Combinando Provisão Dinâmica de Recursos e Rejuvenescimento de Software

Raquel Vigolvino Lopes

Tese de Doutorado submetida à Coordenação dos Cursos de Pós-Gradução em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Francisco Vilar Brasileiro, Dr.
Orientador
Walfredo Cirne, Dr.
Orientador

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

L864c
2007 Lopes, Raquel Vigolvino
Combinando provisão dinâmica de recursos e rejuvenescimento de software /Raquel Vilgovino Lopes. – Campina Grande: 2007.
194f. : il

Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro Engenharia Elétrica e Informática.
Referências.
Orientadores: Dr. Francisco Vilar Brasileiro e Dr. Walfredo Cirne

1. Sistemas de Gerência-Coordenação. 2. Provisão Dinâmica. 3. Rejuvenescimento de Processos. I. Título.

CDU 004.75(043)

COMBINANDO PROVISÃO DINÂMICA DE RECURSOS E REJUVENESCIMENTO DE SOFTWARE

RAQUEL VIGOLVINO LOPES

Tese Aprovada em 26.11.2007

FRANCISCO VILAR BRASILEIRO, Ph.D., UFCG
Orientador

WALFREDO DA COSTA CIRNE FILHO, Ph.D., UFCG
Orientador(Ausencia Justificada)

CARLOS ANDRÉ GUIMARÃES FERRAZ, Ph.D., UFPE
Componente da Banca

VIRGÍLIO AUGUSTO FERNANDES ALMEIDA, Dr., UFMG
Componente da Banca (Ausência Justificada)

PÉRICLES REZENDE BARROS, Ph.D., UFCG
Componente da Banca

DALTON DARIO SEREY GUERRERO, D.Sc., UFCG
Componente da Banca

CAMPINA GRANDE – PB
NOVEMBRO - 2007

Agradecimentos

Agradeço primeiramente a Deus. Passei por muitos momentos difíceis ao longo desses anos e sei que foi Ele quem me sustentou... E foi por acreditar Nele que sempre busquei aprender, me melhorar e superar as situações mais difíceis que passei em minha vida. "A quem muito foi dado, muito será cobrado". Espero poder retribuir de alguma forma tudo que me foi concedido.

Agradeço de forma muito especial aos meus pais, que investiram em minha educação desde o meu nascimento e me ensinaram desde muito cedo que o conhecimento e a educação são a maior herança que eles me deixam. Acho que nunca poderei retribuir tudo que fizeram por mim... Além deste investimento, eles são meus melhores amigos, a quem imito e de quem me orgulho de ser filha.

Apesar dele não entender nada ainda, agradeço ao meu filho, Guilherme. Na realidade, ele não facilitou meu trabalho em nada, muito pelo contrário. Tantas noites mal dormidas ao lado dele até atrapalharam um pouco. Mas ele é meu grande incentivo para me tornar uma pessoa melhor e nos momentos mais difíceis ele — de alguma forma inexplicável — me incentivava a perseverar.

Agradeço aos meus orientadores, Fubica e Walfredo, pela paciência que tiveram comigo, por me deixarem muito à vontade para pesquisar no tema de meu interesse, pelos ensinamentos, pelas conversas e pela amizade que acabou surgindo naturalmente ao longo desses anos.

Agradeço aos meus irmãos de sangue, Camila e Felipe, a quem muito amo, por estarem comigo, mesmo que apenas em pensamento. E agradeço também aos meus muitos amigos — a todos eles — irmãos por opção. Gostaria de falar de cada um deles aqui, mas serei obrigada a escolher apenas alguns.

Agradeço a todos que fazem o Laboratório de Sistemas Distribuidos (LSD), sem exceção. Muitos já terminaram seus trabalhos e já partiram, mas ficam guardados em meu coração todos os momentos que vivemos juntos. Osorinho, Ana, Esther, Andrey, Nelson e Gustavo, muito obrigada por estarem por perto.

Outros ainda estão por aqui e puderam acompanhar bem de perto minha caminhada completa de doutorado, dando seu ombro amigo em muitos momentos e em outros me ajudando tecnicamente, dando sugestões de melhoria para meu trabalho e me incentivando. Lívia, Ayla, Lauro, Nazareno, Mila, Bárbara, Dudu, Iury, Rodrigo e tantos outros (é muita gente!!!). Meu amigos queridos do LSD, muito obrigada por tudo, nunca esquecerei vocês!

Ainda falando aos meus amigos do LSD, quero agradecer especialmente à equipe de suporte. Zane, Thiago, Matheus, Randolph, que não estão mais aqui, mas que me ajudaram enormemente nas configurações dos meus experimentos. E também a Moisés, Roberto e Karla. Vocês todos contribuíram enormemente para que meu trabalho fosse realizado com sucesso. Muito obrigada!

Continuando no LSD, quero agradecer ao OurGrid nas pessoas de William e Walfredo, responsáveis, respectivamente, pelo desenvolvimento e idéia do mesmo. Sem o OurGrid eu provavelmente ainda estaria rodando simulações até hoje. Obrigada!

Agradeço, ainda no LSD, a Cleide, que manteve minha mesa sempre limpinha, sem arrumar minha bagunça "organizada", sua fé é um exemplo para mim. Agradeço a Keka, sempre pronta a nos ajudar com uma declaração, um ofício, e preparando com tanto carinho nossas viagens.

Agradeço também, de coração, a Jacques e a Peter, que não puderam me orientar no doutorado, mas que certamente, contribuíram enormente para a minha formação. As conversas com Jacques sobre modelagem, validação, etc. foram de grande importância no desenvolvimento da minha tese.

Ah! Agradeço enormemente a Vaneide, Cristina e Silvana, que cuidam tão bem de Guilherme na minha ausência. E minha ausência foi bem grande... Sem vocês (e mainha para coordenar tudo!) eu não teria tido a tranquilidade necessária para realizar meu trabalho. Muito obrigada!

Finalmente, agradeço a algumas amigas de sempre: Erikinha, Ju, Milene, Taciana e Thiciane, que mesmo de longe me deram força, incentivo... Acreditaram em minha capacidade, quando eu mesma cheguei a duvidar. Muito obrigada meninas!

Há muito mais a quem agradecer... A todos aqueles que, embora não nomeados, me brindaram com seus inestimáveis apoios em distintos momentos e por suas presenças, o meu reconhecido e carinhoso muito obrigada!

Todos vocês são co-autores deste trabalho.

Resumo

Sistemas de provisão dinâmica de recursos gerenciam automaticamente a capacidade da aplicação de forma que ela se adeque a sua demanda corrente. Estes sistemas são importantes em se tratando de aplicações que recebem uma carga de trabalho que varia ao longo do tempo. Sistemas de rejuvenescimento objetivam detectar e recuperar aplicações de falhas de software através do reinício dos componentes em falha. Tradicionalmente, sistemas de provisão dinâmica de recursos e sistemas de rejuvenescimento de software são construídos e estudados de forma isolada e independente. Argumenta-se que existe uma ampla classe de aplicações, como por exemplo, aplicações Web, que precisam ser controladas simultaneamente tanto no aspecto de provisão dinâmica de recursos quanto no aspecto de rejuvenescimento de software. No entanto, não são conhecidos estudos que explorem os efeitos da coexistência de tais sistemas atuando sobre a mesma aplicação. Nesta tese são apresentados diversos cenários nos quais sistemas de provisão dinâmica de recursos e de rejuvenescimento independentes não atuam eficientemente ao controlar uma mesma aplicação. Nestes cenários, tais sistemas, sem o conhecimento um do outro, tomam decisões que levam a aplicação a oferecer uma pior qualidade de serviço. Além disso, as falhas em componentes causam mais danos à qualidade de serviço da aplicação do que causariam se ocorressem em uma infra-estrutura super-provida de recursos estaticamente. Técnicas simples de coordenação das atividades destes sistemas foram propostas e estudadas e percebeu-se que é possível e interessante coordenar a ação dos sistemas de provisão dinâmica e rejuvenescimento de forma a melhorar a qualidade de serviço da aplicação gerenciada.

Abstract

Dynamic provisioning systems change application capacity in order to use an appropriate number of resources to accommodate current load. These systems benefit applications with time-varying workloads. Rejuvenation systems detect/forecast software failures and restart one or more components of the application in order to bring them to a healthy state. Traditionally, these systems have been developed unaware of one another. We argue that many applications need to be controlled by both systems simultaneously. This is the case, for instance, of Web-based applications. However, we are not aware of research works that study the effects of the coexistence of these systems, actuating over the same application. We identified scenarios in which these systems cannot efficiently actuate over the same application when they are not aware of each other. Component failures cause more damage to the application quality of service in these scenarios than they would cause if they occured in a static over-provisioned infrastructure. Our results show that when both systems coexist independently of each other, application quality of service degrades in comparison with the quality of service provided when each system is acting alone. Simple coordination techniques between dynamic provisioning and rejuvenation actions were proposed. By applying these techniques we can improve the application quality of service.

Conteúdo

Lista de Símbolos e Abreviaturas

DPS: Dynamic Provisioning System.

RS: Rejuvenation System.

QoS: Quality of Service.

AWoF: Application Without Faults.

AWF: Application With Faults.

AWoF+DPS: Composição que modela um *e-service* sem faltas de software que tem sua capacidade gerenciada por um sistema de provisão dinâmica.

AWF+RS: Composição que modela um *e-service* com faltas de software gerenciado por um sistema de rejuvenescimento.

AWF+DPS+RS: Composição que modela um *e-service* com faltas de software gerenciado por um sistema de provisão dinâmica de recursos e por um sistema de rejuvenescimento independentes.

AWF+DPS⋆RS: Composição que modela um *e-service* com faltas de software gerenciado por um sistema de provisão dinâmica de recursos e por um sistema de rejuvenescimento que colaboram entre si.

$\mathbf{A_e}$: Disponibilidade da aplicação e.

$\overline{\mathbf{R}}_\mathbf{e}$: Tempo médio de resposta da aplicação e.

PDPS: Perfect Dynamic Provisioning System.

$\mathbf{l}^{\mathbf{avail}}_{\mathbf{PDPS}}$**:** Perda em termos de disponibilidade causada por ter a aplicação sem faltas de software gerenciada por um PDPS, se comparada à disponibilidade máxima que a aplicação pode oferecer.

$\mathbf{l}^{\mathbf{avail}}_{\mathbf{RS}}$**:** Perda em termos de disponibilidade causada por ter a aplicação com faltas de software gerenciada por um RS, superprovida de recursos, se comparada à disponibilidade máxima que a aplicação pode oferecer. Representa a perda de disponibilidade devido à ocorrência de falhas e rejuvenescimentos.

$\mathbf{l}^{\mathbf{avail}}_{\mathbf{PDPS+RS}}$**:** Perda em termos de disponibilidade causada pela atuação do PDPS e pela ocorrência das falhas e rejuvenescimentos quando PDPS e RS atuam simultaneamente de

forma independente.

$\mathbf{l^{rej}_{DPS}}$**:** Quantidade de requisições rejeitadas por conta da atuação do DPS ao gerenciar uma aplicação sem faltas de software. Representa a perda de requisições devido à provisão dinâmica de recursos.

$\mathbf{l^{rej}_{RS}}$**:** Quantidade de requisições rejeitadas por conta da atuação do RS ao gerenciar uma aplicação com faltas de software, superprovida de recursos. Representa a perda de requisições devido à ocorrência de falhas e rejuvenescimentos.

$\mathbf{l^{rej}_{DPS+RS}}$**:** Quantidade de requisições rejeitadas por conta da atuação do DPS e da ocorrência das falhas e rejuvenescimentos quando DPS e RS atuam simultaneamente de forma independente.

$\mathbf{l^{t}_{DPS}}$**:** Perda em termos de tempo de resposta causada por ter a aplicação sem faltas de software gerenciada por um DPS, se comparada ao tempo mínimo de resposta que a aplicação poderia oferecer caso tivesse capacidade infinita (demanda média de serviço).

$\mathbf{l^{t}_{RS}}$**:** Perda em termos de tempo de resposta causada por ter a aplicação com faltas de software gerenciada por um RS, superprovida de recursos, se comparada ao tempo mínimo de resposta que esta mesma aplicação, com a mesma capacidade poderia oferecer caso fosse livre de faltas de software. Representa a perda em termos de tempo de resposta devido à ocorrência de falhas e rejuvenescimentos.

$\mathbf{l^{t}_{DPS+RS}}$**:** Perda em termos de tempo de resposta causada pela atuação do DPS e pela ocorrência das falhas e rejuvenescimentos quando DPS e RS atuam simultaneamente de forma independente.

$\mathbf{g^{t}_{DPS*RS}}$**:** Ganho em termos de disponibilidade medido quando as ações do DPS e do RS estão coordenadas.

$\mathbf{g^{t}_{DPS*RS}}$**:** Ganho em termos de tempo de resposta medido quando as ações do DPS e do RS estão coordenadas.

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação e relevância

Com o aumento da complexidade da infra-estrutura de tecnologia da informação (TI) e de sua importância tanto para os negócios das empresas quanto para o dia-a-dia das pessoas, a gerência destes sistemas vem passando por diversas mudanças ao longo do tempo. Tradicionalmente, o homem foi o coordenador e o principal tomador de decisões no âmbito da gerência destes sistemas. No entanto, devido à complexidade associada aos sistemas atuais, incluir o homem no processo de gerência passou a ser caro (GANEK; CORBI, 2003; MENASCÉ et al., 2003) e muitas vezes ineficiente (BROWN; PATTERSON, 2001; OPPENHEIMER; GANAPATHI; PATTERSON, 2003). Surgiu então a necessidade de automatizar a gerência dos sistemas (GANEK; CORBI, 2003; MENASCÉ et al., 2003), movimento conhecido por computação autônoma (*autonomic computing*).

Esta necessidade de automação da gerência vem sendo também estimulada pelo surgimento de um novo modelo de entrega de serviços de TI: a computação sob demanda (*utility computing*) (EJASENT, 2001; ERIKSEN, 2003; RAPPA, 2004). Em um ambiente de computação sob demanda obtém-se infra-estrutura, aplicações e processos de negócio através da Internet de forma segura e escalável (RAPPA, 2004). É um modelo funcionalmente semelhante aos serviços de distribuição de água e energia elétrica. Da mesma forma que é a empresa de energia elétrica que se preocupa com a geração, transporte e distribuição de energia, existirão empresas especializadas em implantar, manter e oferecer serviços de TI quando os seus clientes o desejarem. Neste ambiente, os usuários pagam apenas pelos serviços consumidos. Assim, os lucros de quem provê computação sob demanda dependem da sua agilidade e eficiência em oferecer para os clientes o serviço desejado na hora desejada (ERIKSEN, 2003). Desta forma, a agilidade da gerência torna-se fundamental.

Para gerenciar agilmente tal ambiente dinâmico e complexo é preciso reduzir ao máximo a necessidade de intervenção humana no processo de gerência.

Motivados pela importância imediata da automação da gerência, diversos pesquisadores têm proposto sistemas de gerência que não apenas monitoram a aplicação e seu ambiente, mas que tomam decisões para os mais diversos fins com base nestas informações (RANJAN et al., 2002; HONG et al., 2002; CANDEA et al., 2003; MENASCÉ et al., 2003; DIACONESCU; MOS; MURPHY, 2004; AIBER et al., 2004; GUANGZHI et al., 2004). Cada um destes sistemas lida com características específicas das aplicações gerenciadas. Por exemplo, alguns responsabilizam-se por detectar e recuperar aplicações de ataques; outros lidam com a provisão dinâmica de recursos; outros com a configuração de desempenho automática mais adequada, etc.

Para que seja possível afastar completamente a necessidade de intervenção humana no processo de gerência de tais sistemas, acredita-se que é preciso realizar uma gerência holística. Sendo assim, seria necessária a coexistência de diversos sistemas automáticos de gerência gerenciando a mesma aplicação, sob diferentes aspectos. Para se chegar a este ponto, no entanto, é importante definir: (i) que sistemas de gerência são fundamentais e (ii) estudar como estes sistemas interagem e como podem colaborar entre si.

Nesta tese consideram-se especialmente dois tipos de sistemas automáticos já propostos na literatura: sistemas de provisão dinâmica de recursos e sistemas de rejuvenescimento[1] de software.

Um sistema de provisão dinâmica de recursos (*Dynamic Provisioning System* - DPS) visa adequar automaticamente a capacidade de uma aplicação à demanda corrente da mesma (EJASENT, 2001; RANJAN et al., 2002; LASSETTRE et al., 2003; URGAONKAR et al., 2005). Tradicionalmente, administradores de tais aplicações sujeitas a cargas variáveis utilizavam-se de super-provisão de recursos (GRIBBLE, 2001). Para super-prover uma aplicação de recursos tenta-se prever a demanda máxima a ser recebida pela aplicação e aloca-se a quantidade de recursos necessária para atender a esta demanda. Quando a super-provisão é realizada observa-se uma baixa utilização média dos recursos (EJASENT, 2001), uma vez que a carga média da aplicação é inferior à carga de pico. A super-provisão é uma técnica simples, porém cara e sujeita a erros. É cara porque torna necessário o uso de uma quantidade de recursos sempre maior que a necessária e, sujeita a erros porque depende de uma previsão da carga de pico correta e de um mapeamento desta carga em quantidade de recursos também correta. Sistemas de provisão dinâmica são desejáveis em quaisquer ambientes que hospedem aplicações cuja carga varia ao longo do tempo. Em ambientes de

[1] Os termos rejuvenescimento e reinício serão usados neste trabalho como sinônimos

computação sob demanda estes sistemas são obrigatórios.

Sistemas de rejuvenescimento de software (*Rejuvenation Systems* - RS) lidam com falhas de software que ocorrem enquanto a aplicação está sendo executada. Estes sistemas detectam ou prevêem a ocorrência de falhas de software e levam a aplicação a um estado completamente operacional através do rejuvenescimento (reinício) da aplicação ou do componente em falha (AVRITZER; BONDI; WEYUKER, 2005; LI; VAIDYANATHAN; TRIVEDI, 2002; GARG et al., 1998; HUANG et al., 1995).

De acordo com Oppenheimer *et al* (OPPENHEIMER; GANAPATHI; PATTERSON, 2003), falhas de software e erros de operadores humanos são atualmente uma das maiores causas de falhas de aplicações de Internet. Portanto, lidar com tais falhas de forma automática vem também sendo uma atividade importante para tais aplicações.

Os serviços escaláveis que recebem uma carga variável ao longo do tempo representam uma ampla classe de aplicações que se beneficiariam ao ser gerenciadas por estes sistemas simultaneamente. Por constituir um *serviço*, tais aplicações executam por um período de tempo ilimitado. Uma simulação de previsão do tempo, por exemplo, executa por um dado período, e quando a previsão é concluída a aplicação é encerrada, não sendo considerado um serviço. Já servidores de bancos de dados ou servidores Web precisam estar sempre em execução, à espera de pedidos de clientes, constituindo assim serviços. Além de ser um serviço, tais aplicações recebem uma carga variável ao longo do tempo. Os clientes de tais serviços submetem requisições que geram processamento no lado servidor e, após o processamento, uma resposta é gerada e enviada ao cliente. Finalmente, tais aplicações são escaláveis, isto é, é fácil aumentar ou diminuir a sua capacidade, sem modificar seu código ou suas especificações. O aumento de capacidade é realizado através do aumento da quantidade de recursos que executam a aplicação. Geralmente, estas aplicações são executadas em clusters. Neste trabalho, convencionou-se chamar estas aplicações de *e-services*.

Um grupo muito importante de *e-services* é o das aplicações baseadas em Web. O acesso a estas aplicações é feito tipicamente através de um navegador Web via Internet por clientes globalmente distribuídos. A cultura que se formou em torno destas aplicações leva seus clientes a esperarem que elas estejam em perfeito funcionamento 24 horas por dia, 7 dias por semana. Isto implica em um ótimo desempenho e uma alta disponibilidade. Manter esta qualidade de serviço é uma atividade complexa, especialmente quando existem razões econômicas que impedem que a aplicação permaneça constantemente super-provida de recursos.

As aplicações baseadas em Web estão sujeitas a uma carga altamente variável (CRO-

VELLA; BESTAVROS, 1996; BARFORD; CROVELLA, 1998; ARLITT; KRISHNAMURTHY; ROLIA, 2001; MENASCÉ et al., 2003) e difícil de ser prevista. Sendo assim, estas aplicações apresentam uma demanda de recursos que varia com o tempo, justificando-se a importância de um sistema de provisão dinâmica de recursos.

Igualmente importantes para as aplicações Web (e para os *e-services* em geral) são os sistemas de detecção e recuperação de falhas de software. Esta importância se deve especialmente à natureza de execução permanente destas aplicações. Algumas faltas de software podem escapar de todos os testes e se manifestar apenas durante a execução prolongada da aplicação. Com o passar do tempo em execução estas aplicações podem entrar em estados não esperados que não foram estudados na fase de testes. Tipicamente, estas faltas de software são realmente difíceis de ser capturadas e são classificadas como Heisenbugs (GRAY, 1986) e faltas relacionadas ao envelhecimento dos processos (VAIDYANATHAN; TRIVEDI, 2001). Ambos os tipos de falta são ativados sob certas condições que não são facilmente reproduzíveis.

Um outro fato que aumenta a possibilidade de existência de faltas de software em tais aplicações é que muitas delas precisam ser desenvolvidas em um tempo mínimo para que sejam úteis ao negócio (*time to market*) e, além disso, estão permanentemente passando por atualizações que refletem as mudanças do negócio, aumentando a probabilidade de introdução de faltas de software.

Apesar da importância de sistemas de provisão dinâmica e de rejuvenescimento para uma grande classe de aplicações, não são conhecidos estudos que explorem os efeitos da coexistência de tais sistemas atuando sobre a mesma aplicação. Cada um destes sistemas vem sendo proposto de forma isolada e independente. De fato, sistemas de provisão dinâmica não consideram a possibilidade de falhas de software. Da mesma forma, sistemas de rejuvenescimento não consideram que a carga a que estas aplicações são submetidas varia substancialmente, e que elas podem estar com uma provisão limitada de recursos em diversos momentos.

Nos estudos iniciais deste trabalho, um esquema de provisão dinâmica de recursos para aplicações de duas camadas foi experimentado e, surpreendentemente, observou-se que mesmo quando mais recursos eram providos para a aplicação, a qualidade de serviço da aplicação permanecia baixa. Investigando mais profundamente, descobriu-se que a aplicação apresentava faltas de software. Sistemas de provisão dinâmica usam funções que relacionam métricas de sistema (carga, consumo de recursos como CPU, etc.) com o número de máquinas a ser provido à aplicação. Porém, quando faltas de software leval à ocorrência de falhas, estas funções podem não mais refletir a realidade, uma vez que haverá

componentes da aplicação aparentemente disponíveis, consumindo recursos, processando pedidos, mas não se comportando mais de acordo com as suas especificações. Ao associar o esquema de provisão dinâmica com um esquema de rejuvenescimento percebeu-se uma melhora na qualidade de serviço da aplicação gerenciada, ao mesmo tempo em que houve uma economia de recursos, especialmente quando falhas e rejuvenescimentos ocorriam (LOPES; CIRNE; BRASILEIRO, 2004). Verificou-se então que é possível e interessante que haja uma colaboração entre sistemas de provisão dinâmica e de rejuvenescimento.

Analisando as pesquisas já realizadas não é possível afirmar se o comportamento emergente da união simples de sistemas de provisão dinâmica de recursos e de rejuvenescimento construídos de forma independente será satisfatória no que diz respeito à qualidade de serviço da aplicação e ao custo de mantê-la.

Este trabalho possibilita uma coexistência mais harmoniosa entre sistemas de provisão dinâmica e de rejuvenescimento, de forma que *e-services* sejam automatica e simultaneamente gerenciados quanto a estes aspectos. Situações em que a união simples de sistemas de provisão dinâmica de recursos e de rejuvenescimento não é eficiente, especialmente em termos da qualidade de serviço da aplicação gerenciada, foram identificadas e analisadas (vide Capítulo 5). Para solucionar tal problema, foram elaboradas técnicas que visam coordenar as ações de sistemas de provisão dinâmica de recursos e de rejuvenescimento quando atuam sobre uma mesma aplicação. Ao aplicar essas técnicas de coordenação observou-se que em muitos cenários a aplicação gerenciada ofereceu melhor qualidade de serviço sem acréscimo substancial de custo (vide Capítulo 6). Estas técnicas permitem que os sistemas legados de provisão dinâmica e de rejuvenescimento sejam reaproveitados.

1.2 Objetivos

1.2.1 Objetivos gerais

O objetivo deste trabalho é evidenciar a não ortogonalidade de *DPS* e *RS* e identificar uma coordenação adequada das ações destes sistemas de forma que eles possam colaborar entre si. Uma vez identificadas tais colaborações, possibilita-se a gerência automática mais eficiente de *e-services* em dois aspectos distintos: (i) adequar a capacidade de tais aplicações à demanda corrente e (ii) detectar e recuperar tais aplicações de falhas de software através de rejuvenescimento.

1.2.2 Objetivos específicos

No sentido de atingir o objetivo descrito acima, definiram-se as seguintes metas:

1. Caracterizar o comportamento emergente de sistemas de provisão dinâmica de recursos e sistemas de rejuvenescimento ao atuarem sobre uma mesma aplicação sem coordenação. Neste sentido, pretende-se avaliar o impacto em termos de qualidade de serviço e custo da aplicação, de se ter um *DPS* e um *RS* atuando de forma independente e simultânea sobre uma aplicação;

2. Elaborar técnicas de coordenação que permitam colaboração entre o *DPS* e o *RS*, de forma a gerar um terceiro sistema capaz de gerenciar um *e-service* mais eficazmente. Espera-se que a atividade coordenada dos dois sistemas melhore a qualidade de serviço e minimize o custo de manter a aplicação gerenciada em relação à qualidade de serviço e custo encontrados quando não há coordenação entre os sistemas;

3. Comparar, em termos de qualidade de serviço e de custo da aplicação, os resultados do sistema coordenado com os resultados dos sistemas que atuam sem coordenação entre os sistemas;

4. Realizar medições em um ambiente mais próximo do real com o objetivo de validar as idéias principais desta tese, bem como validar o modelo de simulação utilizado;

5. Estudar a qualidade de serviço e o custo da aplicação gerenciada por um sistema de provisão dinâmica perfeito e um sistema de rejuvenescimento com e sem coordenação;

6. Identificar analiticamente situações que indiquem quando a disponibilidade ou o tempo de resposta da aplicação serão afetados devido ao rejuvenescimento de um ou mais nós.

1.3 Metodologia

A metodologia utilizada segue uma abordagem de estudo comparativo baseado em modelos de componentes. Sendo assim, esta metodologia está associada à aplicação de um modelo baseado em componentes. Este modelo pode ser composto pelos seguintes componentes individuais:

- Uma aplicação do tipo *e-service*;
- Um sistema gerador de carga (*LGS - Load Generator System*);

- Um sistema de injeção de erros (*SEIS - Software Error Injection System*);
- Um sistema de provisão dinâmica (*DPS - Dynamic Provisioning System*);
- Um sistema de rejuvenescimento de software (*RS - Rejuvenation System*);
- Um módulo acoplado ao DPS e ao RS que permite a coordenação entre as atividades destes sistemas (*Coord*).

Uma visão de alto nível deste modelo encontra-se na Figura 1.1.

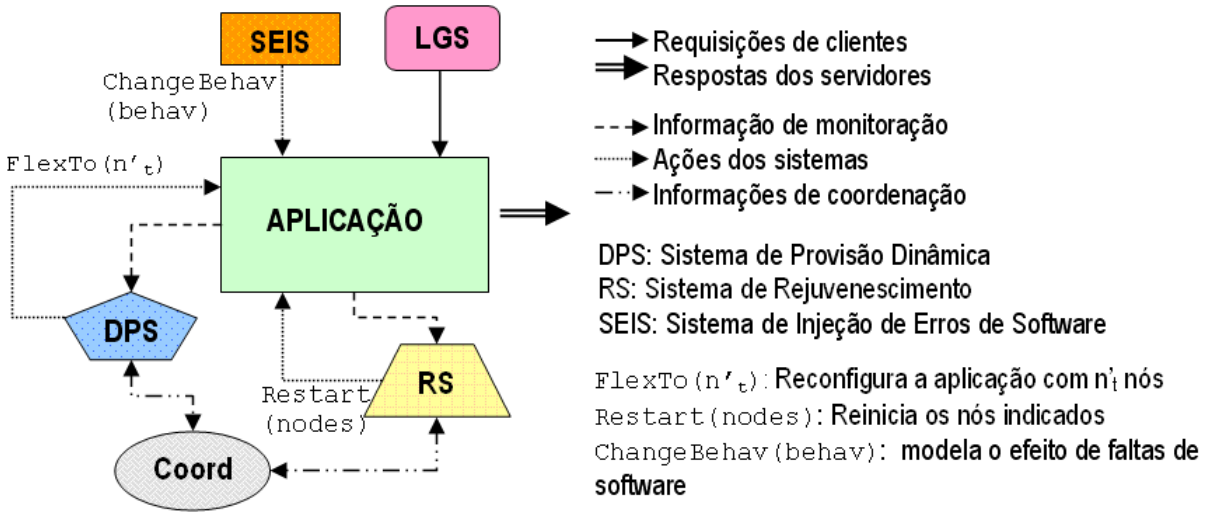

Figura 1.1: Visão de alto nível do modelo baseado em componentes

A aplicação, como já mencionado, é um *e-service*. Esta aplicação é executada em um cluster com $r(t)$ nós no instante t com balanceamento de cargas.

O *LGS* envia requisições para a aplicação de acordo com algum modelo de carga. Cada requisição recebida pela aplicação indica a quantidade de tempo de serviço necessária para servir a requisição.

Os sistemas de gerência (*DPS* e *RS*) formam, cada um, um laço de controle. Mais detalhes sobre estes sistemas serão apresentados nas seções 2.1.2 e 2.2.4. De forma simplificada, eles coletam informações que servem como base para decisões de gerência. Enquanto o *DPS* modifica a capacidade da aplicação em termos de número de nós utilizados de acordo com a demanda atual, o *RS* prevê/detecta e recupera a aplicação de falhas através de rejuvenescimentos. Um componente da aplicação que estiver sendo rejuvenescido permanece indisponível, até que o rejuvenescimento se complete. Estes sistemas podem ou não ter suas atividades coordenadas, de acordo com a presença ou não do componente *Coord*.

Finalmente, o *SEIS* introduz erros no sistema modificando o comportamento da aplicação de forma que os efeitos de faltas de software sejam modelados.

1.3.1 Composições do modelo

Foram estudadas quatro diferentes composições do modelo. Em todas as composições o componente *LGS* e a aplicação estão presentes. A seguir são apresentadas as quatro composições, omitindo-se o componente *LGS*, que permanece o mesmo em todos os casos:

1. *Composição AWoF+DPS*. Esta composição modela um *e-service* sem faltas de software (*Application Without Faults* - AWoF) que tem sua capacidade gerenciada por um sistema de provisão dinâmica;

2. *Composição AWF+RS*. Esta composição modela um *e-service* com faltas de software (*Application With Faults* - AWF) gerenciado por um sistema de rejuvenescimento. Como o componente DPS não está presente nesta composição, haverá super-provisão estática de recursos;

3. *Composição AWF+DPS+RS*. Esta composição engloba todos os componentes, exceto o módulo coordenador *Coord*. Não existe comunicação entre *DPS* e *RS*, isto é, eles agem sobre a mesma aplicação de forma independente;

4. *Composição AWF+DPS*RS*. Esta composição engloba todos os componentes. Os sistemas de provisão dinâmica de recursos e de rejuvenescimento se comunicam e atuam sinergeticamente devido à atuação do *Coord*.

A comparação dos resultados de diferentes composições com configurações afins é importante porque oferece subsídios para realizar o estudo proposto. A composição *AWoF+DPS* precisa ser estudada porque indica a qualidade de serviço da aplicação e o custo de mantê-la quando o *DPS* atua sobre a infra-estrutura da aplicação e falhas não ocorrem. Ao comparar estes resultados com os resultados obtidos ao analisar a composição *AWF+DPS+RS* é possível identificar as perdas em termos de qualidade de serviço e custo devido à ocorrência de falhas e de rejuvenescimentos. Ao identificar a qualidade de serviço e o custo da aplicação na composição *AWF+RS* e comparar com os resultados obtidos para a composição *AWF+DPS+RS* identificam-se as perdas devido à atuação do *DPS* em um ambiente com falhas. Em resumo, as composições *AWoF+DPS* e *AWF+RS* oferecem os limites superiores de qualidade de serviço e inferiores de custo que são necessários para que se possa identificar as perdas que ocorrem quando *DPS* e *RS* atuam simultaneamente.

Finalmente, ao comparar a qualidade de serviço e o custo da aplicação nas composições *AWF+DPS*RS* e *AWF+DPS+RS* pode-se identificar os ganhos advindos das técnicas de coordenação. Para estas comparações, os resultados da composição *AWF+DPS+RS* são limites inferiores de qualidade de serviço e superiores de custo.

1.3.2 Métricas de interesse

Como já mencionado na sub-seção anterior, as comparações entre composições diferentes serão realizadas à luz de métricas de interesse. As métricas de interesse associadas à metodologia estão relacionadas à qualidade de serviço do *e-service* e ao custo de executá-lo. Estas métricas foram escolhidas porque assume-se que sistemas de gerência (incluindo sistemas automáticos de gerência) buscam atuar sobre a aplicação gerenciada (neste caso o *e-service*) de tal forma que esta ofereça a qualidade de serviço esperada pelo menor custo.

Métricas relacionadas à qualidade de serviço

Em se tratando de *e-services*, que são tipicamente aplicações interativas, consideram-se importantes as seguintes métricas: disponibilidade (A) e tempo de resposta (R).

A disponibilidade de um *e-service* no intervalo de tempo $(t - \Delta t, t]$ pode ser calculada segundo a Equação 1.1. Nesta equação, S e U significam, respectivamente, a quantidade total de requisições processadas com sucesso e com insucesso no intervalo de tempo em questão.

$$A = \frac{S}{S + U} \tag{1.1}$$

Uma requisição é processada com sucesso se ela for aceita para ser processada e, após ser aceita, se ela retornar a resposta esperada. Na prática, uma requisição de uma aplicação Web é processada com sucesso se a resposta vier com código de resposta HTTP 200 (OK). Uma requisição não é processada com sucesso quando ela não é aceita para ser processada ou quando ela é aceita, porém devido a algum erro no servidor a resposta é diferente da esperada. Na prática, uma requisição pode ser rejeitada, por exemplo, quando o servidor está passando por um momento de sobrecarga e não pode aceitá-la. Também é possível que o servidor aceite processar a requisição, porém não consiga realizá-la completamente devido a algum erro interno. O código de tais respostas é diferente do código HTTP 200 de sucesso.

Os tempos de resposta medidos e usados nas avaliações das composições consideram o tempo de permanência das requisições respondidas com sucesso no sistema.

Métrica relacionada ao custo

Como métrica de custo será considerado o número médio de nós ativos em um determinado intervalo de tempo. A partir deste valor, pode-se, por exemplo, computar o preço que se paga de fato pelos recursos utilizados. Para tal, é preciso multiplicar esse número médio de nós ativos pelo custo dado em moeda por unidade de tempo de manter um nó ativo.

O DPS visa decidir por uma quantidade de nós que seja suficiente para atender a demanda da aplicação. Assim, medir o número médio de nós ativos escolhido pelo DPS em diferentes cenários é uma interessante métrica de comparação.

1.4 Seleção de técnicas de avaliação

A metodologia apresentada na seção anterior pode ser aplicada seguindo-se diferentes técnicas de avaliação. De fato, a metodologia propõe que modelos que englobem os componentes citados sejam comparados segundo as métricas de interesse. Em princípio, a metodologia não está ligada a uma forma específica de técnica de modelagem/avaliação. Assim, quaisquer das técnicas a seguir podem ser usadas para implementar a metodologia:

- Construção e avaliação de modelos analíticos;
- Construção e avaliação de modelos de simulação;
- Preparo e avaliação de experimentos de medição.

De acordo com (JAIN, 1991) cada uma destas técnicas tem suas vantagens e desvantagens. É preciso identificar a que melhor se adequa ao estudo que se deseja realizar. Um modelo é uma abstração, ou uma visão generalizada de um sistema real (MENASCE; ALMEIDA; DOWDY, 2004). Sendo assim, qualquer modelo que não seja o próprio sistema alvo de estudo ou uma cópia perfeita do mesmo é uma simplificação da realidade. A quantidade de detalhes presente em um modelo depende do objetivo que se quer alcançar ao aplicar o modelo. Identificar um modelo que seja útil ao objetivo perseguido e ao mesmo tempo que seja o mais simples possível é um desafio da atividade de modelagem.

Em geral, modelos analíticos, para serem tratáveis matematicamente, requerem muitas simplificações e hipóteses, o que pode comprometer a precisão dos resultados. Simulações podem trazer mais detalhes, e, por isso, estão tipicamente mais próximas da realidade que os modelos analíticos (JAIN, 1991). Apesar de serem tipicamente os modelos mais complexos, as medições experimentais nem sempre garantem resultados muito próximos da realidade, pois dependem de uma gama enorme de parâmetros de configuração, que precisam estar

bem sintonizados com os valores encontrados no mundo real (JAIN, 1991). Assim, mesmo medições experimentais podem ainda não refletir totalmente a realidade.

A análise matemática típica indica o comportamento do sistema uma vez que ele alcança o estado permanente (*steady state*). Normalmente, ela considera o regime permanente dos sistemas estudados, pois nesse regime a manipulação algébrica necessária pode ser complexa, mas ainda é tratável. Como exemplo, os sistemas nesse regime tendem a manter o seu comportamento em torno de médias representativas. Porém quando se passa para o estado transiente a manipulação algébrica é extremamente complexa, chegando em alguns casos a nem ser possível. Falhas e reconfigurações desviam o sistema do seu estado permanente (HARVERKORT et al., 2001), levando-o a um estado transiente. A princípio, acredita-se ser bastante complexa a aplicação de análise matemática no âmbito desta pesquisa, uma vez que o sistema a ser estudado apresenta pontos de reconfiguração - em que mais ou menos recursos são usados - e pontos de regeneração - em que um ou mais nós são rejuvenescidos. Estes pontos de reconfiguração e regeneração estão intimamente ligados à carga recebida pelo *e-service*, que traz um nível a mais de imprevisibilidade e incerteza ao sistema como um todo.

A realização apenas de experimentos também não se mostra adequada. Os experimentos no contexto deste trabalho são caros em termos de recursos e de tempo. Seriam necessários recursos dedicados, que pudessem representar os servidores dedicados do centro de dados. Além disso, os sistemas sob estudo precisariam ser implementados, uma vez que não são encontrados gratuitamente[2] ou não existem implementações disponíveis. Nestas condições, as experimentações são caras em termos da quantidade de recursos necessários, tempo de implementação e tempo de realização. Varrer um grande espaço de possibilidades demandaria um enorme tempo.

Diante do exposto, decidiu-se não optar por apenas um tipo de modelo. Um modelo analítico foi desenvolvido com o objetivo de identificar situações em que a qualidade de serviço da aplicação é diminuída por conta da retirada de um ou mais nós para rejuvenescimento. O estudo comparativo foi realizado através do projeto, implementação e instanciação de modelos de simulação. No âmbito desta pesquisa, as simulações são interessantes pois: (i) permitem que uma grande área do espaço de possibilidades seja estudada em um tempo inferior ao necessário caso experimentos reais estivessem sendo realizados e (ii) são mais facilmente implementados que experimentos, uma vez que os sistemas reais que se deseja estudar não estão facilmente acessíveis, ou não existem, tendo estes que ser implementados.

[2]É possível encontrar DPSs comerciais, como por exemplo, o HP Global Workload Manager (HEWLETT-PACKARD, 2005)

Ao trabalhar com modelos de simulação, é necessário identificar quão bons são os modelos. Esta avaliação do modelo é realizada em dois passos. Um dos passos é investigar se a implementação do modelo está correta. Esta etapa chama-se verificação (JAIN, 1991). O outro é identificar se as hipóteses e simplificações do modelo são razoáveis. Este processo chama-se validação (JAIN, 1991). O modelo de simulação precisa ser validado com um outro modelo. Este outro modelo, é, neste caso, um modelo de medição, muito mais próximo do sistema real. Este não é o sistema real, pois o sistema real está em um ambiente real de centro de dados, com aplicações reais e cargas reais. O modelo de medição foi oganizado em um ambiente de testes controlado. Com este objetivo, experimentos de medição foram realizados e os resultados comparados com os resultados obtidos através do modelo de simulação.

1.5 Contribuições

Em linhas gerais, as contribuições desta tese são as seguintes: 1) estudo de problemas que ocorrem quando sistemas de provisão dinâmica e de rejuvenescimento atuam sobre uma mesma aplicação sem coordenação. Tal estudo está apresentado no Capítulo 5; 2) aumento do nível de autonomia de aplicações do tipo *e-service* através das técnicas de coordenação propostas. As técnicas e os resultados experimentais são apresentados no Capítulo 6; 3) utilização de metodologia própria para estudo de interações e coordenações entre as ações de sistemas de gerência que atuam sobre um mesmo alvo. Tal metodologia é apresentada ainda neste capítulo; 4) estudo analítico de situações em que o rejuvenecimento afeta a qualidade de serviço da aplicação (vide Capítulo 3) e 5) identificação de caraterísticas benéficas à provisão dinâmica de recursos e rejuvenescimento de software. Esta identificação se deu através de projetos experimentais, descritos no Apêndice A.

1.6 Organização do documento

O restante deste documento está organizado como segue.

No Capítulo 2 apresentam-se características importantes de sistemas de provisão dinâmica de recursos e de rejuvenescimento e o estado da arte nestas áreas de pesquisa. Além disso, são também apresentados outros trabalhos que, assim como este, visam estudar interações entre sistemas de controle/gerência que atuam sobre um mesmo alvo. À luz do estado da arte, descreve-se, ainda no Capítulo 2, o problema que se deseja resolver neste trabalho e os requisitos que se pretende atender com a solução proposta.

No Capítulo 3 é apresentado um modelo analítico simples que indica em que situações o rejuvenescimento prejudica a qualidade de serviço da aplicação em termos de disponibilidade e tempo de resposta.

Um modelo de simulação é apresentado no Capítulo 4. No Apêndice A são apresentados diversos resultados de projetos experimentais realizados utilizando-se este modelo. Através destes projetos foi possível identificar os parâmetros que mais influenciam os resultados de disponibilidade da aplicação gerenciada.

No Capítulo 5 o modelo de simulação é instanciado e diversos cenários são analisados com o intuito de investigar se sistemas de provisão dinâmica e de rejuvenescimento coexistem harmonicamente ao gerenciar a mesma aplicação. Observou-se em diversas escalas que a união pura destes sistemas não é eficiente em termos de qualidade de serviço da aplicação.

No Capítulo 6 técnicas de coordenação entre *DPS* e *RS* são propostas e os resultados de qualidade de serviço e custo da aplicação gerenciada obtidos através de simulações são analisados.

O modelo de simulação usado nos capítulos anteriores foi validado através de experimentos de medição e verificado através de testes de unidade e testes de limites. Os resultados são apresentados no Capítulo D.

Finalmente, no Capítulo 7 são apresentadas as conclusões e contribuições deste trabalho, bem como idéias para projetos futuros.

Capítulo 2

Revisão da literatura relacionada

Neste capítulo discutem-se trabalhos relacionados em três áreas mais relevantes para esta pesquisa: alocação dinâmica de recursos, rejuvenescimento de software e sistemas de diversos tipos que interagem e/ou colaboram entre si.

2.1 A alocação dinâmica de recursos

O desejo de acomodar as variações de carga sofridas pelos *e-services* não é novo. Os sistemas de provisão dinâmica de recursos surgiram para solucionar este problema da variabilidade de carga. Tradicionalmente, isto vem sendo feito através da super-provisão de recursos (GRIBBLE, 2001; EJASENT, 2001). Para que a super-provisão seja realizada é necessário determinar a quantidade de recursos necessária para atender a demanda máxima da aplicação e alocar estaticamente esta quantidade de recursos para a aplicação. A super-provisão é cara, pois em grande parte do tempo os recursos ficam sub-utilizados (EJASENT, 2001). Além disso, a super-provisão pode não ser eficaz, uma vez que nem sempre é possível prever corretamente a carga máxima futura de uma aplicação.

Os sistemas de provisão dinâmica de recursos lidam com a alocação de recursos dinamicamente, permitindo que os recursos sejam alocados sob demanda. Assim, os recursos são usados de forma mais eficiente, isto é, são menos sub-utilizados. As aplicações alvo dos sistemas de provisão dinâmica de recursos são tipicamente *e-services*; especialmente aqueles acessíveis via Internet.

Os *e-services* possuem geralmente clientes distribuídos e o fluxo de requisições destes clientes que chega no lado servidor é bastante variável. A carga recebida por aplicações Web e de comércio eletrônico, por exemplo, tem esse comportamento (CROVELLA; BESTAVROS, 1996; BARFORD; CROVELLA, 1998; MENASCÉ et al., 2000; ARLITT; KRISHNAMURTHY; RO-

LIA, 2001; MENASCÉ et al., 2003). As cargas Web são ditas auto-similares, isto é, existem semelhanças entre um "pedaço" de uma carga Web e o seu todo. Isto ocorre porque cargas Web apresentam variações em todas as escalas de tempo em que são observadas (CROVELLA; BESTAVROS, 1996).

As cargas de aplicações de comércio eletrônico também apresentam invariantes semelhantes às encontradas nas cargas Web de conteúdo estático puro (MENASCÉ et al., 2000, 2003). Arlitt *et al* estudaram a escalabilidade de uma grande aplicação de comércio eletrônico (ARLITT; KRISHNAMURTHY; ROLIA, 2001). Eles identificaram que a carga da aplicação estudada chega a ser nove vezes maior no horário de pico em relação ao horário de menor carga. Todos os estudos de cargas - sejam Web puras, sejam cargas que envolvem a recuperação de conteúdos dinâmicos - sugerem que a quantidade de trabalho requerida pelos clientes de uma aplicação varia bastante de acordo com o tempo e também depende de outras questões subjetivas como, por exemplo, promoções, propagandas, dentre outras. Assim, a quantidade de recursos necessária para que este trabalho seja realizado também pode variar para mais ou para menos, justificando e motivando o uso de sistemas de provisão dinâmica de recursos.

Para ser provida de recursos dinamicamente, uma aplicação precisa ser escalável. Os *e-services* são ditos escaláveis porque é possível aumentar e diminuir sua capacidade, sem a necessidade de interromper a aplicação. Uma aplicação é escalável quando sua capacidade de serviço aumenta previsivelmente à medida em que a quantidade de recursos cedida à aplicação aumenta.

Existem duas formas distintas de aumentar a capacidade de uma aplicação. Quando recursos (mais memória, por exemplo) são adicionados a um mesmo nó diz-se que há uma escalabilidade vertical. A escalabilidade horizontal ocorre quando mais nós são ativados para executar a aplicação. *e-services* são escaláveis tanto vertical quanto horizontalmente.

São verticalmente escaláveis pois, ao detectar gargalos em um nó, é possível aumentar a quantidade ou poder de certo recurso, aumentando a capacidade da aplicação. Por exemplo, ao adicionar mais memória em um servidor Web Apache pode-se aumentar a quantidade de requisições atendidas simultaneamente pelo servidor sem perda de desempenho. Como *e-services* são tipicamente executados em cluster com balanceamento de carga, então ao adicionar mais nós aumenta-se a capacidade da aplicação. Em ambos os casos o aumento da capacidade não requer modificações no código da aplicação.

Diante do exposto, o surgimento de sistemas de provisão dinâmica de recursos pode ser vista como uma evolução natural e eficiente da gerência de capacidade de *e-services*. Os sistemas de provisão dinâmica podem aumentar a capacidade de *e-services* tanto vertical,

quanto horizontalmente. Para aumentar verticalmente, no entanto, é preciso que o ambiente seja virtualizado de forma que se possa mudar as quantidades de recursos nas máquinas virtuais facilmente.

2.1.1 Atividades que compõem a alocação dinâmica de recursos

A alocação dinâmica de recursos em centros de dados envolve uma série de atividades que se complementam. Em um centro de dados é possível que diversas aplicações diferentes estejam sendo executadas, todas competindo pelos mesmos recursos. Pesquisas nesta área podem envolver uma ou mais das atividades listadas a seguir e ilustradas na Figura 2.1:

1. Decidir sobre a quantidade de recursos a ser cedida a uma aplicação ao longo do tempo. Esta atividade é realizada por sistemas específicos ligados à aplicação, preparados para monitorar a aplicação e decidir sobre sua provisão de recursos (CHASE et al., 2001; RANJAN et al., 2002; DOYLE et al., 2003; LASSETTRE et al., 2003; ROLIA et al., 2003b; CHANDRA; GONG; SHENOY, 2003b; URGAONKAR et al., 2005; BENNANI; MENASCE, 2005). Nesta proposta de tese, tais sistemas são chamados de sistemas de provisão dinâmica de recursos (*DPS* - Dynamic Provisioning System). Na Figura 2.1 são apresentados cinco sistemas de provisão dinâmica: *DPS-A*, *DPS-B*, *DPS-C*, *DPS-D* e *DPS-E*. Tais sistemas representam, respectivamente, as aplicações A, B, C D e E no sistema global de alocação dinâmica;

2. Decidir sobre aceitar ou não pedidos de recursos individuais de cada aplicação. É factível que diversas aplicações estejam competindo pelos recursos do ambiente computacional, e, neste caso, deve existir um gerente global dos recursos (ilustrado por um retângulo na Figura 2.1) que decide que pedidos por recursos serão aceitos (CHASE et al., 2001; BYDE; SALLE; BARTOLINI, 2003; ROLIA et al., 2003b; BENNANI; MENASCE, 2005);

3. Controlar a admissão de novas aplicações (representado na Figura 2.1 por uma elipse). Esta tarefa também é realizada por uma entidade de gerência de recursos global. Um ambiente de computação sob demanda pode ser bastante dinâmico no que diz respeito às aplicações em execução neste ambiente em cada instante de tempo. É possível que aplicações antigas não mais sejam processadas e que novas aplicações sejam submetidas. Assim, é necessária a realização de controle de admissão para evitar que os recursos do ambiente computacional se tornem insuficientes para atender a demanda de todas as aplicações (ROLIA et al., 2003a, 2003b; ROLIA; ZHU; ARLITT, 2003);

4. Implementar as modificações necessárias para a mudança de capacidade das aplicações. Decisões de alocação de recursos irão requerer reconfigurações de topologia dos recursos, instalação e configuração de software em tais recursos, dentre outras atividades. Os ambientes virtuais em que as aplicações são executadas precisam passar por modificações. Estas modificações são realizadas por atuadores e, idealmente, não interrompem as aplicações e não necessitam da participação de operadores humanos (ROLIA; SINGHAL; FRIEDRICH, 2000; APPLEBY et al., 2001; GOLDSACK et al., 2003). Um atuador está representado na Figura 2.1 por um pentágono.

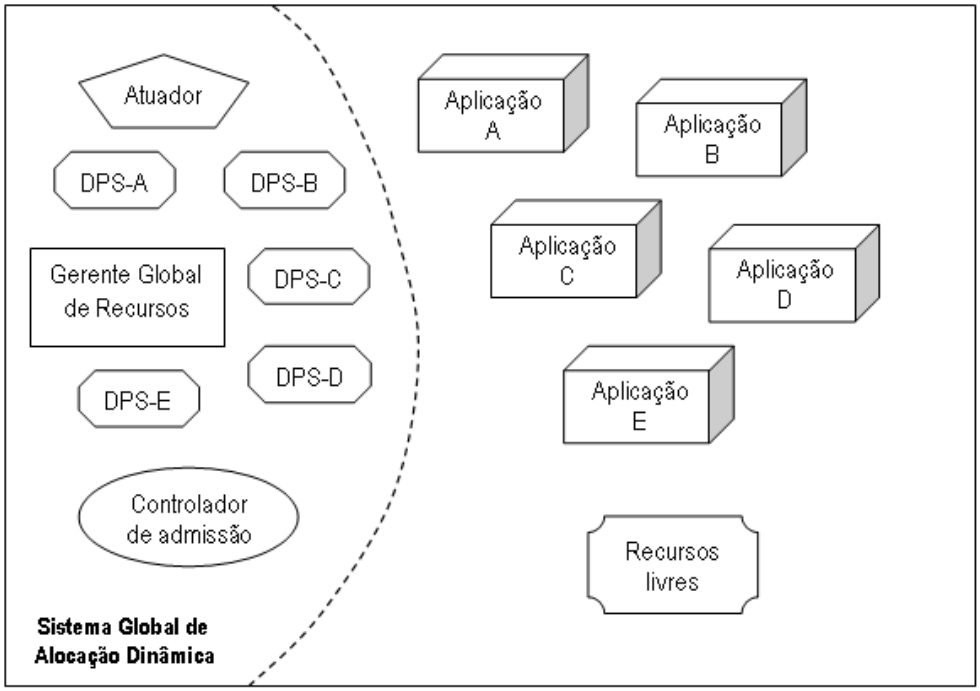

Figura 2.1: Componentes do sistema global de alocação dinâmica de recursos

Os sistemas de provisão dinâmica de recursos (item 1) compõem um dos objetos de estudo desta proposta de tese. Assim, no restante desta seção apenas este componente do sistema global de alocação dinâmica de recursos será tratado.

2.1.2 Sistemas de provisão dinâmica de recursos

Todo sistema de provisão dinâmica está calibrado para decidir a respeito da melhor alocação de recursos para uma dada aplicação. Sistemas de provisão dinâmica de recursos geralmente formam um laço de controle de realimentação (*feedback control loop*) semelhante ao apresentado na Figura 2.2. A aplicação a ser dinamicamente provida de recursos é monitorada

e medida periodicamente, gerando amostras de informações de gerência. Estas informações de gerência são, por exemplo, utilização média de CPU dos servidores, quantidade de requisições processadas no último intervalo de medição, dentre outras. Estas informações são usadas pelo controlador para alimentar um algoritmo de decisão, que irá identificar a quantidade mais adequada de recursos que deve ser alocada para a aplicação no momento.

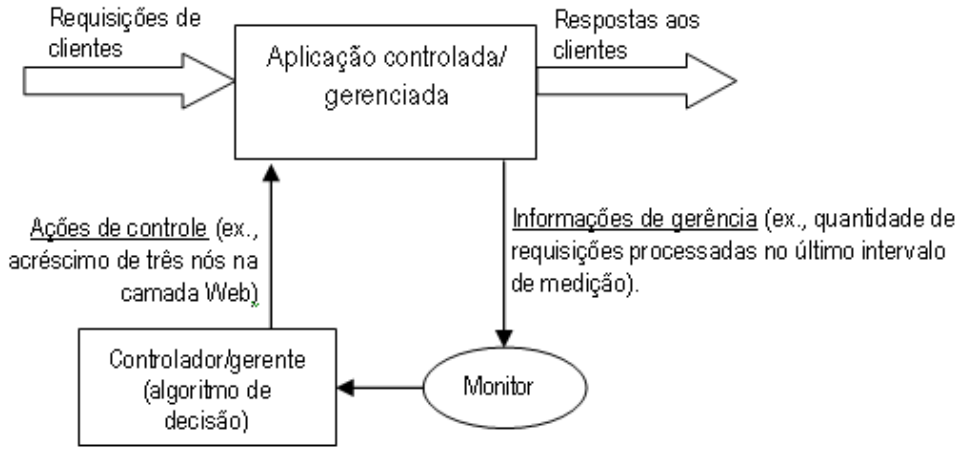

Figura 2.2: Laço de controle de realimentação formado por sistemas de provisão dinâmica

Nos últimos cinco anos, diversos sistemas de provisão dinâmica de recursos foram propostos. Basicamente, o que muda de um sistema para outro são: (i) as informações de monitoração recuperadas através da monitoração/medição da aplicação; (ii) o algoritmo de decisão que usa as informações de monitoração para decidir a respeito da melhor alocação e (iii) a forma de compartilhamento dos recursos do ambiente computacional. Nesta seção diversos destes sistemas propostos na literatura serão apresentados.

Os algoritmos de decisão perseguem algum objetivo. Exemplos simples de objetivos que podem ser buscados são manter a utilização de CPU dos servidores em torno de um alvo, ou os tempos de resposta das aplicações em torno de um alvo.

Em relação ao compartilhamento dos recursos computacionais entre as aplicações, duas formas são conhecidas. Uma das formas é associar servidores a apenas uma aplicação por vez. Neste caso, se o servidor estiver alocado para alguma aplicação, então ele será exclusivo desta aplicação. Este ambiente é conhecido como um ambiente de servidores dedicados.

A segunda forma de compartilhamento de recursos em ambientes computacionais permite que o mesmo nó seja compartilhado entre várias aplicações. Este ambiente é conhecido por ambiente de recursos compartilhados. Quando aplicações com requisitos importantes de desempenho e disponibilidade - como é o caso de grande parte das aplicações Web -

são executadas em ambientes de recursos compartilhados, é preciso isolar uma aplicação da outra, de forma que problemas em uma aplicação não gerem problemas nas demais aplicações que compartilham com elas o mesmo recurso. Este isolamento é tipicamente realizado através do uso de máquinas virtuais, como proposto em (BARHAM et al., 2003; GRAUPNER et al., 2003).

Sistemas de provisão dinâmica recuperam informações de gerência periodicamente, em intervalos chamados *intervalos de medição* (CHASE et al., 2001; RANJAN et al., 2002; DOYLE et al., 2003; LASSETTRE et al., 2003; ROLIA et al., 2003b; CHANDRA; GONG; SHENOY, 2003b; URGAONKAR et al., 2005; BENNANI; MENASCE, 2005). Assim, quando menciona-se que uma dada informação de gerência é recuperada, subentende-se que é em momentos pré-determinados pelos intervalos de medições. Em geral, a métrica recuperada é uma média representativa do último intervalo de medição.

Os algoritmos de decisão são executados também periodicamente, após *intervalos de adaptação*. O valor dos intervalos de adaptação podem ser configurados como sendo iguais ou maiores que o valor dos intervalos de medição, porém nunca inferiores a estes.

Sistemas de provisão diâmica de recursos propostos

A maioria dos sistemas de provisão dinâmica propostos na literatura usa técnicas complexas para identificar a quantidade ideal de recursos a ser alocada à aplicação. Uma vez que as características comuns a todos os sistemas de provisão dinâmica foram apresentadas na seção anterior, esta seção destina-se a evidenciar as diferenças entre alguns sistemas já propostos.

Encontram-se na literatura, por exemplo, sistemas de provisão dinâmica que visam manter a utilização de CPU das máquinas do centro de dados em torno de um alvo (RANJAN et al., 2002; CHASE et al., 2001). Ranjan *et al* apresentam um *DPS* para um ambiente com servidores dedicados, enquanto Chase *et al* consideram recursos compartilhados.

O *DPS* proposto por Ranjan *et al* monitora a utilização de CPU dos servidores, a quantidade de requisições recebidas pela aplicação e a quantidade de requisições processadas pela aplicação (RANJAN et al., 2002). Com base nestes dados o algoritmo de decisão calcula quantos servidores são necessários para manter a utilização média dos servidores em torno de um alvo no próximo intervalo de adaptação.

O DPS proposto por Chase *et al* considera-se uma abordagem baseada em mercado. O objetivo do sistema é usar a quantidade mínima de recursos ativos e manter os recursos livres desligados para economizar energia (CHASE et al., 2001). Neste caso, o sistema proposto é um misto de sistema de provisão dinâmica e gerente global de recursos, pois

ele não apenas decide sobre a quantidade de recursos, mas também distribui os recursos considerando todas as aplicações do centro de dados. Além de visar manter a utilização de CPU dos servidores ativos em torno de uma utilização alvo, o algoritmo de alocação tenta maximizar os lucros do centro de dados.

Um mecanismo de provisão dinâmica de recursos baseado em modelos de desempenho de aplicações foi proposto por Doyle *et al* (DOYLE et al., 2003). Os modelos de desempenho usados relacionam métricas da aplicação (tempo de resposta, por exemplo) com métrica de recursos (utilização de memória, CPU, etc.) e são usados para prever o efeito de alocações sobre a qualidade de serviço da aplicação. Este sistema considera recursos compartilhados entre diversas aplicações.

Informações de carga da aplicação, sejam informações históricas, sejam informações provenientes apenas do último intervalo de medição, servem como base para alguns sistemas de provisão dinâmica (LASSETTRE et al., 2003; URGAONKAR et al., 2005; URGAONKAR; SHENOY, 2005; ROLIA et al., 2003a, 2003b; ROLIA; ZHU; ARLITT, 2003; CHANDRA; GONG; SHENOY, 2003b, 2003a; BENNANI; MENASCE, 2005; RANJAN et al., 2002).

O *DPS* proposto por Lassettre *et al* baseia-se em um sistema preditor de carga (LASSETTRE et al., 2003). Este sistema usa a carga da aplicação em um passado próximo para prever a carga do futuro próximo (escala de poucos minutos). A idéia é que existem tempos de migração em que um servidor está sendo preparado para ativação e, portanto, se a decisão de alocação for tomada apenas quando há a real necessidade, o desempenho da aplicação já estará degradado. O objetivo é assegurar que uma quantidade de recursos satisfatória para atender à demanda da aplicação estará apta a atender as requisições da aplicação quando necessário.

Nakadai e Taniguchi (2007) propõem um sistema de provisão dinâmica que também monitora a carga da aplicação e tenta manter os tempos de resposta em torno de um alvo. Neste caso, são considerados servidores heterogêneos e classes de serviço diferenciadas.

Urgaonkar e Shenoy (2005) combinam técnicas tais como provisão dinâmica, controle de admissão de requisições e degradação de desempenho das requisições admitidas para lidar com sobrecargas extremas. Em um acordo de nível de serviço define-se para cada aplicação do centro de dados o lucro proveniente de uma requisição ser aceita. Assim como no sistema proposto por Lassettre *et al* (2003), o *DPS* usa a carga do último intervalo para prever a carga do próximo intervalo. Com base nesta carga e no lucro associado às requisições desta aplicação, o sistema decide a respeito da quantidade de máquinas que vai ser oferecida a cada aplicação. Assim como sugerido também por Chase *et al* (2001), o sistema de provisão dinâmica está acoplado ao gerente global dos recursos e adota um mecanismo baseado em

mercado para decidir sobre a alocação das máquinas entre aplicações de forma que o centro de dados obtenha maiores lucros.

Um *DPS* que lida com aplicações de múltiplas camadas é apresentado em (URGAONKAR et al., 2005). Neste sistema modela-se a aplicação como uma rede de filas, onde cada servidor é uma fila com servidor único, que recebe uma carga genérica. Este modelo é usado para identificar a quantidade de servidores em cada camada capaz de lidar com a demanda de pico prevista para o próximo intervalo de adaptação. O objetivo é que filas de espera nunca sejam formadas. O *DPS* recebe como informação de gerência as demandas de serviço das requisições que chegam em cada camada e a taxa de chegada de requisições da aplicação.

Ambos os sistemas propostos por Urgaonkar *et al* combinam controle de admissão e alocação dinâmica (URGAONKAR; SHENOY, 2005; URGAONKAR et al., 2005). O algoritmo de provisão dinâmica é executado periodica e automaticamente, assim como os demais sistemas propostos na literatura, mas também pode ser executado sob demanda pelo sistema de admissão quando a taxa de rejeição de requisições estiver alta. Os demais sistemas propostos na literatura não são executados sob demanda.

Em (ROLIA et al., 2003a, 2003b; ROLIA; ZHU; ARLITT, 2003), Rolia *et al* assumem que a demanda de recursos de aplicações pode ser estatisticamente prevista através da análise de dados históricos. Baseando-se em previsões realizadas com base em dados históricos de demanda de recursos é possível assegurar com alguma probabilidade que um recurso será cedido a uma aplicação quando ela precisar. No entanto, não há mecanismos que lidem com variações de carga inesperadas, que não tenham sido capturadas nos dados históricos.

Bennani e Menascé (BENNANI; MENASCE, 2005) propõem um sistema misto que engloba o *DPS* e o gerente global. Os autores consideram aplicações que recebem requisições de diversas classes e usam modelos analíticos de desempenho para decidir sobre a alocação dos recursos entre as aplicações. Uma função de utilidade global é usada para decidir sobre a alocação das máquinas entre diversas aplicações. A função de utilidade é maior quanto menores forem os tempos de resposta das aplicações do tipo *online* e quanto maiores forem as vazões das aplicações do tipo *batch*. Cada aplicação tem sua carga e desempenho monitorados periodicamente, e o gerente global usa estas informações e a função de utilidade para decidir sobre a alocação das máquinas entre as várias aplicações.

Em (ALMEIDA et al., 2006b; CUNHA et al., 2007) propõe-se um sistema de gerência de capacidade para ambientes compartilhados cujo objetivo é maximizar os lucros do centro de dados. Define-se um modelo de SLA (*Service Level Agreement*) em dois níveis: normal e sobrecarga. Penalidades são pagas quando, no nível normal, o SLA é violado; prêmios são recebidos quando capacidade extra é oferecida em momentos de sobrecarga. Modelos

analíticos de desempenho baseados em filas são associados ao modelo de penalidades e prêmios com o objetivo de decidir sobre a fração de recursos que será cedida a cada aplicação do centro de dados.

Em (ALMEIDA et al., 2006a) um outro modelo de SLA é proposto em que o cliente do centro de dados paga pelo que consome apenas quando o SLA não é violado. Considerando esta característica e os custos dos recursos necessários para manter tais SLAs sem violação decide-se sobre as frações de recursos que devem ser alocadas para cada classe de serviço Web. O objetivo do sistema de provisão dinâmica de recursos é minimizar as perdas de rendimentos devido a violações de SLAs e o custo de usar os recursos do centro de dados.

Chandra *et al* (CHANDRA; GONG; SHENOY, 2003b, 2003a) também apresentam um sistema de provisão dinâmica para ambientes de recursos compartilhados que combina um sistema de provisão dinâmica e um gerente global de recursos. O sistema proposto considera um centro de dados com recursos compartilhados e baseia-se na carga e nos tempos de resposta médios medidos após cada intervalo de medição. Com base nestes dados tenta-se prever a carga em um futuro próximo. Esta carga prevista, juntamente com os tempos de resposta medidos são usados para decidir sobre a quantidade de recursos a ser oferecida a cada aplicação do centro de dados, tendo como objetivo manter os tempos de resposta das aplicações inferiores ao tempo de resposta desejado.

Finalmente, em (HELLERSTEIN; KATIRCIOGLU; SURENDRA, 2005), os autores propõem um DPS que visa minimizar as penalidades sofridas por conta de violações de acordos de nível de serviço. Este DPS, baseia suas decisões em um modelo analítico de desempenho e um modelo analítico de custo. O modelo de desempenho considera a possibilidade de falha de servidores.

Um resumo das principais características dos sistemas de provisão dinâmica estudados são apresentados na Tabela 2.1.

É interessante observar que a informação de gerência "carga" está presente em quase todos os sistemas de provisão dinâmica de recursos e esta informação é combinada com outras informações. De posse destas informações o *DPS* decide sobre a quantidade de recursos a ser alocada à aplicação de forma que o valor de alguma métrica importante seja mantido em torno de um alvo - seja tempo de resposta, seja utilização de CPU, seja tamanho das filas.

Tabela 2.1: Sumário de características de alguns *DPSs*

DPS	*Objetivo*	*Métricas coletadas*	*Tipo de compartilhamento dos recursos*	*Realiza gerência global dos recursos*
Ranjan *et al* (RANJAN et al., 2002)	Manter a utilização em torno de um alvo	Utilização, quantidade de requisições recebidas e processadas	Servidores dedicados	Não
Chase *et al* (CHASE et al., 2001)	Manter a utilização em torno de um alvo e maximizar lucros	Utilização, tamanho da fila e vazão	Servidores compartilhados	Sim
Doyle *et al* (DOYLE et al., 2003)	Manter o tempo de resposta em torno de um alvo	Carga	Servidores compartilhados	Sim
Lassettre *et al* (LASSETTRE et al., 2003; NAKADAI; TANIGUCHI, 2007)	Manter o tempo de resposta em torno de um alvo	Carga	Servidores dedicados	Não
Nakadai *et al* (NAKADAI; TANIGUCHI, 2007)	Manter o tempo de resposta em torno de um alvo	Carga	Servidores dedicados e heterogêneos	Sim
Urgaonkar *et al* (URGAONKAR; SHENOY, 2005)	Maximizar os lucros do centro de dados	Carga	Servidores dedicados	Sim
Urgaonkar2 *et al* (URGAONKAR et al., 2005)	Não formar filas	Carga	Servidores dedicados	Não

Tabela 2.1: Sumário de características de alguns *DPSs*

DPS	*Objetivo*	*Métricas coletadas*	*Tipo de compartilhamento dos recursos*	*Realiza gerência global dos recursos*
Rolia *et al* (ROLIA et al., 2003a, 2003b; ROLIA; ZHU; ARLITT, 2003)	Garantir com alguma probabilidade que a aplicação terá o recurso quando precisar	Dados históricos de demandas da aplicação	Servidores compartilhados	Sim
Bennani *et al* (BENNANI; MENASCE, 2005)	Manter tempo de resposta pequeno	Carga e informações de desempenho	Servidores dedicados	Sim
Almeida, Cunha *et al* (ALMEIDA et al., 2006b; CUNHA et al., 2007; ALMEIDA et al., 2006a)	Maximizar os lucros do centro de dados	Carga	Servidores compartilhados	Sim
Chandra *et al* (CHANDRA; GONG; SHENOY, 2003b, 2003a)	Manter os tempos de resposta inferiores a um limiar	Carga e tempo de resposta	Servidores compartilhados	Sim

Recentemente, estudos começaram a ser realizados com o intuito de oferecer provisão dinâmica de recursos também para a camada de dados (TATON et al., 2006; SOUNDARARAJAN; AMZA, 2006). A provisão dinâmica desta camada envolve novos desafios, especialmente a migração dos dados que precisa ser realizada para que uma nova réplica do banco de dados seja ativada. Este trabalho de tese não considera a camada de dados como camada alvo e por isso não serão apresentados detalhes sobre tais sistemas. É interessante ressaltar, no entanto, que em (SOUNDARARAJAN; AMZA, 2006) o próprio sistema de provisão dinâmica é usado para reagir a falhas por parada dos servidores de dados acrescentando mais nós a esta camada específica. Não existe o conceito de rejuvenescimento e as falhas não são

recuperadas pelo sistema, que apenas reage a elas. A solução proposta nesta tese também considera que o sistema de provisão dinâmica pode ser usado para recuperar falhas.

2.2 Detecção e Recuperação de Faltas de Software

Nesta seção apresenta-se o estado da arte em detecção e recuperação de faltas intermitentes de software. Antes de discutir sobre estas pesquisas propriamente ditas, faz-se necessário classificar e definir tais faltas.

2.2.1 Classificação e definição de faltas de software intermitentes

No contexto deste trabalho, aplica-se a terminologia de faltas, erros e falhas definida por Avizienis *et al* (AVIZIENIS et al., 2004). A falha de um serviço é definida como a transição de um serviço correto para um serviço incorreto. Um serviço é caracterizado como incorreto quando pelo menos um dos seus estados externos - isto é, aqueles estados que são perceptíveis a partir da interface do serviço - desvia-se do comportamento correto. Ao desvio do estado correto para o incorreto dá-se o nome de erro. A causa de um erro é uma falta que foi ativada. As faltas podem permanecer em um estado de dormência. Neste caso, existe a falta, mas o erro e a falha não.

Quando se pode reproduzir a ativação de uma falta, diz-se que esta falta é *hard* (AVIZIENIS et al., 2004). Estas faltas são de natureza determinística e, conseqüentemente, podem ser mais facilmente detectadas e corrigidas. Estas faltas são também conhecidas como Bohrbugs (GRAY, 1986). Quando as faltas de software são ativadas por combinações complexas de fatores internos e ambientais, torna-se muito difícil reproduzi-las e corrigi-las. Estas faltas são classificadas como *soft* ou elusivas (AVIZIENIS et al., 2004) ou intermitentes, ou ainda, Heisenbugs (GRAY, 1986). A grande maioria das faltas que permanece com o software em produção são faltas elusivas. Dada a dificuldade de reprodução de tais faltas, elas podem passar despercebidas em todos os testes realizados durante a fase de desenvolvimento do software e manifestar-se durante a execução da aplicação.

Mais recentemente, as faltas de software relacionadas ao envelhecimento do software foram também definidas e classificadas (VAIDYANATHAN; TRIVEDI, 2001, 2005). As falhas de software relacionadas ao envelhecimento dos processos são causadas pelo acúmulo da ativação de faltas de software. Observou-se que, uma vez iniciada uma aplicação, condições potenciais de faltas acumulam-se com o tempo, levando a erros que se revelam como degradação de desempenho, falhas por parada ou ambos (VAIDYANATHAN; TRIVEDI, 2001). As

faltas de software relacionadas ao envelhecimento podem ser *hard* ou elusivas, dependendo da reprodutibilidade (VAIDYANATHAN; TRIVEDI, 2005).

2.2.2 Rejuvenescimento como prevenção e recuperação de faltas intermitentes de software

Já em 1986, Gray argumentava que a maioria das faltas de software são intermitentes, e ao reiniciar o programa em falha devido a tais faltas, o programa volta a funcionar adequadamente (GRAY, 1986). Nesta época, Gray referia-se aos Heisenbugs, que eram então responsáveis por 90% das falhas de software das aplicações.

Mais recentemente, em 1995, o reinício programado do software para lidar com faltas intermitentes recebeu o nome de rejuvenescimento de software (HUANG et al., 1995). O rejuvenescimento de um processo vai trazê-lo a um estado completamente operacional.

Os termos rejuvenescimento e reinício serão usados aqui como sinônimos. É comum encontrar na literatura a distinção entre o reinício pró-ativo, conhecido como rejuvenescimento, e o reativo, chamado simplesmente de reinício. Em geral, considera-se possível detectar a iminência de falhas causadas por faltas de envelhecimento, porém, em se tratando de Heisenbugs, deve-se reagir à falha, uma vez que tais faltas não oferecem sinais de que vão ser ativadas (VAIDYANATHAN; TRIVEDI, 2001). No contexto deste trabalho, no entanto, não importa se o reinício foi ou não pró-ativo; importa apenas que parte da aplicação deixa de operar por estar sendo reiniciada ou por estar em falha.

Desde que o rejuvenescimento foi proposto como uma medida preventiva e curativa para falhas de software, pesquisas começaram a ser realizadas com o intuito de identificar os tempos ótimos para rejuvenescer as aplicações. Todo rejuvenescimento implica em custo. Quando uma aplicação está sendo rejuvenescida ela fica indisponível ou tem a sua capacidade reduzida. Assim, se o rejuvenescimento ocorrer com mais freqüência que o ideal, vai incorrer em um custo ainda maior. Por outro lado, se ele ocorrer com menos freqüência que o ideal, o custo também pode ser alto, uma vez que a aplicação pode falhar ou ter seu desempenho degradado.

Duas abordagens são freqüentemente adotadas ao se estudar rejuvenescimento de software: (i) estudos feitos através de modelos analíticos e (ii) estudos baseados em medições. Nas subseções a seguir sistemas propostos na literatura são apresentados considerando se a metodologia usada.

2.2.3 Abordagem analítica

A maior parte dos estudos da área de rejuvenescimento de software faz uso de modelos analíticos para identificar os melhores momentos de rejuvenescer as aplicações gerenciadas. Nesta seção são apresentadas as características principais de tais modelos e alguns resultados obtidos através dos mesmos.

As principais características dos modelos analíticos estudados são:

1. *O* modelo de falha da aplicação. As aplicações podem falhar por parada (HUANG et al., 1995) ou por degradação do desempenho (GARG et al., 1996b; PFENING et al., 1996; AVRITZER; WEYUKER, 1997), ou ambos (GARG et al., 1998; TRIVEDI; VAIDYANATHAN; GOSEVA-POPSTOJANOVA, 2000; VAIDYANATHAN; TRIVEDI, 2005). A forma como as falhas são modeladas varia de um sistema para outro. Em alguns modelos a aplicação passa de um estado completamente operacional para um estado provável de falha (HUANG et al., 1995; GARG et al., 1997, 1998; DOHI; GOSEVA-POPSTOJANOVA; TRIVEDI, 2000; DOHI et al., 2002); em outros a aplicação vai tendo seu estado degradado gradualmente (BOBBIO; SERENO; ANGLANO, 2001; PFENING et al., 1996; AVRITZER; WEYUKER, 1997; TRIVEDI; VAIDYANATHAN; GOSEVA-POPSTOJANOVA, 2000; LI; VAIDYANATHAN; TRIVEDI, 2002). Quanto mais eficiente for a representação e captura do envelhecimento do software, mais eficiente é o sistema de rejuvenescimento (TRIVEDI; VAIDYANATHAN; GOSEVA-POPSTOJANOVA, 2000);

2. *P*olítica de rejuvenescimento. Esquemas de rejuvenescimento são geralmente classificados com base no evento que dispara o rejuvenescimento. Tais eventos podem ser baseados: (i) no tempo (por exemplo, todo domingo, às 4 horas da manhã); (ii) em métricas da aplicação (por exemplo, utilização de memória); ou (iii) na quantidade de trabalho já realizado (por exemplo, depois que x requisições forem processadas) (HUANG et al., 1995). Alguns sistemas consideram ainda a combinação de (i) e (ii);

3. *A* métrica de interesse. A métrica de interesse é a métrica que se pretende maximizar (ou minimizar - dependendo do seu significado) com o sistema de rejuvenescimento. Esta métrica é usada para comparar o sistema com e sem rejuvenescimento e para comparar políticas distintas de rejuvenescimento. Na maioria dos casos esta métrica é a disponibilidade dos sistemas ou quantidade de transações perdidas. O custo do rejuvenescimento está intimamente relacionado a esta métrica. Quanto melhor a decisão do sistema de rejuvenescimento, menor o custo do mesmo e conseqüentemente maior (ou menor, dependendo do significado) é o valor da métrica de interesse;

4. *A* duração da aplicação considerada. Em alguns trabalhos a aplicação gerenciada - aquela que será rejuvenescida periodicamente - é de longa duração (HUANG et al., 1995; PFENING et al., 1996; GARG et al., 1996b; AVRITZER; WEYUKER, 1997; GARG et al., 1998; VAIDYANATHAN; TRIVEDI, 2005; TRIVEDI; VAIDYANATHAN; GOSEVA-POPSTOJANOVA, 2000; DOHI; GOSEVA-POPSTOJANOVA; TRIVEDI, 2000; OKAMURA; MIYAHARA; DOHI, 2002) e em outros casos são aplicações com tempo finito de execução (GARG et al., 1996a). Neste trabalho, as aplicações de longa duração são consideradas, e portanto nesta seção apenas as pesquisas que envolvem tais aplicações serão apresentados.

Um dos trabalhos pioneiros na área de rejuvenescimento foi (HUANG et al., 1995). O modelo apresentado considera uma aplicação de longa duração que falha por parada e pode ser usado para encontrar o intervalo T_1 ótimo entre rejuvenescimentos. Em (GARG et al., 1995) o modelo apresentado por Huang é estendido, e tem como principal objetivo demonstrar que existe um custo associado ao rejuvenescimento dos sistemas. Mais tarde, Garg *et al* (GARG et al., 1996b, 1998) introduziram uma política de rejuvenescimento que leva em consideração não apenas o tempo, mas também a carga do sistema.

Garg *et al* (1996) estudam duas políticas de rejuvenescimento. A primeira é baseada apenas no tempo — já proposta anteriormente — e a segunda que considera também a quantidade de requisições no sistema. Neste segundo caso, uma vez passado um intervalo mínimo de tempo T_1 a aplicação é rejuvenescida apenas se a quantidade de requisições no sistema for inferior a uma dado limiar. Depois de passado um intervalo de tempo máximo T_2, então a aplicação é rejuvenescida independentemente da quantidade de requisições no sistema. O estudo analítico destes modelos indica que a probabilidade de haver requisições perdidas é menor quando a carga do sistema é levada em consideração. Isto ocorre porque evita-se o rejuvenescimento nos momentos de pico de carga, reduzindo-se assim a quantidade de requisições/pacotes perdidos por conta do rejuvenescimento.

Estas duas políticas analisadas em (GARG et al., 1996b) foram também estudadas por outros pesquisadores (GARG et al., 1997, 1998; TRIVEDI; VAIDYANATHAN; GOSEVA-POPSTOJANOVA, 2000). Em todos os estudos foram consideradas aplicações que recebem transações e observou-se que políticas de rejuvenescimento que levam em consideração a carga corrente da aplicação são mais adequadas. O modelo proposto em (GARG et al., 1998) é o mais poderoso, uma vez que considera os dois tipos de falha e permite, uma maior flexibilidade e poder ao modelá-las e permite também a modelagem de chegada e processamento das transações.

Pfening *et al* (1996) apresentam um modelo teórico que indica o melhor momento

de rejuvenescer uma aplicação cujas falhas de software causam degradação na taxa de serviço. O estado da aplicação é descrito pela quantidade de pacotes na fila da aplicação e o tempo que se passou desde o último rejuvenescimento. O modelo proposto indica, para um dado estado, se é melhor rejuvenescer a aplicação ou continuar o processamento. Usa-se então uma característica da aplicação - neste caso o tamanho da fila - para decidir se o rejuvenescimento deve ocorrer ou não no momento.

Bobbio *et al* (2001) também usam um parâmetro da aplicação para decidir sobre o rejuvenescimento. Naquele trabalho considera-se que o processo de degradação de processos devido a faltas de envelhecimento pode ser modelado como uma seqüência de faltas sucessivas que aumentam o valor de um parâmetro da aplicação. Quando o valor deste parâmetro ultrapassa determinado limiar, a aplicação falha. Sendo assim, na prática, ao monitorar este parâmetro é possível obter sinais de degradação e identificar o melhor momento de rejuvenescer a aplicação. Os autores sugerem duas políticas de rejuvenescimento: uma baseada no tempo e outra baseada em alertas. Na primeira tenta-se descobrir o tempo médio até que a aplicação passe do estado inicial robusto para um estado em que o parâmetro monitorado atinge um valor que leva a aplicação a falhar. Este tempo encontrado define o intervalo entre rejuvenescimentos. Na segunda política o parâmetro escolhido é monitorado periodicamente e quando ele ultrapassa um certo limiar o rejuvenescimento ocorre.

Dohi *et al* (DOHI; GOSEVA-POPSTOJANOVA; TRIVEDI, 2000; DOHI et al., 2002) consideram que a recuperação de falhas nem sempre rejuvenesce a aplicação, de forma que o rejuvenescimento é necessário mesmo após a recuperação de uma falha. Nestes trabalhos os autores apresentam modelos analíticos que decidem sobre o tempo fixo T_1 entre rejuvenescimentos (rejuvenescimento baseado no tempo), dado um modelo de falhas que é uma extensão do modelo apresentado em (HUANG et al., 1995). Os modelos tentam encontrar um T_1 que maximize a disponibilidade das aplicações.

Avritzer e Weyuker (1997) consideram uma aplicação cuja capacidade vai diminuindo com o passar do tempo. O sistema de rejuvenescimento proposto neste trabalho identifica quantos pacotes seriam perdidos devido à degradação do sistema e quantos seriam perdidos caso o sistema fosse rejuvenescido. A decisão é tomada de forma a minimizar a quantidade de pacotes perdidos. Na prática, é difícil convencer uma organização a forçar a indisponibilidade do serviço para restaurar o mesmo. Assim, uma característica importante dos sistemas aptos a terem sua capacidade restaurada é que seja possível restaurar parte da capacidade do sistema sem deixar o serviço indisponível. Isto ocorre, por exemplo, em aplicações que são executadas em clusters.

O uso de rejuvenescimento de software em sistemas de clusters vem se mostrando efi-

ciente, aumentando a disponibilidade dos sistemas (XIE; HONG; TRIVEDI, 2004; CASTELLI et al., 2001; VAIDYANATHAN et al., 2001). Em (VAIDYANATHAN et al., 2001; CASTELLI et al., 2001) são apresentados modelos que podem ser usados para avaliar o impacto de diferentes políticas de rejuvenescimento na disponibilidade de sistemas de clusters. Dois modelos são apresentados. O modelo que representa rejuvenescimento baseado no tempo, que pode ser usado para determinar o melhor intervalo de tempo fixo entre rejuvenescimentos. O outro modelo introduz a possibilidade de realizar rejuvenescimento baseado no estado corrente do sistema. Em (XIE; HONG; TRIVEDI, 2004) as 2 técnicas de rejuvenescimento propostas por Garg (GARG et al., 1996b) foram reavaliadas considerando sistemas de cluster e, mais uma vez, a técnica que leva em consideração a carga mostrou-se mais eficiente.

2.2.4 Abordagem baseada em medições

Os estudos de rejuvenescimento baseados em modelos analíticos precisam que o modelo de falhas da aplicação seja conhecido. Quanto melhor for esta modelagem, mais eficientes são os modelos propostos. Nesta seção são apresentados trabalhos que incentivam políticas de rejuvenescimento baseados em medições, em cujo caso, parâmetros observáveis da própria aplicação gerenciada são usados para decidir sobre o melhor momento de rejuvenescer a aplicação.

Em (VAIDYANATHAN; TRIVEDI, 1999; TRIVEDI; VAIDYANATHAN; GOSEVA-POPSTOJANOVA, 2000; LI; VAIDYANATHAN; TRIVEDI, 2002) são apresentadas abordagens baseadas em medições para estimar o momento em que recursos do sistema operacional vão se exaurir devido ao envelhecimento dos processos. Sistemas de rejuvenescimento não são apresentados, mas os autores motivam o surgimento de políticas de rejuvenescimento baseadas em medições.

Sistemas de rejuvenescimento baseados em medições foram propostos em (CASTELLI et al., 2001; VAIDYANATHAN et al., 2001; HONG et al., 2002; TAI et al., 2005; AVRITZER; BONDI; WEYUKER, 2005, 2007; MATIAS; FILHO, 2006).

Em (CASTELLI et al., 2001; VAIDYANATHAN; TRIVEDI, 2001) apresenta-se uma política de rejuvenescimento para sistemas de clusters baseada em medições. Neste caso, o rejuvenescimento é executado quando a aplicação entra em um estado de degradação ou um estado provável de falha. A técnica baseada em medições mostrou-se mais eficiente para sistemas de clusters que o rejuvenescimento baseado no tempo. Um sistema de rejuvenescimento para clusters da IBM foi apresentado em (CASTELLI et al., 2001).

Em (HONG et al., 2002) é apresentada uma política de rejuvenescimento baseada na quantidade de memória livre no sistema, tendo como alvo servidores Web. Foram deter-

minados vários níveis para a quantidade de memória livre e dependendo da quantidade corrente o rejuvenescimento ocorre também em níveis diferentes. Os autores propõem rejuvenescimento parcial em nível de aplicação (em que alguns processos são reiniciados) e o reinício de hardware.

Mais recentemente, o servidor Web Apache foi novamente experimentado (MATIAS; FILHO, 2006). Observou-se que quando páginas dinâmicas são requisitadas e a carga do sistema está alta ocorre envelhecimento. O desempenho do servidor em termos de tempo de resposta piora subitamente, chegando a atingir valores na ordem de minutos. Assim como em (HONG et al., 2002), os rejuvenescimentos ocorrem quando um limiar de quantidade de memória em uso é atingido e demonstrou-se bastante eficiente.

Tai *et al* (TAI et al., 2005) apresentam um sistema de rejuvenescimento que tem como alvo aplicações distribuídas que precisam manter o estado. O rejuvenescimento deve ser realizado de tal forma que não torne inconsistentes os dados das várias réplicas que executam a aplicação. Os autores propõem uma mistura de rejuvenescimento com base em métricas da aplicação e no tempo. Ao longo da execução da aplicação, o sistema vai conhecendo padrões comportamentais que são precursoras de falhas. Quando estes padrões são encontrados a réplica é rejuvenescida. Adicionalmente, o sistema escolhe randomicamente um momento para rejuvenescer as réplicas seguindo uma ordem circular.

Em (AVRITZER; BONDI; WEYUKER, 2005; AVRITZER et al., 2006) uma aplicação de comércio eletrônico é considerada. Enquanto todos os trabalhos anteriores usam como métrica algum recurso do sistema que vai se perdendo (vazando) com o passar do tempo, a métrica considerada por Avritzer *et al* é o tempo de resposta da aplicação. De forma simplificada, o rejuvenescimento ocorre quando o tempo de resposta coletado está N desvios padrões maior que a média aceitável por K intervalos de medição consecutivos. O valor de N e de K podem ser constantes ou valores dinamicamente escolhidos.

A virtualização foi usada em (SILVA et al., 2007) como um meio de aumentar a disponibilidade e a vazão do Tomcat. Métricas de sistema são coletadas e quando limiares são atingidos o rejuvenescimento é realizado. Para tal, um outro servidor (virtual) é ativado antes que o servidor a ser rejuvenescido seja reiniciado. O servidor em falha deve terminar as requisições já presentes nele antes de ser reiniciado. Enquanto ocorre o reinício o servidor *backup* processa as requisições da aplicação. Esta técnica de rejuvenescimento limpo é muito semelhante à técnica de rejuvenescimento coordenado proposta nesta tese (vide Capítulo 6) e em (LOPES; CIRNE; BRASILEIRO, 2004).

2.2.5 Micro-reinícios

Candea *et al* (CANDEA; FOX, 2001; CANDEA et al., 2003, 2004) propõem microrejuvenescimentos (μRejuvenescimento) para lidar com falhas de software. Esta técnica realiza o reinício de pequenos componentes da aplicação. Os μRejuvenescimentos oferecem muitos dos benefícios do rejuvenescimento e são mais baratos, uma vez que são mais rápidos e envolvem apenas pequeníssimos componentes da aplicação.

Os μRejuvenescimentos, no entanto, não são aplicáveis em quaisquer *e-services*. Para que um *e-service* possa ser microrejuvenescido é preciso projetá-lo e implementá-lo seguindo certas regras (CANDEA; FOX, 2003). As aplicações aptas ao microrejuvenescimento devem, por exemplo, ser formadas por componentes bem pequenos, fracamente acoplados, que mantêm seus estados em locais dedicados. Além disso, a comunicação entre componentes deve ser tal que se um componente que está temporariamente indisponível for acessado, o componente que o chamou saberá que deverá esperar um pouco e então retransmitir a mensagem. Desta forma, os usuários finais da aplicação são pouco afetados pelos μRejuvenescimentos.

Quando o μRejuvenescimento não recupera a aplicação da falha, então o rejuvenescimento tradicional é aplicado (CANDEA et al., 2004). Nem todas as falhas são corrigidas através do μRejuvenescimento e em alguns casos o μRejuvenescimento apenas posterga a realização do rejuvenescimento.

Os resultados com μRejuvenescimento são bastante satisfatórios. No entanto, como as aplicações atuais ainda não estão 100% preparadas para o μRejuvenescimento, e como mesmo aplicando-se o μRejuvenescimento, o rejuvenescimento tradicional de processos ainda é muitas vezes necessário, o trabalho proposto nesta tese considera apenas o rejuvenescimento tradicional de processos. De fato, caso o μRejuvenescimento venha a ser aplicado em todos os *e-services* sem a necessidade do rejuvenescimento, o trabalho desta tese torna-se menos importante, uma vez que os μRejuvenescimentos são rápidos e não diminuem a capacidade da aplicação. É importante ressaltar, no entanto, que será necessário algum tempo até que todas as aplicações legadas venham a ser substituídas.

2.3 Interação entre sistemas de controle

Estudos de interações entre sistemas de controle distintos e independentes que atuam sobre um mesmo alvo ou que têm objetivos conflituosos vêm sendo realizados por pesquisadores em diversas áreas.

O surgimento e disseminação de redes de cobertura (*overlay networks*) sobre a rede

IP (Internet Protocol) possibilita que dois sistemas de controle distintos e independentes decidam sobre o roteamento dos dados. Em (KERALAPURA; TAFT; IANNACCONE, 2004) interações entre estes sistemas distintos são estudadas. Estas interações levam principalmente a oscilações nas rotas entre sistemas autônomos quando falhas ocorrem. Tais oscilações tornam as matrizes de tráfego mais dinâmicas e conseqüentemente mais difíceis de ser estimadas. Erros nesta estimativa são propagados para erros nas tarefas de engenharia de tráfego dependentes da matriz. Além disso, quando existem redes de cobertura que cobrem diferentes sistemas autônomos, o estado da rede em um sistema pode influenciar o estado da rede no outro sistema autônomo, o que não é desejável. O trabalho proposto nesta tese também tem por objetivo estudar interações entre sistemas de controle que atuam sobre um mesmo alvo e, portanto, estão sujeitos a interações não previstas.

Um exemplo de sistemas com objetivos conflituosos é apresentado em (NARASIMHAN, 2002). Se uma aplicação complexa apresenta requisitos de tempo real e tolerância a faltas simultaneamente, então algumas relações conflituosas precisam ser resolvidas. De fato, muito se estudou para se produzir um *middleware* que ofereça garantias de tempo real e também para um *middleware* tolerante a faltas. Porém, quando estes dois requisitos existem simultaneamente a união pura e simples do que foi estudado para ambos os casos isoladamente não é satisfatória, pois existem claramente requisitos conflituosos, alguns deles identificados em (NARASIMHAN, 2002; STANKOVIC; WANG, 1992). Manter simultaneamente a previsibilidade temporal requerida pelos sistemas de tempo real e a consistência forte entre as réplicas de um sistema tolerante a faltas é um desafio. Por exemplo, tanto sistemas de tempo real quanto sistemas tolerantes a faltas requerem uma certa ordenação das operações a serem realizadas. No entanto, cada um destes sistemas ordena estas operações de acordo com critérios diferentes e inconsistência das réplicas pode ocorrer quando estas ordens estão em conflito (WANG et al., 2002). Semelhantemente, a hipótese desta tese é que a união pura e simples de sistemas de provisão dinâmica e de rejuvenescimento que atuam sobre o mesmo alvo não é eficiente, apesar de ambos, individualmente apresentarem um bom desempenho. Em alguns momentos, ações conflituosas podem ocorrer.

Graupner *et al* (2003) atentam para as interações que existem entre os sistemas de gerência tradicionais e os sistemas de virtualização surgiram como um sistema à parte dos sistemas de gerência. A virtualização de recursos computacionais permite que recursos sejam compartilhados entre várias aplicações distintas, porém todas elas têm a impressão de que possuem recursos dedicados. A camada de virtualização traz mudanças que atingem premissas básicas aceitas pelos sistemas de gerência. Por exemplo, esta camada tem a autoridade de constantemente modificar as associações entre recursos subjacentes (que

podem ser os recursos físicos e lógicos, agrupados ou combinados) e entidades virtualizadas, sem que o sistema de gerência tome conhecimento. A associação entre aplicações e recursos subjacentes em ambientes virtualizados muda com mais freqüência do que nos ambientes tradicionais em que não há virtualização. Como estas associações são realizadas sob o controle da camada de virtualização, o sistema de gerência tradicional perde essas associações e informações sobre os recursos subjacentes não mais podem ser correlacionadas às aplicações. Graupner *et al* sugerem que é difícil separar a virtualização da gerência e que estas atividades deveriam ser feitas de forma combinada. Mais especificamente, a camada de virtualização deve ser considerada mais um sistema de gerência, e que deve interagir com os demais sistemas de gerência, cada um capaz de avaliar o estado dos componentes gerenciados, tomar decisões e agir sobre estes componentes. Esta visão de vários sistemas de gerência interagindo entre si é exatamente a visão considerada neste trabalho. As interações entre estes sistemas precisam ser estudadas para que eles possam conviver sinergeticamente.

Na área de teoria de controle estuda-se como sistemas de controle independentes agindo sobre uma mesma planta (sistema controlado) e na presença de incertezas atuam de forma que o estado global desejado do sistema seja atingido. Este problema é conhecido como o problema do controle adaptativo descentralizado (MUKHOPADHYAY, 1999). De acordo com (NARENDRA; OLENG, 2002; MUKHOPADHYAY, 1999; MUKHOPADHYAY; NARENDRA, 1999) sistemas de controle distribuídos precisam de alguma forma trocar informações para reduzir o erro das ações de controle. Em (MUKHOPADHYAY; NARENDRA, 1999), por exemplo, os sistemas de controle atuam em momentos distintos e pré-planejados do tempo e cada vez que um sistema de controle C_i decide a respeito de uma entrada de controle os demais sistemas são comunicados de tal decisão. Em (NARENDRA; OLENG, 2002) o sistema de controle C_i de um subsistema conhece o estado desejado dos demais subsistemas. Enfim, existe um esforço extra que precisa ser despendido para que a coordenação de sistemas de controle independentes ocorra. De fato, problemas de controle adaptativo descentralizado estão sendo cada vez mais comuns e envolvem muitas áreas da ciência e tecnologia, por exemplo, redes de computadores de larga escala (MUKHOPADHYAY; NARENDRA, 1999). Segundo Mukhopadhyay e Narendra (MUKHOPADHYAY; NARENDRA, 1999) este problema se intensifica quando existe um grau acentuado de incerteza envolvendo o sistema dinâmico sendo controlado. Os *e-services* envolvem incertezas tais como falhas de software e variabilidade de carga. A necessidade de um esforço extra que garanta a união eficiente dos sistemas de provisão dinâmica e de rejuvenescimento é, de forma semelhante, enfatizado e buscado nesta tese.

Finalmente, no campo da psicologia humana estudos recentes revelam que existe uma integração entre a cognição e a emoção. Estes sistemas são vistos como sistemas de controle independentes, cada um exercendo funções específicas, que interagem entre si de forma seletiva (GRAY, 2004). Por exemplo, no calor da emoção a habilidade humana de regular o próprio pensamento pode ser diminuída. Ao vivenciar um momento de ansiedade suave o ser humano tem sua capacidade de memória espacial aumentada (GRAY, 2004). Estados emocionais e o controle cognitivo fazem parte de um mesmo time; eles são separáveis, mas não completamente. Para que dois sistemas sejam integrados é preciso mostrar que o todo não é simplesmente a soma das partes. Evidências desta natureza são mostradas com relação aos sistemas de gerência escolhidos para estudo nesta tese.

2.3.1 Relacionamento entre provisão dinâmica de recursos e rejuvenescimento de software

Os sistemas de provisão dinâmica de recursos visam evitar a super-provisão das aplicações enquanto garantem uma dada qualidade de serviço. Espera-se que a quantidade média de recursos provida a uma aplicação dinamicamente seja menor que a quantidade de recursos oferecida a uma aplicação super-provisionada. Sendo assim, os sistemas de provisão dinâmica levam ao:

1. Barateamento do custo de manter as aplicações operando;

2. Aumento da utilização das máquinas que executam as aplicações (se comparada à utilização dos recursos quando a aplicação está sendo estaticamente super-provisionada).

Todavia, quanto mais saturados os recursos, mais propensa a falhas fica a aplicação (GRIBBLE, 2001). Durante a sobrecarga, a aplicação fica mais propensa, por exemplo, a erros tais como condições de corrida e mal comportamento dos coletores de lixo. Segundo (XIE; HONG; TRIVEDI, 2004; VAIDYANATHAN; TRIVEDI, 2005) o tempo que uma aplicação permanece em um estado completamente operacional após um reinício é proporcional à carga que ela recebe. Seja A um *e-service*. Suponha que em t_0 duas instâncias de A, A_1 e A_2, foram implantadas e iniciadas. A instância A_1 está recebendo uma carga média de x; e a instância A_2 está recebendo uma carga média de nx, com $n > 1$. Os resultados obtidos em (VAIDYANATHAN; TRIVEDI, 2005) validam a idéia de que A_2 irá falhar antes de A_1. Ao tempo médio entre falhas consecutivas de uma aplicação dá-se o nome de tempo médio entre falhas (MTBF - *Mean Time Between Failures*).

Pensando sob este aspecto, é factível aceitar a idéia de que o MTBF das aplicações cujos recursos são providos dinamicamente é menor que o MTBF das aplicações super-provisionadas. Sob este ponto de vista, é possível concluir que a quantidade de rejuvenescimentos que ocorre quando um *DPS* está operando pode ser maior que a quantidade de rejuvenescimentos necessários quando a super-provisão estática está sendo realizada.

Uma vez que o *DPS* tenta manter os recursos operacionais de uma aplicação não sub-utilizados, a retirada de um recurso pelo sistema de rejuvenescimento poderá levar a aplicação a um desempenho aquém do esperado. Quando componentes da aplicação falham, seja por parada, seja por desempenho, a qualidade de serviço da aplicação também fica aquém da esperada. Os algoritmos de provisão dinâmica propostos na literatura não estão preparados para lidar com falhas de software, ou com a retirada de componentes da aplicação para rejuvenescimento. Na prática, a eficiência da provisão dinâmica é diminuída quando falhas ocorrem, ou quando uma máquina é retirada sem o conhecimento do *DPS*.

A própria atuação do *DPS* pode ser considerada uma ação de rejuvenescimento: componentes da aplicação passam a ser inseridos e liberados da aplicação. A retirada de recursos da aplicação, no entanto, é feita de forma aleatória, isto é, todos os nós têm a mesma probabilidade de serem retirados da aplicação. Sendo assim, nós em um estado completamente operacional podem ser retirados, enquanto nós em estado provável de falha podem permanecer na aplicação.

Como visto na Seção 2.2.3, uma política bastante estudada e eficiente de rejuvenescimento é atrasar o rejuvenescimento para um momento em que a carga da aplicação é inferior a um certo limiar. Esta política se mostra eficiente porque o ambiente considerado usa provisão estática de recursos, e quando a carga está reduzida, a retirada de um nó não é tão prejudicial. No entanto, quando um sistema de provisão dinâmica está em execução esta política de rejuvenescimento não faz muito sentido, uma vez que a diminuição da carga não leva à super-provisão da aplicação, sendo acompanhada da diminuição também da capacidade da aplicação.

Finalmente, ambos os sistemas precisam saber como diferenciar falhas de software de sobrecarga. Caso esta diferenciação não seja realizada com sucesso, dois problemas podem ocorrer. O primeiro ocorre quando falhas em componentes da aplicação confundem o *DPS*, que resolve adicionar mais máquinas. Este aumento da capacidade da aplicação pode melhorar a qualidade de serviço da aplicação, mas aumenta o custo da mesma e não impede que usuários sejam afetados pelos componentes em falha. Um segundo problema ocorre quando o sistema de rejuvenescimento é confundido por componentes saturados. Os componentes saturados estão mais propensos a falhas e também oferecem um desempenho

aquém do normal esperado. Reiniciar componentes saturados irá piorar ainda mais a qualidade de serviço da aplicação, pois a capacidade da mesma será diminuída.

Diante do exposto, acredita-se que existem interações entre carga e falhas e entre sistemas de provisão dinâmica e de rejuvenescimento que precisam ser melhor investigadas.

2.4 Especificação do problema e objetivo desta tese

Diante do exposto, percebe-se claramente a necessidade de se estudar de forma detalhada a atuação dos sistemas de provisão dinâmica e rejuvenescimento e como eles podem colaborar entre si. Apresenta-se nesta seção mais formal e detalhadamente o problema que esta tese se propõe a estudar e o que se espera alcançar com a solução proposta.

2.4.1 Definição do problema

Dois fatos básicos motivam o desenvolvimento deste trabalho:

- Existe uma ampla classe de aplicações (os *e-services*) que precisam ser controladas tanto no aspecto de provisão dinâmica de recursos quanto no aspecto de rejuvenescimento de software;
- Tradicionalmente, sistemas de provisão dinâmica de recursos e sistemas de rejuvenescimento de software são construídos e estudados de forma isolada e independente.

Seja *qos* uma métrica usada para medir a qualidade de serviço (QoS - *Quality of Service*) de um *e-service* qualquer *e*. Sem perda de generalidade, convenciona-se aqui que quanto maior *qos*, melhor é a qualidade de serviço da aplicação medida. Um exemplo prático de métrica que poderia representar *qos* é a disponibilidade de uma aplicação.

Seja *AWoF* um *e-service* livre de faltas de software que está sendo executado sobre uma infra-estrutura dinâmica, modificada pelas ações de um *DPS* (*Dynamic Provisioning System*), que é um sistema de provisão dinâmica de recursos. O custo, em termos de quantidade de recursos necessária para executar a aplicação, é reduzido devido à ação do *DPS*, se comparado ao custo da mesma aplicação super-provida estaticamente de recursos.

Esta redução de custo alcançada quando o *DPS* atua, no entanto, pode não ser gratuita. Ela pode vir acompanhada de uma diminuição da qualidade de serviço (QoS - *Quality of Service*) da aplicação se, mais uma vez, comparada à qualidade de serviço que seria alcançada no caso em que houvesse super-provisão estática de recursos.

Considera-se que esta diminuição da qualidade de serviço se dá especialmente em momentos em que o *DPS*, por não ser um sistema perfeito, decide por uma quantidade de recursos aquém da realmente necessária para se manter um certo nível de qualidade de serviço. A esta perda de QoS devido à ação do *DPS* chama-se l_{DPS}.

A aplicação *AWoF* totalmente livre de faltas de software, no entanto, não existe na prática. O que existem são aplicações que possuem faltas de software que podem ser ativadas levando a condições de erro e possivelmente a falhas em um ou mais nós que executam a aplicação, sejam de desempenho, sejam falhas por parada. Não existindo a aplicação *AWoF* torna-se necessária a intervenção de um outro sistema de gerência para detectar e recuperar os nós em falha automaticamente. A tal sistema dá-se o nome de *RS* (*Rejuvenation System*).

O *RS* vem sendo estudado considerando-se uma infra-estrutura estática, que geralmente é superprovida de recursos, isto é, uma infra-estrutura preparada para atender as maiores cargas de trabalho esperadas. Apesar da ação do *RS*, é possível que a qualidade de serviço da aplicação *AWF* (aplicação com faltas de software) não seja a máxima possível, por conta do efeito das falhas antes do rejuvenescimento e também por conta dos próprios rejuvenescimentos, que têm um custo associado. Sendo assim, ao ser gerenciada por um *RS*, a qualidade de serviço da aplicação *AWF* pode não ser a máxima possível por duas razões: (i) pela ação das falhas antes do rejuvenescimento e (ii) pelos rejuvenescimentos que diminuem a capacidade da aplicação enquanto ocorrem. A estas perdas de qualidade de serviço devido a falhas e rejuvenescimentos chama-se l_{RS}.

Considerando o *e-service AWF*, é necessária a atuação tanto do *DPS* quanto do *RS*. Mas ao fazer estes sistemas atuarem simultaneamente, perde-se a infra-estrutura superprovida estaticamente com a qual o *RS* lidava. Além disso, o *DPS* passa a atuar sobre uma aplicação cujos nós podem falhar e serem rejuvenescidos pelo *RS*. Ambos os sistemas deixam de atuar nas condições perfeitas para as quais foram estudados.

Esta nova situação leva às seguintes indagações: neste novo cenário, como l_{DPS+RS} se compara às perdas l_{DPS} e l_{RS}? Existem novos fatores que precisam ser considerados? Existem interações que fazem com que as perdas devido ao uso do *DPS* e as perdas devido a falhas e rejuvenescimentos sejam maiores quando *DPS* e *RS* atuam juntos?

A hipótese motivadora desta tese é que existem momentos em que as perdas na qualidade de serviço da aplicação devido à atuação do *DPS* (l_{DPS}) e as perdas devido a falhas e ao rejuvenescimento (l_{RS}) aumentam quando os sistemas atuam simultaneamente.

Alguns fatores apóiam a hipótese apresentada. Primeiramente, quando o *DPS* atua, espera-se que a utilização dos recursos seja maior, aumentando o efeito negativo da falha

de n nós, se comparado ao efeito visto caso a super-provisão estivesse sendo realizada. Em segundo lugar, ao retirar n nós para serem rejuvenescidos, a ausência de tais nós será mais acentuada em um ambiente dinamicamente provido de recursos do que em um ambiente super-provido estaticamente de recursos. Finalmente, pode-se imaginar interações entre *DPS* e *RS* que os levam a tomar decisões diferentes das decisões que tomariam caso estivessem agindo isoladamente em seus ambientes ideais. Por exemplo, nós podem ser rejuvenescidos por conta de sobrecarga em momentos de aumento substancial da carga.

Considerando a aplicação *AWF* e a atuação simultânea do *DPS* e do *RS*, pode-se identificar momentos em que tais sistemas podem agir de forma coordenada, melhorando a qualidade de serviço da aplicação.

O *RS* tem por objetivo rejuvenescer nós da aplicação. Pode-se considerar, sem perda de generalidade, que o rejuvenescimento é a desativação e posterior ativação do mesmo nó. Assim, munido de alguma informação extra (que pode ser oferecida pelo *RS*), o próprio *DPS* pode ser capaz de rejuvenescer nós em falha quando apropriado. Sendo assim, é plausível acreditar que pode-se coordenar as atividades de *DPS* e *RS*. Esta é a segunda hipótese que motivou este trabalho. É possível coordenar as atividades do *DPS* e do *RS* gerando um relacionamento simbiótico entre eles e tornando o sistema coordenado mais eficiente que a união pura dos sistemas.

Ao coordenar as atividades de *DPS* e *RS* leva-se ao surgimento de um sistema combinado *DPS*$\star$*RS*. Este sistema combinado deve gerenciar a aplicação de tal forma que a qualidade de serviço é maior que a qualidade de serviço obtida sem coordenação. Além disso, espera-se que o custo da aplicação gerenciada por este novo sistema coordenado, em termos de número de nós ativos, não seja maior que o custo da aplicação gerenciada pelos sistemas *DPS* e *RS*, sem coordenação.

O trabalho desenvolvido e apresentado neste documento visou não apenas evidenciar a não ortogonalidade de *DPS* e *RS*, mas também identificar uma coordenação adequada das ações destes sistemas de forma que eles possam colaborar entre si.

Nos capítulos que seguem são apresentados resultados analíticos e experimentais realizados com o objetivo de testar a validade de tais hipóteses. No Capítulo 4, um modelo analítico é proposto e usado para mostrar que rejuvenescimentos afetam muito menos as aplicações super-providas de recursos do que as aplicações que executam sobre uma infra-estrutura dinâmica provida por um *DPS*.

No Capítulo 6 são apresentados resultados de experimentos de simulação que indicam que as perdas devido à atuação do *DPS* (l_{DPS}) e as perdas devido a falhas e rejuvenescimentos (l_{RS}) são maiores quando *DPS* e *RS* atuam simultaneamente do que quando eles

atuam isoladamente.

Técnicas de coordenação são propostas e avaliadas no Capítulo 7. Observaram-se ganhos em termos de disponibilidade e de tempo de resposta ao aplicar a coordenação, especialmente nos períodos em que falhas e rejuvenescimentos ocorrem.

Os resultados de simulações indicam que, mesmo sendo o *DPS* perfeito, isto é, $l_{DPS} = 0$, as perdas observadas ao unir o *DPS* perfeito e o *RS* sem coordenação são maiores que as perdas observadas quando apenas o *RS* atua. Neste cenário, em que o *DPS* é perfeito, também foram observados ganhos em termos de qualidade de serviço quando as técnicas de coordenação foram aplicadas (vide Capítulos 6 e 7).

Capítulo 3

Modelo analítico

É importante para este trabalho estudar o comportamento transiente do *e-service* quando ocorre rejuvenescimento. O rejuvenescimento é considerado uma situação transiente, uma vez que ele modifica o valor da taxa de serviço total da aplicação enquanto ocorre e depois que termina. Neste capítulo será estudado em que condições a qualidade de serviço de aplicações piora enquanto o rejuvenescimento ocorre.

Seja j um nó que está apresentando falha de desempenho ao executar uma aplicação e, portanto, precisa ser rejuvenescido. O rejuvenescimento de j vai durar um tempo t_{rejuv} somado ao tempo necessário para processar as requisições residentes no nó. Durante este tempo a taxa de serviço da aplicação será reduzida e o nó j não será capaz de atender requisição alguma, exceto as já presentes nele. Após o rejuvenescimento, a taxa de serviço da aplicação volta a ser a taxa máxima, caso nenhum outro nó esteja apresentando falhas.

Durante o rejuvenescimento de um nó, os demais nós deverão receber as requisições que seriam processadas pelo nó sendo rejuvenescido. Os nós que não estão sendo rejuvenecidos podem ou não ter capacidade suficiente para processar a demanda extra sem afetar a qualidade de serviço da aplicação.

Este capítulo está organizado como segue. Na próxima seção um modelo analítico simples é apresentado. Na seção seguinte, aplica-se este modelo para analisar situações em que o rejuvenescimento afeta a qualidade de serviço da aplicação. Finalmente, conclusões são apresentadas na Seção 3.3.

3.1 Modelo analítico baseado em rede de filas aberta

A Figura 3.1 apresenta um esquema gráfico do modelo analítico proposto. Em um instante t qualquer, existem $r(t)$ nós selecionados para executar o *e-service*. Estes nós são compostos

por filas com disciplina FCFS (*First Come, First Served*) e capacidade b. Em um instante t, requisições chegam a uma taxa $\lambda(t)$. Cada requisição tem uma demanda de serviço d (em unidades de tempo) e portanto todos os nós oferecem teoricamente a mesma taxa máxima de serviço dada por $\mu(t) = \frac{r(t)}{d}$. Quando já existem b requisições em um dado nó, este nó não poderá mais aceitar requisições. Estas requisições rejeitadas são direcionadas para um centro de atraso S_{drop}. Convencionou-se chamar de $\gamma(t)$ a taxa de chegada de requisições em S_{drop}, que é a taxa de rejeição de requisições.

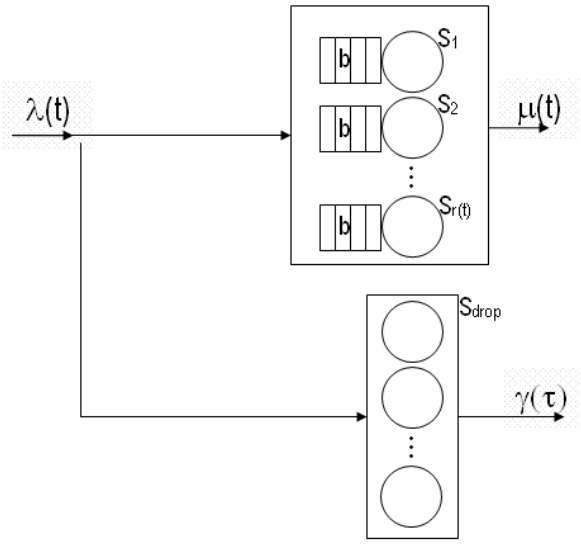

Figura 3.1: Modelo de um *e-service* usando uma rede de filas

Seja r a quantidade total de nós ativos, isto é, que devem estar executando o *e-service*. Destes, r_{fail} nós estão apresentando falha de desempenho, de forma que, para tais nós, a demanda de serviço das requisições é dada por $d_{fail} > d$. Por conta desta degradação no desempenho, os r_{fail} nós serão rejuvenescidos. No início do rejuvenescimento, existem n_0 requisições em cada nó saudável e $n_{0,fail}$ requisições em cada nó em falha. O rejuvenescimento durará um tempo t_{rejuv}, que é a soma do tempo de concluir o processamento das requisições presentes nos nós sendo rejuvenescidos e do tempo necessário para reiniciar a aplicação nos nós ($t_{restart}$).

Definidas estas estas variáveis, desenvolveram-se equações que indiquem em que circunstâncias a disponibilidade e o tempo de resposta da aplicação serão comprometidos devido ao rejuvenescimento de r_{fail} nós.

Considera-se que as requisições são distribuídas igualmente entre os servidores durante o intervalo de tempo em estudo. Como as requisições apresentam a mesma demanda para todos os nós saudáveis e em falha, tem-se que todos os servidores saudáveis apresentam a

mesma carga de trabalho em qualquer instante t e todos os servidores em falha apresentam a mesma carga de trabalho em qualquer instante t.

Para simplificar o modelo, considera-se que, se $r_{fail} > 0$, então o rejuvenescimento inicia-se no tempo $t = 0$. Assim, apenas dois estados são possíveis para o *e-service*: (i) r_{fail} nós estão sendo rejuvenescidos ou (ii) todos os r nós estão saudáveis e ativos.

A taxa de serviço do sistema em um instante t está diretamente relacionada ao número de nós ativos neste instante e ao estado de cada um destes nós. Se todos os nós estivessem sempre ativos e saudáveis, esta taxa seria facilmente representada por $\mu(t) = \frac{r}{d}, \forall t$.

No entanto, quando o rejuvenescimento ocorre, o valor de r varia em função de t, e a taxa de serviço da aplicação no instante t pode ser dada pela Equação 3.1.

$$\mu(t) = \frac{r(t)}{d} \tag{3.1}$$

Onde:

$$r(t) = \begin{cases} r, & \text{Se nenhum nó estiver sendo rejuvenescido} \\ r - r_{fail}, & \text{Se } r_{fail} \text{ nós estiverem sendo rejuvenescidos} \end{cases} \tag{3.2}$$

Esta função apresenta uma descontinuidade. Um exemplo do gráfico desta função é apresentado na Figura 3.1.

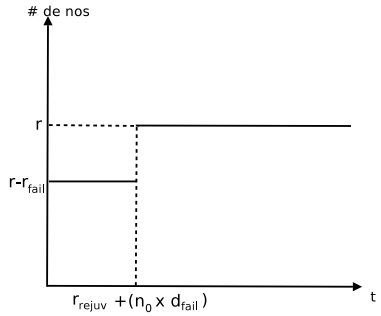

Figura 3.2: Quantidade de nós ativos ao longo do tempo quando $r_{fail} > 0$

Como a taxa de processamento pode variar ao longo do tempo, então é possível que a disponibilidade e o tempo de resposta do *e-service* sejam afetados por conta desta variação. Este é o assunto da próxima seção, porém, antes de iniciá-la, é interessante definir disponibilidade e tempo de resposta.

No contexto deste modelo, a disponibilidade de uma aplicação no intervalo de tempo $[a, b]$ é dada pela Equação 3.3.

$$A_{[a,b]} = \frac{\int_a^b \lambda(t)dt - \int_a^b \gamma(t)dt}{\int_a^b \lambda(t)dt} \tag{3.3}$$

O numerador desta equação indica o número total de requisções completadas com sucesso no intervalo de tempo $[a, b]$ e o denominador indica a quantidade total de requisições recebidas pela aplicação.

O tempo de resposta da aplicação no intervalo $[a, b]$ é dado pelo tempo médio em que as requisições que chegaram neste intervalo permanecem no sistema. Assim, quanto maiores os tamanhos das filas maior será o tempo de resposta da aplicação.

3.2 Quando o rejuvenescimento pode afetar a qualidade de serviço da aplicação

Nesta seção será apresentada uma análise do modelo analítico apresentado na seção anterior. O cenário considerado é o seguinte:

- A quantidade r de nós é suficiente para atender a demanda $\lambda(t)$, $\forall\ t$ desde que todos os nós estejam ativos e saudáveis;
- Se r_{fail} nós forem rejuvenescidos, então é possível que $\lambda(t) > \mu(t)$ no intervalo $[0, t_{rejuv}]$;

Acredita-se que este cenário é factível, uma vez que tradicionalmente, sistemas de provisão dinâmica esforçam-se para adequar perfeitamente a carga à demanda, mantendo sempre que possível o sistema estável (isto é, $\lambda(t) \leq \mu(t)$). Como $\lambda(t)$ pode ser menor que $\mu(t)$, então percebe-se que existe a possibilidade de haver uma provisão extra. Esta provisão extra pode, no entanto, não ser suficiente para manter $\lambda(t) \leq \mu(t)$ quando r_{fail} nós forem rejuvenescidos. Neste caso, o sistema passa a ficar temporariamente instável.

Seja t_{over} o tempo necessário para encher as filas dos $r - r_{fail}$ nós ativos quando r_{fail} nós estiverem sendo rejuvenescidos. Como pressupõe-se que os $r - r_{fail}$ nós poderão não ter capacidade suficiente para garantir a estabilidade do sistema durante o rejuvenescimento, então as filas dos $r - r_{fail}$ nós ativos começarão a crescer, numa taxa α segundo a Equação 3.4.

$$\alpha = \max\left(0, \frac{\overline{\lambda}_{rejuv}}{r - r_{fail}} - \frac{1}{d}\right) \tag{3.4}$$

Nesta equação, $\overline{\lambda}_{rejuv}$ representa a taxa média de chegada de requisições na aplicação durante o rejuvenescimento. Para deixar o modelo menos rígido, aceita-se que esta taxa seja diferente da taxa de chegada após o rejuveenscimento. A fração $\frac{\overline{\lambda}_{rejuv}}{r-r_{fail}}$ indica a taxa de chegada em cada nó que permanece ativo e $\frac{1}{d}$ caracteriza a taxa de saída de requisições em tais nós.

Como já existiam n_0 requisições em tais nós no início do rejuvenescimento, então chega-se à Equação 3.5 como a equação que define em quanto tempo as filas dos nós ativos ficarão completamente cheias:

$$\begin{aligned} n_0 + \alpha \times t_{over} &= b \\ t_{over} &= \frac{b - n_0}{\alpha} \end{aligned} \tag{3.5}$$

Como a taxa de crescimento das filas pode ser nula, então define-se a seguinte equação para definir t_{over}:

$$t_{over} = \begin{cases} \frac{b-n_0}{\alpha}, & \text{Se } \alpha > 0 \\ \infty, & \text{Se } \alpha = 0 \end{cases} \tag{3.6}$$

Nota-se que quanto maior for a taxa de crescimento das filas durante o rejuvenescimento e o número de requisições já presentes nos servidores, então menor é t_{over}. Quanto maior a capacidade dos nós (b), maior é t_{over}.

O tempo total de rejuvenescimento é dado pela Equação 3.7. Este tempo é dado pela soma do tempo necessário para completar as requisições que estavam nos nós em falha no início do rejuvenescimento e do tempo necessário para efetivamente reiniciar a aplicação no nó.

$$t_{rejuv} = (n_{0,fail} \times d_{fail}) + t_{restart} \tag{3.7}$$

Se a taxa de crescimento das filas não for nula e o tempo total necessário para a realização do rejuvenescimento dos r_{fail} nós ultrapassar t_{over}, então o rejuvenescimento afetará não apenas o tempo de resposta, mas também a disponibilidade, uma vez que requisições *começarão* a ser rejeitadas. Diante do exposto tem-se que sempre que a Inequação 3.8 for verdadeira, a disponibilidade será afetada por conta do rejuveenscimento.

$$\frac{b - n_0}{\alpha} < (n_{0,fail} \times d_{fail}) + t_{restart} \tag{3.8}$$

Em relação ao tempo de resposta, um outro estudo pode ser realizado. Considerando que existe um limiar aceitável para o tempo médio de resposta dado por $x \times d$, então é possível identificar se este limiar é violado durante o rejuvenescimento.

Se $x > b$, então o limiar do tempo de resposta de requisições aceitas nunca será afetado. Quando $x < b$, no entanto, este limiar será afetado sempre que $n_0 \leq x$ e $n(t_{rejuv}) > x$, onde $n(t_{rejuv})$ é a quantidade de requisições presentes em cada nó que não foi rejuvenescido imediatamente após o término do rejuvenescimento.

Esta quantidade final de requisições na fila dos nós saudáveis imediatamente após o rejuvenescimento é dada pela Equação 3.9.

$$n(t_{rejuv}) = \max(0, n_0 + \alpha \cdot t_{rejuv}) \tag{3.9}$$

Quando a quantidade final de requisições na fila ultrapassa o limite x, então pode-se identificar através da Equação 3.10 o momento em que esta violação ocorreu (t_x).

$$t_x = \frac{x - n_0}{\alpha} \tag{3.10}$$

Sempre que o rejuvenescimento durar mais que t_x então o limiar de tempo de resposta será violado. Esta inequação é apresentada em 3.11.

$$\frac{x - n_0}{\alpha} < (n_{0,fail} \times d_{fail}) + t_{restart} \tag{3.11}$$

Terminado o rejuvenescimento, o sistema não volta a trabalhar normalmente. Leva-se um tempo, chamado aqui t_{under} até que as filas dos nós que permaneceram ativos voltem a tamanhos aceitáveis, isto é, menores que x. Seja δ, a taxa de redução de cada uma destas filas após o rejuvenescimento. A Equação 3.12 mostra como calcular δ. Nesta equação, $\overline{\lambda}$ é a taxa média de chegada de requisições após o rejuvenescimento.

$$\delta = \frac{1}{d} - \frac{\overline{\lambda}}{r} \tag{3.12}$$

Considerando esta taxa de redução, calcula-se t_{under} de acordo com a Equação 3.13.

$$t_{under} = \frac{n(t_{rejuv}) - x}{\delta} \tag{3.13}$$

Na seção a seguir este modelo é instanciado para que seja possível identificar se violações do tempo de resposta ou da disponibilidade podem ser causadas pelo rejuvenescimento de r_{fail} nós.

3.2.1 Exemplo de instanciação do modelo

Seja, por exemplo uma aplicação cuja demanda de serviço média das requisições é 30 milisegundos. Esta é a mesma demanda usada como exemplo em (HELLERSTEIN; KATIRCIOGLU;

SURENDRA, 2005). Esta aplicação recebe uma taxa média de chegada de 150 requisições por segundo (valor que está dentro do intervalo considerado em (RANJAN et al., 2002; KANODIA; KNIGHTLY, 2003; HELLERSTEIN; KATIRCIOGLU; SURENDRA, 2005).

Seja b igual a $1024 + 250$. 1024 é o tamanho *default* da capacidade de *backlog* de conexões TCP/IP no Linux e 250 é, por *default*, o número máximo de clientes atendidos simultaneamente no servidor Web Apache e no JBoss.

Sejam ainda:

- r igual a 5;
- n_0 igual a 1;
- r_{fail} igual a 1;
- d_{fail} igual $0,21$ segundos.

Estes últimos dados indicam que dentre os cinco nós ativos, existe um que está com o desempenho degradado, de forma que as requisições que chegam neste nó demoram muito mais para serem processadas. Diversas ocorrências de degradação de desempenho em aplicações já foram estudados (LOU; TANG, 2007; AVRITZER; BONDI; WEYUKER, 2005; LI; VAIDYANATHAN; TRIVEDI, 2002; GRIBBLE, 2001; DRUSCHEL; BANGA, 1996). Estas degradações podem ser da ordem de microsegundos (LI; VAIDYANATHAN; TRIVEDI, 2002) até a ordem de segundos (JIA; CHEN; CAI, 2006; AVRITZER; BONDI; WEYUKER, 2005). Faltas no sistema operacional, faltas no middleware e faltas na aplicação propriamente podem ocorrer, levando o sistema a um estado de erro que pode causar falhas de desempenho, reduzindo substancialmente a capacidade de processamento da aplicação. Quanto maior a utilização do servidor, maior a probabilidade da ativação de tais faltas.

Como limiares de disponibilidade e tempo de resposta foram considerados os valores 99,99% (mínimo) e 4 segundos (máximo) respectivamente. Este limiar de tempo de resposta só é satisfeito quando as filas dos servidores são menores ou iguais a 133, sendo este, portanto, o valor de x.

Conhecendo-se os valores acima mencionados, descobre-se que t_{over} é 305 segundos, enquanto t_{rejuv}^{total} é 307 segundos. Portanto, durante o rejuvenescimento as filas dos nós saudáveis se tornam completamente cheias, e há perda de requisições com consequente diminuição da disponibilidade da aplicação durante o rejuvenescimento.

No início do rejuvenescimento, os nós saudáveis ofereciam um tempo de resposta médio de 0,03 segundos, uma vez que não havia filas em tais nós. No fim do rejuvenescimento este tempo de resposta passa a ser 30 segundos.

Para este exemplo, serão necessários 336 segundos até que os nós que permaneceram ativos se recuperem e passem a oferecer um tempo de resposta inferior ao limiar aceitável.

Neste caso, o rejuvenescimento tornou saudáveis os nós em falha, mas sobrecarregou de tal forma os demais nós que a impressão que se pode ter a partir do estado de cada nó no fim do rejuvenescimento é que há r_{fail} nós saudáveis e $r - r_{fail}$ nós em falha.

Se o número de nós ativos fosse maior, isto é, se a aplicação estivesse super-provida de recursos, então o custo do rejuvenescimento em termos de qualidade de serviço da aplicação seria menor. Considerando o mesmo exemplo analisado, porém com $r = 10$, então, no fim do rejuvenescimento, nem a disponibilidade nem o tempo de resposta da aplicação teriam piorado, uma vez que os 9 nós que permanecem ativos têm capacidade suficiente para lidar com a carga de trabalho da aplicação.

3.2.2 Analisando o rejuvenescimento coordenado

A técnica de rejuvenescimento coordenado pode ser expressa da seguinte forma:

- O rejuvenescimento coordenado diminui o tempo de rejuvenescimento de $t_{rejuv} + (n_{0,fail} \times d_{fail})$ para t_{migr}, onde t_{migr} é o tempo necessário para ativar um novo nó. Sempre que $t_{rejuv} + (n_0 \times d_{fail}) > t_{migr}$ então a técnica de coordenação estará contribuindo para a melhor qualidade de serviço da aplicação, especialmente se $t_{restart} + (n_0 \times d_{fail}) > t_{over}$ ou $t_{restart} + (n_0 \times d_{fail}) > t_x$.

Considerando um tempo médio de migração de 34 segundos[1] o que aconteceria se o rejuvenescimento coordenado tivesse sido aplicado no exemplo apresentado na seção anterior? Observa-se que a disponibilidade não teria sido afetada, o tempo de resposta no fim do rejuvenescimento seria 4,3 segundos (um pouco acima da esperada) e em apenas 3 segundos todos os nós já estariam oferecendo um tempo de resposta aceitável.

A outra técnica, de redução coordenada de capacidade, não pode ser avaliada segundo este modelo analítico, uma vez que é necessária a redução de capacidade para analisar esta técnica. No entanto, é possível, diante do modelo exposto, concluir que esta técnica será útil, uma vez que pode-se evitar ou reduzir a quantidade de rejuvenescimentos, ao mesmo tempo em que os nós mais saudáveis (que oferecem maior capacidade de processamento à aplicação) têm maior probabilidade de permaneceram ativos.

[1] Tempo médio de iniciação do JBoss em um computador com uma CPU Intel Pentium 4 de 3.0 GHz, com 400 MB de memória RAM.

3.3 Conclusões parciais

Algumas simplificações foram assumidas como verdade para que o modelo fosse simples. A principal delas foi considerar que *todas as requisições são iguais e demandam um tempo de serviço d dos servidores*. A principal consequência desta simplificação é que todos os nós recebem exatamente a mesma carga e se comportam exatamente da mesma forma ao longo do tempo. Na prática, a carga de cada nó poderá ser diferente, mesmo que todos os nós tenham a mesma capacidade, uma vez que a demanda de cada requisição é diferente. Assim, é possível que alguns nós estejam mais ou menos sobrecarregados que outros em determinados instantes. Esta simplificação, no entanto, não invalida o modelo, uma vez que espera-se do balanceamento de carga perfeito exatamente este comportamento. Na prática, quando existirem nós mais ou menos sobrecarregados, então estes nós serão mais ou menos afetados durante o rejuvenescimento dos r_{fail} nós.

Uma outra simplificação foi considerar que a disponibilidade é afetada apenas quando recursos se exaurem e passam a não mais aceitar requisições. Na prática, acredita-se que quanto mais sobrecarregada uma aplicação, mais propensa a falhas ela está (GRIBBLE, 2001; LOPES; CIRNE; BRASILEIRO, 2004). Tais aplicações são mais sucetíveis a condições de corrida, mal comportamento dos coletores de lixo etc., aumentando a probabilidade da ativação de faltas não determinísticas. Sendo assim, a diminuição da disponibilidade na prática pode ser maior que a apresentada no modelo.

Com este modelo é possível identificar situações em que o rejuvenescimento tem um custo elevado em termos de qualidade de serviço da aplicação. É claro que o rejuvenescimento deve existir, pois é necessário para recuperar falhas de software. No entanto, percebe-se que é possível diminuir o custo de tais rejuvenescimentos em se tratando de aplicações onde há provisão dinâmica de recursos, como foi mostrado na Seção 3.2.2.

Capítulo 4

Modelo de simulação

4.1 Modelo de simulação

No primeiro capítulo foi apresentada a metodologia usada para a realização deste trabalho de tese. Naquele momento foi citado um modelo baseado em componentes. Ainda naquele capítulo foram apresentadas as razões pelas quais um modelo de simulação seria usado como técnica de avaliação. Cada componente deste modelo foi apenas citado, mas nenhum detalhe sobre seu funcionamento foi apresentado.

Nesta seção cada componente do modelo é descrito. Em seguida o modelo é instanciado para a realização de projetos experimentais. Estes projetos experimentais objetivaram identificar os parâmetros mais relevantes do modelo em relação à métrica disponibilidade da aplicação.

4.1.1 A aplicação

A aplicação e é um *e-service* de uma camada, conforme ilustrado na Figuras 4.1(a). A camada única da aplicação é executada em um *cluster* com balanceamento de carga. Aplicações deste tipo são horizontalmente escaláveis, pois é possível mudar sua capacidade ao adicionar/retirar nós do cluster. Quando um nó está ativo, ele recebe requisições dos clientes.

Em cada instante t, existem $r(t)$ nós ativos na camada única da aplicação. O balanceador de cargas recebe requisições de clientes, isto é, do LGS (*Load Generator System*), e as redireciona para um dos $r(t)$ nós ativos da aplicação obedecendo uma ordem circular (*round robin*).

A taxa de chegada de requisições no instante t é dada por $\lambda(t)$, e pode ser extremamente

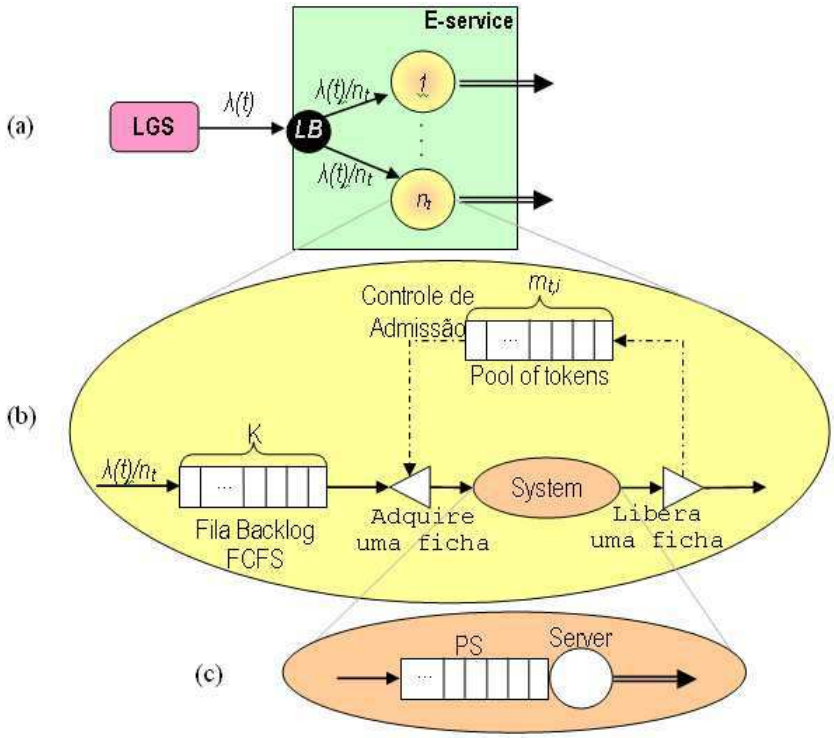

Figura 4.1: O modelo da aplicação

variável, tais quais cargas Web e de comércio eletrônico (BARFORD; CROVELLA, 1998; CROVELLA; BESTAVROS, 1996; ARLITT; KRISHNAMURTHY; ROLIA, 2001; MENASCÉ et al., 2003).

Servidores disponíveis no mercado, como Apache, JBoss e Tomcat, executam múltiplos processos ou múltiplas linhas de execução (*threads*) que processam as requisições recebidas. Cada processo/linha de execução computa uma requisição por vez. A quantidade máxima de processos/linhas de execução determina o nível máximo de concorrência atingido pelo servidor. Os administradores de sistemas são encorajados a mudar este número de forma que o servidor esteja adeqüadamente sintonizado com o poder dos recursos computacionais sobre os quais ele executa.

Aplicando esta mesma idéia, em cada nó i e cada instante t, existe um controle de admissão, composto por um depósito de $m_{t,i}$ fichas. O bloqueio do serviço antes da realização do mesmo diante da indisponibilidade temporária dos recursos necessários para servi-la é modelada aqui segundo o modelo proposto por Menascé, Almeida e Dowdy (2004). Para que uma requisição seja efetivamente servida, ela precisa antes adquirir uma ficha de admissão. Esta ficha fica indisponível, até que a requisição seja completamente servida. Requisições que chegam no nó quando todas as fichas estão em uso esperam em uma fila chamada *Backlog*, até que fichas sejam liberadas. Fichas são liberadas pelas requisições que forem

completamente servidas pelo sistema de fila *Server*. A fila *Backlog* tem capacidade limitada K e obedece uma disciplina "primeiro a chegar, primeiro a ser servido" (*first-come, first-served* - FCFS). Requisições que chegam quando a fila *Backlog* está cheia são rejeitadas. Este sistema de controle de admissão está ilustrado na Figura 4.1(b).

Quando uma requisição obtém uma ficha, ela é recebida por um sistema de fila chamado *Server*, que a processará. Na prática, esta requisição seria servida por uma rede de filas composta por recursos de hardware e de software. No entanto, identificar que partes desta rede de filas estão sendo gargalos foge do escopo deste trabalho. O objetivo neste contexto é apenas de representar a influência que o número de requisições presentes no sistema exerce sobre os tempos de respostas destas requisições. Portanto, considera-se que todas as requisições admitidas no nó são recebidas por uma fila única chamada *Server* e compartilham deste servidor de forma igualitária. Desta forma, se uma requisição é processada em z unidades de tempo quando apenas ela está presente no sistema *Server*, ela passa a ser processada em média em $z \times n$ unidades de tempo quando $n(t)$ requisições iguais a ela estão presentes no servidor no intervalo $(t, t + (z \times n)]$. O sistema *Server* está ilustrado na Figura 4.1(c).

O sistema de fila *Server* obedece uma disciplina de compartilhamento de servidor (*Processor Sharing* - PS), pois esta é a disciplina mais próxima da aplicada pelos sistemas operacionais. Esta disciplina é idêntica à *round robin*, exceto pelo fato de que cada vez que um cliente tem acesso ao servidor, o tempo de serviço obtido por ele é bem pequeno, fazendo com que todas as requisições progridam simultaneamente (MENASCE; ALMEIDA; DOWDY, 2004).

4.1.2 O sistema de geração de carga

O sistema gerador de carga *LGS* deve enviar para o balanceador de cargas da aplicação requisições de clientes. Este sistema deve ser capaz de gerar diferentes tipos de carga, sejam elas constantes ou variáveis. Cada requisição q enviada pelo LGS para o balanceador de cargas deve estar acompanhada de seu tempo de serviço inerente (d_q) e do tempo decorrido entre esta e a próxima requisição. O tempo de serviço inerente indica a quantidade de tempo de serviço mínima necessária no sistema *Server* para que a requisição seja servida.

Na prática, existem duas técnicas distintas de geração de tráfego (KANT; TEWARI; IYER, 2001): a emulação de usuários e a geração de tráfego agregado. A emulação de usuários requer várias linhas de execução que simulam o comportamento de um usuário que está usando a aplicação. A geração de tráfego agregado pode ser realizada através de uma única linha de execução que gera o tráfego tal qual visto no lado servidor.

Quando a emulação de clientes está sendo aplicada, torna-se difícil controlar o tráfego agregado que chega ao servidor, pois o envio de uma requisição só ocorre quando a requisição anterior for satisfeita (KANT; TEWARI; IYER, 2001). A carga oferecida por cada cliente reduz-se quando o servidor está saturado, oferecendo tempos de resposta grandes. Assim, a habilidade de controlar a carga recebida pelo servidor é reduzida.

Neste estudo, deseja-se garantir que a carga recebida no servidor é variável, pois só assim a presença e atuação de um sistema de provisão dinâmica é justificada. Por esta razão, optou-se por aplicar a geração de carga agregada.

Arquivos de *trace*

O sistema gerador de carga recebe como entrada um arquivo chamado arquivo de *trace*. Cada linha deste arquivo representa uma requisição e contém dois campos:

1. O tempo entre requisições. O tempo que se passa entre a chegada da requisição anterior e a requisição corrente;

2. O tempo de serviço inerente da requisição no sistema *Server*.

Desta forma, este sistema gerador de carga é suficientemente simples e poderoso para gerar diferentes tipos de carga, exceto as que consideram o comportamento individual dos usuários, conforme discutido na seção anterior.

A qualidade da carga gerada depende diretamente da atividade de geração do arquivo de *trace*. Este arquivo pode ser gerado com base em ferramentas que tentam modelar a carga semelhante à carga Web real ou com base em distribuições bem conhecidas. Os arquivos de trace usados nesta tese foram gerados através de uma ferramenta de geração de carga agregada de comércio eletrônico chamada GEIST (KANT; TEWARI; IYER, 2001).

4.2 O sistema de injeção de erros

O sistema de injeção de erros (*SEIS*) modifica o comportamento da aplicação de forma que se possa simular o efeito de faltas de software. Faltas podem permanecer em um estado dormente, ou podem ser ativadas levando a condições de erro. Condições de erro podem levar o sistema a uma falha quando o estado externo do sistema é violado (AVIZIENIS et al., 2004).

O *SEIS* introduz erros de software na aplicação gerenciada. Estes erros podem levar a dois tipos de falha: falhas por degradação de desempenho e falhas por parada. As

falhas por degradação de desempenho levam uma aplicação a oferecer um desempenho aquém do esperado. As falhas por parada tornam a aplicação completamente indisponível, impedindo-a de processar as requisições recebidas.

Nesta instanciação do modelo, falhas de software são modeladas mudando-se os tempos de serviço inerentes que acompanham as requisições com base em uma função de degradação particular.

4.2.1 Função de degradação

Aplicações podem falhar de diferentes formas e em diferentes freqüências. É possível encontrar na literatura referências de faltas de software em aplicações Web que levam a uma degradação de desempenho. Por exemplo, diversos experimentos foram realizados com o servidor Web Apache e detectou-se que, mesmo servindo apenas páginas estáticas, o desempenho do servidor degrada 0,06 ms por hora (LI; VAIDYANATHAN; TRIVEDI, 2002) em certas situações de carga. Mais recentemente descobriu-se no Websphere erros de software que, quando ativados, degradam o desempenho a uma taxa de 5% por hora (LOU; TANG, 2007). Em (JIA; CHEN; CAI, 2006) também são apresentados experimentos com o Apache que caracterizaram faltas de envelhecimento que levam a uma alta degradação do desempenho do servidor.

De fato, acredita-se que a ocorrência de faltas de software ocorre devido ao estímulo sofrido pelas aplicações e não apenas pela passagem natural do tempo (VAIDYANATHAN; TRIVEDI, 2005). Seja $\delta_i(t)$ a quantidade de requisições processadas pelo nó i desde seu último reinício até o instante t. A função de degradação deve levar o sistema a uma degradação tanto maior quanto maior for o número de requisições processadas pelo nó.

A função de degradação incrementa a demanda de serviço inerente de cada requisição (d_q) com um valor $S^i_{aging}(t)$ como mostra a Equação 4.1. A nova demanda de serviço da requisição q no nó i passa a ser d^i_q. Este tempo de serviço será o demandado pela requisição no sistema *Server* do nó i para ser satisfeita.

$$d^i_q = d_q + S^i_{aging}(t) \tag{4.1}$$

A função $S^i_{aging}(t)$ pode mudar para refletir diferentes níveis de falha. Considerando, por exemplo, o estudo de caso realizado por Li *et al* (LI; VAIDYANATHAN; TRIVEDI, 2002), a função S_{aging} seria $S^i_{aging}(t) = 4,3 \times 10^{-8}\delta_i(t)$. Neste caso, se uma requisição com tempo inerente de serviço de 10 ms fosse recebida por um nó i que já processou 2 milhões de requisições, então, esta requisição demandaria, na realidade, $10,086$ ms neste nó.

4.3 O sistema de provisão dinâmica

O sistema de provisão dinâmica (*Dynamic Provisioning System - DPS*) modifica a capacidade da aplicação para acomodar a carga corrente. Todos os sistemas de provisão dinâmica de recursos estudados formam um laço de controle com realimentação que atua periodicamente sobre a aplicação, conforme apresentado na Seção 2.1.2. O que muda de um sistema para o outro são as informações de monitoração coletadas e a função objetivo perseguida. Assim, considera-se que o sistema de provisão dinâmica do modelo também apresente as características comuns a todos os sistemas estudados.

A saída de um *DPS* determina o número de nós que devem estar ativos no próximo intervalo de adaptação. Quando este valor é maior que o número de nós atualmente ativos, nós devem migrar de um conjunto de servidores livres para a aplicação. Caso contrário, nós devem ser liberados da aplicação e voltar para o conjunto de nós livres. Quando um nó vai ser desativado, o balanceador de cargas para imediatamente de lhe redirecionar requisições, mas o nó só é completamente desativado quando todas as requisições presentes nele são completadas (RANJAN et al., 2002).

Dois sistemas diferentes de provisão dinâmica de recursos foram aplicados em diferentes estudos experimentais. Um deles é um sistema perfeito que conhece a carga de trabalho da aplicação *a priori*, podendo, portanto, tomar decisões precisas sobre a capacidade da aplicação. O segundo sistema utiliza-se de um algoritmo simples que recebe como parâmetros informações de monitoramento da aplicação e é capaz de decidir sobre a capacidade da aplicação. Como este sistema não conhece a carga da aplicação e sim tenta prevê-la com base em informação passada, ele nem sempre toma decisões precisas.

4.3.1 O sistema perfeito de provisão dinâmica

Seja $\overline{d}(t)$ a demanda média de serviço de cada requisição recebida no intervalo $[t, t + \Delta t]$. Assim, cada nó é capaz de processar, em média, $\frac{1}{\overline{d}(t)}$ requisições por unidade de tempo no intervalo $[t, t + \Delta t]$. Seja $\overline{\lambda}(t)$ a carga média em requisições por unidade de tempo que deve ser recebida pela aplicação no intervalo de tempo $[t, t + \Delta t]$. Considerando estas variáveis, tem-se que no intervalo de tempo $[t, t + \Delta t]$ é preciso que pelo menos $r = \overline{\lambda}(t) \times \overline{d}(t)$ nós estejam ativos. Assim, o *PDPS* decide sobre a quantidade r de nós a ser ativada segundo a Equação 4.2.

$$r = \left\lceil \overline{\lambda}(t) \times \overline{d}(t) \right\rceil + e \tag{4.2}$$

O acréscimo de e nós introduz uma margem de provisão extra para a aplicação. Caso

estes nós não fossem acrescentados, a utilização dos nós ativos tenderia a 100%, e os tempos de resposta poderiam ser muito grandes.

Independente de haver falhas ou não em alguns nós ativos, o *PDPS* sempre decide pela mesma quantidade de nós, pois esta decisão é feita com base apenas na carga recebida e na capacidade dos nós. Isto não acontecia no sistema de provisão dinâmica usado em simulações anteriores, em que a utilização dos nós ativos era também considerada ao tomar decisões de provisão de recursos. Assim, neste capítulo, não será possível realizar uma análise a respeito dos custos do sistema em termos de número de nós ativos, uma vez que se a carga é a mesma, a quantidade de nós usada também é a mesma.

4.3.2 Algoritmo de provisão dinâmica

Nesta instanciação do modelo, o sistema de provisão dinâmica tem por objetivo manter a utilização dos nós ativos da aplicação em torno de uma utilização alvo, tal qual o sistema apresentado em (RANJAN et al., 2002). O algoritmo proposto por Ranjan *et al* foi escolhido por prezar simplicidade e apresentar todas as características principais de um sistema de provisão dinâmica: forma um laço de controle fechado e usa a carga da aplicação como um dos dados de entrada.

O algoritmo de provisão dinâmica coleta informações de monitoração a cada intervalo de adaptação $(t - \Delta t_{adapt,DPS}, t]$. Depois de cada intervalo de adaptação um componente monitor do *DPS* coleta dos nós as seguintes medições:

- X, o número de requisições completamente processadas no último intervalo de adaptação;
- H, o número de requisições que chegaram no último intervalo de adaptação;
- U, a utilização média dos servidores *Server* considerando todos os nós ativos no último intervalo de adaptação.

Dados $r(t)$, o número corrente de nós ativos, e ρ_{target}, a utilização alvo, o *DPS* computa o número adequado de nós para o próximo intervalo como segue (RANJAN et al., 2002):

1. Calcula a utilização média por requisição como sendo $D = \frac{U}{X}$;
2. Calcula a utilização normalizada como sendo $U' = max(H, X) \cdot D$;
3. Calcula o número de nós necessários para que ρ_{target} seja atingido como sendo $N' = \lceil \frac{N \cdot U'}{\rho_{target}} \rceil$.

4.4 O sistema de rejuvenescimento

Como apresentado na Seção 2.2, sistemas de rejuvenescimento não formam, necessariamente, um laço de controle. Estes sistemas podem ser configurados para agir com base no tempo decorrido desde o último rejuvenescimento, ou na quantidade de requisições processadas desde o último rejuvenescimento, ou podem monitorar a aplicação e formar um laço de controle com realimentação.

Os sistemas de rejuvenescimento que tomam decisões com base em informações de monitoração são mais complexos, mas têm, teoricamente, maior chance de identificar melhores momentos para o rejuvenescimento (VAIDYANATHAN et al., 2001). Estimula-se na literatura o surgimento de sistemas de rejuvenescimento baseados em medição (TRIVEDI; VAIDYANATHAN; GOSEVA-POPSTOJANOVA, 2000) e por esta razão, o sistema simulado nesta atividade é baseado em medições. Acredita-se que as faltas de software que ocorrem em aplicações em produção estão diretamente associadas à carga da aplicação. Assim, a ocorrência das falhas de software podem não seguir um padrão bem definido, não sendo propício o uso de algoritmos baseados no tempo decorrido.

No contexto deste trabalho, um sistema de rejuvenescimento forma um laço de controle com realimentação. O sistema monitora cada componente da aplicação e detecta/prevê falhas nestes componentes. Componentes que apresentam comportamento diferente do esperado/aceito por f intervalos de adaptação são rejuvenescidos. Considera-se que os componentes monitorados e sujeitos ao reinício são processos nós ativos do *e-service*.

4.4.1 Algoritmo de rejuvenescimento

No período em que esta atividade foi desenvolvida os sistemas baseados em medição existentes usavam métricas de utilização de recursos que não estavam sendo simulados, mais especificamente, memória livre do sistema (HONG et al., 2002; LI; VAIDYANATHAN; TRIVEDI, 2002). Apenas Avritzer, Bondi e Weyuker (2005) consideravam o rejuvenescimento baseado em monitoração de métricas que afetam a qualidade de experiência dos usuários da aplicação. Por exemplo, tempo de resposta. Para manter a simplicidade do modelo, optou-se por elaborar um sistema de rejuvenescimento baseado em medição especialmente para lidar com o tipo de erro sendo simulado.

O sistema de rejuvenescimento age periodicamente, após intervalos de adaptação. Passado um intervalo de adaptação $(t - \Delta t_{adapt,RS}, t]$, o RS recupera as informações de monitoração que lhe são úteis e decide sobre quais nós precisam ser reiniciados.

Para cada nó ativo i, o RS recupera as seguintes informações após cada intervalo de

adaptação:

- $A_i(t)$, a disponibilidade do nó i no intervalo;
- $R_i(t)$, o tempo médio de resposta das requisições processadas pelo nó i.

De posse destas informações, o *RS* decide sobre quais nós precisam ser reiniciados. Um nó i é reiniciado quando a disponibilidade ou o tempo de resposta medidos no nó violam valores limiares de disponibilidade e de tempo de resposta por f intervalos consecutivos de adaptação.

Devido à natureza das falhas neste modelo - apenas falhas por degradação de desempenho - o sistema de rejuvenescimento espera que todas as requisições já presentes terminem antes de efetivamente reiniciar o nó (CANDEA; FOX, 2001). De fato, segue-se um esquema bastante semelhante ao aplicado pelo *DPS* ao liberar um nó. Quando o nó é selecionado para rejuvenescimento, o balanceador de cargas (*LB*) para de enviar requisições para este nó. Apenas quando todas as requisições presentes no nó são satisfeitas, o nó é efetivamente reiniciado e volta então a fazer parte dos nós ativos no balanceador de cargas.

Em outras situações, em que não se conhece a natureza das falhas, ou que falhas por parada ocorrem, esta técnica não seria favorável, pois não haveria garantias de que o nó realmente terminaria de processar as requisições presentes nele, e, além disso, não haveria garantias de que estas requisições seriam corretamente processadas.

4.5 Conclusões parciais

O modelo de simulação apresentado nesta seção parece factível. No entanto, muitas possibilidades de configuração são possíveis, aumentando muito o número de cenários possíveis que precisariam ser estudados.

Com o objetivo de reduzir este espaço de possibilidades, um estudo experimentalfoi realizado. A descrição completa de como este projeto foi realizado encontra-se no Apêndice A.

O projeto de experimentos permitiu identificar os fatores mais importantes para cada uma das três composições estudadas, a saber: *AWoF+DPS*, *AWF+RS* e *AWF+DPS+RS*.

À luz dos resultados obtidos, foram eliminados os seguintes fatores:

1. Tempo médio de migração em composições em que o *DPS* atua. Apresentou a mesma tendência nos cenários estudados;
2. Intensidade das falhas em composições em que o *RS* atua. Apresentou a mesma tendência - já esperada - em todos os cenários estudados;

3. Tempo de reinício em composições em que o *RS* atua. Foi praticamente irrelevante em todos os cenários estudados;

4. Número de nós ativos nos experimentos em que o *DPS* não atua. Quanto maior o número de nós melhor a disponibilidade (maior super-provisão);

5. Tempo de adaptação do *DPS*. Quanto menor o tempo de adaptação melhor foi a disponibilidade.

Por eliminar os fatores entende-se que eles serão mantidos em certos valores em todos os experimentos. Os demais fatores serão variados em dois níveis.

O número de nós ativos na composição *AWF+RS* mostrou-se significativo, no entanto, a variação deste fator não é importante para o objetivo final do estudo que se propõe aqui. Os modelos onde não ocorre provisão dinâmica são usados como base para comparação com os modelos onde o *DPS* atua. Escolher um número de nós suficiente para processar as requisições dos clientes e ter alguns nós rejuvenescidos já é o bastante.

A utilização alvo do *DPS*, apesar de apresentar forte tendência que se repetiu em todos os experimentos, mostrou-se bastante importante e portanto será variada.

O projeto experimental serviu não apenas para conhecer os fatores mais relevantes para os modelos. Os resultados obtidos foram os resultados esperados e a análise de projeto também serviu como um teste de corretude do simulador. As tendências observadas e apresentadas no decorrer deste capítulo eram esperadas.

Capítulo 5

Resultados da ação simultânea de sistemas de provisão dinâmica e de rejuvenescimento não coordenados

Neste capítulo são apresentados resultados de experimentos de simulação realizados com o objetivo de investigar quanto se perde em termos de qualidade de serviço quando há a ação simultânea não coordenada de sistemas de provisão dinâmica de recursos e rejuvenescimento de software. Para realizar este estudo o modelo genérico baseado em componentes apresentado no Capítulo 5 foi instanciado e analisado em diferentes cenários. Outros resultados de experimentos semelhantes podem também ser encontrados em (LOPES et al., 2005).

5.1 Experimentos que consideram o DPS perfeito

O objetivo desta seção é investigar como a qualidade de serviço de uma aplicação é afetada quando *PDPS* (o sistema de provisão dinâmica perfeito apresentado na Seção 4.3.1) e *RS* atuam simultaneamente.

5.1.1 Sobre a forma de avaliação dos resultados

No Capítulo 3 mencionou-se que existe um custo em termos de qualidade de serviço associado ao uso do *DPS*. Se antes a aplicação era super-provida de recursos, e agora pretende-se diminuir seu custo em termos de número de nós ativos, então esta redução de capacidade poderá ser refletida na qualidade de serviço da aplicação.

A disponibilidade máxima que uma aplicação sem faltas de software (e, portanto, fictícia) pode oferecer é 100% em qualquer instante de tempo. Sendo o *DPS* um sistema perfeito (*PDPS*), espera-se que esta disponibilidade máxima seja alcançada, desde que a aplicação esteja livre de faltas de software. Assim, a perda de qualidade de serviço que se observa ao utilizá-lo é nula em se tratando de disponibilidade ($l^{avail}_{PDPS} = 0$). Ainda considerando a aplicação livre de faltas, a disponibilidade em um sistema garantidamente super-provido de recursos seria também 100%. Quando falhas ocorrem, seus efeitos podem ser refletidos na disponibilidade da aplicação antes que os nós em falha sejam rejuvenescidos(l^{avail}_{RS}). Sendo assim, qualquer queda na disponibilidade que seja observada na composição *AWF+RS* ocorre por conta de tais falhas. Finalmente, quando a disponibilidade for avaliada em um ambiente onde atuam o *PDPS* e o *RS*, todas as perdas observadas ($l^{avail}_{PDPS+RS}$) serão por conta do efeito das falhas e dos custos dos rejuvenescimentos. As perdas acima mencionadas são calculadas de acordo com as equações a seguir. Nestas, termos A_C representam a disponibilidade da aplicação calculada na composição *C*.

$$\begin{aligned} l^{avail}_{PDPS} &= 0 \\ l^{avail}_{RS} &= 1 - A_{AWF+RS} \\ l^{avail}_{PDPS+RS} &= 1 - A_{AWF+DPS+RS} \end{aligned} \tag{5.1}$$

Considerando a disponibilidade do *e-service*, investiga-se a veracidade da seguinte inequação: $l^{avail}_{PDPS+RS} > l^{avail}_{RS}$. Esta inequação afirma que as perdas em termos de disponibilidade aumentam quando *PDPS* e *RS* atuam simultaneamente, se comparadas às perdas observadas para a mesma métrica quando o *RS* atua sobre uma aplicação que executa em uma infra-estrutura estaticamente super-provida de recursos. Em outras palavras, esta inequação indica que os efeitos das falhas e os custos dos rejuvenescimentos são maiores quando *PDPS* e *RS* atuam simultaneamente.

Em se tratando de tempo de resposta, um cálculo menos simples é realizado. Neste caso, não se pode comparar $l^t_{PDPS+RS}$ apenas com l^t_{RS}, uma vez que l^t_{PDPS} pode não ser nula. Considera-se como valor base mínimo para o tempo de resposta o tempo médio de serviço inerente das requisições presentes nas cargas de trabalho ($\overline{d}$). Este seria o menor tempo de resposta possível para esta aplicação ao realizar a carga de trabalho.

A perda em termos de tempo de resposta que se observa ao usar o *PDPS* é dada por l^t_{PDPS}. Esta é a piora do tempo de resposta se comparada ao tempo de resposta da aplicação quando garantidamente nenhuma fila é formada ($\overline{d}$). Sabe-se que o tempo de resposta médio da composição *AWoF+DPS* é dado pelo tempo médio de espera em fila (w) mais o tempo médio necessário para processar as requisições ($\overline{d}$). Neste cenário, o tempo de

espera em fila é afetado unicamente pela capacidade da aplicação (decidida pelo *PDPS*). Sendo $\overline{T}_C$ o tempo médio de resposta da aplicação no cenário C, então a perda l^t_{PDPS} pode ser dada pela seguinte equação:

$$l^t_{PDPS} = \overline{T}_{AWoF+PDPS} - \overline{d} \tag{5.2}$$

A perda em termos de tempo de resposta ao considerar falhas e rejuvenescimentos é dada por l^t_{RS}. Para computar tal perda é necessário conhecer o tempo de resposta da aplicação executando no mesmo ambiente considerado para a composição *AWF+RS*, porém sem a ocorrência de falhas de software. Este tempo é indicado por $\overline{T}_{AWoF}$. Quando se reduz este tempo do tempo médio de resposta computado para a composição *AWF+RS* obtém-se as perdas em termos de tempo de resposta devido ao efeito das falhas de software e aos custos dos rejuvenescimentos (já que a capacidade da aplicação se reduz enquanto os rejuvenescimentos ocorrem).

Se a perda l^t_{RS} fosse computada de forma semelhante ao que foi apresentado para a perda l^t_{PDPS}, o tempo de espera em fila poderia estar sendo computado duas vezes ao somar l^t_{PDPS} e l^t_{RS}. O tempo de resposta médio da composição *AWF+RS* é dado pelo tempo médio de espera em fila $(w_a + w)$ mais o tempo médio necessário para processar as requisições $(\overline{d})$ mais o tempo adicional de processamento devido à ocorrência das falhas (a). O tempo de espera em fila, no entanto, é afetado não apenas pela capacidade da aplicação (w), mas também pelo efeito das falhas antes dos rejuvenescimentos (w_a).

Para evitar uma soma redundante, resolveu-se utilizar o resultado da composição *AWoF*. Esta composição traz uma aplicação livre de faltas de software que executa sobre um ambiente estaticamente super-provido de recursos. Este tempo é indicado por $\overline{T}_{AWoF}$. Sendo assim, o tempo de espera em fila nesta composição é o tempo de espera em fila na composição *AWF+RS* caso nenhuma falha ocorresse. Ao diminuir o tempo médio de resposta medido para a composição *AWoF* do tempo medido para a composição *AWF+RS* obtém-se o tempo de espera em fila devido ao efeito das falhas e o tempo adicional de processamento devido a tais efeitos. A Equação 5.3 mostra tal cálculo.

$$l^t_{RS} = \overline{T}_{AWF+RS} - \overline{T}_{AWoF} \tag{5.3}$$

Nesta equação, não foi preciso subtrair o tempo médio inerente de processamento das requisições $(\overline{d})$, uma vez que ele já está presente no termo $\overline{T}_{AWoF}$.

As perdas l^t_{PDPS} e l^t_{RS} são então somadas e comparada com a perda do tempo de resposta medido para a composição *AWF+DPS+RS*, que é dada por $l^t_{PDPS+RS} = \overline{T}_{AWF+DPS+RS} - \overline{d}$.

No decorrer deste capítulo, as somas de l_{PDPS} e l_{RS}, seja para a disponibilidade, seja para o tempo de resposta, serão referenciadas pelo símbolo $\wedge$.

Esta soma indica quanto se perde por conta do uso do *PDPS* e por conta das falhas e rejuvenescimentos quando *PDPS* e *RS* atuam juntos em relação às perdas observadas quando tais sistemas atuam em seus ambientes isoladamente. Apesar de esta soma envolver sistemas diferentes, acredita-se que ela fornece uma boa base para comparação com $l_{PDPS+RS}$.

5.1.2 Intervalos de confiança

Em todos os experimentos que seguem (inclusive em capítulos subsequentes), o cálculo do número de amostras necessárias para se obter o intervalo de confiança desejado foi feito a partir dos resultados obtidos (em termos da média da disponibilidade, tempo de resposta e número de nós ativos) em uma amostra de n_{sample} experimentos, usando a Equação 5.4.

$$\begin{aligned} n &= t_y^2 \frac{S^2}{E_o^2}, \\ S^2 &= \frac{1}{n_{sample}-1} \sum_{i=1}^{n_{sample}} (x_i - \bar{x})^2 \end{aligned} \tag{5.4}$$

onde,

t_y = valor da variável aleatória T com probabilidade (nível de confiança) y;
T tem distribuição t de Student e, neste trabalho, representa a disponibilidade, o tempo de resposta da aplicação ou o número de nós ativos.
E_o = erro amostral máximo tolerado, o qual é dado em função da unidade de medida da média amostral.
n_{sample} = tamanho da amostra.
S^2 = variância da amostra.

Os intervalos de confiança gerados a partir dos resultados de experimentos apresentados neste capítulo são apresentados no Apêndice A.

5.1.3 Instâncias do modelo

Instâncias das composições *AWoF+PDPS*, *AWF+RS* e *AWF+PDPS+RS* foram criadas e avaliadas. A Tabela 5.1 mostra os fatores que variaram entre diferentes tratamentos.

Tabela 5.1: Parâmetros variáveis usados nos experimentos de simulações

Descrição	**Nível -1**	**Nível 1**
Carga de trabalho	Ver Figura 5.1	Ver Figura 5.2
Capacidade de processamento dos nós que executam a aplicação	15 rps	30 rps
Número de nós em falha	1	2

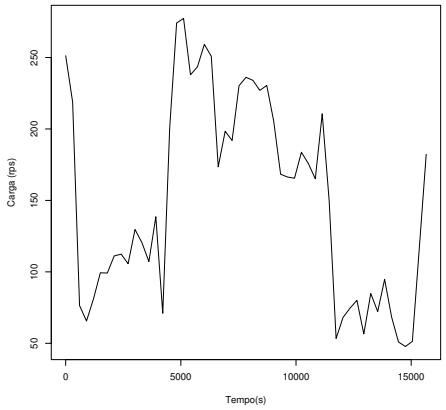

Figura 5.1: Carga de trabalho *down*

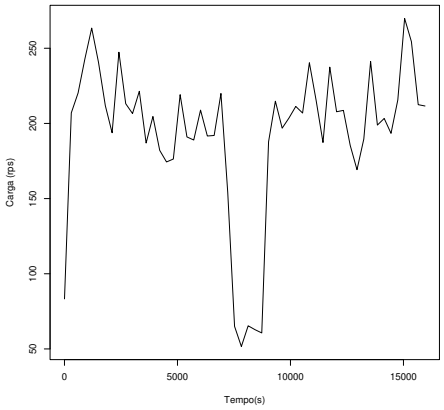

Figura 5.2: Carga de trabalho *up*

Os demais fatores, conforme decidido e discutido no capítulo anterior, permaneceram com o mesmo valor em todos os tratamentos:

- *Tempo médio de migração de um nó*: 1 min;
- *Tempo de reinício de um nó*: 1 min;
- *Tempo de adaptação do PDPS*: 5 min;
- *Tempo de adaptação do RS*: 5 min;
- *Número de intervalos consecutivos de adaptação do RS onde um nó fica com qualidade de serviço aquém do desejável antes de ser rejuvenescido*: 4;
- *Tamanho da fila de Backlog*: 1024;
- *Quantum de serviço dos servidores*: 100 ms;
- *Capacidade de processamento simultâneo de nós de menor capacidade*: 500;

- *Capacidade de processamento simultâneo de nós de maior capacidade*: 1000.

O valor do tempo de migração é um valor aproximado ao medido para a migração completa (transferência, instalação e inicialização) do JBoss em um Pentium 4 de 2,8 GHz com placa de rede de 100 Mbps. O tempo de reinício de um nó foi considerado o mesmo do tempo de migração. Percebe-se que o tempo necessário para parar o nó é bem pequeno (em média, 6 segundos).

O tempo de adaptação do *PDPS* não exerce muita influência, já que o *PDPS* sempre escolhe um número adequado de nós ativos, sendo 5 minutos um valor adequado na prática (RANJAN et al., 2002; MENASCÉ et al., 2003). O tempo de adaptação do *RS* foi considerado o mesmo por questões de simplicidade. Na implementação da simulação o *RS* tem suas informações atualizadas após cada intervalo de adaptação. Quando estes tempos forem diferentes deve-se atualizar as informações de monitoração do *RS* independente do tempo de adaptação, à medida que elas forem sendo coletadas. Esta sincronização é útil quando tais sistemas forem coordenados (assunto apresentado no Capítulo 7).

O número de intervalos consecutivos de adaptação indica o número de intervalos de medição consecutivos em que um nó apresenta desempenho aquém do esperado antes de ser rejuvenescido. Observou-se no estudo do projeto experimental que quanto maior é este valor, então pior é a qualidade de serviço da aplicação, pois os nós em falha permanecem por mais tempo em falha. No entanto, se este número for muito pequeno, o número de rejuvenescimentos não necessários pode ser grande, especialmente durante aumentos da carga. Tentando-se reduzir o número de rejuvenescimentos desnecessários, optou-se por um valor de quatro intervalos, que indica que após 20 minutos (número de intervalos vezes duração do intervalo) com qualidade de serviço aquém da esperada o nó é rejuvenescido.

O valor da capacidade da fila de *Backlog* e o valor da fatia de tempo de serviço do servidor são idênticos aos valores *default* do sistema operacional Linux. O tamanho da fila de *Backlog* no Linux pode ser visto em */proc/sys/net/ipv4/tcp_max_syn_backlog*. Este parâmetro define o número máximo de requisições de conexões que ainda não receberam um reconhecimento. O valor *default* é 1024 para sistemas com mais de 128 MB de memória.

O número estático de nós ativos em experimentos *AWF+RS* estão apresentados na Tabela 5.2.

Estes números de nós foram escolhidos de forma que nenhuma requisição é rejeitada por falta de capacidade de processamento.

Tabela 5.2: Número médio de nós ativos na composição *AWF+RS*

Capacidade dos nós	*Número de nós ativos (Nível 1)*	*Número de nós ativos (Nível -1)*
15 rps	27	33
30 rps	14	17

5.1.4 Resultados obtidos para experimentos que consideraram o *DPS* perfeito (*PDPS*)

Considerando as composições *AWoF+PDPS*, *AWF+RS* e *AWF+PDPS+RS*, esta combinação de parâmetros leva a 28 tratamentos. São 4 tratamentos da composição *AWoF+PDPS*, 8 tratamentos da composição *AWF+RS* e 16 tratamentos da composição *AWF+PDPS+RS*.

Um total de 1036 simulações foram realizadas, referentes a 37 amostras de cada tratamento de cada uma das seguintes composições: (i) *AWoF+PDPS*, (ii) *AWF+RS*, (iii) *AWF+PDPS+RS* e (iv) *AWF+PDPS*RS*. O *PDPS* acrescentou um nó ($e = 1$) além dos $\lceil \overline{\lambda}(t) \times \overline{d}(t) \rceil$.

Gráficos da disponibilidade e dos tempos de resposta da aplicação são apresentados nas Figuras 5.3 e 5.4.

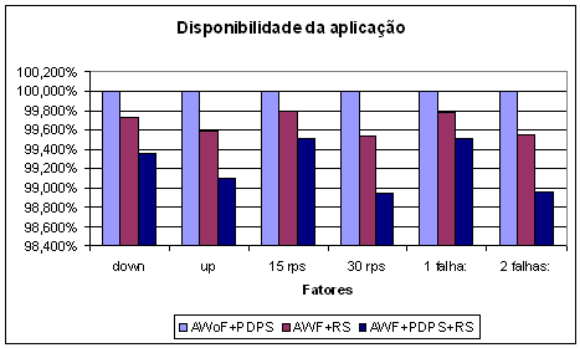

Figura 5.3: Disponibilidade da aplicação considerando diferentes composições

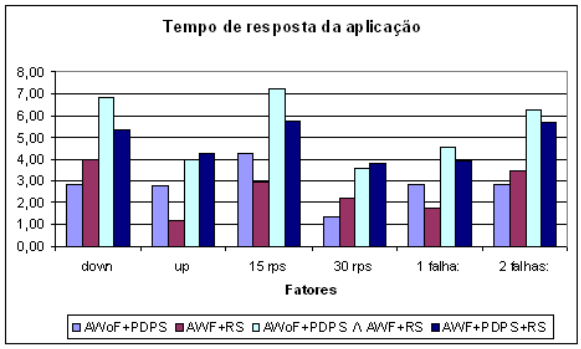

Figura 5.4: Tempo de resposta da aplicação considerando diferentes composições

Observa-se que em termos de disponibilidade, a união de *PDPS* e *RS* sempre resultou em uma perda mais acentuada que a perda observada na composição *AWF+RS*. Em, média, a disponibilidade da composição *AWF+PDPS2+RS* foi 0,441% menor que a disponibilidade da composição *AWF+RS*. Isto indica que o efeito das falhas e os rejuvenescimentos têm um custo muito maior em termos de disponibilidade quando ambos os sistemas atuam juntos.

Em termos de tempo de resposta observa-se que na maioria dos cenários os tempos de resposta da composição *AWF+PDPS+RS* são menores que a "soma" ($\wedge$) dos tempos de resposta das composições *AWoF+PDPS* e *AWF+RS*. Dois fatores contribuíram para este comportamento. Enquanto na composição *AWF+PDPS+RS* as falhas eram detectadas após 4 intervalos de adaptação do *RS*, pois logo seus efeitos eram percebidos, na composição *AWF+RS*, por conta da super-provisão, o efeito das falhas demorava mais para aparecer. Conseqüentemente, os rejuvenescimentos ocorriam mais tardiamente. Em alguns casos eles ocorriam após 12 intervalos de adaptação. Assim, os tempos de resposta, apesar de não violarem o tempo de resposta aceitável segundo o *RS*, ficaram bem maiores que o tempo de resposta da composição *AWoF* e a soma $l^{avail}_{PDPS} \wedge l^{avail}_{RS}$ resultou em uma perda maior que a perda l^{avail}_{PDPS}.

Sendo as perdas em termos de disponibilidade mais acentuadas, analisa-se a seguir tais perdas, considerando os diferentes fatores variáveis entre diferentes tratamentos.

Na Tabela 5.3 são apresentadas as perdas de disponibilidade para cada uma das composições simuladas destacando-se cada um dos fatores que variaram. A última coluna desta tabela indica quão pior é a disponibilidade da aplicação na composição *AWF+PDPS+RS* em relação à disponibilidade da aplicação na composição *AWF+RS*. Esta é também a informação contida no gráfico apresentado na Figura 5.5.

Tabela 5.3: Perdas médias de requisições e tempo de resposta

	down	*up*	*15 rps*	*30 rps*	*1 falha*	*2 falhas*
l^{avail}_{PDPS}	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%
l^{avail}_{RS}	0,263%	0,404%	0,202%	0,465%	0,222%	0,445%
$l^{avail}_{PDPS+RS}$	0,651%	0,898%	0,488%	1,061%	0,498%	1,051%
$l^{avail}_{PDPS+RS} - l^{avail}_{RS}$	0,388%	0,494%	0,287%	0,595%	0,276%	0,606%

Observa-se que existe uma queda substancial da disponibilidade da aplicação quando ambos os sistemas atuam sem coordenação em todos os cenários estudados. Estas perdas são maiores quando a carga *up* é aplicada, quando nós de maior capacidade são usados e quando mais falhas ocorrem.

Quando a carga *up* é aplicada, as falhas ocorrem em períodos de carga crescente e pico. Apesar do *PDPS* ser um sistema perfeito, o acréscimo de um nó tem um efeito menor quando a carga está mais alta e mais nós estão ativos. Assim, para esta carga, as falhas e rejuvenescimentos ocorreram em momentos em que a utilização da aplicação estava um pouco mais intensa.

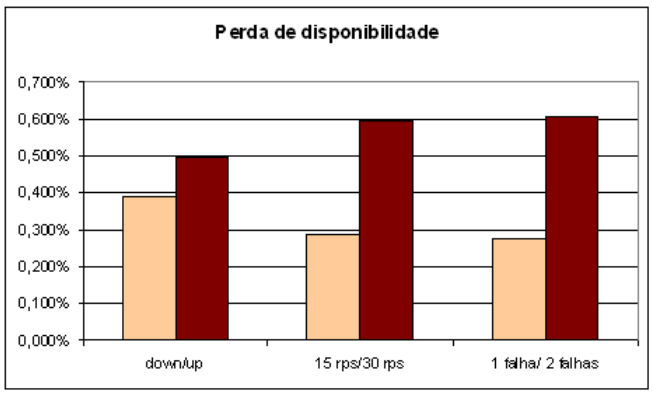

Figura 5.5: Perda em termos de disponibilidade($l^{avail}_{PDPS+RS} - l^{avail}_{RS}$)

Quando nós de maior capacidade são usados, o efeito das falhas é mais perceptível, pois quando um nó apresenta uma falha (seja de desempenho ou por parada) esta falha corresponde a uma porção maior da aplicação se comparada a uma falha em um nó de pequena capacidade.

Finalmente, quando mais nós falham, espera-se que os efeitos de tais falhas sejam mais graves para a aplicação.

As utilizações médias dos nós ativos foi medida durante os experimentos e são apresentadas nas Tabelas 5.4 e 5.5 destacando-se o fator capacidade dos nós, para as cargas *up* e *down* respectivamente.

Tabela 5.4: Utilização média dos nós ativos durante experimentos com carga *up*

Composição	*Capacidade dos nós*	*Utilização*
AWoF+PDPS	15 rps	82%
AWoF+PDPS	30 rps	74%
AWF+RS	15 rps	36%
AWF+RS	30 rps	37%
AWF+PDPS+RS	15 rps	82%
AWF+PDPS+RS	30 rps	75%

Excetuando-se a utilização da aplicação na composição *AWF+RS*, que foi super-provida de recursos, observa-se que as utilizações médias foram elevadas, especialmente quando nós de pequena capacidade foram usados. Seria possível, portanto, questionar se resultados semelhantes a estes seriam percebidos caso a provisão extra assegurada pelo *PDPS* fosse maior. Por esta razão, outros experimentos foram realizados, onde o *PDPS* passou a

Tabela 5.5: Utilização média dos nós ativos durante experimentos com carga *down*

Composição	*Capacidade dos nós*	*Utilização*
AWoF+PDPS	15 rps	80%
AWoF+PDPS	30 rps	71%
AWF+RS	15 rps	32%
AWF+RS	30 rps	32%
AWF+PDPS+RS	15 rps	80%
AWF+PDPS+RS	30 rps	72%

acrescentar dois nós extras ($e = 2$). Este sistema que oferece uma maior provisão extra será chamado de *PDPS2* por questões de organização.

Acrescentando-se dois nós extras as utilizações médias dos nós ativos diminuíram, segundo apresentado nas tabelas 5.6 e 5.7 para as cargas *up* e *down* respectivamente.

Tabela 5.6: Utilização média dos nós ativos durante experimentos com carga *up*

Composição	*Capacidade dos nós*	*Utilização*
AWoF+PDPS2	15 rps	76%
AWoF+PDPS2	30 rps	65%
AWF+RS	15 rps	36%
AWF+RS	30 rps	37%
AWF+PDPS2+RS	15 rps	76%
AWF+PDPS2+RS	30 rps	66%

O *PDPS2* foi usado em novos experimentos de simulação onde as mesmas instâncias anteriormente apresentadas foram criadas e avaliadas. Gráficos de disponibilidade e tempo de resposta são apresentados nas Figuras 5.6 e 5.7.

Resultados muito semelhantes aos observados quando o *PDPS* foi usado foram encontrados. Observa-se que a disponibilidade da aplicação na composição *AWF+PDPS2+RS* é, em média, 0,424% inferior à disponibilidade computada para a aplicação na composição *AWF+RS*. A mesma perda não se observa para os tempos de resposta. Na maioria dos cenários os tempos de resposta da composição *AWF+PDPS2+RS* são menores que os tempos de resposta que consideram a "soma" ($\wedge$) dos tempos de resposta *AWoF+PDPS2* e *AWF+RS*.

Tabela 5.7: Utilização média dos nós ativos durante experimentos com carga *down*

Composição	*Capacidade dos nós*	*Utilização*
AWoF+PDPS2	15 rps	73%
AWoF+PDPS2	30 rps	61%
AWF+RS	15 rps	32%
AWF+RS	30 rps	32%
AWF+PDPS2+RS	15 rps	73%
AWF+PDPS2+RS	30 rps	62%

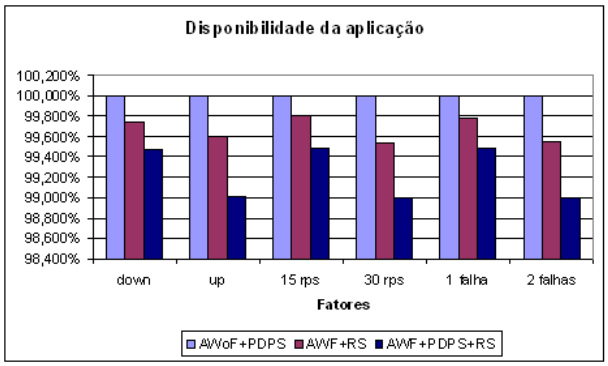

Figura 5.6: Disponibilidade da aplicação considerando diferentes composições

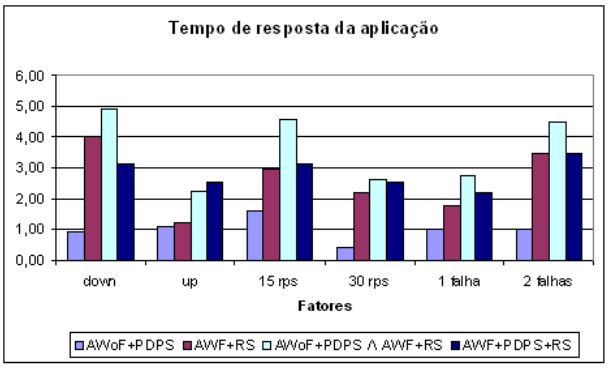

Figura 5.7: Tempo de resposta da aplicação considerando diferentes composições

Na Tabela 5.8 são apresentadas as perdas de disponibilidade para cada uma das composições simuladas destacando-se cada um dos fatores que variaram. A última coluna desta tabela indica quão pior é a disponibilidade da aplicação na composição *AWF+PDPS+RS* em relação à disponibilidade da aplicação na composição *AWF+RS*. Esta é também a informação contida no gráfico apresentado na Figura 5.8.

Observa-se que mesmo com o sistema perfeito de provisão dinâmica que adiciona dois nós extras, existe uma queda da disponibilidade da aplicação quando ambos os sistemas atuam sem coordenação se comparada à perda de disponibilidade $l_{PDPS+RS} - l_{RS}$.

Surpreendentemente, o acréscimo de um nó extra piorou a disponibilidade da aplicação na composição *AWF+PDPS2+RS* em muitos cenários, especialmente quando a carga *up* foi usada. Como a carga *up* inicia baixa e crescente, a(s) falha(s) foram inicialmente mascaradas pela provisão extra de recursos. Por conta disso, o rejuvenescimento acabou sendo adiado, ocorrendo exatamente no período de maior pico de carga. O rejuvenescimento também é bastante adiado no caso da composição *AWF+RS*, no entanto, neste caso, existe

Tabela 5.8: Perdas médias de requisições e tempo de resposta

	down	*up*	*15 rps*	*30 rps*	*1 falha*	*2 falhas*
l^{avail}_{PDPS2}	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%
l^{avail}_{RS}	0,263%	0,404%	0,202%	0,465%	0,222%	0,445%
$l^{avail}_{PDPS2+RS}$	0,535%	0,981%	0,514%	1,002%	0,508%	1,007%
$l^{avail}_{PDPS2+RS} - l^{avail}_{RS}$	0,273%	0,576%	0,312%	0,536%	0,287%	0,562%

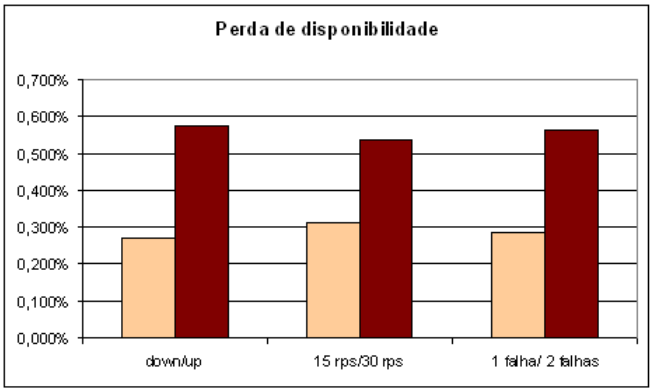

Figura 5.8: Perda em termos de disponibilidade ($l^{avail}_{PDPS+RS} - l^{avail}_{RS}$)

capacidade suficiente para tolerar este rejuvenescimento mesmo nos momentos de pico.

Para a carga *down* a adição de mais um nó foi salutar, diminuindo a diferença entre a disponibilidade da aplicação nas respectivas composições *AWF+RS* e *AWF+DPS2+RS* se comparada à disponibilidade medida quando apenas um nó foi acrescentado. De forma geral, espera-se que esta piora da disponibilidade decresça com o aumento da provisão extra de recursos. Este aumento, no entanto, não deve ser substancial, uma vez que os sistemas de provisão dinâmica existem exatamente para adequar a carga de trabalho à capacidade de trabalho evitando a super-provisão e barateando custos. Mais uma vez observou-se ainda que quanto maior a capacidade dos nós e o número de nós em falha, maior é a diminuição da disponibilidade, como também já foi discutido anteriormente.

As perdas em termos de tempos de resposta, quando ocorreram, não foram substanciais e por isso essa métrica não será avaliada aqui. As razões para este comportamento já foram discutidas anteriormente.

Com estes experimentos é possível concluir que, comumente, mesmo quando o sistema de provisão dinâmica é capaz de tomar ótimas decisões à respeito da capacidade da aplicação, pode-se verificar uma queda da qualidade de serviço da aplicação quando ambos, *PDPS* e *RS* atuam simultaneamente.

5.2 Experimentos que não consideram o *DPS* perfeito

Na seção anterior, analisou-se o que acontece quando o *PDPS* é usado. No entanto, é importante também investigar as conseqüências em termos de qualidade de serviço da aplicação quando o *DPS* não é perfeito. A seguir os resultados de tal estudo são apresentados.

5.2.1 Instâncias do modelo

As composições instanciadas foram as mesmas cujos projetos experimentais foram realizados no capítulo anterior, a saber: *AWoF+DPS*, *AWF+RS*, *AWF+DPS+RS*.

Dois grandes cenários experimentais diferenciados pela intensidade da carga de trabalho foram estudados. No Cenário I as cargas de trabalho são menos intensas que no Cenário II. Nas Figuras 5.9 e 5.10 são apresentadas as cargas de trabalho aplicadas em tratamentos do Cenário I. As cargas de trabalho aplicadas nos tratamentos do Cenário II são apresentadas nas Figuras 5.11 e 5.12.

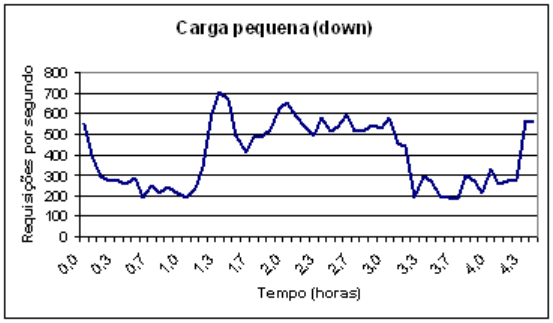

Figura 5.9: Carga de trabalho *down* do Cenário I

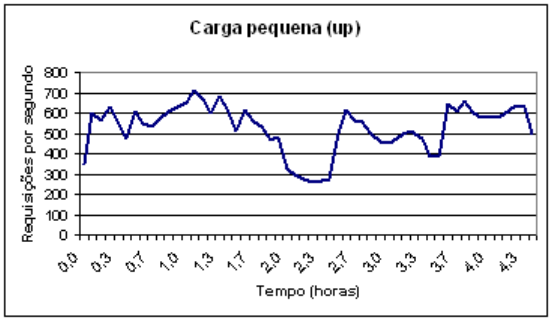

Figura 5.10: Carga de trabalho *up* do Cenário I

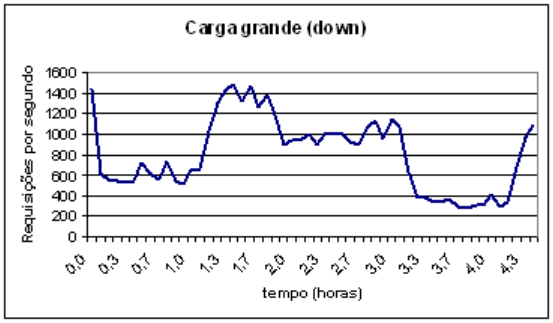

Figura 5.11: Carga de trabalho *down* do Cenário II

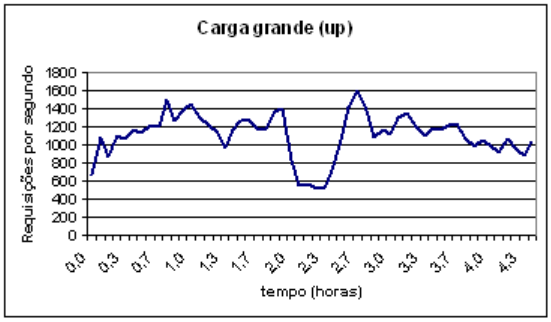

Figura 5.12: Carga de trabalho *up* do Cenário II

Excetuando-se estas cargas de trabalho, todos os outros parâmetros envolvidos na instanciação dos modelos foram os mesmos. A Tabela 5.9 mostra os fatores que variaram entre diferentes tratamentos.

Tabela 5.9: Parâmetros variáveis usados nos experimentos de simulações

Descrição	**Nível -1**	**Nível 1**
Capacidade de processamento dos nós que executam a aplicação	15 rps	45 rps
Utilização alvo perseguida pelo *DPS*	50%	70%
Número de nós em falha	1	2

Os demais fatores, conforme decidido e discutido no capítulo anterior, permaneceram com o mesmo valor em todos os tratamentos:

- *Tempo médio de migração de um nó*: 1 min;
- *Tempo de reinício de um nó*: 1 min;
- *Tempo de adaptação do DPS*: 2,5 min;
- *Tempo de adaptação do RS*: 2,5 min;
- *Número de intervalos consecutivos de adaptação do RS onde um nó fica com qualidade de serviço aquém do desejável antes de ser rejuvenescido*: 6;
- *Tamanho da fila de Backlog*: 1024;
- *Quantum de serviço dos servidores*: 100 ms;

Alguns destes parâmetros repetem os valores já mencionados quando da instanciação dos modelos que envolvem o *DPS* perfeito, não sendo necessário discuti-los novamente.

Para o tempo de adaptação do *DPS* foi considerado o valor do nível que resultou na melhor qualidade de serviço da aplicação e o tempo de adaptação do *RS* foi considerado o mesmo por questões de simplicidade.

O número de intervalos consecutivos de adaptação indica o número de intervalos de medição consecutivos em que um nó apresenta desempenho aquém do esperado antes de ser rejuvenescido. Tentando-se reduzir o número de rejuvenescimentos desnecessários, optou-se por um valor de seis intervalos, que indica que após 15 minutos (número de intervalos vezes duração do intervalo) com qualidade de serviço aquém da esperada o nó é rejuvenescido.

Outros parâmetros que tiveram seus valores modificados nos Cenários I e II são apresentados na Tabela 5.10.

Tabela 5.10: Outros parâmetros usados nos experimentos de simulações I e II

Descrição	**Valor no Cenário I**	**Valor no Cenário II**
Capacidade de processamento simultâneo de nós de menor capacidade	250	500
Capacidade de processamento simultâneo de nós de maior capacidade	500	1000
Intensidade das falhas	100 ms	300 ms

O número de nós em falha variou em diferentes tratamentos. Em alguns apenas um nó falhava de fato e em outros dois nós falhavam.

O número estático de nós ativos em experimentos *AWF+RS* do Cenário I estão apresentados na Tabela 5.11.

Tabela 5.11: Número estático de nós ativos em composições *AWF+RS* do Cenário I

	15 rps		*45 rps*	
	Carga down	*Carga up*	*Carga down*	*Carga up*
Nível -1	78	95	25	31
Nível 1	88	105	30	36

Estes números de nós foram escolhidos de forma que nenhuma requisição é rejeitada por falta de capacidade de processamento.

O número estático de nós ativos nos experimentos *AWF+RS* com carga mais intensas são apresentados na Tabela 5.12.

Mais uma vez, a super-provisão aplicada impede que requisições sejam rejeitadas no cenário *AWF+RS* por conta de falta de capacidade de processamento.

Considerando as composições *AWoF+DPS*, *AWF+RS* e *AWF+DPS+RS*, esta combinação de parâmetros leva a 32 tratamentos para os experimentos com carga menor e 32 tratamentos para os experimentos com carga maior. São 8 tratamentos da composição *AWoF+DPS*, 8 tratamentos da composição *AWF+RS* e 16 tratamentos da composição *AWF+DPS+RS*.

Tabela 5.12: Número estático de nós ativos em composições *AWF+RS* do Cenário II

	15 rps		*45 rps*	
	Carga down	*Carga up*	*Carga down*	*Carga up*
Nível -1	166	200	55	55
Nível 1	176	210	60	60

5.2.2 Sobre a forma de avaliação dos resultados

A análise realizada considera as seguintes métricas:

- Número médio de requisições rejeitadas em um intervalo de tempo. Dois intervalos de tempo serão considerados. O primeiro intervalo considera todo o tempo simulado $[0, t_{fimDaSimulacao}]$. Neste intervalo, $t_{fimDaSimulacao}$ é o instante final de tempo simulado, que é aproximadamente 4,5 horas. O segundo intervalo considera apenas o período de tempo em que ocorreram falhas e rejuvenescimentos $[0, t_{fimDoRejuvenescimento}]$. Em geral, após 15 minutos de simulação os nós em falha já foram rejuvenescidos. Em alguns casos (especialmente para a composição *AWF+RS*), no entanto, isto demora um pouco mais, pois as falhas demoram mais para apresentarem seus efeitos e serem percebidas no ambiente super-provido de recursos;

- Tempo de resposta médio da aplicação no intervalo completo de simulação (dado por $[0, t_{fimDaSimulacao}]$) e durante a ocorrência de falhas e de rejuvenescimentos (dado por $[0, t_{fimDoRejuvenescimento}]$). Esta métrica será estudada apenas quando perdas significativas forem observadas;

- Número médio de nós ativos no período $[0, t_{fimDaSimulacao}]$;

O objetivo da análise é identificar situações em que a união pura de um *DPS* e um *RS* sem coordenação aumenta as perdas de qualidade de serviço devido ao uso do *DPS* e devido às falhas e rejuvenescimentos (l_{DPS+RS}) se comparadas às perdas observadas quando apenas *DPS* (l_{DPS}) ou apenas *RS* (l_{RS}) atuam em seus cenários esperados. Para demonstrar esta hipótese uma análise comparativa dos dados coletados via simulação será realizada.

A comparação em termos de número de requisições rejeitadas será estabelecida da seguinte forma. O sistema *AWF+DPS+RS* é único e não foi estudado em trabalhos anteriores, não podendo ter sua qualidade de serviço comparada com a de um sistema semelhante. A este sistema está associada a perda devido à aplicação do *DPS* e também devido às falhas e rejuvenescimentos (l^{rej}_{DPS+RS}). Para contornar a dificuldade de se encontrar um

parâmetro para comparação, resolveu-se utilizar a soma das perdas independentes medidas nas composições *AWoF+DPS* e *AWF+RS*.

Considera-se que em um ambiente super-provido de recursos onde executa uma aplicação sem falhas (*AWoF*) nenhuma requisição é perdida. Assim, o número de requisições perdidas na composição *AWoF+DPS* indica exatamente a perda de qualidade de serviço devido ao uso do *DPS*, isto é, l^{rej}_{DPS}. Já a quantidade de requisições rejeitadas na composição *AWF+RS* indica as perdas de qualidade de serviço devido às falhas e aos rejuvenescimentos (l^{rej}_{RS}). Isto pode ser afirmado porque os cenários da composição *AWF+RS* foram configurados cuidadosamente para que nenhuma requisição seja perdida por conta de capacidade insuficiente dos recursos providos. Assim, todas as perdas que por ventura sejam observadas são por conta do efeito das falhas.

Somando-se estas perdas, têm-se perdas referentes ao uso do *DPS* (l^{rej}_{DPS}) e perdas referentes ao efeito das falhas e do custo dos rejuvenescimentos (l^{rej}_{RS}). Obviamente, os sistemas individuais não são iguais ao sistema conjunto, no entanto, acredita-se que eles podem oferecer um valor base de comparação para que se possa entender melhor o que se perde ao unir *DPS* e *RS* sem coordenação alguma.

A seguir são apresentadas as equações usadas para encontrar as perdas de interesse mencionadas. Nestas equações o a função rej_C indica o número médio de requisições rejeitadas pela aplicação na composição C.

$$\begin{aligned} l^{rej}_{DPS} &= rej_{AWoF+DPS} \\ l^{rej}_{RS} &= rej_{AWF+RS} \\ l^{rej}_{DPS+RS} &= rej_{AWF+DPS+RS} \\ l^{rej}_{DPS} \wedge l^{rej}_{RS} &= l^{rej}_{DPS} + l^{rej}_{RS} \end{aligned} \tag{5.5}$$

Uma análise em termos de disponibilidade não foi realizada porque somar perdas de disponibilidades não faz muito sentido. No entanto, a métrica número de requisições rejeitadas é uma ótima indicadora da disponibilidade da aplicação, uma vez que seu valor é usado no cálculo da disponibilidade. Conhecendo-se a quantidade total de requisições recebidas pela aplicação no intervalo de tempo estudado, pode-se calcular valores correspondentes de disponibilidade da aplicação no intervalo em questão. Esta transformação será realizada em alguns momentos ao longo das próximas seções.

Em termos de tempo de resposta, uma equação menos simples mas semelhante ao que já foi apresentado na Seção 5.1.1 foi usada. Por não existir um sistema semelhante que possa ser usado para a comparação, resolveu-se usar uma soma das perdas individuais l^t_{DPS}

e l^t_{RS}, tomando-se o cuidado de não repetir termos em tal soma. Tomou-se o tempo médio inerente de serviço das requisições ($\overline{d}$) como sendo o melhor tempo de resposta que se pode esperar da aplicação. As perdas serão então dadas em relação a este valor, segundos as seguintes equações.

$$\begin{aligned} l^t_{DPS} &= \overline{T}_{AWoF+DPS} - \overline{d} \\ l^t_{RS} &= \overline{T}_{AWF+RS} - \overline{T}_{AWoF} \\ l^t_{DPS} \wedge l_{RS} &= l^t_{DPS} + l^t_{RS} \\ l^t_{DPS+RS} &= \overline{T}_{AWF+DPS+RS} - \overline{d} \end{aligned} \tag{5.6}$$

As razões pelas quais tais equações são consideradas já foram discutidas na Seção 5.1.1.

O número médio de nós ativos na composição *AWF+DPS+RS* será comparado apenas com o número médio de nós ativos na composição *AWoF+DPS*, sendo este o valor base.

Além de levar em consideração as somas já mencionadas, é importante ressaltar que a avaliação realizada neste capítulo não leva em consideração os valores brutos das perdas, mas a diferença entre os valores $l_{DPS} \wedge l_{RS}$ e l_{DPS+RS}. Os resultados indicam que na maioria dos cenários tem-se $l_{DPS} \wedge l_{RS} < l_{DPS+RS}$ e serão analisados os fatores que mais afetam esta diferença.

As perdas do tempo de resposta e de custo (número de nós ativos) serão comparados através da seguinte equação:

$$Diff(\%) = \frac{\text{maior valor} - \text{menor valor}}{\text{maior valor}} \tag{5.7}$$

Ao dizer, por exemplo, que a perda de tempo de resposta de uma composição é x% pior que o de outra, indica-se que a maior perda de tempo de resposta é igual ao valor da menor perda de tempo de resposta mais x% deste.

As métricas que envolvem médias no intervalo de tempo $[0, t_{fimDoRejuvenescimento}]$ serão computadas para a composição *AWoF+DPS*, apesar de não haver rejuvenescimento. Tais métricas são necessárias para que se possa computar a soma $l_{DPS} \wedge l_{RS}$. O cálculo leva em consideração o tempo que o *RS* leva para rejuvenescer os nós em falha nas composições *AWF+DPS+RS* correspondentes.

5.2.3 Resultados obtidos para Cenário I: cargas *up* e *down* de menor intensidade

Vinte execuções de cada um dos 32 tratamentos foram realizadas. Os intervalos de 95% de confiança para disponibilidade, tempo de resposta, número de requisições rejeitadas e

número de nós ativos foram gerados e podem ser consultados no Apêndice A.

Os valores médios de disponibilidade e tempo de resposta para todo o intervalo de simulação estudado são apresentados nas Figuras 5.13 e 5.14. Nestes gráficos também são apresentados a disponibilidade e o tempo médio de resposta da "soma" $AWoF{+}DPS \wedge AWF{+}RS$ e da composição $AWoF$. Observa-se visualmente que tanto a disponibilidade quanto o tempo de resposta da aplicação pioraram em relação aos valores de $AWoF{+}DPS \wedge AWF{+}RS$ quando o DPS e o RS atuaram simultaneamente sem coordenação. A Os intervalos de 95% de confiança não mostram sobreposições entre valores computados para a grande maioria dos tratamentos correspondentes de $AWF{+}DPS{+}RS$, $AWF{+}RS$ e $AWoF{+}DPS$.

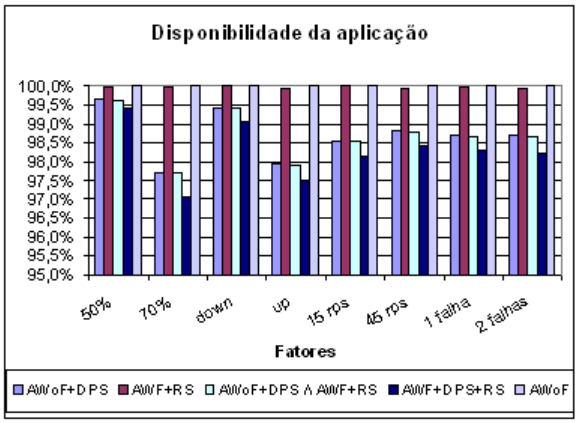

Figura 5.13: Disponibilidade computada no Cenário I

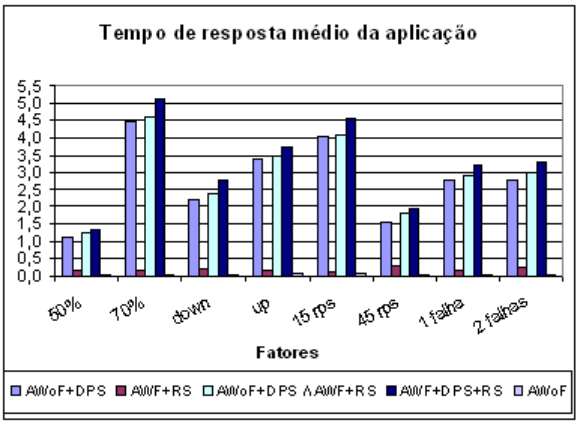

Figura 5.14: Tempo médio de resposta computado no Cenário I

Na Tabela 5.13 são apresentadas as perdas médias computadas para número de requisições rejeitadas e tempo de resposta.

Tabela 5.13: Valores usados nos testes de limite máximo da vazão

	Requisições rejeitadas	*Tempo de resposta*
l_{DPS}	107497	2,75 seg.
l_{RS}	3172	0,14 seg.
$l_{DPS} \wedge l_{RS}$	110669	2,89 seg.
l_{DPS+RS}	140926	3,21 seg.
$l_{DPS+RS} - (l_{DPS} \wedge l_{RS})$	**30257**	**0,32** seg.

Observa-se que em geral, $l_{DPS+RS} > (l_{DPS} \wedge l_{RS})$. Neste cenário específico, observa-se que a perda em termos de número de requisições perdidas (que pode ser refletida na disponibilidade da aplicação) é mais acentuada que a perda de tempo de resposta. Para o

número de requisições perdidas, observa-se que l_{DPS+RS} é 21,5% maior do que $l_{DPS} \wedge l_{RS}$. Já para o tempo de resposta, este percentual cai para 9,9%.

Observa-se que a quantidade média de nós ativos é praticamente a mesma nas composições *AWoF+DPS* e *AWF+DPS+RS*, sendo este último valor ligeiramente maior. Os intervalos de confiança para esta métrica se sobrepõem na maioria dos cenários, não sendo possível afirmar que o custo da aplicação aumenta quando as falhas e rejuvenescimentos são inseridos.

É possível ainda analisar os resultados considerando as perdas observadas para cada uma das métricas, comparando l_{DPS+RS} e $l_{DPS} \wedge l_{RS}$. Nos gráficos que seguem, os valores indicam o resultado da seguinte subtração: $l_{DPS+RS} - (l_{DPS} \wedge l_{RS})$.

Na Figura 5.15 são apresentadas as perdas médias em termos de quantidade de requisições rejeitadas quando *DPS* e *RS* atuam simultaneamente sem coordenação.

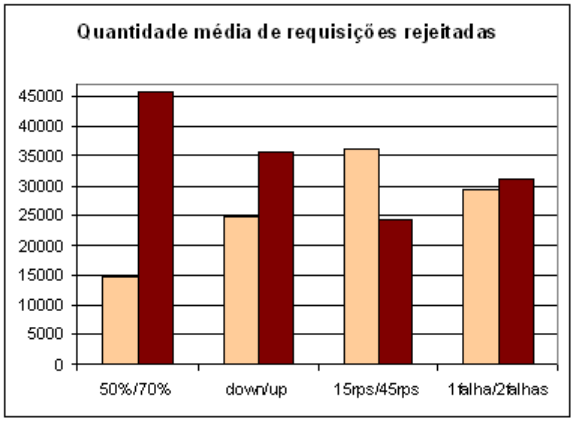

Figura 5.15: Perda total de disponibilidade quando não há coordenação

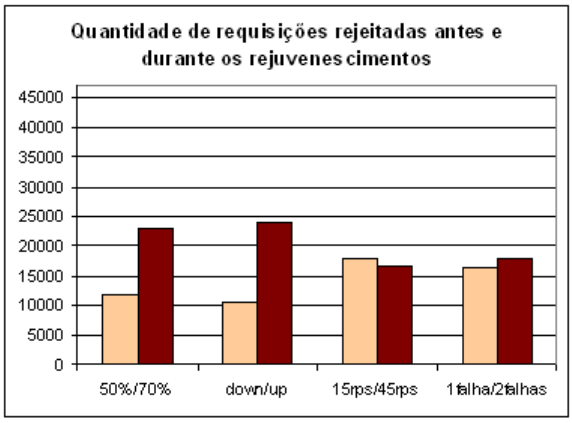

Figura 5.16: Perda de requisições na presença de falhas e rejuvenescimentos

Observa-se que a maior perda em termos de requisições rejeitadas se dá quando aumenta-se a utilização alvo dos nós da aplicação. Quanto maior é a utilização alvo, então:

1. Menor é a capacidade extra que se tem e, conseqüentemente, pior é a qualidade de serviço da aplicação, mesmo quando falhas não ocorrem;

2. Quando falhas ocorrem, os nós em falha recebem uma quantidade de requisições diretamente proporcional à utilização corrente de tais nós e quanto mais requisições os nós em falha recebem, pior é para a qualidade de serviço da aplicação.

Por estas razões, a utilização alvo do *DPS* exerce uma influência bastante intensa sobre a qualidade de serviço da aplicação. Esta característica, possivelmente, não está atrelada

ao algoritmo de provisão dinâmica estudado. De fato, conforme visto no Capítulo 2, os algoritmos de provisão dinâmica tomam decisões tentando manter alguma variável em torno de um alvo. Pode ser, por exemplo, manter o tempo de resposta médio em torno de um alvo. Quanto menos exigente for este alvo, menor será o número de nós necessários para atender as requisições e, conseqüentemente, maior será a utilização dos recursos.

Observa-se também que grande perda em termos de requisições rejeitadas ocorre quando a carga *up* é aplicada. A razão envolve especialmente o momento em que as falhas e rejuvenescimentos ocorreram. Para analisar este fato, é interessante estudar o estado da aplicação durante a ocorrência de falhas e rejuvenescimentos.

A Figura 5.16 mostra um gráfico da perda de requisições computada durante a ocorrência das falhas e dos rejuvenescimentos.

A maioria das rejeições de requisições ocorre antes e durante os rejuvenescimentos. Isto ocorre especialmente para a carga *up*. No início da simulação esta carga começa a subir acentuadamente. Neste momento existem nós em falha e estes nós recebem muito mais requisições que os nós subutilizados da composição *AWF+RS*. Como a carga está subindo, e as decisões são tomadas com base na carga do período anterior, os nós tendem a ficar um pouco mais sobrecarregados nas subidas de carga. Quando o *DPS* atuou, observou-se durante a subida da carga *up* uma utilização média de 60% e 80% respectivamente para cenários em que a utilização alvo é 50% e 70%.

Já quando a carga *down* foi aplicada as falhas e rejuvenescimentos ocorreram em momentos em que a carga estava diminuindo. Com a carga diminuindo, a utilização dos nós tende a ficar um pouco aquém da utilização alvo. Observaram-se utilizações de 46% e 65% respectivamente para cenários em que a utilização alvo foi 50% e 70%. Estas utilizações explicam a maior perda de requisições quando a carga *up* foi aplicada.

Espera-se que não haja sub-utilização intensa dos recursos quando há provisão dinâmica. Assim, é mais provável que os danos causados pelas falhas sejam mais intensos quando o *DPS* atua do que quando o *RS* atua isoladamente sobre uma aplicação executada sobre uma infra-estrutura super-provida estaticamente de recursos.

Voltando a analisar os resultados apresentados no gráfico da Figura 5.15, observa-se também que ao aumentar a capacidade dos nós a perda em termos de quantidade de requisições rejeitadas foi menor. A razão para este comportamento é explicada pelo fato de que as decisões do *DPS* foram melhores quando a capacidade dos nós foi maior, resultando em uma melhor qualidade de serviço da aplicação. Em média, a disponibilidade da aplicação na composição *AWoF+DPS* foi 0,283% melhor quando nós de maior capacidade foram usados. Em se tratando do *RS*, no entanto, observou-se uma preferência oposta, isto é,

quanto maior foi a capacidade dos nós maior foi a quantidade de requisições rejeitadas. Quando há ocorrência de falhas e de rejuvenescimentos, quanto maior a capacidade dos nós, maiores os danos causados pelas falhas e maior é a redução de capacidade causada pelos rejuvenescimentos, o que diminui a qualidade de serviço da aplicação. Observou-se na composição *AWF+RS* que quando nós de menor capacidade foram usados a disponibilidade da aplicação aumentou 0,076%. Sendo a influência da capacidade dos nós maior para o *DPS* do que para o *RS*, então quando tais sistemas atuaram simultaneamente sobre uma aplicação, manteve-se a preferência do *DPS* por nós de maior capacidade.

Finalmente, observou-se que quanto maior é a quantidade de falhas e rejuvenescimentos, maiores são as perdas em termos de quantidade de requisições rejeitadas. Este resultado não requer uma discussão extra, pois é bem aceito o fato de que quanto mais falhas ocorrerem, pior é a qualidade de serviço do *e-service*.

Na Figura 5.17 são apresentadas as perdas em termos de tempo de resposta de se ter um *e-service* gerenciado por um *DPS* e um *RS* sem coordenação entre suas atividades para todo o intervalo de simulação estudado.

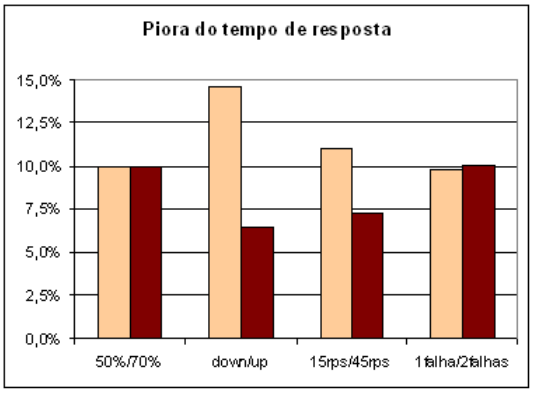

Figura 5.17: Perda total do tempo de resposta quando não há coordenação

Observa-se que a maior perda em termos de tempo de resposta se dá quando se passa da carga *up* para a carga *down*. Os valores médios de aumento do tempo de resposta quando *DPS* e *RS* atuam em relação a $l_{DPS} \wedge l_{RS}$ foram 0,40 seg. para carga *down* e 0,24 seg. para carga *up*.

Além do tempo máximo de resposta ter sido maior para composições onde a carga *up* foi aplicada, observa-se que neste caso as filas ficam cheias com mais freqüência, impossibilitando a piora ainda maior do tempo de resposta. Esta explicação é reforçada pelo fato de as maiores perdas de disponibilidade terem sido observadas exatamente quando a carga *up* é aplicada. Os limites impostos pelo tamanho da fila de *Backlog* e da capacidade

de processamento simultâneo dos servidores impediu que os tempos de resposta piorassem ainda mais quando a carga *up* foi aplicada.

Em relação à utilização média dos nós da aplicação, observa-se que perdas percentuais do tempo de resposta são praticamente as mesmas, independente da utilização alvo que o *DPS* persegue. De fato, a perda média é bem maior quando a maior utilização é perseguida (0,53 seg). Quando busca-se uma utilização menor, a perda em termos de tempo de resposta é de apenas 0,13 segundos, mas como o tempo médio de resposta é mais baixo, então os percentuais de perda se mantêm semelhantes.

Acompanhando a mesma tendência observada para a quantidade de requisições rejeitadas, observa-se que quando maior a capacidade dos nós, menores são as perdas em termos de tempo de resposta. Este comportamento é mais uma vez explicado pelo fato do *DPS* ter tomado melhores decisões quando nós de maior capacidade estiveram envolvidos.

Finalmente, observa-se que as perdas são bastante semelhantes, independente do número de nós em falha. As perdas percentuais do tempo de resposta foram ligeiramente maiores quando mais nós falharam. Uma vez que os nós em falha apresentam degradação de desempenho, o acréscimo de mais nós aumenta o tempo de resposta da aplicação.

É possível ainda realizar uma análise em termos de custo da aplicação considerando o número médio de nós ativos. O custo da aplicação na composição *AWF+DPS+RS* aumentou ligeiramente em alguns cenários e diminuiu em outros quando comparado ao custo da aplicação nos cenários *AWoF+DPS* correspondentes. Em média, o número de nós ativos na composição *AWF+DPS+RS* foi 0,10% maior que na composição *AWoF+DPS*.

Observa-se que em todos os cenários computados a união dos dois sistemas não foi satisfatório em termos de qualidade de serviço da aplicação, aumentando as perdas de requisições e de tempo de resposta se comparadas às perdas observadas quando tais sistemas atuam isoladamente.

5.2.4 Resultados obtidos para Cenário II: cargas *up* e *down* de maior intensidade

Vinte execuções de cada um dos 32 tratamentos foram realizadas e os intervalos de 95% de confiança foram gerados. Para fins de consulta, se necessário, os intervalos de confiança são apresentados no Apêndice A.

Nas Figuras 5.18 e 5.19 são apresentados os valores médios de disponibilidade e tempo de resposta para todo o intervalo de simulação estudado. Nestes gráficos são apresentados os valores computados para as composições *AWoF+DPS*, *AWF+RS*, *AWF+DPS+RS* e

AWoF, além dos valores computados para a "soma" *AWoF+DPS∧AWF+RS*. É possível observar, visualmente, que tanto a disponibilidade quanto o tempo de resposta da aplicação pioraram quando o *DPS* e o *RS* atuaram simultaneamente sem coordenação. Os intervalos de 95% de confiança não mostram sobreposições entre valores computados em tratamentos correspondentes de *AWF+DPS+RS*, *AWF+RS* e *AWoF+DPS*.

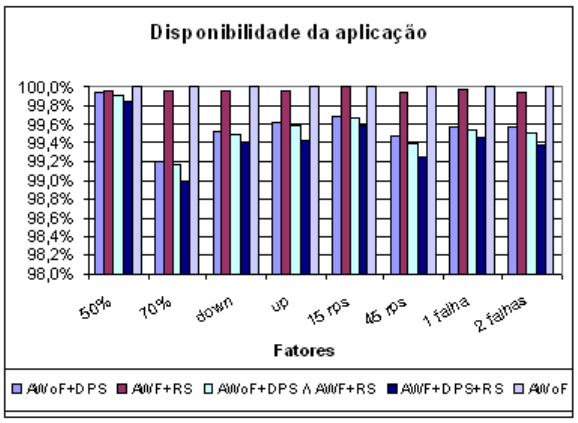

Figura 5.18: Disponibilidade computada no Cenário II

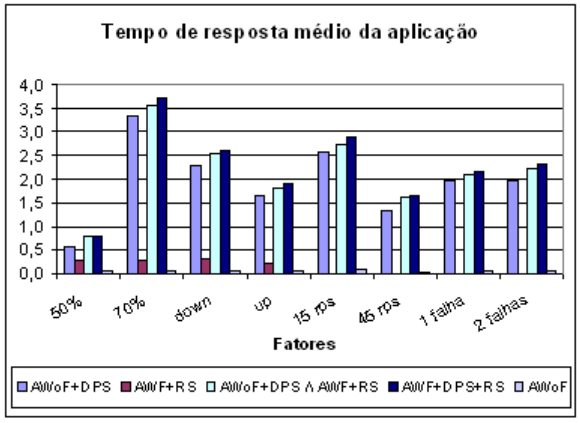

Figura 5.19: Tempo médio de resposta computado no Cenário II

Na Tabela 5.14 são apresentadas as perdas médias computadas para número de requisições rejeitadas e tempo de resposta.

Tabela 5.14: Valores usados nos testes de limite máximo da vazão

	Requisições rejeitadas	*Tempo de resposta*
l_{DPS}	66629	1,92 seg.
l_{RS}	6093	0,21 seg.
$l_{DPS} \wedge l_{RS}$	72722	2,13 seg.
l_{DPS+RS}	91659	2,22 seg.
$l_{DPS+RS} - (l_{DPS} \wedge l_{RS})$	**18938**	**0,09** seg.

Observa-se que em geral, $l_{DPS+RS} > (l_{DPS} \wedge l_{RS})$. Mais uma vez, observa-se que a perda em termos de número de requisições perdidas é mais acentuada que a perda de tempo de resposta. Para o número de requisições perdidas, observa-se que l_{DPS+RS} é 20,7% maior do que $l_{DPS} \wedge l_{RS}$. Já para o tempo de resposta, este percentual cai para 4,1%.

Na Figura 5.20 são apresentadas as perdas em termos de requisições rejeitadas de se ter um *e-service* gerenciado por um *DPS* e um *RS* sem coordenação entre suas atividades. Mais uma vez, todos os valores apresentados nos gráficos a seguir representam a perda em relação ao valor de $l_{DPS} \wedge l_{RS}$ da métrica em questão.

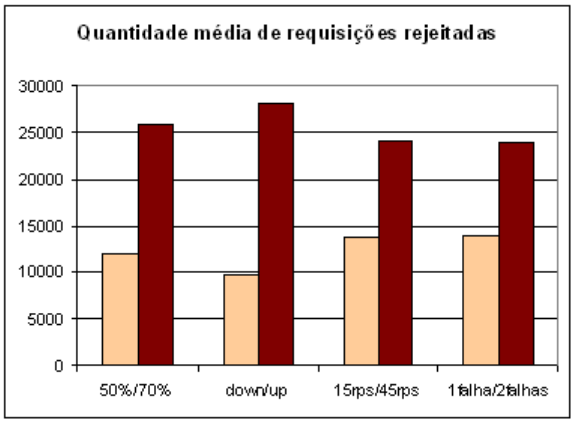

Figura 5.20: Perda em termos de requisições rejeitadas

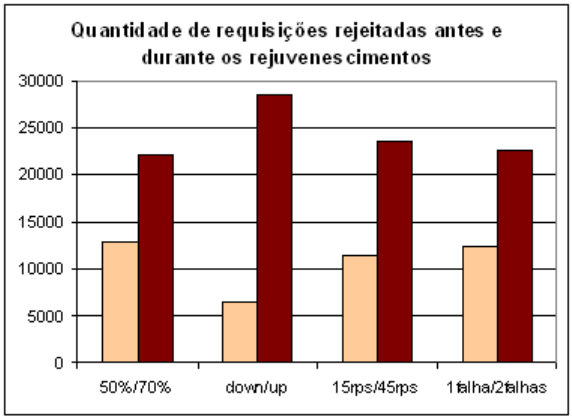

Figura 5.21: Perda de requisições na presença de falhas e rejuvenescimentos

Mais uma vez, observa-se grande perda em termos de requisições rejeitadas quando aumenta-se a utilização alvo da aplicação. Nota-se em todos os experimentos que a utilização alvo do *DPS* exerce uma influência bastante intensa sobre a qualidade de serviço da aplicação.

Há também grande perda em termos de requisições rejeitadas quando a capacidade dos nós aumenta. Observou-se que a combinação de *DPS* com nós de menor capacidade foi mais eficiente em termos de requisições rejeitadas e conseqüentemente em termos de disponibilidade. A disponibilidade média da aplicação nos cenários em que nós de menor capacidade foram aplicados na composição *AWoF+DPS* foi 0,207% melhor que a disponibilidade computada quando nós de maior capacidade foram usados. Além disso, observou-se também que quando falhas e rejuvenescimento estão envolvidos, então quanto maior a capacidade dos nós, maiores são os danos causados por eles. A disponibilidade da aplicação no cenário *AWF+RS* foi 0,066% melhor quando nós de menor capacidade foram usados. Assim, quando o *DPS* e o *RS* atuam em conjunto sobre uma aplicação, então mantém-se a tendência de que ao aumentar a capacidade dos nós a qualidade de serviço da aplicação diminui.

A carga *up* - semelhantemente ao que foi discutido na seção anterior - causou perdas mais intensas que a carga *down* por conta do momento em que as falhas e rejuvenescimentos ocorreram. Para analisar este fato, estudou-se o estado da aplicação durante a ocorrência de falhas e rejuvenescimento.

A Figura 5.21 mostra um gráfico da perda em durante a ocorrência das falhas e dos rejuvenescimentos. O gráfico mostra que a quantidade de requisições perdidas neste período é maior que a "soma" das perdas l_{DPS} e l_{RS} em termos de quantidades de requisições

rejeitadas no mesmo período. Evidencia-se novamente que os danos causados pelas falhas tendem a ser mais intensos quando o *DPS* atua do que quando o *RS* atua sobre uma infra-estrutura super-provida estaticamente de recursos.

Mais uma vez observou-se que a maioria das rejeições de requisições ocorre antes e durante os rejuvenescimentos, especialmente para a carga *up*. Esta carga começa a subir acentuadamente quando existem nós em falha a serem rejuvenescidos. Quando o *DPS* atuou, observou-se durante o crescimento da intensidade da carga *up* uma utilização média de 59% e 79% respectivamente para cenários em que a utilização alvo é 50% e 70%. Como a intensidade da carga está aumentando, os nós tendem a ficar mais sobrecarregados levando os nós em falha a receberem muito mais requisições que os nós subutilizados da composição *AWF+RS*. Já quando a carga *down* foi aplicada as falhas e rejuvenescimentos ocorreram em momentos em que a carga estava diminuindo e a utilização dos nós estava um pouco aquém da utilização alvo. Estas utilizações explicam a maior perda de requisições quando a carga *up* foi aplicada.

Finalmente, observa-se que quanto maior é a quantidade de falhas e rejuvenescimentos, maiores são as perdas em termos de requisições rejeitadas. Este resultado não requer uma discussão extra, pois é bem aceito o fato de que quanto mais falhas ocorrerem, pior é a qualidade de serviço do *e-service*.

As perdas em termos de tempo de resposta para este cenário foram bastante pequenas e por isso não serão analisadas.

Finalmente, observa-se que o custo da aplicação em termos de número de nós ativos aumentou, em média, 0,12% quando o *RS* e as falhas foram introduzidas. Apesar dos valores médios de custo terem sido sempre maiores na composição *AWoF+DPS*, os intervalos de confiança apresentaram sobreposições, especialmente quando a carga *up* esteve envolvida, não sendo possível afirmar para todos os cenários estudados que o custo aumenta. De fato, espera-se um custo maior por duas razões: (i) as falhas inseridas degradam o desempenho dos nós, aumentando sua utilização e influenciando o *DPS* a decidir por um número maior de nós e (ii) durante o rejuvenescimento de um nó a utilização dos demais nós que deverão receber a carga que seria enviada para o nó sendo rejuvenescido sobe, influenciando mais uma vez o *DPS* a acrescentar mais nós. Este aumento do custo indica que o *DPS* foi influenciado pelas falhas e pelos rejuvenescimentos.

5.2.5 Influência do número médio de nós ativos

Nas Figuras 5.22 e 5.23 são apresentados os números médios de nós ativos durante os experimentos que compuseram os cenários I e II respectivamente.

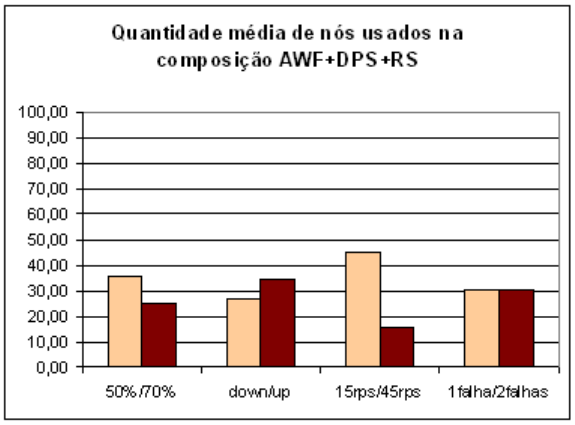

Figura 5.22: Número médio de nós ativos no Cenário I

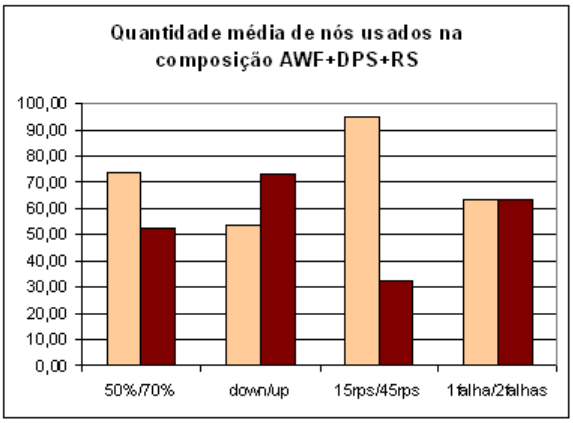

Figura 5.23: Número médio de nós ativos no Cenário II

Observou-se que, apesar da intensidade das falhas no Cenário I terem sido inferiores às utilizadas no Cenário II, as perdas em termos de qualidade de serviço foram mais intensas no cenário I. Na Tabela 5.15 a perda em termos de disponibilidade é apresentada considerada cada carga de trabalho aplicada separadamente.

Tabela 5.15: Perdas de disponibilidade por carga aplicada nos cenários I e II

		Perda de disponibilidade
Cenário I	*Carga down*	0,374%
	Carga up	0,410%
Cenário II	*Carga down*	0,073%
	Carga up	0,154%

Quando o intervalo $[0, t_{fimDoRejuvenescimento}]$ é considerado, estas perdas crescem, segundo apresentado na Tabela 5.15. Este crescimento se dá porque a maioria das perdas computadas no intervalo inteiro de simulação se deu no momento em que ocorreram falhas e rejuvenescimentos.

Estes resultados indicam que, sob condições semelhantes de carga de trabalho, as falhas e rejuvenescimentos influenciam mais negativamente uma aplicação onde menos nós estão ativos. Este resultado é bastante racional, uma vez que, quanto menor é a quantidade total de nós ativos, maior é a perda de capacidade que se tem quando um nó falha ou é desativado para ser rejuvenescido.

Uma análise semelhante em termos de tempo de resposta não faz sentido uma vez que existem limites superiores para o tempo de resposta impostos pelo tamanho da fila de

Tabela 5.16: Perdas de disponibilidade por carga aplicada nos cenários I e II durante ocorrência de falhas e rejuveenscimento

		Perda de disponibilidade
Cenário I	*Carga down*	3,015%
	Carga up	8,406%
Cenário II	*Carga down*	0,786%
	Carga up	3,456%

Backlog e pela capacidade de processamento simultâneo dos servidores. Assim, não há liberdade suficiente para que se possa analisar em que situações o tempo de resposta foi mais negativamente influenciado.

5.3 Conclusões parciais

Através de experimentos de simulação foi possível identificar que independentemente da qualidade do sistema de provisão dinâmica, é possível identificar diversos cenários em que a união simples de um *DPS* e um *RS* não é satisfatória em termos de qualidade de serviço da aplicação. Observa-se que, independentemente da qualidade das decisões tomadas pelo *DPS*, as perdas devido ao uso do *DPS* e as perdas devido às falhas de software e rejuvenescimentos são maiores quando *DPS* e *RS* atuam juntos do que a soma de tais perdas quando os sistemas atuam isoladamente.

Ao analisar os resultados experimentais que envolveu o *DPS* percebe-se especialmente três fatores que contribuem mais fortemente para a ineficiência da união simples do *DPS* e *RS*:

- *Utilização alvo do DPS*. Quanto maior a utilização alvo do *DPS*, pior a disponibilidade da aplicação nos momentos de falha e de rejuvenescimento;
- *Carga de trabalho no momento da falha e rejuvenescimento*. Quando as falhas e rejuvenescimentos ocorrem em momentos em que a carga está crescendo, observa-se uma maior utilização dos nós, piorando a qualidade de serviço da aplicação;
- *Número de nós ativos no momento da falha e rejuvenescimento*. Quando o número de nós ativos no momento em que há ocorrência de falhas e rejuvenescimentos é menor, então a qualidade de serviço é mais negativamente influenciada.

Além disso, nos cenários em que o *DPS* foi configurado com utilização alvo maior foram observados rejuvenescimentos devido à sobrecarga dos nós, especialmente em períodos de subidas bruscas da carga de trabalho. Como será discutido no capítulo que segue, esta possibilidade de confundir sobrecarga com degradação de desempenho deve ser levada em consideração quando *DPS* e *RS* precisam atuar simultaneamente.

Os fatores capacidade dos nós e número de nós em falha também exerceram influência sobre o comportamento emergente do sistema. Observou-se, como esperado, que quanto maior a quantidade de nós que falham, maior é a perda (l^{avail}_{DPS+RS} ou l^{rej}_{DPS+RS}), pois (i) maior é a quantidade de requisições recebida por nós em falha e (ii) maior é a redução de capacidade devido aos rejuvenescimentos. Já o fator capacidade dos nós nem sempre mantém uma tendência quando o DPS não perfeito foi usado.

Quando apenas o *RS* atua sobre uma aplicação super-provida de recursos e fixando-se qualitativa e quantitativamente as falhas, observa-se que quanto maior a capacidade dos nós, pior é a qualidade de serviço da aplicação. Este comportamento é também observado quando *PDPS* e *RS* atuam simultaneamente. No entanto, nem sempre é observado quando *DPS* e *RS* atuam juntos. Observou-se que a preferência do *DPS* em relação à capacidade dos nós é mantida quando ambos, *DPS* e *RS*, atuam sem coordenação. Isto demonstra que nem sempre as mesmas propriedades observadas quando *DPS* e *RS* atuam isoladamente são também observadas quando eles atuam juntos.

O simulador usado para obter os resultados apresentados neste capítulo foi verificado e validado através de testes de limite e experimentos com um modelo de medição correspondente. A descrição completa destas atividades pode ser encontrada no Apêndice D. Os experimentos de medição, apesar de apresentarem valores pontuais diferentes dos computados pelo modelo de simulação, apresentaram tendências que seguem os mesmos padrões dos encontrados no modelo de simulação. Como no decorrer desta tese os valores em si não foram usados, mas sim as tendências de diferenças entre tais valores, então considera-se que o modelo de simulação usado é adequado para este fim.

No capítulo que segue serão discutidas e analisadas técnicas que podem ser úteis quando *DPS* e *RS* precisam atuar sobre uma mesma aplicação.

Capítulo 6

Coordenação de atividades de provisão dinâmica e rejuvenescimento

Em experimentos realizados com uma aplicação demo de comércio eletrônico pôde-se identificar indícios de que colaborações entre as atividades de provisão dinâmica e de rejuvenescimento são possíveis e interessantes (LOPES; CIRNE; BRASILEIRO, 2004). Neste capítulo, colaborações entre estes sistemas são apresentadas com detalhes e estudadas através de simulações.

6.1 Técnicas de coordenação

As técnicas de coordenação propostas e estudadas são:

1. *Diminuição coordenada de capacidade.* Esta técnica é realizada pelo sistema de provisão dinâmica, com o auxílio do *RS* quando for detectada a necessidade de diminuição da capacidade da aplicação, isto é, quando nós precisarem ser removidos da aplicação. O *DPS* deve selecionar para remoção os nós mais propensos a falhar (ou já em falha), em vez de escolher aleatoriamente quais nós serão liberados. Para que esta técnica seja realizada, o *DPS* deverá perguntar ao *RS* quais são os nós mais propensos a falha no momento. O *RS*, por sua vez, deverá manter atualizada uma lista com esta informação;

2. *Rejuvenescimento coordenado.* Esta técnica é realizada pelo sistema de rejuvenescimento com o auxílio do *DPS*. Quando um nó precisa ser rejuvenescido, o *RS* solicita ao *DPS* que remova o nó em falha (aquele que deve ser rejuvenescido) e adicione

um novo nó completamente operacional. Esta técnica só pode ser aplicada quando existirem nós livres;

3. *Quorum mínimo para detecção de falhas.* Esta técnica visa eliminar a quantidade de rejuvenescimentos devido a degradação de desempenho causada por sobrecarga nos nós e não por falhas de fato. O *RS* só considera nós em falha se a quantidade de nós que parecem estar em falha no mesmo instante for menor que o quorum mínimo. Esta técnica será melhor discutida posteriormente.

Para que os sistemas de rejuvenescimento e provisão dinâmica atuem sobre a mesma aplicação, ainda que sem coordenação, é preciso que o *DPS* tenha a capacidade de avisar aos interessados (no caso, o *RS*) sobre ativação/desativação de nós. É possível imaginar um ambiente em que isso já existe para fins de contabilidade. Mas neste caso, a capacidade de avisar aos interessados sobre ativação e desativação de nós é uma condição *sine qua non* para a atuação em conjunto dos dois sistemas. Esta notificação é necessária porque o *RS* precisa saber quem são os nós ativos no momento, de forma que ele possa monitorá-los apropriadamente. Quando a provisão dinâmica não era utilizada, o *RS* não precisava de um método específico de gerência de réplicas, pois o conjunto de nós ativos era estático. Assim, o *DPS* precisa ser modificado para notificar interessados sobre desativação e ativação de nós; e o *RS* precisa ser preparado para gerenciar um conjunto dinâmico de nós.

Além da modificação supra-citada, os sistemas de provisão dinâmica e rejuvenescimento tradicionais precisam de outras modificações, pois não estão preparados para ter suas atividades coordenadas. Ao implementar a coordenação destes sistemas para a realização dos experimentos, foi possível identificar extensões necessárias em cada um destes sistemas para que eles trabalhem de forma coordenada.

As seguintes funcionalidades foram acrescentadas no sistema de provisão dinâmica:

1. Capacidade de receber requisições de desativação condicionada do nó i. Esta desativação é dita condicionada porque o nó i só será desativado se houver um outro nó livre para ser imediatamente ativado. Quando nenhum nó estiver livre, o *DPS* deve avisar à entidade solicitante que não poderá realizar a operação. Neste caso, a entidade solicitante, que é o *RS*, deverá realizar o rejuvenescimento tradicional do processo em falha;

2. Quando o *DPS* decide pela desativação de nós, os nós a serem desativados não serão aleatoriamente escolhidos. O *DPS* deve ter a capacidade de solicitar a uma outra entidade (que no caso é o *RS*) a indicação de quais nós devem ser desativados.

No sistema de rejuvenescimento as seguintes modificações foram realizadas para que suas atividades pudessem ser coordenadas com as atividades do *DPS*:

1. O *RS* deve ter a capacidade de ter uma lista ordenada dos nós segundo seu estado. Assim, o *RS* deve ser capaz de indicar a uma outra entidade sobre quais são os n nós em pior estado;

2. O *RS* deve ter a capacidade de, tendo decidido pelo rejuvenescimento do nó i, solicitar ao *DPS* que realize a desativação condicionada de tal nó. Caso o *DPS* não possa realizar tal desativação, então o *RS* deve ser capaz de proceder com o rejuvenescimento tradicional do nó;

3. O *RS* deve ter a capacidade de adiar rejuvenescimentos quando o quorum mínimo for ultrapassado.

Estas modificações envolvem questões práticas de segurança que devem ser resolvidas em uma implementação real.

Uma técnica muito semelhante à de rejuvenescimento coordenado foi recentemente proposta em (SILVA et al., 2007), mostrando-se eficiente para melhorar a qualidade de serviço da aplicação. Esta técnica requer que os servidores sejam executados em um ambiente de virtualização. Quando um nó precisa ser reiniciado um servidor virtual de *backup* é ativado na mesma máquina do nó que será rejuvenescido. Este servidor virtual de *backup* processa as requisições da aplicação enquanto o servidor de fato é rejuvenescido. No contexto desta tese, a virtualização não é necessária porque o próprio sistema de provisão dinâmica já atua no sentido de ativar um novo nó enquanto o nó em falha é desativado.

É interessante dissertar um pouco mais à respeito do quorum mínimo. Percebe-se que, se o *DPS* não é um sistema perfeito, então é possível que quando a carga aumente abruptamente, o *DPS* não acompanhe com o aumento de capacidade na mesma velocidade, gerando momentos de sobrecarga para os nós ativos. Dependendo da métrica usada para decidir se um nó está em falha ou prestes a falhar, esta sobrecarga pode ser facilmente confundida com degradação de desempenho, conforme mostrado em (LOPES; CIRNE; BRASILEIRO, 2004). Esta confusão faz o *RS* reiniciar os nós sobrecarregados, causando geralmente a piora da qualidade de serviço da aplicação, pois mesmo usando a técnica de redução coordenada de capacidade, a aplicação ficará com um nó a menos durante o período de migração do nó adicionado no lugar dos nós em "falha". Quando o ambiente foi super-provido de recursos estes falso positivos não foram verificados, nem em experimentos de medição, nem

nos experimentos de simulação. Assim, não se tem conhecimento de sistemas de rejuvenescimento que posterguem o rejuvenescimento na tentativa de diferenciar entre nós em falha e nós sobrecarregados.

Em (LOPES; CIRNE; BRASILEIRO, 2004) foi testado através de experimentos de medições um sistema coordenado em que o *RS* decide pelo rejuvenescimento de acordo com um quorum mínimo. Naquele trabalho, o quorum mínimo é a maioria dos nós ativos. Quando mais da metade dos nós ativos estão aparentemente em falha, então o *RS* não rejuvenesce tais nós. Estes experimentos de medição envolveram apenas cinco nós e a carga era de pequena intensidade e pequena duração. Ter um quorum mínimo de mais da metade funcionou porque resultou em um valor pequeno, que era no máximo 3 nós. Para os experimentos apresentados neste capítulo, no entanto, este mesmo quorum não foi adequado. O número total de nós ativos é muito grande para considerar este quorum. O que acontece na prática é que apesar das ações do *DPS*, um ou mais nós, muitas vezes, ficam sobrecarregados durante os aumentos da carga de trabalho, com desempenho aquém do aceitável, só se recuperando completamente depois de alguns intervalos de adaptação. Antes da recuperação, no entanto, eram reiniciados pelo *RS*. Porém, nem todos os nós ativos ficam com o desempenho degradado. Em geral, observou-se que apenas os nós que já estavam ativos desde o início do aumento da carga de trabalho ficam com o desempenho mais degradado. Por esta razão o quorum mínimo de metade não foi eficiente neste novo ambiente, pois era muito grande. Resolveu-se, então, optar por um número fixo para este quorum mínimo, que possa ser configurado pelos administradores da aplicação. De fato, o quorum mínimo é bastante dependente dos tipos de falha que ocorrem mais comumente e a freqüência com que estas falhas ocorrem.

Os sistemas *DPS* e *RS* apresentados no Capítulo 5 tiveram suas ações coordenadas através destas técnicas gerando a composição *AWF+DPS⋆RS*. A seguir são apresentados os resultados obtidos através de simulação.

6.2 Sobre a forma de avaliação dos resultados

O objetivo da análise é identificar se as técnicas de coordenação são eficientes no sentido de minimizar as perdas de qualidade de serviço geradas ao juntar *DPS* e *RS* sem coordenação. Pretende-se avaliar tais técnicas comparando a qualidade de serviço da aplicação medida nas composições *AWF+DPS+RS* e *AWF+DPS*RS*. A disponibilidade e o tempo de resposta são as métricas de interesse para esta análise.

Os ganhos em termos de disponibilidade e tempo de resposta são calculados respecti-

vamente de acordo com as Equações 6.1 e 6.2.

$$g^{avail}_{DPS*RS} = A_{AWF+DPS*RS} - A_{AWF+DPS+RS} \tag{6.1}$$

$$g^{t}_{DPS*RS} = \overline{T}_{AWF+DPS+RS} - \overline{T}_{AWF+DPS*RS} \tag{6.2}$$

Os ganhos de resposta e de custo (número de nós ativos), assim como no Capítulo 5, serão transformados em percentuais de ganho através da seguinte equação:

$$Diff(\%) = \frac{\text{maior valor} - \text{menor valor}}{\text{maior valor}} \tag{6.3}$$

Ao dizer, por exemplo, que o tempo de resposta de uma composição é x% melhor que o de outra, indica-se que o melhor tempo de resposta é igual ao valor do pior tempo de resposta menos x% deste.

Os cenários gerais estudados são os mesmos cenários que consideravam o *DPS* perfeito (*PDPS*) apresentados na seção 5.1.3 e os Cenários I e II apresentados na Seção 5.2.1 do Capítulo 5.

6.3 Resultados obtidos para cenários em que o DPS perfeito foi usado

Os mesmos cenários simulados para as composições *AWoF+PDPS*, *AWF+RS* e *AWF+PDPS+RS* foram também simulados para a composição *AWF+PDPS*RS*. Os intervalos de 90% de confiança da disponibilidade e do tempo de resposta para a composição *AWF+PDPS*RS* foram gerados e são apresentados no Apêndice B.

Gráficos de disponibilidade e tempo de resposta da aplicação computados nas composições *AWF+DPS+RS* e *AWF+DPS*RS* são apresentados nas Figuras 6.1 e 6.2.

Os ganhos médios em termos de disponibilidade e tempo de resposta ao usar as técnicas de coordenação foram respectivamente de 0,188% e 19,01%.

Na Tabela 6.1 são apresentados os ganhos de disponibilidade e tempo de resposta para cada uma das composições simuladas destacando-se os fatores que variaram. Esta é também a informação contida nos gráficos apresentados nas Figuras 6.3 e 6.4.

Quando a carga *up* é aplicada, a técnica de coordenação usada é sempre o rejuvenescimento coordenado, uma vez que nenhum nó é desativado antes do rejuvenescimento. Esta

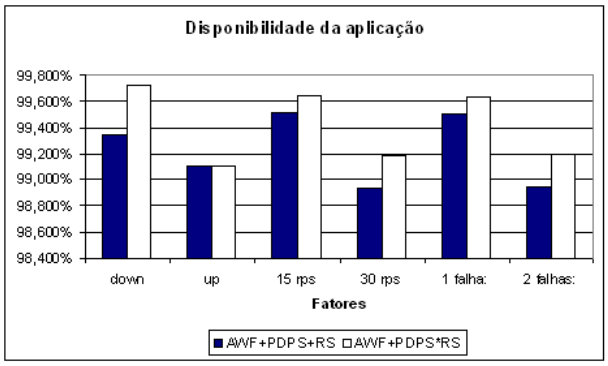

Figura 6.1: Disponibilidade da aplicação considerando diferentes composições

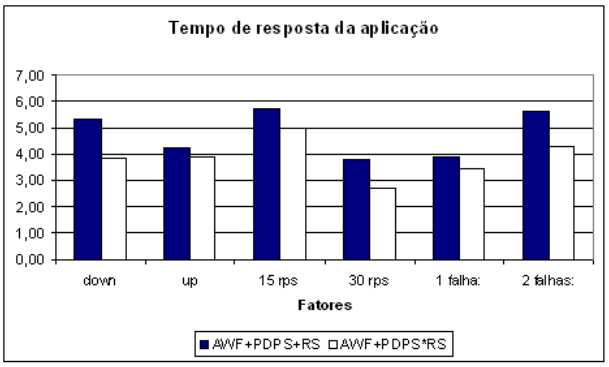

Figura 6.2: Tempo de resposta da aplicação considerando diferentes composições

Tabela 6.1: Ganhos médios de disponibilidade

	down	*up*	*15 rps*	*30 rps*	*1 falha*	*2 falhas*
$g^{avail}_{PDPS*RS}$	0,376%	0,000%	0,127%	0,249%	0,128%	0,249%
$g^{t}_{PDPS*RS}$	27,57%	8,74%	13,30%	28,18%	12,17%	24,10%

técnica não resulta em ganhos de disponibilidade - pelo menos quando o *PDPS* é usado. O ganho desta técnica é apenas no tempo de resposta da aplicação.

Ao aplicar o rejuvenescimento coordenado, observa-se que o momento em que o nó em falha é retirado da aplicação é o mesmo, independentemente dos sistemas terem suas atividades coordenadas. Assim, todos os efeitos de se ter um ou mais nós em falha são sentidos da mesma forma. No entanto, uma vez que o nó será rejuvenescido, o sistema que tem as suas atividades coordenadas apresenta vantagem pois o tempo de rejuvenescimento

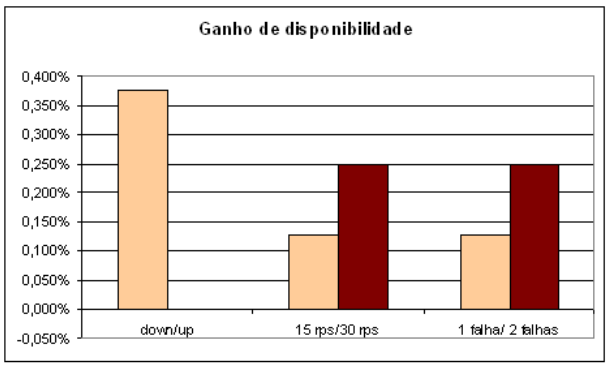

Figura 6.3: Ganho em termos de disponibilidade

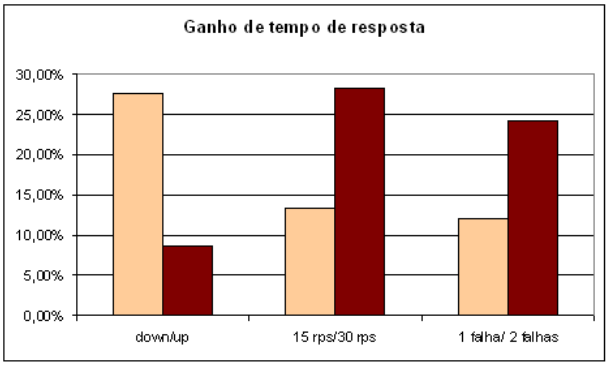

Figura 6.4: Ganho em termos de tempo de resposta

é menor. Não é preciso esperar que o nó processe todas as requisições pendentes, seja rejuvenescido e retorne às suas atividades. O nó em falha é retirado e outro nó saudável é ativado após o tempo de migração. Assim, o tempo de rejuvenescimento é reduzido, passando a ser igual ao tempo de migrar um nó do estado inativo para o estado ativo. Espera-se que este tempo de migração seja menor que o tempo de rejuvenescimento, uma vez que o tempo total de rejuvenescimento é, pelo menos, a soma do tempo de desativação do nó com o tempo de migração do mesmo.

Já quando a carga *down* é aplicada a técnica de coordenação usada foi sempre a desativação coordenada de nós. Esta desativação coordenada mostra-se bastante eficiente. Ambos, disponibilidade e tempo de resposta, melhoraram substancialmente quando os sistemas tiveram suas atividades coordenadas.

Como não há sobreposição dos intervalos, pode-se afirmar que para as amostras observadas, em termos de tempo de resposta, a composição *AWF+PDPS+RS* é sempre pior que a composição *AWF+DPS*RS*, mesmo ao variar a capacidade dos nós e o número de nós em falha.

Observa-se que as técnicas de coordenação - inclusive o rejuvenescimento coordenado que não foi eficiente em termos de disponibilidade - melhoraram o tempo de resposta da aplicação gerenciada.

Experimentos com o *PDPS2* foram também realizados. Os intervalos de 90% de confiança da disponibilidade e do tempo médio de resposta foram gerados e podem ser consultados no Apêndice B. Gráficos de disponibilidade e tempo de resposta da aplicação computados nas composições *AWF+DPS+RS* e *AWF+DPS*RS* são apresentados nas Figuras 6.5 e 6.6.

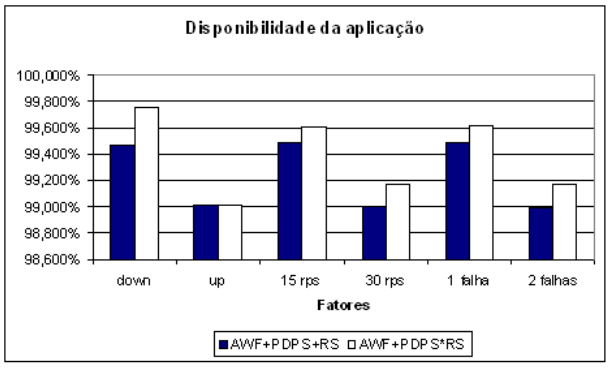

Figura 6.5: Disponibilidade da aplicação considerando diferentes composições

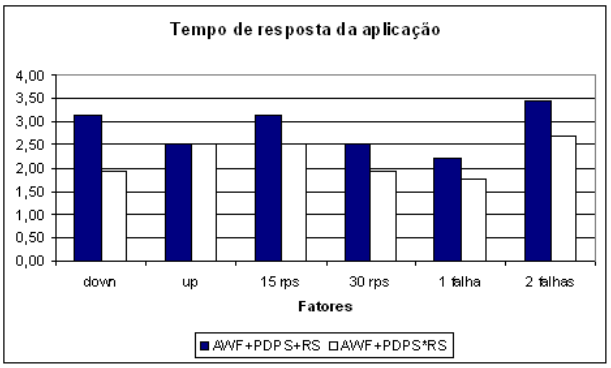

Figura 6.6: Tempo de resposta da aplicação considerando diferentes composições

As técnicas de coordenação proporcionaram ganhos em termos de disponibilidade e

tempo de resposta em todos os cenários. Em média, a disponibilidade melhorou 0,148% e o tempo de resposta 21,08%.

A Tabela 6.2 apresenta os ganhos de disponibilidade e tempo de resposta obtidos ao aplicar as técnicas de coordenação. Estes resultados estão ilustrados nos gráficos das Figuras 6.7 e 6.8.

Tabela 6.2: Ganhos médios de disponibilidade

	down	*up*	*15 rps*	*30 rps*	*1 falha*	*2 falhas*
$g^{avail}_{PDPS*RS}$	0,296%	0,000%	0,119%	0,176%	0,121%	0,174%
$g^{t}_{PDPS*RS}$	38,47%	1,07%	19,89%	24,07%	20,32%	22,68%

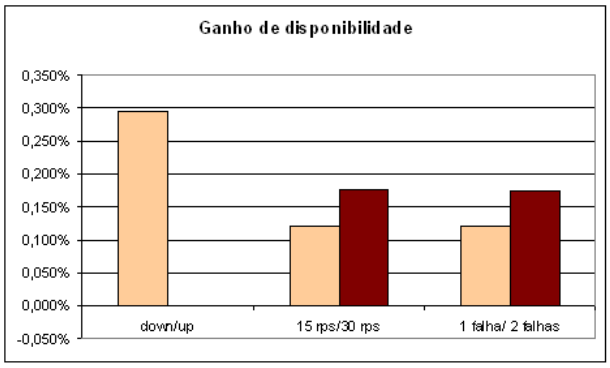

Figura 6.7: Ganho em termos de disponibilidade

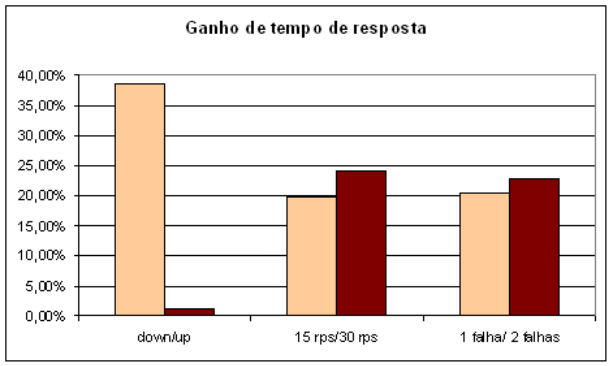

Figura 6.8: Ganho em termos de tempo de resposta

A técnica de rejuvenescimento coordenado, mais uma vez, só melhorou o tempo de resposta, ainda assim, muito superficialmente. Já a técnica de desativação coordenada de nós, como também discutido anteriormente, mostrou-se eficiente, melhorando substancialmente a disponibilidade e o tempo de resposta da aplicação.

6.4 Resultados obtidos para Cenário I: cargas *up* e *down* de menor intensidade

Vinte simulações de cada um dos tratamentos foram realizadas, totalizando 320 simulações da composição *AWF+DPS⋆RS*. Intervalos de 95% de confiança das composições *AWF+DPS*RS* foram identificados para as métricas disponibilidade, tempo de resposta e número de nós ativos e tais intervalos podem ser encontrados no Apêndice B.

A disponibilidade da aplicação e os tempos de resposta medidos em diferentes tratamentos são apresentados nas Figuras 6.9 e 6.10.

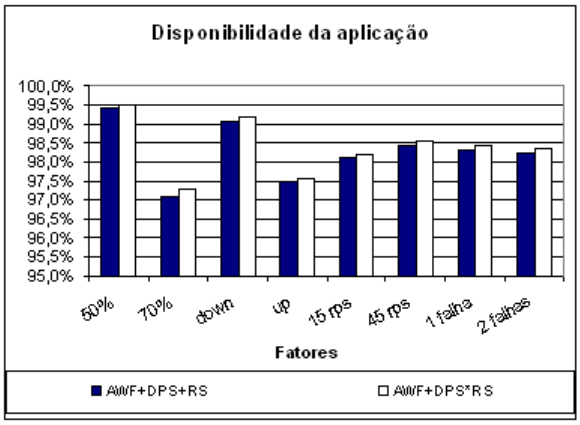

Figura 6.9: Disponibilidade computada no Cenário I

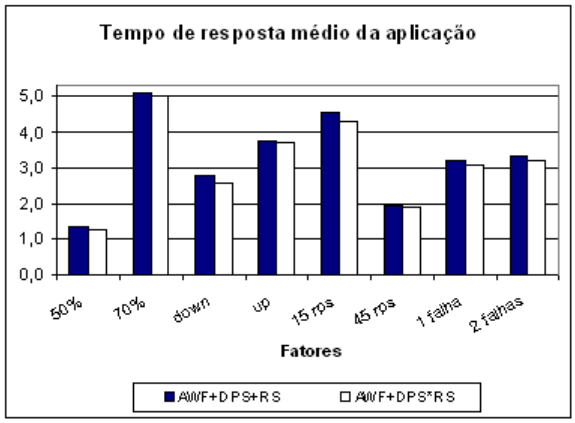

Figura 6.10: Tempo médio de resposta computado no Cenário I

Observa-se visualmente que houve ganho em termos de disponibilidade e tempo de resposta na maioria dos casos em que a coordenação foi realizada. O ganho médio de disponibilidade foi de 0,119% e o de tempo de resposta foi, em média, de 4,2%.

Na Figura 6.11 são apresentados ganhos em termos de disponibilidade ao aplicar as técnicas de coordenação. O baixo ganho em termos de disponibilidade pode passar a idéia inicial de que a disponibilidade não muda substancialmente ao aplicar as técnicas de coordenação. No entanto, pequenas variações na disponibilidade podem representar grandes diferenças em termos de número de requisições servidas, como pode ser visto na Figura 6.12. Nesta figura são apresentadas as quantidades médias de requisições que foram atendidas a mais por conta das técnicas de coordenação. Observa-se que milhares de requisições foram atendidas com sucesso por conta das técnicas. Em média, 8947 requisições foram atendidas por conta das técnicas de coordenação. A perda l^{rej}_{DPS*RS} é, em média, 29,6% menor que a perda l^{rej}_{DPS+RS} quando todo o intervalo de tempo simulado foi considerado. Durante os períodos de ocorrência de falhas e rejuvenescimentos esta diminuição da perda de requisições por conta das técnicas de coordenação foi de 42,4%.

Observou-se que todas as técnicas de coordenação foram usadas. A técnica de rejuvenescimento coordenado foi mais usada quando a carga *up* foi aplicada. Já a técnica de redução coordenada de capacidade foi mais aplicada durante a realização da carga *down*. No entanto, não houve uma exclusividade de uma técnica ou outra para uma carga ou outra. A técnica do quorum mínimo só foi usada quando a utilização alvo do *DPS* foi de 70%, independente da carga de trabalho realizada.

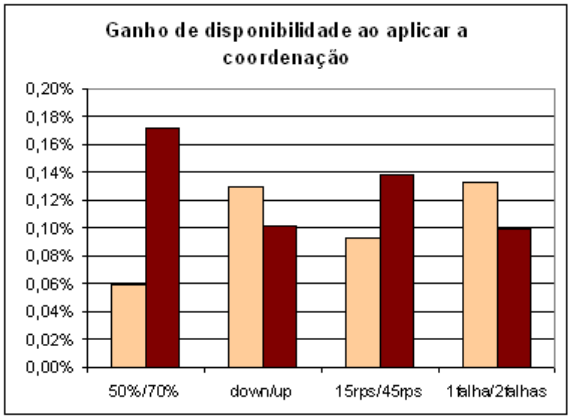

Figura 6.11: Ganho de disponibilidade ao aplicar a coordenação no Cenário I

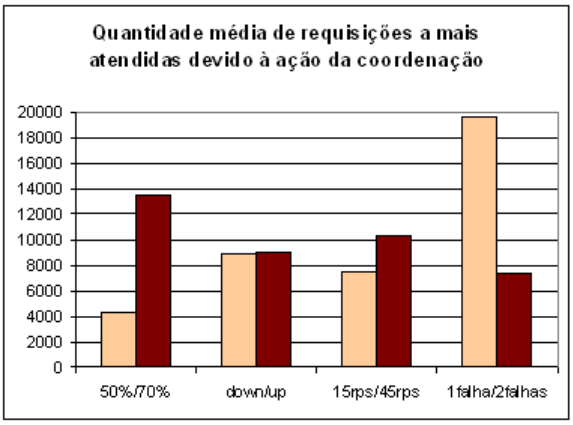

Figura 6.12: Requisições aceitas ao aplicar a coordenação no Cenário I

Em termos de utilização alvo do *DPS*, o maior ganho se deu quando a utilização foi maior. Foi neste caso também que a união sem coordenação de *DPS* e *RS* foi menos eficiente. Como discutido no Capítulo 5, a união sem coordenação de *DPS* e *RS* foi menos eficiente pois: (i) os nós em falha recebem mais requisições e (ii) a capacidade extra para abrigar a carga durante os rejuvenescimentos é menor. Como estas falhas e estes rejuvenescimentos causam muito mais danos em termos de qualidade de serviço da aplicação, então a retirada antecipada de nós em falha ou a redução do tempo de rejuvenescimento traz, conseqüentemente, mais benefícios.

Os ganhos em termos de disponibilidade medidos quando a carga *down* foi aplicada foram maiores. Apesar de alguns rejuvenescimentos coordenados terem sido necessários quando esta carga foi aplicada, a grande maioria dos nós em falha foram retirados antecipadamente através da desativação coordenada de recursos. Este maior ganho de disponibilidade quando a carga *down* foi aplicada sugere a superioridade desta técnica em relação às demais.

Observou-se que quando os nós de maior capacidade foram usados o ganho de disponibilidade também foi maior. A mesma razão pela qual a falha ou rejuvenescimento de nós de maior capacidade é mais danosa para a aplicação explica o fato de as técnicas também serem mais benéficas para tais nós. Quando um nó de maior capacidade que está em falha é retirado pelo *DPS* através de redução coordenada, então o benefício é maior se comparado ao benefício de se retirar um nó de menor capacidade. Foi por esta razão que as técnicas de coordenação se mostraram mais eficientes quando nós de maior capacidade estiveram envolvidos.

Finalmente, a disponibilidade da aplicação melhorou mais quando apenas uma falha

ocorreu. Em geral, um nó em falha foi retirado através da redução coordenada. Nos cenários em que ocorreram duas falhas, então, em alguns casos, especialmente quando a carga *up* foi aplicada, o segundo nó foi retirado através do rejuvenescimento coordenado. É por esta razão que a melhora quando apenas um nó falhou foi superior à melhora observada quando dois nós falharam. O primeiro nó em falha foi mais freqüentemente retirado antecipadamente da aplicação e os efeitos das falhas foram reduzidos.

Os ganhos mais substanciais devido à coordenação se dão durante a ocorrência de falhas e de rejuvenescimentos. Na Figura 6.13 são apresentadas as quantidades médias de requisições atendidas devido à coordenação durante a ocorrência de falhas e rejuvenescimentos. Esta é a quantidade total média de requisições rejeitadas pela aplicação em composições *AWF+DPS+RS* diminuída da quantidade total média de requisições rejeitadas pela aplicação nas composições *AWF+DPS*RS* correspondentes. Estes ganhos correspondem especialmente à atuação das técnicas de redução coordenada e rejuvenescimento coordenado.

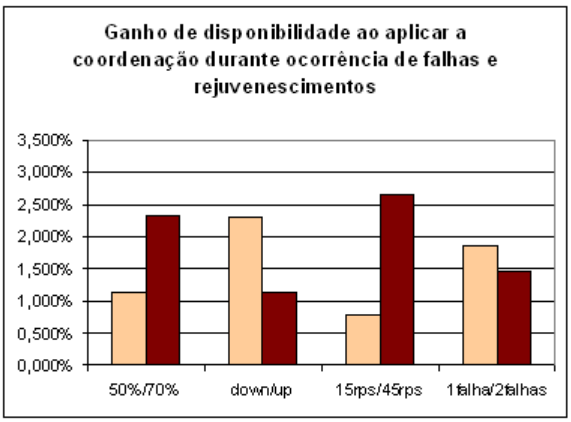

Figura 6.13: Quantidade de requisições aceitas devido à coordenação durante ocorrência de falhas e rejuvenescimentos

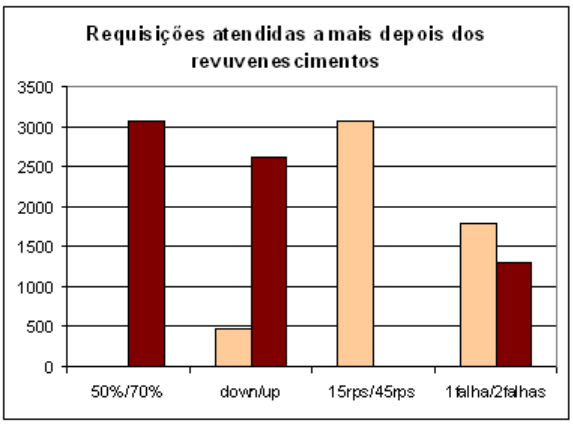

Figura 6.14: Quantidade de requisições aceitas devido à coordenação após a ocorrência de falhas e rejuvenescimentos

Recuperadas as falhas, as técnicas que ainda continuam sendo usadas são o quorum mínimo e a redução de capacidade coordenada. Depois que as falhas são recuperadas, os ganhos observados são bem menores, conforme indicado na Figura 6.14. Os maiores ganhos são observados para cenários com alta utilização alvo, carga *up* e nós de menor capacidade. Estas são exatamente as características dos cenários em que a técnica de quorum mínimo foi utilizada. A Tabela 6.3 traz os percentuais de ocorrência do quorum mínimo em diferentes configurações.

Os ganhos em termos de tempo de resposta para todo o intervalo estudado estão ilustrados na Figura 6.15. Não são observados ganhos substanciais, apesar dos intervalos de

Tabela 6.3: Percentual de ocorrências do quorum mínimo

Utilização do *DPS*	50%	0%
	70%	100%
Carga de trabalho	*up*	72,5%
	down	27,5%
Capacidade dos nós	15 rps	81,6%
	45 rps	18,4%
Número de falhas	1	54,3%
	2	45,7%

confiança não se sobreporem na maioria dos cenários (vide Apêndice B). O maior ganho em termos de tempo de resposta ocorreu nos casos em que a utilização alvo do *DPS* foi menor. Este foi o caso em que a união simples do *DPS* e do *RS* foi mais eficiente. Sendo assim, em ambos os cenários (com e sem coordenação) ocorreram menos filas cheias e, conseqüentemente este é o cenário em que se pode avaliar melhor o efeito da coordenação em termos de tempo de resposta. Quando as filas enchem com mais freqüência, então os ganhos advindos da coordenação aparecem muito mais em termos de disponibilidade do que em termos de tempo de resposta, razão pela qual observa-se na maioria dos casos que, quanto maior foi o ganho de disponibilidade, menor foi o do tempo de resposta. Este resultado sugere que as técnicas de coordenação trazem também ganhos em termos de tempo de resposta da aplicação, e quanto menores forem as perdas de requisições, maiores são os ganhos em termos de tempo de resposta.

Os ganhos de tempo de resposta quando a carga *down* foi aplicada são bem maiores do que os ganhos quando a carga *up* foi aplicada. Além do que foi anteriormente discutido (em geral, as rejeições de requisições devido a filas cheias terem sido menores para esta carga), este resultado sugere mais uma vez a eficiência da técnica de redução coordenada de capacidade, uma vez que esta foi a técnica mais comumente utilizada quando tal carga foi aplicada.

Quando nós de maior capacidade foram usados, os ganhos de tempo de resposta foram maiores. Como já discutido anteriormente, nós de maior capacidade exercem maior influência para a aplicação, seja negativamente - quando não há coordenação - seja positivamente - quando a coordenação é aplicada e tais nós são antecipadamente retirados.

Repetindo os resultados já encontrados na seção anterior, os ganhos em termos de tempo de resposta foram mais intensos durante a ocorrência de falhas e de rejuvenescimento. Estes

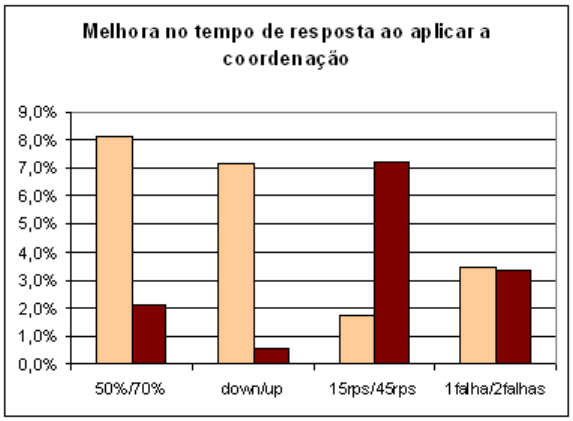

Figura 6.15: Melhora do tempo de resposta devido à coordenação durante ocorrência de falhas e rejuvenescimentos

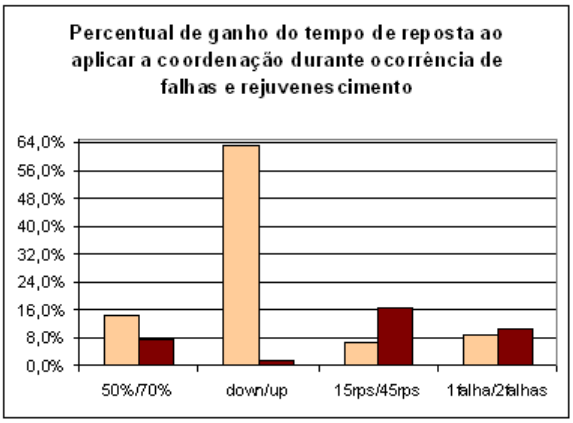

Figura 6.16: Melhora do tempo de resposta devido à coordenação durante a ocorrência de falhas e rejuvenescimentos

ganhos são ilustrados no gráfico da Figura 6.16. O maior ganho foi observado quando a carga *down* foi aplicada, evidenciando o poder da técnica de rejuvenescimento coordenado, aplicada mais freqüentemente para esta carga. Em seguida, acompanhando as mesmas tendências observadas quando todo o intervalo de tempo foi considerado, os maiores ganhos são vistos quando para a menor utilização e para nós de maior capacidade.

Quando as técnicas de coordenação foram aplicadas observou-se, além dos ganhos já discutidos, uma quantidade média de nós ativos ligeiramente inferior à quantidade média de nós ativos na composição *AWF+DPS+RS*. Os intervalos de confiança para a quantidade média de nós se sobrepõem em poucos intervalos (vide Apêndice B), sendo possível afirmar que a coordenação diminuiu o custo do *DPS* em 0,12%, em média. Esta redução de custo não é substancial, mas demonstra que as técnicas de coordenação, ao mesmo tempo que melhoram a qualidade de serviço da aplicação, não aumentam o custo em termos de número de nós ativos.

6.5 Resultados obtidos para Cenário II: cargas *up* e *down* de maior intensidade

Vinte simulações de cada um dos tratamentos foram realizadas, totalizando 320 simulações da composição *AWF+DPS⋆RS*. Intervalos de 95% de confiança das composições *AWF+DPS*RS* foram identificados para as métricas disponibilidade, tempo de resposta e número de nós ativos e tais intervalos podem ser encontrados no Apêndice B.

A disponibilidade da aplicação e os tempos de resposta medidos em diferentes tratamentos são apresentados nas Figuras 6.17 e 6.18.

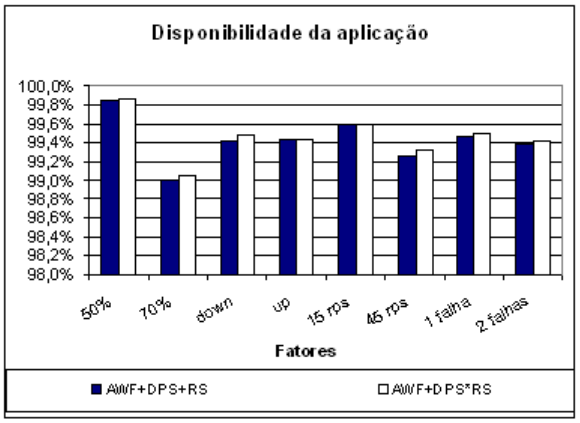

Figura 6.17: Disponibilidade computada no Cenário II

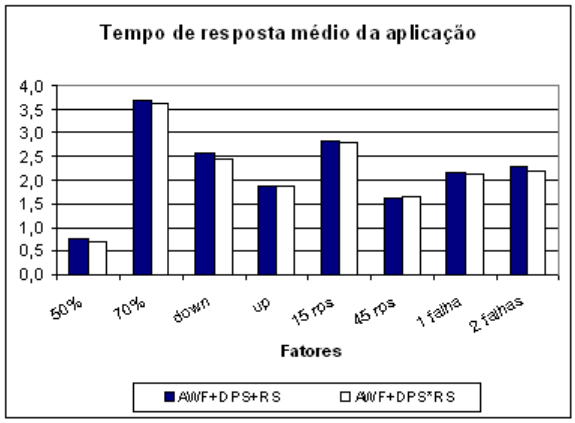

Figura 6.18: Tempo médio de resposta computado no Cenário II

Neste cenário em que o número médio de nós ativos é maior, observam-se ganhos menores em termos de disponibilidade e tempo de resposta quando as técnicas de coordenação são aplicadas. O ganho médio de disponibilidade foi de 0,039% e o de tempo de resposta foi, em média, de 4,9%. Este aumento de disponibilidade resultou em uma média de 5290 requisições sendo atendidas a mais por conta das técnicas de coordenação. A perda l^{rej}_{DPS*RS} é, em média, 27,93% menor que a perda l^{rej}_{DPS+RS} quando todo o intervalo de tempo simulado foi considerado. Durante os períodos de ocorrência de falhas e rejuvenescimentos esta diminuição da perda de requisições por conta das técnicas de coordenação foi de 28,7%.

Na Figura 6.19 são apresentados ganhos em termos de disponibilidade ao aplicar as técnicas de coordenação. Neste cenário houve o uso exclusivo da técnica de redução de capacidade quando a carga *down* foi aplicada e de rejuvenescimento coordenado quando a carga *up* foi realizada. Evidencia-se através destes experimentos a superioridade da técnica de redução coordenada de capacidade, já percebida no Cenário I.

Na Figura 6.20 são apresentadas as quantidades médias de requisições que foram atendidas por conta das técnicas de coordenação. Observa-se que em muitos casos, milhares de requisições foram atendidas com sucesso por conta das técnicas. As mesmas tendências observadas no Cenário I se mantém pelas mesmas razões, não sendo necessário explicar novamente as razões pelas quais uma ou outra configuração foram mais ou menos beneficiadas pela coordenação.

Os ganhos mais substanciais devido à coordenação são vistos durante a ocorrência de falhas e de rejuvenescimentos. Na Figura 6.21 são apresentadas as quantidades médias de

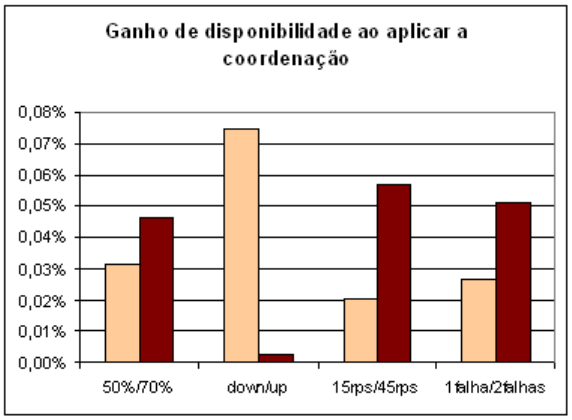

Figura 6.19: Ganho de disponibilidade ao aplicar a coordenação no Cenário II

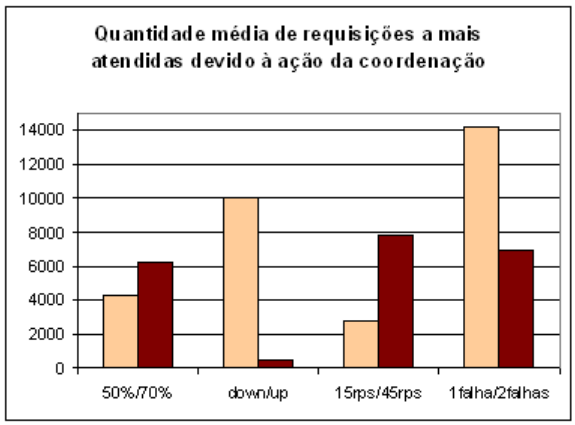

Figura 6.20: Requisições aceitas ao aplicar a coordenação no Cenário II

requisições atendidas devido à coordenação durante a ocorrência de falhas e rejuvenescimentos. As mesmas tendências discutidas na seção anterior são vistas aqui, não havendo novamente a necessidade de discussão extra. Depois que as falhas são recuperadas, os ganhos observados são bem menores, conforme indicado na Figura 6.22. Em alguns cenários, inclusive, observam-se perdas em relação ao cenário *AWF+DPS+RS*. Os maiores ganhos são observados para cenários com alta utilização alvo, carga *down* e nós de menor capacidade. Estas são exatamente as características dos cenários em que a técnica de quorum mínimo foi utilizada. A Tabela 6.4 traz os percentuais de ocorrência do quorum mínimo em diferentes configurações.

Tabela 6.4: Percentual de ocorrências do quorum mínimo

Utilização do *DPS*	50%	0%
	70%	100%
Carga de trabalho	*up*	30%
	down	70%
Capacidade dos nós	15 rps	100%
	45 rps	0%
Número de falhas	1	50%
	2	50%

Analisando mais detalhadamente os resultados, percebeu-se que para a carga *down* quando os nós eram de maior capacidade, e dois nós iniciavam em falha, o *RS* realizou o rejuvenescimento de um nó sobrecarregado durante a simulação. Este rejuvenescimento

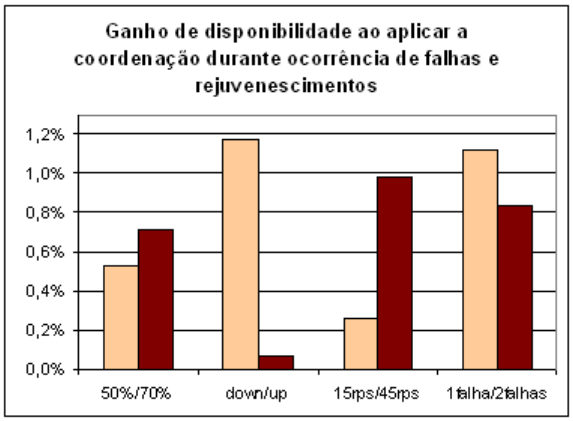

Figura 6.21: Quantidade de requisições aceitas devido à coordenação durante ocorrência de falhas e rejuvenescimentos

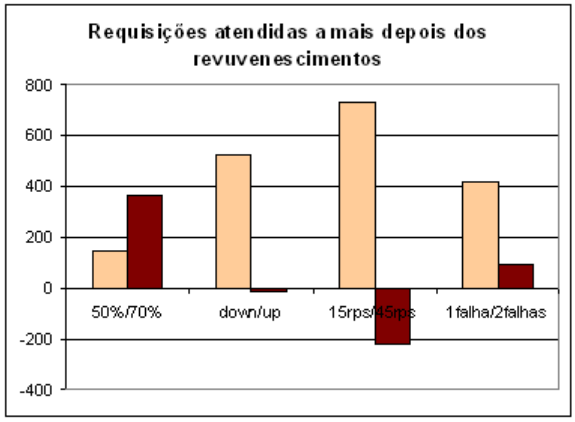

Figura 6.22: Quantidade de requisições aceitas devido à coordenação após a ocorrência de falhas e rejuvenescimentos

ocorreu tanto para o sistema coordenado como para o não coordenado, uma vez que o quorum mínimo não era suficiente para impedir tais rejuvenescimentos. Quando a coordenação foi aplicada, este rejuvenescimento ocorreu em 86% dos experimentos com esta configuração. Já quando a coordenação não foi aplicada, este rejuvenescimento ocorreu em 64% das execuções. Esta foi a razão pela qual a quantidade de requisições perdidas quando nós de maior capacidade foram aplicados foi maior quando a coordenação foi aplicada.

Os ganhos em termos de tempo de resposta computados para todo o intervalo estudado e para o intervalo em que ocorreram falhas e rejuvenescimentos são ilustrados respectivamente nos gráficos das Figuras 6.23 e 6.24. Estes ganhos reforçam a eficiência da técnica redução coordenada de capacidade aplicada quando a carga *down* foi realizada. A técnica de rejuvenescimento coordenado não resultou em ganhos de tempo de resposta.

Quando as técnicas de coordenação foram aplicadas observou-se, além dos ganhos já discutidos, que a quantidade média de nós utilizados foi 0,19% inferior à quantidade média de nós ativos na composição *AWF+DPS+RS*. Esta redução equivale a 0,11 nós. Os intervalos de confiança para a quantidade média de nós não se sobrepõem (vide Apêndice B), sendo possível afirmar que a coordenação reduziu ligeiramente o custo em termos de número de nós ativos.

6.6 Influência do número de nós ativos

No Capítulo 5 percebeu-se que quando o número de nós ativos aumenta, as falhas e rejuvenescimentos influenciam menos a qualidade de serviço da aplicação gerenciada simulta-

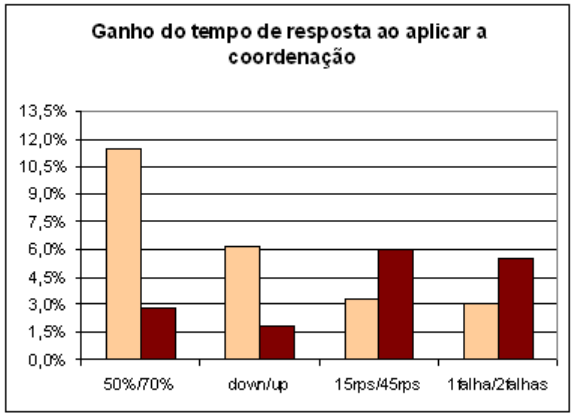

Figura 6.23: Melhora do tempo de resposta devido à coordenação durante ocorrência de falhas e rejuvenescimentos

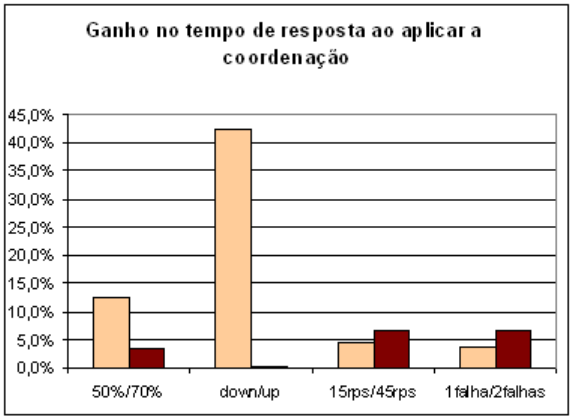

Figura 6.24: Melhora do tempo de resposta devido à coordenação durante a ocorrência de falhas e rejuvenescimentos

neamente por um *DPS* e um *RS* sem coordenação. De forma semelhante, percebe-se aqui que quando o número de nós ativo aumenta, os ganhos observados ao aplicar as técnicas de coordenação também diminuem.

Observou-se que os ganhos em termos de qualidade de serviço foram mais intensos no Cenário I, onde, em média, 30 nós permaneceram ativos durante o intervalo completo de simulação. A quantidade média de nós ativos neste cenário está ilustrada na Figura 5.22 apresentada no Capítulo 5. No Cenário II uma média de 63 nós permaneceram ativos durante o intervalo completo de simulação. A quantidade média de nós ativos neste cenário está ilustrada na Figura 5.23 apresentada no Capítulo 5. Na Tabela 6.5 são apresentados os ganhos de disponibilidade observados nos Cenários I e II por carga de trabalho.

Tabela 6.5: Ganhos de disponibilidade por carga aplicada nos Cenários I e II

		Ganho de disponibilidade
Cenário I	*Carga down*	0,134%
	Carga up	0,104%
Cenário II	*Carga down*	0,075%
	Carga up	0,003%

Quando o intervalo $[0, t_{fimDoRejuvenescimento}]$ é considerado, estes ganhos crescem, segundo apresentado na Tabela 6.6. Quando este intervalo foi considerado os ganhos de disponibilidade também foram inferiores no Cenário II.

Estes resultados indicam que as técnicas de coordenação influenciam mais positivamente

Tabela 6.6: Ganhos de disponibilidade por carga aplicada nos cenários I e II durante ocorrência de falhas e rejuveenscimento

		Ganho de disponibilidade
Cenário I	*Carga down*	2,356%
	Carga up	1,152%
Cenário II	*Carga down*	1,169%
	Carga up	0,073%

uma aplicação onde menos nós estão ativos, pois quanto menor é a quantidade de nós ativos, menor influência este nó terá, seja negativamente, seja positivamente para a aplicação.

6.7 Conclusões parciais

Três técnicas simples de coordenação entre atividades de provisão dinâmica e rejuvenescimento foram propostas. Estas técnicas foram estudadas através de simulações e observou-se que elas são eficientes, melhorando a qualidade de serviço da aplicação.

É plausível imaginar que existe um limite que indica quanto as técnicas de coordenação podem melhorar a qualidade de serviço de uma aplicação. Falhas ocorrem em um ambiente onde a utilização dos recursos é intensa (pelo menos espera-se que seja) e existe um certo tempo para que sejam detectadas e recuperadas. Uma vez que as falhas ocorram, não há como reverter os danos causados, ou reduzi-los aos mesmos efeitos percebidos em um ambiente super-provido estaticamente de recursos. A técnica de rejuvenescimento coordenado visa minimizar o custo dos rejuvenescimentos diminuindo o período necessário para realizá-lo. A técnica de quorum mínimo visa apenas evitar que o *RS* rejuvenesça nós desnecessariamente. Já a técnica de redução coordenada de capacidade pode ser considerada uma técnica de prevenção de falhas. O *DPS* está constantemente adicionando e retirando nós da aplicação. Ao retirar sempre os nós mais propensos a falhas está-se reduzindo a probabilidade de um nó ativo falhar. Esta foi a técnica que se mostrou mais eficiente segundo os estudos realizados.

Quanto mais poderoso for o *RS* ao reconhecer nós que estão na iminência de falhar, mais poderosa se torna esta técnica de coordenação. Na realidade, se o *RS* não tiver esta capacidade, então esta técnica não tem efeito algum, pois os nós a serem desativados acabarão sendo escolhidos aleatoriamente, tal qual o são quando não há coordenação. Formas de caracterizar o estado de servidores fogem do escopo deste trabalho, mas já vêm sendo

tema de pesquisa de alguns grupos de pesquisa (HELLERSTEIN, 1996; AVRITZER; BONDI; WEYUKER, 2005; AVRITZER et al., 2006; ZHANG; HELLERSTEIN, 2000; AVRITZER; BONDI; WEYUKER, 2007; FOX; KICIMAN; PATTERSON, 2004; ANDRZEJAK; SILVA, 2007).

Ao aplicar as técnicas de coordenação entre *DPS* e *RS* perde-se o conceito de rejuvenescimento. Em ambas as técnicas não ocorre rejuvenescimento em si, mas uma colaboração entre *DPS* e *RS* de forma que os nós em falha são retirados da aplicação.

Ao aumentar o número de nós ativos percebe-se que ao mesmo tempo que a influência de falhas e rejuvenescimentos é menor, é menor também o ganho de se aplicar a coordenação. Quando o número de nós ativos é maior, então a falha ou rejuvenescimento de um nó influenciará muito menos a qualidade de serviço da aplicação e, conseqüentemente, a recuperação de tal falha

As técnicas de coordenação descritas aqui podem ser, em sistemas futuros, características do próprio *DPS*. O próprio *DPS*, que já deve estar munido de uma série de monitores, pode passar a detectar quais nós estão em falha, rejuvenescendo-os quando adequado e escolhendo os mais propensos a falhar durante a redução de capacidade. Para o *DPS* é também mais simples diferenciar entre sobrecarga e falhas, uma vez que ele também tem conhecimento sobre a carga corrente da aplicação. Portanto, acredita-se que não é preciso necessariamente que haja um outro sistema, o de rejuvenescimento, desde que o *DPS* esteja preparado para realizar as atividades de recuperação de falhas.

Finalmente, observou-se que as técnicas de coordenação não apenas melhoram a qualidade de serviço da aplicação, mas também reduzem o custo de mantê-la em termos de número de nós ativos. Em geral, há um *tradeoff* entre qualidade de serviço e custo, de forma que quanto melhor a qualidade de serviço, maior é o custo. As técnicas de coordenação, no entanto, melhoram a qualidade de serviço porém sem acrescentar no custo, pois elas possibilitam a sinergia entre os sistemas de provisão dinâmica e rejuvenescimento.

O simulador usado para obter os resultados apresentados neste capítulo foi verificado e validado através de testes de limite e experimentos com um modelo de medição correspondente. A descrição completa destas atividades pode ser encontrada no Apêndice D. Foi possível observar também através de medições melhoras na disponibilidade e tempo de resposta da aplicação quando a coordenação foi aplicada. Os experimentos de medição, apesar de apresentarem valores pontuais diferentes dos computados pelo modelo de simulação, apresentaram tendências que seguem os mesmos padrões dos encontrados no modelo de simulação. Como no decorrer desta tese os valores em si não foram usados, mas sim as tendências de diferenças entre tais valores, então considera-se que o modelo de simulação usado é adequado para este fim.

Capítulo 7

Conclusões e Trabalhos futuros

7.1 Resultados e contribuições

O objetivo desta tese foi estudar situações em que sistemas de provisão dinâmica e de rejuvenescimento que atuam sobre uma mesma aplicação e desenvolver técnicas que permitam uma coexistência mais harmônica entre tais sistemas. A seguir são apresentadas as contribuições obtidas no decorrer desta pesquisa.

Inicialmente foi possível identificar analiticamente situações em que a qualidade de serviço de aplicações piora enquanto o rejuvenescimento ocorre, especialmente quando um sistema de provisão dinâmica está atuando (detalhes no Capítulo 3). Sempre que o rejuvenescimento de um ou mais nós durar mais do que o tempo necessário para encher as filas dos servidores que permaneceram ativos, então a disponibilidade da aplicação será afetada por conta do rejuvenescimento. Na prática, este tempo pode ser menor do que o tempo necessário para encher as filas, como percebido em experimentos de medição em que a própria sobrecarga no nó degradado causa falhas antes mesmo que as filas fiquem cheias. Os tempos de resposta serão afetados sempre que o rejuvenescimento durar tempo suficiente para aumentar as filas dos servidores que permanecem ativos até que um limiar seja ultrapassado em que os tempos de resposta não são mais satisfatórios.

O rejuvenescimento de software é uma técnica bem conhecida e bem aceita, sendo bem aceito também o fato de que ele incorre em algum custo. Assim, além do custo em termos de qualidade de serviço proveniente das falhas, existe também um custo proveniente de se retirar máquinas de uma aplicação para serem rejuvenescidas. Através de experimentos de simulação e de medição foi possível mostrar que em ambientes em que os *e-services* estão super-providos de recursos o custo das falhas e dos rejuvenescimentos é menor do que em ambientes em que há a provisão dinâmica de recursos.

Através de experimentos de simulação evidenciou-se que a união simples de um *DPS* e um *RS* resulta em perdas maiores em termos de qualidade de serviço da aplicação devido à atuação do *DPS* e devido a falhas e rejuvenescimentos do que as perdas que seriam observadas caso estes sistemas estivessem agindo isoladamente. Além disso, foi possível identificar situações em que esta união simples é ainda mais ineficiente em termos de qualidade de serviço da aplicação. De acordo com os resultados obtidos foram identificados fatores que contribuem mais fortemente para a ineficiência da união simples do *DPS* e *RS*, a saber: (i) maior utilização mantida pelo *DPS* nos recursos, (ii) aumento da carga de trabalho no momento da falha e rejuvenescimento e (iii) pequeno número de nós ativos no momento da falha e rejuvenescimento.

Obviamente, uma vez que se aceita o fato de que as aplicações podem falhar, não é possível excluir a ação do *RS*, capaz de recuperar as falhas. Também não se pode excluir o *DPS* por questões de custo e por questões operacionais relacionadas ao novo ambiente de entrega sob demanda. Uma vez evidenciado que falhas e rejuvenescimento causam muito mais danos em uma aplicação onde ambos, *DPS* e *RS*, atuam simultaneamente, o próximo passo foi estudar formas de minimizar estes danos.

Técnicas de colaboração entre *DPS* e *RS* foram propostas e ganhos de qualidade de serviço foram observados (detalhes no Capítulo 6), além de redução de custo em termos de número de nós ativos. Através de simulações foi possível observar que as técnicas de coordenação melhoram a qualidade de serviço da aplicação se comparada à qualidade da aplicação medida quando os sistemas atuavam sem coordenação. Estas técnicas podem ser seguidas para se construir um sistema de gerência novo - como foi feito no DynAlloc-SR (LOPES; CIRNE; BRASILEIRO, 2004) - e podem também ser seguidas para coordenar as atividades de sistemas já existentes, requerendo algumas mudanças em tais sistemas legados.

Os ganhos não tornam a qualidade de serviço da aplicação tão boa quanto a qualidade observada em uma aplicação sem falhas na qual o *DPS* atua, nem tão boa quanto a qualidade observada em um ambiente com falhas, super-provido de recursos em que apenas o *RS* atua. De fato, as falhas causam danos irreversíveis à qualidade de serviço da aplicação, especialmente se ocorrem em momentos em que os recursos estão sobrecarregados.

No entanto, especialmente a técnica de redução coordenada de capacidade, apresentou um elevado grau de eficiência quando o *RS* é capaz de identificar corretamente os nós em falha. Ao munir o *DPS* com informação para, ao reduzir a capacidade da aplicação, retirar os nós em falha ou mais propensos a falhar, então pode haver prevenção de falhas e os ganhos de se aplicar as técnicas de coordenação são maiores. As informações sobre nós em

falha não precisam vir, necessariamente, de um sistema de rejuvenescimento. O próprio sistema de provisão dinâmica pode ter monitores que o auxiliem a decidir sobre quem deve ser o próximo nó a ser desativado em vez de realizar esta escolha aleatoriamente como é feito tradicionalmente.

Observou-se também através de experimentos de simulação que mesmo quando o *DPS* é um sistema perfeito (*PDPS*) e toma sempre as melhores decisões à respeito do número de nós a serem ativados, a união pura de *PDPS* e *RS* não é satisfatória em termos de qualidade de serviço da aplicação. Neste contexto, as técnicas de coordenação foram aplicadas e observou-se uma melhora substancial da qualidade de serviço da aplicação, especialmente quando a técnica de redução coordenada de capacidade foi aplicada.

Finalmente, através de experimentos de medição foi possível observar, na prática, comportamentos semelhantes aos verificados através dos experimentos de simulação tanto no que se refere aos danos causados pelas falhas e rejuvenescimentos em um ambiente onde *DPS* e *RS* atuam sem coordenação quanto aos benefícios de se coordenar as atividades destes sistemas (detalhes no Capítulo D).

Diante do exposto, esta tese traz as seguintes contribuições à comunidade científica:

1. Estudo de interações que ocorrem quando sistemas de provisão dinâmica e de rejuvenescimento atuam sobre uma mesma aplicação sem coordenação. Os trabalhos disponíveis na literatura sobre provisão dinâmica de recursos não consideram que aplicações podem falhar e, portanto, os sistemas de provisão dinâmica existentes não estão preparados para lidar com falhas automaticamente. Da mesma forma, os sistemas de rejuvenescimento de software propostos na literatura não consideram a possibilidade de se ter um conjunto de máquinas que muda dinamicamente de tamanho ao longo do tempo para não deixar os recursos subutilizados. Com este estudo realizado, estas lacunas foram preenchidas;

2. Aumento do nível de autonomia de aplicações do tipo *e-service* através das técnicas de coordenação propostas. As técnicas de coordenação propostas definem novas funcionalidades para o *DPS* e o *RS* de forma que eles começam a colaborar entre si para uma melhor qualidade de serviço da aplicação. Como já mencionado ao longo desta tese, em especial no Capítulo 2, encontra-se na literatura propostas de sistemas de provisão dinâmica, bem como propostas sistemas de rejuvenescimento. No entanto, tais sistemas, apesar de úteis para uma ampla classe de aplicações, não estavam preparados para atuar sobre uma mesma aplicação. O trabalho desenvolvido nesta tese possibilita uma colaboração entre o *DPS* e o *RS* de forma que as perdas

em termos de QoS de se ter provisão dinâmica, falhas e rejuvenescimentos juntos são reduzidas. Esta colaboração permite, portanto, que *e-services* sejam gerenciados simultaneamente sob o aspecto da provisão dinâmica de recursos e do rejuvenescimento de software de forma mais eficiente;

3. Utilização de metodologia própria para estudo de interações e coordenação entre as ações de sistemas de gerência que atuam sobre um mesmo alvo. Com a tendência de computação autônoma e gerência holística da aplicação, diversos sistemas de gerência terão que atuar sobre uma mesma aplicação (WILKES; MOGUL; SUERMONDT, 2004). A metodologia de estudo usada nesta tese foi aplicada para resolver o problema desta tese, no entanto, acredita-se que ela possa ser adaptada para estudar interações entre outros sistemas de gerência. Não foram encontrados na literatura exemplos de metodologias que permitam o estudo das interações entre sistemas autonômicos de gerência distintos;

4. Estudo analítico de situações em que rejuvenescimento afeta a qualidade de serviço da aplicação. Um modelo analítico foi proposto para analisar as condições em que o rejuvenescimento de um ou mais nós sobrecarrega os nós que permanecem ativos. Este modelo considera que todas as requisições demandam a mesma quantidade de serviço, dada por uma média das requisições reais, que são diferentes. Apesar da média ser uma métrica sumarizadora representativa, nem sempre os resultados do modelo analítico estarão condizentes com a realidade, especialmente quando ocorrerem picos e vales bem definidos no intervalo estudado;

5. Um modelo de simulação que se propõe a simular o comportamento de uma aplicação gerenciada por um *DPS* e/ou um *RS*, na presença ou não de falhas foi proposto e validado através de experimentos de medição;

6. Identificação através de projetos experimentais de características benéficas à provisão dinâmica de recursos e rejuvenescimento de software. O projeto experimental aumentou o nível de conhecimento sobre o que influencia mais fortemente a qualidade de serviço da aplicação em termos de disponibilidade quando gerenciada por um *DPS* apenas, por um *RS* apenas e por ambos. Com este estudo foi possível identificar e variar os parâmetros mais importantes para tais sistemas, inclusive quando atuam simultaneamente. Este estudo foi importante, não apenas em termos metodológicos, mas também foi um trabalho bastante original no sentido de aplicar a técnica de projeto experimental para obter informações sobre um problema real. Apesar do

projeto experimental ser muito útil em diferentes situações, não é comum encontrar na literatura o uso de tal técnica.

7.2 Atividades futuras

As seguintes atividades ficam como sugestão para pesquisas futuras:

1. *Caracterizar o comportamento de aplicações de múltiplas-camadas gerenciadas por DPS e RS não coordenados e coordenados.* Esta tese considera apenas aplicações de uma única camada. Esta decisão foi inicialmente tomada porque no início desta pesquisa (2004) os sistemas de provisão dinâmica ainda não consideravam múltiplas camadas, apesar de alguns grupos de pesquisa já estarem começando a desenvolver pesquisas nesta área. Suspeita-se que os resultados encontrados para aplicações de uma camada também serão válidos para aplicações de múltiplas camadas, porém, é também possível que existam outras situações que não são capturadas quando apenas uma camada existe.

2. *Realizar um estudo semelhante ao realizado dando ênfase ao aspecto de métricas de negócio.* Recentemente, uma nova área de pesquisa chamada "Business-driven IT Management" vem estudando o impacto da tecnologia da informação nos processos de negócios das empresas. Esta área de pesquisa traz indicativos de que é interessante tomar decisões na área de tecnologia da informação baseadas em dados do negócio, como sugere (BARTOLINI; SALLÉ, 2004; SAUVÉ et al., 2006). Não foi realizado nesta tese um estudo a respeito das perdas de não ter a coordenação e dos ganhos de aplicá-las para os negócios. Observou-se que não aplicando as técnicas de coordenação há perdas de disponibilidade e de tempo de resposta, que certamente serão refletidas no negócio. Mas não foi possível quantificar estas perdas, nem identificar, sob a ótica do negócio, se os ganhos de aplicar a coordenação são significativos. Quando a coordenação é aplicada as perdas são menores e em alguns casos o custo em termos de número de nós ativos é ligeiramente maior, indicando que em certos momentos, a coordenação traz ganhos e aumento de custo;

3. *Identificar algoritmos mais dinâmicos para o estabelecimento do quorum mínimo.* Observou-se que configurar de forma estática o quorum mínimo nem sempre resultou nas melhores decisões. Quando o *RS* decide inapropriadamente pelo rejuvenescimento de um número de nós aquém do quorum mínimo, tais nós serão rejuvenescidos, apesar de não estarem em falha. O quorum mínimo não pode ser tão pequeno, a ponto de

adiar muito o rejuvenescimento de nós em falha. Ao mesmo tempo, ele não pode ser muito grande, de forma que muitos nós sejam rejuvenescidos indevidamente durante períodos de sobrecarga. A escolha estática deste valor nem sempre é apropriada, conforme se viu nos resultados experimentais, mesmo quando se tem a ciência exata da quantidade de nós que está em falha. Assim, sugere-se uma investigação posterior à respeito da escolha deste valor.

Bibliografia

AIBER, S. et al. Autonomic self-optimization according to business objectives. In: *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*. Nova York, NY, EUA: IEEE Computer Society, 2004. p. 206–213.

ALMEIDA, J. et al. Resource management in the autonomic service-oriented architecture. In: *3rd IEEE International Conference on Autonomic Computing (ICAC '06)*. Dublin, Ireland: [s.n.], 2006.

ALMEIDA, V. A. F. et al. Self-adaptive sla-driven capacity management for internet services. In: *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*. Vancouver, Canadá: IEEE Press, 2006. p. 557–568. ISBN ISBN 1-4244-0143-7.

ANDRZEJAK, A.; SILVA, L. M. Deterministic models of software aging and optimal rejuvenation schedules. In: *Integrated Network Management*. [S.l.: s.n.], 2007. p. 159–168.

APPARAO, P.; MAKINENI, S.; NEWELL, D. Characterization of network processing overheads in xen. *vtdc*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 2, 2006.

APPLEBY, K. et al. Oceano - sla based management of a computing utility. In: *7th IFIP/IEEE International Symposium on Integrated Network Management*. [S.l.: s.n.], 2001. p. 855 –868.

ARLITT, M.; KRISHNAMURTHY, D.; ROLIA, J. Characterizing the scalability of a large web-based shopping system. *ACM Trans. Inter. Tech.*, ACM Press, v. 1, n. 1, p. 44–69, 2001. ISSN 1533-5399.

AVIZIENIS, A. et al. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Sec. Comput.*, v. 1, n. 1, p. 11–33, 2004.

AVRITZER, A. et al. Performance assurance via software rejuvenation: Monitoring, statistics and algorithms. *dsn*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 435–444, 2006.

AVRITZER, A.; BONDI, A.; WEYUKER, E. J. Ensuring stable performance for systems that degrade. In: *WOSP '05: Proceedings of the 5th international workshop on Software and performance*. New York, NY, USA: ACM Press, 2005. p. 43–51. ISBN 1-59593-087-6.

AVRITZER, A.; BONDI, A.; WEYUKER, E. J. Ensuring system performance for cluster and single server systems. *J. Syst. Softw.*, Elsevier Science Inc., New York, NY, USA, v. 80, n. 4, p. 441–454, 2007. ISSN 0164-1212.

AVRITZER, A.; WEYUKER, E. J. Monitoring smoothly degrading systems for increased dependability. *Empirical Softw. Engg.*, Kluwer Academic Publishers, Hingham, MA, EUA, v. 2, n. 1, p. 59–77, 1997. ISSN 1382-3256.

BARFORD, P.; CROVELLA, M. Generating representative web workloads for network and server performance evaluation. In: *SIGMETRICS '98/PERFORMANCE '98: Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*. Madison, Wisconsin, EUA: ACM Press, 1998. p. 151–160. ISBN 0-89791-982-3.

BARHAM, P. et al. Xen and the art of virtualization. In: *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM Press, 2003. p. 164–177. ISBN 1-58113-757-5.

BARTOLINI, C.; SALLÉ, M. Business driven prioritization of service incidents. In: *DSOM*. [S.l.: s.n.], 2004. p. 64–75.

BENNANI, M. N.; MENASCE, D. A. Resource allocation for autonomic data centers using analytic performance models. In: *ICAC '05: Proceedings of the Second International Conference on Automatic Computing*. Washington, DC, EUA: IEEE Computer Society, 2005. p. 229–240. ISBN 0-7965-2276-9.

BOBBIO, A.; SERENO, M.; ANGLANO, C. Fine grained software degradation models for optimal rejuvenation policies. *Perform. Eval.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 46, n. 1, p. 45–62, 2001. ISSN 0166-5316.

BROWN, A.; PATTERSON, D. A. To err is human. In: *EASY '01: Proceedings of the First Workshop on Evaluating and Architecting System Dependability*. Göteborg, Suécia: [s.n.], 2001.

BROWN, A. B. et al. Benchmarking autonomic capabilities: Promises and pitfalls. *icac*, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, p. 266–267, 2004.

BYDE, A.; SALLE, M.; BARTOLINI, C. *Market-Based Resource Aladdress for Utility Data Centers*. [S.l.], set. 2003. Disponível em: <www.hpl.hp.com/techreports/2003/HPL-2003-188.html>.

CANDEA, G.; FOX, A. Recursive restartability: Turning the reboot sledgehammer into a scalpel. In: *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*. [s.n.], 2001. p. 125–132. Disponível em: <citeseer.ist.psu.edu/candea01recursive.html>.

CANDEA, G.; FOX, A. Crash-only software. In: *Proceedings of HotOS'03: 9th Workshop on Hot Topics in Operating Systems*. [S.l.: s.n.], 2003. p. 67–72.

CANDEA, G. et al. Microreboot - a technique for cheap recovery. In: *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*. [S.l.: s.n.], 2004. p. 31–44.

CANDEA, G. et al. Jagr: An autonomous self-recovering application server. In: *5th International Workshop on Active Middleware Services*. [S.l.: s.n.], 2003.

CASTELLI, V. et al. Proactive management of software aging. *IBM Journal of Research and Development*, v. 45, n. 2, p. 311–332, 2001.

CHANDRA, A.; GONG, W.; SHENOY, P. Dynamic resource aladdress for shared data centers using online measurements. *SIGMETRICS Perform. Eval. Rev.*, ACM Press, v. 31, n. 1, p. 300–301, 2003. ISSN 0163-5999.

CHANDRA, A.; GONG, W.; SHENOY, P. J. Dynamic resource aladdress for shared data centers using online measurements. In: *IWQoS 2003: Proceeding of the Eleventh IEEE/ACM International Workshop on Quality of Service*. [S.l.]: Springer Verlag LNCS, 2003. p. 381–400.

CHASE, J. S. et al. Managing energy and server resources in hosting centres. In: *Symposium on Operating Systems Principles*. [s.n.], 2001. p. 103–116. Disponível em: <citeseer.ist.psu.edu/chase01managing.html>.

COCHRAN, W. G.; COX, G. M. *Experimental Designs*. 2. ed. [S.l.]: Wiley, 1957. ISBN 0471162035.

CONSTANTINESCU, C. et al. Dependability benchmarking of computing systems - panel statement. In: *DSN*. [S.l.: s.n.], 2005. p. 400.

CROVELLA, M. E.; BESTAVROS, A. Self-similarity in world wide web traffic: evidence and possible causes. In: *SIGMETRICS '96: Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. Philadelphia, Pennsylvania, EUA: ACM Press, 1996. p. 160–169. ISBN 0-89791-793-6.

CUNHA Ítalo S. et al. Self-adaptive capacity management for multi-tier virtualized environments. In: *Integrated Network Management*. IEEE, 2007. p. 129–138. Disponível em: <http://dblp.uni-trier.de/db/conf/im/im2007.html#CunhaAAS07>.

DEVORE, J. L. *Probability and Statistics for Engineering and the Sciences*. 5th. ed. [S.l.]: Duxbury, 2000.

DIACONESCU, A.; MOS, A.; MURPHY, J. Automatic performance management in component based software systems. In: *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*. Nova York, NY, EUA: IEEE Computer Society, 2004. p. 214–221.

DOHI, T.; GOSEVA-POPSTOJANOVA, K.; TRIVEDI, K. S. Analysis of software cost models with rejuvenation. In: *Proceedings of the 5th IEEE International Symposium on High Assurance Systems Engineering (HASE 2000)*. Albuquerque, New Mexico, United States: IEEE Computer Society, 2000.

DOHI, T. et al. Discrete availability models to rejuvenate a telecommunication billing application. In: *Proceedings of the 7th IEEE International Symposium on High-Assurance Systems Engineering (HASE 2002)*. [S.l.: s.n.], 2002. p. 159–166.

DOYLE, R. et al. Model-based resource provisioning in a web service utility. In: *Proceedings of the USENIX Symposium on Internet Technologies and Systems USITS 2003*. [S.l.: s.n.], 2003.

DRUSCHEL, P.; BANGA, G. Lazy receiver processing (lrp): a network subsystem architecture for server systems. In: *OSDI '96: Proceedings of the second USENIX symposium on Operating systems design and implementation*. New York, NY, USA: ACM Press, 1996. p. 261–275. ISBN 1-880446-82-0.

DURãES, J.; VIEIRA, M.; MADEIRA, H. Dependability benchmarking of web-servers. In: HEISEL, M.; LIGGESMEYER, P.; WITTMANN, S. (Ed.). *SAFECOMP*. Springer, 2004. (Lecture Notes in Computer Science, v. 3219), p. 297–310. ISBN 3-540-23176-5. Disponível em: <http://dblp.uni-trier.de/db/conf/safecomp/safecomp2004.html>.

EJASENT. *Utility Computing: Solutions for the Next Generation IT Infrastructure*. [S.l.], 2001. Disponível em: <www.ejasent.com/technology/whitepapers2.html>.

ERIKSEN, L. *Why Utility Computing Will Suceed Where ASPSs and Outsourcing Failed*. 2003. Disponível em: <http://www.utilitycomputing.com/news/355.asp (visitada em 24/06/2005)>.

FOX, A.; KICIMAN, E.; PATTERSON, D. Combining statistical monitoring and predictable recovery for self-management. In: *WOSS '04: Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*. New York, NY, USA: ACM Press, 2004. p. 49–53. ISBN 1-58113-989-6.

GANEK, A. G.; CORBI, T. A. The dawning of the autonomic computing era. *IBM Systems Journal*, IBM, v. 42, n. 1, p. 5–18, 2003.

GARG, S. et al. Minimizing completion time of a program by checkpointing and rejuvenation. In: *SIGMETRICS '96: Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. Philadelphia, Pennsylvania, United States: ACM Press, 1996. p. 252–261. ISBN 0-89791-793-6.

GARG, S. et al. Modeling and analysis of load and time dependent software rejuvenation policies. In: *Proceedings of th 3rd International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS3)*. Blomingdale, IL, EUA: [s.n.], 1996. p. 35 – 39.

GARG, S. et al. Analysis of software rejuvenation using markov regenerative stochastic petri net. In: *Int. Symposium on Software Reliability Engineering (ISSRE'95), Toulouse, France, October 24–27, 1995*. [S.l.: s.n.], 1995.

GARG, S. et al. On the analysis of software rejuvenation policies. In: *Proceedings of the 12th Annual Conference on Computer Assurance (COMPASS'97)*. Gaithersberg, MD: [s.n.], 1997.

GARG, S. et al. Analysis of preventive maintenance in transactions based software systems. *IEEE Trans. Comput.*, IEEE Computer Society, Washington, DC, EUA, v. 47, n. 1, p. 96–107, 1998. ISSN 0018-9340.

GENTLEMAN, R. et al. *R - The R Project for Statistical Computing*. 1997. Disponível em: <http://www.r-project.org/ (visitada em 28/09/2006)>.

GOLDSACK, P. et al. Smartfrog: Configuration and automatic ignition of distributed applications. In: *HPOVUA '03: Proceedings of the 2003 HP Openview University Association Conference*. [S.l.: s.n.], 2003.

GRAUPNER, S. et al. *Impact of virtualization on management systems*. [S.l.], 2003. Disponível em: <http://www.hpl.hp.com/techreports/2003/HPL-2003-125.pdf>.

GRAY, J. Why do computers stop and what can be done about it? In: *Symposium on Reliability in Distributed Software and Database Systems*. [S.l.: s.n.], 1986.

GRAY, J. Integration of emotion and cognitive control. *Current Directions in Psychological Science*, 2004.

GRIBBLE, S. D. Robustness in complex systems. In: *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*. [s.n.], 2001. p. 21–26. Disponível em: <citeseer.ist.psu.edu/gribble01robustness.html>.

GUANGZHI, Q. et al. Online monitoring and analysis for self-protection against network attacks. In: *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*. Nova York, NY, EUA: IEEE Computer Society, 2004. p. 324–325.

GUPTA, D.; GARDNER, R.; CHERKASOVA, L. *XenMon: QoS Monitoring and Performance Profiling Tool*. [S.l.], 2005. Disponível em: <http://www.hpl.hp.com/techreports/2005/HPL-2005-187.html>.

HARVERKORT, B. R. et al. (Ed.). *Performability Modelling: Techniques and Tools*. [S.l.]: John Wiley & Sons Ltd., 2001. ISBN 0-471-49195-0.

HELLERSTEIN, J. L. An approach to selecting metrics for detecting performance problems in information systems. *2nd IEEE International Workshop on Systems Management*, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, p. 30–39, 1996.

HELLERSTEIN, J. L.; KATIRCIOGLU, K.; SURENDRA, M. An on-line, business-oriented optimization of performance and availability for utility computing. *IEEE Journal*

on Selected Areas in Communications, IEEE Computer Society, Los Alamitos, CA, USA, v. 23, n. 10, p. 2013–2021, 2005. ISSN 0733-8716.

HEWLETT-PACKARD. *HP Global Workload Manager¿Improving server CPU utilization technical overview*. ago. 2005. Disponível em: <http://h71028.www7.hp.com/ERC/downloads/5983-0505EN.pdf (visitada em 24/06/2005)>.

HONG, Y. et al. Closed loop design for software rejuvenation. In: *Workshop on Self-Healing, Adaptive, and Self-Managed Systems*. [S.l.: s.n.], 2002.

HUANG, Y. et al. Software rejuvenation: Analysis, module and applications. In: *Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing*. [S.l.]: IEEE Computer Society, 1995. p. 381–390.

JAIN, R. K. *The Art of Computer Systems Performance Analysis*. [S.l.]: Wiley, 1991. ISBN 0471503363.

JIA, Y.-F.; CHEN, X.-E.; CAI, K.-Y. Chaotic analysis of software aging in web server. In: *SOSE '06: Proceedings of the Second IEEE International Symposium on Service-Oriented System Engineering (SOSE'06)*. Washington, DC, USA: IEEE Computer Society, 2006. p. 117–120. ISBN 0-7695-2726-4.

KANODIA, V.; KNIGHTLY, E. W. Ensuring latency targets in multiclass web servers. *IEEE Trans. Parallel Distrib. Syst.*, IEEE Press, Piscataway, NJ, USA, v. 14, n. 1, p. 84–93, 2003. ISSN 1045-9219.

KANT, K.; TEWARI, V.; IYER, R. Geist: A generator of e-commerce and internet server traffic. In: *Proceedings of the 2001 IEEE International Symposium on Performance Analysis of Systems and Software*. [S.l.]: IEEE Computer Society, 2001. p. 49–56.

KERALAPURA, R.; TAFT, N.; IANNACCONE, C.-N. C. G. Can isps take the heat from overlay networks? In: *ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*. [S.l.: s.n.], 2004.

LASSETTRE, E. et al. Dynamic surge protection: An approach to handling unexpected workload surges with resource actions that have dead times. In: *14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*. [S.l.]: Springer, 2003. (Lecture Notes in Computer Science, v. 2867), p. 82–92. ISBN 3-540-20314-1.

LI, L.; VAIDYANATHAN, K.; TRIVEDI, K. S. An approach for estimation of software aging in a web server. In: *International Symposium on Empirical Software Engineering*. [S.l.: s.n.], 2002.

LOPES, R.; CIRNE, W.; BRASILEIRO, F. Improving dynamic provisioning systems using software restarts. In: *15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'2004)*. Davis, CA, EUA: Springer, 2004. (Lecture Notes in Computer Science, v. 3278), p. 220 – 231.

LOPES, R. V. et al. Can dynamic provisioning and rejuvenation systems coexist in peace? In: *16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2005)*. Barcelona, Espanha: Springer, 2005. (Lecture Notes in Computer Science, v. 3775), p. 245–256.

LOU, T.; TANG, J. *Solving performance degradation problems in WebSphere applications*. 2007. Disponível em: <http://www.ibm.com/developerworks/websphere/library/techarticles/0706_lou/0706_lou.html (visitada em 04/09/2007)>.

MATIAS, R.; FILHO, P. J. F. An experimental study on software aging and rejuvenation in web servers. In: *COMPSAC '06: Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*. Washington, DC, USA: IEEE Computer Society, 2006. p. 189–196. ISBN 0-7695-2655-1.

MENASCÉ, D. et al. Generating representative web workloads for network and server performance evaluation. In: *II ACM Conference on Electronic Commerce*. Minneapolis, MN, EUA: [s.n.], 2000.

MENASCE, D. A.; ALMEIDA, V. A. F.; DOWDY, L. W. *Performance by Design: Computer Capacity Planning by Example*. Upper Saddle River, NJ, EUA: Prentice Hall PTR, 2004. ISBN 0-13-090673-5.

MENASCÉ, D. A. et al. A hierarchical and multiscale approach to analyze e-business workloads. *Performance Evaluation*, Elsevier Science Publishers B. V., v. 54, n. 1, p. 33–57, 2003. ISSN 0166-5316.

MENASCE, D. A.; VIRGILIO, A. F. A. *Scaling for E Business: Technologies, Models, Performance, and Capacity Planning*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000. ISBN 0130863289.

MUKHOPADHYAY, S. Distributed control and distributed computing. *SIGAPP Appl. Comput. Rev.*, ACM Press, v. 7, n. 1, p. 23–24, 1999.

MUKHOPADHYAY, S.; NARENDRA, K. S. Decentralized adaptive control using partial information. In: *American Control Conference*. [S.l.: s.n.], 1999. v. 1, p. 34 – 38.

NAKADAI, S.; TANIGUCHI, K. Server capacity planning with priority allocation for service level management in heterogeneous server clusters. In: *Integrated Network Management*. IEEE, 2007. p. 753–756. Disponível em: <http://dblp.uni-trier.de/db/conf/im/im2007.html#NakadaiT07>.

NARASIMHAN, P. Trade-offs between real-time and fault tolerance for middleware applications. In: *Workshop on Foundations of Middleware Technologies*. Irvine, CA, EUA: [s.n.], 2002.

NARENDRA, K. S.; OLENG, N. O. Decentralized adaptive control. In: *American Control Conference*. [S.l.: s.n.], 2002. v. 5, p. 3407 – 3412.

OKAMURA, H.; MIYAHARA, S.; DOHI, T. Dependability analysis of a client/server software system with rejuvenation. In: *ISSRE*. [S.l.: s.n.], 2002. p. 171–182.

OPPENHEIMER, D.; GANAPATHI, A.; PATTERSON, D. A. Why do internet services fail, and what can be done about it? In: *USITS '03: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*. Seattle, WA, EUA: [s.n.], 2003.

PFENING, A. et al. Optimal software rejuvenation for tolerating soft failures. *Perform. Eval.*, v. 27/28, n. 4, p. 491–506, 1996.

RANJAN, S. et al. Qos-driven server migration for internet data centers. In: *Proceedings of the International Workshop on Quality of Service*. Miami, Florida, EUA: [s.n.], 2002. p. 3 – 12.

RAPPA, M. A. The utility business model and the future of computing services. *IBM Systems Journal*, IBM, v. 43, n. 1, p. 32–42, 2004.

ROLIA, J. et al. Statistical service assurancecs for applications in utility grid environments. In: *Proceedings of the Tenth IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telcommunication Systems*. Forth Worth, Texas: [s.n.], 2003. p. 247–256.

ROLIA, J. et al. Grids for enterprise applications. In: *9th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2003)*. Seattle, WA, EUA: Springer, 2003. (Lecture Notes in Computer Science, v. 2862), p. 129–147. ISBN 3-540-20405-9.

ROLIA, J.; SINGHAL, S.; FRIEDRICH, R. Adaptive internet data centers. In: *In SSGRR'00 Conference*. L'Aquila, Italy: [s.n.], 2000. Disponível em: <citeseer.ist.psu.edu/rolia00adaptive.html>.

ROLIA, J.; ZHU, X.; ARLITT, M. F. Resource access management for a utility hosting enterprise applications. In: *Proceeding of the 2003 International Symposium on Integrgated Management*. [S.l.: s.n.], 2003. p. 549–562.

SAUVÉ, J. P. et al. Business-driven decision support for change management: Planning and scheduling of changes. In: *DSOM*. [S.l.: s.n.], 2006. p. 173–184.

SILVA, L. M. et al. Using virtualization to improve software rejuvenation. In: *NCA*. IEEE Computer Society, 2007. p. 33–44. Disponível em: <http://dblp.uni-trier.de/db/conf/nca/nca2007.html#SilvaASTA07>.

SOUNDARARAJAN, G.; AMZA, C. Reactive provisioning of backend databases in shared dynamic content server clusters. *ACM Trans. Auton. Adapt. Syst.*, ACM Press, New York, NY, USA, v. 1, n. 2, p. 151–188, 2006. ISSN 1556-4665.

STANKOVIC, J. A.; WANG, F. *The integration of scheduling and fault tolerance in real-time systems*. [S.l.], 1992.

TAI, A. T. et al. A performability-oriented software rejuvenation framework for distributed applications. In: *Proceedings of the 2005 International Conference on Dependable Systems and Networks*. Yokohama, Japão: IEEE Computer Society, 2005. ISBN 0769522823.

TATON, C. et al. Self-sizing of clustered databases. In: *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*. Washington, DC, USA: IEEE Computer Society, 2006. p. 506–512. ISBN 0-7695-2593-8.

TRIVEDI, K. S.; VAIDYANATHAN, K.; GOSEVA-POPSTOJANOVA, K. Modeling and analysis of software aging and rejuvenation. In: *Proceedings of the 33rd Annual Simulation Symposium (SS 2000)*. [S.l.: s.n.], 2000. p. 270 – 279.

URGAONKAR, B.; SHENOY, P. Cataclysm: policing extreme overloads in internet applications. In: *WWW '05: Proceedings of the 14th international conference on World Wide Web*. Chiba, Japan: ACM Press, 2005. p. 740–749. ISBN 1-59593-046-9.

URGAONKAR, B. et al. Dynamic provisioning for multi-tier internet applications. In: *ICAC '05: Proceedings of the Second International Conference on Autonomic Computing*. Seattle, WA, EUA: IEEE Computer Society, 2005.

VAIDYANATHAN, K. et al. Analysis and implementation of software rejuvenation in cluster systems. In: *SIGMETRICS/Performance*. [S.l.: s.n.], 2001. p. 62–71.

VAIDYANATHAN, K.; TRIVEDI, K. S. A measurement-based model for estimation of resource exhaustion in operational software systems. In: *ISSRE '99: Proceedings of the 10th International Symposium on Software Reliability Engineering*. Washington, DC, EUA: IEEE Computer Society, 1999. p. 84. ISBN 0-7695-0443-4.

VAIDYANATHAN, K.; TRIVEDI, K. S. Extended classification of software faults based on aging. In: *Proceedings of the 12th International Symposium on Software Reliability Engineering*. [S.l.: s.n.], 2001.

VAIDYANATHAN, K.; TRIVEDI, K. S. A comprehensive model for software rejuvenation. *IEEE Transactions on Dependable and Secure Computing*, v. 2, n. 2, p. 124–137, 2005.

WANG, Y. et al. Solving the group priority inversion problem in a timed asynchronous system. *IEEE Trans. Computers*, v. 51, n. 8, p. 900–915, 2002.

WILKES, J.; MOGUL, J.; SUERMONDT, J. Utilification. In: *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC*. New York, NY, USA: ACM Press, 2004. p. 13.

XIE, W.; HONG, Y.; TRIVEDI, K. S. Software rejuvenation policies for cluster systems under varying workload. In: *PRDC*. [S.l.: s.n.], 2004. p. 122–129.

ZHANG, F.; HELLERSTEIN, J. L. An approach to on-line predictive detection. In: *MASCOTS '00: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. Washington, DC, USA: IEEE Computer Society, 2000. p. 549. ISBN 0-7695-0728-X.

Apêndice A

Projeto de experimentos

No Capítulo 4 foram apresentados diversos componentes que podem ser combinados para formar diferentes composições, conforme também sugerido no Capítulo 1. Cada um destes componentes está associado a um conjunto de fatores, que são variáveis independentes de configuração do componente em questão.

Nesta seção são apresentados conceitos básicos de projeto de experimentos e resultados obtidos ao aplicar uma técnica de projeto de experimento nos modelos a serem estudados nesta tese.

Um projeto de experimentos tem por objetivo projetar os experimentos de tal forma que se obtenha a maior quantidade de informações com a menor quantidade de experimentos possível. Experimentos, neste contexto, podem ser simulações ou medições em ambientes reais (ou próximos aos reais).

Pode-se realizar projeto de experimentos para diversas finalidades, dentre elas: decidir entre alternativas (experimentos comparativos), identificar que fatores influenciam mais uma variável de resposta (experimentos seletivos), ajustar/otimizar o processo experimental, etc.

No contexto deste trabalho o projeto experimental foi realizado para decidir a respeito dos fatores a serem variados. Espera-se reduzir a quantidade de tais fatores, reduzindo conseqüentemente o número de experimentos a serem realizados. Além disso, como será visto no decorrer desta seção, o projeto experimental serviu para trazer mais conhecimento sobre o sistema estudado.

A.0.1 Terminologia

Para entender o projeto experimental realizado é necessário introduzir alguns termos importantes. Para tal, será usado um exemplo de uma aplicação sem falhas (*AWoF*) cujos recursos são gerenciados por um sistema de provisão dinâmica (*DPS*). Esta composição chama-se *AWoF+DPS*, conforme explicado anteriormente.

A aplicação executa sobre servidores de um centro de dados, que apresentam uma certa capacidade de processamento e recebe um fluxo de requisições de clientes que pode variar ao longo do tempo em intensidade. O *DPS* tenta manter a utilização dos recursos em torno de um alvo. Além disso, quando um recurso é selecionado para executar a aplicação, ele precisa ser preparado para tal em um certo intervalo de tempo (tempo de migração). À luz deste exemplo (composição *AWoF+DPS*) são apresentados a seguir termos geralmente usados no projeto e análise de experimentos (JAIN, 1991):

- **Variável de resposta (y).** A variável de resposta (ou variável dependente) é o resultado do experimento, isto é, aquilo que se deseja medir. No exemplo do experimento *AWoF+DPS*, a disponibilidade da aplicação poderia ser a variável de resposta considerada;

- **Fatores.** Um fator é uma variável independente (que pode ser controlada) que pode assumir diferentes valores e que afeta a variável de resposta. Através de uma análise de experimentos é possível quantificar o efeito dos diversos fatores sobre a variável de resposta. Por exemplo, a utilização alvo (ρ_{target}) do *DPS* poderia ser considerado um fator. Este fator poderia assumir teoricamente qualquer valor no intervalo [0,1]. Às diferentes alternativas de valor que um fator assume chama-se nível. Na prática, poder-se-ia analisar um experimento *AWoF+DPS* em que o fator utilização alvo do *DPS* assume três níveis diferentes: 40%, 65% e 90%. É comum identificar o número de fatores pela letra k;

- **Replicação.** É possível repetir o mesmo experimento r vezes. A esta repetição dá-se o nome de replicação;

- **Interação.** Dois fatores interagem se o efeito verificado para um fator sobre a variável de resposta depende do nível do outro fator. Sejam A e B fatores e y a variável de resposta. A Figura A.1 ilustra duas situações distintas. Na Figura A.1(a) não há interação entre os fatores A e B. Nota-se que, independentemente do nível do fator B, a variável de resposta decresce o mesmo valor quando o fator A passa do nível -1 para o nível 1. Já na Figura A.1(b) observa-se que o decréscimo em y quando o fator

A passa do nível -1 para o nível 1 é muito maior quando o fator B está no nível 1. Este comportamento exemplifica uma interação entre os fatores A e B.

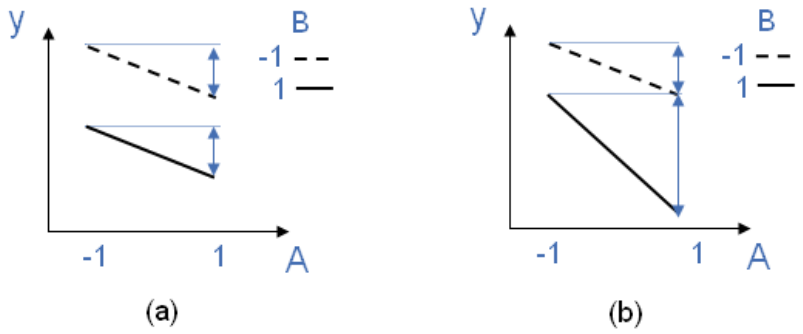

Figura A.1: Exemplo de interação e não interação entre fatores

Uma análise adequada de experimentos permite quantificar o efeito dos fatores na variável de resposta, e quando associada a um estudo de significância, permite identificar quais fatores são estatisticamente significativos para a variável de resposta. Em alguns casos, os efeitos dos fatores na variável de resposta são devidos a variações randômicas causadas por erros de medição e outros fatores que não estão sendo controlados.

Os três tipos de projetos de experimentos mais usados são (JAIN, 1991):

1. **Projeto simples.** Inicia-se com uma certa configuração para os parâmetros e varia-se um fator por vez para identificar o efeito que os fatores exercem sobre a variável de resposta. Este método não leva em consideração as interações entre fatores, podendo levar a resultados errôneos quando existem interações;

2. **Projeto fatorial completo.** Realiza experimentos para cada um dos níveis de todos os fatores. Assim, para realizar um estudo de desempenho com k fatores, com o iésimo fator apresentando n_i níveis, seriam necessários $n = \prod_{i=1}^{k} n_i$ experimentos. Caso haja replicação, r réplicas de cada experimento são executadas, então o número de experimentos cresce para $n \times r$. Este tipo de projeto de experimentos é o mais completo, no entanto, quando o número de fatores e/ou níveis é muito grande, torna-se custoso demais;

3. **Projeto fatorial fracionado.** Este tipo de projeto é indicado quando o número de experimentos a ser realizado com projeto fatorial completo é muito grande. Usa-se aqui apenas uma fração de todas as possibilidades. O número de experimentos é menor, mas não é possível estudar todas as possíveis interações entre os fatores.

A quantidade de fatores identificado para o estudo das composições apresentadas no Capítulo 3 é grande e um estudo completo não seria indicado. Apesar disto, resolveu-se utilizar o projeto fatorial completo e reduzir o nível de todos os fatores para dois, uma vez que é este o projeto experimental mais completo. Como os experimentos a serem realizados são de simulação, então não há custos tão elevados ao se escolher tal projeto. Além disso, a replicação foi realizada quando necessário para que fosse possível identificar variações em y devido a erros experimentais. Este tipo de projeto de experimento é conhecido como projeto fatorial completo com replicação ($2^k r$).

No projeto fatorial $2^k r$ cada um dos 2^k tratamentos de um projeto fatorial completo 2^k são repetidos r vezes. Desta forma, são realizados $2^k \times r$ experimentos. O modelo de regressão linear múltipla é considerado e a partir dele é possível encontrar o efeito dos fatores sobre a variável de resposta y. Em outras palavras, o projeto experimental oferece subsídios para identificar que fatores influenciam mais a variável de resposta y. Como há replicação, é possível também identificar a variação de y devido a erros experimentais. Mais detalhes de como o projeto fatorial completo é realizado podem ser encontrados na literatura, em especial no livro de Jain (1991).

Apesar dos percentuais de variação explicados pelos fatores darem uma boa idéia da influência que estes têm sobre a variável de resposta, é possível que um fator j que está associado a um baixo percentual seja de fato significativo estatisticamente para a variável de resposta. A análise de variância (ANOVA) é usada para identificar que fatores são estatisticamente significantes para a variação de y.

Esta análise serve para identificar formalmente se existe diferença entre médias. A hipótese nula H_0 é que independente do nível do fator j, todas as médias são iguais. A análise de variância vai verificar se a hipótese nula é ou não verdadeira. São realizados 2^k testes para se identificar se os fatores e as interações são ou não significantes.

Explicações detalhadas de como a análise de variância é realizada podem ser encontradas em bibliografia específica da área (DEVORE, 2000; JAIN, 1991).

Oito projetos experimentais foram realizados: dois para a composição *AWoF+DPS*, três para a composição *AWF+RS*, dois para a composição *AWF+DPS+RS* e um para a composição *AWF+PDPS*RS*. Observou-se que os resultados de diferentes projetos para a mesma composição ofereceu resultados semelhantes.

O que diferenciou um projeto de outro para uma mesma composição foi a intensidade da carga de trabalho aplicada. Quando o *PDPS* foi usado, a composição *AWoF+PDPS* não foi estudada porque sempre retorna 100% de disponibilidade, não sendo possível identificar que fatores contribuem mais ou menos para esta métrica.

Quando o *DPS* não perfeito foi usado as composições *AWoF+DPS*, *AWF+RS* e *AWF+DPS+RS* foram estudadas. Neste caso, alguns projetos consideraram cargas de trabalho de menor intensidade e outros cargas de maior intensidade.

Nas seções que seguem são apresentados os resultados de tais projetos.

A.0.2 Projeto de experimentos para a composição *AWoF+DPS*

Projeto da composição *AWoF+DPS* considerando carga de menor intensidade

A composição $AWoF + DPS$ modela uma aplicação livre de falhas cujos recursos são geridos por um sistema de provisão dinâmica. Os resultados obtidos nesta seção e nas seções a seguir foram obtidos com o auxílio da ferramenta R (GENTLEMAN et al., 1997).

Na Tabela A.1 encontra-se a descrição dos fatores que estão associados a este primeiro projeto experimental.

Tabela A.1: Fatores aplicados nos experimentos *AWoF+DPS*

Fator	*Descrição*	*Valor no nível -1*	*Valor no nível 1*
A	Variabilidade da carga submetida à aplicação	Ver Figura A.2	Ver Figura A.3
B	Capacidade de processamento dos nós que executam a aplicação	15 rps	45 rps
C	Utilização alvo perseguida pelo *DPS*	50%	70%
D	Tempo médio de migração dos nós	1 min.	2 min.
E	Tempo de adaptação do *DPS*	2,5 min.	5 min.

Na Tabela A.2 encontram-se os percentuais de variação da disponibilidade da aplicação explicado por alguns fatores e interações. Nesta tabela, optou-se por apresentar apenas os percentuais dos fatores principais e das interações maiores que 1%. O percentual de variação devido a erros experimentais (que não pode ser explicado pelos fatores e interações) foi de 0,05% e o coeficiente de determinação R^2 calculado foi 99,95%.

Para estes experimentos, os fatores que demonstraram ser mais importantes para a disponibilidade da aplicação foram a variabilidade da carga de trabalho e a utilização alvo do *DPS*.

De forma geral, os seguintes resultados foram encontrados:

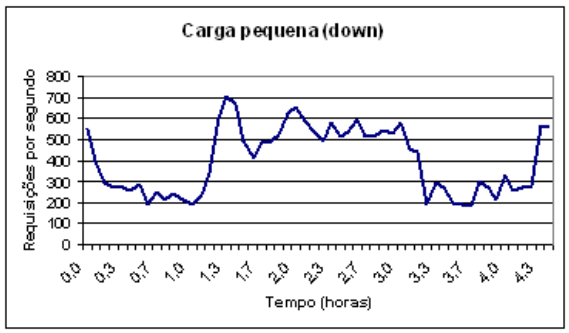

Figura A.2: Carga de trabalho menos variável e intensa, com um pico completo (*down*)

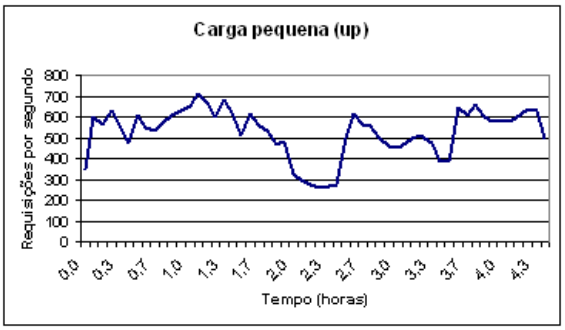

Figura A.3: Carga de trabalho mais variável e intensa, com dois picos completos (*up*)

Tabela A.2: Percentuais de variação de y devido aos fatores

Carga de trabalho ($A_{\%}$)	11,75%
Capacidade dos nós ($B_{\%}$)	2,52%
Utilização alvo ($C_{\%}$)	55,06%
Tempo médio de migração ($D_{\%}$)	2,03%
Tempo de adaptação do *DPS* ($E_{\%}$)	6,56%

1.Quanto mais variável e intensa é a carga, pior é a disponibilidade da aplicação;

2.Quanto maior a capacidade dos nós, melhor é a disponibilidade da aplicação;

3.Quanto maior é a utilização, pior é a disponibilidade da aplicação;

4.Quanto maior é o tempo de migração, pior é a disponibilidade da aplicação;

5.Quanto maior é o tempo de adaptação, pior é a disponibilidade da aplicação.

Com o objetivo de analisar mais formalmente os resultados resolveu-se realizar uma análise de variância (ANOVA).

Na Figura A.4 encontram-se um gráfico quantil-quantil dos resíduos e um gráfico dos resíduos versus resposta esperada.

Em relação à independência dos resíduos observa-se que a grande maioria dos resíduos acumula-se em torno de zero para valores maiores de y. Segundo Jain (JAIN, 1991), esta tendência pode ser desconsiderada, uma vez que o valor dos resíduos é mais de uma ordem de magnitude menor que os valores das respostas esperadas.

Observa-se que o gráfico quantil-quantil não resultou em uma reta perfeita. O histograma dos resísuos assemelha-se ao de uma distribuição não normal de cauda longa. Segundo Cochran e Cox (COCHRAN; COX, 1957), quando as hipóteses assumidas pela ANOVA

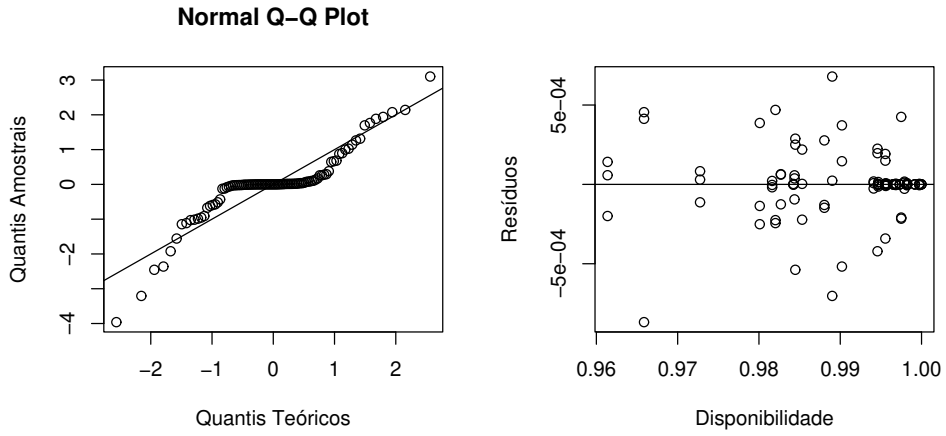

Figura A.4: Testes gráficos para realização da ANOVA

são violadas, a regra geral é que o nível de significância encontrado deve ser na realidade maior. Se, por exemplo, testa-se os dados em um nível de 5% de significância, então, de fato, o nível testado é maior que 5%. Em muitos casos as violações não são tão graves a ponto de invalidar a análise de variância.

Segundo a análise de variância realizada, todos os fatores e todas as interações de segundo grau são consideradas significativas, com nível de significância 0,1%. Como as hipóteses da ANOVA não foram completamente satisfeitas, o nível de significância de fato pode ser maior que 0,1%.

Projeto da composição *AWoF+DPS* considerando carga de maior intensidade

Na Tabela A.3 encontra-se a descrição dos fatores que estão associados a este projeto experimental.

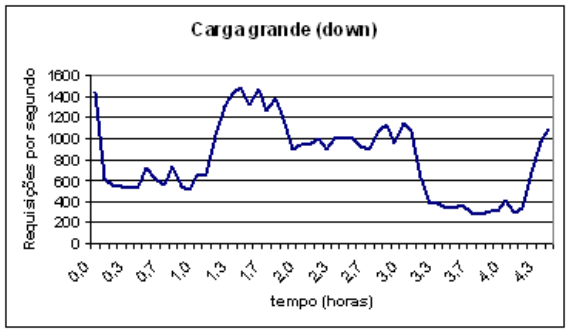

Figura A.5: Carga de trabalho menos variável e intensa, com um pico completo (*down*)

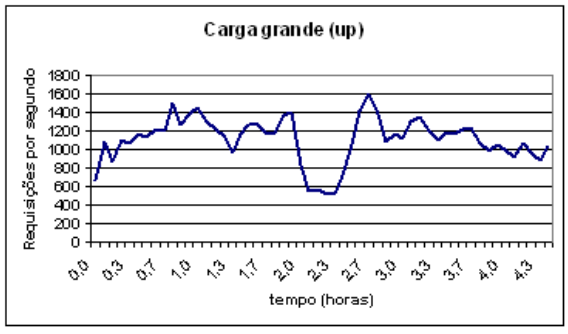

Figura A.6: Carga de trabalho mais variável e intensa, com dois picos completos (*up*)

Tabela A.3: Fatores aplicados nos experimentos *AWoF+DPS*

Fator	*Descrição*	*Valor no nível -1*	*Valor no nível 1*
A	Variabilidade da carga submetida à aplicação	Ver Figura A.5	Ver Figura A.6
B	Capacidade de processamento dos nós que executam a aplicação	15 rps	45 rps
C	Utilização alvo perseguida pelo *DPS*	50%	70%
D	Tempo médio de migração dos nós	1 min.	2 min.
E	Tempo de adaptação do *DPS*	2,5 min.	5 min.

Na Tabela A.4 encontram-se os percentuais de variação da disponibilidade da aplicação explicado por alguns fatores e interações. Mais uma vez, optou-se por apresentar apenas os percentuais dos fatores principais e das interações maiores que 1%. O percentual de variação devido a erros experimentais foi de 0,123% e o coeficiente de determinação R^2 calculado foi 99,877%.

Tabela A.4: Alguns percentuais de variação de y devido aos fatores e interações

Carga de trabalho ($A_{\%}$)	11,37%
Capacidade dos nós ($B_{\%}$)	0,53%
Utilização alvo ($C_{\%}$)	58,20%
Tempo médio de migração ($D_{\%}$)	2,55%
Tempo de adaptação do *DPS* ($E_{\%}$)	9,12%

Para estes experimentos, os fatores que demonstraram ser mais importantes para a disponibilidade da aplicação também foram a variabilidade da carga de trabalho e a utilização alvo do *DPS*.

De forma geral, os seguintes resultados foram encontrados:

1.Quanto mais variável e intensa é a carga, melhor é a disponibilidade da aplicação;

2.Quanto maior a capacidade dos nós, pior é a disponibilidade da aplicação;

3.Quanto maior é a utilização, pior é a disponibilidade da aplicação;

4.Quanto maior é o tempo de migração, pior é a disponibilidade da aplicação;

5.Quanto maior é o tempo de adaptação, pior é a disponibilidade da aplicação.

Os dois primeiros itens citados são exatamente opostos aos dois primeiros itens citados para o primeiro projeto experimental realizado. Isto sugere que a eficiência do *DPS* está intimamente ligada a detalhes da carga de trabalho e capacidade dos nós, de forma que não é possível afirmar, de forma geral que há uma tendência de comportamento da aplicação dependendo da carga e da capacidade dos nós.

Já as observações feitas nos três últimos itens são idênticas às realizadas no primeiro projeto experimental. Sendo assim, há uma evidência de que estas tendências sejam gerais, para quaisquer cenários *AWoF+DPS* que possam ser estudados.

Com o objetivo de analisar mais formalmente os resultados resolveu-se realizar uma análise de variância (ANOVA).

Um gráfico quantil-quantil dos resíduos e um gráfico dos resíduos versus resposta esperada são apresentados na Figura A.7.

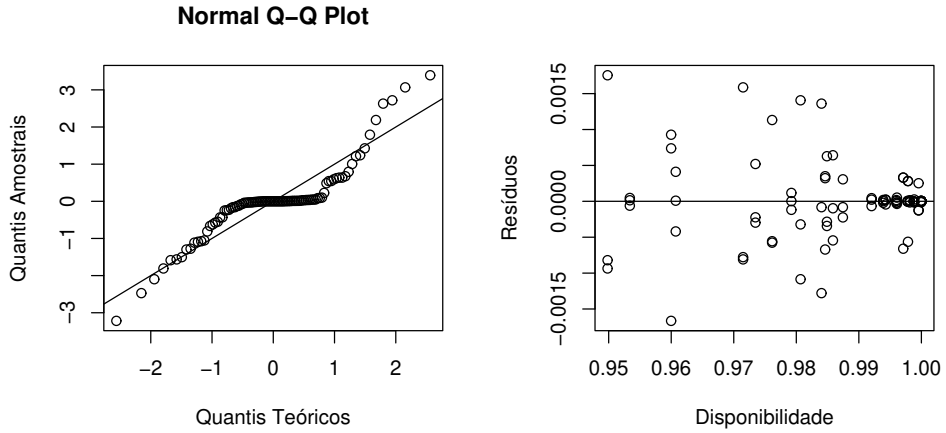

Figura A.7: Testes gráficos para realização da ANOVA

Os gráficos de teste da ANOVA são bastante semelhantes aos encontrados no primeiro projeto realizado, valendo aqui a mesma discussão anterior.

Segundo a análise de variância realizado, todos os fatores foram significativos em um nível de significância de 0,1%.

Conclusões parciais do projeto experimental de composições $AWoF + DPS$

As seguintes tendências foram observadas em ambos os projetos experimentais realizados para a composição *AWoF+DPS*:

•Quanto menor é a utilização alvo, melhor é a disponibilidade da aplicação;

•Quanto menor é o tempo de migração, melhor é a disponibilidade da aplicação;

•Quanto menor é o tempo de adaptação, melhor é a disponibilidade da aplicação;

Como já discutido anteriormente, não se pode tirar conclusões gerais sobre a influência da carga de trabalho e da capacidade dos nós.

Conclui-se, portanto, que nas composições em que o *DPS* está presente é interessante variar a *capacidade dos nós* e a *carga de trabalho*, uma vez que estes fatores têm influência sobre a variável de resposta, mas não apresentam tendências evidentes. Além disso, como o percentual de variação de y devido à *utilização alvo* foi bastante elevado em ambos os experimentos, é interessante também variar este fator.

Projeto da composição *AWF+RS* considerando carga de menor intensidade

Nesta seção serão apresentados os resultados do projeto de experimentos para os experimentos *AWF+RS*. Esta composição modela uma aplicação com falhas, super-provida de recursos, gerenciada por um sistema de rejuvenescimento *RS*.

As composições *AWF+RS* não levaram a uma variação de disponibilidade entre réplicas de um mesmo tratamento. A única variável aleatória do experimento é o tempo de migração e rejuvenescimento dos nós, que segue uma distribuição normal. O tempo de migração não importa nestes experimentos, uma vez que todos os nós são ativados no início e desativados apenas depois que toda a carga de trabalho é realizada. A variação do tempo de rejuvenescimento de uma réplica para outra não é suficiente para mudar a disponibilidade, uma vez que os nós ativos durante o rejuvenescimento são capazes de atender toda a demanda da aplicação. Há uma variação nos tempos de resposta, mas esta variável não foi analisada aqui.

Por esta razão realizou-se para esta composição o projeto experimental fatorial completo sem replicação.

Mais uma vez, a variável de resposta considerada é a disponibilidade da aplicação. Na Tabela A.5 encontra-se a descrição dos fatores que estão associados a este experimento.

Os limiares inferior e superior de disponibilidade e tempo de resposta respectivamente considerados pelo *RS* foram, 99.99% e 2 segundos. O limiar inferior da disponibilidade foi estipulado por ser adequado para muitos *e-services* (MENASCE; ALMEIDA; DOWDY, 2004), como por exemplo, lojas virtuais de comércio eletrônico. O limiar de tempo de resposta

Tabela A.5: Fatores aplicados nos experimentos *AWF+RS*

Fator	*Descrição*	*Valor no nível -1*	*Valor no nível 1*
A	Variabilidade da carga submetida à aplicação	Ver Figura A.2	Ver Figura A.3
B	Capacidade de processamento dos nós que executam a aplicação	15 rps	45 rps
C	Número de nós ativos durante experimento	ver linha 1 da Tabela A.6	ver linha 2 da Tabela A.6
D	Intensidade das falhas	100 ms.	300 ms
E	Número de nós em falha em $t = 0$	1	2
F	Tempo mínimo de rejuvenescimento de um nó	1 min.	2 min.

Tabela A.6: Número estático de nós ativos

	15 rps		*45 rps*	
	Carga down	*Carga up*	*Carga down*	*Carga up*
Nível -1	78	95	25	31
Nível 1	88	105	30	36

foi bastante inferior ao limiar considerado na prática, conhecido como "regra dos 8 segundos" (MENASCE; VIRGILIO, 2000), pois a simulação não leva em consideração os tempos de transmissão das mensagens através da rede e outros custos.

Na Tabela A.7 encontram-se os percentuais de variação da disponibilidade da aplicação explicado pelos fatores.

Tabela A.7: Alguns percentuais de variação de y devido aos fatores

Carga de trabalho ($A_{\%}$)	5,02%
Capacidade dos nós ($B_{\%}$)	43,71%
Número de nós ativos ($C_{\%}$)	0,51%
Intensidade das falhas ($D_{\%}$)	3,35%
Número de nós em falha ($E_{\%}$)	10,47%
Tempo de reinício ($F_{\%}$)	$1,57 \times 10^{-10}$%
Tempo de adaptação do RS ($G_{\%}$)	5,78%

Para estes experimentos, observou-se que a capacidade dos nós e a quantidade de nós em falha são os fatores mais importantes para a disponibilidade da aplicação.

Analisando os resultados, conclui-se, para este projeto experimental que:

1. Quanto mais variável e intensa foi a carga, pior foi a disponibilidade da aplicação. De fato, no início do experimento existe um ou dois nós em falha. Como a carga *up* inicia crescente, então as falhas ocorrem em momentos em que a carga está mais elevada, piorando a disponibilidade, e o rejuvenescimento ocorre em um momento de pico. Já na carga *down* observa-se que ela inicia alta, mas logo decresce e conseqüentemente as falhas têm uma conseqüência menos danosa na disponibilidade da aplicação. Rejuvenescer os nós nos períodos de menor carga faz todo o sentido em um ambiente de super-provisão estática, como sugerido em alguns artigos de rejuvenescimento;

2. Quanto maior a capacidade dos nós, pior foi a disponibilidade da aplicação. Quando nós mais poderosos falham, isto causa um impacto maior na qualidade de serviço da aplicação por duas razões. A primeira é que com a falha de um nó de alta capacidade, uma porção maior da aplicação fica degradada, se comparada à falha de um nó menos poderoso. A segunda é que quando os nós são mais poderosos há a tendência de haver uma quantidade menor de nós, assim, a falha de um nó equivaleria à falha de mais de um nó em um cenário em que os nós são menos poderosos;

3. Quanto maior o número de nós ativos, maior é a disponibilidade da aplicação. Quando nós estão em falha, eles receberão menos requisições, já que elas serão distribuídas

igualmente entre todos os nós. Além disso, quando nós precisam ser rejuvenescidos, quanto mais capacidade extra, melhor;

4. Quanto maior a intensidade das falhas, pior a disponibilidade da aplicação;

5. Quanto maior o número de nós em falha, pior é a disponibilidade da aplicação;

6. O tempo de reinício do nó não influenciou a disponibilidade da aplicação;

7. Quanto maior o tempo de reconfiguração, pior é a disponibilidade da aplicação. Este resultado já era esperado, pois quanto maior o tempo de reconfiguração, maior o tempo em que os nós permanecem em falha antes de ser rejuvenescidos.

Projeto da composição *AWF+RS* considerando carga de maior intensidade

Mais uma vez, a variável de resposta considerada é a disponibilidade da aplicação. Na Tabela A.8 encontra-se a descrição dos fatores que estão associados a este experimento.

Tabela A.8: Fatores aplicados nos experimentos *AWF+RS*

Fator	*Descrição*	*Valor no nível -1*	*Valor no nível 1*
A	Variabilidade da carga submetida à aplicação	Ver Figura A.5	Ver Figura A.6
B	Capacidade de processamento dos nós que executam a aplicação	15 rps	45 rps
C	Número de nós ativos durante experimento	ver linha 2 da Tabela A.9	ver linha 2 da Tabela A.9
D	Intensidade das falhas	300 ms.	500 ms
E	Número de nós em falha em $t = 0$	1	2
F	Tempo mínimo de rejuvenescimento de um nó	1 min.	2 min.

Tabela A.9: Número estático de nós ativos

	15 rps		*45 rps*	
	Carga down	*Carga up*	*Carga down*	*Carga up*
Nível -1	166	200	55	55
Nível 1	176	210	60	60

Na Tabela A.10 encontram-se os percentuais de variação da disponibilidade da aplicação explicado pelos fatores considerados.

Tabela A.10: Alguns percentuais de variação de y devido aos fatores e interações

Carga de trabalho ($A_{\%}$)	2,12%
Capacidade dos nós ($B_{\%}$)	52,91%
Número de nós ativos ($C_{\%}$)	0,052%
Intensidade das falhas ($D_{\%}$)	1,65%
Número de nós em falha ($E_{\%}$)	14,01%
Tempo de reinício ($F_{\%}$)	$2,8 \times 10^{-26}$%
Tempo de adaptação do *RS* ($G_{\%}$)	12,40%

Os resultados em termos de percentuais de variação são semelhantes aos obtidos para a carga menor. Além disso, todas as tendências observadas naquela ocasião são repetidas aqui.

Foi realizado ainda um projeto experimental para a composição *AWF+RS* considerando uma carga de intensidade ainda menor que a apresentada na seção anterior. As mesmas tendências foram observadas e por esta razão os resultados de tal projeto foram omitidos deste texto.

Conclusões parciais do projeto experimental de composições *AWF+RS*

Conclui-se que nas composições em que o *RS* está presente é interessante variar a *capacidade de processamento* dos nós que executam a aplicação e o *número de nós em falha* pois, segundo o estudo experimental, estes fatores exercem grande influência sobre a disponibilidade da aplicação. Além disso, é interessante variar a *carga de trabalho* de forma que rejuvenescimentos e falhas ocorram em momentos de carga diferentes, conforme ocorreu nestes testes.

A.0.3 Projeto de experimento da composição *AWF+DPS+RS* considerando cargas de menor intensidade

Esta composição modela uma aplicação com falhas, gerenciada por um sistema de provisão dinâmica e um sistema de rejuvenescimento simultaneamente. Já foram estudadas as composições em que o *DPS* e o *RS* atuavam sozinhos. Com o estudo apresentado nesta seção é possível identificar se, ao atuarem simultaneamente, os fatores que se mostraram importantes nas seções anteriores mantêm-se importantes.

Mais uma vez, a variável de resposta considerada é a disponibilidade da aplicação. Na Tabela A.11 encontra-se a descrição dos fatores que estão associados a este experimento.

Tabela A.11: Fatores aplicados nos experimentos *AWF+DPS+RS*

Fator	*Descrição*	*Valor no nível -1*	*Valor no nível 1*
A	Variabilidade da carga submetida à aplicação	Ver Figura A.2	Ver Figura A.3
B	Capacidade de processamento dos nós que executam a aplicação	15 rps	45 rps
C	Número inicial de nós em falha	1	2
D	Tempo médio de migração de um nó	1 min.	2 min.
E	Utilização alvo perseguida pelo *DPS*	50%	70%
F	Intensidade das falhas	100 ms.	300 ms
G	Tempo mínimo de rejuvenescimento de um nó	1 min.	2 min.
H	Tempo de adaptação do *DPS*	2,5 min.	5 min.

Na Tabela A.12 encontram-se os percentuais de variação da disponibilidade da aplicação explicados pelos fatores. O percentual de variação devido a erros experimentais foi de 0,268%.

Tabela A.12: Alguns percentuais de variação de y devido aos fatores

Variabilidade da carga ($A_{\%}$)	15,78%
Capacidade dos nós ($B_{\%}$)	0,20%
Número de nós em falha ($C_{\%}$)	0,37%
Tempo de migração ($D_{\%}$)	1,35%
Utilização alvo do *DPS* ($E_{\%}$)	53,26%
Intensidade das falhas ($F_{\%}$)	0,06%
Tempo de reinício ($G_{\%}$)	$2,92 \times 10^{-4}$%
Tempo de adaptação do *DPS* ($H_{\%}$)	9,68%

Para estes experimentos, observou-se que a variabilidade da carga, a utilização alvo e o período de adaptação do *DPS* são os fatores mais importantes para a disponibilidade da aplicação.

Os gráficos quantil-quantil dos resíduos e resíduos versus resposta esperada encontram-se na Figura A.8.

Segundo a análise de variância realizada, todos os fatores foram significativos em um nível de significância de 0,1%, exceto o tempo de reinício. Como as hipóteses da ANOVA

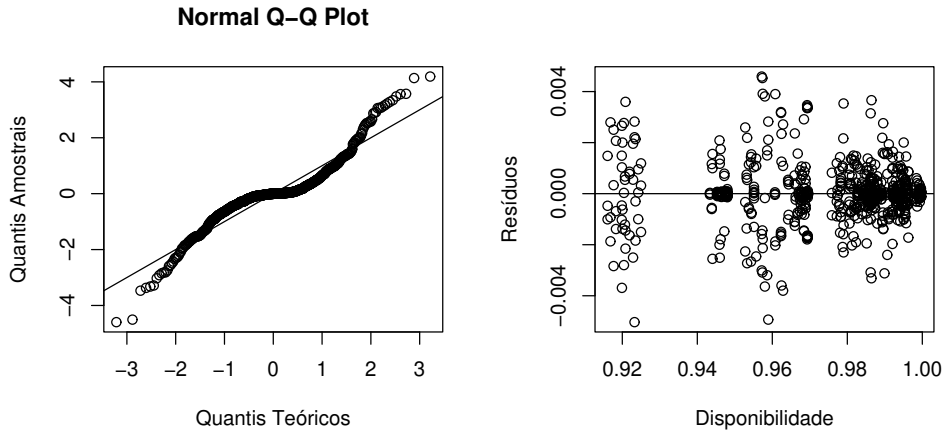

Figura A.8: Testes gráficos para realização da ANOVA

não foram completamente satisfeitas, o nível de significância de fato pode ser maior que 0,1% (COCHRAN; COX, 1957).

As seguintes tendências foram observadas:

1. Quando a variabilidade da carga aumenta, a disponibilidade da aplicação diminui;
2. Quando a capacidade dos nós aumenta, então, em geral, a disponibilidade da aplicação aumenta, mas em algumas poucas situações, diminui (será discutido posteriormente);
3. Quando o número de nós em falha aumenta, a disponibilidade da aplicação diminui;
4. O tempo de migração aumenta, piora a disponibilidade da aplicação;
5. Quando a utilização alvo aumenta, a disponibilidade da aplicação diminui;
6. Quando o tempo de reinício aumenta, a disponibilidade da aplicação não parece ser afetada;
7. Quando o tempo de adaptação aumenta, a disponibilidade da aplicação diminui.

Em geral, quando a capacidade dos nós aumenta, a disponibilidade da aplicação também aumenta. No entanto, quando a carga *down* é aplicada e a utilização alvo do *DPS* é 50%, observa-se que ao aumentar a capacidade dos nós, então diminui-se a disponibilidade da aplicação. Este foi o único cenário em que a disponibilidade da aplicação diminuiu ao aumentar a capacidade dos nós.

É interessante analisar os resultados do fator capacidade dos nós à luz dos resultados deste mesmo fator nas composições *AWoF+DPS* e *AWF+RS* correspondentes. Observa-se

que na composição *AWoF+DPS* quanto maior a capacidade dos nós, melhor foi a disponibilidade da aplicação. Já na composição *AWF+RS*, observa-se que quanto maior a capacidade dos nós, pior foi a disponibilidade da aplicação. Ao unir estes dois sistemas observa-se que aumentar a capacidade nem sempre leva a uma melhor disponibilidade. Conclui-se que as mais variadas combinações podem ocorrer em cenários como estes, em que o que é bom para o *RS* isoladamente não o é para o *DPS* isoladamente e vice-versa.

Projeto de experimentos da composição *AWF+DPS+RS* considerando cargas de maior intensidade

Na Tabela A.13 encontra-se a descrição dos fatores que estão associados a este experimento.

Tabela A.13: Fatores aplicados nos experimentos *AWF+DPS+RS* para carga grande

Fator	*Descrição*	*Valor no nível -1*	*Valor no nível 1*
A	Variabilidade da carga submetida à aplicação	Ver Figura A.5	Ver Figura A.6
B	Capacidade de processamento dos nós que executam a aplicação	15 rps	45 rps
C	Número inicial de nós em falha	1	2
D	Tempo médio de migração de um nó	1 min.	2 min.
E	Utilização alvo perseguida pelo *DPS*	50%	70%
F	Intensidade das falhas	300 ms.	500 ms
G	Tempo mínimo de rejuvenescimento de um nó	1 min.	2 min.
H	Tempo de adaptação do *DPS*	2,5 min.	5 min.

Na Tabela A.14 encontram-se os percentuais de variação da disponibilidade da aplicação explicado por alguns fatores e interações.

As seguintes tendências foram observadas:

1. Quando a variabilidade e intensidade da carga aumentam, a disponibilidade da aplicação também aumenta;

2. Quando a capacidade dos nós aumenta, então a disponibilidade da aplicação diminui;

3. Quando o número de nós em falha aumenta, a disponibilidade da aplicação diminui;

Tabela A.14: Alguns percentuais de variação de y devido aos fatores

Variabilidade da carga ($A_{\%}$)	9,44%
Capacidade dos nós ($B_{\%}$)	1,89%
Número de nós em falha ($C_{\%}$)	0,13%
Tempo de migração ($D_{\%}$)	2,09%
Utilização alvo do *DPS* ($E_{\%}$)	65,82%
Intensidade das falhas ($F_{\%}$)	$2,5 \times 10^{-3}$%
Tempo de reinício ($G_{\%}$)	0,053%
Tempo de adaptação do *DPS* ($H_{\%}$)	7,59%

4.Quando o tempo de migração aumenta a disponibilidade da aplicação diminui;

5.Quando a utilização alvo aumenta, a disponibilidade da aplicação diminui;

6.Quando o tempo de reinício aumenta, a disponibilidade da aplicação não é afetada;

7.Quando o tempo de adaptação aumenta, a disponibilidade da aplicação diminui.

Estas tendências estão em concordância com a grande maioria das tendências encontradas nos estudos realizados para *AWoF+DPS* e *AWF+RS* para as cargas de maior intensidade.

Na Figura A.9 encontram-se um gráfico quantil-quantil dos resíduos e um gráfico dos resíduos versus resposta esperada.

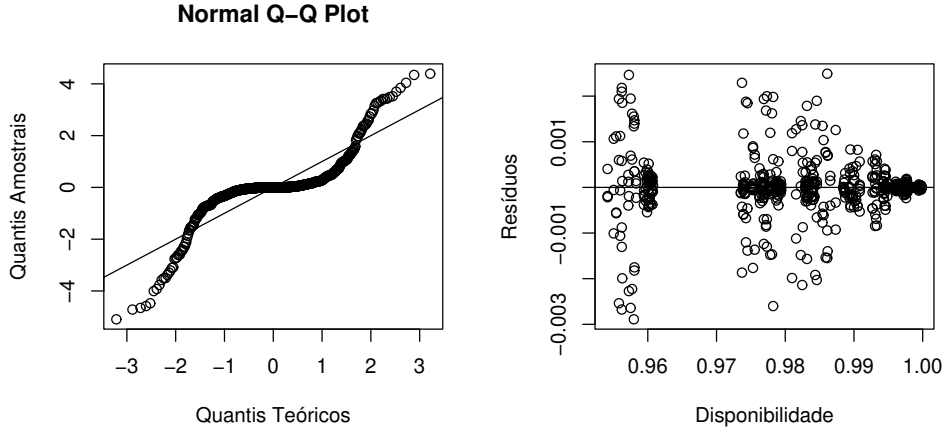

Figura A.9: Testes gráficos para realização da ANOVA

Segundo a análise de variância realizado, todos os fatores foram considerados significativos, com nível de significância 0,1%, exceto o tempo de reinício, que não foi significativo

e o fator intensidade das falhas, que foi significativo com nível de significância igual a 1%.

Conclusões parciais do projeto experimental de composições *AWF+DPS+RS*

Observa-se que os mesmos fatores considerados significativos para as composições *AWoF+DPS* e *AWF+RS* também são significativos quando *DPS* e *RS* atuam juntos. As tendências também são observadas, o que facilita o processo de seleção de fatores.

Quanto à capacidade dos nós, percebe-se que para o modelo *AWoF+DPS* era melhor ter nós de maior capacidade, que resultavam numa provisão extra maior e para o modelo *AWF+RS* era melhor ter nós de capacidade menor, que influenciam menos a disponibilidade da aplicação quando falham ou quando são rejuvenescidos. Quando *DPS* e *RS* atuam simultaneamente percebe-se que nós de capacidade menor resultaram em uma melhor disponibilidade da aplicação. É interessante, portanto, variar a capacidade dos nós ao estudar este modelo variando especialmente a capacidade dos nós.

A.0.4 Projeto de experimentos da composição *AWF+PDPS*RS*

O PDPS foi configurado para acrescentar um nó, além dos $\lceil \overline{\lambda}(t) \times \overline{d}(t) \rceil$ nós (i.e., $e = 1$).

A seguir, os fatores envolvidos são listados:

Tabela A.15: Fatores aplicados nos experimentos *AWF+PDPS*RS* para carga grande

Fator	*Descrição*	*Valor no nível -1*	*Valor no nível 1*
A	Variabilidade da carga submetida à aplicação	Ver Figura A.10	Ver Figura A.11
B	Capacidade de processamento dos nós que executam a aplicação	15 rps	30 rps
C	Número inicial de nós em falha	1	2
D	Tempo médio de migração de um nó	0,5 min.	1 min.
E	Intensidade das falhas	300 ms.	1 seg.
F	Tempo mínimo de rejuvenescimento de um nó	0,5 min.	1 min.
G	Tempo de adaptação do *RS* e do *PDPS*	2,5 min.	5 min.

A tabela A.16 apresenta os percentuais de variação associados a cada um dos fatores listados anteriormente para a composição *AWF+DPS*RS*.

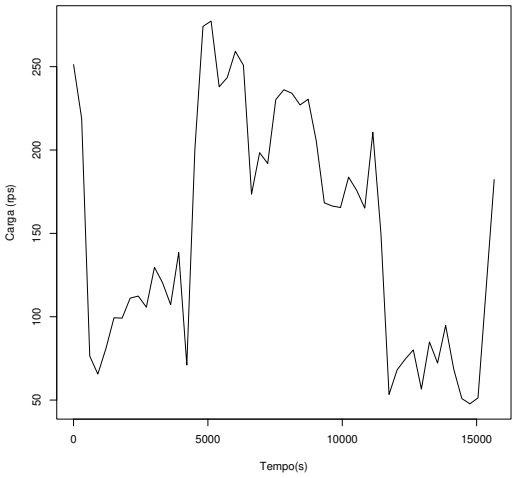

Figura A.10: Carga de trabalho menos variável e intensa, com um pico completo (*down*)

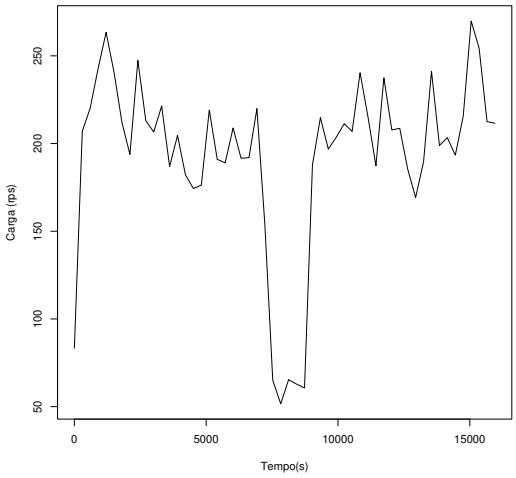

Figura A.11: Carga de trabalho mais variável e intensa, com dois picos completos (*up*)

Tabela A.16: Resultado do projeto experimental para a composição AWF+PDPS*RS

Fator	*Percentual de variação*
Variabilidade da carga ($A_{\%}$)	42,9%
Capacidade dos nós ($B_{\%}$)	21,4%
Número de nós em falha ($C_{\%}$)	22,2%
Tempo de migração ($D_{\%}$)	1,6%
Intensidade das falhas ($E_{\%}$)	0,8%
Tempo de reinício ($F_{\%}$)	0,00014%
Tempo de adaptação do *DPS* ($G_{\%}$)	0,0018%

O coeficiente de determinação (R^2) deste modelo foi 99,92% e o percentual de variação devido a erros ou outros fatores não considerados foi 0,08%.

A análise de variância indicou que dos fatores citados, apenas o tempo de rejuvenescimento não é estatisticamente significante. Os demais fatores foram considerados significativos e apresentaram tendências idênticas às observadas para a composição *AWF+DPS+RS*. Sendo assim, apenas os fatores carga de trabalho, capacidade dos nós e número de nós em falha serão modificados em estudos subseqüentes.

Os resultados da composição *AWF+PDPS+RS* apresentaram um coeficiente de determinação muito baixo e um percentual de variação devido a erros muito alto. Por esta razão estes resultados não foram apresentados aqui. Em todo caso, a análise de variância foi

feita para esta composição e o resultado foi bastante semelhante ao resultado da análise de variância obtido para a composição *AWF+PDPS*RS*. Assim, os mesmos fatores considerados importantes para a composição *AWF+PDPS*RS* serão também considerados quando a composição *AWF+PDPS+RS* for estudada em capítulos seguintes.

A.1 Conclusões parciais

O projeto de experimentos permitiu identificar os fatores mais importantes para cada uma das três composições estudadas.

Com o intuito de reduzir o número de experimentos a ser realizado à luz dos resultados obtidos, pretende-se eliminar os seguintes fatores:

1. Tempo médio de migração em composições em que o *DPS* atua. Apresentou a mesma tendência nos cenários estudados;

2. Intensidade das falhas em composições em que o *RS* atua. Apresentou a mesma tendência - já esperada - em todos os cenários estudados;

3. Tempo de reinício em composições em que o *RS* atua. Foi praticamente irrelevante em todos os cenários estudados;

4. Número de nós ativos nos experimentos em que o *DPS* não atua. Quanto maior o número de nós melhor a disponibilidade (maior super-provisão);

5. Tempo de adaptação do *DPS*. Quanto menor o tempo de adaptação melhor foi a disponibilidade.

Por eliminar os fatores entende-se que eles serão mantidos em certos valores em todos os experimentos. Os demais fatores serão variados em dois níveis.

O número de nós ativos na composição *AWF+RS* mostrou-se significativo, no entanto, a variação deste fator não é importante para o objetivo final do estudo que se propõe aqui. Os modelos onde não ocorre provisão dinâmica são usados como base para comparação com os modelos onde o *DPS* atua. Escolher um número de nós suficiente para processar as requisições dos clientes e ter alguns nós rejuvenescidos já é o bastante.

A utilização alvo do *DPS*, apesar de apresentar forte tendência que se repetiu em todos os experimentos, mostrou-se bastante importante e portanto será variada.

O projeto experimental serviu não apenas para conhecer os fatores mais relevantes para os modelos. Os resultados obtidos foram os resultados esperados e a análise de projeto

também serviu como um teste de corretude do simulador. As tendências observadas e apresentadas no decorrer deste capítulo eram esperadas.

Apêndice B

Intervalos de confiança de experimentos de simulação apresentados no Capítulo 6

B.1 Intervalos de confiança da disponibilidade e tempo de resposta de experimentos que usaram o *PDPS*

Na tabela B.1 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade e tempo de resposta dos experimentos onde o sistema de provisão dinâmica perfeito de recursos foi aplicado.

Tabela B.1: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	90%	0,05%
Tempo de resposta [T. R. (s)]	90%	5%

Tabela B.2: Intervalos de confiança da disponibilidade e tempo de resposta para a composição AWoF+PDPS

Capacidade	*Carga*	**Disponibilidade**	**T. R. (s)**
down	15 rps	100,000% ± 0,000%	4,48 ± 0,06
	30 rps	100,000% ± 0,000%	1,27 ± 0,00
up	15 rps	100,000% ± 0,000%	4,19 ± 0,03
	30 rps	100,000% ± 0,000%	1,46 ± 0,01

Tabela B.3: Intervalos de confiança da disponibilidade e do tempo de resposta para a composição AWF+RS

Capacidade	*Carga*	**# nós em falha**	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	99,903% ± 0,000%	3,33 ± 0,01
		2	99,800% ± 0,000%	6,72 ± 0,02
	45 rps	1	99,749% ± 0,000%	2,19 ± 0,02
		2	99,498% ± 0,000%	4,08 ± 0,03
up	15 rps	1	99,831% ± 0,000%	0,80 ± 0,00
		2	99,660% ± 0,000%	1,50 ± 0,01
	45 rps	1	99,631% ± 0,000%	0,98 ± 0,01
		2	99,260% ± 0,000%	1,90 ± 0,03

Tabela B.4: Intervalos de confiança da disponibilidade e do tempo de resposta para a composição AWF+PDPS+RS

Capacidade	*Carga*	*# de nós em falha*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	99,768% ± 0,026%	5,66 ± 0,12
		2	99,499% ± 0,038%	6,97 ± 0,24
	45 rps	1	99,443% ± 0,049%	2,98 ± 0,17
		2	98,686% ± 0,089%	5,87 ± 0,51
up	15 rps	1	99,593% ± 0,000%	4,89 ± 0,03
		2	99,187% ± 0,000%	5,82 ± 0,04
	45 rps	1	99,205% ± 0,000%	2,38 ± 0,02
		2	98,422% ± 0,000%	4,09 ± 0,06

B.2 Intervalos de confiança da disponibilidade e tempo de resposta de experimentos que usaram o *PDPS2*

Na tabela B.5 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade e tempo de resposta dos experimentos onde o sistema de provisão dinâmica perfeito de recursos foi aplicado.

Tabela B.5: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	90%	0,05%
Tempo de resposta [T. R. (s)]	90%	5%

Tabela B.6: Intervalos de confiança da disponibilidade e tempo de resposta para a composição AWoF+PDPS2

Capacidade	*Carga*	**Disponibilidade**	**T. R. (s)**
down	15 rps	100,000% ± 0,000%	1,58 ± 0,02
	30 rps	100,000% ± 0,000%	0,39 ± 0,00
up	15 rps	100,000% ± 0,000%	1,77 ± 0,08
	30 rps	100,000% ± 0,000%	0,44 ± 0,00

B.3 Intervalos de confiança da disponibilidade, tempo de resposta, número de nós ativos e número de requisições rejeitadas

B.3.1 Cenário I

Na tabela B.8 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade, tempo de resposta, número de nós ativos e número de requisições rejeitadas.

Tabela B.7: Intervalos de confiança da disponibilidade e do tempo de resposta para a composição AWF+PDPS2+RS

Capacidade	*Carga*	*# de nós em falha*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	99,763% ± 0,033%	2,84 ± 0,16
		2	99,562% ± 0,039%	3,97 ± 0,21
	45 rps	1	99,511% ± 0,046%	2,09 ± 0,20
		2	99,022% ± 0,067%	3,86 ± 0,30
up	15 rps	1	99,539% ± 0,000%	2,57 ± 0,02
		2	99,080% ± 0,000%	3,43 ± 0,02
	45 rps	1	99,153% ± 0,000%	1,53 ± 0,01
		2	98,306% ± 0,000%	2,80 ± 0,02

Tabela B.8: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	95%	0,08%
Tempo de resposta [T. R. (s)]	95%	5%
Número de nós ativos	95%	0,5%
Número de requisições rejeitadas	90%	10%

Tabela B.9: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWoF+DPS

Capacidade	*Carga*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**	**# nós ativos**	**# de requisições rejeitadas**
down	15 rps	50%	99,997% ± 0,001%	0,77 ± 0,02	46,27 ± 0,01	204 ± 43
		70%	99,092% ± 0,027%	5,41 ± 0,05	32,91 ± 0,02	60312 ± 1483
	45 rps	50%	99,833% ± 0,000%	0,34 ± 0,00	15,89 ± 0,01	11112 ± 1
		70%	98,845% ± 0,026%	2,33 ± 0,05	11,43 ± 0,01	76738 ± 1428
up	15 rps	50%	99,537% ± 0,001%	2,27 ± 0,02	59,98 ± 0,01	40209 ± 65
		70%	95,596% ± 0,028%	7,68 ± 0,04	41,42 ± 0,02	382632 ± 2011
	45 rps	50%	99,314% ± 0,020%	0,97 ± 0,01	20,54 ± 0,01	59609 ± 1437
		70%	97,362% ± 0,022%	2,54 ± 0,03	14,55 ± 0,01	229157 ± 1591

Tabela B.10: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWF+RS

Capacidade	*Carga*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**	**# nós ativos**	**# de requisições rejeitadas**
down	15 rps	1	100,000% ± 0,000%	0,09 ± 0,00	100 ± 0,00	0 ± 0
		2	100,000% ± 0,000%	0,11 ± 0,00	100 ± 0,00	0 ± 0
	45 rps	1	99,984% ± 0,000%	0,22 ± 0,00	36 ± 0,00	1046 ± 0
		2	99,968% ± 0,000%	0,41 ± 0,00	36 ± 0,00	2132 ± 0
up	15 rps	1	100,000% ± 0,000%	0,11 ± 0,00	88 ± 0,00	0 ± 0
		2	100,000% ± 0,000%	0,13 ± 0,00	88 ± 0,00	0 ± 0
	45 rps	1	99,915% ± 0,000%	0,17 ± 0,00	30 ± 0,00	7377 ± 0
		2	99,829% ± 0,000%	0,31 ± 0,00	30 ± 0,00	14819 ± 0

Tabela B.11: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWF+DPS+RS

Capacidade	*Carga*	*# de nós em falha*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**	**# nós ativos**	**# de requisições rejeitadas**
down	15 rps	1	50%	99,981% ± 0,002%	0,98 ± 0,03	46,38 ± 0,02	1235 ± 94
			70%	98,634% ± 0,019%	6,22 ± 0,04	32,86 ± 0,01	90751 ± 1031
		2	50%	99,969% ± 0,001%	1,12 ± 0,03	46,43 ± 0,02	2033 ± 80
			70%	98,565% ± 0,018%	6,39 ± 0,05	32,91 ± 0,02	95372 ± 979
	45 rps	1	50%	99,593% ± 0,032%	0,56 ± 0,04 [1]	16,00 ± 0,01	27066 ± 1778
			70%	98,254% ± 0,063%	2,96 ± 0,04 [2]	11,40 ± 0,01	116015 ± 3494
		2	50%	99,453% ± 0,032%	0,77 ± 0,05	16,07 ± 0,02	36346 ± 1738
			70%	97,994% ± 0,039%	3,15 ± 0,03	11,43 ± 0,01	133266 ± 2144
up	15 rps	1	50%	99,343% ± 0,010%	2,53 ± 0,02	60,07 ± 0,01	57080 ± 720
			70%	94,583% ± 0,004%	8,20 ± 0,04	41,11 ± 0,01	470651 ± 291
		2	50%	99,348% ± 0,001%	2,56 ± 0,04	60,17 ± 0,01	56686 ± 55
			70%	94,448% ± 0,004%	8,50 ± 0,02	41,09 ± 0,01	482400 ± 269
	45 rps	1	50%	99,013% ± 0,003%	1,13 ± 0,01	20,59 ± 0,01	85717 ± 194
			70%	96,974% ± 0,075%	2,97 ± 0,04	14,52 ± 0,01	262935 ± 5423
		2	50%	98,869% ± 0,003%	1,24 ± 0,01	20,66 ± 0,01	98256 ± 192
			70%	97,249% ± 0,003%	2,71 ± 0,01	14,69 ± 0,01	238998 ± 202

B.3.2 Cenário II

Na tabela B.12 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade, tempo de resposta, número de nós ativos e número de requisições rejeitadas.

Tabela B.12: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	95%	0,05%
Tempo de resposta [T. R. (s)]	95%	5%
Número de nós ativos	95%	0,5%
Número de requisições rejeitadas	90%	12%

[1] $\varepsilon = 8\%$
[2] $\varepsilon = 8\%$

Tabela B.13: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário II para a composição AWoF+DPS

Capacidade	*Carga*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**	**# nós ativos**	**# de requisições rejeitadas**
down	15 rps	50%	99,997% ± 0,004%	0,95 ± 0,01	93,65 ± 0,02	926 ± 525
		70%	99,400% ± 0,015%	5,19 ± 0,13	66,70 ± 0,02	80756 ± 2014
	45 rps	50%	99,785% ± 0,020%	0,55 ± 0,02	31,84 ± 0,02	28894 ± 2749
		70%	98,904% ± 0,047%	2,43 ± 0,07	22,72 ± 0,02	147319 ± 6303
up	15 rps	50%	100,000% ± 0,000%	0,39 ± 0,01	126,82 ± 0,02	0 ± 0
		70%	99,303% ± 0,004%	3,74 ± 0,03	90,23 ± 0,01	126786 ± 799
	45 rps	50%	99,997% ± 0,000%	0,038 ± 0,01	43,13 ± 0,01	543 ± 76
		70%	99,187% ± 0,016%	2,04 ± 0,06	30,76 ± 0,01	147989 ± 2873

Tabela B.14: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário II para a composição AWF+RS

Capacidade	*Carga*	*# de nós em falha*	**Disponibilidade**	**T. R. (s)**	**# nós ativos**	**# de requisições rejeitadas**
down	15 rps	1	99,998% ± 0,000%	0,22 ± 0,00	200 ± 0,00	222 ± 0
		2	99,997% ± 0,000%	0,36 ± 0,00	200 ± 0,00	451 ± 0
	45 rps	1	99,947% ± 0,000%	0,25 ± 0,00	68 ± 0,00	7059 ± 0
		2	99,895% ± 0,000%	0,47 ± 0,00	68 ± 0,00	14144 ± 0
up	15 rps	1	99,994% ± 0,000%	0,17 ± 0,00	176 ± 0,00	1098 ± 0
		2	99,988% ± 0,000%	0,24 ± 0,00	176 ± 0,00	2171 ± 0
	45 rps	1	99,957% ± 0,000%	0,16 ± 0,00	60 ± 0,00	7807 ± 0
		2	99,913% ± 0,000%	0,28 ± 0,00	60 ± 0,00	15788 ± 0

Tabela B.15: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário II para a composição AWF+DPS+RS

Capacidade	*Carga*	*# de nós em falha*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**	**# nós ativos**	**# de requisições rejeitadas**
down	15 rps	1	50%	99,983% ± 0,003%	0,93 ± 0,04	93,81 ± 0,02	2297 ± 443
			70%	99,292% ± 0,009%	5,67 ± 0,02	66,79 ± 0,02	95161 ± 1186
		2	50%	99,972% ± 0,004%	1,07 ± 0,04	93,86 ± 0,03	3715 ± 582
			70%	99,253% ± 0,012%	5,86 ± 0,09	66,82 ± 0,02	100365 ± 1564
	45 rps	1	50%	99,725% ± 0,017%	0,79 ± 0,04	31,93 ± 0,02	36976 ± 2222
			70%	98,742% ± 0,021%	2,65 ± 0,05	22,72 ± 0,01	168993 ± 2861
		2	50%	99,654% ± 0,020%	1,01 ± 0,05	31,98 ± 0,03	46474 ± 2714
			70%	98,632% ± 0,033%	2,86 ± 0,05	22,75 ± 0,02	183803 ± 4420
up	15 rps	1	50%	99,960% ± 0,000%	0,58 ± 0,00	126,91 ± 0,02	7219 ± 42
			70%	99,173% ± 0,001%	4,16 ± 0,08	90,18 ± 0,01	150508 ± 232
		2	50%	99,922% ± 0,000%	0,65 ± 0,00	127,02 ± 0,02	14193 ± 52
			70%	99,110% ± 0,001%	4,22 ± 0,03	90,21 ± 0,01	161924 ± 199
	45 rps	1	50%	99,816% ± 0,001%	0,56 ± 0,02	43,12 ± 0,01	33522 ± 255
			70%	98,997% ± 0,001%	2,15 ± 0,00	30,76 ± 0,01	182540 ± 185
		2	50%	99,670% ± 0,002%	0,68 ± 0,02	43,19 ± 0,02	60141 ± 323
			70%	98,798% ± 0,001%	2,30 ± 0,01	30,81 ± 0,00	218720 ± 227

B.4 Intervalos de confiança da disponibilidade e tempo de resposta durante ocorrência de falhas e rejuvenescimentos

Na tabela B.16 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade e tempo de resposta computados durante a ocorrência de falhas e rejuvenescimentos.

Tabela B.16: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	90%	1%
Tempo de resposta [T. R. (s)]	90%	5%

Tabela B.17: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWoF+DPS

Capacidade	*Carga*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**
down	15 rps	50%	100,000% ± 0,000%	0,15 ± 0,00
		70%	100,000% ± 0,0007%	0,86 ± 0,01
	45 rps	50%	100,000% ± 0,000%	0,08 ± 0,00
		70%	100,000% ± 0,000%	0,92 ± 0,02
up	15 rps	50%	91,632% ± 0,014%	23,83 ± 0,35
		70%	41,320% ± 0,119%	56,64 ± 0,23
	45 rps	50%	91,857% ± 0,013%	9,36 ± 0,10
		70%	66,218% ± 0,432%	20,84 ± 0,33

B.4.1 Cenário II

Na tabela B.20 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade e tempo de resposta computados durante a ocorrência de falhas e rejuvenescimentos.

Tabela B.18: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWF+RS

Capacidade	*Carga*	*# de nós em falha*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	100,000% ± 0,000%	0,12 ± 0,00
		2	100,000% ± 0,000%	0,18 ± 0,00
	45 rps	1	99,726% ± 0,000%	3,19 ± 0,00
		2	99,441% ± 0,000%	6,29 ± 0,00
up	15 rps	1	100,000% ± 0,000%	0,36 ± 0,00
		2	100,000% ± 0,000%	0,62 ± 0,00
	45 rps	1	98,617% ± 0,000%	1,96 ± 0,00
		2	97,216% ± 0,000%	3,93 ± 0,00

Tabela B.19: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWF+DPS+RS

Capacidade	*Carga*	*# de nós em falha*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	50%	99,728% ± 0,030%	2,57 ± 0,37
			70%	99,008% ± 0,118%	4,29 ± 0,38
		2	50%	99,503% ± 0,025%	4,75 ± 0,45
			70%	98,026% ± 0,166%	6,88 ± 0,59
	45 rps	1	50%	97,240% ± 0,506%	3,15 ± 0,65
			70%	95,390% ± 0,671%	3,91 ± 0,53
		2	50%	94,772% ± 0,552%	6,47 ± 0,87
			70%	90,542% ± 0,732%	7,33 ± 0,72
up	15 rps	1	50%	88,904% ± 0,131%	26,38 ± 0,22
			70%	32,777% ± 0,043%	60,32 ± 0,32
		2	50%	89,050% ± 0,013%	26,63 ± 0,48
			70%	30,688% ± 0,041%	64,89 ± 0,17
	45 rps	1	50%	86,681% ± 0,048%	11,54 ± 0,06
			70%	59,180% ± 1,111%	25,21 ± 0,54
		2	50%	84,017% ± 0,044%	13,35 ± 0,10
			70%	64,662% ± 0,032%	24,70 ± 0,14

Tabela B.20: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	95%	0,5%
Tempo de resposta [T. R. (s)]	95%	5%

Tabela B.21: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWoF+DPS

Capacidade	*Carga*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**
down	15 rps	50%	100,000% ± 0,000%	0,12 ± 0,00
		70%	100,000% ± 0,000%	0,39 ± 0,00
	45 rps	50%	100,000% ± 0,000%	0,05 ± 0,00
		70%	100,000% ± 0,000%	0,18 ± 0,01
up	15 rps	50%	100,000% ± 0,000%	4,23 ± 0,16
		70%	84,029% ± 0,108%	44,63 ± 0,44
	45 rps	50%	99,935% ± 0,010%	4,99 ± 0,21
		70%	82,573% ± 0,148%	20,69 ± 0,46

Tabela B.22: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWF+RS

Capacidade	*Carga*	*# de nós em falha*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	99,973% ± 0,000%	1,48 ± 0,02
		2	99,944% ± 0,000%	2,88 ± 0,01
	45 rps	1	99,128% ± 0,000%	1,98 ± 0,03
		2	98,250% ± 0,000%	3,92 ± 0,10
up	15 rps	1	99,902% ± 0,000%	0,71 ± 0,01
		2	99,807% ± 0,000%	1,32 ± 0,01
	45 rps	1	99,043% ± 0,000%	1,21 ± 0,00
		2	98,060% ± 0,000%	2,45 ± 0,01

Tabela B.23: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário II para a composição AWF+DPS+RS

Capacidade	*Carga*	*# de nós em falha*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	50%	99,717% ± 0,059%	1,66 ± 0,02
			70%	99,555% ± 0,071%	2,05 ± 0,02
		2	50%	99,516% ± 0,066%	3,14 ± 0,06
			70%	99,059% ± 0,109%	3,69 ± 0,07
	45 rps	1	50%	98,773% ± 0,289%	2,07 ± 0,15
			70%	98,110% ± 0,357%	2,30 ± 0,14
		2	50%	97,522% ± 0,303%	4,25 ± 0,14
			70%	96,052% ± 0,500%	4,46 ± 0,17
up	15 rps	1	50%	99,126% ± 0,006%	6,83 ± 0,18
			70%	81,172% ± 0,034%	47,37 ± 0,47
		2	50%	98,279% ± 0,007%	7,44 ± 0,10
			70%	79,713% ± 0,028%	47,94 ± 0,39
	45 rps	1	50%	95,944% ± 0,034%	7,17 ± 0,09
			70%	78,379% ± 0,021%	22,34 ± 0,25
		2	50%	92,701% ± 0,038%	8,33 ± 0,10
			70%	73,941% ± 0,030%	23,55 ± 0,58

Apêndice C

Intervalos de confiança de experimentos de simulação da composição AWF+DPS*RS apresentados no Capítulo 7

C.1 Intervalos de confiança da disponibilidade e tempo de resposta de experimentos que usaram o *PDPS*

Na tabela C.1 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade e tempo de resposta dos experimentos onde o sistema de provisão dinâmica perfeito de recursos foi aplicado.

Tabela C.1: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	90%	0,05%
Tempo de resposta [T. R. (s)]	90%	5%

Tabela C.2: Intervalos de confiança da disponibilidade e do tempo de resposta para a composição AWF+PDPS*RS

Capacidade	*Carga*	*# de nós em falha*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	99,926% ± 0,000%	4,93 ± 0,06
		2	99,851% ± 0,000%	5,31 ± 0,06
	45 rps	1	99,797% ± 0,000%	2,04 ± 0,01
		2	99,328% ± 0,000%	3,34 ± 0,02
up	15 rps	1	99,593% ± 0,000%	4,71 ± 0,03
		2	99,187% ± 0,000%	5,34 ± 0,03
	45 rps	1	99,205% ± 0,000%	2,33 ± 0,01
		2	98,422% ± 0,000%	3,33 ± 0,03

C.2 Intervalos de confiança da disponibilidade e tempo de resposta de experimentos que usaram o *PDPS2*

Na tabela C.3 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade e tempo de resposta dos experimentos onde o sistema de provisão dinâmica perfeito de recursos foi aplicado.

Tabela C.3: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	90%	0,05%
Tempo de resposta [T. R. (s)]	90%	5%

Tabela C.4: Intervalos de confiança da disponibilidade e do tempo de resposta para a composição AWF+PDPS2*RS

Capacidade	*Carga*	*# de nós em falha*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	99,934% ± 0,000%	1,99 ± 0,01
		2	99,869% ± 0,000%	2,36 ± 0,01
	45 rps	1	99,825% ± 0,000%	1,14 ± 0,01
		2	99,414% ± 0,000%	2,43 ± 0,01
up	15 rps	1	99,539% ± 0,000%	2,59 ± 0,02
		2	99,080% ± 0,000%	3,37 ± 0,02
	45 rps	1	99,153% ± 0,000%	1,52 ± 0,01
		2	98,306% ± 0,000%	2,74 ± 0,03

C.3 Intervalos de confiança da disponibilidade, tempo de resposta, número de nós ativos e número de requisições rejeitadas

C.3.1 Cenário I

Na tabela C.5 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade, tempo de resposta, número de nós ativos e número de requisições rejeitadas.

Tabela C.5: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	95%	0,08%
Tempo de resposta [T. R. (s)]	95%	5%
Número de nós ativos	95%	0,5%
Número de requisições rejeitadas	90%	10%

Tabela C.6: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWF+DPS*RS

Capacidade	*Carga*	*# de nós em falha*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**	**# nós ativos**	**# de requisições rejeitadas**
down	15 rps	1	50%	99,996% ± 0,000%	0,848 ± 0,01	46,32 ± 0,01	275 ± 0
			70%	98,719% ± 0,015%	6,07 ± 0,04	32,81 ± 0,01	85138 ± 832
		2	50%	99,982% ± 0,000%	0,95 ± 0,01	46,36 ± 0,01	1218 ± 0
			70%	98,716% ± 0,018%	6,08 ± 0,04	32,82 ± 0,01	85304 ± 995
	45 rps	1	50%	99,714% ± 0,000%	0,42 ± 0,01	15,96 ± 0,00	19020 ± 17
			70%	99,485% ± 0,042%	2,79 ± 0,02	11,39 ± 0,01	100637 ± 2319
		2	50%	99,697% ± 0,000%	0,46 ± 0,01	15,98 ± 0,01	20141 ± 19
			70%	98,204% ± 0,038%	2,96 ± 0,02	11,40 ± 0,01	119354 ± 2099
up	15 rps	1	50%	99,408% ± 0,007%	2,48 ± 0,02	60,04 ± 0,01	51438 ± 477
			70%	94,817% ± 0,003%	8,29 ± 0,02	41,19 ± 0,01	450294 ± 221
		2	50%	99,352% ± 0,007%	2,51 ± 0,03	60,08 ± 0,01	56286 ± 492
			70%	94,634% ± 0,004%	8,66 ± 0,02	41,19 ± 0,01	466256 ± 270
	45 rps	1	50%	99,028% ± 0,001%	1,12 ± 0,00	20,56 ± 0,00	84438 ± 65
			70%	97,286% ± 0,002%	2,66 ± 0,00	14,55 ± 0,01	235814 ± 169
		2	50%	98,883% ± 0,002%	1,23 ± 0,01	20,60 ± 0,00	97040 ± 146
			70%	97,249% ± 0,003%	2,73 ± 0,01	14,63 ± 0,01	238996 ± 234

C.3.2 Cenário II

Na tabela C.7 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade, tempo de resposta, número de nós ativos e número de requisições rejeitadas.

Tabela C.7: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	95%	0,05%
Tempo de resposta [T. R. (s)]	95%	5%
Número de nós ativos	95%	0,5%
Número de requisições rejeitadas	90%	12%

Tabela C.8: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário II para a composição AWF+DPS*RS

Capacidade	*Carga*	*# de nós em falha*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**	**# nós ativos**	**# de requisições rejeitadas**
down	15 rps	1	50%	100,000% ± 0,000%	0,86 ± 0,04	93,81 ± 0,02	2297 ± 443
			70%	99,332% ± 0,007%	5,64 ± 0,07	66,79 ± 0,02	95161 ± 1186
		2	50%	100,000% ± 0,000%	0,90 ± 0,02	93,86 ± 0,03	3715 ± 582
			70%	99,332% ± 0,008%	5,65 ± 0,18	66,82 ± 0,02	100365 ± 1564
	45 rps	1	50%	99,788% ± 0,001%	0,66 ± 0,05	31,93 ± 0,02	36976 ± 2222
			70%	98,828% ± 0,002%	2,52 ± 0,05	22,72 ± 0,01	168993 ± 2861
		2	50%	99,772% ± 0,001%	0,74 ± 0,02	31,98 ± 0,03	46474 ± 2714
			70%	98,800% ± 0,002%	2,60 ± 0,02	22,75 ± 0,02	183803 ± 4420
up	15 rps	1	50%	99,961% ± 0,000%	0,57 ± 0,01	126,91 ± 0,02	7219 ± 42
			70%	99,173% ± 0,001%	4,00 ± 0,08	90,18 ± 0,01	150508 ± 232
		2	50%	99,924% ± 0,000%	0,63 ± 0,00	127,02 ± 0,02	14193 ± 52
			70%	99,109% ± 0,001%	4,15 ± 0,05	90,21 ± 0,01	161924 ± 199
	45 rps	1	50%	99,824% ± 0,001%	0,55 ± 0,00	43,12 ± 0,01	33522 ± 255
			70%	98,996% ± 0,001%	2,18 ± 0,06	30,76 ± 0,01	182540 ± 185
		2	50%	99,684% ± 0,001%	0,68 ± 0,01	43,19 ± 0,02	60141 ± 323
			70%	98,799% ± 0,001%	2,31 ± 0,02	30,81 ± 0,00	218720 ± 227

C.4 Intervalos de confiança da disponibilidade e tempo de resposta durante ocorrência de falhas e rejuvenescimentos

C.5 Cenário I

Na tabela C.9 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade e tempo de resposta computados durante a ocorrência de falhas e rejuvenescimentos.

Tabela C.9: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	95%	0,5%
Tempo de resposta [T. R. (s)]	95%	5%

C.6 Cenário II

Na tabela C.11 são apresentados os níveis de confiança considerados para a geração dos intervalos de confiança da disponibilidade e tempo de resposta computados durante a ocorrência de falhas e rejuvenescimentos.

Tabela C.10: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário I para a composição AWF+DPS*RS

Capacidade	*Carga*	*# de nós em falha*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	50%	100,000% ± 0,000%	0,36 ± 0,00
			70%	100,000% ± 0,000%	2,15 ± 0,04
		2	50%	99,733% ± 0,000%	2,11 ± 0,03
			70%	100,000% ± 0,000%	2,06 ± 0,02
	45 rps	1	50%	99,655% ± 0,000%	0,77 ± 0,03
			70%	99,226% ± 0,000%	1,66 ± 0,03
		2	50%	99,334% ± 0,000%	1,43 ± 0,04
			70%	95,109% ± 0,001%	3,86 ± 0,11
up	15 rps	1	50%	90,091% ± 0,022%	25,71 ± 0,26
			70%	33,774% ± 0,049%	60,42 ± 0,08
		2	50%	89,048% ± 0,008%	26,09 ± 0,33
			70%	31,501% ± 0,047%	65,16 ± 0,27
	45 rps	1	50%	86,966% ± 0,012%	11,35 ± 0,04
			70%	64,729% ± 0,037%	22,76 ± 0,05
		2	50%	84,288% ± 0,031%	13,31 ± 0,14
			70%	64,776% ± 0,040%	24,68 ± 0,09

Tabela C.11: Níveis de confiança e erros considerados para a geração dos intervalos de confiança

	Nível de confiança	*Erro máximo (ε)*
Disponibilidade	95%	0,5%
Tempo de resposta [T. R. (s)]	95%	5%

Tabela C.12: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação no Cenário II para a composição AWF+DPS*RS

Capacidade	*Carga*	*# de nós em falha*	*Utilização alvo*	**Disponibilidade**	**T. R. (s)**
down	15 rps	1	50%	100,000% ± 0,059%	1,66 ± 0,02
			70%	99,555% ± 0,071%	2,05 ± 0,02
		2	50%	99,516% ± 0,066%	3,14 ± 0,06
			70%	99,059% ± 0,109%	3,69 ± 0,07
	45 rps	1	50%	98,773% ± 0,289%	2,07 ± 0,15
			70%	98,110% ± 0,357%	2,30 ± 0,14
		2	50%	97,522% ± 0,303%	4,25 ± 0,14
			70%	96,052% ± 0,500%	4,46 ± 0,17
up	15 rps	1	50%	99,126% ± 0,006%	6,83 ± 0,18
			70%	81,172% ± 0,034%	47,37 ± 0,47
		2	50%	98,279% ± 0,007%	7,44 ± 0,10
			70%	79,713% ± 0,028%	47,94 ± 0,39
	45 rps	1	50%	95,944% ± 0,034%	7,17 ± 0,09
			70%	78,379% ± 0,021%	22,34 ± 0,25
		2	50%	92,701% ± 0,038%	8,33 ± 0,10
			70%	73,941% ± 0,030%	23,55 ± 0,58

Apêndice D

Verificação e validação do modelo de simulação

Neste apêndice são apresentadas atividades realizadas com o intuito de verificar e validar o modelo de simulação proposto e utilizado nos capítulos 5 e 6 desta tese.

A verificação do modelo de simulação foi realizada de duas formas: através de testes de unidade e testes de limites. Mais detalhes destas atividades são apresentados na Seção D.1.

A validação do modelo foi realizada através de alguns experimentos de medição. Nestes experimentos foram utilizadas ferramentas comerciais. Cenários de teste bastante semelhantes aos cenários de medição foram estudados através de simulações e seus resultados comparados qualitativamente.

Após a análise dos resultados, pode-se afirmar que:

1. Ao variar alguns parâmetros da simulação e dos experimentos de medição verificou-se que a aplicação - em termos de qualidade de serviço e custo - se comporta de forma semelhante;

2. Ao usar *middleware* real, também se verifica (assim como foi verificado via experimentos de simulação) que a união dos sistemas de provisão dinâmica e rejuvenescimento sem coordenação é ineficiente em termos de qualidade de serviço da aplicação;

3. Ao realizar a coordenação dos sistemas verificou-se que, na prática, as técnicas de coordenação foram eficientes.

Como será visto mais adiante, não foi possível realizar uma comparação quantitativa, uma vez que o ambiente real traz muito mais complexidades e custos ("*overheads*"), que não foram considerados no ambiente de simulação. No entanto, as tendências de cada cenário podem ser comparadas e estas tendências validam os resultados obtidos.

D.1 Verificação do modelo de simulação

Testes de unidade foram criados para testar cada entidade de simulação. Por se tratar de detalhes muito específicos de implementação, tais testes não serão discutidos aqui.

Os testes de limites, que são testes mais gerais, são apresentados na subseção a seguir.

D.1.1 Análise de valores limites

Com o intuito de testar o simulador na presença de situações limite, os seguintes testes foram realizados utilizando-se o simulador:

- *Limite superior da vazão.* A vazão da aplicação é a quantidade de requisições que é processada por unidade de tempo. Se as requisições possuem uma demanda de serviço média d, então a vazão de uma aplicação com r nós ativos não pode ser superior a $\frac{r}{d}$. Este teste tem o objetivo de verificar se este limite superior para a vazão está sendo obedecido pelo simulador;

- *Limites superior e inferior da utilização.* A utilização de um servidor em um intervalo de tempo indica o percentual de tempo em que o servidor permaneceu ocupado no intervalo de tempo em questão. Sendo um percentual, o valor da utilização deve estar dentro do intervalo $[0, 1]$. Este teste tem o objetivo de verificar se este intervalo está sendo obedecido de forma adequada;

- *Limites superior e inferior do tempo de resposta.* Tendo cada servidor uma capacidade de atender c requisições "simultaneamente" e sendo b o tamanho da fila de *Backlog* de cada servidor, então uma requisição poderá passar um máximo de $(b+c) \times d$ unidades de tempo para obter serviço. Além disso, mesmo que não haja requisições na fila, o tempo mínimo de serviço de qualquer requisição deve ser no mínimo dada pela sua demanda de serviço d, com $d > 0$;

- *Limites superior e inferior da disponibilidade.* A disponibilidade da aplicação em um intervalo de tempo é o percentual de requisições atendidas com sucesso durante este intervalo. Sendo um percentual, o valor da disponibilidade deve estar dentro do intervalo $[0, 1]$. Este teste tem o objetivo de verificar se este intervalo está sendo obedecido de forma adequada.

D.1.2 Teste do limite superior da vazão

Seja r o número de nós ativos. Seja λ a taxa de chegada de requisições na aplicação. Cada requisição tem uma demanda de serviço de d unidades de tempo. Sendo assim, a taxa máxima de serviço desta aplicação é dada por $\frac{r}{d}$. Se todas as filas de todos os servidores estiverem vazias e $\lambda < \frac{r}{d}$, então a vazão da aplicação vai ser exatamente igual a λ. Mesmo que λ cresça e passe a ser maior que a taxa máxima de serviço da aplicação, a vazão da aplicação não deverá ser maior que $\frac{r}{d}$, pois este é um limite máximo para a vazão.

O teste do limite superior da vazão consiste em gerar uma carga de trabalho constante e superior a $\frac{r}{d}$ e verificar se a vazão da aplicação está ultrapassando o limite de $\frac{r}{d}$ para a vazão. Os seguintes valores foram usados neste teste:

Tabela D.1: Valores usados nos testes 1, 2 e 3 de limite máximo da vazão

Teste 1	*Teste 2*	*Teste 3*
$\lambda = 50$ rps	$\lambda = 500$ rps	$\lambda = 5000$ rps
$d = 0,050$ s	$d = 0,050$ s	$d = 0,050$ s
$r = 2$	$r = 20$	$r = 200$
$b = 6000$	$b = 6000$	$b = 6000$
$c = 500$	$c = 500$	$c = 500$
$\frac{r}{d} = 40$	$\frac{r}{d} = 400$	$\frac{r}{d} = 4000$

Todos os testes tiveram uma duração de 10 minutos de simulação e em todos eles o valor de b e c foram escolhidos de forma que nenhuma requisição fosse rejeitada.

Para os testes 1, 2 e 3, respectivamente, foram medidas as seguintes vazões durante todo o tempo de simulação: 40, 400 e 4000 requisições por segundo. Estes são exatamente os valores esperados, uma vez que para os testes 1, 2 e 3, respectivamente, são exatamente estas as taxas máximas de serviço dos r servidores ativos.

Outros testes também foram realizados com o objetivo de verificar se a vazão se torna exatamente igual à taxa de serviço quando $\lambda < \frac{r}{d}$. Na Tabela D.2 são apresentados os valores usados em tais testes.

Para os testes 4, 5 e 6 respectivamente obteve-se uma vazão de 35 rps, 350 rps e 3500 rps durante todo o tempo simulado, valores exatamente iguais à taxa de chegada de requisições, conforme esperado.

Os testes apresentados nesta subseção foram usados para testar outros limites, conforme será visto nas subseções a seguir.

Tabela D.2: Valores usados nos testes 4, 5 e 6 de limite máximo da vazão

Teste 4	*Teste 5*	*Teste 6*
$\lambda = 35$ rps	$\lambda = 350$ rps	$\lambda = 3500$ rps
$d = 0,050$	$d = 0,050$ s	$d = 0,050$ s
$r = 2$	$r = 20$	$r = 200$
$b = 6000$	$b = 6000$	$b = 6000$
$c = 500$	$c = 500$	$c = 500$
$\frac{r}{d} = 35$	$\frac{r}{d} = 350$	$\frac{r}{d} = 3500$

D.1.3 Testes de limites superior e inferior do tempo de resposta

A quantidade máxima de requisições presentes em um nó caracteriza um limite máximo para o tempo de resposta da seguinte forma. Seja a uma requisição que é aceita e ocupa inicialmente o último lugar da fila de *backlog*. Sendo assim, existem no servidor c requisições e no *backlog* b requisições. Se cada requisição leva um tempo d para ser processada completamente, então a requisição a permanecerá no sistema por um tempo igual a $(b+c) \times d$. Nenhuma requisição poderá permanecer por mais tempo no sistema, pois o lugar que a ocupou inicialmente é o pior caso para qualquer requisição aceita. Assim, espera-se que o tempo de resposta cresça à medida que as filas aumentam, mas não ultrapasse o limiar $(b + c) \times d$ depois que todas as filas de *backlog* estiverem cheias.

Os testes 7, 8 e 9 descritos na Tabela D.3 foram realizado para estudar limites superiores do tempo de resposta. Estes testes são bastante semelhantes aos testes 1, 2 e 3, diferindo apenas no tamanho da fila de *backlog* (b) e na capacidade de processamento simultâneo dos nós (c). Os valores de b e c foram modificados de forma que as filas ficassem cheias antes do término da simulação.

Tabela D.3: Valores usados nos testes de limite máximo do tempo de resposta

Teste 7	*Teste 8*	*Teste 9*
$\lambda = 50$ rps	$\lambda = 500$ rps	$\lambda = 5000$ rps
$d = 0,050$ s	$d = 0,050$ s	$d = 0,050$ s
$r = 2$	$r = 20$	$r = 200$
$b = 1024$	$b = 1024$	$b = 1024$
$c = 250$	$c = 250$	$c = 250$
$(b + c) \times d = 63,7$	$(b + c) \times d = 63,7$	$(b + c) \times d = 63,7$

Para estes valores o tempo de resposta não pode ultrapassar 63,7 segundos. Todas as requisições atendidas a partir do sétimo minuto permaneceram no sistema por 63,69

segundos, um valor praticamente igual ao limite teoricamente estabelecido e este limite nunca foi ultrapassado.

O limite inferior do tempo de resposta deve ser igual à demanda de serviço das requisições. Os testes 4, 5 e 6 (ver Tabela D.2) foram usados para realizar este teste. Nestes experimentos filas nunca são formadas, uma vez que a taxa de chegada é menor que a taxa máxima de serviço dos nós. Observou-se em todos os resultados que o tempo de resposta era exatamente o mesmo, e igual a 50 ms, conforme esperado.

D.1.4 Teste de limites superior e inferior da utilização

Os testes 1, 2 e 3 cujos parâmetros estão apresentados na Tabela D.1 foram realizados para testar o limite superior da utilização. A taxa de chegada de requisições é suficientemente grande para manter os servidores ativos completamente ocupados durante todo o tempo de simulação.

A utilização foi computada para cada intervalo de 10 segundos e a cada minuto as seis últimas utilizações computadas são usadas para se gerar uma média de utilização de todos os nós. Para os teste 1, 2 e 3 respectivamente obteve-se utilização média igual a 100% em todos os intervalos computados.

Para o testar o limite inferior foram realizados os testes 4, 5 e 6 apresentados na Tabela D.2. A utilização média dos nós ativos manteve-se em 87,5% conforme esperado durante todo o intervalo de tempo de simulação estudado.

D.1.5 Testes dos limites superior e inferior da disponibilidade

Os testes 4, 5 e 6 apresentados na Tabela D.2 foram realizados com o intuito de testar o limite máximo da disponibilidade. Nos três testes a disponibilidade computada para cada minuto foi de 100%. Nenhuma requisição foi rejeitada em nenhum momento.

Os testes 10, 11 e 12 apresentados na Tabela D.4 foram realizados com o objetivo de testar o limite mínimo da disponibilidade. Este teste é bastante irreal, mas estressa o simulador de forma que não apenas os limites da disponibilidade, mas também da vazão podem ser testados.

Para a realização destes testes o valor do "quantum" simulado mudou de 100 ms (usado em todos os experimentos) para 600 s. Observou-se que tanto a disponibilidade quanto a vazão foram iguais a zero durante os dez primeiros minutos de simulação. O tempo de simulação foi aumentado para que o décimo primeiro minuto pudesse ser visto.

Na Tabela D.5 são apresentados os valores computados para disponibilidade e vazão da

Tabela D.4: Valores usados nos testes de limite mínimo da disponibilidade

Teste 10	*Teste 11*	*Teste 12*
$\lambda = 50$ rps	$\lambda = 500$ rps	$\lambda = 5000$ rps
$d = 600$ s	$d = 600$ s	$d = 600$ s
$r = 2$	$r = 20$	$r = 200$
$b = 1024$	$b = 1024$	$b = 1024$
$c = 250$	$c = 250$	$c = 250$

aplicação no décimo primeiro minuto de simulação. É neste intervalo que são computadas as saídas com sucesso das primeiras requisições atendidas. Por esta razão, neste intervalo a disponibilidade e a vazão não são nulas.

Tabela D.5: Valores computados no décimo primeiro minuto de simulação

	Teste 10	*Teste 11*	*Teste 12*
Disponibilidade	$6,667 \times 10^{-4}\%$	$6,667 \times 10^{-4}\%$	$6,667 \times 10^{-4}\%$
Vazão	0,033 rps	0,333 rps	3,33 rps

Todos os valores computados para disponibilidade e vazão foram exatamente os valores esperados.

D.2 Experimentos de medição para validação do modelo

Para validar o modelo foram realizados experimentos de medição em um ambiente de testes dedicado para tal finalidade.

Inicialmente, cogitou-se a possibilidade de usar *benchmarks* de confiabilidade para realizar esta validação. *Benchmarks* de confiabilidade são processos que testam os atributos de confiabilidade de um sistema. Eles oferecem meios de caracterizar quantitativamente o comportamento de sistemas e componentes na presença de faltas (DURãES; VIEIRA; MADEIRA, 2004). Depois de pesquisar sobre o assunto, foi possível concluir que tais *benchmarks* ainda não estão padronizados de fato (CONSTANTINESCU et al., 2005) e, por conseqüência, a aplicação de tais *benchmarks* não tornaria mais simples a atividade de testes experimentais. Portanto, apesar de *benchmarks* serem uma estratégia poderosa de teste, optou-se por não utilizar uma *benchmark* específica.

De fato, pretende-se testar aqui a aplicação e os sistemas de gerência na presença das falhas de desempenho. Recentemente, *benchmarks* para sistemas autônomos também

começaram a ser estudadas. As *benchmarks* de confiabilidade podem ser consideradas como um tipo de *benchmark* de sistemas autônomos (*autonomic benchmark*) (BROWN et al., 2004). Assim como para *benchmarks* de confiabilidade, não existe ainda uma padronização, mas seu objetivo geral é permitir avaliar progressos no campo da computação autônoma (BROWN et al., 2004). Se tais *benchmarks* existissem, seria interessante usá-las uma vez que este trabalho de tese propõe mecanismos para aumentar o nível de autonomia de *e-services*. Porém, sendo a pesquisa nesta área ainda bastante incipiente, optou-se por também não usar tal abordagem.

Nas subseções a seguir o ambiente experimental de medições é descrito com maiores detalhes.

D.2.1 Ambiente experimental de medição

O ambiente de testes experimental está ilustrado na Figura D.2.1.

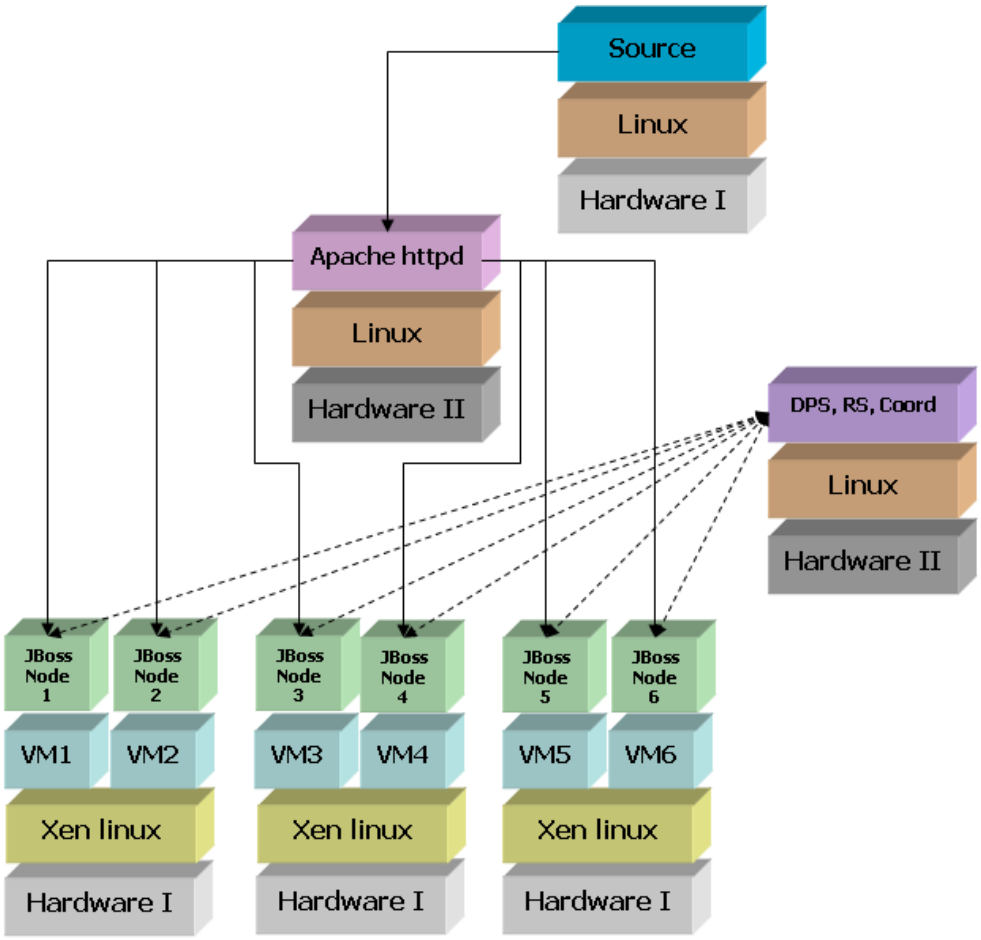

Figura D.1: Arquitetura de alto nível do ambiente de testes experimentais

É possível identificar nesta figura máquinas realizando os seguintes papéis:

- *Fonte geradora de tráfego (Source).* A fonte geradora de tráfego envia requisições para o balanceador de carga, como se fosse um cliente da aplicação. A fonte geradora de tráfego é uma aplicação Java que lê um arquivo de *trace* gerado pela ferramenta GEIST (KANT; TEWARI; IYER, 2001). Cada linha do arquivo de trace representa uma requisição e contém o tempo passado entre a chegada desta requisição e da anterior e o tempo de serviço demandado pela requisição em questão. Esta fonte geradora de tráfego armazena em um arquivo de saída o tempo de resposta e o código da resposta recebida, sendo possível calcular a disponibilidade e o tempo de resposta médio da aplicação;

- *Balanceador de carga.* O balanceador de carga recebe as requisições do cliente e as repassa para um dos nós ativos. O software utilizado para realizar esta função foi o Apache Httpd versão 2.2 habilitado com os módulos *mod_proxy* e *mod_proxy_balancer* e configurado para realizar *proxing* reverso (balanceamento de carga) por requisição (*round robin*);

- *Nós que executam a aplicação (Node1 a Node6).* Estes nós recebem requisições do balanceador de cargas, processam estas requisições e enviam a resposta para o balanceador de carga. Nestes nós foi instalado o JBoss 4.0 onde foi implantada uma aplicação Servlet chamada AgingApplication. Esta aplicação recebe requisições contendo demandas de serviço (em milissegundos) e realiza tarefas que ocupam a CPU por este período aproximado de tempo. Além disso, assim como a aplicação simulada, é possível que exista um fator de envelhecimento configurável que é acrescentado ao tempo de serviço e que aumenta à medida que mais requisições vão sendo processadas pelo nó. Esta aplicação foi implementada especificamente para a realização destes experimentos de medição;

- *Entidade de gerência.* Em uma máquina separada foram instalados sistemas de gerência que são capazes de monitorar e controlar a aplicação em cada um dos seis nós que podem executá-la:

  - *Sistema de provisão de recursos.* É o sistema responsável por decidir sobre a quantidade de nós que deve estar ativa ao longo dos experimentos de medição. Dois sistemas distintos foram implementados:

    - *Sistema de provisão dinâmica de recursos (DPS).* O sistema de provisão dinâmica de recursos decide sobre a quantidade de nós que deve estar ativa ao

longo do tempo durante a realização dos experimentos de medição. O sistema usado implementa o método de provisão dinâmica proposto em (RANJAN et al., 2002);

  - *Sistema de provisão estática de recursos (SPS).* O sistema de provisão estática ativa uma quantidade estática de nós que deve ficar ativa durante todo o experimento de medição.

- *Sistema de rejuvenescimento de software (RS).* Este sistema monitora cada nó ativo e rejuvenesce aqueles que apresentam disponibilidade ou tempo de resposta inferior a um dado limiar por um dado período de tempo. O rejuvenescimento é realizado no nível de *middleware*, isto é, o rejuvenescimento consta de terminar as requisições presentes no nó e reiniciar o JBoss;
- *Sistema de provisão dinâmica de recursos coordenado.* É o mesmo sistema de provisão dinâmica apresentado anteriormente, porém com algumas modificações que permitem a coordenação entre as tarefas deste sistema e do sistema de rejuvenescimento.
- *Sistema de rejuvenescimento de software coordenado.* É o mesmo sistema de rejuvenescimento anteriormente apresentado, com algumas modificações que permitem a coordenação entre o sistema de rejuvenescimento e de provisão dinâmica.

O cenário de testes contou com seis máquinas. Destas, quatro eram do tipo I descrito na Tabela D.6 e duas eram do tipo II descrito na Tabela D.7.

Tabela D.6: Configurações de hardware das máquinas tipo I usadas nos experimentos de medição

Descrição	*Configuração*
Número de CPUs	2
Configuração de cada CPU	Pentium 4, 3,00GHz
Memória	1 GB

O sistema Xen de paravirtualização (BARHAM et al., 2003) foi usado para criar 2 máquinas virtuais em 3 máquinas do tipo I. Os nós que executam efetivamente a aplicação estão executando sobre máquinas virtuais criadas e gerenciadas pelo Xen Linux sobre máquinas do tipo I. Como haviam duas CPUs em cada uma destas máquinas, configurou-se o Xen para usar CPUs distintas em máquinas virtuais distintas. Cada máquina virtual foi configurada para usar 380 MB de memória RAM.

Tabela D.7: Configurações de hardware das máquinas tipo II usadas nos experimentos de medição

Descrição	*Configuração*
Número de CPUs	1
Configuração de cada CPU	Pentium 4, 2,80GHz
Memória	1 GB

D.2.2 Faltas de software inseridas

Faltas de software são inseridas em um dos nós ativos no início de alguns experimentos e recuperadas através do sistema de rejuvenescimento ao longo dos experimentos. A aplicação *agingApplication* em um dos nós já inicia com desempenho bastante degradado, o que garante o rejuvenescimento ao longo do experimento. Cada requisição recebida por este nó demora mais do que o esperado para ser processada, levando o nó a uma falha de desempenho.

Como cada requisição está associada a um tempo de processamento, experimentos foram realizados com o intuito de identificar uma função que demande uma média de um milissegundo do processador.

A função apresentada a seguir foi usada. Experimentos realizados demonstraram que uma execução desta função nas máquinas virtuais usadas requer um tempo aproximado de processamento de um milissegundo.

```
for( double x = -1.9; x < 1.9; x+=0.001 ) {
        double numerador = Math.pow( Math.E, Math.pow(x, 2) * (-0.5) );
        double denominador = Math.sqrt(2 * Math.PI);
        double fx = numerador / denominador;
}
```

Esta função foi executada dez mil vezes isoladamente em cada um dos seis nós. Cada execução teve o tempo de processamento computado e os intervalos de 98% confiança destes tempos em cada nó são apresentados na Tabela D.8.

Quando uma requisição demanda, por exemplo, dez milissegundos, então a função indicada no algoritmo anteriormente apresentado será executada dez vezes.

Tabela D.8: Resultados dos experimentos realizados para identificar a função de um milissegundo

	IC(98%;ε=1,5%)
Node1	[0,99, 1,00]
Node2	[0,98, 1,01]
Node3	[0,98, 1,00]
Node4	[0,99, 1,01]
Node5	[0,98, 1,00]
Node6	[0,99, 1,01]

D.2.3 Medições realizadas

As medições indicam a qualidade de serviço e o custo do sistema sob teste na presença de falhas ao realizar a carga de trabalho. Três medições foram realizadas: disponibilidade, tempo médio de resposta e quantidade média de nós usados.

A disponibilidade média foi computada como sendo o percentual de requisições processadas com sucesso durante o experimento.

O tempo médio de resposta foi calculado como sendo a média aritmética de todas as respostas respondidas com sucesso pela aplicação durante o experimento. Entende-se que uma resposta foi respondida com sucesso quando o código HTML de resposta foi o código 200 (OK).

O número médio de nós usado durante os experimentos onde o sistema de provisão dinâmica de recursos executou é a média aritmética do número de nós ativos em cada intervalo de adaptação do DPS.

D.2.4 Monitores e atuadores do experimento de medição

Os sistemas de gerência precisam de monitores que coletem informações usadas para a tomada de decisão, bem como atuadores, responsáveis por atuar na aplicação gerenciada para garantir que as decisões tomadas serão realizadas de fato.

Os monitores do DPS são dois. Um deles é um script shell que se comunica com a fonte de tráfego para recuperar informações tais como quantidade de requisições enviadas e processadas no fim de cada intervalo de adaptação. O outro é o monitor de utilização de memória de cada nó ativo. Este também um script shell que usa como ferramenta de monitoração o xenmon.py (GUPTA; GARDNER; CHERKASOVA, 2005) e envia para o gerente DPS a utilização medida no último minuto. O DPS possui 2 atuadores. Um deles é

responsável por iniciar a aplicação, migrando o nó do estado inativo para o ativo. A única função deste atuador é iniciar o JBoss. O outro atuador do DPS é responsável por desativar um nó. Este atuador retira o nó em questão da lista de nós conhecidos pelo balanceador de cargas para garantir que nenhuma nova requisição é recebida enquanto as requisições já presentes no nó são processadas. Ele aguarda enquanto todas as requisições são atendidas e depois para o JBoss no nó em questão. Finalmente o nó desativado é reinserido na lista de nós conhecidos pelo balanceador de cargas.

O RS possui sondas que enviam requisições de teste para a aplicação monitorada de tempos em tempos, com o intuito de identificar a disponibilidade e o tempo de resposta da aplicação. O RS tem um atuador que é capaz de rejuvenescer a aplicação. Primeiro o nó a ser rejuvenescido é retirado da lista de nós do balanceador de cargas. Depois este atuador aguarda enquanto o nó termina de processar as requisições já presentes nele. Finalmente, depois que todas as requisições são atendidas o JBoss do nó em questão é reiniciado e reinserido na lista de nós conhecidos pelo balanceador de cargas.

D.2.5 Configuração e procedimentos experimentais

Esta seção dedica-se a apresentar os cenários de testes experimentais realizados.

A carga de trabalho contém as tarefas que o sistema sob teste tem que realizar. Duas cargas de trabalho distintas foram geradas com a ferramenta GEIST (KANT; TEWARI; IYER, 2001) e usadas nos experimentos de medição. Ao aplicar uma das cargas, apresentada na Figura D.2, foi possível estudar casos em que o rejuvenescimento ocorre em um momento de carga alta e crescente. Ao aplicar a segunda carga, apresentada na Figura D.3, foi possível estudar o rejuvenescimento que ocorre durante um momento em que a carga está baixa e diminuindo. Estas cargas são chamadas de *up* e *down* respectivamente no decorrer deste texto.

Quatro cenários experimentais distintos foram testados para cada uma das cargas anteriormente apresentadas:

- *AWoF+DPS.* Neste cenário todos os nós da aplicação estão saudáveis e o sistema de provisão dinâmica de recursos foi usado para gerenciar a quantidade de nós ativos ao longo dos experimentos.

- *AWF+RS.* Neste cenário, um dos nós inicia com o desempenho degradado, de forma que este nó é rejuvenescido uma vez durante o experimento pelo sistema de rejuvenescimento. A provisão de recursos é estática e todos os seis nós ficam ativos durante todo o experimento.

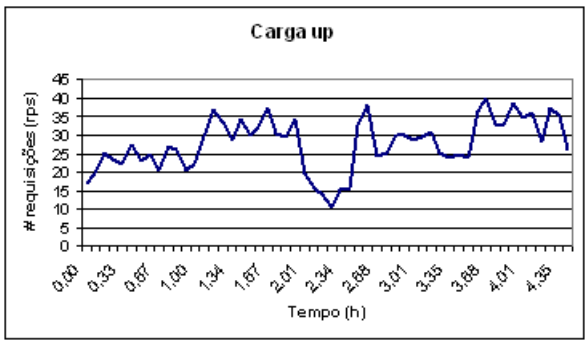

Figura D.2: Gráfico da carga *up* usada nos experimentos

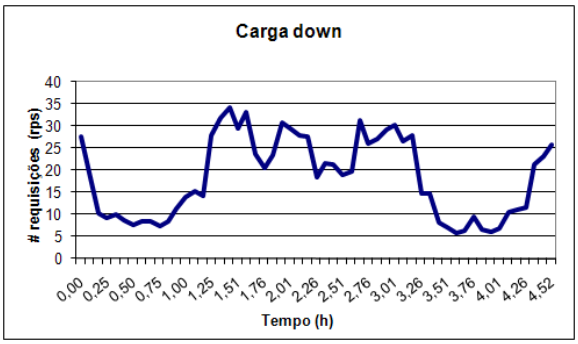

Figura D.3: Gráfico da carga *down* usada nos experimentos

- *AWF+DPS+RS.* Neste cenário os sistemas de provisão dinâmica de recursos e de rejuvenescimento atuam sobre a aplicação sem coordenação entre suas ações. Um dos nós inicia com o desempenho degradado, de forma que ele é rejuvenescido durante o experimento.

- *AWF+DPS*RS.* Cenário idêntico ao anterior, porém há coordenação entre o DPS e o RS.

Quando um nó está em falha, configurou-se que ele leva trezentos milissegundos a mais para processar uma requisição se comparado aos demais nós.

O sistema de provisão estática inicia os seis nós no início do experimento e só os desativa depois que toda a carga de trabalho foi realizada.

O sistema de provisão dinâmica de recursos ativa inicialmente três ou cinco nós para as cargas *up* e *down* respectivamente. Depois, ao longo do experimento, este sistema utiliza-se do algoritmo de provisão dinâmica proposto em (RANJAN et al., 2002) para decidir sobre a quantidade de nós a ser ativado em cada intervalo de adaptação. Cada intervalo de adaptação foi configurado como sendo cinco minutos e a utilização alvo perseguida pelo DPS foi configurada como sendo 58%.

O sistema de rejuvenescimento reinicia nós que permaneceram com a disponibilidade ou o tempo de resposta aquém ou além dos valores limiares para estas variáveis durante 3 intervalos de monitoração consecutivos do sistema de rejuvenescimento. Cada intervalo foi configurado como sendo de 5 minutos. O limiar de disponibilidade considerado foi 99,99% e o limiar de tempo de resposta foi três segundos.

D.3 Instanciações do modelo de simulação

Os valores usados para instanciar o modelo de simulação são apresentados na Tabela D.3.

Tabela D.9: Parâmetros configurados para os experimentos de simulação

Parâmetro	*valor*
Carga de trabalho	Cargas up e down apresentadas nas Figuras D.2 e D.3 respectivamente
Número máximo de nós ativos	6. O mesmo valor configurado nos experimentos de medição
Número mínimo de nós ativos	1. O mesmo valor configurado nos experimentos de medição
Tamanho da fila de backlog	1024. O mesmo tamanho default do Linux
Capacidade de processamento simultâneo	250. O mesmo valor configurado para o parâmetro MaxThreads do JBoss, que determina o número máximo de requisições que podem ser processadas simultaneamente.
Intervalo de adaptação do DPS e do RS	5 minutos. O mesmo valor configurado nos experimentos de medição
Quantidade de nós em falha no início dos experimentos onde o RS atua	1. O mesmo valor configurado nos experimentos de medição.
Envelhecimento do nó em falha	300 milissegundos. O mesmo valor configurado nos experimentos de medição.
Número inicial de nós ativos quando o DPS atua	3 para a carga *up* e 5 para a carga *down*
Tempo médio de migração dos nós	34111 ms. Testes foram realizados com os nós usados nos experimentos e observou-se que este era o tempo médio de inicialização do JBoss em tais máquinas. A variância foi de 15 s.
Tempo médio de reinicialização dos nós	40111 ms. Experimentos demonstraram que o JBoss leva, em média, 6 segundos para parar nos nós usados nos experimentos. Este valor foi somado ao tempo médio de migração.

Número de intervalos consecutivos em que um nó apresenta QoS aquém do esperado antes de ser rejuvenescido	3. O mesmo valor configurado nos experimentos de medição

Os mesmos cenários medidos foram também simulados e os resultados de sua comparação são apresentados na seção a seguir.

D.4 Resultados dos experimentos de medição

Nesta seção são apresentados os resultados dos experimentos de medição e comparados com os resultados de simulações correspondentes.

Nove experimentos de medição foram executados para os cenários em que há inserção de faltas. Os cenários *AWoF+DPS* foram executados 5 vezes. No total, foram 64 experimentos. Os experimentos de simulação foram executados cinco vezes cada um, exceto o cenário *AWF+DPS+RS* para a carga *down*, que foi realizado 150 vezes para que o intervalo de 95% de confiança da disponibilidade pudesse ser obtido.

Nas subseções a seguir cada cenário é avaliado individualmente.

D.4.1 Avaliação dos experimentos *AWoF+DPS*

Oito experimentos desta composição foram realizados para cada uma das cargas *up* e *down*. Nas Tabelas D.10 e D.11 são apresentados os intervalos de de confiança identificados para a disponibilidade, tempo de resposta e número médio de nós ativos na composição *AWoF+DPS*, quando as cargas *up* e *down* foram aplicadas.

Tabela D.10: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação para a carga *up*

	Disponibilidade	T. R. (s)	# nós ativos
Medição	$99,998\% \pm 0,002\%$	$3,04 \pm 0,12$	$3,64 \pm 0,06$
Simulação	$100,000\% \pm 0,000\%$	$1,29 \pm 0,01$	$3,21 \pm 0,00$

Para a obtenção do intervalo de 95% de confiança da disponibilidade nos experimentos de medição foi considerado um erro de 0,005%. O tempo de resposta apresenta uma variabilidade bem maior e para gerar o intervalo de 50% de confiança considerou-se um erro de 10%. Para o número médio de nós ativos o intervalo de 95% de confiança foi gerado considerando um erro de

Tabela D.11: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação para a carga *down*

	Disponibilidade	T. R. (s)	# nós ativos
Medição	$99,995\% \pm 0,005\%$	$2,74 \pm 0,12$	$2,60 \pm 0,08$
Simulação	$100,000\% \pm 0,000\%$	$0,73 \pm 0,01$	$2,28 \pm 0,00$

6% quando a carga *up* foi aplicada. Já quando a carga *down* foi aplicada o intervalo de 90% de confiança considerou um erro de 8%.

Para obter os intervalos de 95% confiança da disponibilidade, tempo de resposta e número médio de nós ativos dos experimentos de simulação foram considerados respectivamente erros de 0,02%, 2% e 2%.

Observa-se que, em termos de disponibilidade, os resultados obtidos via simulação e via medição para o cenário *AWoF+DPS* foram bastante semelhantes. Isto indica que, com os parâmetros usados e livre de falhas, o modelo de simulação e de medição se comportam de forma bastante semelhante.

Os tempos de resposta do modelo de medição foram maiores que o tempo de resposta do modelo de simulação. Este resultado já era esperado, uma vez que o modelo de simulação não considera custos de comunicação e outros processamentos que não sejam o das requisições da aplicação.

Além dos custos já esperados, ainda existe o adicional por conta da virtualização usada. De acordo com (APPARAO; MAKINENI; NEWELL, 2006) *et al.* a virtualização trouxe mudanças significantes para a comunicação em rede. Os pacotes precisam passar por camadas adicionais de processamento, que introduzem custos substanciais. No modelo de simulação não foi adicionado custo algum de transmissão de requisições e respostas pela rede. Além disso, as CPUs das máquinas precisam ainda ser compartilhadas com o domínio 0 (supervisor) que precisa gerenciar os domínios virtuais visitantes.

As tendências de tempo de resposta, no entanto, se mantém, pois observou-se que em ambos os modelos os tempos de resposta foram menores quando a carga *down* foi aplicada.

Em termos de número médio de nós ativos observou-se que o modelo de simulação usou uma quantidade um pouco menor de nós. As utilizações médias do modelo de simulação foram, em geral, menores que as utilizações médias do modelo de medição, o que justifica o DPS ter escolhido uma quantidade um pouco menor de máquinas em diversos momentos ao longo dos experimentos. Como já citado, nas medições a CPU não fica completamente dedicada à aplicação como o é nas simulações.

D.4.2 Avaliação dos experimentos *AWF+RS*, *AWF+DPS+RS* e *AWF+DPS*RS*

Apresentação dos resultados experimentais da composição *AWF+RS*

Dez experimentos da composição *AWF+RS* foram realizados para cada uma das cargas *up* e *down*. Nas Tabelas D.12 e D.13 são apresentados os intervalos de confiança identificados para a disponibilidade, tempo de resposta e número médio de nós ativos na composição *AWF+RS*, quando as cargas *up* e *down* foram aplicadas.

Tabela D.12: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação na composição *AWF+RS* ao aplicar a carga *up*

	Disponibilidade	T. R. (s)	# nós ativos
Medição	$99,685\% \pm 0,030\%$	$0,99 \pm 0,07$	6 ± 0
Simulação	$100,000\% \pm 0,000\%$	$0,87 \pm 0,00$	6 ± 0

Tabela D.13: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação na composição *AWF+RS* ao aplicar a carga *down*

	Disponibilidade	T. R. (s)	# nós ativos
Medição	$99,744\% \pm 0,034\%$	$0,854 \pm 0,06$	6 ± 0
Simulação	$100,000\% \pm 0,000\%$	$2,07 \pm 0,00$	6 ± 0

Para a obtenção do intervalo de 95% de confiança da disponibilidade nos experimentos de medição foi considerado um erro de 0,005%. Para gerar o intervalo de 95% de confiança do tempo de resposta considerou-se um erro de 2%. O número médio de nós ativos foi sempre seis durante todo o experimento.

Para obter os intervalos de 95% confiança da disponibilidade, tempo de resposta e número médio de nós ativos dos experimentos de simulação foram considerados respectivamente erros de 0,02%, 2% e 2%.

Apresentação dos resultados experimentais da composição *AWF+DPS+RS*

Quatorze experimentos da composição *AWF+DPS+RS* foram realizados para cada uma das cargas aplicadas. Nas Tabelas D.14 e D.15 são apresentados os intervalos de confiança identificados para a disponibilidade, tempo de resposta e número médio de nós ativos na composição *AWF+DPS+RS*, quando as cargas *up* e *down* foram aplicadas.

Tabela D.14: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação na composição *AWF+DPS+RS* ao aplicar a carga *up*

	Disponibilidade	T. R. (s)	# nós ativos
Medição	$99,600\% \pm 0,035\%$	$4,20 \pm 0,18$	$3,77 \pm 0,13$
Simulação	$99,580\% \pm 0,000\%$	$3,25 \pm 0,02$	$3,33 \pm 0,00$

Tabela D.15: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação na composição *AWF+DPS+RS* ao aplicar a carga *down*

	Disponibilidade	T. R. (s)	# nós ativos
Medição	$99,697\% \pm 0,025\%$	$4,46 \pm 0,24$	$2,67 \pm 0,04$
Simulação	$99,897\% \pm 0,006\%$	$3,51 \pm 0,12$	$2,40 \pm 0,05$

Para a obtenção do intervalo de 95% de confiança da disponibilidade nos experimentos de medição foi considerado um erro de 0,03%. Para gerar o intervalo de 50% de confiança do tempo de resposta considerou-se um erro de 15% quando a carga *up* foi aplicada e 20% quando a carga *down* foi aplicada. Já o intervalo de 95% de confiança do número de nós ativos foi gerado considerando-se um erro de 12% quando a carga *up* foi aplicada e 4% quando a carga *down* foi aplicada.

Para obter os intervalos de 95% confiança da disponibilidade, tempo de resposta e número médio de nós ativos dos experimentos de simulação foram considerados respectivamente erros de 0,008%, 12% e 2%.

Apresentação dos resultados experimentais da composição *AWF+DPS*RS*

Nove experimentos da composição *AWF+DPS*RS* foram realizados para cada uma das cargas *up* e *down*. Nas Tabelas D.16 e D.17 são apresentados os intervalos de confiança identificados para a disponibilidade, tempo de resposta e número médio de nós ativos na composição *AWF+DPS*RS* quando as cargas *up* e *down* foram aplicadas respectivamente.

Tabela D.16: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação na composição *AWF+DPS*RS* ao aplicar a carga *up*

	Disponibilidade	T. R. (s)	# nós ativos
Medição	$99,655\% \pm 0,046\%$	$4,55 \pm 0,16$	$3,63 \pm 0,13$
Simulação	$99,580\% \pm 0,000\%$	$3,26 \pm 0,02$	$3,30 \pm 0,00$

Para a obtenção do intervalo de 95% de confiança da disponibilidade nos experimentos de medição foi considerado um erro de 0,05%. Para gerar o intervalo de 50% de confiança do tempo

Tabela D.17: Intervalos de confiança da disponibilidade, do tempo de resposta e do número de nós ativos da aplicação na composição *AWF+DPS*RS* ao aplicar a carga *down*

	Disponibilidade	T. R. (s)	# nós ativos
Medição	$99,791\% \pm 0,055\%$	$3,09 \pm 0,10$	$2,69 \pm 0,04$
Simulação	$100,000\% \pm 0,000\%$	$1,48 \pm 0,00$	$2,33 \pm 0,00$

de resposta considerou-se um erro de 16% para a experimentos onde a carga *up* foi aplicada e 11% quando a carga *down* foi aplicada. Já o intervalo de 95% de confiança do número de nós ativos foi gerado considerando-se um erro de 8%.

Para obter os intervalos de 95% confiança da disponibilidade, tempo de resposta e número médio de nós ativos dos experimentos de simulação foram considerados respectivamente erros de 0,008%, 2% e 2%.

Análise dos resultados

Excetuando-se os cenários *AWF+DPS+RS* e *AWF+DPS*RS* onde a carga *up* foi aplicada, observou-se que a disponibilidade da aplicação foi sempre maior no modelo de simulação. Excetuando-se o cenário *AWF+RS* onde a carga *down* foi aplicada, os tempos de resposta foram menores no modelo de simulação. O número médio de nós ativos foi sempre menor no modelo de simulação.

Apesar destas diferenças mencionadas, as tendências em termos de disponibilidade em ambos os modelos são semelhantes (vide Tabelas D.18 e D.19).

Tabela D.18: Tendências de disponibilidade dos modelos de simulação e de medição onde a carga *up* foi aplicada

	Modelo de medição	*Modelo de simulação*
Diferença entre *AWF+RS* e *AWF+DPS+RS*	0,086%	0,420%
Diferença entre *AWF+DPS+RS* e *AWF+DPS*RS*	0,055%	0,000%

Tabela D.19: Tendências de disponibilidade dos modelos de simulação e de medição onde a carga *down* foi aplicada

	Modelo de medição	*Modelo de simulação*
Diferença entre *AWF+RS* e *AWF+DPS+RS*	0,048%	0,103%
Diferença entre *AWF+DPS+RS* e *AWF+DPS*RS*	0,095%	0,103%

Com o intuito de simplificar a análise, considera-se aqui que $l_{DPS} = 0$. De fato, o *DPS* mostrou-se bastante eficiente, sendo o número de rqeuisições rejeitadas sempre bastante pequeno

(poucas unidades ou poucas dezenas) na composição *AWoF+DPS*. Por esta razão, os resultados da composição *AWF+DPS+RS* estão sendo comparados diretamente com os resultados da composição *AWF+RS* para a métrica disponibilidade.

Em ambos os modelos o cenário *AWF+DPS+RS* apresenta disponibilidade pior, se comparados aos cenários *AWF+RS*. Quando a carga *up* é aplicada, esta diferença é maior do que quando a carga *down* é aplicada. Quando a carga *down* é aplicada, ocorre em alguns momentos que o DPS escolhe aleatoriamente o nó em falha para ser desativado. Esta aleatoriedade torna muito alta a variância das disponibilidades e tempos de resposta. Tanto nos experimentos quanto nas simulações foi necessária uma quantidade maior de experimentos para que o intervalo de 95% de confiança fosse gerado.

Em ambos os modelos a disponibilidade da aplicação é melhor no cenário AWF+RPS*RS quando a carga *down* é aplicada. Quando a carga *up* é aplicada, apesar da média das disponibilidades medidas terem sido maiores quando houve coordenação, não se pode dizer que estes valores são diferentes, pois os intervalos de 95% de confiança correspondentes se sobrepõem.

Os tempos de resposta também apresentam tendências semelhantes em ambos os modelos. O tempo de resposta dos cenários *AWF+DPS+RS* foi em ambos os modelos sempre pior que o pior tempo de resposta entre os cenários *AWoF+DPS* e *AWF+RS*. O tempo de resposta médio do cenário *AWF+DPS*RS* com carga *up* no modelo de simulação foi em média igual ao tempo de resposta do modelo *AWF+DPS+RS*, enquanto que no modelo de medição foi maior. Os intervalos de 50% de confiança, no entanto, se sobrepõem, não sendo possível afirmar alguma diferença. Para a carga *down* o tempo de resposta é melhor quando há coordenação entre os sistemas.

Finalmente, observa-se que o número médio de nós ativos no modelo de simulação é sempre um pouco menor que o número médio de nós ativos no modelo de simulação. A razão desta pequena diferença já foi discutida na seção anterior.

Em todos os resultados apresentados nesta tese, enfatizou-se especialmente estas diferenças discutidas nesta seção. Os valores quantitativos em si não foram alvo de nenhuma análise. Sendo assim, considera-se que o modelo de simulação é fiel ao modelo de medição.

Apesar de já considerar, através destes experimentos de medição, o modelo de simulação validado, esforços foram realizados no sentido de identificar as razões destas diferenças. De fato, observou-se claramente que na grande maioria dos casos em que um nó inicia o experimento em falha a qualidade de serviço da aplicação nos experimentos de medição foi pior que a qualidade de serviço da aplicação nas simulações. Três razões foram encontradas. Elas são discutidas na subseção a seguir e em seguida novas simulações com novos parâmetros que visam diminuir estas diferenças foram realizadas e analisadas.

Comportamento de nós em falha nas medições e simulações

A primeira grande diferença entre o modelo de simulação e de medição é que, na medição, o intervalo de reconfiguração do *RS* é maior que na simulação. Afinal, terminado o intervalo de monitoração, o *RS* ainda precisa se comunicar com os monitores para requisitar informações, depois ele vai executar o algoritmo e decidir pelo rejuvenescimento ou não de cada nó ativo. Na simulação estas tarefas não levam tempo algum, porém nos experimentos de medição estas atividades levam algum tempo para ocorrer. Observou-se que o RS leva, em média, 1,5 minutos para realizar estas atividades. Isto contribui para a uma qualidade de serviço pior nos experimentos de medição já que o nó em falha vai passar mais tempo antes de ser rejuvenescido.

Uma segunda razão para uma qualidade de serviço pior da aplicação nos experimentos de medição é que, nestes, o nó com desempenho degradado apresenta uma qualidade de serviço aquém do nó com desempenho degradado na simulação. Os seguintes testes foram realizados exatamente iguais para a simulação e medição:

- *Carga de trabalho.* Taxa de chegada constante de dez requisições por segundo (uma requisição a cada 100 milissegundos), com cada requisição associada a uma demanda simbólica de um milissegundo. A duração desta carga de trabalho é de 80 segundos, totalizando 800 requisições;
- *Nós da aplicação.* Apenas um nó ativo, com degradação de desempenho de 300 milissegundos;

Na simulação o tamanho da fila de *Backlog* foi configurado como sendo 1024 e a quantidade de requisições que podem ser servidas simultaneamente foi de 250.

Com esta carga de trabalho, espera-se da simulação que nenhuma requisição seja perdida. A quantidade de requisições não é suficiente para encher a fila de *Backlog*, assim, os tempos de resposta serão altos, no entanto, nenhuma requisição será perdida.

Como esperado, obteve-se da simulação uma disponibilidade de 100% e um tempo de resposta bastante elevado, de 109 segundos.

Para o experimento correspondente de medição obteve-se uma disponibilidade de apenas 80,5%, e um tempo de resposta de 80 segundos.

Com estes experimentos foi possível demonstrar que, na prática, a degradação de desempenho causa uma piora muito maior da disponibilidade da aplicação do que na simulação. Sabe-se que aplicações sobrecarregas estão mais sucetíveis a falhas (GRIBBLE, 2001), pois estão mais propensas a condições de corrida, mau comportamento de coletores de lixo, etc., aumentando a probabilidade de ocorrência de faltas não determinísticas como Heisenbugs. Este fato não é considerado no modelo de simulação, e, portanto, a disponibilidade da aplicação é maior neste modelo. Da mesma forma, como as filas do modelo de simulação são maiores, os tempos de resposta em alguns casos também o são, como será visto mais adiante.

Finalmente, percebeu-se que o balanceador de cargas começa a diminuir a quantidade de requisições enviadas para o nó com desempenho degradado. Em experimentos *AWF+RS*, por exemplo, o nó lento recebe, antes de ser rejuvenescido, 27% menos requisições que os demais nós quando a carga *up* é aplicada e 16% menos quando a carga *down* é aplicada. Este fato, ao contrário do anterior, contribui para que qualidade de serviço da aplicação seja melhor, no entanto, não é suficiente para balancear a queda de disponibilidade devido às faltas inseridas.

D.4.3 Resultados de simulação para novas configurações

Diante do exposto, decidiu-se realizar novos experimentos de simulação, com configuração de alguns parâmetros modificada de forma a refletir as diferenças anteriormente citadas.

Na Tabela D.20 são apresentados novos valores de alguns parâmetros usados em todos os novos cenários simulados. A outra modificação realizada neste modelo de simulação foi configurar o balanceador de cargas para enviar menos tarefas para o nó em falha. Na Tabela D.21 são apresentadas as configurações deste novo balanceador de cargas para cada um dos cenários analisados.

Tabela D.20: Parâmetros configurados para os experimentos de simulação

Parâmetro	*valor*
Tamanho da fila de backlog	500. Somado à nova capacidade de processamento simultâneo, dá um valor igual à quantidade de requisições média processada com sucesso nos experimentos com um nó em falha descritos anteriormente
Capacidade de processamento simultâneo	150
Número de intervalos consecutivos em que um nó apresenta QoS aquém do esperado antes de ser rejuvenescido	5, para a carga *up* e 4 para a carga *down*. Observou-se que para a carga up e down respectivamente que o RS leva em média 25 minutos e 20 minutos antes de rejuvenescer o nó em falha

Tabela D.21: Quantidade de requisições enviada para o nó em falha antes do rejuvenescimento

Cenário	*valor*

AWF+RS	Quando a carga *up* foi aplicada, os nós em falha receberam 20% menos de requisições que os demais nós antes de ser rejuvenescido. Quando a carga down foi aplicada este valor caiu para 10%. Nos experimentos os valores médios encontrados foram 27% e 16% respectivamente
AWF+DPS+RS e *AWF+DPS*RS*	Quando a carga up foi aplicada, configurou-se o balanceador de cargas para enviar para o nó em falhas 40% menos requisições. Quando a carga down foi usada este valor caiu para 30%. Nas medições os valores medidos foram 48% e 33% respectivamente

Os resultados de disponibilidade para esta nova configuração são apresentados nas Figuras D.4 e D.5. Nestas figuras e nas que seguem, os valores apresentados para medições são repetições dos anteriormente apresentados.

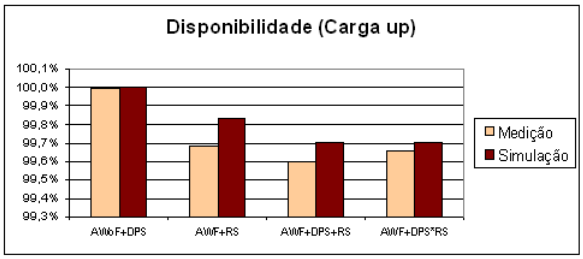

Figura D.4: Disponibilidade da aplicação nos experimentos de medição e simulação onde a carga *up* foi aplicada

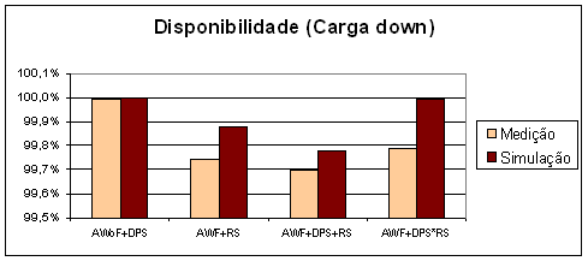

Figura D.5: Disponibilidade da aplicação nos experimentos de medição e simulação onde a carga *down* foi aplicada

Observa-se que a disponibilidade computada nos experimentos de simulação passou a seguir completamente as mesmas tendências que as disponibilidades medidas, sendo, porém, superiores a estas em todos os casos. Houve uma grande redução das disponibilidades computadas nos cenários *AWF+RS*, que antes era 100%.

Os tempos de resposta computados são apresentados nas Figuras D.6 e D.7.

Os tempos de resposta também seguem tendências semelhantes, sendo todos os tempos simulados, exceto no cenário *AWF+RS*, inferiores aos tempos de resposta das medições, conforme já esperado. Como visto em testes anteriores, as simulações oferecem um tempo de resposta maior e uma disponibilidade também maior diante de falhas, uma vez que requisições só deixam de ser atedidas com sucesso devido às filas dos servidores ativos estarem cheias.

Mais uma vez, a qualidade de serviço da aplicação nos cenários *AWF+DPS+RS* foi sempre inferior à qualidade de serviço do pior caso entre *AWoF+DPS* e *AWF+RS*. Além disso, quando

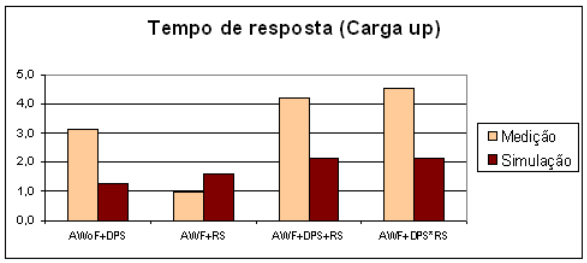

Figura D.6: Tempo de resposta médio da aplicação nos experimentos de medição e simulação onde a carga *up* foi aplicada

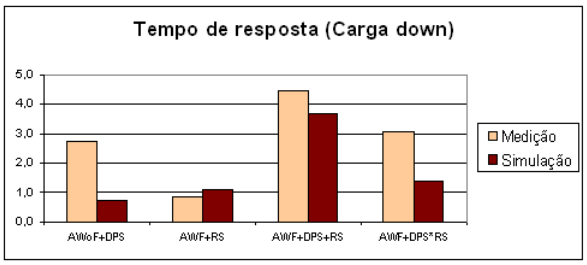

Figura D.7: Tempo de resposta médio da aplicação nos experimentos de medição e simulação onde a carga *down* foi aplicada

a coordenação existia entre DPS e RS a qualidade de serviço aumentou em relação ao cenário *AWF+DPS+RS* correspondente apenas quando a carga *down* foi aplicada.

Em termos de número médio de nós ativos não houve diferenças substanciais após a modificação dos parâmetros de simulação e por esta razão os gráficos foram omitidos.

D.5 Conclusões parciais

Neste capítulo foram descritas algumas atividades realizadas com o objetivo de verificar e validar o modelo de simulação utilizado.

Com os testes de limite aumentou-se a segurança nas medidas computadas no modelo de simulação, tais como disponibilidade, tempo de resposta e utilização, garantindo que limiares bem conhecidos não serão jamais ultrapassados.

Os experimentos de medição, apesar de apresentarem valores pontuais diferentes dos computados pelo modelo de simulação, apresentaram tendências que seguem os mesmos padrões dos encontrados no modelo de simulação. Como no decorrer desta tese os valores em si não foram usados, mas sim as tendências de diferenças entre tais valores, então considera-se que o modelo de simulação usado é adequado para este fim.

Os experimentos de medição, além de validarem o modelo de simulação, permitiram ver que, na prática, a união dos dois sistemas sem coordenação também não é eficiente e que a coordenação é eficiente em alguns casos.