UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"

Pós-Graduação em Ciência da Computação

César de Souza Aguiar

Modelo de virtualização distribuída aplicado ao gerenciamento e replicação de *clusters* multiuso

UNESP

2008

César de Souza Aguiar

Modelo de virtualização distribuída aplicado ao gerenciamento e replicação de *clusters* multiuso

Orientadora: Profª.Drª. Roberta Spolon Ulson

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Sistemas de Computação, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

UNESP
2008

AGRADECIMENTOS

Agradeço a minha orientadora Profª.Drª Roberta Spolon Ulson, que recebe adequadamente este título pelo incentivo, correções e adequações realizadas neste trabalho e por diversas vezes, ter se disposto a resolver problemas para que esse projeto chegasse aos seus objetivos.

Ao meu amigo Daniel Igarashi que compartilhou diversas tardes de fim de semana pesquisando e aplicando o conhecimento obtido, proporcionando novas idéias na elaboração do projeto, além de todo seu apoio pessoal.

Ao Cleyton Messias e Gabriel Pedroso, por toda força e pelo auxílio através de iniciação científica no desenvolvimento do projeto.

À UNESP, que proporcionou recursos para que os experimentos do projeto fossem concluídos.

Ao Prof. Dr. Eduardo Martins Morgado, por todo apoio fornecido.

Aos meus queridos pais, Dorothy e João, irmãs, Luciana e Denise, sobrinha Luisa, sogra Benilde, minha querida avó Izabel, e especialmente para minha namorada Vanessa, por toda preocupação, incentivo e suporte, até nos momentos em que estive menos presente.

E por fim, aos meus amigos, que de forma geral me proporcionaram condições para que este trabalho se concretizasse.

"Qualquer tecnologia suficientemente
avançada parece ser mágica."
Arthur C. Clarke

Sumário

Lista de Abreviaturas e Siglas ..... **v**

Lista de Figuras ..... **vi**

Lista de Tabelas ..... **vii**

1 Introdução **1**

1.1 Contextualização 1
1.2 Motivação 1
1.3 Objetivos 2
1.4 Estrutura da Monografia 3

2 Sistemas de Arquivos Distribuídos e Paralelos **4**

2.1 Considerações Iniciais 4
2.2 Conceitos e Definições 4
2.2.1 Gerenciamento de *Cache* 6
2.2.2 Técnicas de *Prefetching* 7
2.2.3 Tolerância a Falhas e Protocolos Consistentes 8
2.2.4 Sistemas de Arquivos Virtuais 9
2.3 Sistemas de Arquivos Distribuídos 10
2.3.1 NFS 11
2.3.2 AFS 12
2.3.3 CODA 13
2.4 Sistemas de Arquivos Paralelos 14
2.4.1 PVFS 14
2.4.2 NFSP 16
2.4.3 pNFS 17
2.4.4 Sistema de Arquivos Híbrido (pNFS / PVFS2) 18
2.5 Considerações Finais 19

3 *Clusters* Multiuso, *Grids* e Computação *Commodity* **21**

3.1 Considerações Iniciais 21
3.2 *Boot* Remoto 21
3.2.1 LTSP 22
3.2.2 DRBL 23
3.2.3 SLIM 24
3.3 *Clusters* Multiuso 25
3.4 Ambientes em *Grid* 28
3.5 Considerações Finais 28

4 Máquinas Virtuais **30**

4.1 Considerações Iniciais 30
4.2 Estrutura das Máquinas Virtuais 30
4.3 Técnicas de Virtualização 32
4.3.1 Emulação 32
4.3.2 Virtualização nativa (“virtualização completa”) 32
4.3.3 Paravirtualização 33
4.4 Máquinas Virtuais em Sistemas Distribuídos 34
4.4.1 Servidores de Máquinas Virtuais 34
4.4.2 Migração de Máquinas Virtuais 35
4.4.3 Máquinas Virtuais e *Grids* 36
4.5 Isolamento e Segurança 37

4.6 Considerações Finais 38

5 FlexLab 39

5.1 Considerações Iniciais 39
5.2 Requisitos do FlexLab 41
5.3 Estrutura e Implementação 42
5.3.1 Sistema de Arquivos Distribuído 44
5.3.2 Balanceamento de Carga 44
5.3.3 *Boot* Remoto 46
5.3.4 *Clusters* Multiuso e *Grids* 47
5.4 Trabalhos Relacionados 48
5.5 Considerações Finais 49

6 Experimentos 51

6.1 Considerações Iniciais 51
6.2 Experimentos para definir a configuração do FlexLab 51
6.2.1 Comparações de técnicas de *boot* remoto 51
6.2.2 Comparação entre NFSv3, PVFS2 e pNFS/PVFS2 53
6.2.3 Otimização de desempenho 56
6.3 Experimentos do FlexLab 58
6.3.1 Experimentos de *Boot* do FlexLab 58
6.3.2 Prova de conceito para *boot* paralelizado de imagens virtuais 60
6.3.3 Comparativo do FlexLab com um *boot* convencional de disco rígido 61
6.4 Considerações Finais 62

7 Conclusão 65

7.1 Conclusões Gerais 65
7.2 Contribuições deste Trabalho 66
7.3 Trabalhos Futuros 67

Referências Bibliográficas 69

Anexo A 76

AFS: Andrew File System
API: Application Program Interface
COD: Cluster On Demand
CODA: Constant Data Availability
DHCP: Dynamic Host Configuration Protocol
DRBL: Diskless Remote Boot in Linux
DSM: Distributed Shared Memory
EXT3: Third Extended File System
FAT: File Allocation Table
HPFS: High Performance File System
LAN: Local Area Network
LTSP: Linux Terminal Server Project
MV: Máquina Virtual
NASD: Network Attached Storage Devices
NFS: Network File System
NFSP: Netware File Sharing Protocol
NIS: Network Information Service
NTFS: New Technology File System
NTP: Network Time Protocol
pNFS: Parallel Network File System
PVFS: Parallel Virtual File System
PXE: Preboot Execution Environment
RAID: Redundant Arrays of Inexpensive Disks
RSH: Remote Shell
SAD: Serviço de Arquivos Distribuído
SAN: Storage Area Networks
SAP: Sistemas de Arquivos Paralelos
SLIM: Single Linux Image Management
SVM: Shared Virtual Memory
TCO: Total Cost of Ownership
TCP: Transmission Control Protocol
TFTP: Trivial File Transfer Protocol
UDP: User Datagram Protocol
VFS: Virtual File System
VMM: Virtual Machine Monitor
VSG: Volume Storage Group
WAN: Wide Area Network
XDR: External Data Representation

Lista de Figuras

Figura 1 - Distribuição dos volumes entre os servidores 14
Figura 2 - Estrutura da implantação do pNFS com PVFS2..... 18
Figura 3 - Modelo de escrita entre pNFS e PVFS2 19
Figura 4 - Tempo de *Boot* vs. Número de Computadores 25
Figura 5- Estrutura de máquinas virtuais..... 31
Figura 6 - Estrutura de um Servidor de Máquina Virtual..... 35
Figura 7 - FlexLab: *Boot* remoto de imagens virtuais em máquinas heterogêneas com balanceamento de carga entre servidores 40
Figura 8 - Modelo de *Boot* Distribuído do FlexLab: diferentes tipos de sistemas de arquivos distribuídos são utilizados dependendo dos dados requisitados 43
Figura 9 - Inclusão de servidor extra com balanceamento de carga à medida com que são acrescentados novos clientes ao *boot* remoto..... 45
Figura 10 - Computadores clientes auxiliam na distribuição de blocos já carregados para outros computadores..... 46
Figura 11 - *Cluster* multiuso distribuindo aplicações *grid* paralelamente enquanto sistemas operacionais convencionais são utilizados..... 48
Figura 12 - Comparação de velocidade na cópia de 12 MB com blocos de 8 KB 54
Figura 13 - Comparação de operações de leitura com 1 cliente em ambientes com NFSv3 e pNFS/PVFS2 55
Figura 14 - Comparação de operações de escrita com um cliente em ambientes com NFSv3 e pNFS/PVFS2 56
Figura 15 - *Boot* de 10 clientes com otimizações de *cache* no *middleware* 58
Figura 16 - Gráfico comparativo do tempo de *boot* do FlexLab 60

Lista de Tabelas

Tabela 1 - Diferença entre *clusters* e computadores pessoais 26
Tabela 2 - Configuração de *hardware* para os testes em soluções de *boot* remoto..... 52
Tabela 3 - Comparação entre DRBL e SLIM..... 52
Tabela 4 - Configuração de *hardware* para os experimentos de balanceamento de carga no FlexLab..... 53
Tabela 5 - Comparação de velocidade de leitura em KB/s entre pNFS/PVFS2 e NFSv3 54
Tabela 6 - Comparação de velocidade de escrita em KB/s entre pNFS/PVFS2 e NFSv3 55
Tabela 7 - Configuração de *hardware* para os experimentos de *boot* no FlexLab..... 58
Tabela 8- Tempo de *boot* de um Windows 2003 virtualizado e distribuído para um número diversificado de clientes sem disco rígido..... 59
Tabela 9- Tempo de *boot* de diversas máquinas virtuais inicializadas em um mesmo cliente simultaneamente 61
Tabela 10 - Comparativo de tempo de *boot* em um cliente a partir de diferentes modelos de inicialização 62

Resumo

Este trabalho apresenta um modelo de *boot* remoto para computadores *commodity* utilizando máquinas virtuais e sistemas de arquivos distribuídos e paralelos. O modelo proposto pode substituir o *boot* local com disco rígido por um *boot* através da rede de comunicação, aumentando assim a flexibilidade e manutenibilidade do parque de máquinas, além de permitir que dezenas de sistemas operacionais distintos sejam inicializados sem a necessidade de um disco rígido nos clientes, reduzindo dessa forma o custo em *hardware* e diminuindo a complexidade de instalação e manutenção de software, implantando um único ponto centralizado de gerenciamento. O projeto analisa maneiras de otimizar a transmissão de blocos de dados com técnicas de localidade de dados, sistemas de arquivos distribuídos e balanceamento de carga para implementar um ambiente robusto e de virtualização distribuída. O modelo também auxilia implementações de *clusters* multiuso e LAN *grids* para computadores *commodity,* provendo ferramentas para aproveitar recursos computacionais ociosos em conjuntos de computadores conectados. Neste estudo foram analisados diferentes modelos de sistemas de arquivos distribuídos, detalhando suas principais características e utilizações, e foram realizados experimentos com a virtualização distribuída juntamente com balanceamento de carga. A implantação de um sistema de arquivos híbrido através da integração de PVFS2 com pNFS trouxe melhorias de até 16% na velocidade de operações de leitura e permitiu maior escalabilidade da solução, assim como o gerenciamento de *cache* que permitiu a melhora de até 37% na velocidade de *boot* do *middleware*. Os resultados obtidos também viabilizaram o uso da solução para um grande número de computadores e possibilitaram o *boot* escalável de imagens virtuais remotamente.

PALAVRAS-CHAVE: Sistemas Distribuídos, *Cluster* Multiuso, *Grids*, Máquinas Virtuais, *Boot* Remoto

Abstract

This work presents a remote boot model to commodity computers using virtual machines and distributed and parallel file system. The proposed model can replace the local hard disk boot to a boot over the network of communication, thereby increasing the flexibility and maintainability of the group of machines, and with that allowing dozens of different operating system to be initialized without the need of a hard disk on customers, thus reducing the cost in hardware and reducing the complexity of installation and maintenance of software, implementing a centralized management unit. The project examines ways to optimize the data block transmission with techniques of data locality, distributed file system and load balancing to implement a robust environment for distributed virtualization. The model also helps implementations of multiuse clusters and grids to commodity computers, providing tools to take advantage of idle computing resources in connected computers. In this study it was analyzed different models of distributed file system, detailing their main characteristics and uses, it was also conducted experiments with distributed virtualization along with load balancing, which showed improvements in the overall performance of the system. The deployment of a hybrid filesystem by mixing PVFS2 with pNFS brought improvements of up to 16% in the speed of operations for reading and allowed greater scalability of the solution, as well as the management of *cache* that allowed the improvement of up to 37% in speed the boot of middleware. The results also made possible the use of the solution for a large number of computers and allowed a scalable boot of virtual images remotely.

KEYWORDS: Distributed System, Multiuse Clusters, Grids, Virtual Machines, Remote Boot

Capítulo
1

Introdução

1.1 Contextualização

Tem-se observado nos últimos vinte anos uma evolução significativa nas aplicações quanto às suas características e à demanda por recursos computacionais de processamento e armazenamento de dados. Modelos de programação paralela, *boot* remoto e máquinas virtuais estão cada vez mais presentes em pesquisas acadêmicas e áreas que dependam de alta taxa de manutenibilidade e segurança. Além disso, muitas áreas de pesquisa, como *clusters* e *grids*, necessitam de avanços nos meios de comunicação para melhorar a eficiência de implantação e manutenção dos sistemas. (BUTT, et al. 2002).

A velocidade de acesso aos arquivos a partir de seus meios físicos de armazenamento (como dispositivos magnéticos e óticos) não evoluiu na mesma proporção que a velocidade dos processadores ou da transmissão de dados, seja na própria máquina (como no acesso à memória ou registradores) ou remotamente (como no acesso às outras máquinas via rede). Isso torna o acesso aos sistemas de arquivos um gargalo para diversas aplicações, especialmente aquelas que estão em um contexto de *boot* remoto. Para minimizar esse problema têm sido utilizados sistemas de arquivos paralelos no intuito de fornecer alto desempenho no acesso aos dados para um número crescente de clientes em uma rede local.

1.2 Motivação

As redes de alta velocidade e o desenvolvimento de processadores de baixo custo têm motivado a adoção de sistemas distribuídos para resolução de problemas relacionados à implantação e gerenciamento de redes de computadores. Contudo,

mesmo com a grande velocidade de processamento e de transmissão de dados, os servidores ainda são gargalos em sistemas distribuídos que implantam *boot* remoto em diversos clientes simultaneamente.

Além do gargalo de rede obtido em sistemas distribuídos muitos problemas infra-estruturais de rede em manutenibilidade de *software*, *backup*, segurança e implantação de sistemas operacionais são fatores que necessitam de cuidados (CLARK, et al. 2005). A distribuição de imagens virtuais através da rede de comunicação e a abstração de *hardware* obtida com máquinas virtuais surgem como um modelo recente de sistemas distribuídos e necessitam de estudos de viabilidade e balanceamento de carga (KOTZ & ELLIS, 2001).

Soluções existentes baseadas em *thin-clients* (BARATTO, KIM & NIEH, 2005) permitem apenas o gerenciamento centralizado de recursos computacionais, ou seja, a maior parte do processamento é centralizado no servidor. Dessa forma são necessários servidores de grande porte para suportar, com estabilidade, um ambiente que necessite de muitos clientes simultaneamente, e mesmo assim não forneceriam um poder computacional adequado para situações como as encontradas em *grids* e *clusters*.

A tendência para computação *commodity* (ROY, et al. 2004) e o avanço das capacidades de armazenamento aumentaram consideravelmente o desempenho e a quantia de recursos computacionais disponíveis. Entretanto, muitos dos recursos computacionais nas universidades e empresas ainda são ociosos comparado a todo potencial disponível. Computadores *desktop* em escritórios e salas de informática não são utilizados durante a maior parte do seu tempo de vida, até mesmo pequenos *clusters* em grupos de pesquisa e desenvolvimento não utilizam todo seu potencial, desperdiçando recursos computacionais. De outro lado, há muitas aplicações úteis que não podem ser executadas devido à falta de poder computacional, pouca memória e espaço em disco limitado, que podem se beneficiar de um ambiente de *clusters comodity*.

1.3 Objetivos

Este trabalho tem como objetivo apresentar um modelo de *boot* remoto virtualizado baseado em *clusters* multiuso e sistemas de arquivos distribuídos que

permitam balancear a carga de requisição aos dados dos servidores. Também são propostas ferramentas que viabilizam a implantação de sistemas de *grids* através de um *middleware* configurável.

1.4 Estrutura da Monografia

Além do capítulo introdutório, em que foram vistos os objetivos e motivações para este trabalho, a monografia se divide em mais 6 capítulos descritos a seguir:

Capítulo 2 - Descreve as diversas técnicas e conceitos em sistemas de arquivos distribuídos e paralelos, assim como estudos com *cache* e balanceamento de carga.

Capítulo 3 - Descreve técnicas de *cluster* multiuso e *grids* na implantação de um sistema híbrido aproveitando-se dos recursos ociosos em ambiente de computação *comodity*.

Capítulo 4 - Detalha conceitos, técnicas e ferramentas para gerenciamento de máquinas virtuais distribuídas.

Capítulo 5 - Descreve o FlexLab, modelo criado para distribuição de máquinas virtuais em ambientes de *hardware* heterogêneo e suas ferramentas para auxiliar a implantação de aplicações distribuídas.

Capítulo 6 - Apresenta os experimentos realizados e os resultados obtidos para o modelo proposto.

Capítulo 7 - Conclui a dissertação abordando as contribuições deste projeto e trabalhos futuros.

Anexo A - Apresenta alguns dos *scripts* para implantação do *cluster* multiuso, balanceamento de carga e otimizações de *cache*, assim como os métodos utilizados para os experimentos de *benchmark*.

Capítulo
2

Sistemas de Arquivos Distribuídos e Paralelos

2.1 Considerações Iniciais

Os sistemas de arquivos distribuídos e paralelos possibilitam o compartilhamento de diretórios através da rede como se fossem diretórios locais, permitindo a execução de programas e sistemas operacionais com recursos de rede.

Este capítulo apresenta uma breve revisão abordando os sistemas de arquivos distribuídos e paralelos descrevendo técnicas de gerenciamento de *cache*, *prefetching*, balanceamento de carga e sistemas de arquivos virtuais.

2.2 Conceitos e Definições

Esta seção apresenta conceitos e definições disponibilizados pelos sistemas de arquivos distribuídos e paralelos, assim como algumas de suas características, qualidades e soluções.

A seguir estão especificadas algumas das características mais desejadas em sistemas de arquivos distribuídos e paralelos (TANENBAUM, 1992):

Disponibilidade

Depender de um serviço de rede que está indisponível por questões de falha de *hardware*, manutenção ou falta de energia elétrica é uma situação indesejável para qualquer sistema de arquivos distribuídos, ainda mais quando ocorre perda dos dados na transmissão.

É comum sistemas ficarem indisponíveis não por falha do próprio servidor, mas por falha na rede de comunicações. Caso um cliente deseje requisitar um arquivo que

está em um servidor inacessível em determinado momento, o sistema operacional do cliente bloqueia o processo que solicitou até que o servidor volte a ficar acessível. Uma das soluções para se resolver este problema é a replicação dos dados, ou seja, o mesmo arquivo, ou partes dele, deve estar em diferentes servidores. Assim, caso algum servidor fique fora da rede, outro fornecerá os mesmos arquivos. Alguns sistemas de arquivos distribuídos foram criados com essas características, como o CODA, detalhado na seção 2.3.3.

Escalabilidade

Os sistemas distribuídos são, em geral, projetados e configurados pensando-se na configuração da rede no momento da implementação do sistema. Porém, essa rede pode aumentar e dezenas ou centenas de novos clientes podem ser conectados ao sistema. Se essa situação não for considerada no projeto da rede, dificilmente um sistema de arquivos distribuído apresentará desempenho adequado para servir a todos os clientes após esse crescimento. Sistemas de arquivos paralelos permitem a escalabilidade do sistema apenas adicionando mais servidores de replicação na rede de comunicação (CARNS, et al. 2005)

Segurança

A possibilidade de compartilhar recursos é um requisito básico para o desenvolvimento de sistemas distribuídos, porém, torna-se necessária a adoção de mecanismos para garantir a segurança no acesso a áreas restritas de usuários e de sistema. O modelo de segurança fornece confiança e integridade às aplicações que executam na camada de usuários, garantindo mútua confiança entre o fornecedor de recursos e os clientes. Caso sejam submetidos compartilhamentos sem o nível de segurança habilitado a integridade dos sistemas computacionais pode estar comprometida por recursos maliciosos e reciprocamente, a integridade dos recursos pode estar violada por usuários maliciosos (BUTT, et al. 2002).

O controle no acesso aos arquivos é uma das medidas de segurança para protegê-los. Contudo, caso haja outros computadores no caminho entre dois computadores confiáveis, existe o risco de que dados sejam interceptados ou adulterados no caminho. Uma forma de se resolver esse problema é criptografar as informações antes de enviá-las (ANTHONY & CHRISTIAN, 2003).

Outros mecanismos para o controle da segurança no acesso e manipulação de dados armazenados remotamente podem ser encontrados em Galli (GALLI & DOREEN, 2000).

Operações Atômicas

Uma operação em um arquivo é dita atômica quando as etapas da mesma não podem ser percebidas por outros processos exteriores a esta operação (TANENBAUM, 1992). Assim, antes de uma operação o arquivo apresenta um estado e após, outro estado, sem que apresente nenhum estado intermediário durante o processo. Caso alguma etapa falhe durante a operação, o arquivo volta ao estado inicial.

Dessa forma, ou todas as etapas são realizadas com sucesso, ou nenhuma será realizada. Operações de leitura, escrita, criação ou remoção de um arquivo são implementadas de forma atômica pela maioria dos sistemas de arquivos distribuídos.

2.2.1 Gerenciamento de *Cache*

Dispositivos de armazenamento de dados, como discos rígidos apresentam desempenho consideravelmente inferior ao da memória principal (VOELKER, 1998), limitando o desempenho de aplicações paralelas e distribuídas com altas taxas de acesso à informação. Uma das soluções para esse problema consiste na utilização da memória principal para o armazenamento temporário de dados por meio de mecanismo de *cache*. Nesse contexto, é possível definir *cache* como uma técnica que possibilita armazenar os dados mais relevantes na memória, reduzindo a necessidade de acesso aos dispositivos de armazenamento. Embora essa solução apresente tempo de acesso significativamente menor, pode-se sobrecarregar e ocupar toda a memória principal. Para evitar esse problema, os seguintes parâmetros devem ser considerados (TANENBAUM, 1992):

- Determinações da relevância dos blocos de dados, verificando quais blocos têm maior número de acessos durante o processo examinado, melhorando dessa forma a eficiência do espaço reservado para *cache*;
- Utilização de mecanismos de gerenciamento de *cache*, mantendo os dados consistentes e atualizados;
- Utilização de políticas de propagação das alterações dos dados para os dispositivos de armazenamento;

Dados mantidos em *cache* devem ser organizados de forma a aumentar a eficiência de acesso. Em geral, são utilizados métodos simples de organização dos dados tais como listas encadeadas, listas duplamente encadeadas, árvores binárias e tabelas *hash*. Esses métodos não apresentam escalabilidade, o que geralmente influencia a adoção de algoritmos mais sofisticados, como árvores B, árvores B+, bitmaps e *frames*.

Para armazenar os dados mais relevantes em *cache*, devem-se utilizar políticas em *software* que manipulem a possível falta de espaço e sejam capazes de escolher quando e quais blocos são descartados. Além de servir para armazenamento temporário de dados, um sistema de *cache* pode ser utilizado com o mecanismo de pré-busca (*prefetching*) (DODONOV, MELLO & YANG, 2006).

2.2.2 Técnicas de *Prefetching*

Conhecendo a priori as futuras requisições das aplicações, pode-se carregar dados na memória com antecedência (KOTZ & ELLIS, 2001) minimizando a latência de acessos. A técnica que possibilita efetuar leitura antecipada de dados é denominada de *prefetching*, ou leitura antecipada. Como o mecanismo de *prefetching* é, geralmente, utilizado nos intervalos entre as requisições consecutivas de acesso aos dados pela aplicação, esse mecanismo também é conhecido como a redução do tempo ocioso.

A seqüência em que os dados são lidos caracteriza os diferentes padrões de acesso e, conhecendo esses padrões, um algoritmo de *prefetching* pode ser utilizado para efetuar a leitura antecipada de dados. A tarefa de determinar os padrões de acesso é fundamental e representa um dos maiores desafios para a utilização de *prefetching*. É possível dividir os padrões de acesso em seqüenciais e não-seqüenciais. (DODONOV, MELLO & YANG, 2006). Os primeiros acessam os dados em uma seqüência, enquanto os não-seqüenciais são caracterizados por acessos a regiões independentes de maneira pseudo-aleatória para a aplicação. Entre os principais padrões de acesso seqüenciais pode-se citar *one-block look-ahead, n-block look-ahead* e *infinite-block look-ahead.* Os padrões de acesso não-seqüenciais são dependentes de aplicações, sendo que a sua avaliação é mais complexa. A determinação dos padrões de acesso seqüenciais pode ser feita por meio de “dicas” de processos (*hints*) ou por meio de análise da seqüência de blocos lidos pela aplicação (KOTZ & ELLIS, 2001).

2.2.3 Tolerância a Falhas e Protocolos Consistentes

Existem dois tipos de implementações para um serviço de arquivos, acesso remoto e cópia remota, que podem usar um estado específico ou não. Na cópia remota de blocos de dados o cliente recebe uma cópia do arquivo para seu uso e quando terminado o processo pode retornar ou não as alterações ao servidor. É possível também que o serviço lide com replicações de arquivos, o que ajuda o acesso concorrente, permitindo que as aplicações utilizem mais rapidamente os dados, além de permitir o uso do computador quando o servidor estiver fora do ar (GUILHERMO, et al. 2001). Porém, esses aprimoramentos para o usuário final necessitam de diversas técnicas para tolerância a falhas e consistência de dados ao sistema de arquivos em rede.

A técnica mais intuitiva de coerência de memória é a consistência rígida, em que a operação de leitura retorna o valor de escrita mais recente. Este tipo de coerência é atingido somente quando existe uma noção global de tempo que possa fornecer uma ordem determinística para todas as leituras e escritas (KOTZ & ELLIS, 2001). A escrita mais recente é um conceito ambíguo em sistemas distribuídos, em que não existe tempo global, por isso foram desenvolvidas técnicas para manter a coerência em sistemas de arquivos compartilhados.

Muitas aplicações críticas exigem um regime de operação sem interrupções, assim como o *boot* de imagens que pode usar técnicas para diminuição de latência de memória com garantia de integridade dos dados (HILDEBRAND, WARD & HONEYMAN, 2006). A existência de redundância de *hardware*, inerente às arquiteturas paralelas, oferece um suporte natural às técnicas de tolerância a falhas.

Durante a transmissão dos dados entre servidores e clientes, podem ocorrer falhas, seja por excesso de tráfego de pacotes pela rede ou por algum dos servidores estar sobrecarregado. Além disso, podem ocorrer falhas de *hardware*, especialmente dos mecanismos de armazenamento e de transmissão. Esses problemas acontecem em grande parte porque os sistemas distribuídos são implementados sobre redes de computadores que não são totalmente confiáveis, por este motivo, a maior parte da complexidade de sua implementação está em levar em conta esse ambiente propício a falhas.

Um sistema distribuído tolerante a falhas deve usar um protocolo de comunicação com capacidade para detecção de erros de transmissão, assim como no TCP (*Transmission Control Protocol*), caso uma mensagem chegue alterada ou se perca até o seu destino, o protocolo percebe o erro e retransmite a mensagem. A preocupação também ocorre com os *hardwares* que podem apresentar falhas, por exemplo, um disco rígido pode deixar de funcionar de um momento para outro, seja por excesso de uso ou até mesmo por descargas elétricas. Nesse caso, é permitido criar soluções com redundância física do equipamento (realizada via *hardware*) ou redundância controlada pelo próprio sistema distribuído. O uso do protocolo UDP (*User Datagram Protocol*) é aplicado para garantir tolerância a queda de conexão, sendo que após a volta da conexão o fluxo é restabelecido sem precisar iniciar um novo *socket* como acontece no protocolo TCP.

Alguns problemas de desempenho aparecem ao lidar com a interface no nível do *hardware*. O SO (Sistema Operacional) não sabe diferenciar entre regiões "ativas" e "não-ativas" de memória. Como resultado, uma implementação ingênua de migração pode transferir muitas páginas de memória não usadas pelo SO convidado. A técnica *ballon driver* (CLARK, et al. 2005) pode ser utilizada nesse caso para descobrir as páginas que devem ser transferidas.

Independente do problema, o sistema deve evitar que o cliente fique aguardando uma resposta por muito tempo, ou que seus dados sejam danificados ou até mesmo perdidos, acarretando em problemas de disponibilidade e confiabilidade do sistema.

2.2.4 Sistemas de Arquivos Virtuais

O VFS (*Virtual File System*) tem grande utilidade em sistemas de arquivos, pois cria uma camada de abstração entre sistemas de arquivos concretos. O propósito do VFS é permitir que uma aplicação cliente utilize diferentes tipos de sistemas de arquivos de uma maneira uniforme. O VFS pode, por exemplo, pode ser utilizado para viabilizar o acesso a dispositivos locais de forma transparente na rede de comunicação, sem que a aplicação cliente note a diferença. A virtualização de sistemas de arquivo cria uma ponte entre diferentes tipos de sistemas de arquivos a partir de sistemas operacionais distintos como Windows, Mac-OS e Unix. Portanto, aplicações podem usar os arquivos sem se preocupar com os tipos de sistemas de arquivos existentes.

Para que haja uma comunicação entre o *kernel* e o sistema de arquivos real o VFS cria uma interface (ou um contrato), o que possibilita incluir suporte para outros sistemas de arquivos concretos apenas adicionando mais um contrato. O VFS garante que arquivos possuam identificações numéricas únicas, denominadas vnodes, mesmo dentro de um ambiente de rede. Os vnodes separam as operações no sistema de arquivos da semântica de sua implementação.

Acima da interface com o VFS, o sistema operacional opera em termos de vnodes; abaixo desta interface, o sistema operacional pode ou não implementar inodes. A interface VFS pode conectar o sistema operacional a uma grande diversidade de sistemas de arquivos, como por exemplo, FAT (*File Allocation Table*) (FAT, 1980), HPFS (*High Performance File System*) (HPFS, 1989), NTFS (*New Technology File System*) (NTFS, 1998) e EXT3 (*Third Extended Filesystem*) (EXT3, 1999).

2.3 Sistemas de Arquivos Distribuídos

Os SAD (Sistemas de Arquivos Distribuídos) possibilitam o compartilhamento de diretórios através da rede como se fossem diretórios locais, permitindo a execução de programas e sistemas operacionais com recursos de rede. A maior vantagem desse modelo é permitir que os dados sejam requisitados somente quando necessários, podendo ser compartilhados por diversos computadores simultaneamente. Também é possível o compartilhamento de recursos da rede e dispositivos de armazenamento, o que possibilita a redução de custo com leitores de mídia e discos rígidos nos computadores que inicializaram através da rede, assim como garantir a integridade das operações realizadas nos arquivos em redes de comunicação.

Há diversas formas de armazenamento de arquivos que podem influenciar no desempenho de leitura e escrita nos mesmos, por exemplo, arquivos que compõem um banco de dados podem ser armazenados em formato de registros, arquivos que são usados por aplicações multimídia ou imagens de máquinas virtuais costumam ser armazenados em formato contínuo no disco para otimizar sua leitura.

As propriedades dos arquivos também são garantidas pelo sistema de arquivos distribuídos, como permissões de leitura, escrita e execução, além de qualquer outra informação relevante, também chamados de metadados.

A seguir serão apresentados alguns dos sistemas de arquivos distribuídos mais pesquisados.

2.3.1 NFS

O NFS (*Network File System*) (SUN MICROSYSTEMS INC 1989) (HITZ, LAU & MALCOLM, 1994) é o serviço de rede mais importante que utiliza o RPC (*Remote Procedure Call*). Esse sistema permite a utilização de arquivos que se encontram em computadores remotos, exatamente da mesma maneira como um usuário requisita qualquer arquivo localmente. Isto é possível graças à combinação de funcionalidades do sistema operacional no lado do cliente (que usa o sistema de arquivos remoto) e um servidor NFS no lado do servidor (que provê os arquivos de dados). Este acesso ao arquivo é completamente transparente ao cliente e funciona com uma variedade de servidores e arquiteturas de computadores.

Na primeira e segunda versões do NFS, o cliente era responsável pelo controle de acesso aos arquivos, sem nenhuma validação por parte do servidor. Essa característica foi modificada no NFSv3, em que o servidor passou a tomar tal decisão automaticamente, usando o mesmo esquema de segurança dos sistemas de arquivos locais Unix. Após alguns anos do lançamento da especificação do protocolo NFSv3, foi criada a versão NFSv4 que apresenta novas características em segurança e *cache*, as quais provocam alterações conceituais no que se conhecia até então sobre o NFS (KON, 2007).

No NFSv4, o servidor mantém o estado dos arquivos em conjunto com os clientes, diferentemente das versões anteriores. Assim, é possível que um determinado cliente pergunte ao servidor o que outros clientes estão fazendo com determinado arquivo. Isso pode indicar à máquina cliente que é necessário ou não realizar um *cache* dos dados de forma mais agressiva.

Uma característica do NFS é a transparência que o sistema de arquivos possibilita ao usuário final, que nem sequer percebe estar lidando com arquivos remotos. No NFSv4 os controles de bloqueios e estado são nativos do protocolo, isso é ainda mais evidente, dando a impressão de que se está usando o sistema de arquivos local. Além dessas características, o fato de ter sua especificação aberta para que qualquer pessoa possa implementar seu servidor ou cliente permitiu que o NFS se

tornasse o sistema de arquivos distribuído mais utilizado (HITZ, LAU & MALCOLM, 1994).

2.3.2 AFS

O projeto AFS (*Andrew File System*) (HOWARD, 1988) foi iniciado em 1983 na Universidade Carnegie-Mellon, com apoio da IBM. Seu objetivo era projetar e implementar um sistema distribuído para o ambiente acadêmico de ensino e pesquisa, que ofereceria a cada professor e aluno uma estação de trabalho com um sistema operacional compatível com o UNIX BSD. Além dessa característica seria possível, a partir de qualquer um dos computadores com AFS, ter a mesma visão do sistema. Essa transparência de localização deveria ser altamente escalável, podendo aceitar de 5 a 10 mil estações de trabalho nessa rede.

Diferentemente do NFS, toda informação sobre os nomes dos arquivos e diretórios é armazenada nos servidores do AFS. Deste modo, a manutenção dos clientes é realizada facilmente e a uniformidade do espaço de nomes é uma conseqüência natural da configuração dos servidores.

Quando um cliente necessita de um arquivo remoto, ele pergunta a todos os servidores por sua localização, que é então guardada em *cache* local para futuras consultas (KAZAR & LEON 1988).

A fim de atingir seus objetivos, foram adotadas algumas regras para o desenvolvimento do AFS (HOWARD, 1988):

- As operações devem ser realizadas no cliente, e não no servidor, distribuindo assim as tarefas entre as máquinas disponíveis, evitando sobrecarregar o servidor;

- Utilização de *cache* para diminuição de tráfego dos dados e a carga dos servidores;

- Manter arquivos temporários na máquina local, replicar nos servidores arquivos executáveis que são muito usados e raramente alterados;

- Replicar serviços e informações sempre que possível, evitando limitar a escalabilidade de todo o sistema à capacidade da máquina central;

- Confiar no menor número possível de entidades (segurança);

- Agrupar os procedimentos sempre que possível. Por exemplo, realizar uma leitura de 30 KB é muito mais eficiente que realizar 30 leituras de 1 KB.

O AFS-3, ou simplesmente AFS, foi uma iniciativa de tornar o sistema comercial no início de 1990. Para tanto, era necessário adotar alguns padrões, como por exemplo, o VFS da SUN, possibilitando integrá-lo a outros sistemas de arquivos, o que acabou inviabilizando sua área de atuação.

2.3.3 CODA

O CODA (*Constant Data Availability*) (SATYNARAYANAN, 1989) começou a ser desenvolvido em 1987 pela Universidade de Carnegie Mellon, EUA. Desenvolvido a partir do AFS, o CODA oferece uma alta disponibilidade através da adoção de uma técnica otimista de replicação. A grande peculiaridade deste sistema são as ferramentas que permitem o acesso ao sistema de arquivos a partir de computadores portáteis e a possibilidade de operacões desconectadas durante a qual os clientes podem utilizar o sistema de arquivos sem estabelecer contato com os servidores (KISTLER & SATYANARAYANAN, 1992).

Para que os arquivos sejam acessados mesmo com o servidor desligado, o CODA implementa alguns mecanismos de replicação não presentes no AFS, uma vez que foi criado para lidar com estações de trabalho portáteis ou que permanecem conectadas dos servidores por curtos períodos de tempo.

Cada volume do CODA (um conjunto de diretórios do sistema de arquivos) é associado a um VSG (*Volume Storage Group*), que consiste de um conjunto de servidores que replicam o volume. Este conjunto de servidores acessíveis é chamado de AVSG (*Accessible Volume Storage Group*). O sistema é separado por volumes e caso um servidor esteja indisponível outro assume seu lugar. Esse modelo é mostrado na Figura 1 (SATYNARAYANAN, 1989).

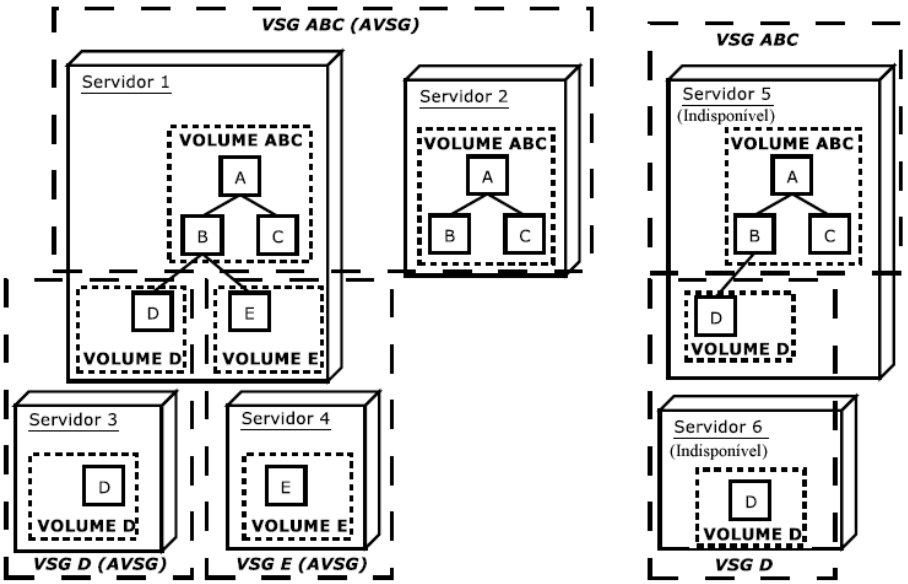

Figura 1 - Distribuição dos volumes entre os servidores no sistema de arquivos distribuídos do CODA

2.4 Sistemas de Arquivos Paralelos

Nos últimos anos, a evolução dos processadores e redes para computadores de baixo custo foi maior se comparada com o aumento do desempenho dos discos de armazenamento de dados. Com isso, muitas aplicações encontram dificuldades em atingir o pleno uso dos processadores, pois estes necessitam esperar até que os dados cheguem para serem utilizados. Uma forma de resolver esse tipo de gargalo é a adoção de Sistemas de Arquivos Paralelos (SAP), que utilizam a velocidade da rede local, além dos recursos de cada máquina, para suprir a deficiência de desempenho no uso isolado de cada disco.

As seções a seguir apresentam uma breve descrição sobre alguns dos sistemas de arquivos paralelos mais utilizados.

2.4.1 PVFS

O PVFS (*Parallel Virtual File System*) (CARNS, et al. 2000) é um sistema de arquivos paralelos escalável e de código aberto, projetado para trabalhar com grandes volumes de dados em pesquisas acadêmicas e ambientes de produção. O PVFS utiliza grandes buffers de dados nas transferências para prover alto desempenho e

escalabilidade paralela em aglomerados de computadores. Em geral, o PVFS oferece quatro características:

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

Para um sistema de arquivos paralelo ser utilizado de maneira eficiente, ele deve prover um espaço de nomes único em todo o aglomerado de computadores, e deve ser possível acessá-lo através dos utilitários mais comuns. Para prover acesso de alto desempenho para os clientes do aglomerado, os dados armazenados no PVFS estão distribuídos entre os vários nós que compõem o aglomerado, assim como o BRIDGE (DIBBLE, SCOTT & ELLIS, 1988) que foi um dos primeiros sistemas de arquivos que considerou distribuir não somente a carga entre os servidores, mas também partes dos arquivos.

Cada um dos nós do PVFS é chamado de I/O node. Separando os dados em vários nós, as aplicações passam a ter mais caminhos para encontrá-los, passando pela rede e chegando aos discos em que estão armazenados, em diferentes computadores. Isso elimina o gargalo de E/S e aumenta o potencial total da banda para múltiplos clientes.

Utilizar mecanismos tradicionais de chamadas de sistema para acesso a arquivos pode ser bem conveniente, mas pode causar uma sobrecarga para o sistema como um todo, especialmente o *kernel*. Assim, é possível copiar os arquivos do PVFS utilizando uma API (Application Programming Interface) nativa. Existem bibliotecas com as operações mais utilizadas que requisitam diretamente os serviços PVFS, evitando a necessidade de utilização do *kernel* local. Essas bibliotecas, como a ROMIO MPI-IO, podem ser utilizadas pelas aplicações ou por outras bibliotecas para acesso de alta velocidade ao PVFS (ROSS, LIGON & CARNS, 2002).

2.4.2 NFSP

Criado e atualmente mantido pelo *Laboratoire Informatique et Distribution* do *Institut d'Informatique et Mathématiques Appliquées de Grenoble* (IMAG), França, o NFSP (*Netware File Sharing Protocol*) (LOMBARD & DENNEULIN, 2001) representa a união da estabilidade, confiabilidade e compatibilidade do NFS com o alto desempenho do PVFS. A idéia é aplicar todos os objetivos do PVFS através do uso do seu próprio código dentro da implementação do NFS.

Da mesma maneira que no PVFS, o objetivo do NFSP é oferecer alto desempenho em aglomerados de computadores, de forma transparente para os aplicativos clientes, que enxergam os vários servidores como sendo um sistema de arquivos único e consistente. O NFSP usa o mesmo modelo de servidores do PVFS, ou seja, um servidor de metadados (contendo informações sobre data de modificação e permissões de acesso), e vários servidores de dados dedicados, em que os dados dos arquivos a serem armazenados se encontram distribuídos.

A diferença fundamental com relação ao PVFS é o protocolo de comunicação entre servidores e clientes, que é o NFS. Dessa forma, é possível utilizar qualquer cliente NFS para copiar e manipular as informações contidas nos servidores, diminuindo de forma considerável a manutenção e administração dos computadores clientes.

O NFSP utiliza o protocolo NFSv2 (RFC1094) em sua estrutura, construído sobre os mecanismos do SunRPC, utilizando o formato XDR (*External Data Representation*) para dados. O SunRPC pode ser usado tanto sobre comunicações UDP como TCP. Através desse modelo qualquer cliente NFS pode utilizar o NFSP como se estivesse usando um servidor NFS convencional. No lado do servidor há dois processos em execução, o rpc.mount, que implementa o protocolo para inicializar o sistema de arquivos, e o rpc.nfsp que é o responsável por tratar as requisições dos clientes e transmitir as informações solicitadas. No lado do cliente basta ter um cliente NFS compatível para montar o sistema de arquivos.

A segurança aplicada no NFSP é a mesma apresentada no NFS e PVFS, que consiste em verificar as permissões de acesso aos arquivos de forma não segura. Mas, assim como no PVFS, esse é um sistema de arquivos desenvolvido para acesso de uma

rede segura, como um aglomerado. Existem implementações iniciais de redundância dos dados armazenados através da adoção de VIOD (*Virtual* IOD) ao invés de IOD (IO *Deamon*). Um IOD é um nó do sistema de arquivos usado para armazenar dados. Um VIOD representa vários IODs que possuem os mesmos dados, ou seja, trocam cópias entre si dos dados que armazenam. O intuito não é apenas criar alta disponibilidade, mas também proporcionar melhor acesso concorrente, ao distribuir a carga entre os IODs do mesmo VIOD.

O NFSP partiu da idéia do uso de vários sistemas de arquivos locais que estariam distribuídos em uma rede de computadores, podendo requisitá-los de uma forma simples, usando clientes já prontos, diminuindo assim a tarefa de administração e manutenção destes. Para isso, foi criado um servidor NFS que possui características do PVFS, ou seja, é um sistema de arquivos paralelo que distribui os dados dos arquivos entre vários servidores, além de fornecer uma interface já bem conhecida para comunicação com seus clientes.

2.4.3 pNFS

O pNFS (*Parallel Network File System*) é uma versão de alto desempenho do NFSv4 que fornece acesso direto aos armazenamentos de sistemas de arquivos paralelos preservando o sistema operacional e plataforma de independência de *hardware*. Esse sistema de arquivos pode aumentar a taxa de escrita em arquivos independente do seu tamanho superando ineficiências de sistemas de arquivos paralelos (HILDEBRAND, WARD & HONEYMAN, 2006).

Embora sistemas de arquivos paralelos separem a camada de controle do fluxo de dados, há uma forte integração entre controle e protocolos de dados, dessa forma os usuários precisam se adaptar a diversas semânticas para cada repositório de dados. O pNFS permite que as aplicações usem semânticas de arquivos de sistema comuns entre os repositórios e o protocolo de metadados do NFSv4 fornece um patamar de consistência dentro de cada repositório de sistema.

Algumas semânticas são dependentes de camadas de *drivers* e protocolos de armazenamento que podem alterar o modo como as aplicações se comportam. Por exemplo, o Panasas Activescale (PANASAS, 2004) que suporta o protocolo seguro OSD (WEBER, 2004), força os clientes a instalarem autenticações, integrações e

mecanismos de privacidade. Ao se alterar alguma característica dependente de camadas de *drivers* é possível que clientes sem essa característica não consigam mais executar leituras e escritas ao sistema de arquivos distribuído. Uma maneira de estabelecer maior independência de protocolos é o uso de um sistema híbrido com sistema de arquivos virtual, por exemplo, pNFS/PVFS, detalhado na seção a seguir.

2.4.4 Sistema de Arquivos Híbrido (pNFS / PVFS2)

Um dos maiores problemas de sistemas de arquivos distribuídos é garantir robustez, escalabilidade e abstração de recursos. Diversos sistemas de arquivos distribuídos modernos como o AFS, CODA, xFS, NFS, NFSP, pNFS, PVFS foram pesquisados. Muitas características, como sistema de arquivos virtual e escalável só foram atingidas com um sistema de arquivos híbrido de PVFS2 com pNFS, como encontrado no projeto Direct-pNFS (HILDEBRAND & HONEYMAN, 2007).

Um benefício do pNFS é a capacidade de ser compatível com sistemas de armazenamento nativos de computadores clientes enquanto continua a suportar todas características do NFSv4. O modelo utiliza o pNFS como camada de abstração e o PVFS2 para executar o balanceamento de carga. O servidor de balanceamento distribui o tráfego entrante ou requisições de recursos provenientes dos nodos que executam os mesmos programas entre computadores do sistema, como visto na Figura 2 (HILDEBRAND, WARD & HONEYMAN, 2006).

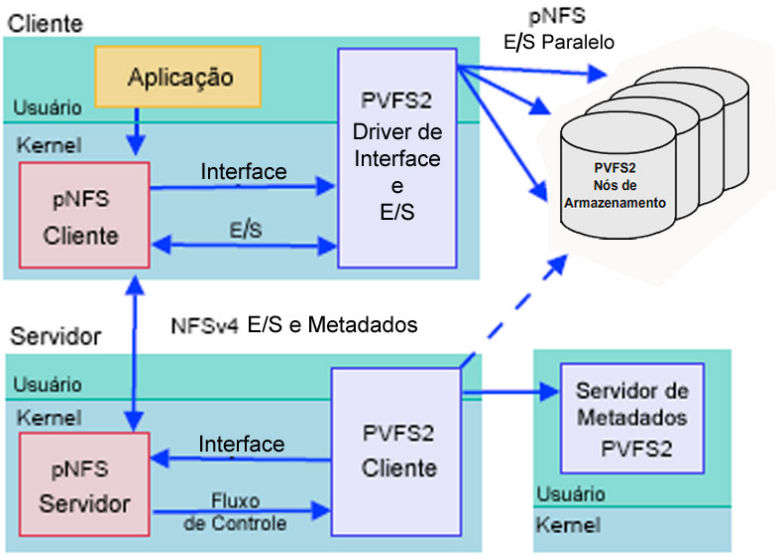

Figura 2 - Estrutura da implantação do pNFS com PVFS2

O PNFS estende o NFSv4 adicionando um I/O *driver* e uma interface ao sistema de arquivos de retorno. O servidor pNFS obtém um mapa de um arquivo do sistema de armazenamento e transfere-o para o cliente PNFS e subseqüentemente para o *driver* que permite o acesso paralelo ou direto aos dados.

O cliente pNFS utiliza o *driver* do PVFS2 para toda operação de E/S. O servidor pNFS obtém o arquivo do servidor de metadados e transfere de volta para o cliente pNFS, depois para a camada de *driver* PVFS2 que faz todo acesso aos dados de forma paralela ou direta dependendo do tamanho do arquivo requisitado, como visto na Figura 3 (HILDEBRAND, WARD & HONEYMAN, 2006).

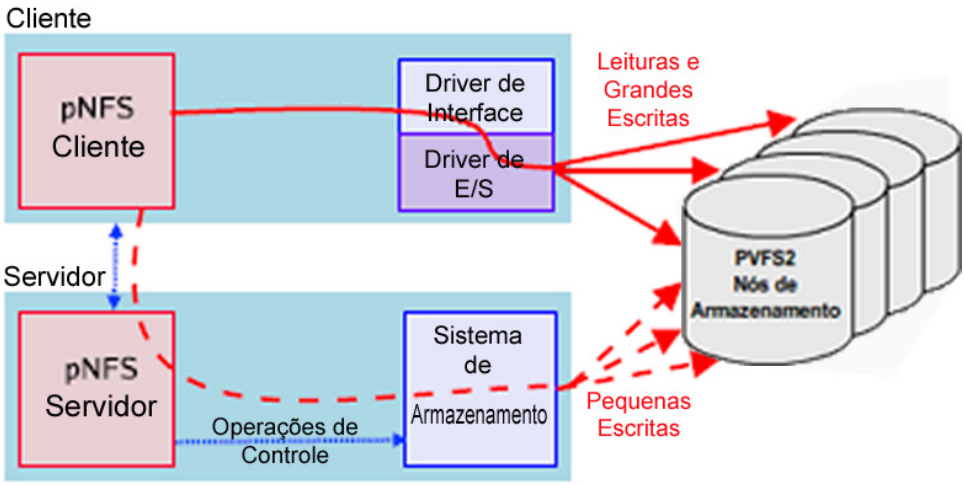

Figura 3 - Modelo de escrita entre pNFS e PVFS2

Todos os nós estão responsáveis em controlar os pedidos. Se um nó falhar, as requisições são redistribuídas entre os nós disponíveis no momento. Este tipo de solução é normalmente utilizado em conjuntos de servidores de web, o balanceamento de carga é necessário para desencadear a distribuição dos arquivos de grande porte no servidor e dissipar o trabalho entre os computadores da rede, sendo assim, computadores auxiliares podem realizar a distribuição de blocos de dados quando já carregado em memória.

2.5 Considerações Finais

Este capítulo apresentou as características fundamentais de sistemas de arquivos distribuídos e paralelos como forma de estabelecer transparência e escalabilidade no acesso aos dados remotos. Muitos serviços oferecidos pelos sistemas de arquivos distribuídos são tão transparentes aos usuários que estes nem percebem a complexidade

envolvida no acesso aos seus dados. Infelizmente tal complexidade gera custos, que costumam estar associados ao baixo desempenho do sistema como um todo.

Cabe ao projetista e aos desenvolvedores dos sistemas de arquivos a tarefa de ponderar desempenho e qualidade de serviço, para suprir as necessidades dos usuários conforme as aplicações às quais se destina tal sistema de arquivos. Tecnologias existentes de acesso a dados como NFS, pNFS, NFSP, PVFS podem falhar em satisfazer todos requisitos, deficiência essa que pode ser resolvida com a implantação de sistemas híbridos como o pNFS / PVFS2.

Capítulo
3

Clusters Multiuso, *Grids* e Computação *Commodity*

3.1 Considerações Iniciais

Nas últimas décadas, o poder de processamento dos computadores pessoais cresceu constantemente, seguindo a taxa de crescimento exponencial da lei de Moore (MOORE, 1965). Componentes *commodity* já são uma realidade (processadores, memórias, conexões de rede e discos) que resultam em uma relação de preço e desempenho mais atrativa para os sistemas construídos a partir desses componentes.

Nesse intuito, a computação *commodity* ganhou espaço em pesquisas acadêmicas e sistemas distribuídos que necessitam de recursos livres para serem executados. Atualmente é possível criar *clusters* e *grids* a partir de laboratórios de informática e computadores pessoais, aproveitando assim, recursos pouco utilizados durante a maior parte do dia.

Neste capítulo são apresentadas tecnologias importantes na evolução da infra-estrutura de sistemas de computadores distribuídos aplicados à computação *commodity*: *desktop grids* e *clusters* multiuso (KURMANN, RAUCH & STRICKER, 2003).

3.2 *Boot* Remoto

O sistema de *boot* remoto permite que sistemas operacionais em máquinas clientes sejam inicializados através de requisições a um ou mais servidores, melhorando dessa forma a manutenção e gerenciamento de *software* dos computadores e facilitando a criação de sistemas distribuídos que utilizam *clusters e grids*.

Algumas características encontradas no *boot* remoto são:

- Eliminação de instalações individuais de cada sistema operacional de um grupo de computadores, conceito "*do-one-used-by-all*", faça uma vez e use para todos;
- Gerenciamento de *software* é facilitado, pois só precisa ser atualizado no sistema operacional base;
- Permite facilmente recuperar e realizar *backup´s* dos sistemas operacionais utilizados pelos usuários. Dados dos usuários também podem ser centralizados no servidor;
- É flexível por ser utilizado tanto para gerenciamento de computadores de trabalho como ambientes em *clusters*;
- Permite a implementação de *boot diskless*, ou seja, as estações clientes não necessitam de disco rígido para a inicialização do sistema operacional. A ausência de disco se traduz em uma redução significativa no custo inicial de um computador (cerca de 1/6) e uma redução significativa na taxa de avarias do equipamento;

As técnicas de *boot* remoto utilizam *stream* de *software* e gerenciamento de *cache* para melhorar a usabilidade do sistema (EYLON, et al. 2001). Duas das técnicas utilizadas nesse intuito são o *block stream* (KUACHAROEN, MOONEY & MADISETTI, 2002) e o DSM (*Distributed Shared Memory*) (PROTIC, 1996).

Para implementar um *boot* remoto podem ser utilizadas diferentes tecnologias. As seções a seguir apresentam algumas implementações existentes.

3.2.1 LTSP

O LTSP (*Linux Terminal Server Project*) (MCQUILAN, 2002) é uma solução de código aberto que agrega um conjunto de serviços para *boot* de máquinas clientes com ou sem recursos de armazenamento e com pequeno poder de processamento local, técnica conhecida como *thin-client* (BARATTO, KIM & NIEH, 2005). Isto possibilita sua implementação em computadores com configuração de *hardware* mínima, só é

necessário um servidor com alto processamento e boa quantia de memória física e volátil.

A solução tem baixo TCO (*Total Cost of Ownership*), ou seja, é barata para ser implementada e mantida por administradores de rede, geralmente utilizada para reaproveitar *hardwares* que estão obsoletos por não priorizar o uso de recursos locais dos micros clientes.

3.2.2 DRBL

O DRBL (*Diskless Remote Boot Linux*) (DRBL, 2002) é uma solução de código fonte aberto para gerenciar a implantação de sistemas operacionais Linux em múltiplos clientes. O DRBL usa PXE (*Preboot Execution Environment*) (INTEL CORPORATION, 1999), NFS (HITZ, LAU & MALCOLM, 1994) e NIS (*Network Information Service*) (BROWN, 1998) para fornecer o acesso aos computadores clientes.

O sistema inicializado não faz requisição ao disco rígido local, somente aos dados do servidor, portanto, caso a máquina cliente tenha disco rígido, os sistemas operacionais nele instalados não serão afetados. A unidade de disco rígido é opcional para um cliente DRBL, e quando disponível pode ser configurado para fazer *swap* ao se necessitar de mais memória.

As máquinas clientes podem ser inicializadas em dois modos (BLAKE, et al. 2004):

1. Modo somente leitura: Todos os dados do sistema operacional ficam protegidos em modo somente leitura, portanto, ao reinicializar a máquina o sistema operacional estará intacto, facilitando o gerenciamento da manutenção do grupo de computadores caso usuários ou *softwares* realizem operações que prejudiquem o funcionamento do sistema;

2. Modo escrita: Esse modo serve para que o sistema operacional seja liberado para operações de escrita, instalação de programas e novas configurações são permitidas, assim, é possível gerenciar as imagens dos sistemas operacionais que estarão disponíveis no modo somente leitura;

3.2.3 SLIM

O conceito fundamental do SLIM (*Single Linux Image Management*) (SLIM, 2004) é o gerenciamento centralizado de sistemas operacionais com rapidez e leveza na inicialização. Todos os clientes compartilham a mesma imagem, instalação e administração do ambiente. Tanto o DRBL quanto o SLIM aproveitam o uso do processamento local nos clientes, diferentemente de soluções *thin-client* (LAY & NIEH, 2006), em que o processamento é centralizado no servidor.

Os requisitos para incluir um computador no *boot* remoto são: habilitar o *boot* através de PXE, conectar o cabo de rede na mesma rede do servidor e ligar o computador. Caso o cliente não suporte o *boot* através de PXE pode ser inserido um disquete com as informações de *boot*. Há três formas de se inicializar clientes no SLIM:

1. **Cópia total à RAM:** O sistema operacional é todo copiado para a memória volátil do computador, após isso a máquina não necessita mais de requisições ao servidor, aumentando a localidade das informações e conseqüentemente a velocidade de processamento. Porém, o *boot* é mais lento já que necessita copiar todo sistema operacional para a RAM.

2. **Cópia total para o disco rígido (caso existir):** O sistema operacional é todo copiado para o disco rígido e após a cópia não há mais necessidade de acesso ao servidor. Ocorre melhora na localidade dos dados, porém, é mais lento que a cópia para RAM, já que a velocidade dos discos rígidos é inferior a das memórias.

3. **Cópia quando necessário:** Este modelo copia apenas partes do sistema operacional que estão sendo usadas no momento e armazena em *cache*, dessa forma a máquina é inicializada com rapidez, porém, todo acesso a áreas que não estão em *cache* devem ser requisitadas no servidor, deixando a usabilidade do sistema operacional mais lenta.

Um servidor pode suportar até centenas de máquinas com um desempenho considerável como visto na Figura 4 (ROY, et al. 2004). Nesse ambiente um único servidor foi responsável por distribuir o sistema operacional Fedora Core 3 para os

clientes e verificar que tanto a cópia para o disco rígido quanto a cópia para RAM demoram mais para inicializar o sistema operacional conforme novos clientes são acrescentados ao servidor.

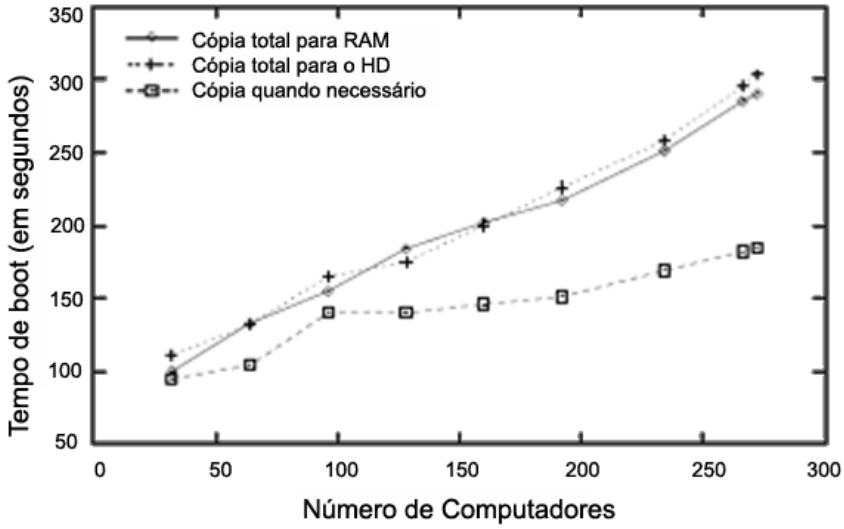

Figura 4 - Tempo de *Boot* vs. Número de Computadores

3.3 *Clusters* Multiuso

Devido à evolução do mercado de grandes sistemas de armazenamento e de componentes dos computadores, nós de *clusters* e computadores *desktop* acabaram se aproximando em sua estrutura. A diferença primordial entre computadores *desktop* e nós de um *cluster* está centrada em suas operações e modos de uso.

Tradicionalmente *clusters* são instalados em *racks* de salas designadas para servidores. Computadores *desktop*, por outro lado, são instalados em escritórios e laboratórios. A

Tabela 1 apresenta a diferença entre c*lusters* e computadores pessoais. *Clusters* tradicionais são estáticos em seu uso, geralmente executam com um sistema operacional específico nas máquinas que compõem o grupo e são ociosos durante a maior parte do tempo. Já o *cluster* multiuso pode ser utilizado para vários propósitos, desde máquinas para escritórios durante o dia até nós de armazenamento e processamento durante as noites e horários fora do expediente, como pode ser observado no *cluster* Beowulf (BEOWULF, 1994).

Tabela 1 - Diferença entre *clusters* e computadores pessoais

	Cluster	Computador Pessoal
Instalação	Montado em *rack*	Mesa
Ambiente	Sala de Servidores	Laboratórios, Escritórios
Tempo de operação	O dia inteiro	Horário comercial
Interação com usuário	Sem monitor, teclado e mouse	Monitor, teclado e mouse

Para que grupos de usuários com propósitos diferentes usem o mesmo grupo de computadores é necessário que haja troca de sistemas operacionais com muita facilidade, e essa característica é obtida facilmente com *cluster* multiuso aplicando distribuição automática de sistemas operacionais conhecido como COD (*Cluster On Demand*) (MOORE, et al. 2002), permitindo que ao reiniciar o computador seja possível escolher a imagem de *boot* que se encontra no servidor. Diferentes classes de estudantes geralmente têm variações em suas demandas por *softwares* instalados e configurações previamente implementadas por administradores de infra-estrutura. Durante o dia pode-se usar um ambiente com *softwares* para adolescentes e à noite *softwares* para estudo de adultos, por exemplo. Os nós do *cluster* permitirão com que dezenas de ambientes sejam configurados para cada necessidade dos grupos de pessoas.

A criação de uma imagem do sistema operacional aplicada no *boot* remoto é parecida com instalações convencionais e os sistemas operacionais são utilizados sem diferenças estruturais. A imagem criada pode ser armazenada para *backup* e aplicada a futuras restaurações do sistema operacional, dessa forma, mesmo que o sistema operacional pare de funcionar é possível restaurá-lo novamente (ZHOU & XIE, 2006).

Um típico *cluster* multiuso de computadores *commodity* é o *cluster* da Patagônia no Departamento de Ciência de Computação em Zurich, formado de *hardware commodity* do maior fabricante de computadores *desktop*. As máquinas foram totalmente conectadas através de rede Gigabit no ano de 2000 (STRICKER, 2000). A instalação permitiu a existência de múltiplos sistemas operacionais até então instalados nos discos rígidos, ou seja, duas instalações diferentes de Windows e duas instalações de Linux para pesquisa e educação (RAUCH, KURMANN & MULLER-LAGUNEZ, 1999). Foi utilizado Oberon para gerenciamento da baixa atualização de SO (REISER, 1991) (GUTKNECHT & WIRTH, 1992). O *cluster* foi instalado em uma sala de aula e

foi utilizado pelos estudantes da faculdade durante o dia, e durante a noite e fins de semana estudantes da pesquisa acadêmica utilizavam o *cluster* em pesquisas científicas e experimentos da área.

Outro *cluster* multiuso criado a partir de computadores *commodity* é o *cluster* Xibalta (XIBALTA, 2003) que foi instalado durante o verão de 2001 no Departamento de Ciência da Computação de Zurich. O modelo não é uma representação típica de *cluster* multiuso com computadores *desktop* já que é instalado em *racks* no centro de computação, porém é composto de *hardware* e *software commodity* e usado por diferentes grupos com diferentes *softwares* e sistemas operacionais. Os diferentes grupos de usuários usam características de preferência do *cluster* nos trabalhos diários, mas para testes computacionais de grande escala todos os nós do *cluster* são reiniciados em um ambiente desejado (KURMANN, RAUCH & STRICKER, 2003). O resultado deste modelo é a facilidade de criar um *cluster* com *hardware commodity* que pode ser comprado em qualquer loja de computadores. Com a miniaturização e integração dos componentes, os computadores *commodity* incluem características embutidas como interconexão de rede, cartões gráficos e adaptadores de disco. Através destas características integradas a diferença entre nós *clusters* e computadores interpessoais se tornam menores.

O uso flexível dos computadores como *clusters* multiuso se depara com alguns problemas e questões interessantes em pesquisas de sistemas operacionais. Para melhorar o acesso às informações distribuídas em servidores é importante a necessidade de um modelo de análise para predição de comportamento em *stream* de dados (DODONOV, MELLO & YANG, 2006) que fluam entre os nós do *cluster*.

Outra característica a ser implementada é a replicação de dados em sistemas de arquivos paralelos como PVFS (CARNS, et al. 2000), por exemplo, para evitar gargalos de busca a dados. As aplicações que lidam com armazenamento precisam gerenciar questões de acesso em diferentes nós do *cluster*. Esses pontos de pesquisa também podem ser melhorados para alto desempenho de E/S através de SAN (*Storage Area Networks*) (TELIKEPALLI, DRWIEGA & YAN, 2004), NASD (*Network Attached Storage Devices*) (MENASCÉ, PENTAKALOS & YELENA, 1996) e RAID (*Redundant Arrays of Inexpensive Disks*) (CHEN & TOWSLEY, 1996).

3.4 Ambientes em *Grid*

O *Grid* é uma infra-estrutura de computação que conecta múltiplos recursos computacionais de diversos computadores para permitir a execução de aplicações com alta demanda de recursos computacionais, memória e espaço em disco. A idéia do *grid* é utilizar todo recurso computacional ocioso para permitir que novos tipos de aplicações sejam criados ou estender aplicações existentes utilizando recursos de sistemas distribuídos altamente conectados (FOSTER, KESSELMAN & TUECKE, 2001).

Desktop grids podem utilizar *sandboxes* (FIGUEIREDO & DINDA, 2002), permitindo que seja possível o uso de máquina virtual para execução de aplicações paralelas assim como no projeto VMPLANTS (KRSUL, GANGULY & ZHANG, 2004). Também é possível implementar a utilização de algoritmos de escalonamento em *grids* para obtenção de um *middleware grid* mais flexível e robusto (FONSECA, et al. 2005) (KAPADIA, FIGUEIREDO & FORTES, 2001).

A diminuição das diferenças em *hardware commodity* entre os nós de um *cluster* e computadores pessoais está permitindo a inclusão de ambientes em *grid*, ou seja, o grande aumento de desempenho dos *hardwares commodity* sugere unir o poder de um acúmulo de computadores pessoais em um recurso de *hardware* compartilhado. Os *grids* podem ser executados tanto em redes WAN (Wide Area Network) como em redes em LAN (Local Area Network), quando junto com *clusters* multiuso são utilizados unicamente como LAN *grids*, possibilitando o aproveitamento dos recursos locais da rede (FIGUEIREDO & FORTES, 2004).

Grids e *clusters* multiuso não são mutuamente exclusivos, é possível inicializá-los em conjunto, o *cluster* multiuso pode ser usado durante o dia para tarefas de trabalho ou estudo juntamente com um *middleware* de *grid* configurado e executando no plano de fundo. As mesmas máquinas também podem ser utilizadas durante a noite em períodos fora do expediente, garantindo uma quantia de recursos computacionais disponíveis e facilmente gerenciáveis com *clusters* e que podem ser aplicados a *grids*.

3.5 Considerações Finais

A ampla utilização de computadores *commodity* conduziu a sistemas computacionais poderosos, superando sistemas tradicionais com um preço atrativo. Em

diversos grupos de computadores existem quantias enormes de poder computacional e armazenamento instalado, porém, a maioria desses recursos computacionais é mal utilizada. A utilização desses recursos pode ser otimizada com a adoção de *clusters* multiuso e *grids desktop*. Um ambiente mais flexível e escalável de distribuição de sistemas operacionais simplifica o gerenciamento dos recursos distribuídos e habilita seu uso para computação de alto desempenho.

Cluster multiuso e *grids commodity* podem ser utilizados além de pesquisas acadêmicas, por exemplo, em empresas ou laboratórios de informática. *Cluster* multiuso ainda permite maior organização do gerenciamento de *software*, maior poder de processamento e armazenamento centralizado.

Capítulo
4

Máquinas Virtuais

4.1 Considerações Iniciais

As técnicas de virtualização têm grande importância para projetos de sistemas computacionais, utilizadas com o objetivo principal de desacoplar os recursos físicos de um *hardware* da sua representação lógica. (BARHAM & DRAGOVIC, 2003). Este capítulo apresenta os conceitos das VM´s (Máquinas Virtuais), assim como modelos de implementação, migração e segurança aplicados a sistemas virtualizados.

4.2 Estrutura das Máquinas Virtuais

Para que um sistema computacional seja virtualizado, é necessário um componente de gerenciamento, denominado VMM (*Virtual Machine Monitor*) ou *hypervisor*, o qual consiste de uma camada de *software* que provê uma interface entre os recursos reais e as máquinas virtuais, fornecendo uma abstração dos recursos computacionais para as mesmas (ROSE, 2004). O VMM virtualiza a CPU, o sistema de gerenciamento de memória e o sistema de entrada/saída (E/S).

O gerenciamento de aplicações que realizam poucas operações de E/S é pouco penalizado pela camada de virtualização aplicada no sistema operacional, já aquelas que utilizam muitas operações de E/S, realizam muitas trocas de contexto entre o espaço físico do sistema hospedeiro e o VMM, dessa forma, têm seu desempenho penalizado (BARHAM & DRAGOVIC, 2003). Portanto, dentre os componentes de *hardware* virtualizados (CPU, memória e E/S) como pode ser visto na Figura 5 (ROSE, 2004), o que gera maior sobrecarga, são operações de E/S, sendo que uma requisição deve ser tratada por múltiplas camadas de protocolos existentes entre o espaço de usuário e o *hardware* do sistema.

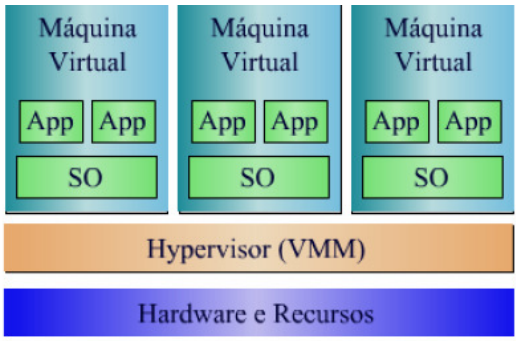

Figura 5- Estrutura de máquinas virtuais

Podem ser utilizadas duas maneiras de se iniciar máquinas virtuais (FIGUEIREDO & DINDA, 2002):

- **VM *Reboot*:** O Sistema Operacional é carregado na máquina virtual a partir da inicialização do sistema;
- **VM *Restore*:** O sistema operacional é restaurado a partir de um pós *boot* (*warm state*);

Para armazenar as imagens virtuais são considerados 2 processos:

- ***Disk-diff*:** O armazenamento dos discos virtuais é criado de maneira não persistente, ou seja, as modificações são armazenadas em arquivos de diferenciação (*diff*). Esse processo é dividido em dois modelos:
  - **Disco-nativo:** armazenado no disco rígido (sistema de arquivos nativo do *host*);
  - ***Disk-loopback*:** reside em uma partição NFS criada no host simulando um sistema de arquivos remoto;
- ***Disk*-réplica:** uma cópia do disco é criada no sistema de arquivos da máquina host antes da inicialização do sistema;

As máquinas virtuais são ambientes que se apresentam como um sistema operacional convidado no *hardware*, porém, são simuladas em *software* dentro do sistema hospedeiro. A simulação dos *drivers* deve ser robusta para que o sistema hospedeiro consiga agir como um sistema real. Existem diversas técnicas de virtualização, como descritas nas seções a seguir.

4.3 Técnicas de Virtualização

A virtualização pode ser definida como uma técnica que combina ou divide recursos computacionais para prover um ou mais ambientes operacionais de execução. Os ambientes criados através dessa técnica são chamados máquinas virtuais. Existem várias técnicas utilizadas para fornecer virtualização, as principais são: emulação, virtualização nativa e paravirtualização. Essas técnicas são detalhadas nas seções seguintes.

4.3.1 Emulação

Nesse tipo de solução o VMM simula todo o *hardware*, permitindo que um sistema operacional sem modificações seja executado em um processador completamente diferente do *hardware* nativo. Um exemplo de emulação é o QEMU (BELLARD, 2004).

O QEMU é um *software open source* que emula alguns tipos de processadores, permitindo a virtualização de um sistema operacional dentro de outro. O QEMU pode ser acoplado ao modulo acelerador chamado KQEMU, utilizando este módulo o QEMU repassa algumas instruções diretamente para o processador real da máquina *host*, melhorando seu desempenho em até 5 vezes (ZHANG, et al. 2006).

4.3.2 Virtualização nativa (“virtualização completa”)

A Virtualização nativa também é conhecida como virtualização híbrida ou acelerada. Este modelo baseia-se na análise de desempenho e emprega técnicas de aceleração de *hardware*. O uso do processador nativo e *drivers* de rede são os recursos mais comuns onde é empregada a virtualização nativa.

Esta solução reduz a maioria das despesas de manutenção da paravirtualização (seção 4.3.3), no que tange a quantidade de alterações necessárias no sistema operacional convidado (SMITH & NAIR, 2005). Porém, há limitações quanto ao número de sistemas operacionais convidados executando eficientemente em um VMM com virtualização nativa (ROSE, 2004), pois a máquina virtual somente simula parcialmente o *hardware*, permitindo que um sistema operacional funcione isoladamente no computador, dessa forma, o sistema operacional convidado exige

muito do *hardware* hospedeiro. Exemplos desse tipo de virtualização são o VMware (VMWARE, 1998) e o Virtual Box (VIRTUAL BOX, 2007).

O VMware e o Virtual Box não são emuladores, trabalham uma camada abaixo da emulação, na qual o processador pode executar diretamente instruções da máquina virtual. Quando isto não é possível, o código é convertido de forma a que o processador não precise trocar de contexto, ganhando tempo na execução de instruções (SUGERMAN & VENKITACHALAN, 2001).

4.3.3 Paravirtualização

A paravirtualização é um tipo especial de virtualização no qual o sistema operacional virtualizado deve ser devidamente modificado para permitir que instruções especiais sejam executadas em detrimento às instruções de máquina padrão. A paravirtualização oferece benefícios significantes de suporte a *drivers* e interfaces. Além disso, o próprio *hypervisor,* o qual fornece uma interface entre os recursos reais e as máquinas virtuais, é protegido de eventuais erros e problemas com *drivers* e pode ser empregado independente do dispositivo físico instalado.

Os sistemas operacionais virtualizados são mais portáteis quando virtualizados pelo *hardware*, eles são virtualizados em níveis baixos e a gerência do *hardware* funciona sob o controle do *hypervisor*. A máquina virtual não simula o *hardware*, porém, oferece uma API especial que requer modificações do *kernel* do sistema operacional hóspede. Uma das implementações que utilizam paravirtualização é o XEN (BARHAM & DRAGOVIC, 2003). O XEN é um monitor de máquinas virtuais para plataformas x86, desenvolvido pela Universidade de Cambridge, o qual permite que múltiplos sistemas operacionais compartilhem recursos de *hardware* garantindo funcionalidade e desempenho. Para contornar o problema da não virtualização da arquitetura IA-32 (ROBIN & IRVINE, 2000), o XEN utiliza a abordagem de paravirtualização em contrapartida à virtualização nativa.

O *kernel* do sistema operacional da máquina hóspede é modificado de modo que o VMM execute tarefas protegidas que não sejam direcionadas diretamente ao processador. No XEN, o *hypervisor* é executado em nível 0 (*ring 0*), de modo que apenas ele possa utilizar o *hardware* diretamente. Os *kernels* dos sistemas operacionais hóspedes são modificados para executarem no nível 1 (*ring 1*), isto é, as máquinas

virtuais não são capazes de acionar diretamente o *hardware*. As aplicações nos sistemas operacionais hóspedes continuam a executar no nível 3 (*ring 3*). Nessa abordagem, o escalonamento das máquinas virtuais no *hardware* é realizado pelo *hypervisor*. Devido à modificação das interfaces dos *kernels* nos hóspedes, as máquinas virtuais convidadas também necessitam de modificações em seu *kernel*.

4.4 Máquinas Virtuais em Sistemas Distribuídos

O uso de máquinas virtuais ganhou um novo sentido quando utilizada no contexto de sistemas distribuídos, possibilitando que sistemas operacionais completamente diferentes possam ser inicializados sem a necessidade de instalá-lo no *hardware* real.

A flexibilidade de sistemas operacionais e um ambiente de execução que abstrai o *hardware* são características fundamentais para o uso de ambientes virtuais em sistemas distribuídos (CLARK, et al. 2005), oferecendo isolamento e mecanismos de segurança complementares para o sistema operacional.

Dessa forma, máquinas virtuais podem servir como ferramenta para facilitar a implementação de sistemas distribuídos em sistemas operacionais distintos, sem a preocupação de configurações adicionais distintas para cada ambiente de execução, permitindo uma implantação mais simples de sistemas em *grid*.

4.4.1 Servidores de Máquinas Virtuais

Os servidores de máquinas virtuais permitem que a virtualização seja centralizada em um único computador, cujas máquinas virtuais são iniciadas sobre a mesma plataforma e podem ser utilizadas com intuitos diversos. Dessa forma é possível ter vários servidores com diferentes sistemas operacionais, por exemplo, um servidor de *e-mail* em linux e um servidor de páginas em windows no mesmo computador sendo executado sobre um monitor de máquinas virtuais.

Dentre os milhões de equipamentos atualmente em produção, a média de utilização de seus recursos é de em torno de 5% a 10%, ou seja, a ociosidade de processamento está em torno de 90% a 95%. A virtualização de servidores permite que os administradores aproveitem esta característica instanciando diversas máquinas

virtuais em um mesmo equipamento, devidamente isoladas e encapsuladas, como visto na Figura 6 (SMITH & NAIR, 2005). Para clientes de tecnologia da informação isto reflete em economia, em média, de 10 servidores para 1, chegando, em alguns casos de até 30 servidores para 1 (ROSE, 2004).

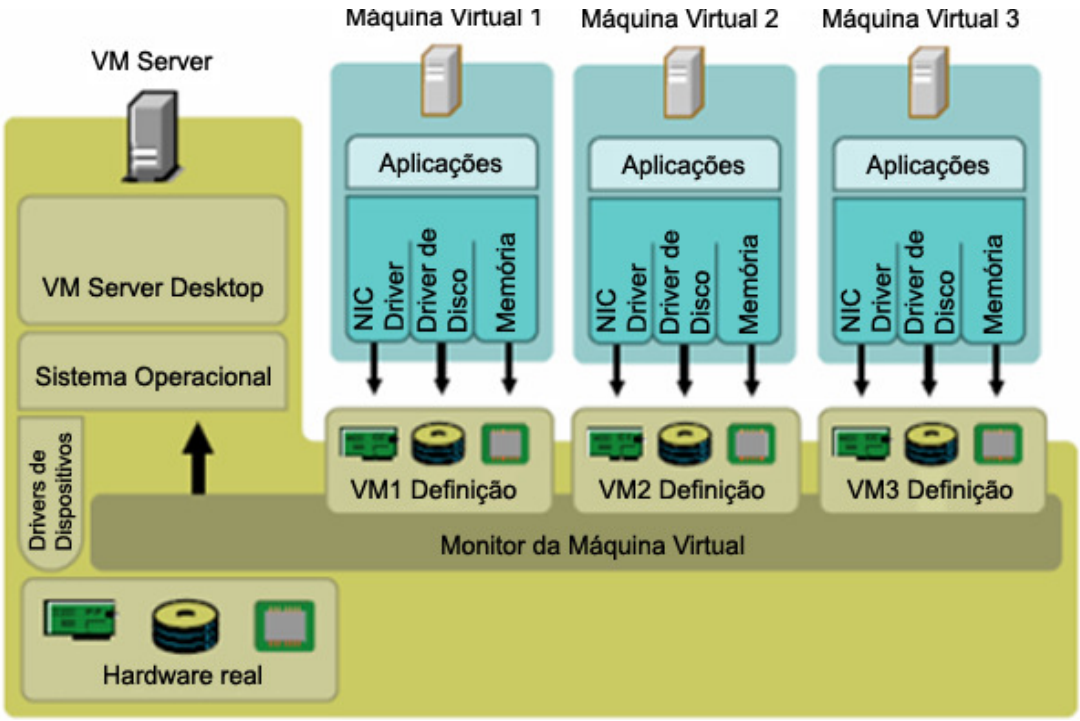

Figura 6 - Estrutura de um Servidor de Máquina Virtual

Outra grande vantagem dos servidores de virtualização é o tempo economizado para trocar o *hardware* real do computador caso ocorra falha do equipamento, sendo necessário somente migrar a máquina virtual para outro equipamento. Caso ocorrer de falhar o sistema operacional da máquina virtual, somente é necessário realizar a cópia de uma imagem virtual base previamente criada e alterá-la conforme as necessidades.

4.4.2 Migração de Máquinas Virtuais

A virtualização permite o encapsulamento da camada de *software* em uma máquina virtual, assim como o desacoplamento do sistema operacional convidado e suas aplicações do *hardware* em que está sendo executado, permitindo que uma máquina virtual seja migrada para outro computador. Além de facilitar as operações de manutenção do *hardware*, a migração de uma máquina virtual pode ser realizada com técnicas de balanceamento de carga (GUILHERMO, et al. 2001) e predição de falhas (DODONOV, MELLO & YANG, 2006).

O aumento da capacidade de processamento dos atuais sistemas computacionais, bem como as melhorias na robustez e da velocidade das redes de computadores, possibilitou a execução e a atualização de aplicações remotamente (PHILIP, et al. 2000), viabilizando assim o uso de máquinas virtuais em ambientes de *grid.* A capacidade de migração permite a melhoria na qualidade de serviço, a um custo operacional baixo, quando comparado a instalações individuais de sistemas operacionais.

4.4.3 Máquinas Virtuais e *Grids*

Máquinas virtuais e *grids* podem trabalhar de forma conjunta oferecendo isolamento, tolerância a falhas e mecanismos de segurança complementares para o sistema operacional. Há diversas formas de se utilizar ambientes *grid* com máquinas virtuais, uma delas é o VMPlant (KRSUL, GANGULY & ZHANG, 2004), que fornece configuração automatizada e criação de máquinas virtuais flexíveis.

Máquinas virtuais são implementadas juntamente com um *grid* em dois cenários, os quais necessitam de características distintas para gerenciamento dos blocos de dados através do sistema de arquivos virtual. No primeiro cenário, o usuário do *grid* é alocado para uma máquina virtual com um disco virtual persistente no servidor de estados. Isso acarreta a suspensão do estado da MV quando os usuários abandonam ou voltam de um determinado estado da MV. No momento que a sessão é iniciada, a MV deve ser instanciada no sistema do servidor, e depois que a sessão for finalizada as modificações realizadas pelo usuário devem ser refletidas no servidor de estados (WHITAKER, SHAW & GRIBBLE, 2002).

No segundo cenário, pode-se usar sistemas de arquivos que permitam instanciação de clones de máquinas virtuais usando tratadores de metadados para o arquivo de estados da memória. Ao invés de copiar o disco virtual inteiro, somente atalhos simbólicos são criados no servidor refletindo o estado do disco virtual (ZHAO, ZHANG & FIGUEIREDO, 2006). Máquinas virtuais, que uma vez configuradas para atender as exigências de aplicações, podem, então, serem copiadas (clonadas) e dinamicamente instanciadas para fornecer execução heterogênea através de ambientes de recursos computacionais remotos. Em combinação com um *middleware* para gerenciamento de usuários, recursos e dados, a funcionalidade permite que um sistema

grid seja facilmente implantado em máquinas com *hardware* e sistema operacional heterogêneos (FIGUEIREDO, KAPADIA & FORTES, 2001).

Sistemas em *grid* necessitam de alguns requisitos básicos que também podem ser fornecidos pelas máquinas virtuais:

Segurança – Os sistemas de computação distribuída devem proteger a integridade dos recursos do computador a que são agregados. As tarefas devem prevenir o acesso e modificações no conjunto de dados do *grid*.

Ambiente de execução limpo – O estado da máquina cliente (sistema de arquivos e registros) deve continuar o mesmo após a execução de tarefas do *grid*.

Moderação - As sub-tarefas executadas nas máquinas *desktop* precisam ficar fora do caminho das aplicações do usuário. Os clientes do *grid* compartilham recursos (computacionais, armazenamento, e recursos de rede) com outras aplicações e usuários da máquina. Entretanto, o uso desses recursos pelas aplicações do *grid* deve ser moderado e não agressivo quando houver competição por processamento entre as aplicações.

Integridade – Alguns ambientes necessitam que o sistema garanta a integridade da computação distribuída, preocupando-se com o isolamento de dados das aplicações e programas.

4.5 Isolamento e Segurança

Com a evolução da informática, o surgimento dos PC's e das redes de computadores, tornou-se possível o compartilhamento de recursos e informações. Este compartilhamento aumentou a facilidade do usuário obter essas informações, e conseqüentemente cresceu a probabilidade de intrusos compartilharem deste acesso. Isto fez crescer a importância da segurança nos sistemas de computação e em aplicações que utilizam *grids*. Diferentemente de antigos sistemas mainframes, os atuais sistemas distribuídos possuem um grande número de pontos de acesso, sendo difícil restringir o acesso a todos eles (ZHAO, ZHANG & FIGUEIREDO, 2004).

Máquinas Virtuais clássicas possibilitam um alto nível de segurança em comparação a um sistema operacional multi-programado com mecanismos de

redundância e independência implementados entre o monitor de máquinas virtuais (VMM) e o sistema operacional (MADNICK & DONOVAN, 1973).

O projeto Revirt (DUNLAP, et al. 2002) usa a virtualização para melhorar a segurança do sistema no acesso aos serviços remotos na rede de comunicação. Revirt permite que uma seqüência de instruções ocorrida anteriormente seja executada novamente, conhecida como "*replay*". O administrador de rede é responsável por procurar evento de invasão ao sistema em uma análise pós-ataque. O *replay* é implementado através de dois contextos de execução que são idênticos em todos os aspectos. O *replay* recria o sistema provendo um nível de detalhes similar a de um depurador, mas com menos sobrecarga e impacto no comportamento do sistema.

Virtualmente, todos os computadores possuem canais de comunicação para acesso autorizado às suas facilidades, e é através destes que o acesso não autorizado ocorre. Nos sistemas distribuídos, os computadores estão conectados a uma rede e os seus sistemas operacionais oferecem uma interface de comunicação padrão que permite o estabelecimento de canais de comunicação virtuais. Dessa forma, as máquinas virtuais são um atrativo para sistemas em *grids* que isolam os recursos da máquina virtual do acesso externo e encapsulam os sistemas de segurança junto com a replicação das imagens virtuais (JITHENDAR, ZHAO & FIGUEIREDO, 2004).

4.6 Considerações Finais

Máquinas virtuais são utilizadas em diversas áreas de pesquisa para solucionar problemas e aprimorar os sistemas de computação. Características como segurança, e disponibilidade de sistemas são fundamentais para melhorar o gerenciamento e usabilidade de computadores. As técnicas de virtualização auxiliam com que aplicações de sistemas distribuídos sejam aprimoradas, garantindo a manutenibilidade e gerenciamento de *software* fornecido pela camada de abstração das máquinas virtuais.

Capítulo
5

FlexLab

5.1 Considerações Iniciais

Três tendências merecem destaque na computação atual, (1) computadores *commodity* que melhoram a taxa preço/benefício dos computadores, (2) a utilização dos sistemas distribuídos e paralelos para aplicações de alto desempenho e (3) a virtualização, que aplica uma camada de abstração do *hardware* e facilita o gerenciamento de sistemas operacionais. Este trabalho propõe um modelo de virtualização distribuída denominado FlexLab, através da aplicação de *stream* de *software*, balanceamento de carga e virtualização, fundamentados nos capítulos anteriores. O FlexLab fornece ferramentas para implantação e manutenção de sistemas operacionais e *softwares* centralizados em uma unidade de gerenciamento, facilitando a administração de computadores na rede, além de auxiliar na implantação e manutenção de aplicações em *clusters* multiuso e LAN *grids*.

Tradicionalmente, para conseguir que a mesma máquina inicializasse mais de um sistema operacional era preciso criar múltiplos discos com seus respectivos sistemas operacionais. Com o advento das máquinas virtuais foi possível criar sistemas operacionais aninhados, porém, ainda era necessário copiar o disco virtual para a máquina local, o que dificultava o gerenciamento centralizado dessas imagens. O FlexLab permite que uma máquina virtual seja selecionada no momento da inicialização, podendo carregá-la em modo somente leitura ou leitura e escrita, facilitando assim a manutenção dos sistemas operacionais e *softwares* em um único ponto de controle.

Há duas vertentes de pesquisa no FlexLab, a primeira centrada em encontrar melhores práticas e técnicas de virtualização, tema de outro projeto de pesquisa em

andamento. A segunda vertente consiste em melhorar o processo de distribuição dos sistemas virtualizados e o gerenciamento de aplicações distribuídas, no qual este trabalho está situado. Esta vertente busca formas de escalar o projeto até dezenas de computadores sem onerar o servidor de imagens virtuais, permitindo que diversos sistemas operacionais sejam inicializados simultaneamente utilizando o mesmo conjunto de servidores, como mostrado na Figura 7, em que diferentes *hardwares* inicializam diferentes sistemas operacionais remotamente.

Figura 7 - FlexLab: *Boot* remoto de imagens virtuais em máquinas heterogêneas com balanceamento de carga entre servidores

Entre as possíveis utilizações para o FlexLab estão:

- Gerenciamento de computadores em empresas, escolas e faculdades, permitindo que o usuário possa escolher o sistema operacional que deseja utilizar no momento do *boot*. Por exemplo, é possível preparar uma imagem virtual (Linux ou Windows) com todos os aplicativos utilizados durante uma disciplina de Biologia, otimizando dessa forma o uso dos computadores durante o período de aula com ferramentas específicas para biologia, sem sobrecarregar ou danificar sistemas operacionais de outras disciplinas;

- Implantação de *clusters* multiuso, ou seja, *clusters* implementados através de computadores *commodity*, permitindo que máquinas subutilizadas sejam aproveitadas em computação de alto desempenho,

por exemplo, inicializando um *cluster* em computadores de laboratórios de faculdades durante os períodos de inatividade dos computadores;

- Gerenciamento de LAN *grids*, ou seja, máquinas virtuais preparadas para serem executadas em plano de fundo enquanto os computadores são utilizados normalmente pelos usuários;

5.2 Requisitos do FlexLab

Para que o FlexLab suporte diversas características e situações encontradas e se porte de maneira uniforme o projeto deve atender aos requisitos:

- Permitir o *boot* remoto de diversos computadores simultaneamente de maneira escalável com balanceamento de carga, sem onerar os servidores;
- Permitir o reaproveitamento de computadores commodity através de *clusters* multiuso e gerenciar operações para ligar e desligar os computadores automaticamente durante períodos de ociosidade;
- Gerenciar máquinas virtuais aplicadas a ambientes em *Grid*;
- Gerenciar o uso de *cache* tanto na máquina cliente como no servidor de imagens virtuais;
- Permitir o gerenciamento e manutenção das máquinas virtuais no próprio computador do servidor, sem a necessidade de ligar um computador cliente para executar uma atualização;
- Ser flexível para suportar a maior gama de *hardwares* encontrados no mercado. Os computadores incluídos ao FlexLab necessitam da seguinte configuração mínima:

Clientes

  - Placa de rede com suporte a *boot* por rede (encontrado na maior parte dos computadores);

- Computador equivalente a um Pentium III com no mínimo 64 MB de RAM (mais variação da memória necessária para a execução da máquina virtual);
- Rede de 100Mb/s.

Servidores

- Equivalente a um computador Pentium IV;
- Mínimo de 512 MB de RAM;
- Rede de 100Mb/s.

5.3 Estrutura e Implementação

A capacidade de processamento dos atuais sistemas computacionais, bem como o avanço da robustez e da velocidade das redes de computadores, viabilizou a execução de aplicações remotamente nas máquinas, juntamente com a camada extra de virtualização, possibilitando que ambientes como o FlexLab se tornem viáveis (GALLI & DOREEN, 2000).

O FlexLab é estruturado em duas camadas (AGUIAR, et al. 2008)(a), a primeira inicializa o *middleware* (mini Linux) contendo suporte para a virtualização e a segunda camada é responsável por realizar o *boot* do sistema operacional virtualizado (Linux, Windows, etc) com técnicas de balanceamento de carga (PHILIP, et al. 2000). O projeto foi concebido através de tecnologias de sistemas de arquivos distribuídos, *boot* remoto, balanceamento de carga e virtualização.

O *boot* do *middleware* FlexLab (mini Linux customizado) foi configurado para utilizar diferentes tecnologias em transmissão de arquivos dependendo da camada de uso, como visto na Figura 8.

Durante o *boot* do *middleware* são aplicados diferentes tratamentos para as diversas áreas do sistema operacional, como os computadores podem não ter disco rígido, a cópia é realizada diretamente na memória RAM dos computadores clientes. Aumentando dessa forma a localidade dos dados, ou seja, melhorando a velocidade de acesso aos dados mais utilizados no sistema operacional, deixando as informações mais próximas à unidade processadora.

O acesso aos dados é realizado de três formas no FlexLab, (1) para as pastas de sistema como /usr, /opt, /bin, /sbin, /etc , o acesso remoto é realizado através de NFSv3 com o protocolo UDP que garante resiliência e rapidez de tráfego, (2) para as pastas do Linux que precisam ser individualizadas por computador como /etc, /dev, /root são copiadas diretamente para a memória RAM através de rsync, e por último (3), o FlexLab requisita as máquinas virtuais através do sistema de arquivos distribuído híbrido (pNFS/PVFS) que executa o balanceamento de carga entre os servidores.

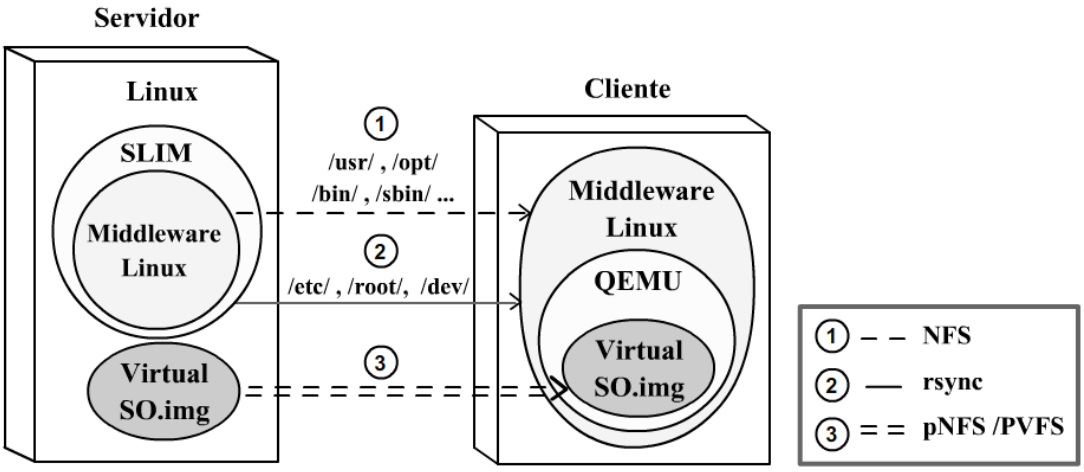

Figura 8 - Modelo de *Boot* Distribuído do FlexLab: diferentes tipos de sistemas de arquivos distribuídos são utilizados dependendo dos dados requisitados

O m*iddleware* foi estruturado com base em um mini kernel Linux (Fedora Core 3) compilado para garantir desempenho em ambientes distribuídos e configurado para aceitar a maioria dos *hardwares* disponíveis com reconhecimento de *drivers*.

A segurança é melhorada pelas características implícitas do NFSv3 e do PVFS2 com pNFS, porém é necessário que a rede de comunicação já tenha uma segurança primária implantada.

Outra característica do FlexLab é permitir que as imagens virtuais possam ser atualizadas remotamente através de *merge* de arquivos, por exemplo, ao atualizar as escolas ou empresas que utilizam a imagem X com um novo *software* instalado, somente as partes que sofreram alteração na imagem serão transmitidas, não necessitando a transmissão da imagem virtual inteira que normalmente contém diversos Giga Bytes. Esse modelo também pode ser útil para aplicações em *grids* que necessitam de alta taxa de atualização de seus ambientes. Há diversas tecnologias que podem

realizar o *merge* dos arquivos, entre elas estão o rsync (TRIDGELL & MACKERRAS, 1998) e o Unison (UNISON, 2001).

5.3.1 Sistema de Arquivos Distribuído

O sistema de arquivos distribuído escolhido para o *middleware* do FlexLab foi o NFSv3 por sua estabilidade, alta resiliência e compatibilidade. Diferentemente do AFS ou CODA, o NFSv3 é compatível com a maioria dos sistemas operacionais, permitindo que a segunda camada virtualizada do FlexLab também possa utilizar os recursos do NFSv3.

O projeto necessitou de configurações adicionais no sistema de arquivos distribuídos, tanto no tamanho dos blocos transmitidos como na forma em que eles eram trafegados. Em sistemas de arquivos distribuídos muitas vezes há necessidade de se copiar uma pasta inteira porque não é possível saber se ocorreu alteração em algum conteúdo dentro da mesma, fazendo com que o sistema sobrecarregue o tráfego da rede.

Em determinadas situações, mesmo quando se identifica alteração em um certo arquivo, o mesmo precisa ser replicado para todos os clientes e boa parte dos sistemas realiza a cópia inteira do arquivo. No FlexLab foi estudado e aplicado o rsync (TRIDGELL & MACKERRAS, 1998) junto com NFSv3 (HITZ, LAU & MALCOLM, 1994), o rsync copia somente as partes do arquivo que realmente sofreram alterações e compacta os dados, melhorando assim o tráfego na rede, característica fundamental em sistemas distribuídos com *boot* remoto.

5.3.2 Balanceamento de Carga

Para melhorar a velocidade de acesso remoto foi utilizado distribuição de carga para operações E/S (Entrada e Saída) entre os servidores através do sistema híbrido de PVFS2 (CARNS, et al. 2000) e pNFS (HILDEBRAND & HONEYMAN, 2007), dessa forma, o servidor de imagens virtuais não é sobrecarregado, podendo ser escalável conforme o número de novos clientes são incluídos.

Um dos maiores problemas de sistemas de arquivos paralelos é garantir robustez, escalabilidade e abstração de acesso aos arquivos. Diversos sistemas de arquivos paralelos modernos como o NFSP, pNFS, PVFS foram pesquisados, e os

requisitos foram atingidos utilizando um sistema de arquivos híbrido de PVFS2 com pNFS, como encontrado no projeto Direct-pNFS (HILDEBRAND & HONEYMAN, 2007).

O sistema híbrido de pNFS com PVFS2 cria dois caminhos para a transmissão dos dados, um deles, transmitindo informações de metadados (informações que definem em qual servidor de E/S os dados requisitados devem ser lidos) através de NFSv4, e a camada convencional de PVFS2 para transmitir os blocos contínuos das imagens virtuais.

Na medida em que o número de clientes conectados a um servidor aumenta é possível acrescentar servidores extra para operações de E/S como observado na Figura 9, até mesmo um nó cliente com disco rígido pode se transformar em um servidor de E/S para balancear a carga do *middleware*.

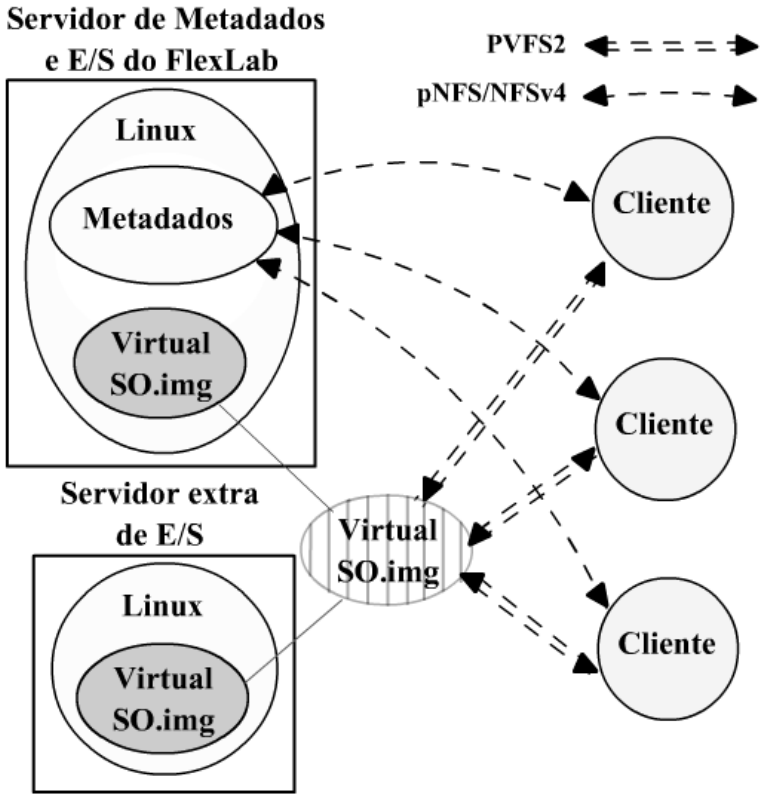

Figura 9 - Inclusão de servidor extra com balanceamento de carga à medida com que são acrescentados novos clientes ao *boot* remoto.

Em um *boot* remoto os blocos carregados inicialmente pelo sistema operacional são geralmente os mesmos, assim, é possível guardar na memória dados mais relevantes do *kernel* e otimizar o processo de *boot* aumentando a localidade dos dados quando

vários sistemas operacionais são inicializados simultaneamente (KUACHAROEN, MOONEY & MADISETTI, 2002). Cada computador cliente pode agir como um servidor para distribuir blocos que já foram recebidos para outros computadores como mostrado na Figura 10.

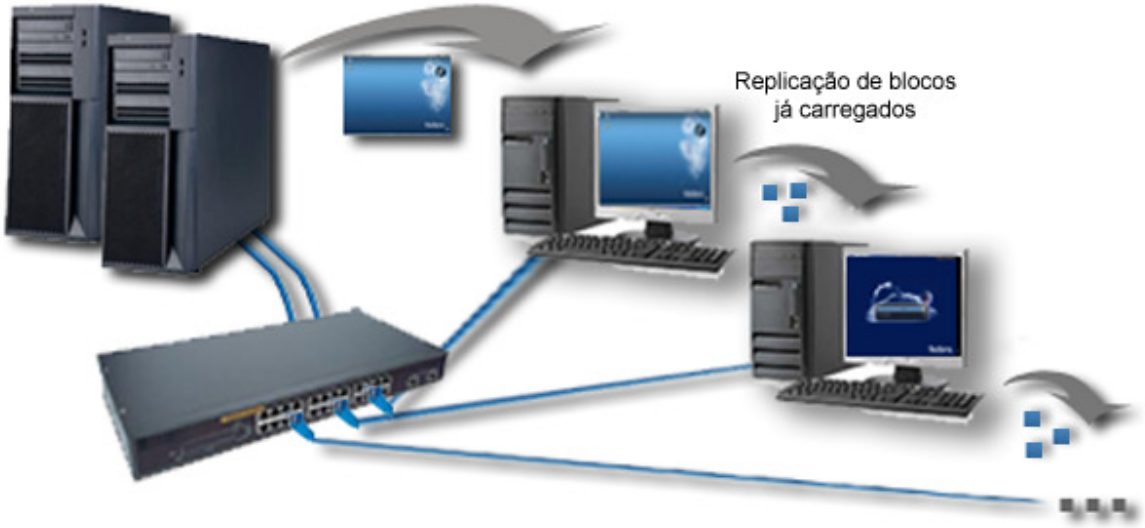

Figura 10 - Computadores clientes auxiliam na distribuição de blocos já carregados para outros computadores

5.3.3 *Boot* Remoto

A escolha do sistema de *boot* remoto é fundamental para um bom funcionamento do sistema. No FlexLab, existem dois momentos de *boot*, o primeiro é o *boot* do *middleware* e o segundo o *boot* do sistema virtualizado, além da camada de *software* intermediária introduzida pela solução para otimização do sistema.

Para o *boot* do *middleware* foi utilizada a tecnologia SLIM com NFSv3, já para o *boot* da camada de virtualização foi utilizado o PVFS2 com pNFS. Dessa forma, são priorizadas leituras de arquivos pequenos para o *boot* do *middleware* e leituras de arquivos grandes para o disco contínuo das máquinas virtuais.

Boot* do *middleware

O *boot* do *middleware* utiliza o SLIM como centralizador e gerenciador das configurações de *boot*. O SLIM é implementado através da combinação de DHCP (*Dynamic Host Configuration Protocol*) (DROMS, 1997), PXE (*Preboot Execution Environment*) (INTEL CORPORATION, 1999), TFTP (*Trivial File Transfer Protocol*) (SOLLINS, K. 1992), rsync (TRIDGELL & MACKERRAS, 1998) e NFSv3 (HITZ, LAU & MALCOLM, 1994).

As máquinas sem disco rígido são inicializadas através de PXE e reconhecidas na rede através do DHCP, a partir desse momento o TFTP copia os dados iniciais do *kernel* Linux para o *boot* do *middleware*, carregando o restante dos dados por NFSv3 e rsync. Após o carregamento inicial os dados que não estão na memória RAM são requisitados sob demanda.

***Boot* do sistema virtualizado**

O sistema de virtualização utilizado no *middleware* foi o QEMU com suporte de virtualização acelerada (KQEMU). Na camada de virtualização foi utilizado a técnica de *disk-diff* com *disk-loopback* para realizar o armazenamento remoto, ou seja, as informações são enviadas para os computadores clientes somente quando há necessidade, diminuindo o tráfego da rede durante períodos de inatividade.

5.3.4 *Clusters* Multiuso e *Grids*

O FlexLab disponibiliza um agendador de tarefas no *middleware* que permite ligar as máquinas remotamente e controlar ambientes em *grids* e *clusters* nos horários ociosos, por exemplo, durante a madrugada. O ambiente explora a combinação de computadores *commodity*, máquinas virtuais e *clusters* multiuso, permitindo que aplicações *grids* sejam executadas paralelamente com o sistema operacional virtualizado utilizado pelo usuário, como mostrado na Figura 11.

O FlexLab se diferencia de outros modelos de agendamento como no projeto Parallax (WARFIELD, ROSS & FRASES) que não permite um gerenciamento independente das estações, já o FlexLab, permite o controle de quantos e quais *grids* serão executados em cada cliente.

Tanto o *cluster* multiuso como aplicações de LAN *grid* podem ser executadas sem a camada de virtualização, diminuindo a abstração de sistema operacional, porém, aumentando a velocidade de processamento, características fundamentais para execução de processos paralelizados.

Figura 11 - *Cluster* multiuso distribuindo aplicações *grid* paralelamente enquanto sistemas operacionais convencionais são utilizados

5.4 Trabalhos Relacionados

Na implementação de técnicas de *boot* remoto algumas soluções preferem a abordagem dos *thin-clients* priorizando o processamento no servidor como o THINC (BARATTO, KIM & NIEH, 2005), diferentemente do FlexLab que prioriza a utilização de recursos locais das máquinas clientes.

Outros sistemas de *boot* remoto como o Ardence BXP (BXP, 2003) permitem que computadores sem disco rígido iniciem um sistema operacional remotamente, porém, esses modelos são vinculados ao *hardware* e não é possível realizar atualizações nas imagens dos sistemas operacionais sem antes inicializar um cliente no *hardware* específico, o que dificulta o gerenciamento e o suporte remoto da solução.

Alguns trabalhos relacionam a distribuição de máquinas virtuais em conjunto com ambientes em *grid.* Os trabalhos encontrados mais significativos com os propósitos do FlexLab foram o VIGO (XU, ADABALA & FORTES, 2005) e o VMPlants (KRSUL, GANGULY & ZHANG, 2004). Também foram encontrados trabalhos que buscam maneiras de otimizar aplicações em *grids* como o SIMGRID (HENRI, et al.

2006). O FlexLab utilizou como base algumas das características dos projetos pesquisados para viabilizar a utilização de *grids* com máquinas virtuais e o reaproveitando de recursos ociosos dos computadores.

Os projetos Parallax (WARFIELD, ROSS & FRASES, 2005) e Usher (MCNETT, et al. 2007) também aplicam a idéia de virtualização distribuída encontrada no FlexLab, porém, não se preocupam com o balanceamento de carga no *boot* simultâneo de máquinas virtuais e não aplicam a idéia de *cluster* multiuso para reaproveitamento dos recursos disponíveis nos computadores.

5.5 Considerações Finais

Recursos tecnológicos disponíveis em organizações estão subutilizados em relação ao seu potencial. Estatísticas mostraram que os computadores convencionais usam em torno de 5% a 10 % do seu potencial, e servidores utilizam 20% da capacidade (KURMANN, RAUCH & STRICKER, 2003).

A velocidade de acesso aos arquivos a partir de meios físicos de armazenamento não evoluiu na mesma proporção que a velocidade dos processadores ou da transmissão de dados, isso torna o acesso aos arquivos um gargalo para muitas aplicações, especialmente para aquelas que buscam informações armazenadas remotamente. Para reduzir esse problema foi utilizado um sistema de arquivos paralelos no acesso aos dados para um número crescente de clientes dentro do FlexLab, provendo a escalabilidade de *boot* para dezenas de clientes simultaneamente.

Os sistemas computacionais presentes nas áreas científicas e comerciais podem apresentar heterogeneidade de *hardware* e *software*, resultando em obstáculos para a implantação de aplicações multi-tarefas. Além disso, c*lusters* multiuso e *grids* computacionais têm emergido no cenário de computação *commodity* por representar uma opção de processamento de tarefas a um baixo custo de implantação.

O FlexLab fornece ferramentas que facilitam o gerenciamento de aplicações *clusters* multiuso e *grids* em computadores *commodity*, podendo ser utilizado por aplicações distribuídas, permitindo o aproveitamento de recursos ociosos dos computadores em empresas ou instituição de ensino, além de possibilitar o

gerenciamento e manutenção centralizados dos sistemas operacionais e aplicativos através de imagens virtuais.

Capítulo
6

Experimentos

6.1 Considerações Iniciais

Os experimentos realizados neste trabalho consistiram em extrair uma base de informações referente a cópias e leituras de blocos de arquivos, testando a viabilidade dos sistemas de arquivos distribuídos e paralelos em suas principais funções, para que dessa forma fosse definida a estrutura do FlexLab. Em um segundo momento foram capturados os tempos de inicialização de sistemas operacionais e abertura de aplicativos no FlexLab, tanto do SO hospedeiro (*middleware*) quanto do SO virtualizado.

6.2 Experimentos para definir a configuração do FlexLab

Diversos fatores influenciam no tempo de *boot* remoto das máquinas clientes como largura de banda, protocolos habilitados, nível de segurança, tamanho dos blocos transmitidos, dimensão do arquivo e número de *threads* em execução no servidor. O gerenciamento de *cache* também tem grande influência sobre a velocidade da cópia, permitindo maior localidade para os próximos blocos de dados. Dessa forma os testes foram realizados para descobrir quais tecnologias e configurações do *middleware* e dos sistemas de arquivos distribuídos e paralelos são mais adequados aos propósitos do FlexLab.

6.2.1 Comparações de técnicas de *boot* remoto

Para viabilização de um sistema de *boot* remoto foram analisadas três tecnologias para computadores com ou sem disco rígido baseados em Linux, o DRBL (Diskless Remote Boot in Linux), o SLIM (Single Linux Image Management) (SLIM, 2004) e o LTSP (Linux Terminal Server Project). Contudo, o LTSP foi inicialmente

descartado pelo fato do processamento dos clientes ser realizado no servidor, o que inviabilizou seu uso para o FlexLab, pois o processamento local nos clientes remotos é fundamental para permitir escalabilidade de computadores e execução de aplicações que necessitam de alto processamento como em aplicações multimídia, *clusters* ou *grids*.

Diversos fatores foram considerados para escolha da tecnologia que melhor se adequasse às necessidades do projeto como a velocidade das operações de escrita e leitura, juntamente com a adequação aos requisitos do FlexLab.

Para os experimentos realizados nas comparações do sistema de *boot* foi utilizada a configuração de *hardware* da Tabela 2 e foi instalado o sistema operacional Fedora Core 6 no servidor.

Tabela 2 - Configuração de *hardware* para os testes em soluções de *boot* remoto

Tipo	Configuração
Servidor	Pentium 4, 512 MB de RAM, Disco rígido de 60 GB IDE, rede de 100Mb/s
Clientes	Pentium 4, 512 MB de RAM, sem disco rígido, rede de 100Mb/s
Switch	100 Mb/s

Para as comparações entre o DRBL e o SLIM, ambos os sistemas foram instalados no servidor com Fedora Core 6 e distribuíram uma imagem do próprio Fedora Core 6 para dois clientes. Na captura do tempo de execução de aplicativos e *boot* do sistema operacional foram utilizados *scripts* na inicialização do computador cliente (vide Anexo A). Os tempos obtidos nos experimentos estão mostrados na Tabela 3.

Tabela 3 - Comparação entre DRBL e SLIM

	Cliente no DRBL	Cliente no SLIM
Boot do Fedora Core 6	0’42’’	0’46’’
Inicialização do Open Office	0’06’’	0’04’’

É possível notar que o DRBL obtém melhor tempo que o SLIM com uma diferença de 4 segundos na inicialização do Fedora Core 6, porém, para execução de aplicações o SLIM tem maior velocidade (YANG, PING-I & YA-LING, 2005), por exemplo, na abertura do Open Office há diferença de 2 segundos em relação ao DRBL. Essa diferença ocorre devido ao tipo de montagem de cada sistema, o DRBL utiliza NFS desde o início do *boot* diminuindo o tempo de inicialização, já o SLIM copia partes do sistema operacional para a RAM do cliente, aumentando o tempo de *boot*, porém, melhorando a execução do mesmo após a inicialização. Devido a essas características, o DRBL congestiona mais a rede de comunicação quando comparado ao

SLIM, como visto em Yang (YANG, PING-I & YA-LING, 2005), dessa forma, o SLIM foi escolhido como o *boot* remoto da solução FlexLab.

6.2.2 Comparação entre NFSv3, PVFS2 e pNFS/PVFS2

Para definir o modelo distribuído mais adequado para as necessidades do projeto foi utilizado o IOzone Filesystem Benchmark (NORCOTT & CAPPS, 2003) como ferramenta de *benchmark*. O IOzone permitiu que fossem analisados desempenhos do NFSv3, PVFS2 e pNFS/PVFS, colhendo dados de leitura e escrita em arquivos nos servidores de 1 até 30 clientes. Os testes foram realizados seguindo a configuração de *hardware* da Tabela 4.

Tabela 4 - Configuração de *hardware* para os experimentos de balanceamento de carga no FlexLab

Tipo	Configuração
Servidor 1	Intel Core2 Duo 2.0 Ghz, 1 GB de RAM, Disco rígido de 120 GB SATA, rede de 1Gb/s
Servidor 2	Intel Core2 Duo 2.0 Ghz, 1 GB de RAM, Disco rígido de 120 GB SATA, rede de 1Gb/s
Clientes	Pentium 4, 512 MB de RAM, Diskless, rede de 100Mb/s
Switch	100Mb/s

Durante os experimentos de balanceamento de carga foi notado que um cliente utilizando PVFS2 juntamente com dois servidores balanceando carga realizavam operações de leitura e escrita mais lentas do que as mesmas operações no NFSv3 com apenas um servidor e um cliente, constatando que as operações de controle do balanceamento de carga oneram o processo quando utilizado com poucos clientes, como observado em Ross (ROSS, LIGON & CARNS, 2002). Para contornar esse problema foi utilizado o sistema híbrido de PVFS2 com pNFS, o qual aumentou a velocidade de leitura em relação ao NFSv3, como visto na Figura 12. Isso pode ser explicado pelo fato do pNFS alterar o caminho das operações de controle (metadados), que passam a ser realizadas pelo pNFS, deixando somente a troca de dados do sistema operacional por PVFS2.

Experimentos realizados mostraram que o PVFS2 junto com pNFS otimizou em até 16% as operações de leitura em relação ao PVFS2 sem a utilização do pNFS.

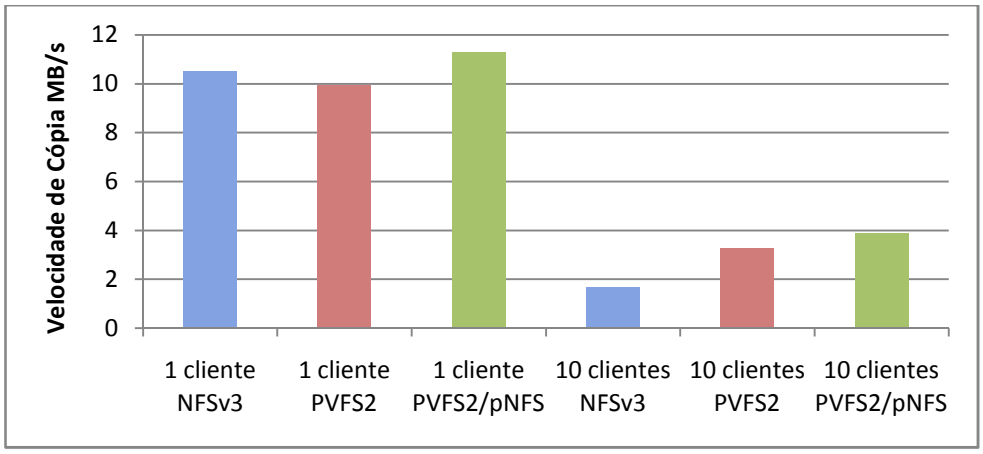

Figura 12 - Comparação de velocidade na cópia de 12 MB com blocos de 8 KB

Os experimentos foram realizados considerando a configuração da Tabela 4, o NFSv3 utiliza o servidor 1, já o PVFS2 e o pNFS/PVFS2 utilizam balanceamento de carga através dos servidores 1 e 2.

Outra vantagem do pNFS/PVFS2 é a escalabilidade conforme são acrescentados clientes. Considerando 10 clientes realizando uma cópia por NFSv3 de 12MB a blocos de 8KB, a taxa de leitura cai 4,5 vezes em relação ao mesmo experimento com apenas um cliente. Já o pNFS/PVFS2 com dois servidores de I/O a taxa de leitura cai apenas 2,3 vezes em relação ao experimento com um cliente, indicando que o balanceamento de carga diminui o gargalo da rede de comunicação em 2,2 vezes em relação ao sistema sem balanceamento de carga, como mostrado na Figura 12.

O NFSv3 foi criado para suportar poucos processos executando acesso ao mesmo arquivo, e por este motivo, pode garantir um acesso mais rápido quando utilizado com poucos clientes. Já o PVFS2 foi projetado para o acesso simultâneo de um arquivo por diversos clientes, aumentando assim o desempenho para aplicações que necessitam dessa característica. Dessa forma, operações de leitura do pNFS/PVFS2 oferecem um desempenho em média 10% melhor que o NFSv3, como visto na Tabela 5 e representado na Figura 13.

Tabela 5 - Comparação de velocidade de leitura em KB/s entre pNFS/PVFS2 e NFSv3

	Tamanho do bloco de leitura (KB)								
	4	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
pNFS/PVFS2	11299	11314	11315	11307	11314	11309	11278	11286	11240
NFSv3	10562	10550	10563	10669	10562	10537	10538	10564	10563

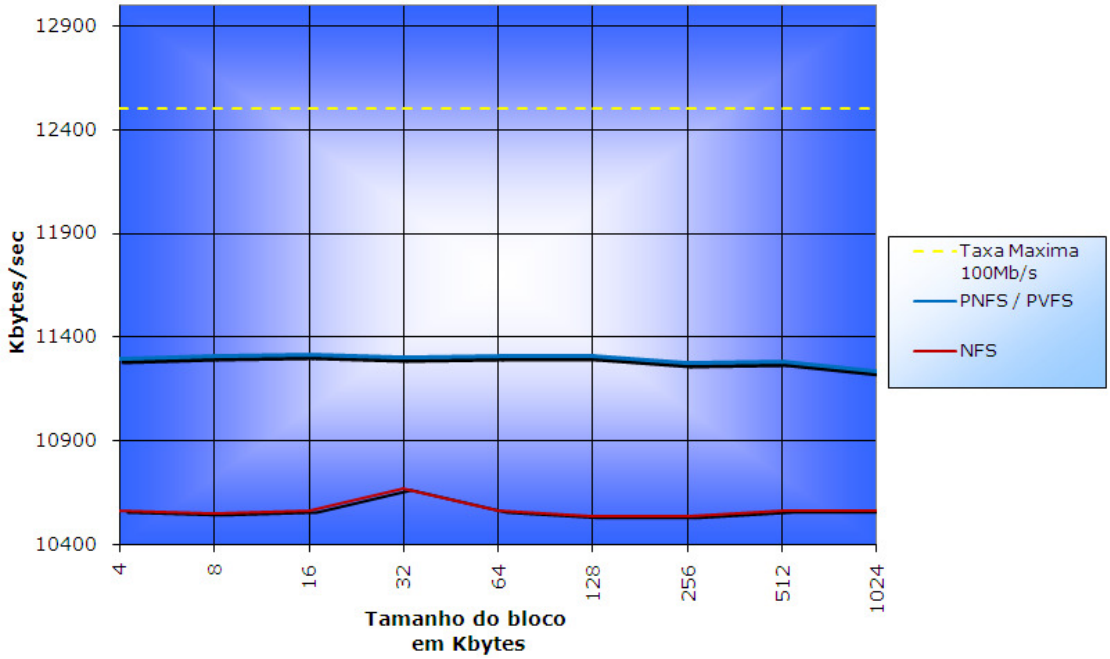

Figura 13 - Comparação de operações de leitura com 1 cliente em ambientes com NFSv3 e pNFS/PVFS2

Operações de escrita no NFSv3 têm melhor desempenho que as mesmas operações no PVFS/pNFS, como detalhado na Tabela 6 e mostrado na Figura 14. Esse fato pode ser explicado devido à carga extra de processamento quando se implanta balanceamento de carga com o pNFS/PVFS, sendo necessário a escrita do bloco transmitido em todos servidores de E/S disponíveis.

Outra explicação para um melhor desempenho em escrita do NFSv3 é a camada extra de controle que o pNFS/PVFS aplica para garantir a coerência de dados aplicado durante operações de escrita em vários servidores, o que leva a perda de desempenho para esse tipo de operação. Contudo, é possível notar a estabilidade do sistema com a implantação de balanceamento de carga através de pNFS/PVFS em relação ao NFSv3 puro, sendo que diminui as oscilações no acesso ao dados devido ao balanceamento da carga entre os servidores de E/S.

Tabela 6 - Comparação de velocidade de escrita em KB/s entre pNFS/PVFS2 e NFSv3

	Tamanho do bloco de escrita (KB)								
	4	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
pNFS/PVFS2	8242	8222	8217	8224	8214	8211	8198	8172	8178
NFSv3	9678	9255	9253	9099	8703	9525	7906	9027	9305

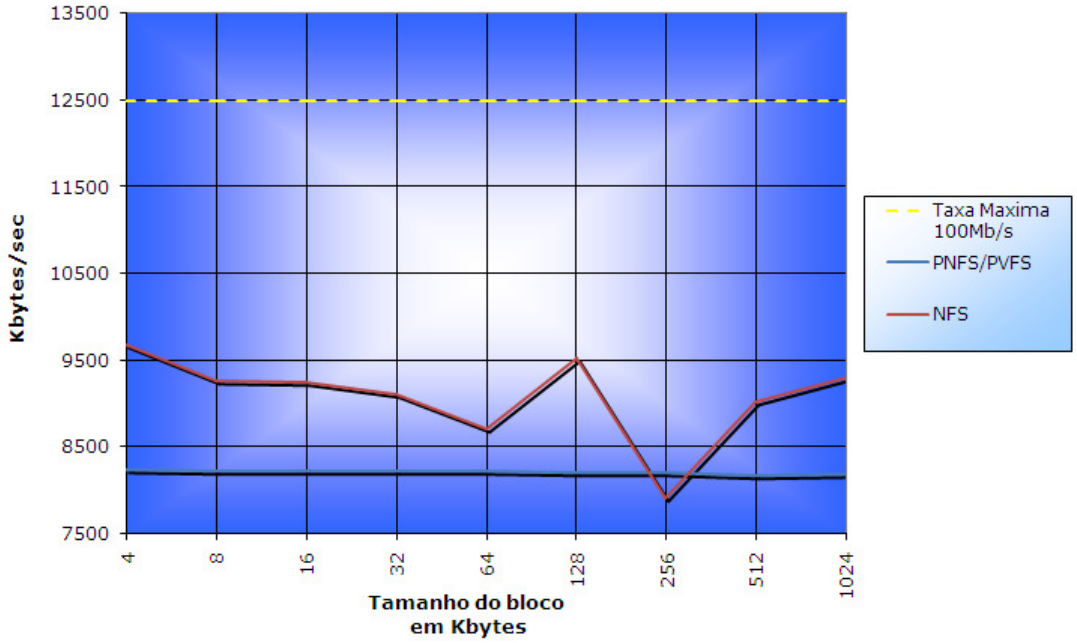

Figura 14 - Comparação de operações de escrita com um cliente em ambientes com NFSv3 e pNFS/PVFS2

6.2.3 Otimização de desempenho

Em busca de um melhor desempenho foram realizadas configurações adicionais no FlexLab, tanto no *middleware* Linux como no sistema de arquivos distribuídos e paralelos.

Sistema de arquivos distribuído

A escolha do NFSv3 como o sistema de arquivos distribuídos para a primeira camada do FlexLab foi devido ao uso da memória RAM, tanto o AFS como o CODA utilizam o *cache* agressivo (ZHAO & FIGUEIREDO, 2006), necessitando de mais memória RAM do que o NFSv3, o que dificultaria o uso desses sistemas em máquinas sem disco rígido e com memória RAM limitada.

Outro motivo para adoção do NFSv3 foi a compatibilidade entre os sistemas operacionais, permitindo que a segunda camada virtualizada do FlexLab também possa utilizar os recursos do NFSv3, diferentemente do AFS e CODA, que oferecem pouca compatibilidade com diversos sistemas operacionais.

Middleware

Para melhorar o tempo de *boot* do *middleware* algumas configurações extras foram realizadas. As áreas críticas do *middleware* que utilizam acesso demasiado ao servidor foram configuradas para terem seus dados alocados diretamente na RAM do cliente, dessa forma, as operações mais executadas no *middleware* já se encontram na memória e não precisam ser requisitadas ao servidor novamente. Experimentos realizados em 10 clientes (computadores disponíveis para os experimentos), na configuração da Tabela 4, com apenas um servidor, mostraram que o fluxo de dados na rede de comunicação diminui em 37% para pacotes UDP após as alterações de *cache* no *middleware*.

Para otimizar o desempenho do acesso aos arquivos de máquinas virtuais, que se encontram remotos, foram utilizadas configurações de *cache* no *middleware* (ZHAO & FIGUEIREDO, 2006) e nas implementações do pNFS/PVFS2 favorecendo o acesso aos arquivos contínuos.

O tamanho do bloco de transmissão foi configurado para 8 KB no *boot* do *middleware* (UDP) e 36 KB para o *boot* das máquinas virtuais (TCP) (KUACHAROEN, MOONEY & MADISETTI, 2002). Dessa forma a inicialização do sistema virtualizado se torna mais rápida em 8% comparada aos padrões de 4 KB no *middleware* e 8 KB na máquina virtual. Isso porque blocos maiores são mais aconselháveis quando se trabalham com arquivos contínuos como em máquinas virtuais (ZHAO, ZHANG & FIGUEIREDO, 2006). Aumentando o tamanho do bloco acima de 8KB no *middleware* e 36KBytes para máquina virtual a velocidade começa a diminuir novamente.

O *software* utilizado para verificar o fluxo de rede foi o ntop (NTOP, 2007). Após as alterações no *middleware* o tempo de *boot* diminuiu aproximadamente 10 segundos em relação ao *boot* sem as customizações, como visto na Figura 15, que mostra a diferença no tempo de *boot* do *middleware* em segundos antes e após as otimizações realizadas.

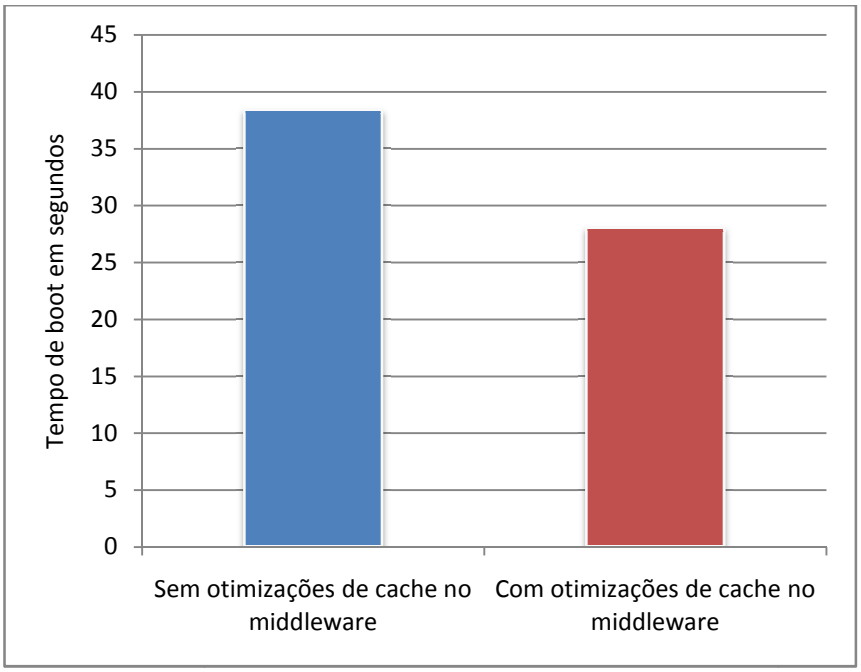

Figura 15 - *Boot* de 10 clientes com otimizações de *cache* no *middleware*

6.3 Experimentos do FlexLab

Os experimentos no Flexlab se basearam na extração de informações de tempo na inicialização das máquinas remotas, os valores são capturados através de *scripts* (vide Anexo A) que executam a leitura do *log* do DHCP e armazenam o tempo total de inicialização dos sistemas operacionais. Para garantir que os resultados sejam precisos é necessário que todos os computadores estejam com os horários do sistema operacional sincronizados. Para que essa característica fosse atingida foi utilizado o *Network Time Protocol* (NTP) (NTP, 1998).

Tabela 7 - Configuração de *hardware* para os experimentos de *boot* no FlexLab

Tipo	Configuração
Servidor 1	Intel Core2 Duo 2.0 Ghz, 1 GB de RAM, Disco rígido de 120 GB SATA, rede de 1Gb/s
Servidor 2	Intel Core2 Duo,2.2 Ghz, 2 GB de RAM, Disco rígido de 320 GB SATA, rede de 1Gb/s
Clientes	Intel Core 2 Duo 2.2 Ghz, 2 GB de RAM, Diskless, rede de 1Gb/s
Switch	100Mb/s

6.3.1 Experimentos de *Boot* do FlexLab

Foram realizados experimentos de *boot* no FlexLab para verificar o tempo extra exigido conforme clientes são acrescentados ao processo de inicialização remota.

A partir da sincronização de horários realizada, os *scripts* no *middleware* (vide Anexo A) se encarregam de inicializar os serviços necessários para controle de fluxo no momento do *boot*. Os experimentos foram realizados em rede *switch* isolada de 100Mb/s e seguiram a configuração de *hardware* citada na Tabela 7. Para comunicação e execução de processos remotos foi utilizado o *Remote Shell* (RSH) (RSH, 1994), podendo assim inicializar qualquer aplicação ou configuração remotamente através do servidor.

Os experimentos apresentados na Tabela 8 mostraram o tempo de *boot* de um ambiente virtualizado e distribuído em que os computadores clientes são ligados remotamente por *Wake-On-Lan*, sendo o Windows 2003 o sistema operacional virtualizado do experimento. O sistema de arquivos distribuído da primeira camada do FlexLab é o NFSv3 e da segunda camada o pNFS/PVFS2. O tempo de *boot* é a soma do tempo do *middleware* (mini kernel Linux) mais o tempo real de inicialização da imagem virtualizada (Windows 2003).

Tabela 8- Tempo de *boot* de um Windows 2003 virtualizado e distribuído para um número diversificado de clientes sem disco rígido

Cenários	Middleware FlexLab	Inicialização da Máquina Virtual	Tempo Total de *Boot*
1 cliente	0’12’’	0’36’’	0’48’’
5 clientes	0’23’’	1’08’’	1’31’’
10 clientes	0’33’’	1’51’’	2’24’’
15 clientes	0’50’’	2’55’’	3’45’’
30 clientes	1,44’’	4’23’’	6,07’’

Com o experimento é possível notar o aumento de 43 segundos no tempo de *boot* quando se aumenta o número de clientes de um para cinco computadores e mais 53 segundos quando são adicionados mais 5 clientes. Já ao adicionar 30 clientes são acrescentados 5 minutos e 19 segundos em relação ao *boot* de um único cliente. Durante os experimentos todas as máquinas foram ligadas simultaneamente através de *Wake-On-Lan* (vide Anexo A).

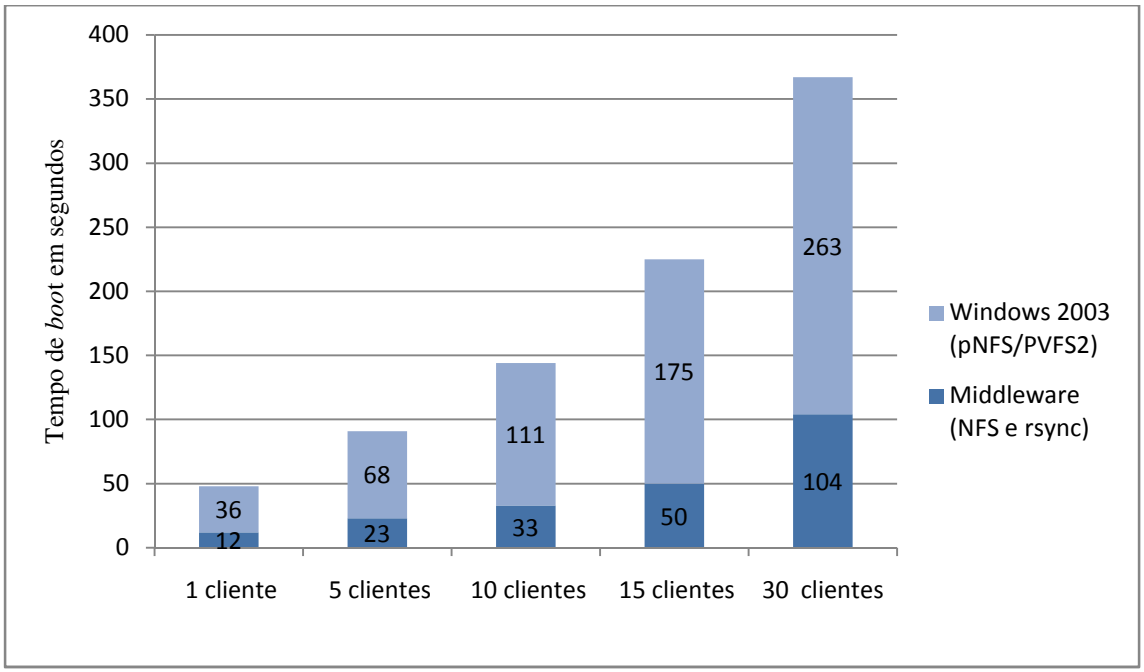

Figura 16 - Gráfico comparativo do tempo de *boot* do FlexLab

Através do gráfico mostrado na Figura 16 é possível notar a curva de crescimento do tempo de *boot* conforme são acrescentados novos clientes. Um detalhe importante é que mesmo depois de 30 clientes inicializarem remotamente, o próximo cliente a ser inicializado leva em média somente 55 segundos no *boot*, já que o servidor volta a ficar ocioso, dessa forma, novos clientes inicializados remotamente não terão um tempo de *boot* com valor alto.

Outra informação relevante é o fato de que ao adicionar um novo servidor de entrada e saída ao balanceamento de carga, o tempo é diminuído em aproximadamente 25% do tempo original, o mesmo acontece ao adicionar o terceiro servidor de balanceamento de carga. Os experimentos não contemplaram rede GB nos clientes, o que poderia ter diminuído ainda mais o tempo de *boot*.

6.3.2 Prova de conceito para *boot* paralelizado de imagens virtuais

Para viabilizar implementações de LAN Grid e analisar o comportamento da solução foi elaborado um experimento em que cada máquina cliente inicializa mais de um sistema operacional virtualizado simultaneamente, esse experimento é mostrado na Tabela 9.

Tabela 9- Tempo de *boot* de diversas máquinas virtuais inicializadas em um mesmo cliente simultaneamente

Clientes	Cenário	Total de Máquinas Virtuais	Tempo Total
30 clientes	5 instâncias do Windows 2003 inicializadas em cada cliente	150	7’15’’
30 clientes	40 instâncias de um mini Linux (Kernel 2.4) inicializadas em cada cliente	1200	3’38’’

No primeiro experimento foram inicializadas 5 instâncias do Windows 2003 em cada cliente. Ao ligar os 30 computadores foram totalizadas 150 máquinas virtuais simultaneamente no FlexLab com o tempo total de 7’15’’. O experimento mostrou que o *cache* do computador cliente age de forma agressiva visto que o tempo de *boot* de 150 máquinas virtuais cresce apenas 1 minuto e 8 segundos comparado ao *boot* de 30 clientes com apenas uma máquina virtual do Windows 2003, fato que pode ser reparado comparando a Tabela 8 com a Tabela 9.

No segundo experimento mostrado na Tabela 9 foram inicializadas 40 máquinas virtuais (mini Linux Kernel 2.4) em cada computador cliente, cada um utilizando 20 MB de RAM do computador. Como foram utilizados 30 computadores, são totalizadas 1200 máquinas virtuais simultaneamente no *cluster* multiuso, permitindo dessa forma que milhares de máquinas virtuais sejam inicializadas no FlexLab, tolerando, por exemplo, que aplicações de *grids* ou terminal remoto sejam aplicadas em escala.

Os experimentos forneceram informações relevantes para que aplicações em LAN Grid (através de máquinas virtuais) sejam pertinentes no ambiente proposto, sendo que o FlexLab provê uma ferramenta que facilita a implantação de sistemas distribuídos através de *clusters* multiuso (AGUIAR, et al., 2008)(b).

6.3.3 Comparativo do FlexLab com um *boot* convencional de disco rígido

Para validar o experimento foram criados diferentes modelos de *boot* como mostrado na Tabela 10. O experimento utilizou a configuração de *hardware* da Tabela 4, constatando, dessa forma, que o modelo distribuído e virtualizado é apenas 24 segundos mais lento que um sistema operacional inicializado do disco rígido e 18 segundos mais lento que o sistema operacional virtualizado a partir do disco rígido.

Nota-se que o valor total de *boot* também inclui o tempo de *boot* do *middleware*, por volta de 22 segundos do tempo total.

Tabela 10 - Comparativo de tempo de *boot* em um cliente a partir de diferentes modelos de inicialização

Modelo	Acesso	Middleware	Windows 2003	Tempo Total
Windows 2003 Sem Virtualização	Disco Rígido	--	0'40''	0'40''
Windows 2003 Virtualizado	Disco Rígido	--	0'46''	0'46''
Windows 2003 Virtualizado e Distribuído	Remoto	0'22''	0'42''	1'04''

A partir dos experimentos foi constatado que o *boot* da máquina virtual distribuída (Windows 2003) sem o *middleware* (42 segundos) é mais lento em dois segundos em relação ao *boot* da máquina virtual a partir do disco rígido (40 segundos) (disco rígido de 5400 rpm). Esse experimento indica que um *boot* de uma máquina virtualizada com balanceamento de carga pode ser mais rápido do que um *boot* através de disco rígido, caso a velocidade de rede de comunicação e o disco rígido do servidor sejam mais rápidos do que o a velocidade de acesso ao disco rígido do cliente (os experimentos não contemplaram rede de 1Gb/s).

Os experimentos também não contemplaram testes de aplicações LAN *grids*, visto que a idéia do projeto é criar um ambiente para facilitar a execução dessas ferramentas, mas a sua aplicação em si e resultados obtidos junto ao FlexLab podem ser abordados em trabalhos futuros.

Características extras como utilizar os próprios clientes para auxiliar a distribuição de dados do *middleware* diminuíram o tempo de *boot*, contudo, ainda é uma função limitada a 4 computadores e resultou em uma melhora de 14% no tempo do boot do *middleware* em relação ao sistema convencional, sendo uma característica que também deve ser trabalhada em projetos futuros.

6.4 Considerações Finais

Através dos experimentos realizados foi possível escolher e configurar as tecnologias que mais se adequaram ao FlexLab. Configurações de *cache* e tamanho de blocos para os sistemas de arquivos distribuídos foram fundamentais para garantir o

sucesso dos experimentos, assim como melhorias na configuração do *middleware* Linux, o qual serviu como base para um bom funcionamento de todo conjunto.

A implantação de um sistema de arquivos através da mistura de PVFS2 com pNFS trouxe melhorias ao FlexLab de até 16% em operações de leitura e permitiram a escalabilidade da solução juntamente com o balanceamento de carga agregado ao PVFS2, possibilitando o crescimento do número de clientes conectados ao sistema. O gerenciamento da memória alocada e a diminuição da sobrecarga de fluxo de dados com pacotes UDP também foram essenciais para uma melhora de até 37 % na velocidade de *boot* do *middleware*.

Com os experimentos foi possível verificar o aumento do tempo de *boot* conforme computadores clientes são adicionados ao conjunto. Também foi possível notar que ao adicionar um novo servidor de E/S ao balanceamento de carga o tempo é diminuído em aproximadamente 25% do tempo original.

Para viabilização dos experimentos com máquinas virtuais distribuídas foi constatado que o modelo distribuído e virtualizado é apenas 24 segundos mais lento que um sistema operacional real inicializado do disco rígido e 18 segundos mais lento que o sistema operacional virtualizado inicializado a partir do disco rígido, já contando o tempo de *boot* do *middleware.* Dessa forma, pode-se confirmar a viabilidade do uso de sistemas operacionais virtualizados através da rede de comunicação.

Mesmo notando que o desempenho do sistema de arquivos paralelos é pior que o desempenho do NFSv3 para operações de escrita, o pNFS/PVFS2 foi escolhido como melhor opção para o FlexLab já que traz maior estabilidade ao sistema e operações de leitura mais rápidas, uma vez que esse tipo de operações são mais comuns em *boot* de sistemas operacionais convencionais.

Os experimentos mostraram que o FlexLab permite que os computadores sejam gerenciados remotamente de forma a aumentar o seu tempo de uso, concentrando além do uso normal dos computadores com a camada de virtualização, também a possibilidade de executar aplicações paralelizadas como *clusters* multiuso e LAN *grids* a partir de *scripts* de automatização (vide Anexo A). Dessa forma, o FlexLab fornece ferramentas de gerenciamento no servidor, permitindo a administração centralizada de

computadores na rede, além de auxiliar a implantação e manutenção de aplicações distribuídas.

Capítulo
7

Conclusão

7.1 Conclusões Gerais

A motivação por trás da virtualização distribuída tem grande importância na conectividade de computadores e sistemas, tanto em redes locais, como em redes de larga distância. Novas oportunidades para aplicar a virtualização surgem neste contexto, em que os computadores estão cada vez mais conectados e aplicações em *clusters* e *grids* estão cada vez mais presentes. O FlexLab provê um veículo rápido para viabilizar idéias de pesquisas em sistemas de computação distribuída além de facilitar o gerenciamento e replicação de sistemas operacionais distribuídos.

As técnicas de gerenciamento de memória aplicadas ao FlexLab e o balanceamento de carga aplicado nos experimentos aumentaram a importância da localização dos dados, diminuindo dessa forma a latência para o carregamento das imagens virtuais, o que viabiliza o uso de máquinas virtuais em instituições de ensino e empresas, podendo se aproveitar de recursos ociosos dos computadores pouco utilizados. O modelo do FlexLab permite que sistemas operacionais sejam centralizados no servidor e que todo gerenciamento também seja vinculado ao mesmo, permitindo um fácil gerenciamento e manutenção das imagens virtuais remotamente.

Dessa forma, o FlexLab fornece ferramentas que facilitam o gerenciamento de aplicações em *clusters* multiuso e LAN *grids* em computadores *commodity*, podendo ser utilizado tanto por aplicações científicas quanto para gerenciamento de parque de máquinas convencionais. Os experimentos mostraram que o FlexLab permite que um dezenas de computadores sejam gerenciados remotamente de forma a aumentar o seu tempo de uso e diminuindo o custo de manutenção do mesmo.

Os experimentos também permitiram a viabilidade para execução de aplicações LAN *Grid* virtualizadas, sendo que o FlexLab fornece uma camada que facilita a implantação e manutenção de sistemas distribuídos através de *clusters* multiuso.

O FlexLab permite que os laboratórios de computadores sejam gerenciados de forma a aumentar o seu tempo de uso, já que permite além do uso normal dos computadores com a camada de virtualização, também a possibilidade de executar aplicações paralelizadas como *clusters* e *grids*.

Muitas das exigências por ambientes mais robustos e flexíveis podem ser auxiliadas por programação distribuída, que venha a ser o núcleo do processamento e distribuição dos dados. A diversidade de plataformas utilizadas nos computadores se torna uma barreira na atualização de *software*, uma vez que *hardwares* diferentes dificultam a entrada de novas tecnologias. Esta diversidade pode ser abstraída com o FlexLab, dessa forma, a união do poder de processamento distribuído e a flexibilidade das máquinas virtuais trazem grandes benefícios para pesquisas em sistemas de computação.

7.2 Contribuições deste Trabalho

O desenvolvimento do FlexLab abordou diversas tecnologias e técnicas que podem ser utilizadas como base para futuros trabalhos e pesquisas acadêmicas:

- Um estudo bibliográfico crítico sobre as principais tecnologias que envolvem aplicações distribuídas como os sistemas de arquivos distribuídos, balanceamento de carga e gerenciamento de *boot* remoto.

- Abordagens de possíveis utilizações e técnicas que envolvem a virtualização e como elas podem se integrar e aprimorar características pouco implementadas em ambientes distribuídos.

- Uma análise do uso de aplicações que envolvem *cluster* multiuso aproveitando recursos ociosos em laboratórios de informática e estações de trabalho, viabilizando o uso de aplicações distribuídas em computadores *commodity*.

- A utilização de LAN *grids* implantados a partir de *clusters* multiuso e controlados por um *middleware* gerenciável, permitindo a criação de novas aplicações para computadores ociosos.

- A viabilização de um pequeno *middleware* Linux compatível com a maioria dos *hardwares* existentes que permite a inicialização de um sistema virtualizado remotamente.

- O modelo do FlexLab, um ambiente de *boot* remoto com balanceamento de carga aplicável ao gerenciamento e replicação de sistemas operacionais virtualizados remotamente.

7.3 Trabalhos Futuros

Como indicativo para continuidade deste trabalho são citados:

- Adaptar o FlexLab para a internet de alta velocidade, disponibilizando imagens virtuais para o *boot* de máquinas via TCP/IP, dessa maneira, um usuário pode ter acesso a um sistema operacional individualizado de qualquer lugar que tenha conexão com a internet.

- Criar um sistema de gerenciamento remoto para controle das imagens virtuais, para que administradores de rede consigam atualizar e gerenciar o laboratório de informática através da internet, assim como visualizar resultados dos experimentos de aplicações distribuídas que se aproveitam dos períodos de ociosidade dos computadores.

- Elaborar aplicações de *grids* que utilizem os recursos do FlexLab e que aproveitem os recursos disponíveis durante períodos de inatividade dos laboratórios de informática.

- Aprimorar o recurso de balanceamento de carga através dos computadores clientes para um número indeterminado de máquinas, permitindo que requisições de blocos de dados não sejam exclusivamente distribuídos por servidores, mas também por computadores clientes que já carregaram blocos de dados previamente.

- Desenvolver aplicações de virtualização distribuída em dispositivos móveis, permitindo que o ambiente virtual do usuário seja carregado independentemente do dispositivo utilizado, assim como a melhor utilização de recursos de virtualização aplicados diretamente em *hardware*.

Referências Bibliográficas

AGUIAR, C. S; CRUZ, D. I.; ULSON, R. S.; CAVENAGHI, M. A. (2008). **The Application of Distributed Virtual Machines for Enterprise Computer Management: a two-tier network file system for image provisioning and management** . *32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008) - (a ser publicado)(a)* .

AGUIAR, C. S; CRUZ, D. I.; ULSON, R. S.; CAVENAGHI, M. A. (2008). **The Application of Virtualized Multiuse Clusters for LAN Grid Network Computer Management**. *Poster in 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2008) - (a ser publicado))(b).*

ANTHONY, H; CHRISTIAN, D.,J. (2003). **Cryptographic Access Control in a Distributed File System**. *8th ACM Symposium on Access Control Models and Technologies.* Junho 2003, Nova York, p. 158-165.

BARATTO, R; KIM, L.; NIEH. J. (2005). **THINC: A Virtual Display Architecture**. *ACM Symposium on Operating Systems Principles.* 2005, p. 277 - 290 .

BARHAM, P.; DRAGOVIC, B. (2003). **Xen and the Art of Virtualization**. *19th ACM Symposium on Operating Systems Principles (SOSP 2003).* 2003, p. 164 -177.

BELLARD, F. (2004). **QEMU**. Disponível em: http://fabrice.bellard.free.fr/qemu/. Acesso em: 08 out. 2007.

BEOWULF. (1994). **Beowulf**. Disponível em: http://www.beowulf.org/. Acesso em: 23 set. 2007.

BLAKE M.I; HUANG, K.; SHIAU, S.; WANG, H. T. (2004). **Diskless Remote Boot Linux**. Disponível em: http://drbl.nchc.gov.tw/. Acesso em: 10 out. 2007.

BROWN, P. (1998). **Introducing the Network Information Service for Linux**. *Linux Journal.* 1998, 50es (Jun. 1998), artigo 4.

BUTT, A. R.; ADABALA, S.; KAPADIA, N. H.; FIGUEIREDO, R. J. (2002). **Fine-grain access control for securing shared resources in computational grids**. *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS* 2002) Washington, DC, EUA, p. 22-24.

BXP. (2003). Disponível em: http://www.ardence.com. Acesso em: 15 ago. 2007.

CARNS, P. H.; ROSS, R. B. L.; WALTER B.; WYCKOFF, P. 2005. **A network abstraction layer for parallel i/o**. *19th International Parallel and Distributed Processing Symposium (IPDPS 2005)* IPDPS. IEEE Computer Society, Washington, DC, p. 213-220.

CARNS, P. H.; ROSS, R. B.; L., WALTER B.; WYCKOFF, P., et al. (2000). **PVFS: A Parallel File System For Linux Clusters**. *Proceedings of the 4th Annual Linux Showcase and Conference. Linux J.* 2000, p. 317-327.

CHEN, S.; TOWSLEY, D. (1996). **A Performance Evaluation of RAID Architectures**. *IEEE Transactions on Computers.* p. 1116 - 1130 .

CLARK, C.; FRASES, K.; HAND, S.; HANSENY, G. (2005). **A live migration of virtual machines**. *Second Symposium on Network Systems Design and Implementation Volume 2* (May 02 - 04, 2005). USENIX Association, Berkeley, CA, p. 273-286.

DIBBLE, P.; SCOTT, M.; ELLIS, C. (1988). **Bridge: A high-performance file system for parallel processors**. *Eighth International Conference on Distributed.* Amsterdam. p. 154-161.

DODONOV, E.; MELLO, R.; YANG, L. T. (2006). **Adaptive technique for automatic communication access pattern discovery and data prefetching in distributed applications using neural networks and stochastic models**. *ISPA 2006*, p. Sorrento, Itália. p. 292-303.

DRBL. (2002). **DRBL**. Disponível em: http://drbl.sourceforge.net/. Acesso em: 15 nov. 2007.

DROMS, R. (1997). **Dynamic Host Configuration Protocol. *DHCP***. Disponível em: ftp://ftp.rfc-editor.org/in-notes/rfc2131.txt. Acesso em: 10 nov. 2007.

DUNLAP, G. W.; KING, S. T.; CINAR, S., BASRAI, M.; CHEN, P. M. (2002). **ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay**. *ACM SIGOPS Operating Systems Review.* 2002, Boston, MA. p. 211–224.

EXT3. (1999). **Third Extended Filesystem**. Disponível em: http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html. Acesso em: 05 ago. 2007.

EYLON, D; AMIT, R.; YEHUDA, V. (2001). **Method and System for Executing Network Streamed Application**. US Patent 6574618.

FAT. (1980). **File Allocation Table**. Disponível em: http://home.no.net/tkos/info/fat.html. Acesso em: 23 set. 2007.

FIGUEIREDO, R. J.; FORTES, J.A.; DINDA, P. (2002). **A Case For Grid Computing On Virtual Machines**. ICDCS. IEEE Computer Society, Washington, DC, p. 550-556.

FIGUEIREDO, R. J.; KAPADIA, N. H.; FORTES, J. A. (2004). **Seamless Access to Decentralized Storage Services in Computational Grids via a Virtual File System**. *In Cluster Computing Journal.* 2004, p. 113-122.

FIGUEIREDO, R. J.; KAPADIA, N. H.; FORTES, J. A. (2001). **Towards Coordinated Work on the Grid: Data Sharing Through Virtual File Systems**. *In Mini-Workshop on Internet Process Coordination and Ubiquitous Computing*. December 2001, Florida. CRC Press p.151-16.

FONSECA, A.A.; VIANNA, N.T; MOURA, L.T.; MENEZES, C.; BOERES, F. R. (2005). **Um Ambiente para o Desenvolvimento e Avaliação de Algoritmos de Escalonamento para Grades Computacionais**. *Anais do 5º Workshop em Sistemas Computacionais de Alto Desempenho*. Foz do Iguaçu - PR – Brasil. p.120,132.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. (2001). **The anatomy of the grid: Enabling scalable virtual organizations**. *International Journal of Supercomputer Applications 2001*. p. 200-222.

GALLI. A; DOREEN, L. (2000). **Distributed Operating Systems**. s.l. : Prentice Hal, 2000. ISBN-10: 0132199084.

GUILHERMO, C.; ALVAREZ, A.; ELIZABETH, B.; SUSIE, G.; THEODORE, H. (2001). **Minerva: An automated resource provisioning tool for large-scale storage systems**. *ACM Transactions on Computer Systems (TOCS)*. vol. 19, no4, p. 483-518

GUTKNECHT, J.; WIRTH, N. (1992). **Project Oberon**. *ACM Press* 1992. ISBN 0-201-54428-9-8.

HENRI, C.; KAYO, F.; ARNAULD, L.; MARTIN, Q. (2006). **The SIMGRID Project: Simulation and Deployment of Distributed Applications**. *The 15th IEEE International Symposium on High Performance Distributed Computing (HPDC'06)* Paris. p 385-386.

HILDEBRAND, D.; HONEYMAN, P. (2007). **Direct-pNFS: Scalable, transparent, and versatile Direct-pNFS: Scalable, transparent, and versatile**. *HPDC'07*. p. 199-208

HILDEBRAND, D.; WARD, L.; HONEYMAN, P. (2006). **Large Files, Small Writes, And pNFS**. *ICS06.* Cairns, Queensland, Australia, 28 de Junho. pp 116-124.

HITZ, D.; LAU, J.; MALCOLM, M. (1994). **File system design for an NFS file server appliance**. *Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference*. USENIX Association, Berkeley, CA, p. 19-20

HOWARD, J. (1988). **Overview of the Andrew File System**. *Usenix 1988 Winter Conference Proceedings.* 1988, p. 23-26.

HPFS. (1989). **High Performance File System**. Disponível em: http://pages.cs.wisc.edu/~bolo/shipyard/hpfs.html. Acesso em: 25 jun. 2007.

INTEL CORPORATION. (1999). **Preboot Execution Environment (PXE) Specification**. Disponível em: http://www.pix.net/software/pxeboot/archive/pxespec.pdf. Acesso em: 28 nov. 2007.

JITHENDAR, P.; ZHAO, M.; FIGUEIREDO, R.J. (2004). **Support for Data-Intensive, Variable-Granularity Grid Applications via Distributed File System Virtualization - A Case Study of Light Scattering Spectroscopy**. *Second International Workshop on Challenges of Large Applications in Distributed Environments (CLADE 2004).* 2004, p. 12-21.

KAPADIA, N. H.; FIGUEIREDO, R. J.; FORTES, J. A. B. (2001). **Enhancing the Scalability and Usability of Computational Grids via Logical User Accounts and Virtual File Systems**. *Heterogeneous Computing Workshop (HCW 2001)*. IPDPS. IEEE Computer Society, Washington, DC p.82-82.

KAZAR, A.; LEON, M. (1988). **Synchronization and Caching Issues in the Andrew File System**. *Usenix 1988 Winter Conference Proceedings*. Dallas, Texas. p. 155 - 162

KISTLER, J.J; SATYANARAYANAN, M. (1992). **Disconnected Operation in the Coda File System**. *ACM Transactions on Computer Systems*. Vol. 10, p. 3-25.

KON, F. (2007). **Distributed file systems past, present and future a distributed file system**. Disponível em http://choices.cs.uiuc.edu/»f-kon/. Acesso em: 23 dez. 2007.

KOTZ, D.; ELLIS, C. S. (2001). **Practical prefetching techniques for multiprocessor file systems**. *IEEE Computer Society Press.* 2001, p. 245–258.

KRSUL, I.; GANGULY, A.; ZHANG, Z. (2004). **VMPlants: Providing and Managing Virtual Machine**. *2004 ACM/IEEE conference on Supercomputing*. 2004, Washington, DC, EUA. p. 2-10.

KUACHAROEN, P.; MOONEY, V.; MADISETTI, V. K. (2002). **Software Streaming via Block Streaming**. *Tech. Rep. GIT-CC.* 2002, p. 2-63.

KURMANN, C.; RAUCH, F.; STRICKER, T. M. (2003). **Cost/Performance Tradeoffs in Network Interconnects for Clusters of Commodity PCs**. *In Workshop on Communication Architecture for Clusters, in conjunction with International Parallel and Distributed Processing Symposium (IPDPS).* p. 22-26.

LAY, A, M.; NIEH, J. (2006). **On the performance of wide-area thin-client computing**. *ACM Transactions on Computer Systems (TOCS).* 2006, p. 175 - 209.

LOMBARD, P.; DENNEULIN, Y. (2001). **NFSP**: **A distributed NFS server for cluster of workstation**. Disponível em http://ka-tools.sourceforge.net/publications/nfsp-ipdps01. Acesso em: 13 ago. 2008.

MADNICK, S. E.; DONOVAN, J. J. (1973). **Application and analysis of the virtual machine approach to information system security and isolation**. In Proc. ACM SIGARCH-SYSOPS Workshop. Março 1973, p. 210–224.

MCNETT, M.; GUPTA, D.; VAHDAT, A.; VOLKER, G. (2007). **Usher: An Extensible Framework for Managing Clusters of Virtual Machines**. *Large Installation System Administration Conference LISA '07* Berkeley, CA, EUA. p. 167-181.

MCQUILLAN, M. (2002). **Linux Terminal Server Project**. *Linux Terminal Server Project.* Disponível em: http://www.ltsp.org/. Acesso: 03 ago. 2007.

MENASCÉ, D. A.; PENTAKALOS, O. I.; YELENA, Y. (1996). **An analytic model of hierarchical mass storage systems with network-attached storage devices**. *ACM SIGMETRICS Performance Evaluation Review*. p. 180-189.

MOORE, G. E. (1965). **Moore's Law**. Disponível em: http://www.intel.com/research/silicon/mooreslaw.htm. Acesso em: 23 jul. 2007.

MOORE, J.; IRWIN, D.; GRIT, L.; SPRENKLE, S. (2002). **Managing mixed-use clusters with Cluster-on-Demand**. *Technical report, Duke University, Department of Computer Science. Duke University, Nov. 2002*. p. 2-17.

NORCOTT, W.D.; CAPPS, D. (2003). **IOZone Filesystem Benchmark**. Disponível em: http://www.iozone.org/. Acesso: 25 nov. 2007.

NTFS. (1998**). New Technology File System**. Disponível em http://www.ntfs.com/. Acesso em: 10 jun. 2007.

NTOP. (2007). **ntop**. Disponível em: http://www.ntop.org/. Acesso em: 22 jul. 2007.

NTP. (1998). NTP. Disponível em: http://info.connect.com.au/docs/ntp/. Acesso em: 20 set. 2007.

PANASAS. (2004). **Panasas Inc. Panasas ActiveScale File System**. Disponível em: http://www.panasas.com. Acesso em: 16 set. 2007.

PHILIP, H.C.; WALTER, B.; LIGON, III; ROBERT, B.; RAJEEV, R. T. (2000). **PVFS: A parallel virtual file system for linux clusters**. *In Proceedings of the 4th Annual Linux Showcase and Conference*. 80es (Nov. 2000), p. 5-6.

PROTIC, J. (1996). **Distributed Shared Memory: Concepts and Systems**. *IEEE Parallel and Distributed Technology.* vol. 04, no. 2, p. 63-79.

RAUCH, F.; KURMANN, C.; MULLER-LAGUNEZ, B. M. (1999). **Patagonia - A Dual Use Cluster of PCs for Computation and Education**. *2nd Workshop Cluster Computing*, p. 65-75.

REISER, M. (1991). **The Oberon System**. *ACM Press*. 1991; ISBN 0-201-54422-9.

ROBIN, J. S.; IRVINE, C.E. (2000). **Analysis of the Intel Pentium's ability to support a secure virtual machine monitor**. *Proceedings of the 9th USENIX Security Symposium*, Denver, Colorado. p. 129–144.

ROSE, R. (2004**). Survey of System Virtualization Techniques**. Disponível em: http://www.robertwrose.com/vita/rose-virtualization.pdf. Acesso em: 07out. 2007.

ROSS, R.; LIGON, W.; CARNS, P.l. (2002). **Parallel Virtual File System**. Disponível em: http://www.parl.clemson.edu/pvfs2/sc2002-whitepaper-pvfs.pdf. Acesso em: 08 nov. 2007.

ROY, H. S.; LEE, D. C.; HUNG, D.; CHOLI, W.; FRANCIS, C.M. (2004**). On Managing Execution Environments for Utility Computing**. *Network Research Workshop, 18th Asia-Pacific Advanced Network Meetings (APAN 2004).* Cairns, Australia, p. 175-182.

RSH. (1994). **Remote Shell**. Disponível em: http://unixhelp.ed.ac.uk/CGI/man-cgi?rsh. Acesso em: 20 nov. 2007.

SATYANARAYANAN, M. (1989). **Coda: A highly available file system for a distributed workstation environment**. *Second IEEEWorkshop onWorkstation Operating Systems*. p. 447-459.

SLIM. (2004). **SLIM**. **Single Linux Image Management**. Disponível em http://slim.cs.hku.hk/. Acesso em: 10 mai. 2007.

SMITH, E.J.; NAIR, R. (2005). **Virtual Machines: Versatile Platforms for Systems**. 2005, p. 50-65.

SOLLINS, K. (1992). **TFTP: Trivial File Transfer Protocol**. Disponível em: http://tools.ietf.org/html/rfc1350. Acesso em: 20 set. 2007.

STRICKER M. S. (2000). **Patagonia. A Dual Use Cluster of PCs for Computation and Education**. http://www.cs.inf.ethz.ch/CoPs/patagonia/. Acesso em: 12 set. 2007.

SUGERMAN, J.; VENKITACHALAN, J. (2001). **Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor**. *USENIX Annual Technical Conference* 2001. Boston, MA. p. 440 – 452.

SUN MICROSYSTEMS Inc. (1989). **NFS: Network file system protocol specification**. *Request for Comments (RFC) 1094, Network Working Group*.

TANENBAUM, A. S. (1992). **Modern Operating Systems**. *Prentice Hall.* 1992. ISBN 0130313580.

TELIKEPALLI, D.; DRWIEGA, R.; YAN, T.J. 2004. **Storage area network extension solutions and their performance assessment**. *Communications Magazine, IEEE.* 2004, p. 56- 63.

TRIDGELL, A.; MACKERRAS, P. (1998). **The rsync algorithm. *Rsync***. Disponível em: http://rsync.samba.org/rsync/tech_report/. Acesso em: 22 set. 2007.

UNISON. 2001. **Unison**. Disponível em: http://www.cis.upenn.edu/~bcpierce/unison/. Acesso em: 10 out. 2007.

VIRTUAL BOX. (2007). **Virtual Box**. Disponível em: http://www.virtualbox.org/. Acesso em: 16 nov. 2007.

VMWARE. (1998). **VMware**. *VMware.* http://www.vmware.com/. Acesso em: 14 nov. 2007.

VOELKER, G. M. (1998). **Implementing cooperative prefetching and caching in a globally-managed memory system**. *Proceedings of the Joint International Conference on Measurement an Modeling of Computer System. ACM Press.* 1998, p. 33-43.

WARFIELD, A.; ROSS, R.; FRASER, K. (2005). **Parallax: Managing Storage for a Million Machines**. *Proceedings of the 10th conference on Hot Topics in Operating Systems.* Volume 10. USENIX Association, Berkeley, CA, p. 4-4.

WEBER, R.O. (2004). **SCSI Object-Based Storage Device Commands (OSD)**. *Storage Networking Industry Association. Working draft NCITS TBD-200X.*, Revision 3, p. 5-5.

WHITAKER, A. M.; SHAW, S. D.; GRIBBLE, D. (2002). **Lightweight Virtual Machines for Distributed and Networked Applications**. *Technical Report 02-02-01, Universidade de Washington.* 2002. Monterey, CA. p. 10-10.

XIBALTA. (2003). **Xibalba**. *Computer Science Cluster of ETH Z¨urich.* Disponível em: http://www.xibalba.inf.ethz.ch/, 2003. Acesso em: 10 set. 2007.

XU, J.; ADABALA, S.; FORTES, J. A. (2005). **Towards Autonomic Virtual Applications in the In-VIGO System**. *Second International Conference on Autonomic Computing (ICAC'05)*. Washington, DC, p. 15-26.

YANG, C.; PING-I, C.; YA-LING, C. (2005). **Performance Evaluation of SLIM and DRBL Diskless PC Clusters on Fedora**. *Sixth International Conference on Parallel and Distributed Computing (PDCAT'05)*. Washington, DC, p. 479-482.

ZHANG, Y.; GU, Y.; WANG, H.; WANG, D. (2006). **Virtual-Machine-based Intrusion Detection on File-aware Block Level Storage**. *18th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'06)*, p. 185-192.

ZHAO, M J; ZHANG, H.; FIGUEIREDO, R. (2004). **Distributed File System Support for Virtual Machines in Grid Computing**. *High Performance and Distributed Computing*. Washington, DC, p. 202-211.

ZHAO, M.; ZHANG, J.; FIGUEIREDO, R. J. (2006). **Distributed File System Virtualization Techniques Supporting On-Demand Virtual Machine Environments for Grid Computing**. *Cluster Computing*. p. 45 - 56.

ZHAO; MING; FIGUEIREDO, R. J. (2006). **Application-Tailored Cache Consistency for Wide-Area File Systems**. *International Conference on Distributed Computing Systems (ICDCS 2006)*. Washington, DC. p. 41-41.

ZHOU, Z.; XIE, Y. (2006). **Virtual Disk based Centralized Management for Enterprise Networks**. *SIGCOMM'06.* ACM 1-59593-417-0/06/0009, p. 11-15.

Anexo A

Script para gravar *log* de inicialização dos clientes no servidor

```
#!/bin/sh

## CRIA ARQUIVO QUE REGISTRA A HORA QUE CADA MAQUINA LOGOU NO DHCP

## COPIA ARQUIVO ANTIGO PARA PASTA DE BACKUP
fileBKP="/testesNFS/backup/result_`date +%T`_d`date +%d`_m`date
+%m`.txt"
cp /testesNFS/resultados.txt $fileBKP

## REMOVE ARQUIVO RESULTADOS.TXT E CRIA NOVO ATRIBUINDO TODAS AS
##PERMICOES

rm -f /testesNFS/resultados.txt
touch /testesNFS/resultados.txt
chmod 666 /testesNFS/resultados.txt

##REGISTRA HORA QUE CADA MAQUINA LOGOU NO LOG
echo "--------------------------------" >>

/linux/images/fc6/home/flexlab/resultados.txt
grep DHCPOFFER /var/log/messages | awk '{print $6,"-->", $10,"-->"
,$3, $1, $2 }' >> /testesNFS/resultados.txt


echo "--------------------------------" >>
/linux/images/fc6/home/flexlab/resultados.txt
```

Script para verificar uso de memória RAM dos clientes

```
#!/bin/sh

##VERIFICA QUANTIDADE DE MEMÓRIA UTILIZADA NOS CLIENTES

MEMTOTAL=$(cat /proc/meminfo | grep MemTotal | awk '{print $2}')
FREEMEM=$(free | grep buffers/cache | awk '{print $4}')

echo “MEMÓRIA LIVRE = “$FREEMEM
echo “MEMÓRIA TOTAL = “$MEMTOTAL
```

Script para inicializar experimento com IOZone em todos clientes simultaneamente

```
#!/bin/sh

##SCRIPT PARA COPIAR ARQUIVO DE 12 MB COM BLOCO DE 4KB EM TODOS
##CLIENTES

mkdir /opt/testes_iozone
/opt/iozone/bin/iozone -t 1  -i 0 -i 1 -c -z -r 4 -s 12m -b
/opt/testes_iozone/resultados_testes.xls -+m /iozone_gabriel/script1
```

Script em python para ligar clientes e capturar o horário

```
#!/usr/bin/python
#coding: utf-8

import os
import time

##INSERIR MAC DOS CLIENTES
clients = []

## TRECHO PARA LIMPAR LOG DO DHC, APÓS ESSE PROCEDIMENTO É EXECUTADO O
##SCRIPT GREPDHCP.SH QUE REGISTRA A HORA QUE CADA MAQUINA RECEBEU O IP
##PELO DHCP

os.system("rm /var/log/messages")
os.system("touch /var/log/messages")
os.system("/etc/init.d/syslog restart")

##COMANDO WAKEUP PARA LIGAR AS MÁQUINAS REMOTAMENTE
wake = "wakeup %s"

for cli in clients:
      cmd = wake %  cli
      print cmd
      os.system(cmd)

scriptDHCP="sh /FlexLab/scriptsFlexLab/grepdhcp.sh"
os.system(scriptDHCP)
```

Script em python para inicializar aplicações remotamente e capturar o tempo de execução

```
#!/usr/bin/python

## THREAD PARA ENVIAR COMANDOS RSH PARA VARIAS MAQUINAS CLIENTES

import threading
import os

##ÚLTIMO IP DA LISTA DO DHCP
finalIP = 230

##CLASSE DEFINIDA CADA CLIENTE
class MyThread(threading.Thread):
      def run(self):
            global finalIP
            finalIP= finalIP +1
            ##COMANDO UTILIZADO PARA INICIAR UMA APLICAÇÃO REMOTAMENTE
            cmd = 'rsh IP.%s -l guest sh
            /home/flexlab/scripts/startoofice.sh' % finalIP
            os.system(cmd)
            print cmd


##LAÇO DE N REPETICOES PARA INICIALIZAR AS THREADS

for x in range(N):
      MyThread().start()
```

Script para gerenciar a inicialização do PXE

```
PROMPT 1
##COMANDO PARA MOSTRAR MENU NO BOOT
DISPLAY messages

##PRIMEIRA IMAGEM DE BOOT
LABEL 1
KERNEL vmlinuz-2.6.9-1.667smp_v1
APPEND root=/dev/ram0 imageroot=192.168.2.252:/linux/images/fc3
initrd=initrd-2.6.9-1.667smp_v1

##SEGUNDA IMAGEM DE BOOT
LABEL 2
KERNEL vmlinuz-2.6.18-1.2798.fc6
APPEND root=/dev/ram0 imageroot=192.168.2.252:/linux/images/fc6
initrd=initrd-2.6.18-1.2798.fc6
```

Script para inicializar o *Middleware* gerenciando o *cache* do FlexLab

```
!/bin/ash
## CRIA VARIÁVEIS COM AS CONFIGURAÇÕES DE CÓPIA PARA O CACHE
if [ -f /mnt/nfsroot/sbin/hwclock ]; then
      /mnt/nfsroot/sbin/hwclock --hctosys
fi
if [ -f /mnt/nfsroot/usr/bin/rsync ]; then
      echo "carregado por rsync"
      CP="/mnt/nfsroot/usr/bin/rsync -azv --delete"
      EXCLUDE="--exclude=rpm --exclude=scrollkeeper"
else
      CP="/mnt/nfsroot/bin/cp -u"
      EXCLUDE=""
Fi

## MONTA O SISTEMA NFS
$CP mnt/nfsroot/root sysroot/
mkdir -p mnt/nfsroot

echo "CONSTRUINDO UM SYSTEMA DE ARQUIVOS, AGUARDE...."
$CP mnt/nfsroot/opt/flexlab/X11 sysroot/

mkdir -p sysroot/proc
mkdir -p sysroot/home
mkdir -p sysroot/root
mkdir -p sysroot/misc
mkdir -p sysroot/initrd
mkdir -p sysroot/lib
mkdir -p sysroot/usr
mkdir -p sysroot/bin
mkdir -p sysroot/sbin
mkdir -p sysroot/mnt
mkdir -p sysroot/mnt/nfsroot
mkdir -p sysroot/mnt/cdrom
mkdir -p sysroot/mnt/floppy
mkdir -p sysroot/tmp
mkdir -p sysroot/opt
mkdir -p sysroot/etc
mkdir -p sysroot/var
mkdir -p sysroot/dev
mkdir -p sysroot/boot
```

```
## EXECUTA CÓPIA PARA RAM
if [ -f /ramdisk ]; then
      cp -a sysroot/dev/ram1 sysroot/dev/root
      echo "########## Cópia para RAM"

else
      vmlinuz-2.6.9-1.667smp installed to /tftpboot
cp -a dev/root sysroot/dev
      echo "######### ramdisk nao encontrado"

fi

## ASSOCIA PASTAS AO NFS RESPECTIVO
mount --bind /mnt/nfsroot/lib sysroot/lib
mount --bind /mnt/nfsroot/usr sysroot/usr
mount --bind /mnt/nfsroot/bin sysroot/bin
mount --bind /mnt/nfsroot/sbin sysroot/sbin
mount --bind /mnt/nfsroot/boot sysroot/boot
mount --bind /mnt/nfsroot/opt sysroot/opt
mount --bind /mnt/nfsroot/etc sysroot/etc
mount --bind /mnt/nfsroot/var sysroot/var
mount --bind /mnt/nfsroot/home sysroot/home
mount --bind /mnt/nfsroot/dev sysroot/dev

exit 0
```

Script para montagem do sistema de arquivos virtual e inicialização do *middleware*

```
#!/bin/ash

INIT_PARAM="$#"
export INIT_PARAM

## INICIALIZA MONTAGEM VIRTUAL DO SYSTEMA DE ARQUIVOS DO MIDDLEWARE
FLEXLAB

PATH=/mnt/nfsroot/bin:/mnt/nfsroot/sbin:/mnt/nfsroot/usr/sbin:/mnt/nfs
root/usr/bin:$PATH
export PATH

LD_LIBRARY_PATH=/mnt/nfsroot/lib:/mnt/nfsroot/usr/lib
export LD_LIBRARY_PATH

## MONTA SISTEMA RAIZ DO NFS
if [ ! -d /driver_loaded ]; then
      mount -t proc /proc /proc
      mount -o remount,rw /dev/ram0 /
      ## DRIVER CARREGADO
      umount /proc
      /linuxrc
      echo > /ramdisk
fi

# VERIFICA EXISTENCIA DE PLACAS DE REDE
cont=0
etherif=
for i in 0 1 2; do
      ifconfig eth$i up > /dev/null 2>&1
      if [ "$?" = 0 ]; then
            etherif="$etherif "$i
```

```
      else
            break;
      fi
done

## CASO NÃO ENCONTRAR PLACA DE REDE
if [ -z "$etherif" ]; then
      /busybox/bin/cat
fi

## PERCORRE PLACAS DE REDE ATIVANDO O CLIENTE DHCP
for i in $etherif; do
      # RECEBE IP POR DHCP
      udhcpc -n -i eth$i -s /etc/dhcpc.bound
done

umount /proc
if [ ! -z "$rescue" ]; then
      PATH=$PATH:/busybox/bin:/busybox/sbin:/busybox/usr/bin
      export PATH
      /bin/ash
fi

## EXECUTA SCRIPT PARA MONTAR A PARTIÇÃO NFS
mountnfs
if [ ! -z "$dtlogin" ]; then
      /bin/chkstartx

fi

## SCRIPT PARA INICIALIZAÇÕES DO SISTEMA
mountrt
cpfs

## TROCA CONTEXTO DA PASTA RAIZ DO SISTEMA OPERACIONAL PARA O NFS
/bin/pivot_root
umount /initrd/mnt/nfsroot > /dev/null 2>&1

echo "INICIALIZA PARÂMETROS: $INIT_PARAM"
if [ "$INIT_PARAM" -eq "0" ]; then
      exec /sbin/init
else
      exec /sbin/init $INIT_PARAM
fi
```

Script para inicializar sistema de arquivos híbrido (PVFS2 com pNFS)

```
#!/bin/bash

## CONFIGURAÇÃO PARA ROTEAR INTERNET NOS CLIENTES
route add default gw IP

## CRIA PASTAS PARA USO DO FLEXLAB
mkdir /home/guest
mkdir /home/flexlab

## ATUALIZA RELÓGIO DOS CLIENTES ATRAVÉS DE NTP PARA DATA IGUAL A DO
#SERVIDOR
/usr/sbin/ntpdate IP
mkdir -p /instaladores
```

```
## CONFIGURA INICIALIZAÇÃO DO QEMU E KQEMU
sbin/modprobe kqemu

umount /dev/shm/
mount -t tmpfs -o size=171m none /dev/shm

mkdir /mnt/pvfs2
mkdir /mnt/nfs4

## CARREGA MÓDULO PVFS2 NOS CLIENTES
insmod /lib/modules/2.6.17-CITI_NFS4_ALL-pnfs-1/kernel/fs/pvfs2/pvfs2-
pnfs.ko
/usr/local/bin/pvfs2-layout/sbin/pvfs2-client -p /usr/local/bin/pvfs2-
layout/sbin/pvfs2-client-core

## MONTA PASTA NFS4 NOS CLIENTES
mount -t nfs4 -o "rw,async,hard,intr,rsize=4096,wsize=4096"  node251:/
/mnt/nfs4
```