

Sistemas de Arquivos Paralelos: Reduzindo o gargalo no acesso ao disco

Alfredo Goldman¹, Roberto Pires de Carvalho¹

¹Instituto de Matemática e Estatística
Universidade de São Paulo (USP) – São Paulo, SP – Brasil

{gold, carvalho}@ime.usp.br

***Abstract.** During the last years, the grown on the speed to access a hard disk drive have not increased at the same ratio found when talking about processors and networks. Therefore, many applications can't reach the full use of the processors, because they fatally have to wait the arrival of needed data before processing. A popular way to solve this is the adoption of parallel file systems, which use several disks to store data, which reduces the bottleneck caused by constantly accessing only one disk.*

In this study, we will show that a parallel file system, PVFS2 in the case, can be faster than a local file system, even having just one machine as a client.

***Resumo.** No decorrer dos últimos anos o aumento das velocidades de acesso aos discos rígidos não aumentou na mesma proporção que os aumentos das velocidades de processamento e transmissão de dados em redes. Com isso, muitas aplicações não atingem o pleno uso dos processadores, pois fatalmente têm que esperar os dados chegarem do disco para serem processados. Uma forma popular para resolver isso é a adoção de sistemas de arquivos paralelos, que por utilizarem vários discos para armazenar os dados, acabam reduzindo o gargalo provocado pelo acesso constante a apenas um disco.*

Neste estudo, mostraremos que um sistema de arquivos paralelo, no caso o PVFS2, pode ser mais rápido do que um sistema de arquivos local, mesmo no caso de termos como cliente apenas uma máquina.

1 Introdução

A velocidade de acesso aos arquivos a partir de seus meios físicos de armazenamento, como dispositivos magnéticos, óticos, entre outros, não evolui na mesma proporção que a velocidade dos processadores ou da transmissão de dados, sejam estes internamente à máquina (como no acesso à memória, registradores, etc) ou externamente (como no acesso às outras máquinas via rede). Podemos observar esse fato na figura 1, que compara a evolução de cada tecnologia ao decorrer dos anos, em uma escala logarítmica [de Carvalho 2005].

Isso torna o acesso aos arquivos um gargalo para muitas aplicações, especialmente para aquelas que buscam informações armazenadas remotamente. Para reduzir esse problema, no lugar de um sistema de arquivos remoto mais simples, costumam-se usar sistemas de arquivos paralelos, que são especializados em fornecer alto desempenho no acesso aos dados para um número crescente de clientes dentro de uma rede local. Essa solução também melhora o desempenho de aplicações distribuídas.

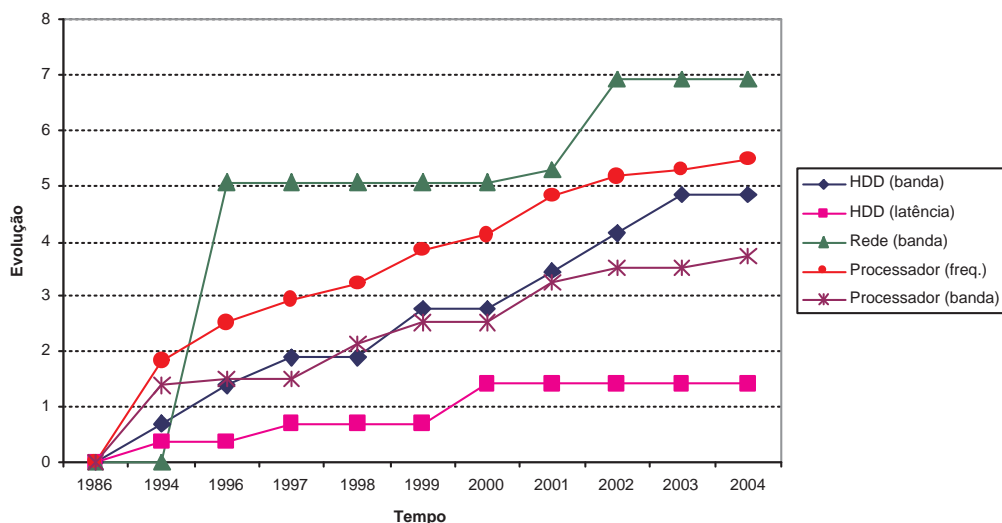


Figura 1. Evolução tecnológica normalizada

Neste trabalho mostramos que esta mesma solução pode ser usada por aplicações que manipulam uma grande quantidade de dados em uma só máquina, sejam elas paralelas ou não, para resolver o gargalo provocado pelo sistema de arquivos local.

Esse artigo está organizado da seguinte forma: na seção 2 apresentaremos o PVFS2, sistema de arquivos paralelo usado para validar nossa proposta. Em seguida, na seção 3, comentaremos sobre a motivação de se comparar o PVFS2 com o sistema de arquivos local Ext3. Abordaremos o ambiente de testes usado na seção 4, e na seção 5 detalharemos como eles foram realizados. Na seção 6 comentaremos sobre os resultados encontrados. Por fim, na seção 7, apresentaremos as conclusões obtidas.

2 PVFS e PVFS2

Atualmente, os aglomerados de PCs têm se tornando cada vez mais populares para aplicações paralelas. Com isso, a demanda por software para esse tipo de plataforma tem crescido muito. Já podemos encontrar todo tipo de software para o ambiente de computação paralela, como sistemas operacionais confiáveis, sistemas de armazenamento de dados local, e sistemas de envio de mensagens.

O *Parallel Virtual File System* [Carns et al. 2000, Haddad 2000] se encaixa na área de sistemas de E/S paralelo, pois é um sistema de arquivos distribuído desenvolvido para prover alto desempenho e escalabilidade paralela para aglomerados de PCs linux. Em geral, o PVFS promete 4 características:

- Um espaço de nomes consistente para todo o aglomerado;
- Acesso transparente para programas e aplicações já existentes, sem a necessidade de recompilá-los;
- Distribuição física de dados dos arquivos em múltiplos discos e múltiplos nós;
- Alto desempenho no acesso em modo usuário.

Quando uma aplicação acessa o PVFS para obter os dados de um determinado arquivo, várias máquinas serão acessadas, de forma transparente. Por acessar várias

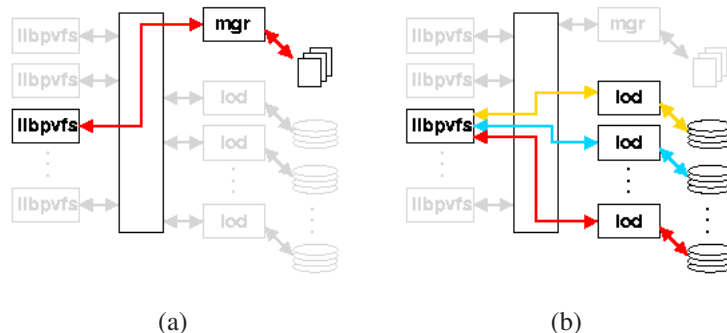


Figura 2. Clientes acessando o PVFS

máquinas, utiliza-se de vários caminhos pela rede para chegar aos respectivos discos em que os dados estão armazenados. Isso elimina o gargalo de E/S quando se tem toda a informação armazenada em uma só máquina, distribuindo a carga e aumentando o potencial total da banda para múltiplos clientes.

2.1 Os componentes do PVFS

O **servidor de meta-dados (mgr** na figura 2(a)) gerencia todos os dados que constituem informações sobre o arquivo, como seu nome, sua localização na hierarquia de diretórios, seu dono, seus atributos, e como seus dados estão distribuídos entre os vários nós de dados do sistema. Ele realiza todas as operações sobre os meta-dados dos arquivos atômicamente, evitando assim ter que implementar esquemas complexos de concorrência, *locks*, consistência, etc, para múltiplos acessos simultâneos.

O **servidor de dados (iod** na figura 2(b)) gerencia o armazenamento do conteúdo dos arquivos, bem como a recuperação dos mesmos, nos discos locais conectados nos nós. Esse servidor grava os dados dos arquivos do PVFS em um sistema de arquivos local, através de chamadas a funções tradicionais, como *read()*, *write()* e *mmap()*. Isso significa que pode-se usar qualquer tipo de sistema de arquivos local, como Ext2, Ext3 ou Reiser [von Hagen 2003], por exemplo. Adicionalmente é possível usar suporte a RAID para que cada nó possua tolerância a falhas de disco de forma transparente e confiável para todo o sistema.

A **API nativa do PVFS** possibilita acesso em modo usuário aos servidores do PVFS. Esta biblioteca, chamada de *libpvfs*, cuida das operações necessárias para mover dados entre os clientes e servidores, mantendo-as transparentes para o usuário. Para operações que necessitam de meta-dados, a biblioteca se comunica diretamente com o servidor de meta-dados, conforme figura 2(a). Para acesso aos dados dos arquivos, o servidor de meta-dados é deixado de lado e os servidores de dados são acessados diretamente, conforme figura 2(b).

O **suporte no kernel do linux para o PVFS** provê as funcionalidades necessárias para se poder usar o comando *mount* nos clientes. Isso permite acesso aos arquivos do PVFS sem necessitar alterações nas aplicações ou programas já existentes. Esse suporte não é necessário para se usar o PVFS, mas ele traz uma enorme conveniência para a interatividade com o sistema. Ele se utiliza da biblioteca *libpvfs* para realizar essas operações.

2.2 PVFS2

O PVFS2 é uma reimplementação das melhores características da primeira versão do PVFS, usando uma nova arquitetura para torná-lo mais flexível, aumentando a qualidade de serviço disponibilizada. Isso possibilitou a implementação de novas características, técnicas e inovações que foram sendo discutidas e requisitadas durante as correções de defeitos da primeira versão.

Algumas das novidades [Ross et al. 2002, PVFS2 Development Team 2003] implementadas (ou ainda a serem implementadas) no PVFS2 são:

- Suporte modular para múltiplos protocolos de rede e armazenamento;
- Acesso a dados estruturados não-contínuos;
- Distribuição de dados de forma flexível, utilizando-se de padrões de acesso aos dados;
- Servidores de meta-dados distribuídos, reduzindo um possível gargalo;
- Mapeamento flexível de referências de arquivos para determinados servidores;
- Redundância de dados e meta-dados.

Dos itens acima, já temos disponível (considerando a versão 1.3.0 do PVFS2) o suporte modular para protocolo de rede e armazenamento de dados e a possibilidade de se utilizar múltiplos servidores de meta-dados, além de mudanças estruturais na arquitetura do projeto visando aprimorar o tempo de resposta entre clientes e servidores.

3 PVFS2 vs. Ext3

O nosso principal objetivo neste artigo é propor uma comparação entre uma única máquina, como cliente, usando o sistema de arquivos paralelo PVFS2, e o sistema de arquivos local Ext3.

À primeira vista pode parecer estranho compará-los, já que têm propósitos distintos, porém, caso haja largura de banda suficiente na rede para o tráfego dos dados, um sistema de arquivos paralelo pode ser mais rápido que um sistema de arquivos local. Para deixar mais clara a idéia, a figura 3 ilustra uma rede muito rápida interligando servidores e clientes que possuem dispositivos de armazenamento notavelmente mais lentos.

Assim, muitas requisições enviadas ao PVFS2, mesmo que vindas de um único cliente, utilizam toda a banda disponível dos discos dos servidores, já que não têm a rede como gargalo, enviando o resultado para o cliente rapidamente. Comparando essa banda agregada com a banda disponível pelo sistema de arquivos local, percebemos a clara vantagem em se utilizar o PVFS2.

Este foi o assunto principal de [de Carvalho 2005], porém, por não possuímos um aglomerado conectado em uma rede de alta velocidade, os resultados iniciais [de Carvalho 2005] foram obtidos através da redução da velocidade dos discos locais de todas as máquinas envolvidas. Dessa forma, foi possível simularmos uma rede de dados mais rápida que o sistema de armazenamento local.

A partir dos resultados dessa simulação, estimamos o desempenho do PVFS2 em uma rede conectada a 1Gbit/s como sendo de aproximadamente 109MB/s, considerando que a velocidade máxima da rede seria de 125MB/s, e que não haveria nenhum outro gargalo a não ser o disco.

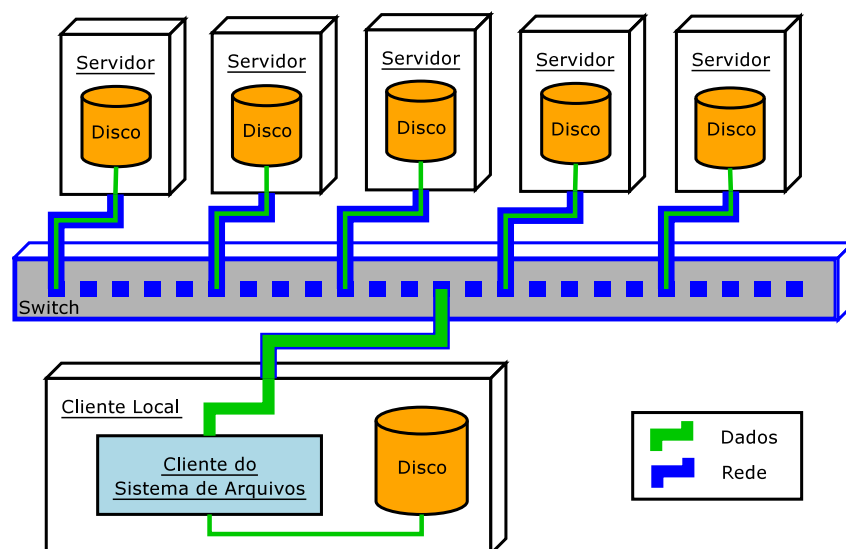


Figura 3. Sistemas de arquivos paralelos vs. locais

Máquina	Modelo do processador	Memória	Modelo do disco
Cliente 1	Intel Pentium 4 CPU 2.80GHz	514692 kB	ST340014A
Cliente 2	Intel Pentium 4 CPU 1.80GHz	482500 kB	ST340014A
Serv. Meta-Dados	AMD Athlon XP 1800+	1035576 kB	Maxtor 6E040L0
Serv. Dados 1	AMD Athlon 1.65GHz	483260 kB	SAMSUNG SP0411N
Serv. Dados 2	AMD Athlon 1.65GHz	483260 kB	SAMSUNG SP0411N
Serv. Dados 3	AMD Athlon 1.65GHz	483260 kB	SAMSUNG SP0411N
Serv. Dados 4	AMD Athlon 1.65GHz	483260 kB	SAMSUNG SP0411N
Serv. Dados 5	Intel Pentium 4 CPU 2.66GHz	514692 kB	ST340014A
Serv. Dados 6	Intel Pentium 4 CPU 1.70GHz	1035576 kB	Maxtor 6E040L0
Serv. Dados 7	AMD Athlon XP 1800+	1035576 kB	Maxtor 6E040L0
Serv. Dados 8	Intel Pentium 4 CPU 2.66GHz	514692 kB	ST340014A

Tabela 1. Configuração das máquinas usadas nos testes

Dispondo agora de uma rede gigabit, realizamos testes comparativos de desempenho entre o PVFS2 e o Ext3, onde verificaremos quais condições são necessárias para que um sistema seja mais eficiente que o outro.

4 Ambiente

Para a realização dos testes, contamos com um aglomerado de 12 máquinas, todas conectadas entre si através de placas de rede Gigabit, ligadas a um *switch* de mesma velocidade. A distribuição Linux usada nas máquinas é baseada em RedHat, com *kernel 2.6.12*. Como essas máquinas não possuíam uma configuração homogênea, o tipo do processador, quantidade de memória e modelo do disco de cada uma pode ser encontrada na tabela 1. Em todos os testes não temos clientes atuando como servidores.

Na figura 4 temos a velocidade média de leitura dos dados a partir dos discos¹ e a

¹Obtida usando-se a ferramenta *hdparm*.

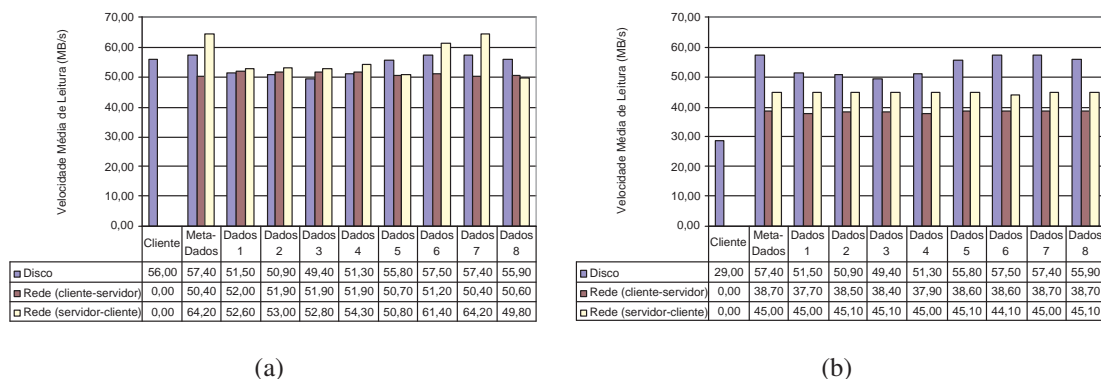


Figura 4. Velocidade dos discos e da rede no (a) cliente 1 e no (b) cliente 2, acessando os servidores de dados

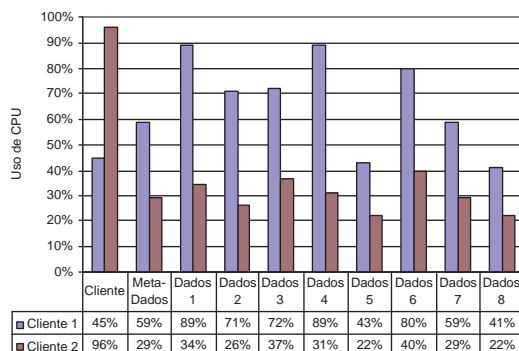


Figura 5. Uso de CPU observado durante a medição da velocidade da rede dos clientes 1 e 2

velocidade média de transferência de dados entre as máquinas². Esta última depende da direção para onde os dados estão indo, cliente-servidor ou servidor-cliente. No sentido cliente-servidor, a velocidade não varia de acordo com o servidor, o que mostra que o cliente é um limitador da velocidade de envio dos dados. No sentido servidor-cliente, o cliente recebe os dados mais rapidamente do que quando envia, e dependendo do servidor, a velocidade pode aumentar mais que em outros. Isso acontece devido às diferenças entre as máquinas, pois temos algumas mais robustas que outras. Na figura 5 podemos ver que essa conclusão faz sentido, pois temos algumas máquinas cuja CPU esteve no limite de utilização durante esses testes de velocidade da rede, enquanto que outras não.

Na figura 4(b) podemos notar que as velocidades da rede se mantêm independentemente dos servidores acessados, o que já não ocorre na figura 4(a). Isso ocorre devido ao cliente 1 ser uma máquina mais nova e com maior poder de processamento se comparada ao cliente 2. Na figura 5 podemos ver que o cliente 2 usa muito mais CPU que o cliente 1 durante a análise da velocidade da rede, tornando-o um gargalo em todos os acessos ao servidor. Já o cliente 1, em alguns momentos ele foi o gargalo, em outros o servidor foi o gargalo.

O objetivo de se ter dois clientes é poder comparar um cliente mais lento com

²Obtida usando-se a ferramenta *iperf*.

um mais rápido, e verificar o impacto disso no acesso ao PVFS2. Em nenhum momento temos os dois clientes acessando o PVFS2 de forma concorrente.

Instalamos o PVFS2 versão 1.3.0 em todos os servidores, seguindo as instruções de instalação do *Quick Guide*, disponível no site oficial do sistema. Nenhuma configuração foi alterada, com relação à instalação padrão. Nos clientes, instalamos a interface UNIX para acesso ao PVFS2. Tal interface é um módulo do *kernel* que permite ao usuário montar o PVFS2 e utilizá-lo como um sistema de arquivos comum, de forma transparente para as aplicações.

5 Os Testes

Para compararmos o comportamento do PVFS2 e do Ext3, preparamos alguns testes que realizam acesso concorrente a arquivos armazenados nesses dois sistemas de arquivos, levando em consideração a combinação das seguintes variáveis:

- **Tipo do teste:** Leitura ou escrita;
- **Tamanho do arquivo:** 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 256MB, 512MB ou 1GB;
- **Tamanho do bloco de leitura ou escrita:** 1KB, 4KB, 16KB, 64KB, 256KB ou 1MB;
- **Quantidade de threads no cliente acessando os servidores:** 1, 2, 4, 8, 16 ou 32;
- **Cache:** Dados no cache do servidor ou dados em disco;
- **Sistema de arquivo:** Ext3 ou PVFS2 com 1, 2, 3, 4, 5, 6, 7 ou 8 servidores de dados;

5.1 PSplit

Para a realização dessas tomadas de tempo, criamos o **PSplit** (ou *Parallel Split*, que tem como principal tarefa criar vários arquivos de mesmo tamanho, a partir do conteúdo de um arquivo maior, utilizando-se de várias threads. O PSplit também pode ser usado para somente leitura (sem escrita), assim como gerar dados aleatórios e gravá-los em vários arquivos, sempre utilizando-se de múltiplas threads.

Ao final do processamento, o PSplit informa o tempo total gasto para ler ou gravar todos os dados, além da velocidade de leitura média em bytes/s, entre a abertura do arquivo até o último byte se lido ou gravado.

5.2 Threads e Processos

Quanto ao uso de várias threads ou vários processos, o sistema operacional Linux os trata de forma muito parecida nas trocas de contexto, o suficiente para não afetar o desempenho dos nossos testes. Mais informações sobre diferenças entre processos e threads podem ser encontradas em [Aas 2005].

5.3 Influência do Cache

Tanto o Ext3 como o PVFS2 se utilizam de cache para melhorar o desempenho no acesso repetido aos mesmos dados, que pode ocupar toda a memória física disponível. Além disso, o PVFS2 armazena seus dados em disco no servidor, utilizando-se do Ext3 para isso, o que gera mais um nível de cache.

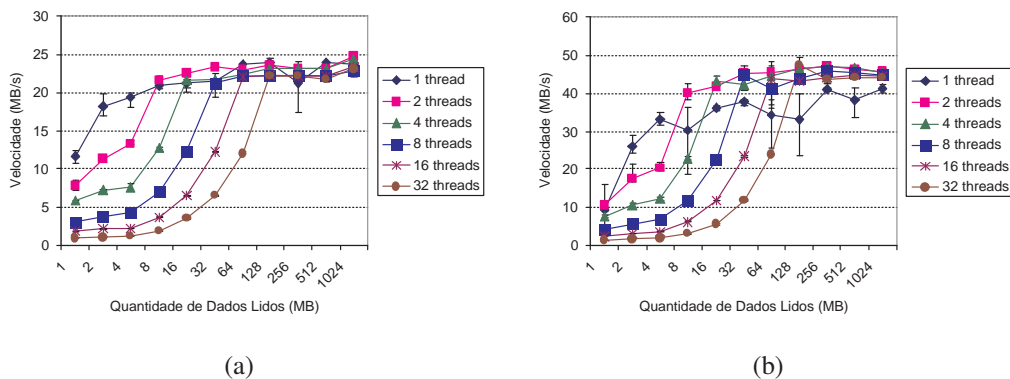


Figura 6. Cliente 1 lendo dados em blocos de 64KB do PVFS2 com (a) 1 e (b) 2 servidores de dados, variando a quantidade de dados lidos e número de threads

Isso pode influenciar os resultados de nossos testes, pois caso os dados a serem lidos já estejam no cache local do servidor, não haverá acesso ao disco, e caso eles estejam no cache local do cliente, não haverá nem acesso à rede.

Dessa forma, realizamos dois tipos de testes: limpando os dados do cache antes de acessar o sistema de arquivos e sem limpar os dados do cache, para podermos analisar a influência de tal mecanismo sobre o desempenho no acesso a grandes quantidades de dados.

5.4 Gráficos

Para minimizar os erros amostrais, cada um dos testes foi repetido 10 vezes. Ao final, calculamos a média dos resultados como o valor a ser usado nos gráficos. Neles, existem também barras de erro que foram calculadas a partir dos valores mínimo e máximo encontrados.

6 Resultados

Analisando os resultados obtidos em nossos testes, pudemos constatar que, enquanto algumas das variáveis adotadas influenciaram de forma considerável os resultados, outras foram praticamente irrelevantes para decidirmos quando o PVFS2 é melhor que o Ext3, em termos de desempenho.

Dentre elas, o tamanho do bloco de leitura não chegou a influenciar no desempenho de ambos os sistemas de arquivos de forma considerável, tanto para leitura quanto para escrita. Notamos apenas que o uso de blocos de 64KB nos dá um desempenho um pouco melhor (cerca de 5%) do que usando outros valores.

A figura 6(a) mostra o comportamento do PVFS2 com apenas um servidor de dados sendo acessado por múltiplas threads, enquanto a figura 6(b) mostra a mesma situação, mas com 2 servidores de dados. Nesses testes, com apenas 1 thread, tivemos alguns picos de desempenho bom e ruim muito acentuados, o que acabou influenciando nossa média (conforme podemos perceber pelas barras de erro). Acreditamos que isso seja causado exclusivamente pelo comportamento do disco dos servidores, que por estarem respondendo a requisições de apenas uma thread, podem ficar ociosos tempo suficiente para saírem da posição ideal de leitura dos dados requisitados, pois com apenas uma thread ela pede

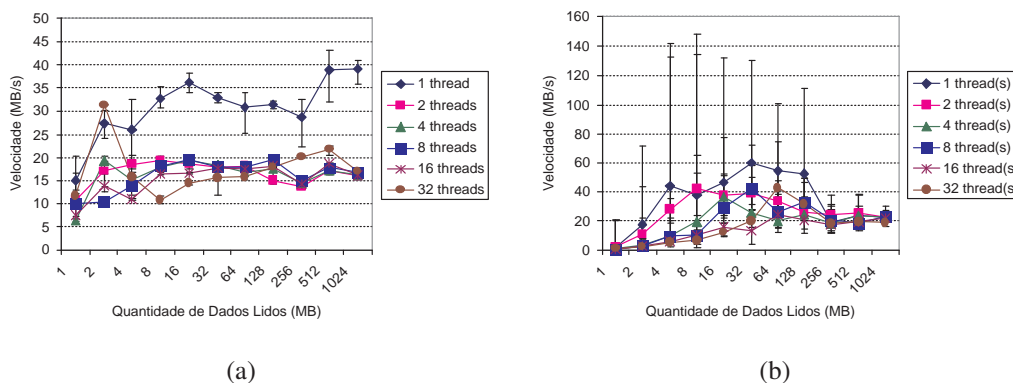


Figura 7. Cliente 1 (a) lendo e (b) escrevendo dados no Ext3, em blocos de 64KB, variando a quantidade de dados e número de threads

dados a apenas um servidor por vez. Quando a concorrência aumenta, eles não ficam ociosos e precisam ler dados a todo instante, aumentando a vazão.

Na mesma figura, vemos que o comportamento do PVFS2 com muitas threads e poucos dados nos mostra o que havia sido observado em [de Carvalho 2005]: a sobrecarga que cada thread de nossos testes sofre antes de começar a ler os dados é muito grande. Nisso inclui contactar o servidor de meta-dados, requisitar informações sobre a localização dos dados e conectar no servidor de dados. Notamos que para que uma thread do PSplit atinja seu máximo desempenho com o PVFS2, ela deve ler pelo menos 4MB. Já na escrita o PVFS2 melhora o desempenho gradualmente conforme a quantidade de dados a serem escritos aumenta [de Carvalho 2005].

O Ext3, nessa mesma situação, melhora o desempenho aos poucos conforme a quantidade de dados a serem lidos aumenta, conforme vemos na figura 7(a). Porém, ao aumentarmos a quantidade de threads, o desempenho cai. Na escrita (figura 7(b)) ocorre o mesmo, mas não de forma homogênea, devido à implementação da escrita do Ext3, que primeiramente envia os dados para a memória cache, evitando bloquear o escritor, para depois realizar a descarga para o disco.

Para simplificar nossa análise, e dadas as conclusões às quais chegamos, fixamos o tamanho do bloco de dados de leitura e escrita em 64KB, e a quantidade de dados a serem lidos em 1GB, para todos os demais testes. Com isso, podemos comparar, no mesmo gráfico, Ext3 e PVFS2, variando agora a quantidade de servidores de dados.

Uma das consequências dessa decisão é a pouca relevância do uso de cache do Ext3 nos nossos testes de leitura de dados, pois temos mais dados para serem lidos do que memória cache disponível. Já o PVFS2, por ter vários servidores e por distribuir o conteúdo do arquivo entre eles, possui uma maior quantidade de memória cache distribuída. Mas, mesmo assim, a diferença entre a média dos resultados, usando-se ou não o cache, variou em torno de 1%, conforme podemos observar ao comparar a figura 8 com a figura 9.

Com relação ao aumento da quantidade de servidores de dados e da quantidade de threads (figura 9(a)), percebemos que existe um ganho considerável no desempenho, até atingindo o limite da velocidade da rede (verificado na figura 4(a)). Na figura 9(b) temos

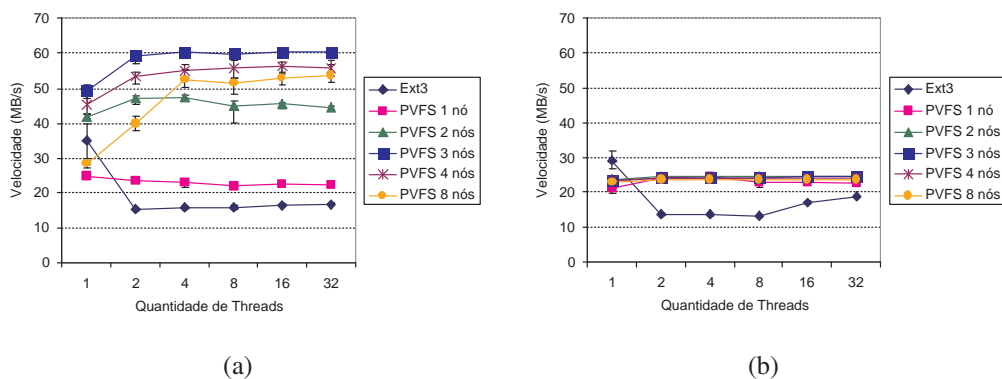


Figura 8. Comparação entre as várias configuração do PVFS2 com o Ext3, sem limpar os dados da memória cache. Em (a) temos o cliente 1 e em (b) o cliente 2, lendo um arquivo de 1GB, com blocos de 64KB

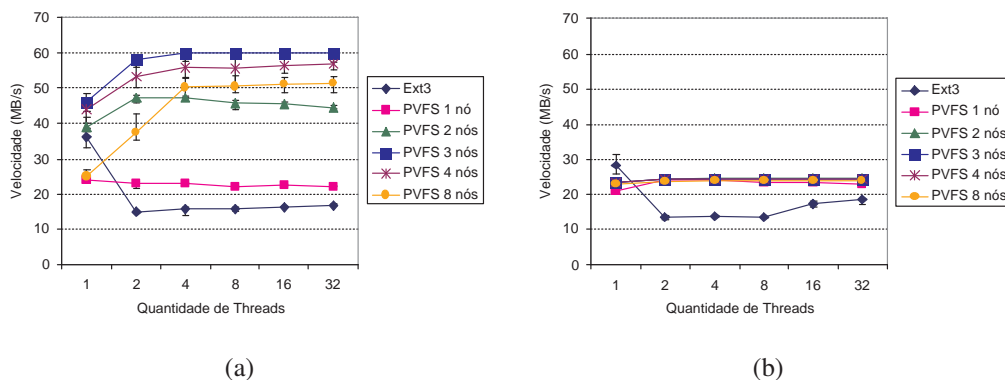


Figura 9. Comparação entre as várias configuração do PVFS2 com o Ext3, limpando-se os dados da memória cache. Em (a) temos o cliente 1 e em (b) o cliente 2, lendo um arquivo de 1GB, com blocos de 64KB

o mesmo caso, porém usando o cliente 2 para acesso ao sistema de arquivos. Note que o desempenho é bem inferior se comparado com o cliente 1. Isso já era esperado, pois por possuir menor poder de processamento, o cliente 2 não tem uma resposta rápida para tratar os dados que vêm pela rede, conforme verificado nas figuras 4(b) e 5, onde a simples medição da velocidade da rede já foi limitada pela CPU.

Uma observação importante sobre os gráficos da figura 9, além de alguns outros, é que omitem os resultados do PVFS2 com 5, 6 e 7 servidores. Isso foi necessário pois, nessas configurações, o desempenho foi muito próximo do PVFS2 com 8 servidores, que acaba “escondendo” o resultado por trás de suas retas. Além disso, ao usarmos 3 servidores de dados temos o melhor desempenho dos nossos testes. Isso aconteceu devido ao nosso arranjo de servidores (tabela 1), pois devido à rede possuir máquinas muito heterogêneas, tentamos colocar aquelas que tiveram melhor desempenho de disco e rede como sendo os primeiros servidores.

As figuras 10 e 11 representam o uso de CPU pelos clientes e pelos servidores durante os testes de leitura de dados. Perceba-se que para ambos os casos, conforme

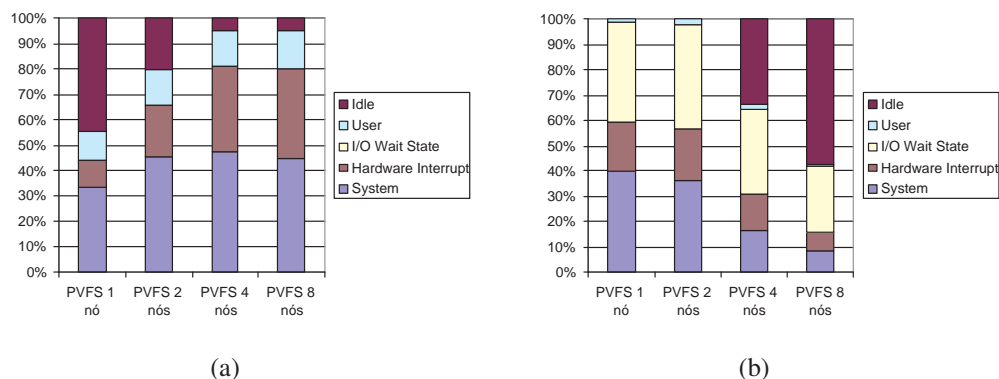


Figura 10. Uso observado da CPU pelo (a) cliente 1 e (b) servidores durante acesso ao PVFS2

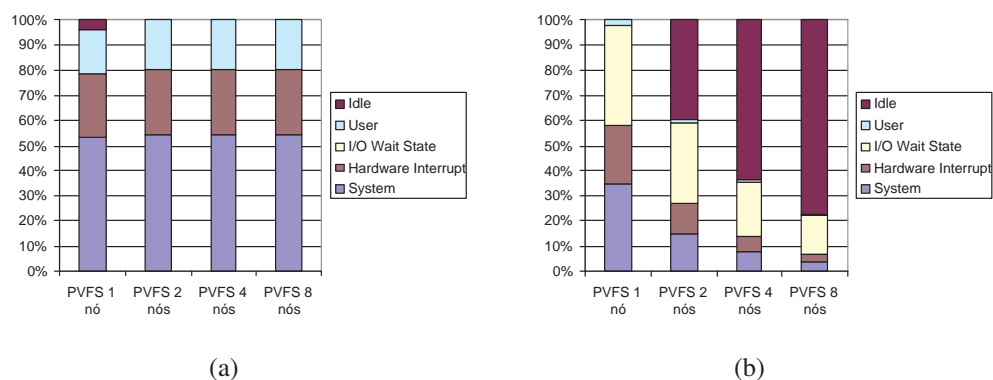


Figura 11. Uso observado da CPU pelo (a) cliente 2 e (b) servidores durante acesso ao PVFS2

aumentamos a quantidade de servidores de dados, o uso de CPU dos servidores diminui e o dos clientes aumenta. Isso se deve a termos mais servidores, que são acessados por apenas um cliente, de forma intercalada, gerando uma distribuição de carga. Assim, os servidores deixam de ser gargalo e o próprio cliente se torna o limitante para atingir um desempenho maior. O uso de CPU durante o acesso ao Ext3 ficou em 100% em ambos os clientes testados, sendo que aproximadamente 95% da CPU estava em *I/O Wait State*, ou seja, aguardando dados do barramento (no caso, vindos do disco).

Comparando o Ext3 com o PVFS2 na figura 9, notamos que quando o Ext3 é acessado por apenas uma thread, de forma seqüencial, seu desempenho é muito bom. Porém, ao usarmos 2 threads, o desempenho do Ext3 degrada muito, enquanto que o PVFS2 melhora de forma considerável, conforme aumentamos a concorrência, até se estabilizar, quando configurado com mais de um servidor de dados.

Já na escrita, que pode ser observada na figura 12, o desempenho do Ext3 se apresenta melhor do que na leitura (devido à escrita em cache, para depois passar os dados para o disco). Como o PVFS2 necessita enviar os dados pela rede (pois o cache de escrita está no servidor), e o acesso ao servidor de meta-dados é maior devido à necessidade de se criar o arquivo, há uma perda de desempenho grande com relação à leitura. Mas

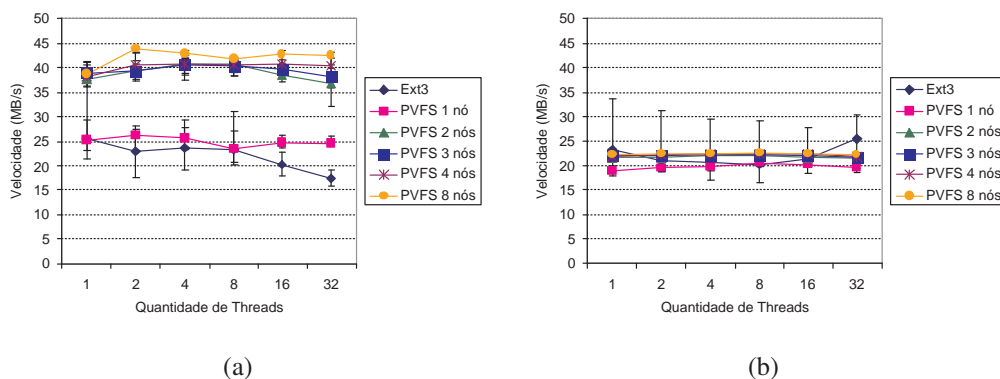


Figura 12. Comparação entre as várias configuração do PVFS2 com o Ext3. Em (a) temos o cliente 1 e em (b) o cliente 2, escrevendo 1GB de dados, com blocos de 64KB, em tantos arquivos quanto threads

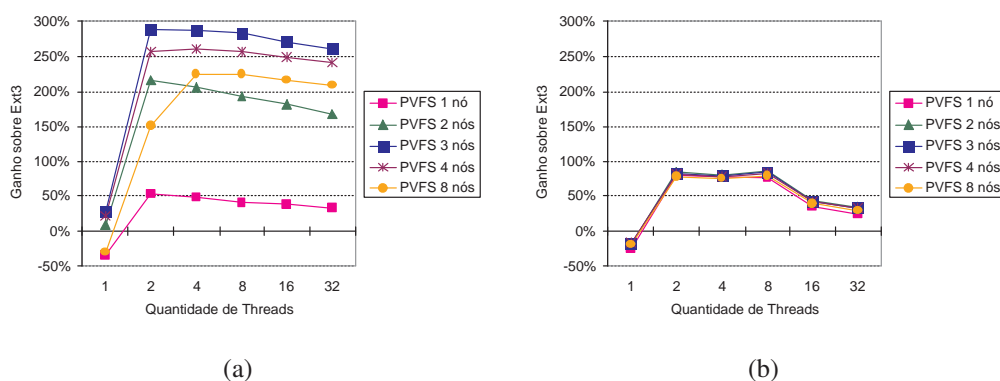


Figura 13. Ganho percentual do PVFS2 sobre o Ext3 usando o (a) cliente 1 e o (b) cliente 2, ao aumentarmos o acesso concorrente e o número de servidores de dados, ao ler 1GB de dados em blocos de 64KB

mesmo com essa melhora do Ext3 e piora do PVFS2, este é ainda mais eficiente que aquele também na escrita.

7 Conclusões dos Experimentos

A partir da análise dos resultados da seção anterior podemos concluir que o PVFS2 realmente pode ser mais eficiente que o Ext3. Na figura 13, montada a partir dos resultados já analisados, podemos constatar mais facilmente o quanto o Ext3 é menos eficiente quando temos um maior nível de concorrência no acesso. A diferença de velocidade chega a ser quase 3 vezes maior ao usarmos o PVFS2.

Porém, até agora comparamos somente a velocidade média do tráfego de dados de ambos os sistemas de arquivos, mas não a eficiência deles para usar os meios de armazenamento e tráfego de dados. Na figura 14 temos uma comparação entre eles com relação ao aproveitamento da velocidade média agregada dos discos, isto é, podemos verificar a eficiência no aproveitamento da velocidade disponibilizada pelos discos dos servidores. Podemos notar que o aproveitamento do Ext3 cai drasticamente ao aumentarmos o nível de concorrência, enquanto que o PVFS2 se mantém, embora não melhore muito.

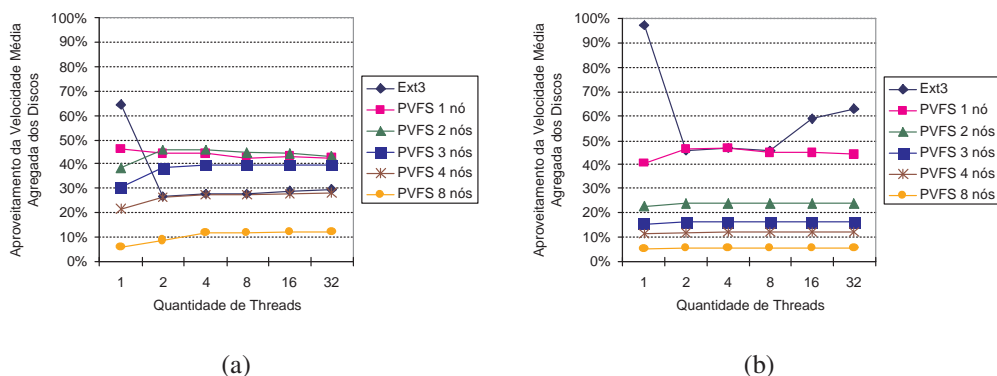


Figura 14. Aproveitamento da velocidade média agregada disponibilizada pelos discos dos servidores, para o (a) cliente 1 e o (b) cliente 2, lendo 1GB de dados do PVFS2, em blocos de 64KB

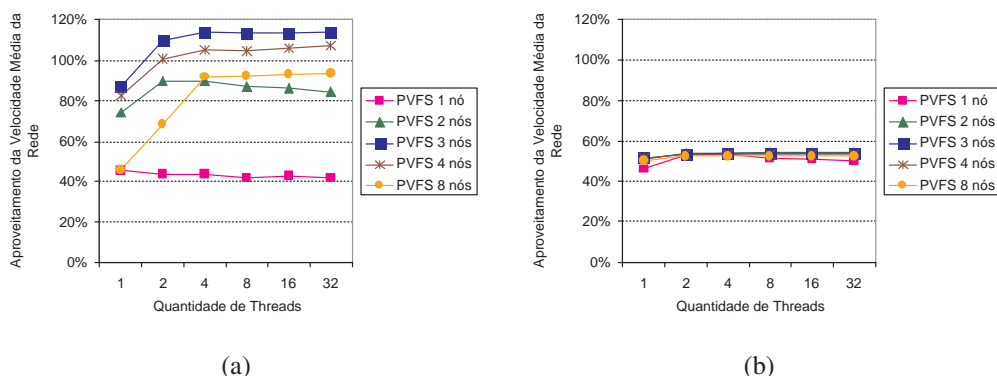


Figura 15. Aproveitamento da velocidade média disponibilizada pela rede, para o (a) cliente 1 e o (b) cliente 2 lendo 1 GB de dados do PVFS2, em blocos de 64KB

Com relação ao aproveitamento da velocidade da rede, vemos na figura 15 que ambos os clientes PVFS2 usados em nossos testes estão limitados pela velocidade máxima verificada da rede, que, por sua vez, está limitada pela CPU de uma das pontas da transmissão dos dados. Em alguns casos o desempenho do PVFS2 ultrapassa esse limite, o que pode acontecer já que a velocidade máxima teórica da rede (125MB/s) é muito maior do que a verificada (veja figuras 4(a) e 4(b)).

8 Conclusão

Dessa forma, concluímos que o gargalo do acesso aos dados do disco foi resolvido com o PVFS2, mesmo usando apenas um cliente. Como os servidores são acessados aleatoriamente, acabam recebendo as requisições de forma balanceada, que foi confirmado nas figuras 10 e 11, o que melhora as respostas do PVFS2, não deixando-o se tornar um gargalo.

Já o Ext3 não possui um desempenho muito bom quando se tem acesso concorrente, ficando muito inferior ao PVFS2. Mesmo quando o acesso é mono-tarefa, o PVFS2 pode ser melhor, desde que a máquina cliente tenha capacidade suficiente no acesso à rede de alta velocidade.

Isso mostra que temos como gargalo principal o cliente, ou mais especificamente seu poder de processamento e transmissão de dados, conforme podemos observar na figura 13. Nela, vemos claramente que um cliente mais potente consegue obter melhor desempenho do PVFS2 do que um cliente mais fraco, acessando o mesmo sistema de arquivos, o que mostra que tanto a rede como a CPU são mais importantes que a velocidade do disco quando o objetivo é atingir alto desempenho no acesso ao sistema de arquivos.

Entre os possíveis trabalhos futuros, temos o estudo de aplicações reais que usam uma grande quantidade de dados, como, por exemplo, as de bio-informática, assim como o estudo do impacto da redundância (através da duplicação) de informações.

9 Agradecimentos

Gostaríamos de agradecer ao empenho do Edson Moreira e da Tereza Carvalho, pelo suporte necessário para o acesso a uma das redes de alta velocidade que usamos durante o início dos testes, e também ao Ricardo Andrade, pelas rápidas respostas durante a fase inicial da preparação do ambiente. Em especial, agradecemos à Christiane Nishibe, por toda paciência, dedicação e ajuda que nos deu durante a realização de todos os testes finais usados nesse estudo.

Referências

- [Aas 2005] Aas, J. (2005). Understanding the Linux 2.6.8.1 CPU Scheduler. Technical report, Silicon Graphics, http://josh.trancesoftware.com/linux/linux_cpu_scheduler.pdf. Acessado em junho de 2005.
- [Carns et al. 2000] Carns, P. H., III, W. B. L., Ross, R. B., and Thakur, R. (2000). PVFS: A parallel virtual file system for linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317 – 327. <http://www.parl.clemson.edu/pvfs/el2000/extreme2000.ps>, (Best Paper Award).
- [de Carvalho 2005] de Carvalho, R. P. (2005). Sistema de Arquivos Paralelos: Alternativas para a redução do gargalo no acesso ao sistema de arquivos. Master's thesis, Instituto de Matemática e Estatística da Universidade de São Paulo. <http://www.ime.usp.br/~carvalho/defesa/mestrado.ps.gz>.
- [Haddad 2000] Haddad, I. F. (2000). PVFS: A parallel virtual file system for linux clusters. *Linux Journal*, 80:74 – 82. <http://www.linuxjournal.com/article/4354>.
- [PVFS2 Development Team 2003] PVFS2 Development Team (2003). Parallel Virtual File System, Version 2. Technical report, Clemson University. <http://www.pvfs.org/pvfs2/pvfs2-guide.html>, acessado em março de 2005.
- [Ross et al. 2002] Ross, R., Ligon, W., and Carns, P. (2002). Parallel Virtual File System. <http://www.parl.clemson.edu/pvfs2/sc2002-whitepaper-pvfs.pdf>.
- [von Hagen 2003] von Hagen, B. (2003). Exploring the Ext3 Filesystem. What is Journaling? Technical report, Linux Planet. <http://www.linuxplanet.com/linuxplanet/reports/4136/3/>, acessado em novembro de 2003.