OTIMIZAÇÃO DE SERVIDORES GNU/LINUX PARA SISTEMAS GERENCIADORES DE BANCO DE DADOS

Janssen dos Reis Lima

PROJETO FINAL SUBMETIDO AO CORPO DOCENTE DA FACULDADE PROFESSOR MIGUEL ÂNGELO DA SILVA SANTOS (FeMASS) COMO PARTE DA SISTEMÁTICA DE AVALIAÇÃO POR PROJETOS (SAP).

Banca Examinadora:

_____

Prof. Alfredo Manhães (Orientador)

_____

Prof. Afonso Pinheiro

_____

Prof. Douglas Valiati

MACAÉ, RJ - BRASIL

JULHO DE 2010

RESUMO:

Sabe-se que não é de hoje que as empresas armazenam informações em banco de dados em computadores. Há muito já não se utiliza planilhas, seja em papel ou eletrônica, para controlar um fluxo de caixa e cadastro de clientes, por exemplo. O uso de sistemas de banco de dados ultrapassou a barreira das pesquisas em universidades e hoje é utilizado amplamente em diversas organizações, desde uma simples transação de venda até uma transação de ações negociadas em bolsas de valores.

Com o avanço das tecnologias de sistemas e dos próprios computadores, muitas vezes o consumo destes recursos não é levado em consideração, sendo sempre a primeira opção a atualização das tecnologias de hardware e software. Isso gera um investimento adicional, o que para uma grande empresa pode não ser algo que afete o orçamento, para uma empresa de pequeno ou médio porte isso influência e muito.

A proposta deste projeto é apresentar um estudo de caso que ajuda a área de TI (Tecnologia da Informação) da empresa a selecionar a melhor configuração dos recursos disponíveis em um sistema GNU/Linux, tais como: escalonadores de E/S (Entrada/Saída), sistema de arquivos etc. Este estudo também mostra que muitos administradores de banco de dados (DBA) não percebem que este tipo de abordagem é essencial para uma boa manutenção destes sistemas.

PALAVRAS-CHAVE: SGDB, Banco de Dados, GNU/Linux, Desempenho.

LISTA DE ABREVIATURAS E SIGLAS

ATA	Advanced Technology Attachment
CFQ	Complete Fair Queue
CIFS	Common Internet File System
CPU	Central Processing Unit
DAFS	Direct Access File System
DBA	Database Administrator
DMA	Direct Memory Access
EXT3	Third Extended File System
FIFO	First In, First Out
FS	File System
GB	Giga Bytes
I/O	Input/Output
IDE	Integrated Drive Electronics
JFS	Journal File System
MB	Mega Bytes
NAS	Network Attached Store
NFS	Network File System
OLTP	Online Transaction Processing
PIO	Programmed I/O
RAID	Redundant Arrays of Independent Disks
RAM	Random Access Memory
SAN	Storage Area Network
SAS	Serial Attached SCSI
SATA	Serial ATA
SCSI	Small Computer Systems Interface
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
TB	Tera Bytes
TI	Tecnologia da Informação
UDMA	Ultra DMA
XFS	eXtent File System

SUMÁRIO

1. INTRODUÇÃO

1.1. RESUMO

A tarefa de manter um sistema de informação funcionando plenamente não é tão simples e depende de vários fatores, entre eles a manutenção de sua infraestrutura, seja ela física ou lógica. Tratando-se de um ambiente em que os dados são de grande importância, a tarefa se torna mais complicada.

A idéia que surgiu para se trabalhar no assunto deste projeto foi justamente conciliar uma boa prática de analisar as tecnologias empregadas em SGBD's antes de sua implementação. Este projeto irá demonstrar através de um estudo de caso, a importância de se fazer um estudo deste porte para que os sistemas consigam utilizar o máximo de desempenho que um computador possa oferecer, e também saber aproveitar melhor estes recursos para manter o sistema seguro contra falhas e gargalos que podem fazer com que se gaste além do necessário. Sendo que em muitos casos, as empresas já entregam uma máquina para uma equipe sem ao menos consultar o que será utilizado.

Nas pesquisas que foram realizadas, foi notado através de participações em fóruns [1] que os administradores de banco de dados se preocupam muito pouco com o bom desempenho dos computadores. Para a maioria deles, uma máquina de última geração com bastante memória e discos rápidos é o suficiente, e que o desempenho do sistema se dará através de aperfeiçoamento nos códigos SQL's dos bancos de dados. Até esse ponto tudo bem. Porém, não é só de desempenho de SQL`s que um sistema se tornará mais rápido. Um SGBD também depende da maneira como um computador irá tratar as requisições de leitura e escrita em disco, a utilização correta da memória RAM etc. Só para exemplificar: sem um estudo de qual sistema de arquivos utilizar, um sistema poderá estar subutilizado pelo simples fato de um sistema de arquivos estar consumindo um tempo desnecessário de disco e também de CPU.

O trabalho desenvolvido teve por objetivo apresentar através de um estudo de caso, o quão importante é escolher as melhores configurações de parâmetros de configuração para que o sistema utilize com bom desempenho os recursos que lhe são fornecidos. Foi analisado o comportamento dos sistemas de arquivos e escalonadores de I/O. Também foi feito um estudo das tecnologias de armazenamento de discos e de ferramentas de *benchmarking* para analisar o desempenho em sistemas de arquivos e discos e em SGBD`s. Com isso, foi possível

observar o comportamento do PostgreSQL, fazendo um cruzamento de teste entre sistema de arquivos e escalonadores de I/O.

Este projeto foi baseado em tecnologias de software livre justamente para que qualquer pessoa ou empresa possa utilizar para testes e verificar que, em certos casos, é possível utilizar sistemas confiáveis a baixo custo. Baseado neste estudo, fica evidente que se possa realizar testes com outros SGBD's e até mesmo utilizando em outros sistemas operacionais.

Para apresentar este estudo, o material foi estruturado em capítulos como segue: o capítulo dois esclarece a finalidade dos SGBD's e expõe as características do PostgreSQL. No capítulo três são demonstradas as tecnologias de sistemas que envolvem um SGBD. O capítulo quatro descreve o estudo de caso que foi feito e mostra através de gráficos os resultados obtidos com os testes. Por último, o capítulo cinco apresenta a conclusão e as perspectivas de trabalhos futuros.

Obs.: Como se trata de um estudo de um sistema científico corporativo, seu nome neste projeto não será citado, sendo substituído por Sistema Protótipo.

1.2. OBJETIVOS

1.2.1. Objetivo geral

Definir métricas para verificação de desempenho de sistemas de arquivos e escalonadores de I/O para a utilização do SGBD PostgreSQL em sistemas operacionais GNU/Linux.

1.2.2. Objetivos específicos

Estudar as tecnologias envolvidas e elaborar um estudo de caso para definir a configuração que tiver o melhor desempenho nos testes desenvolvidos através das métricas definidas para utilização no novo equipamento do Sistema Protótipo.

1.3. JUSTIFICATIVA

O motivo para a realização do estudo foi a possibilidade de conseguir resultados melhores do que os atuais. Anteriormente, o Sistema Protótipo fora implementado com todas as configurações de sistema operacional padrões da instalação, sem definição de escalonador de I/O, sistema de arquivo ideal para as tecnologias de armazenamento etc. Com o estudo realizado, será possível definir de forma consistente e com provas sucintas o melhor ambiente para o sistema desenvolver um desempenho satisfatório, seguro e ideal.

1.4. METODOLOGIA

O trabalho foi realizado através de pesquisa bibliográfica, desenvolvendo um estudo de caso a partir do conhecimento adquirido.

2. SISTEMA GERENCIADOR DE BANCO DE DADOS

De acordo com Navathe (2005), um sistema gerenciador de banco de dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados.

Este capítulo tem o propósito de expor as características de um dos SGBD`s mais utilizados atualmente na comunidade de software livre e também em diversas organizações, o PostgreSQL.

2.1. SISTEMAS DE BANCO DE DADOS

Banco de dados é um recurso que foi desenvolvido para facilitar a busca de informações, eliminando o uso de planilhas e arquivos de papéis, integrando os dados de aplicações e fornecendo segurança no acesso aos dados.

Segundo Date (2004), os sistemas de banco de dados têm a finalidade de gerenciar as informações de um banco de dados, que é um repositório para uma coleção de arquivos de dados computadorizados. Sendo assim, é uma camada que fica entre o banco de dados e os usuários, podendo também ser interações de softwares diretamente com o banco de dados.

2.2. POSTGRESQL

PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional (SGBDOR), baseado no POSTGRES versão 4.2 desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. O POSTGRES foi pioneiro em vários conceitos que somente se tornaram disponíveis muito mais tarde em alguns sistemas de banco de dados comerciais. [2]

2.2.1. História

A implementação do POSTGRES iniciou-se no ano de 1986 com a liderança do professor Michael Stonebraker. Desde então o POSTGRES passou por várias versões até que, em 1992, a Illustra Information Technologies pegou o código fonte do POSTGRES e o comercializou. O projeto foi finalizado oficialmente em 1993, pois estava consumindo grande

parte do tempo que deveria ser dedicado a pesquisas de banco de dados. A última versão do POSTGRES de Berkeley foi a 4.2.

Em 1994, sob um novo nome, o Postgres95 foi liberado na Web por Andrew Yu e Jolly Chen, após adicionaram um interpretador da linguagem SQL. Como descendente de código aberto do original do POSTGRES de Berkley. Com isso, o aumento da comunidade de usuários aumentou consideravelmente, o que acabou ganhando novos desenvolvedores que ajudaram no crescimento do Postgres95.

Em 1996, foi escolhido um novo nome, PostgreSQL, para refletir o relacionamento entre o POSTGRES original e as versões mais recentes com capacidade SQL. Essa mudança também refletiu na migração da versão 1.09 para a versão 6.0

Este SGBDOR tem sido usado para implementar muitos aplicativos diferentes de pesquisa e de produção, incluindo: sistema de análise de dados financeiros, banco de dados de informações médicas, e vários sistemas de informações geográficas. Também tem sido usado como ferramenta educacional por várias universidades. [3]

Para se ter idéia do crescimento do PostgreSQL no Brasil, recentemente a Dataprev (Empresa de Tecnologia e Informações da Previdência Social) migrou a tecnologia proprietária MS SQL Server para PostgreSQL. [4] Foram 2.462 tabelas migradas contendo cerca de 1,7 bilhão de registros. Segundo informações do gestor do projeto, o banco atualmente consome 500 GB. Este é o maior banco de dados sociais da América Latina. Outra empresa do governo a utilizar o PostgreSQL é a Caixa Econômica Federal (CEF). [5] Todos os projetos que envolvam equipamentos x86 da CEF já nascem com PostgreSQL, incluindo mais de vinte e um mil terminais de autoatendimento, além dos sistemas de jogos, todos funcionando em GNU/Linux.

2.2.2. Características

PostgreSQL um software multi-plataforma com suporte a vários sistemas operacionais e também arquitetura de hardwares. Antes de uma versão estável do PostgreSQL ser lançada, ela é testada em diversas plataformas pelos desenvolvedores do projeto. Uma lista das plataformas suportadas pode ser acessada em http://pgdocptbr.sourceforge.net/pg82/supported-platforms.html. Neste endereço também estão listadas as plataformas não suportadas. Esta lista mostra as plataformas que não foram testadas pelos desenvolvedores oficiais do projeto, mas mesmo assim podem ser suportadas com alguns ajustes feitos por desenvolvedores externos do projeto.

Hoje, o PostgreSQL encontra-se na versão 8.4 e foi lançado em julho de 2009. PostgreSQL roda em todos os grandes sistemas operacionais, incluindo GNU/Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) e MS Windows. Possui suporte para armazenamento de objetos binários, incluindo figuras, sons ou vídeos.

Como todo software, o PostgreSQL também possui os seus limites. A tabela 1 nos mostra estes limites. Pode-se observar que os limites estão associados à quantidade de informações gravadas em disco, e isso também está associado diretamente com o limite que os sistemas de arquivos possuem de acordo com o sistema operacional que está instalado. Por exemplo: Não adianta o PostgreSQL ter um tamanho máximo de Bando de Dados ilimitado se o sistema operacional estiver com um sistema de arquivo que não tem capacidade para armazenar estas informações. Também tem de se pensar no limite de capacidade física de armazenamento em discos (o que não ocorre caso os dados estiverem sendo armazenados em sistemas de armazenamento externo, como será explicado no capítulo três).

Tabela 1: Limites do PostgreSQL

Limite	**Valor**
Tamanho máximo do Bando de dados	Ilimitado
Tamanho máximo de uma Tabela	32 TB
Tamanho máximo de uma Linha	1.6 TB
Tamanho máximo de um Campo	1 GB
Máximo de Linhas por Tabela	Ilimitado
Máximo de Colunas por Tabela	250-1600 dependendo do tipo de coluna
Máximo de Índices por Tabela	Ilimitado

Fonte: http://www.postgresql.org.br/sobre

Pode-se observar na tabela 1 que o PostgreSQL também trata de limites como tamanho de um campo. Este tipo de informação não será tratado nesta monografia, pois foge muito do escopo da proposta que está sendo apresentada. Porém, uma vasta informação está disponível no endereço <http://pgdocptbr.sourceforge.net/pg82/datatype.html> para quem tiver mais interesse neste assunto.

2.2.3. Arquitetura

O PostgreSQL utiliza o modelo cliente-servidor. Uma sessão consiste no trabalho conjunto entre dois processos: o processo servidor, *postmaster*, que recebe conexões das

aplicações clientes com o banco de dados e executa ações no banco de dados em nome dos clientes, e o processo cliente, *postgres*, que deseja executar operações de banco de dados [2]

2.2.4. Estrutura física

O PostgreSQL permite uma administração bem flexível em sua estrutura, devido a ela ser possível de ser personalizada. A base de sua estrutura está localizada no diretório '/var/lib/pgsql/' (referência em sistemas Red Hat). Neste diretório estão contidos vários subdiretórios e arquivos de controle. Também estão contidos quatro diretórios principais: pg_xlog, pg_clog, global e base. O subdiretório pg_xlog contém os logs de transações e no pg_clog ficam os logs de pontos de controle. No subdiretório global ficam armazenadas as tabelas que são compartilhadas por todas as bases de dados, as quais ficam armazenadas no subdiretório base. [6]

No PostgreSQL, quando uma tabela ou índice excede 1GB é dividido em segmentos de 1GB. Esta organização evita problemas em plataformas que possuem limitação de tamanho de arquivo, mas também permite um ganho de desempenho, partindo do princípio que o PostgreSQL sabe em que parte foi gravado tal registro.

2.3. TÉCNICAS DE TUNING

O processo de *tuning* é algo que só deve ser realizado quando houver uma real necessidade, pois envolve grande quantidade de processos, tanto do SGBD como do sistema operacional. [7] Este projeto irá tratar apenas do *tuning* de alguns subsistemas referentes a sistemas operacionais. Porém, existe também o *tuning* de Banco de Dados, que são obtidos através de aperfeiçoamento de SQL's e configurações personalizadas do SGBD.

Algumas técnicas de *tuning* são simples e de fácil compreensão, mas outras requerem conhecimento sobre o sistema operacional ou hardware. Está técnica requer muita análise, devendo sempre trabalhar cada caso de uma forma diferente.

Antes de realizar alterações no SGBD, o DBA deve considerar fatores externos ao SGBD, tais como o *hardware* e o *software* disponíveis ao SGBD. Toda essa otimização visa melhorar a capacidade do computador de processar os dados do sistema, fazendo com que processe uma carga maior de trabalho no mesmo intervalo de tempo.

Melhorar o *hardware* é a opção que trará um retorno visivelmente mais rápido. Porém, não é a realidade de toda empresa. Por isso, deve-se verificar uma possível melhora no desempenho através dos recursos disponíveis.

3. TECNOLOGIA DE SISTEMAS UTILIZADOS EM SGBD'S

Este capítulo define os conceitos relacionados a algumas tecnologias que são utilizadas num ambiente de um SGBD e também fornece informações básicas sobre ferramentas de avaliação de desempenho.

3.1. TECNOLOGIA DE ARMAZENAMENTO

O desempenho dos sistemas computacionais depende, em parte, dos dispositivos de armazenamento. [8] Melhorias nas tecnologias de armazenamento são cruciais para o desempenho de sistemas, principalmente de SGBD's, que necessitam armazenar e recuperar dados no disco a todo o momento.

O objetivo desta seção é fazer uma pequena comparação entre os dispositivos de armazenamento utilizados para manter os dados registrados por SGBD's. Não é objetivo deste projeto entrar em detalhes dos dispositivos de armazenamento secundário, tais como: discos ópticos, fitas DAT, dentre outros. A literatura atual disponibiliza vasto material sobre essas mídias. O foco é mostrar as tecnologias que realmente são utilizadas quando se deseja armazenar dados oriundos de SGBD's, e isto inclui as tecnologias de discos magnéticos e suas interfaces. Também será feita uma pequena comparação entre duas tecnologias de armazenamento externo que estão sendo utilizadas por grandes organizações: SAN e NAS. Cada uma com suas vantagens e desvantagens, estas tecnologias possuem praticamente o mesmo objetivo: permitir que diversos servidores tenham acessos a discos externos de modo rápido e confiável.

3.1.1. ATA

Mais conhecido como interface IDE (Integrated Drive Electronics), ATA (Advanced Technology Attachment) (devido à introdução de novas tecnologias, atualmente esta interface também é conhecida como Parallel ATA) é uma interface de unidade ligeiramente lenta. Foi a primeira tecnologia de discos com um controlador integrado ao drive. [9] Os discos mais antigos tinham um controlador em uma placa separada do drive.

Em toda sua história, a interface ATA passou por vários padrões de melhoria, tais como: Autodetecção de condições adversas e falhas (SMART), DMA (Direct Memory Access) e UDMA. O modo de transferência DMA foi das mais importantes, pois anteriormente era utilizado o modo de transferência PIO (Programmed I/O), que é uma forma de acesso a disco com ajuda da CPU. [10] Com a introdução do DMA, os dados são transferidos diretamente entre disco e memória, sem a interferência da CPU, que fica liberada para realizar outras tarefas.

Esta tecnologia é utilizada, em sua maioria, em computadores mais baratos, que têm como principal objetivo, rodar aplicativos monousuário. Seu uso em sistemas de Banco de Dados não é recomendável, devido à baixa taxa de transferência que os discos conseguem atingir.

3.1.2. SATA

Foi projetada para melhorar o desempenho das tecnologias IDE/ATA, mantendo a economia. Tem um rendimento melhor e mais rápido quando comparado aos discos paralelos do padrão ATA, isso devido a sua tecnologia serial, que faz uma melhor comunicação entre processadores e seus periféricos, ao contrário do padrão ATA, que sofre com interferências eletromagnéticas entre as vias, causando ruídos nos dados e deixando a transmissão mais lenta. [8] [10]

3.1.3. SCSI

Amplamente usado em *workstations* e servidores de médio e alto desempenho. A tecnologia SCSI pode dar suporte a até dezesseis dispositivos (além de discos, também outros dispositivos como scanner, impressora, dentre outros) em um único barramento. A quantidade de dados que pode transferir no barramento de dados é muito mais rápida e utiliza menos capacidade de CPU durante uma operação de E/S, apesar de ser uma tecnologia paralela. [8] [10]

3.1.4. SAS

A tecnologia SAS faz uso dos comandos SCSI, com a diferença da comunicação ser feita de forma serial. Foi concebida para obter melhor desempenho na transmissão de dados,

juntamente com a confiabilidade, alto desempenho e outros atributos que a tecnologia SCSI proporciona. [8] [10]

3.1.5. Comparativo entre as tecnologias de discos

Até aqui, foram demonstrados nos tópicos anteriores que as tecnologias de discos possuem muitas diferenças uma das outras. Cada uma delas foi melhorando ao passar dos anos e de acordo com as necessidades que o mercado demandava. Apenas para efeito de comparação e uma melhor visualização, a tabela 2 exibe a taxa máxima de transferência dos dados em cada tecnologia. Vale ressaltar que nem sempre a maior taxa será a melhor escolha, devido a fatores como: capital financeiro, necessidade tecnológica etc.

Tabela 2: Comparativo de taxa de transferência de dados entre as tecnologias

Tecnologia	**Taxa de transferência máxima**
IDE/ATA	133 MB/s
SATA	600 MB/s
SCSI	640 MB/s
SAS	1.2 GB/s

Fonte: Adaptação [8] [10]

3.1.6. SAN

É uma infraestrutura com dois ou mais dispositivos comunicando através de um protocolo serial SCSI, como *Fibre Channel* ou iSCSI. Utiliza redes de alta velocidades na comunicação entre os discos e os servidores, podendo alcançar velocidades de 4 Gb/s na transferência de dados. Porém existe a limitação da velocidade dos discos nela conectados. A figura 1 exibe um exemplo de uma SAN e os dispositivos que nela se integram. [11]

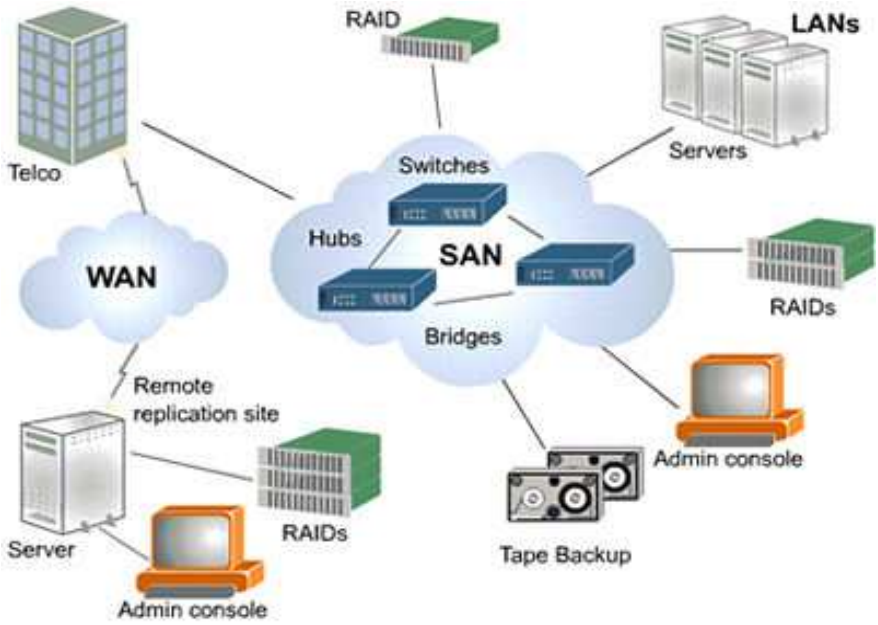


Figura 1: Integração entre SAN e dispositivos diversos

Fonte: http://www.gta.ufrj.br/grad/04_1/san/

3.1.7. NAS

É um computador (ou dispositivo) dedicado para compartilhamento de arquivos via NFS (Network File System), CIFS (Common Internet File System), ou DAFS (Direct Access File System). É uma alternativa a SAN devido a sua semelhança, porém não é o ideal quando se fala em desempenho para Banco de dados. [12]

3.1.8. SAN x NAS

Existem muitas teorias em relação à definição do melhor sistema de armazenamento externo. Tratando-se deste projeto, a melhor opção para um Banco de Dados seria SAN, pois fornece protocolos de camada, ou seja, o cliente (neste caso o servidor) visualiza um dispositivo de disco. Já NAS fornece protocolos de arquivos, com sistema de arquivos definido e uma área compartilhada. De acordo com Preston (2002, p. 41), “Existem alguns Bancos de Dados que exigem o acesso direto ao dispositivo. [...] Apenas discos presos a SAN podem fornecer acesso direto ao dispositivo. Arquivadores NAS podem apenas compartilhar arquivos - e não dispositivos”.

A tabela 3 mostra as diferenças entre SAN e NAS.

Tabela 3: Diferenças entre SAN e NAS.

	SAN	**NAS**
Protocol	Serial SCSI-3	NFS/CIFS
Shares	Raw disk and tape drives	Filesystems
Examples of shared	/dev/rmt/0cbn	\\filer\C\direcory\filename.doc
Items	/dev/dsk/c0t0d0s2	/nfsmount/directory/filename.txt
	\\.\Tape0	
Allows	Different servers can access the same raw disk or tape drive (not typically seen by the end user)	Different users can access the same filesystem or file
Replaces	Replaces locally attached disk and tape drives; with SANs, hundreds of systems can now share the same disk or tape drive	Replaces Unix NFS servers and NT CIFS servers that offer network shared filesystems

Fonte: Preston

A tabela 3 explica que SAN substitui discos conectados localmente, fazendo com que centenas de sistemas compartilhem o mesmo disco. Já NAS substitui servidores Unix NFS e servidores Windows CIFS oferecendo sistemas de arquivos compartilhados.

3.1.9. Escolhendo uma tecnologia de armazenamento para SGBD

Até aqui foram descritas as tecnologias disponíveis que podem servir um SGBD, porém a escolha depende de diversos fatores, principalmente econômicos. SAN é uma infraestrutura relativamente cara, porém possui vantagens em relação ao armazenamento em disco local e também ao NAS. Isso vai depender de que forma a informação tem valor para a empresa, pois apesar de um armazenamento local possuir desvantagens em relação ao SAN, existe meios de se contornar uma possível perda de dados, como por exemplo: utilizar RAID (*Redundant Arrays of Independent Disks*) para fazer espelhamento dos discos locais e até mesmo uma política de backup em um servidor distinto, ou, utilizar uma tecnologia de armazenamento externo mais barato para o backup de dados, como o NAS.

Essa escolha não depende somente do DBA ou de qualquer outro profissional de TI. E sim, de quem está investindo em tecnologia. Se a organização tiver dinheiro para bancar a melhor configuração para o cenário que será utilizado, seria ótimo. Porém, caso essa não seja a realidade da organização e for entregue para a equipe de TI uma estrutura pronta, com a escolha dos equipamentos e tecnologias de disco e outros itens, será importante fazer uma

análise para verificar que configurações serão melhores adaptadas para se obter o melhor desempenho. E é justamente este o propósito deste projeto.

3.2. SISTEMAS DE ARQUIVOS

Não será abordado nesta seção o conceito de Sistema de Arquivos. Como breve definição, pode-se dizer que a principal função de um sistema de arquivos é realizar a organização dos dados gravados em dispositivos de armazenamento, para que posteriormente possam ser acessados e manipulados.

As informações armazenadas em arquivos devem ser persistentes, armazenadas de forma segura e não sofrer danos quando ocorrer uma finalização abrupta de um processo. Neste contexto, é inaceitável que tais informações sejam perdidas, sobretudo em Sistemas de Banco de Dados, que armazenam informações que devem ser mantidas durante um período indeterminado de tempo.

Esses arquivos são gerenciados pelo sistema operacional, que fica com a responsabilidade de guardar as informações contidas neles. O mecanismo do sistema operacional que lida com arquivos é conhecido como Sistema de Arquivos.

Uma das razões para se obter ganho de desempenho em um sistema de banco de dados, é escolher corretamente um sistema de arquivos adequado ao SGBD que será utilizado. Isto porque o acesso aos dados de um sistema, em sua grande maioria, é feito através de dispositivos lentos, como os discos.

Cada sistema de arquivos tem uma finalidade específica: Alguns trabalham melhor com arquivos grandes; outros realizam buscas mais rápidas que os demais; alguns prezam pela segurança dos dados ao invés do desempenho etc. Nesta seção, serão abordadas as características dos sistemas de arquivos mais utilizados em GNU/Linux.

3.2.1. Ext3

É a versão otimizada do Ext2, que além de melhorias como programas de *restore* e personalização, foi adicionada a função de *journaling*.

O recurso de *journaling*, presente também em outros sistemas de arquivos, garante que um desligamento não planejado do sistema não gere inconsistência nos arquivos e também garante um reinicio mais rápido. Diferentemente do Ext2, que verifica a consistência

de todos os arquivos no sistema, o Ext3 apenas verifica os arquivos posteriores à última entrada no *journal*.

O sistema de arquivos Ext3 conta com três opções de configuração: *journal*, *writeback* e *ordered*. A opção *ordered* é opção padrão do sistema, e garante que os arquivos e metadados modificados são gravados diretamente no disco. No modo *writeback*, somente os metadados são gravados no *journal*. O arquivo só é modificado durante a sincronização do sitema de arquivos. O modo *journal* é o mais seguro, porém é também o mais lento, pois registra as modificações nos metadados e nos arquivos. [13]

3.2.2. XFS

Construído pela empresa SGI com o objetivo de substituir o sistema de arquivos EFS, usado no seu sistema operacional IRIX. Ele dispõe de várias opções para ajuste de desempenho. É o único sistema de arquivos que oferece suporte próprio para quotas.

O XFS é um sistema de arquivos de 64 bits. Sendo assim, tem suporte para abrigar arquivos de grandes proporções. É o único sistema de arquivos com *journaling* capaz de oferecer redimensionamento em tempo de execução. [13] [14]

3.2.3. ReiserFS

Ao contrário do XFS e JFS, este sistema de arquivos visa os arquivos de tamanho reduzido. É ideal para utilização em servidores de *news*, onde é comum a utilização de arquivos com menos que 1024 bytes.

ReiserFS também utiliza o recurso de *journaling*. É um sistema que dispõe de diversos utilitários para ajuste de desempenho, porém é muito pobre no que tange em ferramentas de recuperação. [13]

3.2.4. JFS

É o sistema de arquivos de código aberto fabricado pela IBM. Segundo a própria IBM, este sistema de arquivos visa servidores de banco de dados, servidores de arquivos e qualquer outra aplicação com grande necessidade de espaço em disco. Assim como o XFS, também é um sistema de arquivos de 64 bits. [13] [15]

3.3. ESCALONAMENTO DE I/O

O Kernel do GNU/Linux é responsável por controlar o acesso ao disco por meio de agendamento de requisições de I/O. Esse recurso é muito importante para sistemas que utilizam, de forma concorrente, o acesso à leitura e escrita em discos rígidos. Com isso, o sistema tentará aperfeiçoar a utilização da cabeça de leitura/gravação dos discos de modo a economizar as mesmas e ter o melhor desempenho possível para atender as requisições. [16] [17]

Os objetivos de utilizar algoritmos de escalonamento de I/O são:

- Minimizar o tempo desperdiçado de procura no disco rígido.
- Priorizar certas solicitações de I/O pelos processos.
- Dar uma parte da largura de banda de disco para cada processo em execução.
- Garantir que certos pedidos serão atendidos antes de um prazo determinado.

Existem diversos algoritmos que tem o propósito de organizar a leitura e escrita no disco. Os mais utilizados atualmente e que estão incluídos por padrão no Kernel 2.6 do GNU/Linux são: NOOP; CFQ; *Anticipatory* e *Deadline*.

Cada um destes algoritmos implementa níveis de recursos e características diferentes. É importante mencionar que cada algoritmo pode se adaptar melhor a determinadas cargas de trabalho e, por isso, devem ser realizados testes para escolher o que melhor se adapta. A idéia é economizar o uso da cabeça de leitura e gravação do disco. Caso não existissem estes algoritmos nenhum tipo de técnica seria adotado. Neste caso, as requisições seriam tratadas em um FIFO (primeiro a entrar, primeiro a sair). Nos dias de hoje, isso é inadmissível, pois os sistemas exigem rapidez e agilidade nos tratamentos e disponibilidade dos dados e, sem uma política de escalonamento a cabeça de leitura/gravação iria perder muito tempo para atender as requisições. Observe a figura 2.

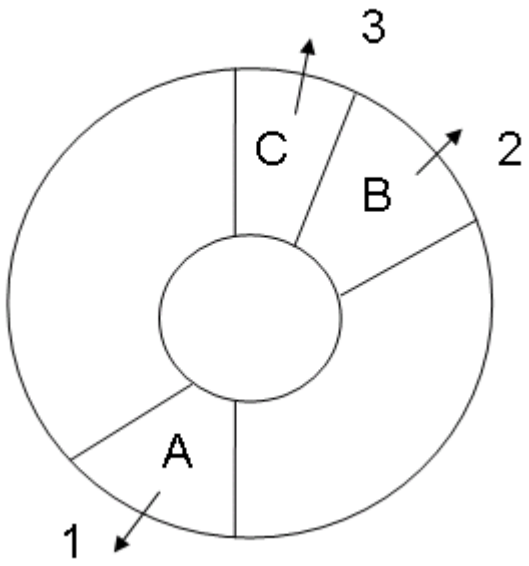


Figura 2: Tratamento de requisições de leitura em disco.

Fonte: Própria

Suponha que chegaram requisições para acesso ao disco na seguinte seqüência: ABC. Conforme ilustrada na figura 2, a cabeça de leitura/gravação do disco será movida até o setor A, em seguida será novamente movida até o setor B, e por último para o setor C. Observe que é uma operação muito custosa, e o tempo perdido para atender as três requisições é lento.

Os algoritmos de escalonamento foram criados justamente para melhorar o desempenho na utilização de dispositivos de armazenamento físico, de forma a utilizar mais rápido e atender diversas requisições de processos concorrentes. Em seguida, serão mostradas as características de cada um dos algoritmos. [17]

3.3.1. NOOP

Técnica extremamente simples que praticamente não adiciona nenhum recurso. É um algoritmo com apenas uma fila, no mesmo estilo de FIFO, e utiliza uma quantidade mínima de CPU. O único recurso extra deste algoritmo é a execução de junção entre as últimas requisições apenas. Junção é o processo de agrupar setores idênticos a fim de ser realizada a operação uma única vez. Portanto, este mecanismo tenta melhorar os últimos itens da fila para buscar uma possibilidade de resumir operações de I/O. [16] [17] [18]

3.3.2. CFQ

Complete Fair Queue. O principal conceito desde algoritmo é permitir que haja justiça entre os processos do sistema ao utilizar recursos de I/O. Este algoritmo tenta distribuir a largura de banda de I/O entre todas as requisições. Através de um processo interno, este escalonamento procura criar filas independentes para cada processo que queira usar os recursos de I/O. CFQ é o algoritmo padrão na maioria das distribuições GNU/Linux, justamente por ser uma boa opção em sistemas onde o recurso de I/O não deve ser monopolizado por nenhum processo. [16] [17] [18]

3.3.3. Anticipatory

Este algoritmo apresenta um atraso controlado para que algumas ações possam ser antecipadas. Isso é uma boa técnica para realizar operações de áreas de disco que estão próximas. Com as áreas adjacentes é possível realizar um processo de junção ou reordenação (controle devido dos movimentos de cabeça de leitura), melhorando o desempenho e a eficiência dos acessos ao disco. Um problema que pode ocorrer com o uso desde algoritmo é a presença de uma latência maior, causada pela espera das requisições. [16] [17] [18]

3.3.4. Deadline

Este algoritmo oferece uma garantia de execução em tempo real das operações de I/O. Ele utiliza um conjunto de filas e estas são orientadas em tempos de execução. Esta política favorece as operações de leitura, pois tem um prazo de expiração menor se comparado com o prazo das operações de escrita. [16] [17] [18]

3.3.5. Definindo um escalonador

O GNU/Linux oferece muitos recursos para personalização de sistemas. E pensando em oferecer mais praticidade, foi adicionado ao kernel 2.6 o recurso de não ser mais necessário reiniciar o sistema para que as alterações de escalonador tornem-se efetivadas. Portanto, o correto é testar cada escalonador em seu ambiente para obter os resultados e a partir daí analisar para se obter uma resposta com dados comprovados que a escolha seja a melhor possível.

Para verificar qual o escalonador que está ativo em seu sistema execute o comando conforme o exemplo abaixo:

```
[janssen@notebook ~]$ cat /sys/block/sda/queue/scheduler
noop anticipatory deadline [cfq]
```

O termo entre colchetes exibe qual o escalonador está ativado no sistema para o dispositivo **sda** (o primeiro disco SATA instalado no sistema GNU/Linux). Para alterar o escalonador de I/O, é preciso executar o seguinte comando:

```
[janssen@notebook ~]$ echo deadline > /sys/block/sda/queue/scheduler
```

3.4. FERRAMENTAS PARA ANÁLISE DE DESEMPENHO

Existem muitas ferramentas para testar o desempenho de um determinado sistema. Neste capítulo serão apresentadas ferramentas de *Benchmarking* para testes de disco e de banco de dados, já que a proposta desta pesquisa é aperfeiçoar um sistema para uso de SGBD. E para um banco de dados o que realmente importa é o desempenho de discos (podendo-se incluir sistema de arquivos) e o número de transações que serão bem sucedidas no banco de dados.

O principal objetivo de realizar este tipo de teste é fazer uma comparação de desempenho entre uma série de configurações pré-estabelecidas para ajudar na escolha da que tiver o melhor desempenho. A vantagem de utilizar uma ferramenta de *Benchmarking* é a facilidade de executar procedimentos de forma automatizada. Isso garante que a métrica utilizada será a mesma para todas as configurações.

Tratando-se de desempenho, é interessante fazer os testes com as ferramentas adequadas para o ambiente em questão. A escolha de ferramentas mal adequadas pode, futuramente, prejudicar o desempenho do sistema, levando a crer que o trabalho realizado não surtiu efeito.

3.4.1. IOzone

Esta ferramenta é uma excelente opção para investigar o desempenho de escrita e leitura no subsistema de I/O, pois oferece diferentes tipos de testes que tornam possível analisar o subsistema sob diferentes perspectivas. [19]

O Iozone cria arquivos de acordo com os parâmetros passados para a execução dos testes. Estes arquivos podem ser de até 4 GBytes, além de utilizar blocos de valores entre 64 bytes e 16 Kbytes.

Alguns tipos de testes que podem ser realizados pelo Iozone são:

- Write: tempo gasto para criar um novo arquivo.
- Rewrite: escrever em um arquivo já criado.
- Read: ler um arquivo já criado.
- Reread: ler um arquivo que foi recentemente lido.
- Random Read: ler em partes aleatórias de um arquivo.
- Random Write: escrever em partes aleatórias de um arquivo.

Um detalhe interessante que chama a atenção desta ferramenta é a possibilidade de gerar gráficos tridimensionais com o *software* gnuplot (disponível em http://www.gnuplot.info) através dos arquivos de saída que são gerados. Um exemplo da plotagem do resultado de um teste pode ser visualizado na figura 3.

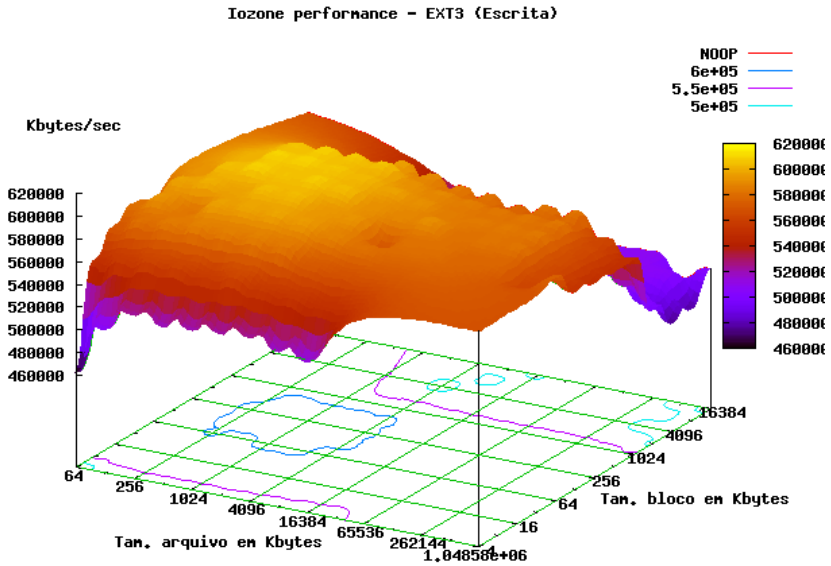


Figura 3: Plotagem 3D do resultado do desempenho de escrita do FS EXT3

Fonte: Extraído do *software* gnuplot

No exemplo da figura 3, o gráfico gerado exibe os dados coletados através dos testes de escrita utilizando o sistema de arquivos EXT3 e o escalonador NOOP. Observe que este

gráfico possui a taxa em que a operação foi executada, em Kbytes/sec, o tamanho do arquivo e o tamanho do bloco. Perceba que quanto maior o tamanho do arquivo e o tamanho do bloco o desempenho sofre variação. Com a visualização deste gráfico, é possível compreender que o sistema tem o seu melhor desempenho atingido na área representada pela área azul na base do gráfico, também representada na parte "montanhosa" do gráfico pela cor amarela, como pode ser observada na legenda à direita. Isso representa que o sistema observou o melhor desempenho neste ponto do gráfico. A utilização do *software* gnuplot auxilia bastante na leitura dos resultados obtidos pelos testes.

3.4.2. BenchmarkSQL

BenchmarkSQL é um *software* que testa o desempenho de banco de dados através do padrão TPC - *Transaction Processing Performance Council*. Todo material referente a esse padrão está disponível em http://www.tpc.org. O TPC possui diversos *benchmarks* para diversos tipos de testes, como: transações, OLTP, *Data Mining*, entre outros. Para os objetivos desde projeto, foi escolhido um *benchmark* que realizasse um teste em uma base de dados que simule um ambiente real do sistema utilizado, e um *benchmark* no padrão TPC-C tem esse objetivo, pois simula um ambiente de vendas de uma fábrica com diversas transações concorrentes, de maneira que possa forçar o acesso ao banco de dados tanto na gravação de dados como também na leitura, podendo ser simultaneamente. [20]

O BenchmarkSQL é um *software* de código aberto desenvolvido em Java e com recursos simples. Sua instalação é fácil e sua interface é bastante amigável.

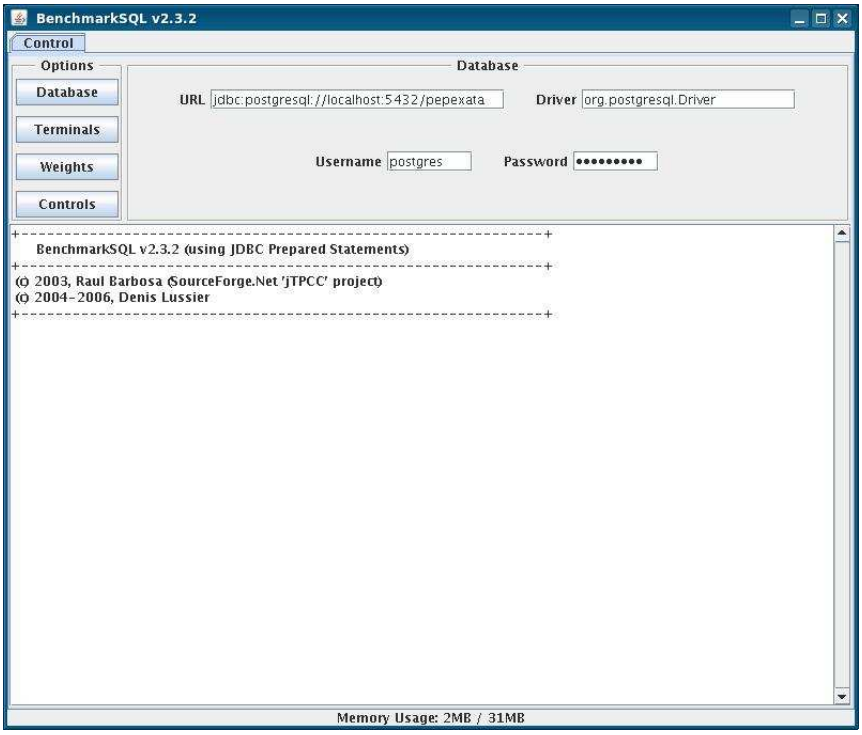


Figura 4: Tela inicial do BenchmarkSQL com suas opções de configuração.

Fonte: Capturada do próprio *software.*

A imagem 4 exibe a tela inicial do BenchmarkSQL, onde é possível indicar a base de dados que será testada, o *driver* de conexão e o usuário que se conectará ao banco para que o teste possa ser realizado. Após a realização do teste, o programa exibe um relatório com a média de transações efetuadas por minuto, o horário inicial e final do teste e a quantidade total de transações que foram feitas. Uma amostra deste relatório é mostrada na figura 5.

```
Measured tpmC  =60000*13219/600051 = 1.321

Session Start   2009-07-21 09:48:01
Session End     2009-07-21 09:58:01
Transaction Count      29303
```

Figura 5: Relatório do resultado final de um teste executado pelo BenchmarkSQL.

Fonte: Capturada do próprio *software.*

3.4.3. Iostat

Iostat é um aplicativo que está contido no pacote Systat (http://pagesperso-orange.fr/sebastien.godard/). Este aplicativo analisa uma série de informações sobre o

subsistema de I/O, podendo identificar a utilização de leitura e escrita nos discos. É possível obter informações da quantidade de transferências por segundo, blocos lidos e escritos por segundo, entre outras informações.

Este aplicativo auxilia a investigação do funcionamento dos discos e sistemas de arquivos. É ideal para monitorar constantemente essas métricas para observar se está ocorrendo sobrecarga no sistema.

4. ESTUDO DE CASO

O estudo de caso foi realizado visando obter um melhor desempenho do sistema que terá seu parque de máquinas atualizado. Atualmente o sistema utiliza um computador Pentium 4 2.8 Mhz com 2 MB de memória RAM, HD IDE de 80GB. Os principais *softwares* instalados são: GNU/Linux CentOS 3.8 kernel 2.4, Java JDK 1.5.0_11, PostgreSQL 7.4 e sistema de arquivos Ext3.

Para substituição do parque de máquina, foi iniciado um projeto para atualizar a versão do sistema operacional e também dos principais softwares, como a versão do PostgreSQL e também do Java. Neste projeto foi definido que deveria ser testado o equipamento com outro sistema de arquivo e verificar o desempenho do banco de dados do aplicativo.

4.1. EQUIPAMENTO UTILIZADO

Para a execução dos testes foi utilizado o computador que servirá de base para a nova imagem do sistema. Sua configuração é a seguinte: Intel Core 2 Duo E4600 2.4 GHz, 4GB RAM e HD SATA 160 GB.

Foi instalado o sistema operacional GNU/Linux CentOS 5.3 kernel 2.6. Para a realização dos testes com outros sistemas de arquivos, foi necessário atualizar o CentOS através do repositório *Plus* da distribuição, pois o sistema de arquivos padrão que vem instalado e disponível é o Ext3.

4.2. FERRAMENTAS

A priori, foi cogitada a utilização de duas ferramentas *Benchmark* para comparar o desempenho entre os Sistemas de Arquivos e outras duas para comparar o desempenho de transações de Banco de Dados.

Com a evolução dos testes e complexidade do tema, foram definidos um *Benchmark* para cada teste devido as seguintes razões:

- O software IOZone possui mais recursos que o *software* Bonnie++, além de gerar saídas para gráficos no formato Excel e também plotagem de gráficos tridimensionais com o software gnuplot.

- O *software* pgBench, que vem disponível no pacote *contrib* do PostgreSQL não foi utilizado, devido ter sido constatado que sua tecnologia de desenvolvimento utiliza o padrão TPC-B, que está obsoleto, segundo informações do consórcio TPC (Transaction Processing Performance Council), entidade regida por empresas de tecnologia interessadas nos resultados deste tipo de teste. Portanto, foi buscada uma solução que abrangesse ao padrão TPC-C, que atualmente é o padrão para testes em ambientes OLTP.

Com isso, ficou definido a utilização dos *softwares* de código aberto IOZone (para verificar o desempenho em Sistemas de Arquivos) e BenchmarkSQL (para verificar o desempenho em Banco de dados), descartando o uso das ferramentas pgBench e Bonnie++. Na tabela 4, pode ser verificada a relação dos principais *softwares* utilizados no testes.

A escolha do BenchmarkSQL foi interessante, porque além de utilizar o padrão TPC-C, este *software* foi desenvolvido em Java, e utiliza conexões JDBC, o que seria um ambiente bastante similar ao que encontramos atualmente no sistema para algumas aplicações.

Outro fato importante foi a definição de como seriam realizados os testes. A princípio seriam realizados diversos testes de simulação do ambiente do sistema, tais como: conversão do BD para binário, conversão de binário para BD, *backup* e *restore* do banco de produção etc. Porém, foi notado que alguns testes seriam desnecessários, devido ao tempo em que os mesmos eram executados (por exemplo: não dava para comparar o desempenho entre os sistemas de arquivos pelo pouco tempo que o teste era executado). Com isso, foi feito um *script* incluindo vários testes que simulassem um ambiente do sistema, porém, ao invés de fazer *backup* e *restore* do banco de produção, foram realizados *backup* e *restore* do banco gerado pelo BenchmarckSQL que possuía mais de 1GB de dados (simulação igualmente a uma situação real de dados armazenados no ambiente de desenvolvimento). O resultado disto foi denominado *Benchmarck* PROTOTIPO. Como incremento, o *benchmark* realizava outras tarefas, tais como: truncamento, limpeza e remoção de tabelas.

Tabela 4: Softwares utilizados para a realização dos testes

Software	**Versão**	**Descrição**
Gnuplot	4.0.0-14.el5	Plotagem tridimensional de gráficos
Postgresql	8.1.11-1.el5_1.1	SGBD
postgresql-jdbc	8.1.407-1jpp.4	Drivers JDBC para conexão com PostgreSQL
Iozone	3.326	Benchmark de I/O
BenchmarkSQL	2.3.2	Benchmark de SQL
Java JDK	1.6.0_14	Kit de desenvolvimento Java com compilador, interpretador e utilitários
Pgadmin3	1.8.4-1.rhel5	Ferramenta para administração do PostgreSQL
reiserfs-utils	3.6.19-2.4.1	Utilitários para ReiserFS
Xfsprogs	2.9.4-1	Utilitários para XFS
Jfsutils	1.1.12-1	Utilitários para JFS
CentOS	5.3	Sistema GNU/Linux baseado em RHEL
Kernel	2.6.18-128.1.10.el5.centos.plus	Suporte a pacotes adicionais

A respeito dos *softwares* mencionados na tabela 4, será explicada a instalação e configuração das três ferramentas de *benchmark* que foram utilizadas no estudo de caso. Os demais *softwares* não serão abordados, devido a ser fora do escopo deste projeto.

4.2.1. IOzone

Para a instalação do IOzone foi necessário baixar a versão 3.326 no formato de pacote RPM (*Red Hat Package Manager*), disponível em http://www.iozone.org/src/current/iozone-3-326.i386.rpm. Para instalar basta executar o comando *rpm –ivh iozone-3-326.i386.rpm*.

A execução do *benchmark* IOzone pode ser realizada de forma simples, pois ele pode ser executado em modo automático com a opção –a. Porém, como o estudo de caso já tinha suas métricas definidas, o comando para execução dos testes ficou da seguinte maneira: *iozone -s 1g -r 4k -r 8k -i 0 -i 2*, onde a opção –s 1g é usada para criar um arquivo de 1GB, –r

é usada para criar blocos de 4k e 8k, e –i 0 e –i 2 utilizado para selecionar os modos de escrita e leitura, respectivamente.

4.2.2. BenchmarkSQL

Como a ferramenta BenchmarkSQL foi desenvolvida em Java, é pré-requisito que esteja instalado no sistema uma versão da máquina virtual Java. Foi necessário baixar e instalar a versão 1.6.0_14 do JDK, disponível no site da Oracle em http://java.sun.com/javase/downloads/widget/jdk6.jsp. Também é necessário a instalação do PostgreSQL Server com o *driver* JDBC. O PostgreSQL pode ser baixado via *software* YUM, disponível no sistema operacional GNU/Linux CentOS. Para este projeto, o PostgreSQL foi selecionado durante a instalação do CentOS.

Para instalar o BenchmarkSQL, baixar o arquivo disponível em http://sourceforge.net/projects/benchmarksql/ e descompacta-lo no diretório /opt/BenchmarkSQL. Antes iniciar as etapas

Sua utilização consiste em quatro etapas:

- Criação das tabelas no PostgreSQL. Executar o *script* runSQL.sh.
- Carga das tabelas. Executar o *script* loadData.sh.
- Criação de índices. Executar o *script* runSQL.sh.
- Execução do *benchmark*. Executar o *script* runBenchmark.sh.

Todos os comandos devem ser executados informando como parâmetro o arquivo de configuração da conexão com o banco de dados.

4.2.3. Benchmark Protótipo

O Benchmark Protótipo foi desenvolvido em *Shell Script* e tem o objetivo de obter o tempo total de realização dos procedimentos do Sistema Protótipo em relação a cada escalonador de I/O e sistema de arquivos. O teste consiste nas seguintes etapas:

- Conversão de uma tabela do banco de dados para arquivo binário.
- Conversão de arquivo binário para tabela no banco de dados.
- Backup e restauração do banco de dados.

Para sua execução, é necessário executar o comando bench_backup.sh <parâmetro>. O parâmetro a ser passado é o escalonador que será utilizado.

4.3. TESTES

O objetivo desta análise foi para escolher o sistema de arquivos com o melhor desempenho e o escalonador de I/O mais adequado para ser utilizado na nova imagem do Sistema Protótipo.

O primeiro FS a ser testado foi XFS, seguido de EXT3, JFS e ReiserFS. Para cada FS, foram realizados os testes na seguinte seqüência:

- IOzone
- BenchmarkSQL
- Benchmark PROTOTIPO

O teste realizado com o IOzone foi para criar arquivos crescentes até o limite de 1 GB com blocos de tamanho de 4k e 8k. Para cada escalonador foi realizado um teste. Com o arquivo gerado, foi possível gerar gráficos com o gnuplot para, no final, comparar o desempenho entre os sistemas de arquivos. O resultado dos testes estão representados pelos gráficos que estão no anexo I deste projeto.

Com o auxílio dos gráficos gerados pelo gnuplot através das saídas do IOzone, foi possível comparar cada sistema de arquivo e escalonador de I/O. É possível verificar que os sistemas de arquivos XFS e JFS realmente trabalham melhor com arquivos de grande capacidade, sem contar que mantém as operações de escrita e de leitura estáveis, não oscilando como acontece com os sistemas de arquivos EXT3 e, principalmente, o ReiserFS. Estas oscilações são melhores visualizadas nas bases dos gráficos, conforme podem ser vistas no anexo I deste projeto.

Na segunda fase do teste foi a executado o BenchmarkSQL. Esta ferramenta foi utilizada para cada sistema de arquivo vezes a quantidade de escalonador, totalizando trinta e dois, pois o intuito foi analisar o desempenho em um banco pequeno e sem transações concorrentes, e o outro teste com o banco em crescimento e transações concorrentes, justamente para verificar o desempenho do SGBD.

A última etapa do teste foi a execução do script criado para simular operações rotineiras que são utilizadas pelo sistema.

4.4. RESULTADOS

Foi necessário representar os resultados em gráficos em forma de barras para que a análise pudesse ser feita sem ocorrência de erros.

Na figura 6, pode-se observar que o sistema de arquivos JFS obteve vantagem sobre todos os demais sistemas de arquivos, inclusive tendo um desempenho 77,55% melhor em relação ao sistema de arquivos EXT3 (quando comparado JFS/Deadline contra EXT3/CFQ, representado pela figura 7), que é utilizado na imagem do sistema protótipo na configuração atual. Este teste simula um ambiente do sistema protótipo, onde em média ocorrem dez conexões ao banco sem concorrência de transações, ou seja, com um único depósito para relacionamento.

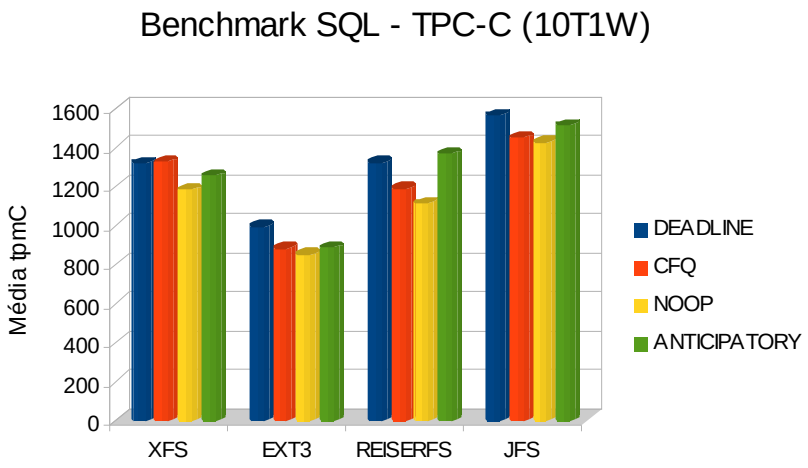


Figura 6: Desempenho do FS com BenchmarkSQL. BD sem concorrência de transações.

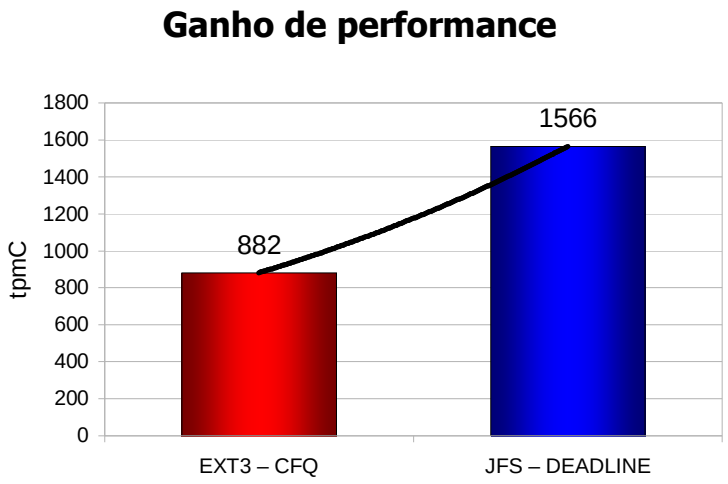


Figura 7: Representação de 77,55% de ganho de desempenho entre JFS/Deadline e EXT3/CFQ.

Como teste para comparação entre transações concorrentes, foi obtido o resultado conforme exibido na figura 8.

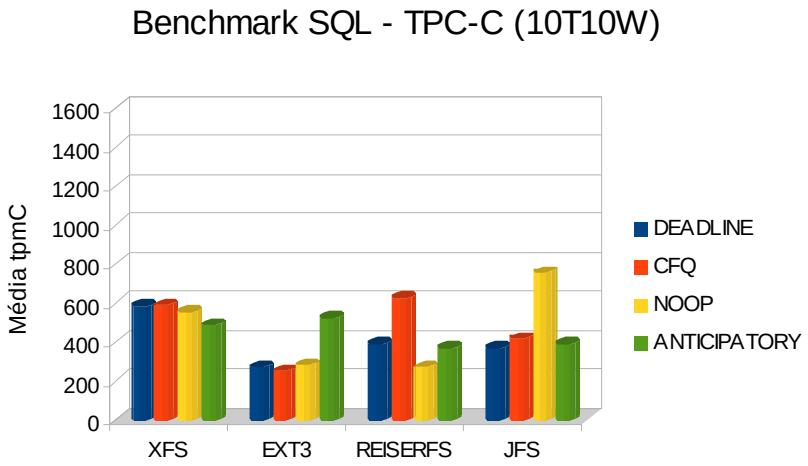


Figura 8: Desempenho do FS com BenchmarkSQL. BD com concorrência de transações

No teste executado com concorrência entre transações, o FS XFS obteve o melhor desempenho na média, porém não foi muito superior ao JFS, tendo inclusive um menor desempenho quando comparado com o escalonador NOOP do FS JFS e CFQ do ReiserFS.

Como forma de simulação do sistema, o último teste relacionado à banco de dados foi o script feito para simular as operações deste sistema. A figura 10 mostra o desempenho

entre os sistemas de arquivos na execução destas operações. Quanto menor o tempo de execução, melhor o desempenho. Mais uma vez, o FS JFS obteve o melhor resultado entre os sistemas de arquivos testados. O ganho de desempenho em relação ao FS EXT3 não foi superior a 7%, porém destaca-se a informação de que algumas operações do sistema utilizam a partição "/" do sistema operacional, na qual utiliza o FS EXT3 para ler/gravar dados destas operações. Pode-se verificar neste teste, que o único FS que teve melhor desempenho que o EXT3 foi o JFS.

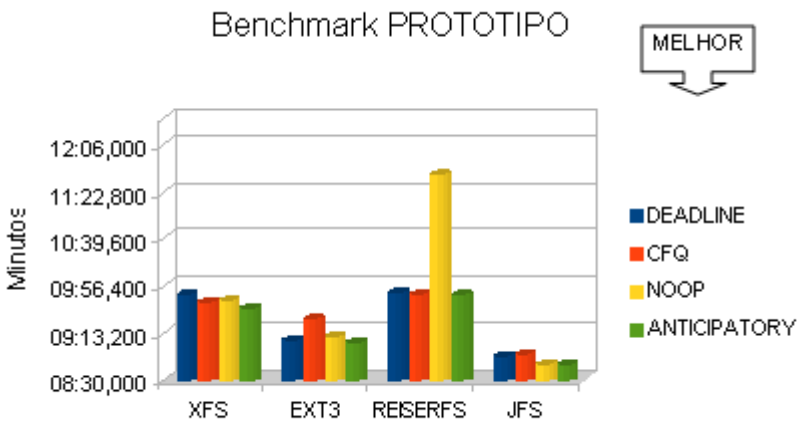


Figura 10: Desempenho de FS em simulação de operações do Sistema Protótipo

Tendo em vista que o FS JFS obteve o melhor desempenho nos testes de *Benchmark* de banco de dados, foi realizado o último teste para escolha do escalonador, no qual ficou sendo monitorada uma máquina de campo num período de um dia. Neste teste, foi verificado o volume de acesso de leitura/escrita ao disco, conforme demonstrado na figura 11. O gráfico demonstra que há mais requisições de leitura quanto de escrita, levando a conclusão de que o escalonador a ser escolhido é o *Deadline*.

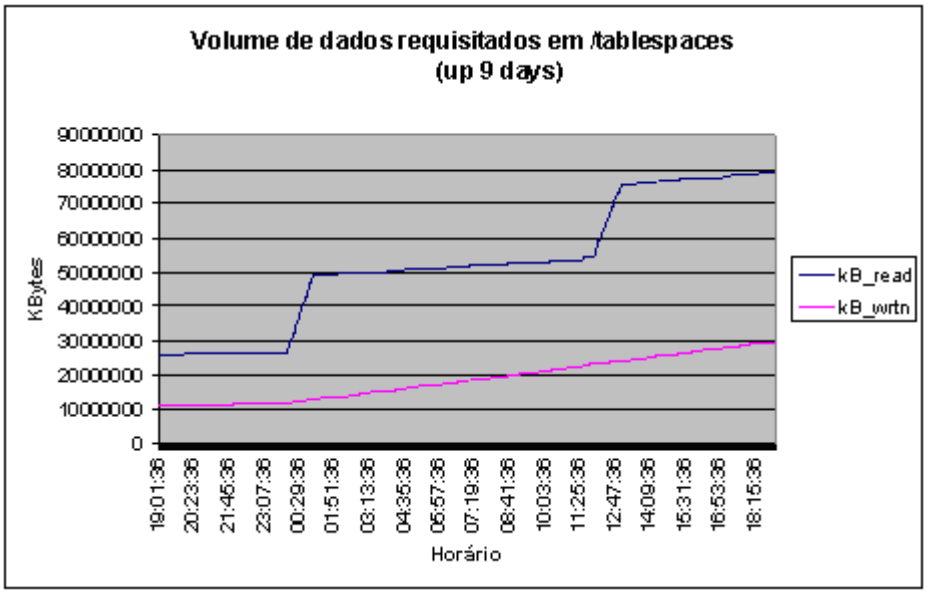


Figura 11: Acesso de leitura/escrita na partição /tablespaces de máquina de campo.

Por último, foi realizado um teste de integridade dos Sistemas de Arquivos JFS e EXT3, visando verificar a capacidade de recuperação destes FS em caso de perda de dados. Na simulação, foi testada a gravação em disco utilizando várias instâncias do utilitário “dd” e em determinado momento desligando o equipamento de forma abrupta. A recuperação do JFS obteve uma ligeira vantagem em relação ao EXT3, sendo imperceptível sua recuperação em uma partição com 35GB.

5. CONCLUSÃO

5.1. CONCLUSÕES GERAIS

Este projeto teve como objetivo abordar o *tuning* de servidores GNU/Linux para utilização de SGBD, que é um processo evolutivo de análise e otimização que pode ser realizado em diferentes aspectos, com a finalidade de melhorar o seu desempenho de acordo com a aplicação das métricas que foram utilizadas.

Nos testes de Banco de Dados realizados pelo BenchmarkSQL, pode-se notar que o FS JFS obteve melhor desempenho em relação aos outros FS. O mesmo pode-se afirmar nos testes de gravação, conforme as métricas definidas para a realização dos testes. Em relação à escolha do melhor FS nos testes de escrita e leitura pelo IOzone, pode-se constatar que o desempenho dos FS JFS e XFS são relativamente idênticos, ocorrendo empate técnico entre os dois FS no quesito desempenho de escrita. No quesito desempenho de leitura, o FS ReiserFS obteve o melhor resultado. Porém, o objetivo do teste não foi verificar apenas leitura em disco, e sim o desempenho de um servidor GNU/Linux para a utilização de um SGBD. Portanto, o desempenho com operações em banco de dados é que conta para a esta análise.

O escalonador escolhido foi o *Deadline*, em razão de ter obtido melhor desempenho nos testes relacionados a banco de dados. Como embasamento, nos testes de requisição de disco, verificou-se que o acesso à leitura é superior ao acesso de escrita, sendo este teste uma ótima referência para a escolha.

Para facilitar o processo de testes, foram desenvolvidos *scripts* para simular o ambiente do Sistema Protótipo. O que foi bastante útil para comparar as métricas nos diferentes sistemas de arquivos e escalonadores. Conclui-se que no momento de escolher uma estrutura para armazenar um SGBD, é importante que se faça uma análise para obter um bom desempenho no uso de um sistema.

5.2. PROPOSTAS DE TRABALHOS FUTUROS

Com base na proposta deste projeto e o os testes realizados, pode-se a partir deste, testar as configurações internas do PostgreSQL e também configurações avançadas de uso de memória do GNU/Linux. Também é válido realizar um estudo com as mesmas configurações feitas neste estudo de caso em outro sistema operacional, sendo possível fazer uma comparação entre os sistemas operacionais. Porém, é necessário levar em consideração os

aspectos de sistemas de arquivos, pois alguns sistemas operacionais não utilizam certos sistemas de arquivos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] PostgreSQL. **Lista oficial de discussão sobre PostgreSQL no Brasil.** Disponível em <http://listas.postgresql.org.br/cgi-bin/mailman/listinfo>. Acesso em 20 mai 2010.

[2] PostgreSQL. **Documentação do PostgreSQL**. Disponível em <http://pgdocptbr.sourceforge.net/>. Acesso em 20 mai 2010.

[3] NAVATHE, Shamkant; ELMASRI, Ramez E., **Sistemas de Banco de Dados**, 4.ª edição, São Paulo, Ed. Addison-Wesley, 2004.

[4] CASTRO, Jorge de. **Dataprev aperfeiçoa sistema para previnir fraudes.** Disponível em <http://portal.dataprev.gov.br/tag/postgresql/>. Acessado em 20 mai 2010

[5] CASTRO, Jorge de. **Governo Federal prioriza uso de tecnologia livre.** Disponível em <http://portal.dataprev.gov.br/tag/caixa-economica-federal/>. Acessado em 20 mai 2010.

[6] ALMEIDA, Clailson de. A estrutura física do PostgreSQL. **SQL Magazine,** São Paulo, ago 2009, pp. 44-50.

[7] VALE, Vinicius Aquino do. Aplicando técnicas de tuning para melhoria de desempenho em Banco de Dados. **SQL Magazine.** São Paulo, set 2009, pp. 38.

[8] OLIVEIRA, Edmar Welington. **Tecnologia de Discos**. Campinas, 2008. Disponível em <http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/065819-T.pdf>. Acessado em: 11 out 2009.

[9] TORRES, Gabriel; LIMA, Cássio. **Tudo que você precisa saber sobre discos rígidos ATA-66, ATA-100 e ATA-133**. Disponível em <http://www.clubedohardware.com.br/artigos/1055>. Acessado em: 21 out 2009.

[10] FARIA, Fábio Augusto. **Tecnologia de Discos Rígidos: IDE, SATA, SCSI e SAS.** Campinas, 2008. Disponível em <http://www.ic.unicamp.br/~ducatte/mo401/1s2008/T2/079734-t2.pdf>. Acessado em: 11 out 2009.

[11] PINTO, Marcelo Seabra. **Storage Area Network.** Rio de Janeiro, 2004. Disponível em <http://www.gta.ufrj.br/grad/04_1/san/>. Acessado em: 05 out 2009.

[12] PRESTON, W. Curtis. **Using SANs and NAS**. 1 ed. Sebastopol. O'Reilly, 2002.

[13] REITTER, Jörg. Fundamentos dos sistemas de arquivos com journaling. **Linux Magazine**, São Paulo, set 2004, pp. 19-22.

[14] SGI. **XFS: A high-performance journaling filesystem.** Disponível em <http://oss.sgi.com/projects/xfs>. Acessado em: 20 jun 2010.

[15] Sourceforge. **JFS Project Web Site.** Disponível em <http://jfs.sourceforge.net/>. Acessado em: 20 jun 2010

[16] Redhat. **Choosing an I/O Scheduler for Red Hat® Enterprise Linux® 4 and the 2.6 Kernel.** Disponível em <http://www.redhat.com/magazine/008jun05/features/schedulers/>. Acesso em 27 mai 2010.

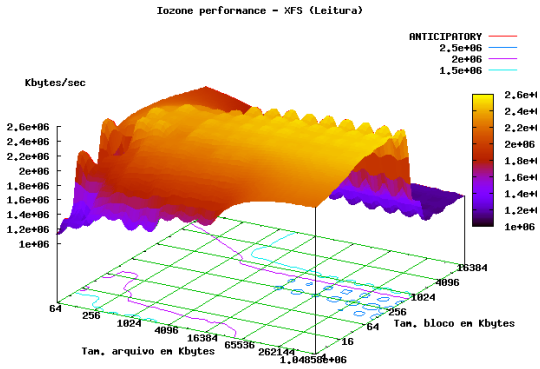
[17] ALVES, Maicon Melo. **Linux: Performance e Monitoramento.** 1 ed. Rio de Janeiro. Brasport, 2009.

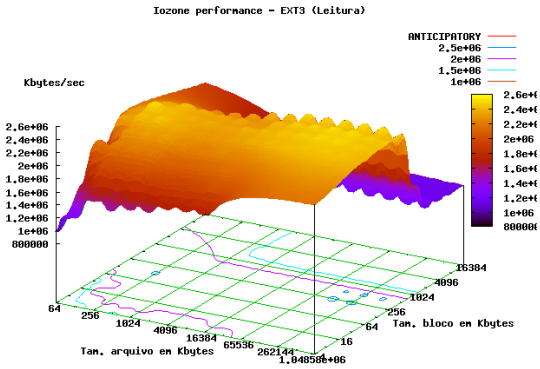
[18] UFRJ. **Block IO - The Block I/O Layer.** Disponível em <http://www.cos.ufrj.br/~vitor/aulas/COS773/>. Acesso em 27 mai 2010.

[19] IOzone. **IOzone Filesystem Benchmark.** Disponível em <www.iozone.org>. Acesso em 27 mai 2009.

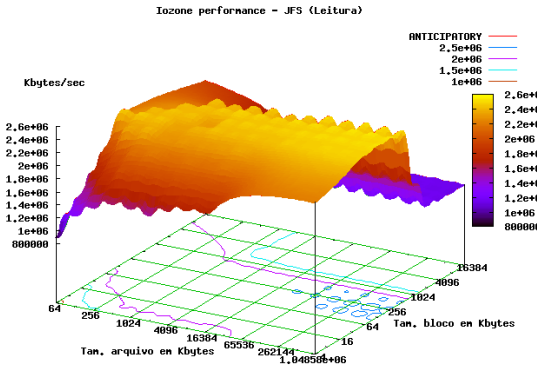
[20] SQL Magazine. **Um software para Benchmark.** Disponível em <www.devmedia.com.br/articles/viewcomp.asp?comp=3942>. Acesso em 27 mai 2009.

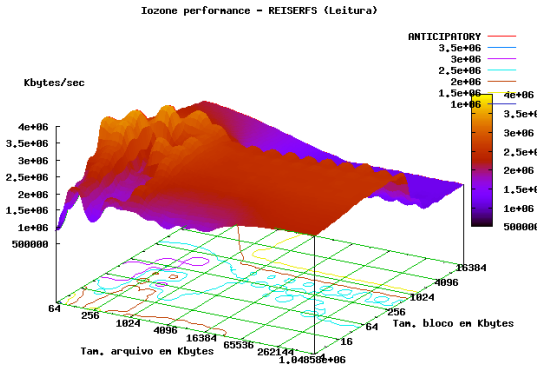
ANEXO I – PLOTAGEM DOS RESULTADOS OBTIDOS PELO IOZONE

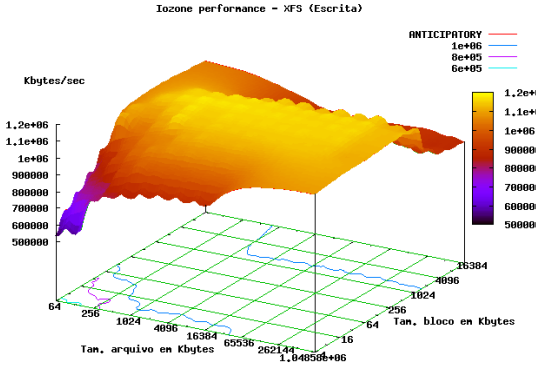


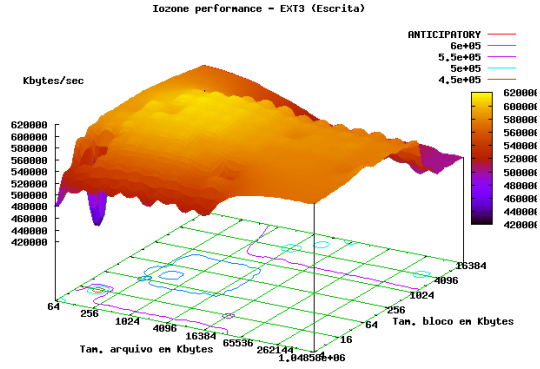


ANTICIPATORY - LEITURA

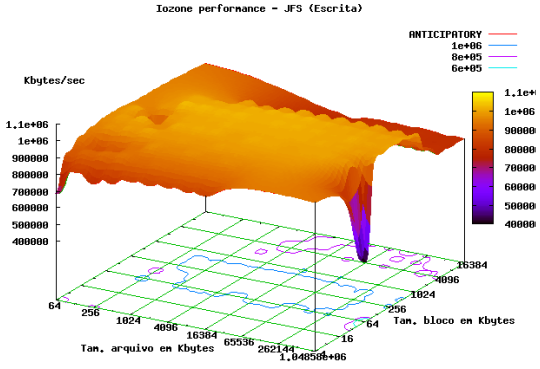


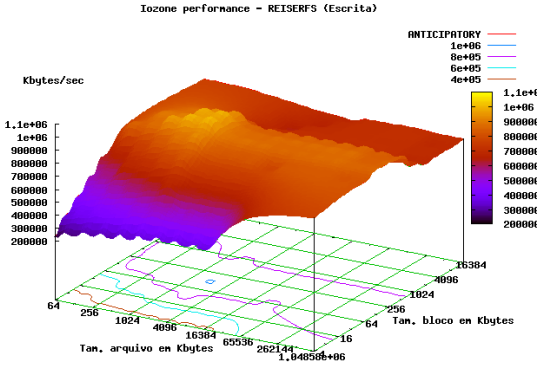


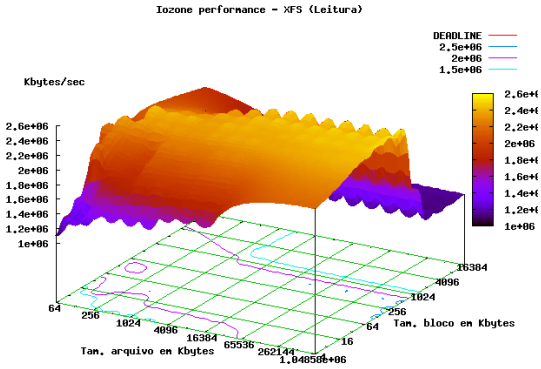


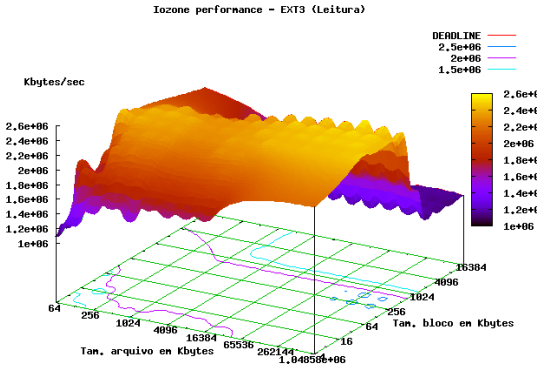


ANTICIPATORY - ESCRITA

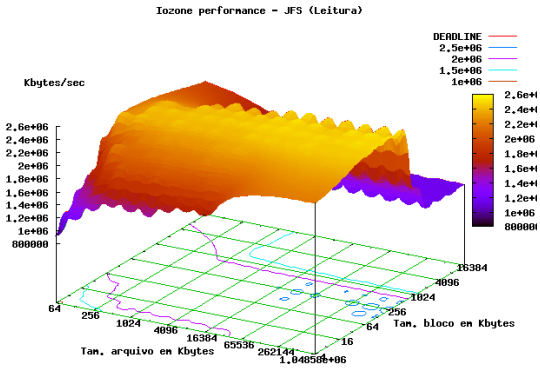


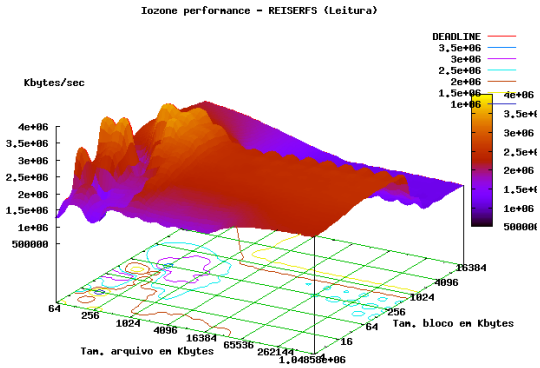


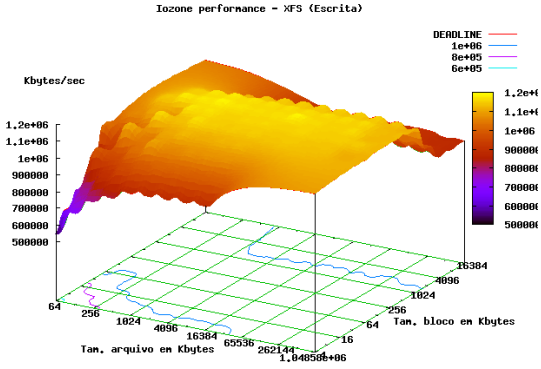


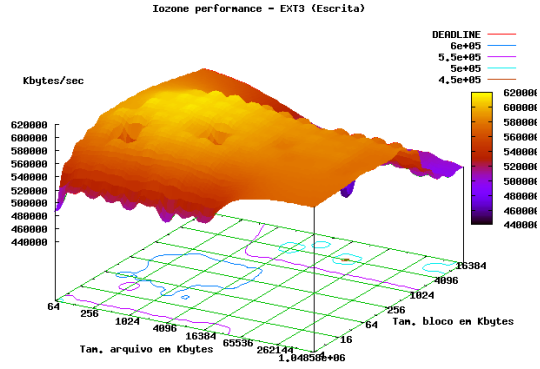


DEADLINE - LEITURA

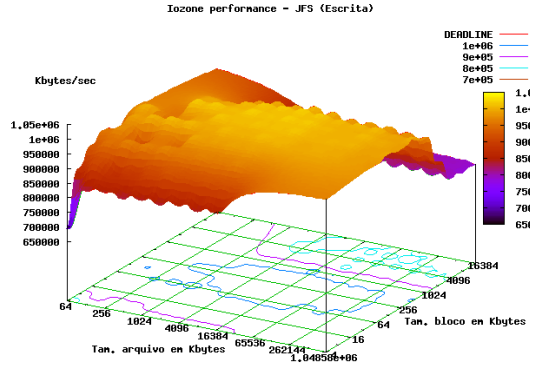


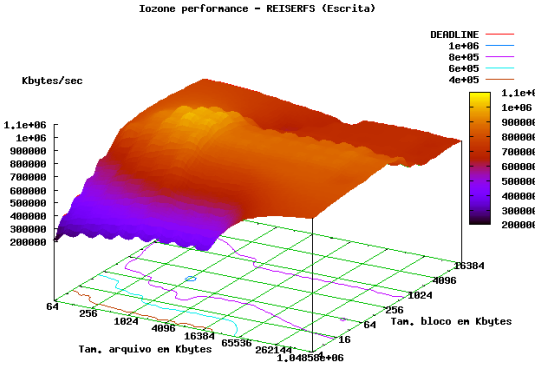


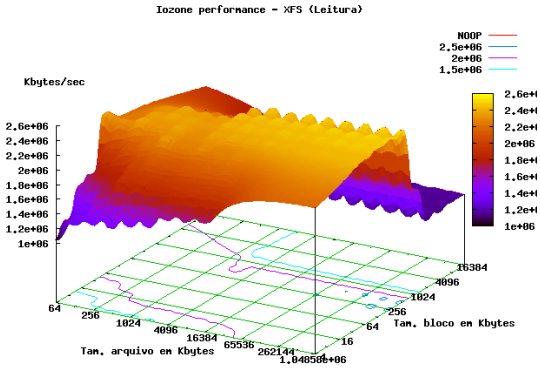


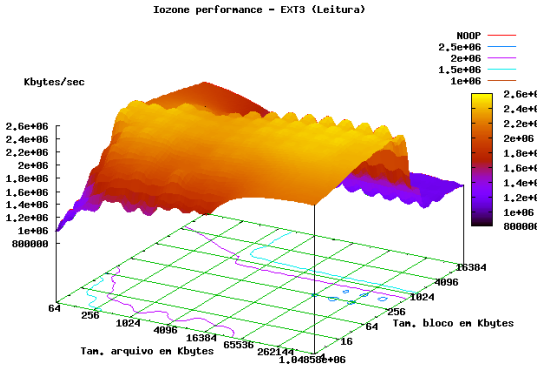


DEADLINE - ESCRITA

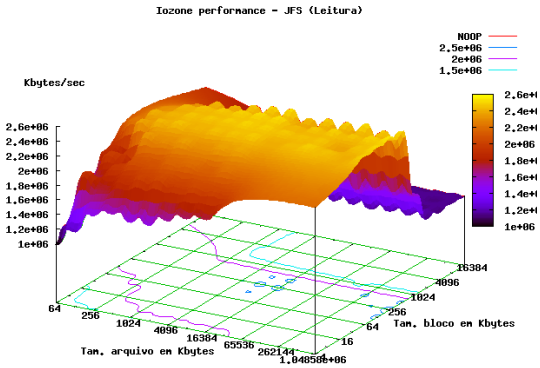


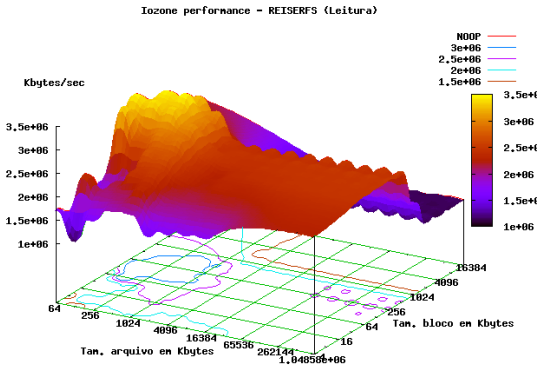


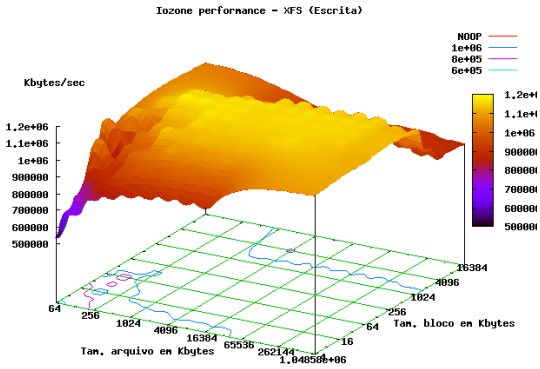


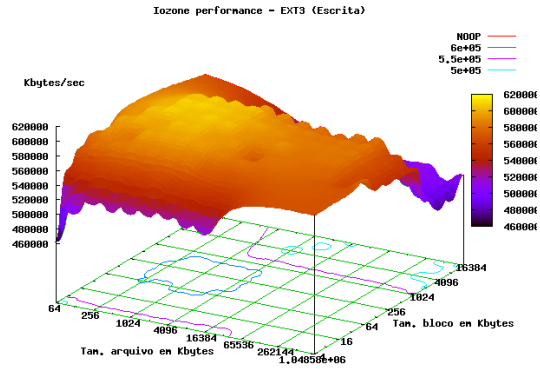


NOOP - LEITURA

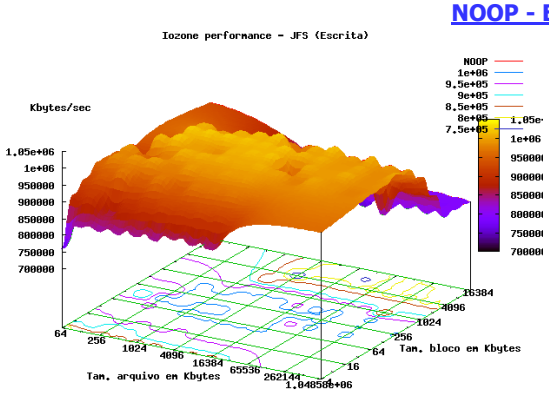


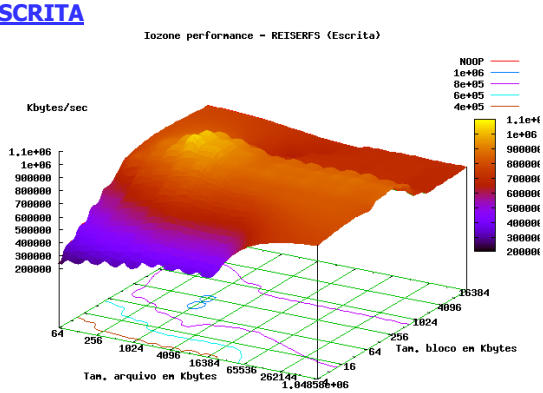


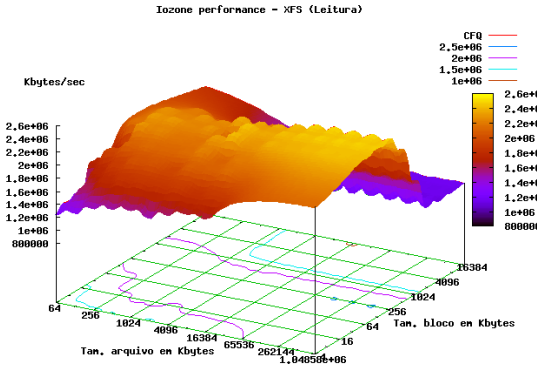


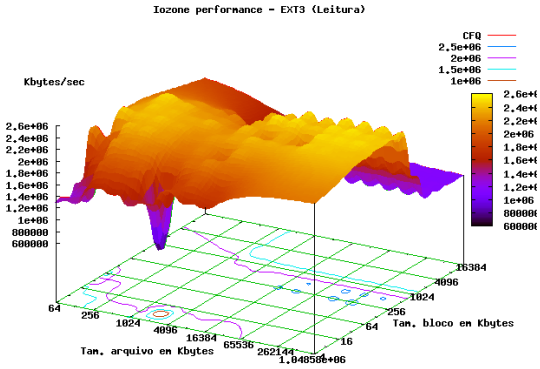


NOOP - ESCRITA

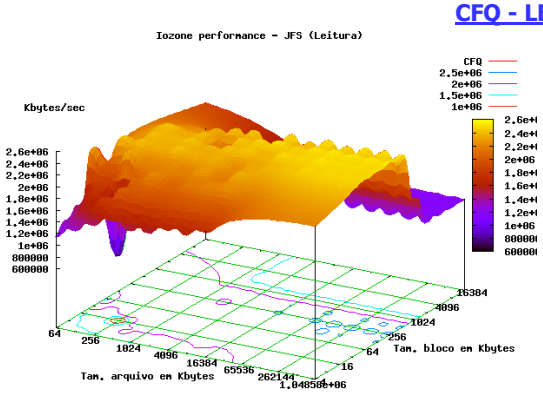


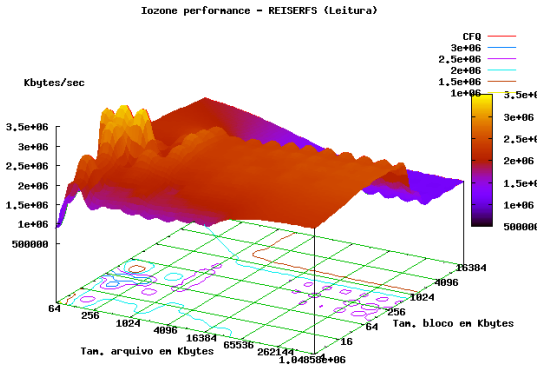


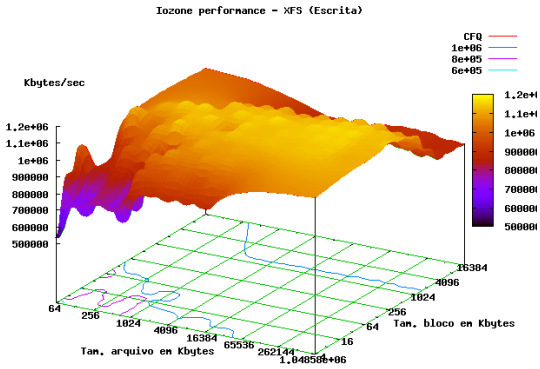


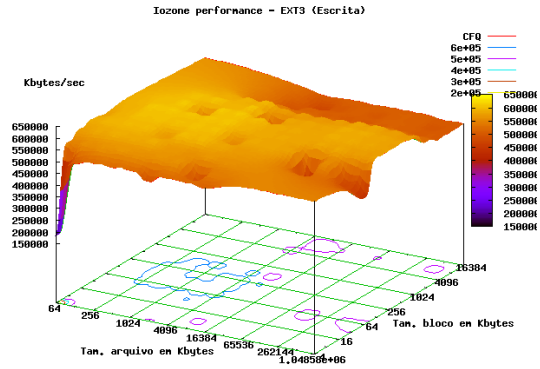


CFQ - LEITURA









CFQ - ESCRITA