Sistemas de Arquivos Paralelos: Alternativas para a redução do gargalo no acesso ao sistema de arquivos

Roberto Pires de Carvalho
Instituto de Matemática e Estatística
Universidade de São Paulo
São Paulo, SP – Brasil
carvalho@ime.usp.br

Alfredo Goldman
Instituto de Matemática e Estatística
Universidade de São Paulo
São Paulo, SP – Brasil
gold@ime.usp.br

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 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 a substituição de um sistema de arquivos local, no caso o Ext3, por um sistema de arquivos paralelo, no caso o PVFS2, pode aumentar de forma considerável o desempenho de aplicações concorrentes, e até mesmo seqüenciais, que utilizam uma grande quantidade de dados.

1 Introdução

A velocidade de acesso aos arquivos a partir de seus meios físicos de armazenamento (como dispositivos magnéticos, óticos, etc) não evoluiu 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 a outras máquinas via rede) [3].

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.

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 concorrentes ou não, para resolver o problema do gargalo provocado pelo sistema de arquivos local.

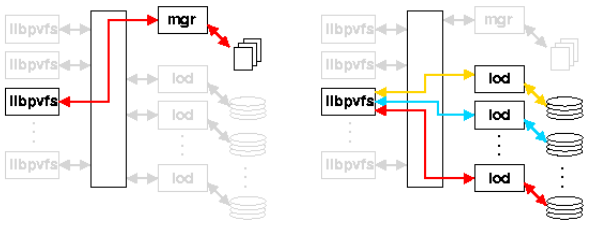
Este resumo de dissertação está organizado da seguinte forma: na seção 2 apresentamos o PVFS2, sistema de arquivos paralelo usado para validar nossa proposta. Em seguida, na seção 3, comentamos sobre a motivação de se comparar o PVFS2 com o sistema de arquivos local Ext3. Na seção 4 encontra-se o estudo premilinar realizado. Detalhes sobre a realização dos testes e ambiente estão na seção 5, enquanto que os resultados estão na seção 6. Por fim, na seção 7, apresentamos as conclusões obtidas.

2 PVFS2

O *Parallel Virtual File System* [2, 6, 9] é um sistema de arquivos distribuído desenvolvido para prover alto desempenho e escalabilidade para aglomerados de PCs linux. Ele foi escolhido para esse estudo por ter seu código fonte aberto, suporte ao sistema operacional Linux, licença de uso gratuita e estar em constante desenvolvimento. Outros possíveis candidatos levados em consideração, mas que não apresentavam todas essas características, foram NFSp [7], CEFT-PVFS [11] e Google File System [4].

Em geral, o PVFS2 promete 4 características:

- 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.



(a) (b)

Figura 1. Clientes acessando o PVFS2

Quando uma aplicação acessa o PVFS2 para obter os dados de um determinado arquivo, várias máquinas são acessadas, de forma transparente. Por acessar várias máquinas, utiliza-se de vários caminhos pela rede para chegar aos respectivos discos em que os dados estão armazenados. Isso elimina o gargalo de entrada e saída de dados quando se tem toda a informação armazenada em uma só máquina, distribuindo a carga e aumentando o potencial total da banda.

2.1 Os componentes do PVFS2

O **servidor de meta-dados** (**mgr** na figura 1(a)) gerencia todos os dados relativos ao 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. O PVFS2 permite que se tenha mais de um servidor de meta-dados simultâneo para distribuição de carga, evitando, assim, torná-lo um gargalo no acesso aos meta-dados dos arquivos.

O **servidor de dados** (**iod** na figura 1(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 PVFS2 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 [10], por exemplo.

No momento em que este estudo foi realizado, o PVFS2 não suportava alta disponibilidade, sendo que tal característica estava programada para ser apresentada em versões futuras. Porém, sempre 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 PVFS2** possibilita acesso em modo usuário aos servidores do PVFS2. 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 1(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 1(b).

O **suporte no *kernel* do linux para o PVFS2** provê as funcionalidades necessárias para se poder usar o comando *mount* nos clientes. Isso permite acesso aos arquivos do PVFS2 sem alterações nas aplicações ou programas já existentes. Esse suporte não é necessário para se usar o PVFS2, mas ele traz uma enorme conveniência para a interatividade com o sistema, pois devido à transparência no acesso pode-se considerar o uso do PVFS2 para qualquer aplicação que acesse uma grande quantidade de dados. Ele se utiliza da biblioteca *libpvfs* para realizar essas operações.

3 PVFS2 vs. Ext3

Nosso objetivo é comparar o desempenho do sistema de arquivos paralelo PVFS2 contra o desempenho do sistema de arquivos local Ext3, usando uma única máquina como cliente. 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.

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.

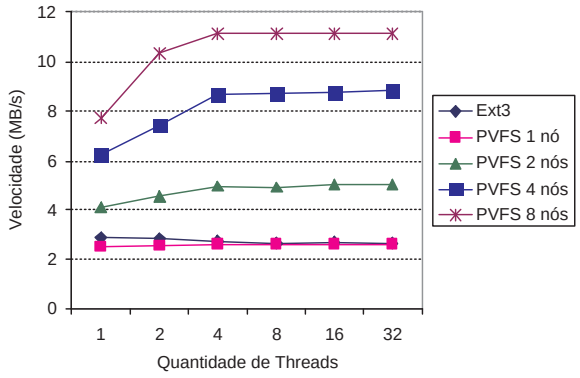
4 Estudo Preliminar

Nosso estudo foi realizado em duas etapas: simulação [3] e verificação [5]. Essas etapas foram necessárias por não possuirmos um aglomerado conectado em uma rede de alta velocidade no início do estudo. Nesta seção comentamos brevemente sobre a primeira etapa, que pode ser encontrada com maiores detalhes em [3].

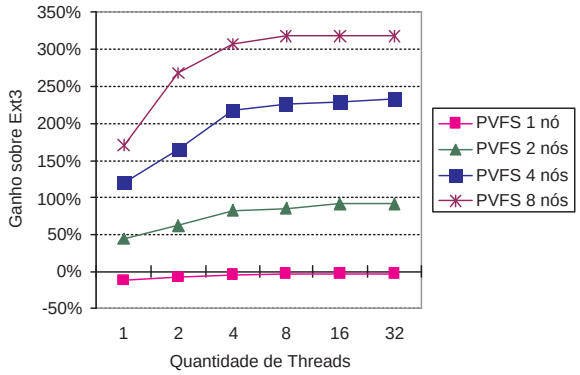
Como a velocidade da rede usada em nossos testes iniciais era mais lenta que a velocidade dos discos (aproximadamente 12MB/s contra 25MB/s, respectivamente), a velocidade dos discos locais de todas as máquinas envolvidas foi reduzida para que pudéssemos simular uma rede de dados mais rápida que o sistema de armazenamento local.

A figura 2 mostra os resultados encontrados, onde nota-se claramente que se a rede for realmente mais rápida que os discos, há um ganho considerável do PVFS2 sobre o Ext3 quando temos apenas um cliente acessando-os, principalmente de forma concorrente.

Ao aumentarmos a velocidade dos discos gradualmente, percebemos que a vantagem do PVFS2 sobre o Ext3 cai

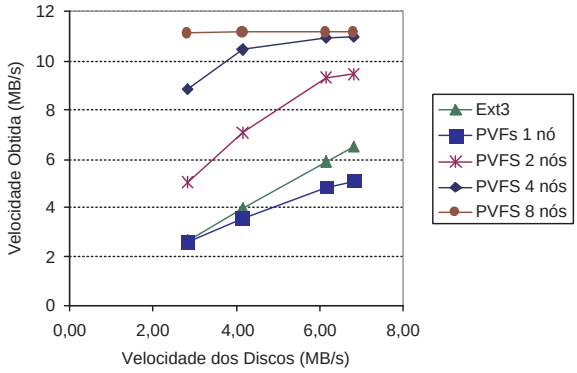


(a)

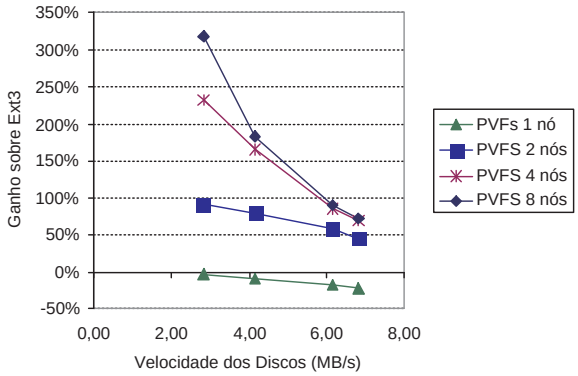


(b)

Figura 2. (a) Velocidade e (b) ganho do PVFS2 sobre o Ext3 ao aumentar a concorrência no acesso, com discos na velocidade 2,5MB/s



(a)



(b)

Figura 3. (a) Velocidade e (b) ganho do PVFS2 sobre o Ext3 ao variarmos a velocidade dos discos, usando 32 threads

rapidamente, conforme visto na figura 3. A partir desses resultados, 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, que os discos não usam mais do que 92% da banda disponível (em média) e que não haveria nenhum outro gargalo a não ser o próprio disco.

5 Sobre os Testes

Para comparar o comportamento do PVFS2 e do Ext3 sobre uma rede Gigabit, 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 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, desde a abertura do arquivo até o último byte 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 [1].

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.

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.

Assim, para podermos analisar a influência de tal mecanismo sobre o desempenho no acesso a grandes quantidades de dados, realizamos dois tipos de testes: limpando os dados do cache antes de acessar o sistema de arquivos e sem limpar os dados do cache, acessando o mesmo arquivo logo após ele já ter sido lido.

Para limpar o cache do lado do cliente, apenas desmontamos o sistema de arquivos PVFS2, e para limparmos o cache dos servidores e do Ext3, criamos um programa que aloca e preenche toda a memória física da máquina. O preenchimento é necessário pois o gerenciamento de memória do Linux 2.6 foi desenvolvido levando-se em conta programas que alocam muita memória mas não a utilizam. Dessa forma, se um programa pedir memória, o Linux só irá realmente alocá-la quando ela for modificada.

5.4 Amostras

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 representam os valores mínimo e máximo encontrados.

5.5 Ambiente Utilizado

Para a realização dos testes, contamos com um aglomerado de 12 máquinas[1], sendo 9 servidores, 2 clientes e 1 *gateway* não utilizado, todas conectadas entre si através de placas de rede Gigabit e um *switch* de mesma velocidade. A distribuição Linux usada nas máquinas foi a RedHat, com *kernel* 2.6.12.

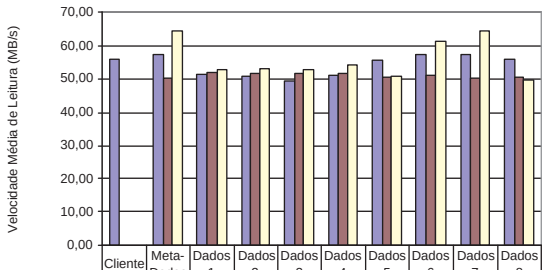
Instalamos o PVFS2 versão 1.3.0 em todos os servidores, seguindo as instruções do *Quick Guide* [8] do sistema. Nenhuma configuração foi alterada com relação à instalação padrão. Também instalamos a interface UNIX para acesso ao PVFS2 nos clientes. Tal interface é um módulo do *kernel* que permite ao usuário montar o PVFS2 e utilizá-lo como um sistema de arquivos comum, de forma totalmente transparente para as aplicações.

Utilizamos dois clientes para testar os sistemas de arquivos, sendo um deles um Pentium 4 2,8GHz (cliente 1) e o outro um Pentium 4 1,8GHz (cliente 2). O objetivo de se ter dois clientes é para podermos comparar o desempenho de um cliente mais lento com o de um mais rápido e verificar o impacto disso no acesso ao PVFS2. Em nenhum momento temos os dois clientes acessando o PVFS2 de forma simultânea.

Na figura 4 temos a velocidade média de leitura dos dados a partir dos discos (obtida usando-se a ferramenta *hdparm*) e a velocidade média de transferência de dados entre as máquinas (obtida usando-se a ferramenta *iperf*). 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 seus componentes não são todos da mesma marca ou modelo.

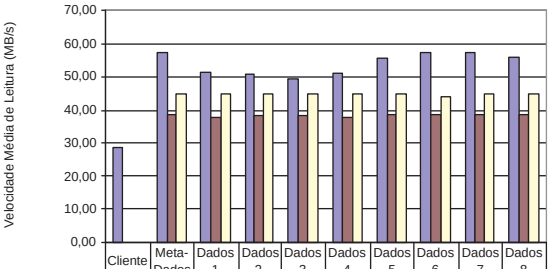
Na figura 4(b) podemos notar que as velocidades da rede se mantém independentemente dos servidores acessados, o que não ocorre na figura 4(a). Isso se deve ao fato do cliente 1 ser uma máquina com maior poder de processamento se comparada ao cliente 2. Na figura 5 podemos ver que o cliente 2 usa muito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

[1]Gostaríamos de agradecer ao Edson Cáceres, pelo empréstimo das máquinas e da rede, e à Christiane Nishibe, pelo suporte prestado durante os testes.



	Cliente	Meta-Dados	Dados 1	Dados 2	Dados 3	Dados 4	Dados 5	Dados 6	Dados 7	Dados 8
Disco	56,00	57,40	51,50	50,90	49,40	51,30	55,80	57,50	57,40	55,90
Rede (cliente-servidor)	0,00	50,40	52,00	51,90	51,90	51,90	50,70	51,20	50,40	50,60
Rede (servidor-cliente)	0,00	64,20	52,60	53,00	52,80	54,30	50,80	61,40	64,20	49,80

(a)



	Cliente	Meta-Dados	Dados 1	Dados 2	Dados 3	Dados 4	Dados 5	Dados 6	Dados 7	Dados 8
Disco	29,00	57,40	51,50	50,90	49,40	51,30	55,80	57,50	57,40	55,90
Rede (cliente-servidor)	0,00	38,70	37,70	38,50	38,40	37,90	38,60	38,60	38,70	38,70
Rede (servidor-cliente)	0,00	45,00	45,00	45,10	45,10	45,00	45,10	44,10	45,00	45,10

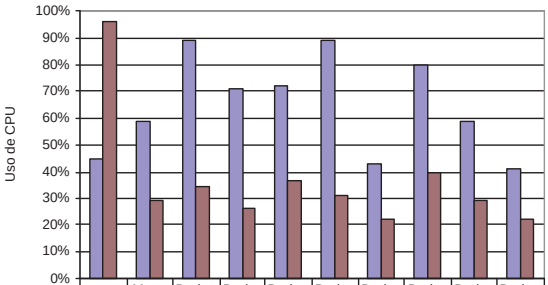
(b)

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

6 Resultados

Analisando os resultados obtidos em nossos testes iniciais, constatamos que enquanto algumas das variáveis adotadas influenciaram de forma considerável os resultados, outras foram praticamente irrelevantes. Dentre elas, o tamanho do bloco de leitura não chegou a influenciar muito no desempenho de ambos os sistema de arquivos [5], 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 tamanhos.

A figura 6(a) mostra o comportamento do PVFS2 com apenas um servidor de dados sendo acessado por múltiplas threads. Nota-se que o PVFS2 com muitas threads e poucos dados nos mostra o que já havia sido observado em [3]: a sobrecarga que cada thread de nossos testes sofre antes de começar a ler os dados é muito grande. Nisso inclui-se contactar o servidor de meta-dados, requisitar informações sobre a localização dos dados e conectar no servidor de dados.



	Cliente	Meta-Dados	Dados 1	Dados 2	Dados 3	Dados 4	Dados 5	Dados 6	Dados 7	Dados 8
Cliente 1	45%	59%	89%	71%	72%	89%	43%	80%	59%	41%
Cliente 2	96%	29%	34%	26%	37%	31%	22%	40%	29%	22%

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

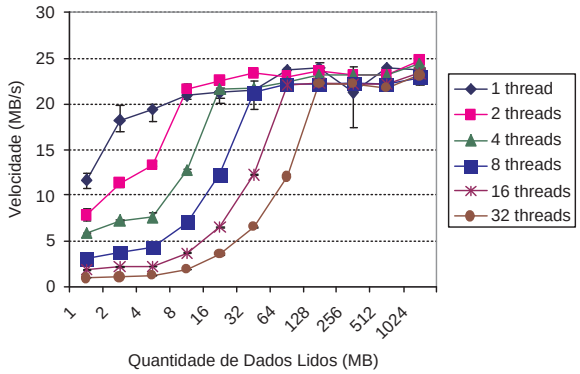
A figura 6(b) mostra o comportamento do Ext3. Note que ele melhora o desempenho aos poucos conforme a quantidade de dados a serem lidos aumenta. Porém, ao aumentarmos a quantidade de threads, o desempenho cai, mostrando que ele não lida muito bem com acesso concorrente.

Assim, dadas as conclusões às quais chegamos, podemos simplificar nossa análise sem comprometer nossos resultados, fixando 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 e de threads.

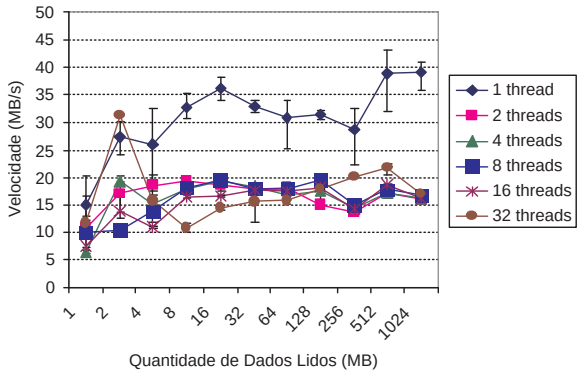
Uma das conseqüencias dessa decisão é a pouca relevância do uso de cache do Ext3 nos nossos testes de leitura de dados, pois temos mais dados para serem lidos do que memória cache disponível. Já o PVFS2, por ter vários servidores e por distribuir o conteúdo do arquivo entre eles, possui uma maior quantidade 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%.

Com relação ao aumento da quantidade de servidores de dados e da quantidade de threads (figura 7(a)), percebemos que existe um ganho considerável no desempenho, até atingir o limite da velocidade da rede (verificado na figura 4(a)). Na figura 7(b) temos o mesmo caso, porém usando o cliente 2. Note que o desempenho deste último é bem inferior ao primeiro. Isso já era esperado, pois por possuir menor poder de processamento, tal cliente não tem uma resposta rápida o suficiente para tratar os dados que vêm pela rede através dos múltiplos servidores de dados, 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 7

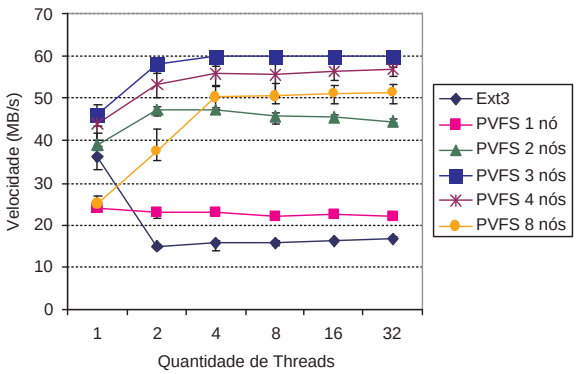


(a)

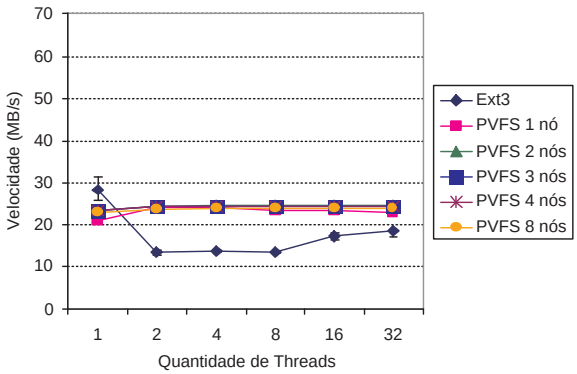


(b)

Figura 6. Leitura através do (a) PVFS2 com 1 servidor de dados e do (b) Ext3, variando a quantidade de dados e número de threads



(a)



(b)

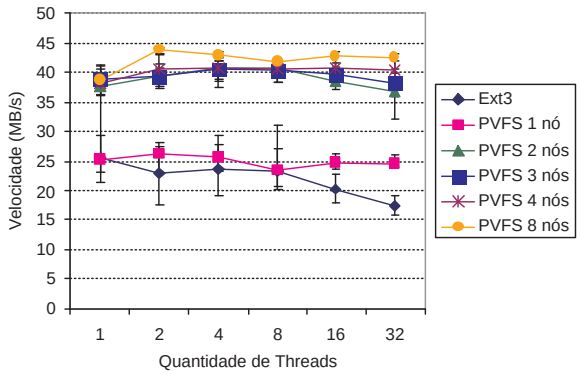
Figura 7. Desempenho do (a) cliente 1 e do (b) cliente 2, lendo 1GB de dados do PVFS2 e do Ext3 em blocos de 64KB

em diante, é que eles 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, o que acaba "escondendo" o resultado por trás de suas retas no gráfico. Além disso, usando 3 servidores de dados conseguimos o melhor desempenho dos nossos testes. Isso se deve ao nosso arranjo de servidores, pois devido à rede possuir máquinas heterogêneas, tentamos colocar aquelas que tiveram melhor desempenho de disco e rede como sendo os primeiros servidores.

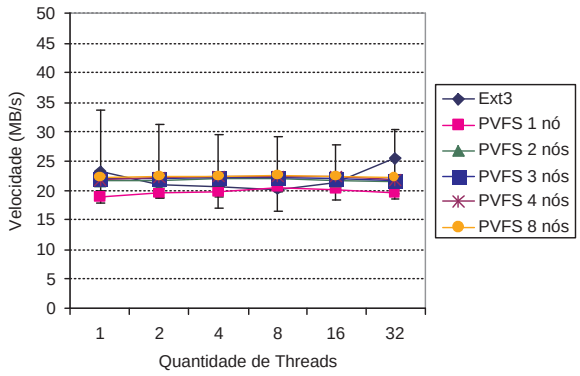
Comparando o Ext3 com o PVFS2 na figura 7, notamos que quando o Ext3 é acessado por apenas uma thread, de forma seqüencial, seu desempenho é muito bom. Porém, ao usarmos 2 threads, o desempenho do Ext3 degrada, enquanto que o PVFS2 melhora de forma considerável, especialmente quando se aumenta a quantidade de servidores de dados. E, nesse caso, conforme aumentamos a concorrência, o desempenho do PVFS2 aumenta até se estabilizar próximo ao limite verificado da velicidade da rede.

Já na escrita, que pode ser observada na figura 8, o desempenho do Ext3 se apresenta melhor do que na leitura (devido à escrita em cache, para depois enviar 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 acentuado devido à necessidade de se criar o arquivo, há uma perda de desempenho com relação à leitura. Mas mesmo com essa melhora do Ext3 e piora do PVFS2, até na escrita este é mais eficiente que aquele, pelo menos com relação ao cliente 1.

Quando analisamos o cliente 2, cujo teste de desempenho de escrita pode ser encontrado na figura 8(b), vemos que o PVFS2 tem desempenho quase idêntico ao teste de leitura. Isso ocorre devido ao poder limitado do processador deste cliente, que não suporta uma quantidade elevada

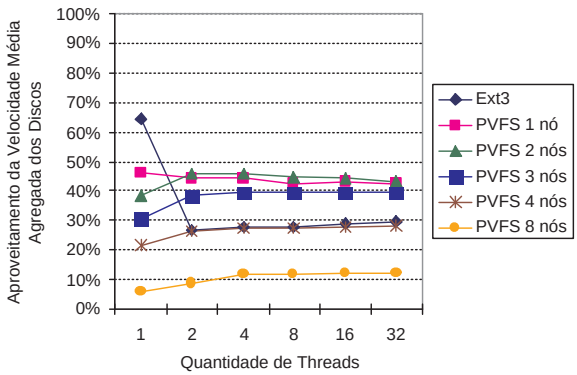


(a)

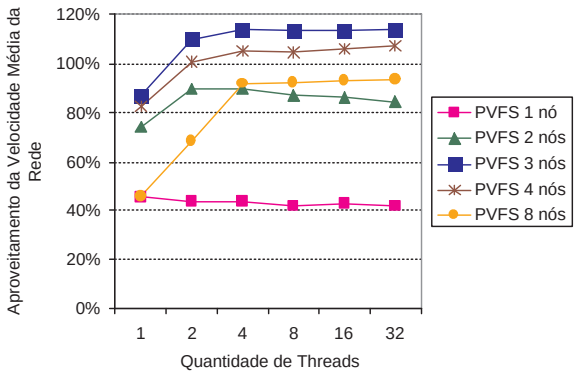


(b)

Figura 8. Desempenho do (a) cliente 1 e do (b) cliente 2, escrevendo 1GB de dados no PVFS2 e no Ext3 em blocos de 64KB



(a)



(b)

Figura 9. Aproveitamento da velocidade média agregada (a) dos discos dos servidores e (b) da rede de dados, para o cliente 1

de dados vindo pela rede. Já o Ext3 tem a mesma melhora observada no cliente 1, aumentando o desempenho de tal forma a se igualar ao PVFS2.

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 9(a) 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.

Com relação ao aproveitamento da velocidade da rede, vemos na figura 9(b) que o cliente 1 está limitado pela velocidade máxima verificada da rede, que, por sua vez, está limitada pela CPU de pelo menos uma das pontas da transmissão dos dados (veja figura 5). Em alguns casos o desempenho do PVFS2 ultrapassa esse limite (de aproximadamente 53MB/s), o que pode acontecer já que a velocidade máxima teórica da rede (125MB/s) é muito maior do que a verificada em nossos testes (veja figura 4).

No caso do cliente 2, não faz sentido comparar o aproveitamento das bandas agregadas de disco e rede, já que em todos os testes ele esteve sempre limitado pelo seu poder de processamento, nunca chegando ao limite da velocidade média da rede.

7 Conclusão

A partir da análise dos resultados da seção anterior podemos concluir que o PVFS2 realmente pode ser mais eficiente que o Ext3. Na figura 10, montada a partir dos re-

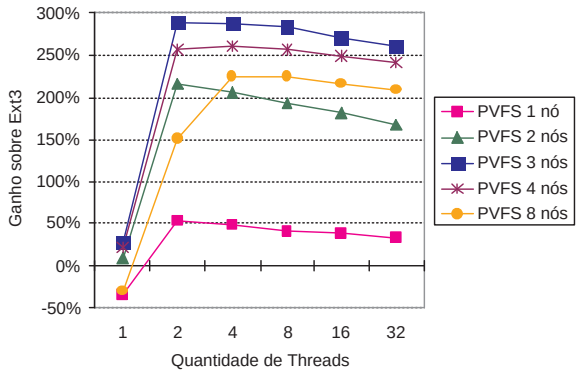


Figura 10. Ganho do PVFS2 sobre o Ext3 ao aumentar a concorrência no acesso, usando o cliente 1

sultados 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.

Dessa forma, concluímos que o gargalo no 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, 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 de processamento e no acesso à rede de alta velocidade.

Assim, temos como gargalo principal o cliente, ou mais especificamente seu poder de processamento e transmissão de dados, conforme observa-se na figura 7. 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.

Referências

[1] J. Aas. Understanding the Linux 2.6.8.1 CPU Scheduler. Technical report, Silicon Graphics, `http://josh.trancesoftware.com/linux/linux_cpu_scheduler.pdf`, February 2005. Acessado em junho de 2005.

[2] P. H. Carns, W. B. L. III, R. B. Ross, and R. Thakur. PVFS: A parallel virtual file system for linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317 – 327, October 2000. `http://www.parl.clemson.edu/pvfs/el2000/extreme2000.ps`, (Best Paper Award).

[3] R. P. de Carvalho. 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, September 2005. `http://www.ime.usp.br/~carvalho/defesa/mestrado.pdf`.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, New York, NY, USA, 2003. ACM Press. Acessado em fevereiro de 2004: `http://www.cs.rochester.edu/sosp2003/papers/p125-ghemawat.pdf`.

[5] A. Goldman and R. P. de Carvalho. Sistemas de arquivos paralelos: Reduzindo o gargalo no acesso ao disco. `http://www.ime.usp.br/~carvalho/sbrc2006/artigo.pdf`, 2006.

[6] I. F. Haddad. PVFS: A parallel virtual file system for linux clusters. *Linux Journal*, 80:74 – 82, December 2000. `http://www.linuxjournal.com/article/4354`.

[7] P. Lombard and Y. Denneulin. NFSP: A distributed NFS server for cluster of workstations. `http://ka-tools.sourceforge.net/publications/nfsp-ipdps01.pdf`, 2001.

[8] PVFS2 Development Team. Parallel Virtual File System, Version 2. Technical report, Clemson University, September 2003. `http://www.pvfs.org/pvfs2/pvfs2-guide.html`, acessado em março de 2005.

[9] H. Ramachandran. Design and implementation of the system interface for PVFS2. Master's thesis, Clemson University, December 2002. `ftp://ftp.parl.clemson.edu/pub/techreports/2002/PARL-2002-008.ps`.

[10] B. von Hagen. Exploring the Ext3 Filesystem. What is Journaling? Technical report, Linux Planet, 2003. `http://www.linuxplanet.com/linuxplanet/reports/4136/3/`, acessado em novembro de 2003.

[11] Y. Zhu, H. Jiang, X. Qin, D. Feng, and D. R. Swanson. Improved read performance in CEFT-PVFS: Cost Effective, Fault-Tolerant Parallel Virtual File System. In *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003)*, pages 730–735, May 2003. Acessado em junho de 2005: `http://www.cs.nmt.edu/~xqin/pubs/ccgrid03.pdf`.