UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ANÁLISE DE DESEMPENHO DE SOLUÇÕES PARA ARMAZENAMENTO ESTÁVEL DE DADOS

Erick B. PASSOS

Campina Grande - PB
Julho - 2003

Análise de Desempenho de Soluções para Armazenamento Estável de Dados

Erick B. Passos

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

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Sistemas Distribuídos

Francisco V. Brasileiro

Walfredo Cirne

Campina Grande, Paraíba, Brasil

Resumo

Aplicações tolerantes a falhas freqüentemente necessitam gravar dados de forma estável para garantir uma computação confiável. Para dar suporte a esse requisito, diversas formas de se estabilizar os dados têm sido propostas, como gravação síncrona em disco e replicação de dados pela rede. Ao mesmo tempo que diferem em arquitetura, essas propostas diferem em desempenho. Nesse trabalho estudamos dois problemas: i) A gravação estável de dados em si, explorando sua arquitetura e suas implementações de fato, e ii) A análise de desempenho dessas soluções, ou seja, as variáveis e formas de se comparar serviços de armazenamento estável de dados. Fizemos um trabalho abrangendo o estudo de soluções, estudo de métodos de avaliação, a implementação de uma solução baseada em replicação de dados pela rede, e a realização de experimentos para avaliar o desempenho dessa implementação comparada a soluções consagradas. Com os resultados obtidos nos experimentos comprovamos a viabilidade das soluções baseadas em replicação de dados e identificamos uma série de pontos críticos na utilização de soluções de armazenamento estável de dados.

Abstract

Fault tolerant applications often need to stabilize data in order to provide dependable computation. Ideas for services for data stabilizing abound, for example: synchronous operations on magnetic disks, data replication through the network, or even non-volatile RAM. This different ideas lead to different performance fingerprints. In this dissertation we focus on two problems: i) The stable storage problem, with architecture and implementations, and ii) The performance evaluation of this kind of service. Our work comprehends the study of the stable storage problem and solutions, study of methods for performance evaluation of this solutions, implementation of a network data-replication based solution and the performance evaluation of this solution with the classic ones, based on synchronous data write procedures. With the results, we assure the viability of our implementation and identify a series of trade offs in stable storage systems.

Agradecimentos

Ao meus orientadores, Fubica, por sua orientação durante o programa de mestrado, sempre objetiva e motivante, e Walfredo, pelas suas "sacações"e comentários, sempre elucidativos.

À Aninha e Vera pela paciência no atendimento e pela simpatia.

À meus amigos do *Recanto Zen*, Amâncio, Lauro, Érika, Cabelo, Cláudia, Ladjane, e Juliana, pelos momentos descontraídos e pela companhia.

Gostaria também de fazer um agradecimento especial a meus pais que sempre me deram o suporte, seja ele logístico ou emocional, em todas as fases de minha educação.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Sistemas computadorizados são cada vez mais utilizados para guardar, processar e distribuir informação. Com o crescimento da demanda, as pessoas passam a depender desses sistemas para fazer compras, viajar, e em certos casos até confiam a vida a esses sistemas, como em sistemas de controle de vôo e metrôs. Falhas nesses sistemas podem trazer perda de tempo e dinheiro, problemas legais ou mesmo risco de vida. De alguma forma esses sistemas precisam evitar a ocorrência de falhas tornando-se cada vez mais confiáveis.

As falhas podem ocorrer tanto em *hardware*, por defeitos nos componentes ou uso fora de especificação[1], quanto em *software*, por imperfeições na implementação. Duas abordagens são usadas para tratar falhas: i) prevenção de falhas, onde se procura evitar que falhas ocorram; ii) tolerância a falhas, onde se procura contornar a existência das mesmas, sendo esta última nossa área de estudos.

1.1 Tolerância a Falhas

Podemos dizer que tolerância a falhas compreende o estudo das formas de detectar falhas em *hardware* e *software* e de manter o bom funcionamento do sistema mesmo na presença dessas falhas. Ao longo dos anos, uma série de serviços básicos tolerantes a falhas foi idealizada. Através desses serviços básicos, é possível se construir aplicações mais robustas.

Uma forma simples de se tolerar falhas em uma aplicação é através do uso de pontos de salvaguarda [Pan94], onde periodicamente o estado global do sistema é armazenado. No caso

[1] Sistema elétrico em sobrecarga por exemplo.

de uma falha ser detectada o sistema pode ser recuperado para um estado anterior consistente. Essa técnica é particularmente útil em casos de problemas que demoram um longo tempo para serem computados onde o prejuízo de uma reinicialização seria custoso demais. Para ser implementada, necessita-se de um serviço de armazenamento estável de dados para se salvar periodicamente as informações referentes ao estado do sistema.

Outra importante abstração para a tolerância a falhas em aplicações é o conceito de transação. Uma transação é um conjunto de operações que deve ser realizada de forma atômica, ou seja, existe uma ordem implícita e apenas dois resultados possíveis podem ser atingidos, ou a transação executa corretamente ou não executa de maneira alguma. Para garantir a estabilidade das informações referentes às transações já realizadas, esses sistemas também dependem de um serviço de armazenamento estável de dados.

Para essas duas categorias de aplicações tolerantes a falhas podemos ver que um serviço de armazenamento estável de dados é necessário para sua implementação. A forma mais comum de se prover esse serviço é através da utilização de equipamentos auxiliares de armazenamento tais como discos magnéticos. Discos têm sido utilizados por terem um alto grau de confiabilidade em comparação com outras tecnologias, bem como uma grande capacidade de armazenamento. As possíveis falhas que um disco pode sofrer são tratáveis através de técnicas relativamente simples. Mas, apesar de serem confiáveis, o desempenho desses discos não evolui no mesmo ritmo que outras tecnologias tais como redes de transmissão de dados e memória volátil.

Esse baixo desempenho relativo tem sido contornado pela a maioria das aplicações através do uso de *caches* locais em memória volátil para otimizar o acesso a disco. Aplicações que têm requisitos de desempenho fazem uso dessa *cache* enquanto aplicações que necessitam de garantias de estabilidade nas operações de gravação sofrem as penalidades por conta do baixo desempenho.

Em certos casos isso pode não vir a ser um problema pelo fato de nem toda aplicação tolerante a falhas ter grandes requisitos de desempenho. Entretanto, nos últimos anos temos visto um crescimento rápido nesses requisitos em sistemas baseados em transações, tais como sistemas *on-line*, por conta da grande quantidade de operações simultâneas. Além disso, a quantidade de dados carregando a informação de controle de uma transação ou o estado interno de um sistema aumentou consideravelmente. Isso tem motivado o estudo de técnicas

alternativas para prover estabilidade dos dados.

Como serviços de armazenamento de dados são providos basicamente através de hardware em conjunto com o sistema operacional [Pan94], sua *interface* para as aplicações é o conjunto de operações de gravação dos sistemas de arquivo. Os trabalhos recentes nessa área têm explorado duas abordagens para otimizar esse desempenho, sendo uma delas o projeto e implementação de novas formas de se estruturar os sistemas de arquivo [RO92; Rei; ext; xfs; jfs; MG99] e a outra é evitar o uso imediato do disco nas operações de gravação segura [CNC+96; BCPS02; LGG+91]

Neste nosso trabalho, implementamos um sistema de arquivos baseado em uma técnica de replicação de dados em memória remota, evitando as penalidades de performance obtidas pelo uso imediato de discos [BCPS02]. Essa implementação foi usada para se avaliar essa técnica de gravação estável em relação a outras abordagens nesse trabalho de avaliação de desempenho.

1.2 Avaliação de Desempenho

Partindo do ponto de vista do desenvolvedor de aplicações tolerantes a falhas surge um outro problema. Como comparar essas técnicas diferentes de se armazenar dados? Em sistemas de arquivo, variáveis como concorrência pelo uso do disco e padrão de carga[2] têm enorme influência sobre o resultado final em uma comparação de desempenho. Isso decorre do fato de diferentes formas de se organizar os dados e escalonar o uso do disco levam a diferentes padrões de comportamento do serviço.

A forma mais simples de se medir o desempenho desses sistema é através do uso de *micro benchmarks*, ou seja, aplicações que medem o desempenho dos sistemas de arquivo medindo o custo de cada operação individualmente repetidas vezes, com uma certa variação na granularidade, e fazendo um sumário dos resultados através de médias, variâncias e picos. Outra forma é submeter o serviço de armazenamento a uma carga padronizada[3] e comparar o comportamento deste no decorrer do processo de gravação e leitura dos dados dessa carga. Isso é a técnica chamada de *macro benchmark*.

[2] Quantidade, granularidade e velocidade com que os dados chegam ao sistema de arquivos.

[3] Carga de trabalho que seja parecida com o comportamento de certas aplicações como sistemas de bancos de dados ou sistemas que fazem salvaguardas.

Apesar de estar sendo bastante explorada pela literatura, essa área ainda carece de um estudo comparativo mais completo que utilize técnicas de *micro* e *macro benchmarks* para obter informações sobre as atuais implementações desses serviços. As comparações existentes costumam utilizar apenas uma dessas técnicas ou então comparar dois serviços similares[4].

Ferramentas de *micro benchmark* são boas quando se deseja obter uma idéia geral do comportamento do serviço de armazenamento estável de dados, mas só *macro benchmarks* e aplicações reais são representativas em relação aos padrões de carga que ocorrem em um ambiente de produção de um sistema tolerante a falhas.

Nosso trabalho procurou comparar as principais técnicas de implementação de serviços de armazenamento estável através da utilização de uma aplicação real, que utiliza armazenamento estável de dados, para se obter um padrão de comportamento desses serviços. Também comparamos essas soluções através das técnicas tradicionais de avaliação de desempenho (*micro* e *macro benchmarks*) de sistemas de armazenamento.

Nossa contribuição para a área é ajudar administradores de sistemas que precisam encontrar a solução mais adequada para seu contexto, bem como descobrir a forma pela qual variáveis tais como concorrência afetam o desempenho desses sistemas. Ao final da leitura deste trabalho, o leitor deverá ser capaz de entender o funcionamento de serviços de armazenamento estável, entender as diferentes formas de se avaliar seu desempenho e tolerância quanto a falhas e, por fim, poder responder às seguintes questões:

- Dados um contexto de uso[5] e um padrão de carga de uma aplicação[6], qual a técnica de armazenamento estável de dados mais eficiente em termos de custo e desempenho?
- Quais são os melhores parâmetros para se avaliar a qualidade de um serviço de armazenamento estável de dados sabendo quais são os requisitos da aplicação?

[4]Dois serviços de armazenamento estável que utilizam a mesma técnica para gravar os dados, seja esta disco, rede ou memória local não volátil.

[5]Modelo de sistema, qualidade e desempenho do equipamento de disco e rede de transmissão de dados.

[6]Tamanho médio dos arquivos, frequência de gravação, variância do tamanho dos blocos gravados bem como seu tamanho, nível de concorrência observado, etc.

1.3 Estrutura da Dissertação

Nos próximos dois capítulos são discutidos os conceitos básicos e os trabalhos relacionados com este. No Capítulo 2, mostraremos a definição formal do que é um serviço de armazenamento estável e como esses serviços têm sido implementados nos diversos sistemas computacionais. No Capítulo 3, exploraremos os conceitos relativos à avaliação de desempenho de sistemas de armazenamento de dados e mostraremos como os parâmetros de comparação podem influir nos resultados obtidos.

No Capítulo 4, detalhamos as implementações de três sistemas de armazenamento estável de dados que foram utilizadas nos nossos experimentos. O Capítulo 5, traz a descrição formal dos testes que foram executados e uma análise detalhada dos dados obtidos dos mesmos. Nessa análise está nossa principal contribuição para esta linha de pesquisa.

O Capítulo 6 traz algumas observações finais e conclui o trabalho.

Capítulo 2

Armazenamento Estável de Dados

Aplicações tolerantes a falhas frequentemente necessitam armazenar dados de forma persistente. Um serviço de armazenamento estável visa fornecer essa abstração para o desenvolvedor, deixando-o livre para tratar outras questões inerentes à tolerância a falhas.

Neste capítulo, iremos descrever o problema do armazenamento estável de dados, e formalizar uma abstração de um serviço desse tipo que seja robusto. Iremos também discutir as principais tecnologias utilizadas para dar suporte à armazenamento estável de dados. Por fim, faremos uma exploração das abordagens usadas por sistemas operacionais modernos para prover esse serviço através da interface de armazenamento padrão, os sistemas de arquivo.

O objetivo deste capítulo é familiarizar o leitor com o problema em questão, ilustrando as principais dificuldades, as soluções encontradas e as abordagens mais utilizadas por sistemas reais.

2.1 Armazenamento Estável

Armazenamento estável é uma abstração de um serviço ideal para persistência de dados tolerante a falhas. Em outras palavras, assume-se que o sistema computacional oferece uma forma de armazenamento que preserva seu conteúdo mesmo na presença de falhas [Pan94]. Como na prática é impossível prever todas as falhas de *hardware* que podem ocorrer em um sistema computacional, serviços de armazenamento estável provêem uma forma de tolerância simplificada, ou seja, tolerando apenas certos tipos de falhas.

A forma mais comum de se prover um serviço de armazenamento estável é através da uti-

lização de dispositivos de armazenamento secundários não voláteis, tais como discos magnéticos. Apesar de serem relativamente confiáveis, discos não podem ser considerados seguros o suficiente para prover armazenamento estável no contexto de muitas aplicações, a não ser auxiliados por alguma técnica de redundância.

Para entender como serviços de armazenamento estável são implementados, precisamos saber os tipos de falha que os dispositivos físicos de armazenamento podem sofrer e que tipo de eventos indesejáveis essas falhas podem provocar. Processos operam um disco basicamente através de duas operações [Pan94]:

- **procedimento** escrever(endereço, dado)
- **procedimento** ler(endereço) retorna (estado, dado).

O procedimento de escrita recebe dois parâmetros: i) o endereço físico no disco, onde os dados devem ser gravados; e ii) os dados propriamente ditos. O procedimento de leitura recebe como parâmetro o endereço físico a ser lido e retorna os dados ali contidos e um *estado* que pode indicar *bom* ou *ruim*. Se o *estado* é *bom* significa que os dados não estão corrompidos, se retorna *ruim* significa que os dados lidos estão corrompidos.

Embora os dispositivos físicos atuais já tratem alguns tipos de falhas, outras não podem ser tratadas neste nível. As falhas mais comuns que podem ocorrer com um disco são:

- *Falhas Transientes*: O disco se comporta de forma imprevisível por um curto período de tempo.
- *Setor Danificado*: Um setor do disco é permanentemente danificado e os dados ali gravados são perdidos.
- *Falha na Controladora*: A controladora de discos falha. O conteúdo gravado nos discos não é perdido mas fica inacessível enquanto a controladora não for reparada.
- *Falha de Disco*: O disco é perdido por uma falha física, como por exemplo uma quebra da cabeça de leitura. O dados são irrecuperáveis.

Vamos agora enumerar alguns dos eventos indesejáveis provocados por esses tipos de falhas durante um procedimento de leitura de dados:

1. *Erro de Leitura Simples*: A leitura do endereço *a* retorna *estado ruim* mas deveria retornar *bom* porque os dados não estão corrompidos. Isso é provocado por falhas transientes e não costuma perdurar muito tempo.

2. *Erro de Leitura Persistente*: A leitura de um endereço *a* retorna *estado ruim* e sucessivas leituras continuam com o mesmo comportamento. Isso é normalmente provocado por um setor danificado (ou por uma falha no disco ou na controladora).

3. *Erro não Detectado*: Endereço *a* é *ruim* mas leituras retornam *estado bom*, ou endereço *a* é *bom*, mas a leitura retorna dados diferentes daqueles que foram armazenados.

Se um procedimento de escrita ocorre como desejado, ele deve mudar o conteúdo do endereço *a* para *d*. Entretanto, falhas podem ocorrer e os possíveis eventos indesejados são os seguintes:

1. *Escrita Nula*: Conteúdo do endereço *a* não é modificado.

2. *Escrita Errada*: Endereço *a* passa a ser (*ruim, d*)

Além dos eventos relacionados a operações de leitura e escrita, existem certos tipos de eventos indesejados que podem ocorrer não estando diretamente relacionados com essas operações, mas apenas com a passagem do tempo. Esses tipos de eventos costumam afetar certas partes do disco e costumam ocorrer infreqüentemente. Esses eventos são:

1. *Corrupção*: Um endereço passa de (*bom, d*) para (*ruim, d*).

2. *Recuperação*: Um endereço passa de (*ruim, d*) para (*bom, d*).

3. *Erro não Detectado*: Um endereço muda de (*s, d*) para (*s, d'*)
   com $d \neq d'$

2.2 Implementações

Dependendo dos requisitos de confiabilidade da aplicação e da capacidade necessária, um serviço de armazenamento estável pode ser construído utilizando-se apenas um ou um conjunto de discos. Vamos agora descrever as formas mais comuns de se implementar esse serviço.

2.2.1 Usando um único disco

É possível se implementar um sistema de armazenamento estável usando-se um único disco. Para tanto vamos descrever duas operações chamadas de *Leitura-Cuidadosa* e *Escrita-Cuidadosa* [Pan94]:

- *Leitura-Cuidadosa*: Nesta operação o sistema faz repetidas leituras até que o *estado* retorne *bom* ou seja atingido um limite no número de leituras. Esta operação por si só já pode contornar eventos do tipo *erro de leitura simples*.

- *Escrita-Cuidadosa*: Nesta operação o sistema faz uma escrita seguida de uma leitura até que a leitura retorne *estado bom*. Isso elimina os eventos *Escrita Errada* e *Escrita Nula*.

Através dessas duas operações é possível se implementar as operações *Escrita-Estável* e *Leitura-Estável* que garantem uma certa tolerância a falhas. Para tanto vamos considerar um disco que não sofra uma falha completa e que um evento indesejado aleatório afete apenas um certo grupo de setores chamados setores relacionados. Uma unidade estável de dados é composta por um par de endereços localizados em setores não relacionados a um mesmo evento aleatório. As operações estáveis sobre esses endereços podem ser implementadas assim: uma operação estável de leitura faz uma *Leitura-Cuidadosa* de um dos endereços do par, se o *estado* for *ruim*, faz a leitura a partir do outro endereço do par; uma operação de escrita estável faz uma *Escrita-Cuidadosa* em um dos endereços do par e em seguida faz o mesmo no endereço seguinte. Dessa forma são evitados os problemas causados por eventos aleatórios.

2.2.2 Usando mais de um disco

A forma mais simples de se implementar um serviço de armazenamento estável de dados usando mais de um disco é a técnica de espelhamento [Pan94]. Essa técnica faz o mesmo que a implementação descrita na seção anterior criando unidades estáveis compostas por unidades de alocação de dois ou mais discos. Dessa forma podem-se evitar problemas relacionados a falhas completas de disco.

O grande problema dessa solução está na carga extra imposta a toda operação de gravação uma vez que os dois discos devem ser acessados sucessivamente. Essa carga extra pode ser evitada com a utilização de mais de uma controladora de discos, aumentando-se o custo do sistema.

Uma alternativa existente para melhorar a escalabilidade de serviços de armazenamento baseados em discos magnéticos é a técnica conhecida como RAID (Redundant Array of Inexpensive Disks) [CLG+94]. Nessa técnica um conjunto de discos é usado para simular um único disco, de grande capacidade, através da "alternância de bits". Essa técnica oferece um melhor desempenho uma vez que leitura e gravação de um conjunto de *bits* pode ser feita em paralelo. Entretanto, dessa forma, o sistema não poderá suportar falhas já que um único disco danificado pode inviabilizar a recuperação da informação.

A solução é colocar discos redundantes carregando informações de paridade do conjunto. Dessa forma, a falha de um disco não ocasionará perda de informação. Existem vários níveis de redundância que podem ser utilizados fazendo variar o número de falhas de discos que podem ser toleradas e também o desempenho máximo que se pode obter [CLG+94].

2.3 Sistemas de Arquivo

Já discutimos como falhas de discos podem ser toleradas para se construir unidades de armazenamento confiáveis a partir de discos que podem sofrer essas falhas. Agora iremos discutir como essas unidades de armazenamento estáveis são utilizadas para se construir um serviço de armazenamento estável de dados no sistema computacional.

Os sistemas operacionais normalmente fornecem um serviço de armazenamento de dados através dos respectivos sistemas de arquivo. Para poder garantir a recuperação dos dados armazenados, sistemas de arquivos têm que manter informações sobre a localização desses dados. Essas informações (meta-dados) também ficam armazenadas em disco e a forma usada para organizá-las tem influência no desempenho desses sistemas de arquivo.

Para evitar perda excessiva de desempenho, sistemas de arquivo geralmente operam de dois modos distintos: i) um modo de melhor desempenho, onde não se garante a persistência dos dados armazenados logo após a operação de gravação; e ii) um modo estável onde o sistema garante, a cada operação de gravação, que os dados são armazenados de forma

definitiva em um dispositivo secundário (disco).

Na próxima seção faremos uma revisão sobre a semântica de gravação de dados fornecida através da interface de sistemas de arquivos que seguem o padrão POSIX [Org]. Em seguida veremos alguns detalhes sobre as principais técnicas de estruturação de dados e meta-dados nesses sistemas de arquivo.

2.3.1 Semântica de Gravação em Sistemas de Arquivos POSIX

Devem existir dois modos de operação para o procedimento de gravação de dados em sistemas de arquivo POSIX: gravação assíncrona[1], não estável; e síncrona ou estável. No modo de operação assíncrono, o sistema operacional faz uso de *cache* em memória volátil para melhorar o desempenho da gravação de dados. No modo de operação síncrono o sistema operacional só faz uso da *cache* para otimizar a leitura dos dados. Esses modos de operação podem ser visualizados na Figura 2.1.

O modo de operação a ser usado nas gravações é indicado pela forma que a aplicação abre o arquivo. Se a aplicação abre o arquivo incluindo a diretiva *O_SYNC* então todas as operações de gravação sobre aquele arquivo serão executadas de forma síncrona. Caso essa diretiva não seja usada, as operações serão assíncronas. O modo de operação síncrono pode ser obrigatoriamente usado caso o sistema de arquivos tenha sido montado como síncrono ou o diretório onde o arquivo está armazenado é configurado para só aceitar operações síncronas.

Outra forma de se garantir a estabilidade de operações de gravação sobre um arquivo é fazer uso da função *fsync*. Essa função recebe como parâmetro um arquivo e varre a *cache* do sistema de arquivos procurando blocos modificados que ainda não tenham sido gravados em disco, em seguida realiza essa operação de gravação em disco.

Quando se faz uso de operações assíncronas em um sistema de arquivos, uma falha por parada na máquina pode levar a inconsistências nesse sistema. Para poder tolerar esse tipo de falha, sistemas de arquivo fazem uso de ferramentas de recuperação da consistência (como o *fsck - file system check*) ou técnicas de gravação que mantenham a consistência pelo menos

[1] Na verdade modo atrasado (*delayed*), onde os dados ficam gravados em memória volátil durante um certo intervalo de tempo. No modo assíncrono original a gravação é iniciada imediatamente mas a aplicação não é bloqueada durante esse período.

dos meta-dados. Nas próximas seções vamos ver algumas das principais técnicas para manter essa consistência, bem como prover o serviço de gravação estável de dados.

2.3.2 Sistemas baseados em nós-i

A maioria dos sistemas de arquivo POSIX é derivada do sistema de arquivos desenvolvido para o BSD Unix, mais conhecido como FFS [MJLF84], e utiliza uma estrutura estática para manter as informações relativas aos dados contidos no disco. Três estruturas básicas (meta-dados) mantém o controle dessa informação:

- *Estrutura* Super-Bloco: mantém informações sobre um sistema de arquivos como um todo, ou seja, informações sobre blocos livres, número de arquivos entre outras.
- *Estrutura* nó-i: mantém informações sobre um arquivo específico. Entre essas informações destacam-se tamanho, ponteiros para blocos de dados, data de modificação, informações sobre usuário e grupo donos do arquivo entre outras.
- *Estrutura* Entrada-de-diretório: Faz a ligação entre um caminho de arquivo e o nó-i correspondente.

Para entender como esses meta-dados podem ser alterados por operações de escrita sobre arquivos e a importância da ordenação dessas operações podemos usar dois exemplos básicos:

Exemplo 1: Uma aplicação abre um arquivo e acrescenta cerca de 10Kbytes de dados ao mesmo, sendo que o tamanho de bloco do sistema de arquivo é 8Kbytes. Para corretamente alocar os 10Kbytes extras, o sistema de arquivos deve alocar pelo menos mais um bloco, gravar os dados no mesmo e criar um ponteiro para esse bloco no nó-i do arquivo. Caso os dados sejam gravados fisicamente em disco (no bloco recentemente alocado), mas o nó-i não tenha sido completamente armazenado em disco, uma falha no fornecimento de energia fará com que esses dados sejam perdidos, uma vez que o sistema não tem como localizar o bloco onde estão os dados na ausência do ponteiro relativo ao mesmo no nó-i do arquivo.

Exemplo 2: Uma aplicação cria um arquivo temporário e depois move esse arquivo para seu destino final[2]. Essa operação consiste em modificar as entradas de diretório. É preciso

[2]Essa é a operação básica usada por editores de texto para salvar arquivos e servidores de e-mail para lidar com mensagens recebidas e encaminhá-las a seus destinos locais

remover a entrada do diretório temporário que aponta para o arquivo e criar uma nova entrada no diretório destino do mesmo. Caso a entrada temporária seja removida antes da nova ser criada e ocorra uma falha geral do sistema, o arquivo será definitivamente perdido. Isso exemplifica a importância da ordenação entre operações de sistemas de arquivos.

Essas operações sobre meta-dados e a importância de sua ordenação, juntamente com a distribuição espacial dessas informações ao longo do disco, causam sérios problemas de desempenho nesses sistemas de arquivos baseados em nós-i. Por esse motivo, grande parte das implementações desses sistemas de arquivos faz uso de *caching* para melhorar o desempenho das operações de escrita sobre arquivos, quando requisitos de estabilidade não são necessários.

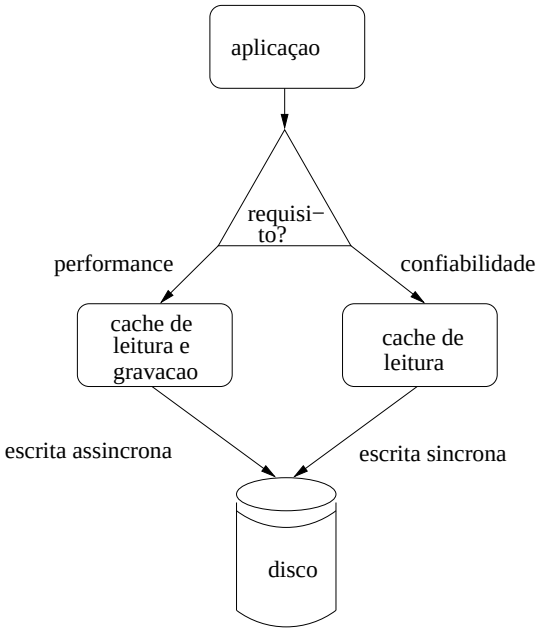

Figura 2.1: Modos de operação de um sistema de arquivos padrão POSIX

A grande maioria dos sistemas de arquivos baseados em nós-i provê armazenamento estável através da sincronia dessas operações sobre dados e meta-dados [MJLF84; CTT94]. Aplicações que necessitam de estabilidade nas operações de escrita sofrem de uma série de problemas de desempenho provocados por essa necessidade de sincronia nessas operações [GP94]. Além disso, problemas como concorrência e o baixo desempenho dos discos magnéticos, quando comparados com outras tecnologias de armazenamento e transmissão de dados, tais como memória volátil e redes [CP93a], tornam essa solução inviável

em certos casos. O uso de *arrays* de discos como solução pode minimizar esse problema mas é uma solução que tem limitações de escala.

Nas próximas seções vamos ver algumas técnicas para minimizar a carga de trabalho extra para se gravar os meta-dados relativos a operações de escrita em sistemas de arquivo POSIX baseados em discos magnéticos.

2.3.3 Sistemas de arquivos baseados em diários e jornais

Uma solução proposta para minimizar a perda de desempenho de sistemas de arquivo baseados em nós-i foi a utilização de sistemas de arquivos baseados em *logs*, ou diários [RO92]. Nesse tipo de sistema de arquivos, dados e meta-dados são gravados seqüencialmente nas operações de escrita, evitando-se o excesso de movimentação das cabeças de gravação do disco. Informações suficientes são mantidas em memória para facilitar a busca e recuperação de informação a partir desse diário.

Para manter o sistema consistente e não sobrecarregar a capacidade do disco, um processo de limpeza é periodicamente executado buscando informações que sejam redundantes e procurando desfragmentar o disco. Esse processo de limpeza é o grande problema dos sistemas de arquivos baseados em diários porque é extremamente ineficiente e ao mesmo tempo necessário em sistemas de arquivo com grande carga de utilização.

Uma das soluções propostas para lidar com esse problema foi a utilização desses diários no sistema de arquivos, mas apenas para operações em meta-dados. Esses sistemas que usam um diário (ou jornal) para gravar as operações sobre meta-dados são chamados de sistemas de arquivo jornalados (journaled file systems) [jfs; xfs; ext; Rei]. Nesses sistemas, qualquer operação sobre meta-dados é gravada nesse diário antes de ser gravada em disco na sua estrutura original garantindo estabilidade nas operações sobre esses meta-dados mesmo quando armazenamento estável não é um requerimento. O diário só contem informação relativa a operações sobre meta-dados, a estabilidade dos dados em si ainda depende do uso de gravação síncrona nesses sistemas de arquivos.

Sistemas de arquivos baseados em nós-i sofrem de outro problema além da perda de desempenho. Eles são incapazes de lidar com arquivos ou discos muito grandes. Com o atual desenvolvimento computacional demandando cada vez mais recursos, entre eles uma grande agilidade e capacidade nos sistemas de armazenamento, esses sistemas baseados em

nós-i estão se tornando obsoletos.

Para suportar arquivos e discos cada vez maiores uma opção é a modificação das estruturas existentes, como mapas de blocos livres, adicionando-se mais bits para lidar com a informação extra. Entretanto, essa solução não é muito escalável, devido à forma linear que essas estruturas guardam as informações. Mapas de bits muito extensos obviamente levam a um maior tempo gasto cada vez que se tem de gravar alguma informação nos mesmos. Além disso, paradas no fornecimento de energia, ou falhas do sistema operacional, podem levar a inconsistências no sistema de arquivos devido à maioria das aplicações de usuário operarem de modo assíncrono. Tradicionalmente, esse problema era tratado através de um processo de checagem realizado sempre que o sistema operacional é reiniciado após uma pane inesperada, e periodicamente para prevenir falhas no sistema de arquivos. Esse procedimento de checagem precisa varrer toda a partição de disco em busca de inconsistências. Com o grande aumento de capacidade dos discos atuais esse procedimento é por demais lento e pode atrasar consideravelmente o processo de inicialização do sistema.

Nos sistemas baseados em jornal a checagem de disco na inicialização é substituída por um procedimento mais simples, que lê a informação contida no diário de operações sobre meta-dados, fazendo as correções necessárias para colocar o sistema em um estado consistente. Esses sistemas de arquivo tendem a ter uma ligeira perda de desempenho em certas operações assíncronas por causa da carga extra provocada pelo uso do diário. Entretanto, eles garantem uma maior estabilidade do sistema de armazenamento e um processo de recuperação mais ágil.

2.3.4 Replicação remota de buffers

Soluções de armazenamento estável de dados baseadas na replicação remota de *buffers* [BCPS02; LGG+91] são uma outra alternativa para otimizar o desempenho de aplicações que necessitam desse serviço. A idéia consiste em replicar-se remotamente em memória os dados e meta-dados referentes a operações de gravação estável submetidas ao sistema de arquivos, para garantir tolerância a falhas por *crash* da máquina onde essas operações estão sendo processadas, e mesmo assim evitar a perda de performance obtida quando se utiliza a gravação síncrona em discos.

Arquitetura

Como já vimos anteriormente, um sistema de arquivos tradicional opera de duas formas: aplicações que não requerem estabilidade dos dados gravados podem usar um modo de maior performance onde os dados enviados para gravação ficam em memória até que o sistema possa gravá-los em disco sem prejudicar o desempenho da mesma. Aplicações que requerem estabilidade dos dados gravados utilizam a forma de gravação síncrona, onde os dados são enviados diretamente para o disco bloqueando a aplicação até que essa operação tenha sido finalizada. Esses modos de operação podem ser visualizados na Figura 2.1.

Um serviço baseado na replicação de *buffers* tenta evitar a perda de performance provocada pela forma de gravação síncrona utilizando a memória de máquinas remotas para prover tolerância a falhas, e, localmente, gravação assíncrona para prover desempenho. Os modos de operação de um sistema de arquivos que utiliza essa solução podem ser vistos na Figura 2.2.

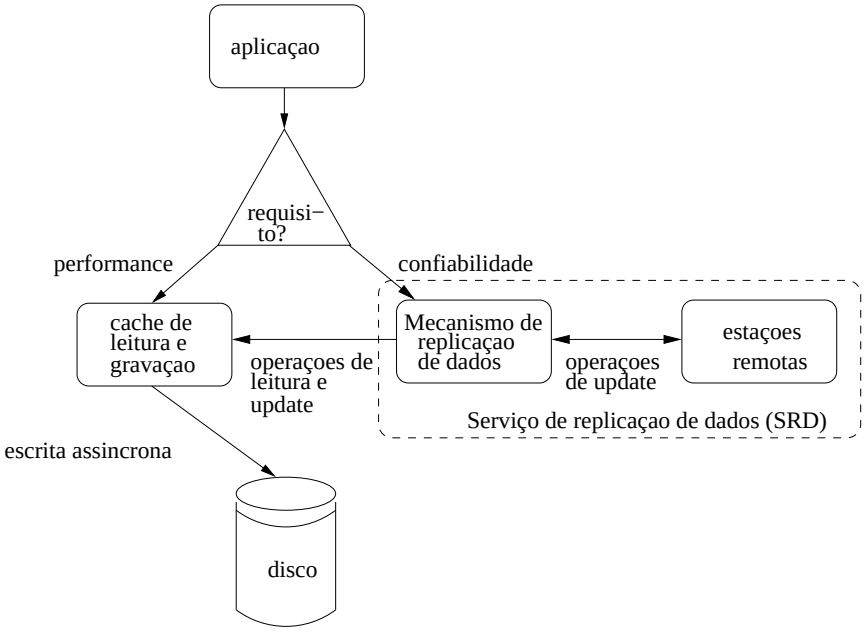

Figura 2.2: Modos de operação de um sistema de arquivos que utilize replicação remota de *buffers*

Essa solução assume que a rede não pode corromper mensagens mas pode atrasar arbitrariamente o envio e a entrega das mesmas. A rede também pode eventualmente estar particionada de forma que a replicação não se torne possível em um dado momento. O siste-

ma deve de alguma forma detectar essa partição e encontrar uma forma alternativa de manter a estabilidade dos dados. Os modos de operação de um sistema que seja capaz de detectar partições na rede[3] está representado na Figura 2.3.

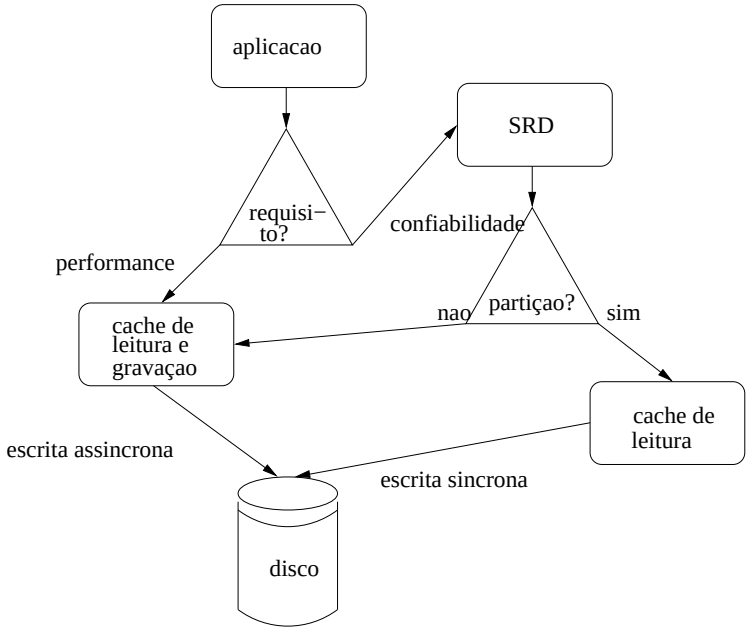

Figura 2.3: Modos de operação de um sistema de arquivos que utilize replicação remota e detecte partições na rede

Sistemas de arquivo e demais soluções de armazenamento estável de dados que utilizam replicação remota de *buffers* dependem de um procedimento de recuperação de dados sempre que ocorram de falhas no fornecimento de energia ou no sistema operacional da estação onde os dados estavam sendo armazenados [BCPS02; LGG+91]. Embora apresentem pequenas diferenças na forma de se replicar os dados, seja por espelhamento ou alternância de bits, todas essas soluções executam um procedimento de recuperação de dados que procura, nas demais estações que compõem o serviço de replicação, por dados e meta-dados que eventualmente não estejam ainda estabilizados em disco, da mesma forma que o escalonador do sistema de arquivos varre a *cache* de memória volátil em busca de blocos não gravados em disco num sistema de arquivos local. Após a execução desse procedimento de recuperação esses sistemas de arquivo estão consistentes e atualizados com os últimos dados que foram gravados pelas aplicações estáveis antes da falha.

[3]Partições reais ou virtuais provocadas pela queda do desempenho.

2.4 Sumário

Neste capítulo mostramos os principais problemas a serem contornados para se desenvolver sistemas de armazenamento de dados tolerantes a falhas. Apresentamos também como lidar com as falhas que podem ocorrer com os dispositivos físicos mais usados para se implementar tal serviço criando um sistema de armazenamento tolerante a essas falhas.

Por fim, vimos como sistemas de arquivos se utilizam desses sistemas de armazenamento físico para prover uma *interface* de armazenamento de objetos (arquivos) para as aplicações de usuários. Mostramos as principais formas de organização utilizadas para se armazenar, buscar e recuperar esses objetos e as implicações no desempenho de cada uma dessas alternativas.

Capítulo 3

Avaliação de Desempenho de Sistemas de Armazenamento de Dados

Temos acompanhado nas ultimas décadas um crescimento espantoso na capacidade de processamento dos *microchips*. Esse crescimento cria uma tendência de se considerar a velocidade dos processadores como a principal, quando não a única, métrica de performance de um sistema computacional [CP93a].

Comparadas com os processadores, que têm evoluído em um ritmo bastante acelerado, tecnologias de armazenamento de dados têm se desenvolvido a passos lentos [CLG+94]. Com isso, armazenamento e recuperação de dados tende a se tornar cada vez mais o gargalo para muitas categorias de aplicações. Mais que isso, essa diferença de desempenho entre processadores e dispositivos de armazenamento, tornam a performance dos últimos cada vez mais relevante na avaliação geral da qualidade de um sistema.

A importância do estudo de métricas e ferramentas de avaliação de desempenho é, portanto, uma conseqüência desse diferente ritmo de evolução dessas tecnologias. Existem diversas ferramentas e técnicas para avaliar o desempenho de serviços de armazenamento de dados, gerando resultados tanto distintos quanto, certas vezes, conflitantes. Algumas idéias, entretanto, foram se tornando consenso na literatura, como por exemplo: uma ferramenta de avaliação de desempenho de sistemas de armazenamento de dados ideal deve impor ao sistema uma carga semelhante àquela que será submetida por usuários reais, para ajudar desenvolvedores a identificar os principais pontos de falha [CP93a; SFGM93; Mil96]. Além disso, essas ferramentas devem ajudar administradores e usuários a entender

melhor o comportamento desses sistemas, auxiliando na hora de uma compra.

Neste capítulo, vamos explorar esses dois aspectos da avaliação de desempenho de sistemas de armazenamento de dados: métricas de performance e representatividade das ferramentas existentes em relação às cargas aplicadas a esses sistemas em ambientes reais de produção.

3.1 Métricas de Performance

Por envolver um grande número de variáveis, aleatórias ou não, a avaliação de desempenho para sistemas de armazenamento pode ser feita através de um grande número de métricas. Ao se fazer um estudo de performance de sistemas de armazenamento estável de dados, é fundamental entender o que essas métricas avaliam e como as diversas variáveis afetam o resultado das mesmas.

A principal métrica considerada para esses sistemas é a vazão. Esta pode ser medida de duas formas [CP93b; CP93a]:

- Taxa de entrada e saída: Normalmente medida em número de operações de entrada e saída por unidade de tempo;
- Taxa de fluxo de dados: Normalmente medida em número de *bytes* lidos ou gravados por unidade de tempo.

Taxa de entrada e saída é normalmente usada quando se deseja prever a performance de um sistema de armazenamento de dados ao ser submetido a aplicações que executam muitas operações em curtos períodos de tempo, sendo cada uma delas sobre uma pequena quantidade de dados, como por exemplo servidores de correio eletrônico sobrecarregados, ou aplicações baseadas em transações. Taxa de fluxo de dados é mais adequada para se ter uma idéia do desempenho de aplicações que trabalham com grandes volumes de dados, como aplicações científicas ou aplicações baseadas em salvaguardas por exemplo.

Entretanto, métricas que avaliam apenas a vazão dos dados não são suficientes para se comparar a performance de dois sistemas de armazenamento porque, dependendo do requisito da aplicação, não adianta um sistema ser capaz de armazenar um volume imenso de

dados em apenas algumas poucas unidades de tempo se leva também esse mesmo tempo para armazenar uma pequena quantidade de dados.

A segunda métrica mais importante, portanto, é o tempo de resposta, ou latência [CP93a; CP93b]. Essa medida indica quanto tempo o sistema leva para acessar ou iniciar uma gravação de dados. Essa métrica é normalmente relacionada à avaliação de performance de leituras mas também tem grande importância para sistemas baseados em transações que necessitam de um sistema de armazenamento que seja capaz de gravar pequenas quantidades de dados simultaneamente.

Essas duas métricas são complementares porque discos magnéticos, por causa de sua arquitetura de acesso à mídia de gravação, se comportam de formas distintas dependendo da forma de gravação usada: seqüencial ou por intervalos. Quando acessados seqüencialmente, caso mais comum de se ocorrer em sistemas que armazenam grandes volumes de dados de uma vez, discos atingem os seus picos de desempenho. Quando acessados em intervalos, como por exemplo em aplicações baseadas em transações, a cabeça de gravação tem que se reposicionar a cada operação, aumentando a latência e, por conseqüência, diminuindo a vazão.

Capacidade de armazenamento também é uma métrica importante para administradores de sistema, principalmente quando levada em consideração junto com o custo inerente ao equipamento. Gráficos de desempenho e capacidade são muito úteis para comparação quando analisados em paralelo ao custo do sistema. Se capacidade ou custo não fossem métricas importantes, os sistemas computacionais já teriam deixado de usar discos magnéticos em detrimento de outras formas de memória não volátil, como *flash memory*. Entretanto, essas formas alternativas e rápidas de memória não volátil têm um alto custo para cada *byte* de capacidade quando comparadas com os discos magnéticos [CP93b].

A última métrica importante para sistemas de armazenamento de dados é a confiabilidade. Essa é uma métrica de muita importância para sistemas de armazenamento estável de dados. Normalmente, fabricantes expressam essa medida através do *tempo médio para falha*, que informa a expectativa de tempo útil de uso do equipamento antes que esse venha a sofrer uma falha. No presente trabalho, vamos considerar que o tempo médio para falha de um disco é suficiente para atender nossos requisitos de confiabilidade.

Algumas dessas métricas são triviais de serem comparadas, tais como o custo e a capa-

cidade. As restantes são por demais influenciadas pelo ambiente de execução e as diferentes cargas de trabalho aplicadas ao sistema de forma que uma avaliação criteriosa se torna necessária.

Na próxima seção vamos analisar aspectos de metodologias e ferramentas úteis para avaliar o desempenho de sistemas de armazenamento estável de dados.

3.2 Ferramentas

Diversos trabalhos têm sido publicados apresentando e avaliando técnicas e ferramentas de avaliação de desempenho e qualidade de equipamentos computacionais. Ainda não se chegou a um consenso sobre a forma ideal de se avaliar um sistema de armazenamento porque para cada perfil de uso que o sistema venha a se submeter, uma metodologia diferente deve ser aplicada para se avaliar seu desempenho. Neste trabalho focamos a nossa análise na avaliação de desempenho de sistemas de armazenamento estável de dados sob o ponto de vista de um administrador de sistemas, mais especificamente um administrador de sistemas onde as seguintes categorias de aplicações predominam: sistemas baseados em transações, servidores de correio eletrônico e servidores de arquivo. Nossa análise, portanto, estará voltada para ferramentas e técnicas que sejam úteis a essa categoria de usuários.

Existe uma série de características que são desejáveis em uma ferramenta de avaliação de desempenho de um sistema. A principal delas é que essa ferramenta deve ajudar desenvolvedores e usuários a entender por que o sistema se comporta da maneira que vem fazendo [CP93a; CP93b]. Outra característica importante é que a ferramenta de avaliação deve corretamente representar a carga de uso que o sistema vai sofrer quando estiver instalado. Dessa forma, pode-se prever com mais acurácia esse comportamento [SFGM93; Mil96; CP93a; CP93b; MS96; MK].

É comum o uso de medidas fornecidas pelos fabricantes dos equipamentos de armazenamento, como discos por exemplo, para ilustrar o funcionamento de sistemas de armazenamento de dados. Entre essas medidas estão tempo médio de acesso, taxa de transferência nominal do sistema de armazenamento, etc. Essa medidas podem trazer alguma informação inicial, mas ajudam pouco a prever o comportamento de tais sistemas em um ambiente de produção [CP93b; CP93a].

Uma boa ferramenta de avaliação de desempenho deve poder isolar razões para baixa performance e assim ajudar desenvolvedores a melhorar esses sistemas. Para tanto, essas ferramentas devem ser limitadas pelas operações de entrada e saída. Em outro caso, outros fatores podem estar influindo na avaliação desejada. Em certos casos, entretanto, uma ferramenta não limitada por entrada e saída pode ser útil, como veremos mais adiante.

Embora essa classificação não seja unanimidade, vamos considerar a existência de dois tipos principais de ferramentas para avaliação de desempenho de sistemas de armazenamento estável de dados: i) ferramentas sintéticas, também chamadas de *micro benchmarks*; e ii) ferramentas baseadas em aplicações [CP93b].

3.2.1 Ferramentas Sintéticas

As ferramentas sintéticas utilizam o sistema de armazenamento invocando diretamente as chamadas de sistema relativas a leitura, gravação e modificação dos dados e medindo o custo dessas operações. Por invocar essas funções diretamente, essas ferramentas têm uma maior capacidade de serem limitadas por essas funções, o que é uma característica desejável para essa categoria de ferramentas.

Medidas importantes que podem ser avaliadas através dessa categoria de ferramentas incluem:

- Taxa de vazão do sistema de arquivos;
- Latência relativa a todas as operações;
- Influência de concorrência e fragmentação dos dados[1] sobre a performance.

A principal limitação dessas ferramentas está no fato de que apesar de serem limitadas por operações de entrada e saída, elas não estão realizando nenhum trabalho computacional prático, ou seja, não representam bem nenhuma categoria de aplicação real, sendo assim pouco úteis para prever o comportamento dos sistemas em um ambiente de produção [SFGM93; Mil96].

Alguns projetos se destacam entre essas ferramentas, entre esses podemos citar:

[1] Distribuição espacial, no disco, dos vários blocos de dados que compõem um objeto, como um arquivo por exemplo.

- *Bonnie* [Bra] - Ferramenta que aplica uma série de cargas de trabalho sobre um único arquivo incluindo leituras, gravações, acessos concorrentes, entre outras coisas; e
- *IOStone* [PB90] - Ferramenta de *micro benchmark* baseada em cargas de uso[2] de estações de trabalho observadas em ambientes de produção.
- *IOZone* [Nor] - Ferramenta flexível, portável para uma grande variedade de sistemas operacionais que permite fazer avaliações de operações síncronas.

Essas ferramentas são úteis principalmente para projetistas que desejam entender o comportamento dos sistemas de armazenamento sob determinada carga específica, mas são de pouca ajuda para administradores que necessitam escolher entre um sistema e outro para utilização porque as cargas reais a que um sistema será submetido são muito mais variáveis que aquelas submetidas por essas ferramentas.

3.2.2 Avaliação Baseada em Aplicações

A utilização de aplicações reais para avaliar o desempenho de sistemas de armazenamento traz a vantagem de se poder representar com mais propriedade a carga a que o sistema deverá ser submetido em um ambiente de produção. Isso é feito pelo uso de diversas aplicações como compiladores, bancos de dados, editores, em variadas combinações de forma a simular um sistema em produção.

Esse tipo de avaliação de desempenho tem o potencial de ajudar bem mais o administrador na escolha de um sistema de armazenamento. Entretanto, a escolha da carga de trabalho a ser aplicada é um aspecto muito importante para o resultado final e tem sido objeto de estudo de diversos trabalhos [SFGM93; Mil96].

Outro problema encontrado nesse tipo de avaliação é decorrente do fato de que muitas vezes essas aplicações reais não são limitadas pelas operações de entrada e saída, mascarando os resultados. Um cuidado deve ser tomado na instrumentação dessas aplicações para se tentar medir apenas o que é relevante para a avaliação em questão. Essa é, entretanto, uma característica desejável de uma ferramenta de avaliação em certos casos, como o do presente

[2] Apesar de baseada em cargas reais, essa ferramenta ainda é considerada sintética por fazer a medição de performance das operações individualmente.

trabalho, porque ajuda-nos a encontrar quão relevante um ganho de desempenho é obtido ao se mudar de um sistema para outra. É interessante descobrir qual o ganho prático em desempenho que um sistema com ótimos resultados obtidos através de ferramentas sintéticas vai provocar quando submetido a um teste de carga com uma aplicação real que não seja totalmente limitada por operações de entrada e saída.

3.3 Sumário

Neste capítulo, abordamos o problema da avaliação de desempenho de sistemas de armazenamento de dados, primeiramente mostrando a sua relevância frente ao diferente ritmo de evolução no desenvolvimento dos dispositivos de armazenamento e de processamento. Identificamos as principais métricas que são importantes para essa avaliação e mostramos algumas formas de combiná-las para se formar resultados ilustrativos.

No final, classificamos as ferramentas usadas para esse fim em duas categorias básicas, ferramentas sintéticas e aplicações. Embora essa classificação não seja uma unanimidade, ela ilustra de forma simples os dilemas encontrados por aqueles que desejam realizar uma avaliação desse tipo.

No próximo capítulo iremos analisar as implementações de sistemas de arquivo que foram usadas neste trabalho, focando essa análise na semântica de armazenamento estável.

Capítulo 4

Implementações

Nos dois capítulos anteriores procuramos introduzir o leitor no contexto de sistemas de armazenamento estável de dados e de metodologias usadas na avaliação desses sistemas. Neste capítulo, vamos apresentar alguns detalhes de três implementações de sistemas de arquivo, que provêem armazenamento estável de dados, usadas como referência e analisadas em nosso trabalho.

Vamos analisar a estrutura que esses sistemas de arquivo utilizam para organizar seus dados e os correspondentes meta-dados, focando essa análise no impacto provocado no desempenho dos mesmos. Também será explorada a semântica de gravação estável dos mesmos.

Os sistemas utilizados foram o Linux EXT2 [CTT94], o ReiserFS [Rei] e o SaliusFS [BCPS02]. O EXT2 é o sistema de arquivos padrão para o popular sistema operacional Linux e é um sistema baseado em nós-i como o FFS [MJLF84]. O ReiserFS é um sistema de arquivos moderno, baseado em árvores balanceadas, que utiliza técnicas de jornal para melhorar a eficiência na recuperação de falhas, e finalmente o SaliusFS, que foi implementado por nós para este trabalho a partir de um projeto anterior [Sta]. O SaliusFS é um sistema de arquivos baseado no código do Linux EXT2 e utiliza replicação remota de dados pela rede para prover tolerância a falhas.

4.1 Sistema de Arquivos EXT2

O sistema de arquivos EXT2 [CTT94] (Second Extended File System) foi desenvolvido para ser utilizado pelo sistema operacional Linux e é uma evolução do *Extended File System*,

que por sua vez foi desenvolvido a partir do sistema de arquivos do sistema operacional Minix [Tan87].

O principal objetivo no desenvolvimento do sistema de arquivos EXT2 foi adicionar novas funcionalidades e remover limitações existentes no sistema de arquivos do Minix. Essas limitações restringiam o tamanho do sistema de arquivos, o tamanho dos nomes e o número de arquivos a valores muito pequenos para um sistema operacional com o potencial do Linux.

Como a maioria dos sistemas de arquivo para Linux, o EXT2 traz uma série de características básicas derivadas de outros sistemas de arquivo para UNIX [MJLF84; Tan87; MK]. Arquivos são representados por uma estrutura chamada nó-i, diretórios são arquivos contendo uma lista de apontadores para outros arquivos.

Iremos agora explicar a organização estrutural básica dos dados e meta-dados no sistema de arquivos EXT2.

4.1.1 Organização

A estrutura de dados básica de um sistema de arquivos EXT2 é o superbloco. O superbloco é uma sequência fixa de bytes gravados no início de cada partição EXT2. O tamanho dessa sequência depende do tamanho da partição e do número de blocos existentes na mesma. Além da cópia principal no início de cada partição, existem cópias de *backup* do superbloco distribuídas ao longo da mesma.

No superbloco ficam gravadas informações sobre o tamanho de cada bloco de dados, o número de blocos e nós-i existentes na partição, além de dois mapas de bits: um para indicar a utilização dos blocos de dados e um para marcar os nós-i livres. Além dessas informações estruturais, o superbloco mantém informações sobre o tipo de sistema de arquivos, última data de montagem, entre outras informações.

Um nó-i contém as informações relativas a um arquivo individual no sistema. Essas informações incluem o *id* do usuário, o *id* do grupo, data e hora do último acesso, data e hora da última modificação, entre outras. Além disso, estão no nó-i os apontadores para os blocos de dados do arquivo. Esses apontadores são de três tipos:

- Apontadores Diretos: são ponteiros para blocos contendo dados.

- Apontadores Indiretos: são ponteiros para blocos contendo apontadores diretos.
- Apontadores Indiretos Duplos: são ponteiros para blocos contendo apontadores indiretos.

Diretórios são arquivos especiais que contêm uma lista de nós-i com um nome de arquivo para cada um. São organizados de forma hierárquica. Quando o sistema recebe uma requisição para um caminho de arquivo, faz a pesquisa a partir do diretório raiz, cujo nó-i está indicado no superbloco, até encontrar o nó-i correspondente ao arquivo. Esse nó-i fica, então, armazenado na memória e chamadas de localização subsequentes não exigem mais acesso direto às estruturas no disco.

4.1.2 Modos de Operação

O sistema de arquivos EXT2 faz uso intenso de memória volátil para otimizar o seu desempenho. O uso de *cache* de leitura é útil em todo caso, mas o uso dessa memória volátil nas operações de gravação de dados, ou modificação de meta-dados, pode comprometer a confiabilidade do sistema.

Existem dois modos básicos de operação para o sistema de arquivos EXT2 nas operações de gravação sobre um arquivo, dependendo do uso ou não de *cache* em memória volátil: i) modo síncrono, onde todas as operações de gravação sobre meta-dados e dados são gravadas diretamente em disco antes da aplicação que as solicitou ser desbloqueada; e ii) modo assíncrono, onde o sistema faz uso de *cache* para otimizar a performance das operações, inclusive de gravação, comprometendo a confiabilidade.

Para se garantir que os dados gravados estejam estáveis após o término da operação é necessário que os mesmos sejam gravados em disco antes do desbloqueio da aplicação. Existem quatro formas de se garantir esta estabilidade:

- Utilizando a diretiva *O_SYNC* na abertura do arquivo - Essa diretiva garante que todas as operações de gravação sobre esse arquivo serão realizadas de forma síncrona;
- Fazendo gravação assíncrona e em seguida executando-se a chamada de sistema *fsync(fd)* - Essa chamada de sistema grava em disco todos os dados e meta-dados modificados no arquivo indicado por *fd*;

- Montando-se todo o sistema de arquivos como síncrono - É possível forçar que todas as operações de gravação sobre um sistema de arquivos sejam feitas de forma síncrona desde que essa opção seja indicada no momento da montagem;

- Adicionando-se a diretiva de sincronia no diretório que contém o arquivo que está sendo utilizado - Dessa forma, todos os arquivos daquele diretório se comportam como se tivessem sido abertos com a diretiva *O_SYNC*.

A semântica original do FFS [MJLF84] indicava que todas as operações sobre meta-dados deveriam ser síncronas, independentemente do uso ou não de cache para a gravação dos dados em si. Entretanto, essa semântica resultava em um desempenho muito baixo nesse sistema de arquivos, de forma que a grande maioria das novas implementações de sistemas de arquivos baseadas no FFS relaxaram essa semântica em favor do desempenho. Recentes implementações do FFS para o sistema operacional *FreeBSD* [1] trazem uma técnica alternativa, denominada *soft updates* [MG99], para a semântica de gravação dos meta-dados. Nessa técnica, as operações sobre meta-dados não são gravadas diretamente em disco, ficam armazenadas logicamente em um grafo organizados através de um esquema escolhido no momento da criação da partição. Periodicamente, o escalonador do sistema de arquivos grava operações sobre meta-dados no disco, seguindo as diretivas apontadas pelo esquema do grafo. Tais diretivas podem ser próximas da semântica original, ou seja, levando a uma atualização quase síncrona dos meta-dados, ou mais relaxada, levando a um desempenho superior mas ainda garantindo pelo menos consistência no sistema de arquivos através da ordenação das operações sobre os meta-dados [MG99].

Existe muita controvérsia entre os desenvolvedores de sistema de arquivos e desenvolvedores de aplicações tolerantes a falhas sobre os prejuízos em confiabilidade provocados por esse relaxamento de semântica. Como esse tópico extenderia demais nosso trabalho, preferimos estudar apenas o caso onde essa semântica não é relaxada, ou seja, apenas aplicações que utilizam gravação síncrona operam sobre os diretórios onde executamos os nossos experimentos.

[1] Sistema operacional baseado na licença GPL desenvolvido por voluntários através da internet e organizado na Universidade da California em Berkeley - www.freebsd.org.

4.2 ReiserFS

O sistema de arquivos ReiserFS é um projeto desenvolvido utilizando-se técnicas de sistemas gerenciadores de bancos de dados em sistemas de arquivos. O objetivo principal dos desenvolvedores desse sistema de arquivos é trazer ganhos de desempenho para aplicações que trabalham de forma semelhante a SGBDs, ou seja, aplicações que operam sobre uma grande quantidade de arquivos pequenos sem prejudicar a performance para operações sobre arquivos grandes.

Tradicionalmente, se achava que objetos pequenos (arquivos entre eles) deveriam ser tratados em outra camada, que não o sistema de arquivos, porque o custo de se otimizar as operações sobre esses arquivos pequenos comprometeria o desempenho dos sistema de arquivos nas operações sobre arquivos grandes. Dessa forma, pequenos objetos (registros em uma base de dados) sempre foram tratados por SGBDs e a pesquisa nessa área apontou o uso de árvores balanceadas como a melhor estrutura para organizar os mesmos.

O desenvolvimento do sistema de arquivos ReiserFS tem mostrado que, incluindo-se modificações específicas para um sistema de arquivos, algoritmos de arvores balanceadas podem tornar eficiente o processamento de pequenos objetos diretamente pelo sistema de arquivos sem comprometer o desempenho das operações sobre arquivos grandes [Rei].

Toda a estrutura organizacional do ReiserFS é uma árvore, assim como o EXT2, estando a grande diferença no fato de que no EXT2, essa árvore é estática, rigidamente dependente da hierarquia de diretórios, do tamanho dos blocos e da disposição dos mesmos. No ReiserFS essa árvore é balanceada e os nós da mesma são ordenados por uma chave que é parte componente dos meta-dados de cada objeto (diretório, arquivo, ou nó interno da árvore). Pelo fato de que todos os galhos dessa árvore terem exatamente a mesma profundidade espera-se um ganho geral de desempenho ao se realizar operações de leitura e gravação.

O ReiserFS é também mais eficiente em termos de utilização de espaço. Sabemos que quanto maior o bloco de dados de um sistema baseado em nós-i, melhor seu desempenho em termos de performance, mas maior se torna o desperdício de espaço toda vez que um arquivo não preenche um bloco inteiro com seus últimos *bytes* de dados. No ReiserFS isso é evitado de duas formas: i) existem blocos de dados não formatados, ou seja, sem tamanho fixo. São usados para guardar sequências maiores que 4kbytes de um arquivo; ii) existem

blocos formatados que servem para guardar as porções finais de mais de um arquivo, ou seja, os últimos *bytes* de um arquivo podem estar compartilhando um bloco com o final de outro arquivo para evitar desperdício de espaço.

Da mesma maneira que sistemas de arquivos baseados em nós-i, o ReiserFS faz a localização de um arquivo específico varrendo a árvore do topo para os galhos. Existem duas diferenças, entretanto: i) No EXT2 um galho pode crescer indefinidamente, dependendo do número de sub-diretórios existentes para encontrar o arquivo, e ii) No ReiserFS uma mesma chave pode representar mais de um arquivo, caso esses sejam pequenos. Existe muita controvérsia entre pesquisadores se o uso de árvores balanceadas é eficiente ou não neste caso, uma vez que minimiza o tamanho dos galhos dessas árvores. Esperamos encontrar algumas respostas com nossos experimentos descritos no capítulo 5.

Além disso, o ReiserFS é um sistema de arquivos baseado em jornal, o que torna desnecessária a execução de um procedimento de checagem em caso de reinicialização provocada por uma pane inesperada.

Vamos agora analisar as estruturas internas do ReiserFS.

4.2.1 Organização

Uma partição ReiserFS é uma árvore balanceada onde cada bloco dessa partição é um nó da mesma. Existem três tipos de nós:

- Nó interno - Nós internos não contém dados, armazenam ponteiros para sub-árvores. O bloco raiz da partição informa qual é o nó interno de mais alto nível no momento. Sempre que se precisa aumentar o tamanho da árvore adiciona-se um nó interno no topo da mesma;

- Nó formatado - Nós formatados é a estrutura onde se armazenam itens do sistema de arquivos e suas chaves de pesquisa. Itens podem ser de quatro tipos: i) Item indireto, ii) Item direto, iii) Item estático, ou iv) Item de diretório;

- Nó não formatado - São os últimos nós dos galhos da árvore e contém dados de arquivos maiores que 4Kbytes.

O número de itens agrupados em um nó formatado é variável, dependendo do tamanho de cada um deles e do tamanho dos blocos na partição. Apenas o tamanho máximo dos galhos na árvore é fixado no momento da criação da partição, sendo seu valor padrão 5 (cinco).

A Figura 4.1 mostra a estrutura de um nó interno da árvore. Esses nós não contém dados e as únicas informações existentes são as referências a outros nós e as relativas chaves de pesquisa. No cabeçalho do bloco fica armazenado o seu número, e sua chave de pesquisa.

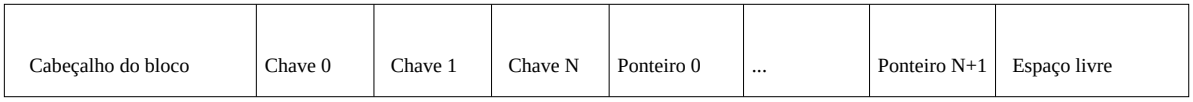

Figura 4.1: Estrutura de um nó interno do ReiserFS em um bloco de disco

A Figura 4.2 representa a estrutura de um nó formatado. Esses nós armazenam meta-dados de arquivos e diretórios e podem, também, armazenar pequenas quantidades de dados de arquivos. Os cabeçalhos de ítens armazenam as chaves de pesquisa dos mesmos e indicam o tipo de informação contida nesses ítens. Essas informações podem ser, além de meta-dados de arquivos e diretórios, entradas de diretório ou dados de arquivos pequenos entre outras coisas.

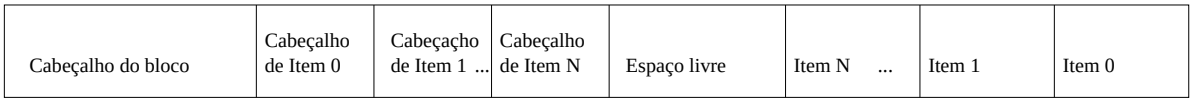

Figura 4.2: Estrutura de um nó formatado do ReiserFS em um bloco de disco

Como foi visto anteriormente, os itens armazenados em nós formatados podem ser de quatro tipos:

- Item de dados estáticos - Este tipo de item armazena as informações equivalentes a um nó-i no UFS [RT74], ID do dono e grupo do arquivo, número de links, data de modificação, tamanho, permissões etc;
- Item direto - Armazena os dados de arquivos pequenos;
- Item indireto - Armazena um grupo de ponteiros para nós não formatados para um arquivo grande;

- Item de diretório - É o equivalente a um bloco de dados de um diretório, armazena uma tabela contendo nomes de arquivos e suas correspondentes estruturas de meta-dados para localização.

Na Figura 4.3 podemos ver uma representação simplificada da hierarquia dessas estruturas no ReiserFS. O nó raiz da árvore está representado por (1); um ponteiro para um nó formatado pode ser visto em (2); (3) é um item indireto, contendo um ponteiro para um nó não formatado de dados representado por (4).

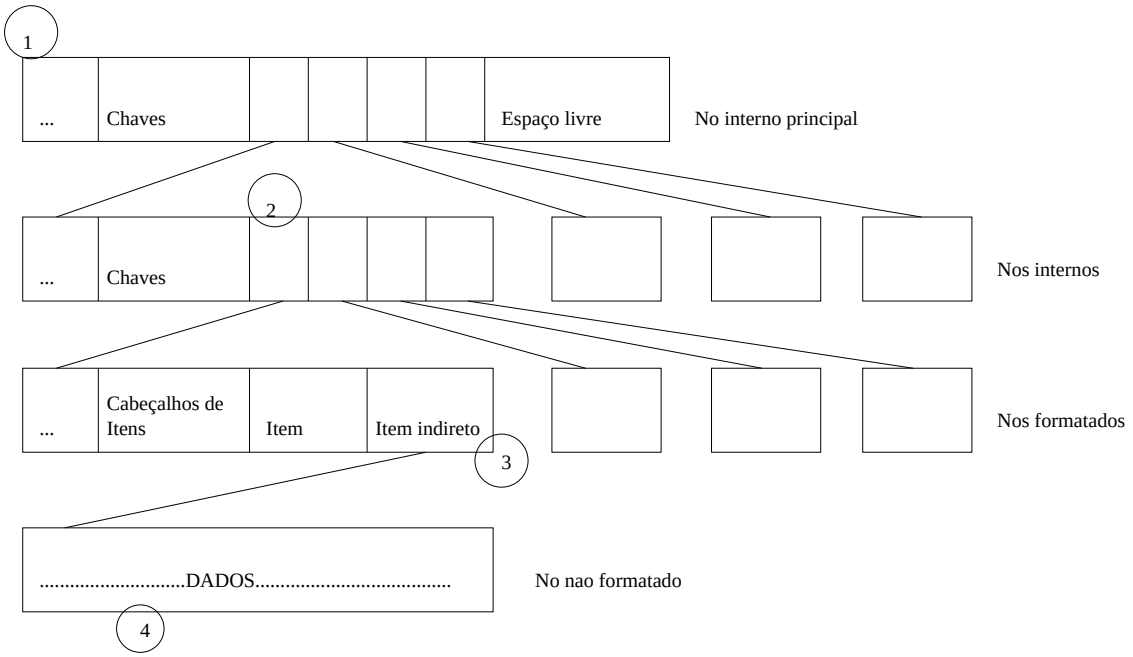

Figura 4.3: Estrutura simplificada de uma partição ReiserFS

4.2.2 Modos de operação

O ReiserFS, assim como o EXT2, pode ser acessado de duas formas distintas: i) modo assíncrono, onde se usa *cache* de leitura e gravação; e ii) modo síncrono (seguro), onde se usa *cache* apenas para leitura de dados. Da mesma forma que o EXT2, o ReiserFS sofre penalidades no desempenho quando está operando de forma síncrona, entretanto, como a forma de distribuição dos meta-dados e dados nesses dois sistemas de arquivos difere bastante, esperamos encontrar diferenças de performance nos nossos experimentos.

4.3 SaliusFS

O sistema de arquivos Salius foi projetado com o objetivo de otimizar o desempenho nas operações de gravação estável através da replicação remota de dados. Esse projeto foi apresentado como uma dissertação de mestrado [Sta] e teve sua implementação [BCPS02] realizada por nós para possibilitar a elaboração deste trabalho.

Essa implementação foi baseada no código do sistema de arquivos EXT2 encontrado na versão 2.2.19 do núcleo do sistema operacional Linux. Por ser derivado desse sistema, o Salius compartilha a maior parte do código com o sistema EXT2, adicionando-se, apenas, as funcionalidades necessárias para prover a replicação de dados, evitar o uso de gravação síncrona e manter a consistência dessas operações.

Um sistema de arquivos SaliusFS é montado sobre uma partição EXT2 normal, compartilhando, portanto, a mesma estrutura de organização de dados e meta-dados. Para suportar a mesma semântica de gravação do sistema de arquivos EXT2, algumas funções básicas, como *file_write(), fsync(), open()* foram modificadas para incluir a replicação remota de dados sempre que estabilidade fosse um requisito da aplicação.

O funcionamento básico do SaliusFS pode ser definido assim: sempre que o requisito da aplicação for estabilidade dos dados na gravação utilize gravação assíncrona para melhorar a performance e garanta a estabilidade usando o serviço de replicação remota de dados em memória volátil.

4.3.1 Organização

O sistema de arquivos SaliusFS foi implementado a partir do código do sistema EXT2 encontrado no núcleo do sistema operacional Linux na sua versão 2.2.19. Por esse motivo, o código dos dois sistema é semelhante, diferindo apenas nos pontos onde foram inseridas as funções relativas à replicação de dados.

Todas as estruturas de disco do SaliusFS são idênticas àquelas definidas para o EXT2. Por esse motivo o SaliusFS deve ser montado sobre partições EXT2. Na próxima seção vamos explicar a diferença nos modos de operação entre os dois sistemas.

4.3.2 Modos de operação

O sistema SaliusFS da suporte aos mesmos modos de operação do sistema EXT2, mas implementa essa funcionalidade de maneira distinta. Assim como no EXT2, existem quatro formas de se estabilizar os dados de um arquivo: usando gravação síncrona através da diretiva *O_SYNC*, marcando o diretório como síncrono, montando toda a partição como síncrona, ou gravando de forma assíncrona e executando-se a chamada de sistema *fsync(fd)* ao final da operação.

Para as três primeiras formas de gravação o sistema SaliusFS opera da seguinte forma a cada execução da chamada de sistema write(fd,dados): Os dados são gravados no arquivo de forma **assíncrona** inicialmente, após o que o sistema envia através de *UDP multicast* um pedido de replica dos dados gravados; após terem sido recebidas suficientes confirmações de replicas o sistema desbloqueia a aplicação que solicitou a gravação.

Para dar suporte à quarta forma de se estabilizar os dados, através da chamada *fsync(fd)* o sistema Salius teve de sofrer uma adaptação especial que gerou uma pequena limitação. Não havia forma de se diferenciar uma aplicação que estava usando gravação assíncrona para depois estabilizar os dados através de *fsync(fd)* daquelas que usavam gravação assíncrona por não usarem gravação estável. A opção de se implementar a estabilização dos dados no momento da chamada *fsync* era por demais custosa e teve de ser contornada. A alternativa foi usar replicação de dados nas gravações assíncronas e criar uma chamada *fsync(fd)* que não realiza operação alguma. Dessa forma, garantimos a estabilidade dos dados gravados, mas penalizamos as aplicações que não precisam dessa estabilidade.

Essa alternativa foi necessária para se dar o suporte a essa forma de estabilização de dados, que é muito usada por certas categorias de aplicações, inclusive por servidores de correio eletrônico. Espera-se que essas formas de gravação que utilizam apenas memória volátil e replicação em rede tenham uma performance melhor que aquela apresentada pelas soluções baseadas em gravação síncrona em disco para estabilização de dados.

Nas próximas seções vamos detalhar um pouco mais a forma de implementação do sistema SaliusFS, explicando os mecanismos de replicação e recuperação de dados

4.3.3 Serviço de Replicação de Dados

O serviço de replicação de dados implementa um protocolo que garante a estabilidade dos dados operação a operação através de cópias remotas em memória volátil ou gravação síncrona sempre que a replicação não estiver disponível. O serviço depende de duas instâncias básicas: o cliente (núcleo do sistema operacional com suporte a SaliusFS), máquina onde está baseado o sistema de arquivos em disco; e um grupo de servidores, que são aplicações (*Daemons*) que executam em máquinas remotas.

Os servidores ficam bloqueados aguardando mensagens vindas dos clientes. Essa mensagens podem ser de dois tipos:

- Pedido de réplica: nessa mensagem o cliente envia um bloco de dados para ser armazenado no servidor, o servidor deve responder com uma mensagem de confirmação se a mensagem tiver sido corretamente recebida e armazenada.

- Pedido de recuperação: o servidor deve responder essa mensagem enviando os blocos replicados em sua memória, referentes a o cliente que enviou a mensagem de recuperação.

Esses servidores guardam os blocos provenientes dos clientes remotos em uma lista encadeada, mantendo informações de tempo de permanência das mesmas. Para evitar que o servidor esgote a sua capacidade de memória, um protocolo de limpeza, descrito na especificação do sistema de arquivos SaliusFS [BCPS02], é executado de forma a excluir dados não mais necessários e manter seguros dados ainda potencialmente vulneráveis.

Os clientes operam da seguinte forma: toda vez que uma operação de gravação segura é requisitada, os dados são inicialmente gravados de forma assíncrona no sistema de arquivos, ou seja, são , na maioria dos casos, armazenados em memória volátil local. Após essa gravação, esses dados são enviados para o grupo de servidores através de *UDP multicast*. O cliente , então espera um número mínimo de confirmações de que a réplica foi armazenada nos servidores, para então desbloquear a aplicação.

Caso o número de réplicas não seja atingido, o sistema espera um tempo máximo por mais respostas e se essas não chegam, procede a gravação dos dados de forma síncrona para garantir sua estabilidade.

Esse protocolo garante que os dados referentes a uma gravação fiquem seguros por estarem presentes em um grupo de máquinas até que possa ser armazenado em disco. Quanto maior o número de réplicas exigido, maior a confiabilidade nas operações de gravação.

4.3.4 Serviço de Recuperação de Dados

Caso ocorra uma falha na máquina hospedeira do sistema de arquivos, levando a uma reinicialização, um procedimento de recuperação dos dados remotos precisa ser executado. Como não é possível saber *a priori* se existem ou não dados replicados que ainda não tenham sido gravados em disco de forma segura torna-se necessária a execução dos seguintes procedimentos de checagem:

- Envio de um pedido de recuperação por parte do cliente para o grupo de servidores de replica (*UDP Multicast*);
- Aguardo pela chegada de todas as mensagens vindas dos servidores de replica até que todos os servidores do grupo tenham enviado uma mensagem indicando a não existência de novas réplicas;
- Ordenação das réplicas recebidas pelo *time stamp* existente nas mesmas;
- Gravação segura dos dados e meta-dados contidos nessas replicas em disco;
- Envio de mensagem indicando fim da operação de recuperação para o grupo de servidores de réplica para que o espaço ocupado por essas réplicas seja liberado.

Como, em cada mensagem de replicação, são enviados todos os meta-dados relativos ao arquivo, esse protocolo garante que, após a execução da última mensagem de replicação, os meta-dados serão os mesmos vistos pela aplicação no momento da última operação de gravação estável. Observe que para garantir a estabilidade dos meta-dados, é preciso uma operação de gravação estável sobre o arquivo. Caso tenha sido feita uma operação, de forma assíncrona, exclusivamente sobre meta-dados, o protocolo não garante a estabilidade da mesma.

Para validar esse protocolo de recuperação de dados, foi implementada uma ferramenta que usamos na inicialização do sistema sempre que o mesmo tiver sido desligado acidentalmente ou esteja se recuperando de uma falha por *crash*. Esta ferramenta se encarrega de

enviar a mensagem de pedido de recuperação ao grupo de replicas, ordenar as mensagens recebidas e escrever os dados, de forma segura, em disco.

Foram executados dois conjuntos de experimentos para se verificar o funcionamento da ferramenta de recuperação de dados. No primeiro, uma aplicação gravava, sequencialmente, 100Kbytes de dados em um arquivo de texto de forma segura. Em seguida, a máquina era desconectada da alimentação elétrica, forçando-se uma parada semelhante a uma falha por *crash*. Ao se reiniciar, foi verificado que de cada 5 execuções, em 4 o arquivo não havia sido estabilizado ainda em disco. A ferramenta de recuperação era, então, executada, inclusive nos casos em que o arquivo já havia sido estabilizado antes da parada, para que os dados fossem recuperados a partir das réplicas. Foi observado que, em todos os experimentos o arquivo foi totalmente recuperado pela ferramenta.

Um segundo tipo de experimento foi executado para se verificar a capacidade do sistema de arquivo Salius de lidar com aplicações concorrentes. Dois aplicativos idênticos, iniciados simultaneamente, gravavam dados, de forma segura, sobre um sistema de arquivos Salius. de forma semelhante ao experimento anterior, a máquina era desligada abruptamente, impedindo o sistema de estabilizar os dados em disco. Novamente a ferramenta de recuperação conseguiu recuperar os dados dos diferentes arquivos em todas as tentativas.

4.4 Gravação estável passo a passo

Nesta seção vamos explicar com alguns detalhes como funciona uma gravação estável em cada um dos três sistemas de arquivos para termos uma estimativa do custo relativo a cada passo.

4.4.1 Abertura do arquivo

Sempre que uma aplicação abre um arquivo, seja somente para leitura, seja para gravação, algumas operações são realizadas. Caso o arquivo não exista, ele deve ser criado, caso exista, deve ser localizado na estrutura de diretórios e ter os seus meta-dados carregados na memória.

Os sistemas EXT2 e SaliusFS utilizam a mesmas estruturas de disco, portanto realizam essas operações da seguinte forma: i) o arquivo é localizado na estrutura de diretórios de

forma linear, partindo-se do diretório raiz até chegar ao sub-diretório onde o arquivo se encontra. Essa operação pode ser um pouco longa caso o arquivo se encontre dentro de uma estrutura de diretórios muito complexa com vários níveis. Caso o arquivo não exista o primeiro passo é procurar no superbloco por um nó-i livre e alocar o mesmo para esse arquivo. Em seguida, os meta-dados desse arquivo devem ser gravados no nó-i alocado. O último passo é a localização do diretório onde o arquivo vai ser gravado e a geração de uma entrada de diretório apontando para o arquivo. Observe que essa operação pode ser recursiva caso o diretório não exista ainda e precise ser criado em tempo de execução.

Para o sistema de arquivos ReiserFS as operações são as mesmas, estando a grande diferença apenas na localização do diretório que é realizada em uma árvore balanceada com profundidade conhecida não maior que cinco, por definição. Dessa forma, evitam-se longos procedimentos de localização em estruturas de diretórios com profundidades grandes. As estruturas de meta-dados do ReiserFS incluem uma informação a mais a ser gravada, a chave de pesquisa, mas nesse momento, isso ainda não deve comprometer seu desempenho. Observe que, caso a árvore atual esteja saturada, todo um novo braço deve ser criado, adicionando-se inclusive um novo nó raiz. Isso ocorre com frequência em uma partição ReiserFS recém formatada que ainda não está preenchida com arquivos.

No sistema SaliusFS essas operações poderiam ser executadas de forma assíncrona, mas por questões de consistência no sistema de arquivos, foram mantidas de forma síncrona já que espera-se que não tenham um impacto tão grande na performance.

4.4.2 Alocação de blocos

Após o arquivo ter sido aberto ou criado e suas estruturas de meta-dados estarem todas carregadas na memória, inclusive os ponteiros para o nó-i, superbloco e diretórios, os blocos para gravação de dados devem ser alocados. Observe que essa operação pode ser realizada de duas formas: i) Mudando-se temporariamente o *offset* do arquivo para que o sistema pré-aloque espaço para a gravação dos dados, e ii) Alocando-se os blocos em tempo de execução das operações de gravação em si, onde os blocos são alocados passo a passo de acordo com o espaço demandado a cada operação.

Cada operação de alocação de espaço para um arquivo requer as seguintes modificações de estruturas no sistema EXT2: Marcação, no mapa de bits de blocos livres do superbloco,

dos blocos, necessários para a gravação da quantidade de dados requisitada, como usados; criação de um ponteiro para esse bloco, no nó-i do arquivo, ou em um ponteiro indireto; e a gravação no nó-i do arquivo do novo *offset*.

No sistema ReiserFS essa operação requer a mesma gravação no mapa de bits do superbloco, a gravação dos ponteiros para os blocos na estrutura equivalente ao nó-i e o novo *offset*. Até aqui essa operação foi semelhante nos dois sistemas, entretanto o ReiserFS pode alocar um único bloco de tamanho grande para se gravar os dados, funcionalidade que os outros sistema de arquivo não oferecem. Isso pode levar a uma melhor performance na gravação de dados com blocos previamente alocados.

No sistema SaliusFS esse passo não requer nenhuma operação direta em disco, visto que a informação contida nas replicas pode recuperar os dados necessários à realização do mesmo.

4.4.3 Gravação dos dados nos blocos

Nesse passo, o sistema de arquivos já teve o espaço alocado para a gravação dos dados e a única tarefa restante é a gravação dos mesmos.

Para um sistema de arquivos EXT2 isso ocorre na seguinte ordem: i) os dados são gravados nos blocos necessários, observe que no caso de uma operação de gravação sobre uma quantidade relativamente pequena de dados (32Kbytes) isso pode levar à alocação e gravação em oito (8) blocos diferentes. em um sistema fragmentado pelo uso isso pode levar a um custo relativamente grande. O último passo é a gravação do novo *offset* no nó-i do arquivo.

No sistema SaliusFS a operação de gravação ocorre de forma assíncrona, não necessitando de nenhuma operação em disco, a não ser nas fases anteriores. É nesse momento que a replicação de dados ocorre. Inclusive o novo *offset* não é efetivamente gravado no nó-i até que ocorra uma sincronização programada pelo escalonador de disco.

No sistema ReiserFS esse passo corresponde à gravação efetiva dos dados nos blocos, entretanto observe que mesmo uma gravação de uma quantidade relativamente grande de dados (256Kbytes por exemplo) pode ser efetuada em um único bloco sequencial caso este tenha sido pré alocado.

4.5 Conclusão

Neste capítulo fizemos uma análise das implementações dos três sistemas de arquivo que foram utilizados em nossos testes.

Mostramos as diferentes estruturas organizacionais que esses sistemas de arquivo utilizam para manter as informações referentes aos dados gravados e a semântica de gravação estável de cada um deles. Certos tópicos, como por exemplo a semântica de gravação de meta-dados em operações assíncronas, são bastante controversos e exigem uma discussão extensa demais para ser incluída neste trabalho. Preferimos, portanto, considerar apenas o caso onde se utiliza exclusivamente operações síncronas.

No final do capítulo incluímos uma análise passo a passo de uma seqüência de gravação de dados nos três sistemas de arquivo. Identificamos, nessa análise, os pontos críticos de utilização de disco nos três sistemas e possíveis melhorias para o sistema SaliusFS. Vamos agora apresentar os experimentos realizados sobre os sistemas para estudar o desempenho dos mesmos.

Capítulo 5

Resultados e Análise

Neste capítulo iremos descrever os experimentos que foram conduzidos para avaliar a performance dos sistemas de armazenamento estável escolhidos. Também iremos apresentar os resultados obtidos nesses experimentos e analisar esses resultados com o objetivo de melhor entender o funcionamento dessas abordagens para armazenamento estável de dados.

Inicialmente, iremos descrever o ambiente de testes utilizado. Em seguida, faremos uma descrição detalhada dos dois procedimentos de avaliação de desempenho, explicando o objetivo específico de cada um deles.

Na segunda parte do capítulo explicaremos os detalhes dos resultados obtidos em cada teste, fazendo uma análise dos mesmos.

5.1 Ambiente de testes

Nosso ambiente de testes consistiu-se de uma rede local de microcomputadores do tipo PC com a seguinte configuração:

- Processadores AMD Duron 800MHz
- 128Mbytes de memória RAM (tempo de acesso de 5ns)
- Discos rígidos de 20Gb e 5400RPM
- Adaptadores de rede ethernet de 100Mbits/segundo

Os microcomputadores estavam interligados através de um barramento compartilhado [1] de 100Mbits/segundo, isolado de qualquer outra fonte de tráfego. Como apenas os microcomputadores usados nos testes estavam ligados a esse barramento, pudemos manter a carga submetida à rede sob controle.

Todos os microcomputadores estavam executando o sistema operacional Red Hat Linux 7.2 com o kernel 2.2.19-salius [2] e nenhum serviço de rede desnecessário foi iniciado, evitando, assim, a ocorrência de tráfego extra na rede e concorrência excessiva pelo uso dos processadores.

Os microcomputadores estavam organizados da seguinte forma:

- Um microcomputador principal onde os sistemas de arquivo eram testados;
- Dois (2) microcomputadores executando servidores de replicação Salius;
- Um microcomputador usado para inserir carga na rede e no microcomputador principal através de NFS;
- Um microcomputador para gerar as cargas para o servidor SMTP e medir o desempenho do mesmo sob o ponte de vista do cliente.

5.2 Descrição dos experimentos

Dois conjuntos de experimentos foram executados nessa avaliação: o primeiro consistiu de avaliações sintéticas baseadas em ferramentas de *micro-benchmark* para sistemas de arquivo e o segundo consistiu da comparação do desempenho de um servidor SMTP, que utiliza armazenamento estável, executando sobre os três sistemas de arquivo sob avaliação.

Para os experimentos sobre o sistema de arquivos Salius, restringimos sua operação ao seguinte caso: dois servidores de replicação executando e o cliente aguardando apenas uma confirmação de armazenamento remoto. Essa escolha foi baseada nos resultados obtidos previamente que mostraram esse como sendo o caso médio na curva de desempenho desse sistema [BCPS02]. A variância entre os casos sempre foi estatisticamente deseconsiderável de forma que entendemos essa restrição como não prejudicial à validade dos resultados.

[1] HUB sem comutação de *frames*

[2] Kernel 2.2.19 customizado com o suporte ao sistema de arquivos Salius.

5.2.1 Experimentos sintéticos

O objetivo desse conjunto de experimentos é ajudar a compreender como diversas variáveis (tamanho de arquivos e de registros, intervalo entre operações etc.) influenciam no desempenho das operações de gravação estável nos sistemas de arquivo testados.

Dividimos esse conjunto de experimentos em duas etapas:

1. Avaliações livres de interferência externa

2. Avaliações com interferência externa

Na primeira etapa os sistemas de arquivos foram testados em um ambiente livre de qualquer interferência que pudesse afetar o desempenho das operações de gravação estável, ou seja, não existia concorrência pelo uso do disco, da rede, ou do processador. Nessa etapa avaliamos a influência das variáveis relacionadas diretamente com a operação de gravação em si, ou seja, tamanho dos arquivos e registros utilizados, etc.

Na segunda etapa procuramos entender a forma como interferências externas afetam o desempenho desses sistemas de armazenamento estável. Para testar a influência do acesso concorrente ao disco e à rede executamos scripts em uma máquina remota que montava um diretório da máquina principal de testes através de NFS e executava operações de leitura e gravação sobre esse diretório gerando uma carga de uso. Tal carga consiste em 80% de leituras e 20% de gravações. As gravações foram feitas através do comandos *cat* enviando o conteúdo de um arquivo pequeno (64Kbytes) previamente criado em outro disco para um novo arquivo no disco onde se desejava criar a carga de gravação. As leituras eram obtidas através do mesmo comando sendo que a partir do disco onde se desejava criar a carga de leitura. O *script* criou essas cargas alternando entre gravações e leituras nas proporções indicadas e remontando o sistema de arquivos periodicamente para evitar o uso exclusivo da memória *cache* do sistema operacional. Como o sistema de arquivos sendo testado na máquina principal era exportado através de NFS pudemos combinar o acesso concorrente à rede e ao disco em apenas uma instância de experimentos.

A ferramenta utilizada para esse conjunto de testes sintéticos foi o *IoZone benchmark tool* [3]. As duas etapas de testes sintéticos foram executadas da seguinte forma:

[3] www.iozone.org

Cada sistema de arquivos a ser avaliado passou por uma bateria de experimentos onde a influência das seguintes variáveis era observada:

1. Tamanho do arquivo (32, 320, 3200 e 32000Kbytes)
2. Tamanho do registro de gravação (2, 4, 8, 16 e 32Kbytes)
3. Forma de alocação dos blocos de arquivo (pré alocados ou alocados em tempo de execução)

Em seguida, a mesma bateria de testes era executada simultaneamente com o uso de NFS para medir a influência da utilização concorrente do sistema de arquivos.

5.2.2 Experimentos com um servidor SMTP

Resultados obtidos através de ferramentas de análise sintéticas nos dão uma visão geral do comportamento dos sistemas. São úteis para entendermos como diversos fatores alteram o desempenho de tais sistemas. Entretanto, nem sempre essas ferramentas representam corretamente a carga à qual esses sistemas seriam submetidos em ambientes de produção reais.

Nossa idéia foi avaliar a performance de uma aplicação real que usa armazenamento estável de dados, ao ser submetida a uma carga, obtida previamente através do monitoramento dessa mesma aplicação em um ambiente de produção. Escolhemos um servidor de correio eletrônico por ser uma aplicação bastante comum que tem um comportamento semelhante a um sistema baseado em transações no que diz respeito à utilização de um sistema de armazenamento estável.

As cargas a que submetemos o nosso servidor SMTP de teste foram obtidas a partir dos logs de utilização do servidor *anjinho.dsc.ufpb.br* entre os dias 17 de fevereiro e 30 de abril de 2002. Comparamos esses logs com outros logs de servidores de correio eletrônico obtidos da Light Infocon e da AESPI (Associação de Ensino Superior do Piauí) e observamos que os gráficos estatísticos de tamanho e intervalo entre as mensagens são auto similares em relação ao tempo, ou seja, o gráfico de tamanho e intervalo entre mensagens obtido com a análise dos logs relativos a um dia de utilização do servidor são semelhantes aos gráficos obtidos através da análise do log completo de três meses.

A partir dessa análise inicial resolvemos executar nossos experimentos submetendo nosso servidor de testes a duas cargas, uma delas obtida a partir dos logs do servidor *anjinho.dsc.ufpb.br*, a segunda sendo uma carga sintética. A primeira carga foi obtida a partir do log relativo a um dia que fosse bastante semelhante à carga média do período, a segunda carga foi simulada de forma a testar a saturação de utilização do sistema de correio eletrônico, mandando rajadas de mensagens em tamanhos variados para verificar a vazão suportada pelo servidor.

As medições de performance foram feitas sob o ponto de vista do cliente. Medimos o tempo que o servidor gasta para processar cada mensagem sob o ponto de vista do cliente remoto que a está enviando, analisando, assim, o impacto real no desempenho desses serviços de correio eletrônico observado pelos usuários quando se usam as diversas formas de armazenamento estável.

5.3 Resultados e análise

Nesta seção apresentaremos diversos gráficos e tabelas mostrando todos os resultados obtidos nos experimentos sintéticos e baseados no servidor de correio eletrônico, bem como uma análise desses resultados.

5.3.1 Resultados dos experimentos sintéticos

Inicialmente apresentaremos os resultados individuais para cada um dos sistemas de arquivo testados para verificar a influência dos fatores tamanho do arquivo, tamanho do registro utilizado e existência ou não de concorrência. Em seguida apresentaremos alguns gráficos comparando os resultados obtidos pelos sistemas de arquivo avaliados.

Resultados para o sistema de arquivos Ext2:

O sistema de arquivos Ext2 atinge uma vazão máxima de cerca de 1.5Mbytes/segundo nas operações de gravação de dados, para blocos de até 32Kbytes, devido à alta sobrecar-

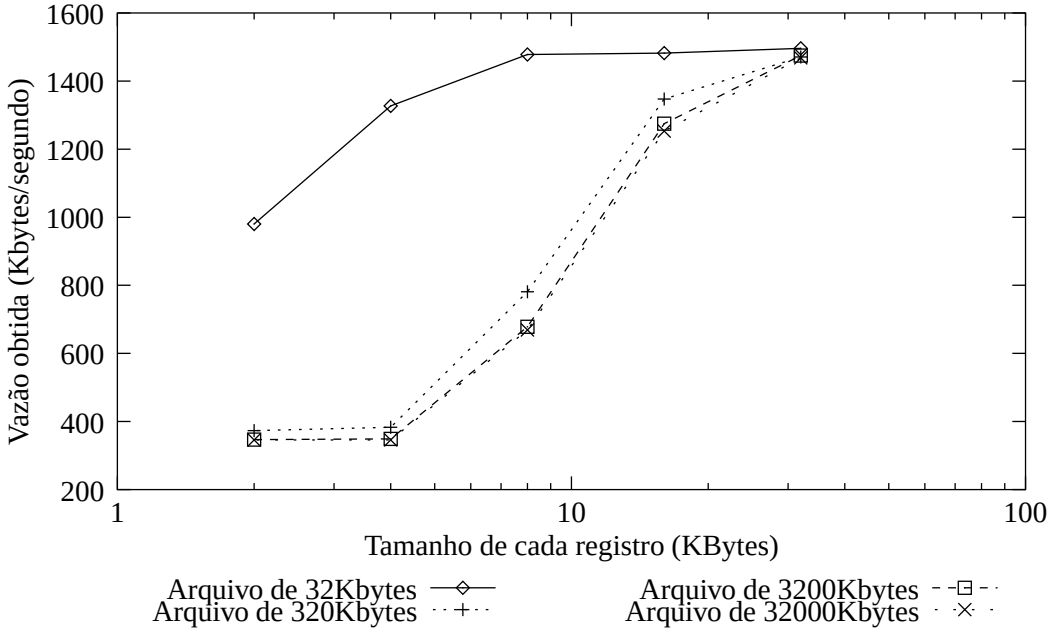

Figura 5.1: Desempenho do sistema Ext2 em gravações síncronas sem concorrência

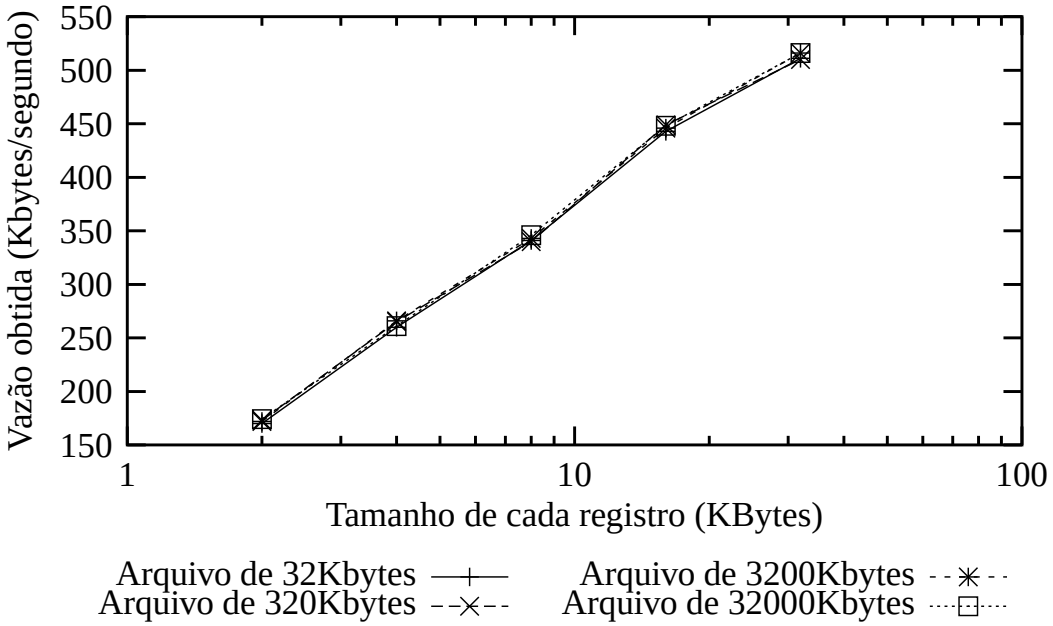

Figura 5.2: Desempenho do sistema Ext2 em gravações síncronas com concorrência

ga imposta ao disco para se gravar os meta-dados a cada operação, uma vez que os blocos da dados necessários para armazenar os mesmos são alocados em tempo de execução e essa operação exige gravação síncrona em disco, isso pode ser observado na figura 5.1.

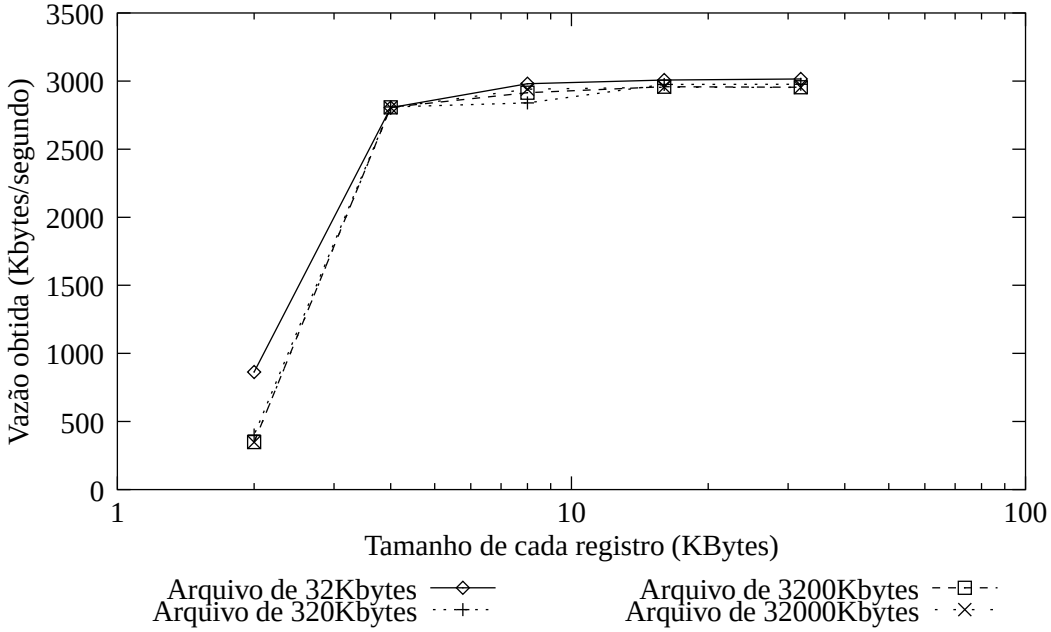

Figura 5.3: Desempenho do sistema Ext2 em regravações síncronas sem concorrência

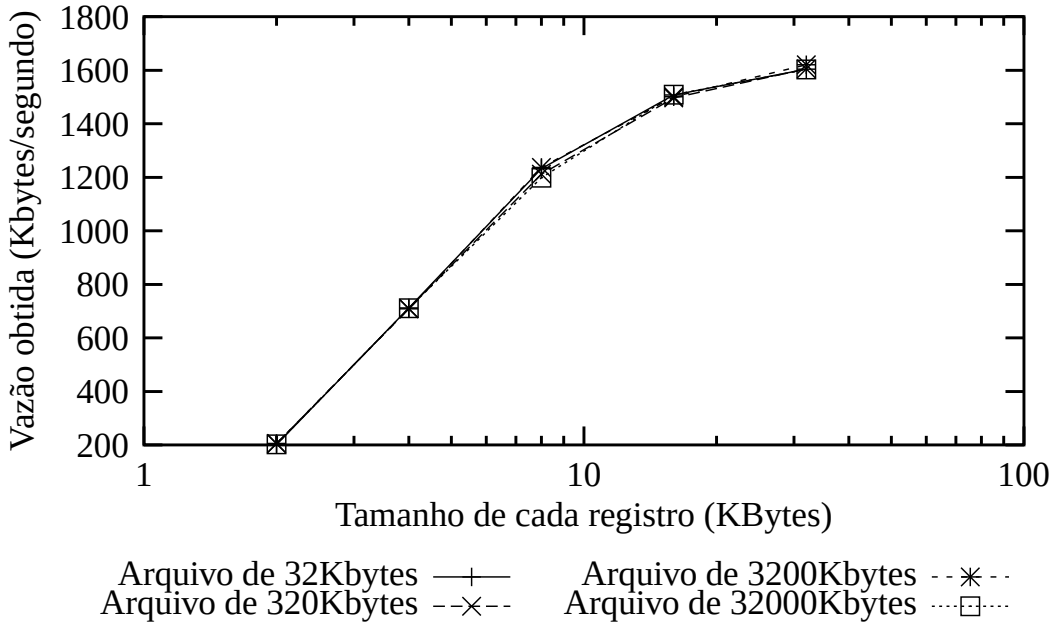

Figura 5.4: Desempenho do sistema Ext2 em regravações síncronas com concorrência

Nas operações de regravação de dados (figura 5.3), com blocos previamente alocados, o sistema Ext2 consegue melhorar sua performance em até 100% atingindo uma vazão máxima de 3Mbytes/segundo. Mesmo nesse caso, ainda existe sobrecarga em cada operação de

gravação devido à necessidade de se gravar informações como o *offset* do arquivo no nó-i correspondente, além dos dados em sí.

A existência de concorrência pelo uso do disco afeta consideravelmente a performance do sistema Ext2, principalmente nas operações de gravação de dados. Nesse caso, como pode ser visto na figura 5.2, a vazão não consegue ultrapassar 500Kbytes/segundo. O desempenho desse sistema sofreu, também, uma considerável queda na performance das regravações chegando a um pico 1.6Mbytes/segundo, observe a figura 5.4.

Resultados para o sistema de arquivos Reiser:

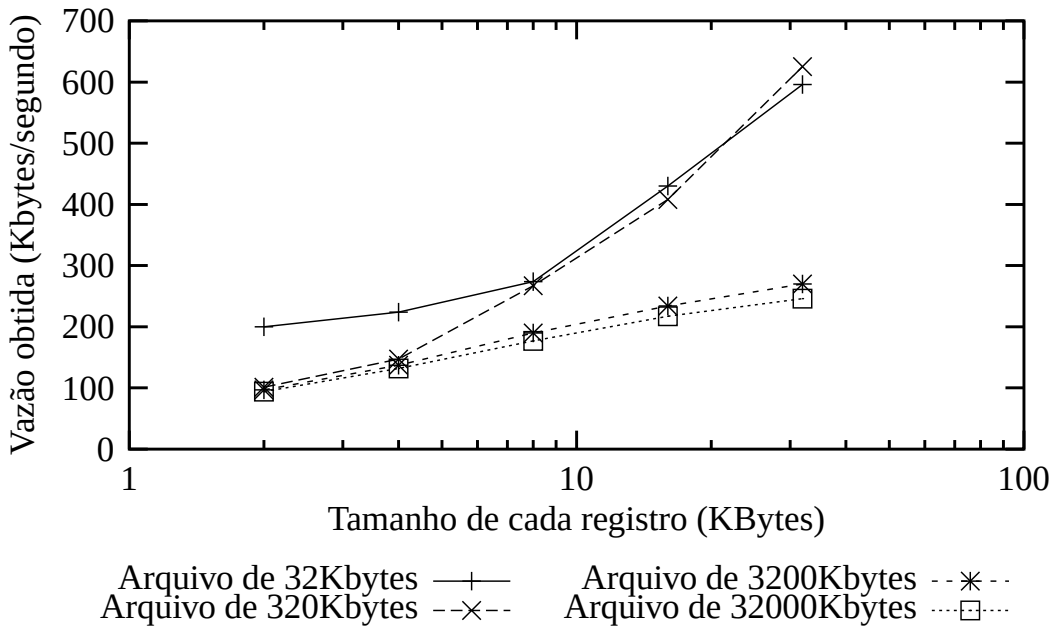

Figura 5.5: Desempenho do sistema ReiserFS em gravações síncronas sem concorrência

Nas operações de gravação livres de concorrência, figura 5.5, o desempenho do sistema é bastante limitado pelo tamanho do registro utilizado, que provoca mais sobrecarga ao sistema operacional pelo maior número de operações. Na gravação de arquivos pequenos (32Kbytes) o sistema ReiserFS havia apresentado, inicialmente, um resultado muito superior em relação ao desempenho do sistema Ext2 e em relação ao seu próprio resultado para outros tamanhos de arquivo. Imaginou-se que isso se devia a uma melhor performance desse sistema nas

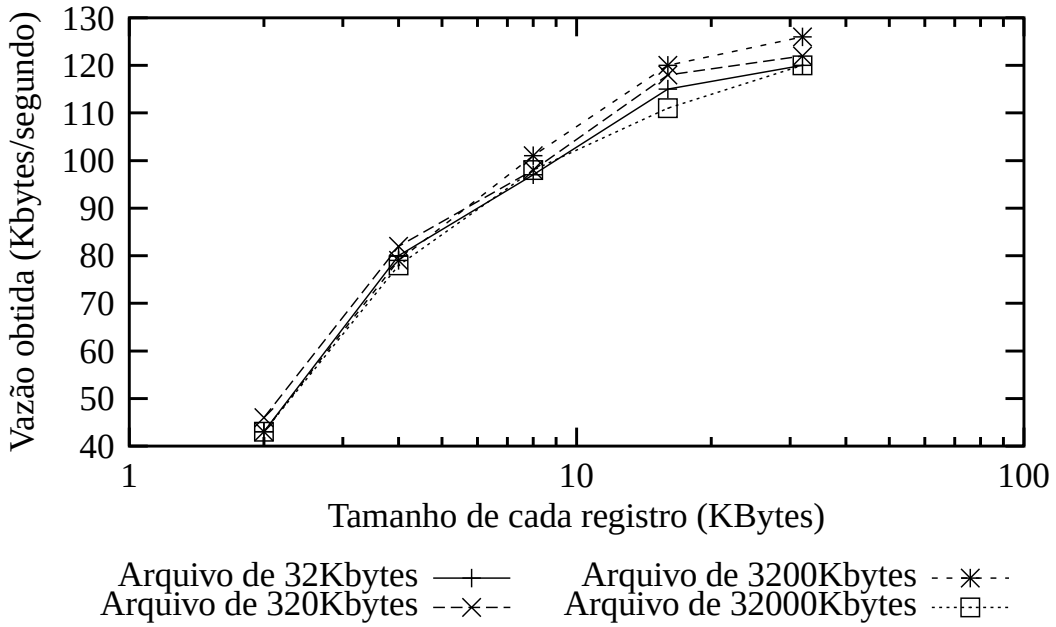

Figura 5.6: Desempenho do sistema ReiserFS em gravações síncronas com concorrência

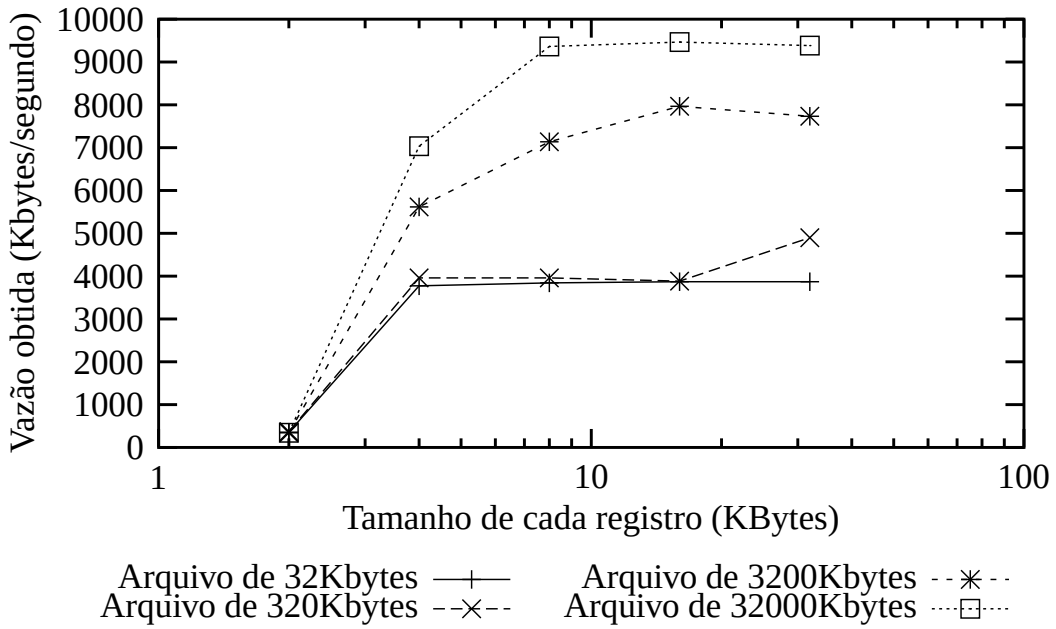

Figura 5.7: Desempenho do sistema ReiserFS em regravações síncronas sem concorrência

operações sobre arquivos pequenos, que é um dos objetivos do projeto do ReiserFS. Após o resultado das execuções completas dos testes verificamos que essa diferença era provocada por um uso eficiente da *cache* existente no disco por parte desse sistema nessas operações (o

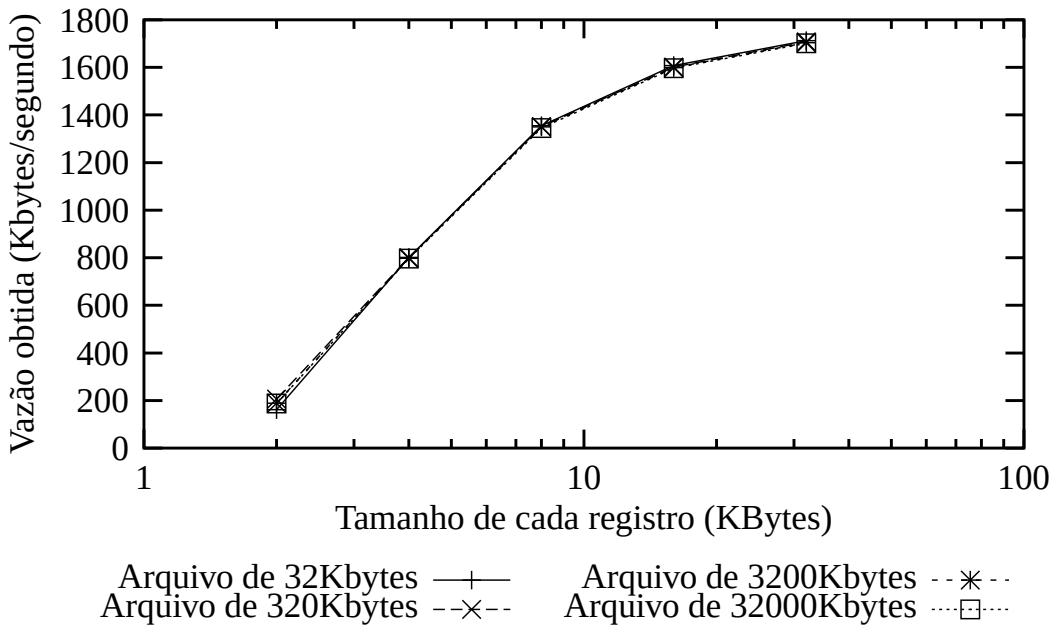

Figura 5.8: Desempenho do sistema Reiser em regravações síncronas com concorrência

sistema Ext2 também teve um ganho nas operações de gravação sobre arquivos de 32Kbytes). De um modo geral, o sistema ReiserFS teve um desempenho inferior àquele obtido pelo sistema Ext2 nas operações de gravação livres de concorrência.

Ao analizar os resultados obtidos por esse sistema de arquivos nas operações de regravação, Figura 5.7, podemos observar duas coisas interessantes: i) a performance na gravação dos arquivos de 32Kbytes foi extremamente alta em algumas instâncias de experimentos, até ultrapassando a vazão máxima do disco, chegando a valores de quase 30Mbytes/s, o que nos levou a executar novos testes que mostraram que isso era resultado da utilização da *cache* existente no disco (512Kbytes); e ii) Quanto maior o registro, bem melhor será a performance de regravação nesse sistema. Isso se deve ao fato de que ao se pré-alocar blocos no reiserFS, este faz uso de grandes blocos não formatados [4], permitindo que o disco atinja sua máxima vazão. O desempenho desse sistema nas regravações livres de concorrência chegou a ser três vezes superior àquele obtido pelo sistema Ext2.

Ao se adicionar concorrência ao uso do disco o sistema ReiserFS sofre uma perda de desempenho semelhante àquela sofrida pelo sistema Ext2, mantendo uma performance novamente inferior nas operações de gravação, figura 5.6. Nas operações de regravação o

[4] semelhantes aos *Extents* usados em outros sistemas de arquivos UNIX

impacto é maior que aquele observado no Ext2, levando o desempenho a se equiparar àquele obtido pelo primeiro sistema de arquivos, observe a figura 5.8. Aparentemente, a adição de concorrência no uso do disco limitou a performance do ReiserFS a um patamar de 1.7Mbytes/segundo, desde que não exista outro tipo de sobrecarga nas operações, isso levou o sistema a se equiparar ao EXT2 na regravação de dados, caso onde era bastante superior sem a existência de concorrência.

Resultados para o sistema de arquivos Salius:

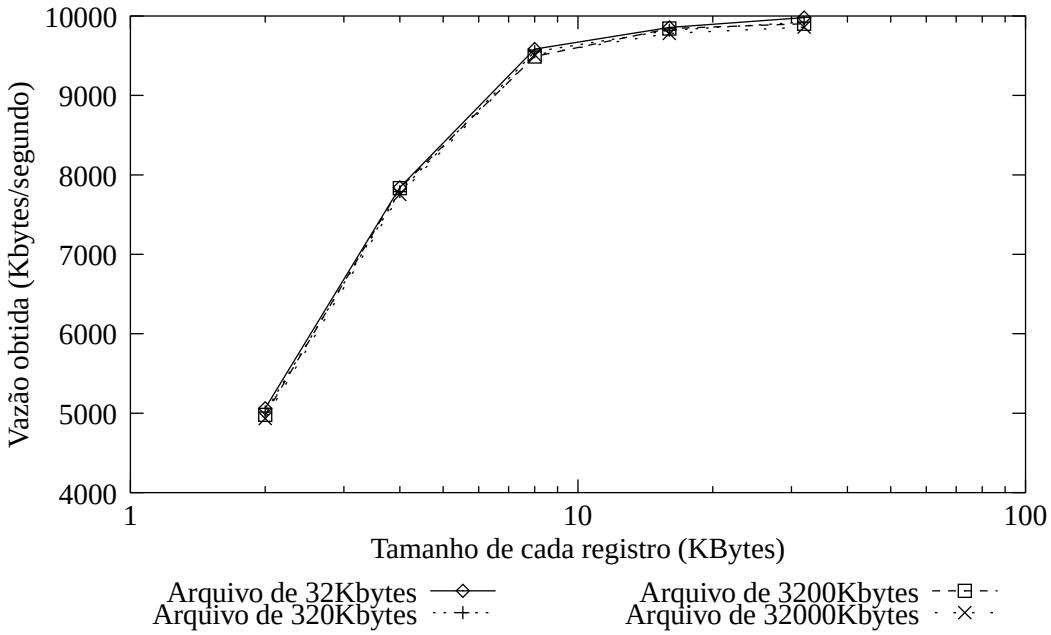

Figura 5.9: Desempenho do sistema Salius em gravações síncronas sem concorrência

O sistema de arquivos Salius tem um comportamento bem mais estável que os demais sistemas. Isso se reflete na menor variância encontrada nas diversas instâncias do mesmo teste que eram executadas para se chegar a um resultado médio. Como esse sistema faz todas as operações sobre dados e meta-dados relativas às chamadas de sistema *write* de forma assíncrona, o fator limitante para a performance do mesmo passa a ser a vazão da rede e a sobrecarga imposta pelo tamanho dos registros utilizados.

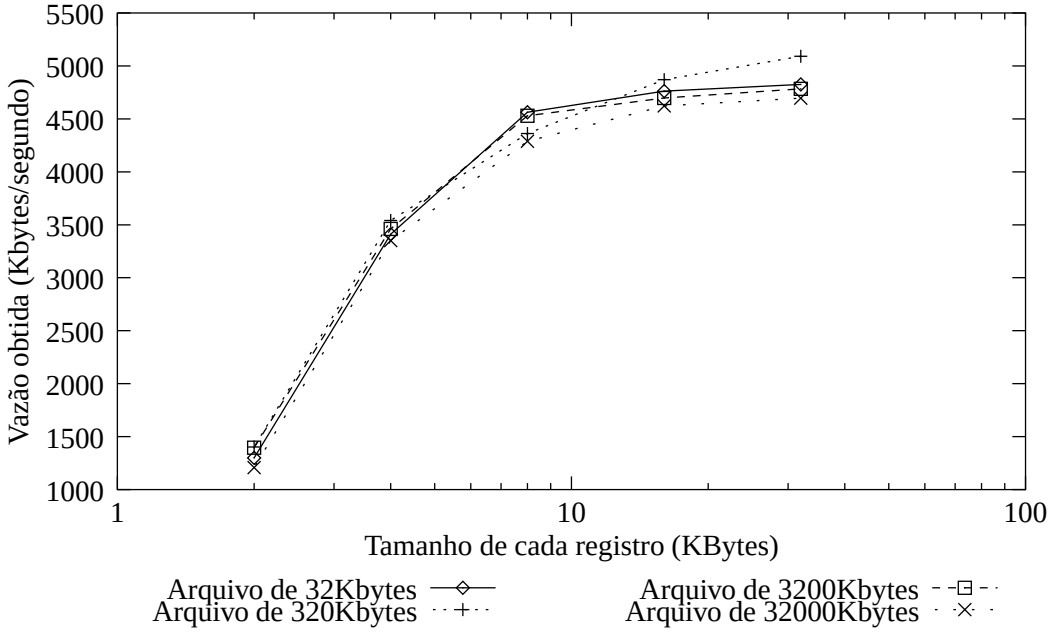

Figura 5.10: Desempenho do sistema Salius em gravações síncronas com concorrência

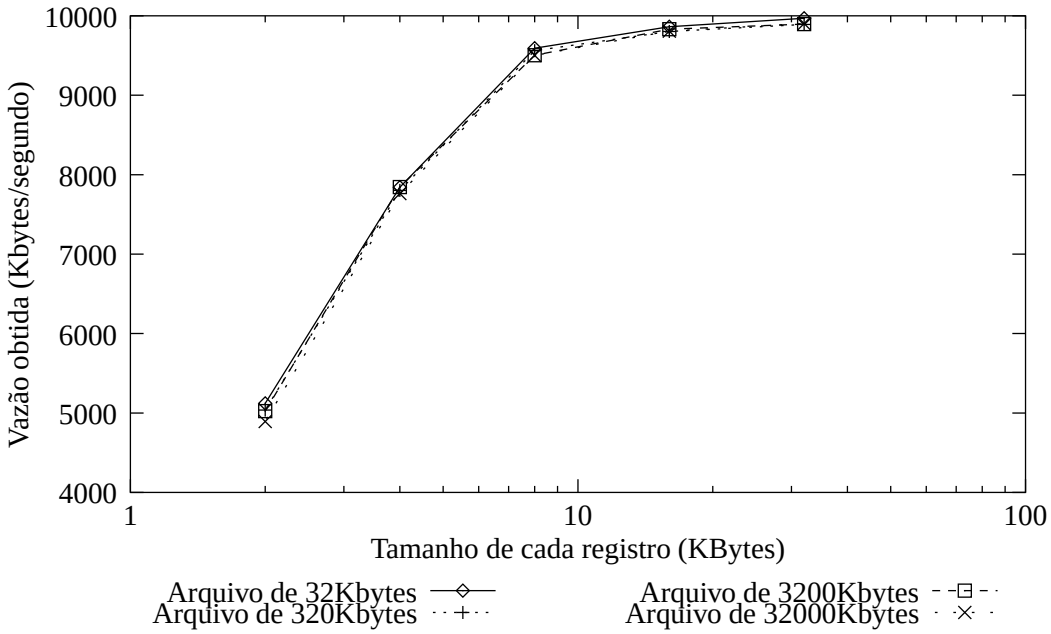

Figura 5.11: Desempenho do sistema Salius em regravações síncronas sem concorrência

Como já era esperado, devido aos resultados prévios encontrados em nosso trabalho anterior com esse sistema [BCPS02], as operações de gravação e regravação de dados têm desempenho extremamente semelhante (figuras 5.9 e 5.11), já que a pré alocação de blocos não têm influência sobre uma operação que vai ser realizada sem o uso síncrono do disco.

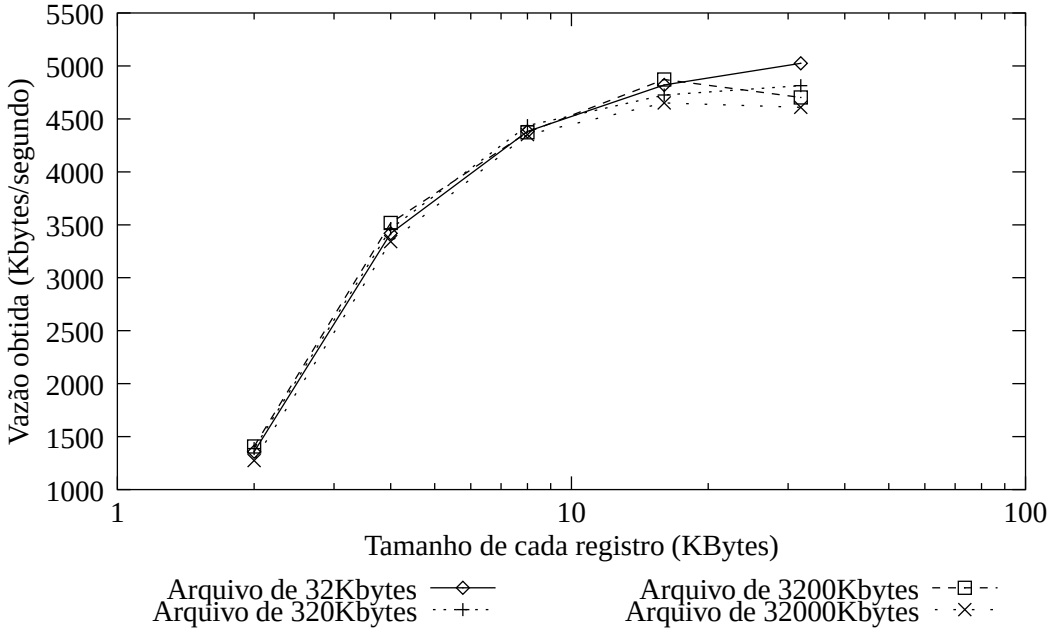

Figura 5.12: Desempenho do sistema Salius em regravações síncronas com concorrência

Em todos os testes livres de concorrência (figuras 5.9 e 5.11), o principal fator limitante de performance foi a sobrecarga imposta pelo tamanho do registro. A vazão máxima chegou a pouco menos que 10Mbytes/segundo, o que corresponde a 80% da vazão máxima de uma rede *Fast Ethernet*.

A concorrência provocada pela carga imposta através de NFS degradou consideravelmente o desempenho em todas as operações do sistema salius, mas essa degradação foi inferior à observada nos experimentos com os outros sistemas de arquivo. A vazão máxima caiu cerca de 50% atingindo um pico de pouco menos que 5Mbytes/segundo, isso pode ser constatado nas figuras 5.10 e 5.12.

Gráficos comparativos de desempenho:

Para comparar o desempenho dos três sistemas criamos gráficos que mostram a vazão obtida por estes separados por tamanho de arquivo, existência de concorrência e tipo de operação (gravação ou regravação). Incluímos apenas os resultados para dois tamanhos de arquivos, 32Kbytes e 32000Kbytes por serem estes os que trouxeram informações mais significativas.

Gráficos comparando a performance de gravação

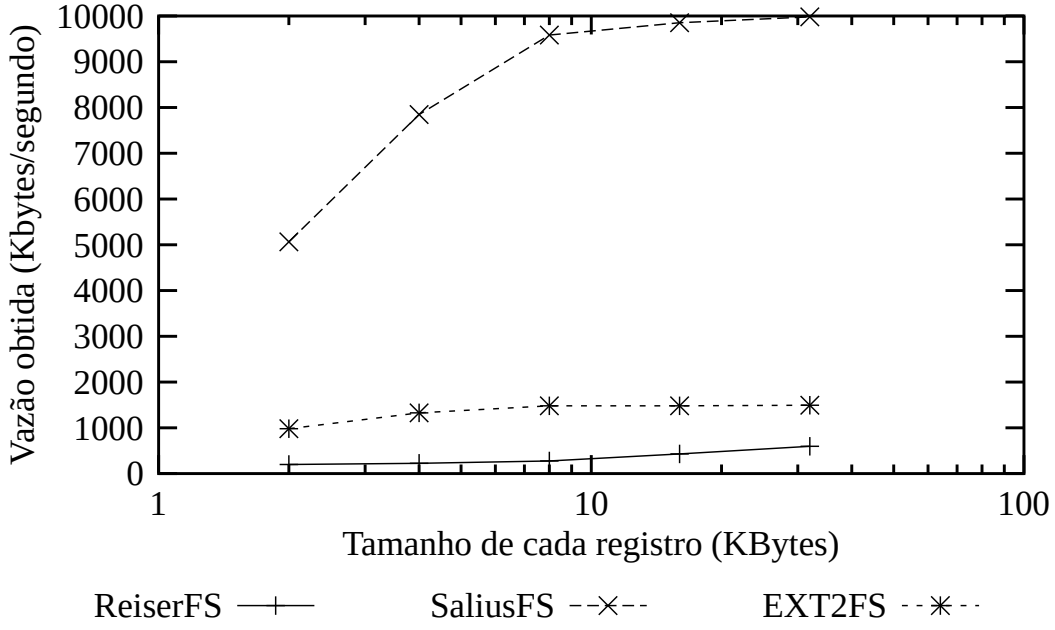

Figura 5.13: Desempenho comparativo para gravações síncronas de um arquivo de 32Kbytes sem concorrência

Podemos observar nesses gráficos que o sistema de arquivos salius obteve um desempenho bastante superior aos demais nas operações de gravação de dados, principalmente quando existia a presença de forte concorrência pelo uso do disco quando esse sistema chegou a ser 9 vezes mais ágil que os demais (figuras 5.14 e 5.16). Sem a existência de concorrência o sistema Salius chegou a ser de 3 a 6 vezes mais rápido, dependendo do tamanho do arquivo sendo gravado (figuras 5.13 e 5.15).

Gráficos comparando a performance de regravação

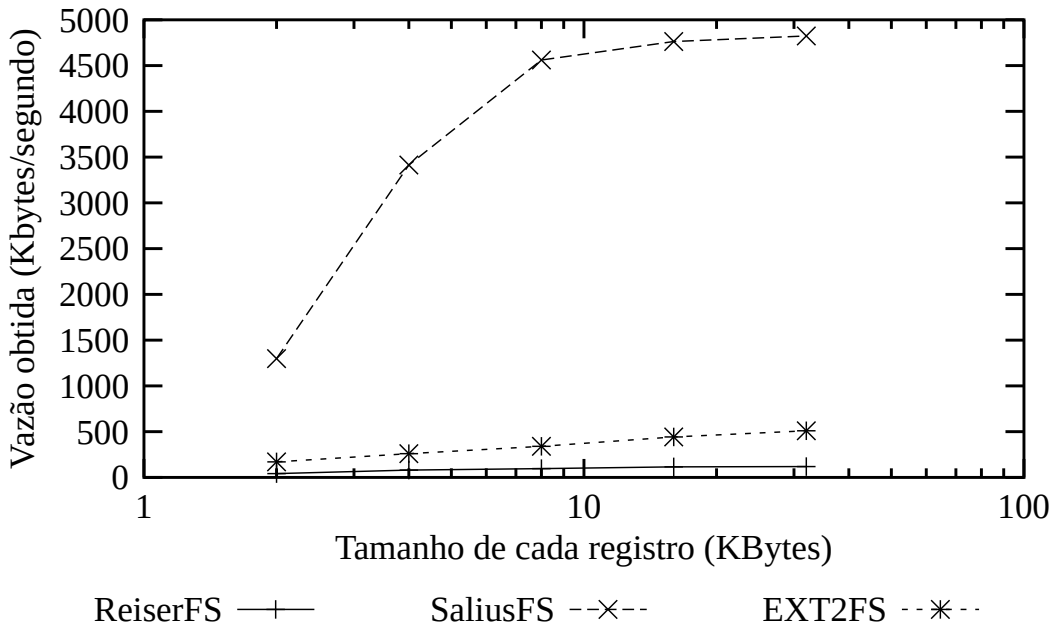

Figura 5.14: Desempenho comparativo para gravações síncronas de um arquivo de 32Kbytes com concorrência

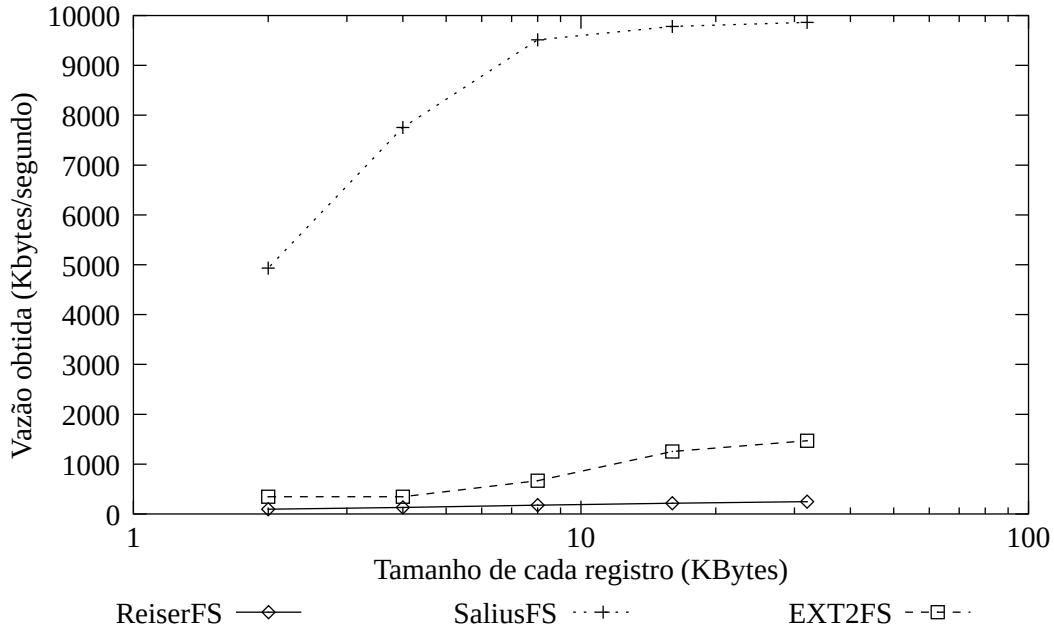

Figura 5.15: Desempenho comparativo para gravações síncronas de um arquivo de 32000Kbytes sem concorrência

Nos experimentos com regravação de dados temos um quadro bastante diferente, onde o sistema de arquivos Reiser chega a rivalizar o sistema Salius na performance de regra-

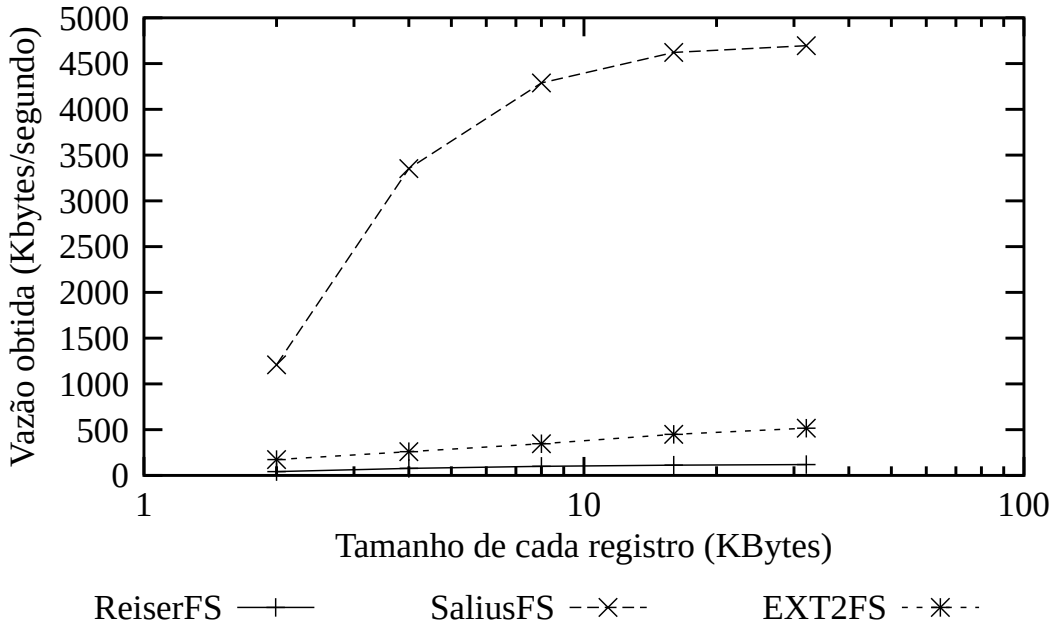

Figura 5.16: Desempenho comparativo para gravações síncronas de um arquivo de 32000Kbytes com concorrência

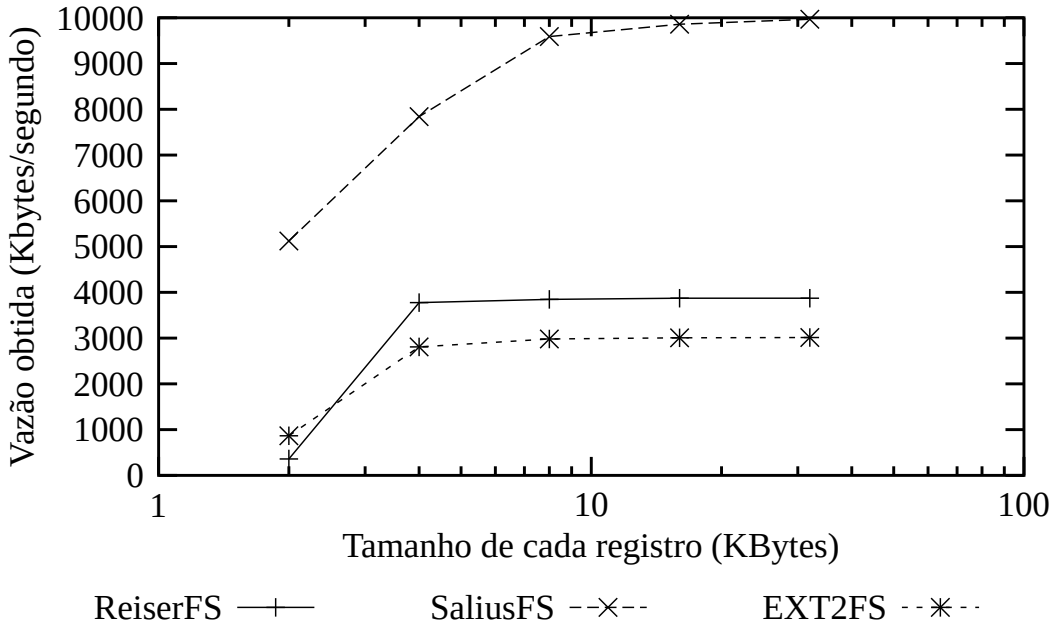

Figura 5.17: Desempenho comparativo para regravações síncronas de um arquivo de 32Kbytes sem concorrência

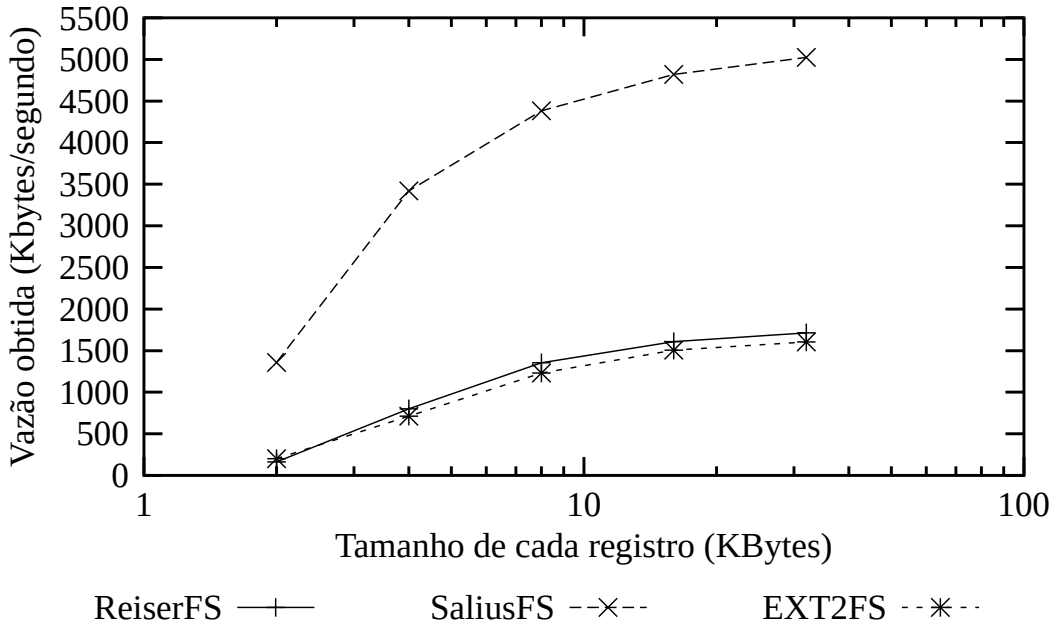

Figura 5.18: Desempenho comparativo para regravações síncronas de um arquivo de 32Kbytes com concorrência

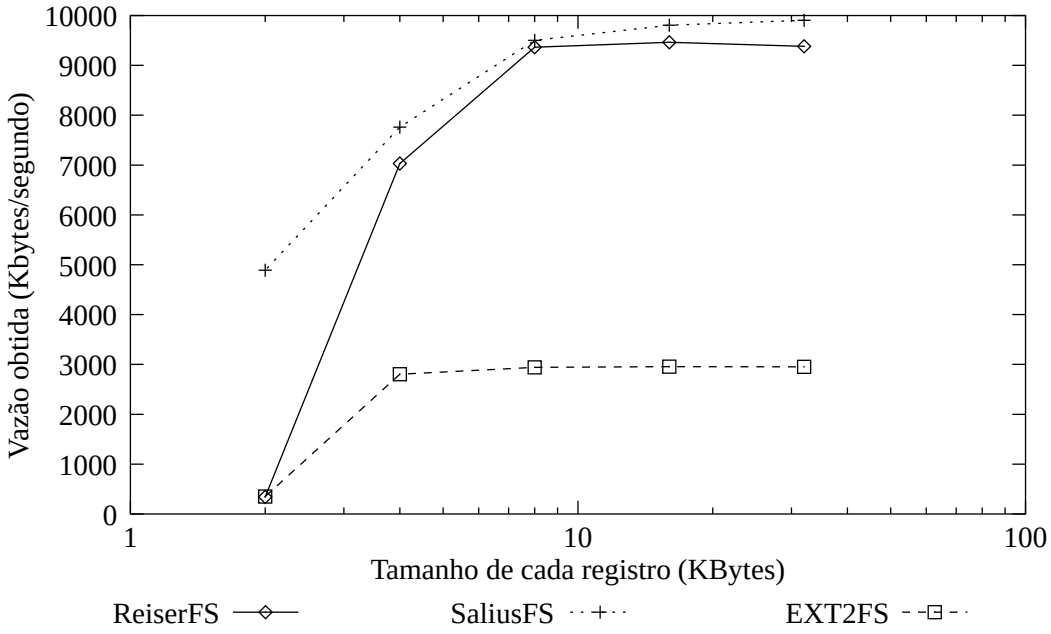

Figura 5.19: Desempenho comparativo para regravações síncronas de um arquivo de 32000Kbytes sem concorrência

vação de arquivos grandes quando não existia concorrência, figura 5.19. O sistema Ext2 apresentou um desempenho bem inferior aos demais nesses experimentos de regravação. A

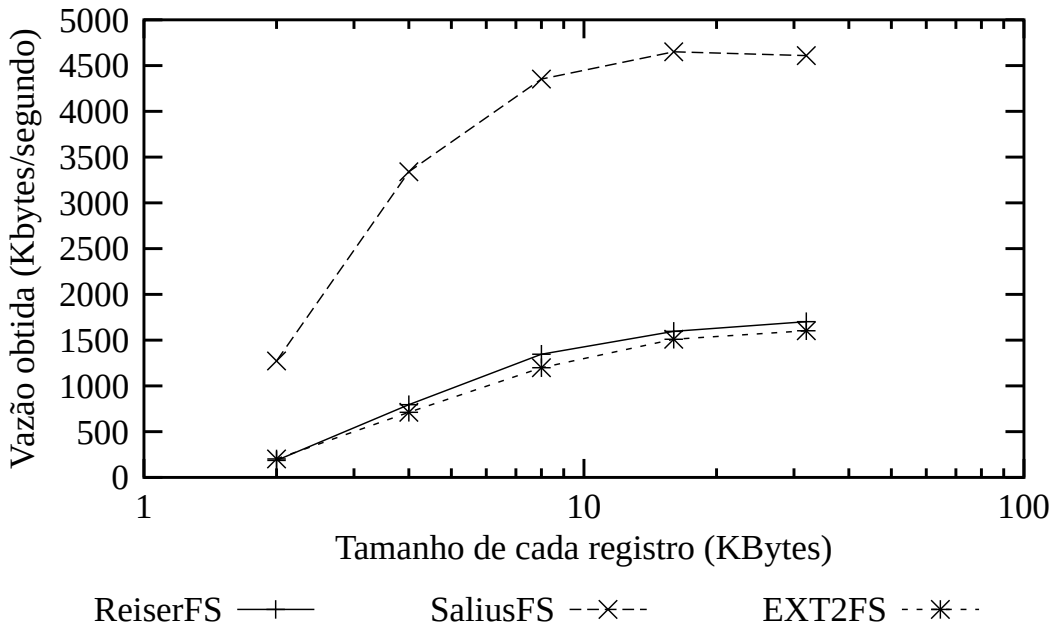

Figura 5.20: Desempenho comparativo para regravações síncronas de um arquivo de 32000Kbytes com concorrência

performance do sistema Salius foi cerca de 3 vezes superior aos demais sistemas quando havia concorrência pelo uso do disco (figuras 5.18 e 5.20).

Nesta primeira parte desses resultados podemos ver que o tamanho do arquivo teve pouca influência no desempenho do sistema de arquivo Salius tanto para gravações quanto para regravações em arquivos com blocos pré-alocados. Esse fato já havia sido observado em um trabalho anterior [BCPS02]. O desempenho na gravação e regravação no sistema de arquivos Reiser também não é muito afetado pelo tamanho do arquivo utilizado para teste com uma exceção. Em alguns testes realizados utilizando o menor tamanho de arquivo (32Kbytes) esse sistema apresentou um desempenho excepcionalmente alto, inclusive ultrapassando a vazão máxima do disco em utilização. Isso aconteceu porque esse sistema faz um uso eficiente da memória *cache* existente no disco (512Kbytes), sendo esta suficiente para agrupar as operações sobre dados e meta-dados em uma pequena gravação de um arquivo. O resultado do teste foi obtido através da média da vazão obtida em 10 instâncias do experimento, entretanto, cada instância era realizada após uma recriação do sistema de arquivos, através do

comando *mkreiserfs*, o que explica o uso eficiente da *cache* do disco. Ao se repetir os experimentos, aumentando o número de instâncias para 50, e deixando-se de recriar o sistema de arquivos antes de cada uma, os resultados atingiram os valores esperados, que foram usados nos gráficos.

O tamanho do registro utilizado em cada operação de gravação mostrou ter bem mais influência que o tamanho do arquivo nos resultados dos experimentos. Podem-se fazer duas observações importantes através desses resultados: i) A existência de concorrência pelo uso do disco deteriora bastante o desempenho dos sistemas de arquivo que fazem uso de gravação síncrona para estabilizar os dados (compare a figura 5.15 com a figura 5.16), e ii) O sistema de arquivos Ext2 apresenta um desempenho bem inferior aos demais sistemas nas operações de regravação de um arquivo com blocos pré-alocados (figuras 5.17 a 5.20).

A influência da concorrência pelo uso do disco foi um pouco maior que a esperada, enquanto a, relativamente pequena, influência do tráfego gerado na rede sobre o sistema Salius já era esperada pelos resultados obtidos em outro trabalho de comparação [BCPS02]. A perda de desempenho do sistema Salius foi maior que a desse trabalho citado, provavelmente porque nestes novos experimentos a carga gerada pelo compartilhamento de dados através de NFS é bem maior que a carga simulada utilizada nos experimentos prévios, onde utilizamos uma ferramenta sintética para saturar a rede. O resultado ruim na regravação de dados obtido pelo sistema de arquivos Ext2 em relação ao sistema ReiserFS, veja figura 5.19, é explicado pelo fato desse último fazer uso de *extents* [5], aproximando-se dos resultados obtidos pelo sistema de arquivos Salius.

Segue um sumário das observações feitas através da análise dos resultados dos experimentos sintéticos:

- A concorrência pelo uso do disco pode vir a prejudicar as operações estáveis nos sistemas baseados em gravação síncrona ao ponto de inviabilizar a sua utilização;
- A existência de tráfego em uma rede ethernet prejudicou o desempenho do sistema Salius, entretanto esse prejuízo foi menor quando comparado àquele sofrido pelos outros sistemas com a concorrência pelo uso do disco;

[5] Grandes blocos não formatados que, se previamente alocados, permitem uma maior vazão por serem contíguos.

5.3.2 Resultados dos experimentos com um servidor SMTP

A análise dos dados obtidos através do monitoramento do servidor de correio eletrônico *anjinho.dsc.ufpb.br* no período citado anteriormente nos mostrou os seguintes resultados:

- 2.6Gbytes é o total de dados que foram recebidos pelo servidor;
- 14Kbytes é o tamanho médio das mensagens;
- 9.8Mbytes é o tamanho da maior mensagem;
- 202bytes é o tamanho da menor mensagem;
- Houve pouca variância nesse tamanho [6].

Baseando-se nesses dados, escolhemos um dia que apresentasse um perfil de carga semelhante a esse. O dia escolhido para ter sua carga submetida ao nosso servidor de testes foi 5 de março de 2002 que apresentou o seguinte perfil de mensagens:

- 83.3Mbytes é o total de dados que foram recebidos pelo servidor;
- 12.4Kbytes é o tamanho médio das mensagens;
- 2.6Mbytes é o tamanho da maior mensagem;
- 227bytes é o tamanho da menor mensagem;
- Houve pouca variância nesse tamanho [7].

O teste consistiu em se verificar o tempo que o servidor de correio eletrônico demorava para processar o recebimento [8] do total de mensagens relativas a esse dia. Para tanto, para cada mensagem obtida do *log* do dia 5 de março era enviada uma, de igual tamanho ao servidor de teste. O resultado consiste do tempo gasto pelo servidor para receber todas as mensagens.

[6] A grande maioria das mensagens ($\tilde{9}0\%$) era de tamanho entre 10 e 16 Kbytes

[7] A grande maioria das mensagens ($\tilde{9}2\%$) era de tamanho entre 10 e 16 Kbytes

[8] Ao receber uma mensagem, o servidor a armazena de forma estável na fila para ser processada posteriormente.

Para se testar um novo sistema de arquivos, a partição do disco rígido era recriada e o servidor de correio eletrônico era reimplantado sobre essa partição. Os resultados obtidos através da média de três instâncias de teste para cada sistema de arquivos foi o seguinte:

Sistema de arquivos	Salius	Ext2	ReiserFS
Tempo gasto(segundos)	302.3	374.6	646.8

O resultado desse teste nos trouxe algumas surpresas e por esse motivo resolvemos estender os testes sobre o servidor de correio eletrônico para verificar o comportamento do mesmo sob cargas diversificadas.

Para os testes simulados resolvemos testar diversos tamanhos de mensagens e verificar a influência do número de seções simultâneas [9] sobre o desempenho, imaginando que os sistemas de arquivo tradicionais apresentem uma melhora no seu desempenho com o aumento do número de seções paralelas, principalmente com mensagens pequenas. Cada instância do teste consistia-se do envio de 500 mensagens de tamanho fixo para o servidor variando-se o número de seções simultâneas entre 5, 10 e 20 e medindo-se o tempo gasto para completar o envio dessas 500 mensagens. Foram testadas mensagens de tamanhos 1, 10, 100 e 1000Kbytes. Os resultados obtidos , separando-os por sistema de arquivo testado, foram os seguintes (resultados em segundos por grupo de 500 mensagens):

Sistema de arquivos Salius:

Número de seções	1Kbyte	10Kbytes	100Kbytes	1000Kbytes
5	12.19	13.51	35.61	152.81
10	8.78	9.29	30.88	145.74
20	6.54	7.09	28.29	142.27

Sistema de arquivos Ext2:

[9] Cada seção simultânea é equivalente a um programa cliente de usuário conectado ao servidor de correio eletrônico

Número de seções	1Kbyte	10Kbytes	100Kbytes	1000Kbytes
5	17.97	18.73	42.15	168.67
10	11.47	12.43	35.36	150.33
20	8.84	9.21	33.92	146.20

Sistema de arquivos Reiser:

Número de seções	1Kbyte	10Kbytes	100Kbytes	1000Kbytes
5	29.57	33.82	53.62	203.85
10	27.00	31.43	49.99	197.89
20	24.39	28.82	46.85	193.78

Para simplificar o entendimento, criamos gráficos baseados nesses resultados mostrando a vazão obtida pelo servidor sobre cada um dos sistemas de arquivo. Os gráficos estão separados pelo número de seções e mostram a vazão atingida pelos sistemas variando-se o tamanho de cada mensagem.

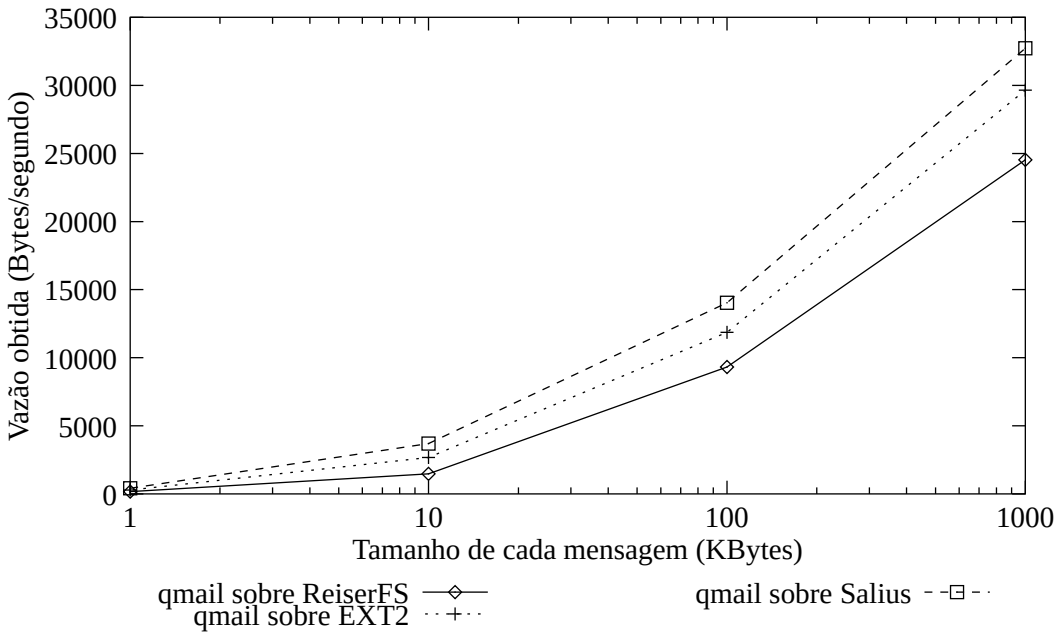

Figura 5.21: Desempenho comparativo do servidor de correio eletrônico com 5 seções abertas simultaneamente

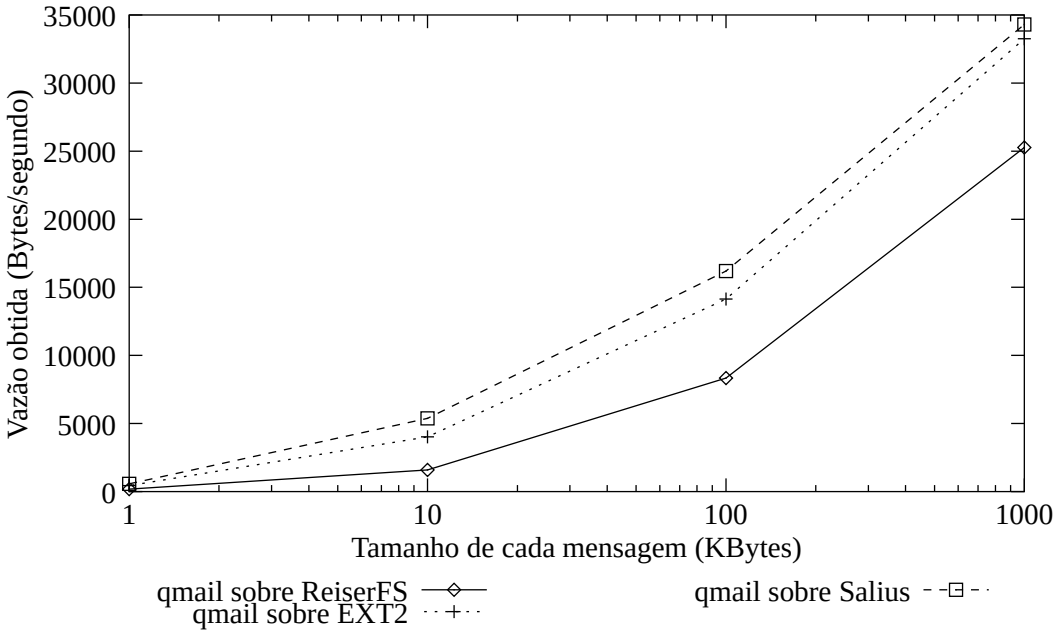

Figura 5.22: Desempenho comparativo do servidor de correio eletrônico com 10 seções abertas simultaneamente

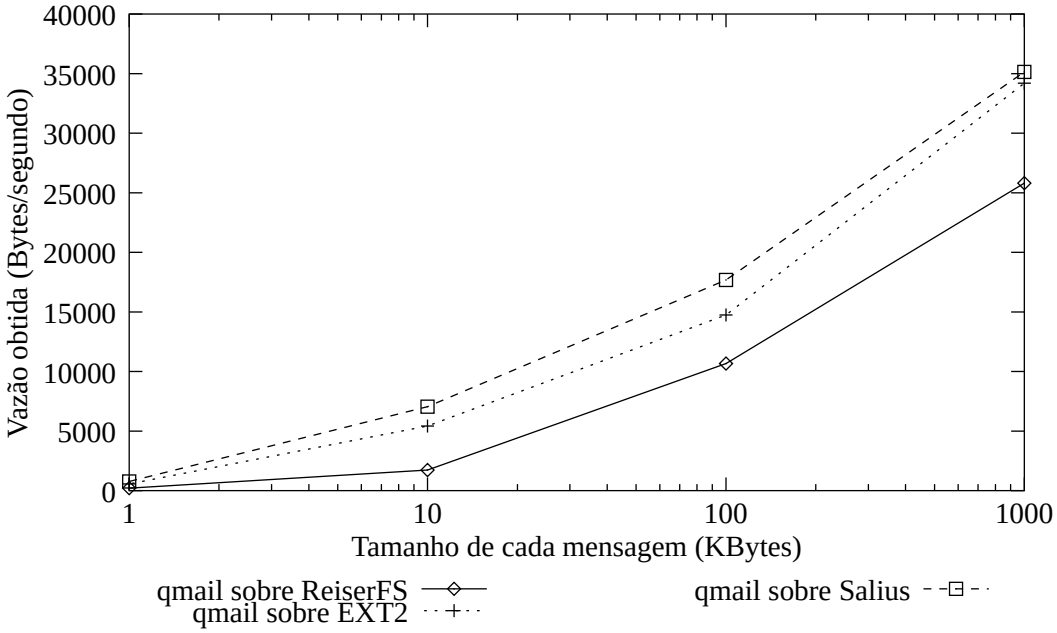

Figura 5.23: Desempenho comparativo do servidor de correio eletrônico com 20 seções abertas simultaneamente

Como pode-se observar, o sistema de arquivos Salius leva uma ligeira vantagem em todos os testes em relação ao sistema Ext2, enquanto o sistema ReiserFS apresentou um desempenho bastante inferior aos outros dois, o que foi uma surpresa, uma vez que esse sistema foi projetado para apresentar vantagens em operações sobre arquivos pequenos, o que é o caso da maioria das instâncias dos experimentos com o servidor de correio eletrônico. A explicação para esse fraco desempenho do sistema de arquivos ReiserFS pode estar justamente na sua principal característica, o uso de árvores balanceadas para estruturar os meta-dados. Isso nos leva a uma forma de localização de arquivos com desempenho constante, sem pior ou melhor caso, além disso, no caso de sistemas de arquivo ainda não muito utilizados, com poucos arquivos criados, a árvore precisa ser constantemente reestruturada, adicionando-se novos braços e um novo topo, levando à criação de meta-dados extras nos nós internos de pesquisa, fato que não se torna necessário nos outros dois sistemas.

É importante salientar um detalhe, entretanto. Ao se realizar os experimentos com mensagens de 1Mbyte o sistema Salius mostrou uma limitação importante. Como as rajadas de mensagens eram muito rápidas, os servidores de replicação não tinham tempo suficiente para liberar a memória alocada para as primeiras cópias recebidas, tendo sua capacidade esgotada, degradando bastante o desempenho do sistema. Isso ocorre porque os servidores de replicação esperam 60 segundos antes de remover qualquer réplica da memória volátil, aguardando que estas tenham sido estabilizadas no cliente que as solicitou. A solução foi aumentar, momentaneamente, a capacidade de memória das máquinas onde os servidores de replicação executavam para 320Mbytes. Isso solucionou esse problema para essa vazão (500Mbytes em pouco mais de 2 minutos) mas deixou evidente que a vazão, a longo prazo, do sistema Salius está limitada pela quantidade de memória volátil existente nas máquinas remotas.

Após essa adaptação era esperado um melhor desempenho do sistema Salius baseando-se nos resultados dos experimentos sintéticos. Como o sistema ReiserFS mostrou-se bastante inferior aos demais nos testes anteriores quando se tratava de gravação de novos dados, o seu fraco desempenho já era esperado. A pouca diferença entre o sistema Ext2 e o sistema Salius foi uma surpresa mas explica-se pelo fato de que o processo de receber uma mensagem de correio eletrônico envolve muito mais operações que gravar dados em arquivos através da chamada de sistema *write*. Operações exclusivamente sobre os meta-dados de um arquivo

como mover, adicionar entrada de diretório, entre outras ainda são realizadas de forma síncrona no sistema de arquivos Salius. Aparentemente, o servidor de correio eletrônico *qmail* é limitado mais por essas operações que por aquelas aceleradas no sistema Salius, como *write* e *fsync*.

5.4 Conclusão

Neste capítulo descrevemos uma série de experimentos realizados com as três implementações de sistemas de arquivo com suporte à gravação estável de dados escolhidos. Os resultados destes experimentos foram apresentados e analisados com o objetivo de se chegar a algumas conclusões sobre as vantagens e desvantagens dessas três abordagens utilizadas para se estabilizar dados. Uma parte dos resultados correspondeu às espectativas baseadas no projeto desses sistemas de arquivo. Alguns resultados, entretanto, trouxeram algumas surpresas como a pouca vantagem do sistema Salius sobre o Ext2 nos testes com o servidor de correio eletrônico.

A partir dos resultados obtidos desses experimentos, chegamos às seguintes conclusões práticas sobre gravação estável de dados em um contexto geral:

- A utilização de memória remota como alternativa para a gravação síncrona de dados é viável em praticamente todas as situações;
- Extensões [10] são uma boa alternativa para aplicações que trabalhem com gravação de grandes arquivos.

As seguintes conclusões foram feitas no contexto de servidores de correio eletrônico:

- Processar mensagens de correio eletrônico é uma tarefa limitada por operações sobre meta-dados mais que por operações de gravação de dados, portanto obtém-se pouca vantagem ao se utilizar um sistema que usa memória remota para estabilizar dados;
- A implementação de árvores B utilizada, ReiserFS versão 3, não se mostrou uma alternativa viável em relação aos nós-i nesse contexto.

[10] Grandes blocos contíguos não formatados, não obrigatoriamente em conjunto com árvores B.

Capítulo 6

Conclusão

Neste capítulo final da dissertação, iremos fazer um sumário dos objetivos que foram traçados ao longo do seu desenvolvimento e uma análise comparativa entre estes objetivos e os resultados apresentados. Além disso, iremos apresentar uma cronologia dos trabalhos e projetos que serviram de base para esta dissertação. Por último, faremos algumas considerações finais sobre o programa de mestrado.

O objetivo geral deste trabalho foi elucidar as principais questões que surgem na mente de administradores de sistemas ao se deparar com uma aplicação que faz uso de armazenamento estável de dados. Para tanto o trabalho envolveu duas linhas de pesquisa: i) tolerância a falhas e ii) análise de desempenho. Fizemos uma abordagem ligada à área de tolerância a falhas no Capítulos 2, em que foram mostrados os principais problemas existentes na criação de serviços de armazenamento estável de dados e algumas soluções, e no Capítulo 4, em que foram detalhadas as implementações de três sistemas de arquivo enfocando a semântica de gravação estável de dados.

Mais especificamente, procuramos entender o problema do armazenamento estável, analisar algumas implementações utilizadas e criar uma implementação alternativa às existentes. Ao estudar avaliação de desempenho de sistemas de armazenamento, nossos objetivos eram entender as variáveis existentes e sua influência nesse âmbito.

O ponto de partida para o desenvolvimento deste trabalho foi o projeto de implementação de um sistema de arquivos baseado na replicação remota de buffers [Sta; BCPS02]. Esse sistema de arquivos havia sido idealizado em um outro trabalho de dissertação mas sua implementação não pode ser feita naquele momento. O desenvolvimento

desse sistema de arquivos durou cerca de seis meses, tendo sido iniciado no começo de 2001. Ao longo desse processo, o projeto foi alterado para melhor se adequar às tecnologias usadas no seu desenvolvimento. Nesse momento, ao se fazerem avaliações preliminares de desempenho para a publicação de artigos, surgiu a ideia de desenvolver um estudo de avaliação de desempenho de serviços de armazenamento estável de dados, incluindo o nosso sistema de arquivos.

A partir do segundo semestre de 2001 o sistema de arquivos Salius [BCPS02] estava sendo comparado com o sistema que foi a base de seu desenvolvimento, o Ext2, e decidimos que esse seria o nosso objeto de estudo: Avaliar sistemas de arquivo no âmbito de armazenamento estável de dados. Passamos então a recolher bibliografia para estudo dessa área já pensando no desenvolvimento da dissertação.

Nesse período foi verificado que avaliação de desempenho desse tipo de sistema é uma tarefa complexa e que está fortemente acoplada ao tipo de aplicação que vai utilizá-lo. Outra observação que pôde ser feita foi que, apesar de existirem muitos trabalhos importantes sobre metodologias de avaliação, existia uma carência de publicações científicas trazendo comparações feitas entre sistemas de arquivo usando tais metodologias. As publicações existentes sempre procuravam mostrar as vantagens de algum projeto desenvolvido pelo autor e limitando-se a um pequeno número de experimentos.

Nesse momento tínhamos a ambição de criar uma espécie de tratado final sobre sistemas de armazenamento estável de dados, o que era claramente impossível de ser alcançado no âmbito do programa de mestrado, visto que teria que envolver muitos outros sistemas de arquivo, muito mais estudo sobre outras alternativas de gravação, sem falar na evolução constante que essas tecnologias sofrem com o passar dos anos. Por fim, resolvemos limitar o nosso estudo a técnicas de criação de serviços de armazenamento estável, algumas implementações importantes de sistemas de arquivo que pudessem ser avaliadas no nosso ambiente e restringir os experimentos a dois contextos: i) algumas avaliações sintéticas para trazer uma idéia sobre o comportamento desses sistemas de arquivo, e ii) uma avaliação de desempenho de uma aplicação de produção que utilize armazenamento estável.

A escolha de um servidor de correio eletrônico como a aplicação a ser usada como ferramenta de avaliação mostrou-se uma decisão acertada porque, longe de mostrar resultados extremamente favoráveis a nossa implementação de um sistema de arquivos, o que foi mos-

trado nos experimentos sintéticos, o que se viu foi que a limitação de performance provocada pelos discos nesse tipo de aplicação não está ligada as operações de gravação de dados em si, e sim às operações exclusivamente sobre meta-dados. Para nós isso foi uma surpresa inesperada e trouxe mais à evidência trabalhos que já exploram essa lacuna [GP94].

Quanto aos objetivos propostos na introdução do trabalho acreditamos que os resultados obtidos e as conclusões extraídas dos mesmos trazem uma boa quantidade de informação para aqueles interessados em entender um pouco mais sobre sistemas de armazenamento estável de dados e procuram uma fonte inicial de informação sobre a avaliação de desempenho desse tipo de sistema. Dessa forma, consideramos que este trabalho é bastante útil e conclui um processo que foi iniciado a alguns anos com a criação do projeto Salius [Sta].

Fazendo um resumo abstrato das conclusões tiradas deste trabalho, podemos dizer que, a forma de implementação utilizada no sistema de arquivo tem uma grande importância sobre a performance do mesmo em armazenamento estável, basta, para tanto, observar a grande diferença de performance em regravações observada nos três sistemas dependendo do contexto. Duas idéias já consagradas confirmaram sua força, replicação remota de dados e uso de grandes blocos (*extents*) para gravação do conteúdo de arquivos. Pelo menos no contexto de nossos experimentos com servidores de correio eletrônico, o uso de árvores B não demonstrou ser uma boa alternativa para gravação estável de dados.

Para finalizar, gostaríamos de acrescentar que este trabalho de dissertação trouxe-nos motivação especial por dois motivos: i) sempre foi nosso interesse o desenvolvimento de *software* básico e esse projeto foi a primeira oportunidade de trabalhar com o código fonte de um sistema operacional, e ii) foi um trabalho que envolveu duas áreas distintas de pesquisa, tolerância a falhas e avaliação de desempenho. A primeira motivação foi plenamente satisfeita pois tivemos que estudar a estrutura interna do sistema operacional na prática. Isso se mostrou útil nos momentos, na implementação, em que alguma coisa não funcionava e precisávamos depurar os erros. Como se faz para depurar o núcleo de um sistema operacional? Conhecimento sobre a estrutura interna do código ajudou bastante nesse ponto. Sobre a segunda motivação podemos dizer que trabalhos multidiciplinares (não é exatamente este o caso mas são áreas de pesquisa distintas) tendem a trazer resultados mais práticos e esse era um dos nossos objetivos.

Consideramos que o trabalho desenvolvido contribuiu da forma esperada para aqueles in-

teressados no estudo desta área e também que foi um desafio necessário dentro do programa de mestrado.

Bibliografia

[BCPS02] Francisco V. Brasileiro, Walfredo Cirne, Erick Passos, and Tatiana S. Stanchi. Using remote memory to stabilize data efficiently on an ext2 linux file system. In *XX Simpósio Brasileiro de Redes de Computadores*, 2002.

[Bra] T. Bray. Bonnie benchmark source code, 1990. ftp://www.cs.umbc.edu/pub/elm/iobenchmarks/bonnie.sh.

[CLG+94] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[CNC+96] Peter M. Chen, Wee Teck Ng, Subhachandra Chandra, Christopher Aycock, Gurushankar Rajamani, and David Lowell. The rio file cache: Surviving operating system crashes. In *Architectural Support for Programming Languages and Operating Systems*, pages 74–83, 1996.

[CP93a] P. M. Chen and D. A. Patterson. A new approach to I/O performance evaluation—self-scaling I/O benchmarks, predicted I/O performence. In *Proceedings of the 1993 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 1–12, Santa Clara, CA, USA, 10–14 1993.

[CP93b] Peter M. Chen and David A.. Patterson. Storage performance—metrics and benchmarks. In *Proceedings of the IEEE, 81(8):1151–1165, August 1993.*, 1993.

[CTT94] R. Card, T. Ts'o, and S. Tweedie. Design and implementation of the second extended filesystem. In *Proceedings of the First Dutch International Symposium on Linux*, december 1994.

[ext] Linux ext3 file system development homepage. http://beta.redhat.com/index.cgi?action=ext3.

[GP94] G. R. Ganger and Y. N. Patt. Metadata update performance in file systems. In *Proceedings of the USENIX 1994 Symposium on Operating Systems Design and Implementation*, pages 49–60, Monterey, CA, USA, 14–17 1994.

[jfs] Ibm jfs journaled file system homepage. http://www-124.ibm.com/developerworks/oss/jfs/.

[LGG+91] Barbara Liskov, Sanjay Ghemawat, Robert Gruber, Paul Johnson, Liuba Shrira, and Michael Williams. Replication in the Harp file system. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, pages 226–38. Association for Computing Machinery SIGOPS, 1991.

[MG99] Marshall Kirk McKusick and Gregory R. Ganger. Soft updates: A technique for eliminating most synchronous writes in the fast filesystem. pages 1–17, 1999.

[Mil96] Ethan L. Miller. Towards scalable benchmarks for mass storage systems. In *Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies*, pages 515–528, sep 1996.

[MJLF84] M. McKusick, W. Joy, S. Leffler, and R. Fabry. A fast file system for unix. In *Proceedings of the ACM Transactions on Computer Systems*, august 1984.

[MK] L. W. McVoy and S. R. Kleiman. Extent-like performance from a unix file system.

[MS96] Larry W. McVoy and Carl Staelin. lmbench: Portable tools for performance analysis. In *USENIX Annual Technical Conference*, pages 279–294, 1996.

[Nor] W. Norcott. Iozone benchmark source code, version 2.01. ftp://www.cs.umbc.edu/pub/elm/iobenchmarks/iozone01.

[Org] IEEE Standards Organization. Ieee posix certifcation authority. http://standards.ieee.org/regauth/posix/.

[Pan94] Pankaj Jalote. *Fault Tolerance in Distributed Systems*. Prentice Hall ,Inc, New York, NY, 1994.

[PB90] A. PArk and J. C. Becker. Iostone: A synthetic file system benchmark. In *Computer Architecture News 18(2)*, pages 45–52, 1990.

[Rei] Hans Reiser. Reiserfs v.3 whitepaper. http://www.namesys.com/content_table.html.

[RO92] Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems*, 10(1):26–52, 1992.

[RT74] Dennis W. Ritchie and Ken Thompson. The UNIX time-sharing system. *Communications of the ACM*, 17(7):365–375, July 1974.

[SFGM93] Michael Stonebraker, Jim Frew, Kenn Gardels, and Jeff Meredith. The SEQUOIA 2000 storage benchmark. pages 2–11, 1993.

[Sta] Tatiana Simas Stanchi. Serviço de armazenamento estável com recuperação pra frente baseada na replicação remota de buffers. Dissertação COPIN, Campina Grande, fevereiro de 2000.

[Tan87] Andrew S. Tanenbaum. MINIX: A UNIX clone with source code for the IBM PC. *the USENIX Association newsletter*, 12(2):3–9, March 1987.

[xfs] Sgi's xfs homepage. http://www.sgi.com/software/xfs/.

Apêndice A

Análise Estatística dos Resultados

Sempre que se faz experimentos envolvendo muitas variáveis, é importante isolá-los da interferência de variáveis externas, garantindo assim, a representatividade dos resultados. A estatística é uma ferramenta útil para indicar que esse objetivo foi atingido. Nos nossos experimentos, isolamos, o melhor possível, o ambiente de execução dos testes para evitar essas interferências. Nos experimentos que envolviam concorrência, entretanto, interferência era uma das variáveis sendo testadas. Nesse grupo de experimentos, foi mais difícil se chegar a um valor representativo da performance dos sistemas de arquivo. Essa pequena análise sobre os resultados tem o objetivo de validar essa representatividade.

A.1 Objetivos

Vamos chamar cada ponto em um gráfico (ponto onde as variáveis sendo testadas eram fixas) de *experimento*. Para se chegar a um valor representativo, livre de interferências, eram executadas *n* instâncias de cada experimento. Nosso objetivo foi chegar a um valor, para *n*, suficientemente grande para que a média encontrada na amostra fosse representativa em relação à média da população de todas as possíveis instâncias desse experimento.

Para verificar se o tamanho de nossa amostra (n) era suficientemente grande, tomamos o seu desvio padrão, σ, e dividimos por $\sqrt{n}$, chamando esse valor encontrado de coeficiente de variação, q. Quanto menor for o valor de q, em relação à média da população (m), mais próximos estamos de um valor representativo para a mesma, através de nossa amostra de média M. Como não sabemos o valor de m, fizemos uma aproximação usando o valor da

média da amostra, *M*. Consideramos suficiente para nossos objetivos que, para cada amostra de um experimento, o valor encontrado para *q* fosse no máximo 10% do valor da média da amostra, *M*.

A.2 Análise

Aqui podem ser observados os valores, para todos os experimentos, das médias e do coeficiente de variação, *q*, além da relação percentual entre esses dois valores.

Para cada sistema de arquivos foram feitas duas tabelas, uma agrupando os experimentos livres de concorrência, e outra trazendo os resultados dos experimentos com concorrência. Pode ser observado que, o coeficiente de variação é mais baixo nos experimentos livres de concorrência e em todos os experimentos com o sistema de arquivos Salius. Apenas nos experimentos envolvendo concorrência, para os sistemas de arquivos Reiser e EXT2, encontramos amostras onde o coeficiente de variação, *q*, fica próximo do nosso limite estabelecido de 10%.

Em seguida, mostraremos algumas figuras apresentando os gráficos comparativos da evolução dos resultados dos três sistemas de arquivo para dois experimentos: um livre de concorrência e um envolvendo concorrência.

A.2.1 Sistema de arquivos EXT2

Acima os resultados para os experimentos livres de concorrência.

Arquivo (KB)	Registro (KB)	Write (KB/s)	*q* (%)	Rewrite (KB/s)	*q* (%)
32	2	980	20.38 (2.08%)	863	27.27 (3.16%)
32	4	1327	38.35 (2.89%)	2805	84.71 (3.02%)
32	8	1478	26.01 (1.76%)	2980	72.71 (2.44%)
32	16	1482	48.01 (3.24%)	3008	56.24 (1.87%)
32	32	1496	21.54 (1.44%)	3015	39.19 (1.30%)
320	2	373	10.89 (2.92%)	401	17.28 (4.31%)
320	4	383	4.28 (1.12%)	2810	44.67 (1.59%)
320	8	781	20.22 (2.59%)	2840	81.22 (2.86%)
320	16	1347	67.21 (4.99%)	2975	68.12 (2.29%)
320	32	1471	33.39 (2.27%)	2976	50.88 (1.71%)
3200	2	347	16.51 (4.76%)	349	3.97 (1.14%)
3200	4	349	6.80 (1.95%)	2807	160.84 (5.73%)
3200	8	678	10.44 (1.54%)	2914	40.79 (1.40%)
3200	16	1275	20.78 (1.63%)	2958	79.86 (2.70%)
3200	32	1475	26.99 (1.83%)	2955	82.74 (2.80%)
32000	2	346	13.90 (4.02%)	348	7.76 (2.23%)
32000	4	346	5.22 (1.51%)	2805	46.56 (1.66%)
32000	8	669	20.00 (2.99%)	2941	32.05 (1.09%)
32000	16	1253	14.91 (1.19%)	2954	153.31 (5.19%)
32000	32	1468	39.19 (2.67%)	2953	95.08 (3.22%)

Resultados para os experimentos com concorrência:

Arquivo (KB)	Registro (KB)	Write (KB/s)	*q* (%)	Rewrite (KB/s)	*q* (%)
32	2	170	6.76 (3.98%)	201	8.68 (4.32%)
32	4	260	10.84 (4.17%)	712	32.68 (4.59%)
32	8	341	14.83 (4.35%)	1232	59.87 (4.86%)
32	16	443	20.06 (4.53%)	1507	47.01 (3.12%)
32	32	511	24.06 (4.71%)	1604	51.80 (3.23%)
320	2	173	8.45 (4.89%)	204	7.99 (3.92%)
320	4	265	8.13 (3.07%)	710	47.00 (6.62%)
320	8	340	11.08 (3.26%)	1212	112.95 (9.32%)
320	16	449	16.43 (3.66%)	1496	47.87 (3.20%)
320	32	510	17.74 (3.48%)	1605	55.69 (3.47%)
3200	2	172	9.09 (5.29%)	205	7.66 (3.74%)
3200	4	266	18.91 (7.11%)	708	28.39 (4.01%)
3200	8	343	30.62 (8.93%)	1237	52.94 (4.28%)
3200	16	446	13.69 (3.07%)	1501	68.29 (4.55%)
3200	32	516	16.77 (3.25%)	1620	78.08 (4.82%)
32000	2	174	5.96 (3.43%)	202	6.24 (3.09%)
32000	4	261	9.44 (3.62%)	711	59.93 (8.43%)
32000	8	346	13.14 (3.80%)	1198	42.40 (3.54%)
32000	16	448	17.83 (3.98%)	1509	94.01 (6.23%)
32000	32	516	21.46 (4.16%)	1603	143.14 (8.93%)

A.2.2 Sistema de arquivos Reiser

Resultados para os experimentos livres de concorrência:

Arquivo (KB)	Registro (KB)	Write (KB/s)	*q* (%)	Rewrite (KB/s)	*q* (%)
32	2	200	7.18 (3.59%)	358	24.95 (6.97%)
32	4	224	12.70 (5.67%)	3776	41.53 (1.10%)
32	8	274	6.41 (2.34%)	3845	203.40 (5.29%)
32	16	430	23.39 (5.44%)	3873	125.09 (3.23%)
32	32	596	12.03 (2.02%)	3872	102.99 (2.66%)
320	2	101	2.24 (2.22%)	370	7.69 (2.08%)
320	4	147	2.48 (1.69%)	3959	59.78 (1.51%)
320	8	267	8.46 (3.17%)	3959	254.56 (6.43%)
320	16	408	5.58 (1.37%)	3881	143.98 (3.71%)
320	32	625	17.81 (2.85%)	4894	150.24 (3.07%)
3200	2	97	1.01 (1.05%)	345	8.62 (2.50%)
3200	4	137	3.46 (2.53%)	5617	108.40 (1.93%)
3200	8	190	8.20 (4.32%)	7134	96.30 (1.35%)
3200	16	234	5.14 (2.20%)	7967	387.19 (4.86%)
3200	32	270	11.07 (4.10%)	7731	164.67 (2.13%)
32000	2	94	1.76 (1.88%)	343	9.98 (2.91%)
32000	4	132	7.61 (5.77%)	7032	164.54 (2.34%)
32000	8	177	2.76 (1.56%)	9364	165.74 (1.77%)
32000	16	217	6.59 (3.04%)	9463	113.55 (1.20%)
32000	32	246	3.05 (1.24%)	9383	307.76 (3.28%)

Resultados para os experimentos com concorrência:

Arquivo (KB)	Registro (KB)	Write (KB/s)	*q* (%)	Rewrite (KB/s)	*q* (%)
32	2	43	1.33 (3.11%)	162	6.01 (3.71%)
32	4	80	2.63 (3.29%)	801	31.87 (3.98%)
32	8	97	3.36 (3.47%)	1355	57.58 (4.25%)
32	16	115	4.19 (3.65%)	1608	72.68 (4.52%)
32	32	120	4.59 (3.83%)	1712	82.00 (4.79%)
320	2	46	1.84 (4.02%)	206	6.30 (3.06%)
320	4	82	3.44 (4.20%)	800	44.00 (5.50%)
320	8	98	4.29 (4.38%)	1351	43.77 (3.24%)
320	16	118	5.38 (4.56%)	1601	95.09 (5.94%)
320	32	122	5.78 (4.74%)	1708	147.57 (8.64%)
3200	2	43	2.11 (4.92%)	189	5.91 (3.13%)
3200	4	79	2.45 (3.11%)	798	27.13 (3.40%)
3200	8	101	3.59 (3.56%)	1350	49.54 (3.67%)
3200	16	120	4.76 (3.97%)	1599	63.00 (3.94%)
3200	32	126	4.77 (3.79%)	1704	71.73 (4.21%)
32000	2	43	2.40 (5.60%)	187	8.37 (4.48%)
32000	4	78	5.78 (7.42%)	796	37.81 (4.75%)
32000	8	98	9.05 (9.24%)	1345	40.61 (3.02%)
32000	16	111	3.44 (3.10%)	1597	57.97 (3.63%)
32000	32	120	3.93 (3.28%)	1701	82.66 (4.86%)

A.2.3 Sistema de arquivos Salius

Resultados para os experimentos livres de concorrência:

Arquivo (KB)	Registro (KB)	Write (KB/s)	*q* (%)	Rewrite (KB/s)	*q* (%)
32	2	5061	153.34 (3.03%)	5122	89.63 (1.75%)
32	4	7843	96.46 (1.23%)	7842	91.75 (1.17%)
32	8	9584	259.72 (2.71%)	9592	294.47 (3.07%)
32	16	9855	607.06 (6.16%)	9862	343.19 (3.48%)
32	32	9981	238.54 (2.39%)	9970	272.18 (2.73%)
320	2	5014	297.83 (5.94%)	5038	108.82 (2.16%)
320	4	7789	161.23 (2.07%)	7801	124.03 (1.59%)
320	8	9550	258.80 (2.71%)	9561	97.52 (1.02%)
320	16	9821	170.88 (1.74%)	9802	440.10 (4.49%)
320	32	9923	319.52 (3.22%)	9892	311.59 (3.15%)
3200	2	4979	70.70 (1.42%)	5023	129.59 (2.58%)
3200	4	7834	227.18 (2.90%)	7845	156.90 (2.00%)
3200	8	9491	104.40 (1.10%)	9503	135.89 (1.43%)
3200	16	9843	253.94 (2.58%)	9831	554.46 (5.64%)
3200	32	9903	476.33 (4.81%)	9895	288.93 (2.92%)
32000	2	4933	110.99 (2.25%)	4892	146.27 (2.99%)
32000	4	7754	355.90 (4.59%)	7763	187.86 (2.42%)
32000	8	9512	183.58 (1.93%)	9501	175.76 (1.85%)
32000	16	9781	133.99 (1.37%)	9807	124.54 (1.27%)
32000	32	9861	158.76 (1.61%)	9903	402.06 (4.06%)

Resultados para os experimentos com concorrência:

Arquivo (KB)	Registro (KB)	Write (KB/s)	*q* (%)	Rewrite (KB/s)	*q* (%)
32	2	1299	40.13 (3.09%)	1358	18.19 (1.34%)
32	4	3412	44.01 (1.29%)	3420	96.78 (2.83%)
32	8	4562	125.91 (2.76%)	4381	99.01 (2.26%)
32	16	4762	318.57 (6.69%)	4821	81.47 (1.69%)
32	32	4825	117.73 (2.44%)	5024	56.26 (1.12%)
320	2	1402	48.64 (3.47%)	1387	76.14 (5.49%)
320	4	3541	75.06 (2.12%)	3462	112.51 (3.25%)
320	8	4361	141.29 (3.24%)	4437	118.91 (2.68%)
320	16	4871	87.67 (1.80%)	4725	99.22 (2.10%)
320	32	5091	117.09 (2.30%)	4814	73.65 (1.53%)
3200	2	1396	20.52 (1.47%)	1407	93.42 (6.64%)
3200	4	3461	102.09 (2.95%)	3518	137.55 (3.91%)
3200	8	4529	52.08 (1.15%)	4374	135.15 (3.09%)
3200	16	4698	123.55 (2.63%)	4872	122.77 (2.52%)
3200	32	4783	255.41 (5.34%)	4702	91.68 (1.95%)
32000	2	1207	27.88 (2.31%)	1274	17.45 (1.37%)
32000	4	3352	171.62 (5.12%)	3341	169.05 (5.06%)
32000	8	4288	84.90 (1.98%)	4351	101.81 (2.34%)
32000	16	4623	87.37 (1.89%)	4651	136.27 (2.93%)
32000	32	4695	77.93 (1.66%)	4610	108.79 (2.36%)

A.3 Gráficos de comparação

Na figura A.1, pode-se ver um gráfico da evolução dos valores de desempenho encontrados em um experimento sem concorrência (regravação de um arquivo de 3200Kbytes usando registro de 8Kbytes) , para os três sistemas de arquivo.

Na figura A.2, pode-se ver um gráfico da evolução dos valores de desempenho encontrados em um experimento com concorrência (regravação de um arquivo de 3200Kbytes usando registro de 8Kbytes) , para os três sistemas de arquivo. Pode-se notar que o sistema

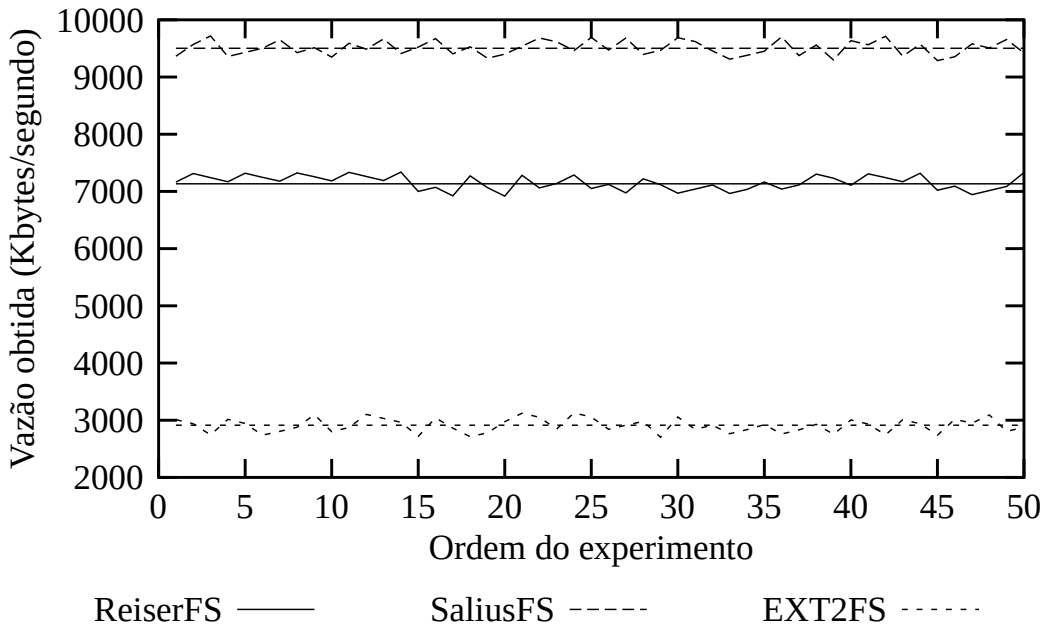

Figura A.1: Evolução do desempenho dos sistemas de arquivo sem concorrência

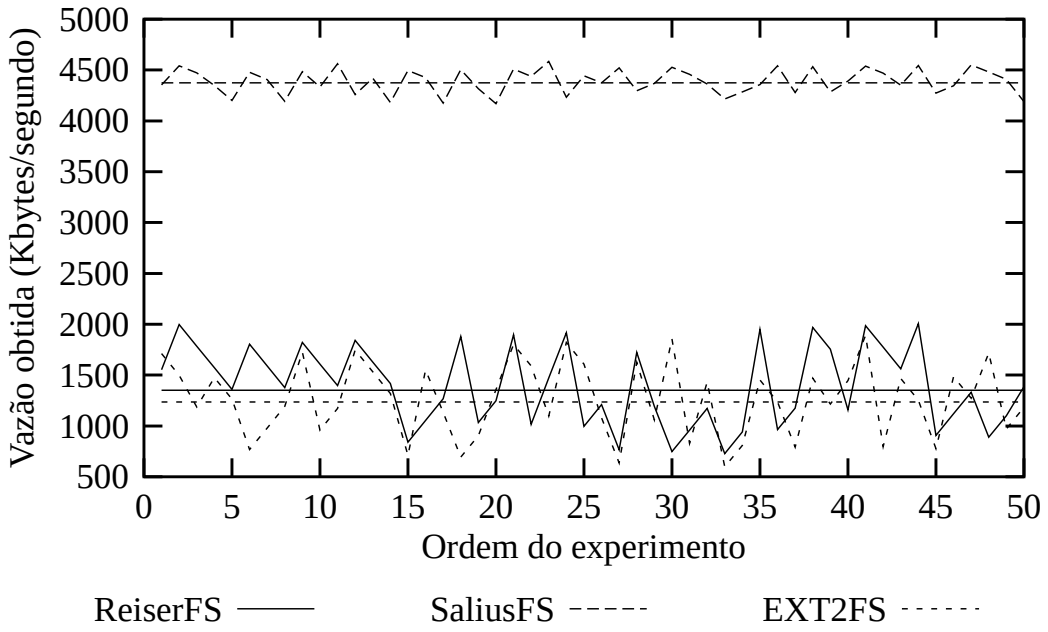

Figura A.2: Evolução do desempenho dos sistemas de arquivo com concorrência

Salius é mais estável que os demais.

A.4 Conclusão

Consideramos que o número de instâncias de experimentos, 50, foi suficientemente grande para chegarmos a valores representativos para o desempenho dos sistemas de arquivos sendo testados.