Utilização de Recursos Alocados pelo Usuário para Armazenamento de Dados no Sistema de Arquivos dNFSp

Everton Hermann, Danilo Fukuda Conrad, Francieli Zanon Boito,
Rodrigo V. Kassick, Rafael B. Avila, Philippe O. A. Navaux
Instituto de Informática - Universidade Federal do Rio Grande do Sul
Porto Alegre - RS - Brasil
{ehermann, dfconrad, fzboito,rvkassick, avila, navaux}@inf.ufrgs.br

Resumo

Este artigo apresenta um modelo de alocação de servidores temporários para o armazenamento de dados no sistema de arquivos dNFSp. Esta funcionalidade permite a adaptação do sistema distribuído às necessidades de desempenho e armazenamento de diferentes aplicações. Será apresentado um modelo de tratamento do dinamismo de servidores de dados, assim como a implementação deste modelo seguida de uma avaliação de desempenho.

1. Introdução

A utilização de *cluster* de computadores tem crescido muito rapidamente dentro da área de computação paralela. Um número significativo de aplicações científicas executadas neste tipo de plataforma necessitam efetuar uma grande quantidade de operações de entrada e saída em armazenamento secundário (discos, sistemas de arquivos em rede, etc.). Por isso, o desempenho de tais operações é crucial e pode ser o principal fator na determinação do tempo de execução de uma aplicação [12].

O crescimeto da diferença de velocidade entre processador e dispositivos de armazenamento tem agravado ainda mais esta situação, tornando as operações de entrada e saída no ônus mais severo pago por aplicações que fazem uso massivo do sistema de armazenamento. Uma das alternativas que permitem amenizar este problema com baixo custo é a utilização dos recursos de armazenamento dos nós de um cluster através de um sistema de arquivos paralelo. Isto permite agregar as capacidades de transferência de rede e de disco de vários destes nós, oferecendo uma significativa melhoria de desempenho.

Em conseqüência desta necessidade de sistema de arquivos paralelos, existe uma quantidade significativa de trabalhos que abordam sistemas de arquivos que sejam escaláveis e façam bom uso dos padrões existentes de hardware e de software. Um bom exemplo deste tipo de sistema de arquivos é o dNFSp[10, 8, 2]. pois faz uso do protocolo NFS[11], largamente utilizado em clusters e outros sistemas. No dNFSp as funcionalidades de um servidor NFS tradicional são divididas entre os nós de um cluster, dedicando algumas máquinas para o armazenamento de dados enquanto outras são responsáveis pela gerência de metadados dos arquivos.

A utilização de um protocolo padrão como o NFS permite que clientes de diversas arquiteturas utilizem o sistema de arquivos paralelo sem dificuldade, uma vez que só necessitam da implementação tradicional do protocolo. Outros sistemas largamente utilizados em cluster como o PVFS [3, 9] e o Lustre [4] possuem um protocolo específico para a comunicação entre clientes e servidores, o que os torna incompatíveis com soluções ja existentes de protocolos de armazenamento.

Para a instalação de um sistema de arquivos distribuído em um *cluster* de computadores é possível utilizar de forma exclusiva uma parte dos nós disponíveis neste conjunto de máquinas. Neste caso, um bom dimensionamento da quantidade de nós ocupados pelo sistema de arquivos faz-se essencial. Este dimensionamento é, no entanto, dificultado pelas diferentes necessidades de armazenamento das váriоas aplicações paralelas que executam em um cluster : se são disponibilizados muitos nós para o sistema de arquivos, serão consumidos recursos que poderiam ser utilizados para outros fins. Se o sistema de arquivos for distribuído em um número pequeno de máquinas, é possível não estar disponibilizando o grau de paralelismo necessário a aplicações que façam uso massivo do armazenamento.

Para que seja possível disponibilizar um sistema de arquivos paralelo que contemple aplicações com grandes necessidades de armazenamento e ainda assim não manter recursos ociosos quando executando aplicações mais simples, será apresentado um modelo que introduz novas funcionalidades ao dNFSp, permitindo a criação dinâmica de servi-

dores de dados. Desta forma, é possível dedicar nós dinamicamente alocados como recursos temporários de armazenamto do sistema de arquivos. Isso possibilita que o sistema seja adaptado às necessidades da aplicação, utilizando eficientemente a estrutura computacional disponível.

2. Sistema de Arquivos Paralelos

Em sistemas de arquivos paralelos, a inserção e remoção dinâmica de servidores de dados é levada em consideração por diversos motivos, principalmente expandir o sistema e suportar tolerância a falhas. Exemplos deste tipo de sistema de arquivos são o xFS[1], o Expand (XPN) [5] e o Lustre.

No xFS, quando uma nova maquina se junta ao sistema, o sistema pode adicioná-la ao mapeador de gerenciadores. Este mapeador está presente em todos os nós do sistema e mantém as informações necessárias para encontrar os dados de um determinado arquivo. A cada mudança na configuração do sistema de arquivos, o mapeador deve ser redistribuído entre os nós, para que os arquivos possam ser acessados corretamente. A funcionalidade de reconfiguração não foi implementada nos protótipos desenvolvidos pela equipe do xFS, porém faz parte do modelo proposto pelo sistema de arquivos.

O sistema de arquivos Expand (XPN) [5] combina vários servidores NFS para oferecer um particionamento distribuído de arquivos. Esta solução não inclui alterações no servidor NFS. Trata-se da implementação de uma biblioteca que oferece uma interface POSIX ao programador. Desta forma, o XPN não funciona como um sistema de arquivos montável em um diretório do sistema.

Uma das funcionalidades projetadas para o Expand é permitir o redimensionamento do sistema de arquivos através da inserção de novos servidores. Porém, para que os dados estejam adequados à nova configuração, é necessário redistribuí-los quando um novo servidor é inserido. Apesar de fazer parte dos projetos, até o momento não existem publicações onde constam resultados da implementação.

O sistema de arquivos Lustre [4] foi projetado com o objetivo de remover os gargalos tradicionalmente encontrados em sistemas de arquivos para *cluster*. Sua arquitetura é dividida em vários tipos de serviços. Os servidores de metadados (MDSs, Metadata Servers) contêm o esquema de diretórios do sistema de arquivos, permissões, e atributos estendidos para cada objeto. Os OST (Object Storage Targets) são responsáveis pelo armazenamento e a transferência dos dados de um arquivo. Tanto um quanto o outro pode operar em pares (*active* / *failover*) e podem automaticamente ocupar o lugar um do outro em casos de falha.

O sistema de arquivos Zebra [6] distribui os dados dos arquivos em diversos servidores assegurando que a perda de um único servidor não afeta a disponibilidade dos dados. O esquema de fracionamento utilizado pelo Zebra é semelhante ao do RAID nível 5, onde os dados e a paridade são divididos entre os discos. O bloco de paridade é obtido a partir de um ”ou exclusivo”(XOR) dos blocos de dados. A definição do nó que armazena o bloco de paridade é feita através de distribuição circular entre os nós. Através do uso desta técnica o Zebra pode suportar a perda de nós, visto que existem dados redundantes.

Apesar de existir uma variedade de sistemas de arquivos para cluster que planeja a inserção e remoção dinâmica de servidores de dados, poucos possuem esta funcionalidade completamente implementada. Quando este fator é levado em conta, o principal objetivo é tolerância a falhas e não oferecer garantias de desempenho e adaptação às necessidades específicas de uma aplicação, como é a proposta deste trabalho.

3. O sistema de arquivos dNFSp

O sistema de arquivos dNFSp teve sua origem no NFSp (NFS parallel), desenvolvido no laboratório ID/IMAG em Grenoble. o NFSp é uma extensão da implementação tradicional do NFS, que divide as funções do servidor entre diversos nós de um cluster [10]. O principal objetivo do NFSp é aumentar a desempenho e a escalabilidade em relação ao seu antecessor mantendo-se compativel com os clientes NFS.

Do mesmo modo que o PVFS [3, 9], o NFSp utiliza servidores de entrada e saida (I/O daemons, IODs) para acessar os dados em disco. Quem se encarrega do papel de servidor NFS é o *nfspd*, definido pelos seus criadores como *meta-servidor*. Este servidor é visto pelos clientes como um servidor NFS tradicional e é responsável por repassar as requisições de leitura e escrita ao IOD que possui os dados em questão.

A arquitetura utilizando um único meta-servidor permite um grande ganho de desempenho nas operações de leitura, uma vez que os IODs podem enviar os dados diretamente aos clientes. As operações de escrita, no entando, precisam passar pelo meta-servidor que, por ser único no sistema, transforma-se no gargalo de desempenho.

O dNFSp tem como objetivo resolver esta limitação do NFSp através da distribuição dos metadados em vários servidores. Estes servidores funcionam da mesma maneira que o servidor centralizado do NFSp, recebendo as requisições dos clientes e repassando-as aos IODs, compartilhados por todos meta-servidores. No entanto, cada meta-servidor atende a apenas uma parte dos clientes, os quais ainda funcionam como clientes NFS tradicionais. Com a utilização de múltiplos meta-servidores, quando os clientes acessam o sistema de arquivos simultaneamente, a carga é distribuida entre os servidores, possibi-

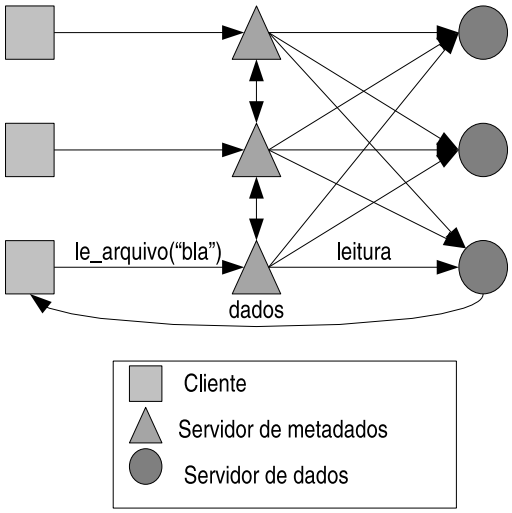

Figura 1. dNFSp

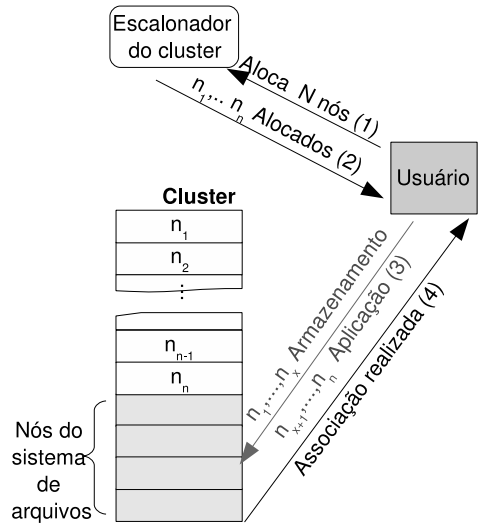

Figura 2. Alocação de nós pelo usuário

litando assim obter um melhor desempenho nas operações de escrita.

A Figura 1 ilustra uma operação de leitura no dNFSp. O cliente envia uma requisição de leitura do NFS para o meta-servidor ao qual está associado. O meta-servidor passa a requisição ao IOD. Ao recebe-la, o IOD envia os dados diretamente ao cliente.

4. Modelo de Inclusão Iniciada Pelo Usuário

Devido à grande variedade de aplicações que são executadas em um *cluster*, nem sempre é possível satisfazer as necessidades de armazenamento de todos os usuários. O pouco conhecimento prévio que o administrador tem sobre as aplicações executadas torna o dimensionamento do sistema de arquivos em uma tarefa bastante complexa. Uma má escolha na quantidade de servidores destinados a este fim pode resultar num desperdício de recursos quando o sistema de arquivos é super-dimensionado. Por outro lado, a disponibilização de poucos nós de armazenamento pode implicar um baixo desempenho a algumas aplicações, principalmente àquelas que efetuam um número significativo de operações de entrada e saída.

O usuário, no entanto, possui maior controle sobre o perfil de suas aplicações que o administrador do *cluster*, o que lhe permite determinar com maior precisão o padrão de acesso aos dados antes mesmo de iniciar a execução. Com isso, um melhor uso dos recursos pode ser obtido se o usuário tiver a possibilidade de redimensionar o sistema de arquivos de acordo com as necessidades de suas aplicações.

Para possibilitar esta escolha, é preciso oferecer ferramentas que permitam ao usuário definir o número de nós que uma aplicação deseja dedicar ao armazenamento. Este número deve ser contabilizado junto com os nós de processamento no momento da reserva de recursos junto ao escalonador do *cluster*(PBS, OAR, etc.)

Como ilustrado na Figura 2, o usuário aloca um número de nós suficiente ao armazenamento e à execução de sua aplicação. No momento em que estes recursos são disponibilizados, é preciso informar ao sistema de arquivos que uma aplicação deseja dedicar alguns nós para o armazenamento. Fazendo uso das ferramentas de configuração, o usuário indica quais nós serão responsáveis pelo processamento e quais vão estar armazenando dados. Ao receber a solicitação do usuário, o sistema de arquivos faz as associações necessárias para que os novos nós sejam integrados no sistema.

Neste cenário, os novos nós de armazenamento são chamados de **servidores de dados temporários**. Estes nós armazenam informações pertencentes a uma aplicação durante a sua execução. Os demais servidores de dados, inseridos pelo administrador no momento da instalação do sistema de arquivos são chamados de **servidores de dados permanentes**. É neles que ficam armazenados os dados durante todo o tempo de vida do sistema de arquivos.

No momento de sua inserção no sistema, cada servidor temporário é associado a um grupo de armazenamento. Um grupo de armazenamento é composto por um ou mais servidores permanentes de dados e no máximo um servidor temporário de por aplicação. Não existe replicação de dados entre os servidores temporários de um mesmo grupo: as únicas informações escritas nestes são os dados criados pela aplicação a eles associada. Os dados armazenados nos servidores permanentes também não são copiados para os servidores temporários no momento de sua inserção.

Todas as operações de leitura e escrita feitas pelos clientes são encaminhadas aos servidores temporários. Como

não existe replicação entre os servidores permanentes e temporários, é possível que algum dado requisitado não seja encontrado. Isso ocorre principalmente na leitura de dados anteriores à execução da aplicação, ou então dados criados por outras aplicações. Neste caso, a requisição de leitura é repassada ao servidor permanente que pertence ao mesmo grupo do nó que recebeu a requisição. Se a informação existir ela será encontrada nos servidores permanentes.

O acesso aos servidores de dados temporários é restrito aos nós da aplicação associada a eles. Isso permite minimizar a interferência das operações de E/S de uma aplicação em outra. Portanto, aplicações que possuem nós temporários tem acesso exclusivo a eles, enquanto que as demais apenas têm acesso aos servidores permanentes.

Após o final da execução de uma aplicação os recursos associados a ela devem ser liberados. Porém, antes é preciso transferir os dados dos servidores temporários para os permanentes. Este procedimento garante que os dados criados pela aplicação possam ser acessados posteriormente.

5. Implementação

A implementação do dNFSp utiliza um esquema de servidores de dados virtuais (VIOD). Cada VIOD representa um grupo de armazenamento responsável por guardar uma fração do arquivo e é formado por um ou mais IODs. Juntamente com o gerenciador de servidores de dados (iod_mgmt), um VIOD funciona como uma camada de abstração entre os meta-servidores e os servidores de dados (IOD), abstraindo dos meta-servidores a maneira como os dados são armazedos e permitindo alterações dinâmicas de IODs.

O iod_mgmt está acoplado ao servidor de metadados, de forma que ambos executam no mesmo nó. Portanto, existe um iod_mgmt para a cada meta-servidor. Grande parte da implementação da inserção de nós iniciada pelo usuário é feita no iod_mgmt. É através dele que os comandos do usuário são enviados. O lançador de comandos permite executar diferentes ações: criar uma aplicação, inserir clientes da aplicação, finalizar uma aplicação, entre outras ações não relacionadas à inserção dinâmica iniciada pelo usuário.

Apesar de toda a configuração ser tratada pelo iod_mgmt, são ainda os servidores de dados e metadados os responsáveis pelas operações em arquivo. A única função do iod_mgmt é receber as solicitações do usuário, avaliar a situação atual do sistema de arquivos e tomar decisões de como processar a solicitação. O resultado é repassado aos servidores de dados e metadados para ser posto em pratica.

5.1. Interface com o Usuário

A interface entre o usuário e o gerenciador de servidores de dados é feita através de linha de comando. Três novas opções foram criadas para definir a associação de nós a uma aplicação. O primeiro deles é o create-execution, que tem como função notificar o sistema que se deseja executar uma aplicação com servidores de dados inseridos pelo usuário. Esta opção cria um espaço para o gerenciamento desta nova aplicação e retorna ao usuário um identificador único para a execução criada, que deve ser utilizado nos passos subseqüentes.

Para definir quais nós fazem parte de uma aplicação, é utilizada a opção application-node. Esta opção recebe como parâmetro o endereço do nó e o identificador da aplicação. Ao receber esta solicitação, o iod_mgmt insere este nó na lista de clientes da aplicação e transmite esta informação aos servidores de metadados.

A inserção de servidores temporários em um VIOD é feita pelo próprio IOD durante sua inicialização. A diferenciação entre um IOD temporário dos demais é feita através da opção application-iod. Ela recebe como parâmetro o identificador da aplicação à qual o IOD estará associado. Ao ser inicializado com esta opção, o servidor temporário anuncia ao sistema de arquivos que um novo servidor temporário acaba de ser inserido.

Após a inseção de todos os servidores de dados e a associação dos clientes à aplicação, o usuário pode iniciar a execução. A existência de servidores de dados temporários é transparente à aplicação, uma vez que o sistema de arquivos continua se comportando como um sistema NFS padrão. Desta forma, não é preciso qualquer alteração na aplicação para executá-la em um ambiente com servidores de dados exclusivos.

Ao final da execução, é preciso notificar o iod_mgmt que os servidores temporários não são mais necessários. Isto é feito através do comando delete-execution, seguido do identificador da aplicação. Ao processar esta operação, o iod_mgmt envia uma mensagem aos servidores de dados temporários indicando a qual IOD permanente os dados devem ser transferiridos. Finalizada a sincronização entre os servidores de dados, os nós podem ser liberados e o iod_mgmt solicita aos servidores de metadados a exclusão da aplicação.

Um exemplo de execução utilizando esta interface é mostrado na Figura 3. Para facilitar a utilização desta ferramenta, foi desenvolvido um lançador de aplicações chamado dnfsprun. Este lançador interage com o escalonador do cluster, no caso testado o OAR. Desta forma, é necessário informar apenas o número de nós destinados ao armazenamento, além das informações usuais para executar a aplicação. Por exemplo, a linha seguinte pode ser utilizada para executar uma aplicação utilizando MPI que ne-

```
n1:#iod_mgmgt --create-execution
0
n1:#iod_mgmgt --application-node="client1 0"
n1:#ssh iod1 iodng --applicationiod=0
n1:#ssh client1 mytests.sh
Resultados da aplicacao
n1:#iod_mgmgt --delete-execution=0
```

Figura 3. Exemplo de inicialização de nós exclusivos

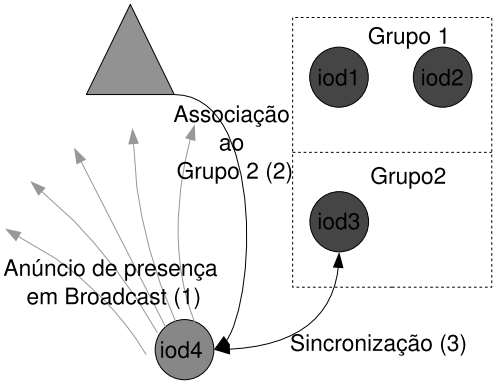

Figura 4. Alocação de nós pelo usuário

cessite 4 nós de armazenamento:

```
n1:#dnfsprun.sh 4 mpirun  -machinefile \
$DNFSP_CLIENTS -np 4 btio.bin
```

5.2. Tratamento da Inserção

No modelo proposto a inserção de novos servidores é baseada na estrutura de VIODs. Cada novo servidor de dados passa a fazer parte de um dos VIODs, de acordo com a configuração atual do sistema.

A inserção de servidores temporários é iniciada no momento do lançamento do servidor de dados. Para determinar se um servidor faz parte de um grupo de servidores temporários de uma aplicação é preciso utilizar a opção applicationiod seguida do identificador da aplicação, como ilustrado anteriormente.

O procedimento de execução do servidor é ilustrado na Figura 4. Inicialmente, o IOD envia uma mensagem em *broadcast* sinalizando a sua existência. Esta mensagem é tratada pelo iod_mgmt que que analisa a configuração do sistema de arquivos e define o grupo de armazenamento (VIOD) do qual o novo nó vai fazer parte. Em seguida é enviada uma mensagem ao IOD notificando a nova configuração.

Apenas um dos iod_mgmt trata a requisição do IOD. Além de notificar o IOD sobre o grupo de armazenamento que ele deve fazer parte, o iod_mgmt deve notificar todos os demais iod_mgmt da nova configuração. Em seguida, cada iod_mgmt repassa as modificações aos seus respectivos servidores de meta-dados.

5.3. Caminho de dados com IODs Temporários

Apesar das decisões de configuração serem feitas pelo iod_mgmt, durante a execução, o caminho percorrido pelos dados é definido pelos servidores de metadados. Uma tabela de mapeamento entre nós de armazenamento e nós de computação permite determinar de forma direta quais servidores de dados pertencem à mesma aplicação de um determinado cliente.

Ao receber a configuração do iod_mgmt, o servidor de metadados constrói uma tabela *hash* associando o endereço dos clientes com uma aplicação. Esta tabela é utilizada no tratamento de requisições de escrita e leitura. Através dela, o servidor de metadados procura a aplicação e a lista de servidores de dados associada ao emissor. Se não existe nenhuma entrada nesta tabela, considera-se que o cliente não faz parte de uma aplicação e a requisição é transmitida a um dos IODs permanentes. No caso de existir uma entrada na tabela, o IOD escolhido faz parte da lista de servidores temporários da aplicação.

Em consequência disso, requisições de um mesmo tipo podem ser tratadas por grupos de nós diferentes, dependendo da configuração do sistema. Esta multiplicidade de caminhos de dados pode evitar sobrecarga de servidores. Por pertencerem a apenas uma aplicação, os IODs temporários tendem a ser menos requisitados que os servidores permanentes e portanto oferecem maior desempenho.

As requisições de leitura transmitidas aos servidores de dados temporários nem sempre podem ser respondidas por eles. Isso ocorre nos casos onde os dados não existem no servidor temporário. Neste caso, a requisição é transmitida a um dos servidores permanentes que pertence ao mesmo VIOD e, portanto, é responsável por tratar do mesmo conjunto de dados.

A Figura 5 ilustra a execução de três aplicações. Duas das aplicações não possuem servidores de dados temporários, e por isso concorrem pelo acesso aos servidores de dados permanentes. Por outro lado, os nós que pertencem ao Cliente 3 (C3) possuem servidores de dados temporários, o que permite acessar estes recursos exclusivamente durante suas operações de leitura e escrita. O acesso aos servidores permanentes é feito apenas quando os dados não são encontrados nos servidores temporários. Nesta figura a presença dos servidores de metados foi abstraída, no entanto todas as requisições devem

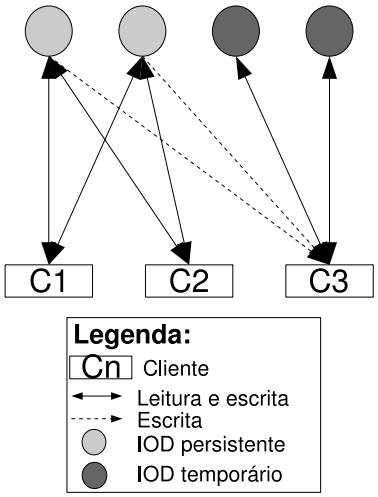

Figura 5. Execução de 3 aplicações, sendo que uma possui IODs temporários

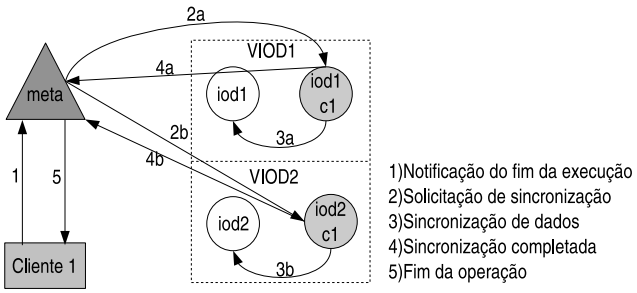

Figura 6. Finalização da execução de uma aplicação

passar por ele antes de chegar nos servidores de metadados.

5.4. Tratamento da Finalização

Quando a aplicação chega ao final de sua execução, é preciso notificar o sistema de arquivos que se deseja retirar os servidores de dados temporários associados a ela e iniciar a replicação dos dados. Uma vez que os servidores de dados temporários deixam de existir após a alocação dos recursos, as informações devem ser copiadas para os IODs permanentes.

Ao receber notificação de finalização 6, o gerenciador de servidores de dados envia uma mensagem aos servidores de dados temporários. Nesta mensagem esta incluído o identificador do IOD persistente ao qual ele deve transferir seus dados. Após a transmissão é possível excluir os servidores de dados temporários e liberar os nós alocados.

6. Resultados

Para avaliar o mecanismo de inserção de servidores de dados iniciada pelo usuário, foi desenvolvido um conjunto de testes englobando diversas situações que fazem parte deste cenário. O primeiro caso tem como objetivo avaliar o ganho de desempenho ao executarmos uma aplicação com servidores de dados exclusivos em uma situação onde existem outras aplicações executando no *cluster*. No segundo caso, a analise é voltada para descrever o impacto das diferentes etapas do ciclo de vida da aplicação nas demais aplicações executando no *cluster* no mesmo instante. Neste caso o objetivo é observar o quanto o desempenho da aplicação é afetado pelas aplicações executando no *cluster* ao mesmo tempo que ela.

Os testes apresentados nesta seção foram executados no *i-cluster2* [7] instalado em Montbonnot Saint Martin, France, na sede do INRIA Rhône-Alpes. O cluster é composto por 100 nós. Cada nó é equipado com processadores duais Itanium2 900 MHz com 3 Gb de memória e armazenamento em disco com capacidade de 72Gb, 10000 rpm, SCSI. Todos os nós estão inter-conectados utilizando rede 1 Gigabit Ethernet, rede Fast Ethernet e rede Myrinet. Os experimentos foram realizados utilizando a rede 1 Gigabit Ethernet. O sistema operacional instalado nos nós é a versão estável do Debian GNU/Linux, com a versão 2.6.15 do núcleo do Linux.

6.1. Desempenho no uso de nós exclusivos

Neste teste, foi feito uma instalação do dNFSp contendo 27 clientes, 9 meta-servidores e 9 IODs. Os 27 clientes foram divididos em 3 aplicações compostas de 9 clientes cada uma. Sobre esta configuração foram executados testes de leitura e escrita seqüencial de dados, onde cada cliente escreveu e leu 1GB de dados. Em seguida foram inseridos servidores de dados temporários a uma das aplicações em um número igual ao de servidores de dados permanentes. Para completar o teste, a mesma quantidade de servidores de dados do caso anterior, somando-se permanentes e temporários, foi distribuída na forma de servidores de dados permanentes. As três configurações testadas são ilustradas de forma simplificada na figura 7. Foram feitas 10 execuções de cada caso, e o valor exibido no gráfico é o tempo médio dos nós de cada aplicação. Os resultados do teste de escrita podem ser observados na Figura 8. Como esperado, a execução utilizando servidores de dados exclusivo trouxe os melhores resultados para a aplicação que possui servidores temporários. As demais aplicações tiveram resultados iguais entre si, visto que compartilham o mesmo

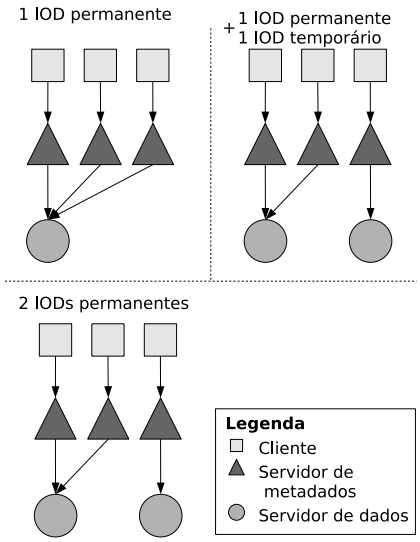

Figura 7. Descrição dos cenários utilizados para avaliar o desempenho da inserção dinâmica de servidores de dados iniciada pelo usuário

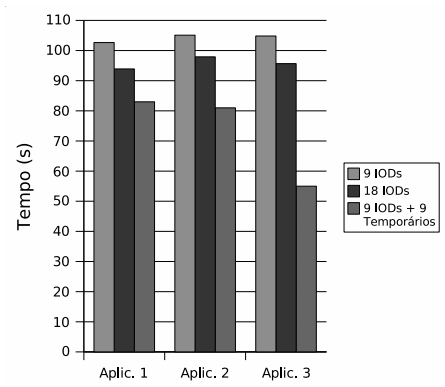

Figura 8. Resultado da escrita de dados utilizando servidores de dados temporários

recurso. No caso onde existiam 18 servidores de dados permanentes houve um ganho de desempenho das aplicações sem IODs temporários, mas para a aplicação 3 este não foi o melhor resultado entre as três configurações testadas. Isso mostra que o uso de servidores de dados exclusivos para a aplicação pode oferecer vantagens de desempenho que não seriam obtidas se os recursos fossem compartilhados com as demais. O ganho da aplicação 3 utilizando servidores temporários ao ser comparado com o primeiro é de 45%.

Já na Figura 9 temos o comparativo da leitura dos dados. Neste caso não existe a participação do servidor de metadados na transferência de dados, visto que os dados são transferidos diretamente entre o servidor de dados e o cliente. Como o tempo de transferência é menor, o impacto da existência de servidores de dados temporários é menos visível, mas mesmo assim foi possivel obter um ganho de desempenho de 11% utilizando servidores temporários.

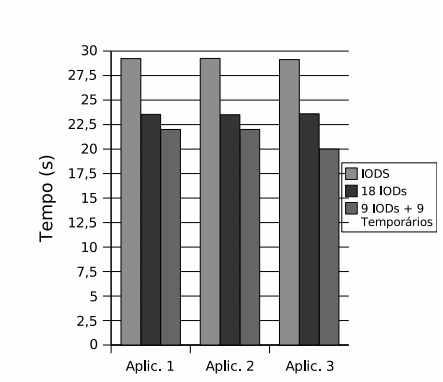

Figura 9. Resultado da leitura de dados utilizando servidores de dados temporários

6.2. Impacto dos Servidores Temporários nas Demais Aplicações

Este teste tem como objetivo traçar um perfil da taxa de transferência de dados das aplicações durante a utilização do mecanismo de servidores de dados temprários. O cenário de testes é o mesmo da seção anterior. No entanto, é feita a transferência de dados contínuamente, obtendo-se a taxa de transferência de cado em determinados instantes da execução. A Figura 10 mostra o perfil dos resultados obtidos. Inicialmente existem apenas 9 servidores de dados compartilhados pelas três aplicações (t1). Portanto a taxa de transferência consumida por cada uma é igual. Após 100 segundos de teste (t1), a Aplicação 3 insere 9 servidores de dados temporários. Em consequência disso ela tem sua taxa de transferência quase dobrada.

Nos 100 segundos subsequentes as aplicações continuam a transferir, até que no instante t2 a Aplicação 3 finaliza sua execução. Neste instante, os dados armazenados nos servidores temporários são transferidos para os permanentes. Como a operação de replicação não envolve os servidores de metadados, tanto a Aplicação 1 quanto a Aplicação 2 passam a ter uma maior taxa de transferência. Ao finalizar a replicação no instante t3 as aplicações 1 e 2 passam a obter taxas de transferência ainda maiores pois os servidores de dados permanentes não estão mais envolvidos na replicação e portanto podem se dedicar plenamente às operações destas duas aplicações.

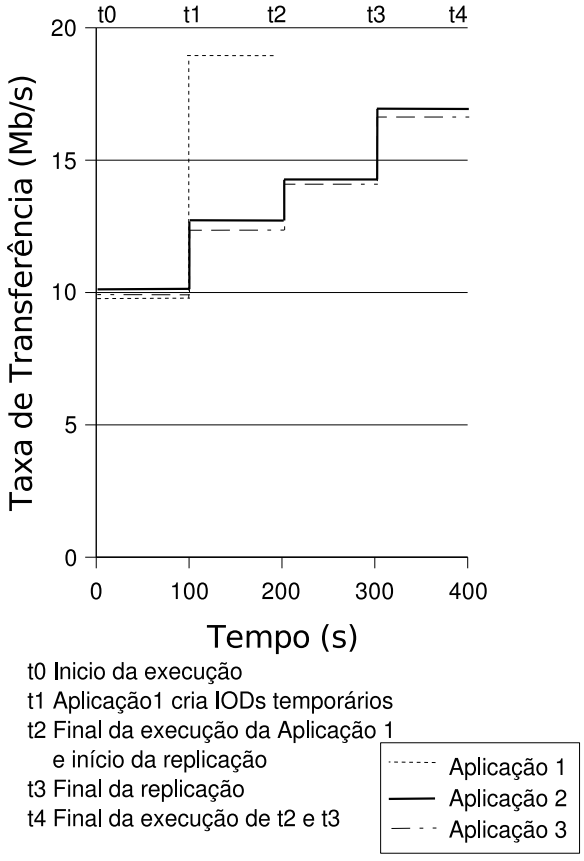

Figura 10. Impacto do mecanismo de IODs temporários nas demais aplicaçoes executando no cluster

7. Conclusão

O desempenho do armazenamento de dados pode ser um fator determinante no desempenho final de algumas aplicações. Por isso é necessário que uma arquitetura de alto desempenho, com é o cluster, ofereça soluções que satisfaçam estas necessidades. Para facilitar esta tarefa, foi apresentado neste trabalho um modelo que permite ao usuário adaptar o sistema de arquivos às suas necessidades. Isto é feito através da inclusão de servidores de dados temporários. Estes recursos são alocados juntamente com os nós de processamento, e servem de armazenamento exclusivo para a aplicação.

Para validar este modelo foi feita uma implementação tendo como base o sistema de arquivos dNFSp. Testes de desempenho foram executados e mostraram um ganho de desempenho na ordem de 50% nas operações de escrita, e de 10% nas operações de leitura.

Atualmente está em desenvolvimento a expansão deste modelo para a supressão de nós causadas por falha. Este tratamento, aliado a técnicas de replicação pode aumentar a disponibilidade do sistema.

Referências

[1] T. Anderson, M. Dahlin, J. Neefe, D. Pat-terson, D. Roselli, and R. Wang. Serverless network file systems. In *In Proceedings of the 15th Symposium on Operating System Principles. ACM*, pages 109–126, Copper Mountain Resort, Colorado, December 1995.

[2] R. B. Ávila, P. O. A. Navaux, P. Lombard, A. Lebre, and Y. Denneulin. Performance evaluation of a prototype distributed NFS server. In J.-L. Gaudiot, M. L. Pilla, P. O. A. Navaux, and S. W. Song, editors, *Proc. of the 16th Symposium on Computer Architecture and High Performance Computing*, pages 100–105, Foz do Iguaçu, Oct. 2004. Washington, IEEE.

[3] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. PVFS: a parallel file system for Linux clusters. In *Proc. of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, 2000. Best Paper Award.

[4] Cluster File Systems, Inc. Lustre: A scalable, high-performance file system, 2002. Available at http://www.lustre.org/docs/whitepaper.pdf (July 2004).

[5] F. Garcia-Carballeira, A. Calderon, J. Carretero, J. Fernandez, and J. M. Perez. The design of the expand parallel file system. *International Journal of High Performance Computing Applications*, 17(1):21–37, 2003.

[6] J. H. Hartman and J. K. Ousterhout. The Zebra striped network file system. In H. Jin, T. Cortes, and R. Buyya, editors, *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, pages 309–329. IEEE Computer Society Press and Wiley, New York, NY, 2001.

[7] i-cluster 2, 2006. Available at: <http://i-cluster2.inrialpes.fr>. Access in: May 2006.

[8] R. Kassick, C. Machado, E. Hermann, R. Ávila, P. Navaux, and Y. Denneulin. Evaluating the Performance of the dNFSP File System. In *5th Int. Symposium on Cluster Computing and the Grid (CCGrid 2005)*, Cardiff, UK, May 2005. IEEE Press.

[9] R. Latham, N. Miller, R. Ross, and P. Carns. A next-generation parallel file system for linux clusters. *LinuxWorld*, 2(1), January 2004.

[10] P. Lombard and Y. Denneulin. nfsp: A distributed nfs server for clusters of workstations. In *IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium*, page 352, Washington, DC, USA, 2002. IEEE Computer Society.

[11] Sun Microsystems, Inc. RFC 1094: NFS: Network File System Protocol specification, Mar. 1989.

[12] Y. Zhu, H. Jiang, X. Qin, D. Feng, and D. Swanson. Improved read performance in a Cost-Effective, Fault-Tolerant Parallel Virtual File System (CEFT-PVFS). In *Proceeding of IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, Tokyo, Japan, May 2003.