

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RN
BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

SISTEMAS DE ARQUIVOS COM JOURNALING DISPONÍVEIS PARA O LINUX

SISTEMAS DE ARQUIVOS COM JOURNALING DISPONÍVEIS PARA O LINUX

Rafael Melo Veras

Monografia apresentada ao curso de
Bacharelado em Sistemas de Informação da
Faculdade Natálense para o
Desenvolvimento do Rio Grande do Norte
como requisito parcial para a obtenção do
grau de Bacharel em Sistemas de
Informação.

Orientador:

Alcides Ferreira da Rocha Neto

**NATAL/RN
2006**

NATAL/RN
2006

RAFAEL MELO VERAS

SISTEMAS DE ARQUIVOS COM JOURNALING DISPONÍVEIS PARA O LINUX

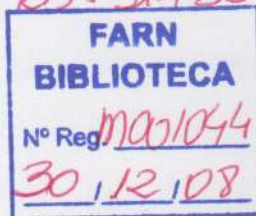
Monografia apresentada ao curso de Bacharelado em Sistemas de Informação da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte como requisito parcial para a obtenção do título de Bacherel em Sistemas de Informação.

Orientador:

Aluizio Ferreira da Rocha Neto

NATAL/RN
2006

RB-31756 M



004.451.9

V5845

21006

ex 01

Catálogo na Publicação – Biblioteca da FARN
Setor de Processos Técnicos

Veras, Rafael Melo

Sistemas de arquivos com journaling disponíveis para o Linux / Rafael Melo
Veras. – Natal: FARN, 2006.
39f.

Orientador: Aluizio Ferreira da Rocha Neto (M.Sc.)

Monografia (Bacharelado em Sistemas de Informação) – Faculdade Natalense
para o Desenvolvimento do Rio Grande do Norte.

1. Sistema de Arquivos – Monografia. 2. Sistema Operacional Linux –
Monografia. 3. Journaling – Monografia. I. Título.

RN/FARN/BC

CDU 004.451.9

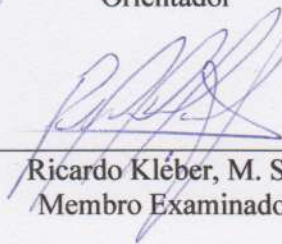
LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO
DO RIO GRANDE DO NORTE
SISTEMAS DE ARQUIVOS COM JOURNALING DISPONÍVEIS PARA O LINUX

Rafael Melo Veras

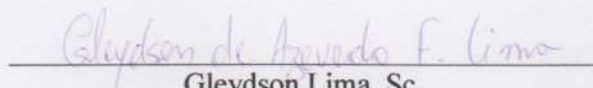
Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte e aprovada em __ de ____ de 2006 pela banca examinadora composta pelos seguintes membros:



Aluizio F. Rocha Neto, M. Sc.
Orientador



Ricardo Kléber, M. Sc.
Membro Examinador



Gleydson Lima, Sc.
Membro Examinador

RESUMO

Esta monografia visa ajudar usuários do sistema operacional Linux a entender um pouco mais sobre os sistemas de arquivos com *journaling*, conhecendo o funcionamento de cada um deles, suas principais características e os benefícios trazidos por essa nova funcionalidade, que proporciona uma rápida recuperação do sistema de arquivos após uma falha no sistema, restaurando o sistema de arquivos para um estado consistente em uma questão de segundos ou minutos. Além desse estudo esta monografia também demonstra testes comparativos realizados com o intuito de descobrir qual sistema de arquivos possui o melhor desempenho em situações diferentes, com isso descobrir para que finalidade o mesmo é mais adequado e assim poder ajudar os usuários sem conhecimentos avançados neste assunto a decidir qual sistema de arquivos se adapta melhor às suas necessidades.

Palavras-Chaves: Sistema de arquivos, *Journaling*, Desempenho.

ABSTRACT

This monograph aims at to help users of the operational system Linux to more understand a little on the systems of archives with journaling, knowing the functioning of each one of them, its main characteristics and the benefits brought for this new functionality, that provides a fast file system recovery after a system crash, restoring the file system to a consistent state in a matter of seconds or minutes. Beyond this study this monograph also demonstrates comparative tests carried through with intention to discover which file system posses optimum performance in different situations, with this to discover so that purpose the same is more adjusted and thus to be able to help the users without advanced knowledge in this subject to decide which file system is more adjusted for its reality.

Key words: File system, Journaling, Performance.

SUMÁRIO

1	INTRODUÇÃO.....	10
1.1	PROBLEMA DE PESQUISA.....	10
1.2	OBJETIVOS.....	11
1.2.1	Objetivos Específicos.....	11
1.3	ORGANIZAÇÃO DO TRABALHO	11
2	REFERENCIAL TEÓRICO.....	12
2.1	LINUX.....	12
2.2	SISTEMA DE ARQUIVOS.....	12
2.3	JOURNALING.....	13
2.4	EXT3 (<i>Third Extended File System</i>)	14
2.4.1	Tipos de <i>Journaling</i> no EXT3	14
2.5	REISERFS.....	15
2.5.1	Características do sistema de arquivos ReiserFS.....	16
2.6	JFS.....	16
2.6.1	Logs.....	17
2.6.2	Journaling.....	17
2.6.3	Tamanho variável de bloco.....	17
2.6.4	Alocação dinâmica de <i>inodes</i> de disco	18
2.6.5	Checagem/Reparação do Sistema de Arquivos	18
2.6.6	Tamanho máximo de um arquivo	18
2.7	XFS.....	18
2.7.1	Capacidade.....	19
2.7.2	Tamanho do Bloco.....	19
2.7.3	Alocação Extensiva.....	19
2.7.4	Journaling.....	19
2.8	COMPARATIVO DE LIMITAÇÕES DOS SISTEMAS DE ARQUIVOS.....	20
3	METODOLOGIA.....	21
4	DESENVOLVIMENTO.....	22
4.1	OS TESTES.....	22
4.2	REALIZANDO OS TESTES.....	22
4.3	SCRIPT DOS TESTES	26
4.4	RESULTADOS	29
5	CONCLUSÃO.....	38
	REFERENCIAL BIBLIOGRÁFICO	39

LISTA DE FIGURAS

Figura 1: Formatando a partição com o sistema de arquivos EXT2	24
Figura 2: Formatando a partição com o sistema de arquivos EXT3	24
Figura 3: Formatando a partição com o sistema de arquivos JFS	25
Figura 4: Formatando a partição com o sistema de arquivos REISERFS	25
Figura 5: Formatando a partição com o sistema de arquivos XFS.....	25
Figura 6: <i>Script</i> dos testes realizados	28
Gráfico 1: Gráfico do teste removendo o arquivo de 1Kb	33
Gráfico 2: Gráfico do teste descompactando o arquivo do kernel	33
Gráfico 3: Gráfico do teste compactando o arquivo do kernel	34
Gráfico 10c: Gráfico do teste copiando o arquivo do kernel	34
Gráfico 11: Gráfico do teste movendo o arquivo do kernel	35
Gráfico 12: Gráfico do teste removendo o arquivo do kernel	35

LISTA DE GRÁFICOS

Gráfico 1: Gráfico do teste criando 10 mil arquivos em um diretório	30
Gráfico 2: Gráfico do teste copiando o diretório com os 10 mil arquivos	30
Gráfico 3: Gráfico do teste removendo o diretório com os 10 mil arquivos	31
Gráfico 4: Gráfico do teste criando um arquivo de 1GB.....	31
Gráfico 5: Gráfico do teste movendo o arquivo de 1GB.....	32
Gráfico 6: Gráfico do teste copiando o arquivo de 1GB	32
Gráfico 7: Gráfico do teste removendo o arquivo de 1GB.....	33
Gráfico 8: Gráfico do teste descompactando o <i>source do kernel</i>	33
Gráfico 9: Gráfico do teste compactando o <i>source do kernel</i>	34
Gráfico 10: : Gráfico do teste copiando o <i>source do kernel</i>	34
Gráfico 11: Gráfico do teste movendo o <i>source do kernel</i>	35
Gráfico 12: Gráfico do teste removendo o <i>source do kernel</i>	35

LISTA DE TABELAS

Tabela 1: Limitações dos sistemas de arquivos.....	20
Tabela 2: Informações de hardware da máquina de teste.....	23
Tabela 3: Informações de software da máquina de teste.....	23
Tabela 4: Sumário do resultado dos testes referente ao tempo.....	36
Tabela 5: Sumário do resultado dos testes referente ao consumo da CPU.....	37

Uma das características mais importantes do Linux é o suporte para vários sistemas de arquivos diferentes. Isto torna muito flexível e fácil de trabalhar com muitos outros sistemas operacionais. Foi sempre de escita, o Linux suporta mais de 20 sistemas de arquivos diferentes, como oes extio: EXT, EXT2, XIA, MINIX, UMSDOS, NTFS, VFAT, FAT, NTFS, 9y9 PROC, SMI, NLP, IS0960, SYSV, HFS, AFPA, UFS, UFS, AFPA, UFS, UFS e finalmente os sistemas com suporte a *journaling*, como EXIA, ReiserFS, JFS e etc.

Devido à diversidade de sistemas de arquivos com *journaling* suportados pelo Linux e suas diferentes descomposições e modos de operação, surge a necessidade de analisá-los e compará-los.

Este trabalho visa analisar um estudo sobre cada um desses sistemas de arquivos com *journaling* citados acima, esclarecendo o funcionamento, analisar o desempenho, e aplicação de cada um deles, podendo ser útil para ajudar um usuário a fazer a escolha de qual sistema de arquivos é mais apropriado para suas necessidades ou apenas ter mais conhecimento sobre o assunto.

4.1 PROBLEMA DE PESQUISA

Uma das principais dificuldades para o desenvolvimento deste projeto foi a escassez de material na língua portuguesa e a ausência de uma literatura completa sobre este assunto.

Muitos usuários usam o sistema de arquivos padrão de sua distribuição sem saber se esta opção é a melhor para suas necessidades e para o equipamento que possui. Tendo em vista que esta escolha não é trivial se faz necessário o esclarecimento sobre estes sistemas de arquivos.

1 INTRODUÇÃO

Com o crescimento do Linux e sua crescente utilização em sistemas de servidores, novas características se tornam necessárias a ele. Uma delas é a de *Journaling*, que consiste em um mecanismo que fornece uma rápida recuperação do sistema de arquivos após uma falha.

Uma das características mais importantes do Linux é o suporte para vários sistemas de arquivos diferentes. Isto torna muito flexível e fácil de coexistir com muitos outros sistemas operacionais. Em tempo de escrita, o Linux suporta mais de 20 sistemas de arquivos diferentes, entre eles estão: EXT, EXT2, XIA, MINIX, UMSDOS, MSDOS, VFAT, NFS, NTFS, SysV PROC, SMB, NCP, ISO9660, SYSV, HPFS, AFFS, UFS, UDF, ADFS, BFS, QNX4 e atualmente os sistemas com suporte a *journaling*, como EXT3, ReiserFS, JFS E XFS.

Devido à diversidade de sistemas de arquivos com *journaling* suportados pelo Linux e seus diferentes desempenhos e modos de operação, surgiu a necessidade de testá-los e compará-los.

Este trabalho visa realizar um estudo sobre cada um desses sistemas de arquivos com *journaling* citados acima, esclarecendo o funcionamento, analisar o desempenho, e aplicação de cada um deles, podendo ser útil para ajudar um usuário a fazer a escolha de qual sistema de arquivos é mais apropriado para suas necessidades ou apenas ter mais conhecimento sobre o assunto.

1.1 PROBLEMA DE PESQUISA

Uma das principais dificuldades para o desenvolvimento deste projeto foi a escassez de material na língua portuguesa e a ausência de uma literatura completa sobre este assunto.

Muitos usuários usam o sistema de arquivos padrão da sua distribuição sem saber se esta opção é a melhor para suas necessidades e para o equipamento que possui. Tendo em vista que esta escolha não é trivial se faz necessário o esclarecimento sobre esses sistemas de arquivos.

1.2 OBJETIVOS

Realizar uma análise dos principais sistemas de arquivos com *journaling* disponíveis para o sistema operacional Linux, detalhando seus funcionamentos e realizar testes comparativos de desempenho.

1.2.1 Objetivos Específicos

A seguir, as etapas para o desenvolvimento deste trabalho:

- Estudo dos sistemas de arquivos: Ext3, ReiserFS, JFS, XFS
- Análise do funcionamento do *journaling* em cada um desses sistemas de arquivos
- Realização de testes de desempenho
- Comparação do desempenho
- Análise geral sobre cada sistema de arquivos.

1.3 ORGANIZAÇÃO DO TRABALHO

No Capítulo 2 será abordado o Referencial Teórico, com uma rápida introdução ao sistema operacional Linux, e a descrição de cada sistema de sistema de arquivos com *journaling*, detalhando seu funcionamento e suas principais características, o capítulo 3 apresenta a metodologia usada no desenvolvimento desta monografia, no capítulo 4 é elaborado o desenvolvimento onde os testes comparativos de desempenho são realizados, depois da análise feita dos resultados é chegada a conclusão no capítulo 5.

2 REFERENCIAL TEÓRICO

2.1 LINUX

Segundo (Wikipedia-LINUX, 2006), GNU/Linux ou simplesmente Linux, é um popular sistema operacional livre, composto pelo núcleo (*kernel*) Linux e pelas bibliotecas e ferramentas do projeto GNU. É um sistema do tipo Unix.

Tecnicamente falando, Linux é um núcleo (ou "cerne"; em Inglês 'kernel') o que propriamente se refere ao sistema de software que oferece uma camada de abstração referente a equipamentos como discos, controle de sistema de arquivos, multi-tarefa, balanceamento de carga, rede e segurança. Um núcleo (ou "cerne"; em Inglês 'kernel') não é um sistema operacional completo. (Wikipedia-LINUX, 2006).

Sistemas completos construídos em torno do *kernel* do Linux usam o sistema GNU que oferece um interpretador de comandos, utilitários, interfaces gráficas, bibliotecas, compiladores e ferramentas, bem como muitos outros programas como o editor Emacs etc. Por essa razão, Richard M. Stallman, criador e líder do projeto GNU, solicita aos usuários que se refiram ao Linux como o sistema completo GNU/Linux.

O Linux hoje funciona em dezenas de plataformas, desde *mainframes* até um relógio de pulso, passando por várias arquiteturas: Intel, StrongARM, PowerPC, Alpha etc., com grande penetração também em dispositivos embarcados, como *handheld*, PVR, vídeo-jogos e centrais de entretenimento etc. (Wikipedia-LINUX, 2006).

2.2 SISTEMA DE ARQUIVOS

Segundo (Wikipedia-Sistema_de_ficheiros, 2006), sistema de arquivos é a forma de organização de dados nos discos de armazenamento. Sabendo do sistema de arquivos de um determinado disco, o Sistema Operacional pode decodificar os dados armazenados e lê-los ou gravá-los.

Fazendo analogias, tal organização assemelha-se a uma biblioteca escolar. O bibliotecário organiza os livros conforme o seu gosto, cuja busca, convenientemente, procura deixar mais fácil, sem ocupar muitas prateleiras e assegurando a integridade deste. Ainda, certamente, organiza os livros segundo suas características (assunto, censura, etc). Depois de organizados, ou durante a organização, o bibliotecário cria uma lista com todos os livros da biblioteca, com seus assuntos, localizações e códigos respectivos.

O Sistema Operacional seria o bibliotecário da "biblioteca de dados" do computador: o disco de armazenamento. Exatamente igual à organização de uma biblioteca, o Sistema Operacional guarda os dados nos espaços vazios do disco, rotulando-os com um FCB (*File Control Block*, Bloco de Controle de Arquivo) e ainda criando uma lista com a posição deste dado, chamada de MFT (*Master File Table*, Tabela de Arquivos Mestre). Sabendo a posição do arquivo a ser aberto/gravado, o Sistema Operacional solicita a leitura desta, decodifica/codifica e realiza a abertura/gravação do dado.

Um sistema de arquivo é, assim, uma forma de criar uma estrutura lógica de acesso a dados numa partição. Sendo assim, também é importante referir que nunca poderá ter 2 ou mais tipos de sistemas de arquivos (formatados) numa mesma partição.

2.3 JOURNALING

Segundo (Wikipedia-Journaling_file_system, 2006), um sistema de arquivos com *Journaling* é um sistema de arquivos que registra todas as mudanças (gravação, atualização de dados) para um Jornal (*log* circular em uma área especialmente alocada) antes de realmente escrevê-los para o sistema de arquivo principal.

Após escrever completamente as mudanças elas são removidas do Jornal.

Segundo (ALECRIM, 2004), esta tecnologia foi criada para evitar ou diminuir a perda de dados, em sistemas de arquivos sem *Journaling*, no caso do EXT2 após um desligamento incorreto (uma queda de energia, por exemplo) poderia deixar o sistema de arquivo em um estado inconsistente. Assim depois de cada boot o programa FSCK (*file system check* ou *file system consistency check* é uma ferramenta usada para checar a consistência de um sistema de arquivos no Unix e clones) é acionado para cada sistema de arquivo (partição) que não foi desmontado corretamente e isto demora muito. Um sistema de arquivo com *Journaling* evita isso, verificando (modificações pendentes) somente as áreas que foram recentemente modificadas/acessadas que estavam gravadas no Jornal e não foi possível de serem gravadas. Dessa forma o risco de perda de dados e o tempo para checar o sistema de arquivo após uma falha diminuiriam drasticamente.

Existem vários sistemas de arquivos disponíveis com a tecnologia *Journaling*, como o XFS, desenvolvido originalmente pela Silicon Graphics e posteriormente disponibilizado com código aberto, o ReiserFS, desenvolvido especialmente para Linux, JFS, desenvolvido originalmente pela IBM mas também liberado com código aberto, e o mais conhecido deles: o EXT3, desenvolvido pelo Dr. Stephen Tweedie juntamente com outros colaboradores.

2.4 EXT3 (*Third Extended File System*)

Segundo (ALECRIM, Emerson, 2004), o sistema de arquivos EXT3 é basicamente o sistema de arquivos EXT2 com recursos de *Journaling*. Talvez, essa seja a razão de seu uso amplo: ele é totalmente compatível com EXT2 (que foi um sistema de arquivos muito usado), o que nenhum outro sistema de arquivos baseado em *Journaling* é.

O EXT3 passou a ser efetivamente suportado pelo *kernel* do Linux a partir da versão 2.4. Conseqüentemente, todas as distribuições Linux lançadas com esse *kernel* ou superior, tem suporte padrão para EXT3.

(ALECRIM, Emerson, 2004) define o seguinte:

No EXT3, o código de *Journaling* usa uma camada chamada "*Journaling Block Device*" (JBD). A JBD foi criada com o propósito de implementar *Journal* em qualquer tipo de dispositivo com base em blocos de dados. Por exemplo, o código EXT3 informa e "pede autorização" à JBD para efetuar as mudanças, antes de modificar/adicionar qualquer dado no disco. Sendo assim, é o JBD que verdadeiramente "gerencia" o *journal*. O fato mais interessante disso é que, a JBD funciona como uma entidade independente, permitindo que não só o EXT3 a use, mas também outros sistemas de arquivos.

A JBD utiliza um método diferente de outros *journalings* para recuperação de informações. Ao invés de armazenar as informações em bytes que depois devem ser implementados, a JBD grava os próprios blocos modificados do sistema de arquivos. Assim, o EXT3 também armazena "réplicas" completas dos blocos modificados em memória para rastrear as operações que ficaram pendentes. A desvantagem desta forma de trabalho é que o *journal* acaba sendo maior. No entanto, o EXT3 não precisa lidar com a complexidade dos *Journalings* que trabalham gravando bytes.

2.4.1 Tipos de *Journaling* no EXT3

Segundo (AMARAL, 2004), o EXT3 suporta três diferentes modos de trabalho do *Journaling*. São eles:

- **Journal:** grava todas as mudanças em sistema de arquivos. É o mais lento dos três modos, mas é o que possui maior capacidade de evitar perda de dados;
- **Ordered:** grava somente mudanças em arquivos *metadata* (arquivos que guardam informações sobre outros arquivos), mas guardam as atualizações no arquivo de dados antes de fazer as mudanças associadas ao sistema de arquivos. Este *journaling* é o padrão nos sistemas de arquivos EXT3;

- **Writeback:** também só grava mudanças para o sistema de arquivo em *metadata*, mas utiliza o processo de escrita do sistema de arquivos em uso para gravação. É o mais rápido *Journaling* EXT3, mas o menos confiável. O modo *Ordered* é o padrão no EXT3, mas é possível especificar qual o modo que você deseja usar, através da atualização do arquivo *fstab*.

2.5 REISERFS

Segundo (NOGUEIRA, Roger Fonseca, 2003), foi concebido por Hans Reiser e é mantido hoje por *The Naming System Venture*. O núcleo deste sistema de arquivos é baseado em árvores B*, uma versão melhorada de árvores B+. A principal diferença é que todos os objetos do sistema de arquivos estão numa mesma árvore B*. Isto significa que não existe uma árvore diferente para cada diretório, mas cada diretório possui uma sub-árvore da principal. Este tipo de abordagem em que todos diretórios, blocos de arquivos, atributos de arquivos, *links*, etc., são organizado dentro de uma mesma árvore, exigiu que técnicas mais complexas de indexação (*hashing*) fossem implementadas no ReiserFS.

Outra diferença deste sistema de arquivos é que ele ainda não suporta *extents*, que são conjuntos de blocos lógicos contíguos utilizados para melhorar a localidade espacial. E também contrastando com os outros sistemas de arquivos, ele exige um mínimo de 8192 blocos de alocação para o *journal*, enquanto os demais utilizam por volta de 1024 blocos.

O ReiserFS, por sua vez, trabalha com o conceito de *journaling*. Contudo armazena em *log* apenas informações *metadata*, ou seja, informações referentes ao tamanho (espaço ocupado) do arquivo e suas permissões. É mais rápido que o EXT3 e mais lento (obviamente) que o EXT2. No caso de um desligamento incorreto é possível localizar todas as operações que não haviam sido completadas, restaurando a consistência do sistema de arquivos sem a necessidade de vasculhar arquivo por arquivo, como faz o SCANDISK do Windows ou o FSCK no Linux. Isso significa dizer que o ReiserFS consegue recuperar o sistema de arquivos em fração de segundos (caso ocorra algum sinistro) e a perda de pastas e arquivos é praticamente nula. Em compensação, os arquivos que eventualmente estiverem sendo gravados no exato momento em que acabou a energia ficarão com seus dados alterados. Você continuará tendo acesso aos arquivos normalmente, mas o conteúdo estará truncado ou incompleto. Na realidade isso pode ocorrer porque o ReiserFS não armazena as informações em si dos arquivos, apenas suas informações *metadata*.

2.5.1 Características do sistema de arquivos ReiserFS

As características principais do ReiserFS são as seguintes:

- Projetado por Hans Reiser, tem a grande vantagem de ser o primeiro sistema de arquivos baseado em *Journaling* a ser incluído como parte integrante do *kernel* do Linux.
- Utiliza uma abordagem radicalmente diversa da alocação baseada em blocos usada no EXT2 para armazenar as informações, usando “*balanced trees*”.
- Ainda não oferece suporte a quotas para usuários e grupos.
- Permite arquivos com até 4 Gb de tamanho.
- Possui otimizações para trabalhar com arquivos pequenos, reduzindo a fragmentação.

2.6 JFS

Segundo (BEST, Steve, 2000), o *Journaled File System* (JFS) é um sistema de arquivo desenvolvido para sistemas de alta performance. Maleável e robusto sua vantagem em cima dos sistemas de arquivos não jornalados é a sua capacidade de reinicializar rapidamente. O JFS pode restaurar o sistema de arquivo para um estado consistente em questão de segundos.

O sistema de arquivo JFS também usa a estrutura *inode* para armazenar a localização dos blocos de cada arquivo nas estruturas físicas do disco, esses blocos podem variar de 512 à 4096 bytes, a alocação dos *inodes* é feita conforme vai sendo necessário.

Agregados: A ferramenta de criação do sistema de arquivos, *mkfs*, cria um agregado que é completamente contido dentro de uma partição. Um agregado é um *array* dos blocos do disco que contém um formato específico que inclui um superbloco e um mapa de alocamento. O superbloco identifica a partição como um agregado JFS, enquanto o mapa de alocamento descrever o estado de alocação de cada bloco de dados dentro do agregado. O formato inclui também os *filesets* iniciais e as estruturas de controle necessárias para descrevê-lo. O *fileset* é uma entidade montável.

Filesets: Um *fileset* contém arquivos e diretórios. Arquivos e diretórios são representados persistentemente por *inodes*; cada *inode* descreve os atributos de um arquivo ou diretório e serve como o ponto de partida para encontrar os dados, arquivos ou diretórios no

disco. JFS usa também *inodes* para representar outros objetos do sistema de arquivo, tais como o mapa que descreve o estado de alocação e o local no disco de cada *inode* no *fileset*.

O superbloco agregado, o mapa de alocação de disco, o descritor de arquivos, o mapa de *inodes*, os *inodes*, os diretórios e as estruturas de endereçamento representam as estruturas de controle ou metadados do JFS.

2.6.1 Logs

Os *logs* do JFS são mantidos em cada agregado e usados para gravar informações sobre operações nos metadados. O *log* tem um formato que também é ajustado pelo utilitário de criação do sistema de arquivos. Um único *log* pode ser usado simultaneamente por múltiplos *filesets* montados no agregado.

2.6.2 Journaling

Ele somente armazena em *logs* os metadados. A recuperação de um *log* somente restaura a consistência das relações estruturais e o estado da alocação de recursos. Ele não armazena em *log* dados sobre os arquivos ou recupera esses dados para um estado consistente.

Conseqüentemente, alguns dados de arquivos podem ser perdidos durante a recuperação.

2.6.3 Tamanho variável de bloco

O JFS suporta tamanhos de bloco de 512, 1024, 2048 e 4096 bytes por sistema de arquivos, permitindo que os usuários otimizem a utilização do espaço baseado no seu ambiente de aplicação. Os tamanhos de bloco menores reduzem a quantidade de fragmentação interna dentro dos arquivos e diretórios e são mais eficientes em relação ao espaço. Entretanto, os blocos pequenos podem aumentar o comprimento do caminho desde que as atividades do alocamento do bloco podem ocorrer com mais frequência do que se forem usados blocos grandes. O tamanho padrão do bloco é 4096 bytes tendo em vista que o desempenho, seja mais importante do que a utilização do espaço, e geralmente esta consideração é preliminar para servidores.

2.6.4 Alocação dinâmica de *inodes* de disco

O JFS aloca dinamicamente o espaço para *inodes* de disco conforme vai sendo necessário, liberando esse espaço quando não for mais necessário. Esse suporte impede que os *inodes* se localizem em regiões fixas do disco.

2.6.5 Checagem/Reparação do Sistema de Arquivos

O objetivo do utilitário de checagem/reparação é proporcionar um estado estável ao sistema de arquivos para prevenir falhas e corrupção do mesmo, com um objetivo secundário de preservar os dados no caso de corrupção. O utilitário pode descartar dados a fim de garantir a consistência do sistema de arquivos. Qualquer arquivo ou subdiretório órfão pela deleção de um diretório corrompido é colocado no diretório *lost+found* localizado na raiz do sistema de arquivos.

2.6.6 Tamanho máximo de um arquivo

O tamanho máximo de arquivo é o tamanho do maior arquivo que a estrutura virtual do sistema de arquivo (*VFS – Virtual File System*) suporta. Por exemplo, se o *frame work* suportar somente 32 bits, então isto limita o tamanho do arquivo.

2.7 XFS

Segundo (Wikipedia-XFS, 2006), XFS é um, sistema de arquivos de alta performance com *Journaling* criado pela Silicon Graphics Inc. para seu sistema operacional IRIX.

XFS é o sistema de arquivos *Journaling* mais velho disponível para sistemas UNIX, e tem um maduro, estável e bem testado código. O desenvolvimento do XFS foi começado pela Silicon Graphics, em 1993, com a primeira distribuição sendo vista no IRIX 5.3 em 1994.

O sistema de arquivos foi liberado sob a *GNU General Public License* em maio de 2000, e portado para o Linux, com a primeira distribuição com suporte aparecendo em 2001/2002.

O XFS foi fundido a linha principal do Kernel do Linux 2.4 e 2.6, tornando-se quase que universal sua disponibilidade em sistemas Linux. Os programas da instalação das distribuições SuSE, Gentoo, Mandriva, Slackware, Fedora, Ubuntu e Debian Linux, todas disponibilizam o XFS como uma opção para sistema de arquivos.

2.7.1 Capacidade

Segundo (Wikipedia-XFS, 2006), XFS é um sistema de arquivos *Journaling* de 64-bits, com consistência garantida do sistema de arquivos. Suporta um tamanho máximo do sistema de arquivos de 9 Exabytes, embora esteja sujeito aos limites do bloco impostos. Em sistemas Linux de 32-bit, os limites de tamanho dos arquivos e do sistema de arquivos é de 16 Terabytes.

2.7.2 Tamanho do Bloco

Segundo (Silicon Graphics, Inc - XFS, 2006), o tamanho de bloco mínimo do sistema de arquivo é 512 bytes. O tamanho de bloco máximo é o tamanho da página do *kernel*, que é 4K na arquitetura x86 e é ajustada no *kernel* quando compilado para a arquitetura IA64 (páginas de até 64 kilobytes). Assim, o XFS suporta tamanhos de bloco do sistema de arquivos de até 64 kilobytes (de 512 bytes, em potência de 2), quando o tamanho da página do *kernel* o permite.

2.7.3 Alocação Extensiva

Segundo (NOGUEIRA, 2003), XFS usa árvores *B+tree* extensivamente no lugar da estrutura tradicional de sistema de arquivos. *B+trees* usam um método altamente eficiente de indexação para indexar entradas em diretórios, gerenciar dimensões de arquivos, localizar espaço livre, e acompanhar as localizações das informações dos índices de arquivo. Como resultado, ler sistemas de arquivos e recuperar informações deles acontece rapidamente - sem usar grandes quantidades de recursos do sistema.

2.7.4 Journaling

Segundo (Wikipedia-XFS, 2006), XFS fornece *journaling* para os metadados do sistema de arquivo, onde as atualizações do sistema de arquivos são primeiro escritas para um jornal em série antes que os blocos reais do disco sejam atualizados. O jornal é um *buffer* circular de blocos do disco que nunca é lido em uma operação normal do sistema de arquivos. Pode ser armazenado dentro da seção dos dados do sistema de arquivos (um *log* interno), ou em um dispositivo separado para minimizar a disputa do disco. As atualizações do jornal são executadas de forma assíncrona para evitar incorrer uma falha no desempenho. Se ocorrer uma falha no sistema, as operações imediatamente antes da falha podem ser refeitas usando

dados no jornal, que permite que XFS garanta a consistência do sistema de arquivos. A recuperação é executada automaticamente quando o sistema de arquivos é montado, e é independente do tamanho do sistema de arquivos. Onde os dados recentemente modificados não foram nivelados ao disco antes de uma falha de sistema, XFS assegura-se de que todos os blocos de dados não escritos estejam zerados no *reboot*, prevenindo quaisquer edições de segurança possíveis que se levantem dos dados residuais.

2.8 COMPARATIVO DE LIMITAÇÕES DOS SISTEMAS DE ARQUIVOS

A Tabela 1 abaixo sumariza as informações dos limites referente ao tamanho máximo do nome de um arquivo, caracteres permitidos no nome de diretórios, tamanho máximo de um arquivo e tamanho máximo do volume, de cada sistema de arquivo com *journaling* testados, incluindo também o EXT2, um sistema de arquivo sem *journaling*, apenas para referência.

Nome do sistema de arquivo	Tamanho máximo do nome do arquivo	Caracteres permitidos no nome dos diretórios	Tamanho máximo de um arquivo	Tamanho máximo do volume
EXT2	255 bytes	Qualquer byte exceto nulo	2GB	4TB
EXT3	255 bytes	Qualquer byte exceto nulo	2GB	4TB
REISERFS	4032 bytes/255 caracteres	Qualquer byte exceto nulo	8Ti	16Ti
JFS	255 bytes	Qualquer byte exceto nulo	4PB	4PB
XFS	255 bytes	Qualquer byte exceto nulo	16TB	16TB

Tabela 1: Limitações dos sistemas de arquivos

3 METODOLOGIA

Esta monografia é uma pesquisa exploratória e descritiva quando elaborada a partir de material já publicado, constituído principalmente de livros, artigos de periódicos e materiais da Internet, baseado nesse material foi realizado um estudo sobre a técnica *journaling* e sobre os sistemas de arquivos que possuem essa funcionalidade, mostrando seu funcionamento e suas principais características. Realizando também um estudo de caso com o objetivo de realizar uma pesquisa sólida e detalhada. Foi utilizado um computador com o sistema operacional Linux, com dois discos rígidos, um onde executava o sistema operacional e o outro onde foram feitos os testes práticos de desempenho baseado em tarefas comuns, feitas no dia-a-dia, como a criação e remoção de arquivos, com arquivos de diferentes tamanhos, compactação e descompactação, para que se pudesse obter dados e assim realizar um comparativo entre os sistemas de arquivos em questão e tirar conclusões sobre cada um destacando suas possíveis utilizações.

6. Copiar o arquivo de 1GB

7. Remover o arquivo de 1GB

8. Descompactar o source do kernel

9. Compactar o source do kernel

10. Copiar o source do kernel

11. Mover o source do kernel

12. Remover o source do kernel

4.2 REALIZANDO OS TESTES

Os testes foram realizados em um disco rígido adicional instalado na máquina de teste. Foi criada uma partição (*dev/sdd1*) no disco de teste, esta seguida formatada e então foi executado o script de teste. O script foi executado duas vezes em cada sistema de arquivos, depois foi feita a média dos dois resultados gerados e com essa média foram gerados os gráficos. Foram realizados os testes também com o sistema de arquivos EXT2 apenas como referência de um sistema de arquivos sem *journaling*.

4 DESENVOLVIMENTO

4.1 OS TESTES

Ao escolher os testes para verificar o desempenho de cada sistema de arquivos, pensou-se em alguns testes bem práticos, relacionado a tarefas do dia-a-dia, como copiar, mover, remover, compactar e descompactar arquivos. Baseado nisso foram elaborados 12 testes, que são:

1. Criar 10 mil arquivos em um diretório
2. Copiar o diretório com os 10 mil arquivos
3. Remover o diretório com os 10 mil arquivos
4. Criar um arquivo de 1GB
5. Mover o arquivo de 1GB
6. Copiar o arquivo de 1GB
7. Remover o arquivo de 1GB
8. Descompactar o *source* do *kernel*
9. Compactar o *source* do *kernel*
10. Copiar o *source* do *kernel*
11. Mover o *source* do *kernel*
12. Remover o *source* do *kernel*

4.2 REALIZANDO OS TESTES

Os testes foram realizados em um disco rígido adicional instalado na máquina de teste. Foi criada uma partição `/dev/hdd1` no disco de teste, em seguida formatada e então foi executado o script de teste. O script foi executado duas vezes em cada sistema de arquivos, depois foi feita a média dos dois resultados gerados e com essa média foram gerados os gráficos. Foram realizados os testes também com o sistema de arquivos EXT2 apenas como referência de um sistema de arquivos sem *journaling*.

Especificações da máquina de teste:

Hardware	
Placa Mãe	MSI KT6V
Processador	AMD Athlon(tm) XP 2600+ (2085.632 MHz)
Memória RAM	512 MB
SWAP	256 MB
Disco Rígido	Seagate ST34311A 4.33GB Ultra ATA/66 5400RPM

Tabela 2: Informações de hardware da máquina de teste

Software	
Distribuição do Linux	Ubuntu Linux (Dapper) Com todos os pacotes atualizados em: 26/10/2006.
Kernel	2.6.15-27-386 (Kernel padrão mais recente da distribuição)
e2fsprogs (mkfs.ext2 e mkfs.ext3)	1.38-2ubuntu2
jfsutils (mkfs.jfs):	1.1.8-1
reiserfsprogs (mkfs.reiserfs):	3.6.19-1
xfsprogs (mkfs.xfs):	2.7.7-1

Tabela 3: Informações de software da máquina de teste

Os testes foram realizados com apenas 1 usuário logado no sistema, em modo texto (sem a interface gráfica em funcionamento) e apenas com processos essenciais sendo executados no momento.

Os processos que estavam sendo executados durante os testes eram: init, ksoftirqd/0, watchdog/0, events/0, khelper, kthread, kblockd/0, kacpid, pdflush, pdflush, aio/0, kswapd0, kseriod, ata/0, ata_hotplug/0, scsi_eh_0, scsi_eh_1, khubd, kjournald, shpchpd_event, kgameportd, syslogd, dd, klogd, sshd, getty, getty, getty, getty, getty, getty, udevd, sshd, bash.

```

Superblock backups stored on blocks:
21785, 39304, 143340, 328176, 544812, 819320, 942736

Writing inode tables done!
Creating journal (26768 blocks): done
Writing superblocks and filesystem information: done

This filesystem will be automatically checked every 30 seconds or
180 days, whichever comes first. Use tune2fs -l or -l to override.

```

Figura 2: Particionando a partição com o sistema de arquivos EXT3

Criando os sistemas de arquivos:

Todos os sistemas de arquivos foram criados com as opções padrões.

EXT2:

```
root@explode:~# mkfs.ext2 /dev/hdd1
mke2fs 1.38 (30-Jun-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
529056 inodes, 1056502 blocks
52825 blocks (5.00%) reserved for the super user
First data block=0
33 block groups
32768 blocks per group, 32768 fragments per group
16032 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 27 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Figura 1: Formatando a partição com o sistema de arquivos EXT2

EXT3:

```
root@explode:~# mkfs.ext3 /dev/hdd1
mke2fs 1.38 (30-Jun-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
529056 inodes, 1056502 blocks
52825 blocks (5.00%) reserved for the super user
First data block=0
33 block groups
32768 blocks per group, 32768 fragments per group
16032 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 39 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Figura 2: Formatando a partição com o sistema de arquivos EXT3

JFS:

```

root@explode:~# mkfs.jfs -q /dev/hdd1
mkfs.jfs version 1.1.8, 03-May-2005
/
Format completed successfully.
4226008 kilobytes total disk space.

```

Figura 3: Formatando a partição com o sistema de arquivos JFS**REISERFS:**

```

root@explode:~/monografia# mkfs.reiserfs -q /dev/hdd1
mkfs.reiserfs 3.6.19 (2003 www.namesys.com)

A pair of credits:
Many persons came to www.namesys.com/support.html, and got a question
answered
for $25, or just gave us a small donation there.

Alexander Zarochentcev (zam) wrote the high low priority locking code,
online
resizer for V3 and V4, online repacker for V4, block allocation code, and
major
parts of the flush code, and maintains the transaction manager code. We
give
him the stuff that we know will be hard to debug, or needs to be very
cleanly
structured.

```

Figura 4: Formatando a partição com o sistema de arquivos REISERFS**XFS:**

```

root@explode:~# mkfs.xfs -f /dev/hdd1
meta-data=/dev/hdd1          isize=256    agcount=8, agsize=132062
blks
      =                       sectsz=512    attr=0
data      =                   bsize=4096    blocks=1056496, imaxpct=25
      =                       sunit=0       swidth=0 blks, unwritten=1
naming    =version 2          bsize=4096
log        =internal log      bsize=4096    blocks=2560, version=1
      =                       sectsz=512    sunit=0 blks
realtime   =none              extsz=65536   blocks=0, rtextents=0

```

Figura 5: Formatando a partição com o sistema de arquivos XFS

4.3 SCRIPT DOS TESTES

O script de teste, mostrado na Figura 6, foi feito em *bash*, um interpretador de comandos, uma espécie de tradutor entre o sistema operacional e o usuário, normalmente conhecido como *shell* que permite a execução de seqüências de comandos direto no *prompt* do sistema ou escritas em arquivos de texto (como foi usado nesta ocasião), conhecidos como *shell scripts*.

```

1  #!/bin/bash
2
3  #####
4  # Rafael M. Veras - 04 / 10 / 2006
5  # Orientador: Aluizio Rocha
6  # Script para teste de performance do FS
7  #
8  #####
9  # Ordem dos testes...
10 #
11 # 1. Criar 10 mil arquivos em um diretório
12 # 2. Copiar o diretório com os 10 mil arquivos
13 # 3. Remover o diretório
14 # 4. Criar 10 mil arquivos
15 # 5. Copiar os 10 mil arquivos
16 # 6. Remover os 10 mil arquivos
17 # 7. Criar um arquivo de 1GB com o dd
18 # 8. Mover o arquivo
19 # 9. Copiar o arquivo
20 # 10. Remover o arquivo
21 # 11. Descompactar o source do kernel
22 # 12. Compactar o source do kernel
23 # 13. Copiar
24 # 14. Mover
25 # 15. Remover
26 #
27 #####
28
29 TIMEFORMAT='%Tempo:\t %Rs\nCPU:\t %P%'
30 DIR="$PWD/dirteste"
31 SODIR="/localdisk"
32 FST=$(mount|grep /dev/hdd1 |awk '{print $5}')
33 KSOURCE="linux-2.6.18.tar.gz"
34 EXTEN=".tar.gz"
35 KSRCDIR=$(basename $KSOURCE $EXTEN)
36
37 echo "===== ";
38 echo " Sistema de arquivos: $FST ";
39 echo "===== ";
40 echo "";
41
42 function create {
43     if [ "$1" == "files" ]; then
44         mkdir $DIR
45         echo -e "\nCriando 10 mil arquivos:";
46         cd $DIR
47         time touch file{1..10000}
48         cd - > /dev/null

```

```

49         elif [ "$1" == "lgb" ]; then
50             echo -e "\nCriando um arquivo de 1GB:";
51             time dd if=/dev/zero of=filelgb bs=1024
count=1000000 > /dev/null 2>&1
52         else
53             exit 1
54         fi
55     }
56
57     function move {
58         if [ "$1" == "lgb" ]; then
59             echo -e "\nMovendo o arquivo de 1GB:"
60             time mv filelgb $SODIR
61             mv $SODIR/filelgb .
62         elif [ "$1" == "kernel" ]; then
63             echo -e "\nMovendo o source do kernel:"
64             time mv $KSRCDIR $SODIR
65             mv $SODIR/$KSRCDIR .
66         else
67             exit 1
68         fi
69         rm -rf tmp
70     }
71
72     function copy {
73         if [ "$1" == "dir" ]; then
74             echo -e "\nCopiando o diretorio:";
75             time cp -rp $DIR "$DIR-copy"
76             rm -rf "$DIR-copy"
77         elif [ "$1" == "files" ]; then
78             echo -e "\nCopiando os 10 mil arquivos:";
79             mkdir temp
80             cd $DIR
81             time cp file* ../temp/
82             cd - > /dev/null
83             rm -rf temp
84         elif [ "$1" == "lgb" ]; then
85             echo -e "\nCopiando o arquivo de 1GB:"
86             time cp -p filelgb copy-filelgb
87             rm -f copy-filelgb
88         elif [ "$1" == "kernel" ]; then
89             echo -e "\nCopiando o source do kernel:"
90             time cp -r $KSRCDIR $KSRCDIR-copy
91             rm -rf $KSRCDIR-copy
92         else
93             exit 1
94         fi
95     }
96
97     function remove {
98         if [ "$1" == "files" ]; then
99             echo -e "\nRemovendo os 10 mil arquivos:";
100             cd $DIR
101             time rm -f file{1..10000}
102             cd - > /dev/null
103         elif [ "$1" == "lgb" ]; then
104             echo -e "\nRemovendo o arquivo de 1GB:"
105             time rm -f filelgb
106         elif [ "$1" == "dir" ]; then
107             echo -e "\nRemovendo o diretorio:";
108             time rm -rf $DIR

```

```

109         elif [ "$1" == "kernel" ]; then
110             echo -e "\nRemovendo o source do kernel:"
111             time rm -rf $KSRCDIR
112         else
113             exit 1
114         fi
115     }
116
117     function ksource {
118         if [ "$1" == "descompactar" ]; then
119             echo -e "\nDescompactando o kernel:"
120             time tar -zxvf $KSOURCE
121         elif [ "$1" == "compactar" ]; then
122             echo -e "\nCompactando o kernel:"
123             time tar -czf new-$KSOURCE $KSRCDIR
124             rm -rf new-$KSOURCE
125         else
126             exit 1
127         fi
128     }
129
130     # 1
131     create "files"
132     sync && sleep 10
133     # 2
134     copy "dir"
135     sync && sleep 10
136     # 3
137     remove "dir"
138     sync && sleep 10
139     # 4
140     create "lgb"
141     sync && sleep 10
142     # 5
143     move "lgb"
144     sync && sleep 10
145     # 6
146     copy "lgb"
147     sync && sleep 10
148     # 7
149     remove "lgb"
150     sync && sleep 10
151     # 8
152     ksource "descompactar"
153     sync && sleep 10
154     # 9
155     ksource "compactar"
156     sync && sleep 10
157     # 10
158     copy "kernel"
159     sync && sleep 10
160     # 11
161     move "kernel"
162     sync && sleep 10
163     # 12
164     remove "kernel"
165     sync && sleep 10
166
167     echo -e "Terminou.\a"
168     exit 0

```

Figura 6: Script dos testes realizados

O script foi dividido em 5 funções:

1. Função create (linha 42): usada para criar os 10 mil arquivos com o comando *touch* e o arquivo de 1GB com o comando *dd*.
2. Função move (linha 57): usada para fazer os testes para mover o *source do kernel* e o arquivo de 1GB da partição de teste para a partição em que o S.O. esta rodando usando o comando *mv*.
3. Função copy (linha 72): realiza os testes de copia, usada para copiar um diretório, os 10 mil arquivos criados com o *touch* na função *create*, o arquivo de 1GB e o *source do kernel*.
4. Função remove (linha 97): usada para remover os arquivos criados com a função *create* e o *source do kernel*, utilizando o comando *rm*.
5. Função ksource (linha 117): função usada para compactar e descompactar o *source do kernel*, usando o comando *tar*.

Depois de criadas as funções apenas foram chamadas passando-se os parâmetros para realizar a sequência de teste que foi descrita no tópico 4.1. Entre cada teste o comando *sync* e *sleep* foram rodados, para esvaziar os *buffers* do sistema de arquivos (efetivar as mudanças no disco) e aguardar 10 segundos para evitar que o próximo teste seja afetado pelo anterior. Para se obter os resultados de tempo decorrido (em segundos) e consumo de CPU (percentual) utilizado para a realização da tarefa foi usado o comando *time*. A porcentagem de uso da CPU foi obtida pelo *time* somando-se o tempo gasto em segundos usando a CPU no modo usuário mais o tempo gasto em segundos usando a CPU no modo sistema e dividindo esta soma pelo tempo decorrido durante a execução da tarefa.

4.4 RESULTADOS

Para uma melhor compreensão dos resultados gerados pelo *script* de teste, serão mostrados gráficos de desempenho de cada um dos testes para cada sistema de arquivos avaliado.

No primeiro teste realizado, o de criar 10 mil arquivos em um diretório, o ReiserFS liderou, completando a tarefa em apenas 0,4 segundos (Gráfico 1).



Gráfico 1: Gráfico do teste criando 10 mil arquivos em um diretório

Copiando o diretório contendo os 10 mil arquivos criados no teste 1 (Gráfico 2), o ReiserFS também foi o mais rápido. Logo atrás veio o Ext2, porém com uma diferença considerável de 1.9 segundos. Nesses dois primeiros testes o JFS ficou na média de tempo dos demais sistemas de arquivos, mas com um consumo muito menor da CPU.



Gráfico 2: Gráfico do teste copiando o diretório com os 10 mil arquivos

No teste do Gráfico 3, novamente o ReiserFS foi o mais rápido com 0.4 segundos para completar a tarefa, e novamente o Ext2 foi o que mais se aproximou do ReiserFS. Nos três

primeiros testes percebe-se uma superioridade elevada do ReiserFS possivelmente pelo uso das árvores balanceadas.



Gráfico 3: Gráfico do teste removendo o diretório com os 10 mil arquivos

No teste do Gráfico 4, dentre os sistemas de arquivos com *journaling*, o JFS obteve o melhor desempenho realizando a tarefa com o mesmo tempo do ReiserFS, mas usando aproximadamente sete vezes menos a CPU, o XFS também teve um desempenho semelhante.

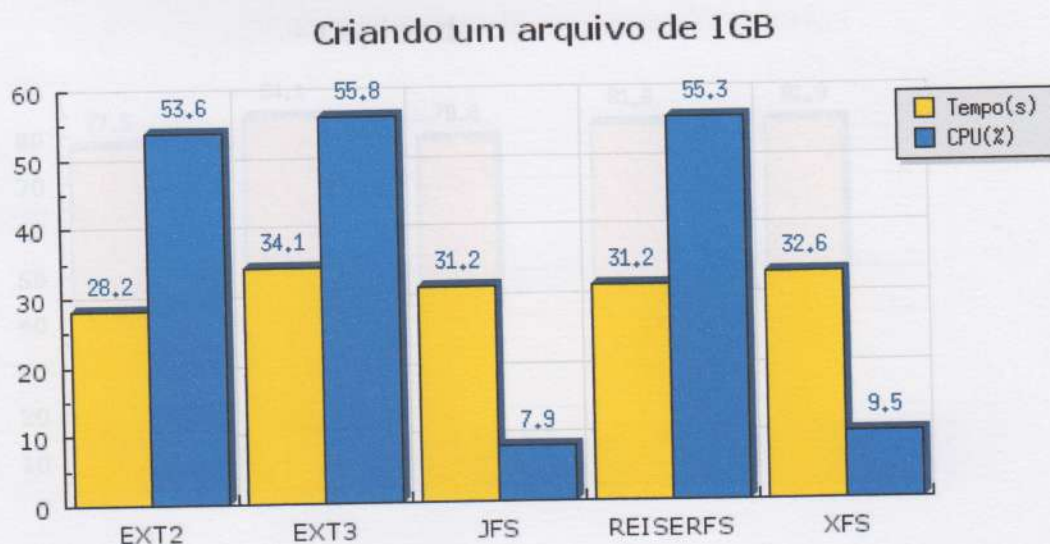


Gráfico 4: Gráfico do teste criando um arquivo de 1GB

O mesmo resultado se repete ao mover o arquivo de 1GB (Gráfico 5), só que neste teste o ReiserFS foi um segundo mais rápido, mas usando aproximadamente seis vezes mais a CPU.

Movendo o arquivo de 1GB

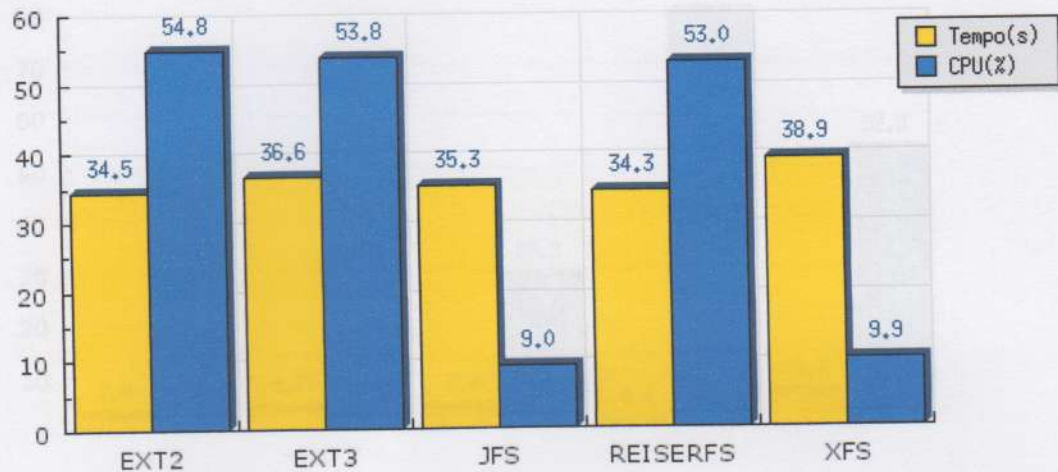


Gráfico 5: Gráfico do teste movendo o arquivo de 1GB

Copiando o arquivo de 1GB (Gráfico 6), o JFS conseguiu ser o mais rápido dentre os sistemas de arquivos com *journaling*, com o menor uso da CPU, utilizando apenas 5.6%.

Copiando o arquivo de 1GB

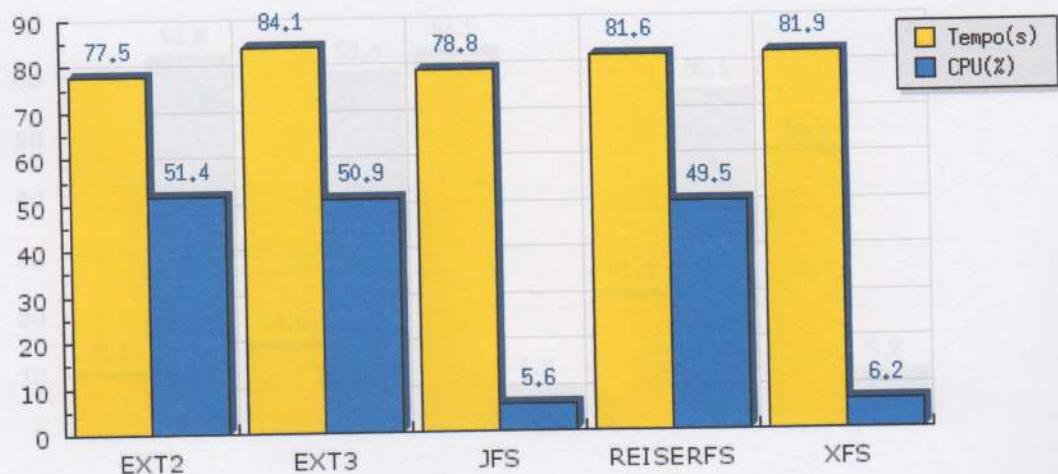


Gráfico 6: Gráfico do teste copiando o arquivo de 1GB

No teste do Gráfico 7, percebe-se claramente que trabalhar com arquivos grandes o JFS e o XFS saem-se muito bem, considerando o baixo uso da CPU. O ReiserFS foi bem rápido, mas com alto uso da CPU.

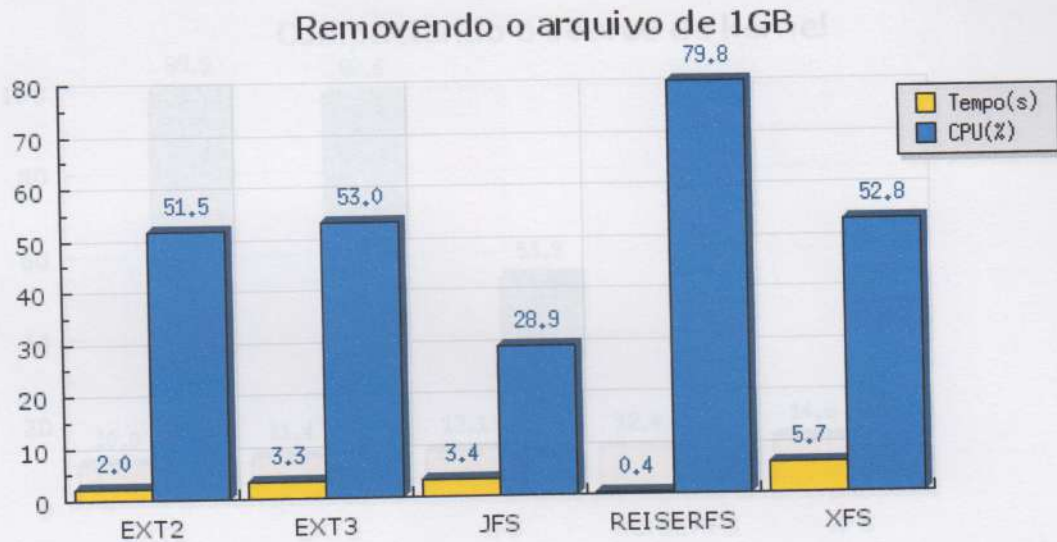


Gráfico 7: Gráfico do teste removendo o arquivo de 1GB

O teste do Gráfico 8 consiste na descompactação do código fonte do *kernel*. O Ext2 e Ext3 foram bem superiores aos demais.

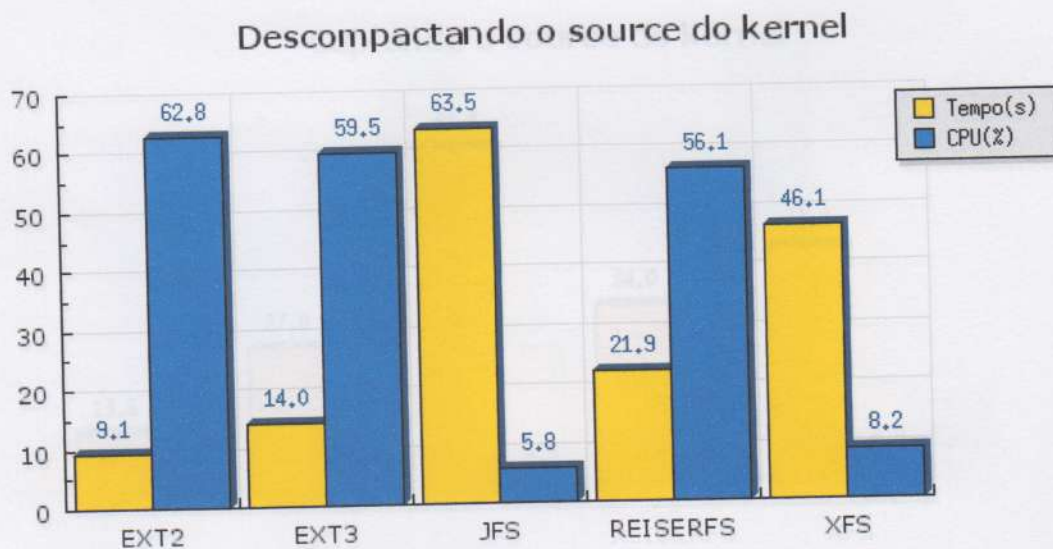


Gráfico 8: Gráfico do teste descompactando o source do kernel

Ao compactar o código fonte do *kernel* (Gráfico 9), novamente o Ext2 e o Ext3 foram mais rápidos, mas com um alto uso da CPU. O JFS realizou a tarefa com um tempo bem próximo dos demais e com o menor uso da CPU.

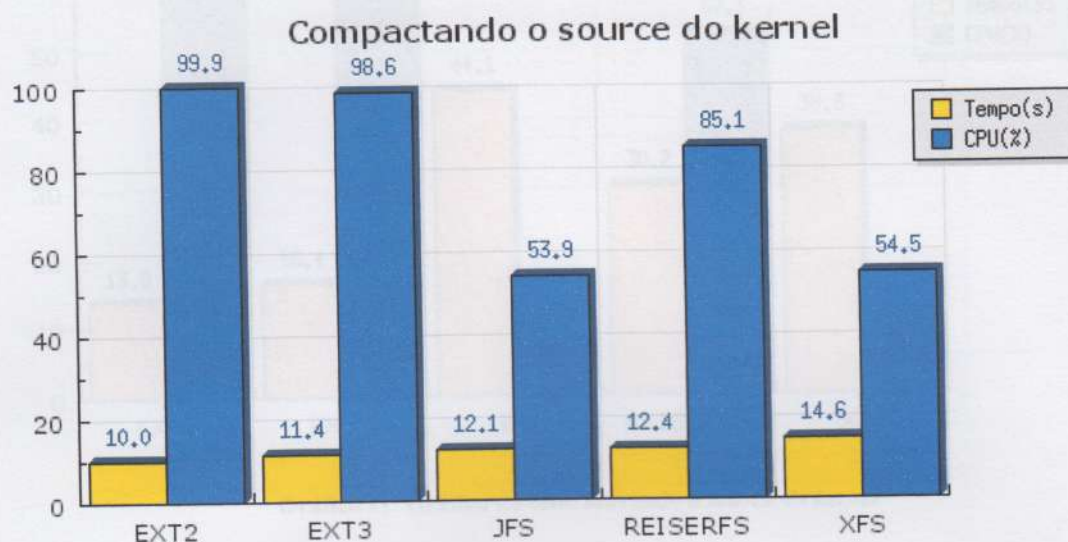


Gráfico 9: Gráfico do teste compactando o *source* do *kernel*

No teste do Gráfico 10, novamente o Ext2 e o Ext3 foram mais rápidos e com os maiores uso da CPU. Desta vez o JFS foi o muito lento, porem usou menos de 5% da CPU.

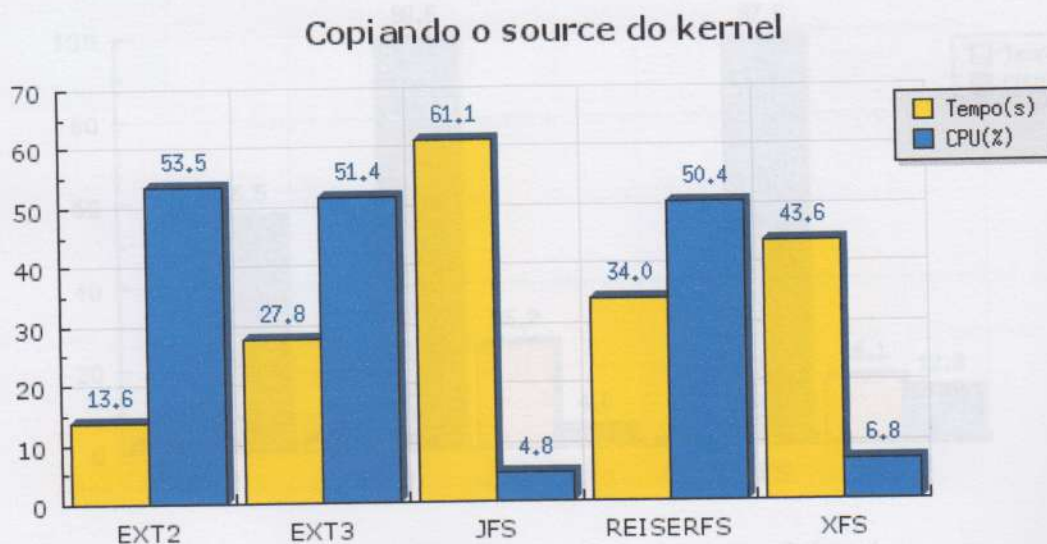


Gráfico 10: : Gráfico do teste copiando o *source* do *kernel*

O resultado do teste do Gráfico 11 é semelhante aos três testes anteriores.

Movendo o source do kernel



Gráfico 11: Gráfico do teste movendo o source do kernel

Por fim, o teste do Gráfico 12. Ficou claro que ao se trabalhar com muitos arquivos e diretórios o Ext3 liderou com certa vantagem, sendo o sistema de arquivos com *journaling* de melhor desempenho.

Removendo o source do kernel

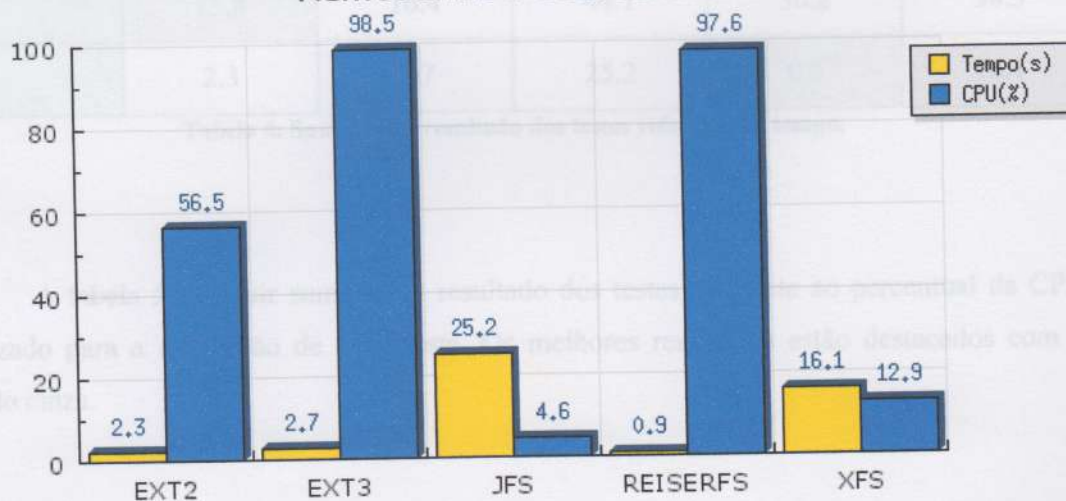


Gráfico 12: Gráfico do teste removendo o source do kernel

A tabela 4 a seguir sumariza o resultado dos testes referente ao tempo gasto em segundos para a realização de cada teste. Os melhores resultados estão destacados com o fundo cinza.

Teste	EXT2	EXT3	JFS	REISERFS	XFS
01	2.4	6.3	6.2	0.4	11.4
02	2.5	5.1	5.9	0.6	11.7
03	1.2	2.5	14.1	0.4	9.5
04	28.2	34.1	31.2	31.2	32.6
05	34.5	36.6	35.3	34.3	38.9
06	77.5	84.1	78.8	81.6	81.9
07	2.0	3.3	3.4	0.4	5.7
08	9.1	14.0	63.5	21.9	46.1
09	10.0	11.4	12.1	12.4	14.6
10	13.6	27.8	61.1	34.0	43.6
11	13.8	16.4	44.1	30.2	38.3
12	2.3	2.7	25.2	0.9	16.1

Tabela 4: Sumário do resultado dos testes referente ao tempo.

A tabela 5 a seguir sumariza o resultado dos testes referente ao percentual da CPU utilizado para a realização de cada teste. Os melhores resultados estão destacados com o fundo cinza.

Teste	EXT2	EXT3	JFS	REISERFS	XFS
01	99.9	97.9	8.4	97.3	7.8
02	99.3	96.6	11.0	97.6	10.1
03	97.7	98.6	4.6	97.3	15.0
04	53.6	55.8	7.9	55.3	9.5
05	54.8	53.8	9.0	53.0	9.9
06	51.4	50.9	5.6	49.5	6.2
07	51.5	53.0	28.9	79.8	52.8
08	62.8	59.5	5.8	56.1	8.2
09	99.9	98.6	53.9	85.1	54.5
10	53.5	51.4	4.8	50.4	6.8
11	58.2	57.5	6.8	52.3	11.2
12	56.5	98.5	4.6	97.6	12.9

Tabela 5: Sumário do resultado dos testes referente ao consumo da CPU.

5 CONCLUSÃO

Ao final de todos os testes, ficou clara a divisão dos sistemas de arquivos em 3 grupos, o primeiro ao se trabalhar com arquivos pequenos, o segundo com arquivos grandes e o terceiro com muitos arquivos e diretórios. O ReiserFS se encaixou no primeiro grupo, tendo um bom desempenho no geral e se destacando nos três primeiros testes com arquivos pequenos, sendo extremamente rápido nestes testes. Já ao trabalhar com arquivos grandes, que caracteriza o segundo grupo, o XFS e o JFS foram os melhores, destacando-se pelo pouco tempo gasto para a realização das tarefas e com um baixíssimo uso da CPU. Nos últimos testes, ao se trabalhar com um grande número de arquivos e diretórios, que constitui o código fonte do *kernel*, (teste 8 ao teste 12) o Ext3 foi o melhor sem dúvida, ficando no terceiro grupo.

Sendo assim, baseado no estudo feito e no resultado dos testes, conclui-se que o ReiserFS tem um bom desempenho e é uma boa escolha para ser usado em um computador pessoal ou em servidores que possuam muitos arquivos de pequeno tamanho e onde a economia de espaço seja fundamental.

O Ext3 também poderia ser uma boa escolha para um computador pessoal e por ser padrão na maioria das distribuições Linux. Entretanto, o Ext3 possui uma limitação um tanto grave, que é o tamanho máximo de um arquivo ser de 2GB. No entanto por ser um sistema de arquivos bem maduro, estável, e de desempenho satisfatório, seu uso é mais comum em servidores em geral.

O JFS e o XFS são sistemas de arquivos desenvolvidos para sistemas de alta performance, capazes de manipular grandes arquivos, sendo mais indicado para grandes servidores, rodando muitos processos, como um servidor de banco de dados com uma grande base de dados, por exemplo.

REFERENCIAL BIBLIOGRÁFICO

- ALECRIM, Emerson – Sistema de arquivos ext3. Disponível em: <http://www.infowester.com/linext3.php>. Acesso em 08/05/2006.
- BEST, Steve, - JFS overview. Disponível em: <ftp://www6.software.ibm.com/software/developer/library/jfs.pdf>. Acesso em 12/05/2006.
- NOGUEIRA, Roger Fonseca, UFPEL. Acesso em 20/05/2006.
- Silicon Graphics, Inc – XFS. Disponível em: <http://oss.sgi.com/projects/xfs/index.html>. Acesso em 12/06/2006.
- WIKIPEDIA – Journaling. Disponível em: http://en.wikipedia.org/wiki/Journaling_file_system. Acesso em 18/04/2006.
- WIKIPÉDIA – LINUX. Disponível em: <http://pt.wikipedia.org/wiki/Linux>. Acesso em 16/04/2006.
- WIKIPEDIA – Sistema de ficheiros. Disponível em: http://pt.wikipedia.org/wiki/Sistema_de_arquivos. Acesso em 15/04/2006.
- WIKIPÉDIA – XFS. Disponível em: <http://en.wikipedia.org/wiki/XFS>. Acesso em 12/06/2006.