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

Metrologia da Captura e Reprodução de Rastros de Sistemas de Arquivos

Thiago Emmanuel Pereira da Cunha Silva

Tese 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 Doutor em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Sistemas e Computação

Francisco Vilar Brasileiro
(Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S586m Silva, Thiago Emmanuel Pereira da Cunha.
Metrologia da captura e reprodução de rastros de sistemas de arquivos / Thiago Emmanuel Pereira da Cunha Silva. – Campina Grande, 2016.
116f. : il.

Tese (Doutorado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2016.
"Orientação: Prof. Francisco Vilar Brasileiro".
Referências.

1. Ciência da Computação - Metrologia. 2. Sistemas de Arquivos. 3. Avaliação de Desempenho. I. Brasileiro, Francisco Vilar. II. Universidade Federal de Campina Grande, Campina Grande (PB). III. Título.

CDU 004:006.91(043)

Resumo

Métodos para avaliação de desempenho têm, por muito anos, apoiado a adoção, desenvolvimento e operação de sistemas de arquivos. Em particular, a reprodução de rastros de utilização é um método bastante popular. Apesar de sua popularidade, alguns trabalhos produzidos recentemente contestam — embora sem comprovação empírica — a qualidade dos resultados obtidos com esse método.

Em nossa opinião, este ceticismo se deve, em larga medida, à falta de métodos suficientemente embasados para detectar e quantificar os erros das medições baseadas em reproduções de rastros. Uma vez que não sabemos como diagnosticar o problema, é provável que novos trabalhos adotem métodos de reprodução inadequados ou criem novos métodos sem preocupações metodológicas mais amplas. Em outras palavras, estamos fadados a repetir os erros anteriores ou introduzir novos erros.

Nossa tese é que podemos usar metrologia — a ciência de obter boas estimativas para medições conduzidas por meio de instrumentos de medição imperfeitos — para melhorar a qualidade da avaliação de desempenho de sistemas de arquivos baseada em rastros.

Neste documento, desenvolvemos esta tese por meio de duas provas de conceito. A primeira considera métodos de captura de rastros, enquanto a segunda considera métodos de reprodução de rastros. Em ambas, consideramos ferramentas populares descritas na literatura. Na primeira prova de conceito, ao aplicar o protocolo de metrologia, descobrimos fontes de erro que tornam as medições de captura de rastros bastante tendenciosas, embora precisas. Nós também mostramos como compensar as fontes de erro por meio de um processo de calibração — prática ausente na literatura. Ainda, descobrimos que os métodos de captura são sensíveis ao efeitos de cargas de fundo, as quais, quando não identificadas, podem comprometer o processo de calibração. Na segunda, o protocolo de metrologia também nos permitiu observar fontes de erros que enfraquecem a qualidade das medições de reprodução. Embora o procedimento de calibração não seja aplicável na segunda prova de conceito, nossos resultados apontaram limitações nas ferramentas de reprodução que quando reconsideradas, permitiram a melhoria na qualidade dos métodos considerados.

Agradecimentos

Agradeço ao governo brasileiro por financiar meus estudos. Agradeço bastante ao Professor Fubica, pela paciência e dedicação na minha orientação desde 2005 (quando entrei no LSD). Agradeço aos meus camaradas de laboratório, aí incluídos os colegas do BeeFS (Patrick, Gonzaguinha, Johnny e Carlinha), Nigini, Abmar, Fabio e Lesandro Ponciano (por todas as conversas não relacionadas com nossas teses) e Livia Sampaio, minha co-orientadora informal. Por fim, vivas à minha família, minha esposa Heloísa (e ao meu filho, que no momento em que compilo esse código fonte, ainda não nasceu), por suportarem minha falta de humor com a infinita esperança de que ela fosse passageira (nada mudou do mestrado para hoje).

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nesta tese, consideramos a avaliação de desempenho baseada na reprodução de rastros de utilização de sistemas de arquivos. Em comparação com outros métodos, tais como modelagens analíticas [16, 18, 20], simulações [64], e execução de *benchmarks* [2, 3, 6, 14, 15, 23, 35, 42, 51], acredita-se que a reprodução de rastros proporciona a representação mais próxima de experimentos com cargas de trabalho reais.

Todavia, recentemente a convicção sobre a qualidade deste método tem sido contestada. Usando as palavras de Traeger et al. [65], "How to do this [reproduzir a carga de trabalho real] accurately is still an open question, and the best we can do right now is take the results with a degree of skepticism.".

Qual a razão para esse ceticismo? Nossa revisão da literatura indica pelo menos uma possibilidade: há pouca análise comparativa sobre os métodos conhecidos; como escolher entre um ou outro método? Há diferenças entre eles?

Para preencher esta lacuna, conduzimos uma prova de conceito preliminar comparando os métodos de captura e reprodução mais populares [55]. O resultado dessa prova de conceito confirma e embasa as suspeitas da comunidade: de fato, os métodos de captura e reprodução propostos na literatura podem levar a conclusões diferentes.

Com esse estudo identificamos uma segunda fonte para o ceticismo: não há metodologia amplamente aceita para analisar os métodos de captura e reprodução. Sem isso, embora saibamos que os métodos não sejam perfeitos, não sabemos quão errados eles são. Ou ainda, quais são as fontes destes erros e como mitigá-los.

Tivesse maior intuição, poderia ter suspeitado, sem o esforço inicial da prova de conceito,

que outras ciências — por exemplo, física, química e biologia — consideram problemas similares. Em todas essas ciências, objetos de estudo são observados por meio de instrumentos de medição. Os instrumentos de medição são limitados, e portanto, as observações contém erros. O estudo desses erros de medição constitui o objetivo da metrologia.

Nossa tese é que metrologia pode também ser aplicada para entender os problemas de medição em avaliação de desempenho de sistemas de arquivos baseada em rastros. O propósito deste documento é exercitar esta tese e suas implicações para medições baseadas em rastros de utilização de sistemas de arquivos. Para isso, aplicamos o arcabouço de metrologia na análise de duas provas de conceito, cobrindo as fases de captura e reprodução de rastros.

O restante desse documento é dividido no que segue. Nos Capítulos 2 e 3 procuramos embasar a intuição de que metrologia pode ser aplicada nos cenários de captura e reprodução de rastros. No Capítulo 2 descrevemos os métodos de avaliação de desempenho de sistemas de arquivos, com ênfase nos métodos de reprodução de rastros, e na crítica que tem sido feita a esses métodos recentemente. Em seguida, no Capítulo 3, discorremos sobre os aspectos básicos de metrologia. Apresentamos o vocabulário básico que será usado ao longo do documento e uma visão geral do arcabouço teórico de metrologia. Ainda nesse capítulo, relacionamos os problemas dos métodos de captura e reprodução de rastros com os problemas tratados por metrologia em outras ciências. Com isso, dadas as convergências, apontamos as oportunidades de aplicação de metrologia nos cenários de captura e reprodução de rastros. No Capítulo 4 detalhamos o arcabouço teórico de metrologia que será aplicado nas prova de conceito. Nos Capítulos 5 e 6 descrevemos e discutimos as provas de conceito de captura e reprodução, respectivamente. Nos Capítulo 5 avaliamos experimentalmente dois métodos populares de captura de rastros. Em um dos métodos, a ferramenta de captura utilizada executa no espaço do usuário, enquanto no outro método a ferramenta de captura executa como um módulo do núcleo do sistema operacional. O capítulo foca na validação dos métodos de captura, no exame de suas fontes de erros e por fim, da redução de parte desses erros por um processo de calibração. Em sequência a prova de conceito sobre captura, no Capítulo 6 descrevemos a prova de conceito de reprodução de rastros. Nesse capítulo, novamente, avaliamos dois métodos populares. Os métodos que consideramos se diferenciam pela arquitetura da ferramenta de reprodução: baseada em compilação ou baseada em eventos. Nessa prova

de conceito também consideramos a validação dos métodos de reprodução bem como o exame das fontes de erro. Esse exame é posteriormente usado como base para a modificação do reprodutor baseado em eventos que levou à redução da incerteza de suas medições. Após apresentar as provas de conceito, no Capítulo 7 avaliamos em retrospectiva a aplicabilidade de metrologia nos cenários de captura e reprodução de rastros. Nele também discutimos outros problemas da área, nos quais poderíamos aplicar alguma consideração metrológica. Nessa discussão final, para tornar a leitura mais fluida, evitamos qualquer construção matemática. Para o leitor interessado em uma discussão mais aprofundada, apresentamos no Apêndice A um complemento para o arcabouço teórico de metrologia.

Por fim, o leitor pode também consultar as publicações [53] e [54], que são bases para as provas de conceito descritas nos Capítulos 5 e 6, respectivamente.

Capítulo 2

Avaliação de desempenho de sistemas de arquivos

Neste capítulo discutiremos alguns dos métodos conhecidos de avaliação de desempenho de sistemas de arquivos. Discutiremos as limitações, bem como a adequação de cada um destes métodos. Nessa discussão, tomamos como base a compilação produzida por Traeger et al. [65]. Na Seção 2.2, consideramos o método de avaliação baseado na execução de *microbenchmarks*. Por sua vez, na Seção 2.3 levamos em conta a avaliação baseada em *macrobenchmarks*. Por fim, nas Seções 2.4.1 e 2.4.2 descrevemos os métodos conhecidos para captura e reprodução de rastros de utilização. Antes de discutir os métodos de avaliação, na Seção 2.1 apresentamos de maneira resumida alguns conceitos de sistemas de arquivos.

2.1 Sistemas de arquivos

Sistemas operacionais baseados em `UNIX` fornecem recursos para gerenciamento de armazenamento permanente de dados. Arquivos são a abstração básica que suportam esse gerenciamento. Do ponto de vista do sistema operacional, arquivos são contêineres sequenciais de *bytes* nos quais dados são armazenados através de operações de escrita. O acesso aos dados pode ser feito tanto de maneira sequencial quando aleatória, com base no deslocamento do arquivo em cada operação; no modo sequencial, o deslocamento no início de uma operação é igual ao deslocamento ao fim da operação anterior, já no modo aleatório, não há relação entre os deslocamentos.

Os arquivos são organizados em uma estrutura hierárquica de árvore em um sistema de arquivos. Além de arquivos, essa estrutura contém também diretórios. Os arquivos sempre são componentes terminais. Tanto arquivos quanto diretórios são identificados por um nome. Arquivos abaixo de um mesmo diretório não podem ter um mesmo nome. Para identificar unicamente um arquivo é preciso determinar seu caminho completo, que é composto dos nomes de todos os componentes, dá raiz do sistema de arquivos até o arquivo em questão.

O sistema de arquivos também mantém um conjunto de atributos para cada arquivo, por exemplo: tipo do arquivo (diretório, regular, FIFO entre outros), tamanho do arquivo, identificadores de usuário e grupo do dono do arquivo, permissões de acesso e marcações de tempo para criação, acesso e modificação. Os atributos por vezes são chamados também de metadados e podem ser obtidos por chamadas ao sistema como *stat*, *lstat* e *fstat*.

Numa perspectiva arquitetural, uma implementação genérica de sistemas de arquivos em um sistema operacional `UNIX` segue um modelo conforme a Figura 2.1 [32]:

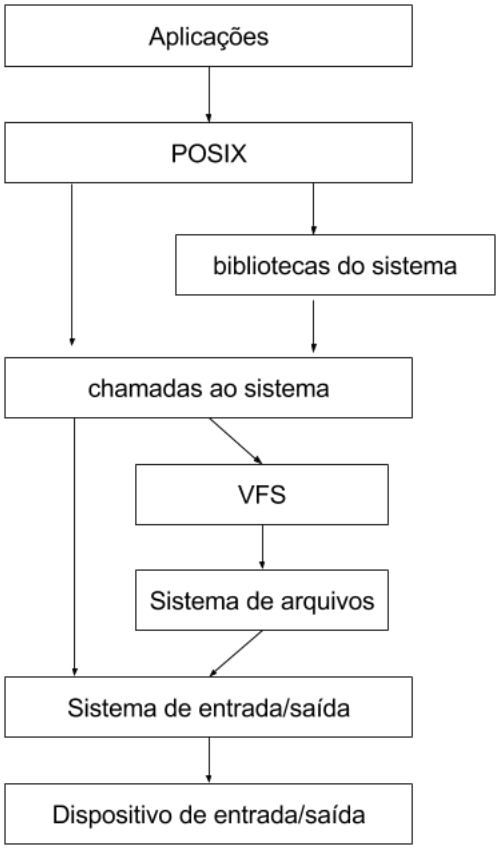


Figura 2.1: Modelo arquitetural típico em um sistema de arquivos `UNIX`.

Para acessar as funcionalidades dos sistemas de arquivos, as aplicações interagem com

a interface POSIX disponível no sistema operacional. Isso pode ser feito tanto através de chamadas ao sistema quando pelo uso de bibliotecas como *libc* (que por sua vez usam as chamadas ao sistema). Uma implicação direta do uso dessa interface é que as aplicações não precisam ser modificadas para usar sistemas de arquivos diferentes. Outro ponto importante desse modelo arquitetural é que implementações particulares de sistemas de arquivos não estão diretamente acopladas com as chamadas ao sistema. Ao invés disso, essa chamadas usam o **VFS** (*virtual file system interface*), como interface comum mediadora para as diferentes implementações de sistemas de arquivos.

2.2 *Microbenchmarks*

Microbenchmarks são aplicações escritas para testar um aspecto específico, de maneira isolada, do sistema de arquivos. Um exemplo é o *benchmark* Bonnie [23], que avalia o desempenho de operações de escrita sequencial, leitura sequencial e aleatória. Outro exemplo é o *benchmark* do Sprite LFS [57], que além de considerar leituras e escritas, sequenciais e aleatórias, também avalia o desempenho para criar e remover arquivos.

Uma alternativa mais flexível é o uso de *microbenchmarks* geradores, tais como o IOZone [51], Iometer [3] e Filebench [2]. Basicamente, essas ferramentas permitem o exercício de um número maior de funções do sistema de arquivos. Por exemplo, o *benchmark* Iozone permite avaliar o desempenho das seguintes operações: i) *write* - criação e escrita em arquivos; ii) *re-write* - escrita em arquivos já criados; iii) *read* -leitura em arquivo já criados; iv) *re-read* - leitura em um arquivo que foi lido recentemente; v) *random-read* - leitura em posições aleatórias de um arquivo; vi) *random-write* - escrita em posições aleatórias de um arquivo; vii) *random-mix* - escrita e leitura em posições aleatórias de um arquivo; viii) *backwards read* - leitura do fim para o início de um arquivo; ix) *record re-write* - escrita para um região específica de um arquivo. Além disso, é possível configurar o tamanho do arquivo a ser manipulado, o tamanho do bloco usado para escrita e leitura, o atraso usado entre cada operação, o número de processos usados para executar a carga de trabalho, entre outros parâmetros.

A despeito da disponibilidade de boas ferramentas de livre acesso, via de regra cria-se um novo *benchmark* específico para cada avaliação. Segundo Traeger et al. [65], criou-se

uma nova ferramenta de propósito específico em 62 artigos dentre os 106 considerados pelos autores. Essa abordagem tem uma desvantagem clara. Uma vez que, raramente, o código fonte do *benchmark* é publicizado, é difícil reproduzir os resultados.

2.3 *Macrobenchmarks*

Enquanto que os *microbenchmarks* avaliam operações específicas, de maneira isolada, do sistema de arquivos, os *macrobenchmarks* exploram múltiplas operações em uma carga de trabalho aproximada da carga de trabalho real.

Macrobenchmarks populares incluem: Postmark [42], criado para representar uma carga de trabalho de acesso a dados e metadados de arquivos pequenos e com vida curta, carga típica de servidores de *e-mail*; a família de *benchmarks* criado pela *Transaction Processing Performance Council* (TPC) [15], para avaliação de desempenho de sistemas de banco de dados; e o SPEC SFS para a avaliação de servidores de arquivos NFS [14].

Usar ferramentas públicas como essas para a avaliação de desempenho tem uma vantagem clara: é mais fácil comparar o desempenho de um sistema com os seus concorrentes. Entretanto, é necessário avaliar a adequação da ferramenta escolhida para como o cenário real alvo do sistema sob avaliação. Ou seja, é importante julgar se a carga de trabalho gerada é uma boa aproximação da carga de trabalho real. De modo geral, isso não costuma ser levado em consideração; a popularidade é a justificativa dada para a escolha das ferramentas em muitos dos artigos compilados por Traeger et al. [65].

Um caso emblemático desse problema é o uso do Andrew *benchmark*, que tornou-se popular após ter sido usado na avaliação de desempenho do influente Andrew File System [35]. Esse *benchmark* considera a carga de trabalho gerada pela compilação de um dado programa. Ele tem como entrada a estrutura de diretórios contendo o código fonte do programa e sua execução prossegue em cinco etapas:

1. Cópia da estrutura de diretórios para o sistema de arquivos sob avaliação.

2. Cópia dos arquivos fontes do programa a ser compilado para a estrutura de diretórios criada na fase anterior.

3. Execução da operação *stat* para cada um dos arquivos copiados na fase anterior.

4. Leitura do conteúdo dos arquivos (em sua totalidade) copiados na segunda fase.

5. Compilação do código fonte.

Embora a carga de trabalho gerada por essa ferramenta possa ter sido relevante quando foi criada, em 1988, em máquinas modernas a memória principal costuma ser mais do que suficiente para armazenar todo o conteúdo dos arquivos que serão compilados, portanto, após a execução da segunda fase, grande parte das requisições não serão respondidas pelo sistema de arquivos.

2.4 Avaliação baseada em rastros

A metodologia de reprodução de rastros pode ser dividida em 4 etapas. A primeira etapa envolve a captura do rastro de utilização. Para isso o sistema sob monitoramento precisa ser instrumentado. Em seguida, na segunda etapa, usualmente, um fragmento do rastro capturado é selecionado para a experimentação. Essa seleção acontece tanto para experimentar com fragmentos de características diferentes como para evitar os problemas associados à manipulação do rastro completo, que costuma ser bastante grande. Na terceira etapa, o fragmento selecionado é efetivamente reproduzido. O fator chave da etapa de reprodução é a geração de uma carga de trabalho equivalente à carga de trabalho do fragmento tomado como entrada. Além disso, ela também envolve a configuração do sistema que será avaliado, aí incluído a criação da estrutura de diretórios e seus arquivos. Por fim, na quarta etapa, o desempenho do sistema de arquivos alvo da avaliação é medido, em termos de alguma métrica de desempenho coerente com o tipo das aplicações associadas ao sistema.

2.4.1 Captura

O rastro de utilização é uma representação da carga de trabalho submetida ao sistema de arquivos pelas aplicações em execução no ambiente de produção. A maneira mais acurada de obter essa representação da atividade das aplicações seria instrumentá-las. Infelizmente, isso nem sempre é possível. Nesse trabalho nós consideramos que, no ambiente de produção, as aplicações em execução são arbitrárias, desse modo, não é possível ter acesso ao seu código para que a instrumentação seja feita.

Como parece ser a regra em se tratando da metodologia da avaliação de desempenho de sistema de arquivos, não há consenso em como essa instrumentação deve ser feita. Um método popular consiste na captura do rastro ao nível das chamadas ao sistema relacionadas ao sistema de arquivos (ex. *open*, *read* e *close*) [49, 52, 55, 66]. Há duas alternativas nesse método que se diferenciam pela maneira como o sistema operacional é instrumentado. A primeira alternativa emprega ferramentas que executam no espaço do usuário [66], que por sua vez, se baseiam na chamada ao sistema *ptrace*. A segunda alternativa consiste em adicionar emendas ao sistema operacional com o código de instrumentação [49].

Um segundo método aplicável em sistemas de arquivos distribuídos, consiste em capturar os pacotes enviados entres clientes e servidores através da rede [21, 27, 28, 50]. Para tanto, esse método emprega ferramentas tais como *tcpdump* [4] e *wireshark* [5], que permitem filtrar apenas pacotes de um determinado protocolo, por exemplo, NFS [59] e CIFS [34].

O método baseado na captura de pacotes tem uma vantagem em relação à instrumentação das chamadas ao sistema: é necessário interferir em menos componentes. Enquanto a captura de pacotes é feita tipicamente no comutador ou na máquina em que está instalado o servidor de arquivos, a instrumentação de chamadas ao sistema precisa ser feita em todas as máquinas clientes. Embora tenha a vantagem da praticidade, o método baseado na captura de pacotes representa pior a carga de trabalho real do que o método baseado na instrumentação dos clientes, uma vez que as requisições atendidas pelo *cache* dos clientes do sistema de arquivos não geram requisições remotas (e portanto não podem ser capturadas).

2.4.2 Reprodução

Assim como observamos sobre a captura, há bastante diversidade nos métodos de reprodução de rastros. As principais diferenças entre esses métodos sãos os modelos de reprodução — abertos ou fechados [58] — e as arquiteturas adotadas pelas ferramentas de reprodução — baseadas em compilação ou em eventos. Com respeito aos modelos de reprodução, no fechado, a submissão de novas requisições pode depender do atendimentos das anteriores, enquanto no modelo aberto, não há essa dependência. As ferramentas baseadas em compilação [41, 66] geram o código-fonte de um programa que emula a atividade registrada nos rastros: a carga de trabalho reproduzida é gerada pela execução deste programa. Os reprodutores baseados em eventos [50, 55] são programas genéricos que reproduzem ras-

tros arbitrários tomados como entrada. Na Seção 2.4.2 discutimos os modelos reprodução encontrados na literatura. Em seguida, na Seção 2.4.2, descrevemos os projetos mais representativos de reprodutores baseados em compilação e em eventos.

Modelos de reprodução

Há dois problemas relacionados com o modelo de reprodução: i) a relação de dependência que qualquer par de requisições pode ter, e portanto a ordem na qual as requisições devem ser reproduzidas; e ii) o instante de tempo no qual uma requisição deve ser reproduzida.

Duas abordagens consideram esses problemas, uma baseada em experimentação [48] e outra baseada em inferência [50, 55, 66]. Na abordagem baseada em experimentação, assume-se que a carga de trabalho que será capturada pode ser controlada. Isso sendo possível, o atendimento de algumas requisições é artificialmente atrasado durante a captura. Em seguida, observa-se quais requisições subsequentes foram postergadas como resultado do atraso; assume-se que há dependência entre estas e aquela. Essa abordagem tem uma limitação: é necessário realizar múltiplas coletas, de tal forma que seja possível experimentar várias opções de atraso das requisições; ainda, é preciso que a carga de trabalho seja a mesma em todas as coletas, caso contrário, não é possível distinguir entre os efeitos de um atraso controlado e uma possível modificação da carga de trabalho.

Por sua vez, na abordagem baseada em inferência, as relações de dependência e temporização são deduzidas com base em um único rastro. Os reprodutores `ARTC` [66] e `TBBT` [50] definem duas classes de políticas equivalentes para a dedução da relação de ordem. A primeira classe — composta pela política *ROOT* do reprodutor `ARTC` e pela política *FS dependency* do reprodutor `TBBT` — se baseia na semântica de modificação do sistema de arquivos; por exemplo, uma requisição de leitura ou escrita para um arquivo não pode ser iniciada antes que a requisição que cria o arquivo tenha terminado. A segunda classe — composta pelas políticas *temporal* do reprodutor `ARTC` e pela política *conservative* do reprodutor `TBBT` — considera a ordem tal como encontrada no rastro, ou seja, uma requisição só pode ser reproduzida após as requisições anteriores (com base nas marcações de tempo coletadas no rastro) tenham sido reproduzidas. Em relação à inferência de temporização, os reprodutores `ARTC` e `TBBT` também adotam políticas semelhantes. Ambos os reprodutores implementam a política *fullspeed*. Ao adotar essa política, o reprodutor ignora qualquer possível

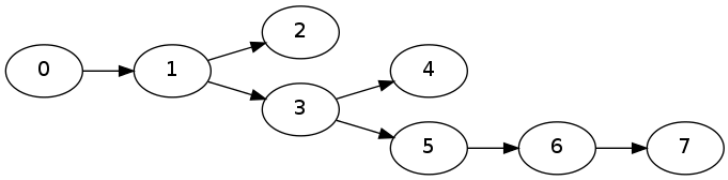
relação entre o atendimento e chegada de requisições e as processa na maior velocidade possível. Adicionalmente, o reprodutor `TBBT` implementa a política de temporização *timestamp*; ao adotar essa política, o reprodutor executa as requisições o mais próximo possível das marcações de tempo encontradas no rastro.

A escolha das políticas de ordenação e temporização de fato determina as características da carga de trabalho que será reproduzida. A Figura 2.2 ilustra os efeitos dessa escolha sobre a reprodução do rastro definido na Tabela 2.1. Nessa tabela, registra-se o início e o fim da execução de 8 requisições, identificadas pelo valor da coluna *i*. O tipo, os argumentos e o valor de retorno das requisições estão indicados nas colunas *type*, *args* e *RV*, respectivamente.

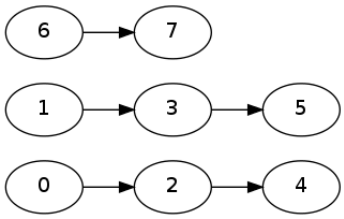
Tabela 2.1: Exemplo de rastro de utilização. Registramos o início e o fim da execução de 8 requisições, identificadas pelo valor da coluna *i*. O tipo, os argumentos e o valor de retorno das requisições estão indicados nas colunas *type*, *args* e *RV*, respectivamente.

i	*início*	*fim*	*type*	*args*	*RV*
0	0	1	open	{*pathname=/parentA/fileA, flags*=32961, *mode*=384}	5
1	2	3	mkdir	{*pathname=/parentB, mode*=493}	0
2	4	6	write	{*fd*=5, *count*=1024}	1024
3	5	6	open	{*pathname=/parentB/.lock, flags*=3101, *mode*=666}	6
4	7	8	close	{*fd*=5}	0
5	7	9	unlink	{*pathname=/parentB/.lock*}	0
6	12	13	creat	{*pathname=/parentC/*}	0
7	14	15	stat	{*pathname=/parentC/.lock*}	−1

As Figuras 2.2a e 2.2b reportam a relação de dependência entre as requisições do rastro da Tabela 2.1 quando se aplica as políticas de ordenação *conservative* e *FS*, respectivamente. A Figura 2.3 representa uma possível reprodução do rastro da Tabela 2.1, respeitando as relações de dependência definidas, e combinando as políticas de temporização *fullspeed* e *timestamp*. Consideramos que a duração da execução das requisições reproduzidas tem o mesmo valor da captura. Para o exemplo criado, percebe-se que a política *fullspeed* ignora os intervalos sem atividade considerados pela política *conservative*. Ainda, a política *FS*



(a) Relação de dependência das requisições segundo a política de ordenação *conservative*.



(b) Relação de dependência das requisições segundo a política de ordenação *FS*.

Figura 2.2: Relação de dependência das requisições segundo as políticas de ordenação *conservative* e *FS*.

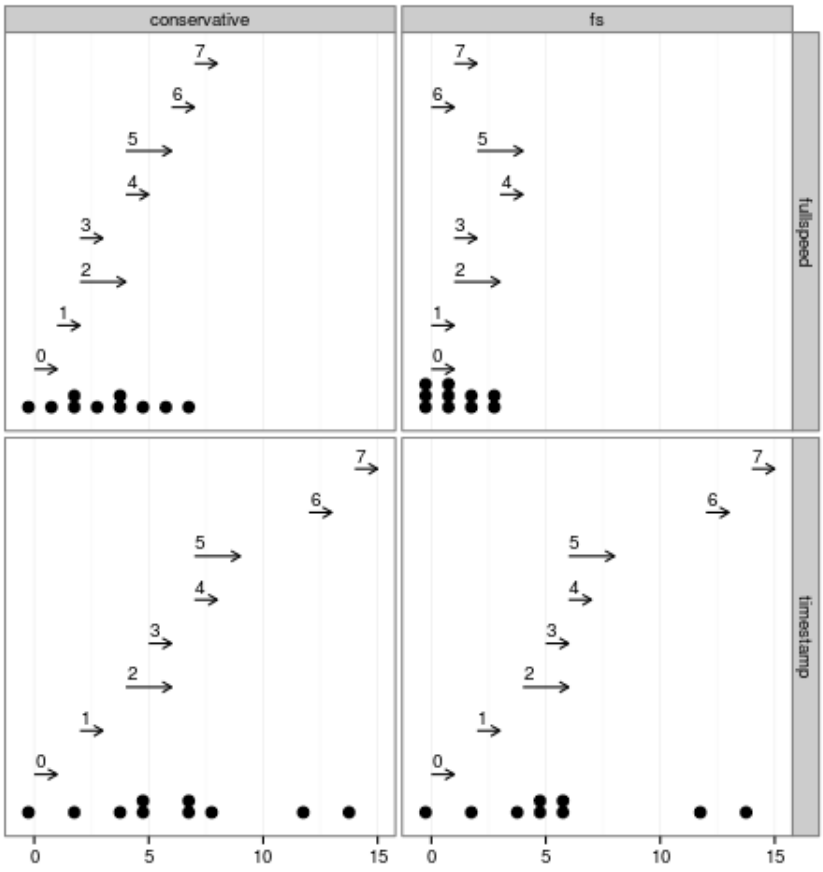


Figura 2.3: Um cenário de reprodução possível para o rastro descrito na Tabela 2.1, obedecendo as relações de dependência definidas nas Figuras 2.2b e 2.2a.

impõe um número menor de relações de dependência entre as requisições, por essa razão, permite uma reprodução com um nível maior de concorrência, tal como acontece em sua combinação com a política *fullspeed*.

Projeto e implementação

Independentemente do método usado para definir a carga de trabalho, há ainda a preocupação de como essa carga é efetivamente reproduzida. Com essa finalidade, podemos discriminar dois tipos de projetos para reprodutores: baseados em compilação ou em processamento de eventos.

No primeiro tipo, um compilador toma como entrada o rastro capturado, e gera um programa que ao ser executado, gera uma carga que é equivalente à carga de trabalho capturada; a reprodução se dá, portanto, pela execução do programa gerado. Esse é o projeto seguido pelo reprodutor `ARTC` [66]. O código gerado por seu compilador cria uma *thread* para cada *thread* que aparece no rastro capturado. Cada uma dessas *threads* reproduz a mesma coleção de requisições com as quais aparece associada no rastro. Além disso, os dados que compõem as requisições, por exemplo, seus argumentos, também são alocados estaticamente durante a compilação. O processo de compilação também é responsável por levar a cabo as regras das políticas de ordenação: cada requisição inclui uma trava indicando qual outras *threads* podem ser bloqueadas, caso uma requisição a ser executada por essas *threads* dependa daquela requisição.

Esse também é o modelo de projeto seguido pelo reprodutor `Replayfs` [41]. Esse reprodutor difere do `ARTC` quanto ao tipo das chamadas capturadas. Enquanto que o `ARTC` coleta e reproduz chamadas ao sistema, o reprodutor `Replayfs` captura funções no nível da API VFS; isso implica que a reprodução precisa ser feita em modo privilegiado. Para tanto, seu compilador gera um módulo que é inserido no núcleo do sistema operacional. Essa decisão de projeto reduz o tempo gasto na troca de contexto, verificação de parâmetros e cópia de *buffers*, que seriam necessários caso a reprodução fosse feita por um processo não privilegiado. Por consequência, a eliminação dessas atividades contribui para uma cronometragem mais acurada das requisições. Embora a motivação para essas otimizações seja coerente, a reprodução no nível da API VFS apresenta desvantagens. A maior delas, como apontam Traeger et al. [65], é que não há portabilidade: versões diferentes de um mesmo sis-

tema operacional podem ter API VFS diferentes. Talvez por esse motivo, além da exigência de inclusão de módulos no núcleo do sistema para a reprodução, o reprodutor *Replayfs* seja pouco popular.

No segundo tipo de projeto, baseado no processamento de eventos, um mesmo programa genérico é responsável por executar uma carga de trabalho arbitrária dada como entrada. Esse é o modelo seguido pelo reprodutor `TBBT` [50] e mais recentemente pelo reprodutor que desenvolvemos [55].

O reprodutor `TBBT` reproduz cargas de trabalho de servidores NFS. Seu projeto é estruturado em termos da execução de 3 *threads*, as quais são responsáveis pela leitura do rastro, envio e recebimento de requisições, respectivamente. A *thread* responsável pela leitura do rastro, lê os dados de entrada de maneira contínua (ou seja, o rastro não é carregado inteiramente ao início da reprodução), os interpreta e insere em um *buffer* de requisições a serem reproduzidas. O número de requisições presentes no *buffer* deve ser grande o suficiente para assegurar que a *thread* que envia as requisições para o servidor NFS sempre tenha trabalho disponível. A *thread* de envio gera requisições para o servidor NFS através de chamadas RPC assíncronas; as respostas para essas requisições são recebidas pela terceira *thread*. A *thread* de envio também é responsável por garantir que as requisições atendam as políticas de ordenação e temporização definidas pelo método `TBBT`. Para isso, travas de escrita e leitura são associadas às requisições. A *thread* de envio tenta o acesso exclusivo às travas, enquanto a *thread* de recebimento de requisições as libera.

Em nosso reprodutor, o qual discutiremos com mais detalhes no Capítulo 6, diferente do que foi adotado no reprodutor `TBBT`, as regras das políticas de ordenação não são decididas em tempo de execução. Ao invés disso, a relação de dependência entre as requisições é especificada nos dados de entrada, junto com a descrição do tipo e argumentos das requisições que serão reproduzidas. Com isso, implementar uma nova política de ordenação não envolve a modificação do código fonte do reprodutor mas sim reprocessar os dados de entrada — o que reduz o esforço para experimentar novos métodos de reprodução.

2.5 Crítica da avaliação de desempenho de sistemas de arquivos

Na área de sistemas de arquivos e de armazenamento há poucos trabalhos dedicados a um estudo sistemático sobre métodos de avaliação de desempenho; geralmente, os esforços estão voltados para o desenvolvimento de novos sistemas. Uma exceção é o abrangente estudo sobre a qualidade da metodologia de avaliação de desempenho de sistemas de arquivos feito por Traeger et al. [65]. Nesse estudo, os autores compilam 106 artigos relacionados a sistemas de arquivos e armazenamento, escritos entre 1999 e 2008, para entender as razões usadas na escolha do método de avaliação de desempenho, bem como a maneira como esses métodos são empregados, além de como os resultados são analisados e descritos. Os autores concluem que a grande maioria dos trabalhos analisados apresenta problemas metodológicos: é emblemático citar que, em apenas dois dos 106 artigos compilados a avaliação experimental foi executada mais de uma vez, o que afeta sobremaneira a validade estatística da maior parte dos trabalhos na área.

Um segundo problema apontado pelos autores é a falta de justificativa embasada para a escolha da metodologia. Por exemplo, em 20 experimentos dos 148 que empregaram *macrobenchmarks* criados por terceiros, usou-se a popularidade das ferramentas como justificativa para sua escolha. Ainda mais grave, em 28 experimentos não foi fornecida justificativa alguma. Diante disso, Traeger et al. indicam que os *macrobenchmarks* são escolhidos por sua popularidade, a despeito de sua adequação [65].

Um terceiro problema diz respeito à falta de suporte à replicação dos resultados. Esse problema é especialmente grave em trabalhos que criam *benchmarks* ad hoc, uma vez que raramente o código fonte das ferramentas criadas é liberado e na falta desses, não se fornecem detalhes suficientes para nova implementação.

Mais importante para o nosso trabalho, o diagnóstico de Traeger et al. sobre reprodução de rastros não é menos grave [65]. Dos 19 artigos compilados que usam esse método, 15 não descrevem a ferramenta de reprodução usada. Além disso, 11 não definem os métodos de ordenação e temporização usados. Ainda, dentre os 5 artigos que capturaram seus próprios rastros para a reprodução, 4 não especificam como isso foi feito. É importante ressaltar que, embora o levantamento feito por Traeger et al. seja relevante para apontar a falta de consenso

no uso dos métodos de avaliação de desempenho, não é suficiente para avaliar a consequência dos problemas metodológicos, nem para contribuir no desenvolvimento de novos métodos. Preencher essa lacuna é um dos principais objetivos desse trabalho.

Posteriormente, em 2011, Tarasov et al. [62] atualizaram o estudo de Traeger et al. [65] (esses dois estudos contam com autores em comum) apresentando conclusões semelhantes. Por exemplo, a criação de *benchmarks* ad hoc continuou bastante popular, a despeito da existência de ferramentas disponíveis que poderiam gerar a mesma carga de trabalho.

Os diagnósticos de Traeger et al. [65] e de Tarasov et al. [62] ainda são válidos. Analisamos as conferências consideradas pelos autores, de 2011 em diante, e encontramos os mesmo problemas metodológicos. Por exemplo, dentre 16 trabalhos que empregaram reprodução de rastros, 8 não descrevem que reprodutor usaram. Por sua vez, em 5 trabalhos os autores desenvolverem reprodutores ad hoc, muito embora os descrevam em poucos detalhes. Por fim, 3 trabalhos fogem à regra: seus autores empregaram reprodutores de uso mais geral, desenvolvidos por outros autores.

A opinião de Tarasov et al. sobre a falta de qualidade na avaliação de desempenho de sistemas de arquivos é representativa: "We claim that file system benchmarking is actually a disaster area full of incomplete and misleading results that make it virtually impossible to understand what system or approach to use in any particular scenario." [62].

Mais recentemente, em sua proposta de doutoramento, Tarasov incorpora essa visão pessimista e desaconselha o uso do método de reprodução de rastros [61]. Segundo o autor, os rastros costumam ser muito grandes, portanto, difíceis de manipular e compartilhar. Além do mais, são inflexíveis para testar cargas de trabalho um pouco diferentes daquelas presentes durante o momento de captura. Como alternativa, Tarasov sugere um retorno aos métodos conhecidos de criação de cargas sintéticas baseadas em rastros de utilização [43, 60].

No início deste trabalho, não conhecíamos nenhum estudo que avaliasse a qualidade da metodologia da reprodução de rastros. Mesmo o trabalho de Zhu et al., que se destaca por propor as políticas de ordenação e temporização discutidas na Seção 2.4.2, não fornece nenhuma avaliação sobre a eficácia das políticas que sugere [50]. Devido a essa lacuna, nós conduzimos em 2013 um trabalho [55] no qual avaliamos as diferenças entre as políticas propostas por Zhu et al. [50]; de fato, a escolha das políticas afeta os resultados obtidos com o reprodutor.

Posteriormente, Zev et al. também consideraram o problema da qualidade dos métodos de reprodução de rastros [66]. Como descrevemos na Seção 2.4.2, os autores desenvolveram novas políticas de reprodução que são equivalente às políticas originalmente descritas por Zhu et al. [50]. Aqueles autores sustentam que a política `ARTC` é precisa o suficiente na inferência da carga de trabalho, e portanto, é possível reabilitar a reprodução de rastros como uma alternativa fiel à experimentação em ambiente real.

Acreditamos que a tese de Tarasov é demasiadamente pessimista; deve haver cenários em que a reprodução de rastros é útil e acurada. Por sua vez, o trabalho de Zev et al. [66] peca pelo excesso de otimismo; como veremos nos capítulos seguintes, a metodologia de reprodução não se restringe à inferência da carga de trabalho. Outros fatores, tais como o processo de captura dos dados, podem afetar os resultados do método.

Desse modo, nossa proposta é um contraponto em relação a esses dois trabalhos. Sustentamos que precisamos avaliar a qualidade de todos os aspectos — da captura à reprodução, incluindo a inferência da carga de trabalho — para decidir sobre a utilidade da metodologia de reprodução de rastros.

Capítulo 3

Conceitos de metrologia

Métodos de avaliação de desempenho são aspectos chave em ciência da computação. São empregados, por exemplo, quando se quer comparar projetos diferentes de um mesmo sistema, configurar um sistema para otimizar seu serviço e planejar a capacidade de infraestruturas de computação para determinada qualidade de serviço. Dada essa relevância, metodologias que sistematizam as atividades de avaliação de desempenho têm sido consideradas há um bom tempo.

Nessa direção, Jain [39] considera os seguintes passos comuns quando se conduz uma avaliação de desempenho:

1. Definir o objetivo da avaliação bem como delimitar o sistema sob estudo;
2. Identificar o serviço provido pelo sistema bem como os resultados possíveis desse serviço;
3. Selecionar as métricas de avaliação de desempenho;
4. Listar os fatores que afetam o desempenho;
5. Selecionar os fatores para estudo;
6. Selecionar a técnica para a avaliação de desempenho;
7. Selecionar a carga de trabalho que será submetida ao sistema;
8. Projetar o experimento;

9. Analisar e interpretar os dados;
10. Apresentar os resultados.

Le Boudec [45] também esquematiza como avaliar o desempenho de sistemas:

1. Identificar o objetivo da avaliação de desempenho;
2. Definir a métrica usada na avaliação de desempenho;
3. Definir a carga de trabalho que será usada na avaliação de desempenho;
4. Escolher o método de solução empregado na avaliação;
5. Estabelecer os fatores, tanto do sistema quanto da carga de trabalho, que podem afetar o desempenho medido.

Em adição aos passos básicos listados acima, Le Boudec considera também importante conhecer os gargalos típicos de determinados tipos de sistemas; é importante que no estabelecimento de fatores, esses gargalos sejam conhecidos. Além disso, Le Boudec também discute como definir hipóteses, projetar experimentos e avaliar a acurácia dos resultados obtidos.

Há muitos pontos em comum nas visões de Jain e Le Boudec. A avaliação de desempenho começa com um objetivo definido, seja ele dimensionar o sistema, entender seu comportamento em caso de sobrecarga, comparar alternativas, entre outros. Métricas objetivas precisam ser escolhidas como critério para a análise de desempenho. É necessário também identificar fatores que impactam o desempenho bem como a carga que será submetida ao sistema; isso não pode ser feito ser conhecer o sistema nem como ele é usado (explicitamente declarado no passo 2 do esquema de Jain e discutido ao longo do texto de Le Boudec). Em seguida, a técnica de avaliação (ou método de solução) precisa ser escolhida; os autores citam a construção de modelos analíticos, a execução de simulações e a condução de experimentos de medição com sistemas reais. Qual seja o método escolhido é importante projetar como o experimento será conduzido, de maneira que seja obtido o máximo de informação com o menor esforço possível. Por fim, os resultados precisam ser analisados. Nessa análise é fundamental reconhecer que resultados obtidos são variedades aleatórias e precisam ser sua variabilidade discutida.

Em particular, Jain discute em profundidade questões relacionadas com a carga de trabalho (passo 7), incluindo como expressar a carga de trabalho e produzi-la de maneira representativa[1], questões e métodos para projetar a condução de experimentos (passo 8) e os métodos estatísticos para analisar e interpretar os resultados obtidos com a avaliação de desempenho (passo 9). Por sua vez, Le Boudec discute em mais detalhes a geração de cargas de trabalho (passo 3), padrões de desempenho e gargalos típicos bem como métodos estatísticos para análise de resultados experimentais.

Alguns aspectos de medição são pouco discutidos tanto por Jain quanto por Le Boudec — entre esses aspectos está a medição de desempenho realizada com sistemas reais. Nesse tipo de medição, ilustrado na Figura 3.1, uma carga de trabalho é submetida ao sistema de fato, num ambiente de experimentação. Os valores das métricas escolhidas são obtidos através de um instrumento de medição. É importante levar em conta, como bem apontado por Brendan Gregg [33], que perturbações podem alterar os resultados das medições, aí incluídas causas como outras cargas de trabalho em execução no ambiente.

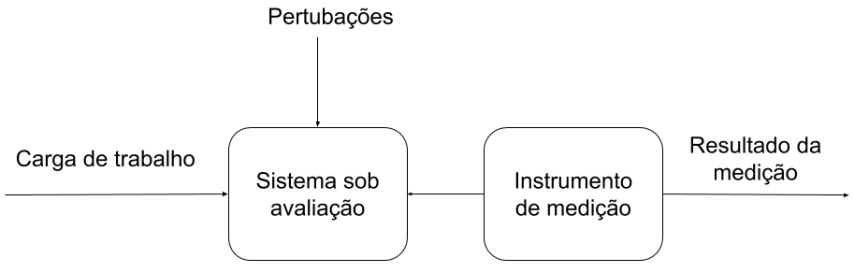


Figura 3.1: Em avaliações de desempenho experimentais, uma carga de trabalho é submetida ao sistema sob avaliação. O desempenho do sistema é medido através de um instrumento de medição. Esse desempenho pode ser afetado por perturbações no ambiente experimental.

[1]Feitelson [30] provê um tratamento extensivo sobre este tópico.

Sobre esse aspecto Le Boudec considera que "... tal como em física, é difícil realizar medições sem provocar nenhuma perturbação ... quando as medições são realizadas pelo próprio sistema sob avaliação, o seu impacto precisa ser analisado com cuidado" [46]. Embora reconheça o problema, está fora do escopo de seu trabalho um estudo de como analisar esse impacto bem como mitigá-lo. Jain considera instrumentos de medição sob o termo "monitor"e compila alguns problemas no projeto de monitores [40]. Por exemplo, ele indica que esses consomem os recursos do sistema sob avaliação e por consequência podem provocar interferência na própria medição. Também cita a resolução do monitor como o grão menor de informação que pode ser coletada (tal como o menor intervalo de tempo em que o monitor pode coletar dados). Assim como Le Boudec, Jain não considera em maior profundidade como avaliar o impacto da interferência dos monitores nos resultados das medições.

Na seção seguinte, descrevemos os conceitos de metrologia. Em linhas gerais, essa disciplina trata dos erros associados com instrumentos em medições experimentais. Metrologia pode ser vista como uma das partes das metodologias descritas por Jain e Le Boudec; sua contribuição principal diz respeito a como melhorar a qualidade de medições experimentais de avaliação de desempenho.

3.1 Metrologia

Na visão de metrologia do Guia para a Expressão da Incerteza de Medição (GUM [26]), a **medição** é um processo experimental usado para obter o valor do **mensurando**, i.e., a grandeza que se pretende medir. [2]

O **procedimento de medição** é feito conforme um **método de medição**, o qual descreve como operar um **instrumento de medição** para obter o valor do mensurando. Assim, a qualidade de uma medição depende da adequação do seu método e do seu instrumento. Além disso, uma vez que os instrumentos de medição muitas vezes são operados por humanos, a habilidade do operador do instrumento também afeta a qualidade da medição.

Devido às imperfeições presentes no método de medição, no seu instrumento, bem como na operação do seu instrumento, o **valor medido de uma grandeza** difere do **valor verda-**

[2]Termos em negrito compõem o jargão de metrologia catalogado no VIM (Vocabulário Internacional de Metrologia) [13, 31].

deiro da grandeza.

O postulado básico da metrologia é que é impossível conhecer o valor verdadeiro de uma grandeza. Como consequência, o problema mais importante tratado pela metrologia diz respeito a como obter estimativas para o valor verdadeiro de uma grandeza por meio de procedimentos de medição imperfeitos [56].

As imperfeições da medição afetam a **exatidão da medição**, que é o grau de concordância entre o valor medido e o valor verdadeiro de uma grandeza. A exatidão da medição se relaciona com os conceitos de **precisão de medição** e **veracidade de mediação**. A precisão de medição quantifica o grau de concordância entre valores medidos obtidos em medições repetidas, enquanto a veracidade de medição quantifica o grau de concordância entre o valor medido típico e um **valor de referência**, i.e., um valor usado no lugar do valor verdadeiro, e desconhecido, do mensurando.

A diferença entre o valor medido e o valor de referência implica no **erro de medição**, o qual apresenta dois componentes que apresentam comportamentos diferentes em medições repetidas: o **erro sistemático** e o **erro aleatório**; enquanto o erro sistemático permanece constante ou varia de maneira previsível, o erro aleatório varia de maneira imprevisível. A **tendência de medição** é a estimativa do **erro sistemático**.

A combinação de precisão e tendência de medição define a **incerteza da medição**, um parâmetro que caracteriza a dispersão dos valores atribuídos a um mensurando.

3.2 Metrologia aplicada na avaliação de desempenho de sistemas de arquivos

Embora os conceitos de metrologia não sejam populares em ciência da computação, não são inéditos. De fato, há um bom número de trabalhos que aplicam algum conceito de metrologia embora, na maioria da vezes, não explicitamente declarado.

Por exemplo, em um dos primeiros trabalhos baseados na análise de rastros de utilização, Ousterhout et al. instrumentaram as chamadas ao sistema do núcleo 4.2 BSD para analisar a atividade do sistema de arquivos [52]. Nesse trabalho, seus autores estavam interessados na quantidade de dados coletados e no impacto do programa coletor de rastros sobre as estações de trabalho em que executava — ou seja, aspectos de **intrusividade**. Em particular,

os autores estavam interessados em reduzir a intrusividade do sistema de captura de rastros de maneira a não perturbar a experiência do usuário na estação de trabalho.

O coletor de rastros *DFSTrace* baseado em instrumentação do núcleo do sistema operacional, desenvolvido há mais de duas décadas no escopo do sistema de arquivos distribuído Coda [35], também incorpora conceitos de metrologia de modo implícito. Os autores da ferramenta usaram o termo *sobrecarga*[3] quando se referiam à medição do erro sistemático da ferramenta. Essa sobrecarga foi avaliada comparando o tempo total de execução do *Andrew Benchmark* [49] em um sistema de arquivos sem instrumentação com o tempo de execução em um sistema de arquivos instrumentado. Em essência, esse é o método para calcular a tendência de medição conforme os protocolos de metrologia. De qualquer modo, embora os autores meçam os erros sistemáticos, não consideram o arcabouço de metrologia relacionado com a correção desse tipo de erro.

Mais recentemente, Anderson considerou os métodos de captura de rastros para servidores NFS corporativos [17]. O autor detectou que as ferramentas de captura de rastros para servidores NFS de então não eram capazes de dar conta de cargas de trabalho tão intensas. Considerando esse cenário, a sua análise teve como objetivo aferir a praticabilidade de novos métodos de captura para cargas de trabalho tão intensas. Esse tipo de análise é entendida em metrologia como um estudo das condições de medição, tais como **rated operating conditions** e **limiting operating conditions** [13].

Anderson et al. desenvolveram a ferramenta `Buttress` como uma solução para os problemas de controle de temporização em reprodutores de rastros de E/S [19]. Eles avaliaram sua ferramenta com respeito ao erro de emissão, a diferença entre o instante de tempo planejando para a reprodução de uma requisição e o instante de tempo no qual a requisição foi de fato reproduzida. Eles analisaram o erro de emissão como uma função do número de requisições de E/S reproduzidas em um segundo. Esse método de análise está em acordo com a definição do **intervalo de medição** em metrologia.

Como descrevemos na seção anterior, Zhu et al. não avaliaram experimentalmente as políticas de ordenação e temporização de seu reprodutor `TBBT` [50]. Em nossa avaliação das políticas do `TBBT`, comparamos o tempo de resposta das requisições reproduzidas com o tempo de resposta original do sistema de arquivos, conforme registrado no rastro captu-

[3] Do termo original *overhead.*

rado [55]; em suma, uma verificação de tendência de medição em termos de metrologia.

Zev et al. também analisaram o reprodutor `ARTC`, descrito no capítulo anterior, através de um estudo de tendência de medição: compararam o tempo total de execução de um *microbenchmark* e o tempo total de reprodução do rastro de execução deste *microbenchmark* [66]. Além disso, os autores também avaliaram as reproduções geradas pela ferramenta `ARTC` quando essas acontecem em um ambiente computacional diferente daquele usado durante a captura do rastro; visto pela ótica da metrologia, avaliaram a ferramenta em **condições de reprodutibilidade**.

Como vimos, é possível perceber conceitos de metrologia em muitos trabalhos de avaliação de desempenho de sistemas de computação. Em comum, esses trabalhos discutem a adequação da aplicação de um método de medição novo ou ainda não amplamente aceito. Em termos de metrologia, trata-se de validação (ou teste) de métodos de medição. No próximo capítulo, discutiremos um protocolo bastante utilizado para conduzir esse tipo de teste. Como descrevemos na Introdução, ainda não há uma análise embasada da adequação dos métodos de captura e reprodução de rastros de utilização de sistemas de arquivos. Assim, usaremos esse protocolo nas provas de conceito descritas no Capítulo 5 e 6.

Capítulo 4

Validação de métodos de medição

Neste capítulo apresentamos o ramo da metrologia dedicado à validação de métodos de medição. Em particular, consideramos a abordagem de validação por um único laboratório (Seção 4.1). Em seguida, consideramos em mais detalhes os procedimentos de calibração de instrumentos de medição (Seção 4.2). Esses aspectos de metrologia tornam o presente documento auto contido e serão a base para o entendimento das provas de conceito apresentadas nos Capítulos 5 e 6. Adicionalmente, no Apêndice A, discutimos alguns aspectos de metrologia que podem ser úteis em outros estudos.

Qualquer aplicação de metrologia requer que a incerteza do procedimento de medição seja adequada ao propósito pretendido da aplicação. **Validação** é o ramo da metrologia que trata de aferir as evidências desta adequação.

Para evidenciar que um método de medição é adequado a um determinado propósito, as técnicas de validação precisam identificar e quantificar as fontes de incerteza que afetam os resultados da medição [29]. Quando possível, isto pode ajudar a reduzir os erros de medição. A incerteza da medição devida aos erros remanescentes pode ser estimada. Essa estimativa apoia a decisão sobre se o método sob validação é adequado ou se é necessário desenvolver um novo método (ou relaxar alguns dos requisitos colocados para sua adoção).

4.1 Validação por um único laboratório

Nesta seção, descrevemos um protocolo clássico de validação que é aplicado para contabilizar fontes de incerteza que são possíveis de controlar em um único laboratório [29]. Usa-

remos este protocolo nas provas de conceito apresentadas nos Capítulos 5 e 6. O protocolo envolve os seguintes passos:

1. Especificação do mensurando;
2. Definição do procedimento de medição;
3. Identificação das fontes de incerteza;
4. Caracterização da medição;
5. Calibração;
6. Determinação da incerteza de medição.

O mensurando e o procedimento de medição são formalizados nos passos *Especificação do mensurando* e *Definição do procedimento de medição*, respectivamente. No passo *Identificação das fontes de incerteza*, as possíveis fontes de incerteza, bem como as limitações típicas dos instrumento de medição escolhidos que podem afetar a qualidade da medição são apontadas. Nessa parte do protocolo é possível ser afetado por fatores que *não sabemos que não sabemos*[1]: fatores importantes e desconhecidos podem ser desconsiderados no levantamento, e portanto, a incerteza na medição será subestimada. Em outras ciências, esse problema é remediado através do incentivo ao uso de instrumentos de mesmo tipo (p.ex. pipetas, béqueres e densímetros), de maneira que suas limitações sejam mais prováveis de já serem conhecidas. Em outras comunidades, os pesquisadores se apoiam editando catálogos com compilações de fontes de incerteza que afetam seus procedimentos de medição [36]. Infelizmente, essa cultura não está presente na comunidade de ciência da computação.

No passo de *Caracterização da medição*, experimentos são conduzidos para a analisar o desempenho do método de medição e dos instrumentos. Normalmente, esse experimentos são analisados em termos dos valores da precisão e tendência, mas outros métricas específicas para cada propósito são também comuns. Por exemplo, a insensibilidade do procedimento de medição quanto aos efeitos das fontes de incerteza definem uma característica

[1]Donald Rumsfeld, ex-Secretário de Defesa norte-americano, em um discurso sobre a falta de provas da existência de armas de destruição em massa no Iraque, definiu que há coisas que *sabemos que sabemos*, coisas que *sabemos que não sabemos*, e coisas que *não sabemos que não sabemos*. Em política e metrologia, as últimas costumam ser as mais difíceis.

importante, a **seletividade**. Quanto mais independente são os valores medidos em relação às fontes de incerteza, maior a seletividade do procedimento de medição. Por outro lado, a **estabilidade** de um instrumento de medição é a propriedade segundo a qual este mantém as suas propriedades metrológicas constantes ao longo do tempo.

No passo de *Calibração*, se estabelece um procedimento de ajuste dos valores medidos para torná-los mais próximos dos valores verdadeiros. O procedimento de calibração, que deve ser aplicado quando a tendência das medições é significantemente alta, é feito através da compensação dos erros sistemáticos, tal como encontrados na etapa de caracterização anterior. Na Seção 4.2, discutimos em mais detalhes como construir curvas de calibração, a ferramenta básica do procedimento de calibração, bem como quantificar a qualidade desse procedimento.

Finalmente, no último passo — *Determinação da incerteza de medição* — a incerteza do método sob validação é encontrada com base nos resultados dos experimentos conduzidos no passo *Caracterização da medição* e após o tratamento destes resultados pelo procedimento de calibração. A incerteza da medição discrimina precisão e tendência (afetadas pelos erros aleatórios e sistemáticos, respectivamente). A componente $u(R_w)$ contabiliza a incerteza em virtude dos erros aleatórios, enquanto a componente $u(bias)$ contabiliza a incerteza devida aos erros sistemáticos. Estas componentes são agregadas em uma única medida — a incerteza combinada, u_c — dada pela seguinte equação [29]:

$$u_c = \sqrt{u(R_w)^2 + u(bias)^2}$$

Ambas as componentes são calculadas com base na execução de n experimentos replicados usando o método de medição sob avaliação. Esses consistem em medições replicadas do mensurando. Em sua execução, é importante tomar medidas para que alterações conhecidas no instrumento de medição ou no ambiente de medição causadas por um experimento anterior sejam reparadas antes da execução do experimento subsequente, de maneira que cada experimento replicado seja conduzido de acordo com as condições esperadas de aplicação do método na prática.

A componente $u(R_w)$ é dada pelo desvio padrão dos valores medidos nos n experimentos; estes valores são usados após correção por calibração, quando possível.

Por sua vez, a componente $u(bias)$ é dada pelos valores medidos (após correção por

calibração) e por um coleção de n valores de referência. Essas referências são usadas para calcular o erro sistemático (a diferença entre um valor medido e o valor de referência) em cada experimento. Seguindo este esquema, a componente $u(bias)$ é definida como

$$u(bias) = \sqrt{RMS_{bias}^2 + u(C_{ref})^2}$$

em que o termo RMS_{bias} é o valor médio quadrático para a tendência de medição encontrado nos n experimentos e o termo $u(C_{ref})$ diz respeito à incerteza nos valores de referência (embora mais confiáveis, há alguma incerteza nos valores de referência). Para o conjunto de n experimentos, o termo RMS_{bias} é dado por

$$RMS_{bias} = \sqrt{\frac{\sum_{i=1}^{n}(bias_i)^2}{n}}$$

Para cada experimento i, o termo $bias_i$ é dado pelo erro percentual relativo entre o valor medido x_i e o valor de referência ref_i. Desse modo, temos

$$bias_i = (x_i - ref_i) \cdot 100$$

Por sua vez, o termo $u(C_{ref})$ é dado por

$$u(C_{ref}) = \sqrt{\frac{\sum_{i=1}^{n} u(ref_i)^2}{n}}$$

em que $u(ref_i)$ é a incerteza do valor de referência adotado no i-ésimo experimento.

Os resultados da incerteza da medição são reportados na forma $y_m \pm u_c$, em que y_m é a média dos valores medidos nas medições replicadas. Esta incerteza combinada indica que os valores medidos se encontram no intervalo $[y_m - u_c, y_m + u_c]$ com um nível de confiança de 68%. Quando se requer níveis de confiança maiores, usa-se as **incertezas expandidas** $y_m \pm 2 \cdot u_c$ e $y_m \pm 3 \cdot u_c$. Esses intervalos correspondem aos níveis de confiança de 95.5% e 99.7%, respectivamente.

4.2 Calibração

Nesta Seção, nós consideramos alguns aspectos da calibração de instrumentos de medição e em particular, alguns métodos para construção de curvas de calibração bem como métodos

para quantificar os potenciais erros devidos às imperfeições da calibração.

4.2.1 Visão geral

O processo de calibração melhora a qualidade das medições ao compensar os erros sistemáticos. O processo de calibração baseia-se na construção e uso de curvas de calibração, tal como mostrado na Figura 4.1. Curvas de calibração relacionam valores de referência com valores medidos; para cada objeto de referência com valor R_0 e um valor medido Y_0 (resultado da medição da referência com o instrumento sob validação), existe um ponto $\langle R_0, Y_0 \rangle$ na curva de calibração. Uma vez construída a curva, a calibração é obtida aplicando-se a relação inversa: dado um valor medido (eixo y) obtém-se um valor corrigido (eixo x).

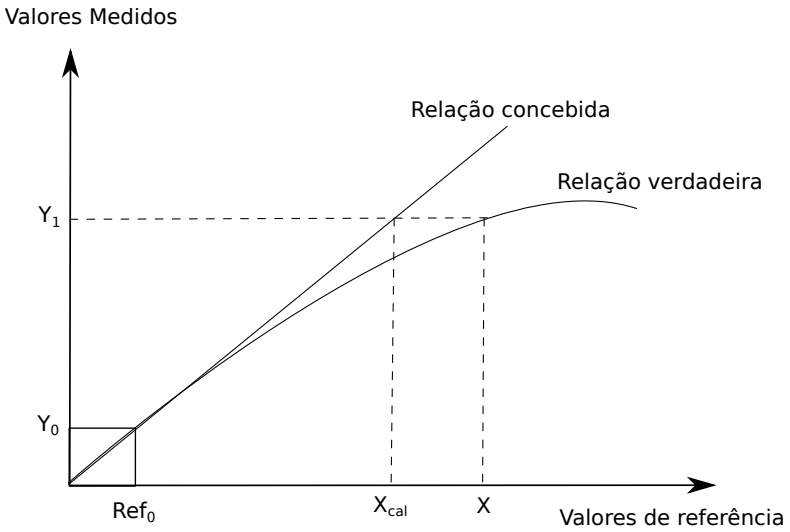


Figura 4.1: Curvas de calibração relacionam valores de referência com os valores medidos destas referências. Desse modo, para uma dada referência com valor R_0 e um valor medido Y_0, obtido por uma medição feita com o procedimento sob validação, temos um ponto $\langle R_0, Y_0 \rangle$ na curva de calibração. A relação definida pela curva de calibração é usada para corrigir as medições; para um dado valor medido, encontra-se um valor correspondente no eixo das referências.

A curva de calibração de um instrumento ideal (perfeito) seria uma linha reta, com inclinação igual a 1 e ponto de intersecção com o eixo y igual a zero (ou seja, os valores medidos são iguais aos valores de referência). Em contraste, as curvas de calibração de instrumentos de mediação reais (imperfeitos) se parecem com os padrões mostrados na

Figura 4.2. As curvas de calibração de instrumentos com *erro de ponto zero* se mostram como linhas paralelas à curva de calibração do instrumento ideal, indicando que o erro sistemático desses ao longo do intervalo de medição é constante. As curvas de calibração de instrumentos com *erro de ganho* apresentam inclinação diferente de 1, indicando que o erro sistemático não é constante ao longo do intervalo de medição, mas varia a uma taxa constante. Finalmente, as curvas de calibração de instrumentos com *erro não-linear* apresentam erro sistemático variável, tal como as curvas de instrumentos de *erro de ganho*, entretanto, a taxa de variação do erro não é constante ao longo do intervalo de medição.

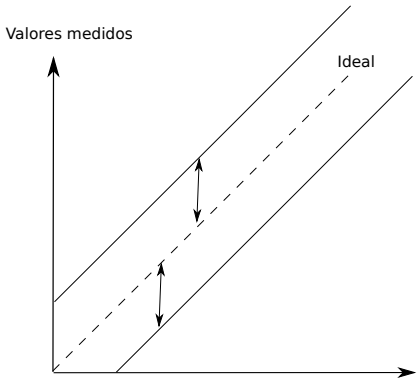
Saber de antemão a categoria de um instrumento de medição ajuda a decidir o número de referências que serão usadas para definir sua curva de calibração. Em certos casos, é difícil ou custoso obter ou preparar as referências. Portanto, é natural tentar minimizar o número de referências que serão usadas.

Quando se tem confiança que o procedimento de medição apresenta *erro de ponto zero* (baseando-se em processos similares já conhecidos) as curvas de calibração podem ser definidas usando o método de `ponto único`. Uma vez que o erro sistemático é constante ao longo do intervalo de medição, uma única referência é necessária para definir a curva de calibração. Quando o instrumento apresenta *erro de ganho*, aplica-se o método de `dois pontos`. Duas referências são necessárias para obter a inclinação e o ponto de interceptação.

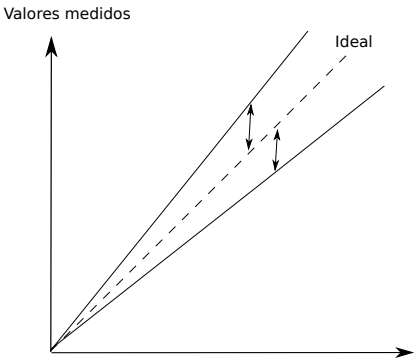
Quando não se pode assumir as premissas para a adoção dos métodos de `ponto único` e `dois pontos`, aplica-se um método de `múltiplos pontos`, como por exemplo, calibração por regressão linear. Embora sejam robustos, métodos de `múltiplos pontos` não são perfeitos. Isso pode ser porque a faixa de valores coberta pelas referências não cobre de maneira adequada o intervalo de medição pretendido para o instrumento, ou ainda devido à incerteza nos valores de referência, que embora baixa, pode ser significativa.

Devido a todas essas limitações, o processo de calibração inclui um passo adicional que considera a quantificação da qualidade da calibração, isto é, a quantificação dos erros relacionados com a correção dos resultados medidos.

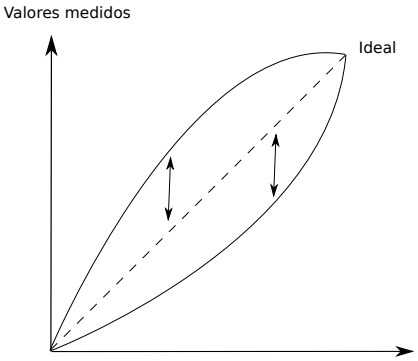
Uma maneira de calcular o erro de calibração leva em conta os intervalos de predição da regressão linear usada para construir a curva. De acordo com esse método, para uma curva com inclinação b_1 e ponto de intersecção b_0, obtida com base em n valores de referência,



(a) Padrão de *erro de ponto zero*.



(b) Padrão de *erro de ganho*.



(c) Padrão de *de erro não-linear*.

Figura 4.2: Padrões típicos de erros de instrumentos de medição. Para instrumentos que apresentam *erro de ponto zero*, a relação entre os valores medidos e os valores de referência aparecem como retas paralelas à relação ideal, nesse caso a magnitude dos erros é constante ao longo do intervalo de medição. Para instrumentos que apresentam *erro de ganho*, essa relação aparece com uma reta de inclinação diferente da relação ideal, assim, o erro é variável ao longo do intervalo de medição mas essa variação é constante ao longo do intervalo de medição. Por fim, a relação para instrumentos que apresentam *erro não-linear* implica não apenas em erros variáveis mas também em tendências de variação diferentes em diferentes faixas do intervalo de medição.

o valor corrigido $\hat{x}_0$ está em um intervalo de confiança $[\hat{x}_0^-,\ \hat{x}_0^+]$ com nível de confiança $(1-\alpha)100\%$. Este intervalo, mostrado na Figura 4.3, é dado por [44]:

$$\hat{x}_0^{\pm} = \hat{x}_0 \pm t_{(1-\alpha/2,n-2)} s_{\hat{x}_0} \tag{4.1}$$

em que $t_{(1-\alpha/2,n-2)}$ é o valor t com $n-2$ graus de liberdade e $s_{\hat{x}_0}$ é a variância dos valores

corrigidos pela calibração dada por:

$$s^2_{\hat{x}_0} = \frac{s^2_{y/x}}{b_1^2}\sqrt{\frac{1}{m} + \frac{1}{n} + \frac{(\hat{x}_0 - \overline{x})^2}{\sum_1^n (x_i - \overline{x})^2}} \tag{4.2}$$

em que $\overline{x}$ é a média dos valores de referência, x_i é o valor do i-ésimo valor de referência e m é o número de experimentos replicados conduzidos com cada valor de referência.

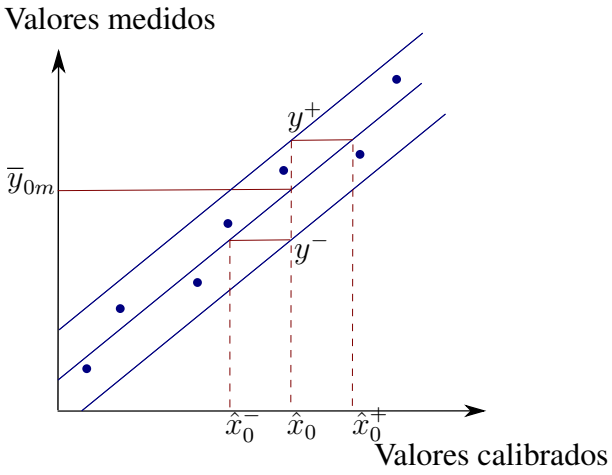


Figura 4.3: Intervalos de confiança para a calibração. A linha interna representa a curva de calibração encontrada por regressão linear, $\hat{x}_0$ é o valor calibrado obtido pela aplicação da curva de calibração na média dos valores medidos $\bar{y}_{0m}$, resultado de m medições replicadas. Devido às incertezas tanto da regressão quando do processo experimental empregado para obter $\bar{y}_{0m}$, um intervalo de confiança limitado por $\hat{x}_0^-$ e $\hat{x}_0^+$ é associado ao valor calibrado $\hat{x}_0$. As linhas externas delimitam os intervalos de predição para a regressão. Assim, os limites de calibração $\hat{x}_0^-$ e $\hat{x}_0^+$ são definidos de maneira que os seus valores correspondentes no eixo y encontrem os limites, y^- and y^+, do intervalo de predição.

Capítulo 5

Captura de rastros de utilização

Neste capítulo, aplicamos o protocolo de validação apresentado no Capítulo 4 para analisar métodos de captura de rastros de utilização de sistemas de arquivos. Na Seção 5.1 consideramos as 6 etapas do protocolo de validação. Nessa seção discutimos o projeto experimental da validação bem como os métodos que aplicamos para caracterizar os procedimentos de captura de rastros. Na Seção 5.2 discutimos os resultados da caracterização dos métodos de captura e a análise da incerteza de medição. Por fim, na Seção 5.3 finalizamos esse capítulo com um conjunto de recomendações derivadas dos resultados obtidos na prova de conceito.

5.1 Método

O protocolo de validação descrito no capítulo anterior é definido em 6 partes: i) especificação do mensurando; ii) definição do procedimento de medição; iii) identificação das fontes de incerteza; iv) caracterização da medição; v) calibração; e vi) determinação da incerteza de medição. A seguir, dedicamos uma subseção para cada uma das partes do protocolo.

5.1.1 Especificação do Mensurando

Nós consideramos a atividade de um sistema de arquivos como uma sequência da execução de suas declarações. Para capturar a atividade de um sistema de arquivos, um subconjunto de suas declarações é instrumentado. Levamos em conta o caso em que a métrica de interesse é o tempo de resposta, e portanto é preciso instrumentar tanto a primeira quanto a última

declaração do conjunto F das funções do sistema de arquivos. O rastro representa a atividade desse sistema como uma sequência de eventos T. Cada evento corresponde à execução de uma função do sistema de arquivos, incluindo informação de temporização e de contexto tal como argumentos e valor de retorno da função capturada. Essa informação dá suporte à reprodução dos rastros. Assim, cada evento $t_k \in T$ é uma tupla $\langle f_k, b_k, e_k, c_k \rangle$, que representa a execução de uma função do sistema de arquivos $f_k \in F$, em que b_k e e_k são as marcações de tempo da execução da primeira e última declarações da função f_k, e c_k é a informação contextual coletada.

Como mencionado anteriormente, nessa prova de conceito o mensurando é o tempo de resposta do sistema de arquivos. Para cada evento $t_k = \langle f_k, b_k, e_k, c_k \rangle$ da sequência T, o tempo de resposta de t_k é dado pelo intervalo que decorre entre a execução da última e primeira declarações de f_k, i.e. $e_k - b_k$.

5.1.2 Procedimento de medição

Nesta seção definimos os procedimentos de captura de rastros. A definição inclui as condições (o ambiente) em que as medições devem ser conduzidas, os instrumentos (as ferramentas de captura de rastros), e os procedimentos empregados para usar esses instrumentos.

Ambiente experimental

Conduzimos os experimentos de captura de rastros em uma estação de trabalho Intel E6550 Core 2 Duo 2.33GHz, com 2 GB de memória principal. A máquina tem dois discos rígidos: um disco SATA de 5400 RPM com 32 MB de cache, e um disco SATA de 7200 RPM com 8 MB de cache. Instrumentamos um sistema de arquivos `ext4` montado no primeiro disco rígido. Na máquina estava instalado o sistema operacional Linux na versão 2.6.32-41. A máquina foi usada de maneira exclusiva durante a execução das medições [1].

[1] Este isolamento se justifica em experimentos que pretendem avaliar os métodos de captura. Entretanto, nos experimentos típicos de captura de rastros, normalmente não é possível ter controle suficiente sobre o ambiente para obter esse isolamento.

Instrumentos de medição

Nessa prova de conceito, consideramos duas ferramentas de captura de rastros: `SystemTap` e `strace`. A ferramenta `SystemTap` permite a interceptação de declarações arbitrárias do núcleo do sistema operacional. Além disso, permite registrar rotinas que serão chamadas quando o sistema operacional executar a declaração instrumentada. Em nossa prova de conceito, usamos rotinas para gerar e armazenar os dados que farão parte do rastro de utilização. Por sua vez, a ferramenta `strace` não permite interceptar declarações arbitrárias do núcleo do sistema operacional, mas permite instrumentar as chamadas ao sistema, aí incluídas aquelas relacionadas com o sistema de arquivos. A Tabela 5.1 descreve o formato do rastro coletado com ambas as ferramentas.

Tabela 5.1: Formato do rastro capturado.

Campo	Descrição
pid	A identificação do processo que chamou a função instrumentada.
tid	A identificação da *thread* que chamou a função instrumentada.
begin	A marcação de tempo para o instante que a função instrumentada foi chamada.
end	A marcação de tempo para o instante que a função instrumentada retornou.
function	O nome da função instrumentada.
args	Os argumentos da função instrumentada.
rvalue	O valor de retorno da função instrumentada.

Procedimento de medição

Os procedimentos de medição começam por limpar as caches de páginas, *dentries* e *inodes*. Para isso, primeiro liberamos as páginas sujas através da execução do comando `sync`. Em seguida, para limpar as caches de fato, manipulamos os arquivos de controle do subsistema de memória virtual presentes em `/proc/sys/vm/`. Mais especificamente, executamos o comando `` `echo 3 > /proc/sys/vm/dropcaches` ``. Isso é feito para assegurar que as alterações nas caches depois de uma medição de captura de rastros não afetem as medições subsequentes. Após a limpeza da cache, o sistema de arquivos alvo é instrumentado e a carga de trabalho que será capturada é iniciada. Quando as medições são realizadas com a

ferramenta `SystemTap`, a instrumentação do sistema de arquivos requer que um módulo do núcleo seja carregado. Tanto a criação quanto o carregamento do módulo são mediados pelo utilitário `stap` da ferramenta `SystemTap`. Quando a medição é feita com a ferramenta `strace`, a ferramenta de instrumentação e a carga de trabalho são iniciadas ao mesmo tempo (o programa gerador de carga é executado conjuntamente com o comando `strace`). Por fim, em ambos os casos, os dados gerados pela captura são armazenados. Para reduzir a possível interferência causada pelo armazenamento desses dados e pelo atendimento das requisições feitas ao sistema de arquivos, o rastro é armazenado em um sistema de arquivos diferente daquele que está sendo instrumentado, montado em um disco diferente.

5.1.3 Identificação das fontes de incerteza

Nesta seção, nós indicamos as possíveis fontes de incertezas dos métodos de captura de rastros. Nós consideramos 4 fontes de incertezas mostradas na Figura 5.1: i) a ferramenta de captura de rastros; ii) a carga de trabalho; iii) a carga de trabalho de fundo; e iv) o ambiente de testes.

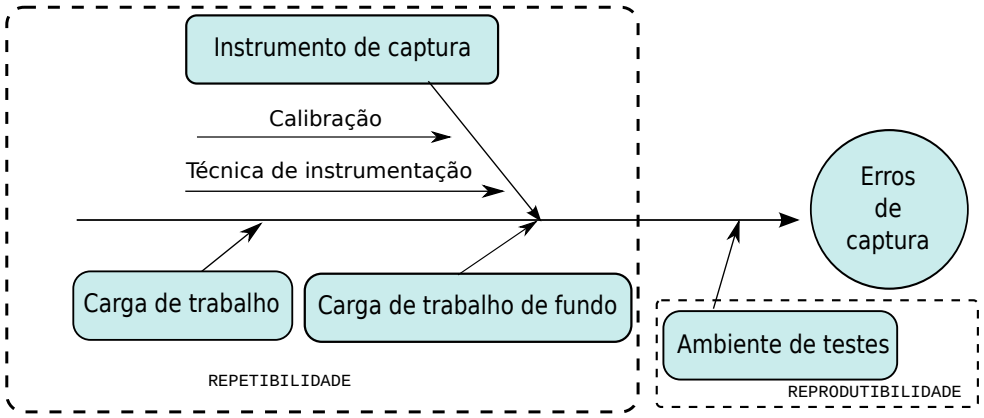


Figura 5.1: Fontes de incerteza que contribuem para erros na captura de rastros de utilização.

As ferramentas de captura afetam as medições por requererem um tempo adicional para executar a intercepção de chamadas. As ferramentas de captura afetam as medições ao impor tempo extra para interceptação das chamadas. O objetivo da interceptação é coletar a informação de contexto, como descrevemos na Seção 5.1.2. Para realizar a intercepção, as ferramentas de captura precisam acessar o espaço de endereçamento do processo alvo da captura. As ferramentas de captura que consideramos adotam técnicas diferentes e es-

sas técnicas impõem diferentes sobrecargas para acessar o espaço de endereçamento alvo. Como ilustra a Figura 5.2, a ferramenta `strace` executa chamadas ao sistema *ptrace* para cada interceptação realizada. Já a ferramenta `SystemTap` evita essas chamadas ao sistema, uma vez que a interceptação acontece em modo *kernel*.

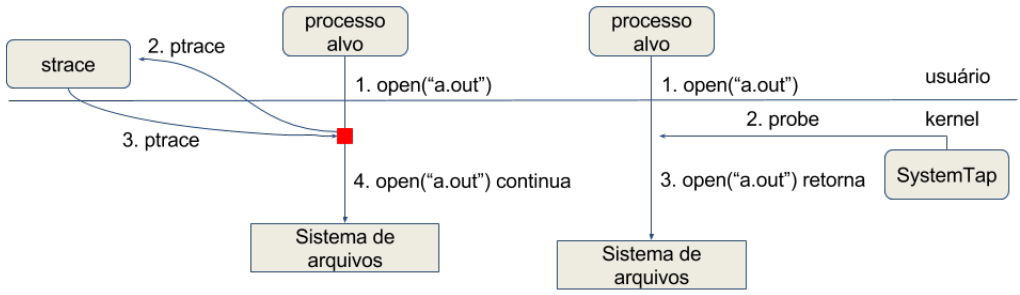


Figura 5.2: A ferramenta `strace` depende da chamada ao sistema *ptrace* para realizas as interceptações enquanto que as interceptações feitas com a ferramenta `SystemTap` acontecem diretamente no núcleo do sistema.

A interceptação de chamadas é uma fonte provável de erros sistemáticos. Como descrevemos na Seção 4.2, esses erros podem ser corrigidos por calibração. Embora esse processo os reduza, ele não é perfeito; por essa razão, a calibração em sí também é uma fonte de incertezas.

A carga de trabalho impacta o comportamento dos sistemas de arquivos, além de outros subsistemas do sistema operacional. Por exemplo, o descarregamento da cache de sistemas de arquivos é dirigido pela quantidade de páginas livres e usadas (ambos os fatores dependem das características da carga de trabalho) [22]. Uma vez que o processo das ferramenta de captura de rastros não está isolado do sistema de arquivos nem de outros processos em modo privilegiado — por exemplo, compartilham CPU e memória — a carga de trabalho pode também impactar os instrumentos de medição, desse modo, afetando os erros de medição.

Adicionalmente, pode ainda existir uma carga de fundo — criada por outros processos que não os clientes dos sistema de arquivos — que também consome os recursos disponíveis e, como consequência, pode afetar o desempenho medido do sistema de arquivos. Em muitos casos, a carga de fundo está fora do controle do procedimento de captura de rastros, e portanto, não pode ser diminuída.

Todas as fontes de incerteza acima podem atuar mesmo em condições de repetibilidade

— condições que incluem o mesmo procedimento de medição, os mesmos operadores, o mesmo sistema de medição, as mesmas condições de operação e o mesmo local, assim como medições repetidas no mesmo objeto ou em objetos similares durante um curto período de tempo [31].

Finalmente, a plataforma de execução, nela incluída o *hardware* e o *software*, também impacta a incerteza de medição. Por exemplo, o sistema de arquivos atua como uma camada de *software* entre as aplicações e o sistema de entrada/saída. Essa interação do sistema de arquivos com as camadas subjacentes define o desempenho do sistema de arquivos. A plataforma de execução é uma fonte de incerteza importante quando se analisa questões de reprodutibilidade, por exemplo, quando se compara medições feitas por laboratórios diferentes. Entretanto, uma vez que nesse trabalho nos concentramos somente nas incertezas que podem ser avaliadas em um único laboratório, nós consideramos somente a variação na versão do núcleo do sistema operacional: 2.6.32-41 e 3.13.0-24. A primeira versão foi escolhida por razão histórica — era ela que usávamos quando iniciamos a pesquisa que deu origem à escrita desse documento, já a segunda é a versão estável mais recente do núcleo Linux.

5.1.4 Caracterização da medição

Para caracterizar o procedimento de medição, os valores medidos são comparados com o valor verdadeiro do mensurando. Como descrito no Capítulo 4, uma vez que o valor verdadeiro do mensurando é desconhecido, valores de referência são adotados como uma aproximação.

Há duas maneiras populares para obter essas aproximações: i) realizar medições com um método de referência (um método com erros conhecidos e menores do que os erros do método sob avaliação); e ii) realizar medições com objetos de referência (objetos com propriedades definidas, por exemplo pesos-padrão mantidos por entidades de metrologia legal).

Em nossa prova de conceito, ilustrada na Figura 5.3, os valores de referência são obtidos usando um instrumento de medição de referência para medir um objeto de referência. Nós adotamos como objeto de referência um *micro-benchmark* que gera chamadas ao sistema relacionadas com o sistema de arquivos em um nível de carga específico. O nível de carga é dado pelo número de *threads* ativas no *micro-benchmark* que geram as chamadas ao sistema. Cada *thread* executa um número configurável de requisições continuamente e manipula um

arquivo diferente de $10GB$. Nós usamos arquivos que são ao menos 4 vezes maiores do que a quantidade de memória principal disponível; isso reduz a probabilidade de que requisições consecutivas para posições aleatórias de um arquivo sejam atendidas pela mesma página da memória em cache [65].

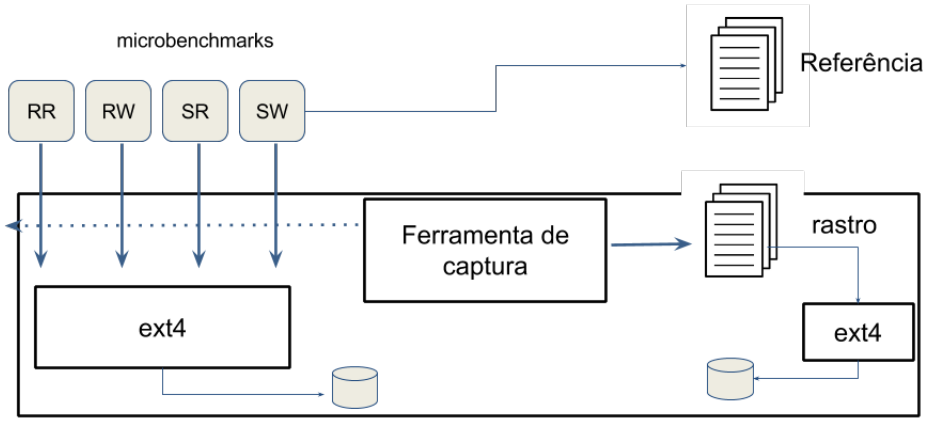


Figura 5.3: As ferramentas de captura interceptam cargas de trabalho geradas por micro-benchmarks. Os dados capturados são armazenados em um disco rígido diferente daquele envolvido com a carga de trabalho gerada. As medições são comparadas com medições de referência, colhidas sem a necessidade da ferramenta de captura.

Para gerar as cargas de trabalho, consideramos 4 níveis de carga e 4 tipos de carga. Os níveis de carga variam de 1 até 4. Os tipos de carga são definidos pela chamada ao sistema requisitada pelas *threads* de trabalho do *micro-benchmark*: i) *random read* (chamada ao sistema *pread*); ii) *random write* (chamada ao sistema *pwrite*); iii) *sequential read* (chamada ao sistema *read*); e iv) *sequential write* (chamada ao sistema *write*). Cada requisição lê ou escreve em blocos de 4096 bytes. Cada *thread* executa uma sequência de 5000 requisições sem espera entre elas.

O sistema de medição de referência simplesmente coleta as marcações de tempo para a execução das requisições no nível da aplicação (adicionamos chamadas para a função de coleta de tempo `clock_gettime` ao *micro-benchmark*), sem a necessidade de executar as instruções de interceptação presentes quando o sistema de arquivos é instrumentado. Essa abordagem fornece medições mais próximas do tempo de resposta do sistema de arquivos percebido pelas aplicações, portanto mais próximos do valor verdadeiro do mensurando do

que as medições realizada com as ferramentas de captura `SystemTap` e `strace`. Note que, na prática, não é factível coletar as marcações de tempo no nível da aplicação em experimentos de captura de rastros; uma vez que as aplicações podem não ser conhecidas de antemão, portanto não é possível modificá-las para adicionar o código de coleta de marcações de tempo.

Determinação da precisão da medição

Para contabilizar a precisão da medição, conduzimos 10 medições replicadas usando cada ferramenta de captura de rastros, em todas as combinações de nível de carga de trabalho e tipos de carga de trabalho. Consideramos 4 níveis de carga de trabalho, variando de 1 até 4. O resultado do experimento é a média do tempo de resposta para todas as operações executadas no experimento.

Analisamos a precisão dos métodos de captura, comparando-a com a precisão do método de referência. Em outras palavras, nosso objetivo foi verificar se as ferramentas `SystemTap` e `strace` aumentam a variabilidade dos resultados medidos em comparação ao método de referência. Para isso, aplicamos um teste *F* para comparar a precisão dos métodos de captura: para um método de medição A, seja $u(R_w^A)$ a precisão do método, ou seja, a componente da incerteza devida aos erros aleatórios. Segundo esse teste, há evidência que um método B é menos preciso do que um método A se $F_r > F_{\alpha,N_B-1,N_A-1}$, em que F_r é a razão entre as variâncias dos métodos de medição:

$$F_r = \frac{u(R_w^B)^2}{u(R_w^A)^2}$$

e F_{α,N_B-1,N_A-1} é o valor crítico da distribuição F com $N_B - 1$ e $N_A - 1$ graus de liberdade ao nível de significância α [12].

Determinação da tendência da medição

A tendência de medição é a diferença entre os valores medidos e os valores de referência. Para analisar a tendência de medição, comparamos as medições de captura e os valores de referência obtidos nos experimentos conduzidos para determinar a precisão da medição, descritos na seção anterior, para as mesmas combinações de nível da carga de trabalho e tipo da carga.

Decidir se a tendência de medição é aceitável ou não é uma questão qualitativa específica do propósito pretendido para o método de medição. De qualquer maneira, existe uma regra prática que pode servir de guia, independente de propósito, para decidir se a tendência é significativa, e portanto, os valores medidos devem ser corrigidos por calibração. De acordo com essa regra, a tendência é significativa caso a sua magnitude seja maior que o dobro da incerteza [37]:

$$|\bar{x} - ref| > 2 \cdot \sqrt{\frac{u(R_w)^2}{n} + u(C_{ref})^2}$$

em que $\bar{x}$ é a média dos valores medidos, ref é o valor de referência adotado, $u(R_w)$ é a componente de precisão descrita no Capítulo 4, n é o número de experimentos replicados conduzidos e $u(C_{ref})$ é a incerteza dos valores de referência, também descrita no Capítulo 4.

5.1.5 Calibração

Tanto quanto sabemos, não há um estudo anterior que descreva os padrões de erros esperados para as medições de captura de rastros. Por essa razão, não podemos adotar antecipadamente métodos mais simples de calibração, tais como os métodos de `ponto único` e `dois pontos`; no lugar destes adotamos o método de calibração de `múltiplos pontos` por regressão linear. Felizmente, em nosso cenário, preparar múltiplos objetos de referência não é um problema: a única desvantagem é o tempo extra necessário para executar um número maior de experimentos.

Para construir a curva de calibração, aplicamos a mesma metodologia descrita na Seção 5.1.4, que foi usada para aferir as características dos métodos de captura. Executamos 10 experimentos replicados, dessa vez considerando uma faixa mais larga para o nível da carga de trabalho, variando de 1 até 8. Para quantificar os erros do procedimento de calibração, usamos o método descrito na Seção 4.2, baseado nos intervalos de predição da regressão linear.

5.1.6 Determinação da incerteza de medição

Nesta prova de conceito, consideramos o cenário metrológico da validação de métodos de captura com respeito aos erros possíveis de serem isolados em um único laboratório. Assim,

a incerteza dos métodos de captura é calculada conforme a incerteza combinada, u_c, dada pela seguinte equação [29]:

$$u_c = \sqrt{u(R_w)^2 + u(bias)^2}$$

Como descrevemos na Seção 4.1, a componente $u(R_w)$ contabiliza a incerteza em virtude dos erros aleatórios, enquanto a componente $u(bias)$ contabiliza a incerteza devida aos erros sistemáticos. Para calcular a incerteza combinada, considera-se que os valores de medição já foram corrigidos pelo procedimento de calibração.

5.2 Resultados

Nesta seção, discutimos os resultados da validação dos métodos de captura de rastros. Na Seção 5.2.1 nós mostramos os resultados da caracterização dos métodos de captura de rastros. Na Seção 5.2.2 reportamos os resultados do procedimento de calibração. Finalmente, na Seção 5.2.4, apresentamos a incerteza combinada dos métodos de captura de rastros. Verificamos por testes de shapiro que tantos as medições de referência quanto as medições com as ferramentas `strace` e `SystemTap` são distribuídas normalmente. No apêndice D, mostramos os testes de normalidade complementares que suportam as análises de precisão, tendência e incerteza.

5.2.1 Caracterização da medição

Nesta caracterização, nós descrevemos os erros aleatórios e sistemáticos observados para as medições de captura, bem como os valores de precisão e de tendência associados. Nossa discussão começa com a análise das fontes de erros devidas ao tipo de carga de trabalho e ao nível da carga de trabalho — fatores relacionados com o instrumento de medição e o objeto medido. Em seguida, analisamos fatores extrínsecos: a influência da carga de fundo e o do ambiente de teste.

A Figura 5.4 mostra as componentes dos erros sistemáticos e aleatórios, $u(bias)$ e $u(R_w)$ respectivamente, tal como definidas no Capítulo 4, para as ferramentas `strace` e `SystemTap`, para todas as combinações de tipos e níveis de carga. Os valores dos erros aleatórios e sistemáticos foram encontramos com base em 50 experimentos replicados.

As componentes destes erros são mostradas como percentuais relativos aos valores de referência.

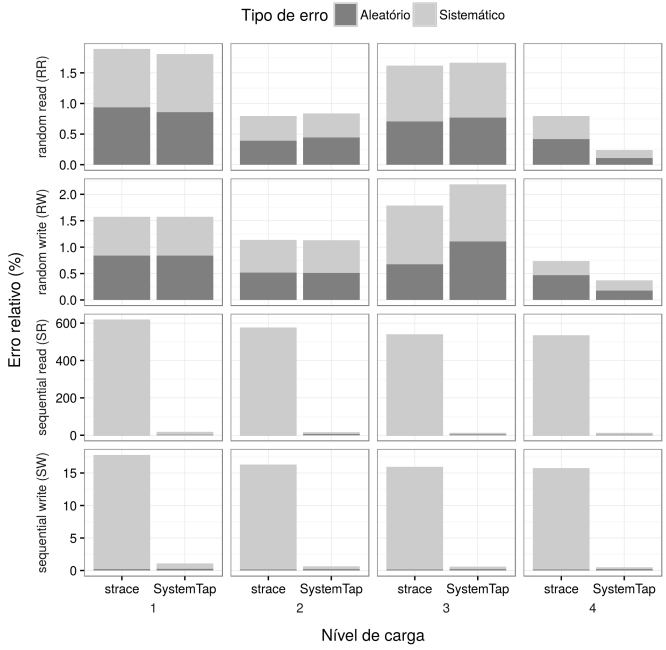


Figura 5.4: Componentes dos erros aleatórios e sistemáticos, $u(R_w)$ e $u(bias)$, respectivamente, das medições de captura de rastros com as ferramentas `SystemTap` e `strace`. As componentes de erros são mostradas como percentuais relativos aos valores de referência.

Estes resultados indicam que, para as cargas de trabalho sequenciais, os erros sistemáticos são maiores do que os erros aleatórios. Como consequência, a incerteza de medição pode ser bastante reduzida pela compensação dos erros sistemáticos realizada pelo procedimento de calibração, como discutiremos na Seção 5.2.2. Por sua vez, para as cargas de trabalho aleatórias, os erros aleatórios são da mesma grandeza que os erros sistemáticos. Esses cenários representam um desafio adicional para a redução da incerteza de medição, uma vez que não é possível aplicar o procedimento de calibração para reduzir erros aleatórios.

Na Tabela 5.2, nós comparamos a precisão das medições feitas pelas ferramentas `SystemTap` e `strace` com a precisão das medições feitas com o método de referência. A Tabela 5.2 mostra a razão F_r, como definida na Seção 5.1.4, com nível de significância de 0.05 e graus de liberdade $N_B - 1$ e $N_A - 1$ iguais a 49 (com base em 50 experimentos replicados). O valores calculados para $F_{1-\alpha/2,N_B-1,N_A-1}$ e $F_{\alpha/2,N_B-1,N_A-1}$ foram de 1.76 e 0.56,

respectivamente. Em todos os cenários, a precisão dos métodos de captura é equivalente à precisão do método de referência, ou seja, a razão F_r fica entre os limites críticos. Assim, podemos concluir que a escolha da ferramenta de captura em sí, não determina a precisão dos valores medidos.

Tabela 5.2: Comparação entre a precisão dos métodos de captura baseados nas ferramentas `SystemTap` e `strace` e a precisão do método de referência. De acordo com um teste *F* com nível de significância de 0.05, a precisão do método baseado na ferramenta `SystemTap` e do método baseado na ferramenta `strace` é equivalente à precisão do método de referência.

	1	2	3	4
Random read (RR)				
`SystemTap`	0.80	1.49	0.59	0.64
`strace`	1.06	1.18	0.57	1.05
Random write (RW)				
`SystemTap`	1.32	0.69	1.00	0.88
`strace`	1.32	0.70	0.57	0.59
Sequential read (SR)				
`SystemTap`	0.71	0.58	0.64	0.62
`strace`	1.62	0.89	1.14	1.02
Sequential write (SW)				
`SystemTap`	1.20	0.81	0.94	1.06
`strace`	0.97	0.65	0.98	0.88

Os resultados mostrados na Figura 5.4 também mostram que o erro total, somado os erros aleatórios e sistemáticos, para as medições com a ferramenta `strace` é maior do que o erro total para as medições com a ferramenta `SystemTap`. Finalmente, os erros para a captura das cargas de trabalho *sequential read* e *sequential write* são maiores do que os erros para a captura das cargas de trabalho *random read* e *random write*. Esses resultados confirmam o esperado: a ferramenta `strace`, baseada na chamada ao sistema `ptrace`, introduz um erro sistemático maior do que o erro introduzido pela ferramenta `SystemTap`, baseada em instrumentação do núcleo do sistema, como mostrado para os resultados das medições das

cargas de trabalho *sequential write* e *sequential read*. Para as cargas de trabalho de acesso aleatório, a magnitude do erro sistemático (menor do que 100 microssegundos) é marginal quando comparada com os valores medidos (cerca de 10.000 microssegundos). Entretanto, com base no critério definido na Seção 5.1.4, o erro sistemático para as medições baseadas na ferramenta `strace` é significativo para todos os cenários das cargas de trabalho *sequential read* e *sequential write*.

Os erros de medição não dependem apenas das ferramentas de captura e das características das cargas de trabalhos capturadas, mas também da interação com o ambiente experimental. Nessa prova de conceito, nós investigamos dois exemplos desta interação: a seletividade quanto à carga de fundo e a interação com o sistema operacional.

Avaliamos a seletividade do método de medição com respeito à utilização da CPU. Para esse fim, nós conduzimos outra medições de captura de rastros tomando como base o mesmo projeto experimental que usamos para analisar os erros aleatórios e sistemáticos. Diferente do que fizemos antes, agora controlamos a utilização da CPU executando um número configurável de processos intensivos em CPU durante a captura dos rastros. Consideramos dois níveis para a utilização da CPU, 0 e 1, com base no número de processos intensivos em CPU.

A Figura 5.5 mostra os valores medidos para a captura das cargas de trabalho *sequential read* e *sequential write* com os respectivos erros associados, usando as ferramentas `SystemTap` e `strace`, enquanto submetidas à carga de fundo. Nós observamos que a captura de rastros feita com a ferramenta `strace` é afetada pela carga de trabalho de fundo — os valores medidos **diminuem** ao passo que **aumentamos** o nível da carga de fundo.

Essa redução ocorre como um efeito colateral do mecanismo de gerenciamento de potência habilitado na máquina em que conduzimos os experimentos. Esse mecanismo, bastante comum atualmente, controla a frequência da CPU de acordo com a variação de sua utilização provocada pelos processos intensivos em CPU. Como consequência, as requisições executam mais rápido, apesar de terem de competir pela CPU com outros processos [47].

A Figura 5.6 mostra o impacto do sistema operacional nas medições de captura. Ela mostra a média dos valores medidos para todas as cargas de trabalho, usando ambas as ferramentas de captura `SystemTap` e `strace`. Mostramos os resultados das medições para duas configurações do sistema operacional, baseadas em duas versões do núcleo Linux:

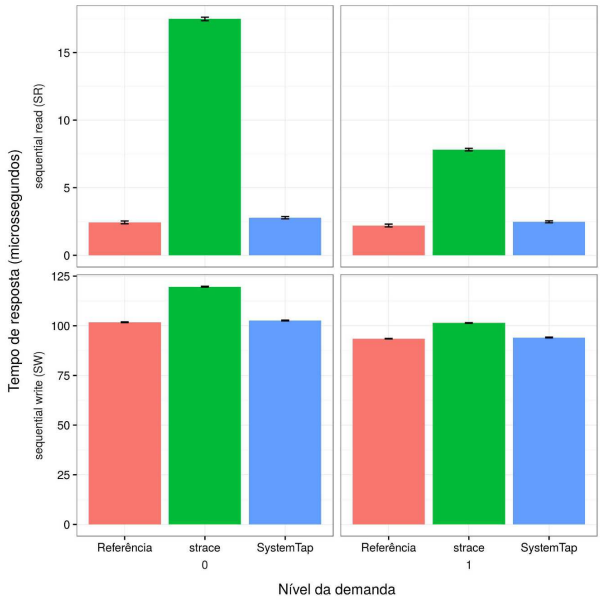


Figura 5.5: O método de captura baseado na ferramenta `strace` não é seletivo em relação à carga de fundo. Os valores medidos diminuem ao passo que aumentamos o nível da carga de fundo.

3.13.0-24 e 2.6.32-41. Nós observamos que as medições feitas com o método de referência, bem como as medições realizadas com as ferramentas `strace` e `SystemTap`, são afetadas pelo núcleo do sistema operacional — para todos os métodos, os valores medidos são maiores quando se emprega o núcleo 3.13.0-24 do que as os valores medidos com o núcleo 2.6.32-41.

Essa caracterização indica que, em alguns cenários, os erros sistemáticos são muito altos para serem ignorados. A seguir, na Seção 5.2.2, nós discutimos como calibrar as medições de captura para compensar os erros sistemáticos. Consideramos a calibração para os erros causados pelos instrumentos de medição e pelas cargas de trabalho capturadas. A carga de trabalho de fundo e a configuração do ambiente de teste, como acabamos de ver, são fatores que costumam ser ignorados na metodologia corrente. Na Seção 5.2.3, nós discutimos o efeitos da falta de informação sobre essas fontes de erro sobre o processo de calibração.

5.2.2 Calibração

Concentramos nossa análise de calibração nas medições realizadas com a ferramenta `strace`, para as carga de trabalho *sequential read* e *sequential write*, uma vez que estas

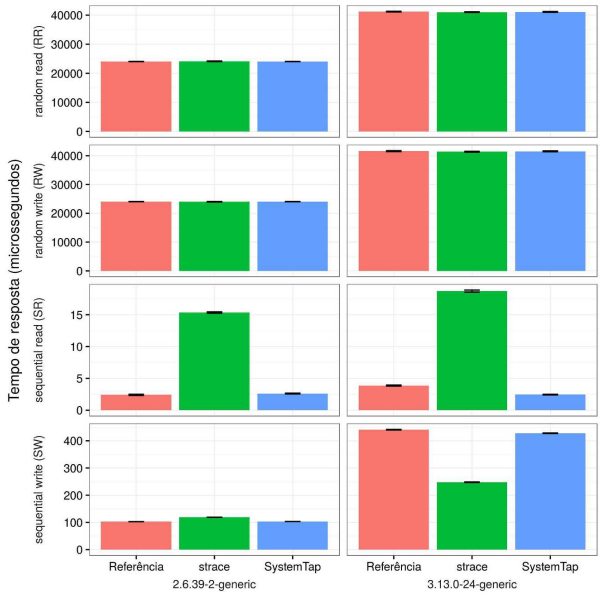


Figura 5.6: As medições de captura de rastros são afetadas pelo núcleo do sistema operacional usado no ambiente de testes. O tempo de resposta medido nas capturas realizadas com a versão 3.13.0-24 do núcleo é maior do que o tempo de resposta medido nas capturas com a versão 2.6.32-41.

apresentaram erro sistemático significativo. Para construir as curvas de calibração, aplicamos a metodologia descrita na Seção 5.1.5.

A Figura 5.7 mostra os resultados da calibração baseados nas curvas de calibração encontradas para as medições realizadas com a ferramenta `strace`. As barras indicam os valores medidos e os valores calibrados para cada nível de carga, bem como os valores de referência correspondentes. Como explicamos na Seção 5.1.5, os valores de referência são dados pela média dos resultados dos experimentos replicados conduzidos com o método de medição de referência. Os valores medidos para a ferramenta `strace` são dados pela média de 50 experimentos replicados. Os valores calibrados são obtidos aplicando as curvas de calibração aos resultados medidos para corrigir os erros sistemáticos. A Figura 5.7 também mostra os intervalos de confiança $(\hat{x}^-, \hat{x}^+)$ calculados de acordo com a Equação 4.1. Nós adotamos um nível de confiança de 95% para definir todos os intervalos de confiança.

O processo de calibração é capaz de compensar os erros sistemáticos das medições realizadas com a ferramenta `strace`; considerando tanto a carga de trabalho *sequential read* quanto *sequential write*, no nível de confiança considerado (95%), não há diferença entre os

valores corrigidos e os valores de referência.

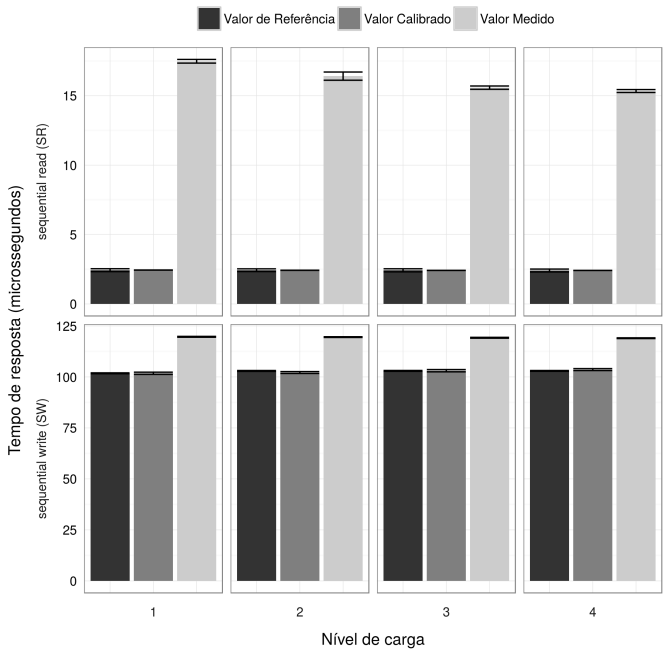


Figura 5.7: Calibração das medições feitas pela ferramenta `strace`. As barras apontam o tempo de resposta médio para as medições de captura feitas com a ferramenta `strace`, o tempo de resposta médio para as medições realizadas com o método de referência e o tempo de resposta calibrado (com os intervalos de confiança de calibração associados). O procedimento de calibração é efetivo na correção dos erros sistemáticos: os valores calibrados se aproximam dos valores de referência.

5.2.3 Robustez da calibração

Como observamos antes, há múltiplos fatores que contribuem para os erros de medição. Alguns deles, por exemplo, o tipo da carga de trabalho, costumam ser considerados durante o procedimento de captura de rastros. Entretanto, há outras fontes de erros, tais como a carga de fundo, que não são considerados na literatura corrente.

Para ressaltar a importância da coleta dessa informação adicional, nós aplicamos as curvas de calibração construídas para a análise da Seção 5.2.2, para corrigir outras medições, afetadas por essas outras fontes de erro.

Quando a calibração é aplicada, espera-se que as medições tenham sido afetadas pelas mesmas fontes de erro que tenham sido consideradas para criar as curvas de calibração.

Quando não é caso, a compensação pode ser insuficiente para prover um resultado de qualidade, ou ainda pior, a compensação pode aumentar os erros de medição.

Nesta análise, começamos por apresentar os resultados da calibração para compensar as medições de carga de trabalho *sequential write* realizadas com a ferramenta `strace` quando sujeitas a uma carga de fundo. Nós consideramos a correção dessas medições tanto com as curvas de calibração discutidas na seção anterior, criadas com base em medições que não estavam sujeitas à carga de fundo, quanto com curva de calibração que levaram em conta medições afetadas pela carga de fundo. Em todos os resultados apresentados nessa seção, os valores típicos das medições são dados pela média de 50 experimentos replicados. Os intervalos de confiança $(\hat{x}^-, \hat{x}^+)$ para a calibração são calculados conforme definido na Seção 4.2.1.

A Figura 5.8 mostra os valores medidos, os valores corrigidos, bem como os valores de referência, para níveis de carga de 1 até 4. Pode ser visto que a curva de calibração baseada nas medições sem a influência da carga de fundo não é capaz de compensar as medições feitas sob a influência dessa fonte de erros. O valor de referência para essa carga de trabalho é de cerca de 90 microssegundos. Os valores medidos com a ferramenta `strace` atingem cerca de 100 microssegundos. A calibração aumenta os valores, ao invés de diminuí-los — os valores corrigidos são de cerca de 150 microssegundos. Por sua vez, as correções feitas com base nas medições feitas sob influência da carga de fundo são capazes de compensar os erros de medição.

Em seguida, a Figura 5.9 apresenta os resultados da calibração das capturas realizadas com a ferramenta `strace` para a carga de trabalho *sequential read*, quando sujeitas às mudanças no sistema operacional usado no ambiente de testes. Nessa análise, nós usamos curvas de calibração criadas com base em medições realizadas em dois ambientes de testes diferentes: o primeiro baseado na versão $2.6.32$-41 do núcleo Linux enquanto o segundo baseado na versão $3.13.0$-24 do núcleo. Usamos essas duas curvas para calibrar as medições realizadas com base na versão $3.13.0$-24 do núcleo Linux. A Figura 5.9 mostra os valores medidos, os valores calibrados com as duas curvas de calibração, bem como os valores de referência. Novamente, os resultados mostram que não há maiores problemas quando a calibração é aplicada para corrigir uma medição feita com o mesmo ambiente de testes que foi usado para construir as curvas de calibração. Entretanto, quando se aplica a curva

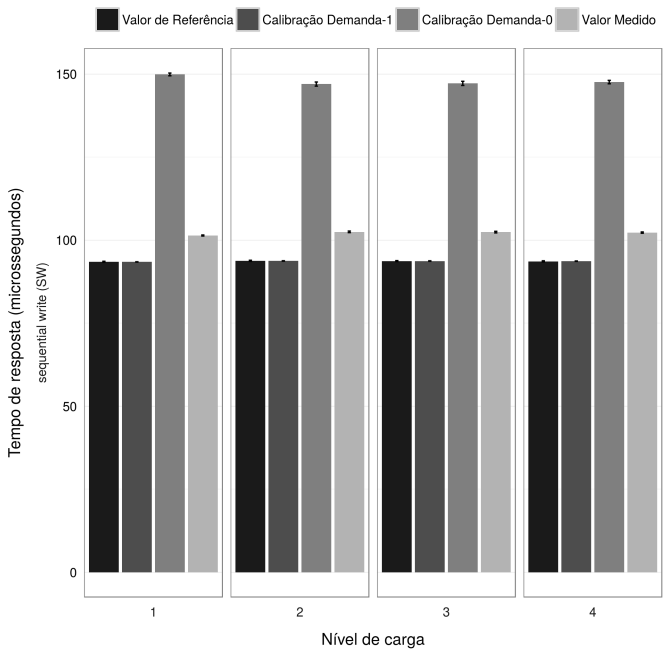


Figura 5.8: O procedimento de calibração definido sem levar em conta a carga de trabalho de fundo não é capaz de corrigir as medições feitas com a ferramenta `strace` sob a influência da carga de fundo.

de calibração criada com base nas medições feitas com o núcleo $2.6.32\text{-}41$ para corrigir as medições realizadas com o núcleo $3.13.0\text{-}24$, a correção não é efetiva; a correção sobrecompensa os erros de medição.

5.2.4 Determinação da incerteza da medição

Após a correção por calibração, nós determinamos a incerteza combinada, u_c, para as medições de captura de rastros. A Tabela 5.3 mostra a incerteza da medição para as ferramentas `SystemTap` e `strace`, para todas as combinações de carga de trabalho e tipo de carga. Para os cenários nos quais a tendência foi significativamente alta, e por isso, demandaram correção por calibração, a tabela também mostra (entre parênteses) a incerteza antes do processo de correção. A incerteza combinada, como descrevemos no Capítulo 4, é reportada como $y_m \pm u_c$, em que y_m é a média dos valores medidos em medições replicadas. Os resultados mostrados na Tabela 5.3 consideram um nível de confiança de 95.5%.

A incerteza da medição para as cargas de trabalho de acesso aleatório é menor do que a incerteza para as medições das cargas de trabalho sequenciais: não passa de 2.8% do valor

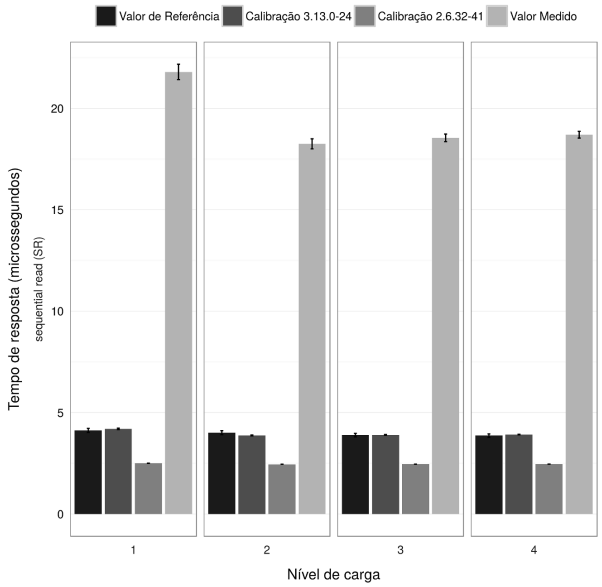


Figura 5.9: O procedimento de calibração para medições realizadas com a ferramenta `strace` é específico para a versão o núcleo do sistema operacional usado no ambiente de testes. O procedimento de calibração definido com base nas medições feitas com o núcleo 2.6.32-41 não é capaz de corrigir as medições feitas com o núcleo 3.13.0-24.

esperado para as medições conduzidas com a ferramenta `SystemTap` e de 2.7% para as medições feitas com a ferramenta `strace`. Por sua vez, a incerteza de medição para as cargas de trabalho sequenciais atinge até 30.5% para as medições realizadas com a ferramenta `SystemTap` e 1237% para a ferramenta `strace`.

Depois da calibração, a incerteza de medição para a carga de trabalho *sequential read* é reduzida e não passa de 9.2% para a ferramenta `strace` e de 9.3% para a ferramenta `SystemTap`. A incerteza da medição para a carga de trabalho *sequential write* também é alta para a ferramenta `strace`, embora seja menor do que a incerteza da carga de trabalho *sequential read*: antes da calibração, a incerteza é de até 35.2% para a ferramenta `strace`. Após a calibração, a incerteza para a ferramenta `strace` para a carga de trabalho *sequential write* não ultrapassa 1.8%.

Tabela 5.3: Incerteza de medição combinada u_c com nível de confiança de 95.5%.

Nível de carga	SystemTap		strace	
	Random read (RR)			
1	$7744.9 \pm 2.5\%$		$7742.4 \pm 2.7\%$	
2	$17945.1 \pm 1.2\%$		$17941.1 \pm 1.1\%$	
3	$23339.1 \pm 2.3\%$		$23377.3 \pm 2.3\%$	
4	$24054.4 \pm 0.3\%$		$24136.6 \pm 1.1\%$	
	Random write (RW)			
1	$8323.6 \pm 2.2\%$		$8325.3 \pm 2.2\%$	
2	$17953.3 \pm 1.6\%$		$17966.8 \pm 1.6\%$	
3	$23223.2 \pm 2.8\%$		$23284.6 \pm 2.5\%$	
4	$24060.7 \pm 0.5\%$		$24014.3 \pm 1.1\%$	
	Sequential read (SR)			
1	$2.4 \pm 8.7\%$	$(2.8 \pm 30.5\%)$	$2.4 \pm 8.6\%$	$(17.5 \pm 1237.0\%)$
2	$2.4 \pm 8.3\%$	$(2.7 \pm 21.3\%)$	$2.4 \pm 8.2\%$	$(16.4 \pm 1150.3\%)$
3	$2.4 \pm 9.3\%$	$(2.6 \pm 20.4\%)$	$2.4 \pm 9.2\%$	$(15.6 \pm 1086.9\%)$
4	$2.4 \pm 8.7\%$	$(2.6 \pm 20.0\%)$	$2.4 \pm 8.7\%$	$(15.3 \pm 1073.8\%)$
	Sequential write (SW)			
1	$102.7 \pm 1.8\%$		$101.8 \pm 1.1\%$	$(119.7 \pm 35.2\%)$
2	$103.3 \pm 1.0\%$		$102.2 \pm 1.8\%$	$(119.5 \pm 32.3\%)$
3	$103.3 \pm 0.9\%$		$103.1 \pm 1.2\%$	$(119.2 \pm 31.5\%)$
4	$103.2 \pm 0.8\%$		$103.6 \pm 1.6\%$	$(119.0 \pm 31.1\%)$

5.3 Discussão

A validação descrita nas seções anteriores serve a três propósitos: indicar como reduzir a incerteza do método de medição; embasar a escolha de uma ferramenta de captura; e ajudar a manter experimentos de captura de longa duração sob controle.

Considerando o propósito de redução da incerteza de medição, a prescrição baseada em nossos resultados é simples: não se deve utilizar rastros de captura de carga de trabalho sequenciais sem a correção adequada dos erros sistemáticos, caso contrário, a incerteza da medição será significantemente alta. Além disso, nós mostramos que as ferramentas de captura de rastros não são seletivas. Por essa razão, é importante manter o ambiente de captura tão isolado quanto possível. Caso não seja possível, é importante registrar a carga de trabalho de fundo, do contrário, o procedimento de calibração não pode ser aplicado de maneira adequada. Também é importante manter um inventário das bibliotecas e programas instalados no ambiente de experimentação, uma vez que o desempenho do sistema de arquivos pode depender destes.

Decidir que método de medição usar não envolve simplesmente escolher o método que apresenta menor incerteza, uma vez que geralmente há alguma consideração de custo envolvida (por exemplo, comprar instrumentos de medição mais capazes ou empregar pessoal mais capacitado etc). Por exemplo, as medições baseadas na ferramenta `SystemTap` são mais complicadas de conduzir: é preciso empregar um sistema operacional no qual a compilação preserve os símbolos de depuração (ou compilar o sistema, caso os símbolos de depuração não estejam disponíveis). Além disso, o desenvolvimento de módulos do núcleo também não é uma tarefa simples, tendo em vista que defeitos nestes podem provocar o travamento do sistema operacional. Em nossa prova de conceito descobrimos que, caso a carga de trabalho seja composta em sua maioria por chamadas de acesso aleatório, ambas as ferramentas de captura apresenta alta precisão e baixa tendência de medição; dessa maneira, para esse cenário, a ferramenta de captura `strace` é a melhor escolha, haja vista que é mais fácil de usar do que a ferramenta `SystemTap`. Quando se considera cargas de trabalho de escrita sequencial, a ferramenta `strace` mais uma vez é a melhor escolha, contanto que um procedimento de correção por calibração seja aplicado. Por outro lado, caso se considere carga de trabalho de leitura sequencial, ambas as ferramentas apresentam alta incerteza e

pode ser o caso que nenhuma delas seja adequada. De qualquer modo, essa é uma decisão claramente dependente dos requisitos associados com a aplicação dos dados capturados.

Finalmente, o terceiro propósito da prova de conceito é ajudar a manter experimentos de captura de rastros sob controle [24]. Isso serve como um controle de qualidade, particularmente útil quando se quer coletar rastros de utilização de longa duração. Por exemplo, pode-se executar o protocolo de validação periodicamente, digamos semanalmente ao longo do período em que o rastro está sendo coletado, para investigar se os resultados da medição estão dentro dos limites esperados de precisão e tendência calculados durante o procedimento de validação. No decorrer desses longos períodos de experimentação, atualizações dos sistema operacionais são aplicadas, parte do maquinário pode falhar ou desviar-se do modo de funcionamento normal, fatores que podem mudar os resultados da medição. Esses fatores podem ser identificados através de execução periódica do protocolo de validação proposto neste trabalho.

Os três propósitos listados são embasados na caracterização dos métodos de medição, aí incluída a incerteza combinada. No apêndice B, compilamos em um catálogo os principais resultados dessa caracterização; assim podem ser consultados de modo independente da prova de conceito.

Capítulo 6

Reprodução de rastros de utilização

Neste capítulo, nós consideramos a validação de métodos de reprodução de rastros. Esses métodos se diferenciam pela arquitetura da ferramenta de reprodução empregada: baseada em compilação ou baseada em eventos. Adotamos a mesma organização apresentada no capítulo anterior em conformidade com o protocolo de validação discutido no Capítulo 4: na Seção 6.1 descrevemos a aplicação do protocolo de validação no cenário de reprodução de rastros. Na Seção 6.2 discutimos os resultados da caracterização e análise da incerteza dos métodos de reprodução. Por fim, assim como fizemos no capítulo anterior, terminaremos esta prova de conceito com um conjunto de recomendações para aplicação dos métodos de reprodução considerados.

6.1 Método

Nesta seção, nós apresentamos a metodologia para teste dos métodos de reprodução de rastros. Dedicamos uma subseção para cada uma das etapas do protocolo.

6.1.1 Especificação do Mensurando

O mensurando que adotamos em nossos procedimentos de reprodução de rastros é o mesmo que adotamos na prova de conceito de captura: o tempo de resposta do sistema de arquivos. Esse mensurando é considerado como uma sequência de eventos T, relacionados com a execução das funções do sistema de arquivos. Cada evento $t_k \in T$ é gerado por uma

requisição feita pelo reprodutor de rastros ao sistema de arquivos. Desse modo, o evento t_k é uma tupla $\langle f_k, b_k, e_k, c_k \rangle$ que representa a reprodução de uma função f_k do conjunto de funções F do sistema de arquivos, b_k e e_k são as marcações de tempo imediatamente antes e depois da reprodução da função e c_k é a informação contextual relacionada, incluindo argumentos e valores de retorno das funções reproduzidas.

De maneira análoga a prova de conceito anterior, para cada evento $t_k = \langle f_k, b_k, e_k, c_k \rangle$ da sequência de eventos T, o tempo de resposta do evento t_k é dado pelo intervalo de tempo que decorre entre as marcações de tempo, ou seja, $e_k - b_k$.

6.1.2 Procedimento de medição

Nesta seção nós descrevemos o ambiente experimental em que os procedimentos de reprodução de rastros foram conduzidos, bem como os reprodutores baseados em eventos e em compilação e por fim os procedimentos de medição que adotamos para usar os reprodutores.

Ambiente experimental

Como será justificado na Seção 6.1.4, em nossa validação dos procedimentos de reprodução empregamos como valores de referência as medições realizadas na prova de conceito de captura de rastros. Para que as medições de reprodução possam ser comparadas de maneira coerente com os valores de referência, empregamos o mesmo ambiente experimental utilizado na prova de conceito descrita no capítulo anterior. Assim, usamos uma estação de trabalho Intel E6550 Core 2 Duo 2.33GHz, com $2GB$ GB de memória principal. Essa estação tem dois discos rígidos: um disco SATA de 7200 RPM com 8 MB de cache, e um disco SATA de 5400 com 32 MB de cache. O reprodutor gerou a carga de trabalho para um sistema de arquivos `ext4` montado no segundo disco rígido. A máquina foi usada de maneira exclusiva durante a execução das medições e tinha o sistema operacional Linux na versão 2.6.32-41 instalado.

Instrumento de medição

Nós consideramos dois reprodutores de rastros, cada um representante de um modelo arquitetural diferente. Como representante da arquitetura baseada em compilação, adotamos o reprodutor `ARTC` [66]. Em nossos experimentos, empregamos o reprodutor `ARTC` sem modificação. Como representante da arquitetura baseada em eventos, desenvolvemos um novo reprodutor tomando como base o projeto do reprodutor `TBBT`, proposto por Zhu et al. [50]; tivemos que implementar uma nova ferramenta porque, até onde sabemos, não há reprodutor baseado em eventos disponível publicamente.

Como discutido anteriormente, o funcionamento dos reprodutores é ajustado por dois controles importantes: as políticas de ordenação e temporização. A política de ordenação define a relação de dependência que pode existir entre duas requisições quaisquer, e portanto, define a ordem em que essas requisições podem ser reproduzidas. Por sua vez, a política de temporização define os instantes de tempo nos quais as requisições devem ser reproduzidas.

Os reprodutores `ARTC` e `TBBT` definem dois grupos de políticas de ordenação equivalentes. O primeiro grupo (composto pela política *ROOT*, no reprodutor `ARTC`, e pela política *FS dependency*, no reprodutor `TBBT`) leva em consideração a semântica das operações do sistema de arquivos para definir a ordem entre as requisições. Por exemplo, uma requisição de escrita para um arquivo não pode ser reproduzida antes da requisição que cria o arquivo. O segundo grupo (composto pela política *temporal*, no reprodutor `ARTC`, e pela política *conservative* no reprodutor `TBBT`) considera a ordem encontrada no rastro capturado (com base nas marcações de tempo do rastro), ou seja, uma requisição será reproduzida somente depois que as requisições anteriores tenham sido.

Uma vez que as políticas de ordenação dos reprodutores `ARTC` e `TBBT` são equivalentes, para facilitar o trabalho do leitor, no que segue, nos referiremos às políticas do primeiro grupo como *FS* e às políticas dos segundo grupo como *temporal*.

Os reprodutores `ARTC` e `TBBT` também adotam políticas de temporização semelhantes. Ambos os reprodutores implementam a política de temporização *fullspeed*. Nessa política, qualquer possível intervalo entre as requisições é ignorado e elas são reproduzidas o mais rápido possível, desde que seja respeitada a política de ordenação. Além da política *fullspeed*, o reprodutor *TBBT* implementa a política de temporização *timestamp*. De acordo com essa política, as requisições são reproduzidas o mais próximo possível das marcações

de tempo originais, conforme o rastro, novamente, desde que seja respeitada a política de ordenação.

A carga de trabalho para os reprodutores baseados em compilação e em eventos foram geradas com base nos mesmos rastros. A Tabela 6.1 descreve o formato dos rastros usados.

Tabela 6.1: Formato do rastro.

Campo	Descrição
pid	A identificação do processo que chamou a função do sistema de arquivos.
tid	A identificação da *thread* que chamou a função do sistema de arquivos.
begin	A marcação de tempo em que a função do sistema de arquivos foi chamada.
end	A marcação de tempo em que a função do sistema de arquivos retornou.
function	O nome da função do sistema de arquivos que foi chamada.
args	Os argumentos da função do sistema de arquivos que foi chamada.
rvalue	O valor de retorno da função do sistema de arquivos que foi chamada.

O reprodutor `ARTC`, enquanto alternativa baseada em compilação, toma o rastro como entrada e gera código fonte que representa a carga de trabalho capturada. O código fonte gerado cria uma *thread* para cada *thread* encontrada no rastro. No código fonte estão definidos estaticamente os dados associados com cada requisição, por exemplo, os argumentos para as funções do sistema de arquivos que serão reproduzidas. No código fonte também está definida a sequência de requisições que cada *thread* reproduzirá — novamente, tal como encontrado no rastro — e a relação de dependência especificada pela política de ordenação escolhida. Para implementar essa relação de ordem, a estrutura de dados que representa uma requisição no código fonte gerado pelo `ARTC` contém a identificação das requisições dependentes. Durante a reprodução do rastro, a *thread* responsável pela execução de uma requisição precisa antes verificar se as requisições predecessoras já foram executadas, bloqueando em uma variável condicional quando não for o caso.

Nossa implementação, ilustrada na Figura 6.1, baseada em eventos segue o projeto do reprodutor `TBBT` [50]. Ela evita os passos de geração do código fonte e compilação. O reprodutor toma como entrada o rastro capturado e o mapeia para um formato de entrada mais estruturado. Nessa estrutura é declarada a relação de ordem entre as requisições. O

reprodutor processa este arquivo formatado e o traduz em chamadas ao sistema para serem reproduzidas. Essas operações são realizadas por dois grupos de *threads*: coordenador e trabalhadores. A *thread* de coordenação gera os eventos que são reproduzidos pelas *threads* de trabalho.

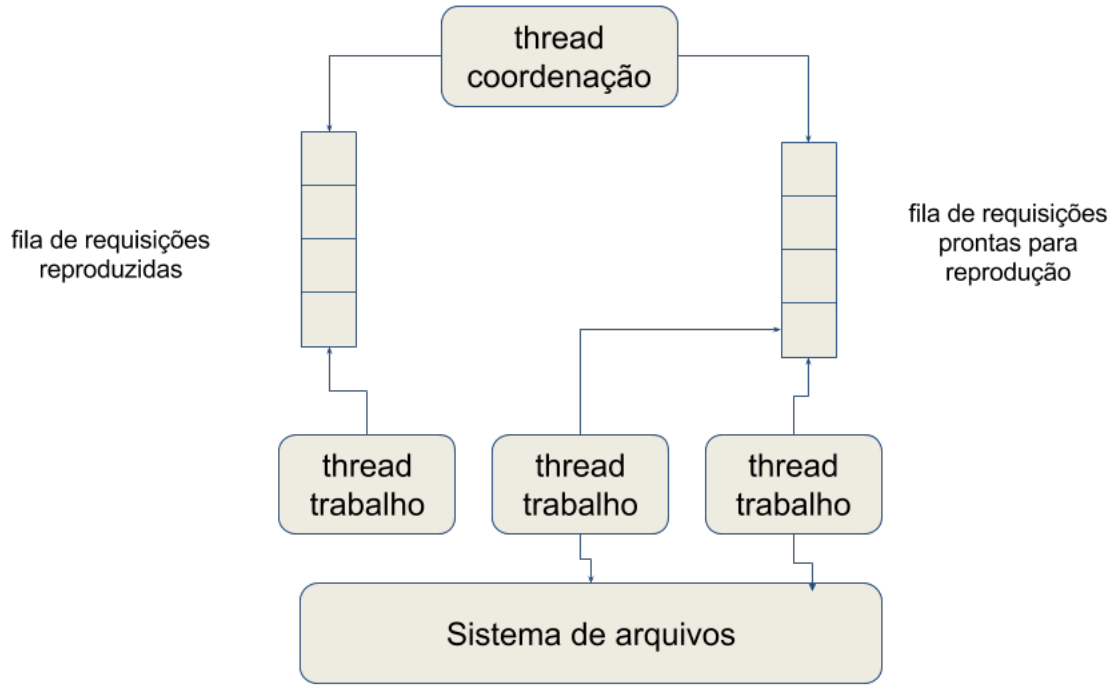


Figura 6.1: O reprodutor processa o rastro como entrada e o traduz em chamadas ao sistema para serem reproduzidas. Essas operações são realizadas por dois grupos de *threads*: coordenador e trabalhadores. A *thread* de coordenação gera os eventos que são reproduzidos pelas *threads* de trabalho.

A geração de eventos se desenvolve de acordo com as regras definidas pelas políticas de ordenação e temporização e pelo progresso feito pelas *threads* de trabalho. O reprodutor utiliza um algoritmo de marcação para gerar os eventos, conforme mostrado na Figura 6.2. Nesse algoritmo, uma requisição pode assumir três estados: i) *indisponível*; ii) *disponível*; e iii) *reproduzida*. A *thread* de coordenação gera um evento para cada requisição *disponível*, e adiciona esses eventos na fila de eventos disponíveis. Uma *thread* de trabalho ociosa retira um evento dessa fila e executa a chamada ao sistema relacionada. Após a reprodução, a *thread* de trabalho adiciona o evento em uma fila de eventos reproduzidos. A *thread* de coordenação retira eventos dessa fila e modifica o estado da requisição de *disponível* para

reproduzida. Em seguida, a *thread* de coordenação muda o estado das requisições que sucedem a requisição que acabou de ser reproduzida, de acordo com a relação de ordem, de *indisponível* para *disponível*.

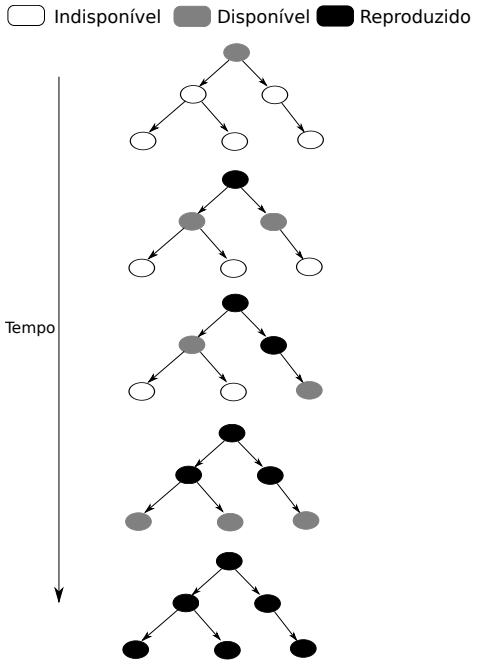


Figura 6.2: O reprodutor baseado em eventos usa um algoritmo de coloração para produzir os eventos que serão reproduzidos pelas *threads* de trabalho. Uma *thread* de trabalho ociosa executa eventos disponíveis (cinza). Após a execução, o coordenador marca o evento como *reproduzido* (preto) e muda a marcação de seus sucessores de indisponíveis (branco) para *disponíveis* (cinza).

Nossa ferramenta baseada em eventos executa como um processo *real time* para evitar a preempção do processo do reprodutor por outros processos em execução no ambiente de experimentação, como recomendado por Joukov et al. [41] e adotado por Tahiri et al. [63].

Procedimento de medição

Antes de reproduzir os rastros, é necessário recriar o estado do sistema de arquivos tal como encontrado durante a fase de captura. Para recriá-lo, usamos uma cópia dos dados coletada logo antes do procedimento de captura. Após essa etapa, o procedimento de medição inicia descarregando a cache de páginas, *dentries* e *inodes* do sistema de arquivos, através da execução do comando `drop_caches`. Isso é feito para assegurar que mudanças na cache

do sistema de arquivos, feitas durante a execução de um experimento de reprodução, não afetem a execução dos experimentos subsequentes. Depois do descarregamento das caches, nós iniciamos a reprodução dos rastros e esperamos sua finalização para coletar os dados gerados pela reprodução. Para reduzir a interferência entre o armazenamento dos dados gerados pelo reprodutor e as requisições da carga de trabalho reproduzida, tanto os dados de entrada usados pelos reprodutores quanto os dados gerados por estes são armazenados em um disco e sistema de arquivos diferentes.

6.1.3 Identificação das fontes de incerteza

Nesta seção, nós descrevemos as fontes de incerteza mais importantes para os métodos de reprodução de rastros, mostradas na Figura 6.3, e discutimos as causas que contribuem para cada uma dessas fontes de incerteza.

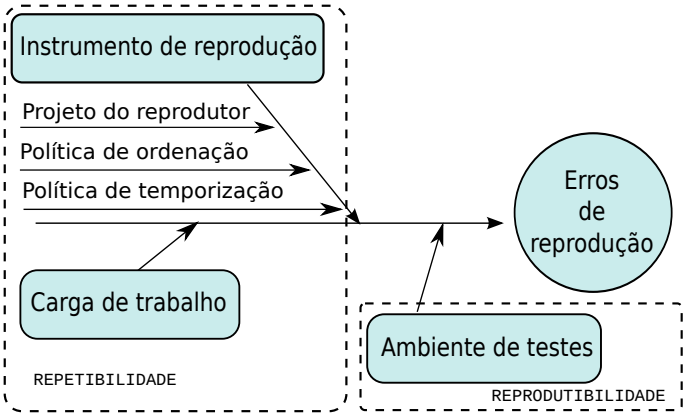


Figura 6.3: Fontes de incerteza que contribuem para erros na reprodução de rastros de utilização.

A carga de trabalho afeta o comportamento do sistema de arquivos e do sistema operacional. Como exemplificamos na Seção 5.1.3, o descarregamento das caches do sistema de arquivos é governado pela proporção entre páginas da memória usadas e livres (ambos os fatores dependem da carga de trabalho) [22]. Uma vez que o processo do programa reprodutor de rastros não está isolado do sistema de arquivos nem do núcleo do sistema operacional — por exemplo, compartilham CPU e memória — a carga de trabalho pode também impactar os instrumentos de medição, desse modo, afetando os erros de medição.

Uma ferramenta de reprodução perfeita deveria ser capaz de reproduzir um rastro de tal modo que os valores do mensurando (em nosso caso, o tempo de resposta do sistema de arquivos) fossem iguais aos valores medidos caso as aplicações alvo do rastro estivessem em execução no ambiente de reprodução.

Naturalmente, tanto decisões de projeto quanto decisões de implementação atrapalham a construção dessa ferramenta ideal. Por exemplo, em ambos os reprodutores baseados em eventos e em compilação, o acesso concorrente feito às regiões críticas pelas *threads* dos reprodutores precisa ser controlado; os atrasos adicionados para controlar esse acesso concorrente são fontes de erros sistemáticos. No reprodutor baseado em eventos, uma vez que há um componente extra para coordenar a reprodução, os erros sistemáticos devidos ao controle de acesso concorrente podem ser ainda maiores.

As políticas de ordenação e temporização também são fontes de erros. Isso porque essas políticas moldam a carga de trabalho que será reproduzida, por exemplo, alterando o nível de concorrência que pode existir na carga de trabalho e a carga de trabalho por si é uma fonte de incertezas, como descrevemos anteriormente.

Finalmente, a plataforma de execução, nela incluída o *hardware* e o *software*, também impacta a incerteza de medição. Por exemplo, o sistema de arquivos atua como uma camada de *software* entre as aplicações e o sistema de entrada/saída. Esta interação do sistema de arquivos com as camadas subjacentes define o desempenho do sistema de arquivos. A plataforma de execução é uma fonte de incerteza importante quando se analisa questões de reprodutibilidade, por exemplo, quando se compara medições feitas por laboratórios diferentes. Entretanto, uma vez que nesse trabalho nos concentramos somente nas incertezas que podem ser avaliadas em um único laboratório, nós não consideramos a influência da plataforma de execução nesta prova de conceito.

6.1.4 Caracterização da medição

Para caracterizar as medições, os valores medidos são comparados como valor verdadeiro da medição. Como descrevemos no Capítulo 3, uma vez que esse valor é desconhecido, adotamos valores de referência como aproximação.

Há duas maneiras populares para obter essas aproximações: i) realizar medições com um método de referência (um método com erros conhecidos e menores do que os erros do

método sob avaliação); e ii) realizar medições com objetos de referência (objetos com propriedades definidas, por exemplo pesos-padrão mantidos por entidades de metrologia legal).

Não podemos adotar a primeira solução, uma vez que não existe um reprodutor de rastros reconhecidamente mais acurado que pudesse ser adotado para obter medições de referências. Por essa razão, adotamos a solução baseada em objetos de referência.

Adotamos como valores de referência as medições para o tempo de resposta do sistema de arquivos como mostradas no rastro capturado. Note que o método de captura também pode ser visto como uma fonte de incerteza, como mostramos no capítulo anterior. Para amenizar este problema, ao invés de usar os dados capturados com as ferramentas `strace` e `SystemTap`, usamos os rastros colhidos no nível de aplicação — as medições de referência usadas no capítulo anterior. Essa abordagem permite usar os valores mais próximos do tempo de resposta do sistema de arquivos, percebido pelas aplicações, sendo assim, mais próximo do valor verdadeiro do mensurando.

Para a caracterização da reprodução de rastros, ilustrada na Figura 6.4, nós capturamos a execução de dois programas que geram cargas de trabalho de complexidade diferentes. O primeiro programa é o *micro-benchmark* usado no capítulo anterior, o qual executa chamadas ao sistema em um nível de carga configurável. O segundo programa é o *macro-benchmark* `filebench` que configuramos para gerar uma carga de trabalho que emula a atividade de um servidor de arquivos.

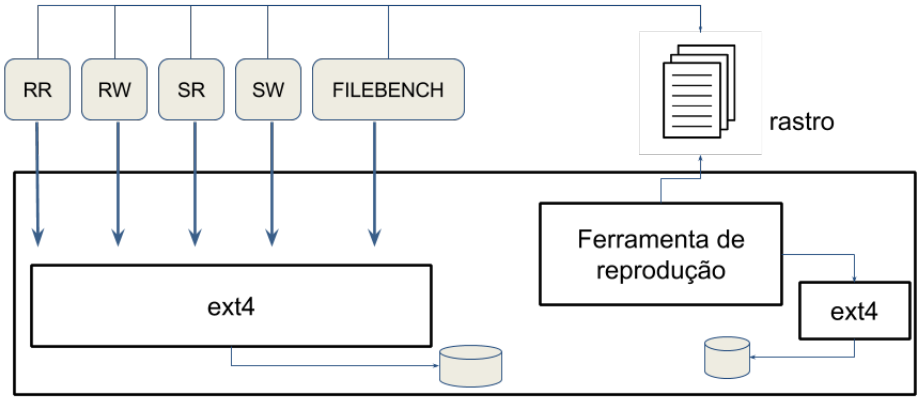


Figura 6.4: As ferramentas de reprodução tomam como entrada rastro captura da interceptação da carga de trabalho gerada por micro e macrobenchmarks.

Como descrito no capítulo anterior, o nível de carga do *micro-benchmark* é definido pelo

número de *threads* no *micro-benchmark* que executam as chamadas ao sistema. Cada *thread* executa um número configurável de chamadas para um arquivo de $10GB$ diferente.

Tal como antes, consideramos 4 níveis de carga e 4 tipos de carga. Os níveis de carga variam de 1 até 4. Os tipos de carga são: i) random read; ii) random write; iii) sequential read; e iv) sequential write. Cada chamada lê ou escreve em blocos de 4096 bytes. Cada *thread* do reprodutor executa 2000 chamadas.

Para cada combinação de nível e tipo de carga, nós geramos rastros em três diferentes modos: i) sem pausa entre as requisições; ii) com uma pausa de $10\mu sec$ entre as requisições; e iii) com uma pausa de $50\mu sec$ entre as requisições (para os cenários com pausa consideramos somente os tipos de carga sequential read e sequential write).

Os rastros gerados com o primeiro modo são usados para as medições tanto do reprodutor baseado em compilação quanto do baseado em eventos, com a política de temporização *fullspeed*. Os rastros gerados com demais modos são usados apenas pelo reprodutor baseado em eventos, com a política de temporização *timestamp*.

Com respeito ao *macro-benchmark*, o programa `filebench` foi configurado para emular a atividade de um servidor de arquivos. Nessa configuração, 4 *threads* executam sequências de operações de criação, remoção, concatenação, leitura e escrita, além de operações sobre metadados de arquivos (chamadas `stat`). O *macro-benchmark* opera sobre um conjunto de 1000 arquivos. O tamanho dos arquivos é dado por uma distribuição gama (com os parâmetros média e gama iguais a 131072 e 1.5 respectivamente). Operações de leitura e escrita envolvem todo o conteúdo do arquivo. O tamanho médio dos blocos usados nas operações de concatenação é de $16KB$. A duração de cada execução do *macro-benchmark* foi de 30 segundos.

Incidentalmente, a carga de trabalho gerada pelo programa `filebench` é mais complexa do que a carga gerada pelo *micro-benchmark*. Por exemplo, na carga gerada pelo `filebench`, o tamanho dos arquivos varia, desse modo, cada *thread* executa uma carga de trabalho diferente. Além disso, os arquivos manipulados por essas cargas podem ser até duas ordens de magnitude menores do que os arquivos manipulados pelo *micro-benchmark*, assim é possível que os arquivos usados pelo `filebench` estejam completamente na *cache* do sistema. Em última instância, o *macro-benchmark* ressalta a importância do conjunto de trabalho em memória do processo gerador de carga, para o desempenho observado do

sistema de arquivos.

Nas seções seguintes, discutiremos os métodos que aplicamos para analisar a precisão e a tendência de medição dos reprodutores baseados em eventos e em compilação.

Determinação da precisão da medição

Para aferir a precisão das medições, realizamos 50 experimentos replicados para cada rastro de utilização. O resultado do experimento é o valor médio do tempo de resposta das chamadas realizadas no experimento.

Nós analisamos a precisão dos métodos de reprodução de rastros os comparando com a precisão dos valores de referência, segundo o mesmo método aplicado na prova de conceito de captura, que repetimos aqui para facilitar a leitura. Aplicamos um teste *F* para comparar a precisão dos métodos de reprodução: para um método de medição A, seja $u(R_w^A)$ a precisão do método, ou seja, a componente da incerteza devida aos erros aleatórios. Segundo esse método, há evidência que um método B é menos preciso do que o método A se $F_r > F_{\alpha,N_B-1,N_A-1}$, em que F_r é a razão entre as variâncias dos métodos de medição:

$$F_r = \frac{u(R_w^B)^2}{u(R_w^A)^2}$$

e F_{α,N_B-1,N_A-1} é o valor crítico da distribuição F com $N_B - 1$ e $N_A - 1$ graus de liberdade ao nível de significância α [12].

Determinação da tendência da medição

Para avaliar o erro sistemático, comparamos as medições feitas com os reprodutores baseados em eventos e em compilação e os valores de referência usados nos experimentos para determinação da precisão da medição, para as mesmas combinações de nível de carga e tipo de carga de trabalho. Para decidir se a tendência de medição dos métodos é aceitável ou não, aplicamos o mesmo critério usado anteriormente: a tendência de uma medição é significativa caso sua magnitude seja maior que o dobro da incerteza [37].

6.1.5 Calibração

Em muitas aplicações de metrologia, os resultados da análise de tendência são a base para a construção de um procedimento de calibração, que posteriormente é aplicado para corrigir os erros sistemáticos. Por exemplo, no capítulo anterior, mostramos que a calibração é fundamental para a correção das medições de captura de rastros.

Embora a calibração seja útil, em alguns casos ela não pode ser aplicada — a reprodução de rastro é um desses casos. Isso acontece porque qualquer procedimento de calibração é específico para um instrumento de medição e para o ambiente experimental que foi usado para determinar a tendência da medição. Uma vez que os métodos de reprodução de rastros são tipicamente aplicados para avaliar um sistema diferente daquele usado para capturar os rastros, ou ainda, para avaliar mudanças no ambiente de experimentação, tais como o uso de um novo componente de *hardware*, as premissas para o emprego do procedimento de calibração não se aplicam.

Ainda que não possamos aplicar os resultados da análise de tendência para definir um procedimento de calibração, eles podem ser usados para identificar problemas no projeto e implementação das ferramentas de reprodução. Quando possível, ao remover essas fontes de problemas, é possível reduzir os erros sistemáticos.

6.1.6 Determinação da incerteza de medição

Nesta validação dos métodos de reprodução temos o mesmo cenário metrológico considerado para a captura de rastros: levamos em conta as fontes de incertezas possíveis de serem avaliadas em um único laboratório. Por isso, mais uma vez, a incerteza combinada, u_c, é dada pela seguinte equação [29]:

$$u_c = \sqrt{u(R_w)^2 + u(bias)^2}$$

Como descrevemos na Seção 4.1, a componente $u(R_w)$ contabiliza a incerteza em virtude dos erros aleatórios, enquanto a componente $u(bias)$ contabiliza a incerteza devida aos erros sistemáticos.

6.2 Resultados

Nesta seção, nós discutimos os resultados da validação dos métodos de reprodução de rastros, inclusos os resultados da caracterização da medição e da determinação da incerteza de medição, tanto para o *micro* quanto para o *macro-benchmark*. Para amparar o uso do protocolo de metrologia, descrevemos em maiores detalhes os resultados da validação da reprodução da carga de trabalho gerada pelo *micro-benchmark*.

Para esta carga de trabalho, na Seção 6.2.1, nós mostramos os resultados da caracterização da reprodução de rastros em termos de precisão e tendência da medição. Na Seção 6.2.2, nos concentramos no uso da análise da tendência de medição para embasar a redução dos erros sistemáticos.

Nas Seções 6.2.1 e 6.2.2, consideramos os resultados para os cenários sem pausa entre as requisições, tanto para o reprodutor baseado em eventos, quanto para o reprodutor baseado em compilação. O reprodutor baseado em compilação não dá suporte à política de temporização *timestamp*, deste modo, os cenários com pausa entre as requisições são exclusivos do reprodutor baseado em eventos. Os resultados para estes último cenários são discutidos na Seção 6.2.3 em conjunto com a análise da medição da incerteza para os cenários sem pausa entre as requisições.

Por fim, na Seção 6.2.3, nós mostramos a incerteza de medição para a carga de trabalho gerada pelo *macro-benchmark*, reproduzidas tanto pela ferramenta baseada em eventos, quanto pela ferramenta baseada em compilação. Verificamos por testes de shapiro que as medições com os reprodutores baseados em eventos e compilação são distribuídas normalmente. Nos apêndices E e F, mostramos os testes de normalidade adicionais que suportam as análises de precisão, tendência e incerteza.

6.2.1 Caracterização da medição

Precisão da medição

Na Tabela 6.2 nós mostramos a razão F_r, como definimos na Seção 6.1.4, para as medições com os reprodutores baseados em compilação e em eventos, com um nível de significância de 0.05, e $N_B - 1$ e $N_A - 1$ níveis de liberdade iguais a 49 (com base em 50 experimentos replicados). O valor calculado para F_{α,N_B-1,N_A-1} é 1.607. Tanto o reprodutor baseado em

compilação quanto o reprodutor baseado em eventos não aumentam a imprecisão observada; em apenas um dos 64 cenários avaliados, a ferramenta baseada em compilação é menos precisa do que as medições de referência.

Tabela 6.2: Comparação entre a precisão dos métodos de reprodução e a precisão das medições de referência. Os métodos de reprodução são precisos: de acordo com o teste *F* com nível de significância de 0.05, em apenas 1 dos 64 cenários avaliados, a precisão das medições de reprodução é menor do que a precisão dos valores de referência.

		Nível de carga			
		1	2	3	4
Random read (RR)					
Compilação	`FS`	0.02	0.45	0.06	0.27
	`Temporal`	0.02	0.07	0.05	0.26
Eventos	`FS`	0.02	0.44	0.14	0.91
	`Temporal`	0.02	0.67	1.10	0.82
Random read (RW)					
Compilação	`FS`	0.07	0.44	0.25	1.58
	`Temporal`	0.11	0.36	0.26	0.56
Eventos	`FS`	0.10	0.36	0.28	0.57
	`Temporal`	0.06	1.01	0.82	0.58
Sequential read (SR)					
Compilação	`FS`	0.05	0.02	0.03	0.01
	`Temporal`	0.03	0.01	0.01	0.01
Eventos	`FS`	0.05	0.19	0.84	0.11
	`Temporal`	0.10	0.13	0.26	0.16
Sequential write (SW)					
Compilação	`FS`	0.28	0.91	0.33	0.25
	`Temporal`	0.34	**1.83**	0.82	0.34
Eventos	`FS`	1.08	1.13	0.83	0.93
	`Temporal`	0.95	1.06	1.17	1.46

Tendência de medição

A Figura 6.5 mostra o tempo de resposta médio para os experimentos replicados de reprodução de rastros e os valores de referência médios (linhas tracejadas), para todas as combinações de nível de carga de trabalho e tipos de carga de trabalho.

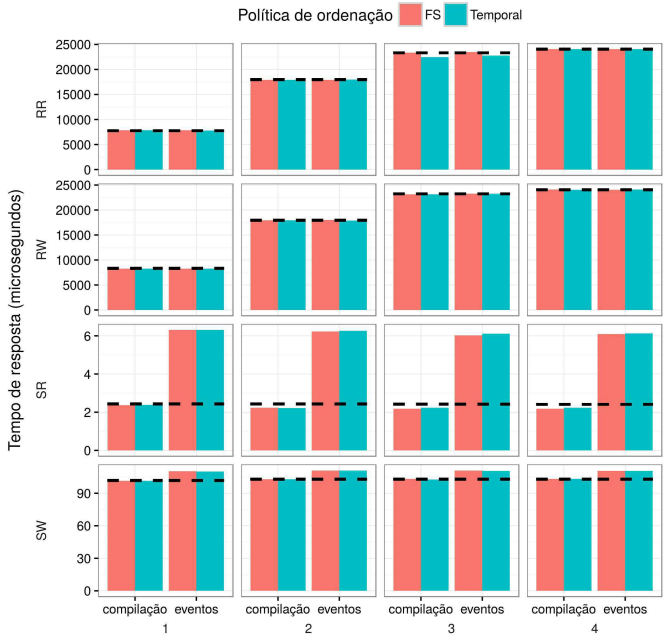


Figura 6.5: Erro sistemático dos procedimentos de reprodução de rastros. Os grupos de barra representam o tempo de resposta médio em medições replicadas com as ferramentas de reprodução baseadas em compilação e eventos. As linhas tracejadas representam os valores de referência médios.

Como base no critério definido na Seção 6.1.4, o erro sistemático das medições realizadas com a ferramenta baseada em compilação são insignificantes, para todas as combinações de tipos de carga de trabalho e níveis de carga. Com base no mesmo critério, o erro sistemático do reprodutor baseado em eventos é significativo para a carga de trabalho do tipo *sequential read*.

6.2.2 Correção dos erros sistemáticos da reprodução

Como dissemos na Seção 6.1.5, embora não seja possível aplicar os resultados da análise dos erros sistemáticos para definir um procedimento de calibração, esses resultados podem ser

usados para identificar problemas no projeto e implementação dos reprodutores de rastros. Ao remover essas fontes de problemas, é possível reduzir os erros sistemáticos. Com base no critério de significância adotado na Seção 5.1.4, os erros sistemáticos mostrados pelo reprodutor baseado em eventos, para a reprodução de carga de trabalho do tipo *sequential read* indicam bons alvos para aperfeiçoamento do reprodutor.

Para reduzir os erros sistemáticos observados para a carga de trabalho do tipo *sequential read* avaliamos a influência do componente coordenador do reprodutor baseado em eventos. Nós modificamos o projeto original do reprodutor, removendo o componente de coordenação e distribuindo suas responsabilidades para as *threads* de trabalho. Nesse novo projeto, os eventos que cada *thread* de trabalho irá executar são definidos no início da execução do reprodutor. Uma variável condicional é adicionada a cada evento para aplicar a política de ordenação; essa variável de guarda sinaliza quando os eventos predecessores de um dado evento já foram executados.

A Figura 6.6 mostra a redução dos erros sistemáticos, para a carga de trabalho de tipo *sequential read*, após realizadas a modificação descrita acima para o reprodutor baseado em eventos. Essa modificação, quando aplicada para a reprodução de cargas de trabalho de tipo *sequential read*, reduz o erro sistemático relativo de mais de 150% para não mais que 25%.

6.2.3 Determinação da incerteza da medição

Nas seções anteriores, nós descrevemos os métodos de reprodução e caracterizamos seus erros. Nesta seção, após as modificações feitas no reprodutor baseado em eventos para mitigar o impacto das fontes de erros sistemáticos, completamos o protocolo de validação calculando a incerteza combinada u_c para as medições das cargas de trabalho geradas pelo *micro-benchmark* (Seção 6.2.3) e pelo *macro-benchmark* (Seção 6.2.3).

Micro-benchmark

A Tabela 6.3 mostra a incerteza combinada que calculamos para os reprodutores baseados em eventos e em compilação, para todas as combinações de tipos de carga de trabalho e níveis de carga de trabalho, bem como políticas de ordenação. Essa tabela mostra os resultados para os rastros sem pausa entre as requisições, reproduzidos com a política de temporização

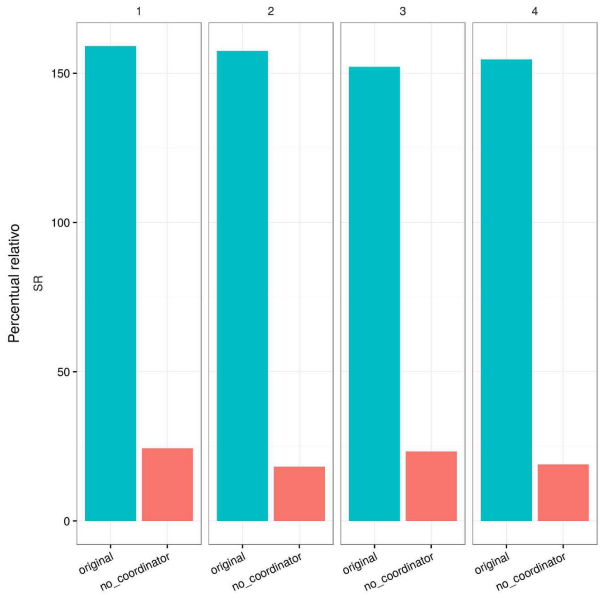


Figura 6.6: Percentual relativo do erro sistemático (com respeito aos valores medidos) do reprodutor baseado em eventos antes (`original`) e depois (`no_coordinator`) das modificações no projeto do reprodutor. O erro sistemático é mostrado para a carga de trabalho de tipo *sequential read*.

fullspeed. Para as reproduções realizadas com o reprodutor baseado em eventos, também mostramos a incerteza observada para as reproduções antes das modificações no código do reprodutor que descrevemos na seção anterior (entre parênteses). A incerteza combinada, com descrevemos na Seção 4.1, é reportada como $y_m \pm 2 \cdot u_c$, em que y_m é o valor médio dos experimentos replicados. Consideramos um nível de confiança de 95.5%.

Para as cargas de trabalho de acesso aleatório, há pouca diferença entre as incertezas combinadas dos reprodutores baseados em eventos e compilação, mesmo depois das modificações feitas no reprodutor baseado em eventos. Para a carga de trabalho de tipo *random read*, a incerteza para a ferramenta baseada em eventos é de 7.57%. Já para a ferramenta baseada em compilação é de 4.8% e de 2.29%, respectivamente antes e depois das modificações. Para a carga de trabalho de tipo *random write*, a incerteza da ferramenta baseada em compilação é de até 2.37%, e de até 2.29% e 10.78% para a ferramenta baseada em eventos, respectivamente antes e depois das modificações.

Para as cargas de trabalho de acesso sequencial, as modificações no reprodutor baseado em eventos reduzem bastante a incerteza de medição, entretanto a incerteza para a ferramenta

baseada em eventos ainda é menor.

Para a carga de trabalho de tipo *sequential read*, a incerteza de ferramenta baseada em compilação não é maior que 20.26%, enquanto a incerteza da ferramenta baseada em eventos é de até 318.29%. antes da sua modificação e de 49.58% após a modificação. Para a carga de trabalho de tipo *sequential write*, a incerteza da ferramenta baseada em compilação é de até 1.03%, enquanto a incerteza da ferramenta baseada em eventos é de até 16.63%. antes da sua modificação e de 5.05% após a modificação.

Além dos rastros sem pausa entre requisições, nós também mostramos a incerteza para a reprodução de rastros com pausa entre as requisições, para as cargas de trabalho *sequential read* e *sequential read* (que são mais afetadas pelos erros sistemáticos). A Tabela 6.4 e a Tabela 6.5 mostram os resultados para a ferramenta baseada em eventos para as combinações de tipos de carga de trabalho, níveis de carga de trabalho bem como políticas de ordenação e a política de temporização *timestamp*. Nós consideramos dois valores para a pausa entre as requisições quando os rastros foram gerados: i) $10\mu sec$; e ii) $50\mu sec$.

A Tabelas 6.4 e a Tabela 6.5 indicam que a incerteza para a reprodução dos rastros com pausa entre as requisições é maior que a incerteza dos cenários sem pausa. Há três razões para isto: i) as fontes de incertezas mostradas na Tabela 6.3, as quais também operam para as reproduções dos rastros sem pausa entre as requisições, retardam a sua execução; ii) as funções de atraso de tempo normalmente usadas pelos reprodutores são inacuradas (nós usamos a função `nanosleep`); e iii) a política de temporização *timestamp* define que uma requisição deve ser reproduzida o mais próximo possível das marcações de tempo tal como se observa no rastro [50]. Quando combinadas, estas condições implicam que o reprodutor, em muitos casos, não espera antes de reproduzir uma requisição, uma vez que esta requisição já está atrasada. Como consequência, as *threads* de trabalho ficam menos propensas a serem preemptadas por causa de um bloqueio, assim, reduzindo o tempo de resposta observado; como a Tabela 6.4 e a Tabela 6.5 mostram, para a carga de trabalho do tipo *random read* e *random write*, os valores medidos médios são menores do que os de referência (em contraste, a Figura 6.5 mostra que, para os cenários de reprodução de rastros sem pausa entre as requisições, os valores medidos médios são maiores do que os valores de referência).

Tabela 6.3: Incerteza de medição combinada $y_m \pm 2 \cdot u_c$ com nível de confiança de 95.5%. Entre parênteses mostramos a incerteza combinada para o reprodutor baseado em eventos antes das modificação em seu projeto.

Nível de carga	`FS`	`Temporal`
	Eventos	
	Random read (RR)	
1	$7843.38 \pm 2.78\%$ ($7828.84 \pm 2.52\%$)	$7843.42 \pm 2.79\%$ ($7818.50 \pm 2.34\%$)
2	$17899.96 \pm 1.18\%$ ($17894.79 \pm 1.23\%$)	$17948.12 \pm 0.97\%$ ($17961.08 \pm 0.75\%$)
3	$23474.29 \pm 2.30\%$ ($23515.92 \pm 2.65\%$)	$22930.96 \pm 4.26\%$ ($22803.71 \pm 4.80\%$)
4	$24062.49 \pm 0.38\%$ ($24052.98 \pm 0.32\%$)	$24051.38 \pm 0.36\%$ ($24062.10 \pm 0.28\%$)
	Random write (RW)	
1	$8317.32 \pm 1.54\%$ ($8278.15 \pm 1.88\%$)	$8302.64 \pm 1.60\%$ ($8277.81 \pm 1.90\%$)
2	$18026.41 \pm 1.65\%$ ($17988.54 \pm 1.49\%$)	$17964.83 \pm 1.75\%$ ($17871.73 \pm 1.70\%$)
3	$23221.51 \pm 2.26\%$ ($23286.40 \pm 2.29\%$)	$22024.04 \pm 10.78\%$ ($23300.97 \pm 2.16\%$)
4	$24057.48 \pm 0.48\%$ ($24063.92 \pm 0.52\%$)	$24028.71 \pm 0.54\%$ ($24076.65 \pm 0.45\%$)
	Sequential read (SR)	
1	$3.02 \pm 49.14\%$ ($6.31 \pm 318.39\%$)	$3.03 \pm 49.58\%$ ($6.31 \pm 318.37\%$)
2	$2.87 \pm 37.41\%$ ($6.22 \pm 312.19\%$)	$2.86 \pm 36.29\%$ ($6.26 \pm 315.09\%$)
3	$2.90 \pm 40.81\%$ ($6.03 \pm 298.52\%$)	$2.99 \pm 47.70\%$ ($6.11 \pm 304.41\%$)
4	$2.80 \pm 33.54\%$ ($6.10 \pm 306.51\%$)	$2.86 \pm 38.86\%$ ($6.13 \pm 309.40\%$)
	Sequential write (SW)	
1	$99.51 \pm 4.51\%$ ($110.25 \pm 16.63\%$)	$99.31 \pm 4.90\%$ ($109.95 \pm 16.05\%$)
2	$100.75 \pm 4.29\%$ ($111.05 \pm 15.76\%$)	$100.92 \pm 3.96\%$ ($110.83 \pm 15.35\%$)
3	$100.63 \pm 4.57\%$ ($110.86 \pm 15.36\%$)	$100.62 \pm 4.58\%$ ($110.76 \pm 15.16\%$)
4	$100.43 \pm 4.94\%$ ($110.75 \pm 15.15\%$)	$100.38 \pm 5.05\%$ ($110.71 \pm 15.06\%$)
	Compilação	
	Random read (RR)	
1	$7835.39 \pm 2.63\%$	$7833.36 \pm 2.59\%$
2	$17902.81 \pm 1.17\%$	$17928.60 \pm 0.89\%$
3	$23355.96 \pm 1.86\%$	$22466.99 \pm 7.57\%$
4	$24056.48 \pm 0.30\%$	$24054.55 \pm 0.30\%$
	Random write (RW)	
1	$8294.25 \pm 1.68\%$	$8288.04 \pm 1.78\%$
2	$17962.95 \pm 1.49\%$	$17925.00 \pm 1.47\%$
3	$23153.32 \pm 2.34\%$	$23145.34 \pm 2.37\%$
4	$24095.63 \pm 0.69\%$	$24042.36 \pm 0.50\%$
	Sequential read (SR)	
1	$2.37 \pm 10.41\%$	$2.39 \pm 9.53\%$
2	$2.23 \pm 18.75\%$	$2.22 \pm 19.16\%$
3	$2.19 \pm 21.24\%$	$2.23 \pm 18.49\%$
4	$2.19 \pm 20.26\%$	$2.23 \pm 17.18\%$
	Sequential write (SW)	
1	$101.37 \pm 0.94\%$	$101.32 \pm 1.03\%$
2	$102.92 \pm 0.59\%$	$102.79 \pm 0.77\%$
3	$103.11 \pm 0.56\%$	$102.69 \pm 0.77\%$
4	$103.34 \pm 0.86\%$	$103.23 \pm 0.69\%$

Tabela 6.4: Incerteza de medição combinada $y_m \pm 2 \cdot u_c$ com nível de confiança de 95.5%, para a reprodução de rastros com pausa de $10\mu s$ entre requisições com o reprodutor baseado em eventos.

Nível de carga	FS	Temporal
	Sequential read (SR)	
1	$6.41 \pm 326.95\%$	$6.37 \pm 323.44\%$
2	$6.44 \pm 329.77\%$	$7.15 \pm 387.97\%$
3	$6.11 \pm 304.53\%$	$7.33 \pm 405.63\%$
4	$6.27 \pm 320.42\%$	$7.07 \pm 387.54\%$
	Sequential write (SW)	
1	$110.30 \pm 16.73\%$	$110.11 \pm 16.37\%$
2	$110.61 \pm 14.92\%$	$110.83 \pm 15.34\%$
3	$110.67 \pm 14.99\%$	$110.11 \pm 14.34\%$
4	$111.15 \pm 15.91\%$	$110.89 \pm 15.41\%$

Tabela 6.5: Incerteza de medição combinada $y_m \pm 2 \cdot u_c$ at a 95.5% com nível de confiança de 95.5% para a reprodução de cargas de trabalho com pausa de $50\mu s$ com o reprodutor baseado em eventos.

Nível de carga	FS	Temporal
	Sequential read (SR)	
1	$6.44 \pm 329.35\%$	$6.45 \pm 330.42\%$
2	$6.32 \pm 320.09\%$	$8.16 \pm 471.07\%$
3	$6.26 \pm 317.24\%$	$7.36 \pm 407.71\%$
4	$6.26 \pm 319.63\%$	$7.47 \pm 420.12\%$
	Sequential write (SW)	
1	$110.15 \pm 16.45\%$	$110.41 \pm 16.95\%$
2	$111.09 \pm 15.85\%$	$111.23 \pm 16.12\%$
3	$111.02 \pm 15.67\%$	$110.90 \pm 15.43\%$
4	$111.01 \pm 15.65\%$	$110.82 \pm 15.27\%$

Macro-benchmark

Após a análise da incerteza das reproduções das cargas de trabalho geradas pelo *micro-benchmark*, nós analisamos a incerteza das reproduções das cargas geradas pelo *macro-benchmark*. A Tabela 6.6 mostra a incerteza combinada que encontramos para os reprodutores baseados em eventos e em compilação. Além da incerteza combinada, também mostramos o valor de referência considerado. Assim como na análise da reprodução das cargas geradas pelo *micro-benchmark*, reportamos a incerteza combinada como $y_m \pm 2 \cdot u_c$, onde y_m é a média dos valores obtidos nos experimentos replicados, considerado um nível de confiança de 95.5%.

A Tabela 6.6 mostra somente os resultados obtidos para a política de ordenação *FS*; tal como nas análises anteriores, não há diferença significativa entre os resultados obtidos com as políticas de ordenação *FS* e *Temporal*. Além disso, a Tabela 6.6 mostra apenas a incerteza para as operações de leitura e escrita, uma vez que estas têm maior impacto sobre o desempenho da carga de trabalho gerada pelo *macro-benchmark* do que as operações sobre metadados tais como a criação e remoção de arquivos.

A Tabela 6.6 indica que a incerteza de medição para a reprodução da carga de trabalho gerada pelo *macro-benchmark*, tanto para o reprodutor baseado em evento quanto para o reprodutor baseado em compilação, é maior do que a incerteza da reprodução da carga de trabalha mais simples, gerada pelo *micro-benchmark*.

Por exemplo, para as operações de leitura, a incerteza de medição para a ferramenta baseada em compilação atinge 87.24% enquanto que a incerteza de medição relacionada com a mesma ferramenta para a carga de trabalho *read* gerada pelo *micro-benchmark* (**SR**) não é maior do que 20.26%. Para as operações *write* da carga de trabalho gerada pelo *macro-benchmark*, a incerteza de medição do reprodutor baseado em compilação é de 58.8% enquanto a incerteza de medição para operações de escrita geradas pelo *micro-bechmark* (**SW**) não ultrapassa 1.03%. Note que, como descrevemos na Seção 6.2.1, as operações *read* e *write* da carga de trabalho geradas pelo *macro-benchmark* são operações sequenciais.

Para o reprodutor baseado em eventos, a incerteza de medição para a reprodução das operações *read* da carga de trabalho gerada pelo *macro-benchmark* atinge 120.54% enquanto a incerteza para a reprodução das operações de leitura para a carga de trabalho gerada pelo *microbenchmark* (**SR**) não é maior do que 48.58%. Para o reprodutor baseado em eventos, a

incerteza de medição para a reprodução de operações *write* da carga de trabalho gerada pelo *macro-benchmark* atinge 88.17% enquanto a incerteza para a reprodução das operações read (**SW**) da carga de trabalho geradas pelo *micro-benchmark* não é maior do que 5.05%.

Tabela 6.6: Incerteza combinada $y_m \pm 2 \cdot u_c$ com nível de confiança de 95.5% para a reprodução da carga de trabalho gerada pelo *macro-benchmark*. As ferramentas de reprodução aplicaram a política de ordenação *FS* em combinação com a política de temporização *fullspeed*.

	Eventos	Compilação	Referência
Read	$20.15 \pm 120.54\%$	$28.60 \pm 87.24\%$	50.72
Write	$46.96 \pm 88.17\%$	$59.28 \pm 58.80\%$	83.95

Observamos que há duas fontes de erro que contribuem para incerteza para a reprodução da carga de trabalho gerada pelo *macro-benchmark*. A primeira fonte é negativa (reduz o valor do mensurando) e contribui fortemente para a geração dos erros sistemáticos. A segunda fonte é positiva (aumenta o valor do mensurando) e explica a diferença entre a incerteza da medição observada para as ferramentas baseadas em eventos e em compilação.

A primeira fonte de erros é causada pela diferença entre os ambientes de captura e reprodução. Mais especificamente, o consumo de memória das ferramentas de reprodução é menor do que o consumo de memória do *macro-benchmark* `filebench`. Como consequência, durante a reprodução, há mais memória disponível para a cache de páginas, e assim, é mais provável que as operações feitas no sistema de arquivos sejam atendidas pelo subsistema de memória ao invés do disco rígido; com isso, reduzindo o tempo de resposta.

A segunda fonte de erro é relacionada com as limitações das ferramentas de reprodução de rastros para corretamente emular a concorrência entre requisições definida pelo rastro capturado. A Figura 6.7 mostra a função de distribuição cumulativa empírica para o nível de concorrência, que observamos para as cargas de trabalho capturadas e reproduzidas, para ambos os reprodutores baseados em eventos e em compilação. O nível de concorrência é dado pelo número de requisições em execução em um dado momento.

A Figura 6.7 mostra que o reprodutor baseado em eventos reduz a concorrência enquanto que o reprodutor baseado em compilação a aumenta, em comparação com a concorrência

observada na carga de trabalho de referência. Para as reproduções realizadas com a ferramenta baseada em eventos, durante a execução de pelo menos 75% da requisições, não havia outras requisições em execução. Por sua vez, para as reproduções realizadas com a ferramenta baseada em compilação, durante a execução de mais da metade da requisições, outras duas requisições estavam em execução ao mesmo tempo. Quanto maior for o nível de concorrência, maior é o tempo em que uma *thread* ficará sem executar após uma preempção.

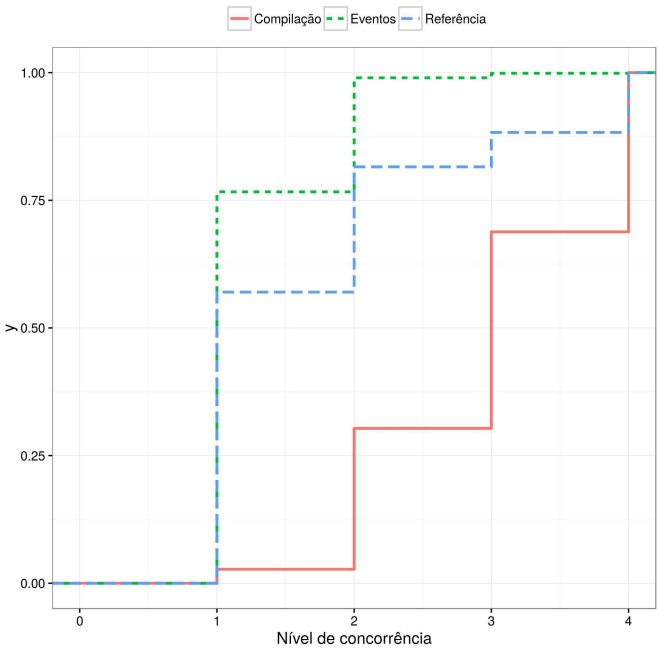


Figura 6.7: Função de distribuição cumulativa para o nível de concorrência encontrada na reprodução das cargas de trabalho gerada pelo *macro-benchmark* e para a carga de trabalho de referência.

6.3 Discussão

O estado-da-prática sobre os métodos de reprodução de rastros é marcado por um bom número de ferramentas, projetos e procedimentos bastante díspares. Nesse cenário, ao caracterizar a qualidade dos métodos disponíveis, o protocolo de validação ajuda não apenas a escolha de alternativas adequadas mas também ajuda a refinar as melhores práticas no uso destes métodos — por derradeiro, contribui para o desenvolvimento de um método padrão para a reprodução de rastros.

Para as cargas de trabalho de tipo *random read* e *random write*, não há grande diferença entre a incerteza dos reprodutores. Além disso, a incerteza é baixa para ambos os reprodutores.

Quando se tem como alvo a reprodução de cargas de trabalho do tipo *sequential read*, nós também argumentamos que os reprodutores baseados em compilação e em eventos são escolhas equivalentes (por uma razão mais sutil). Nesse cenário, mesmo após as modificações feitas no reprodutor baseado em eventos, a incerteza da ferramenta baseada em compilação é menor, para todas as combinações de nível de carga e políticas de reprodução. Entretanto, a diferença entre os valores medidos está na faixa dos nanossegundos. Ou seja, embora tenhamos uma diferença no erro sistemático, é improvável que essa diferença leve a qualquer consequência prática na análise de desempenho de sistema de arquivos.

Observamos também que o reprodutor baseado em eventos apresenta limitações para reproduzir rastros com pausa entre requisições — a incerteza é mais alta do que aquela para a reprodução de rastros sem pausa entre requisições. Por exemplo, para cargas de trabalho de tipo *sequential read* sem pausa entre requisições, a incerteza combinada é de até 49.58%, enquanto para a reprodução da carga de trabalho de mesmo tipo, com pausa entre as requisições chega até 387.97%. Para a carga de trabalho *sequential write*, a incerteza da reprodução de rastros sem pausa não é maior que 5.05%, enquanto que para a reprodução de rastros com pausa é de até 16.73%. Essas observações indicam que, quando se mede uma métrica sensível tal como a que adotamos nesta prova de conceito (o tempo de resposta), as funções de controle de tempo normalmente usadas podem não ser apropriadas para serem empregadas na implementação da política *timestamp*. Em termos de metrologia, isso impõe um **limite de quantificação** na escala de microssegundos. Uma possível solução para reduzir a incerteza nestes cenários seria adotar métodos de controle de tempo de maior resolução, embora menos portáveis, tais como o registrado HPET [1].

Finalmente, observamos que para reproduzir rastros de maneira acurada é importante coletar mais informação sobre o ambiente de captura, além da atividade do sistema de arquivos. Como mostra a análise da incerteza para a reprodução da carga de trabalho gerada pelo *macro-benchmark*, diferenças entre os ambientes de captura e reprodução — no caso, o uso de memória — podem afetar a qualidade das medições. Por exemplo, a Tabela 6.6 aponta que a incerteza para a reprodução das operações do tipo *read* atinge 120.54% com a ferramenta

baseada em eventos e 87.24% com a ferramenta baseada e compilação. A incerteza também é alta para a reprodução das operações do tipo *write*, atinge 88.17% para a ferramenta baseada em eventos e $58,80\%$ para a ferramenta baseada em compilação. Em ambos os casos, a incerteza para a reprodução da carga de trabalho gerada pelo *macro-benchmark* é maior do que incerteza para a reprodução da carga de trabalho gerada pelo *micro-benchmark*. Em nosso caso, como o ambiente experimental de reprodução é o mesmo da captura, é possível mitigar esse problema; por exemplo, aumentando artificialmente o consumo de memória durante a reprodução para emular o consumo maior que foi observado durante a captura. De qualquer modo, esse resultado indica que os métodos de reprodução que consideramos podem não ser adequados para serem usados em condições de reprodutibilidade, por exemplo, quando há diferenças entre os ambientes de experimentação. Da mesma maneira que fizemos para a prova de conceito de captura de rastros, no Apêndice C, compilamos as características dos métodos de reprodução avaliados.

Capítulo 7

Conclusão

Há ainda muito o que ser feito na área de metodologia da avaliação de desempenho de sistemas de arquivos. Por isto, encerrando esse trabalho, é apropriado recapitular os resultados descritos ao longo do documento e indicar algumas implicações desses resultados bem como problemas em aberto na área.

No Capítulo 2, descrevemos os problemas nesta área de avaliação de desempenho. Eles foram identificados primeiro por Zadok et al. [65] e ao longo dos anos houve pouco sinal de mudança. A maior parte dos trabalhos que aplicam metodologias baseadas em reprodução de rastros desenvolve seu próprio ferramental, sem maiores considerações metodológicas (muitas vezes não há qualquer descrição que possibilite a reprodução do trabalho). Há ainda trabalhos que usam ferramentas de captura e reprodução desenvolvidas (ou rastros capturados) por outros autores. Infelizmente, pouco se discute se a aplicação dessas ferramentas é adequada em um contexto diferente.

No Capítulo 3, apresentamos alguns aspectos básicos de metrologia. O principal objetivo da apresentação foi motivar o uso dessa disciplina para entender os problemas dos métodos baseados em reprodução de rastros. Visto através das lentes de metrologia, os experimentos de captura e reprodução de rastros empregam instrumentos de medição para obter um mensurando, geralmente envolvendo alguma medida de duração de tempo, por exemplo, tempo de resposta. As medições são afetadas por fontes de incerteza, tais como a imprecisão dos mecanismos de temporização do sistema operacional, que ocasionam os erros de medição. As fontes de incerteza causam prejuízo à exatidão da medição — o grau de concordância entre o valor medido e o valor verdadeiro.

No Capítulo 4, consideramos com mais detalhes como aplicar metrologia no cenário de captura e reprodução de rastros. Descrevemos um protocolo experimental, popularmente aplicado em outras ciências, por exemplo, em química analítica, para verificar se um procedimento de medição é adequado para o uso pretendido. Esse protocolo considera: o levantamento das fontes de incerteza que afetam os métodos sob análise; o controle dessas fontes e a observação de seus efeitos sobre os erros de medição; a correção destes erros, quando possível; e por fim, a contabilização da incerteza de medição. Procuramos manter a descrição feita nesse capítulo independente de sua aplicação nos cenários de captura e reprodução de rastros — esperamos que o conteúdo possa ser útil em outros contextos.

Em seguida, aplicamos o protocolo descrito em duas provas de conceito, que consideram a análise de métodos de captura e reprodução populares na literatura.

No Capítulo 5, aplicamos o protocolo de validação na análise de dois métodos de captura. O primeiro baseado na ferramenta `strace` que executa no espaço do usuário. O segundo baseado na ferramenta `SystemTap` que executa como um módulo do sistema operacional. É útil pensar nos resultados dessa primeira prova de conceito tanto do ponto de vista da condução de novas medições de captura quanto do uso de rastros capturados por outros. Quando se considera novas capturas, o resultado mais direto do estudo é a organização de um catálogo dos métodos de medição. Tal como em outras ciências, com um catálogo disponível, é mais fácil avaliar os compromissos entre a facilidade de uso e acurácia das ferramentas `strace` e `SystemTap`, então, decidir se um dado método atender o propósito definido. No Apêndice B, compilamos os resultados principais da prova de conceito de captura em um desses catálogos.

É importante ter em mente que a qualidade do método de medição escolhido estará conforme o especificado, caso: i) o procedimento seja conduzido como definido; ii) os materiais usados no experimento e a configuração do ambiente estejam de acordo como o determinado; e iii) as fontes de erro atuem conforme o esperado. Enquanto os dois primeiros fatores dependem da capacidade do laboratório e da proficiência dos operadores, o terceiro fator, muitas vezes, não é possível de ser controlado completamente. Por exemplo, o levantamento das fontes de erros feitos na validação do método não é exaustivo, deste modo, alguns fatores podem ter sido negligenciados. Uma boa prática para verificar a possibilidade de uso de um método de medição é reproduzir os experimentos usados na validação desse método; tão

melhor será caso a incerteza medida seja equiparável àquela encontra na validação.

Em muitos casos, os rastros, caso tenham sido publicados, são usados em propósitos não determinados quando da captura. Em outras palavras, não é possível antecipar se a escolha do método de captura será adequada para todos os usos que serão feitos dos rastros no futuro. Para melhor apoiar o uso posterior dos dados coletados, deve-se incentivar a coleta e publicação dos dados adicionais que permitam tanto o cálculo da incerteza de medição quando a criação das curvas de calibração. Como vimos anteriormente, os métodos de captura não são seletivos com respeito à carga de fundo; é preciso que se colete informações sobre essa carga de fundo, caso contrário, não é possível realizar uma calibração adequada. Além de não serem seletivos, os métodos de captura são afetados pela configuração do ambiente de testes, por exemplo, pelo sistema operacional em uso. Assim, principalmente em medições de longa duração (nas quais o sistema operacional instalado pode ser modificado) esses métodos podem ser inconsistentes — ou seja, podem apresentar erro variável. Nesses casos, para ter um procedimento de calibração apropriado, é preciso manter um inventário da configuração do ambiente de testes durante a captura dos rastros.

No Capítulo 6, aplicamos novamente o protocolo de validação, dessa vez para a análise de dois métodos de reprodução de rastros. O primeiro método emprega uma ferramenta de captura baseada em compilação. O segundo método emprega uma ferramenta baseada em eventos. A primeira observação importante de nossa análise indicou que uma prática sugerida na literatura, a execução do programa de reprodução como um processo *real time*, apresenta um efeito colateral: o aumento dos erros sistemáticos. Acreditamos que em muitos casos essa prática pode ser abandonada; uma vez que o ambiente experimental de reprodução é controlado, é possível evitar a execução de processos e serviços intrusivos. Diferente da prova de conceito anterior, procedimentos de calibração não são aplicáveis em experimentos de reprodução de rastros. De qualquer forma, a análise dos erros sistemáticos pode ser usada para guiar o melhoramento das ferramentas de reprodução. A segunda observação importante de nossa análise indicou que, uma vez otimizada, a incerteza das medições com a ferramenta baseada em eventos se aproxima da incerteza das medições com a ferramenta baseada em compilação. Por fim, observamos que as políticas de temporização descritas na literatura bem como sua implementação no reprodutor baseado em eventos não são adequadas para a reprodução de cargas de trabalho com pausas entre requisições. Enquanto que os

resultados de captura pode ser analisados com dois interesses — da realização da captura e do uso dos rastros — os resultados da prova de conceito de reprodução interessam principalmente ao pesquisador que precisa escolher um método de reprodução. Nessa prova de conceito, tal como fizemos para a prova de conceito de captura, compilamos os resultados principais da prova de conceito de reprodução em um catálogo no Apêndice C.

7.1 Trabalho futuros

Na análise da variabilidade das medições de captura e reprodução, o protocolo adotado nesse documento assume a execução de experimentos em **condições repetíveis** — condições em que não há mudança nos materiais empregados, no ambiente experimental e nos instrumentos de medição durante a execução dos experimentos. Entretanto, na prática, alguns fatores podem aumentar a variabilidade das medições ainda mais. Este é o caso, por exemplo, quando usamos máquina diferentes para reproduzir os rastros — algo comum quando se é necessário comparar resultados obtidos em laboratórios diferentes. Uma vez que no protocolo que adotamos não se leva em consideração estas fontes de erros, a incerteza calculada é, de fato, um limite inferior. Nessa direção, as análises aqui descritas poderiam ser expandidas para considerar **condições reproduzíveis** de experimentação — condições em que há mudanças nos materiais, ambiente ou no instrumento de medição.

Na prova de conceito de captura de rastros, aplicamos um procedimento de calibração para corrigir erros sistemáticos. É importante ter em conta que, embora eficaz, o procedimento de calibração não é perfeito. Como consequência desta limitação, o próprio procedimento de calibração é uma uma fonte de erros, contribuindo então para a incerteza da medição. Para compreender melhor as limitações do procedimento de calibração, é preciso analisar o impacto do uso de curvas de calibração imperfeitas sobre a incerteza das medições de captura de rastros. Em particular, é importante estudar o cenário em que o procedimento de calibração é usado para corrigir cargas de trabalho diferentes daquelas usadas para definir a curva de calibração.

Na prova de conceito de reprodução, observamos limitações na ferramenta de reprodução baseada em eventos na reprodução de cargas de trabalho com pausas entre requisições. É preciso estudar como superar estas limitações. Como um primeiro passo nesta investigação,

poderia se avaliar uma nova implementação da política de temporização *Temporal* que empregue mecanismos de controle de tempo de maior resolução, por exemplo, o mecanismo baseado nos registrador `HPET`. Outra direção de trabalho futuro é implementar essa política de temporização também na ferramenta baseada em compilação, assim, permitindo avaliar se outros aspectos do projeto dos reprodutores afetam a reprodução de cargas de trabalho com pausa entre as requisições.

Tanto na prova de conceito de captura quanto de reprodução, consideramos a carga de trabalho submetida ao sistema de arquivos como uma fonte de erros. Em nossos experimentos, geramos cargas bastante simples. É possível que cargas mais complexas gerem um padrão de erros diferente daquele que observamos nas provas de conceito que conduzimos. Validar os métodos de captura e reprodução com novas cargas de trabalho em ordem crescente de complexidade é outra direção para trabalhos futuros. Por exemplo, podemos usar cargas de trabalho geradas por rastros da atividade de aplicações reais.

Bibliografia

[1] Especificação do hardware HPET. http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/software-developers-hpet-spec-1-0a.pdf/.

[2] Filebench benchmark. http://sourceforge.net/projects/filebench/.

[3] Iometer bechmark. http://www.iometer.org/.

[4] Página web da ferramenta de captura de pacotes *tcpdump*. http://www.tcpdump.org/.

[5] Página web da ferramenta de captura de pacotes *wireshark*. http://www.wireshark.org/.

[6] Xdd. http://sourceforge.net/projects/xdd/.

[7] *ISO 5725-1: Accuracy (Trueness and Precision) of Measurement Methods and Results - Part 1 : General Principles and Definitions*. International Organization for Standardization, Geneva, 1994.

[8] *ISO 5725-2: Accuracy (Trueness and Precision) of Measurement Methods and Results - Part 2 : Basic method for the determination of repeatability and reproducibility of a standard measurement method*. International Organization for Standardization, Geneva, 1994.

[9] *ISO 5725-3: Accuracy (Trueness and Precision) of Measurement Methods and Results - Part 3 : Intermediate measures of the precision of a standard measurement method*. International Organization for Standardization, Geneva, 1994.

[10] *ISO 5725-4: Accuracy (Trueness and Precision) of Measurement Methods and Results - Part 4 : Basic methods for the determination of the trueness of a standard measurement method*. International Organization for Standardization, Geneva, 1994.

[11] *ISO 5725-5: Accuracy (Trueness and Precision) of Measurement Methods and Results - Part 5 : Alternative mthods for the determination of the precision of a standard measurement method*. International Organization for Standardization, Geneva, 1994.

[12] *ISO 5725-6: Accuracy (Trueness and Precision) of Measurement Methods and Results - Part 6 : Use in practice of accuracy values*. International Organization for Standardization, Geneva, 1994.

[13] *International vocabulary of metrology. Basic and general concepts and associated terms (VIM), JCGM 200*. ISO, 2008.

[14] Specsfs2008 benchmark. `http://www.spec.org/sfs2008/`, 2014.

[15] Transaction processing performance council. `http://www.tpc.org`, 2014.

[16] Guillermo A. Alvarez, Elizabeth Borowsky, Susie Go, Theodore H. Romer, Ralph Becker-szendy, Richard Golding, Arif Merchant, Mirjana Spasojevic, Alistair Veitch, John Wilkes, Guillermo A. Alvarez, Elizabeth Borowsky, Susie Go, Theodore H. Romer, Ralph Becker-szendy, Richard Golding, Arif Merchant, Mirjana Spasojevic, Alistair Veitch, and John Wilkes. Minerva: an automated resource provisioning tool for large-scale storage systems. *ACM Transactions on Computer Systems*, 19:483–518, 2001.

[17] Eric Anderson. Capture, conversion, and analysis of an intense nfs workload. In *Proccedings of the 7th USENIX Conference on File and Storage Technologies*, FAST'09, pages 139–152, 2009.

[18] Eric Anderson, Michael Hobbs, Kimberly Keeton, Susan Spence, Mustafa Uysal, and Alistair Veitch. Hippodrome: Running circles around storage administration. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, pages 175–188, Berkeley, CA, USA, 2002. USENIX Association.

[19] Eric Anderson, Mahesh Kallahalla, Mustafa Uysal, and Ram Swaminathan. Buttress: A toolkit for flexible and high fidelity i/o benchmarking. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, FAST'04, pages 45–58, 2004.

[20] Eric Anderson, Susan Spence, Ram Swaminathan, Mahesh Kallahalla, and Qian Wang. Quickly finding near-optimal storage designs. *ACM Trans. Comput. Syst.*, 23:337–374, 2005.

[21] Matt Blaze. Nfs tracing by passive network monitoring. In *Proceedings of the USENIX Winter 1992 Technical Conference*, pages 333–343, Berkeley, CA, USA, 1992. USENIX Association.

[22] Daniel P Bovet and Marco Cesati. *Understanding the Linux kernel*. "O'Reilly Media, Inc.", 2005.

[23] Tim Bray. Bonnie file system benchmark. `http://www.textuality.com/bonnie/`.

[24] Richard K Burdick, Connie M Borror, and Douglas C Montgomery. *Design and analysis of gauge R&R studies: Making decisions with confidence intervals in random and mixed ANOVA models*, volume 17. SIAM, 2005.

[25] Horst Czichos, Tetsuya Saito, and Leslie E Smith. *Springer handbook of metrology and testing*. Springer-Verlag, Berlin, 2nd edition, 2011.

[26] Bureau International des Poids et Mesures, Commission électrotechnique internationale, and Organisation internationale de normalisation. *Guide to the Expression of Uncertainty in Measurement*. International Organization for Standardization, 1995.

[27] Daniel Ellard, Jonathan Ledlie, Pia Malkani, and Margo Seltzer. Passive NFS tracing of email and research workloads. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 203–216, Berkeley, CA, USA, 2003. USENIX Association.

[28] Daniel Ellard and Margo Seltzer. New NFS tracing tools and techniques for system analysis. In *Proceedings of the 17th USENIX conference on System administration*, pages 73–86, Berkeley, CA, USA, 2003. USENIX Association.

[29] Eurachem. *The fitness for purpose of analytical methods: A laboratory guide to method validation and related topics*. LGC Teddington (UK), 1998.

[30] Dror G. Feitelson. *Workload Modeling for Computer Systems Performance Evaluation.* Cambridge University Press, 2015.

[31] Eduarda Filipe, Olivier Pellegrino, Antonio Carlos Baratto, Sérgio Pinheiro de Oliveira, and Victor Manuel Loayza Mendoza. *Vocabulário Internacional de Metrologia– Conceitos fundamentais e gerais e termos associados (VIM 2012)*. Inmetro, Brasil, 1st edition, 2012.

[32] Brendan Gregg. *Systems Performance: Enterprise and the Cloud*, chapter File Systems. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition, 2013.

[33] Brendan Gregg. *Systems Performance: Enterprise and the Cloud*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition, 2013.

[34] Christopher R Hertel. *Implementing CIFS: the common Internet file system*. Prentice Hall Professional, 2004.

[35] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *ACM Trans. Comput. Syst.*, 6(1):51–81, February 1988.

[36] Werner Hässelbarth. *EUROLAB Technical Report No. 1/2006, Guide to the Evaluation of Measurement Uncertainty for Quantitative Test Results*, chapter Annex A1. Eurolab, 2006.

[37] Werner Hässelbarth. *EUROLAB Technical Report No. 1/2006, Guide to the Evaluation of Measurement Uncertainty for Quantitative Test Results*, chapter Estimation of Measurement Uncertainties using Within-laboratory Validation and Quality Control Data. Eurolab, 2006.

[38] Werner Hässelbarth. *EUROLAB Technical Report No. 1/2006, Guide to the Evaluation of Measurement Uncertainty for Quantitative Test Results*. Eurolab, 2006.

[39] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley New York, 1991.

[40] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*, chapter Monitors. Wiley New York, 1991.

[41] Nikolai Joukov, Timothy Wong, and Erez Zadok. Accurate and efficient replaying of file system traces. In *FAST'05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, pages 337–350, Berkeley, CA, USA, 2005. USENIX Association.

[42] Jeffrey Katcher. Postmark: A new file system benchmark. tech. rep. tr3022, network appliance. `http://www.netapp.com/tech_library/3022.html`, 1997.

[43] Zachary Kurmas, Kimberley Keeton, and Kenneth M. Mackenzie. Synthesizing representative I/O workloads using iterative distillation. In *Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003. 11th IEEE/ACM International Symposium on*, pages 6–15. IEEE, 2003.

[44] Irma Lavagnini and Franco Magno. A statistical overview on univariate calibration, inverse regression, and detection limits: Application to gas chromatography/mass spectrometry technique. *Mass spectrometry reviews*, 26(1):1–18, 2007.

[45] Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems*. EPFL Press, Lausanne, Switzerland, 2010.

[46] Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems*, chapter Methodology. EPFL Press, Lausanne, Switzerland, 2010.

[47] Jialin Li, Naveen Kr. Sharma, Dan R. K. Ports, and Steven D. Gribble. Tales of the tail: Hardware, os, and application-level sources of tail latency. In *Proceedings of the 5th ACM Symposium on Cloud Computing*, pages 9:1–9:14, New York, NY, USA, 2014. ACM.

[48] Michael P. Mesnier, Matthew Wachs, Raja R. Sambasivan, Julio Lopez, James Hendricks, Gregory R. Ganger, and David O'Hallaron. //TRACE: parallel trace replay with approximate causal events. In *Proceedings of the 5th USENIX conference on File and Storage Technologies*, pages 153–167, Berkeley, CA, USA, 2007. USENIX Association.

[49] Lily Mummert and Mahadev Satyanarayanan. Long term distributed file reference tracing: Implementation and experience. Technical report, DTIC Document, 1994.

[50] Jiawu Chen Ningning Zhu and Tzi-Cker Chiueh. TBBT: Scalable and accurate trace replay for file server evaluation. In *Proceedings of the 4th USENIX Conference on File and Storage Technologies*, pages 323–336. USENIX Association, 2005.

[51] William D Norcott and Don Capps. Iozone filesystem benchmark. `www.iozone.org`, 2006.

[52] John K. Ousterhout, Hervé; Da Costa, David Harrison, John A. Kunze, Mike Kupfer, and James G. Thompson. A trace-driven analysis of the UNIX 4.2 BSD file system. In *Proceedings of the tenth ACM symposium on Operating systems principles*, pages 15–24, New York, NY, USA, 1985. ACM Press.

[53] Thiago Emmanuel Pereira, Francisco Vilar Brasileiro, and Livia Sampaio. File system trace replay methods through the lens of metrology. In *Proceedings of the 32nd IEEE International Conference on Massive Storage Systems and Technology (MSST)*. IEEE Computer Society, 2016.

[54] Thiago Emmanuel Pereira, Francisco Vilar Brasileiro, and Livia Sampaio. A study on the errors and uncertainties of file system trace capture methods. In *Proceedings of the 9th ACM International Systems and Storage Conference*. ACM, 2016.

[55] Thiago Emmanuel Pereira, Livia Sampaio, and Francisco Vilar Brasileiro. On the accuracy of trace replay methods for file system evaluation. In *Proceedings of the 2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 380–383. IEEE Computer Society, 2013.

[56] SG Rabinovich. *Measurement Errors and Uncertainties: Theory and Practice*, chapter General Information About Measurements. Springer, 3rd edition, 2005.

[57] Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. volume 10, pages 1–15, 1992.

[58] Bianca Schroeder, Adam Wierman, and Mor H. Balter. Open versus closed: a cautionary tale. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, page 239?252, Berkeley, CA, USA, 2006. USENIX Association.

[59] P Staubach, B Pawlowski, and B Callaghan. Nfs version 3 protocol specification. `http://tools.ietf.org/html/rfc1813`, 1995.

[60] V. Tarasov, S. Kumar, J. Ma, D. Hildebrand, A. Povzner, G. Kuenning, and E. Zadok. Extracting flexible, replayable models from large block traces. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, pages 1–9, Berkeley, CA, USA, 2012. USENIX Association.

[61] Vasily Tarasov. Multi-dimensional workload analysis and synthesis for modern storage systems. Dissertation proposal, Stony Brook University, 2013.

[62] Vasily Tarasov, Saumitra Bhanage, Erez Zadok, and Margo Seltzer. Benchmarking file system benchmarking: it *IS* rocket science. In *Proceedings of the 13th USENIX conference on Hot topics in operating systems*, Berkeley, CA, USA, 2011. USENIX Association.

[63] Mojtaba Tarihi, Hossein Asadi, and Hamid Sarbazi-Azad. DiskAccel: Accelerating Disk-Based Experiments by Representative Sampling. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Proc. SIGMETRICS'15, pages 297–308. ACM, 2015.

[64] Chandramohan A Thekkath, John Wilkes, and Edward D Lazowska. Techniques for file system simulation. *Software: Practice and Experience*, 24(11):981–999, 1994.

[65] Avishay Traeger, Erez Zadok, Nikolai Joukov, and Charles P. Wright. A nine year study of file system and storage benchmarking. *ACM Transactions on Storage(TOS'2008)*, 4(2):1–56, May 2008.

[66] Zev Weiss, Tyler Harter, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. Root: replaying multithreaded traces with resource-oriented ordering. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 373–387, New York, NY, USA, 2013. ACM.

Apêndice A

Outros aspectos de metrologia

Em metrologia, os objetos podem ser considerados, grosso modo, por medições ou validações [25][1]. Nessa categorização, medições têm como objetivo a obtenção, de maneira empírica, do valor de um mensurando associado a um objeto. Já validações (também chamadas de testes ou ensaios de medição) têm como objetivo a determinação das características de um objeto ou processo.

De antemão, as medições precisam de um método definido, que especifica como os instrumentos de medição serão usados para obter o mensurando. A definição completa de um método de medição inclui sua incerteza: um parâmetro que caracteriza a dispersão dos valores atribuídos a um mensurando.

Validações são necessárias quando ainda não temos um método definido, padronizado, para determinado propósito — este foi o caso dos estudo descritos nos capítulos anteriores desse documento. A metrologia fornece a base metodológica para conduzir as validações. A escolha entre os diversos métodos de validação dependem tanto do conhecimento dos fenômenos associados com a medição quanto do propósito esperado para o método. A Figura A.1, mostra uma visão geral de alguns métodos de validação. Em comum, todos os métodos começam com a definição do mensurando e com o levantamento das fontes de incerteza que afetam o método. Ao fim, todos os métodos levam ao cálculo da incerteza de medição associada como método sob validação.

Como mostra a Figura A.1, há basicamente 4 métodos de validação. Dentre esses, temos uma abordagem baseada no modelo matemático da lei de propagação de incertezas

[1] Em particular, tal como vista pela comunidade de química analítica, bastante influente em metrologia.

(GUM [26]) e três abordagens baseados em experimentação.

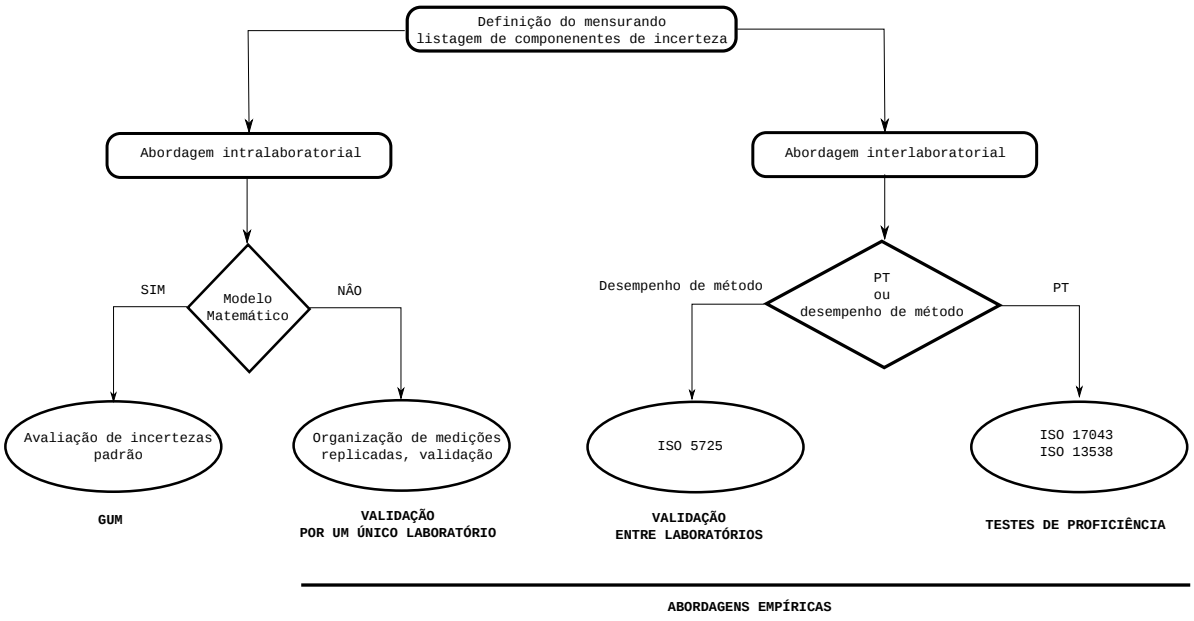


Figura A.1: Métodos para estimativas da incerteza.

Segundo o método GUM, um modelo matemático funcional do processo de medição precisa ser concebido. Esse modelo relaciona o mensurando com todas as outras quantidades das quais o mensurando depende, aí incluídas todas as fontes de erro. Em seguida, são calculadas incertezas-padrão para todas as entradas do modelo. Por fim, a incerteza combinada é calculada com base na lei de propagação de incertezas.

Em muitos casos, definir um modelo matemático para o método de medição é difícil. Por exemplo, os processo físicos subjacentes que relacionam os instrumentos de medições e as fontes de erros podem ser desconhecidos. Os métodos baseados em experimentação são a alternativa nesse cenário.

Os métodos de validação empíricos se diferenciam pelos tipos de erro que consideram para definir a incerteza de medição. Os tipos de erro considerados na validação determinam como as medições que serão realizadas posteriormente, com o método validado, podem ser comparadas. Por exemplo, determina se podemos concluir que os resultados de duas medições são equivalentes ou diferentes.

Em nossas provas de conceito, aplicamos o método empírico de *validação por um único laboratório*. Como o nome aponta, nesse método, as estimativas de tendência de medição e precisão são obtidas organizando experimentos conduzidos em um único laboratório.

Embora seja possível quantificar os efeitos de muitas fontes de erro em um único laboratório, outras são geralmente consideradas agregando resultados de múltiplos laboratórios. Isso porque alguns fatores tais como o uso e desgaste de instrumentos de medição, o ambientes de teste (por exemplo, condições climáticas) e a experiência dos operadores dos instrumentos, são difíceis de manipular em um único laboratório.

Há uma diferença básica entre os dois métodos inter-laboratoriais. Na abordagem de *validação entre laboratórios*, descrita na série de padrões ISO 5725 [7–12], os dados de um conjunto de laboratórios são coletados de uma única vez e publicados. Qualquer outro laboratório capaz de conduzir as medições, pode usar os dados publicados como estimativa da incerteza. Por sua vez, a abordagem de *teste de proficiência* (**PT**) tem como objetivo, ao invés de reuso de dados coletados à priori, a verificação periódica do desempenho de um laboratório [38]. Assim, um laboratório que participa do esforço coletivo do teste de proficiência compara os resultados de suas medições com as medições feitas por outros laboratórios. Nesse método, o desvio padrão das medições feitas pelos laboratórios participantes do teste é visto como uma estimativa da incerteza do método.

Naturalmente, o esforço necessário para proceder com os métodos acima mencionados não é o mesmo. É fácil notar que os esforço coletivo, bem como de coordenação, para conduzir um teste de proficiência é maior do que o esforço para conduzir uma validação por um único laboratório. Na próxima seção introduzimos o conceito de precisão intermediária. Esse conceito é um meio-termo entre a validação por um único laboratório e as abordagens inter-laboratoriais. Considerando o estado-da-prática da avaliação de desempenho em sistemas de arquivos, explorar esse conceito seria um passo natural para avaliações mais complexas de reprodutibilidade, como discutimos na Conclusão (Capítulo 7) desse documento.

A.1 Condições intermediárias de precisão

Em metrologia, a precisão é uma medida da variabilidade das medições. Mesmo quando medições são realizadas com um mesmo instrumento sobre material/objetos idênticos (condições repetíveis), obtém-se resultados diferentes. Isso porque não é factível controlar todas as fontes de incerteza, portanto existirão erros aleatórios em qualquer medição.

Além desses fatores, que estabelecem um limite inferior para a incerteza da medição, há

ainda outros que podem também contribuir para a variabilidade dos resultados, entre esses fatores temos:

- **Tempo**: Se o intervalo de tempo entre medições sucessivas é longo ou curto.
- **Calibração**: Se o equipamento é calibrado ou não entre medições sucessivas.
- **Operador**: Se o mesmo operador ou se operadores diferentes manipulam o instrumento entre medições sucessivas.
- **Instrumento**: Se o mesmo instrumento é usados nas medições.

A influência desses fatores é maior do que a existente quando experimentos são conduzidos em condições repetíveis. Por essa razão, convencionou-se avaliar a precisão dos métodos de medição entre dois extremos: de condições repetíveis para condições reproduzíveis. Essas últimas incorporam variações nos fatores acima listados. O documento *ISO 5725-3* discute condições intermediárias, nais quais um ou mais dos fatores listados acima estão presentes.

Nesse documento *ISO*, o cálculo da precisão intermediária é definido em **M-fatores** ($M = 1, 2, 3$ ou 4), a depender das mudanças que acontecem em um ambiente de experimentação (tempo, calibração, operador ou instrumento). Temos $M = 1$ quando apenas um dos quatro fatores são considerados até $M = 4$, quando todos eles são considerados.

Com respeito ao fator **tempo**, medições conduzidas em intervalos curtos minimizam as mudanças provocadas em condições ambientais que não podem ser garantidas constantes, por exemplo, temperatura e umidade. Em contraste, medições conduzidas em intervalos longos procuram observar os efeitos que estas condições ambientais podem criar.

O fator **calibração** é considerado no cálculo da precisão intermediária em cenários nos quais o processo de calibração é aplicado em períodos regulares. Ou seja, não se trata daqueles métodos que requerem a calibração feita de maneira obrigatória para obter cada medição. Pretende-se, portanto, entender o impacto nos erros devido ao processo contínuo de perda de calibração.

O fator **operador** pode também incluir todo um time de operadores, caso a operação do método em questão empregue um operador para partes específicas do procedimento de medição. Assim, qualquer troca de pessoal na equipe ou ainda mudança de responsabilidades entre os membros da equipe é considerada como uma variação no fator **operador**.

O fator **equipamento**, para cálculo da precisão intermediária, é abrangente a ponto de incluir também partes ou componentes do instrumento de medição. Nesse ponto, o analista precisa atuar para decidir se a influência de um dado componente é importante o suficiente para que tenhamos um equipamento diferente.

A maneira mais simples de calcular a precisão intermediária para medições realizadas em um único laboratório consiste em conduzir uma conjunto de n medições para uma amostra, objeto (ou coleções de amostras/objetos caso eles sejam destruídos pelo processo de medição) com mudanças dos fatores entre a medições. É recomendado que n não seja menor que 15. Esse método pode não ser satisfatório quando comparado com outros métodos. De qualquer modo, em particular, é um método útil para estudar o efeito da mudanças no fator **tempo**, realizando medições sobre determinadas amostras em dias consecutivos ou para estudar o efeito da calibração entre medições consecutivas. Nesse caso, a precisão é dada por:

$$\sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(y_k - \bar{y})^2}$$

em que y_k é a k-ésima medição replicada e $\bar{y}$ é o valor médio de n medições replicadas.

Um método alternativo considera r grupos de medições, cada grupo contendo n resultados de experimentos replicados. Por exemplo, em um laboratório, um conjunto de r amostras podem ser, cada uma, medidas, em seguida, um dos fatores intermediários é mudado, e as r amostras medidas novamente. Até obter n medições para uma das r amostras. Cada grupo de n testes deve ser obtido de um mesmo material (ou um conjunto de amostras presumivelmente idênticas). É recomendado que o valor $t(n-1)$ não seja menor que 15. Nesse caso, a precisão intermediária é definida como:

$$\sqrt{\frac{1}{t(n-1)}\sum_{j=1}^{t}\sum_{k=1}^{n}(y_{jk} - \bar{y_j})^2}$$

em que y_{jk} é o resultado da k-ésima medição do j-ésimo material e $\bar{y_j}$ é o valor médio de n medições sobre o j-ésimo material.

Para o caso mais comum, no qual há dois testes para cada material, a equação acima é simplificada para:

$$\sqrt{\frac{1}{2t}\sum_{j=1}^{t}(y_{j1}-\bar{y_{j2}})^2}$$

Tendência

Apêndice B

Catálogo de medição para captura de rastros

Neste apêndice nós sumariamos os resultados da validação dos métodos de captura `strace` e `SystemTap`. Além da declaração do mensurando, da faixa de valores medidos e do ambiente experimental, a Tabela B.1 e a Tabela B.2 mostram os resultados da caracterização, calibração e incerteza combinada para os métodos `strace` e `SystemTap`, respectivamente.

Tabela B.1: Metrologia do método de captura `strace`. Precisão e Tendência são mostrados como percentuais relativos (com relação ao valor medido médio e ao valor de referência médio, respectivamente). A incerteza combinada é definida com nível de confiança de 95.5%. Mostramos os valores antes da calibração entre parênteses. Precisão, tendência e incerteza são apresentados para todos os níveis da faixa de medição.

Mensurando	Tempo de resposta do sistema de arquivos Ext4.
Faixa de medição	Nível da carga capturada varia de 1 até 4, com base no número de *threads* usadas pelo *microbenchmark*.
Ambiente experimental	Estação de trabalho Intel $E6550$ Core 2 Duo 2.33GHz, com 2 GB de memória principal, com sistema operacional Linux $2.6.32$-41. Esta máquina contém dois discos rígidos: um disco SATA de 7200 RPM e 8 MB de memória cache, e um disco SATA de 5400 RPM com 32 MB de memória cache.

Random read (RR)	
Característica da medição	
Precisão	A precisão relativa é 0.94%, 0.39%, 0.66% e 0.42%.
Tendência	A tendência relativa é 0.26%, 0.17%, 0.23% e 0.35%.
Seletividade	Seletiva com respeito à carga de trabalho de fundo.
Calibração	Não é necessária uma vez que o erro sistemático não é significativo.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.94%, 0.39%, 0.66% e 0.42%.
Incerteza da tendência	A incerteza relativa da tendência é 0.95%, 0.40%, 0.92% e 0.38%.
Incerteza combinada	A incerteza combinada é 2.67%, 1.12%, 2.26%, 1.12%.
Random write (RW)	
Característica da medição	
Precisão	A precisão relativa é 0.84%, 0.52%, 0.68% e 0.47%.
Tendência	A tendência relativa é 0.01%, 0.08%, 0.21% e 0.18%.
Seletividade	Seletiva com respeito à carga de trabalho de fundo.
Calibração	Não é necessária uma vez que o erro sistemático não é significativo.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.84%, 0.52%, 0.68% e 0.47%.
Incerteza da tendência	A incerteza relativa da tendência é 0.74%, 0.62%, 1.02% e 0.27%.
Incerteza combinada	A incerteza combinada é 2.23%, 1.61%, 2.45%, 1.08%.
Sequential read (SR)	
Característica da medição	
Precisão	A precisão relativa é 0.76%, 1.80%, 0.76% e 0.69%.
Tendência	A tendência relativa é 618.47%, 575.14%, 543.41% e 536.89%.
Seletividade	Não é seletiva com respeito a carga de trabalho de fundo.
Calibração	A curva de calibração é dado por $73.51b_1 - 161.96$.
Incerteza da medição	

Incerteza da precisão	A incerteza relativa da precisão é 0.07%, 0.17%, 0.07% e 0.06% (0.76%, 1.80%, 0.76% e 0.69%).
Incerteza da tendência	A incerteza relativa da tendência é 4.29%, 4.09%, 4.62% e 4.33% (618.48%, 575.16%, 543.43% e 536.91%).
Incerteza combinada	A incerteza combinada é 8.58%, 8.19%, 9.24% e 8.65% (1236.96%, 1150.31%, 1086.85% e 1073.81%.)
Sequential write (SW)	
Característica da medição	
Precisão	A precisão relativa é 0.17%, 0.15%, 0.18% e 0.15%.
Tendência	A tendência relativa é 17.58%, 16.13%, 15.77% e 15.57%.
Seletividade	Não é seletiva com respeito a carga de trabalho de fundo.
Calibração	A curva de calibração é dado por $-0.37b_1 + 158$
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.53%, 0.45%, 0.54% e 0.46% e (0.17%, 0.15%, 0.18% e 0.15%).
Incerteza da tendência	A incerteza relativa da tendência é 0.21%, 0.77%, 0.23% e 0.65% (17.58%, 16.13%, 15.77% e 15.57%).
Incerteza combinada	A incerteza combinada é 1.13%, 1.79%, 1.17% e 1.59% (35.15%, . 32.25%, 31.54%, 31.14%).

Tabela B.2: Metrologia do método de captura `SystemTap`. Precisão e tendência são mostrados como percentuais relativos (com relação ao valor medido médio e ao valor de referência médio, respectivamente). A incerteza combinada é definado com nível de confiança de 95.5%. Mostramos os valores antes da calibração entre parenteses. Precisão, tendência e precisão são apresentados para todos os níveis da faixa de medição.

Mensurando	Tempo de resposta do sistema de arquivos Ext4.
Faixa de medição	Nível da carga capturada varia de 1 até 4, com based no número de *threads* usadas pelo *microbenchmark*.

Ambiente experimental	Estação de trabalho Intel $E6550$ Core 2 Duo 2.33GHz, com 1.92 GB de memória principal, com sistema operacional Linux $2.6.32$-41. Esta máquina contém dois discos rígidos: um disco SATA de 7200 RPM e 8 MB de memória cache, e um disco SATA de 5400 RPM com 32 MB de memória cache.

Random read (RR)

Característica da medição	
Precisão	A precisão relativa é 0.82%, 0.44%, 0.68% e 0.11%.
Tendência	A tendência relativa é 0.22%, 0.15%, 0.07% e 0.01%.
Seletividade	Seletiva com respeito a carga de trabalho de fundo.
Calibração	Não é necessária, uma vez que o erro sistemático não é significativo.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.22%, 0.15%, 0.07% e 0.01%.
Incerteza da tendência	A incerteza relativa da tendência é 0.94%, 0.39%, 0.90% e 0.13%.
Incerteza combinada	A incerteza combinada é 2.49%, 1.18%, 2.25% e 0.34%.

Random write (RW)

Característica da medição	
Precisão	A precisão relativa é 0.84%, 0.51%, 1.00% e 0.18%.
Tendência	A tendência relativa é 0.01%, 0.00%, 0.05% e 0.01%.
Seletividade	Seletiva com respeito a carga de trabalho de fundo.
Calibração	Não é necessária, uma vez que o erro sistemático não é significativo.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.84%, 0.51%, 1.00% e 0.18%.
Incerteza da tendência	A incerteza relativa da tendência é 0.74%, 0.62%, 1.00% e 0.19%.
Incerteza combinada	A incerteza combinada é 2.23%, 1.60%, 2.82% e 0.52%.

Sequential read (SR)

Característica da medição	
Precisão	A precisão relativa é 3.16%, 2.59%, 2.81% e 2.57%.

Tendência	A tendência relativa é 14.27%, 9.51%, 8.63% e 8.65%.
Seletividade	Seletiva com respeito a carga de trabalho de fundo.
Calibração	A curva de calibração é dado por $4.97b_1 - 5.38$
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.72%, 0.57%, 0.61% e 0.56% (3.16%, 2.59%, 2.81% e 2.57%).
Incerteza da tendência	A incerteza relativa da tendência é 4.31%, 4.11%, 4.62% e 4.33% (14.90%, 10.36%, 9.79% e 9.67%).
Incerteza combinada	A incerteza combinada é 8.73%, 8.29%, 9.32% e 8.72% (30.46%, 21.34%, 20.36% e 20.01%).

Sequential write (SW)

Característica da medição	
Precisão	A precisão relativa é 0.22%, 0.19%, 0.20% e 0.19%.
Tendência	A tendência relativa é 0.86%, 0.39%, 0.34% e 0.27%.
Seletividade	Seletiva com respeito a carga de trabalho de fundo.
Calibração	Não é necessária, uma vez que o erro sistemático não é significativo.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.22%, 0.19%, 0.20% e 0.19%.
Incerteza da tendência	A incerteza relativa da tendência é 0.88%, 0.45%, 0.40% e 0.33%.
Incerteza combinada	A incerteza combinada é 1.82%, 0.97%, 0.90% e 0.75%.

Apêndice C

Catálogo de medição para reprodução de rastros

Nesta seção nós sumariamos os resultados da validação dos métodos de reprodução baseados em eventos e em compilação. Além da declaração do mensurando, da faixa de valores medidos e do ambiente experimental, a Tabela C.1 e a Tabela C.2 mostram os resultados da caracterização e do cálculo da incerteza combinada para os métodos baseados em compilação e em eventos, respectivamente.

Tabela C.1: Metrologia do método de reprodução baseado em compilação. Precisão e tendência são mostrados como percentuais relativos (com relação ao valor medido médio e ao valor de referência médio, respectivamente). A incerteza combinada é definido com nível de confiança de 95.5%. Precisão, tendência e incerteza são apresentados para todos os níveis da faixa de medição.

Mensurando	Tempo de resposta do sistema de arquivos Ext4.
Faixa de medição	Nível da carga capturada varia de 1 até 4, com base no número de *threads* usadas pelo *microbenchmark*.
Ambiente experimental	Estação de trabalho Intel $E6550$ Core 2 Duo 2.33GHz, com 1.92 GB de memória principal, com sistema operacional Linux $2.6.32$-41. Esta máquina contém dois discos rígidos: um disco SATA de 7200 RPM e 8 MB de memória cache, e um disco SATA de 5400 RPM com 32 MB de memória cache.

Random read (RR)	
Característica da medição	
Precisão	A precisão relativa é 0.13%, 0.24%, 0.23% e 0.07% para a política `FS` e 0.11%, 0.10%, 0.21% e 0.07% para a política `Temporal`.
Tendência	A tendência relativa é 0.94%, 0.39%, 0.14% e 0.02% para a política `FS` e 0.92%, 0.24%, 3.67% e 0.01% para a política `Temporal`.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.13%, 0.24%, 0.23% e 0.07% para a política `FS` e 0.11%, 0.10%, 0.21% e 0.07% para a política `Temporal`.
Incerteza da tendência	A incerteza relativa da tendência é 1.31%, 0.53%, 0.90% e 0.13% para a política `FS` e 1.29%, 0.44%, 3.78% e 0.13%
Incerteza combinada	A incerteza combinada é 2.63%, 1.16, 1.85% e 2.59%, 0.89%, 7.56% e 0.29% para a política `Temporal`.
Random write (RW)	
Característica da medição	
Precisão	A precisão relativa é 0.19%, 0.41%, 0.50% e 0.24% para a política `FS` e 0.25%, 0.37%, 0.51% e 0.14% para a política `Temporal`.
Tendência	A tendência relativa é 0.36%, 0.06%, 0.35% e 0.15% para a política `FS` e 0.44%, 0.16%, 0.39% e 0.07% para a política `Temporal`.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.19%, 0.41%, 0.50% e 0.24% para a política `FS` e 0.25%, 0.37%, 0.51% e 0.14% para a política `Temporal`.
Incerteza da tendência	A incerteza relativa da tendência é para a política `FS` é 0.82%, 0.62%, 1.06% e 0.25% e 0.85%, 0.64%, 1.07% e 0.20% para a política `Temporal`.

Incerteza combinada	A incerteza combinada é 1.67%, 1.48%, 2.34% e 0.69% para a política `FS` e 1.77%, 1.47%, 2.36% e 0.49% para a política `Temporal`.

Sequential read (SR)

Característica da medição	
Precisão	A precisão relativa é 1.03%, 0.68%, 0.89% e 0.43% para a política `FS` e 0.75%, 0.42%, 0.53% e 0.52% para a política `Temporal`.
Tendência	A tendência relativa é 2.78%, 8.41%, 9.52% e 9.15% para a política `FS` e 1.95%, 8.65%, 7.99% e 7.40%para a política `Temporal`.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 1.03%, 0.68%, 0.89% e 0.43% para a política `FS` e 0.75%, 0.42%, 0.53% e 0.52% para a política `Temporal`.
Incerteza da tendência	A incerteza relativa da tendência é 5.10%, 9.35%, 10.58% e 10.12% para a política `FS` e 4.70%, 9.57%, 9.23% e 8.57% para a política `Temporal`.
Incerteza combinada	A incerteza combinada é 10.41%, 18.75%, 21.23% e 20.26% para a política `FS` e 9.52%, 19.16%, 18.48% e 17.17%para a política `Temporal`.

Sequential write (SW)

Característica da medição	
Precisão	A precisão relativa é 0.11%, 0.20%, 0.12% e 0.09% para a política `FS` e 0.12%, 0.29%, 0.19% e 0.11% para a política `Temporal`.
Tendência	A tendência relativa é 0.41%, 0.01%, 0.14% e 0.37% para a política `FS` e 0.46%, 0.14%, 0.26% e 0.27% para a política `Temporal`.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.11%, 0.20%, 0.12% e 0.09% para a política `FS` e 0.12%, 0.29%, 0.19% e 0.11% para a política `Temporal`

Incerteza da tendência	A incerteza relativa da tendência é 0.45%, 0.21%, 0.25% e 0.42%para a política `FS` e 0.50%, 0.26%, 0.34%, 0.33% para a política `Temporal`.
Incerteza combinada	A incerteza combinada é 0.93%, 0.58%, 0.55% e 0.85% para a política `FS` e 1.02%, 0.77%, 0.77% e 0.68% para a política `Temporal`.

Tabela C.2: Metrologia do método de reprodução baseado em eventos. Precisão e tendência são mostrados como percentuais relativos (com relação ao valor medido médio e ao valor de referência médio, respectivamente). A incerteza combinada é definido com nível de confiança de 95.5%. Precisão, tendência e incerteza são apresentados para todos os níveis da faixa de medição.

Mensurando	Tempo de resposta do sistema de arquivos Ext4.
Faixa de medição	Nível da carga capturada varia de 1 até 4, com base no número de *threads* usadas pelo *microbenchmark*.
Ambiente experimental	Estação de trabalho Intel $E6550$ Core 2 Duo 2.33GHz, com 1.92 GB de memória principal, com sistema operacional Linux $2.6.32$-41. Esta máquina contém dois discos rígidos: um disco SATA de 7200 RPM e 8 MB de memória cache, e um disco SATA de 5400 RPM com 32 MB de memória cache.

Random read (RR)

Característica da medição

Precisão	A precisão relativa é 0.11%, 0.24%, 0.33% e 0.13%para a política `FS` e 0.14%, 0.30%, 0.95% e 0.12% para a política `Temporal`.
Tendência	A tendência relativa é 1.04%, 0.40%, 0.65% e 0.04% para a política `FS` e 1.04%, 0.13%, 1.68% e 0.00% para a política `Temporal`.

Incerteza da medição

Incerteza da precisão	A incerteza relativa da precisão é 0.11%, 0.24%, 0.33% e 0.13%para a política `FS` e 0.14%, 0.30%, 0.95% e 0.12% para a política `Temporal`.
Incerteza da tendência	A incerteza relativa da tendência é 1.39%, 0.54%, 1.10% e 0.14% para a política `FS` e 1.39%, 0.39%, 1.91% e 0.13% para a política `Temporal`.
Incerteza combinada	A incerteza combinada é 2.78%, 1.18%, 2.30%, 0.37% para a política `FS` e 2.78%, 0.97%, 4.25% e 0.35% para a política `Temporal`.

Random write (RW)

Característica da medição

Precisão	A precisão relativa é 0.23%, 0.37%, 0.53% e 0.14% para a política `FS` e 0.18%, 0.62%, 0.95% e 0.15% para a política `Temporal`.
Tendência	A tendência relativa é 0.09%, 0.41%, 0.06% e 0.00% para a política `FS` e 0.26%, 0.07%, 5.21% e 0.12% para a política `Temporal`.

Incerteza da medição

Incerteza da precisão	A incerteza relativa da precisão é 0.23%, 0.37%, 0.53% e 0.14% para a política `FS` e 0.18%, 0.62%, 0.95% e 0.15% para a política `Temporal`.
Incerteza da tendência	A incerteza relativa da tendência é 0.74%, 0.74%, 1.00% e 0.19% para a política `FS` e 0.78%, 0.62%, 5.31% e 0.23% para a política `Temporal`.
Incerteza combinada	A incerteza combinada é 1.54%, 1.65%, 2.26% e 0.48% para a política `FS` e 1.59%, 1.75%, 10.78%, 0.54% para a política `Temporal`.

Sequential read (SR)

Característica da medição

Precisão	A precisão relativa é 0.79%, 1.52%, 3.53%, 1.21% para a política `FS` e 1.07%, 1.24%, 1.91%, 1.46% para a política `Temporal`.

Tendência	A tendência relativa é 24.18%, 18.19%, 19.56%, 16.16% para a política `FS` e 24.39%, 17.63%, 23.32%, 18.88% para a política `Temporal`.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão para a política `FS` é 0.79%, 1.52%, 3.53%, 1.21% para a política `FS` e 1.07%, 1.24%, 1.91%, 1.46% para a política `Temporal`.
Incerteza da tendência	A incerteza relativa da tendência é 24.56%, 18.64%, 20.10% e 16.73% para a política `FS` e 24.77%, 18.10%, 23.77% e 19.37% para a política `Temporal`.
Incerteza combinada	A incerteza combinada é 49.14%, 37.41%, 40.80%, 33.53% para a política `FS` e 49.58%, 36.28%, 47.69%, 38.85% para a política `Temporal`.
Sequential read (SW)	
Característica da medição	
Precisão	A precisão relativa é 0.22%, 0.23%, 0.19% e 0.19% para a política `FS` e 0.21%, 0.22%, 0.23%, 0.23% para a política `Temporal`.
Tendência	A tendência relativa é 2.23%, 2.12%, 2.27% e 2.45% para a política `FS` e 2.43%, 1.96%, 2.27% e 2.51% para a política `Temporal`.
Incerteza da medição	
Incerteza da precisão	A incerteza relativa da precisão é 0.22%, 0.23%, 0.19% e 0.19% para a política `FS` e 0.21%, 0.22%, 0.23%, 0.23% para a política `Temporal`.
Incerteza da tendência	A incerteza relativa da tendência é 2.24%, 2.13%, 2.27% e 2.46% para a política `FS` e 2.44%, 1.97%, 2.28% e 2.51% para a política `Temporal`.
Incerteza combinada	A incerteza combinada é 4.50%, 4.29%, 4.56%, 4.93% para a política `FS` e 4.89%, 3.96%, 4.58% e 5.05% para a política `Temporal`.

Apêndice D

Testes de normalidade para as medições de captura de rastros

Mostramos nas Figuras D.1, D.2, e D.3 abaixo, os quantis para as medições de referência de captura, bem como as medições feitas com as ferramentas `strace` e `SystemTap`.

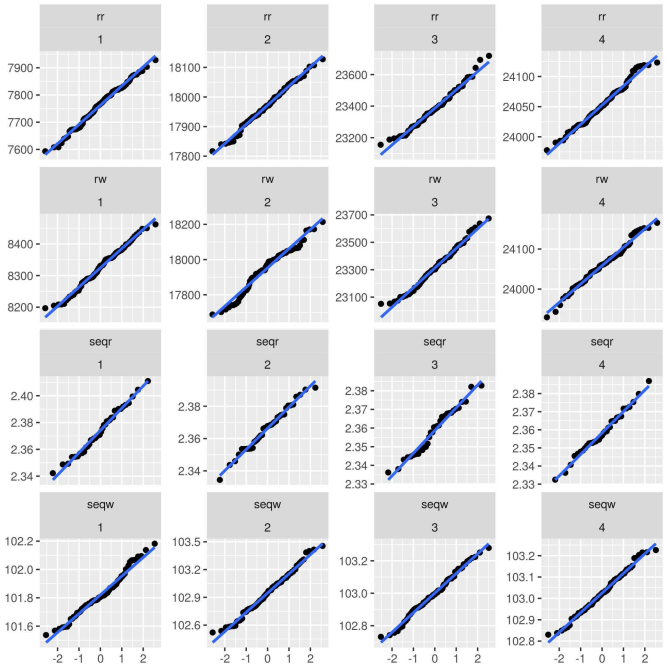


Figura D.1: Quantis para as medições de referência de captura, para todas a combinações de nível e tipo de carga de trabalho

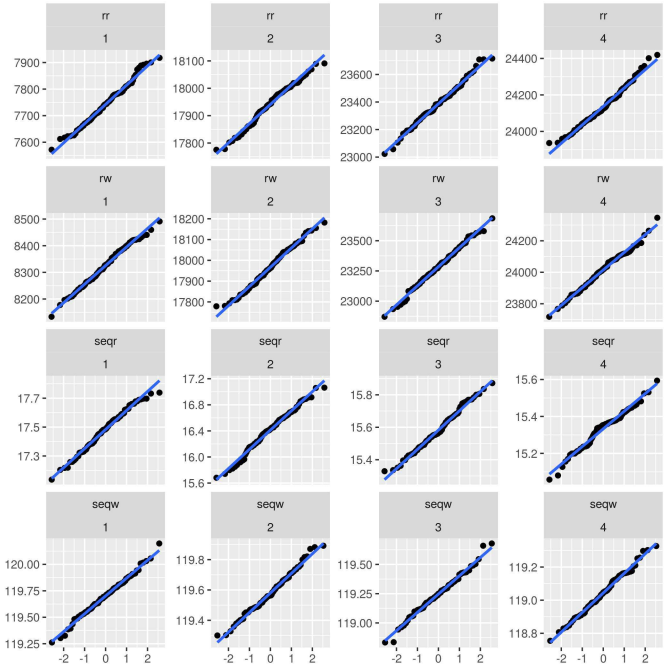


Figura D.2: Quantis para as medições com a ferramenta de captura `strace`, para todas a combinações de nível e tipo de carga de trabalho

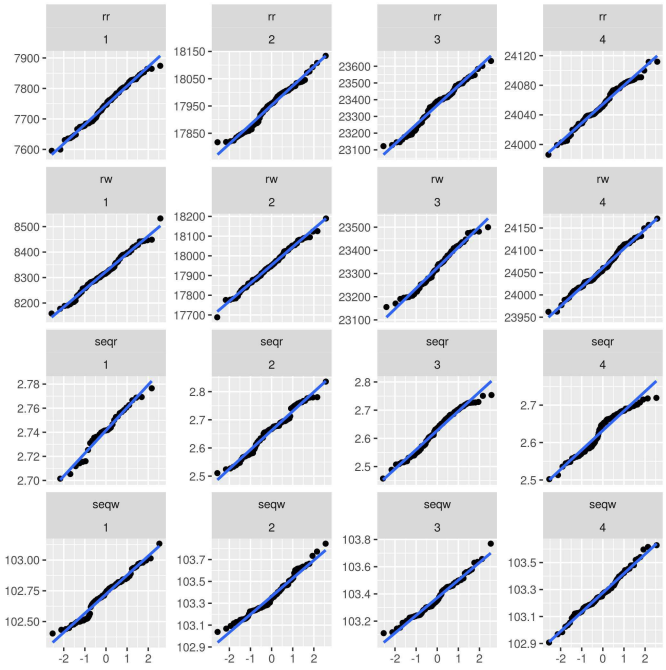


Figura D.3: Quantis para as medições com a ferramenta de captura `SystemTap`, para todas a combinações de nível e tipo de carga de trabalho

Apêndice E

Testes de normalidade para as medições de reprodução dos rastros gerados pelo microbenchmark

Mostramos nas figuras abaixo os quantis para as medições de reprodução com as ferramentas baseadas em compilação e em eventos. As Figuras E.2 e E.1 mostram os quantis para as medições com a ferramenta de reprodução baseada em compilação para as políticas de ordenação `Temporal` e `FS`, respectivamente. Por sua vez, as Figuras E.3 e E.4 mostram os quantis para as medições com a ferramenta de reprodução baseada em compilação para as políticas de ordenação `Temporal` e `FS`, respectivamente.

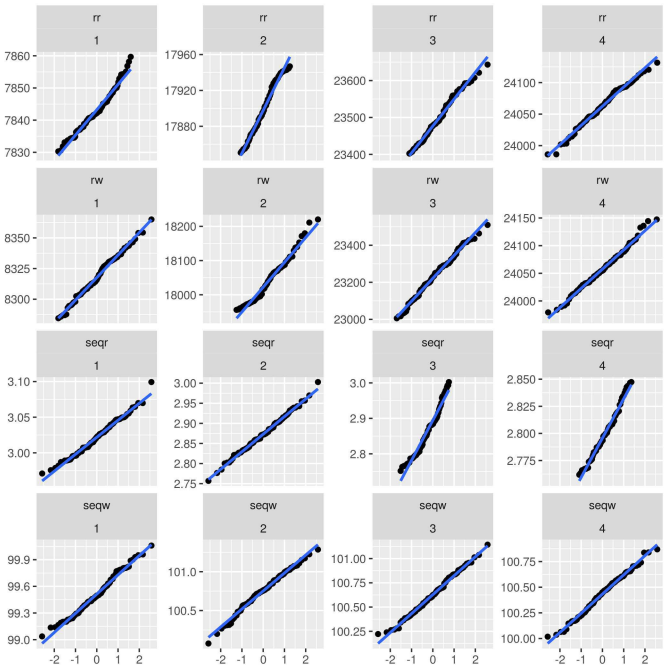


Figura E.1: Quantis para as medições com a ferramenta de reprodução baseada em eventos todas a combinações de nível e tipo de carga de trabalho, com a política de ordenação `FS`.

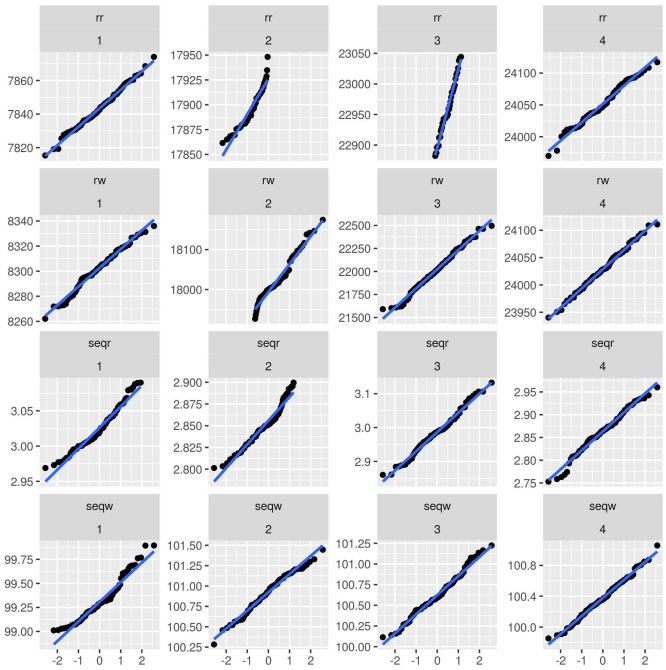


Figura E.2: Quantis para as medições com a ferramenta de reprodução baseada em eventos todas a combinações de nível e tipo de carga de trabalho, com a política de ordenação `Temporal`.

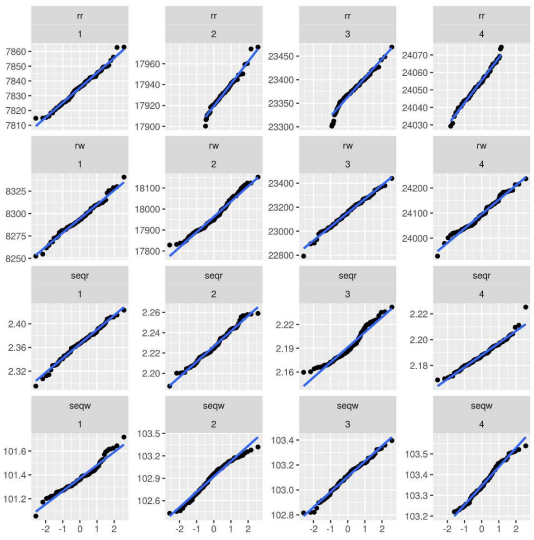


Figura E.3: Quantis para as medições com a ferramenta de reprodução baseada em compilação para todas a combinações de nível e tipo de carga de trabalho, com a política de ordenação `FS` em combinação com a política de temporização `fullspeed`.

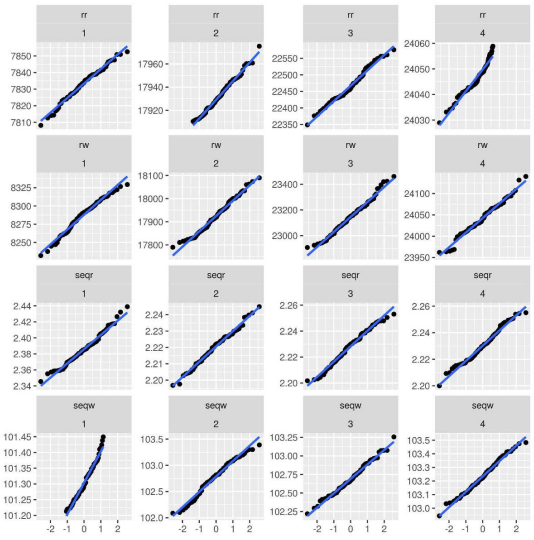


Figura E.4: Quantis para as medições com a ferramenta de reprodução baseada em compilação para todas a combinações de nível e tipo de carga de trabalho, com a política de ordenação `Temporal` em combinação com a política de temporização `fullspeed`.

Apêndice F

Testes de normalidade para as medições de reprodução dos rastros gerados pelo macrobenchmark

Mostramos nas Figuras F.1 e F.2 abaixo os quantis para as medições de reprodução com as ferramentas baseadas em compilação e em eventos, para as cargas de trabalho **read** e **write**, com a política de ordenação `FS` em combinação com a política de temporização `fullspeed`, respectivamente.

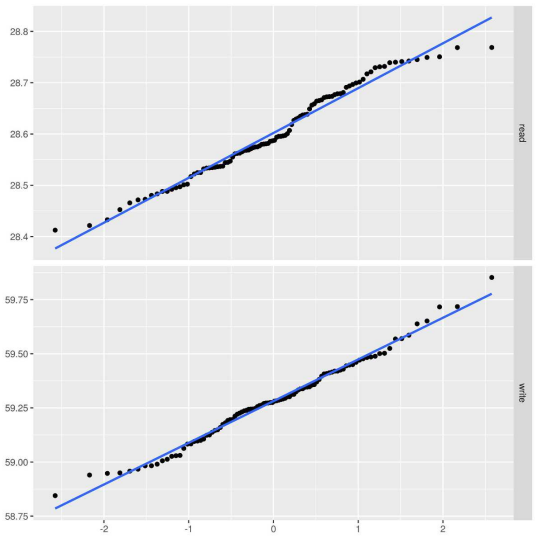


Figura F.1: Quantis para as medições com a ferramenta de reprodução baseada em compilação para as cargas de trabalho **read** e **write**, com a política de ordenação `FS` em combinação com a política de temporização `fullspeed`.

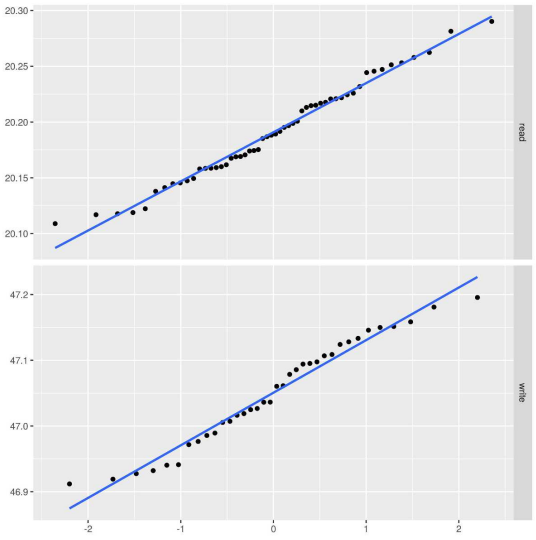


Figura F.2: Quantis para as medições com a ferramenta de reprodução baseada em eventos para as cargas de trabalho **read** e **write**, com a política de ordenação `FS` em combinação com a política de temporização `fullspeed`.