Revista Científica On-line
Tecnologia – Gestão – Humanismo
ISSN: 2238-5819

SISTEMA DE ALTA DISPONIBILIDADE EM BANCO DE DADOS MYSQL UTILIZANDO LINUX LVS

HIGH AVAILABILITY SYSTEM IN MYSQL DATABASE USING LINUX LVS

SISTEMA DE ALTA DISPONIBILIDAD EN LA BASE DE DATOS MYSQL USANDO LINUX LVS

Gustavo Cesar Bruschi[1] (gustavo.bruschi@fatec.sp.gov.br)
Jossany Serra[2] (jossany2@hotmail.com)
Luis Alexandre da Silva[3] (luis.silva51@fatec.sp.gov.br)
Pedro Ostti[4] (pedrostti@gmail.com)
Rafael Offerni[5] (rafa.l.o@gmail.com)

[1,2,3,4,5] FATEC Bauru

Resumo
Este trabalho visa oferecer uma solução de baixo custo no armazenamento de informações em um sistema tolerante a falhas por meio da utilização de um cluster computacional. Viável às pequenas empresas devido a seu baixo custo de implantação, este cluster oferece alta disponibilidade no funcionamento de um banco de dados em função das configurações que permitem monitorar possíveis falhas em nós secundários, continuando seu serviço de forma transparente ao usuário. Conclui-se que esta estrutura pode ser construída com hardwares *low-end* e softwares *open-source*, sendo uma opção viável para empresas se manterem competitivas e eficientes na área de gerenciamento de informações com baixo investimento.
Palavras-chave: Banco de Dados, Disponibilidade, Linux, LVS, MySQL.

Abstract
This paper aims to provide a low cost solution to store information in a fault-tolerant system using a computer cluster. It is addressed to small businesses due to its low cost implementation, this cluster provides high availability in the Database operation due to configurations that allow real time monitoring of possible faults in secondary nodes, continuing its services in a transparent way to the user. It was concluded that this structure can be built using simple, low-end hardware and open-source software, minimizing its costs and becoming a viable option for companies that aims to remain competitive and efficient in data management area with low cost implementation.
Keywords: Database, Availability, Linux, LVS, MySQL.

Resumen
En este trabajo se propone ofrecer una solución de bajo costo para el almacenamiento de información en un sistema tolerante a fallos usando un clúster computacional. Una opción viable para las pequeñas empresas debido a su bajo costo de implementación, este clúster ofrece alta disponibilidad en la operación de una base de datos en función de los ajustes que le permiten supervisar posibles fallos en los nodos secundarios, sin requiere acción del usuario. Fue posible concluir que esta estructura puede ser construida con un hardware de baja gama (low-end) y un software de código abierto, siendo una opción viable para que las empresas sigan siendo competitivas y eficientes en el ámbito de la gestión de la información con una baja inversión.
Palabras clave: Base de Datos, Disponibilidad, Linux, LVS, MySQL.

ISSN: 2238-5819

Introdução

Atualmente a informação é reconhecida como parte do patrimônio de uma empresa, e contando com isso, gerenciar os dados dessas informações de forma a melhor aproveitá-los requer a utilização de equipamentos e sistemas computacionais eficientes. A falha em atender esses requisitos pode representar a perda de competitividade em um mercado cada vez mais definido pelas informações provenientes de produtos, clientes e tendências.

Buscar um ambiente contendo um Sistema Gerenciador de Banco de Dados (SGBD) que gerencie todo o volume de informações pode ser uma boa solução para empresas que necessitem manter os serviços computacionais contínuos. Se associado a um ambiente que possibilite o gerenciamento por *softwares* controladores com mecanismos de suporte a falhas, pode representar para muitas empresas a continuidade do negócio, garantindo o faturamento e o aumento dos recursos financeiros.

Com a crescente expansão tecnológica, diversas áreas de atuação do conhecimento humano passam a agregar sistemas computacionais para lhes permitir maior eficiência em suas funções. Segundo Pereira Filho (2004), essa disseminação de redes e *hardwares* a custos acessíveis faz com que até mesmo tarefas críticas passem a adotar sistemas que são inevitavelmente vulneráveis a falhas. Pereira Filho afirma ser essencial a utilização de técnicas que garantam a disponibilidade dos serviços em casos de falhas que podem acarretar prejuízos materiais, financeiros e, em alguns casos, até mesmo a perda de vidas humanas. Por outro lado, Piedad e Hawkins (2001) ressaltam que buscar disponibilidade em tempo integral aumenta exponencialmente o custo de implementação desses sistemas, o que pode ser impraticável para algumas organizações, uma vez que esse custo será repassado direta ou indiretamente ao empresário. É importante encontrar o equilíbrio e garantir alta disponibilidade para os sistemas realmente essenciais.

Importante considerar que a economia dos recursos financeiros e a eficácia da implantação de um serviço com alta disponibilidade dependerá da escolha do ambiente criado para essa tarefa. Optar ainda pela estrutura em *Cluster* com máquinas de uso comum demonstra que essa arquitetura pode ser implantada por qualquer empresa que não possa despender recursos para a aquisição dos caros equipamentos específicos para esta tarefa.

Possibilitar uma solução de baixo custo para o gerenciamento de informações permite que empresas de pequeno porte possam atualizar-se e manterem-se competitivas no mercado através da implementação de métodos tecnológicos anteriormente acessíveis apenas mediante investimentos pesados em toda sua infraestrutura de *hardware*.

Este trabalho tem por objetivo demonstrar a criação de um servidor de banco de dados de alta disponibilidade, ou seja, com sistemas que suportem possíveis falhas, utilizando *softwares* sem grande valor de investimentos e que façam o gerenciamento das informações, mantendo os serviços em estado contínuo.

ISSN: 2238-5819

Na seção 1 são descritos conceitos importantes para entendimento dos sistemas utilizados no experimento. Cada etapa do experimento, os equipamentos utilizados e os métodos empregados, bem como as adaptações necessárias para finalização do projeto, foram destacadas na seção 2. Os resultados alcançados são explanados na seção 3 e a análise final é apresentada nas Considerações Finais.

1 Conceitos e Definições

1.1 Banco de Dados Relacional e o MySQL

O termo modelo relacional foi idealizado por Edgar Frank Codd e refere-se ao modo como os dados são armazenados e representados ao usuário, estruturados através de tabelas. Segundo Codd (1970), é um conceito abstrato que define que o usuário deve ser protegido da necessidade de saber a exata forma de organização do armazenamento de dados, por não se tratar de uma solução prática.

Os bancos de dados relacionais baseiam-se no princípio de que os dados são armazenados em tabelas relacionadas entre si e que podem rapidamente atribuir relações múltiplas entre os dados. São utilizadas tabelas, linhas e colunas para definir a posição aparente de um dado para o usuário.

Em um banco de dados relacional, a facilidade de acesso aos dados possibilita uma grande variedade de abordagens no tratamento das informações, em oposição ao modelo hierárquico, no qual é necessário que o usuário defina as questões de maneira específica devido a natureza da ligação de seus registros. A linguagem padrão dos bancos de dados relacionais é a Linguagem de Consulta Estruturada, do inglês *Structured Query Language* (SQL), conforme demonstrado na Figura 1.

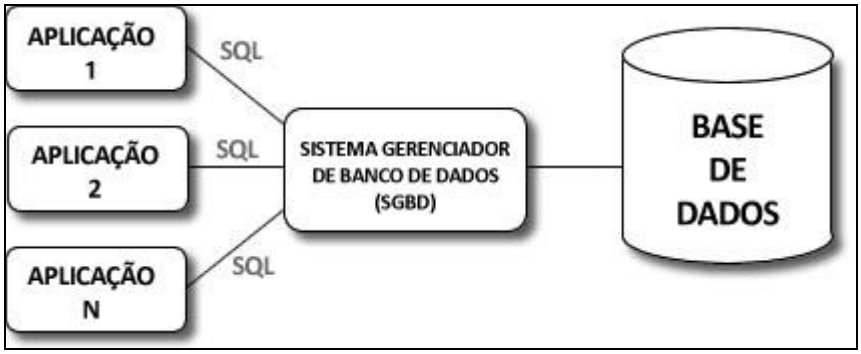

Figura1 - Representação de um Sistema de Banco de Dados
Fonte: Elaborado pelos Autores

Segundo o *site* DB-ENGINES (2014), MySQL é o sistema gerenciador de banco de dados relacionais de código aberto mais utilizado no mundo. Ele permanece em segundo lugar em termos de popularidade perdendo apenas para o banco de dados da empresa ORACLE, porém este não apresenta código aberto. O MySQL, que utiliza a linguagem SQL, foi criado pela

ISSN: 2238-5819

empresa MySQL AB. Essa empresa foi adquirida pela Sun Microsystems no ano de 2008 e esta, por sua vez, foi incorporada à empresa Oracle no ano de 2009.

Apesar de a empresa Oracle ser detentora dos direitos autorais de seu código fonte, o MySQL pode ser utilizado gratuitamente, pois possui dupla licença: uma delas garante utilização gratuita e direitos de uso aos usuários através dos termos de licença *General Public License* (GNU), e outra licença para uso comercial mediante sua compra. Ambas apresentam o mesmo *software*, porém cada uma delas possui privilégios diferentes. Com sua estrutura escrita nas linguagens C e C++, é compatível com diversas plataformas e sistemas operacionais, possuindo ainda suporte total a *multi-threads* usando *threads* diretamente no *kernel*, utilizando múltiplas CPUs, se disponível.

Como um dos SGBDs mais utilizados, o MySQL está presente em empresas públicas e particulares no mundo inteiro, incluindo empresas de alta escala como as gigantes Google, Facebook e Youtube. Entre suas funcionalidades, o banco de dados MySQL se destaca por utilizar técnicas de otimização de desempenho. Segundo Dyer (2008), possui um sistema de alocação de memória muito rápido e baseado em processo (*thread*), tabelas *hash* em memória, que são usadas como tabelas temporárias, funções SQL implementadas por meio de uma biblioteca de classes altamente otimizada, *joins*, que utilizam uma *multi-join* de leitura única otimizada, tabelas em disco baseadas em árvores-B extremamente rápidas com compressão de índices.

Possui característica de suporte pleno a comandos e funções SQL, além de permitir que tabelas de bancos de dados distintos sejam misturadas para a realização de uma pesquisa e, de acordo com o *site* MySQL.com, seu sistema de segurança é flexível e seguro: as senhas e os privilégios de usuários são baseados em estações/máquinas e trafegam em meio criptografado ao conectar-se com o servidor. Outra característica importante é que o MySQL suporta bancos de dados enormes, com número incrivelmente alto de registros, graças ao seu sistema de índices que permite até 32 índices por tabela compostos cada um por até 16 colunas ou partes de colunas.

Esse SGBD possui suporte nativo para ambientes clusterizados através da ferramenta MySQL *Cluster*, que realiza função similar à solução proposta neste trabalho, porém sua licença de uso é atrelada à modalidade de licença paga deste banco de dados. Essa ferramenta pode ser objeto de trabalhos futuros, mas seu custo comercial a exclui como candidata à aplicação na proposta deste trabalho.

A escolha da utilização do MySQL neste projeto se deve à viabilidade e sinergia gerada na utilização de *softwares* de licença gratuita para a criação de um ambiente funcional que apresente alta disponibilidade e funcionalidades adequadas, mantendo um baixo orçamento para sua implementação.

ISSN: 2238-5819

1.2 *Cluster*

Cluster pode ser definido como um agrupamento de computadores também chamados de "nós", conectados entre si e trabalhando em conjunto com a finalidade de aumentar seu desempenho na execução das tarefas, segundo Ferreira (2003).

Os nós podem estar conectados através de uma rede local, que na opinião de Alecrim (2013), são mais comuns os padrões da Ethernet por terem menor custo e permitirem a retirada ou a inclusão de nós sem alterar seu funcionamento.

A estrutura de um *Cluster* pode ser construída por vários nós, sendo uma característica o fato de serem imperceptíveis ao usuário quantas máquinas estão conectadas. Na construção de um *Cluster*, as máquinas utilizadas podem ser Desktops de uso comum em qualquer outra finalidade ou preparadas para *Clustering*. Segundo Faustino e Freitas (2005), as máquinas preparadas para *Cluster* são chamadas multi-processadas, pois podem conter um numero maior de processadores. O agrupamento de nós resulta na soma de recursos computacionais como processador, memória, rede e armazenamento, que pode ser comparado a um supercomputador, já que permite um alto poder computacional. Segundo Bacellar (2012), uma das finalidades do *Cluster* é garantir o poder computacional e a transparência ao usuário.

Na preparação de uma máquina para *Cluster* é necessário considerar os recursos de *hardware* como fator importante no desempenho dos nós, Bacellar (2012) afirma que os recursos de processadores, memória RAM, placa mãe e disco rígido são os que mais influenciam em seu desempenho. Porém, na montagem de um *Cluster* com máquinas comuns, os equipamentos não necessitam da mesma similaridade de *hardware*, e na opinião de Pereira (2004) o sistema Operacional pode ser padrão. Para empresas de médio e pequeno porte pode ser uma excelente solução na utilização de máquinas já existentes.

Segundo Faustino e Freitas (2005), os *Clusters* podem ser de três tipos:

a) Alta Performance – *High Performance Computing* (HPC) – são utilizados para se obter alto desempenho em menor tempo para grandes processamentos. Suas máquinas com até dois processadores possuem maior poder computacional podendo alcançar maior velocidade de processamento. É utilizado na necessidade de se processar grandes quantidades de dados, como em resultados de concursos, eleições, vestibulares ou em grandes variedades de dados como cálculos.

b) Balanceamento de Carga – as tarefas são divididas igualmente entre os nós individuais ou divididas por performance, cada nó recebe uma parte de acordo com sua capacidade computacional. Ferreira (2003) afirma que o sistema de balanceamento de cargas gerencia os nós de forma a direcionar uma requisição para a máquina que estiver com menor número de tarefas. Esse tipo de *Cluster* é normalmente utilizado em sistemas Web, cujas requisições de tarefas podem aumentar de acordo com a demanda e comprometer o desempenho computacional.

ISSN: 2238-5819

c) Alta Disponibilidade – *High Availability* (HA) – os *Clusters* de HA buscam a alta disponibilidade dos sistemas que necessitam de processamento sem paradas através de monitoramento de possíveis falhas de *hardware* (nós) ou de *software*, de replicação de dados entre computadores, para substituição de máquinas com os sistemas em execução e até uso de equipamentos geradores de energia evitando paradas por agentes externos, afirma Ferreira (2003). No Linux HA, o *Cluster* funciona com uma máquina como espelho de outra, na ocorrência de falha de uma máquina outra máquina assume o lugar. Os *Clusters* de HA são de grande utilidade para instituições financeiras, *e-commerce*, hospitais, aeroportos, etc.

Para Alecrim (2013), uma solução em *Cluster* pode ser utilizada com mais de um tipo de *Cluster* são os chamados *Clusters* híbridos, que aliam dois ou mais tipos de *Clusters* em um único sistema. *Clusters* são muito utilizados no campo da computação científica, como Beowulf, GiganetcLAN e Berkeley NOW, que são usados em computação de larga escala, segundo Tang *et al*. (2005).

Para Faustino Jr. e Freitas (2005), são muitas as vantagens de um *Cluster* computacional:

a) Alto desempenho, que através de processamentos paralelos e balanceamento de carga pode-se encontrar solução para problemas complexos ou respostas em menor tempo;

b) Escalabilidade, que é a possibilidade de incluir máquinas ao *Cluster* e aumentar seu desempenho conforme o crescimento e a demanda de serviços;

c) Tolerância a falhas, que é a confiança na resolução do sistema, caso haja alguma falha de aplicativos ou de máquinas;

d) Baixo custo na obtenção de alto desempenho utilizando computadores disponíveis na empresa;

e) Independência de fornecedores na possibilidade de utilização de *softwares* livres e máquinas de diferentes fabricantes.

1.3 Alta Disponibilidade

O termo “alta disponibilidade” é frequentemente utilizado nessa área para descrever a característica de um sistema com alta capacidade de permanecer disponível durante o maior tempo possível durante a execução de serviços críticos. Tem como característica principal a garantia de suas funcionalidades e alta tolerância a falhas de *hardware*, *software* e energia através de dispositivos que gerem redundância no sistema. Para isso, podem ser utilizadas peças de *hardware* que iniciam seu funcionamento automaticamente ao detectar uma falha, assumindo o controle da operação previamente executada. Esta operação é chamada *Failover*.

De acordo com Piedad e Hawkins (2001), quanto maior a redundância do sistema, maior será a proteção e garantia de funcionamento do sistema ao se eliminar os *Single Point Of Failure* (SPOF), minimizando períodos de interrupção no serviço. É necessário apontar que para se alcançar níveis satisfatórios de alta disponibilidade é necessária maior redundância, que por consequência implica altos custos de operação. Uma alternativa aos altos valores vinculados a

ISSN: 2238-5819

essa tecnologia é a utilização de *clusters* criados a partir do agrupamento de *hardwares* mais acessíveis, tornando-os similares às máquinas de grande porte.

Quanto ao índice de tolerância a falhas, Piedad e Hawkins (2001) nos mostram que são utilizados dois parâmetros para mensurar o grau de disponibilidade, chamados *Mean Time Between Failures* (MTBF – tempo médio entre falhas), que registra a frequência de falhas ocorridas e o *Mean Time To Repair* (MTTR – tempo médio de recuperação), que registra o período entre a ocorrência da falha e a total recuperação do sistema ao seu estado operacional.

A disponibilidade de um sistema pode ser calculada pela relação entre seu tempo de vida útil e seu tempo de vida total, pela fórmula: Disponibilidade = MTBF / (MTBF + MTTR). Segundo Pereira Filho (2004), os níveis de disponibilidade podem ser entendidos através da incidência de *downtime* do sistema, conforme demonstrado na Tabela 1.

Tabela 1 - Níveis de Disponibilidade

	UPTIME	DOWNTIME	DOWNTIME POR ANO	DOWNTIME POR SEMANA
1	90%	10%	36.5 dias	16 horas, 51 minutos
2	98%	2%	7.3 dias	3 horas, 22 minutos
3	99%	1%	3.65 dias	1 hora, 41 minutos
4	99.8%	0.2%	17 horas, 30 minutos	20 minutos, 10 segundos
5	99.9%	0.1%	8 horas, 45 minutos	10 minutos, 5 segundos
6	99.99%	0.01%	52.5 minutos	1 minuto
7	99.999%	0.001%	5.25 minutos	6 segundos
8	99.9999%	0.0001%	31.5 segundos	0.6 segundos
9	99.99999%	0.00001%	3.15 segundos	0.06 segundos

Fonte: Pereira Filho (2004)

Algumas das porcentagens dessas ordens de magnitude são algumas vezes referidas pelo número de vezes em que o numeral nove é repetido em sequência, chamados então de “classe de noves”. Como exemplo, a distribuição de água encanada sem interrupções por um período de 99.999% do tempo total medido possui classe cinco de confiabilidade em sua distribuição. Métodos similares são utilizados para medição de disponibilidade de missões militares norte americanas ou para a medição de nível de pureza de substâncias, porém, Marcus (2003) questiona a utilização desse método para a medição da disponibilidade de sistemas informatizados.

Marcus (2003) afirma que associar essas sequências numéricas a padrões de qualidade da disponibilidade dos serviços é fundamentalmente errado por não considerar a diferença de valor do tempo de funcionamento de cada sistema e sua função. O autor exemplifica que um *site* de compras pela internet sofreria um impacto maior com uma indisponibilidade de cinco minutos em uma véspera de natal em comparação a uma indisponibilidade de mesmo período na madrugada de um domingo do mês de agosto.

De qualquer forma, a utilização desse método se popularizou graças à sua fácil classificação e entendimento por profissionais de áreas diversas à de sistemas informatizados.

ISSN: 2238-5819

Em todo caso, após a manutenção ou demais providências necessárias no sistema, é efetuada uma operação chamada *Failback* no qual o serviço provisoriamente deslocado é retornado à sua máquina de origem para a continuidade do serviço.

1.4 Linux

O Linux tem sua origem no Unix, que, segundo Morimoto (2013), é um sistema desenvolvido para uso em servidores, porque permite ser acessado por muitos usuários simultaneamente e que também permite a execução de vários programas ao mesmo tempo. Inicialmente foi desenvolvido para ser uma réplica do Unix, como uma opção de *software* livre e uma alternativa que exigisse menos recursos financeiros como na aquisição do Unix Comercial.

Segundo Mota Filho (2007), o Linux foi criado por Linus B Torvalds em 1991, um estudante finlandês que tinha a vontade de criar um sistema operacional livre similar ao Minix, uma versão simplificada do Unix. O estudante Linus utilizou os aplicativos já criados no Projeto GNU da *Free Software Fundation*, porque se seu *kernel* rodasse no GNU, também rodaria no Unix. Dessa forma, o nome correto deste importante sistema operacional é GNU/Linux, porém comumente chamado apenas como Linux (que significa a junção do nome Linus com Unix).

Para maior entendimento do Linux, dois conceitos importantes são explicados abaixo:

a) O Sistema Operacional é o responsável por fazer a interface entre os programas executados pelos usuários e o computador, como editores de textos, imagens e som, mas o núcleo do sistema operacional possui outras funções essenciais como gerenciamento dos recursos de memória, compartilhamento dos arquivos, discos e os recursos periféricos (físicos do *hardware*), segundo Alencar (2012),

b) O *kernel* é o núcleo do sistema operacional e contém as principais tarefas de gerenciamento. Alencar (2012) explica ainda que o *kernel* roda em camadas privilegiadas de memória e seu acesso é feito através de chamadas, disponibilizadas nas bibliotecas do sistema; já os programas de sistemas rodam em camadas não privilegiadas de memória que são os conjuntos de bibliotecas C (libc), shell (para o usuário digitar seus comandos) e o ambiente gráfico.

Para Alencar (2012), o sistema operacional do Linux possui um *kernel* desenvolvido como núcleo monolítico para a execução principal das tarefas deste sistema, ou seja, as tarefas de escalonamento de processos, gerenciamento de memória, execução de operações de entrada e saída e acessos aos arquivos de processos são executadas dentro do núcleo.

Dentre as características do *kernel* estão que algumas funções de sistemas de arquivos e suporte a rede são compilados e executados sem blocos ou módulos separadamente da parte principal do núcleo, são chamadas as bibliotecas *loadablekernel modules* e, segundo Alencar (2012), podem ser acessadas e descartadas após a execução do *kernel*.

Uma das principais características do GNU/Linux é o sistema de particionamento de arquivos que significa a divisão lógica do dispositivo de armazenamento de arquivos (disco rígido) em partes para executá-las separadamente como se fossem um disco único. Outras importantes

ISSN: 2238-5819

características do GNU/Linux, segundo Ferreira (2003): possui sistema executável em 32 ou 64 bits, sistema gráfico X-Windows, suporte as linguagens Java, C, C++, Pascal, Lisp, Prolog entre outras e suporte aos protocolos de rede TCP/IP, IPX, Apple Talk e NetBios.

A partir do Linux, muitas são as distribuições desenvolvidas e de grande utilização no mercado atual como a ArchLinux, Debian, Fedora, Mandriva, Mint, Opensuse, PCLinuxOS, Puppy, Sabayon, Slackware e Ubuntu. Algumas empresas compilaram o *software* e fornecem o sistema pronto para o uso como a RedHat, SuSE, Mandriva e a Canonical (desenvolvedora do Ubuntu Linux), e outros projetos como Debian ou Gentoo, segundo Alencar (2012).

Ferreira (2003) afirma que o Linux tem seu código-fonte ainda desenvolvido por muitos pesquisadores e programadores de diversos países, porém cada autor mantém os direitos de copyright sobre o código que escreveram.

Por ter um Sistema Operacional aceitável na execução em múltiplas arquiteturas como aplicações para *tablets* e *smatphones* ou ainda em sistemas robustos como mainframes para grandes empresas, é grande a abertura de mercado para o GNU/Linux, segundo Alencar (2012).

O GNU/Linux está licenciado pela GNU que permite a distribuição e a venda de versões desenvolvidas e modificadas do Linux, mas exige que todas as cópias sejam acompanhadas do código-fonte.

1.5 Linux Virtual Server

Para Dantas *et al.* (2002), o Linux Virtual Server (LVS) é uma configuração formada por um servidor virtual construído em um *cluster* de servidores reais em execução no sistema operacional GNU/Linux. Ao cliente externo parecerá que se trata de apenas um servidor real que atende as requisições de entrada e saída de dados.

Um *Cluster* LVS possui dois tipos de computadores: o servidor virtual (*frontend* ou nó mestre) responsável por receber e direcionar as requisições de serviços e os servidores reais (nós escravos) responsáveis por executá-las. Dantas (2002) afirma que as requisições de serviços são recebidas pelo servidor virtual e direcionadas a um dos nós disponíveis do *Cluster*, ou servidores reais, dessa forma, os computadores relacionados no *Cluster* respondem as requisições ao mesmo tempo e cada cliente executa requisições como se estivesse conectado diretamente a um servidor único.

Cota (2005) explica que existem três formas de implementar o servidor virtual: Virtual Server via NAT, Virtual Server via IP *Tunneling* e *Virtual Server* via Roteamento Direto:

a) Virtual Server Via NAT: Os servidores reais podem utilizar qualquer sistema operacional que atenda os protocolos de TCP/IP e também podem utilizar endereços particulares de Internet e apenas um endereço de IP. Todas as requisições são recebidas pelo servidor virtual e este vai designar o nó que irá executar a tarefa, o que pode se tornar um gargalo do sistema num *Cluster* LVS de mais de 20 nós.

ISSN: 2238-5819

b) Virtual Server Via IP *Tunneling*: Todas as requisições são agendadas pelo servidor virtual e são respondidas diretamente dos servidores reais aos clientes externos, portanto o servidor virtual pode receber inúmeras requisições sem se tornar um gargalo do sistema. Importante que o IP Tunneling precisa estar habilitado em todos os servidores do *Cluster* e, segundo Cota (2005), o Sistema Operacional Linux é o mais compatível, dessa forma, dificulta a habilitação para servidores com Sistemas Operacionais diferentes.

c) Virtual Server Via Roteamento Direto: Também como no IP *Tunneling*, no Roteamento Direto, o servidor virtual só agenda as requisições, deixando as tarefas de executar e responder aos servidores reais, porém exige que a interface do servidor virtual e dos servidores reais esteja na mesma parte física.

Cota (2005) afirma que o LVS é um sistema *open source* que através da estrutura em *Cluster* alcança alta disponibilidade para sistemas críticos.

2 Materiais e Métodos

2.1 Arquitetura do Ambiente Proposto

Para demonstrar no experimento a execução das requisições de entrada e saída de dados mantendo a disponibilidade do serviço em caso de falhas, será utilizada uma estrutura em *Cluster* LVS com Banco de Dados MySQL. A utilização do Banco de Dados MySQL é uma alternativa de baixo custo e de boa abrangência para essa finalidade.

Para criação do ambiente necessário para realização dos testes, foi utilizado um computador hospedeiro com a configuração AMD Phenon II x6 3.0GHz, com 8 GB de memória RAM, com dois *Hard Drives* de 500 GB, associados em RAID 0, executando Sistema operacional Windows 7 64 bits, no qual foram instaladas três máquinas virtuais com o *software* VirtualBox Versão 4.3.6. Em cada máquina virtual, foi disponibilizado 1 GB de memória RAM e instalado o sistema operacional Linux CentOS 32 bits Versão 5.4, selecionada por ser uma distribuição estável, bem aceita no mercado, além de ser homologada para a execução do ambiente proposto.

Das três máquinas virtuais, adotou-se uma como nó mestre e duas como nós escravos redundantes. O nó mestre é responsável por receber as requisições do cliente e repassá-las ao nó escravo ativo no momento. Para essa função, foram instalados no nó mestre os *softwares* Piranha e IPVS AdmVersão 1.24, responsáveis pelo gerenciamento dos acessos ao IP Virtual do *Cluster* e redirecionamento de requisições aos nós escravos. Esses *softwares* foram selecionados para essa função por estarem previamente disponíveis no repositório de aplicações do LinuxCentOS.

Os nós escravos, nos quais estão armazenadas as bases de dados, cumprem a função de receber e processar as requisições direcionadas pelo nó mestre para, então, responder diretamente ao cliente através do IP virtual do *cluster*. Simultaneamente, o nó escravo ativo gera um registro binário com as alterações enquanto o nó escravo inativo ocupa-se em atualizar sua base de dados com as informações mais recentes. Para os nós escravos cumprirem suas

ISSN: 2238-5819

funções, foi instalado o MySQL versão 5.0.95, habilitado a base de dados e definida senha para o usuário root.

A Figura 2 representa a arquitetura e o funcionamento do *cluster* LVS, demonstrando os sentidos possíveis do fluxo das requisições.

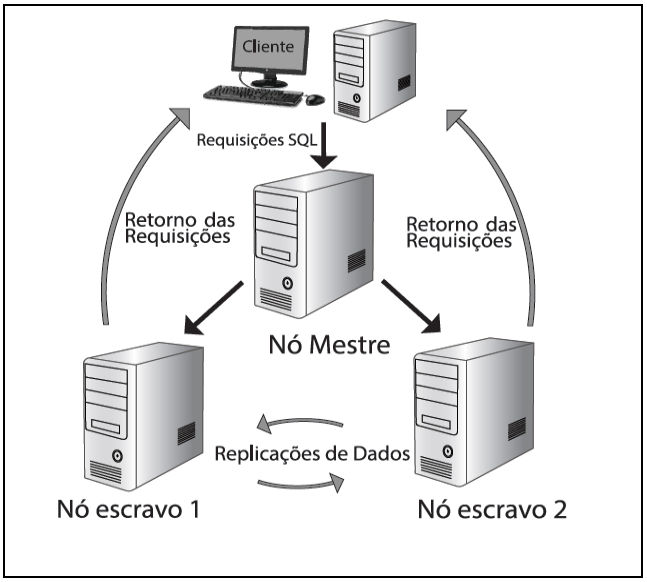

Figura 2 - Modelo de arquitetura e funcionamento
Fonte: Elaborado pelos Autores

2.2 Adaptações do Sistema

No nó mestre as configurações necessárias estão unicamente no *software* responsável por direcionar as requisições para as bases ativas. No *software* Piranha, foi necessária a criação de um script que verifica se os nós escravos estão ativos. Esse script realiza chamadas constantes aos nós escravos e espera retorno positivo do serviço MySQL em pleno funcionamento. Em caso negativo, o *software* Piranha automaticamente redireciona as futuras requisições ao nó escravo que estava anteriormente em espera.

Para que o registro binário de alterações das bases de dados seja devidamente criado nos nós escravos, e para que não haja conflitos de auto-incrementos nas bases de dados, é necessária a edição do arquivo de nome my.cnf, por não se tratar de configurações padrões do MySQL. Estas alterações demonstradas a seguir devem ser efetuadas na seção [mysqld] do arquivo my.cnf.

Nó escravo 1:

```
server-id = 1
auto-increment-increment = 2
auto-increment-offset = 1

log-bin
```

ISSN: 2238-5819

```
master-host = 192.168.0.202
master-user = root
master-password = 123456
master-port = 3306
master-connect-retry = 60
```

Nó escravo 2:

```
server-id = 2
auto-increment-increment = 2
auto-increment-offset = 2

log-bin

master-host = 192.168.0.201
master-user = root
master-password = 123456
master-port = 3306
master-connect-retry = 60
```

Conforme descrito no código acima, é necessário que no arquivo de configuração seja gravado a porta de conexão do MySQL, bem como um usuário e senha com permissão de acesso ao banco de dados.

Após reinicialização do banco de dados, é necessária a execução do comando mysql> start slave e faz-se necessária a habilitação do encaminhamento de pacotes IPV4 no arquivo sysctl.conf do diretório /etc alterando o parâmetro net.ipv4.ip_forwarddo valor 0 para 1.

Por fim, os IPs das máquinas foram definidos da seguinte maneira:

Tabela 2 – Endereços de IP do *cluster* LVS

Elementos LVS	IP Real	IP Virtual
Nó mestre	192.168.0.200	192.168.1.250
Nó escravo 1	192.168.0.201	
Nó escravo 2	192.168.0.202	

Fonte: Elaborado pelos autores

3 Resultados

Para demonstração prática da funcionalidade do *Cluster* foi inserido um dado com sua requisição realizada através do IP virtual 192.168.0.250. Pode ser observado que o dado inserido consta nos dois nós escravos, conforme demonstrado na Figura 3.

ISSN: 2238-5819

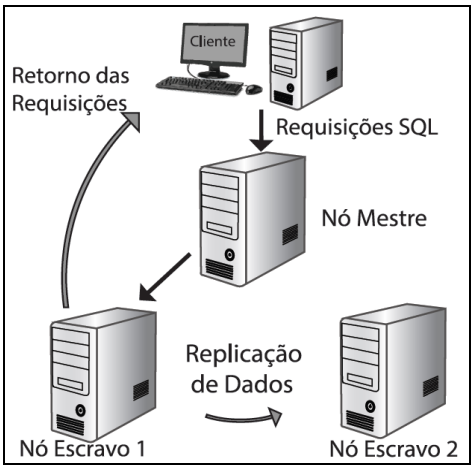

Figura 3 - Recebendo Requisições SQL no Nó escravo 1
Fonte: Elaborado pelos Autores

Ao interromper o serviço do banco de dados do nó escravo 1 e prosseguir com um *update* através do IP virtual, a nova informação é registrada apenas no nó escravo 2, que como demonstrado na Figura 4 se encontra operante e pode ser consultada normalmente de forma independente, sem sofrer influência do problema que afeta o nó desativado.

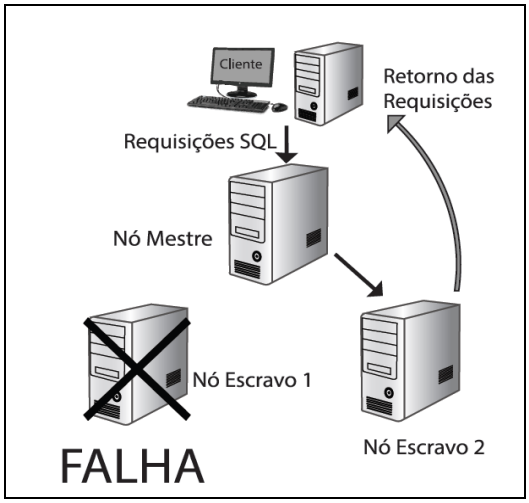

Figura 4 - Interrupção ou Falha no Nó Escravo 1
Fonte: Elaborado pelos Autores

Ao recuperar o sistema falho e reinicializar o serviço de banco de dados, pode-se observar que sua base de dados se atualiza automaticamente com todas as alterações realizadas durante o período de indisponibilidade, mantendo assim a paridade entre as bases de dados através de sua replicação, demonstrado na Figura 5.

ISSN: 2238-5819

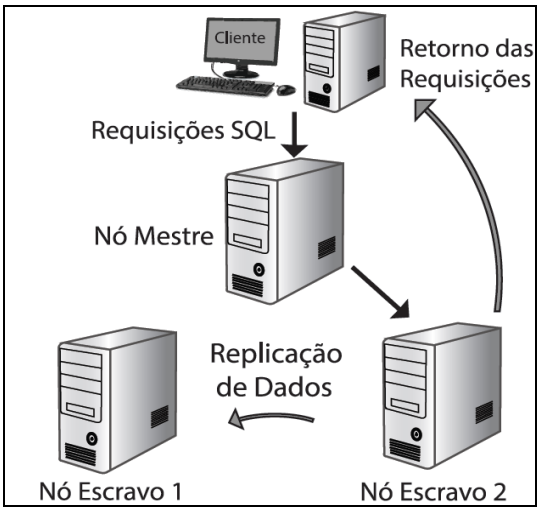

Figura 5 - Retorno da Replicação dos Dados entre os Nós
Fonte: Elaborado pelos Autores

A partir desse ponto, ainda que o nó mestre envie as requisições para o nó escravo 2, é possível interromper o serviço do banco de dados deste nó e continuar os trabalhos de inserção de dados através do IP virtual, que serão automaticamente repassados ao nó ativo, neste caso o nó escravo 1. É possível fazer consultas em tempo real para verificar que não apenas o registro foi realizado com sucesso, mas que a troca de servidores escravos foi bem sucedida durante a falha. Ao reinicializar o serviço do banco de dados do nó escravo 2, observamos que ele automaticamente atualizou sua base de dados, assim como ocorrido anteriormente ao testar o outro nó, verificando a plena funcionalidade do sistema proposto.

Dessa forma, a disponibilidade do serviço é contínua porque na possível falha de qualquer um dos nós escravos, o outro passa a operar imediatamente, pois possui a mesma atualização da base de dados do nó falho.

Considerações Finais

Ainda que longe de atingir padrões de qualidade alcançada por grandes empresas com a aquisição de *hardwares* específicos e contratação de serviços especializados em recuperação de desastres, o *cluster* aqui proposto apresenta uma disponibilidade de serviços proporcionalmente notável em relação ao seu custo. Utilizando apenas *softwares* de código fonte aberto, inserido dentro do trabalho como software *open source*, foi possível obter alta disponibilidade em um ambiente de banco de dados.

Com a finalidade de proteger dados de falhas comuns às quais todo banco de dados está sujeito, esta proposta fornece um bom nível de tolerância a falhas para ambientes de banco de dados de pequeno porte ou até mesmo em servidores responsáveis por tarefas secundárias em grandes empresas. As estratégias para obtenção de alta disponibilidade em qualquer ambiente computacional são abrangentes, oferecendo um nível maior ou menor de proteção, considerando a criticidade e também o valor disponível para investir em um ambiente mais robusto.

ISSN: 2238-5819

Independente da finalidade de sua utilização, esta simples solução potencializa a disponibilidade de dados de uma maneira eficaz e funcional, sem ignorar que este ambiente ainda é dependente de toda a infraestrutura da empresa, como soluções de energia, cabeamento e segurança em geral.

Finalmente, considerando que o desempenho geral do *cluster* também é dependente do poder computacional das máquinas nele agrupadas, e que é possível alcançar maior velocidade no processamento dos dados dividindo as tarefas por nós, podemos apontar como sugestão para trabalhos futuros a implementação de métodos de análise de desempenho e balanceamento de carga, que complementam a utilidade geral do *cluster* de forma a aproveitar uma excelente sinergia com as características de alta disponibilidade aqui demonstradas.

Referências

ALECRIM, E. Cluster: conceito e características, 2013. Disponível em: <http://www.infowester.com/cluster.php> Acesso em: 31 mar. 2014.

ALENCAR, M. S. A História do Linux. **Revista Difusão Científica**, 2012. Disponível em:<http://www.difusaocientifica.com.br/artigos/Historia_Linux.pdf.> Acesso em: 13 abr. 2014.

BACELLAR, H. V. Cluster: Computação de Alto Desempenho, 2012. Disponível em:<http://www.ic.unicamp.br/~ducatte/mo401/1s2010/T2/107077-t2.pdf> Acesso em: 04 abr. 2014.

CODD, E. F. A relational model of data for large shared data banks. **Communications of the ACM Magazine**, v. 13, Issue 6, 1970, p. 377-387. Disponível em:<http://www.dl.adc/citation.cfm?doid=362384.362685> Acesso em: 05 abr. 2014

COTA, A. Alta Disponibilidade com LVS, 2005. Disponível em: <http://www.vivaolinux.com.br/artigo/Alta-Disponibilidade-com-LVS?pagina=2> Acesso em: 13 abr. 2014.

DANTAS, M. A. R. *et al.* Integração de Alta Disponibilidade e Balanceamento de Carga em um Ambiente de Cluster. **Anais WSCAD**, 2002. Disponível em:<http://www.lbd.dcc.ufmg.br/colecoes/wscad/2002/002.pdf> Acesso em: 28 fev. 2014.

DB-ENGINES Ranking. Knowledge Base of Relational and NoSQL Database Management Systems. Disponível em:<db-engines.com/en/ranking> Acesso em: 06 abr. 2014.

DYER, R. J. T. MySQL in a Nutshell, **O'Reilly Media**, 2008, 2nd Edition. Disponível em: <http://books.google.com.br/books?id=lBHE8xdSGwQC&pg=PA549&lpg=PA549&dq=MySQL+in+a+Nutshell&source=bl&ots=9DTnf6qbxD&sig=J_HaEz_Kn78HSH5xBWvx_ePpsZI&hl=pt-BR&sa=X&ei=Fwh9VKmaDoGLgwTu_ICABA&ved=0CFQQ6AEwBg#v=onepage&q=MySQL%20in%20a%20Nutshell&f=false> Acesso em: 04 abr. 2014.

FAUSTINO Jr., E. P.; FREITAS, R. B. Construindo Supercomputadores com Linux - Cluster Beowulf. Centro Federal de Educação Tecnológica de Goiás. Departamento de Telecomunicações, Curso de Redes de Comunicação, 2005. Disponível em: <http://sinergbrasil.com/files/tcc-cluster-beowulf-2005.pdf> Acesso em: 04 abr. 2014

FERREIRA, R. Linux Guia do Administrador do Sistema. 2ª ed. Novatec, 2003. Disponível em:<https://www.novatec.com.br/livros/linuxguiaadm2/capitulo9788575221778.pdf> Acesso em: 13 abr. 2014.

LINUX Virtual Server. Disponível em:<http://www.linuxvirtualserver.org> Acesso em: 06 abr. 2014.

MARCUS, E. L. The myth of the nines. 2003. Disponível em:<http://www.searchstorage.techtarget.com/tip/The-myth-of-the-nines> Acesso em: 06 abr. 2014.

MORIMOTO, C. E. **Servidores Linux** – Guia Prático. Porto Alegre: Sul Editores, 2013.

MOTA FILHO, J. E. Descobrindo o Linux – Entenda o Sistema Operacional GNU/Linux. 2ª edição. São Paulo: Novatec, 2007.

MySQL. Why MySQL?. Disponível em:<http:www.mysql.com/why-mysql/spotlight-on-mysql-oracle-linux> Acesso em: 12 abr. 2014.

PEREIRA FILHO, N. A. **Serviços de Pertinência para Clusters de Alta Disponibilidade**. Dissertação de Mestrado, Instituto de Matemática e Estatística, São Paulo: USP, 2004. Disponível em:<http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04102004-104700/pt-br.php> Acesso em: 04 abr. 2014.

PIEDAD, F.; HAWKINS, M. **High Availability** – Design, Techniques and Processes. Prentice Hall Professional, 2001.

TANG, H. *et al*. Construction and Application of Linux Virtual Server for Scientific Computing. NPC Workshops, p. 287-289. IEEE Computer Society, Shanghai, China, 2008. Disponível em:<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=4663338> Acesso em: 27 fev. 2014.

Recebido em 29/10/2014

Aceito em 13/11/2014