Curso de Engenharia de Computação

SISTEMAS DE ARQUIVO:
ANÁLISE DE DESEMPENHO

THÉO RODRIGUES DE ALMEIDA

Itatiba – São Paulo – Brasil

Dezembro de 2009

Curso de Engenharia da Computação

SISTEMAS DE ARQUIVO:
ANÁLISE DE DESEMPENHO

THÉO RODRIGUES DE ALMEIDA

Monografia apresentada à disciplina de Conclusão de Curso, do Curso de Engenharia da Computação da Universidade São Francisco, sob a orientação do Dr. Alencar de Melo Júnior, como exigência parcial para conclusão do curso de graduação.

Orientador: Prof. Dr. Alencar de Melo Júnior

Itatiba – São Paulo – Brasil

Dezembro de 2009

Sistemas de arquivo
Análise de desempenho

Théo Rodrigues de Almeida

Monografia defendida e aprovada em 11 de Dezembro de 2009 pela **Banca Examinadora** assim constituída:

Prof. Dr. Alencar de Melo Júnior **(Orientador)**

USF – Universidade São Francisco – Itatiba – SP.

Prof. Ms. Cláudio Maximiliano Zaina

USF – Universidade São Francisco – Itatiba – SP.

Profª. Ms. Silvia E. S. Lopes

USF – Universidade São Francisco – Itatiba – SP.

Sonhos não morrem, apenas adormecem na alma da gente.

(Chico Xavier)

A minha avó Maria de Lourdes Almeida, por ser minha segunda mãe. Por ter acreditado e sempre me ajudar todo este tempo.

Agradecimentos

Agradeço a Deus acima de tudo por me dar a sabedoria e oportunidade de estudar. Por me dar o dom da vida e sempre me mostrar o caminho correto.

Ao Prof. Sidney Pio de Campos por me incentivar com o tema abordado nesta monografia, pelas aulas maravilhosas e dinâmicas que me ajudaram a ser um bom profissional.

Ao Prof. Alencar de Melo Júnior, por ser meu orientador e me passar muitos ensinamentos, por me ajudar nos desafios encontrados durante este projeto e por suas aulas que foram de grande valia.

A minha namorada Anelise Dias, que me ajudou nos momentos difíceis sempre com muito amor e compreensão.

Ao meu grande amigo Tob que se tornou uma grande companhia nas noites de estudo.

Por fim, agradeço a todos os meus amigos que sempre estiveram ao meu lado.

Sumário

Lista de Siglas

EB	*Exa Byte*
EFS	*Extent File System*
EXT	*Extend*
EXT2	*Extend 2*
EXT3	*Third Extended File System*
EXT4	*Extend 4*
FAT	*File Allocation Table*
FFS	*Fast File System*
FSCK	*File System Check*
Gb	*Giga Bit*
GB	*Giga Byte*
HPFS	*High Performance File System*
JDB	*Journaling Block Device*
JFS	*Journaling File System*
Kb	*Kilo bit*
KB	*Kilo Byte*
Kernel	*Núcleo do Sistema Operacional*
NTFS	*New Tecnology File System*
PB	*Peta Byte*
REISERFS	*Reiser File System*
SA	*Sistema de Arquivo*
SAS	*Sistemas de Arquivos*
SO	*Sistema Operacional*
SO	*Sistemas Operacionais*
TB	*Tera Byte*
XFS	*X File System*
XFS	*X File System*

Lista de Figuras

Lista de Tabelas

Resumo

O trabalho objetiva comparar alguns dos sistemas de arquivos mais comuns no mercado. Uma vez que existem diversos sistemas de arquivos em amplo uso comercial a decisão por qual sistema deve ser instalado em determinado servidor ou até mesmo em computadores pessoais deve ser baseada nos tipos de aplicação que serão implantadas. Esta decisão deve ter como principais critérios a segurança e velocidade da aplicação que fará uso deste computador. Com isso, surge à necessidade de uma análise profunda de como cada sistema de arquivo se comporta para um determinado tipo de aplicação. O software utilizado para esta análise foi o Benchmark Bonnie++, que tem como objetivo coletar o tempo de escrita por caractere, tempo de escrita por bloco, reescrita por bloco, leitura por caractere, leitura por bloco, busca aleatória, assim como o uso de processamento decorrente em cada análise desenvolvida. O trabalho analisa qual sistema de arquivo terá melhor desempenho considerando um banco de dados de determinado tamanho, um servidor de armazenamento ou um servidor de processamento. Com isso tem-se um método objetivo para a escolha do sistema de arquivo a ser instalado.

Palavras-chave: Sistema de arquivo; análise de desempenho; Bonnie++.

Abstract

The work aims at comparing some of the most common file systems at the market. Once there are several file systems being widely commercially used, the decision on which system to be installed at a certain server or even personal computers must be based upon the application types that will be implanted. This decision must include main criteria, such as security and application speed that will use this computer. Thus, a deep analysis must be performed on how each file system works with a certain application type. The software that was used to this analysis was BenchMark Bonnie++, which aims at collecting the writing time by character, block, rewriting by block, reading by character, reading by block, random search, as well as the due processing use at each developed analysis. The work analyzes which file system will have the best performance considering a certain data bank size, storage server or a processing server. Thereby there is an objective method in order to choose the file system to be installed.

KEY WORDS: System File; Performance Analysis, Bonnie++

1 INTRODUÇÃO

De um modo simples pode-se dizer que um sistema de arquivo (SA) é responsável pelo modo como os dados serão armazenados e manipulados. O SA é responsável pela segurança, velocidade e capacidade de armazenamento em um dispositivo computacional.

Todos os sistemas de arquivos (SAS) utilizam uma hierarquia com diretórios e arquivos, sendo que esta hierarquia garante uma forma organizada de gravar estes arquivos. Com a evolução dos computadores e softwares, veio a necessidade de se criar e aprimorar novos SAS, pois com os mais variados tipos de aplicações em produção, tem-se uma demanda por determinado tipo de armazenamento e processamento. Devido a esta necessidade será feita uma análise detalhada de como se comporta cada SA para uma determinada aplicação.

Existem muitos estudos feitos sobre cada SA existente, onde a maioria ocorre pela própria empresa criadora do SA. Porém não há muitas publicações que mostram a comparação entre eles.

Os principais SAS serão abordados e também os tipos de armazenamentos mais comuns em cada um dos SAS analisados, assim como suas principais características.

Os SAS que foram comparados são: EXT2, EXT3, EXT4, REISERFS, JFS, NTFS e XFS visto que são todos SAS padrões na maioria das instalações Linux. Foram analisados os seguintes indicadores de desempenho: escrita por caractere, escrita por bloco, reescrita, leitura por caractere, leitura por bloco, procura aleatória, tal como o uso da unidade central de processamento (CPU) em cada um dos dados coletados.

O resultado final deste estudo teve como objetivo fornecer uma melhor comparação entre os SAS existentes, e com isso decidir qual SA tornará a aplicação que se deseja implantar mais rápida e segura.

A próxima seção aborda os principais tipos de armazenamento em SA, a saber, alocação contínua, lista encadeada e lista por alocação indexada, assim como a definição do principal recurso de segurança em SA.

A seção 3 aborda as principais características em cada SA analisado. Estas características são, história de surgimento, vantagens e desvantagens.

A seção 4 teve como objetivo mostrar as análises dos testes realizados sobre os SAS, assim como os gráficos correspondentes sobre cada teste executado. Finalmente a seção 5 traz a conclusão dos resultados obtidos com todos os testes realizados.

2 TIPOS DE ARMAZENAMENTO EM SISTEMAS DE ARQUIVO

Basicamente pode-se dividir os sistemas de arquivos em três principais tipos de estruturas: alocação continua, alocação por lista encadeada e alocação por lista indexada. Estas estruturas demonstram como se comportam os arquivos em um determinado disco físico.

No caso da alocação contínua os dados são armazenados de forma contínua. Por exemplo, quando se quer armazenar dois arquivos, o primeiro arquivo é armazenado em um espaço continuo no disco, o armazenamento apenas ocorre se o espaço for suficientemente grande para comportar todo o arquivo.

A maior vantagem neste tipo de alocação é a fácil implementação e uma maior otimização de desempenho, uma vez que os arquivos estariam em sequência na memória (como mostra a Figura 1).

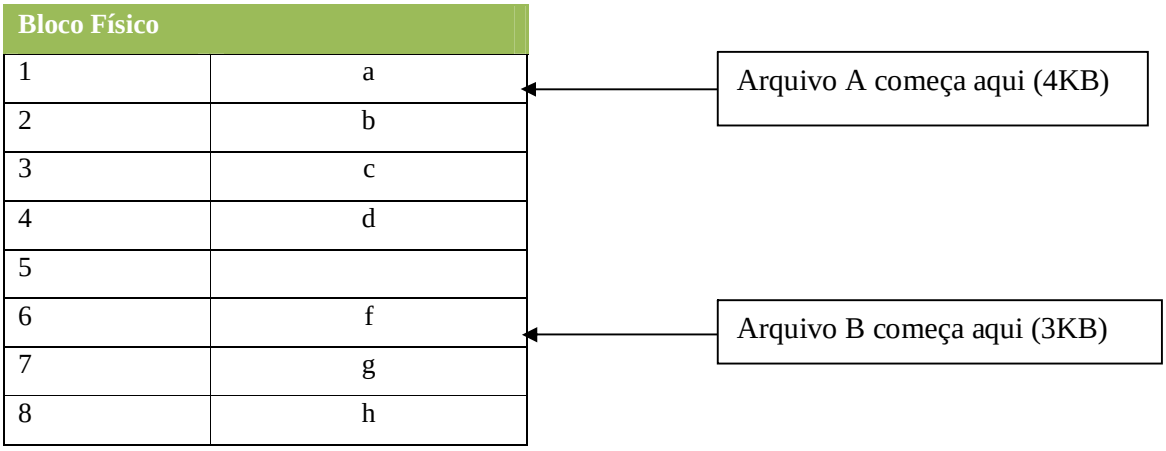

Figura 1 – Alocação continua[1]

Com esta alocação ocorre uma grande perda de espaço, uma vez que se tem um determinado tamanho de blocos, os arquivos não utilizam os espaços que ficam vagos. [1]

No caso da alocação por lista encadeada, os blocos de memória não têm apenas como informação o arquivo, mas também um endereço, onde é armazenado o local da memória onde está o próximo bloco do arquivo (conforme a Figura 2).

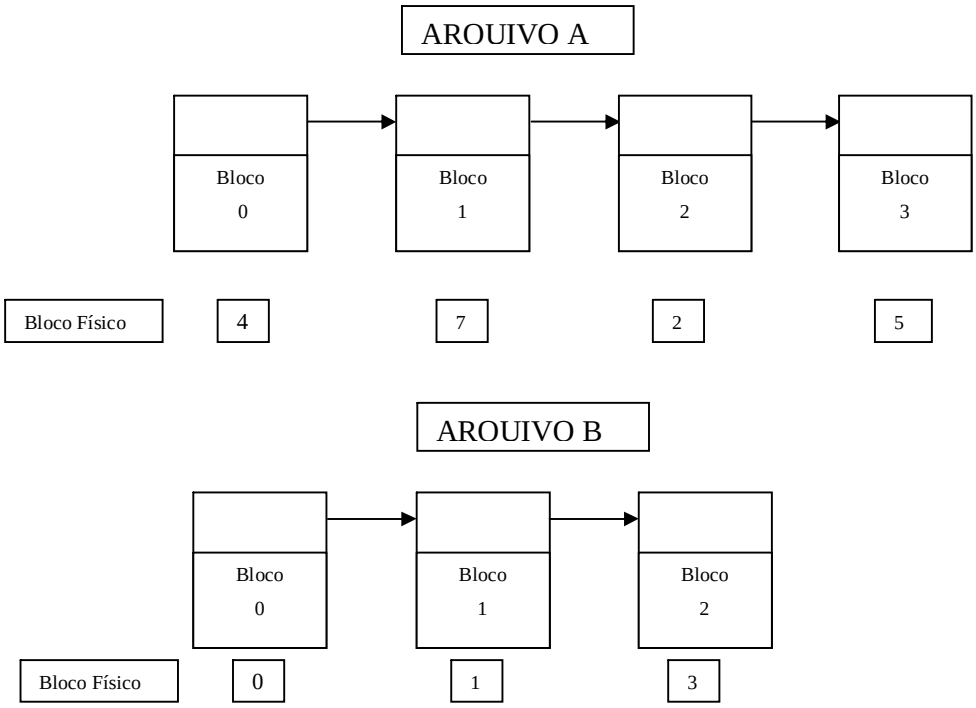

Figura 2 – Alocação lista encadeada [1]

Deste modo não ocorre à perda de espaço, pois qualquer bloco não ocupado pode conter uma parte do arquivo a ser gravado.

Este sistema de alocação promove uma grande fragmentação no disco, dado que um mesmo arquivo pode-se encontrar em diferentes partes do disco, gerando assim perda de velocidade. [1]

A maioria dos SAS atuais utiliza alocação indexada. Esta técnica consiste em manter uma lista dos principais blocos de arquivos. Desta maneira, o bloco principal aponta para os demais blocos que contém a continuação do arquivo (conforme a Figura 3). O SA mantém uma lista atualizada de todos os blocos principais. [1]

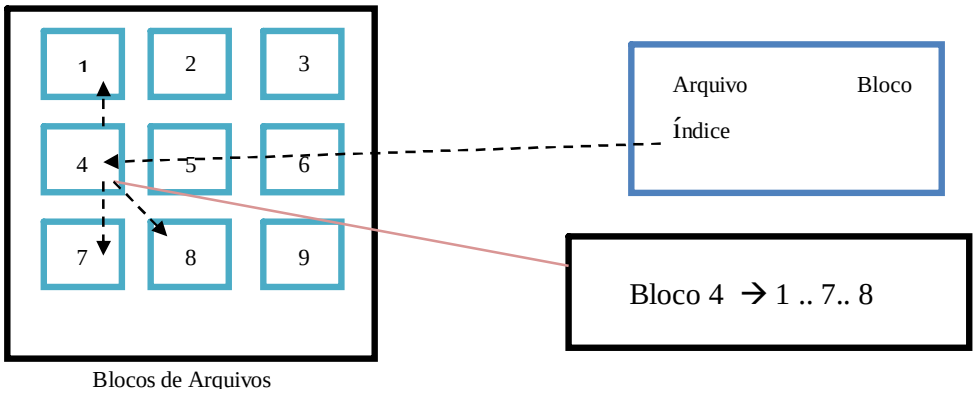

Figura 3 – Alocação Indexada [1]

2.1 Segurança nos sistemas de arquivos utilizando *journaling*

Um dos maiores problemas nos SAS foi encontrar um modo eficiente de prover uma forma segura para recuperação de erros, uma vez que o desligamento abrupto de um sistema pode ocorrer por falta de energia ou até mesmo por descuido do usuário. Uma maneira eficiente de se prover esta recuperação é fazendo uso de *journaling*. Este mecanismo tem a capacidade de acompanhar as mudanças que são implementadas no SA, como por exemplo, gravações e atualizações de dados, antes que as mesmas realmente sejam feitas. Estas informações ficam em uma parte separada do SA. Após as atualizações serem realizadas com sucesso, há a remoção dos dados do *journaling*. Caso ocorra algum problema no processo destas, as informações estão gravadas no *journaling*, e assim que o sistema iniciar novamente há a verificação deste *journaling*. Uma vez que existam mudanças a serem feitas, as mesmas serão aplicadas. Isto evita que possa ocorrer corrompimento de arquivos e até mesma perda de dados. [2]

3 PRINCIPAIS SISTEMAS DE ARQUIVOS

Nesta seção abordar-se-á os principais SAS existentes no mercado, considerando suas características, história de surgimento, pontos fortes e pontos fracos.

3.1 *New technology file system* (NTFS)

O SA NTFS foi inicialmente projetado para versões do Windows em servidores. Seu principal objetivo era produzir um SA flexível, altamente seguro e confiável. A Microsoft resolveu desenvolver o Windows NT para competir no mercado com o Unix, que fez surgir a necessidade de se criar um novo SA, já que o antigo SA *File Allocation Table* (FAT) era cheio de falhas, tais como o tamanho do arquivo que se, superior a 512 MB apresentava problemas e limite de partições com no máximo 5GB. A Microsoft criou assim o sistema NTFS baseado nos conceitos funcionais do SA *High Performance File System* (HPFS). [3] [4]

3.1.1 Características

O NTFS trabalha de forma eficiente no gerenciamento do espaço de disco. De modo que o SA NTFS não usa *clusters* de múltiplos setores e sim armazenamento em uma base por setor de disco. Portanto, há uma menor fragmentação de disco onde o sistema não precise trabalhar com um número pré-determinado de setores. [4]

3.1.2 Controle de falhas

O NTFS implementa um controle transacional. Este sistema funciona de modo que, quando uma alteração é realizada no SA, esta ação ou ocorre em sua totalidade ou não. Quando o sistema operacional (SO) demanda por uma determinada ação no SA, é utilizado um arquivo de metadados chamado $LogFile onde são gravados os componentes desta

operação. Uma vez que toda operação é executada, é gravado no *log*, indicando que tudo ocorreu corretamente. Caso isso não ocorra, a próxima vez que o arquivo de *log* for acessado ele continuará a realizar a operação que não foi finalizada anteriormente. Isso faz com que haja a garantia de correção das falhas em caso de interrupção abrupta do sistema. O problema deste tipo de controle de erros é que o acesso a este arquivo de *log* a cada mudança no sistema demanda um determinado tempo. Para amenizar este problema, o sistema operacional grava este log em *cache* por ser uma memória mais rápida, porém isto faz com que o sistema não fique totalmente garantido de falhas. A solução foi, fazer com que um processo secundário de menor prioridade grave estes dados do *cache* no $LogFile de tempos em tempos, fazendo diminuir em muito a possibilidade de algum erro. Tal erro só aconteceria se estivesse nesta janela de tempo, que atualmente é de 8 segundos.

Como em vários outros SAS, existem versões do NTFS. Atualmente a versão mais recente é a 5.0, que foi inicialmente conhecida com o lançamento do Windows 2000. [4]

3.2 *Second extended file* (EXT2)

Em 1992 iniciou-se a utilização de um SA que é hoje um dos principais do mercado, sendo este, o *extends* (EXT). O primeiro EXT foi introduzido no núcleo do SO (*kernel*) em sua versão 0.96c. No qual na primeira versão, as partições podiam ter um tamanho máximo de 2GB e suportavam apenas arquivos com nomes de no máximo 255 caracteres de tamanho.

O SA EXT2 foi desenvolvido por Rémy Card e Stephen Tweedie em 1995 com o intuito de corrigir os problemas encontrados no EXT. [5]

3.2.1 Vantagens

O EXT2 tornou possível definir na formatação do sistema o tamanho de cada setor de alocação, tendo como o mínimo um tamanho de 512 bytes e um maximo de 4.096 bytes. Desta forma, o EXT2 permite o endereçamento por blocos, onde cada bloco tem o tamanho de 4KB. Há também um super bloco, que é a estrutura básica do SA, sendo que este super bloco é responsável por armazenar a localização dos dados do sistema, quantidade de blocos e estado do SA. O SA EXT2 suporta partições de até 4TB, porém com os mesmo 255 caracteres para nomes de arquivos. [5]

3.2.2 Desvantagens

O EXT2 não implementa *journaling*, assim, quando o sistema é desligado abruptamente pode ocorrer corrompimento de dados, uma vez que eles não são checados quando se inicia novamente o sistema. [5]

3.3 *Third extended file system* (EXT3)

Em 1999 o novo sistema de arquivos EXT3 foi incorporado ao Linux, sua principal mudança foi à introdução de um recurso utilizado em outros SAS, chamado *journaling*. Até os dias de hoje este é um dos SAS mais utilizados nos sistemas operacionais (SOS) Linux. [6]

3.3.1 Características

O SA EXT3 garante a integridade dos dados caso ocorra uma falha na energia ou até mesmo um desligamento acidental. Seu criador Stephen Tweedie, juntamente com outros desenvolvedores na Red Hat, desenvolveram um SA onde o código de *journaling* usa um *Journaling Block Device* (JBD) que é uma camada que implementa o *journal* em qualquer tipo de dispositivo. Seu formato padrão utiliza blocos de dados, sendo assim, o JDB é quem gerencia as transações de disco. Este recurso é muito utilizado uma vez que possamos ter diversos SAS utilizando o JDB, independendo do SA utilizado. O JDB não utiliza a gravação de dados byte a byte, mas sim, grava os próprios blocos modificados de arquivos, e o SA EXT3 também faz a gravação de seus blocos, assim é possível comparar quais foram as informações que ficaram pendentes. O EXT3 suporta 3 tipos de *journaling*: [7]

- *Journal:* Maior garantia contra falhas, utiliza a gravação das mudanças no SA. Por escrever diretamente no SA é o modo mais lento de *journaling*.
- *Ordered*: Padrão do SA EXT3, as mudanças são gravadas em arquivos metadados (arquivos que guardam dados sobre dados), onde os dados são gravados no SA no qual foi instalado o SO.
- *Writeback*: Grava as mudanças em arquivos metadados, porém é utilizado o modelo do *journaling* presente no SA no qual é realizada a operação no arquivo. 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 se deseja usar, por meio de uma mudança do arquivo *fstab*. Por exemplo, pode ser que a linha /dev/hda1/opt tenha sua opção *data* com o valor *ordered*. Este valor pode ser modificado para *writeback* ou *journal*. [7]

3.3.2 Vantagens

A maior vantagem do EXT3 é que a partir do EXT2 sua atualização se torna simples, uma vez que não é necessário fazer *backup* dos arquivos e depois fazer a restauração. Os metadados do SA estão em locais fixos, e há redundância inerente à estrutura de dados, o que faz com que a recuperação de dados seja possível caso houver uma corrupção de arquivos.

Outra característica importante é que o EXT3 suporta grandes tamanhos de arquivos. A Tabela 1 mostra o tamanho do bloco de dados com seu respectivo tamanho suportado de arquivo e o tamanho máximo do SA suportado. [8]

Tamanho do Bloco	Tamanho Máximo Arquivo	Tamanho Máximo SA
1 KB	16 GB	2 TB
2 KB	256 GB	8 TB
4 KB	2TB	16TB
8 KB	2TB	32TB

Tabela 1 – Tamanho de alocação sistema EXT3[8]

3.3.3 Desvantagem

No SA EXT3 não há alocação dinâmica de *inode*, ou seja, ocorre uma fragmentação interna, uma vez que o tamanho dos blocos é fixo, sendo atribuído seu tamanho no momento da instalação, pois sua estrutura é muito similar as do EXT2. Outro problema é que o SA não pode ser checado enquanto é montado para a escrita, o que resulta em um *dump* que é quando ocorrem erros na inicialização do sistema. [8]

3.4 *Four extends file system* (EXT4)

Com as constantes melhorias que foram sendo desenvolvidas no EXT3, houve também alguns problemas que surgiram durante este processo de evolução. Em certo ponto do projeto viu-se a necessidade de fazer uma divisão, ou seja, pegar tudo o que foi feito anteriormente, copiar e mudar aquele código, preservando o SA EXT3. Com isso a partir do *kernel* 2.6.19, o SA EXT4 em sua versão alfa já era suportado. Porém como possuía alguns problemas de incompatibilidade, não era muito aceito no mercado; isto veio a mudar com a versão 2.6.24.4 do *kernel*, onde se passou a ter suporte integral ao novo SA. [9]

3.4.1 Vantagens

Algumas melhorias foram implementadas no EXT4, sendo que a mais visível foi no tamanho de arquivos que podem ser escritos que passou de 2 TB para 1024 PB e sua partição máxima passou de 32TB para 16EB. Essas mudanças são mais visíveis para grandes servidores.

Houve melhorias na pré-alocação em que, na versão EXT3 quando um programa fosse utilizar um espaço físico em um dispositivo de armazenamento, ocorria uma pré-alocação, no qual os *inodes* que caso seria utilizados eram preenchidos de zeros. Em aplicações que esta execução ocorria muitas vezes, isto causava um grande atraso de tempo. Já no EXT4 isto não ocorre, pois não há mais a necessidade de se fazer esta pré-alocação.

Outra melhoria é que não existem mais limites para o número de pastas, este número era limitado a 32000 no EXT3.

No EXT4 a desfragmentação ocorre em tempo de execução, de modo que os arquivos são desfragmentados enquanto são alocados em disco.

A fim de que não se possam apagar arquivos essenciais para o funcionamento do SO, no EXT4 há uma ferramenta que impede que isso ocorra, mesmo tendo permissão máxima do SO para manipulação de arquivos.

É criada também uma validação do *journaling* onde há uma maior chance de não haver falhas na restauração de arquivos corrompidos. [10]

3.4.2 Desvantagens

Como o sistema de arquivos EXT4 ainda está em desenvolvimento não é possível afirmar com certeza o que será uma vantagem e uma desvantagem. Mais existem grandes expectativas sobre suas novas implementações. [10]

3.5 *Reiser file system* (REISERFS)

Em meados de 2001 surgiu o SA denominado REISERFS, sendo que seu nome é originado de seu principal criador Hans Reiser, e também fundador da equipe NAMESYS, que ficou responsável por gerenciar o projeto. Muitas versões do Linux vêm como padrão este SA em sua instalação. [11]

3.5.1 Vantagens

Entre os principais recursos do REISERFS pode-se citar:

- Uso de *journaling* para a prevenção de perda de dados e integridade do sistema. No caso do REISERFS, o *journaling* fica limitado aos arquivos metadados;
- suporte a arquivos de até 1 EB;
- utiliza alocação dinâmica de *inodes*, fazendo com que não haja perda de espaço em disco. Com isso, arquivos grandes podem ser armazenados rapidamente e sem desperdício de espaço. [12]

3.5.2 Desvantagens

Por utilizar alocação dinâmica de *inodes* para armazenamento, o SA REISERFS se torna mais lento e acaba utilizando muita CPU caso necessite acessar ou gravar muitos arquivos pequenos.

Seu criador, Hans Reiser encontra-se preso desde 10 de outubro de 2006 e com isso houve a interrupção de seu desenvolvimento na Namesys. Com isso é impossível determinar como se dará sua evolução, sendo que Hans Reiser é proprietário da Namesys, empresa esta que se encontra fechada. [12]

3.6 *X file system* (XFS)

Desenvolvido pela Silicon Graphics, inicialmente para o sistema operacional IRIX, o SA XFS foi desenvolvido a partir do antigo sistema EFS, que por sua vez tinha seu desenvolvimento baseado no antigo FFS. Seu surgimento se deu no ano de 1994 na versão 5.3 IRIX. Foi mostrado em 1999 um modo de se adaptar o sistema XFS para o Linux porém, somente em maio de 2001 foi lançada a primeira versão do XFS para Linux. Esta versão inicial só era possível de ser incorporada ao Linux por meio de *patches.* Somente em 2004 o SA XFS foi incorporado ao kernel 2.4. [13]

3.6.1 Vantagens

Sua estrutura foi criada para a arquitetura de 64 bits, porém é completamente compatível com arquiteturas de 32 bits. O tamanho máximo de armazenamento em arquiteturas de 64 bits é de 18 EB e o tamanho máximo dos arquivos é de 8 EB.

O tamanho dos blocos é ajustável, podendo variar entre 512 bytes a 64KB. Isto torna o armazenamento de arquivos grandes como pequenos extremamente eficientes.

O XFS utiliza uma técnica diferente para a alocação de arquivos, uma vez que o arquivo é escrito em *buffer* ao invés de alocar espaço de dados. Assim o XFS reserva o número correto de blocos livres que devem ser utilizados para armazenar os dados e somente ocorre à escrita quando muitos dados são enviados para o disco. Isso diminui a fragmentação e melhora o desempenho de escrita e leitura no disco.

O XFS utiliza *journaling* para a recuperação de dados, uma vez que ocorra qualquer falha no SO. [14]

3.6.2 Desvantagens

A principal desvantagem do XFS é o consumo excessivo de recursos quando ocorre uma demanda muito grande de utilização de arquivos pequenos, uma vez que o *buffer* é utilizado a cada escrita de arquivos antes que o mesmo seja gravado em disco. [14]

3.7 *Journaling file system* (JFS)

O JFS é um SA desenvolvido pela IBM, que utiliza licença *open-source*. Seu desenvolvimento ocorreu com o intuito de ser utilizado no Unix dos servidores da IBM.

O JFS foi criado inicialmente para servidores de grande porte. Seu criador Steve Best criou um SA que visava à alta taxa de transferência e processamento que os servidores necessitavam. Com o rápido desenvolvimento de hardwares para computadores pessoais, o uso do JFS se tornou comum para usuários domésticos. [15]

3.7.1 Vantagens

O JFS utiliza *journaling* para prevenir possíveis erros em caso de falha na energia ou desligamento abrupto do sistema. No JFS é possível fazer um redimensionamento nas partições do sistema sem que haja a necessidade de reiniciar o SO. [15]

3.7.2 Desvantagens

O JFS guarda todos seus *logs* em arquivos metadados, em que há a necessidade de recuperar os dados de E/S em outros dispositivos, é necessário uma verificação completa do mesmo, a fim de verificar se o erro é decorrente de um problema no sistema de armazenamento ou do SA, isso faz com que haja uma utilização excessiva de recursos do sistema e conseqüentemente uma grande perda de tempo, isto se torna muitas vezes ineficaz uma vez que, se o problema é caracterizado por falha na mídia de gravação. Este erro foi corrigido a partir do SA JFS2. [15]

4 Análise comparativa

Esta seção apresenta a análise de desempenho para os seguintes SAS: NTFS, EXT2, EXT3, EXT4, REISERFS, JFS e XFS.

4.1 Hardware

O hardware utilizado para os testes foi um computador com CPU core 2 duo 3.0 GHz, com 4 GB de memória a 800 MHz, e HD SATA 2500 Gb.

4.2 Software

Para a análise de desempenho foi instalado o SO Linux Ubuntu 9.04, no qual foi instalado o mínimo possível de softwares que vem com a distribuição de tal SO. Foram criadas 8 partições, onde o SO ficou instalado em uma partição EXT3, uma vez que o EXT3 utiliza JBD, e seu modo padrão de *journaling* é *ordered*. Sendo necessário modificar o *journaling* para o modo *writeback*, uma vez que, este modo utiliza o *journaling* do SA que foi utilizado no momento.

O teste efetuado na partição EXT3 foi realizado em uma partição separada da partição EXT3 que contem o SO, desta forma o teste não foi prejudicado. Para todas as partições foram criados blocos de 4 KB para todos os SA analisados.

Para uma maior exatidão dos dados analisados foram feitas mudanças na localização dos SAS, uma vez que a distância do SA no disco não demonstrou diferença significativa, desta forma, as análises finais não foram comprometidas.

Na análise de desempenho dos SAS foi utilizado o software bonnie++ [16]. A escolha do software deu-se por ser muito utilizado no seguimento de análise de desempenho de SA, sendo que em alguns artigos da revista Linux Magazine [17] [18] os testes com este software podem ser visualizados. Outra característica importante deste software é que há a possibilidade de modificar a configuração da análise a ser desenvolvida.

4.2.1 Bonnie ++

A versão instalada do bonnie++ foi a 1.03e. Uma grande qualidade do bonnie++ é a possibilidade de se configurar todos os testes, determinando o tamanho do arquivo a ser escrito, a memória RAM a ser utilizada para esta operação e o número de vezes que cada um destes testes que serão realizados.

Nos testes foram utilizados arquivos com o tamanho de 2GB, a memória RAM utilizada foi de 1GB. Para a maior eficácia das análises, foram realizados seis testes consecutivos para cada SA. Se algum programa executou em um determinado momento sua *thread*, nos testes seguintes a diferença apareceria nos seus respectivos valores gerados.

No momento de cada teste os softwares que estavam sendo executados eram os mesmos pra cada SA e os dois núcleos do processador também estavam sendo utilizados.

A utilização do bonnie++ ocorre de maneira simples, por meio de linha de comando.

bonnie++ -d /local/ -s X -r Y -x Z –u | bon_csv2html > /bonnie/nome.html

onde:
-d local onde se encontra o SA instalado
-s tamanho do arquivo a ser escrito
-r memória RAM utilizada no teste
-x número de testes a serem realizados
-u usuário *root*
/diretório de saída dos resultados/nome do arquivo a ser gerado com os dados.html

Com os resultados gerados pelos seis testes consecutivos foi realizada uma média aritmética para que a margem de erro fosse a menor possível, uma vez que o SO consumia alguns recursos. Os dados gerados foram mostrados por meio de gráficos para uma melhor compreensão dos resultados.

4.3 Escrita por caractere

Este teste tem como objetivo gravar um arquivo bit-a-bit, fornecendo como resultado a taxa média em Kb/s da escrita por caractere. Esta análise mostra como se comporta a escrita de arquivos pequenos em disco. A visualização dos dados pode ser observada na Figura 4.

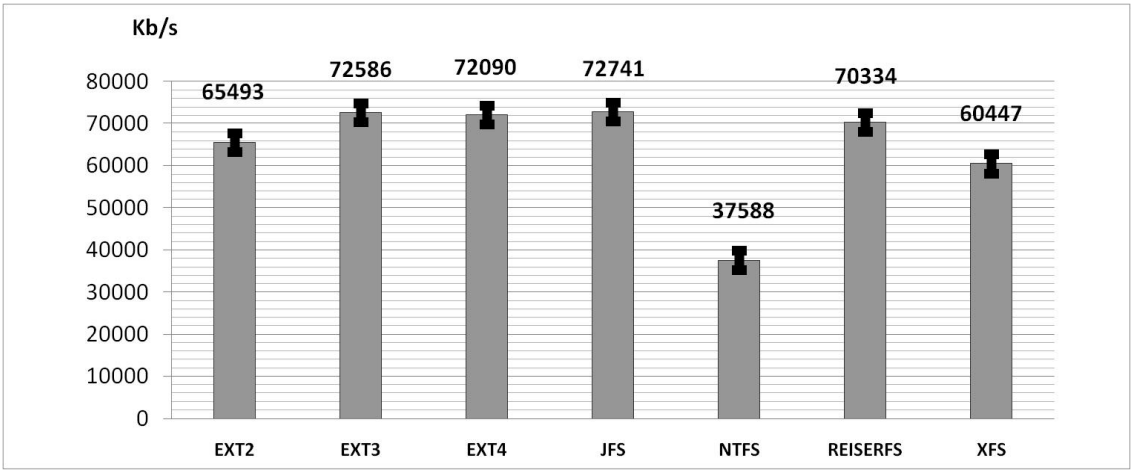

Figura 4 – Escrita por caractere

Com base nos dados mostrados pela Figura 4, observa-se que todos os SAS tiveram desempenho similar, exceto o SA NTFS que foi o mais lento.

4.3.1 Uso de CPU na escrita por caractere

O uso de CPU no decorrer deste teste é descrito pela média da porcentagem gasta de processamento para a escrita por caractere. Os dados podem ser visualizados na Figura 5.

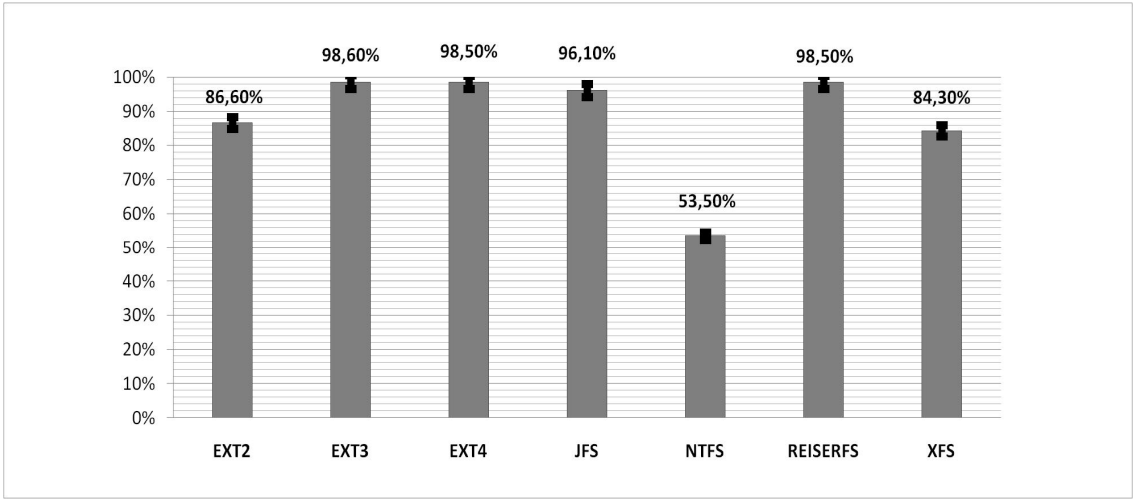

Figura 5 – Uso de CPU escrita por caractere

As maiores taxas de uso de CPU foram nos SAS EXT3, EXT4, JFS, REISERFS. A menor taxa foi no SA NTFS, porém foi o que obteve o pior desempenho na escrita por caractere (Figura 5).

4.4 Escrita por bloco

Neste teste é utilizada a escrita por bloco, onde todos os SAS têm o mesmo tamanho de bloco, que é de 4KB. Este teste visa mostrar como se comporta a escrita de arquivos grandes em disco, uma vez que arquivos menores de 4KB não utilizam a escrita por bloco. Os dados podem ser visualizados na Figura 6.

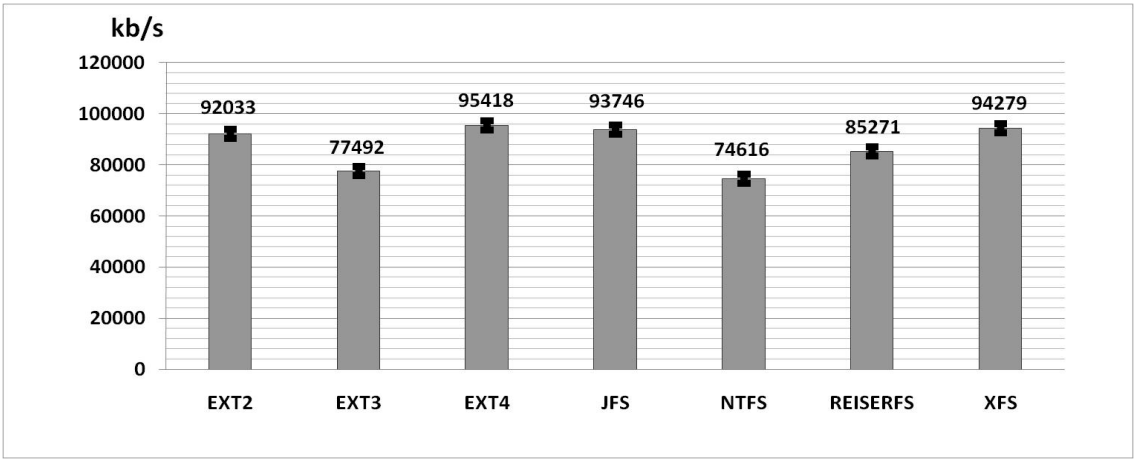

Figura 6 – Escrita por bloco

Com exceção dos SAS EXT3 E NTFS que obtiveram a menor taxa, a escrita por bloco não mostrou diferenças nas taxas obtidas nos demais SAS (Figura 6).

4.4.1 Uso de CPU na escrita por bloco

O uso de CPU no decorrer deste teste é descrito pela média da porcentagem gasta no processamento para a escrita por bloco. Os dados analisados podem ser visualizados na Figura 7.

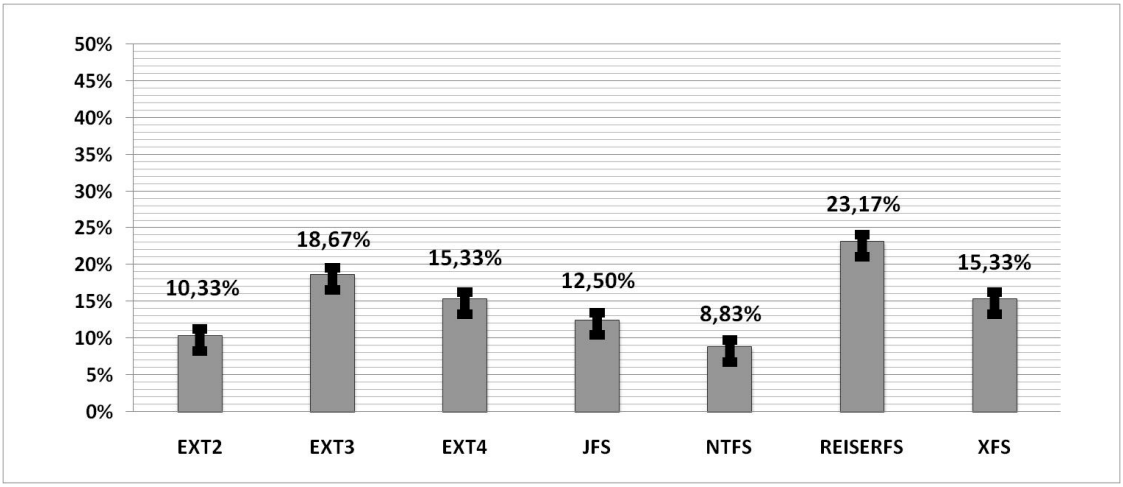

Figura 7 – Uso CPU escrita por bloco

A partir dos dados apresentados na Figura 7, o SA REISERFS foi o que teve maior uso de CPU, os SAS NTFS E EXT2 foram o que obtiveram o menor uso durante o processo de escrita por bloco.

4.5 Reescrita

Este teste visa mostrar o quanto rápido é cada sistema de arquivo para reescrever um arquivo já existente, modificando seu conteúdo por blocos. Os resultados são mostrados na Figura 8.

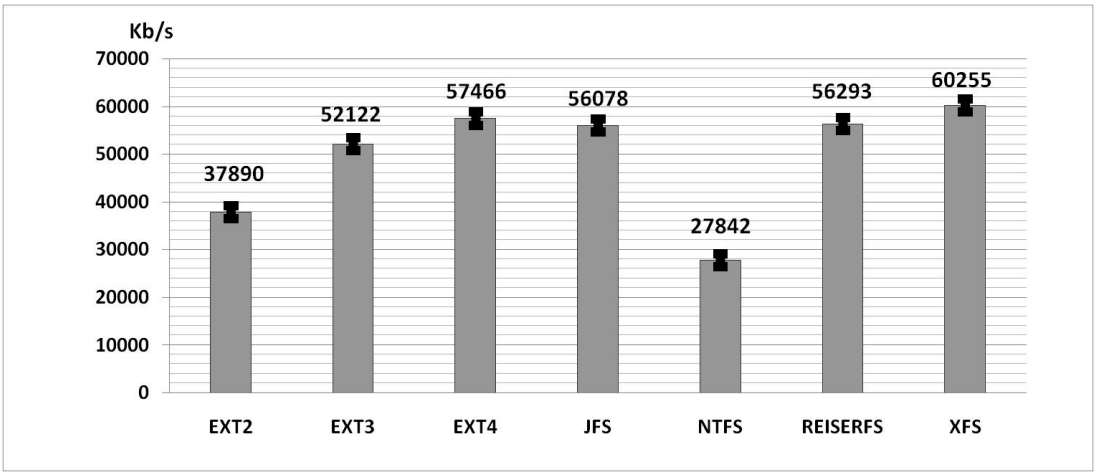

Figura 8 – Reescrita

Observa-se na Figura 8, que a menor taxa foi obtida com o SA NTFS e EXT2. Os demais SAS tiveram taxas similares.

4.5.1 Uso de CPU na reescrita

A reescrita dos dados não utiliza muito processamento, uma vez que seus dados são reescritos por bloco. A Figura 9 descreve o uso do CPU na reescrita de dados.

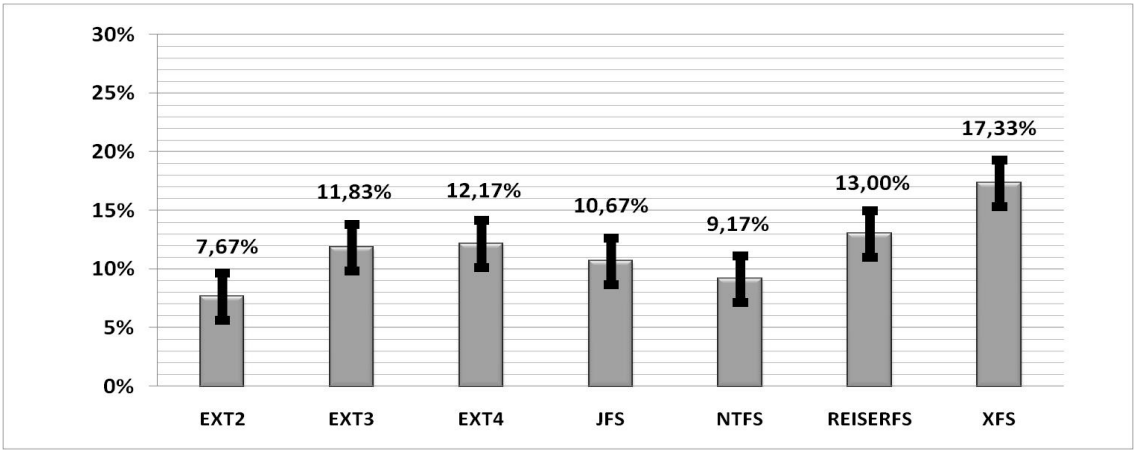

Figura 9 – Uso de CPU reescrita

O menor uso de CPU se deu no EXT2 e no NTFS, já o maior uso de CPU foi o XFS, como pode ser observado na Figura 9.

4.6 Leitura por caractere

Os resultados gerados por estes testes mostram o desempenho de ler, caractere a caractere, todos os dados gravados no arquivo. Conforme pode ser observado, tais resultados são apresentados na Figura 10.

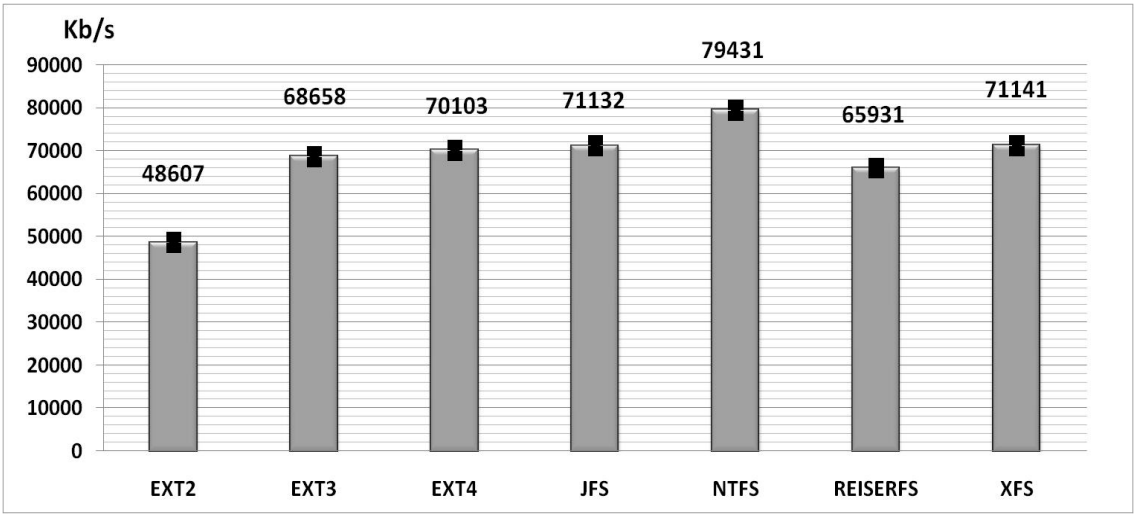

Figura 10 – Leitura por caractere

O SA NTFS mostrou a maior taxa neste teste, como foi apresentado na Figura 10. Também observa-se que a pior taxa de leitura por caractere foi encontrada no EXT2. Nos demais SAS não houve uma diferença significativa.

4.6.1 Uso de CPU na leitura por caractere

Este teste mede o uso médio de processamento para a leitura de caractere a caractere. A Figura 11 mostra o uso do CPU no momento do teste.

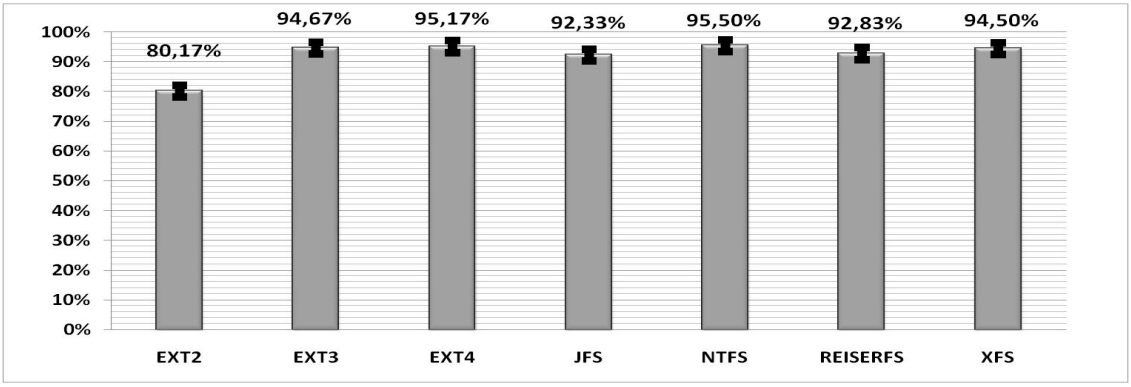

Figura 11 – Uso de CPU leitura por caractere

Como evidenciado na Figura 11, neste teste o SA que obteve o menor uso de CPU foi o EXT2, os outros sistemas de arquivo não apresentaram diferença significativa.

4.7 Leitura por bloco

Neste teste é analisado o tempo de leitura por bloco, onde a leitura ocorre a cada bloco de 4Kb por todo o arquivo. A Figura 12 mostra a taxa em Kb/s para a leitura por bloco.

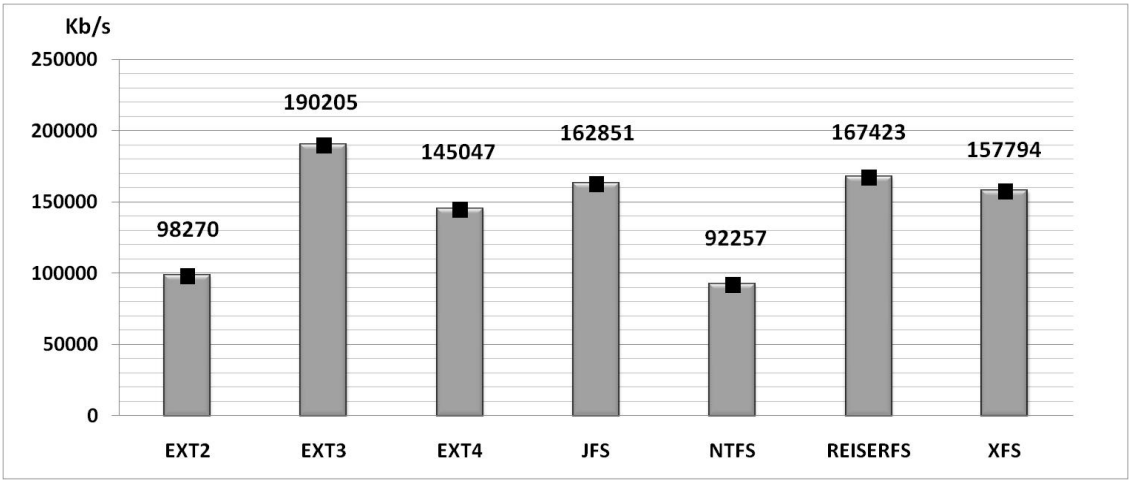

Figura 12 – Leitura por bloco

O SA que se mostrou mais rápido foi o Ext3. Este teste foi o que mostrou a maior variação entre os sistemas analisados. O EXT2 e NTFS apresentaram as piores taxas (Figura 12).

4.7.1 Uso de CPU na leitura por bloco

Este teste mensura a utilização de CPU média no decorrer da leitura do arquivo por blocos. A Figura 13 mostra o percentual de CPU utilizado na leitura por bloco.

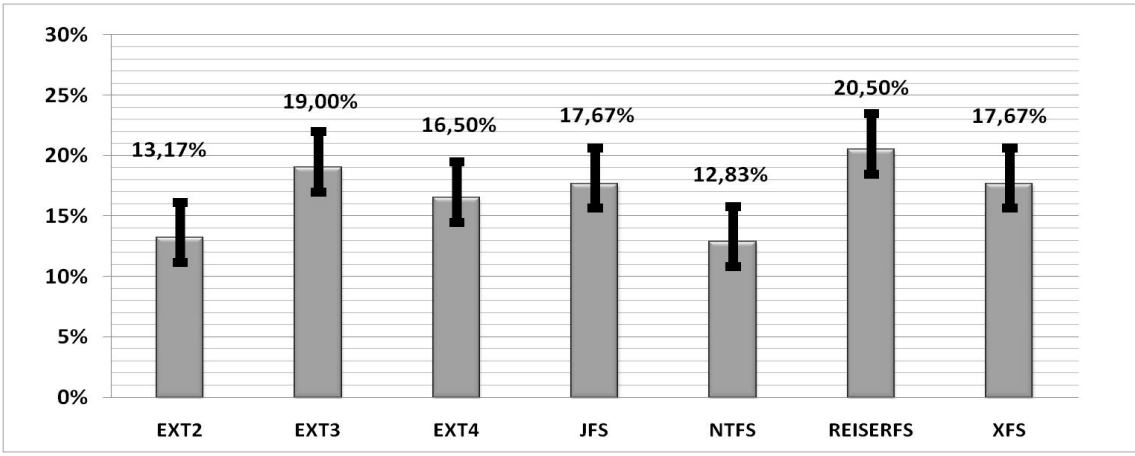

Figura 13 – Uso de CPU leitura por bloco

Tanto o EXT3 como o REISERFS não se mostraram diferentes, o menor uso de CPU se deu com o NTFS, porém o mesmo teve o pior desempenho na Leitura por bloco, como indicado pela Figura 13.

4.8 Busca Aleatória

Este teste visa uma busca por determinado dado gravado dentro do arquivo que foi gerado. A média de todas as buscas é expressa em segundos. A Figura 14 traz como resultado o tempo em segundos que se levou para encontrar um determinado arquivo.

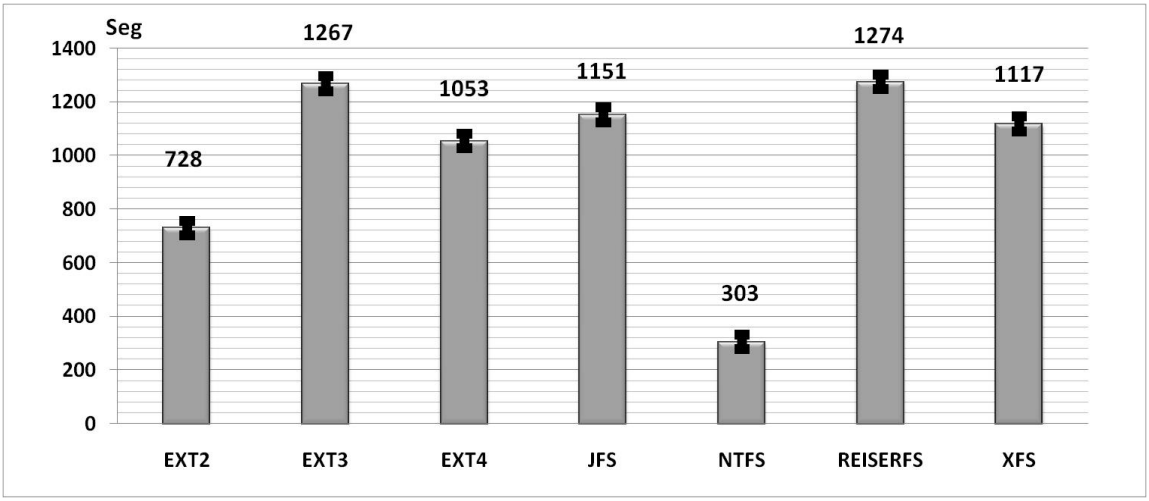

Figura 14 – Busca Aleatória

Conforme os dados apresentados por meio da Figura 14, observa-se que o SA NTFS foi o mais eficaz, o SA REISERFS e EXT3 obtiveram os piores tempos na busca aleatória.

4.9 Tabela Comparativa

Para uma melhor visualização dos dados foram criadas 2 tabelas com os valores dos testes realizados. Na Tabela 2 é visualizado os valores de taxa de escrita por caractere, escrita por bloco, reescrita, leitura por caractere, leitura por bloco e busca aleatória. Já a Tabela 3 mostra a visualização referente ao uso do CPU nos testes realizados.

Sistema de Arquivo	Escrita por caratere Kb/s	Escrita por Bloco Kb/s	REESCRITA Kb/s	LEITURA POR CARACTERE Kb/s	Leitura por bloco Kb/s	Busca Aleatória seg
EXT2	65493	92033	37890	48607	98270	728
EXT3	72586	77492	52122	68658	190205	1267
EXT4	72090	95418	57466	70103	145047	1053
JFS	72741	93746	56078	71132	162851	1151
NTFS	37588	74616	27842	79431	92257	303
REISERFS	70334	85271	56293	65931	167423	1274
XFS	60447	94279	60255	71141	157794	1117

Tabela 2 – Comparação de tempo nos resultados

Sistema de Arquivo	Escrita por caratere	Escrita por Bloco	REESCRITA	LEITURA POR CARACTERE	Leitura por bloco	Busca Aleatória
EXT2	86,6	10,3	7,7	80,2	13,2	x
EXT3	98,6	18,7	11,8	94,7	19,0	x
EXT4	98,5	15,3	12,2	95,2	16,5	x
JFS	96,1	12,5	10,7	92,3	17,7	x
NTFS	53,5	8,8	9,2	95,5	12,8	x
REISERFS	98,5	23,2	13,0	92,8	20,5	x
XFS	84,3	15,3	17,3	94,5	17,7	x

Tabela 3 – Uso de CPU nos resultados

5 CONCLUSÃO

A pesquisa desenvolvida ao longo deste trabalho buscou mostrar as principais diferenças entre alguns SAS mais utilizados no mercado, entre estas diferenças estão tempo de escrita por caractere, tempo de escrita por bloco, reescrita, leitura por caractere, leitura por bloco e busca aleatória assim como o uso de CPU em todas as análises efetuadas. Cada SA demonstrou ser satisfatório para um determinado tipo de aplicação, com isso não houve um único SA que possa ser utilizado para todos os tipos de aplicações. O SA NTFS obteve as principais diferenças entre os demais SAS. Uma vez que o SO utiliza um *drive* para ter compatibilidade com este SA, os dados coletados para este SA podem ter diferentes aspectos quando utilizado em seu SO padrão e desta maneira se torna impossível fazer uma análise comparativa eficiente do NTFS com os demais SAS.

É possível decidir a partir dos dados obtidos, por qual SA utilizar, sendo que, os principais tipos de servidores têm como principal objetivo o armazenamento de pequenos ou grandes arquivos, leitura de arquivos, busca de arquivos e processamento de informações.

Desta forma, os SAS EXT2, EXT4 JFS E XFS demonstram maior eficiência para servidores em que se tem a necessidade de gravar grandes arquivos. Para servidores em que a reescrita é a principal característica os SAS XJF JFS E EXT4 obtiveram as melhores taxas.

O SA EXT3 obteve a melhor taxa para leitura por bloco, desta forma seria o SA mais indicado para servidores onde sua principal necessidade é a leitura de arquivos. O menor tempo de busca aleatória nos SAS ocorreu no SA NTFS, porém devido ao uso de um *drive* externo para utilização deste SA no SO Linux, a busca aleatória que utiliza o menor tempo foi no EXT2, onde servidores de pesquisa que visam à procura por determinados trechos em um arquivo obteriam o melhor desempenho. Devido não utilizar *journaling* o SA EXT2 obteve o menor uso de CPU nas análises, para servidores exclusivos de calculo é o melhor SA que pode ser utilizado.

5.1 Contribuições

Resumidamente, as principais contribuições gerais deste estudo são avaliar as principais diferenças entre SAS mais utilizados nas versões Linux/Unix, assim como ter os testes realizados como base para melhor escolha do SA a ser instalado para um determinado tipo de aplicação.

5.2 Extensões

Este trabalho pode ser continuado com a analise de outros SAS também utilizados no mercado, tais como ZFS, UFS, UFS2, etc. Outra extensão para este trabalho pode ser feito utilizando outros softwares de *Banckmark* tais como *FFmpeg[19], GnuPG[20]* etc.

Referências Bibliográficas

[1] TANENBAUM, ANDREW S.; Sistemas Operacionais: projeto e implementação, tradução de Edson Furmankiewicz, Editora Bookman, 2000.

[2] M. TIM JONES. Anatomy of Linux journaling file systems. Disponível em: <http://www.ibm.com/developerworks/library/l-journaling-filesystems/index.html>. Acesso em: 10 jul. 2009.

[3] ALECRIM EMERSON. Sistemas de arquivos FAT e FAT32. Disponível em: < http://www.infowester.com/fat.php>. Acesso em: 13 dez. 2009.

[4] EMERSON ALECRIM. Sistemas de arquivos NTFS. Disponível em: <http://www.infowester.com/ntfs.php>. Acesso em: 12 ago. 2009.

[5] CARD, RÉMY; TS'O, THEODORE; TWEEDIE, STEPHEN. Design and Implementation of the Second Extended Filesystem. Disponível em: <http://web.mit.edu/tytso/www/linux/ext2intro.html>. Acesso em: 12 ago. 2009.

[6] ALECRIM EMERSON et al. Sistemas de Arquivos EXT3. Disponível em: <http://www.infowester.com/linext3.php>. Acesso em: 15 ago. 2009.

[7] ROBBINS, DANIEL. Common threads: Advanced filesystem implementor's guide. Disponível em: <http://www.ibm.com/developerworks/linux/library/l-fs7.html>. Acesso em: 14 Dez. 2009.

[8] THEODORE Y. Ts'o; TWEEDIE STEPHEN: Planned Extensions to the Linux Ext2/Ext3 Filesystem . Disponível em: http://e2fsprogs.sourceforge.net/extensions-ext23/extensions-ext23.pdf. Acesso em: 14 Dez 2009

[9] MATHUR, AVANTIKA; DILGER, ANDREAS; VIVIER, LOURENT. The new ext4 filesystem. Disponível em: <http://ols.108.redhat.com/2007/Reprints/mathur-Reprint.pdf>. Acesso em: 18 ago. 2009.

[10] JONES M. Tim. Anatomia do Ext4. Disponível em: < http://www.ibm.com/developerworks/br/library/l-anatomy-ext4/index.html>. Acesso em: 14 Dez 2009

[11] ALECRIM EMERSON et al. Introdução ao sistema de arquivos ReiserFS. Disponível em: <http://www.infowester.com/reiserfs.php>. Acesso em: 22 ago. 2009.

[12] The structure of the Reiser file system Disponível em: <http://homes.cerias.purdue.edu/~florian/reiser/reiserfs.php>. Acesso em: 22 ago. 2009.

[13] SILICON GRAPHICS. XFS File System structure. Disponível em: <http://oss.sgi.com/projects/xfs/>. Acesso em: 10 set. 2009.

[14] SILICON GRAPHICS. Getting Started With XFS. Disponível em: <http://oss.sgi.com/projects/xfs/papers/getting_started_with_xfs.pdf>. Acesso em: 10 set. 2009.

[15] IBM. Anatomy of Linux journaling file systems. Disponível em: <http://www.ibm.com/developerworks/linux/library/l-journaling-filesystems/>. Acesso em: 15 set. 2009.

[16] BONNIE++ Disponível em: <http://www.coker.com.au/bonnie++/>. Acesso em: 15 ago. 2009.

[17] MONITORAMENTO DE REDES. Brasil: Linux New Media, v. 31, 01 jul. 2007.

[18] SISTEMAS DE ARQUIVO. Brasil: Linux New Media, v. 2, 1 mar. 2005.

[19] FFMPEG. Disponível em: < http://ffmpeg.org/> Acessado em: 14 Dez 2009

[20] GNUPEG. Disponível em: < http://www.gnupg.org/> Acessado em: 14 Dez 2009