Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Informática

Dissertação de Mestrado

Escalonamento Tolerante a Sabotagem em Grades Computacionais Entre-Pares

Ana Cristina Alves de Oliveira

Campina Grande, Paraíba, Brasil
Agosto - 2007

Escalonamento Tolerante a Sabotagem em Grades Computacionais Entre-Pares

Ana Cristina Alves de Oliveira

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Francisco Vilar Brasileiro

(Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

O49e
2007 Oliveira, Ana Cristina Alves de.
Escalonamento tolerante a sabotagem em grades computacionais entre-pares / Ana Cristina Alves de Oliveira. — Campina Grande: 2007.
87f. : il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Referências.
Orientador: Dr. Francisco Vilar Brasileiro.

1. Sistemas de Processamento Distribuído 2. Escalonamento Adaptativo. 3. Tolerância a Sabotagem. 4. Grades Computacionais. 5. P2P I. Título.

CDU 004.75(043)

"ESCALONAMENTO TOLERANTE A SABOTAGEM EM GRADES COMPUTACIONAIS ENTRE-PARES"

ANA CRISTINA ALVES DE OLIVEIRA

DISSERTAÇÃO APROVADA EM 31.08.2007

PROF. FRANCISCO VILAR BRASILEIRO, Ph.D
Orientador

PROF. MARCO AURÉLIO SPOHN , Ph.D
Examinador

PROF. ANTONIO MARINHO PILLA BARCELLOS, Ph.D
Examinador

CAMPINA GRANDE – PB

"Filho, desde tua mocidade aplica-te à disciplina e até com cabelos brancos encontrarás a sabedoria. Como o lavrador e o semeador, cultiva-a, e espera pacientemente seus bons frutos, porque te cansarás um pouco em seu cultivo, mas em breve comerás de seus frutos."

Eclesiástico 6, 18-20

Resumo

As grades computacionais são uma infra-estrutura para agregar potência computacional. Evoluíram no sentido de formarem comunidades de livre ingresso sobre a Internet e ganharam o título de grades entre-pares (P2P). É importante perceber que quando qualquer usuário pode ingressar e sair livremente de um sistema, este se torna mais suscetível a danos causados por usuários trapaceiros. Uma solução para este problema é aplicar técnicas de tolerância a sabotagem, geralmente baseadas em replicação, para estimar a corretude da computação. Sarmenta verificou que o emprego de técnicas de tolerância a faltas baseada em reputação no escalonamento de tarefas pode proporcionar altos níveis de confiança para os resultados da computação e também minimizar custos com replicação, quando comparados à tradicional técnica de votação para escolha de um resultado correto. Este trabalho tem como objetivo avaliar o uso de heurísticas de escalonamento que se adaptam ao nível de confiança existente para as máquinas em grades P2P, aplicando a técnica de tolerância a faltas proposta por Sarmenta. No entanto, essa técnica pressupõe o conhecimento de propriedades muito difíceis de prever para sistemas de livre ingresso, onde não se conhecem os participantes nem suas intenções. Os resultados obtidos demonstram que não há apenas um método de implementação do escalonamento tolerante a sabotagem. Três heurísticas de escalonamento foram avaliadas e elas apresentaram vantagens e desvantagens, de onde se pôde concluir que, dependendo do conhecimento que se tem sobre a grade, a aplicação de uma heurística pode apresentar um melhor desempenho do que a das demais.

Abstract

The computational grids are infrastructures to agregate computing power. They evolved in the sense of forming free-to-join communities over the Internet and won the title of peer-to-peer (P2P) grids. It is important to notice that when any user can freely join and leave a system, it becomes more succeptible to damages caused by malicious users. A solution to this problem is to apply sabotage tolerance techniques, generally based on replication, to estimate the computation correctness. Sarmenta has verified that the employment of reputation-based fault tolerance techniques in the task scheduling may promote high confidence levels for the computation results, at the same time that minimizes replication costs, when compared to the tradional voting technique to choose a correct result. The objective of this work is to evaluate the usage of scheduling heuristics that adapt themselves to the machines' confidence level in P2P grids, applying the fault tolerance technique proposed by Sarmenta. This technique assumes the knowledge of some properties, which are difficult to foresee. The obtained results demonstrate that there are more than one implementation of sabotage-tolerant scheduling. Three scheduling heuristics were evaluated and they present advantages and disadvantages, concluding that, depending on the grid environment, one heuristic might have a better performance than the others.

Agradecimentos

Agradeço a Deus e a todos a quem Ele concede o dom de cuidar de mim...

Primeiramente, à minha família, mas especialmente aos meus pais Severino e Odete e à minha irmã Karolzinha. Eles sempre me deram exemplos para tudo o que há de bom, incentivando-me e acreditando em mim.

Aos meus professores todos. Amigos e educadores em todos os sentidos. Guiaram-me pela mão ou pelas orelhas e me ajudaram a chegar até aqui. Especialmente aos professores do CEFET-PB (antiga ETFPB) onde estudei o pró-técnico, o curso técnico juntamente com o ensino médio e a graduação. Lembrança especial aos professores Leônidas, meu exemplo, e Kamienski, tão exigente e dedicado, que me orientou na iniciação científica. Destaco meus professores do mestrado na UFCG também. Agradeço, com distinção, ao meu orientador, Fubica, pela paciência que teve comigo e a ajuda para ir até o fim.

Aos amigos do CEFET-PB que, mesmo não estudando mais juntos, nunca perdemos o contato e o carinho. Alguns pude reencontrar na UFCG: Nelson, Robson, Carol, Nazareno, Daniela e Gustavo.

Aos amigos que fiz na UFCG, em especial no LSD. Companheiros de comemorações super engraçadas e trabalhos árduos que nos aproximaram muito. Tenho que ressaltar aqueles que dividiram sala ou projeto comigo no período que passei no LSD e que guardarei sempre com muito carinho: Ayla, Andrey, Lívia, Raquel, Osorinho, Marcelo Iury, Nigini, Fireman, Esther, Elizeu, Lauro, Bruno, Léo e Flávio Peruca.

Aos irmãos que fiz na comunidade católica que participo há sete anos. Contribuiram para que eu procurasse conhecer-me melhor, a buscar ir além dos limites que vejo e tentar ajudar as pessoas de uma forma mais concreta.

Aos que trabalham comigo na Coteminas ou na Springs Global US, especialmente a Daniel pela grande ajuda que me dá todos os dias.

Aos que não são da minha família, nem meus professores, não trabalham comigo ou ainda nem são meus amigos, mas estão lendo esta dissertação, também os agradeço, pois fazem este trabalho valer a pena. Muito obrigada.

Conteúdo

1 Introdução **1**

1.1 Contextualização 1

1.2 Tolerância a Sabotagem 4

1.3 Solução Proposta para Tolerar Sabotagem em Grades P2P 4

1.4 Objetivos 5

1.4.1 Objetivo Geral 5

1.4.2 Objetivos Específicos 5

1.5 Estrutura do Trabalho 6

2 Tolerância a Sabotagem **7**

2.1 Introdução 7

2.2 Modelo do Sistema 7

2.3 Técnicas para Avaliar a Corretude de um Resultado 9

2.3.1 Votação 9

2.3.2 *Spot-checking* e Lista Negra 10

2.4 Cálculo das Credibilidades 13

2.4.1 Credibilidade das Máquinas 14

2.4.2 Credibilidade dos Resultados 15

2.4.3 Credibilidade dos Grupos de Resultados e das Tarefas 15

2.5 Aplicação das Credibilidades 16

2.5.1 Unicamente *Spot-checking* 16

2.5.2 Unicamente Votação (Votação baseada em Margem) 17

2.5.3 Votação e *Spot-checking* Combinados 17

2.5.4 *Spot-checking* por meio de Votação 18

2.5.5 Exemplo 18

3 Escalonamento Adaptativo Tolerante a Sabotagem **20**

3.1 Introdução 20

3.2 Algoritmo de Escalonamento 21

3.3 Heurísticas de Escalonamento 23

3.3.1 Heurística Não-Adaptativa e sem Conhecimento de Máquinas Confiáveis 23

3.3.2 Heurística de Escalonamento Adaptativa com Número de Máquinas Restrito 25

3.3.3 Heurística de Escalonamento Adaptativa Gulosa 26

3.4 Estimativa Adaptativa de Parâmetros 27

3.4.1 Fração de Faltosos Conhecida e Adaptativa 27

3.4.2 Taxa Adaptativa para Geração de Tarefas de Averiguação 28

3.5 Sistema de Reputação 30

3.6 Componentes Principais 30

4 Avaliação de Desempenho **32**

4.1 Introdução 32

4.2 Simulador 32

4.2.1 Ferramentas de apoio 33

4.3 Metodologia 33

4.4 Resultados Obtidos 36

4.4.1 Comportamento da Heurística de Escalonamento Não-Adaptativa . 37

4.4.2 Comportamento das Heurísticas de Escalonamento Adaptativas . . 44

4.4.3 Comportamento a Longo Prazo das Heurísticas Adaptativas 59

4.4.4 Comportamento a Longo Prazo da Heurística Adaptativa com Número de Máquinas Restrito com Valor Mínimo para q 68

4.5 Considerações sobre as Heurísticas de Escalonamento 70

4.5.1 Sobre a Heurística de Escalonamento Não-Adaptativa 70

4.5.2 Sobre a Heurística de Escalonamento Adaptativa com Número de Máquinas Restrito 71

4.5.3 Sobre a Heurística de Escalonamento Adaptativa Gulosa 72

5 Trabalhos Relacionados **73**

5.1 Introdução 73

5.1.1 Zhao et al. 73

5.1.2 Sonnek et al. 74

5.2 Comparações Entre Projetos de Escalonamento Tolerante a Sabotagem . . . 74

5.2.1 Métricas Avaliadas pelos Trabalhos 74

5.2.2 Compartilhamento de Reputações 75

5.2.3 Escalonamento 76

5.2.4 Tratamento de Conluios 77

5.2.5 Sistemas de Reputação 78

5.3 Considerações sobre os Trabalhos Relacionados 79

6 Considerações Finais e Trabalhos Futuros **81**

6.1 Considerações Finais 81

6.2 Trabalhos Futuros 83

Lista de Figuras

1.1 Exemplo de grade P2P: comunidade OurGrid. 2

1.2 Comunidade OurGrid com sabotadores. 3

2.1 Taxa de erro da votação majoritária para vários valores de m e f. 9

2.2 Taxa de erro de *spot-checking* com lista negra versus a taxa de sabotagem. Para vários valores de n, $f = 20\%$ e $q = 10\%$. 12

2.3 Fila de tarefas para escalonamento com herança de credibilidade. 19

3.1 Interações entre os componentes do serviço de escalonamento. 31

4.1 Redundância média nos 5 primeiros *jobs* da heurística não-adaptativa para $s = 50\%$, $f_{real} = \{5\%, 10\%, 20\%\}$, $\varepsilon_{ace} = 0,1\%$, $f = \{0,0001 \cdot f_{real}, f_{real}\}$, $q = \{0, 01\%, 10\%\}$. 41

4.2 *Slowdown* médio nos 5 primeiros *jobs* da heurística não-adaptativa para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$, $f = \{0,0001 \cdot f_{real}, f_{real}\}$, $q = \{0, 01\%, 10\%\}$. 42

4.3 Erro médio obtido nos 5 primeiros *jobs* da heurística não-adaptativa para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$, $f = \{0,0001 \cdot f_{real}, f_{real}\}$, $q = \{0, 01\%, 10\%\}$. 43

4.4 Erro médio obtido nos 5 primeiros *jobs* da heurística não-adaptativa para $s = 50\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = 0,1\%$, $f = 0,0001 \cdot f_{real}$, $q = 10\%$. 44

4.5 Redundância média nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$. 46

4.6 Número final médio de *spot-checks* escalonados nos 5 primeiros *jobs* da heurística adaptativa gulosa para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$. 48

4.7 Número final médio de *spot-checks* escalonados nos 5 primeiros *jobs* da heurística adaptativa com número de máquinas restrito para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$. 49

4.8 *Slowdown* médio nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$. 52

4.9 Variação de f da réplica que contribuiu para o pior *slowdown* do primeiro *job* da heurística adaptativa gulosa para $c = 5\%$, $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$. 53

4.10 Variação de f da réplica que contribuiu para o melhor *slowdown* do primeiro *job* da heurística adaptativa gulosa para $c = 5\%$, $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$. 54

4.11 Erro médio obtido nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$. 57

4.12 Número final médio de sabotadores descobertos nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 1\%$. 58

4.13 Número final médio de máquinas confiáveis nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$. . . 60

4.14 Redundância média nos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0, 1\%\}$. 62

4.15 *Slowdown* médio nos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0, 1\%\}$. 63

4.16 Erro médio nos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0, 1\%\}$. 65

4.17 Número médio de máquinas confiáveis ao final dos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0, 1\%\}$. . . 66

4.18 Número médio de sabotadores descobertos ao final dos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0, 1\%\}$. . . 67

4.19 Redundância, *slowdown*, erro médios nos 35 primeiros *jobs* da heurística adaptativa com número de máquinas restrito para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = 1\%$ e valor mínimo para q de 10%. 69

4.20 Número final de sabotadores médio nos 35 primeiros *jobs* da heurística adaptativa com número de máquinas restrito para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = 1\%$ e valor mínimo para q de 10%. 70

5.1 Modelo do sistema: um servidor mantém armazenadas as taxas de confiabilidade e as utiliza para atribuir tarefas a grupos de máquinas. 75
5.2 Modelo de escalonamento baseado em confiança. 76
5.3 Vazão média (a) e a taxa de sucesso (b) com 100 nós na rede. 77
5.4 Vazão média (a) e a taxa de sucesso (b) com 1000 nós na rede. 77
5.5 Precisão vs. percentual de faltosos; sem sistema de reputação, $s = 50\%$. . . 78
5.6 Sobrecarga vs. percentual de faltosos; sem sistema de reputação, $s = 50\%$. . 78
5.7 Precisão vs. percentual de faltosos; sistema de reputação local, cálculos feitos entre as 500.000 primeiras tarefas submetidas, $s = 50\%$. 79
5.8 Sobrecarga vs. percentual de faltosos; sistema de reputação local, cálculos feitos entre as 500.000 primeiras tarefas submetidas, $s = 50\%$. 79

Capítulo 1

Introdução

1.1 Contextualização

As grades computacionais, também conhecidas na literatura por "*grids*", foram desenvolvidas para proporcionar uma infra-estrutura para execução de aplicações paralelas em máquinas geograficamente dispersas, de modo que um usuário que se conecte à grade tenha acesso à potência computacional como se estivesse ligando um eletrodoméstico a uma tomada para receber energia elétrica[1].

Esse termo foi adotado em meados dos anos 90 para significar a então proposta infra-estrutura de computação distribuída para ciência avançada e engenharia. A partir desse momento, emergiram uma boa compreensão dos problemas que a tecnologia de grade soluciona e um conjunto de aplicações apropriadas a ela [Ora01].

Atualmente, a maior demanda para a computação em grade provém da ciência. A pesquisa científica do século XXI, também chamada de *e-science*, caracteriza-se pelo processamento de alto desempenho para aplicações paralelas de larga escala e análise de dados. As grades computacionais permitem o aumento do poder de computação, contribuindo para diminuir o tempo do processamento de dados e de execução das aplicações [SN02][Fos05][Fos02].

A formação das grades computacionais varia quanto ao tipo de acesso e utilização. O TeraGrid [Pen02] é um exemplo de grade privada com poucos sítios e que utiliza o Globus [FKT01][Fos06] como *middleware*. Visando facilitar o acesso de colaboradores, alguns

[1] Em inglês, a rede de energia elétrica também é chamada *grid* [Sar01].

projetos abriram sua grade para usuários voluntários interessados em participar. Exemplos de grade de computação voluntária são a distributed.net [Hay98][Dis07] e a infraestrutura aberta da universidade de Berkeley chamada BOINC (*Berkeley Open Infrastructure for Network Computing*) [And03]. Esta permite a colaboração com os projetos científicos: *SETI@home* [SET07], *Climateprediction.net* [cli07], *Einstein@home* [Ein07], *LHC@home* [LHC07], *Predictor@home* [Pre07], *Rosetta@home* [Ros07], *Cell Computing* [Cel07] e *World Community Grid* [WCG07]. Várias razões podem encorajar um voluntário a aderir a tais sistemas, como, por exemplo, a satisfação de ajudar um projeto de pesquisa e ter seu nome na lista de contribuintes, recebimento de dinheiro por tempo de processamento ou alguma outra razão econômica.

Apenas contribuir com projetos científicos não permite aos usuários executarem suas próprias aplicações. No intuito de compartilhar ciclos computacionais ociosos de diversos usuários, surgiram as grades entre-pares (ou *peer-to-peer* - P2P)[UJJL05]. Nessas, os usuários são livres para ingressarem, doarem ciclos de CPU e executarem suas próprias aplicações paralelas. A comunidade OurGrid [CBA+06] é um exemplo de grade P2P e a Figura 1.1 [CBA+06] mostra um exemplo de submissão de trabalhos (*jobs*) do usuário para máquinas dessa grade.

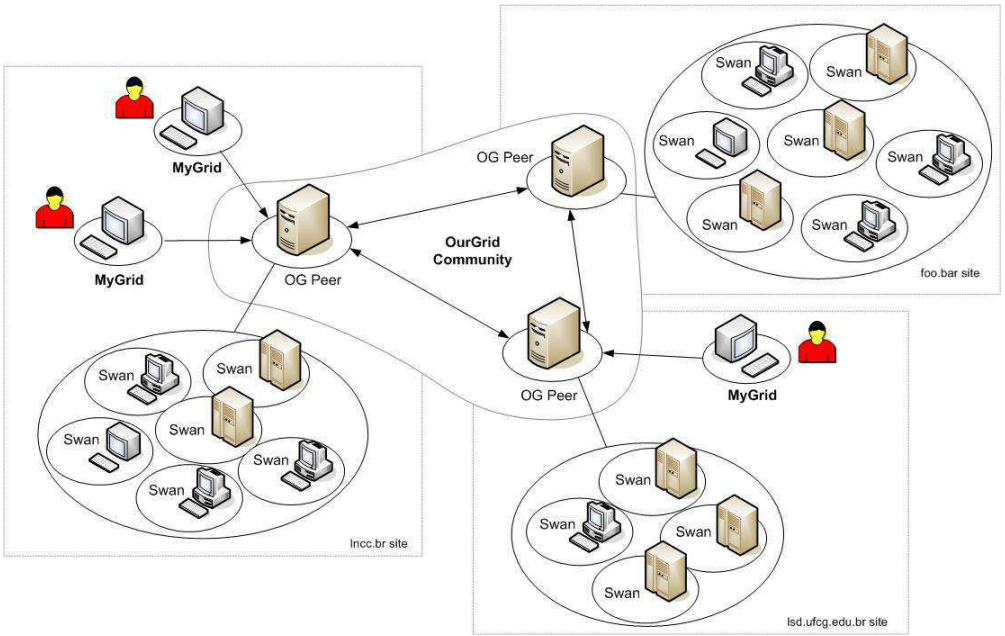


Figura 1.1: Exemplo de grade P2P: comunidade OurGrid.

O usuário utiliza um escalonador de aplicação (*broker*) para submeter seus trabalhos. O

escalonador requisita as máquinas a um *peer* que é responsável por lhe prover tais máquinas de forma transparente (de seu próprio sítio ou recebidas da comunidade). De posse das máquinas, o *broker* faz o escalonamento das tarefas, além disso informa ao usuário sobre o estado das mesmas, erros na execução e quando o trabalho está concluído ou falhou.

As grades P2P estão suscetíveis a um problema já diagnosticado em sistemas de computação voluntária, como o SETI@home: a possibilidade de usuários trapaceiros (por se ter de confiar em máquinas desconhecidas)[Mol00]. No caso do SETI@home, os voluntários trapaceavam para serem considerados os melhores colaboradores na classificação pública dos voluntários.

Várias razões podem motivar a sabotagem em grades P2P. Por exemplo, pode existir alguma vantagem econômica relativa aos ciclos computacionais que o usuário fornece à comunidade. Ele pode beneficiar-se na política de alocação de recursos. O usuário pode ainda se motivar pelo simples fato de ser reconhecido entre os amigos como um colaborador muito participativo. A Figura 1.2 ilustra como sabotadores podem estar distribuídos na comunidade OurGrid. Uma fração f das máquinas é composta por máquinas sabotadoras (fração de sabotadores) e cada uma dessas máquinas tem uma probabilidade s (taxa de sabotagem) de retornar um resultado incorreto.

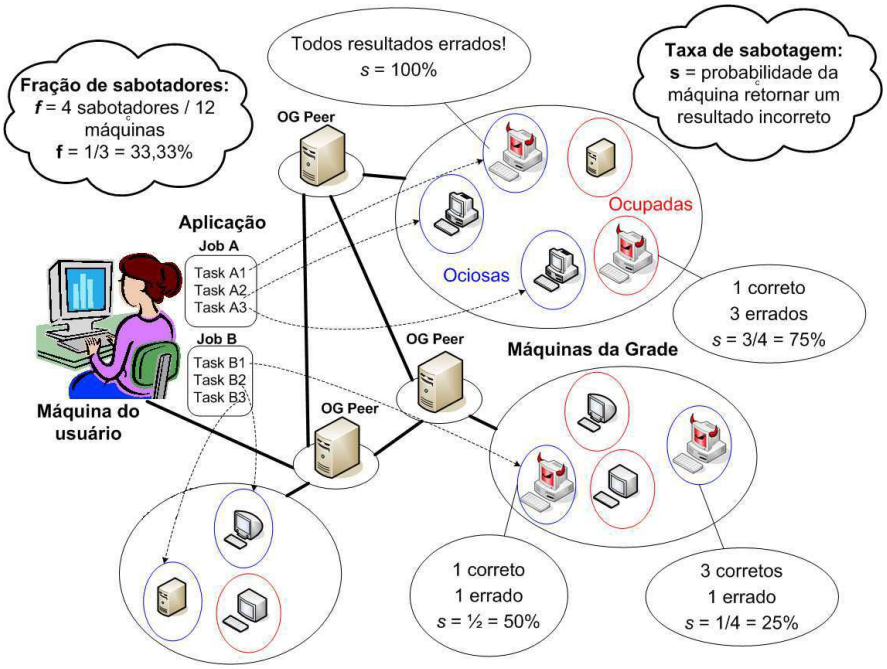


Figura 1.2: Comunidade OurGrid com sabotadores.

1.2 Tolerância a Sabotagem

A princípio, as falhas arbitrárias estavam associadas a problemas de componentes de *hardware* ou erros no desenvolvimento do *software*. Sendo assim, as falhas possuíam uma probabilidade de manisfestarem um erro que podia ser determinada, pois se manifestavam quando era utilizado um componente ou um trecho de código falho. E, determinando as causas da falha, podia-se prever a freqüência ou quando novas falhas poderiam ocorrer.

Sabotagem é um tipo de falha estudada na área de segurança computacional (*security*), que trata de falhas maliciosas [Web02]. Usuários mal intencionados podem fazer o sistema se comportar diferentemente do modo esperado. Não é simples estimar o número provável de sabotagens ou a freqüência com que podem acontecer.

Alguns trabalhos foram propostos na literatura para tolerar falhas arbitrárias (ou bizantinas) de *hardware* ou *software* baseadas em replicação com quóruns definidos de acordo com um número máximo de falhas toleradas. Mas como estabelecer limites para sabotagens em um sistema?

O custo de tolerar sabotagem pode ser alto. As falhas maliciosas podem não seguir uma determinada característica e se manifestarem mesmo com componentes de *hardware* e *software* em perfeito funcionamento. Em geral, as técnicas propostas na literatura para tolerar sabotagem baseiam-se em replicação, no sentido de realizar votação majoritária sobre os resultados. Para tolerar t falhas usando-se o mecanismo de votação, toda computação deve ser repetida pelo menos $2t + 1$ vezes [Web02].

1.3 Solução Proposta para Tolerar Sabotagem em Grades P2P

Uma forma de mensurar a confiabilidade de alguém no mundo físico é através da *reputação* dessa pessoa. Isso é verdade também para redes de computadores. Técnicas como o estabelecimento de reputações positivas e de redes de divulgação podem ser usadas para aumentar a confiança em uma máquina [Ora01].

Este trabalho apresenta um estudo de como tolerar sabotagem em grades computacionais P2P utilizando mecanismos de alto nível baseados em reputação. A solução consiste em

empregar a técnica de *tolerância a faltas baseada em credibilibidade* proposta por Sarmenta [Sar02] para tolerar sabotagem em sistemas de computação voluntária no mecanismo de escalonamento de grades P2P.

Essa técnica foi escolhida pelo fato de ser genérica e conduzir a uma redução de custos, quando comparada à votação tradicional. Foi elaborado um esquema de escalonamento de aplicações *bag-of-tasks* (BoT - aplicações paralelas cujas tarefas são independentes entre si) que limita a probabilidade de erro na aceitação de tarefas e minimiza o custo computacional, em termos da redundância empregada, bem como o *makespan* (tempo necessário para executar uma aplicação na grade).

Aplicar a técnica proposta por Sarmenta não é uma tarefa simples. Esse estudo teórico assume o uso de propriedades que não são conhecidas na prática e devem de alguma forma ser estimadas para que possam ser aplicadas no mecanismo de escalonamento.

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo deste trabalho não é propor a heurística de escalonamento tolerante a sabotagem mais eficiente em todas as situações possíveis. O objetivo geral é apresentar um estudo de como integrar a técnica de *tolerância a faltas baseada em credibilidade* no mecanismo de escalonamento de grades computacionais P2P.

1.4.2 Objetivos Específicos

Os objetivos específicos são implementar a *técnica de tolerância a faltas baseada em credibilidade* no escalonamento de tarefas *bag-of-tasks* em grades computacionais P2P, estudar o comportamento dessa técnica em diferentes estimativas de parâmetros de configuração e elencar os pontos fortes e fracos de cada abordagem.

1.5 Estrutura do Trabalho

O Capítulo 2 explica como as reputações são calculadas no sistema e como ocorre o mecanismo de aceitação de tarefas corretas, a rejeição de tarefas erradas e eliminação de sabotadores da grade. O Capítulo 3 descreve a contribuição deste trabalho ao estado da arte: escalonamento tolerante a sabotagem. Ele ressalta os problemas encontrados na aplicação da técnica de tolerância a faltas baseada em credibilidade, como o sistema de reputação está inserido no mecanismo de escalonamento e as interações entre os módulos do escalonador. A avaliação de desempenho do serviço de escalonamento é apresentada no Capítulo 4. Trabalhos relacionados que propõem soluções para escalonamento tolerante a sabotagem são discutidos no Capítulo 5. Por fim, o Capítulo 6 sintetiza as conclusões do estudo e discute possíveis trabalhos futuros para dar continuidade a este estudo.

Capítulo 2

Tolerância a Sabotagem

2.1 Introdução

As técnicas de tolerância a sabotagem neste trabalho consistem em estabelecer o grau de credibilidade que as tarefas submetidas à grade possuem e apenas aceitar como corretas aquelas que atendem a um requisito mínimo de credibilidade exigido para cada aplicação. O princípio da *tolerância a faltas baseada em credibilidade* proposto por Samenta [Sar01][Sar02] é o fundamento matemático utilizado para estimar as reputações.

Para verificar resultados, Sarmenta [Sar02][Sar01] demonstra que a combinação de *spot-checking* e *votação baseada em credibilidade* é uma prática eficiente. Todavia, deixa claro que outros mecanismos de verificação de resultados podem ser empregados na *tolerância a faltas baseada em credibilidade*.

2.2 Modelo do Sistema

Assume-se neste trabalho que todas as tarefas (*tasks*) são independentes (BoT) e submetidas em um grupo chamado *job*, que constitui uma aplicação paralela. Outra suposição é que todas as réplicas de uma mesma tarefa fornecem resultados idênticos (determinístico).

Assume-se que a grade possui pelo menos um sítio (*peer*) confiável conhecido pelo escalonador (minimamente, o sítio local é confiável). Todas as máquinas oriundas de sítios confiáveis são idôneas. O escalonador pode conhecer outros sítios confiáveis e apenas confia plenamente nas máquinas de tais sítios.

O modelo de falhas adotado é o de falhas arbitrárias para o resultado final da computação. Uma máquina falha (ou sabotadora) pode entregar um resultado incorreto ao final da computação. O sabotador decide quando ou não a falha deve ocorrer.

As máquinas se comunicam por meio de troca de mensagens (arquivos de entrada e saída), entretanto o custo de transferência dos arquivos na rede não foi considerado no estudo. Não foram consideradas perdas de pacotes. Esperam-se que todos os resultados para as tarefas escalonadas sejam recebidos.

Algumas propriedades modelam o comportamento do sistema e são utilizadas para compor a solução do problema. Elas estão elencadas abaixo:

- **Credibilidade -** Cr**:** representa a probabilidade de determinado objeto do sistema estar operando corretamente. Também entendida como **reputação**;
- **Taxa de erro aceitável -** ε_{ace}**:** corresponde à probabilidade máxima de erro que o usuário deseja aceitar. Essa propriedade é definida para cada *job* e se aplica a todas as tarefas do mesmo, ou seja, se $\varepsilon_{ace} = 1\%$, então para cada 100 tarefas do *job*, apenas uma tarefa com resultado errado pode ser equivocadamente aceita como tendo completado corretamente;
- **Limiar de credibilidade -** θ**:** corresponde à credibilidade mínima que deve ser atingida pelo resultado de uma tarefa para garantir que a tarefa seja aceita como correta. É definida por $\theta = 1 - \varepsilon_{ace}$;
- **Fração de faltosos -** f: também chamada de *fração de sabotadores* ou de máquinas maliciosas. Representa o percentual de sabotadores na grade;
- **Taxa de sabotagem -** s**:** exprime a probabilidade de uma máquina sabotadora retornar um resultado incorreto independentemente das demais;
- **Quantidade de máquinas confiáveis -** c**:** fração de máquinas nas quais o sistema confia previamente. Elas são provenientes de sítios confiáveis. A credibilidade dessas máquinas é 100%.

2.3 Técnicas para Avaliar a Corretude de um Resultado

Sarmenta [Sar02] propõe mecanismos para diminuição de custos de tolerância a sabotagem a partir da combinação de **votação** e ***spot-checking***. Esta prática permite que altas taxas de acerto sejam obtidas com baixos níveis de redundância, se comparados à votação majoritária tradicional. Antes de ser discutido como essas duas técnicas podem ser combinadas, elas serão estudadas isoladamente.

2.3.1 Votação

A votação majoritária é uma técnica tradicional que replica as tarefas e espera obter um *quorum* mínimo de resultados idênticos para poder aceitar uma tarefa como correta. A Figura 2.1 [Sar02] apresenta os resultados obtidos por Sarmenta para o esquema de *votação majoritária m-primeiros*. Nesse esquema, o módulo responsável por alocar tarefas, receber os resultados e os comparar espera até que m resultados idênticos sejam recebidos para a mesma tarefa antes de poder aceitá-la.

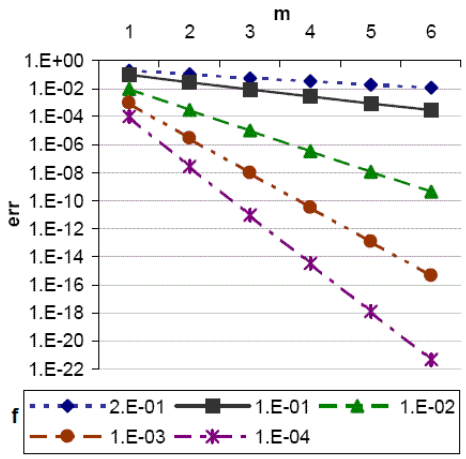


Figura 2.1: Taxa de erro da votação majoritária para vários valores de m e f.

Esse gráfico corresponde à taxa de erro (err) versus o número de resultados idênticos esperados (m) para várias frações de sabotadores (f). Por exemplo, quando f é muito pequeno como $f = 0,01\%$, i.e., 10^{-4}, dois valores iguais ($m = 2$) conduzem a uma taxa de erro inferior a $0,000001\%$, i.e., 10^{-8}, e continua diminuindo exponencialmente à medida que m aumenta. Entretanto, quando f aumenta, é necessária uma redundância bem maior para

obter baixas taxas de erro. Por exemplo, quando $f = 20\%$ (2×10^{-1}), mesmo para $m = 6$, a taxa de erro ainda é maior do que 1%.

A conclusão que pode ser tirada é que essa técnica pode atingir taxas de erro baixas, no entanto, tolerar muitas falhas requer um alto custo. Portanto, é recomendada para os casos em que $f < 1\%$. Esse esquema requer que toda a computação seja realizada pelo menos duas vezes. A redundância esperada para esse esquema de votação é definida por $\frac{m}{1-f}$ [Sar01].

2.3.2 *Spot-checking* e Lista Negra

Spot-checking é uma abordagem para detectar máquinas maliciosas proposta por Sarmenta que testa as máquinas com tarefas de averiguação (ou *spot-checks*), cujos resultados são previamente conhecidos ou existe alguma maneira confiável de os obter. As máquinas recebem tarefas de averiguação com uma probabilidade q, chamada de taxa de geração de *spot-checks*. Isso quer dizer que se $q = 10\%$, então 10% das tarefas deverão ser *spot-checks*. A redundância, em média, fica em torno de $\frac{1}{1-q}$.

Essa técnica permite que sejam indentificados sabotadores sem a necessidade de replicar todo o trabalho. Caso haja muitas tarefas, em relação ao número de máquinas disponível, e o ε_{ace} for grande, então as máquinas podem atingir a credibilidade-alvo apenas sendo aprovadas em *spot-checks*, sem necessitar que se inicie a replicação das tarefas. Portanto, a taxa de erro aceitável pode ser obtida mesmo com uma redundância menor do que dois.

Quando se diagnostica que uma máquina foi reprovada em um *spot-check*, uma solução é banir a máquina da grade. Isso pode ser implementado por meio de uma **lista negra** que armazena algum identificador da máquina. Isso permite que todos os resultados fornecidos por tais máquinas sejam invalidados e que novas tarefas não sejam direcionadas a essas máquinas futuramente.

Existem problemas relacionados também na escolha de um identificador para as máquinas. Se as máquinas forem excluídas pelo IP de suas interfaces de rede, então se elas usarem um protocolo de concessão dinâmica de endereços (DHCP) podem receber em algum momento outro IP e continuarem a sabotar. Ou ainda, se uma máquina não maliciosa receber o IP de uma máquina que está na lista negra, ela não poderá fornecer computação útil ao sistema.

Outra possibilidade é o uso de credenciais para uma máquina ingressar na grade. Antes

de uma máquina se juntar à rede, ela deveria, por exemplo, identificar-se a uma autoridade certificadora e requerer uma credencial válida para o acesso. Quando um sabotador fosse encontrado, as suas credenciais iriam para a lista negra. Credenciais, porém, não se enquadram bem na filosofia de comunidades P2P de livre-ingresso.

Para simplificar, assumiu-se que o identificador inserido na lista negra é suficiente para remover o sabotador do sistema.

A *taxa de erro final média* da técnica de *spot-checking* com lista negra, ε_{scln}, representa a probabilidade de um resultado escolhido aleatoriamente estar incorreto ao final da computação. A Equação 2.1 expressa ε_{scln}, onde n é o número total de tarefas que a máquina executou até a conclusão do trabalho. As tarefas incorretas só podem ser provenientes de sabotadores que sobreviveram até o final da aplicação, pois se um sabotador for pego durante a execução, ele é posto na lista negra e todos os resultados que ele forneceu são eliminados. A probabilidade de um sabotador permanecer até o final da aplicação é dada por $f \cdot (1 - q \cdot s)^n$. A probabilidade da máquina ser um sabotador é f e a de não ser é $1 - f$. Portanto, a probabilidade da máquina sobreviver até o final da aplicação é da máquina não ser um sabotador **ou** de ser um sabotador e permanecer até o final [Sar01], como é definido pela Equação 2.1.

$$\begin{aligned}
\varepsilon_{scln}(q, n, f, s) &= P(\text{resultado escolhido aleatoriamente ser incorreto}) \\
&= s \times P(\text{resultado vir de um sabotador}) \\
&= s \times P(\text{máquina ser um sabotador} \mid \text{sabotador sobreviveu}) \\
&= s \times \frac{P(\text{máquina ser um sabotador } \mathbf{e} \text{ sabotador sobreviveu})}{P(\text{máquina ter sobrevivido})} \\
&= \frac{s \cdot f \cdot (1 - q \cdot s)^n}{(1 - f) + f \cdot (1 - q \cdot s)^n}
\end{aligned} \tag{2.1}$$

A Figura 2.2 [Sar01] mostra o comportamento da taxa de erro em diferentes valores da taxa de sabotagem, para $0 \leq s \leq 1$, com $q = 10\%$ e $f = 20\%$. Quando $s = 0\%$ não existe erro, pois a máquina nunca retorna um resultado incorreto. Vale lembrar que quanto maior for o valor de n, conseqüentemente é maior o número de *spot-checks* pelos quais a máquina estará sujeita. Outro ponto importante é que quanto mais s aumenta, maior é a chance de um sabotador ser pego pelo mecanismo de *spot-checking*. Pode-se observar pelo gráfico que à

medida que n vai crescendo, as taxas de erro vão se tornando cada vez menores.

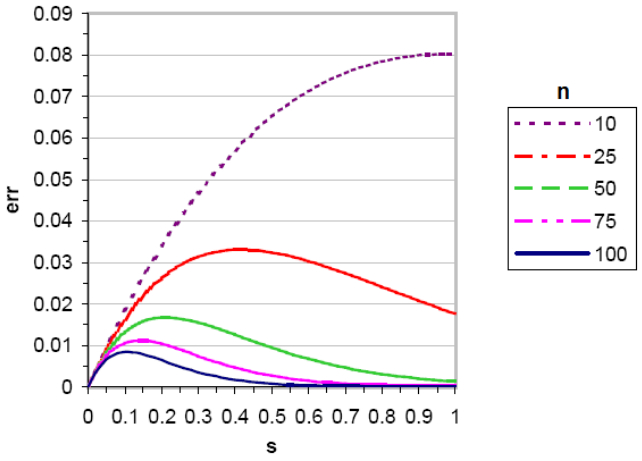


Figura 2.2: Taxa de erro de *spot-checking* com lista negra versus a taxa de sabotagem. Para vários valores de n, $f = 20\%$ e $q = 10\%$.

As taxas de erro crescem no ínicio, quando s é muito baixo, depois atingem um ponto de saturação (onde a taxa de sabotagem pode ser considerada "ótima"para os sabotadores) e começam a decrescer à medida em que s aumenta. A razão é porque, para permanecer no sistema, o sabotador precisa passar em todas as tarefas de averiguação e, se ele tiver uma alta taxa de sabotagem, também será alta a chance de responder com um resultado incorreto a um *spot-check* que lhe tenha sido alocado.

Quando $n = 10$ tarefas, não se percebe redução da taxa de erro de acordo com o gráfico. Nesse caso, o número provável de *spot-checks* seria igual a 1, pois $q = 10\%$. É bastante difícil capturar um sabotador com este número de tarefas, usando exclusivamente o mecanismo de *spot-checking* independentemente do valor s. Por essa razão e tendo em vista que 20% das máquinas são sabotadoras, as taxas de erro são as mais altas.

A Equação 2.1 não possui uma fórmula fechada para o seu ponto máximo exato. Sarmenta [Sar01], no entanto, apresenta um limite superior para ela, como mostra a Equação 2.2, onde s é a probabilidade de uma máquina retornar um resultado incorreto; $f \cdot (1 - q \cdot s)^n$ é a probabilidade de um sabotador permanecer até o final da computação e $1 - f$ é a probabilidade de uma máquina ser honesta.

$$\varepsilon_{scln}(q, n, f, s) < \hat{\varepsilon_{scln}} = \frac{s \cdot f \cdot (1 - q \cdot s)^n}{1 - f} \tag{2.2}$$

O ponto máximo da Equação acontece quando a taxa de sabotagem é expressa pela Equação 2.3. Do ponto de vista do sabotador, esse é o melhor valor para s.

$$\hat{s^*_{scln}}(q,n) = min\left(1, \frac{1}{q \cdot (n+1)}\right) \tag{2.3}$$

O valor máximo de erro para a pior taxa de sabotagem é limitado estrita e assintoticamente pela Equação 2.4, onde e é a base do logarítimo neperiano. Essa equação representa o pior caso de erro, para quando o sabotador conhece n e q antecipadamente e ajusta sua taxa de sabotagem com base neles. Nesse caso, a taxa de erro é inversamente proporcional a n. Quando s é constante, porém, a ε_{scln} decresce exponencialmente no tempo à medida que n aumenta [Sar01].

$$\varepsilon_{scln}(q,n,f,s) < \hat{\varepsilon^*_{scln}}(q,n) < \frac{f}{1-f} \cdot \frac{1}{q \cdot n \cdot e} \tag{2.4}$$

Quanto maior for o número de tarefas de uma aplicação, mais as taxas de erro podem ser reduzidas, pois maior será a contribuição para a computação realizada por cada máquina (maior n por máquina).

2.4 Cálculo das Credibilidades

Neste ponto o objetivo é estimar a credibilidade das tarefas. As consequências de identificar a corretude de uma tarefa são: definir o estado da tarefa, descobrir sabotadores, excluir todos os resultados provenientes dos mesmos e remover os sabotadores da grade.

Para se estimar a credibilidade final do trabalho, i.e., todas as tarefas executadas, são calculadas as credibilidades de quatro objetos do sistema, seguindo a metodologia proposta por Sarmenta. Os objetos são *máquina*, *resultado*, *grupo de resultados* e *tarefa*. A tarefa é a fração da aplicação paralela que é executada em uma máquina da grade, podendo ser replicada e computada em diferentes máquinas. Cada máquina fornece um resultado para a réplica de uma tarefa e esses resultados podem ou não coincidir. Como este trabalho assume que as tarefas são determinísticas, ou seja, só existe um resultado correto para uma tarefa, apenas um grupo de resultados pode apresentar o resultado correto para a tarefa a que ele se refere. Outras formas podem existir para calcular as credibilidades, dependendo das técnicas escolhidas para avaliar a corretude das tarefas.

As credibilidades são calculadas na seguinte ordem:

1. **credibilidade da máquina -** Cr_P**:** cada máquina da grade tem uma credibilidade, que depende do número de *spot-checks* executados ou de conhecimento prévio que o escalonador tenha sobre ela;

2. **credibilidade do resultado -** Cr_R**:** assim que o escalonador recebe um resultado da execução de uma tarefa, a credibilidade do resultado é calculada. Essa depende da credibilidade da máquina que o executou;

3. **credibilidade do grupo de resultados -** Cr_G**:** uma mesma tarefa pode ter vários resultados, pois, graças ao mecanismo de votação, podem existir várias réplicas para uma mesma tarefa. É estabelecida uma credibilidade para cada grupo de resultados iguais obtidos;

4. **credibilidade da tarefa -** Cr_W**:** o objetivo do processo de atribuição de credibilidades é encontrar a credibilidade da tarefa. Essa representa a probabilidade da tarefa estar correta e deve ser igual ou superior à credibilidade-alvo. A credibilidade corrente de uma tarefa é dada pela maior credibilidade entre as credibilidades dos grupos de resultados da tarefa.

2.4.1 Credibilidade das Máquinas

Para calcular a credibilidade de uma máquina, usa-se o número de spot-checks k em que ela foi aprovada. Ou seja, pode-se estimar como uma máquina tende a retornar um resultado correto pelo número de *spot-checks* pelos quais passou. Em quanto mais *spot-checks* ela for aprovada, quanto mais confiança se tem de que a máquina é correta ou, pelo menos, que ela não tem uma taxa de sabotagem alta.

A Equação 2.5 representa a fórmula do cálculo da credibilidade para uma máquina P utilizando as técnicas de ***s**pot-**c**hecking* e **l**ista **n**egra (*scln*), onde f representa a fração de máquinas maliciosas (ou faltosas). Esta credibilidade corresponde à probabilidade de P retornar um resultado correto.

$$Cr_P(P)_{scln} = 1 - \frac{s \cdot f \cdot (1-s)^k}{(1-f) + f \cdot (1-s)^k} \tag{2.5}$$

Como a taxa de sabotagem s de determinada máquina é uma variável difícil de obter, outra equação será aplicada. A Equação 2.6 corresponde a um limite inferior estrito para a probabilidade de uma máquina P retornar um resultado correto, independentemente do valor de s. Sarmenta [Sar01] apresenta mais detalhes sobre como esta equação foi derivada.

$$Cr_P(P)_{scln} = 1 - \frac{f}{1-f} \cdot \frac{1}{k \cdot e} \tag{2.6}$$

2.4.2 Credibilidade dos Resultados

O método seguido para estimar a credibilidade de um resultado R é simples e está expresso na Equação 2.7. Assume-se que é igual à credibilidade da máquina *R.solucionador* que o produziu.

$$Cr_R(R) = Cr_P(R.solucionador) \tag{2.7}$$

2.4.3 Credibilidade dos Grupos de Resultados e das Tarefas

Com o mecanismo de replicação (votação), vários resultados de uma mesma tarefa poderão ser obtidos. Para estimar a credibilidade de uma tarefa, primeiro se agrupam os resultados iguais e depois se estima a credibilidade de cada grupo de resultados.

Se uma tarefa W **tem apenas um resultado** R, então sua credibilidade é estimada como a credibilidade desse resultado, como mostra a Equação 2.8.

$$Cr_W(W) = Cr_R(R) = Cr_P(R.solucionador) \tag{2.8}$$

Como mais resultados podem ser produzidos, formam-se g grupos por critério de igualdade de resultados. O grupo a é representado por G_a, onde $1 \leq a \leq g$. O número de elementos do grupo G_a é igual a m_a.

As credibilidades dos grupos de resultados se baseiam na probabilidade condicional de correção, dada a combinação dos resultados recebidos até o momento do cálculo, como mostrado na Equação 2.9. Lembrando que, como as tarefas possuem resultados determinísticos, apenas um grupo de resultados pode estar correto, no máximo.

$$
\begin{aligned}
Cr_G(G_a) &= \frac{P(G_a \text{ correto}) \cdot P(\text{todos os outros são incorretos})}{P(\text{pegue } g \text{ grupos, onde cada } G_a \text{ tem } m_a \text{ membros})} \\
&= \frac{P(G_a \text{ correto}) \cdot \prod_{i \neq a} P(G_i \text{ incorreto})}{\prod_{j=1}^{g} P(G_j \text{ incorreto}) + \sum_{j=1}^{g} P(G_j \text{ correto}) \cdot \prod_{i \neq j} P(G_i \text{ incorreto})}
\end{aligned}
\tag{2.9}
$$

Dado que a:

- probabilidade de **todos** os resultados $R_{a,i}$ do grupo G_a serem corretos é:
  $P(G_a \text{ correto}) = \prod_{i=1}^{m_a} Cr_R(R_{a,i})$
- probabilidade de **todos** os resultados $R_{a,i}$ do grupo G_a serem incorretos é:
  $P(G_a \text{ incorreto}) = \prod_{i=1}^{m_a} (1 - Cr_R(R_{a,i}))$

2.5 Aplicação das Credibilidades

2.5.1 Unicamente *Spot-checking*

Uma maneira simples de estabelecer reputações aos objetos do sistema é aplicar unicamente a técnica de *spot-checking*. Sarmenta ressalta em seu trabalho [Sar01] que se deve notar que como *spot-checking* reduz a probabilidade de erro inversamente em k, essa probabilidade não decresce exponencialmente. Tendo em vista esse fato, usar *spot-checking* sozinho não é praticável a menos que o erro aceitável não seja muito menor do que f. Caso contrário, serão necessários muitos *spot-checks* até que uma máquina atinja o limiar de credibilidade esperado.

Por exemplo, se $f = 30\%$ e $\varepsilon_{ace} = 0,001\%$, quantos spot-checks serão necessários para que uma máquina atinja a credibilidade-alvo θ?

Solução:

$$
\begin{aligned}
\theta &= 1 - \varepsilon_{ace} \\
&= 1 - 0,00001 \\
&= 0,99999
\end{aligned}
\tag{2.10}
$$

De acordo com a Equação 2.6, para uma máquina M obter $Cr_P(M) = 0,99999$, o número de *spot-checks* necessários é obtido pela Equação 2.11, onde e é a base do logarítmo neperiano.

$$\begin{aligned} k &= \frac{1}{1 - Cr_P(M)} \cdot \frac{f}{(1-f) \cdot e} \\ &= \frac{1}{1 - 0,99999} \cdot \frac{0,30}{(1-0,30) \cdot e} \\ &\cong 15.767 \textit{ spot-checks} \end{aligned} \tag{2.11}$$

Isso quer dizer que para obter a credibilidade-alvo de $99,999\%$ para uma determinada máquina, com $f = 30\%$ e $\varepsilon_{ace} = 0,001\%$, são necessárias no mínimo 15.767 tarefas redundantes de averiguação. O custo de utilizar apenas *spot-checking* nesse caso é muito alto.

2.5.2 Unicamente Votação (Votação baseada em Margem)

A credibilidade das máquinas permanece constante sem o mecanismo de *spot-checking*, contudo, a credibilidade da tarefa aumenta à medida que resultados iguais são recebidos para a mesma. Ao se esperar reunir o número suficiente de resultados para alcançar o limiar de credibilidade θ, um mecanismo *dinâmico* de votação é implementado. Ele difere da votação m-primeiros no sentido de que a redundância não é fixada de antemão, mas varia dependendo de f e θ.

Esse tipo de votação pode ser considerado como *votação baseada em margem* ou *votação m-à-frente*, pois se espera até que um dos grupos de resultados tenha no mínimo uma margem de m resultados a mais do que o outro [Sar01].

2.5.3 Votação e *Spot-checking* Combinados

A união de *spot-checking* e votação funciona aproveitando *spot-checking* e votação em um único esquema e apresenta vantagens com relação ao uso dos dois isoladamente. Inicialmente se atribui uma credibilidade inicial a cada máquina, por exemplo, $1 - f$. Depois se alocam as tarefas e se começa a coleta dos resultados. Caso a fila de tarefas seja grande e o

limiar de credibilidade θ seja baixo, as máquinas aumentarão suas credibilidades sendo aprovadas em *spot-checks* e seus resultados poderão ser aceitos sem a necessidade de votação, podendo, assim, ser obtido um grau de replicação menor do que 2.

Contudo, se θ for muito alto, as máquinas irão requerer várias rodadas de trabalho até que suas credibilidades estejam altas o suficiente. Então, inicia-se a alocação de tarefas redundantes às máquinas e, com isto, o mecanismo de votação dos resultados [Sar01].

2.5.4 *Spot-checking* por meio de Votação

Nesta técnica, quando um resultado é aceito como correto na votação baseada em credibilidade, i.e., um grupo de resultados alcança a credibilidade-alvo, ele é considerado um *spot-check*. Dessa feita, as máquinas que contribuíram para o resultado têm seu número de *spot-checks* aprovados k acrescido de 1 e, em decorrência, sua credibilidade aumentada.

As máquinas que perdem a votação são consideradas sabotadores. O procedimento tomado nessa situação é o mesmo para quando as máquinas são reprovadas em uma tarefa de averiguação. Elas têm seu valor de credibilidade reduzido para zero, são adicionadas à lista negra e todos os resultados que forneceram para a aplicação em execução são invalidados.

Sarmenta [Sar01] defende que a taxa de *spot-checks* deva ser pequena, menor do que 10%. Vale ressaltar que como $k \approx q \cdot n$, o crescimento da credibilidade das máquinas é limitado em k, tal como o desempenho. Essa técnica possibilita um maior crescimento do valor de k e, conseqüentemente, um aumento mais freqüente da credibilidade das máquinas, tornando a votação mais rápida.

A votação baseada em credibilidade funciona bem porque garante que a votação só será efetivada quando a probabilidade de que o resultado está correto for alta o suficiente [Sar01].

2.5.5 Exemplo

A Figura 2.3 [Sar02] representa uma fila de tarefas. As tarefas (na figura, *work*) a serem escalonadas estão numeradas de 0 a 999. Este exemplo considera uma fração de faltosos $f \leq 0,2$ e a credibilidade alvo $\theta = 0,999$. Note que a tarefa 0 (*Work 0*) foi executada pela máquina (ou *worker*) P1 que ainda não fora testada por nenhum *spot-check* ($k = 0$). Nesse exemplo, teve sua credibilidade inicial definida como $1 - f = 0,8$. À medida que

resultados vão sendo recebidos para uma tarefa, eles vão sendo agrupados e a credibilidade de cada grupo é calculada de acordo com a Equação 2.9. Note que se houver apenas um resultado para a tarefa, o cálculo é feito com base na Equação 2.8. As máquinas que vão sendo aprovadas em *spot-checks* aumentam suas credibilidades[1].

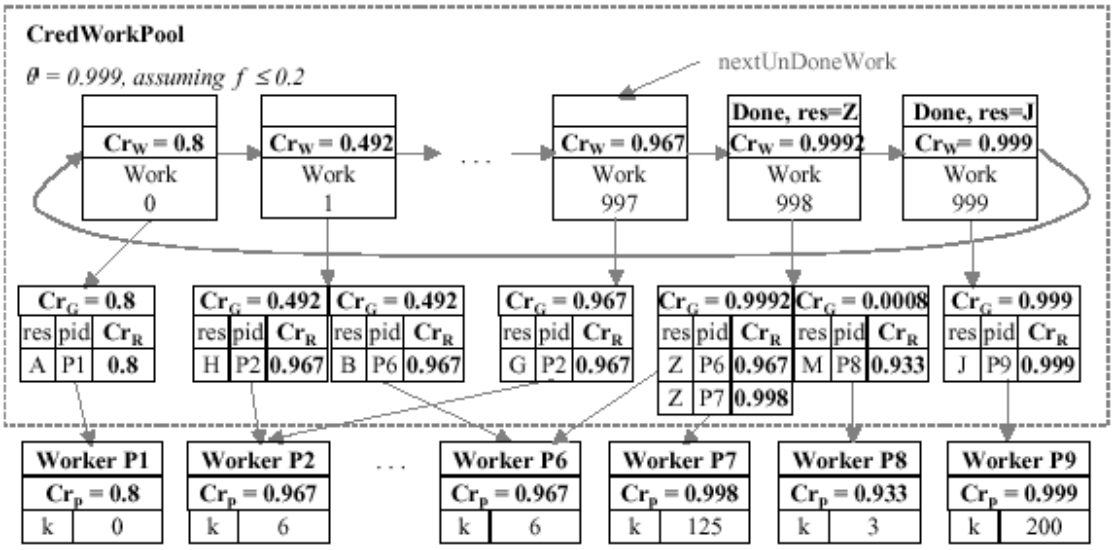


Figura 2.3: Fila de tarefas para escalonamento com herança de credibilidade.

Se um grupo de resultados alcançar o limiar de credibilidade, então este resultado será aceito e poderá ser considerado um *spot-check* passado por todas as máquinas que o produziram. Em contrapartida, as máquinas dos demais grupos serão tratadas como detectadas retornando um resultado errado em um *spot-check*. A credibilidade da tarefa será a credibilidade do grupo vencedor.

[1]A equação empregada para calcular a credibilidade dessas máquinas não foi a mesma adotada por este trabalho (Equação 2.5). Sarmenta [Sar02] utilizou a equação que propôs para o mecanismo de *spot-checking* sem lista negra: $Cr_W(W) = 1 - \frac{f}{k}$.

Capítulo 3

Escalonamento Adaptativo Tolerante a Sabotagem

3.1 Introdução

Sarmenta provou que é possível limitar as taxas de erro na aceitação de tarefas em sistemas de computação suscetíveis a sabotadores [Sar01]. Naquele trabalho é avaliada a técnica genérica de *tolerância a faltas baseada em credibilidade*. Essa técnica pode ser composta de várias outras técnicas que estipulem credibilidades para as tarefas, no entanto, ela será composta neste trabalho por votação baseada em credibilidade e *spot-checking* com lista negra.

Essas técnicas fazem uso de algumas variáveis para estimar as credibilidades das tarefas que nem sempre são conhecidas a priori e que na prática são, muitas vezes, difíceis de estimar. No caso em estudo, essas variáveis são a fração de máquinas maliciosas f e a taxa de geração de *spot-checks*, q, como discutido no Capítulo 2. Como se pode estimar a fração de máquinas desonestas presentes em um sistema de livre ingresso, como as grades P2P? Existe alguma maneira de garantir a taxa de erro aceitável e diminuir custos com tarefas extras, como *spot-checks*, à medida que as máquinas vão aumentando suas reputações?

Essas questões são deixadas em aberto no trabalho de Sarmenta e surgem como dificuldades para a concretização prática das técnicas propostas. As Seções 3.2 e 3.3 exemplificam como unir as soluções propostas e implementar um mecanismo de escalonamento de tarefas tolerante a sabotagem em grades P2P. A Seção 3.4.1 relata como o problema de f desconhe-

cido foi contornado, a Seção 3.4.2 explica como se chegou a um q adaptativo.

3.2 Algoritmo de Escalonamento

Algumas adaptações foram feitas para que a técnica de *tolerância a faltas baseada em credibilidade* pudesse operar no contexto de escalonamento em grades P2P. O Algoritmo 1 representa o funcionamento genérico de um escalonamento tolerante a sabotagem em que apenas são aceitas as tarefas cujas credibilidades são iguais ou superiores a θ. Ele termina quando todas as tarefas atingem essa credibilidade-alvo. O momento de alocar uma tarefa de averiguação a uma máquina é probabilístico e determinado de acordo com o valor de q.

Algoritmo 1 Algoritmo genérico para escalonamento tolerante a sabotagem

```
seja LM a lista de máquinas disponíveis, segundo heurística
seja LT a lista de tarefas para executar
seja M uma máquina
seja T uma tarefa

enquanto houver tarefa em LT que não atingiu θ faça
  enquanto ( houver máquina ociosa em LM ) ∧ ( houver tarefa em LT que não atingiu θ ) faça
    M = a próxima máquina de LM
    se M não estiver na lista negra então
      se ( M não for reconhecidamente confiável ) ∧ ( existirem spot-checks no sistema ) ∧ ( for
      o momento de executar um spot-check ) então
        T = um spot-check
      senão
        T = uma tarefa ainda não concluída
      fim se
      escalone T em M
    fim se
  fim enquanto
fim enquanto
```

Note pelo Algoritmo 1 que podem existir **máquinas reconhecidamente confiáveis**. A quantidade delas é definida pela propriedade c, como apresentado na Seção 2.2. Desde o íni-

cio do processo pode-se ter informação sobre máquinas confiáveis e que esse comportamento não irá mudar.

O Algoritmo 2 mostra como ocorre o processamento de um resultado R executado por uma máquina M qualquer para a tarefa T do *job* J. Ele representa a forma como os resultados são tratados no sistema. Primeiro ponto importante é que caso um sabotador seja identificado na votação de resultados enquanto esteja executando a réplica de alguma tarefa, o resultado fornecido por ele nem será processado. Apenas os resultados originados por máquinas que não estão na lista negra são processados. Caso seja a réplica de um *spot-check*, então, como o resultado é conhecido, verifica-se se o mesmo está correto. Se ele estiver correto, o número de *spot-checks* aprovados k para M é incrementado. Se não, então a máquina é adicionada à lista negra e todos os resultados que ela forneceu para tarefas do *job* em execução são invalidados. De acordo com a corretude do resultado, a credibilidade da máquina M, $Cr_P(M)$, é estipulada e armazenada no sistema.

A credibilidade do resultado R, $Cr_R(R)$, é dada pela credibilidade da máquina. No caso das máquinas previamente confiáveis, essa credibilidade é 100%. Em seguida, o resultado R é inserido no grupo de resultados de T que são iguais a ele ou em um novo grupo se não houver outro resultado idêntico previamente recebido. Depois de alimentar o sistema com as informações relevantes necessárias, chega o momento de calcular a credibilidade da tarefa, $Cr_W(T)$, que é a finalidade de todos os cálculos das credibilidades.

Quando uma tarefa é executada em uma máquina reconhecidamente confiável, a tarefa e o resultado recebido alimentam o mecanismo de *spot-checking* e $Cr_W(T) = 100\%$. Essa tarefa também é contabilizada como um *spot-check* e todas as máquinas que submeteram o mesmo resultado têm o número de *spot-checks* aprovados k incrementado. Caso R tenha sido executado por uma máquina não previamente confiável, então é calculada a credibilidade Cr_G para cada grupo de resultados. A credibilidade corrente da tarefa é a maior credibilidade dentre as credibilidades dos grupos de resultados.

O valor da credibilidade da tarefa é armazenado para eventuais comparações no algoritmo de escalonamento (Algoritmo 1). O último passo do processamento de um resultado é o teste do limiar de credibilidade. Se a credibilidade da tarefa for igual ou superior à credibilidade-alvo, então é feita a limpeza do sistema. As máquinas que não fazem parte do grupo vencedor da votação são consideradas sabotadores, adicionadas à lista negra e seus resultados são

invalidados.

3.3 Heurísticas de Escalonamento

Este trabalho estuda três heurísticas de escalonamento. Dois parâmetros mudam de comportamento entre uma heurística e outra. Eles são a *fração de faltosos*, f, e a taxa de geração de *spot-checks*, q. Outra característica que pode diferir entre as heurísticas é a informação prévia sobre sítios confiáveis, ou seja, sítios em que todas as suas máquinas são confiáveis durante todo o processo de execução de aplicações.

Na Seção 3.3.1 é proposta uma heurística não-adaptativa. Ela possui f e q constantes. As Seções 3.3.2 e 3.3.3 apresentam heurísticas que estimam f e q de modo adaptativo. Para a heurística da Seção 3.3.2, f varia de acordo com o número de máquinas de boa reputação no sistema e para a heurística da Seção 3.3.3 o valor de f varia de acordo com o número de máquinas previamente confiáveis e com o número de sabotadores descobertos. Em ambas as heurísticas adaptativas o valor de q para uma determinada máquina varia de acordo com f, o e_{ace} e o número de tarefas, n, executadas por ela.

3.3.1 Heurística Não-Adaptativa e sem Conhecimento de Máquinas Confiáveis

Na heurística introduzida pelo Algoritmo 3, pode-se perceber que as máquinas são ordenadas aleatoriamente no início da iteração de escalonamento das tarefas. A aleatoriedade foi adotada para representar que todas as máquinas têm a mesma probabilidade de executarem uma tarefa qualquer.

A taxa de geração de *spot-checks*, q, e a *fração de faltosos*, f, permanecem constantes durante todo o processo. Seus valores são pré-fixados desde o início da execução dos trabalhos. A fração de faltosos para essa heurística representa um limite máximo para o número de sabotadores tolerados pelo sistema para que a taxa de erro aceitável seja garantida. A desvantagem de utilizar essa premissa é que o número de usuários maliciosos não é uma variável para a qual se possa estabelecer limites de ocorrência na prática. Sistemas P2P são de livre ingresso e podem conter qualquer número de usuários maliciosos.

Algoritmo 2 Evento de recepção do resultado R

seja R o resultado recebido
seja T a tarefa original desta réplica
seja J o *job* a que T pertence
seja M a máquina que forneceu R
seja V um grupo de resultados

se M não estiver na lista negra **então**
  se R for um *spot-check* **então**
    se R estiver correto **então**
      incremente em 1 o número de *spot-checks* aprovados para M
      calcule $Cr_P(M)$
    senão
      coloque M na lista negra
      invalide todos os resultados fornecidos por M para tarefas de J
      $Cr_P(M) = 0$
    fim se
    armazene $Cr_P(M)$
  fim se
  se M não for reconhecidamente confiável **então**
    $Cr_R(R) = Cr_P(M)$
  senão
    $Cr_R(R) = 1$
  fim se

  insira R no grupo de resultados G_a apropriado
  se M for reconhecidamente confiável **então**
    $Cr_W(T) = 1$
    $V = G_a$
    adicione T à lista de *spot-checks* disponíveis
    incremente em 1 o número de *spot-checks* aprovados para cada máquina que forneceu os resultados de V
  senão
    calcule Cr_G para cada grupo de resultados de T
    $V =$ grupo com a maior Cr_G
    $Cr_W(T) = Cr_G(V)$
  fim se
  armazene $Cr_W(T)$
  se $Cr_W(T) \geq \theta$ **então**
    coloque na lista negra todas as máquinas que forneceram os resultados dos grupos diferentes de V
    invalide todos os resultados das máquinas que forneceram os resultados dos grupos diferentes de V em J
  fim se
fim se

Algoritmo 3 Heurística de Escalonamento com Ordenação Aleatória das Máquinas

LM = lista de máquinas disponíveis ordenada aleatoriamente

Outro detalhe importante para salientar é a credibilidade inicial das máquinas. Como a credibilidade representa a probabilidade de estar correto e o único parâmetro conhecido sobre a grade é f, que corresponde à probabilidade de ser faltoso, entende-se aqui que a probabilidade de se escolher uma máquina correta é $Cr_P = 1 - f$.

Como esta heurística não tem conhecimento sobre máquinas confiáveis, então deve existir de início ao menos uma tarefa cujo resultado pode ser confiavelmente obtido para ser usado no escalonamento como *spot-check* para as máquinas.

3.3.2 Heurística de Escalonamento Adaptativa com Número de Máquinas Restrito

Esta heurística implementa uma fração de faltosos adaptativa à quantidade de máquinas confiáveis. Esse f é descrito na Seção 3.4.1 no tópico em que f é abordado como **agressivo**. A intenção é manter um f baixo, de modo que as máquinas ganhem credibilidade mais rapidamente e o custo computacional também seja baixo. A taxa de geração de *spot-checks* adaptativa é específica para cada máquina e calculada com base no valor da sua credibilidade. A Seção 3.4.2 explica o método empregado para o cálculo de q.

Algoritmo 4 Heurística adaptável à quantidade de máquinas confiáveis

seja L a lista de máquinas disponíveis

sejam m_c e m_d números naturais

seja f um número real

L = lista de máquinas disponíveis ordenada por credibilidade decrescentemente

m_c = número de máquinas em L cuja $Cr_P \geqslant \theta$

m_d = número de máquinas desconhecidas admitido, *definido na configuração do sistema*

$$f = 1 - \frac{m_c}{m_c + m_d}$$

LM = sub-lista com as $m_c + m_d$ primeiras máquinas de L que não estão na lista negra

Esta heurística trata a credibilidade inicial das máquinas desconhecidas como sendo $Cr_P = 50\%$ e corresponde à imprevisibilidade da máquina ser ou não confiável. Essa credi-

bilidade é adotada apenas enquanto a máquina não passou por nenhuma tarefa de averiguação, porque a máquina tem sua credibilidade estimada de acordo com a Equação 2.5 após executar com sucesso um *spot-check*. As máquinas previamente confiáveis ficam de fora do processo de descoberta de reputação, pois suas credibilidades são sempre $Cr_P = 100\%$.

3.3.3 Heurística de Escalonamento Adaptativa Gulosa

A heurística de escalonamento adaptativa gulosa não limita o número de máquinas a serem usadas no escalonamento. Ela escalona as tarefas para todas as máquinas disponíveis que forem doadas pela comunidade P2P. Para manter a fração de faltosos controlada, assumiu-se um f bastante conservador, de acordo com as informações conhecidas pelo sistema. A Seção 3.4.1, na parte que trata sobre o f conservador, comenta o método para esse cálculo de f. O Algoritmo 5 descreve em alto nível as instruções que compõem a heurística.

Algoritmo 5 Heurística adaptativa gulosa

seja L a lista de máquinas disponíveis

sejam m_{pc}, m_s e m_{total} números naturais

L = lista de máquinas disponíveis

m_{pc} = número de máquinas previamente confiáveis

m_s = número de máquinas sabotadoras identificadas até o momento

m_{total} = número total de máquinas disponíveis

f = 1 - $\dfrac{m_{pc}}{m_{total} - m_s}$

$LM = L$

A taxa de geração de *spot-checks* segue o mesmo princípio da taxa utilizada pela *heurística adaptativa com número de máquinas restrito*. O método para estimar q é descrito na Seção 3.4.2.

3.4 Estimativa Adaptativa de Parâmetros

O mecanismo de escalonamento adaptativo proposto neste trabalho possui dois parâmetros: f e q. Duas heurísticas adaptativas são estudadas. Elas diferem no conjunto de máquinas utilizado para escalonar as tarefas e na maneira como lidam com f. A *heurística adaptativa com número de máquinas restrito* utiliza um f mais **agressivo** e a *heurística adaptativa gulosa* emprega um f mais **conservador**. Ambas as heurísticas estimam o q adaptativo da mesma forma. Os métodos para estimar f e q são discutidos na Seção 3.4.1 e na Seção 3.4.2, respectivamente.

3.4.1 Fração de Faltosos Conhecida e Adaptativa

Fração de Faltosos Agressiva

Uma maneira simples de conhecer a fração de faltosos é restringindo-a no mecanismo de escalonamento. Neste momento, a informação sobre máquinas previamente confiáveis é importante para iniciar o funcionamento do sistema. No princípio, utiliza-se apenas as máquinas previamente confiáveis e se adiciona um número m_d de máquinas desconhecidas ao conjunto de máquinas escalonáveis. Como não se conhece a intenção dessas máquinas nem se considera que foram testadas suficientemente, assumir-se-á que são sabotadores.

A idéia é compor um processo gradual de inclusão de máquinas desconhecidas ao escalonamento, de modo à manter o valor de f controlado. Com a continuação, as m_d máquinas ou vão aumentando suas reputações e sendo consideradas confiáveis ou vão sendo excluídas do processo e adicionadas à lista negra. O que faz uma máquina deixar de ser desconhecida é sua credibilidade ser igual ou superior ao limiar de credibilidade. Novas máquinas desconhecidas vão participando do escalonamento, não ultrapassando m_d máquinas desconhecidas por rodada, até que todas possam participar.

Desta forma, a fração de sabotadores usada é determinada, como mostra o Algoritmo 4, onde m_c é o número de máquinas confiáveis, ou seja, aquelas cujas credibilidades são iguais ou superiores à credibilidade-alvo. No início serão apenas as confiáveis, mas, à medida que o *job* executa, todas as que atingirem o limiar. Logo, a fração de máquinas maliciosas usadas no escalonamento, f, torna-se conhecida e poderá ser aplicada no cálculo da credibilidade das máquinas.

Fração de Faltosos Conservadora

A fração de faltosos conservadora requer a presença de máquinas confiáveis no sistema e supõe que são previamente conhecidas pelo escalonador. Ela utiliza suposições fortes para o cálculo de f. Subentende que as máquinas da grade que não são previamente confiáveis são sabotadoras. O valor de f decresce apenas quando sabotadores são identificados e banidos do escalonamento, diminuindo o percentual de possíveis sabotadores no sistema.

Essa medida foi tomada para não restringir o número de máquinas no escalonamento e todas as máquinas disponíveis poderem ser empregadas no escalonamento paralelamente. As suposições da solução encontrada tiveram de ser fortes para que tal atitude fosse tomada e que o valor de f se mantivesse conhecido. Em geral, f possui um valor alto, a menos que o ambiente seja extremamente hostil e um grande percentual de sabotadores seja descoberto.

A heurística adaptativa gulosa, representada pelo Algoritmo 5, implementa f de modo conservador. Uma característica desse método de estimar f é a obtenção de uma corretude alta, porém, ao mesmo tempo, tende a aumentar os custos e a sacrificar o desempenho, pois a credibilidade das máquinas cresce mais lentamente.

3.4.2 Taxa Adaptativa para Geração de Tarefas de Averiguação

A taxa de geração de *spot-checks* também pode ser uma propriedade dinâmica no sistema e que atenda à necessidade que cada máquina tem de ser testada. Faz sentido que as máquinas que foram provadas tenham menor probabilidade de receber uma tarefa de averiguação do que aquelas que ainda não executaram tarefa alguma. Seguindo este raciocínio e atentando para o limite máximo para a taxa de erro de uma máquina, chegou-se a uma taxa de geração de *spot-checks* adaptativa q_{adap}, conforme descrito a seguir.

Sarmenta [Sar01] constatou que a taxa média de erro para um sabotador i que sobreviveu até o final de uma aplicação, com os mecanismos de *spot-checking* e lista negra, ε_i, possui um limite superior $\hat{\varepsilon}_i^*$. Este está expresso na Equação 3.1, onde e é a base do logaritmo neperiano e n_i é o número de tarefas concluídas por i durante a execução da aplicação.

$$\hat{\varepsilon}_i^*(q, n_i) < \frac{f}{1-f} \cdot \frac{1}{q \cdot n_i \cdot e} \tag{3.1}$$

Em suma, sejam ε e $\hat{\varepsilon}^*$ a taxa média de erro e taxa média máxima de erro da aplicação,

respectivamente, pode-se afimar que:

$$\varepsilon = \frac{1}{j}\sum_{i=1}^{j}\varepsilon_i$$

$$\hat{\varepsilon}^* = \frac{1}{j}\sum_{i=1}^{j}\hat{\varepsilon}_i^*$$

onde j é o número de sabotadores que sobreviveram até o final da aplicação. O objetivo de estipular ε_{ace} é garantir que: (i) $\varepsilon \leq \varepsilon_{ace}$ e (ii) $\hat{\varepsilon}^* \leq \varepsilon_{ace}$. Pode-se dizer, sem perda de generalidade, que a taxa de erro para cada sabotador também deve ser, em média, menor do que o ε_{ace}, logo:

$$\varepsilon_{ace} \geq \hat{\varepsilon}_i^* \Rightarrow \varepsilon_{ace} \geq \frac{f}{1-f}\frac{1}{qn_ie}$$

Agora se pode limitar o valor de q para cada sabotador i (q_i) como sendo uma função de f e n_i, haja vista que ε_{ace} e e são constantes:

$$q_i(f, n_i) \geq \frac{f}{1-f} \cdot \frac{1}{\varepsilon_{ace} \cdot n_i \cdot e} \; ; \; \varepsilon_{ace} \neq 0, \; n_i \neq 0$$

Logo, é possível estimar uma taxa de geração de *spot-checks* adaptativa, q_{adap}, à necessidade de cada máquina, de forma a garantir o ε_{ace} para um f conhecido (e também restrito). Isto está representado pela Equação 3.2.

$$q_{adap}(f, n) = \frac{f}{1-f} \cdot \frac{1}{\varepsilon_{ace} \cdot n_i \cdot e} \tag{3.2}$$

Quando uma máquina i ainda não executou tarefas ($n_i = 0$), deve-se estipular um valor para q_{adap}. Nesse caso, os cálculos são feitos com base em n_i inicial igual a 1.

A maneira como q_{adap} foi definido pode acarretar em custos muito altos, principalmente quando ε_{ace} é muito baixo. q_{adap} pode ficar em 100% por um longo período de estabilização. Não obstante toda a conceituação para obter a taxa de geração de *spot-checks* adaptativa, optou-se por adotar um q_{adap} de no máximo 50% e deixar o mecanismo de votação atuando em parceria com o de *spot-checking*. Essa medida interfere apenas para valores de q_{adap} superiores a 50%, o que já representa uma redundância muito alta para tarefas de teste.

$$q_{adap}(f, n) = min\left(0,5; \frac{f}{1-f} \cdot \frac{1}{\varepsilon_{ace} \cdot n_i \cdot e}\right) \tag{3.3}$$

3.5 Sistema de Reputação

O sistema de reputação é atualizado com base nos resultados das execuções das tarefas. Para avaliar a correção das tarefas, os mecanismos de verificação de resultados escolhidos são votação baseada em credibilidade e *spot-checking* **por** votação, seguindo os passos enumerados abaixo:

1. inicialmente, todas as máquinas previamente conhecidas como confiáveis têm credibilidade 1 e as não conhecidas têm credibilidade inicial de acordo com a heurística em execução;

2. as máquinas, com exceção das previamente confiáveis, ganham credibilidade ao serem aprovadas em tarefas de averiguação (*spot-checks*);

3. quando a credibilidade de uma tarefa é igual ou superior ao limiar de credibilidade e se têm, no mínimo, dois resultados de alguma tarefa, pode-se votar nos resultados. Aumenta-se a credibilidade das máquinas que produziram os resultados ganhadores da votação e se remove do sistema (adicionando-se em lista negra) os nós que perderam. A tarefa será considerada concluída e não mais será replicada;

4. caso o limiar de credibilidade não seja atingido, armazenam-se os resultados obtidos por grupos de semelhança e se continua replicando as tarefas e gerando *spot-checks* até que as credibilidades das tarefas cresçam o suficiente para atingir o limiar.

Sarmenta [Sar02] ainda propõe métodos para estimar a credibilidade das tarefas na ausência de lista negra, inclusive apresentando bons resultados em termos de redundância e erro. Entretanto, a avaliação desses métodos não está no escopo deste trabalho.

3.6 Componentes Principais

Em resumo, cinco componentes principais constituem o núcleo do serviço de escalonamento tolerante a sabotagem proposto: 1) verificador de resultados; 2) calculador de credibilidade; 3) módulo para armazenamento de credibilidade; 4) provedor de *spot-checks* e 5) lista negra. A Figura 3.1 mostra as interações entre eles.

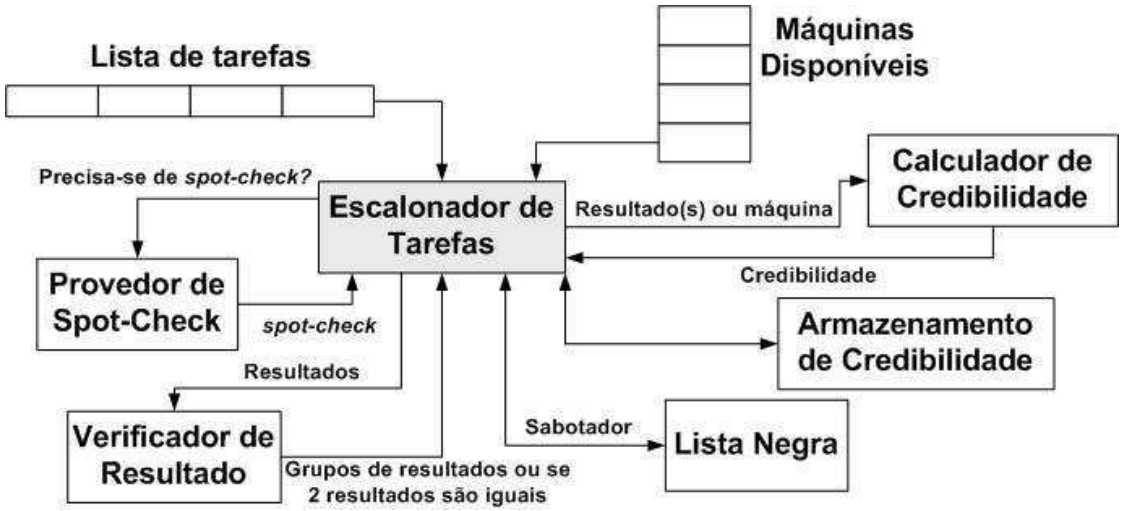


Figura 3.1: Interações entre os componentes do serviço de escalonamento.

No momento em que o escalonador é notificado de que uma máquina completou a réplica de uma tarefa, o escalonador solicita ao módulo *calculador de credibilidade* que estime quais são as credibilidades da máquina que executou a tarefa e daquela tarefa. A credibilidade da máquina é armazenada no componente de *armazenamento de credibilidade*. Os resultados iguais são postos em um mesmo grupo. O componente *verificador de resultado* é que faz este agrupamento, bem como valida os resultados retornados para os *spot-checks*. Sempre que um novo resultado é inserido em algum grupo, as credibilidades de todos os grupos são (re)calculadas pelo *calculador de credibilidade* e o grupo que atingir a credibilidade-alvo ganha a votação. Esses grupos também são armazenados no componente de *armazenamento de credibilidade*.

Se um sabotador for identificado devolvendo um resultado incorreto pelo componente *verificador de resultados*, ele será removido do sistema e todos os resultados que forneceu serão descartados. Para evitar o recebimento de resultados oriundos de sabotadores, existe uma *lista negra* onde uma identificação da máquina sabotadora é armazenada.

Capítulo 4

Avaliação de Desempenho

4.1 Introdução

Este capítulo analisa os resultados obtidos por meio de simulações para o escalonamento tolerante a sabotagem para grades computacionais P2P mediante três heurísticas de escalonamento. O Capítulo 3 apresentou o algoritmo de escalonamento e as heurísticas. Antes de discutir os resultados, este capítulo contém uma breve descrição do simulador desenvolvido e da metodologia empregada para obtenção dos resultados. Ele conclui com algumas considerações sobre as características das heurísticas em estudo, discutindo em que situações qual heurística se adequa melhor.

4.2 Simulador

Duas ferramentas foram empregadas para implementar o simulador. O escalonador de aplicação das tarefas *bag-of-tasks* para grades computacionais P2P foi implementado usando como base um simulador de grades computacionais já existente e de código aberto. O simulador escolhido foi o GridSim [BM02]. Já a geração dos eventos probabilísticos das simulações foi implementada com o auxíliio de bibliotecas do Projeto Colt [Col07].

4.2.1 Ferramentas de apoio

GridSim. O GridSim é uma ferramenta para modelagem e simulação de sistemas paralelos e distribuídos, como grades computacionais e *clusters*. Faz parte do projeto Gridbus, patrocinado pela Universidade de Melbourne (Australia), a Sun Microsystems (USA) e *Victorian Partnership for Advanced Computing* (VPAC) (Melbourne, Australia) [BM02]. Essa escolha diminuiu consideravelmente o esforço necessário para a codificação do simulador, fazendo com que o trabalho se concentrasse na implementação dos algoritmos de escalonamento e dos módulos mandatórios para a técnica de tolerância a falta baseada em credibilidade. A ferramenta GridSim contribuiu de forma direta na manipulação transparente dos recursos da grade, como a definição da arquitetura dos computadores, capacidade de processamento, largura de banda, carga e relógios. Administrou ainda o tempo de execução definido para as tarefas, levando em consideração os tamanhos estipulados para os arquivos de entrada e saída.

Colt. O Projeto Colt disponibiliza um conjunto de bibliotecas de código aberto para computações científica e técnica de alto desempenho [Col07]. Dentre elas, utilizou-se a biblioteca *Jet*, que provê ferramentas matemáticas e estatísticas para análise de dados, histogramas, geração de números aleatórios e distribuições de probabilidade. As duas últimas funcionalidades foram aproveitadas no simulador. Usou-se geração de números aleatórios com distribuição de probabilidade uniforme para escolher as máquinas trapaceiras (uma fração f do número total de máquinas), quando uma máquina deve executar uma tarefa de averiguação (com probabilidade q), qual o momento em que uma máquina trapaceira deve retornar um resultado incorreto (com probabilidade s) e no momento da escolha de para qual máquina escalonar uma tarefa (em heurísticas que fazem essa escolha aleatoriamente).

4.3 Metodologia

Os resultados foram obtidos por meio de simulações do algoritmo genérico de escalonamento apresentado no Algoritmo 1. O recebimento dos resultados fornecidos pelas máquinas para a execução das tarefas é representado pelo Algoritmo 2. Ambos os algoritmos foram discutidos na Seção 3.2. Este estudo avalia três heurísticas de escalonamento acopladas ao algoritmo genérico de escalonamento. Uma delas não é adaptativa e as outras duas são adaptativas. As

heurísticas de escalonamento foram discutidas na Seção 3.3.

A grade é composta por 100 *peers*, cada um com 10 máquinas, totalizando 1.000 máquinas. Cada simulação consiste no escalonamento de 5 *jobs* cada um possuindo um número $n_{tarefas}$, onde $n_{tarefas} = 10.000$. Há herança de reputações, ou seja, as credibilidades estimadas para os objetos do sistema persistem mesmo após a conclusão de um *job* e o próximo *job* também contribui para alimentar o sistema de reputação. O tempo necessário para executar 1 tarefa, t_{tarefa}, é 1 hora.

Cada uma das três heurísticas foi avaliada em 27 cenários diferentes. Os parâmetros variados são a taxa de sabotagem (s), a fração de sabotadores entre todas as máquinas da grade (f_{real}) e a taxa de erro aceitável (ε_{ace}), segundo a Tabela 4.1.

Tabela 4.1: Cenários avaliados.

No. Cenário	s	f_{real}	ε_{ace}	**No. Cenário**	s	f_{real}	ε_{ace}
1	25%	5%	1%	**15**	50%	20%	0,01%
2	25%	5%	0,1%	**16**	50%	70%	1%
3	25%	5%	0,01%	**17**	50%	70%	0,1%
4	25%	20%	1%	**18**	50%	70%	0,01%
5	25%	20%	0,1%	**19**	100%	5%	1%
6	25%	20%	0,01%	**20**	100%	5%	0,1%
7	25%	70%	1%	**21**	100%	5%	0,01%
8	25%	70%	0,1%	**22**	100%	20%	1%
9	25%	70%	0,01%	**23**	100%	20%	0,1%
10	50%	5%	1%	**24**	100%	20%	0,01%
11	50%	5%	0,1%	**25**	100%	70%	1%
12	50%	5%	0,01%	**26**	100%	70%	0,1%
13	50%	20%	1%	**27**	100%	70%	0,01%
14	50%	20%	0,1%				

- **Heurística Não-Adaptativa e sem Conhecimento de Máquinas Confiáveis:** os parâmetros específicos para essa heurística foram a taxa de geração de *spot-checks* (q), que assumiu os valores 0,01% e 10%, e a fração de sabotadores usada no escalona-

mento (f) também sofreu a interferência de um fator e assumiu os valores $0,01\% f_{real}$ e f_{real};

- **Heurística de Escalonamento Adaptativa com Número de Máquinas Restrito:** o parâmetro variado nessa heurística é a fração de sítios confiáveis (c) o qual assume os valores 5%, 10% e 20%. Além desse parâmetro, a heurística possui um valor fixo para o número de máquinas desconhecidas no escalonamento, $m_d = 1$, com possibilidade de variação também. No entanto, este estudo não avaliou outros valores para m_d;

- **Heurística de Escalonamento Adaptativa Gulosa:** possui apenas um parâmetro específico que é a fração de sítios confiáveis (c), também variando de 5%, 10% e 20%.

Como listado, cada heurística foi avaliada com parâmetros particulares a fim de serem estudados comportamentos específicos em seus desempenhos. Foram executadas 35 réplicas dos cenários para cada tratamento particular das três heurísticas. Cada réplica da simulação de um cenário recebe um conjunto de máquinas aleatoriamente escolhido. Porém, na avaliação dos parâmetros específicos de uma heurística, usa-se o mesmo conjunto aleatório de máquinas no escalonamento. Isso quer dizer que ao se avaliar a *heurística adaptativa com número restrito de máquinas*, por exemplo, com o parâmetro c em dois níveis, $c = 5\%$ e $c = 10\%$, o conjunto de máquinas doadas para a n-ésima réplica do tratamento $c = 5\%$ é o mesmo que é entregue ao escalonador na avaliação da n-ésima réplica quando $c = 10\%$.

São entregues ao escalonador um número N de máquinas escolhidas aleatoriamente da comunidade P2P, onde $N = 200$. A alocação é feita da seguinte forma:

1. são estabelecidas quais máquinas são sabotadoras e quais são reconhecidamente confiáveis no conjunto total de máquinas da grade, de acordo com os percentuais definidos por f_{real} e c. Este é exclusivo para as heurísticas adaptativas;

2. um número mínimo de 10 máquinas reconhecidamente confiáveis é entregue ao escalonador, que corresponde ao número de máquinas do sítio local. O intuito disso é que as heurísticas adaptativas tenham certeza de que existem máquinas previamente confiáveis no sistema para gerarem *spot-checks*;

3. as outras $N-10$ máquinas são escolhidas aleatoriamente entre o universo de máquinas da grade e doadas ao escalonador, compondo as N máquinas.

As métricas observadas foram:

- **Redundância**: a razão entre número de tarefas computadas (incluindo *spot-checks*) e o número original de tarefas;
- ***Slowdown***: representa a relação entre o tempo gasto na execução do *job* (*makespan*) na presença do escalonamento tolerante a sabotagem e o tempo que seria gasto para a conclusão do *job* apenas nas máquinas honestas disponíveis na ausência de técnicas de tolerância a sabotagem. Sarmenta também avalia essa métrica em seu trabalho [Sar01][Sar02]. A Equação 4.1 consiste na fórmula usada para calcular o *slowdown* de cada *job*.

$$Slowdown = \frac{makespan_{obtido}}{makespan_{ideal}} \tag{4.1}$$

O $makespan_{obtido}$ é o ***makespan*** mensurado nas simulações e:

$$makespan_{ideal} = n_{rodadas} \times t_{tarefa}$$
$$n_{rodadas} = \frac{n_{tarefas}}{N \cdot (1 - f_{doado})}$$
$$f_{doado} = \frac{\#sab}{N}$$

onde $n_{rodadas}$ é o número de laços que devem ser feitos pelo escalonamento para completar um *job*; f_{doado} é a fração de faltosos dentre as máquinas que foram entregues ao escalonador; $\#sab$ corresponde ao número de sabotadores presentes no conjunto aleatório de máquinas doadas para o escalonamento. Portanto, o valor de f_{doado} vai sendo diminuído com a presença de lista negra à medida que os sabotadores são descobertos;

- ***Erro obtido***: erro médio obtido (ε) na aceitação dos resultados das tarefas ao final da execução do *job*.

4.4 Resultados Obtidos

Para facilitar o estudo da análise do comportamento das heurísticas, os resultados são apresentados para os mesmos cenários nas três heurísticas. Os cenários analisados têm $s = 50\%$,

$\varepsilon_{ace} = 0,1\%$ em três níveis de f_{real}: $f_{real} = 5\%$, $f_{real} = 20\%$ e $f_{real} = 70\%$.

Eles foram escolhidos por sintetizarem todos os comportamentos específicos de cada heurística em níveis intermediários, mas estudando como se comportam em diferentes níveis de f_{real}. Quando se diminui o valor de s, por exemplo, um maior número de sabotadores tende a continuar no sistema e quando $s = 100\%$, todos eles são descobertos por algum teste. Quando o erro aceitável é maior, mais tarefas erradas podem ser aceitas e quando ele é menor, mais se exige do escalonamento, causando mais custos e se obtém, por conseguinte, menos erros ao final da computação.

4.4.1 Comportamento da Heurística de Escalonamento Não-Adaptativa

Introdução

Foi visto na Seção 3.3.1 que a heurística não-adaptativa em estudo assume que f é a fração máxima de faltosos que estarão presentes, que ela é conhecida e que esse valor permanece constante. Assume-se ainda que a taxa de geração de *spot-checks*, q, também é um fator constante.

Esta avaliação busca inicialmente determinar qual o impacto do valor assumido para fração de faltosos no escalonamento no erro final da computação. Por exemplo, se o escalonador assumir um valor para f muito menor do que o real, que erro deve ser esperado? Isso interfere na redundância do processo? Como o *slowdown* se comporta?

Outro ponto que se procurou avaliar é o impacto da taxa de geração de *spot-checks*. Como q é um parâmetro que também deve ser configurado desde o início do escalonamento, como o sistema se comporta se a probabilidade de um *spot-check* ser executado for muito baixa?

Simulou-se a fração de faltosos usada no escalonamento, f, em dois níveis: $f = f_{real}$ e $f = f_{real} \times 10^{-4}$. A taxa de geração de *spot-checks* também foi avaliada em dois níveis: $q = 0,01\%$ e $q = 10\%$. O intervalo de confiança dos resultados em relação à média no escalonamento é de 95%. O erro padrão para os resultados esperados é de 5%.

Para entender o comportamento da heurística, é necessário lembrar como ocorre o estabelecimento de reputações e a decisão de quando uma tarefa está correta. Um tarefa é aceita como correta se sua credibilidade atinge θ. A credibilidade da tarefa é condicionada

à credibilidade da máquina que executou a tarefa, se esta for superior ou igual a θ então a tarefa é considerada concluída. Se não, então outra máquina executará uma réplica da tarefa e isso se repete até que a credibilidade da tarefa esteja acima da credibilidade-alvo. A credibilidade da tarefa é uma probabilidade condicional baseada nas credibilidades das máquinas que executaram réplicas da tarefa (votação baseada em credibilidade), conforme a Equação 2.9.

É preciso entender qual o papel de f e q nesse processo. A credibilidade da máquina é definida pela Equação 2.6[1]. Duas conclusões podem ser tiradas com relação a f e q. A primeira é que quanto menor for f, então maior tende a ser o valor da credibilidade. A segunda é que se q for alto, portanto k tem maior probabilidade de crescer, então a credibilidade da máquina também tende a crescer.

Pela maneira como foram investigados, os resultados obtidos para a heurística não-adaptativa apresentam duas características distintas. A primeira é quando o valor de q é muito baixo. Nesse caso o mecanismo de *spot-checking* deixa de existir. A segunda é quando f é considerado muito baixo. Nesse caso o mecanismo de votação não atua no escalonamento, pois as máquinas recebem credibilidades altas o suficiente para atingir o limiar de credibilidade. Ao analisar-se os resultados, pode-se perceber claramente o modo como essas características se manifestam.

Serão discutidos a seguir redundância, o *slowdown* e o erro nos 5 primeiros *jobs* para $s = 50\%$, $f_{real} = \{5\%, 10\%, 20\%\}$, $\varepsilon_{ace} = 0,1\%$, $f = \{0,0001 \cdot f_{real}, f_{real}\}$, $q = \{0,01\%, 10\%\}$. As Figuras 4.1, 4.2 e 4.3 apresentam essas métricas, respectivamente.

Fração de Faltosos Assumida equivalente à Fração Real de Faltosos

Quando $f = f_{real}$ e $q = 0,01\%$ o custo é constante, porque como praticamente não são gerados *spot-checks* as credibilidades das máquinas não aumentam e o número de réplicas para a votação é o mesmo para garantir ε_{ace}. O *slowdown* também se mantém constante e o erro é inferior ao aceitável, pois o mecanismo de votação garante isso, como mostram as Figuras 4.2 e 4.3.

Quando $f = f_{real}$ e $q = 10\%$, a redundância é maior no primeiro *job* e depois se esta-

[1] $Cr_P(P)_{scln} = 1 - \frac{f}{1-f} \cdot \frac{1}{k \cdot e}$, onde k é o número de *spot-checks* em que a máquina P foi aprovada e e é a base do logarítmo neperiano.

biliza a partir do segundo, pois as máquinas aumentam suas credibilidades pelo mecanismo de *spot-checking* e suavizam o quórum necessário para a votação. No entanto, suas credibilidades não estão altas a ponto de alcançarem o limiar de credibilidade e serem aceitas sem a necessidade de votação. O *slowdown* possui a mesma tendência da redundância. Isso ocorreu, porque como f é bastante alto nem todos os sabotadores conseguiram ser identificados de imediato no primeiro *job*, levando o *slowdown* a ser inferior a 1.

Quando $f_{real} = 70\%$, não foi possível concluir as simulações no tempo esperado para $q = 0,01\%$ e $f = f_{real}$, pois f era muito alto com relação a q e as credibilidades atribuídas às máquinas eram muito pequenas para atingirem θ em um tempo aceitável, mesmo com o mecanismo de votação.

Fração de Faltosos Assumida muito inferior à Fração Real de Faltosos

Quando a fração de faltosos assumida no escalonamento é $f = f_{real} \times 10^{-4}$, as credibilidades estimadas para máquinas tendem a ser altas, de acordo com a Equação 2.5. Para esse valor de f, não é necessário o mecanismo de votação de tarefas, pois as credibilidades atribuídas às máquinas são altas o suficiente para satisfazerem o limiar de credibilidade e levarem as tarefas à conclusão. Portanto, a redundância se mantém constante desde o primeiro *job*.

O primeiro ponto a avaliar é $q = 0,01\%$. Quando se assume esse valor para q, espera-se que 1 *spot-check* seja escalonado a cada 10.000 tarefas. Como visto, 1 a cada *job*, o que torna a redundância do mecanismo de *spot-checking* praticamente nula. Pode-se observar na Figura 4.1 que o custo da redundância quando $q = 0,01\%$ foi 1, o que significa que não houve redundância. Para se ter uma noção, a credibilidade inicial das máquinas é $Cr_P(P) = 1-f$, portanto, antes mesmo de executar um *spot-check*, cada máquina P já tem credibilidade $Cr_P(P) = 1 - 0,00007 = 0,99993$.

O *slowdown* para esse caso também é constante e depende do número de sabotadores presentes. As tarefas são executadas em todo o conjunto disponível de máquinas, pois não ocorre detecção de sabotadores. Como o *slowdown* corresponde ao tempo gasto para a execução das tarefas apenas nas máquinas corretas, quando $f = 5\%$ o *slowdown* é cerca de 5% inferior ao *slowdown* ideal, ou seja, 1. Da mesma maneira ocorre para $f = 20\%$ ou $f = 70\%$, conforme mostra a Figura 4.2.

O erro obtido é praticamente o mesmo do que o erro sem os mecanismo de tolerência a

sabotagem, $\varepsilon \approx s \cdot f$. Por exemplo, quando $s = 50\%$ e $f = 20\%$, $\varepsilon \approx (0,5) \times (0,2) \approx 0,1$, o que pode ser observado na Figura 4.3(b).

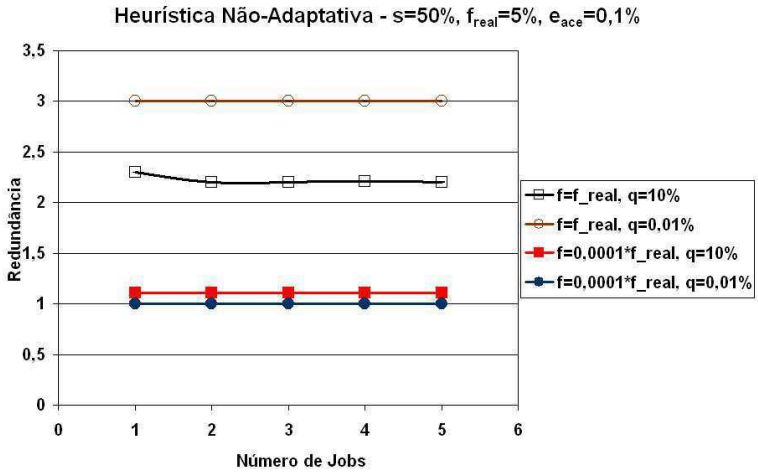
A segunda parte é avaliar $q = 10\%$. Quando $q = 10\%$, o baixo valor de f continua atribuindo altas reputações e evitando a replicação de tarefas e, por conseguinte, a votação dos resultados. Entretanto, a redundância não é nula, pois 10% das tarefas são *spot-checks*. Os resultados demonstram que a redudância foi em média cerca de 1,111. Esses valores condizem com os estudos de Sarmenta [Sar02], que traduzem a redundância da técnica de *spot-checking* como sendo $\frac{1}{1-q}$. Assim sendo, para $q = 10\%$, a redundância deve ser em média $\frac{1}{1-0,1} \approx 1,111$. Pela Equação 2.6, para $f = \frac{0,7}{10.000}$, $k = 1$ e $e = 2,72$, a credibilidade de P é $Cr_P(P)_{scln} = 1 - \frac{7\times10^{-5}}{1-7\times10^{-5}} \frac{1}{1\times2,72} \approx 0,999974$. Essa credibilidade já é suficiente para satisfazer os três erros aceitáveis em estudo sem a necessidade de votação.

Mesmo não existindo votação, o valor de q não é nulo e as máquinas são testadas. Logo, sabotadores são identificados à medida que os *jobs* vão sendo executados. Pode-se observar na Figura 4.2 que o valor do *slowdown* para primeiro *job* é inferior aos demais nos três níveis de f, ainda que discretamente como quando $f = 5\%$. Isso ocorre, devido ao número de máquinas que executam a computação. No primeiro *job* há mais sabotadores presentes e o *slowdown* se torna inferior aos demais. Depois o *slowdown* passa a convergir e ser superior a 1. Esse fato se torna mais evidente quando se observa $f = 70\%$, que requer mais do sistema para a descoberta de sabotadores. Percebe-se claramente que o *slowdown* do primeiro *job* que é inferior a 1, como mostra a Figura 4.2(c).

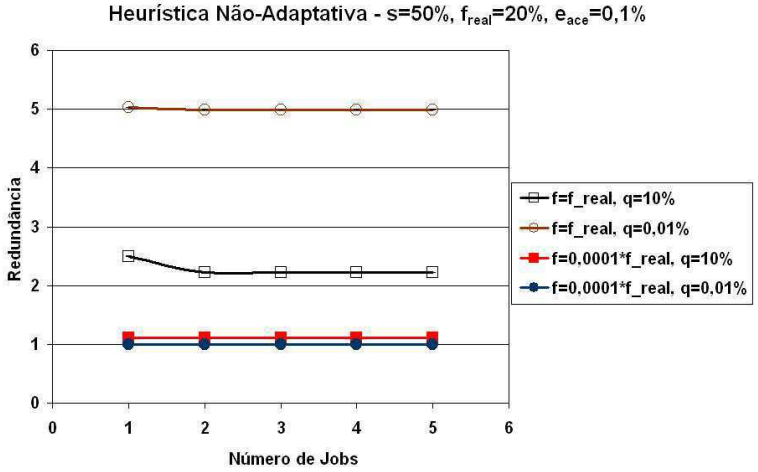
Como é de se esperar, existe erro ao final da computação, no entanto, devido à escala, ele não está presente na Figura 4.3. A Figura 4.4 apresenta o erro obtido quando $f = f_{real} \times 10^{-4}$ e $q = 10\%$. Ele não satisfaz a condição do erro aceitável em todos os casos, levando-se em consideração os intervalos de confiança, mas apresenta uma drástica redução, quando comparado ao mesmo valor de f com $q = 0,01\%$. Porém, eventualmente, todos os sabotadores serão identificados com a atuação do mecanismo de *spot-checking*.

Os estudos para os demais cenários da heurística não-adaptativa apresentaram os mesmos comportamentos com relação a custos, desempenho e corretude ao se variar f e q. O ponto de convergência da redundância e do *slowdown* no pior caso foi o terceiro *job*.

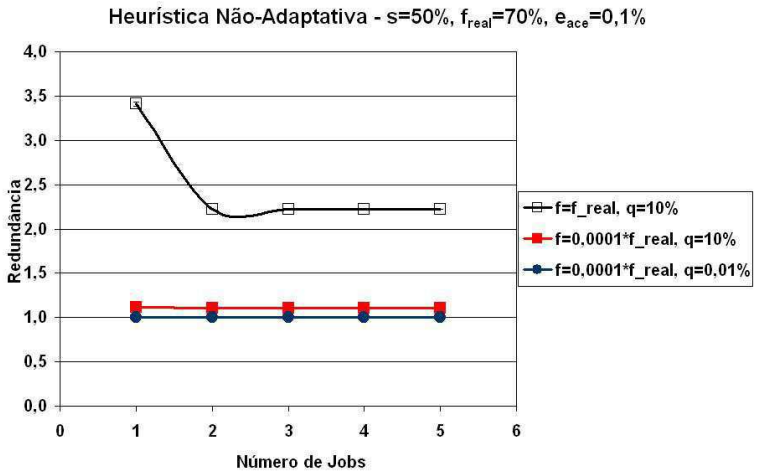
Quando ocorre *spot-checking* e votação, os custos de redundância eventualmente irão cair para 1 mais a redundância do mecanismo de *spot-checking* $(1+\frac{1}{1-q})$ quando as credibilidades



(a) $f_{real} = 5\%$

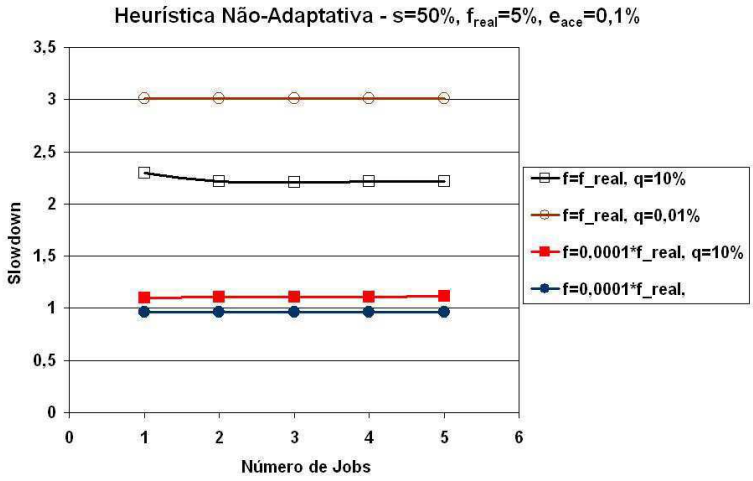


(b) $f_{real} = 20\%$

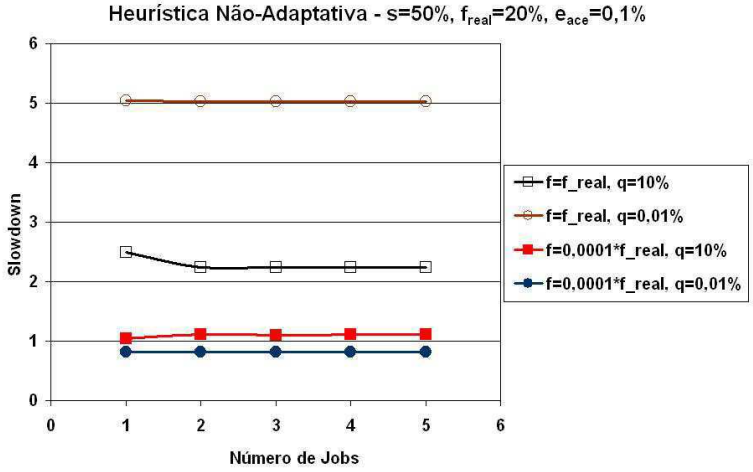


(c) $f_{real} = 70\%$

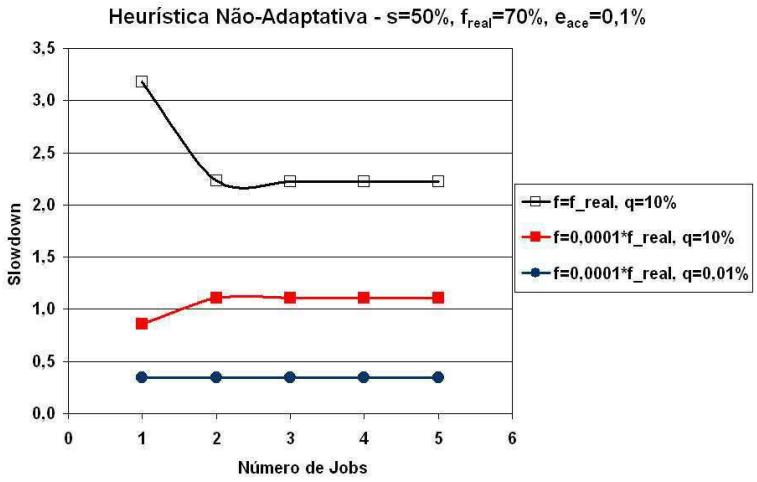
Figura 4.1: Redundância média nos 5 primeiros *jobs* da heurística não-adaptativa para $s = 50\%$, $f_{real} = \{5\%, 10\%, 20\%\}$, $\varepsilon_{ace} = 0,1\%$, $f = \{0,0001 \cdot f_{real}, f_{real}\}$, $q = \{0,01\%, 10\%\}$.



(a) $f_{real} = 5\%$

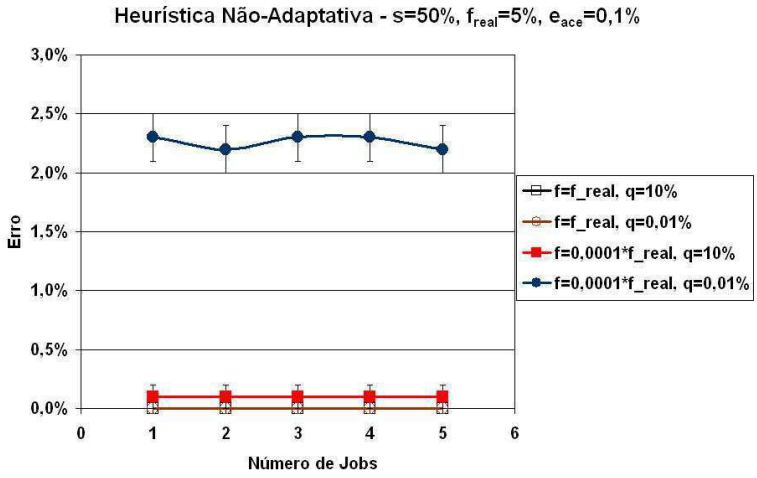


(b) $f_{real} = 20\%$

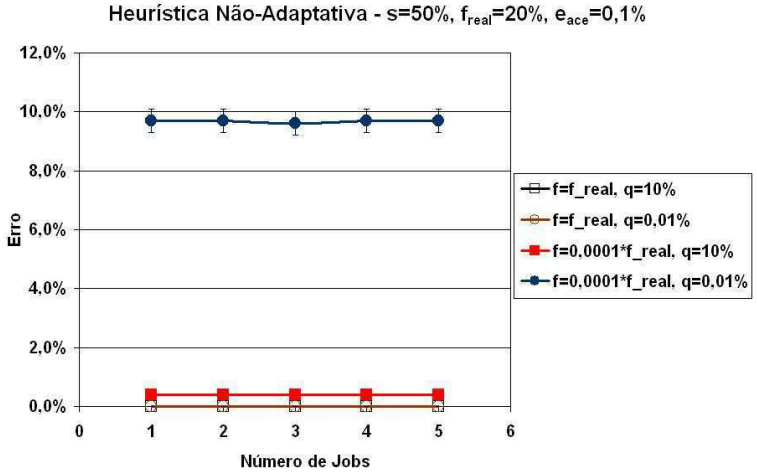


(c) $f_{real} = 70\%$

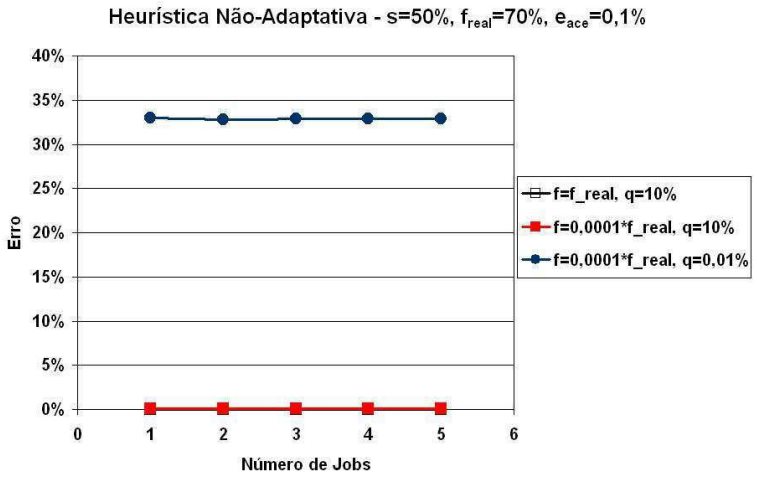
Figura 4.2: *Slowdown* médio nos 5 primeiros *jobs* da heurística não-adaptativa para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$, $f = \{0,0001 \cdot f_{real}, f_{real}\}$, $q = \{0,01\%, 10\%\}$.



(a) $f_{real} = 5\%$



(b) $f_{real} = 20\%$



(c) $f_{real} = 70\%$

Figura 4.3: Erro médio obtido nos 5 primeiros *jobs* da heurística não-adaptativa para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$, $f = \{0,0001 \cdot f_{real}, f_{real}\}$, $q = \{0,01\%, 10\%\}$.

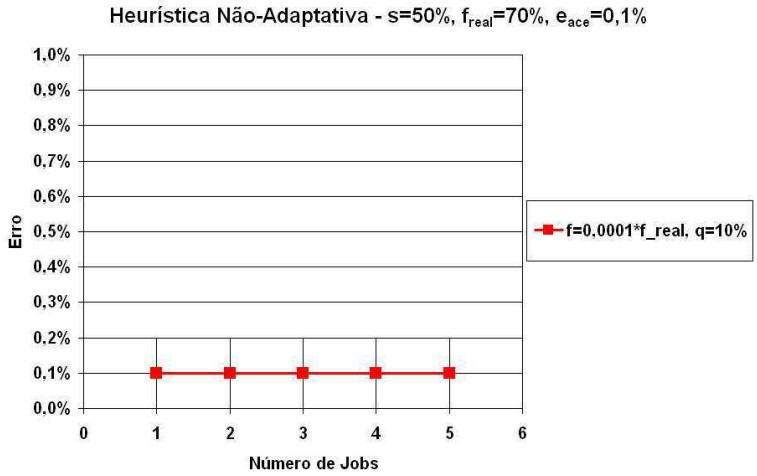


Figura 4.4: Erro médio obtido nos 5 primeiros *jobs* da heurística não-adaptativa para $s = 50\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = 0,1\%$, $f = 0,0001 \cdot f_{real}$, $q = 10\%$.

das máquinas atingirem o limiar de credibilidade. Isso tende a ser aproximadamente no mesmo período, pois todas as máquinas têm a mesma probabilidade de aumentarem suas credibilidades. Todavia, a convergência do *slowdown* depende apenas do instante em que todos os sabotadores são descobertos. Da mesma forma ocorre com o erro.

Este estudo não avaliou $f > f_{real}$. Diante do conhecimento adquirido a partir dessas análises, pode-se concluir que essa condição não causaria impacto na corretude das tarefas, mas iria requerer do mecanismo de escalonamento um custo maior e, consequentemente, causaria perda de desempenho.

4.4.2 Comportamento das Heurísticas de Escalonamento Adaptativas

Introdução

Esta Seção avalia as duas heurísticas adaptativas apresentadas nas Seções 3.3.2 e 3.3.3 do Capítulo 3. O método de escalonamento da *heurística adaptativa com número de máquinas restrito* está identificado nos gráficos como ***adaptativo*** e o da *heurística adaptativa gulosa* como ***adaptativo+***.

As heurísticas adaptativas foram simuladas sob os mesmos cenários da heurística não-adaptativa, cujos resultados foram apresentados na Seção 4.4.1. Os resultados obtidos para as métricas representam a média no escalonamento para 5 *jobs* com herança de credibilidade,

replicados 35 vezes e com intervalos de confiança de 95%.

Para compreender os resultados, vale a pena revisar o método empregado no escalonamento das tarefas para cada heurística. A heurística adaptativa com número de máquinas restrito não escalona tarefas a todas as máquinas disponíveis, mas apenas às máquinas com credibilidade igual ou superior a θ e a uma porção m_d de máquinas desconhecidas. Quanto menor for o valor de m_d, menor será f, conforme o Algoritmo 4. Nos casos em estudo $m_d = 1$. Isso quer dizer que uma nova máquina desconhecida apenas é adicionada ao conjunto possível de máquinas escalonáveis se uma máquina inicialmente desconhecida que fazia parte do escalonamento atingir o limiar de credibilidade, mantendo o número de máquinas desconhecidas igual ou inferior a m_d.

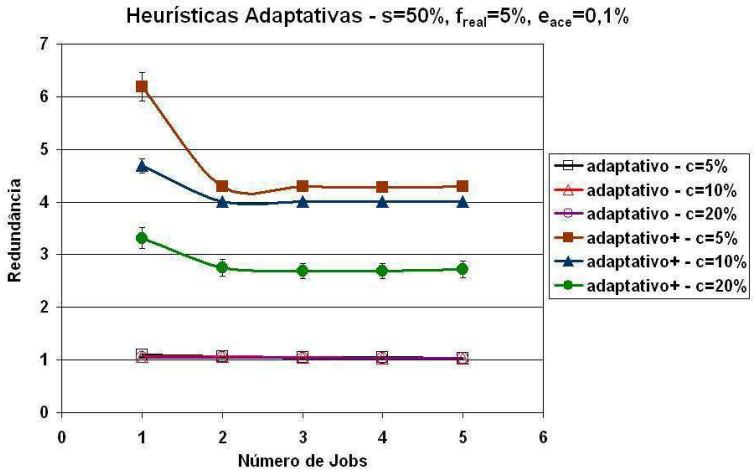
A heurística adaptativa gulosa utiliza todas as máquinas disponíveis para escalonar as tarefas. Por não procurar restringir f no escalonamento, adota um f muito conservador com base nas informações que o escalonador tem sobre o sistema. O valor de f é calculado com base no número de máquinas conhecidas como confiáveis e no número de sabotadores descobertos, o que tende a estimar um f alto quando não se têm muitas máquinas confiáveis ou poucos sabotadores identificados.

Ambas as heurísticas utilizam o mesmo mecanismo para estimar um valor adaptativo para q, descrito na Seção 3.4.2. No início q tende a ser conservador, mas com a continuação, o valor tende a decrescer até o ponto de anular o mecanismo de *spot-checking* para máquinas que tenham credibilidades acima do limiar ou quando f é muito pequeno ou quando a máquina já executou inúmeras tarefas e conseguiu permanecer no sistema.

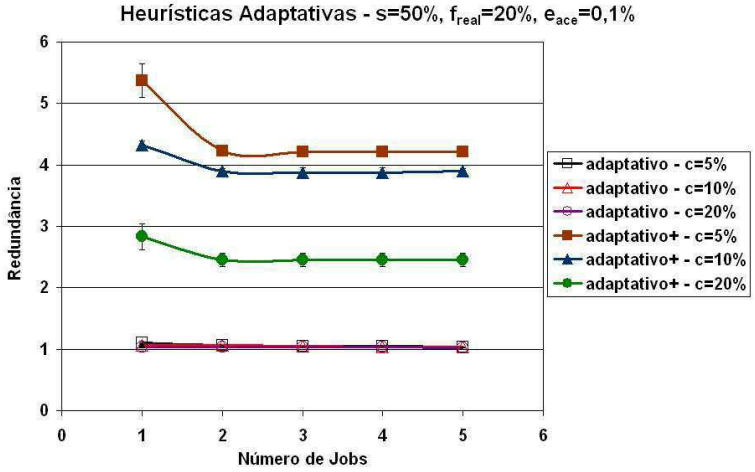
Estudo da Redundância

A heurística adaptativa com número de máquinas restrito tem custos baixos, o valor fica em torno de 1 desde o início da execução dos *jobs*, pois apenas m_d máquinas desconhecidas requerem que as tarefas continuem a ser replicadas após serem executadas por essas máquinas. Nesse estudo, apenas $m_d = 1$ máquinas dessa natureza estão presentes no escalonamento. O baixo custo dessa heurística adaptativa pode ser observado na Figura 4.5.

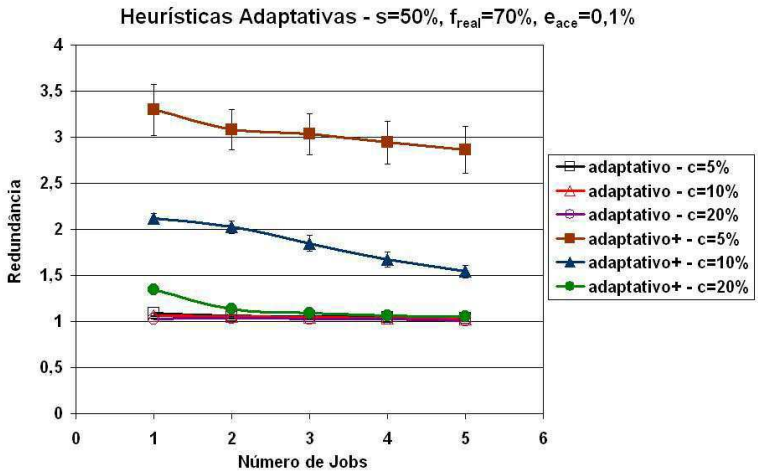
Vale salientar ainda que o valor de f_{real} não interfere na redundância dessa heurística, pois f é calculado com base no número de máquinas que estão de fato sendo utilizadas no escalonamento e não com base no conjunto disponível de máquinas. Isso pode ser percebido



(a) $f_{real} = 5\%$



(b) $f_{real} = 20\%$



(c) $f_{real} = 70\%$

Figura 4.5: Redundância média nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$.

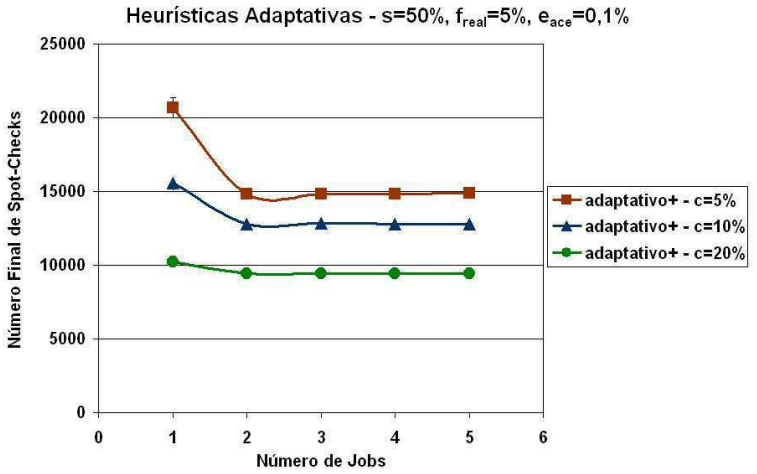
nas Figuras 4.5(a), 4.5(b) e 4.5(c), em que o nível de f_{real} varia entre $f_{real} = 5\%$, $f_{real} = 20\%$ e $f_{real} = 70\%$, respectivamente.

A heurística adaptativa gulosa apresenta maiores custos, como também mostra a Figura 4.5. Diferentemente da heurística adaptativa com número de máquinas restrito no escalonamento, o nível de f_{real} interfere consideravalmente nos custos, em decorrência do método usado para estimar f. Quanto menor for o valor de f_{real}, menos sabotadores são identificados e maior é o f estimado. O outro fator que interfere no valor de f e, consequentemente, nos custos é c, pois o aumento do número de máquinas reconhecidamente confiáveis contribui para a diminuição do valor de f e da redundância. Como esperado, o pior caso de *adaptativo+* encontra-se na Figura 4.5(a), quando $f_{real} = 5\%$ e $c = 5\%$. Em todos os casos, os custos dessa heurística são maiores no primeiro *job*. Isso acontece, porque f é mais conservador no primeiro *job*, pois não há sabotadores na lista negra para que o valor de f seja reduzido, logo a taxa de geração de *spot-checks* é maior e a credibilidade das máquinas também aumenta de forma mais moderada, causando maiores custos. Em seguida, os custos vão sendo reduzidos até o ponto de se estabilizarem. A tendência dos custos das heurísticas adaptativas é convergirem para 1.

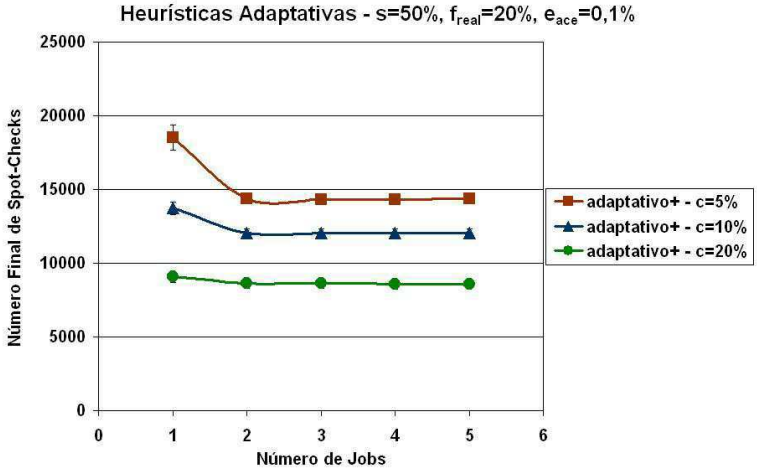
Ainda com relação à heurística adaptativa gulosa, pode-se constatar na Figura 4.5(c) que o melhor caso ocorre quando $f_{real} = 70\%$ e $c = 20\%$. A redundância convergiu rapidamente para 1, indicando que f se tornou favorável para que os 10% de máquinas que não eram sabotadores e também não eram consideradas confiáveis alcançassem o limiar de credibilidade.

Número Final de *Spot-Checks*. A Figura 4.7 apresenta o número de *spot-checks* escalonados para a heurística adaptativa com número de máquinas restrito no escalonamento e a Figura 4.6 apresenta os valores correspondentes para a heurística adaptativa gulosa.

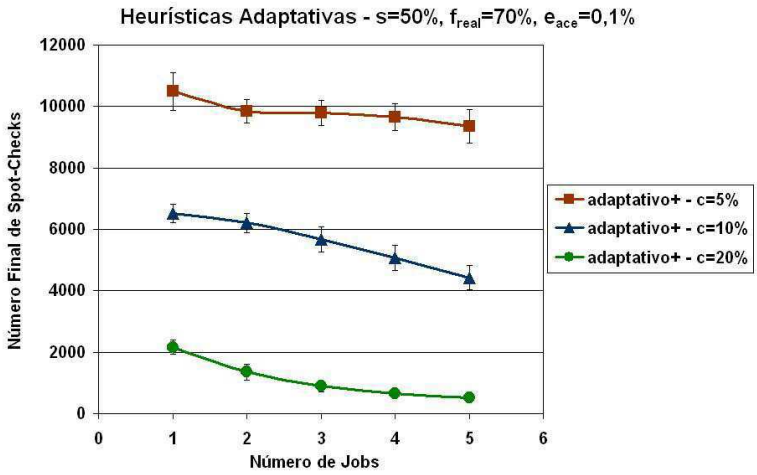
Os dados confirmam que o número de *spot-checks* executados para a heurística adaptativa gulosa é maior no primeiro job, como discutido anteriormente. Quando $f = 5\%$ e $c = 5\%$, o número médio de *spot-checks* no primeiro *job* foi superior a 20.000. Logo, o custo computacional para o primeiro *job* foi aumentado em pelo menos 2 duas vezes devido ao mecanismo de *spot-checking*. O gráfico da Figura 4.5(a) mostra que nesse mesmo ponto a redundância média ficou em torno de 6. De onde se pode concluir que mesmo com esse volume de *spot-checks*, todo o trabalho teve de ser replicado por mais 4 vezes para que



(a) $f_{real} = 5\%$

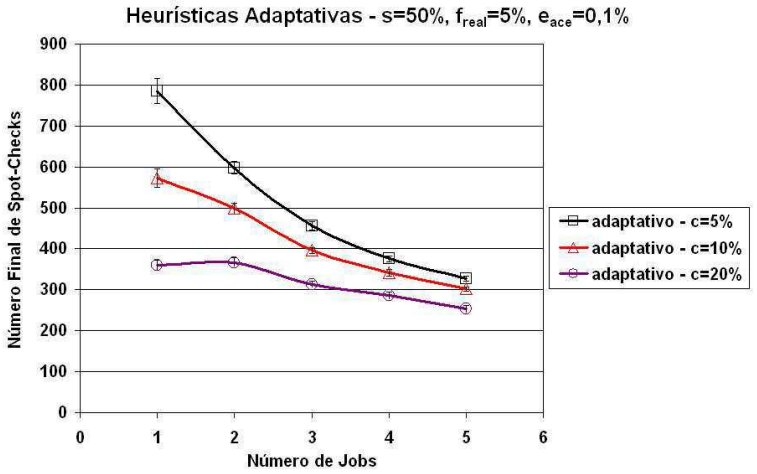


(b) $f_{real} = 20\%$

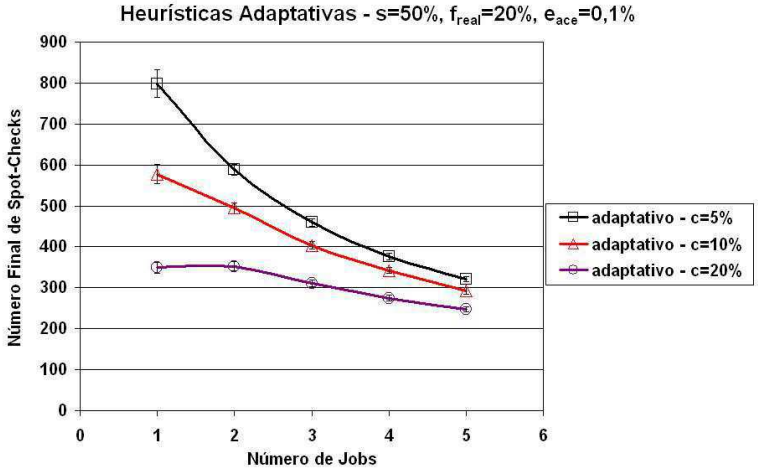


(c) $f_{real} = 70\%$

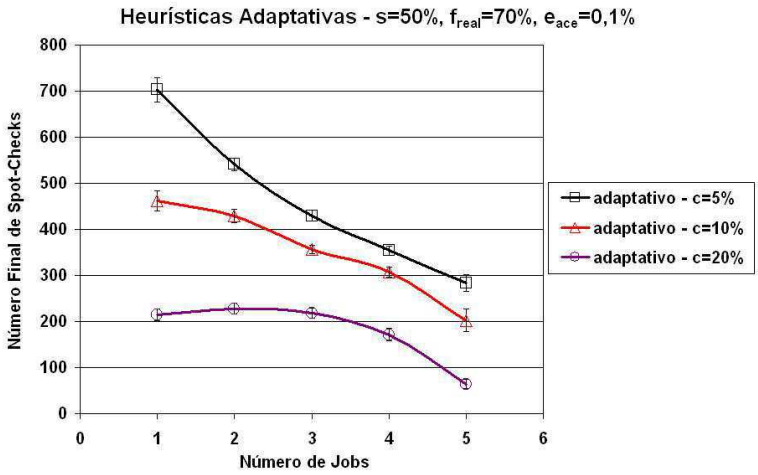
Figura 4.6: Número final médio de *spot-checks* escalonados nos 5 primeiros *jobs* da heurística adaptativa gulosa para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$.



(a) $f_{real} = 5\%$



(b) $f_{real} = 20\%$



(c) $f_{real} = 70\%$

Figura 4.7: Número final médio de *spot-checks* escalonados nos 5 primeiros *jobs* da heurística adaptativa com número de máquinas restrito para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$.

houvesse votação dos resultados e as tarefas pudessem atingir o limiar de credibilidade.

Para compreender o comportamento da taxa de *spot-checks* adaptativa é importante lembrar que o número de *spot-checks* é função de f, de ε_{ace}, de c e do número de tarefas executadas pelas máquinas.

Para *adaptativo+*, o valor de f só variou no primeiro *job*, pois em todos os cenários estudados o número final de sabotadores permanece constante após a execução do primeiro *job*, como mostra a Figura 4.12. Todos eles foram descobertos na fase inicial. Essa é uma das razões pelas quais o valor de q tem a maior redução do primeiro para o segundo *job*.

Como esperado, o maior volume de *spot-checks* para as duas heurísticas adaptativas ocorre quando $c = 5\%$, pois apenas 5% das máquinas são reconhecidamente confiáveis e não precisam ser testadas. À medida que o número de máquinas reconhecidamente confiáveis aumenta, a necessidade de *spot-checks* diminui.

Observando a heurística adaptativa com número de máquinas restrito, pode-se perceber que os volumes de *spot-checks* escalonados para $f_{real} = 5\%$ e $f_{real} = 20\%$ nos três níveis de c foram estatisticamente equivalentes, reforçando a conclusão de que a redundância para essa heurística independe do nível de f_{real}. Esse comportamento, porém, não foi o mesmo quando $f_{real} = 70\%$, pois f_{real} corresponde a uma fração muito alta do número total de máquinas. Uma máquina desconhecida é escolhida por vez para participar do escalonamento. Quando $f_{real} = 70\%$, é alta a probabilidade desta máquina ser um sabotador. Ao se iniciar o processo de averiguação da máquina, é maior a probabilidade da máquina ir para a lista negra e, por conseguinte, do processo de teste desta máquina cessar. O evento de escolher uma máquina sabotadora ocorre com freqüência, pois apenas uma fração de 30% das máquinas não é de sabotadores e, deste montante, se retira uma outra fração c de máquinas que também não são testadas.

Estudo do *Slowdown*

Apesar da redundância da heurística adaptativa com número de máquinas restrito no escalonamento não ter apresentado diferenças entre os níveis de c, observando os gráficos da Figura 4.8 pode-se perceber que existem diferenças para o *slowdown*. O *slowdown* quando $c = 5\%$ é inicialmente menor do que quando $c = 10\%$. O mesmo ocorre entre $c = 10\%$ e $c = 20\%$, porque há um maior número de máquinas presentes no escalonamento aptas a concluirem as

tarefas com 100% de credibilidade. Ao longo das execuções, o *slowdown* nos três níveis vai se estabilizando. Na medida em que a redundância está próxima de 1 e os sabotadores são descobertos, o *slowdown* se aproxima de 1.

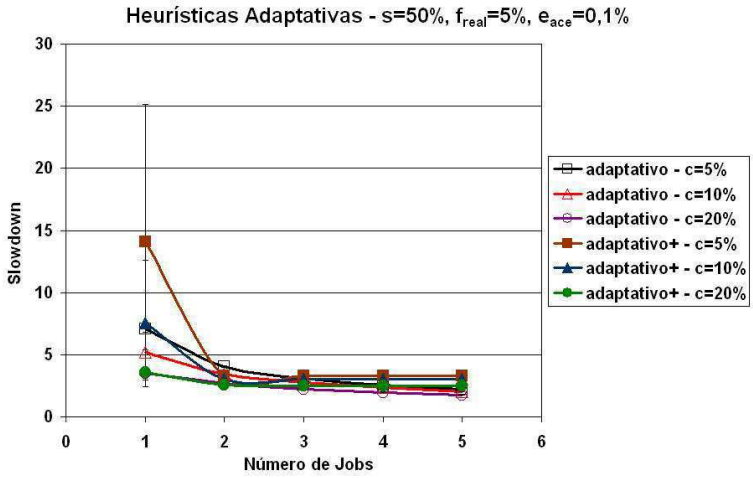
A heurística adaptativa gulosa apresentou maiores custos principalmente para o primeiro *job*, como era de se esperar, o mesmo ocorreu para o *slowdown*. Um fato chama a atenção ao observarmos os gráficos do *slowdown* para *adaptativo+*: a variabilidade do *slowdown* para o nível de $c = 5\%$ no primeiro *job* executado. O intervalo de confiança é muito grande para fornecer os 95% de confiança de que a média real estará dentro do intervalo.

Antes de se investigar com mais profundidade as causas dessa variabilidade, é interessante observar que a adaptação da heurística adaptativa gulosa é muito instável quando há pouco conhecimento sobre o sistema, mas melhora sensivelmente quando ele recebe mais informações. Por exemplo, ao compararmos a razão entre *slowdown* e redundância médios, tem-se que ela é mais divergente quando $c = 5\%$ e quando $c = 20\%$ essa razão é próxima de 1 nos três níveis de f_{real} avaliados.

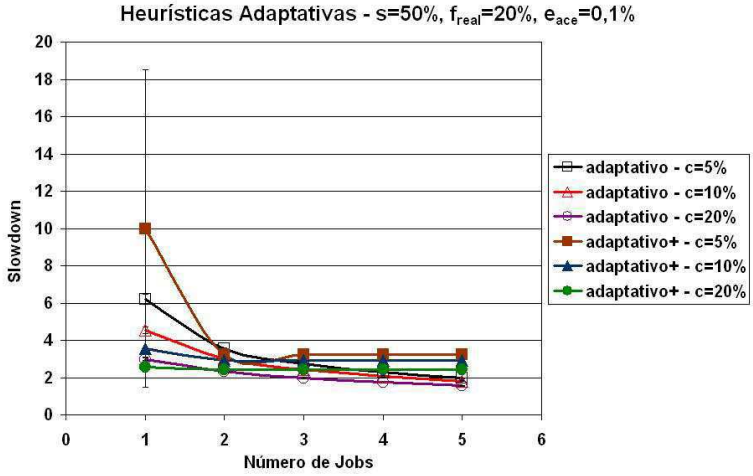
Voltando ao estudo da variabilidade obtida para o *slowdown* quando $c = 5\%$ na heurística adaptativa gulosa, dois fatores externos e aleatórios podem interferir nessa variabilidade de tempo gasto para concluir um *job*. Eles são o número de sabotadores e o número de máquinas confiáveis entre as máquinas que são doadas ao escalonador. O tempo de execução sofre influência da ordem em que os sabotadores são identificados e do tempo necessário para identificar todos eles. Quando os sabotadores são identificados no ínicio da execução das tarefas, o custo de invalidar seus resultados e o tempo para re-escalonar e completar as tarefas é menor do que quando ele já retornou vários resultados e depois é descoberto.

Uma boa maneira de verificar o tempo em que os sabotadores foram descobertos é observar as transições de f durante o processo de escalonamento até que f_{doado} seja zero, ou seja, todos os sabotadores estejam fora do processo. Para apresentar uma noção da alta variabilidade do *slowdown* quando $c = 5\%$, observe como f se comportou na réplica do cenário em estudo no pior caso do *slowdown* do primeiro *job* e compare com a réplica que apresentou o melhor *slowdown* do primeiro *job*. A Figura 4.9 mostra as transições de f ao longo do tempo da réplica com o pior *slowdown* para $c = 5\%$, $s = 50\%$ e $\varepsilon_{ace} = 0,1\%$ em três níveis de f_{real}. A Figura 4.10 corresponde as transições de f para a réplica do melhor *slowdown*.

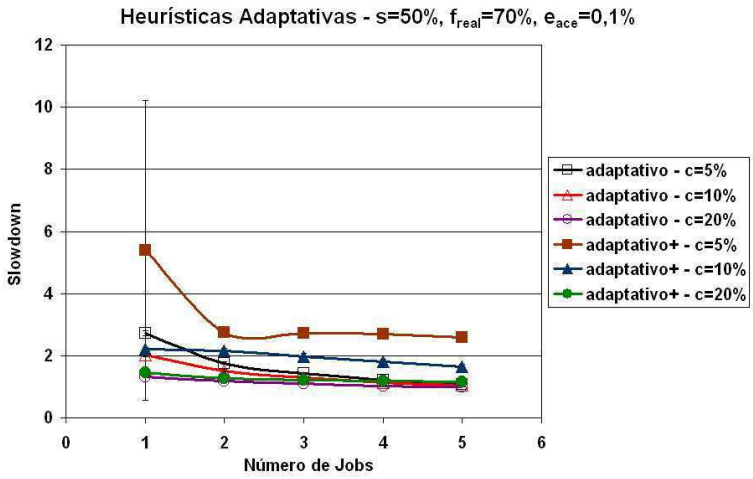
A fórmula geral do *makespan* ideal para executar um *job* é $makespan_{ideal} =$



(a) $f_{real} = 5\%$

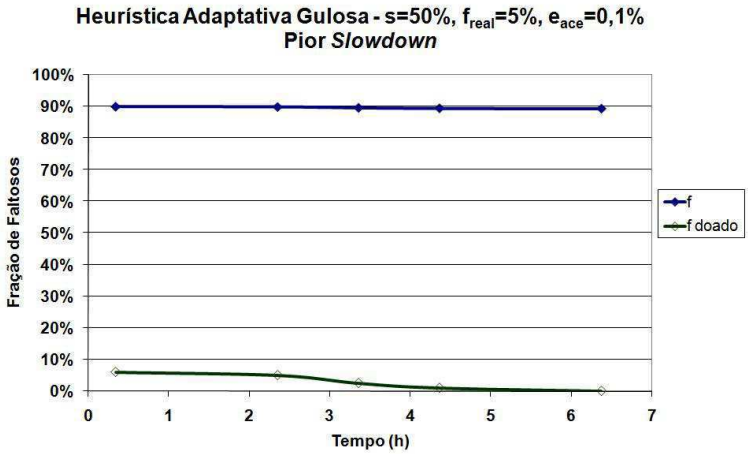


(b) $f_{real} = 20\%$

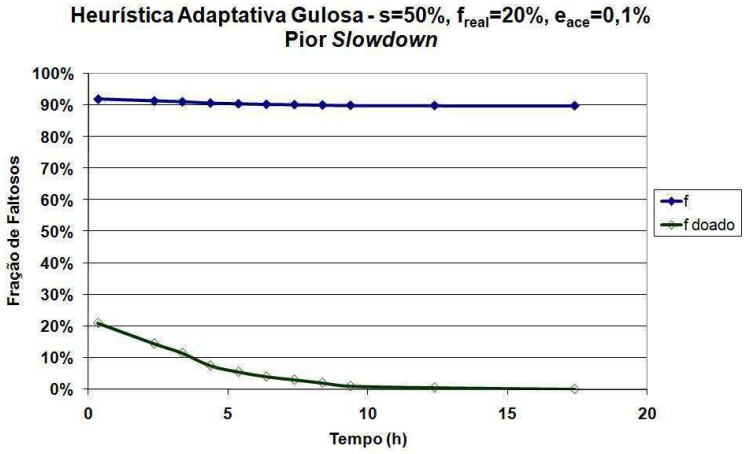


(c) $f_{real} = 70\%$

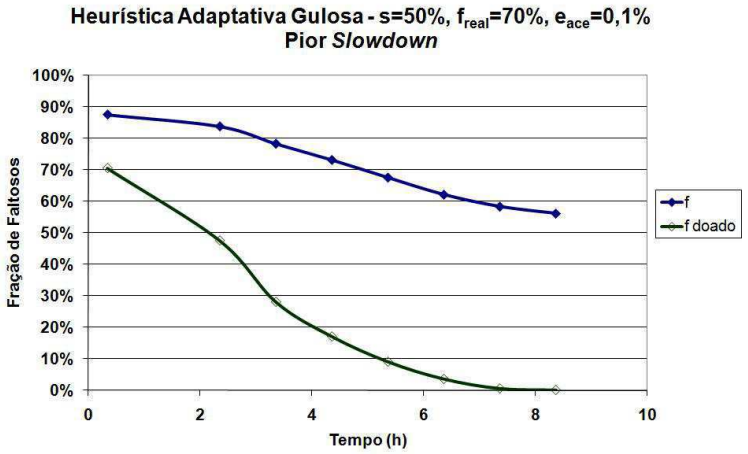
Figura 4.8: *Slowdown* médio nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$.



(a) $f_{real} = 5\%$

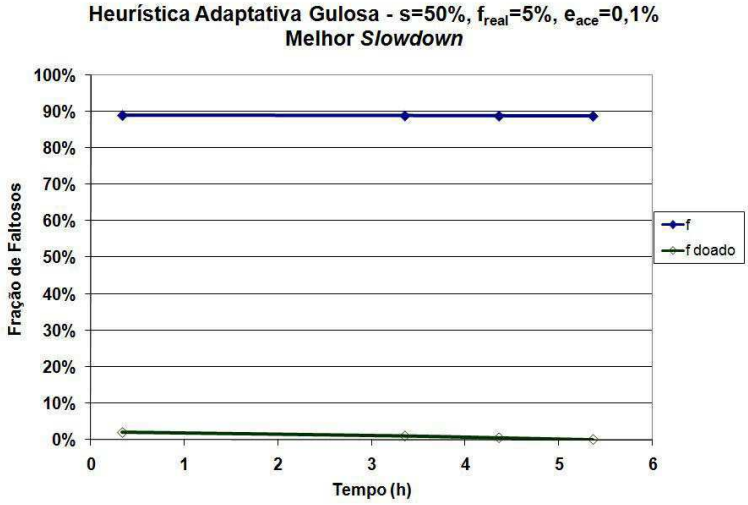


(b) $f_{real} = 20\%$

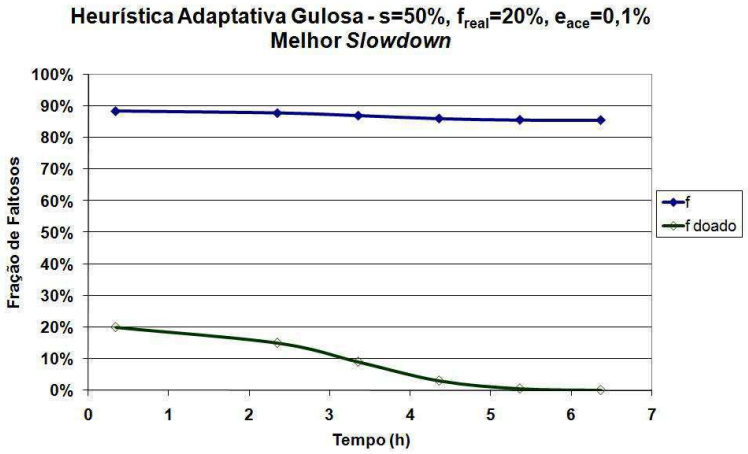


(c) $f_{real} = 70\%$

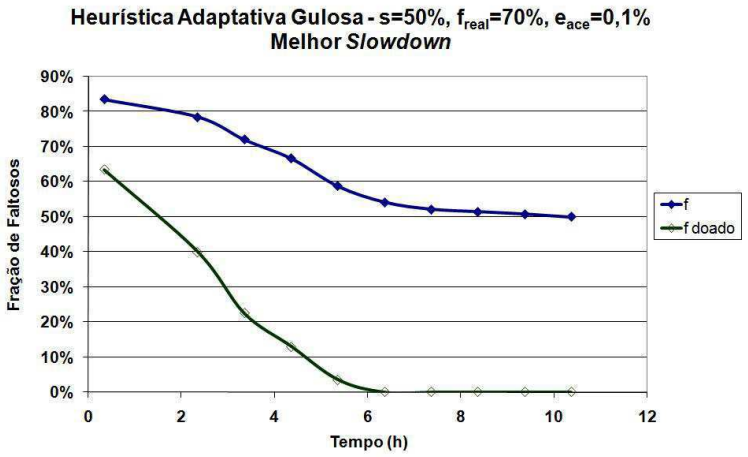
Figura 4.9: Variação de f da réplica que contribuiu para o pior *slowdown* do primeiro *job* da heurística adaptativa gulosa para $c = 5\%$, $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$.



(a) $f_{real} = 5\%$



(b) $f_{real} = 20\%$



(c) $f_{real} = 70\%$

Figura 4.10: Variação de f da réplica que contribuiu para o melhor *slowdown* do primeiro *job* da heurística adaptativa gulosa para $c = 5\%$, $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$.

$\frac{n_{tarefas}}{N \cdot (1 - f_{doado})} \times t_{tarefa}$. O maior *makespan* entre os valores de f_{real} ocorre quando $f_{real} = 5\%$ e equivale a $makespan_{ideal} \approx \frac{10.000}{200 \cdot (1-(0,05))} \times (1h) \approx 53h$. Pode-se confirmar, observando os gráficos, que o valor de f_{doado} em todos os cenários estudados chega a zero ainda no primeiro *job*, pois todos os tempos são inferiores ao *makespan* necessário para executar um *job*. Isso mostra que todos os sabotadores foram identificados ainda no primeiro *job*.

Há uma diferença no tempo gasto para identificar todos os sabotadores quando se compara a pior à melhor réplica. Como esperado, no melhor caso, os sabotadores foram descobertos ($f_{doado} = 0$) mais rapidamente em relação ao pior *slowdown*. Por exemplo, quando $f_{real} = 20\%$, no pior caso do *slowdown* todos os sabotadores foram descobertos após 17h de computação e para o melhor *slowdown* esse valor ocorreu em cerca de 6h.

Há um caso interessante. Quando $f_{real} = 70\%$, o tempo da última transição de f da réplica com o melhor *slowdown* é maior do que o tempo da última transição de f para a réplica com o pior *slowdown*, como mostram as Figuras 4.10(c) e 4.9(c). Contudo, o valor de f_{doado} na melhor réplica chega a zero em cerca de 6h22min, o que acontece em cerca de 8h22min na réplica com o pior *slowdown*. Isso quer dizer que a quantidade esperada de sabotadores é retirada do escalonamento na réplica com o melhor *slowdown* em torno de 2h antes do mesmo evento ocorrer na réplica que resultou no pior *slowdown*. O fato de continuar a ocorrer transições em f, mesmo após o número esperado de sabotadores ter sido removido, ocorre porque máquinas honestas também foram colocadas na lista negra. Elas foram excluídas após algum sabotador, que contribuiu com o mesmo resultado que elas, ser descoberto, tendo, portanto, seus resultados invalidados e eliminando máquinas corretas na votação.

Estudo do Erro

A *heurística adaptativa gulosa*, por fazer suposições conhecidas e fortes a respeito de f, garante que todos os valores obtidos para o erro são inferiores ao erro aceitável. Como ela assume um f conservador, a taxa de geração de *spot-checks* adaptativa também se comporta de modo conservador, conforme a Equação 3.3 que define q_{adap}. Dessa forma, mais *spot-checks* são gerados, especialmente na fase inicial do escalonamento, e por conseguinte, mais sabotadores são identificados além daqueles que são descobertos pela votação dos resultados. Essa heurística detectou todos os sabotadores logo nos primeiros *jobs* executados em todos

os cenários estudados e não houve a presença de tarefas erradas ao final da computação.

Já a *heurística adaptativa com número de máquinas restrito no escalonamento* possui casos em que a condição do erro aceitável não é sastisfeita. A razão é o baixo valor estimado para f, que contribui para reduzir a taxa de geração de *spot-checks* e para levar a máquina desconhecida a obter altas credibilidades mais rapidamente e atingir o limiar esperado, tendo ainda sabotadores vistos como máquinas confiáveis. Logo, essa heurística falha quando $s \neq 100\%$, ou seja, existe a probabilidade de não se detectar o sabotador e também é tão pior quanto maior for o valor de ε_{ace}, pois o valor de θ é mais indulgente.

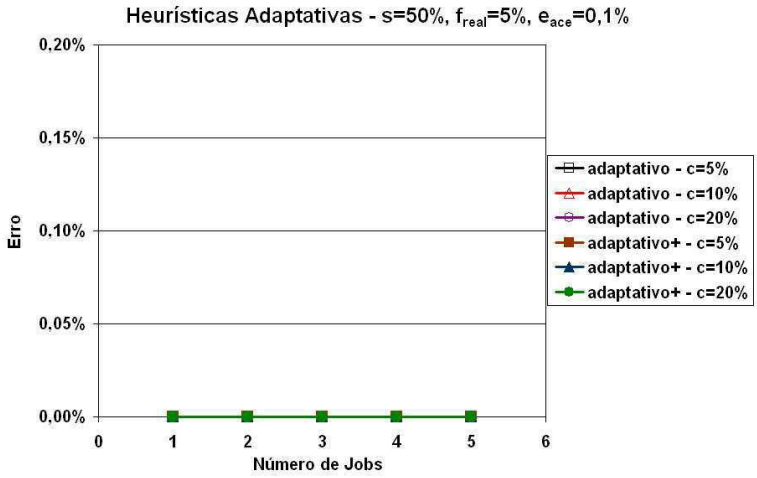
A análise dos cenários possibilitou o diagnóstico de taxas de erro superiores à aceitável para a heurística adaptativa com número de máquinas restrito nos cenários em que $f \geq 20\%$ e $\varepsilon_{ace} \leq 1\%$ para $s < 100\%$. Em todos os casos que $f \leq 5\%$, as taxas de erro são menores do que o erro aceitável. Também não infrigiram a condição esperada para o erro os cenários em que $\varepsilon_{ace} = 0,01\%$ e os cenários em que $s = 100\%$. Esse último caso é um sinal de que todas as máquinas desconhecidas passam por pelo menos 1 teste.

Os piores casos para a taxa de erro, como esperado, foram os cenários 7 e 16, em que se tem a maior fração de sabotadores na grade, $f_{real} = 70\%$, e o maior erro aceitável estudados, $\varepsilon_{ace} = 1\%$, para os níveis $s = 25\%$ e $s = 50\%$. As maiores taxas de erro foram obtidas quando $s = 50\%$, devido à maior probabilidade do resultado estar incorreto, mas também um maior número de sabotadores foi descoberto.

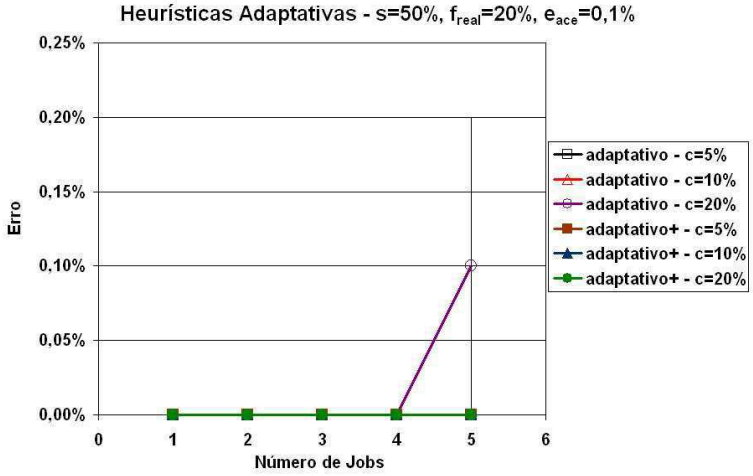
Número Final de Sabotadores. O número final de sabotadores remanescentes no sistema ao final da computação determina o grau de erro que será obtido. A Figura 4.12 apresenta o número médio final de sabotadores descobertos ao final dos cinco primeiros *jobs*.

Como discutido, a heurística adaptativa gulosa identifica todos os sabotadores ainda no primeiro *job*, obtendo uma taxa de erro $\varepsilon = 0$ ao final da computação. As Figuras 4.12(a), 4.12(b) e 4.12(c) demonstram que o valor do número de sabotadores descobertos se manteve constante a partir do primeiro *job* para $f_{real} = 5\%$, $f_{real} = 20\%$ e $f_{real} = 70\%$, respectivamente.

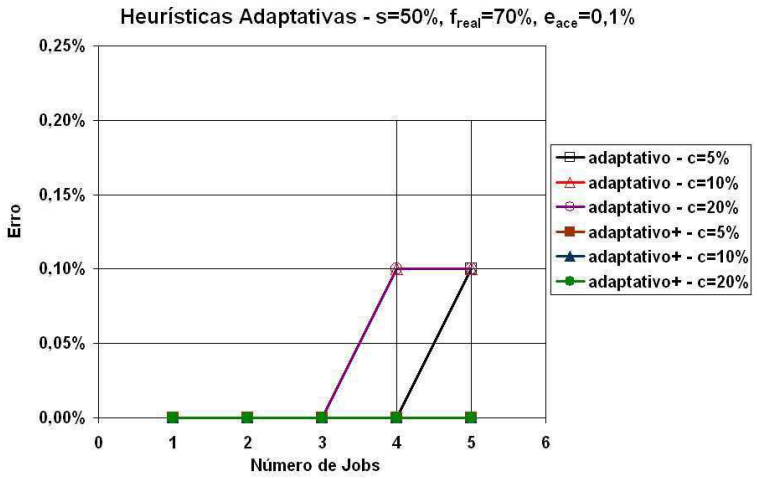
Com relação à heurística adaptativa com número de máquinas restrito, um ponto interessante a se perceber é que o número de sabotadores identificados foi inferior ao de *adaptativo+*, mas o erro obtido não foi tão diferente entre as duas heurísticas. Isso ocorre, porque nem todas as máquinas estão presentes no escalonamento, como é o caso de *adaptativo+*,



(a) $f_{real} = 5\%$

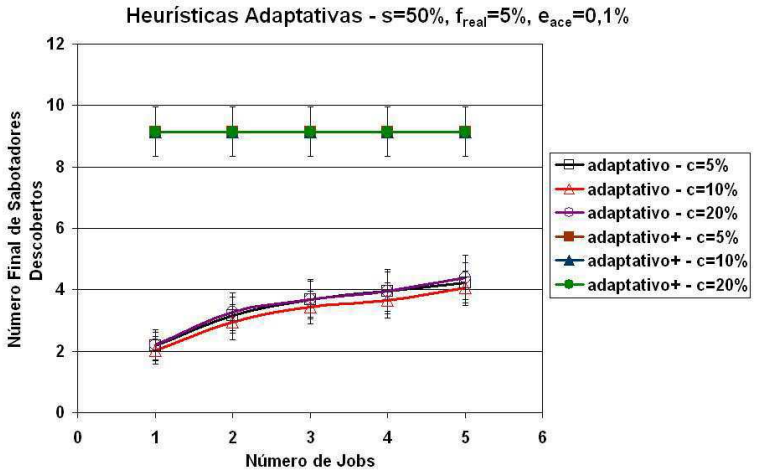


(b) $f_{real} = 20\%$

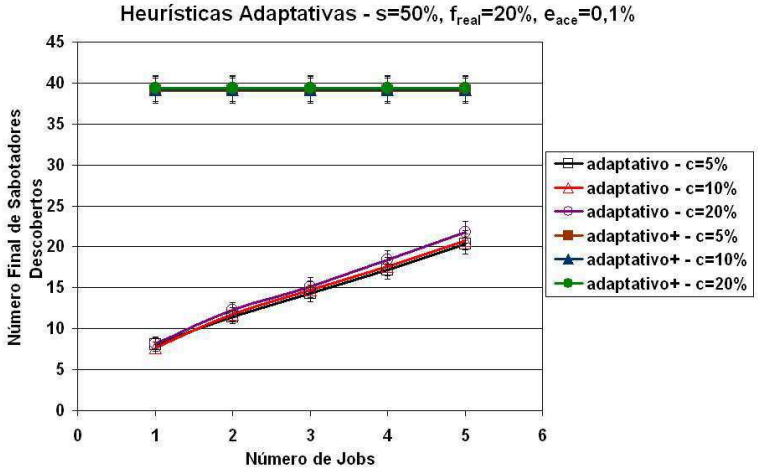


(c) $f_{real} = 70\%$

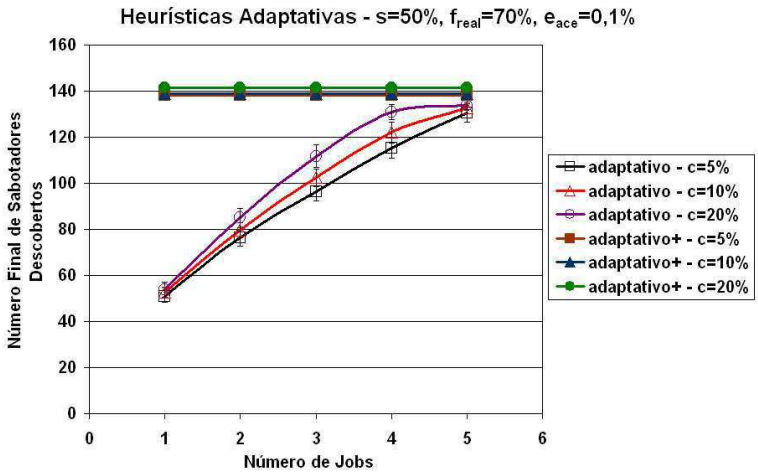
Figura 4.11: Erro médio obtido nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$.



(a) $f_{real} = 5\%$



(b) $f_{real} = 20\%$



(c) $f_{real} = 70\%$

Figura 4.12: Número final médio de sabotadores descobertos nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 1\%$.

logo nem toda a fração de sabotadores entre as máquinas doadas pela comunidade está presente no escalonamento para retornarem resultados incorretos.

Número Final de Máquinas Confiáveis

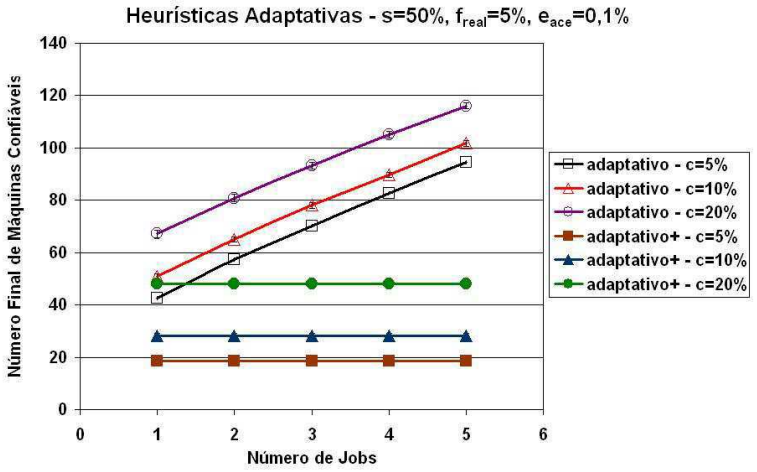
Após as discussões sobre redundância, *slowdown* e erro, que permitiram uma compreensão geral do comportamento das heurísticas e das interações entre os fatores estudados, pode-se também compreender melhor o ganho de credibilidade das máquinas. Os gráficos do número médio de máquinas confiáveis ao final da execução dos *jobs* são apresentados pela Figura 4.13.

Todas as máquinas confiáveis da heurística adaptativa gulosa para $f_{real} = 5\%$ e $f_{real} = 20\%$ que atingiram o limiar de credibilidade o fizeram ainda no primeiro *job*, como mostram as Figuras 4.13(a) e 4.13(b). Duas razões contribuiram para isso: o valor de f alto que conduz a baixas credibilidades e o número de *spot-checks* inicial que é mais alto e permitiu que algumas máquinas obtivessem credibilidades iguais ou superiores a θ. Porém, ao se observar o percentual de máquinas confiáveis ao final, este não é muito superior a c. A Figura 4.13(c), que avalia $f_{real} = 70\%$, mostra que houve um aumento no número final de máquinas confiáveis ao longo das execuções, diferentemente de $f_{real} = 5\%$ e $f_{real} = 20\%$. Como discutido sobre a influência de f na credibilidade das máquinas para a heurística adaptativa gulosa, percebe-se que o valor de f se tornou favorável ao aumento das credibilidades das máquinas, pois 70% das máquinas eram sabotadores e eles foram descobertos, diminuindo o valor de f.

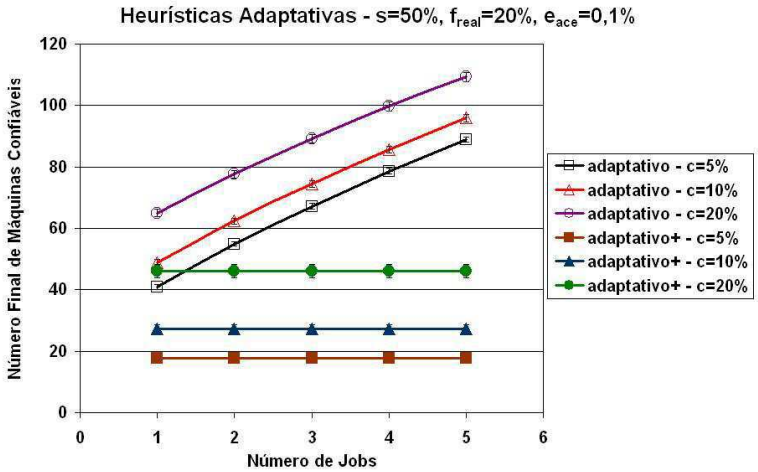
Como esperado, o número final de máquinas confiáveis para a heurística adaptativa com número de máquinas restrito no escalonamento vai aumentando gradualmente à medida que novas máquinas ingressam no escalonamento.

4.4.3 Comportamento a Longo Prazo das Heurísticas Adaptativas

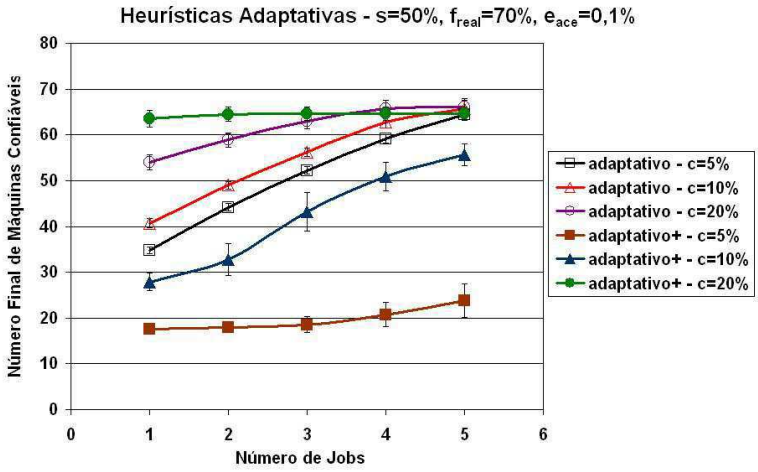
Esta Seção trata do comportamento das heurísticas adaptativas ao longo das execuções de 35 *jobs*, mantendo a herança de credibilidade e a lista negra. O foco desta Seção é estudar o erro obtido para a heurística adaptativa com número de máquinas restrito no escalonamento e o processo de estabilização das métricas avaliadas. Os cenários estudados nessa Seção são



(a) $f_{real} = 5\%$



(b) $f_{real} = 20\%$



(c) $f_{real} = 70\%$

Figura 4.13: Número final médio de máquinas confiáveis nos 5 primeiros *jobs* das heurísticas adaptativas para $s = 50\%$, $f_{real} = \{5\%, 20\%, 70\%\}$, $\varepsilon_{ace} = 0,1\%$.

o sétimo e o oitavo, de acordo com a Tabela 4.1. O cenário 7 representa o pior caso avaliado para o erro da heurística adaptativa com número de máquinas restrito. Os parâmetros estão configurados da seguinte forma: $s = 25\%$, $f = 70\%$ e $\varepsilon_{ace} = 1\%$. Para acompanhar a evolução da heurística, ela será estudada juntamente com um cenário de erro aceitável menor e que também apresenta erro ao final da computação, nesse caso é o cenário 8, em que $\varepsilon_{ace} = 0,1\%$.

Estudo da Redundância

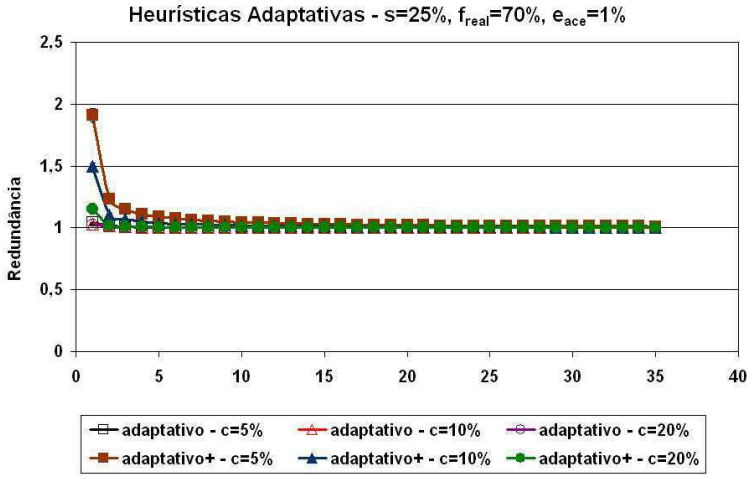
Quando $\varepsilon_{ace} = 1\%$ a redundância de todos os cenários converge para 1 após o quinto *job*, como pode ser observado na Figura 4.14(a). Para $\varepsilon_{ace} = 0,1\%$, o custo de convergência é maior especialmente para *adaptativo+* com $c = 5\%$ e $c = 10\%$, onde a redundância se estabiliza após a execução de 15 e 26 *jobs*, respectivamente, como mostra a Figura 4.14(b).

Estudo do *Slowdown*

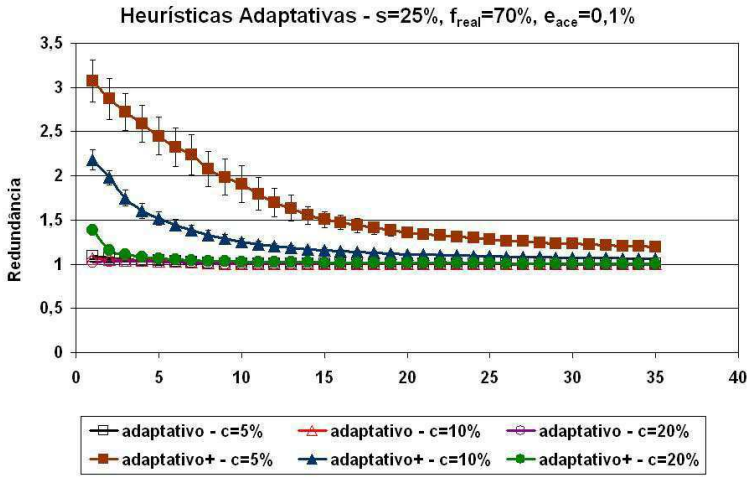
Sempre que o *slowdown* é inferior a 1 é um sinal de que estão sendo executadas tarefas em máquinas maliciosas, portanto o mecanismo de escalonamento não conseguiu identificar todos os sabotadores. Na Figura 4.15 pode-se observar a ocorrência desse fato. Para os dois estudos de ε_{ace}, após a estabilização do *slowdown*, a heurística *adaptativo* obteve um valor para o *slowdown* inferior a 1. Isso indica que sabotadores presentes no sistema não vão sendo removidos ao longo das execuções, pois o valor de q se tornou muito baixo para que o mecanismo de *spot-checking* pudesse causar impacto e as credibilidades das máquinas estão altas o suficiente para os resultados não precisarem de votação. No caso em que $\varepsilon_{ace} = 1\%$, o ponto de convergência do *slowdown* é em torno de 0,5, como mostra a Figura 4.15(a). Como a redundância é em torno de 1, tem-se que as tarefas estão sendo executadas por um número de máquinas que é cerca do dobro do número de máquinas realmente honestas.

Estudo do Erro

A Figura 4.15 apresenta o erro obtido após a execução de 35 *jobs* em um ambiente com $s = 25\%$, $\varepsilon_{ace} = \{1\%, 0,1\%\}$ com $f = 70\%$. Pode ser visto que a heurística adaptativa gulosa não apresenta erro ao final da computação por assumir um f inicial conservador e eliminar sabotadores desde o início do processo. No entanto, a heurística adaptativa com

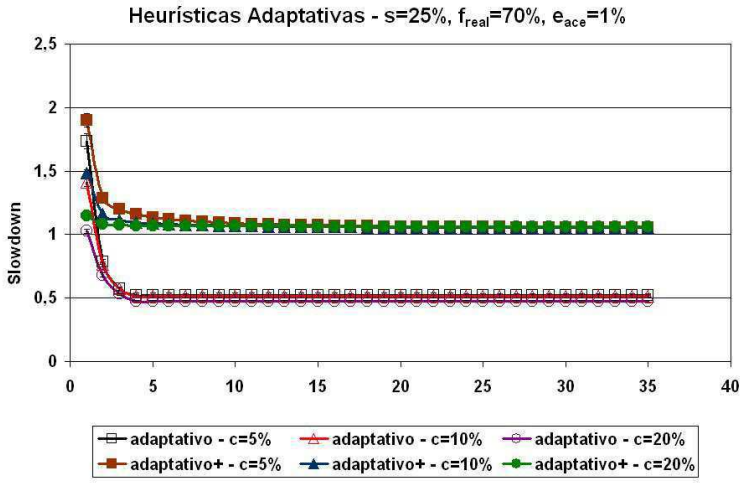


(a) $\varepsilon_{ace} = 1\%$

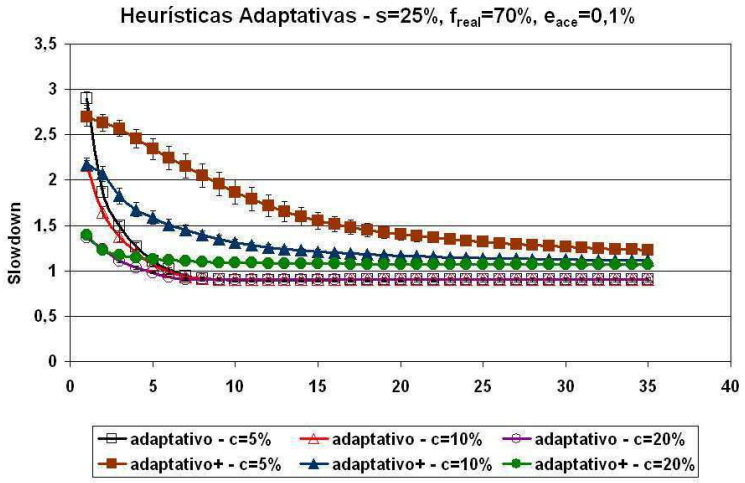


(b) $\varepsilon_{ace} = 0,1\%$

Figura 4.14: Redundância média nos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0,1\%\}$.



(a) $\varepsilon_{ace} = 1\%$



(b) $\varepsilon_{ace} = 0,1\%$

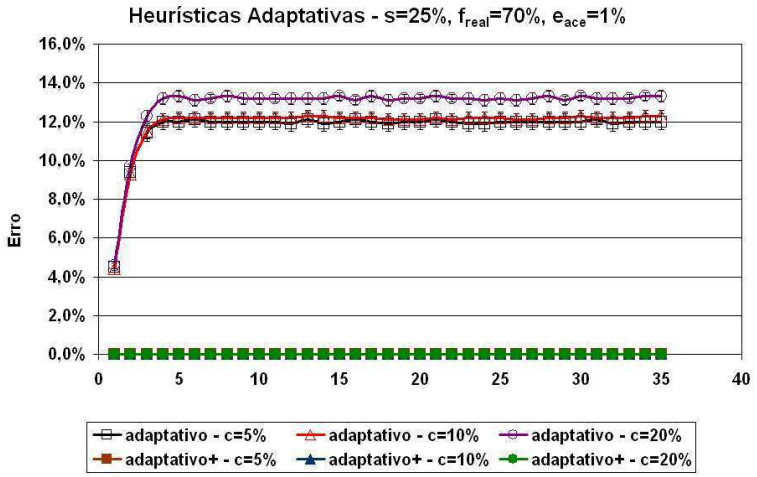
Figura 4.15: *Slowdown* médio nos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0,1\%\}$.

número de máquinas restrito no escalonamento, conforme discutido para o *slowdown* apresentado por ela, não consegue eliminar todos os sabotadores antes que eles alcancem o valor de credibilidade esperado quando a probabilidade de um sabotador retornar um resultado correto é alta, ou seja, s é baixo. Portanto, sem o mecanismo de votação dos resultados para as máquinas com credibilidade igual ou superior a θ e sem o mecanismo de *spot-checking*, todos os resultados provenientes de sabotadores são aceitos. A taxa de erro também apresenta pontos de estabilização. Para $\varepsilon_{ace} = 1\%$ são necessários cerca de 5 *jobs* e para $\varepsilon_{ace} = 0,1\%$ fica em torno do décimo *job*, como mostram as Figuras 4.16(a) e 4.16(b), respectivamente.

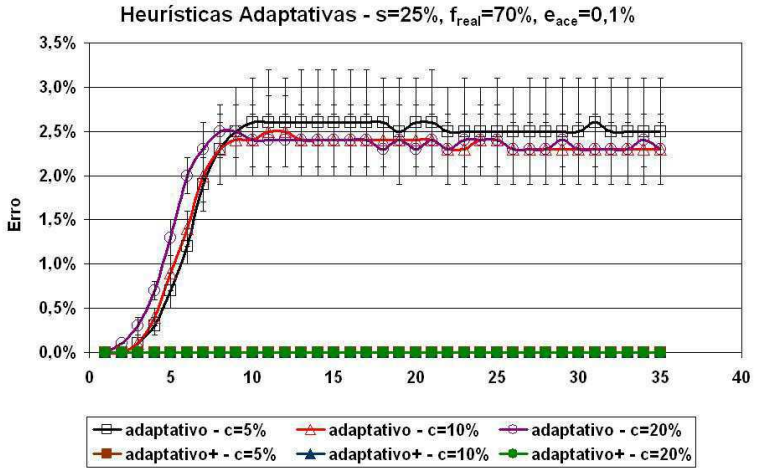
Número Final de Máquinas Confiáveis

Mais detalhes podem ser estudados para exemplificar o comportamento a longo prazo das heurísticas adaptativas. Um bom ponto para a compreensão do que ocorre é observar o número final de máquinas confiáveis a cada *job*, ou seja, o número de máquinas com credibilidade igual ou superior a θ por *job*. Observe na Figura 4.17(a) que, quando $\varepsilon_{ace} = 1\%$, a heurística *adaptativo+* convergiu já ao final do primeiro *job* e serve como referencial para o número médio de máquinas confiáveis que deve ser esperado. Ao observar-se a heurística *adaptativo*, percebe-se que ela converge a partir do quarto *job*, mas confia em cerca de duas vezes mais máquinas do que deveria. Ela apresenta um caso pior quando $c = 20\%$, pois quanto maior o número de máquinas previamente confiáveis, menor é o valor estimado para f e, consequentemente, é o valor atribuído às credibilidades das máquinas.

Quando $\varepsilon_{ace} = 0,1\%$, apresentado na Figura 4.17(b), a heurística *adaptativo+* converge a partir do segundo *job* para $c = 20\%$, no vigésimo *job* para $c = 10\%$ e para $c = 5\%$, mesmo após as 35 execuções, a heurística ainda continua identificando máquinas confiáveis e não converge ainda, mas se aproxima do valor esperado, pois, como discutido, para baixos valores de ε_{ace} e pouco conhecimento sobre máquinas previamente confiáveis, a heurística *adaptativo+* tende a atribuir baixas credibilidades às máquinas e depende da votação para concluir as tarefas. Já a heurística *adaptativo* converge a partir do sétimo *job* e ainda considera máquinas maliciosas como confiáveis. Ao invés de confiar em cerca de 65 máquinas, ela confia em média em torno de 75 máquinas.

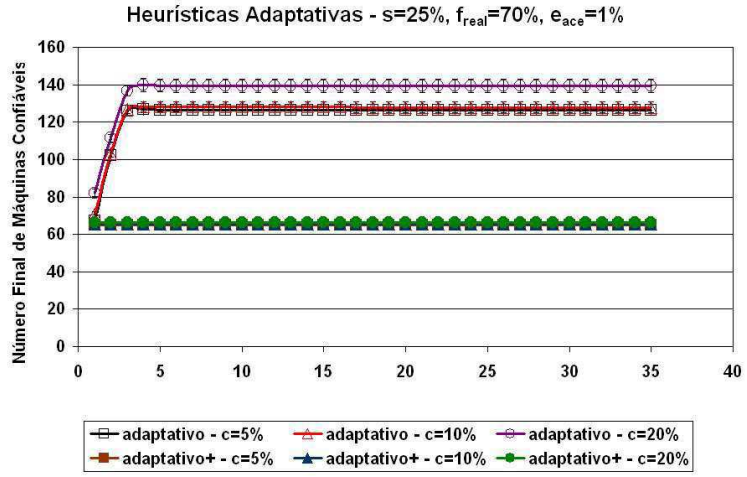


(a) $\varepsilon_{ace} = 1\%$

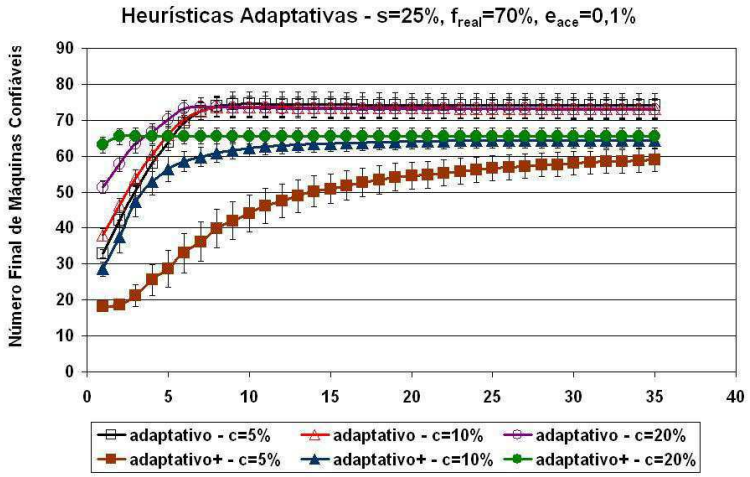


(b) $\varepsilon_{ace} = 0,1\%$

Figura 4.16: Erro médio nos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0,1\%\}$.



(a) $\varepsilon_{ace} = 1\%$

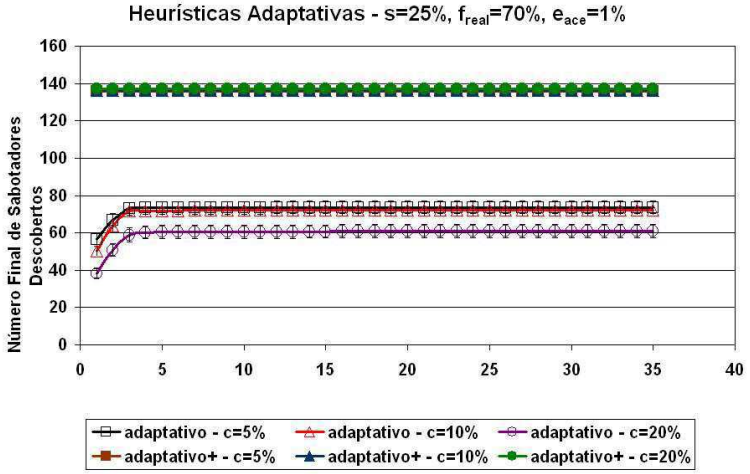


(b) $\varepsilon_{ace} = 0,1\%$

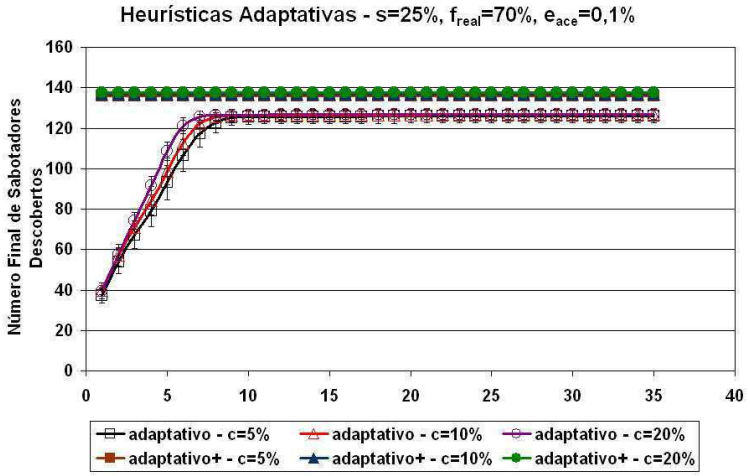
Figura 4.17: Número médio de máquinas confiáveis ao final dos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0,1\%\}$.

Número Final de Sabotadores

A heurística *adaptativo+* identificou todos os sabotadores desde o primeiro *job* para os dois níveis de ε_{ace}, mesmo não tendo identificado todas as máquinas honestas. Já a heurística *adaptativo* estabilizou a descoberta de sabotadores a partir do quinto *job* para $\varepsilon_{ace} = 1\%$ e no décimo *job* para $\varepsilon_{ace} = 0,1\%$, como mostram as Figuras 4.18(a) e 4.18(b). Pode-se perceber que ela se estabilizou e não conseguiu descobrir quais eram todos os sabotadores ao final dos 35 *jobs*. Isso pode ser percebido nas taxas de erro da Figura 4.16.



(a) $\varepsilon_{ace} = 1\%$



(b) $\varepsilon_{ace} = 0,1\%$

Figura 4.18: Número médio de sabotadores descobertos ao final dos 35 primeiros *jobs* das heurísticas adaptativas para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = \{1\%, 0,1\%\}$.

4.4.4 Comportamento a Longo Prazo da Heurística Adaptativa com Número de Máquinas Restrito com Valor Mínimo para q

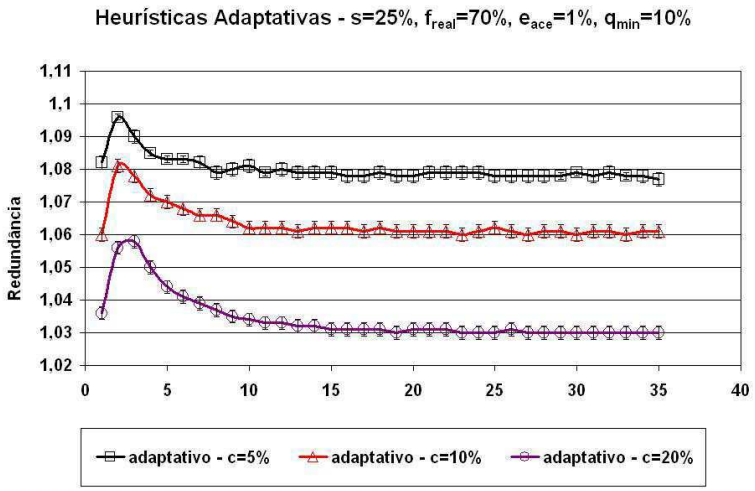
Nos estudos feitos sobre a heurística adaptativa com número de máquinas restrito, pôde-se observar que em determinados casos o valor estimado para q anula o mecanismo de *spot-checking* e a heurística se estabiliza ainda na presença de sabotadores, passando a aceitar resultados incorretos e perdendo a garantia do erro aceitável.

Os resultados a seguir são o produto de simulações a longo prazo da heurística adaptativa com número de máquinas restrito no escalonamento, utilizando o mesmo processo de estimativa adaptativa de q, mas com um valor mínimo de 10%. Os gráficos da Figura 4.19 foram escolhidos para este estudo por apresentarem o caso em que a heurística teve o pior erro quando executada sem o valor mínimo.

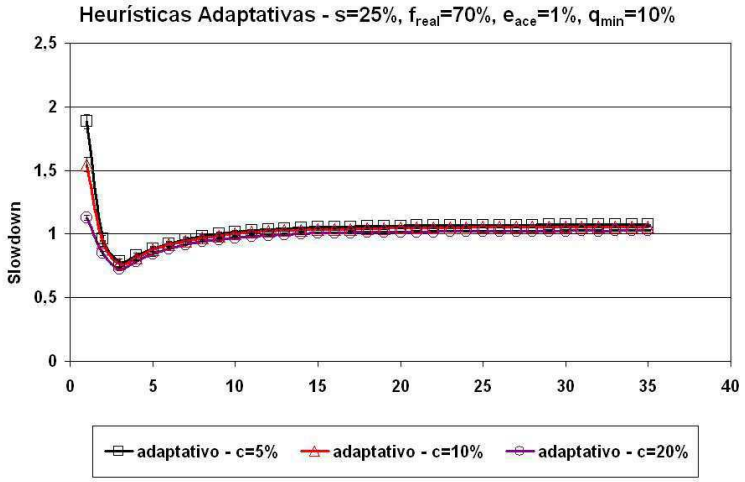
A Figura 4.19(c) mostra que o ε_{ace} é garantido a partir do décimo terceiro job. O impacto nos custos causado pelo limite mínimo de q foi apenas cerca de 10% de redundância extra unicamente para as máquinas que inicialmente eram desconhecidas, uma vez que as máquinas reconhecidamente confiáveis não recebem tarefas de averiguação. Portanto, todos os custos de redundância foram inferiores a 1,11. A partir do décimo quinto *job*, a heurística já havia atingido seu ponto de convergência para todos os níveis estudados de c, conforme mostra a Figura 4.19(a).

Pode-se perceber que os valores para o *slowdown* entre o segundo e o nono *jobs* foram inferiores a 1, indicando que sabotadores foram aceitos como máquinas confiáveis, de acordo com a Figura 4.19(b). É interessante observar que no primeiro *job* o *slowdown* é mais alto e o ε_{ace} é garantido, porque no início a heurística se mantém com um conjunto restrito de máquinas, majoritariamente as previamente confiáveis. Além disso, a taxa de geração de *spot-checks* inicial é mais conservadora, daí o problema anterior de serem aceitos resultados incorretos em uma margem de erro superior à aceitável quando os sabotadores não são descobertos nessa fase inicial.

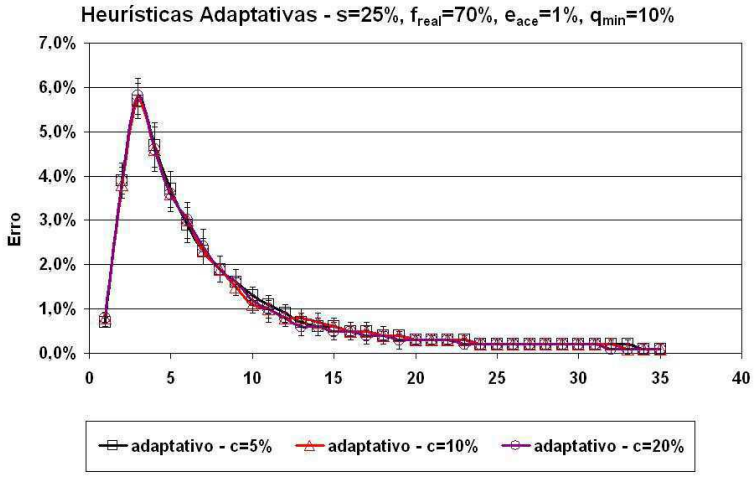
Ao estabelecer-se um valor mínimo para q, as máquinas continuarão sendo testadas e eventualmente todos os sabotadores serão identificados. De acordo com a Figura 4.20, esse momento ocorre depois do décimo *job* e é justamente quando o *slowdown* converge e fica em torno de 1.



(a) Redundância



(b) *Slowdown*



(c) Erro

Figura 4.19: Redundância, *slowdown*, erro médios nos 35 primeiros *jobs* da heurística adaptativa com número de máquinas restrito para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = 1\%$ e valor mínimo para q de 10%.

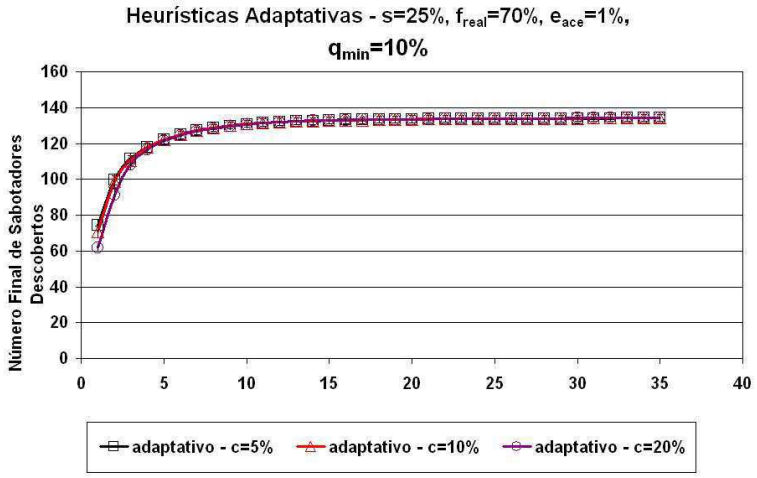


Figura 4.20: Número final de sabotadores médio nos 35 primeiros *jobs* da heurística adaptativa com número de máquinas restrito para $s = 25\%$, $f_{real} = 70\%$, $\varepsilon_{ace} = 1\%$ e valor mínimo para q de 10%.

Para se ter uma idéia de como esse mecanismo atua, as mesmas curvas são encontradas para $s = 50\%$, mas há uma convergência mais rápida, uma vez que também são maiores as chances de se identificar sabotadores. O erro é satisfeito em todos os níveis de c a partir do quinto *job* e fica em torno de zero após o décimo. A redundância e o *slowdown* também convergem no quinto *job*, onde também se pode notar que o número de sabotadores descobertos se estabiliza, podendo restar alguns com determinada variação. De acordo com os intervalos de confiança para o erro, pode-se perceber ainda a possibilidade de erro entre o quinto e o décimo *jobs*, mas depois disso não se detecta mais a presença de erro.

4.5 Considerações sobre as Heurísticas de Escalonamento

4.5.1 Sobre a Heurística de Escalonamento Não-Adaptativa

Esta heurística avalia f com um valor idêntico a f_{real} e outro muito inferior. Em resumo, os resultados demonstram que quando $f = f_{real}$ o erro aceitável é garantido, porque ou os sabotadores são identificados pelo mecanismo de *spot-checking* ou pela votação. Quando f foi muito inferior ao f_{real}, o mecanismo de votação foi anulado, o que limita a descoberta de sabotadores, e a consequente diminuição da taxa de erro, apenas ao mecanismo de *spot-*

checking. Conforme esperado, quanto maior o valor de f, maiores os custos e o impacto no desempenho também.

A avaliação de q também ocasionou impacto nas métricas. Quando $q = 10\%$, o mecanismo de *spot-checking* teve atuação, no entanto, quando avaliado o valor de $q = 0,01\%$, o mecanismo de *spot-checking* foi praticamente anulado e os sabotadores apenas poderiam ser identificados pelo mecanismo de votação. Isso ocasionou um aumento nos custos e degração no desempenho quando $f = f_{real}$, porém, quando $f = f_{real} \times 10^{-4}$, nem o mecanismo de *spot-checking* atuou nem o de votação, resultando em custo baixo custo e bom desempenho, mas altas taxas de erro, $\varepsilon \approx s \cdot f$.

4.5.2 Sobre a Heurística de Escalonamento Adaptativa com Número de Máquinas Restrito

Esta prática apresenta desvantagens. As máquinas maliciosas podem atingir a credibilidade-alvo e serem consideradas confiáveis sem de fato serem. Dois fatores combinados contribuem para isso: m_d e ε_{ace}. As máquinas podem rapidamente atingir a credibilidade esperada se m_d for um número pequeno, pois f também será pequeno. Ter um erro aceitável alto é um fator agravante, pois as máquinas atingirão a credibilidade-alvo mais rapidamente, mesmo sem possuirem os méritos. De acordo com a Equação 2.5, quanto menor for o valor de f e maior o valor de ε_{ace}, maior é a credibilidade atribuída à máquina. Por conseguinte, quanto mais máquinas tiverem credibilidades altas, menor será também o valor de f.

A taxa de geração de *spot-checks* adaptativa também pode causar problema. Uma tarefa, após ser executada em uma máquina que supostamente já passou com sucesso pelo número de *spot-checks* suficiente e atingiu a credibilidade-alvo, é aceita como confiável. Com a continuação, a probabilidade de atribuir uma tarefa de averiguação à máquina se torna bastante pequena, sendo muito difícil excluí-la do escalonamento caso seja maliciosa. Pode-se perceber que no decorrer do tempo o mecanismo de estimativa de reputações poderá não corresponder à credibilidade que realmente deveria ser atribuída à máquina e conduzir o sistema a situações de erro com uma taxa maior do que a aceitável.

No entanto, a avaliação a longo prazo dessa heurística com valor mínimo para q aponta um caminho para a redução das taxas de erro, mantendo o baixo custo e bom desempenho.

4.5.3 Sobre a Heurística de Escalonamento Adaptativa Gulosa

Uma das premissas para a tolerância a faltas baseada em credibilidade é que as credibilidades das máquinas correspondam à probabilidade das máquinas retornarem um resultado correto. Nas discussões sobre a heurística anterior, observa-se que a vantagem de possuir a fração de faltosos controlada no escalonamento também ocasionou vulnerabilidades ao modelo da maneira como foi proposto. Uma vez que nem sempre as credibilidades refletem a intencionalidade da máquina.

Esta heurística continua não supondo um f conhecido nem uma fração limitada para o número de falhas. Baseia-se no conhecimento que o sistema possui, mas de maneira mais forte. Atua, geralmente, com um f alto. Fato que faz com que a credibilidade das máquinas cresça mais lentamente. A grande diferença é que utiliza todo o conjunto disponível de máquinas para realizar replicação de tarefas para diferentes máquinas e, em seguida, votar nos resultados. O mecanismo de votação dos resultados permite obter altas taxas de confiabilidade. Porém, acarreta em custos mais altos também.

O estudo do comportamento a longo prazo dessa heurística mostra que, apesar de ser mais conservadora inicialmente, as máquinas têm suas credibilidades aumentadas com mais freqüência, o que contribuirá para antecipar a convergência da redundância para 1. Todas as máquinas corretas devem atingir a credibilidade-alvo eventualmente ou, ao menos, reduzirem sensivelmente o quórum necessário à votação dos resultados.

Capítulo 5

Trabalhos Relacionados

5.1 Introdução

O trabalho de Sarmenta [Sar02][Sar01] é o fundamento teórico das técnicas de verificação de resultados adotadas neste trabalho de dissertação, porém escalonamento está fora de seu escopo. A principal contribuição desta pesquisa é adaptar seu trabalho de modo a integrá-lo em um serviço de escalonamento para grades computacionais P2P.

Outros trabalhos envolvendo escalonamento tolerante a sabotagem foram propostos. Em particular, dois projetos bastante semelhantes foram encontrados. O primeiro foi proposto por Zhao et al. [ZLG05] e o segundo por Sonnek et al. [SNC06]. Este capítulo discute alguns pontos de cada trabalho e conclui fazendo considerações de como este trabalho se compara ao projetos relacionados.

5.1.1 Zhao et al.

O trabalho Zhao et al. [ZLG05] propõe o uso de **replicação** (votação majoritária) e de esquemas de ***quizes*** para verificação da correção dos resultados. A técnica de *quizes* é bastante semelhante a *spot-checking*, diferindo apenas que enquanto *spot-checks* são tratados como tarefas enviadas independentemente com uma determinada probabilidade, os *quizes* são enviados em um pacote com outras tarefas normais, devendo ser executadas todas juntas. As tarefas do pacote só são aceitas como corretas se todos os *quizes* estiverem corretos. Este projeto foi pioneiro a unir técnicas de estabelecimento de reputações à estratégia de escalona-

mento de grades computacionais P2P, propondo um arcabouço para **escalonamento baseado em confiança**.

Uma característica importante do trabalho de Zhao et al. é que a razão de *quiz* e o fator de replicação são adaptativos. Existe um *módulo verificador de resultados* que executa uma **função de razão de *quiz*** e uma **função de fator de replicação** que recebe o valor da reputação de um trabalhador (máquina) como entrada. Quanto mais confiável for a máquina, menores serão a razão de *quiz* e o fator de replicação. Porém, não entra em detalhes de como essas funções devem ser calculadas e deixa a cargo do desenvolvedor a tarefa de escolher um sistema de reputação para estimar as reputações.

5.1.2 Sonnek et al.

O outro trabalho encontrado é o de Sonnek et al. [SNC06]. Ele propõe apenas o uso de *votação majoritária*, mas com algoritmos de escalonamento que procuram satisfazer uma *probabilidade de correção alvo* (LOC - *likelihood of correctness*). Propuseram quatro algoritmos para formar grupos de redundância que executam uma mesma tarefa (*First-Fit*, *Best-Fit*, *Random* e *Fixed*), procurando satisfazer o LOC alvo. No intuito de limitar as taxas de replicação, adotam um tamanho mínimo ou máximo para os grupos. O resultado será aceito quando for majoritário no grupo de redundância. Entretanto, não mencionam uma maneira eficiente de estimar os tamanhos máximos e mínimos dos grupos de redundância.

5.2 Comparações Entre Projetos de Escalonamento Tolerante a Sabotagem

5.2.1 Métricas Avaliadas pelos Trabalhos

Os trabalhos mensuram algumas métricas que permitem uma noção de comparação entre os trabalhos:

- **Sarmenta:** avalia a redundância (razão entre o número de tarefas alocadas em média e o número de tarefas original da aplicação) e o *slowdown* (razão entre o tempo de execução com redundância e o tempo médio que seria gasto sem redundância);

- **Sonnek et al.:** avaliam a vazão de escalonamento e a taxa de acerto dos resultados (nos gráficos, *success rate*);

- **Zhao et al.:** mensuram o sobrecarga para alcançar um resultado correto e a taxa de acerto (nos gráficos, *accuracy*). Esse sobrecarga significa o percentual de computação extra realizado, ou seja, ter sobrecarga igual a 1 não significa que não houve redundância, mas que 100% do volume original de tarefas foram tarefas extras.

5.2.2 Compartilhamento de Reputações

Quanto ao grau de compartilhamento dos sistemas de reputação dos trabalhos relacionados, Sarmenta [Sar02] e Sonnek et al. [SNC06] adotam sistemas de reputação locais, pois em seus modelos existe um único servidor (ou mestre) que distribui as tarefas entre as máquinas (ou trabalhadores), responsável por administrar os esquemas de verificação de resultados (conforme indicado pela Figura 5.1 em [SNC06]) e estimar as reputações.

Zhao et al [ZLG05], por sua vez, não amarra a necessidade de um servidor central de tarefas, avaliando, inclusive, diferentes sistemas de reputação. Dessa forma, pode haver compartilhamento de reputações entre os pares vizinhos (compartilhamento parcial) ou entre todos os pares da rede (compartilhamento global).

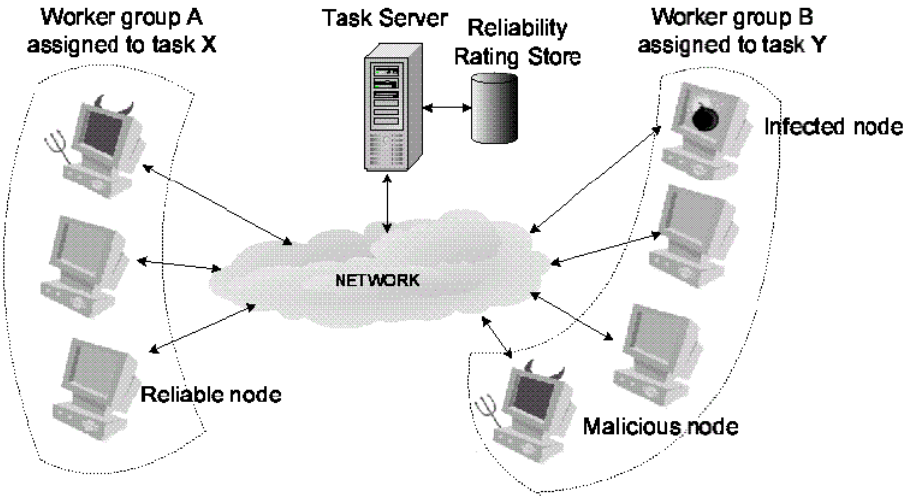


Figura 5.1: Modelo do sistema: um servidor mantém armazenadas as taxas de confiabilidade e as utiliza para atribuir tarefas a grupos de máquinas.

5.2.3 Escalonamento

O modelo de escalonamento proposto por Zhao et at. [ZLG05] submete tarefas em uma grade P2P, possui um módulo verificador de resultados responsável por atualizar os valores de confiança e um escalonador que utiliza sistemas de reputação independentes que podem compartilhar ou não os valores de reputação entre os demais pares da rede de modo transparente, conforme mostra a Figura 5.2 [ZLG05]. Seu trabalho avalia o desempenho do escalonamento com cinco sistemas de reputação diferentes, em três graus de compartilhamento: local, parcial e global.

É interessante notar que os sistemas com maior compartilhamento de reputação apresentaram melhores resultados. Surge uma questão: como acreditar nos valores de reputação oriundos de um par desconhecido? Porém, este problema está fora do escopo do trabalho.

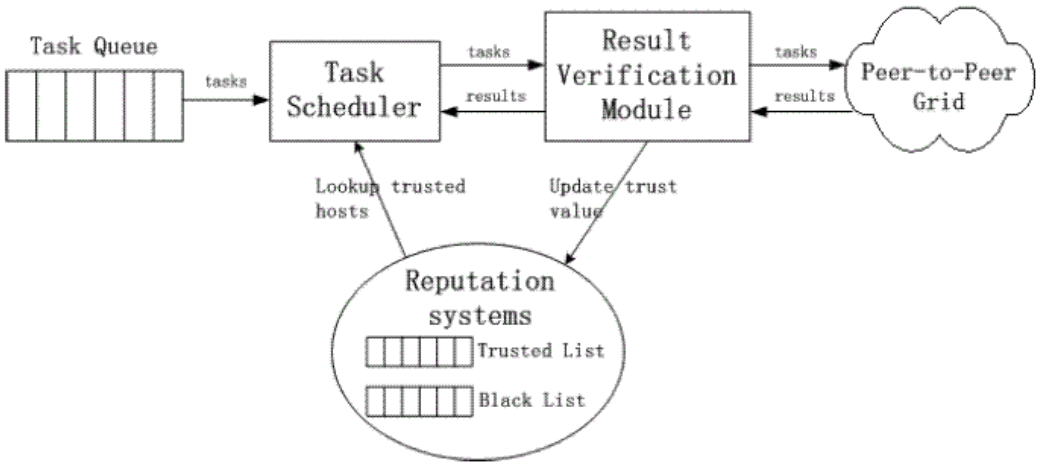
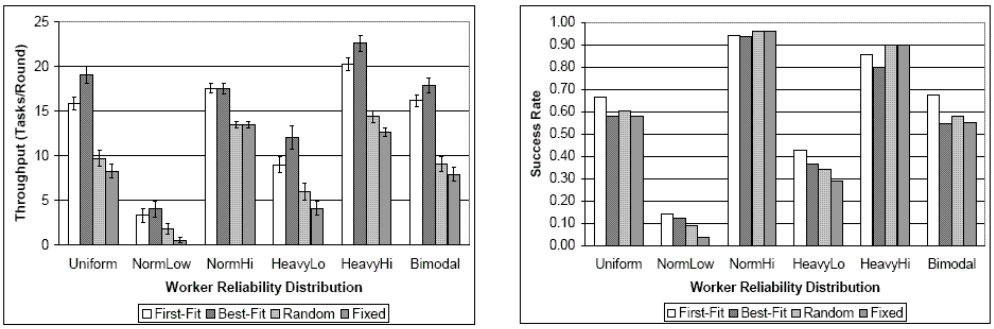


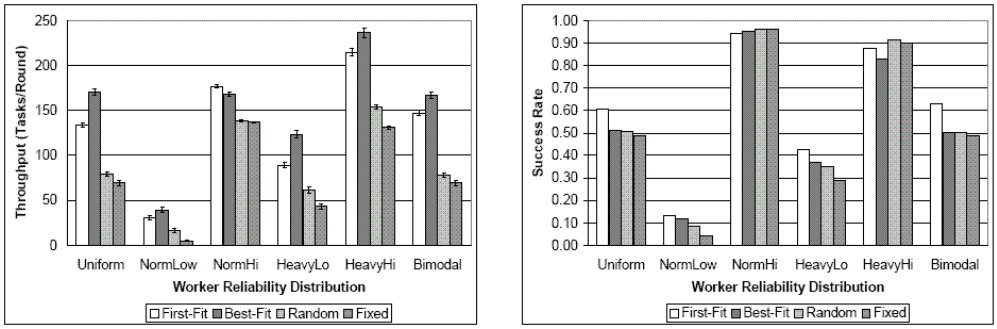
Figura 5.2: Modelo de escalonamento baseado em confiança.

Com relação ao escalonamento proposto por Sonnek et al., mesmo tendo um modelo centralizado, os resultados obtidos não apresentaram perda de desempenho com o aumento da escala. Foram realizadas simulações para seis distribuições diferentes de confiabilidade das máquinas, usando quatro algoritmos de escalonamento de tarefas (*First-Fit, Best-Fit, Random, Fixed*) [SNC06]. A Figura 5.3 [SNC06] mostra a vazão atingida (a) e a taxa de sucesso dos resultados aceitos (b) desses algoritmos com uma rede de 100 nós. Enquanto a Figura 5.4 [SNC06] apresenta as mesmas métricas para uma rede com 1000 nós. Note que os resultados obtidos para a vazão são linearmente proporcionais ao crescimento do número de nós, não afetando muito a taxa de sucesso.



(a) Throughput (b) Success Rate

Figura 5.3: Vazão média (a) e a taxa de sucesso (b) com 100 nós na rede.



(a) Throughput (b) Success Rate

Figura 5.4: Vazão média (a) e a taxa de sucesso (b) com 1000 nós na rede.

5.2.4 Tratamento de Conluios

A abordagem de conluios de sabotadores para ganharem a votação e prejudicarem os que trabalharam corretamente diverge entre os trabalhos. O modelo de Sonnek et al. desconsidera esta possibilidade. Leva em conta que os sabotadores comportam-se independentemente e que o espaço de resultados possíveis é muito grande para coincidirem, conduzindo, pois, a probabilidade de conluio a ser desprezível. Os projetos de Sarmenta e Zhao et al. consideram possível a existência de conluios entre sabotadores. Zhao et al. apresentam uma avaliação sobre o comportamento do sistema em caso de conluios.

No trabalho de Zhao et al. [ZLG05] pode-se perceber que a precisão para se obter um resultado correto aplicando-se a técnica de votação majoritária (replicação) com conluios diminui drasticamente à medida que a fração de nós maliciosos aumenta. Enquanto a acurácia da técnica de *quiz* decai gradualmente, conforme as Figuras 5.5 [ZLG05] e 5.7 [ZLG05]. Sem a possibilidade de conluios, o mecanismo de replicação apresenta as melhores taxas:

uma precisão próxima de 100%.

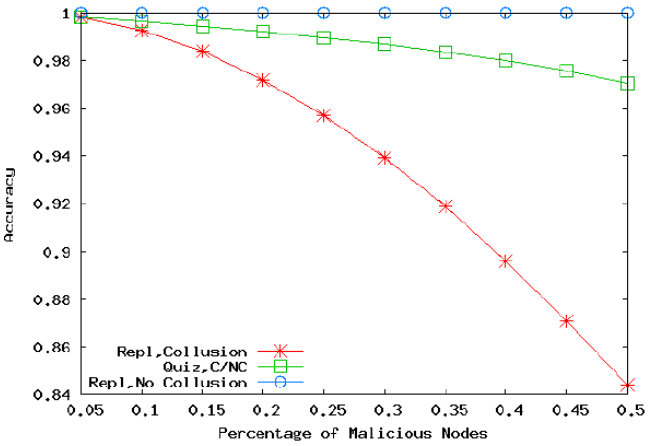


Figura 5.5: Precisão vs. percentual de faltosos; sem sistema de reputação, $s = 50\%$.

5.2.5 Sistemas de Reputação

A Figura 5.6 [ZLG05] apresenta resultados obtidos para o sobrecarga por Zhao et al. [ZLG05] sem a utilização de um sistema de reputação. Vale salientar que a técnica de *quiz* apresentou o pior sobrecarga. Isto se explica porque, quando se detecta um resultado errado, todo o pacote de tarefas é descartado. E como esperado, observa-se que o sobrecarga aumenta com o aumento da fração de faltosos.

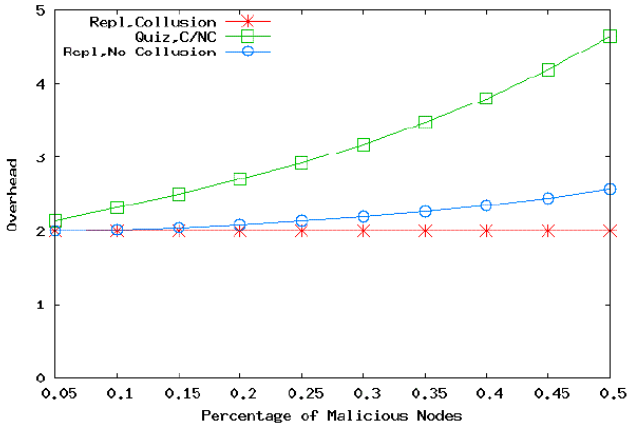


Figura 5.6: Sobrecarga vs. percentual de faltosos; sem sistema de reputação, $s = 50\%$.

As Figuras 5.7 [ZLG05] e 5.8 [ZLG05] vêm demonstrar que o uso de um sistema de reputação influencia na precisão dos resultados e no sobrecarga. Os sabotadores foram ava-

liados com uma taxa de sabotagem constante igual a 50% em três cenários: *quiz*, replicação sem conluio e replicação com conluio. O sistema de reputação foi local.

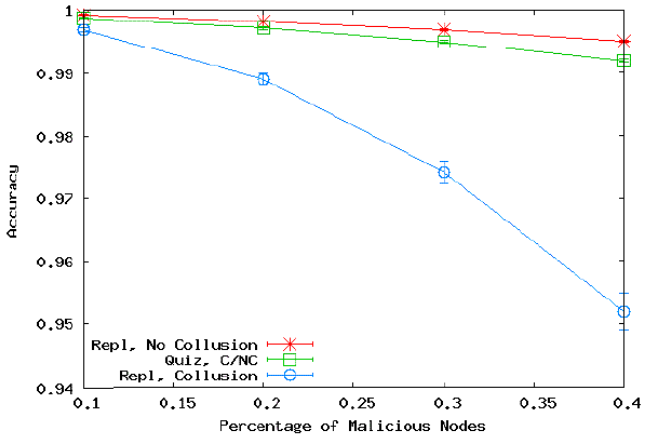


Figura 5.7: Precisão vs. percentual de faltosos; sistema de reputação local, cálculos feitos entre as 500.000 primeiras tarefas submetidas, $s = 50\%$.

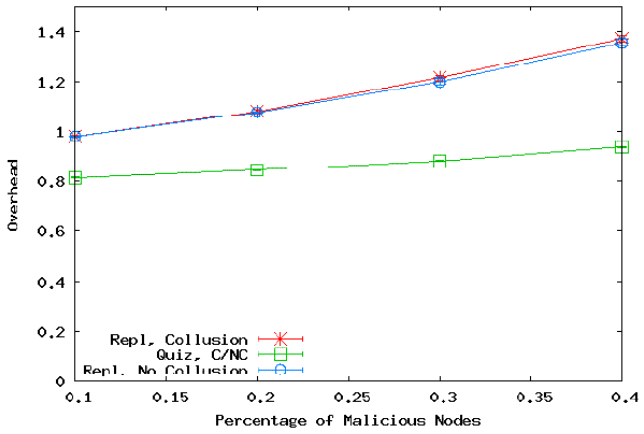


Figura 5.8: Sobrecarga vs. percentual de faltosos; sistema de reputação local, cálculos feitos entre as 500.000 primeiras tarefas submetidas, $s = 50\%$.

5.3 Considerações sobre os Trabalhos Relacionados

Sabe-se que a técnica de votação majoritária apresenta altas taxas de replicação (no mínimo, o trabalho deve ser feito duas vezes). Os esquemas de *quizes*, por sua vez, são escalonados em forma de um pacote com várias tarefas onde se percebe que se algum *quiz* estiver incorreto haverá um desperdício de computação útil para todas as tarefas do pacote também.

Apesar da redundância aumentar a probabilidade de serem aceitos resultados corretos, ela significa desperdício de recursos.

Tanto Zhao et al [ZLG05] quanto Sonnek et al. [SNC06], para aumentarem a vazão de tarefas concluídas com sucesso, propõem algoritmos de escalonamento que utilizam sistemas de reputação, diminuindo, pois, as taxas de erro e a sobrecarga causadas pela execução de tarefas em máquinas não-confiáveis.

Os trabalhos de Zhao et al. [ZLG05] e Sonnek et al. [SNC06] são bastante interessantes, pois estudam alguns problemas que este trabalho não contempla. Eles abrangem outros tipos de falhas, como conluios e faltas no tempo. No entanto, deixam a desejar em alguns pontos. O trabalho de Zhao et al. [ZLG05] não apresenta um estudo sobre as funções para cálculo do fator de replicação e das taxas de *quizes* no artigo. Além disso, deixa a escolha do sistema de reputação em aberto, não cobrindo o modo como estimar as reputações. Sonnek et al. [SNC06], por sua vez, tentam limitar a redundância estabelecendo tamanhos mínimos ou máximos para os grupos de redundância. O trabalho avalia os grupos usando valores fixos para seus tamanhos e não discute como esses valores deveriam ser estimados de modo a obter melhores custos ou desempenho ou corretude.

Este trabalho difere dos demais por apresentar a metodologia necessária para compor um mecanismo de escalonamento tolerante a sabotagem e implementa a técnica de tolerância a faltas baseada em credibilidade proposta por Sarmenta [Sar01][Sar02]. Essa técnica combina outras, proporcionando redução de custos com relação unicamente à técnica de votação. Ela apresenta também técnicas bem definidas para estabelecimento de reputações. Este trabalho também contribuiu com o de Sarmenta, propondo formas de como contornar alguns problemas de implementação prática, no que diz respeito à forma de estimar parâmetros do sistema, i.e., f e q.

Capítulo 6

Considerações Finais e Trabalhos Futuros

6.1 Considerações Finais

Os trabalhos relacionados à área, em geral, ou não tratam de escalonamento ou empregam mecanismos baseados em votação majoritária que apresentam altos índices de replicação. O diferencial deste trabalho é a proposta de um mecanismo de escalonamento tolerante a sabotagem para grades P2P que está aliado a um sistema de reputação com técnicas que requerem baixa redundância e possibilitam atingir baixas taxas de erro na aceitação de tarefas.

Os resultados obtidos neste trabalho comprovam a idéia de que a técnica de tolerância a faltas baseada em credibilidade proposta por Sarmenta pode ser empregada para obter tolerância a sabotagem em ambientes de grades computacionais P2P, entretanto foi necessário contornar problemas de implementação prática, propondo e avaliando formas de estimar alguns parâmetros do sistema, i.e., f e q.

No entanto, implementar a técnica de tolerância a faltas baseada em credibilidade não foi uma tarefa simples, pois esse estudo teórico assume o uso de propriedades que não são conhecidas na prática. Após várias análises, essas propriedades foram estimadas a partir de outras suposições controláveis feitas para o sistema. A avalição de desempenho foi realizada implementando um mecanismo de escalonamento não-adaptativo, que conhecia de antemão as propriedades desconhecidas da grade, que foi comparado a dois mecanismos de escalonamento que as estimavam de modo adaptativo.

Diante dos resultados obtidos para as heurísticas de escalonamento estudadas, conclui-se que elas apresentam características peculiares e sugerem uma abordagem de uso baseada no conhecimento que o escalonador tem sobre o sistema e o tipo de ambiente em questão.

A heurística de escalonamento não adaptativa é indicada primeiramente quando não se tem conhecimento de máquinas confiáveis no sistema. Já as heurísticas de escalonamento adaptativas pressupõem que o escalonador conhece alguma fração de máquinas confiáveis na grade.

A heurística de escalonamento adaptativa com número de máquinas restrito no escalonamento é a que possui a menor redundância. Elas se mantém próxima de 1, ou seja, não ocorre praticamente replicação. Entretanto, é ineficiente com relação à corretude quando o sabotador apresenta baixas taxas de sabotagem e o erro aceitável é igual ou superior a $0,1\%$, pois as máquinas não são testadas suficientemente antes de serem aceitas como confiáveis. Após aceitas, não ocorre votação sobre os resultados fornecidos por elas. Uma maneira de limitar as taxas de erro a longo prazo é estipulando um valor mínimo para q, conforme discutido. O sistema tem uma fase inicial de aprendizagem e, em seguida, consegue garantir que o erro obtido esteja dentro dos limites do erro aceitável com baixo custo.

A heurística de escalonamento adaptativa gulosa é indicada quando o ambiente é muito hostil. Quanto mais sabotadores são identificados, melhor é seu desempenho. Contudo, quando o número de sabotadores identificados é baixo ou o número de máquinas confiáveis conhecidas também é baixo, pode apresentar muita variabilidade no tempo de execução dos *jobs* e baixo desempenho a curto prazo.

Uma vantagem das heurísticas adaptativas é que à medida que outros trabalhos são executados, o desempenho das heurísticas melhora. A redundância, o *slowdown* e o erro tendem a cair, pois mais máquinas confiáveis são agregadas ao escalonamento ou mesmo os sabotadores mais resistentes são identificados em algum momento.

A grande desvantagem das heurísticas adaptativas é o método para calcular q_{adap}. À medida que um sabotador executa tarefas sem ser identificado, a probabilidade de ele ser testado vai diminuindo cada vez mais e ele pode permanecer no sistema retornando resultados incorretos. Nesse caso, os usuários maliciosos poderiam responder corretamente às primeiras tarefas executadas e depois de atingirem o limiar de credibilidade e diminuirem suas probabilidades de receberem *spot-checks*, eles poderiam iniciar o retorno apenas de resultados

incorretos.

6.2 Trabalhos Futuros

Este trabalho propõe técnicas para tolerância a sabotagem, supondo que os sabotadores atuam de maneira independente e os resultados incorretos são gerados de maneira aleatória, com uma probabilidade muito baixa de coincidirem. Os sabotadores não trocam informações entre si para decidirem em que momento e que valor propor. Um possível trabalho futuro é investigar outros tipos de falhas, como a existência de conluios entre os sabotadores na grade.

Outro trabalho futuro interessante é aprimorar o algoritmo de escalonamento estudado, a partir do estudo de novas heurísticas com novas abordagens para a estimativa de f e q ou até mesmo das taxas de sabotagem dos sabotadores. Por exemplo, um bom começo poderia ser estudar o valor de q constante para as heurísticas adaptativas com o valor de f adaptativo, mas ao mesmo tempo não muito alto para evitar que o custo de tarefas que não representam computação útil ao final da execução do trabalho seja alto.

Outro estudo poderia investigar como seriam as taxas de erro para a heurística adaptativa com número restrito de máquinas no escalonamento se o valor de q fosse incrementado periodicamente ao invés de apenas decrescer para tentar eliminar os sabotadores mais resistentes. Outra possibilidade é investigar valores ótimos para o valor mínimo de q ou ainda outros valores para m_d que não reduzissem o valor de f ao ponto das máquinas receberem credibilidades altas demais desde o início da computação sem que de fato sua probabilidade de estar correta represente tais valores.

Com relação à melhoria do estabelecimento de credibilidades, é possível distinguir a credibilidade do resultado da credibilidade do solucionador. Por exemplo, os sítios também poderiam ser um fator a influenciar na credibilidade do resultado. O cálculo da credibilidade dos resultados poderia ter um peso associado ao sítio. Quanto maior fosse o número de máquinas com boa reputação que o sítio possuísse, maiores seriam as credibilidades das máquinas daquele sítio. Considerando-se que quanto mais máquinas de boa reputação um sítio possua, maior a probabilidade que as demais máquinas daquele sítio sejam confiáveis.

Poderia ser estudado também um mecanismo de escalonamento mais inteligente. Por

exemplo, ao se escalonar uma tarefa para uma máquina confiável, dar preferência a uma tarefa que ainda não foi escalonada ou replicar uma tarefa que está com credibilidade baixa a máquinas que possuem credibilidade maior. Para realizar esse escalonamento, poderia ser implementado um módulo que ao escalonar uma tarefa, estimasse qual a credibilidade mínima que a próxima máquina a executar uma réplica para esta tarefa deve ter para que ela seja concluída com o menor custo possível, de acordo com as credibilidades existentes para o conjunto de máquinas disponível.

Por fim, um importante trabalho futuro é o de implementar e avaliar o serviço de escalonamento tolerante a sabotagem em um ambiente de grade computacional P2P em produção, como o OurGrid [CBA+06].

Bibliografia

[And03] David P. Anderson. Public computing: Reconnecting people to science. In *Conference on Shared Knowledge and the Web*, Madrid, Spain, November 2003.

[BM02] Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience (CCPE)*, 14(13-15):1175–1220, November - December 2002.

[CBA+06] Walfredo C. Cirne, Francisco Brasileiro, Nazareno Andrade, Robson Santos, Alisson Andrade, Reynaldo Novaes, and Miranda Mowbray. Labs of the world, unite!!! *Journal of Grid Computing*, June 2006.

[Cel07] Cell computing. http://www.cellcomputing.net/, August 2007.

[cli07] climateprediction.net. http://climateprediction.net/, August 2007.

[Col07] Projeto Colt. http://dsd.lbl.gov/ hoschek/colt/, Agosto 2007.

[Dis07] Distributed.net. http://distributed.net, August 2007.

[Ein07] Einstein@home. http://einstein.phys.uwm.edu/, August 2007.

[FKT01] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 2001.

[Fos02] Ian Foster. The grid: A new infrastructure for 21st century science. *Physics Today*, 55(2):42–47, 2002.

[Fos05] Ian Foster. Service-oriented science. *Science*, 308:814–17, 2005.

[Fos06] Ian Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pages 2–13. Springer-Verlag LNCS 3779, 2006.

[Hay98] Brian Hayes. Collective wisdom. *American Scientist*, March-April 1998.

[LHC07] Lhc@home. http://lhcathome.cern.ch/lhcathome/, August 2007.

[Mol00] David Molnar. The seti@home problem. E-Commerce, 2000.

[Ora01] Andy Oram, editor. *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O'Reilly, Sebastopol, CA, 2001.

[Pen02] Rob Pennington. Terascale clusters and the teragrid. In *Proceedings of the HPC Asia*, pages 407–413, December 2002.

[Pre07] Predictor@home. http://predictor.scripps.edu/, August 2007.

[Ros07] Rosetta@home. http://boinc.bakerlab.org/rosetta/, August 2007.

[Sar01] Luis F. G. Sarmenta. *Volunteer Computing*. PhD thesis, Massachusetts Institute of Technology (MIT), Dept. of Electrical Engineering and Computer Science, March 2001.

[Sar02] Luis F. G. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems (Expanded journal version of CCGrid '01, Best Finalist Paper), Elsevier*, 2002.

[SET07] Seti@home. http://setiathome.berkeley.edu/, August 2007.

[SN02] J. M. Schopf and B. Nitzberg. Grids: Top ten questions. *Scientific Programming, special issue on Grid Computing*, 10(2):103–111, August 2002.

[SNC06] Jason Sonnek, Mukesh Nathan, and Abhishek Chandra. Reputation-based scheduling on unreliable distributed infrastructures. In *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS'06), Lisboa, Portugal, July 2006*, 2006.

[UJJL05] P. Uppuluri, N. Jabisetti, U. Joshi, and Y. Lee. P2p grid: service oriented framework for distributed resource management. In *IEEE International Conference on Services Computing*, volume 1, pages 347–350, July 2005.

[WCG07] World community grid. http://www.worldcommunitygrid.org/, August 2007.

[Web02] Taisy Silva Weber. Um roteiro para exploração dos conceitos de tolerância a falhas. Curso de Especialização em Redes e Sistemas Distribuídos, 2002.

[ZLG05] Shanyu Zhao, Virginia Lo, and Chris GauthierDickey. Result verification and trust-based scheduling in peer-to-peer grids. In *Fith IEEE International Conference on Peer-to-Peer Computing (IEEE P2P)*, pages 31–38, 2005.