ESTUDO TEÓRICO SOBRE CLUSTER LINUX

ALEXANDRE CAVALCANTI BATISTA

2007

ESTUDO TEÓRICO SOBRE CLUSTER LINUX

ALEXANDRE CAVALCANTI BATISTA

LAVRAS
MINAS GERAIS – BRASIL
2007

ALEXANDRE CAVALCANTI BATISTA

ESTUDO TEÓRICO SOBRE CLUSTER LINUX

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Dr. Heitor Augustus Xavier Costa

LAVRAS
MINAS GERAIS – BRASIL
2007

ALEXANDRE CAVALCANTI BATISTA

ESTUDO TEÓRICO SOBRE CLUSTER LINUX

Monografia de Pós-Graduação "*Lato Sensu"* apresentada ao Departamento de Ciência da computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em 22 de setembro de 2007

_____
Prof. Msc. Herlon Ayres Camargo

_____
Profª. Dr. Marluce Rodrigues Pereira

_____
Prof. Dr. Heitor Augustus Xavier Costa
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2007

Dedico este trabalho aos meus pais,
Darcy e Ilvo, aos meus irmãos
Rodrigo e André, e a todos que
me ajudaram a concluir este curso.

Agradecimentos

Agradeço especialmente aos meus pais, aos meus irmãos André e Rodrigo, ao meu amigo Rogério Soares por toda a ajuda e pelos ensinamentos, possibilitando enriquecer o aprendizado deste curso, ao orientador Heitor e aos demais professores do curso.

Resumo

Este trabalho apresenta um estudo teórico de tecnologias de *clusters* em ambiente Linux, focando em aspectos relevantes para o sucesso de um projeto de *cluster*.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Soluções de alta disponibilidade a cada dia têm se tornado mais acessíveis, populares e comuns em ambientes computacionais que buscam proteger e melhor dispor as informações para os usuários.

As soluções de *clusters* são fortemente adotadas pelas grandes organizações empresariais em seus principais sistemas. No entanto, pequenas empresas nem sempre têm os seus sistemas protegidos por alguma solução de alta disponibilidade.

Nos dias de hoje, estas soluções podem ser construídas utilizando *hardware* simples baseados em *chips* Intel. O *software* utilizado é *open source* e livre do pagamento de licença.

Em muitos projetos de alta disponibilidade, a solução fica direcionada somente na infra-estrutura do *cluster*, especialmente os servidores e a aplicação do *cluster*, negligenciando ou não dando a atenção necessária a outros aspectos relevantes de um projeto de alta disponibilidade, por exemplo, a vulnerabilidade do ambiente, as causas da indisponibilidade e a recuperação da informação.

1.1 Motivação

Atualmente, os sistemas de informação são amplamente utilizados pelas organizações. O desejável para estas organizações é estes sistemas estarem sempre disponíveis. Antigamente, a tecnologia de *clusters* era restrita às grandes organizações empresariais devido ao seu alto custo; porém, esta tecnologia tornou-se acessível às organizações de pequeno e médio porte, pois elas podem implementar soluções de alta disponibilidade e utilizar os produtos de *software*

desenvolvidos sob os termos da *General Public License* (GPL), ou seja, sem a necessidade de pagar pelo produto de *software*. Assim, tornou-se imperativo para qualquer profissional de infra-estrutura de tecnologia da informação (TI) entender, aprender e difundir o seu uso dentro das empresas onde trabalham.

1.2 Objetivo

O objetivo deste trabalho é realizar um estudo teórico sobre algumas tecnologias de *cluster* disponíveis, procurando mostrar como elas podem ajudar as empresas e as instituições de ensino e pesquisa a terem um ambiente computacional seguro, confiável, de alta disponibilidade e a baixo custo.

Como resultado final, é esperado que este trabalho sirva de consulta para qualquer profissional de TI, das informações básicas, em virtude da extensão e complexidade do assunto abordado, mas que devem ser levadas em consideração ao planejar uma arquitetura de alta disponibilidade em um ambiente empresarial.

1.3 Metodologia de Desenvolvimento

Este trabalho foi realizado mediante consultas a livros, especialmente livros eletrônicos, e consultas a documentos e páginas sobre o assunto disponíveis na internet. Neste trabalho, foram abordados os principais tipos de *clusters* disponíveis no mercado e algumas questões relativas a projetos de uma solução de alta disponibilidade, por exemplo, a vulnerabilidade de uma solução de alta disponibilidade.

1.4 Estrutura de Monografia

O capítulo 2 apresenta o conceito de *cluster*, os seus tipos e os principais benefícios do uso desta tecnologia.

O capítulo 3 apresenta a comunicação entre os nós do *cluster* que é o componente essencial para sistemas de alta disponibilidade.

O capítulo 4 apresenta o *cluster* de alta disponibilidade, sua definição, os principais usos e características e a sua arquitetura.

O capítulo 5 apresenta o conceito, as causas e as consequências da indisponibilidade e da interrupção do serviço em uma solução em *cluster*.

O capítulo 6 mostra questões relacionadas a vulnerabilidade neste tipo de solução por ser muitas vezes negligenciadas na implementação de um *cluster*.

O capítulo 7 apresenta um *cluster* de balanceamento de carga, o *cluster Linux Virtual Server,* e suas configurações.

O capítulo 8 apresenta o *cluster* de alto desempenho, o conceito e os principais usos desta tecnologia e o principal *cluster* de alto desempenho sob GPL que é o *beowulf*.

O capítulo 9 é a conclusão, apresentando uma síntese do trabalho realizado, bem como sugestão para trabalhos futuros.

2 *Cluster*

2.1 Considerações Iniciais

A necessidade por alta disponibilidade nas organizações empresariais vem crescendo a cada dia. Nas empresas, o acesso contínuo e ininterrupto à informação tornou-se vital para que estas consigam realizar negócios e atendam as necessidade de seus clientes.

Os ambientes computacionais exigem necessidades de muitos computadores para resolver as tarefas que apenas um não seria capaz de gerenciar. Os ambientes de computação envolvem o uso de servidores capazes de realizar grandes tarefas, com cada nó conectado ao outro, formando um único sistema, chamado de *cluster*. Esta tecnologia pode ser usada em uma grande variedade de aplicações que utilizam servidores de banco de dados, web, ftp e *email*, beneficiando-se desta tecnologia na proporção em que a carga de trabalho aumenta em servidores.

Há várias configurações possíveis para a implementação de um *cluster*, sendo elas escalonáveis, podendo ter o seu poder de processamento aumentado somente com a adição de novos servidores sempre que houver necessidade.

Restrito às grandes corporações, como bancos e indústrias, há duas décadas, hoje diversas empresas podem ter um ambiente de computação mais seguro para o seu negócio, a baixo custo, como os *clusters Linux Virtual Server* ou *Beowulf*. A verdade é qualquer um pode ter um *cluster*.

Este capítulo apresenta mais detalhes sobre esta tecnologia, os principais tipos de *clusters* existentes, os principais benefícios e componentes normalmente encontrados em um solução de alta disponibilidade.

2.2 História

Os *clusters* foram inventados quando surgiu a necessidade de acrescentar mais poder de processamento para a execução de uma tarefa, quando um computador não era capaz de suportar o trabalho isoladamente (WIKIPEDIAa, 2007).

Na década de 60, Eugene Amdahl, engenheiro da IBM, procurou um meio de executar tarefas em paralelo, procurando aumentar a capacidade de processamento dos mainframes, interligando-os. Ele desenvolveu diversos *mainframes* para a IBM e, mais tarde, para a companhia que fundou (WIKIPEDIAa, 2007).

Em 1967, Amdahl descreveu matematicamente a arquitetura para a computação em *cluster* e multiprocessadores. Desta forma, definiu a base da engenharia para a computação multiprocessada e para a computação em *cluster,* onde a principal diferença entre os dois tipos de computação está no suporte à comunicação entre os processos dentro do computador, que é o barramento, ou fora do computador, através de redes de computadores (WIKIPEDIAa, 2007).

Assim, a história dos primeiros *clusters* de computadores está de certo modo amarrada a história das primeiras redes de computadores, sendo que uma das motivivações iniciais para o desenvolvimento de redes foi associá-las a recursos computacionais, criando *cluster* de computadores (WIKIPEDIAa, 2007).

A troca de pacotes, conhecida como *Packet switching* , foi inventada pela empresa Rand Corporation em 1962. Utilizando o conceito de troca de pacotes de redes, o projeto Arpanet criou em 1969 a primeira rede baseada em *cluster* de computadores, ligando quatro centros de computação diferentes. O projeto Arpanet cresceu e tornou-se a internet (WIKIPEDIAa, 2007).

O desenvolvimento de *cluster* para fins comerciais e pesquisa ocorreram simultaneamente ao desenvolvimento de redes e ao desenvolvimento de sistema operacional Unix no início dos anos 70 (WIKIPEDIAa, 2007).

A partir daí, diversos fabricantes desenvolveram soluções em *cluster*, como a Dec, IBM e HP, porém suas soluções de *cluste*r eram caras e destinadas para as grandes empresas. Com o advento do software livre, vários produtos foram desenvolvidos sob os termos da GPL, entre eles *clusters*, como o *Linux Virtual Server* e o *Beowulf* , contemplando as mais diferentes necessidades para aplicações que necessitam de alta disponibilidade, a baixo custo.

Nos dias de hoje, os maiores supercomputadores em *cluster* possuem capacidade de processamento que atinge os *petaflops*. *Floating Point Operations Per Second* (FLOPS) (WIKIPEDIAb, 2007) é a abreviação da unidade utilizada para medir a performance do computador.

Aplicações que necessitam desse poder computacional costumam ser científicas, como análise de proteínas, descoberta de princípios ativos na indústria farmacêutica, estudos de meteorologia e correntes das marés, comportamento dos átomos e diversas outras aplicações.

2.3 Conceito

Um *cluster* é um grupo constituído de dois ou mais computadores que trabalham em conjunto como se fosse um único sistema, tendo como objetivo a alta disponibilidade ou a melhora no desempenho da aplicação, através do balanceamento de carga ou da distribuição de tarefas. O objetivo é ter o sistema disponível, mesmo se ocorrer falha em um dos computadores do *cluster*, também chamado de nó.

Segundo PFISTER (2006), *cluster* é um tipo de sistema paralelo ou distribuído que consiste em uma coleção inteira de computadores interconectados, usado como um único e unificado recurso computacional.

Segundo HOCHSTETLER (2004), *cluster*, de uma forma simples, é a combinação de dois ou mais computadores trabalhando juntos para prover uma solução. A idéia de *clusters* é juntar o poder computacional dos nós envolvidos para prover alta escalabilidade e a alta disponibilidade.

A figura 2.1 apresenta uma possível configuração de servidores, interconectados, formando um *cluster* com 3 nós, compartilhando entre eles a *storage*, com estações clientes acessando este *cluster* como se fossem um único servidor.

2.4 Tipos

Existem diversos tipos de *cluster*, entre eles (HOCHSTETLER, 2004):

- *Cluster* de alta disponibilidade (*HA - High Availability*);
- *Cluster* de balanceamento de carga (HS - *Horizontal Scaling);*
- *Cluster* de alto desempenho (HPC - *High Perfomace Computing*).

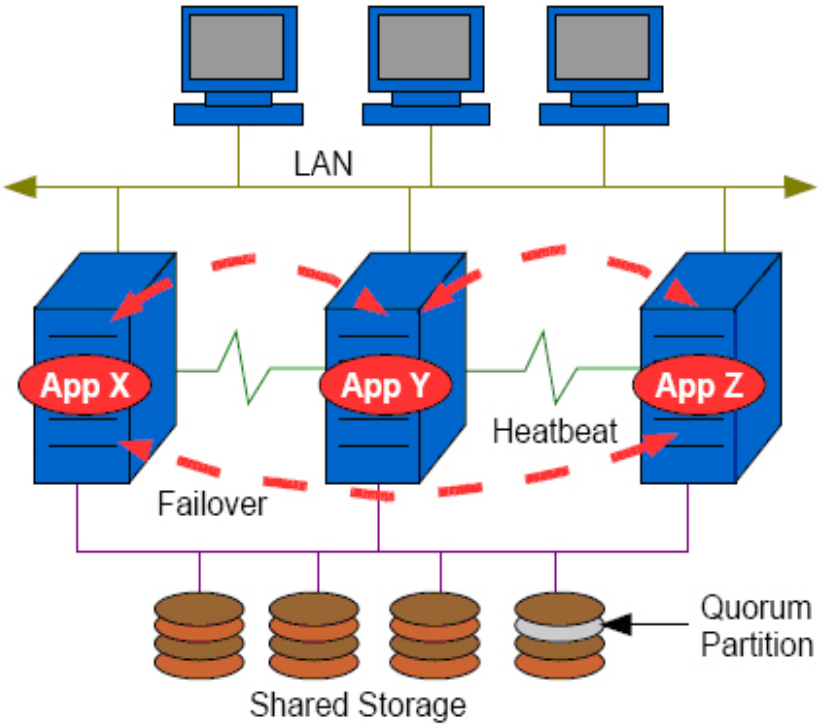

Figura 2.1: Configuração típica de um *cluster* de alta disponibilidade (RED HAT, 2003)

O *cluster* de alta disponibilidade e o de balanceamento de carga são os tipos mais adotados nas empresas. O *cluster* de alta disponibilidade tem por objetivo propiciar um ambiente seguro contra falhas, através da redundância de diversos componentes. Desta forma, o *cluster* de alta disponibilidade torna-se indispensável para empresas que precisam ter suas aplicações sempre disponíveis, como organizações financeiras e aeroportuárias. O *Red Hat Cluster Suite* é um tipo de *cluster linux* de alta disponibilidade. O *cluster* de balanceamento de carga tem por objetivo distribuir as requisições que chegam ao *cluster*. Mesmo no caso de falha em um dos nós, estas requisições são

redistribuídas entre os membros do *cluster*. Neste tipo de *cluster,* as características de *hardware* dos nós podem ser diferentes, propiciando para as empresas um melhor aproveitamento do *hardware* que possuem. O *cluster* de balanceamento de carga é muito usado em aplicações de internet, como *email*. O *Linux Virtual Server* é um tipo de *cluster* de balanceamento de carga.

O *cluster* de alto desempenho é desenhado para ser usado em computação paralela, utilizando grande poder de processamento para obter a resposta para um determinado problema (HOCHSTETLER, 2004). É adotado em laboratórios científicos, para atividades como mapeamento do genoma, estudo de nanomateriais, processamento de imagem e outras aplicações que requerem alto poder de processamento. O *cluster Beowulf* é um exemplo de *cluster* de alto desempenho.

2.5 Benefícios

Nesta seção, são apresentados alguns benefícios proporcionados pela tecnologia em estudo.

2.5.1 Redundância

A adoção de *hardware* e *software* redundantes evita que qualquer erro ou falha em um dos componentes do ambiente do *cluster* cause a perda do processamento ou da informação.

Quando ocorre uma interrupção, que pode ou não ser programada, os serviços são migrados do nó principal para o nó secundário, permitindo a aplicação continuar o processamento normal e de modo transparente para o usuário final. A seguir, são apresentados exemplos de *hardware* comuns em uma solução de *cluster* e que, usualmente, costumam ser redundantes para evitar a

ocorrência de um *single point of failure* (SPOF). S*ingle point of failure* é qualquer *software, hardware* ou componente do ambiente de alta disponibilidade que, se falhar, pode resultar na indisponibilidade da aplicação, podendo requerer intervenção humana para a correção do problema (CNT, 2003).

As figuras 2.2, 2.3 e 2.4 mostram equipamentos de *hardware* comumente utilizados na implementação de uma solução de *cluster*. Os processadores em um *cluster* permitem os processos de consulta utilizarem um determinado conjunto de processadores e outros processos *batches*, por exemplo, utilizarem outros. Normalmente, esse controle de qual processador utilizar e com qual finalidade é determinado por *software*. A *storage* além de armazenar os dados, traz segurança a solução permitindo o espelhamento das informações. Os *switches* também trazem segurança, permitindo que seja configurado um *switch* exclusivo só para o *ping* entre os nós do *cluster*.

A figura 2.2 ilustra processadores que podem trabalhar em conjunto para a realização de uma tarefa ou priorizar o processamento de uma determinada tarefa, sendo isto determinado por software. A figura 2.3 apresenta uma *storage* contendo vários discos e é um componente usual de uma solução de *cluster*. Usualmente, a *storage* é compartilhada por diversas aplicações. A figura 2.4 apresenta o *switch*, que é o equipamento responsável por entregar os pacotes de um computador da origem para outro computador de destino, reduzindo a colisão de pacotes.

As figuras 2.5 e 2.6 ilustram uma solução de *cluster*, com pelo menos dois servidores compartilhando recursos como *storage* e s*witch*, anteriormente citados.

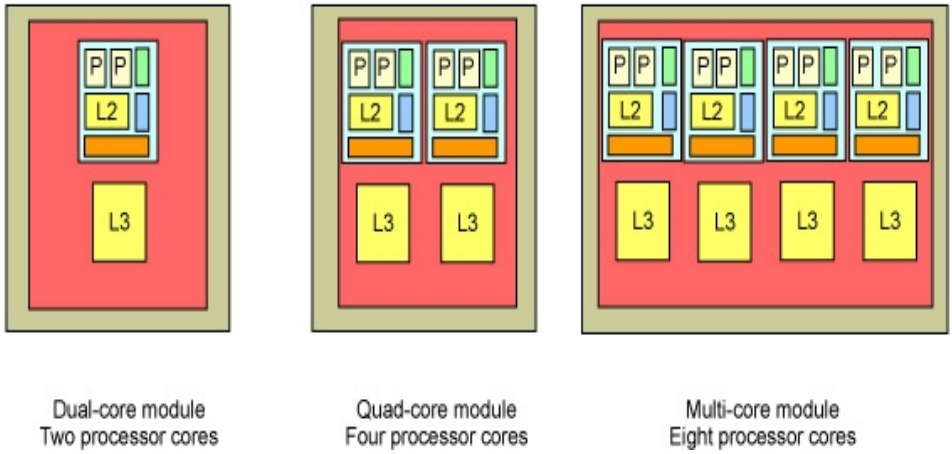

Figura 2.2: Três opções de processadores (IBMa,2003)

Figura 2.3: *Storage* compartilhada (EDIT HEAVEN, 2006)

2.5.2 Escalabilidade

É a capacidade de adicionar mais nós ao *cluster* sempre que necessário (HARRIS, 2004). Por exemplo, quando é detectado perda de desempenho ou pretende-se aumentar a segurança para a aplicação. Não há limite para o número máximo de nós que qualquer solução de *cluster* pode ter.

A figura 2.5 ilustra a escalabilidade das soluções de *clusters*. Se a aplicação necessitar de maior poder de processamento, basta adicionar mais nós aos quatro existentes.

Figura 2.4: *Switches* (CISCO, 2007)

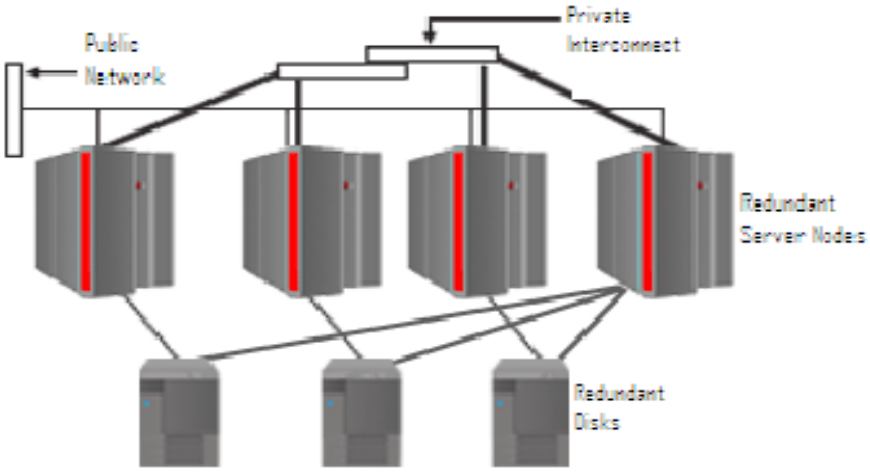

Figura 2. 5: Escalabilidade (IBMb,2003)

2.5.3 Disponibilidade

Devido ao *cluster* possuir no mínimo dois nós e ocorrendo qualquer problema, a disponibilidade dos serviços não deve ser prejudicada.

A figura 2.6 ilustra um *cluster* com quatro nós com grande capacidade quanto a disponibilidade do sistema. Se um dos nós falhar, a aplicação permanece executando nos demais nós.

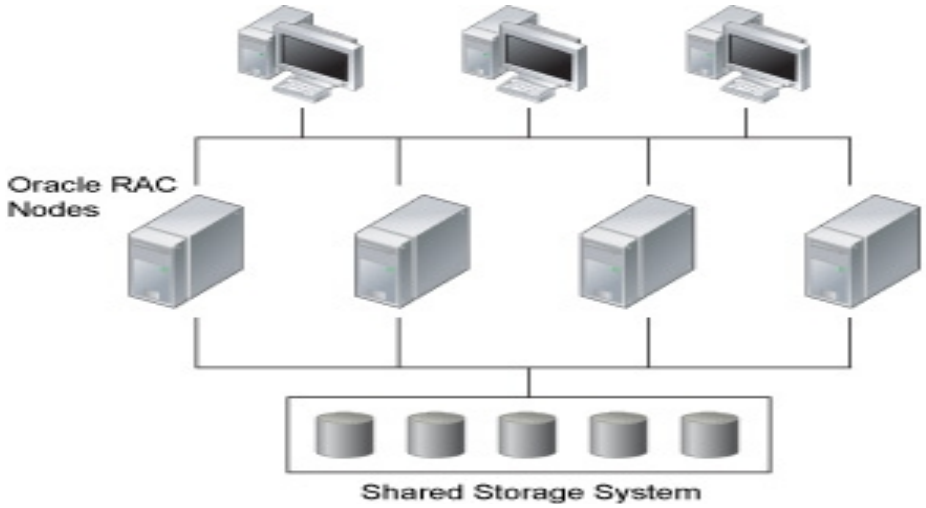

Figura 2.6: Disponibilidade do *Cluster* (ITRAINEDUCATION,2007)

2.5.4 Compartilhamento de Recursos

Ambientes de *cluster,* em geral, são gerenciados dinamicamente, visto que o ambiente computacional pode estar distribuído por espaços geográficos distintos. Em muitas soluções, *clusters* estão dispersos por diferentes regiões de um país. Normalmente, essas soluções procuram se proteger de falhas que podem prejudicar o serviço prestado, por exemplo, um *blackout* no sistema de energia elétrica de uma cidade. Adotando essa solução, a empresa pode minimizar ou evitar possíveis prejuízos.

A figura 2.7 mostra a configuração de um *cluster* com dois nós, onde cada nó está fisicamente localizado em cidades diferentes, sendo um nó principal e outro secundário, com duas redes conectando os *clusters* e utilizada por estes para melhorar a performance.

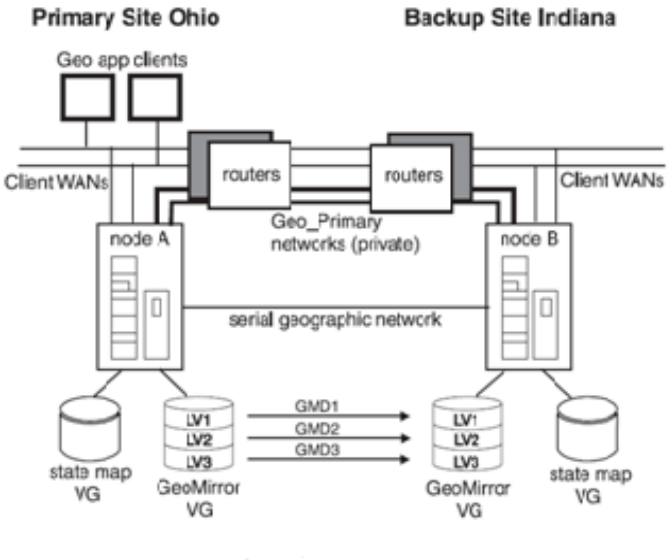

Figura 2.7: *Clusters* com nós em localização física diferente (IBMc,2003)

2.5.5 Requisitos de *Hardware*

Ao adotar uma solução de *cluster* de alta disponibilidade, um cuidado essencial para a construção de um ambiente confiável está na decisão de quanta disponibilidade e performance são desejadas para a aplicação e, a partir daí, definir os componentes que farão parte do ambiente do *cluster*.

Algumas configurações são mais dispendiosas do que outras dependendo do nível de redundância adotada.

Os requisitos mínimos para um *cluster* de alta disponibilidade são (RED HAT , 2006):

- Dois servidores para rodar os serviços do *cluster*;
- Conexão *Ethernet* para enviar *pings* do *heartbeat* para os nós da rede;
- Um *switch* ou *hub* para conectar os nós do *cluster*, bem como os serviços;
- Dispositivo para conexão;
- Discos em *Raid*.

Estes requisitos não protegem a aplicação contra os tipos de falhas, pois, se um dos servidores ou o *switch* falhar, haverá interrupção da aplicação por algum período. Porém, este *downtime* deve ser previsto no planejamento da solução e ser considerado aceitável para a aplicação.

Os requisitos de uma solução com redundância, aumentando a disponibilidade da aplicação, são (RED HAT, 2006):

- Ao menos dois servidores para rodar os serviços do *cluster*;
- Conexão *Ethernet* entre cada nó do *cluster* para enviar *pings* do *heartbeat* para os nós da rede;
- *Array* redundande *Raid* ou uma outra unidade de armazenamento (*Storage*);
- *Network power switches* para cada nó da rede, durante a migração de um nó para outro, quando ocorrer falha em um dos nós;
- Interfaces *Ethernet* configurada para o enlace dos canais;
- Ao menos duas unidades *uninterruptible power supply* (UPS) para alta disponibilidade de energia.

Exemplo de uma configuração de *cluster* sem SPOF (RED HAT, 2006):

- Dois servidores, onde cada nó possui duas interfaces de rede para o acesso pela rede do cliente e conexão no dispositivo;
- Um *switch* que habilita a conexão de múltiplos nós na rede;
- Três cabos de rede em cada nó. Dois cabos para conectar cada nó ao *switch* redundante e um cabo para conectar no dispositivo**;**
- Dois cabos *crossover* para conectar a porta serial em cada nó do *terminal server*;
- Dois *powers switches* fazendo com que cada nó tenha energia antes de reiniciar os serviços;
- *FlashDisk Raid disk array* com *dual controllers.* Dois *Raids* que protegem contra falhas em disco e no *controller*;
- Dois cabos SCSI para conectar cada adaptador no disco *Raid*.

2.6 Considerações Finais

Na década de 60, surgiu a tecnologia de *cluster* e, desde então, esta tem se aperfeiçoado, mostrando ser adequada para dispor e proteger os sistemas para os usuários finais, muitas vezes de maneira ininterrupta durante o ano. Deste modo, consegue-se atender as necessidades de negócios atuais. Com o advento do movimento do *software* livre, apareceram várias soluções de *cluster*, permitindo que essa tecnologia se disseminasse mais, alcançando empresas e instituições de pesquisa que de outra forma não teriam acesso a alta disponibilidade em decorrência do alto custo das soluções proprietárias.

A escolha do *hardware* do *cluster* determina quanta disponibilidade se pretende ter para um sistema, bem como os possíveis SPOF. Uma vez escolhido o *hardware*, é possível planejar a recuperação do sistema, caso ocorra uma falha.

Em qualquer solução em *cluster, storage* é essencial para proteger os dados e as placas ou interfaces de rede, para evitar a lentidão dos sistemas, costumam ser redundantes.

3 Heartbeat

3.1 Considerações Iniciais

Um *cluster* é constituído pelo menos por dois computadores trabalhando em conjunto. A comunicação entre os nós do *cluster* é de vital importância e constitui no aspecto fundamental para a existência do sistema de alta disponibilidade. No *cluster linux*, o programa chamado *heartbeat* é o responsável por monitorar e detectar quando ocorre falha em um dos nós do *cluster*.

O *heartbeat* é parte integrante de algumas distribuições linux, como Red Hat, Suse, Debian e Ubuntu linux. A figura 3.1 ilustra o *heartbeat* configurado em um *cluster* de alta disponibilidade.

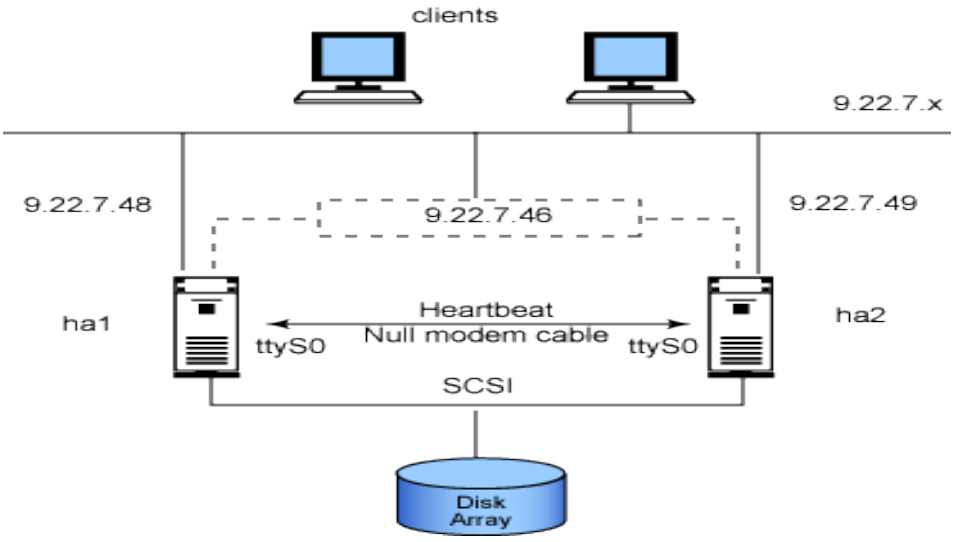

Figura 3.1: *Heartbeat* (IBMd,2003)

3.2 Objetivo

O *heartbeat* é um subsistema que monitora a presença dos nós do *cluster* através de uma série de mensagens que indicam se os membros do *cluster* estão ativos (ROBERTSON, 2000).

3.3 Funcionamento

Estas mensagens provêem o serviço de comunicação entre os nós do *cluster* e indicam que estes nós estão ativos. Estas mensagens ou pacotes se propagam para os demais membros do *cluster* e, por isso, é necessário que o *heatbeat* seja instalado e configurado de modo idêntico nos nós do *cluster*. Estes pacotes são enviados para cada nó do *cluster*, em um intervalo de tempo configurável e determinado de acordo com a criticidade e a disponibilidade desejada para o ambiente do *cluster*.

Quando a mensagem *heartbeat* não propaga aos demais nós do *cluster*, um evento responsável por avisar ao *cluster* quando um nó deixou ou retornou a ser um membro do *cluster* é disparado. Esse evento é conhecido como *cluster transiction* (ROBERTSON, 2000).

O *heartbeat* possui um subsistema, chamado de *cluster manager,* que verifica quais recursos estão em que nós, tomando a ação apropriada durante o *cluster transiction* (ROBERTSON, 2000).

3.4 Segurança

No ambiente de alta disponibilidade, o *heartbeat* é configurado em uma rede privada, onde é destinado um endereço IP virtual, que é o endereço IP do

heartbeat. Esse IP virtual é assumido por outra máquina quando ocorre falhas em um dos nós associado a um determinado serviço.

Cada servidor do *cluster* possui uma interface com diferentes redes para o caso de uma outra falhar ou uma indisponibilidade. As interfaces de rede são programadas com endereço IP virtual que pode ser movido de uma interface para outra, sendo que os servidores comunicam-se entre si através dos cabos de rede (*heartbeat*).

No ambiente de alta disponibilidade, o *heartbeat* é executado em uma rede privada dedicada utilizando endereços IPs virtuais, que é o IP do serviço *heartbeatI.* Isso evita que o *heartbeat* seja executado na mesma interface pública, pois as mensagens entre os servidores podem ser interceptadas, causando uma recuperação automática indesejada.

3.5 Considerações Finais

O *heartbeat* constitui parte essencial no funcionamento do *cluster*. O seu objetivo é reunir continuamente os servidores em uma configuração de *cluster*, para garantir que eles estejam ativos e respondendo. Além disso, ele permite determinar se o servidor principal perdeu a sua habilidade de entregar serviços (BOOKMAN, 2003). Segundo BOOKMAN (2003), o *heartbeat* é uma outra camada de alta disponibilidade que acrescenta uma maneira para assumir automaticamente, se o servidor principal falhar.

4 *Cluster* de Alta Disponibilidade

4.1 Considerações Iniciais

A demanda por alta disponibilidade vem crescendo continuamente entre as organizações empresariais de diversos portes e segmentos. Entre estes segmentos, podem ser citados:

- Bancos e companhias financeiras e de seguros;
- Empresas de serviço público;
- Empresas de comércio eletrônico;
- Supermercados e grandes cadeias de lojas;
- Indústrias de vários segmentos.

Estas organizações precisam cada vez mais de seus sistemas de informações ativos e disponíveis. Prover o acesso contínuo a informação tornou-se essencial para as empresas que conseguem, ao usar esta tecnologia, evitar o prejuízo como a perda de negócios e clientes.

Por causa desses novos requisitos para um negócio, soluções que melhoram a disponibilidade dos serviços prestados pelas organizações têm sido cada vez mais adotados.

A figura 4.1 ilustra um *cluster* de alta disponibilidade composto dos seguintes componentes: i) *sites*; ii) nós; iii) redes; iv) adaptadores de redes; v) discos externos compartilhados; e vi) estações clientes. Estes componentes ajudam as empresas a alcançarem o objetivo da alta disponibilidade.

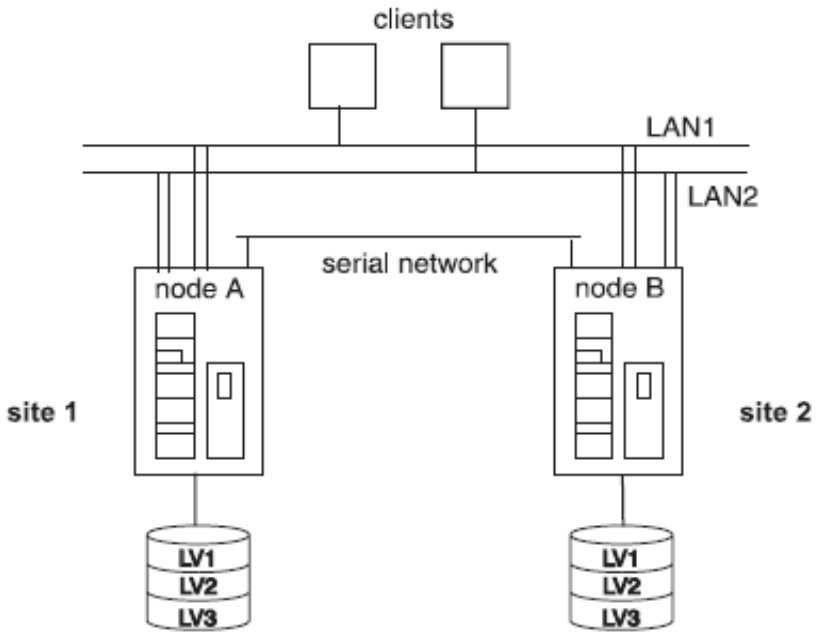

Figura 4.1 : *Cluster* de Alta Disponibilidade (IBMc, 2003)

4.2 Alta Disponibilidade

Alta disponibilidade é o tempo considerado aceitável pela organização para o correto funcionamento de seu sistema de informação. Este tempo difere de sistema para sistema e, para atender a estes diferentes requisitos, há diversas soluções, robustas, inclusive sob os termos da GPL (GUCER, 1999).

O objetivo da alta disponibilidade é manter os serviços prestados por um sistema mesmo que este venha a falhar.

Alta disponibilidade é um ambiente da aplicação que é altamente disponível se possuir a habilidade de recuperar automaticamente, dentro do

mínimo de tempo aceitável. Este tempo depende da criticidade da natureza do negócio e, geralmente, dura alguns poucos minutos (CNT, 2003).

Um *cluster* de alta disponibilidade tem como principal objetivo prover a segurança do ambiente computacional, através da redundância de cada sistema e dos recursos computacionais.

Segundo HOCHSTETLER (2004), um *cluster* de alta disponibilidade é tipicamente construído com a intenção de prover um ambiente seguro contra falhas através da redundância, ou seja, prover um ambiente computacional onde a falha de um ou mais componentes, que pode ser *hardware, software* ou rede, não seja significante para afetar a disponibilidade da aplicação em uso.

SPOF é qualquer componente de *software* e *hardware* do ambiente do *cluster* onde, na ocorrência de falha de um dos componentes, o sistema fica indisponível, precisando de recuperação automática.

Em um *cluster* de alta disponibilidade, um ponto chave é qualquer SPOF poder se recuperar no menor tempo possível, procurando fazer com que a aplicação permaneça funcional para o usuário final.

4.3 Arquitetura

O objetivo de descrever a arquitetura para *cluster* de alta disponibilidade é por ela ser uma das mais comuns encontradas nas empresas, sendo muito comum encontrar soluções híbridas, como um *cluster* de alta disponibilidade, e que possui características do *cluster* de balanceamento de carga.

4.3.1 *Cluster* Ativo-Passivo

Um *cluster* Ativo-Passivo é constituído de dois computadores, onde o processamento ocorre no *cluster* ativo e o *cluster* passivo é utilizado, caso o *cluster* ativo venha a ter algum problema.

Esta configuração apresenta uma indisponibilidade da aplicação quando há a necessidade de migração dos serviços para o nó passivo.

Neste tipo de configuração, a memória e o processador do *cluster* passivo tendem a ser aproveitados para melhorar a performance das transações no *cluster* Ativo.

A figura 4.2 ilustra um *cluster* ativo-passivo, onde a aplicação é toda executada no nó principal App A, estando o nó de *backup* pronto para assumir o serviço do nó principal, se este ficar inoperante.

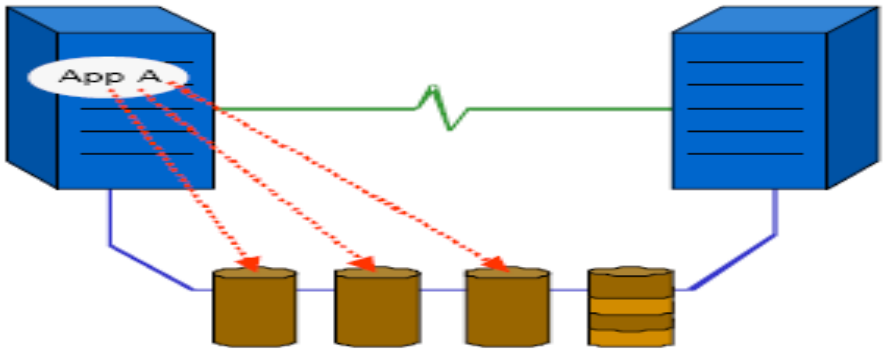

Figura 4.2: *Cluster* Ativo-Passivo (RED HAT, 2003)

4.3.2 *Cluster* Ativo-Ativo

Em um *cluster* Ativo-Ativo, o processamento da informação ocorre em ambos os nós, possibilitando a distribuição da carga das transações entre eles.

Qualquer problema que ocorra em um dos nós o outro permanece ativo, o problema de um dos nós é perceptível para os usuários finais que perdem o acesso ao nó que ficou indisponível. Há interrupção do serviço prestado até que as conexões dos usuários sejam restabelecidas no nó que permanece ativo. Pode haver perda de performance enquanto somente um dos nós estiver operacional, porque os acessos, antes distribuídos entre os nós do *cluster*, agora são realizados somente nó que permanecer operacional.

A figura 4.3 ilustra um *cluster* ativo-ativo, onde diversas aplicações são distribuídas entre ambos os servidores. Se um deles falhar, as aplicações são migradas e executadas em um dos nós que permanecer ativo.

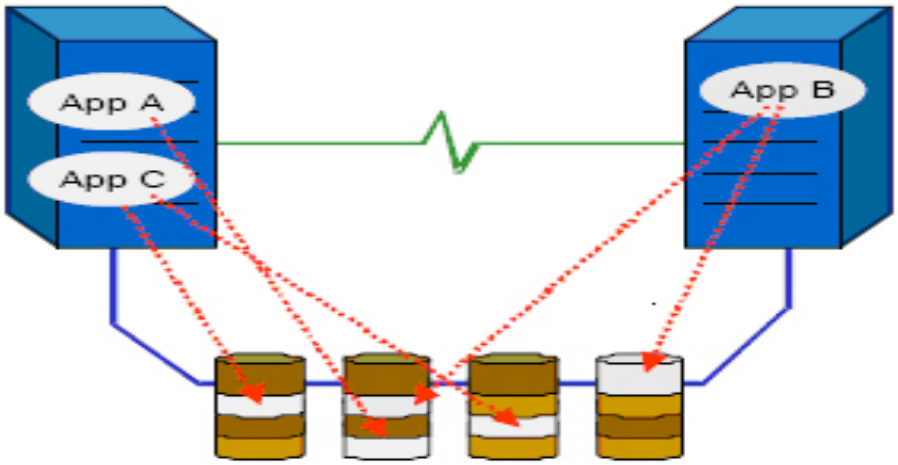

Figura 4.3: *Cluster* Ativo-Ativo (RED HAT, 2003)

4.4 Considerações Finais

Um *cluster* de alta disponibilidade pode ser entendido como uma tecnologia construída para permitir que os sistemas permaneçam acessíveis a maior parte do tempo, sendo seguro contra falhas, através da redundância dos

seus componentes (White Paper, IBM, 2000), e cuja eventual falha não seja significante para prejudicar os usuários finais.

5 Indisponibilidade do *Cluster*

5.1 Considerações Iniciais

Perder acesso a recursos vitais pode custar a uma empresa milhões de dólares em perda de lucro e resultar em diminuição de produtividade do trabalhador. Um negócio ou um recurso que confia no tempo ativo de um serviço pode ser paralisado pela perda de conectividade (BOOKMAN, 2003).

O tempo de manutenção em um ambiente de servidor importante ao negócio significa que os clientes não podem acessar seus dados, causando algum prejuízo para a organização e para os seus clientes (BOOKMAN, 2003).

Desta forma, as empresas procuram determinar os requisitos necessários para alcançar a alta disponibilidade, construindo uma ambiente computacional seguro contra falhas e procurando evitar que qualquer *downtime* possa ter um impacto negativo.

5.2 Conceito

Indisponibilidade ou *downtime* é basicamente o tempo requerido para tornar novamente operacional uma aplicação que fica indisponível. O *downtime* pode ou não ser programado (CNT, 2003). *Downtime* é qualquer interrupção de um serviço de alta disponibilidade, que pode incorrer em prejuízo à organização, onde a sua perda resulta em um inaceitável dano funcional e financeiro (COMPAQ, 1998).

5.3 Causas

Em um ambiente de alta disponibilidade, há vários fatores que podem causar o *downtime* da aplicação. O ambiente de alta disponibilidade é formado pelo conjunto de *software*, *hardware,* pelas pessoas que operam o sistema e pelo ambiente físico onde estão os nós do *cluster*.

O *hardware* pode falhar, mesmo que seja confiável. *Software* pode apresentar *bugs* que necessitam ser corrigidos ou, se não apresentam problemas, precisam ter suas versões atualizadas. Isso pode requerer manutenção ou *downtime* programado.

Além disso, por mais críticos que os sistemas possam ser, os profissionais que os administram podem cometer erros a qualquer momento, durante o funcionamento normal da aplicação ou em uma atividade de manutenção programada, levando, neste caso, um tempo maior do que o esperado para a restauração do ambiente.

O *downtime* pode ser causado por outra área da empresa, por exemplo pela área responsável pela manutenção da energia elétrica. A central de processamento de dados pode precisar de manutenção urgente nas instalações elétricas ou de ar condicionado e cuja responsabilidade é da área de manutenção que, muitas vezes, não tem consciência do quão desastroso pode ser para os negócios, se ocorrer, por exemplo, uma falha elétrica que venha a causar a indisponibilidade da aplicação. Pode ser também uma falha humana, quando inadvertidamente o técnico desliga a tomada do *cluster*.

Há ainda problemas externos, que estão fora do alcance das companhias como a interrupção do fornecimento de energia causado por um *blackout* ou por

outros eventos relacionados a ocorrências de fenômenos da natureza, por exemplo enchentes, terremotos e furacões, que causam *downtime* nas companhias.

Um exemplo claro desse último tipo de *downtime* ocorreu no Vale da Eletrônica em Santa Rita do Sapucaí – MG – onde algumas empresas tiveram o seu funcionamento interrompido devido a inundação de suas instalações causadas por enchentes.

A tabela 5.1 ilustra problemas problemas de *downtime* na Infraero – Empresa de Infraestrutura Aeroportuária – onde a ocorrência de falhas de energia não programadas e por motivos de manutenção acarretam a indisponibilidade de diversos sistemas que possuem alta disponibilidade, prejudicando o usuário final. Na aba *motivo*, é descrito o problema que originou a falha de um sistema. Na aba *sistema*, são apresentados os sistemas afetados. A aba *dependência* indica as dependências onde ocorreu a falta de energia e as abas *inicial, final e indisponibilidade* indicam a hora de início e de fim e o tempo total que o sistema ficou indisponível.

Na tabela 5.1, pode ser visto na linha em amarelo, que por motivos de falta de energia nas dependências do AIBR (Aeroporto Internacional de Brasília), o sistema Tecaplus, sistema do terminal de cargas, ficou indisponível no dia 31 de agosto por um período de 59 minutos, impedindo que qualquer carga pudesse ser retirada pelos clientes.

5.4 Custo

Por custo de *downtime,* entende-se como sendo as conseqüências ou os prejuízos que uma empresa pode ter em decorrência da interrupção da aplicação

bem como os custos relacionados para evitar ou minimizar o *downtime* (CNT, 2003). Quando ocorre o *downtime*, dados podem ser perdidos, transações deixam de ser realizadas, vendas deixam de ser concretizadas e projetos têm seus prazos reavaliados, sem contar a perda de horas de trabalhos dos funcionários. Essas interrupções nos sistemas podem custar às companhias desde alguns reais até milhões de reais em prejuízo, além do prejuízo da imagem.

Tabela 5.1: Indicadores de indisponibilidade do AIBR – Período: 08/2007

Dia	Área	Motivo	Sistema	Dependência	Inicial	Final	Indispon.
1	Infra-Estrutura	Teste de Sistema de Energia - Emergencia	Notes	Servidores	2:35:00	7:55:00	5:20:00
1	Infra-Estrutura	Teste de Sistema de Energia - Emergencia		ST3,	3:48:00	3:55:00	0:07:00
1	Infra-Estrutura	Falta de Energia	Tecaplus	Servidores	3:59:00	4:15:00	0:16:00
1	Infra-Estrutura	Teste de Sistema de Energia - Emergencia		ST2, ST3	4:39:00	4:45:00	0:06:00
2	Infra-Estrutura	Teste de Sistema de Energia - Emergencia		ST6	3:12:00	3:16:00	0:04:00
2	Infra-Estrutura	Teste de Sistema de Energia - Emergencia		ST2	3:32:00	3:36:00	0:04:00
9	Infra-Estrutura	Problemas com o nobreak		ST6	10:18:00	10:33:00	0:15:00
25	Infra-Estrutura	Manutenção do roteador	Sgtai	SEDE	1:00:00	3:00:00	2:00:00
27	Infra-Estrutura	Manutenção do roteador	Internet	SEDE	10:02:00	10:42:00	0:40:00
29	Infra-Estrutura	Manutenção do roteador	Internet	SEDE	1:00:00	1:40:00	0:40:00
29	Servidores	Servidor de Storage	Tecaplus,Ge	Servidores	3:25:00	14:30:00	11:05:00
31	Infra-Estrutura	Manutenção elétrica NB4	Notes	NB4	2:20:00	6:40:00	4:20:00
31	Infra-Estrutura	Falta de energia	Bdo	AIBR	3:15:00	4:17:00	0:27:00
31	**Servidores**	**Falta de Energia**	**Tecaplus**	**Servidores**	**2:26:00**	**3:25:00**	**0:59:00**
31	Servidores	pela MABR. Danificou Firewall mas não	Siv,Sra,Bdo	Servidores	2:20:00	3:25:00	1:05:00

Uma vez que o custo do *downtime* máximo de uma aplicação tenha sido definido, é possível avaliar e definir qual a tecnologia de alta disponibilidade que será adotada pela empresa. Desta forma, o custo da alta disponibilidade representa o montante financeiro do ambiente do *cluster*. Os custos estão relacionados a aquisição dos componentes de *software* e *hardware* e dependem da solução escolhida pela empresa.

Há desde tecnologias que prevêem alguma indisponibilidade a outras que possibilitam a não ocorrência de SPOF. Qualquer configuração definida, o custo maior será de *hardware* (servidores, placas de rede, *hubs, storage*). Quanto

ao *software*, as soluções implementadas utilizando *software* sob a licença GPL não envolvem custos, sendo possível, nos dias de hoje, construir um *cluster* utilizando *software* livre.

A tabela 5.2 apresenta um estudo do Gartner Group (1998) mostrando o custo por hora que o *downtime* pode causar em uma variedade de indústrias.

Tabela 5.2: Estudo dos custos do *downtime* (Fonte: Compaq (1998))

Industry	Application	Average Cost per Hour Downtime
Financial	Brokerage Operations	$6,500,000
Financial	Credit Card Sales	$2,600,000
Media	Pay-per-View	$1,150,000
Retail	Home Shopping (TV)	$ 113,000
Retail	Catalog Sales	$ 90,000
Transportation	Airline Reservations	$ 89,500

5.5 Recuperação de Informação

No planejamento de uma solução de alta disponibilidade, é necessário determinar o impacto que a indisponibilidade da aplicação pode ter e estabelecer possíveis cenários de recuperação da informação. Muitas vezes, as empresas precisam manter as informações de suas aplicações por uma período determinado por lei e, na falta desta, se for requerida, a empresa pode vir a sofrer alguma penalidade.

Por exemplo, negócios construídos com o objetivo de ter 100% de disponibilidade da aplicação, qualquer que seja o desastre do ambiente, este não deve ser motivo para haver qualquer interrupção nos serviços prestados pela solução e a recuperação deve ser transparente para o usuário final. Espera-se, neste caso, que o ambiente tenha redundância total, física e de aplicação, não causando prejuízo para a organização e para os clientes. Por haver necessidade

de *sites* redundantes, normalmente uma solução como esta é encontrada somente em grandes empresas.

Um outro cenário, comum para a recuperação da informação, é a ocorrência de uma falha onde se faz necessária a recuperação dos dados em qualquer servidor da empresa. Esse servidor pode ter configuração inferior e ainda estar compartilhando recursos com outra aplicação, podendo ocorrer perda de performance para a aplicação recuperada e para a aplicação anteriormente existente.

Em *sites* de *e-commerce*, uma transação de compra ou venda pode deixar de ser realizada, mesmo que a aplicação esteja acessível para o cliente, caso apresente, por exemplo, lentidão, comprometendo a conclusão da transação.

5.6 Teoria dos Nove

Uma vez definida a solução de alta disponibilidade, é conhecida quanta disponibilidade será implementada. Estas exigências de tempo ativo são referidas como a teoria dos nove e essa disponibilidade em TI é usualmente medida em porcentagem, ou seja, em quantos noves está previsto para a solução de alta disponibilidade (BOOKMAN, 2003). Esta teoria é uma forma comum de expressar em relatórios e documentar em instruções de trabalho nos ambientes de TI das empresas, a probabilidade de *downtime* que um sistema pode ter e procura determinar os requerimentos de *hardware* e *software* necessários para garantir a alta disponibilidade de uma determinada aplicação.

Desta forma, ter cinco noves de disponibilidade é quase impossível sem algum tipo de redundância total, na forma de uma solução de alta

disponibilidade (BOOKMAN, 2003). Também, outra forma muito comum de representar essa teoria é medir a disponibilidade em termos de horas x dias x período que a aplicação deve ficar ativa. Por exemplo, um sistema 24 x 7 x 365 é considerado um sistema de disponibilidade máxima, possuindo redundância total, onde não é aceitável mais do que 5,25 minutos de indisponibilidade por ano.

A tabela 5.3 apresenta a relação entre a disponibilidade desejada pela solução de alta disponibilidade adotada e o tempo de indisponibilidade considerado aceitável que esta pode apresentar durante o período de um ano em minutos, ano e por semana.

Tabela 5.3: Os nove de disponibilidade (Fonte: Bookman (2003))

Tempo ativo em noves	Tempo de manutenção (%)	Tempo de manutenção por ano	Tempo de manutenção por semana
Um (98%)	2	7.3 dias	3 h e 22 min
Dois (99%)	1	3.65 dias	1 h e 41 min
Três (99,9%)	0.1	8 h e 45 min	10 min e 5 s
Quatro (99,99%)	0.01	52.5 min	1 min
Cinco (99,999%)	0.0001	5.25 min	6 s

5.7 Considerações Finais

Na solução de *cluster*, é possível que este fique indisponível por um determinado período considerado aceitável e dentro do planejado para a solução que foi construída, baseada na teroria dos nove, não importando o motivo que levou a indisponibilidade do serviço prestado. Isto quer dizer que a indisponibilidade maior do que planejada significa que houve falha ao conceber o *cluster*.

6 Vulnerabilidade no Ambiente de Alta Disponibilidade

6.1 Considerações Iniciais

Em um ambiente de alta disponibilidade, após a solução ter sido escolhida, é necessário verificar se há alguma vulnerabilidade no conjunto do ambiente do *cluster*.

6.2 Testes do Ambiente

É importante que o projeto do *cluster* seja testado antes dele se tornar operacional. Os testes no ambiente do *cluster* agregam confiabilidade ao ambiente, reduzindo custos e trabalho.

Os testes servem para verificar a funcionalidade de cada equipamento envolvido na solução, tais como: *switches, storage*, cabos de fibra ótica, servidores e outros. Eles visam também verificar o ambiente do *cluster*, quanto ao desempenho, à segurança e à recuperação, no caso de ocorrer qualquer problema que gere indisponibilidade.

Se qualquer um dos testes resultar em uma resposta inadequada com o projeto do *cluster*, o objetivo da alta disponibilidade estará em risco, havendo possibilidade de falhar, quando o *cluster* se tornar operacional.

Portanto, testar o ambiente do *cluster* é a última fase de um projeto de de alta disponibilidade, possibilitando que, ao entrar em produção, a solução esteja funcional e sem problemas imprevistos.

6.3 Documentação

Documentação costumava ser o ponto fraco em muitas organizações, sendo comum não possuírem documentos dos processos relativos a administração, a manutenção e a recuperação dos seus sistemas.

Nos dias de hoje, as empresas procuram adotar modelos de gestão, que permitam-nas certificar seus sistemas de gestão de qualidade através de organismos internacionais de certificação.

A série ISO 9000, *International Organization for Standardization,* (WIKIPEDIAc, 2007) é um conjunto de normas que formam um modelo de gestão de qualidade em ambientes de produção. A organização deve atender alguns requisitos da ISO 9001 para serem certificadas, dentre eles:

- Padronização dos processos chaves do negócio, processos que afetam o produto e, conseqüentemente, o cliente;
- Implementar e manter os registros adequados e necessários para garantir a rastreabilidade do processo;
- Revisão sistemática dos processos e do sistema de qualidade para garantir a sua eficácia.

Toda empresa que possui a certificação internacional de seus sistemas ou empresas que se preocupam com a documentação dos seus processos conseguem transmitir as informações relacionadas aos ambiente de produção com facilidade, por exemplo para os novos funcionários, permitindo que estes possam atuar com precisão em qualquer situação desde a sua admissão.

6.4 Treinamento

Total Cost of Ownership (TCO) é uma ferramenta, criada pelo Gartner Group, para medir direta ou indiretamente os custos relacionados com os serviços, permitindo analisar e gerenciar um amplo alcance do custo e do risco relacionado com os negócios dos ambientes computacionais (GARTNER, 2007). O investimento em *hardware* e *software* é somente parte do processo da implementação e dos custos de uma solução de alta disponibilidade.

Ao implementar soluções de alta disponibilidade, as empresas preocupam-se somente com a implementação da solução em si, muitas vezes negligenciando o fator humano responsável pela administração do *cluster* e sem treinamento adequado para a correta administração e monitoração do sistema. Segundo Joseph Feiman, vice-presidente de pesquisa da Gartner Group, muitas empresas consideram o treinamento uma questão secundária ou sentem que não é produtivo afastar as pessoas do trabalho por qualquer período de tempo (MICROSOFT,2007).

Uma das consequências que a falta de treinamento pode acarretar é demora para a identificação e solução de problemas, sendo considerado um SPOF de uma solução de alta disponibilidade. O retorno do investimento em treinamento na equipe responsável por manter a infra-estrutura do *cluster* funcionando aumenta a produtividade, a disponibilidade e a credibilidade de um ambiente de missão crítica.

Por isso, uma solução de alta disponibilidade deve contemplar o treinamento dos profissionais responsáveis pela administração e manutenção do ambiente computacional.

6.5 Considerações Finais

A vulnerabilidade de um *cluster* muitas vezes não está no *hardware* ou no *software,* mas na ausência de procedimentos documentados, por exemplo, de testes de recuperação. Ou ainda, na falta de capacitação dos profissionais responsáveis pela administração e pela manutenção do ambiente de alta disponibilidade.

7 *Cluster* de Balanceamento de Carga

7.1 Considerações Iniciais

Balanceamento de carga é uma outra arquitetura de *cluster*, que possui a capacidade de distribuir as requisições dos processos ou conexões entre os membros do *cluster*, reduzindo o tempo de processamento e melhorando o desempenho da aplicação. Em computação, balanceamento de carga é a técnica que distribui o trabalho entre muitos computadores, processos, discos e outros recursos com o objetivo de conseguir uma ótima utilização dos recursos e reduzir o tempo de computação (HARRIS, 2004).

Cluster de balanceamento de carga é, geralmente, uma solução híbrida porque possui características do *cluster* de alta disponibilidade, como a possibilidade de configuração dos nós primário e secundário para o *master load balance,* e as características próprias do *cluster* de balanceamento de carga, que melhora o desempenho e a escalabilidade dos serviços (BOOKMAN, 2003).

A carga do processamento é compartilhada entre os computadores que fazem parte do *cluster* e a quantidade de carga que cada servidor recebe depende dos recursos disponíveis. Nestes *clusters*, os servidores não precisam ser iguais, nem possuir a mesma configuração. Isso os tornam mais flexíveis e mais baratos que outras soluções de *cluster*, sendo especialmente adotado em aplicações web, de e*mail*, ftp, Voip e comércio eletrônico.

Devido a essa flexibilidade, os serviços destinados aos nós podem ser diferentes. Enquanto um ou mais nós podem ser destinados a requisições de processos *batch*, os outros nós podem ser configurados para receber requisições somente de consulta e de curta duração. Além disso, pode-se ter o

processamento distribuído uniformemente entre os membros do *cluster*. Um dos *clusters* de balanceamento de carga mais utilizados em sistemas *open souce* é o *Linux Virtual Server*.

A figura 7.1 apresenta uma configuração de um *cluster* de balanceamento de carga com três servidores e, o balanceador de cargas primário e secundário, que distribui as requisições para os demais servidores do *cluster*.

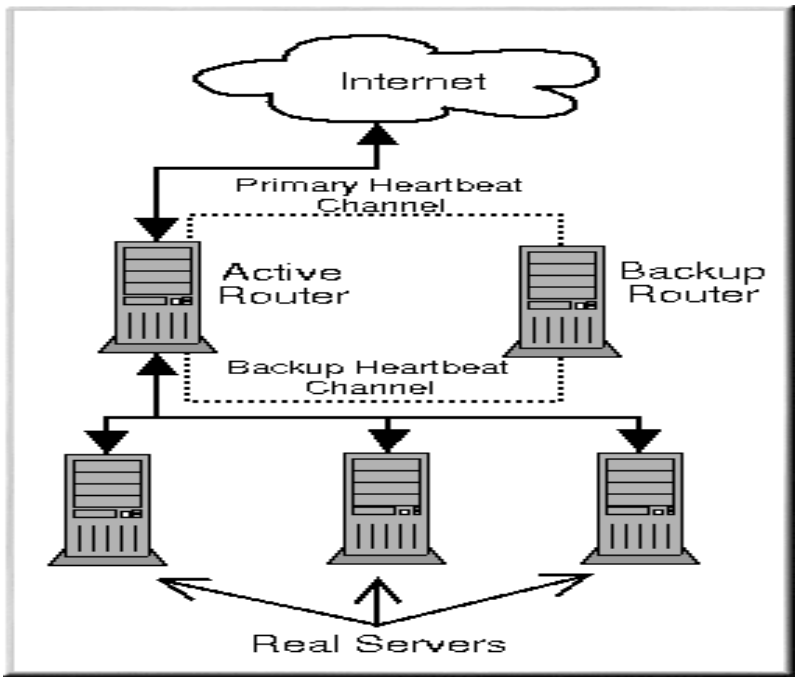

Figura 7.1: *Cluster* de Balanceamento de Carga (RED HAT,2007)

7.2 *Linux Virtual Server*

Linux Virtual Server (LVS) é uma solução de *cluster* de balanceamento de carga, cujo objetivo é construir um servidor capaz de fornecer disponibilidade, escalabilidade e desempenho para as aplicações, sendo esta

arquitetura completamente transparente para os usuários (HOCHSTETLER, 2004). O desenvolvimento do LVS foi motivado pelo grande e rápido crescimento das aplicações web, que resultou em um aumento no tráfego da rede, gerando problemas de sobrecarga em *sites* muito acessados.

O LVS é um grupo de servidores, chamado de servidores reais, que age como um único computador, sendo composto por dois tipos de computadores: o balanceador de cargas e os servidores reais. Esse conjunto é chamado de servidor virtual. O LVS é responsável por realizar o balanceamento de carga das requisições entre os servidores reais que são os servidores destinados a fazer o processamento das requisições. Ele faz a verificação da integridade do *cluster*, checando se os nós estão ativos e quando ocorre falha em um dos servidores reais, o balanceador de carga redistribui as conexões entre os demais servidores do *cluster* (BOOKMAN, 2003).

A principal característica do LVS é a capacidade de melhorar o desempenho através do balanceamento de carga entre os servidores e melhorar a disponibilidade através da redundância dos servidores reais. Por ser uma arquitetura flexível, os servidores podem ser adicionados e removidos automaticamente, permitindo que manutenções possam ser realizadas nos servidores sem que haja necessidade de interromper os serviços prestados para os usuários finais (DOLBIER, 2003).

Outra característica do LVS é ser considerado uma solução de baixo custo, porque o *software* utilizado é livre e o *hardware* existente pode ser aproveitado como mais um servidor real, aceitando máquinas com diferentes configurações e performance, proporcionando mais redundância e escalabilidade para a aplicação.

O balanceamento de carga é feito através de alguns algoritmos estáticos ou dinâmicos. A decisão de qual algoritmo de balanceamento a ser utilizado é feito durante a instalação do *cluster*. O algoritmo escolhido define a forma como serão distribuídas as conexões entre os nós e pode ser modificado a qualquer momento. A diferença de configuração e de capacidade dos servidores que compõe o *cluster* e do serviço prestado pelo *cluster* pesarão na definição de qual será o algoritmo utilizado.

A figura 7.2 mostra um *cluster* com três servidores reais sem redundância do balanceador de cargas. As requisições chegam aos servidores reais pelo servidor de balanceamento de carga, que distribuirá as requisições para os servidores reais.

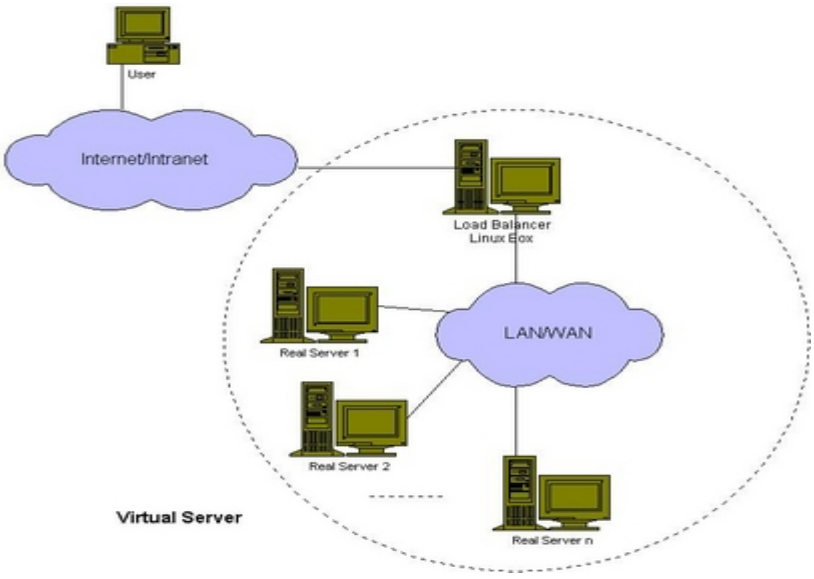

Figura 7.2 : LVS com três servidores reais (HARRIS,2004)

A figura 7.3 mostra alguns servidores reais, onde os dois servidores de balanceamento de carga, o primário e o secundário, compartilham um

dispositivo de *storage*. Se o servidor primário falhar, o secundário assume a tarefa de distribuir as requisições para os servidores reais.

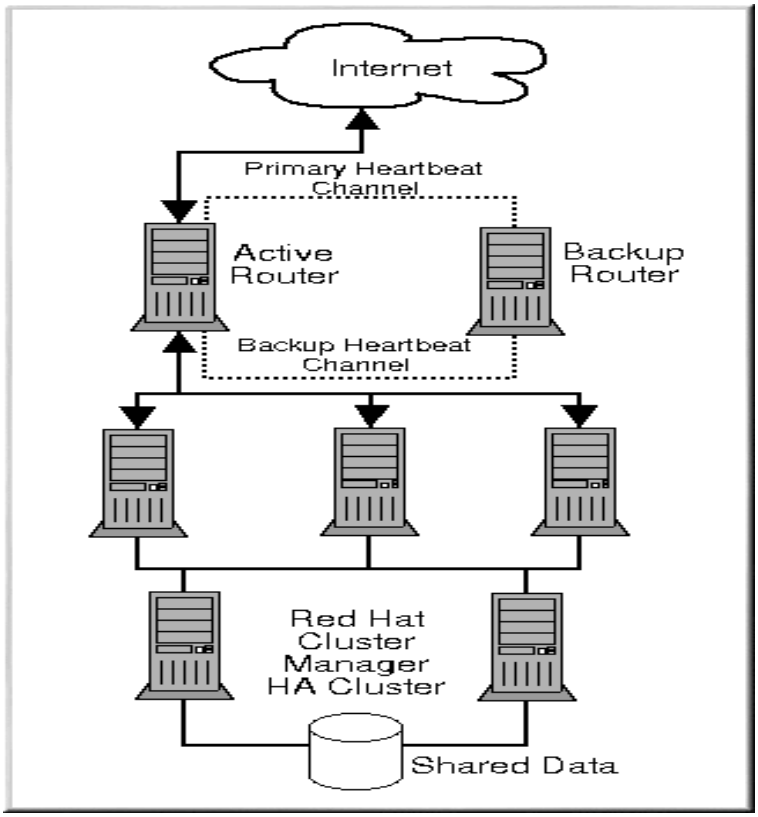

Figura 7.3 : LVS compartilhando uma *storage* (RED HAT,2007)

7.3 Algoritmos de Balanceamento de Carga

Atualmente, existem oito algoritmos de balanceamento de carga, dos quais são citados os mais comumente utilizados, tornando mais compreensível o funcionamento deste tipo de *cluster*.

7.3.1 Algoritmos Estáticos

7.3.1.1 *Round Robin* (RR)

É considerado um algoritmo estático, porque as requisições são distribuídas uniformemente entre os membros do *cluster*, eles recebem um tempo para cada processo, em partes iguais, e possuem a mesma prioridade. Por isso, é considerado o mais simples dentre os existentes.

A unidade de tempo neste algoritmo de escalonamento é denominada de *quantum*. No *round robin*, os processos são armazenados em um fila circular. O escalonador da *central processing unit* (CPU) percorre a fila, alocando a CPU para cada processo durante um *quantum*, e retira o primeiro processo da fila procedendo a execução. Se o processo não termina a execução no *quantum* determinado, o processo é inserido no final da fila e se termina antes de um *quantum*, a CPU é liberada para a execução de novos processos. É indicado para soluções onde os servidores reais têm capacidades iguais (OSDEV, 2006).

A figura 7.4 apresenta o servidor de balanceamento de carga, distribuindo uniformemente as requisições que chegam dos clientes para os servidores reais.

7.3.1.2 *Weighted Round-Robin* (WRR)

É uma variação do algoritmo r*ound robin*, porém determina um peso para cada servidor. As requisições são distribuídas sequencialmente entre os servidores, direcionando maior número de conexões para o servidor de maior peso. É a escolha indicada quando há diferença de capacidades entre os servidores reais (WIKIPEDIAd, 2007). Ao atribuir um peso para um determinado servidor, este receberá um número maior de conexões que outro servidor. Na figura 7.5, o servidor RIP2 possui mais recursos computacionais

que os demais e, por isso, foi-lhe atribuído peso três, que é maior que o peso atribuído aos outros servidores e, assim, o servidor RIP2 aceitará um número maior de conexões.

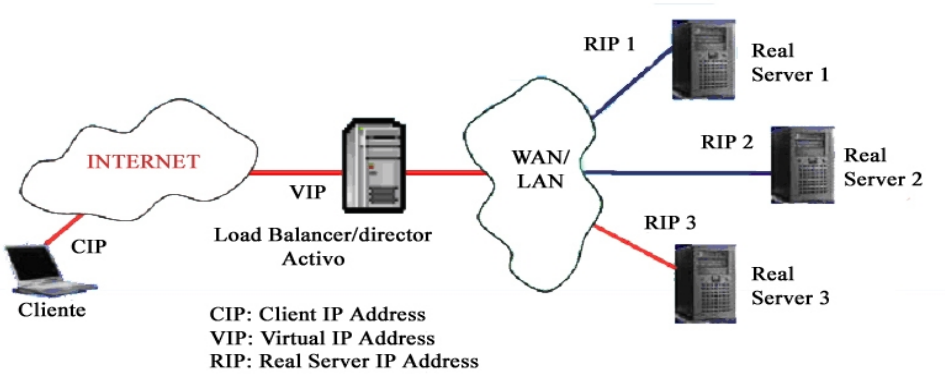

Figura 7.4 : Algoritmo *Round Robin* (BOLINA,2005)

A figura 7.5 apresenta o servidor de balanceamento de carga, que distribui as requisições para os servidores, respeitando um peso atribuído para cada servidor. O servidor de maior peso recebe o maior número de conexões.

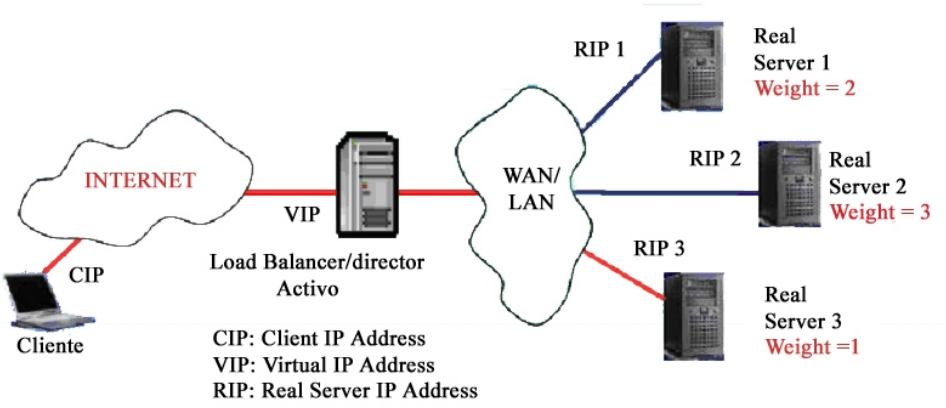

Figura 7.5 : Algoritmo *Weighted Round Robin* (BOLINA,2005)

7.3.2 Algoritmos Dinâmicos

7.3.2.1 *Least Connection* (LC)

Este algoritmo trabalha contando as conexões ativas de cada servidor e distribui as requisições que chegam para o servidor com o menor número de conexões ativas. Ele é indicado para a solução de *cluster* que possui capacidade similar, pois fará a distribuição equilibrada da carga quando houver grandes variações desta (WIKIPEDIAe, 2006).

A figura 7.6 mostra o algoritmo de LC, onde as conexões são direcionadas para o servidor real que apresentar o menor número de conexões ativas.

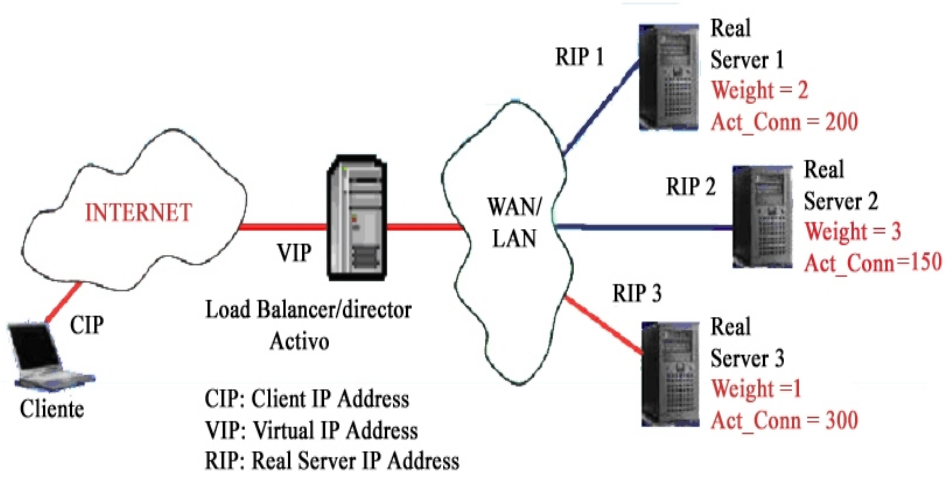

Figura 7.6 : Algoritmo *Least Connection* (BOLINA,2005)

7.3.2.2 *Weighted Least Connection* (WLC)

É uma variação do algoritmo *least connection,* combinando as vantagens deste algoritmo. As requisições são distribuídas para o servidor que possui o maior peso e com menor número de conexões. Este servidor recebe o maior número de conexões e o algoritmo é indicado para *clusters* cujos servidores possuem capacidades diferentes (WIKIPEDIAf, 2007).

A diferença entre os algoritmos WRR e WLC é o primeiro atribui um peso para cada servidor e direciona as conexões para o servidor com o maior peso, enquanto o segundo é um algoritmo baseado no número de conexões, mas sensível ao peso atribuído ao servidor e direciona as conexões para o servidor que possuir o menor número de conexões abertas e com maior peso.

A figura 7.7 apresenta uma combinação entre os algoritmos WRR e LC, onde as conexões são direcionadas para os servidores de maior peso e com menor número de conexões.

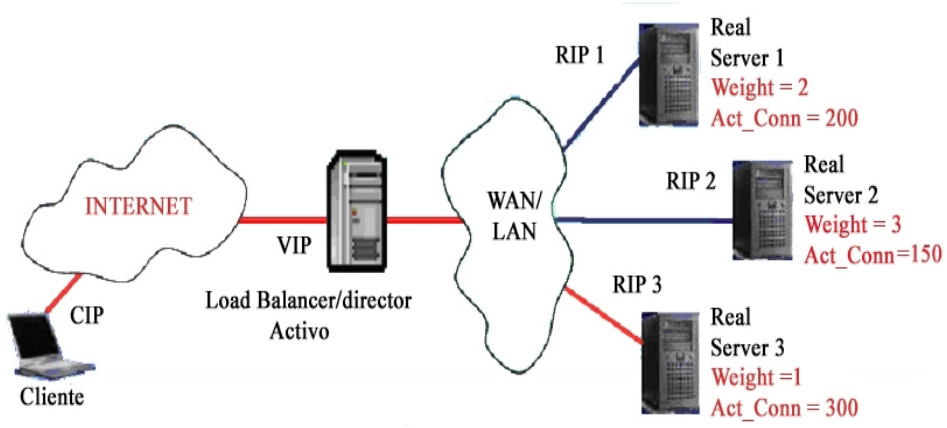

Figura 7.7 : Algoritmo *Weighted Least Connection* (BOLINA,2005)

7.4 Configurações de um Servidor LVS

Uma das configurações de um *cluster* LVS, que busca alcançar as vantagens citadas anteriormente, é possuir uma arquitetura em três camadas e que é a mais recomendada (DOLBIER, 2003):

- *Load Balancer*: representa o servidor virtual e é o responsável por distribuir as requisições dos clientes para os servidores reais;
- *Server Cluster*: são os servidores reais, repositórios dos serviços que prestam, como web, *email* e outros;
- *Shared Storage*: podem ser os bancos de dados que provêem a garantia de acesso concorrente aos dados de um determinado serviço.

As conexões usadas entre o load balancer, server clusters e shared storage ocorrem através de redes de alta velocidade, placas de rede ethernet, preferencialmente gigabit ethernet, procurando evitar que a falta de um componente de hardware adequado para uma solução de cluster venha a ser causa de gargalo ou lentidão da aplicação.

7.5 Considerações Finais

Dentre as arquiteturas de *cluster* existentes, a de balanceamento de carga é sem dúvida a mais barata e acessível, pois não é pré-requisito que os componentes do *cluster* sejam iguais. Este *cluster* permite melhor aproveitamento dos recursos computacionais que a empresa possui, pois os computadores podem ser adicionados como mais um servidor do *cluster*.

8 *Cluster* de Alto Desempenho

8.1 Considerações Iniciais

Clusters de alto desempenho são construídos com o objetivo de melhorar o desempenho da aplicação pelo aumento de poder do processamento e dividir as tarefas em segmentos pequenos entre os nós do *cluster* (FERREIRA, 2001).

Clusters de alto desempenho são desenhados para usar a computação paralela, aplicando maior poder do processamento para solucionar um problema. Além disso, *clusters* de alto desempenho tenta obter o melhor poder computacional que um único computador pode prover. Assim, *clusters* de alto desempenho é um ramo da ciência da computação com o foco no desenvolvimento de supercomputadores, algoritmos de processamento paralelo e *software* relacionados (FERREIRA, 2001).

Este tipo de arquitetura está relacionado especialmente com aplicações científicas, industriais, de engenharia e, mais recentemente, com aplicações voltadas para negócios com banco de dados que precisam recuperar e analisar grandes quantidades de dados. Entre as aplicações que fazem uso desta tecnologia, estão: i) previsão do tempo; ii) estudos geológicos para a indústria do petróleo; iii) processamento e renderização de imagens; e iv) estudos científicos em geral.

Aplicações que podem rodar em um *clusters* de alto desempenho foram escritas para obter o melhor proveito da computação paralela. Neste tipo de *cluster*, para cada um dos nós, pode-se determinar a execução de uma tarefa.

Nesse *cluster*, as tarefas são executadas a partir de um nó, que distribui o processamento entre os demais nós do *cluster*.

Há uma tarefa que inicia a execução em um nó do *cluster*. Enquanto ocorre o processamento, os nós ficam comunicando-se entre si. Esta tarefa é divida em pequenos pedaços com os outros nós. Por exemplo, a tarefa do nó A está realizando um processamento, cujo resultado será utilizado por uma outra tarefa do nó B. Por sua vez, a tarefa do nó B está aguardando o resultado do término da execução da tarefa do nó A para que possa prosseguir com a sua execução. Ou seja, os resultados intermediários de um nó afetam resultados de tarefas futuras de um outro nó. Quando os nós finalizam a execução de sua tarefa, os resultados da tarefa do processo dividido são remontados e a informação é disponibilizada como um resultado único de processamento.

Um *cluster* de alto desempenho implementa a comunicação entre os nós de duas maneiras: i) *Parallel Virtual Machine* (PVM); e ii) *Messaging Program Interface* (MPI). Essas são as chamadas de categorias de programação para sistemas paralelos que compartilham recursos de memória (HOCHSTETLER, 2004).

Estas bibliotecas são um conjunto de rotinas usadas para facilitar a comunicação entre os processos e um padrão de mensagem que possibilita aos programadores uma interface padronizada para escrever códigos portáveis entre plataformas heterogêneas. O código é destinado a capacitar o programador a acessar os nós do *cluster* para solucionar e prover o correto processamento para a solução de problemas computacionais.

A passagem da mensagem é feita através da comunicação entre os processos e enviadas e recebidas através de processadores, utilizando pacotes TCP/IP. Cada processador conversa entre si através de programas MPI ou PVM.

A biblioteca PVM surgiu primeiro que a biblioteca MPI. Entre as características da PVM, ela oferece utilização efetiva da computação paralela. Na PVM, o controle de processos é maior e permite iniciar, parar e controlar os processos em tempo de execução. Na MPI, somente é possível o controle processos através de grupos de tarefas. Na PVM, o controle de recursos é dinâmico, na MPI este controle é estático.

A biblioteca MPI é mais utilizada que a PVM devido a algumas características que a torna mais atraente aos programadores, tais como a definição de uma interface que acrescenta extensões a biblioteca PVM, tornando-a mais fácil de ser utilizada pelos programadores, e a possibilidade de maior escalabilidade, permitindo uma aplicação criar subgrupos de processos que permitem operações de comunicação coletiva para melhorar o alcance dos processos.

Pode-se observar que, embora existam diferenças entre estes modelos de passagem de mensagem, a tendência é características da PVM sejam incorporadas a MPI e vice-versa, procurando facilitar o desenvolvimento de aplicações.

8.2 *Cluster Beowulf*

O *cluster Beowulf* é um dos mais populares projetos de *clusters* de alto desempenho. Ele foi o modelo desenvolvido para a computação de alto desempenho, com baixo custo de *hardware* e como uma alternativa ao alto custo

para implementar um *clusters* de alto desempenho utilizando os supercomputadores (FERREIRA, 2001).

O *Beowulf* foi construído utilizando o sistema operacional linux e outros programas de código aberto como as bibliotecas MPI e PVM, ferramentas de gerenciamento sob os termos da GPL, e *hardware* simples, como os computadores pessoais capazes de executarem o linux e que pode ser adquirido a um custo menor para esse tipo de solução (HOCHSTETLER, 2004).

O desenvolvimento desse *cluster* possibilitou que empresas, instituições de pesquisa e universidades passassem a ter as vantagens da computação paralela de alto desempenho a um preço acessível.

A configuração de um *cluster Beowulf* consiste de um nó denominado *server* e também chamado de *head* ou *master*. Este nó *server* possui monitor, teclado e é responsável pela conexão dos demais nós membros do *clusters*, denominados de *clients*. Os *clients* não possuem teclados e monitores e o acesso ao *server* ocorre via conexão remota. A comunicação entre o *server* e os *clients* ocorre através de uma rede *ethernet* ou outro dispositivo de conexão como o *switch*.

As tarefas do *cluster Beowulf* são iniciadas no nó *server* e a sua execução é distribuída em paralelo entre os demais membros do *cluster* para os *clients*, reduzindo o tempo de processamento da tarefa. Os *clients* são controlados pelo nó *server* e, ao final da execução da tarefa, é retornado o resultado acumulado do processamento, como ocorre nos *clusters* de alto desempenho.

Nesta configuração, pode-se adicionar novos computadores sempre que necessário e, normalmente, os nós do *cluster* são configurados de forma idêntica, com mesma quantidade de memória, rede e discos. No entanto, não é pré-requisito para o seu funcionamento que os nós sejam iguais, pelo contrário, a adição de novos nós com diferentes configurações pode ser feita, permitindo um melhor aproveitamento dos recursos computacionais disponíveis em um ambiente heterogêneo, além de proporcionar economia.

A figura 8.1 ilustra um *cluster Beowulf*, onde o nó *master* ou controlador distribui as conexões para os nós clientes, onde ocorre o processamento das requisições. Uma vez que este processamento termina nos nós clientes, o resultado retorna para o nó *master*.

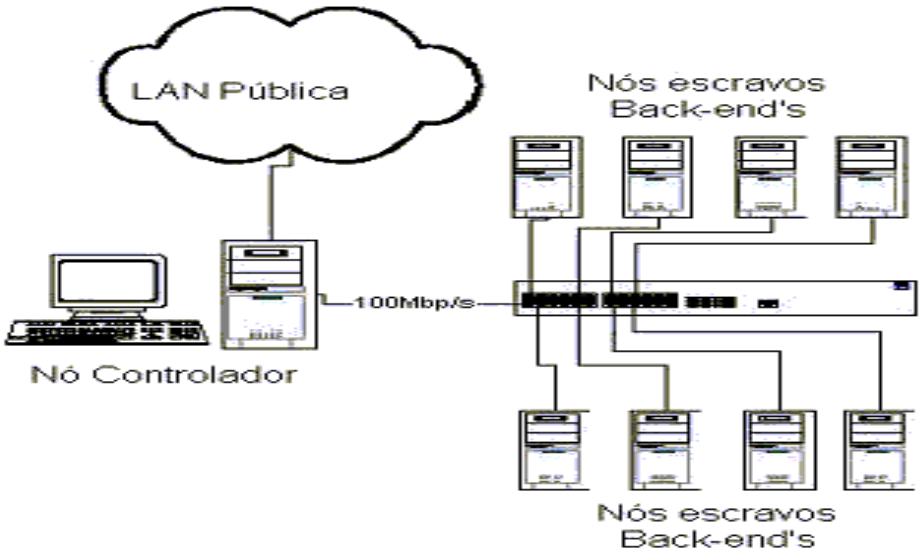

Figura 8.1: *Cluster Beowulf* (PIRES,2004)

A figura 8.2 apresenta o ambiente de um *cluster Beowulf*, onde somente o nó *master* possui teclado e monitor e os nós clientes são os servidores para a execução das tarefas e não possuem teclado ou monitor.

Figura 8.2 : Ambiente de um *cluster Beowulf* (FREITAS, 2003)

Pelo seu custo inferior a soluções proprietárias e aproveitamento dos recursos de *hardware* disponíveis e por *Beowulf* ser construído com *software* livre, sem contar em fatores como confiabilidade e escalabilidade, torna este *cluster* uma opção para processamento que requer computação de alto desempenho.

8.3 Considerações Finais

Clusters de alto desempenho são conhecidos pela utilização em aplicações científicas que precisam de grande poder de processamento. Esta arquitetura trabalha dividindo o processamento de uma grande tarefa em pequenas tarefas distribuídas entre os nós do *cluster*, retornando um resultado ao final do processamento. Justamente por esta característica, *cluster* de alto

desempenho deixa de ser uma tecnologia voltada somente para instituições de pesquisas e torna cada vez mais adotado por empresas em aplicações, por exemplo, de *data mining*.

9 Considerações Finais

9.1 Conclusões

Nos dias de hoje, é comum na maioria das empresas ter seus sistemas usufruindo de alguma forma de contingência que possibilite manter os serviços prestados funcionando, mesmo na ocorrência de uma falha.

Qualquer solução de *cluster* prevê algum tipo de disponibilidade e esta precisa ser meticulosamente planejada nos aspectos que envolve um projeto dessa natureza, para que de fato a prestação contínua e ininterrupta dos serviços possa ser alcançada.

Isto significa escolher adequadamente o ambiente físico, que deve atender as normas de segurança para instalações de CPD, como fiação protegida, garantia de temperatura adequada para a sala dos servidores, além do *hardware* e *software* mais adequado para o sistema em *cluster*.

No entanto, deve-se levar em consideração que somente um ambiente físico adequado pode não ser suficiente para manter uma solução de *cluster* funcionando, sendo importante o treinamento dos profissionais responsáveis por administrar o ambiente.

Para garantir a alta disponibilidade, é importante que políticas de *backup* e *recovery* tenham sido planejadas, documentas e testadas antes do sistema entrar em produção.

9.2 Contribuições

A importância deste trabalho está em salientar aspectos da escolha e da implementação de uma solução de alta disponibilidade que normalmente são negligenciadas ou postas em segundo plano pela equipe de TI e que vem a constituir um ponto de falha do planejamento.

Neste trabalho, foi apresentado o início da tecnologia de *cluster*, que surgiu há mais de 40 anos, mostrando os conceitos sobre *cluster*, os objetivos para o qual foram desenvolvidos, a importância desta tecnologia para as organizações, que propiciou confiabilidade ao ambiente de negócios, tanto para as empresas que ofertam serviços, quanto para os clientes que muitas vezes não conseguem imaginar a complexa infra-estrutura que é criada para permitir que mais e mais serviços possam estar disponíveis 24 horas por dia.

Por isso, além das tecnologias comuns em uma solução de *clusters*, foram abordados aspectos relacionados a vulnerabilidade de um ambiente de alta disponibilidade, as causas e os custos envolvidos quando ocorre um *downtime*.

Foram abordados, os principais requisitos de *hardware* para construir um *cluster* desde uma solução simples até outra solução onde não há SPOF. Além disso, foram apresentados o conceito e o uso do *heartbeat,* explicitando a importância da comunicação entre os *clusters,* e foram descritos os principais tipos de *clusters* existentes sob os termos da GPL.

O movimento do *software* livre possui soluções de *clusters* bastante heterogêneas e consideradas confiáveis e estáveis para cada tipo de *cluster*. Em *clusters* de alta disponibilidade, por exemplo, tem-se o *Red Hat Cluster Suíte.* Para soluções de balanceamento de carga, o *Linux Virtual Server* e, para *clusters*

de alto desempenho, tem-se o *Beowulf*. Portanto, esta tecnologia deixou de ser privilégio de grandes corporações, ficando acessível a todos.

Os diferentes tipos de *clusters* podem ser utilizados para uma variedade de aplicações além das citadas neste trabalho, tais como: Servidores web, servidores de *email*, *Firewalls*, servidores de arquivos, de banco de dados, de DNS (*Domain Name System*), de DHCP (*Dynamic Host Configuration Protocol*) e *Proxy*.

9.3 Trabalhos Futuros

Realizar este trabalho permitiu saber quão abrangente pode ser o assunto e, portanto, como sugestão para trabalhos futuros, pode-se aprofundar mais no estudo de um dos aspectos descritos neste trabalho. Acredita-se que será útil para profissionais de TI que não trabalham diretamente ou que não têm muita familiaridade com o assunto alta disponibilidade.

Dentre estes, pode-se sugerir a realização de um estudo sobre um determinado tipo de *cluster,* um trabalho detalhando o *heartbeat* ou um trabalho sobre os algoritmos descritos no capítulo 7 e, ainda, pode-se realizar um trabalho relacionando somente os aspectos da vulnerabilidade de um ambiente de alta disponibilidade. Outro trabalho interessante pode ser a implementação de um *cluster* descrito neste trabalho e a elaboração de um manual de instalação e configuração do *cluster* escolhido.

Referências Bibliográficas

(BOLINA,2005) BOLINA, M. *Algoritmos de balanceamento de carga*. Disponível em: <http://www.cafelug.org.ar/eventos/sept-03/slides/linux-ha-html/linux-ha.pdf>. Acessado em 16/02/2007.

(BOOKMAN, 2003) BOOKMAN, Charles. *Agrupamento de Computadores em Linux.* Editora Ciência Moderna, 2003.

(CISCO, 2007) CISCO . *Switches*. Disponível em: <http://pccompsoft.com/Cisco%20Switches.jpg>. Acessado em 20/03/2007.

(CNT, 2003) CNT, Computer Network Technology Corporation. *Achieving high availability objectives*. *White paper,* 2003. Disponível em: <http://www. sniaeurope.org/Document.nsf/0AF0A/$FILE/HighAvailablityObjectivespl581.pdf>. Acessado em 18/02/2007.

(COMPAQ, 1998) COMPAQ, COMPAQ COMPUTER CORPORATION. *Architecting and Deploying High-Availability Solutions.* White Paper prepared by CustomSystems Enterprise High Availability Segment. Disponível em: <ftp://ftp.compaq.com/pub/supportinformation/papers/ecg0641198.pdf>. Acessado em 30/01/2007.

(DOLBIER, 2003) DOLBIER, G;BOGDANOVIC, P.; CIMAFRANCA, D; JOHNG, Y; CREDLE JR, R. IBM *Bladecenter, Linux, and Open Source.* IBM Redbooks,*2003.* Disponível em: <http://www.redbooks.ibm.com/redbooks/pdfs/sg247034.pdf>. Acessado em 18/02/2007.

(EDIT HEAVEN, 2006) EDIT HEAVEN. Storage compartilhada. Disponível em: <http://www.editheaven.co.uk/Editshare/editshare.html>. Acessado em 16/02/2007.

(FERREIRA, 2001) FERREIRA,L; KETTMANN, G; THOMASH , A; SILCOCKS, E; CHEN, J; DAUNOIS,JC; ILHAMO, J; HARADA, M; HILL, S; BERNOCCHI, W; FORD, E. *Linux HPC Cluster Installation.* IBM Redbooks, 2001. Disponível em: <http://www.redbooks.ibm.com/redbooks/pdfs/sg246041.pdf>. Acessado em 30/01/2007.

(FREITAS, 2003) FREITAS, AE; VAZ, TB. CLUSTER BEOWULF. *Ambiente de um cluster Beowulf.* Disponível em: <http://twiki.im.ufba.br/pub/Main/TiagoVaz/apresentacao_beowulf_1.0.pdf>. Acessado em 01/03/2007.

(GARTNER, 2007) GARTNER GROUP. *Decision Cost Of Management,* Gartner, *2007.*Disponível em: <http://www.gartner.com/4_decision_tools/measurement/decision_tools/tco/tco.html>. Acessado em 25/08/2007.

(GUCER, 1999) GUCER V; JACOB,B; PIMPLODKAR, N; SPURWAY, D; MITZEL, J; SHIELS D. *High Availability Scenarios for Tivoli* Software. IBM Redbooks *1999.* Disponível em: <http://www.redbooks.ibm.com/redbooks/SG242032.html> . Acessado em 15/02/2007.

(HARRIS, 2004) ARMINGAUD F ; BERLARDI, M; HUNT, M;LIMA, M; MALCHISKY JR, W;RUIBAL, JR;TAYLOR, J.*Linux Handbook.* IBM,2004. Disponível em: <http://www.redbooks.ibm.com/redbooks/pdfs/sg24 7000.pdf>. Acessado em 26/03/2007.

(HOCHSTETLER, 2004) HOCHSTETLER, S; BERINGER, B. *Linux Clustering With CSM and GPFS,* IBM Redbooks, 2004. Disponível em: <http://www.redbooks.ibm.com/abstracts/sg246601.html>. Acessado em 22/08/2007.

(IBMa,2003) IBM. Três opções de processadores. Disponível em: <http://www-128.ibm.com/developerworks/linux/library/l-pow-devoverview/>. Acessado em 18/02/2007.

(IBMb,2003) IBM. Escalabilidade. Disponível em: <http://www-128.ibm.com/developerworks/db2/library/techarticle/adamache/images/Sun_Cluster.gif>. Acessado em 11/03/2007.

(IBMc, 2003) IBM. *High Availability Geographic Cluster for AIX. Concepts and Facilities.* Disponível em: <http://publib.boulder.ibm.com/epubs/pdf/am8hc240.pdf>. Acessado em 20/02/2007.

(IBMd, 2003) IBM. *Heartbeat*. Disponível em: <http://www-128.ibm.com/developerworks/linux/library/l-halinux/figure1.gif>. Acessado em 22/02/2007.

(ITRAINEDUCATION,2007) *Cluster de alta disponibilidade* . Disponível em: <http://www.itraineducation.co.uk/images/racnews.jpg>. Acessado em 28/09/2007.

(MICROSOFT, 2007) MICROSOFT. *¨6 Common IT Security Mistakes and How to Avoid* Then, Microsoft, *2007.*Disponível em: <http://www.microsoft.com/midsizebusiness/security/securitymistakes.mspx>. Acessado em 25/08/2007.

(OSDEV,2006) OSDEV COMMUNITY. Disponível em: <http://www.osdcom.info/content/view/29/39/>
PFISTER, Greg. *Linux cluster white papers*, 2006. Disponível em: <http://www-03.ibm.com/servers/eserver/clusters/whitepapers/linux_wp.pdf>.
Acessado em 06/03/2007.

(PIRES,2004) PIRES, A. *Cluster & Grid Computing.* Disponível em: <http://www.dei.unicap.br/~almir/seminarios/2004.2/ns06/cluster_grid/indice_arquivos/beowulf.htm>. Acessado em 20/03/2007.

(RED HAT, 2003) RED HAT. *Delivering High Availability Solutions with Red Hat.* Disponível em:<http://www.redhat.com/whitepapers/rha/RHA_ClusterSuiteWPPDF.pdf >. Acessado em: 20/02/2007.

(RED HAT, 2006) RED HAT CLUSTER SUITE. *Hardware Installation and Operation System Configuration.* Red Hat, 2006. Disponível em: <http://www.redhat.com/docs/manuals/csgfs/browse/rh-cs-en/ch-hardware.html>. Acessado em 12/03/2007.

(RED HAT, 2007) RED *HAT. Cluster de Balanceamento de Carga.* Disponível em: <http://www.redhat.com/docs/manuals/csgfs/browse/rh-cs-en-3/figs/ha-lvs-overview/typical-lvs.png>. Acessado em 04/10/2007.

(ROBERTSON, 2000) ROBERTSON, A; *Linux-HA Heartbeat System Design.* Usenix Association, *2000.* Disponível em: <http://delivery.acm.org/10.1145/1270000/1268399/p20-robertson.pdf?key1=1268399&key2=3948087811&coll=GUIDE&dl=,&CFID=15151515&CFTOKEN=6184618>. Acessado em 03/03/2007.

(WIKIPEDIAa, 2007) WIKIPEDIA. *Computer cluster*. Disponível em: <http://en.wikipedia.org/wiki/Computer_cluster>. Acessado em 25/08/2007.

(WIKIPEDIAb, 2007) WIKIPEDIA. *Flops*. Disponível em: <http://en.wikipedia.org/wiki/FLOPS>. Acessado em 27/03/2007.

(WIKIPEDIAc, 2007) WIKIPEDIA. *ISO 9000* . Disponível em: <http://pt.wikipedia.org/wiki/ISO_9000>. Acessado em 26/08/2007.

(WIKIPEDIAd, 2007) WIKIPEDIA. *Weighted round robin*. Disponível em: <http://en.wikipedia.org/wiki/Weighted_round_robin>, Acessado em: 24/08/2007.

(WIKIPEDIAe, 2006) WIKIPEDIA. *Least connection.* Disponível em: <http://kb.linuxvirtualserver.org/wiki/Least-Connection_Scheduling>. Acessado em 24/08/2007.

(WIKIPEDIAf, 2007) WIKIPEDIA. *Weighted least connection*. Disponível em:<http://kb.linuxvirtualserver.org/wiki/Weighted_Least-Connection_Scheduling>. Acessado em 24/08/2007.