UNIVERSIDADE FEDERAL DO PAMPA

TIAGO DE FREITAS VARGAS

ALTA DISPONIBILIDADE NA PLATAFORMA TRIXBOX

Bagé
2013

TIAGO DE FREITAS VARGAS

ALTA DISPONIBILIDADE NA PLATAFORMA TRIXBOX

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Orientador: Leonardo Bidese de Pinho

Coorientador: Sandro da Silva Camargo

Bagé
2013

TIAGO DE FREITAS VARGAS

ALTA DISPONIBILIDADE NA PLATAFORMA TRIXBOX

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Trabalho de Conclusão de Curso defendido e aprovado em: 08, de Agosto de 2013.

Banca examinadora:

_____

Prof. Dr. Leonardo Bidese de Pinho
Orientador
Unipampa

_____

Prof. Dr. Sandro da Silva Camargo
Unipampa

_____

Prof. Msc. Cristiano Cachapuz e Lima
Urcamp

Dedico esse trabalho primeiramente a Deus pela inspiração, à minha família, minha mãe Isabel Cristina, fonte de apoio emocional e financeiro durante toda a minha vida, à Cristiani por toda paciência e dedicação, aos meus amigos pelo companheirismo, amizade e incentivo, e a todas as pessoas que me ajudaram de alguma forma, direta ou indiretamente e aos professores pela ajuda e orientação.

"Os que confiam no SENHOR serão como o monte de Sião, que não se abala, mas permanece para sempre."

Salmo 125

RESUMO

Alta disponibilidade é uma das propriedades mais desejadas em sistemas computacionais. Neste trabalho é apresentada uma proposta visando agregar alta disponibilidade a um Sistema VoIP baseado na Plataforma Trixbox, por meio da inserção da ferramenta MySQL Cluster. Para validar a proposta, é implementado um ambiente experimental composto por quatro máquinas virtuais, sendo uma responsável por executar a Plataforma Trixbox, uma pelo nó gerenciador que faz o gerenciamento do cluster e outras duas que executam os componentes replicados de armazenamento e manipulação das consultas SQL. A partir do cenário apresentado é feita a integração do Trixbox com o MySQL Cluster, de modo que todos os dados gerados no Sistema VoIP passam a ser salvos automaticamente no banco de dados do cluster, que realiza, de forma transparente para a aplicação, a replicação dos mesmos. Os resultados experimentais demonstram que o acréscimo no tempo de resposta do banco de dados replicado é pequeno, da ordem de dezenas de milisegundos, e que a consistência é realizada adequadamente após a ocorrência de uma falha por queda.

Palavras-chave: Sistemas distribuídos; Banco de dados; Alta disponibilidade; VoIP.

ABSTRACT

High availability is one of the most desirable properties in computer systems. This work presents a proposal to add high availability to a VoIP system based on Trixbox platform, using MySQL Cluster tool. To validate the proposal, an implementation of an experimental environment is bult composed by four virtual machines, one responsible for running the Trixbox platform, one by the node manager that orchestrates the cluster and two other running replicated storage and SQL manipulation components. From this scenario, Trixbox integration with MySQL Cluster is presented, so that all data generated in the VoIP system becomes automatically saved in the database cluster, which performs, transparently to the application, the replication of data. Experimental results show that the increase in response time generated by the replicated database is small, on the order of tens of milliseconds, and that consistency is performed properly after a fail-stop fault.

Keywords: Distributed systems, Database, High Availability, VoIP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CPUs - *Single Central Processing* (Central Única de Processamento)

GB – *Gigabytes*

IAX - *Inter Asterisk eXchange*

IP - *Internet Protocol* (Protocolo de Internet)

ITU - *International Telecommunication Union* (União Internacional de Telecomunicações)

LANs - *Local - Area Network*

MB - *Megabytes*

MGCP - *Media Gateway Control Protocol*

MySQL - *My Structured Query Language* (Minha Linguagem de Consulta Estruturada)

NDB - *Node Data Base*

PABX - *Private Automatic Branch Exchange* (Central Automática de Comutação Privada)

PCs - *Personal Computer* (Computador Pessoal)

PCM - *Pulse Code Modulation* (Modulação por Código de Pulsos)

QoS - *Quality of Service* (Serviço de Qualidade)

RAM – *Randon Access Memory*

RTP - *Real-time Transport Protocol* (Protocolo de Transporte em Tempo Real)

SGBD - Sistema Gerenciador de Banco de Dados

SGBDD - Sistema Gerenciador de Banco de Dados Distribuídos

SIP - *Session Initiation Protocol* (Protocolo de Iniciação de Sessão)

SO - *Operational System* (Sistema Operacional)

TCP - *Transmission Control Protocol* (Protocolo de Controle de Transmissão)

UDP - *User Datagram Protocol*

VM - *Virtual Machine* (Máquina Virtual)

VoIP - *Voice over Internet Protocol* (Voz sobre Protocolo de Internet)

WANs - *Wide - Area Network*

SUMÁRIO

1 INTRODUÇÃO

O banco de dados é uma coleção logicamente coerente de dados com algum significado inerente que representam algum espaço do mundo real, sendo projetado, construído e populado para uma finalidade específica (ELMARSI e NAVATHE, 2011). Porém, para que um banco de dados seja preciso e confiável o tempo todo, ele precisa ser um reflexo verdadeiro do que representa. Portanto, as mudanças do mundo real precisam ser refletidas no banco de dados o mais breve possível.

A partir da criação do Sistema Gerenciador de Banco de Dados (SGBD) houve uma verdadeira revolução tecnológica, pois os bancos de dados desempenham um considerável papel em grande parte das áreas em que computadores são usados, tendo uma importante contribuição nos Sistemas de Informação. Entretanto, muitas vezes, os dados podem ficar sujeitos a perdas e danos devido a alguma falha no sistema. Por isso, no estudo em questão, busca-se uma solução para tal problema através da implementação de um Sistema de Banco de Dados com Alta Disponibilidade, sendo que nesse sistema, os dados são replicados evitando sua perda.

Para isso, foi implementado um banco de dados distribuído utilizando o sistema gerenciador de banco de dados MySQL Cluster, sendo que os dados a serem sincronizados encontram-se em um SistemaVoIP denominado Trixbox (FONALITY, 2013).

Percebe-se a aplicação desenvolvida no presente trabalho como algo enriquecedor na área de tecnologia, já que, a alta disponibilidade, evidenciada no mesmo, torna-se algo cada vez mais desejada e necessária em sistemas de bancos de dados, diminuindo a ocorrência de falhas, e tornando o sistema mais confiável e satisfatório aos usuários.

Além disso, a proposta em questão pode beneficiar os serviços de telecomunicações, que, por meio da implementação desse sistema, podem garantir uma maior disponibilidade dos dados da instituição onde o Sistema VoIP, que é sistema de conversão de voz sobre IP, estiver implantado.

1.1 Contexto

O desenvolvimento do trabalho gira em torno da sincronização de um banco de dados distribuído, possibilitando o compartilhamento de informações e a alta disponibilidade dos dados, tendo como foco que o sistema esteja operando corretamente e esteja disponível sempre que solicitado.

O estudo também conta com a utilização do Sistema VoIP, sendo este, um sistema de conversação que trabalha a partir do protocolo IP. Tal sistema é integrado no Trixbox que é um sistema integrado de ferramentas responsável por efetuar ligações via internet.

1.2 Motivação

A motivação do presente trabalho ocorreu pelo fato dos sistemas distribuídos serem uma área da ciência da computação que está mudando rapidamente, o que ocasiona uma considerável evolução, disponibilizando assim, mais serviços aos usuários.

Outro motivo que leva ao interesse de tal estudo é quanto ao banco de dados, que atualmente é um componente essencial da vida na sociedade moderna. Embora o banco de dados seja algo indispensável no uso das tecnologias, muitas vezes, pode ocorrer uma falha inesperada, ocasionando a perda parcial ou total dos dados. Por isso, acredita-se que por meio de técnicas de replicação de dados, esse tipo de imprevisto possa ser minimizado, pois os dados estarão disponíveis em mais de um componente do sistema distribuído.

1.3 Objetivos

- Geral:
    - Agregar Alta Disponibilidade a um Sistema VoIP, para minimizar a perda de dados.
- Específicos:
    - Fazer a sincronização de réplicas do banco de dados contido em um Sistema VoIP (Plataforma Trixbox), instanciadas em múltiplas máquinas virtuais, por meio de ferramentas usadas em Computação em Cluster (MySQL Cluster);

- Proporcionar a partir do compartilhamento de dados, um sistema VoIP com alta disponibilidade dos mesmos;

1.4 Metodologia

Para chegar ao objetivo do trabalho, que é fazer com que duas máquinas virtuais sincronizem os dados simultaneamente, sendo gerenciadas por outra máquina, é implementado um sistema de banco de dados distribuídos com a utilização do sistema gerenciador de banco de dados MySQL Cluster. Sendo que os dados replicados são as informações contidas em um banco de dados de um sistema VoIP denominado Trixbox (FONALITY, 2013), utilizado para a gestão dos usuários.

1.5 Estrutura do Trabalho

O restante do trabalho está dividido em quatro capítulos, conforme segue:

- Capítulo 2 (Referencial teórico-prático): descreve o embasamento que integra conceitos e técnicas de três grandes áreas: Sistemas distribuídos com alta disponibilidade; Sistemas VoIP; e Bancos de dados;
- Capítulo 3 (Proposta: Sistema VoIP com alta disponibilidade de dados): apresenta a proposta de um sistema VoIP com alta disponibilidade de dados de gestão dos usuários;
- Capítulo 4 (Validação): expõe a implementação da proposta, descrevendo o ambiente experimental utilizado, apresentando o processo de instalação e adaptação do Trixbox ao MySQL Cluster, e, também, os resultados obtidos;
- Capítulo 5 (Conclusão): apresenta as conclusões sobre o trabalho desenvolvido e aponta os trabalhos futuros a serem realizados.

2 REFERENCIAL TEÓRICO-PRÁTICO

O presente capítulo apresenta o referencial teórico-prático adotado neste trabalho. Em particular são apresentados conceitos de Sistemas Distribuídos, dando ênfase aos Sistemas de Computação em Cluster e à Alta Disponibilidade e replicação. Esse capítulo, apresenta ainda o conceito de Sistemas VoIP e Banco de Dados enfatizando a ferramenta MySQLCluster.

2.1 Sistemas Distribuídos

Os sistemas de computação estão passando por uma revolução, sendo que segundo Tanenbaum e Steen (2007), desde 1945, quando começou a era moderna dos computadores, até aproximadamente 1985, os computadores eram grandes e caros. Conforme estes autores, o resultado é que a maioria das organizações tinha apenas alguns poucos computadores e, na falta de um modo de conectá-los, eles funcionavam independentemente uns dos outros.

A partir de meados da década de 1980, dois avanços tecnológicos começaram a mudar essa situação. Sendo, um deles, o desenvolvimento de microprocessadores de grande capacidade. Pois, inicialmente, eram máquinas de 8 bits, mas logo tornaram-se CPUs de 16, 32 e 64 bits. O outro avanço foi em relação à invenção de redes de computadores de alta velocidade, sendo que redes locais, conhecidas como LANs (local-area networks), possibilitam que máquinas localizadas fisicamente próximas umas das outras se comuniquem entre si, permitindo que a transferência de dados entre elas. Já nas redes de longa distância, referenciadas como WANs (wide-area networks), é possível que máquinas no mundo inteiro se conectem a velocidades que variam de 64 Kbits/s a Gigabits por segundo.

Esse último avanço, como já dito anteriormente, possibilita a comunicação e troca de informações entre computadores de uma mesma rede, seja ela local ou de longa distância, sendo que o resultado desse compartilhamento de dados forma um Sistema Distribuído, amparado pela seguinte conceituação:

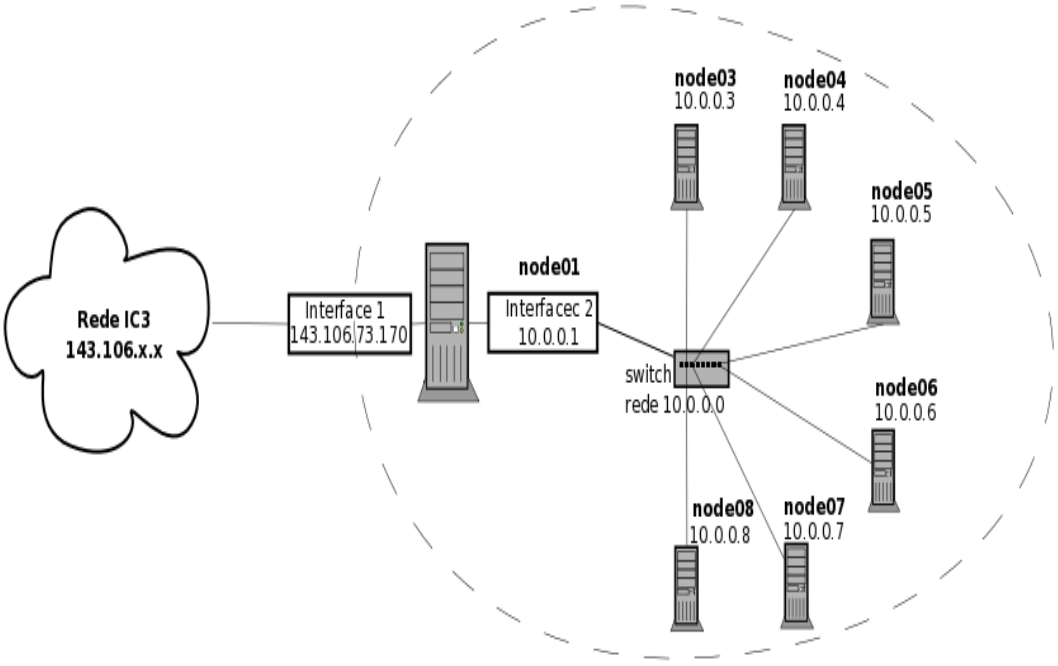
“Um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente” (TANENBAUM; STEEN, 2007, p. 1), ou seja, a partir desta citação percebe-se que em um sistema distribuído os computadores são autônomos, porém precisam colaborar uns com os outros. De forma similar, Coulouris, Dollimore e Kindberg (2007) definem um Sistema Distribuído como sendo aquele no qual os componentes de hardware ou software, localizados em computadores interligados em rede, se

comunicam e coordenam suas ações apenas enviando mensagens entre si. Ainda conforme estes autores, a principal meta de um sistema distribuído é facilitar aos usuários, e às aplicações, o acesso a recursos remotos e seus compartilhamentos de maneira controlada e eficiente.

2.1.1 Sistemas de Computação em Cluster

Conforme Tanenbaum e Steen (2007), na computação de cluster, o *hardware* subjacente consiste em um conjunto de estações de trabalho ou PCs semelhantes, conectados por meio de uma rede local de alta velocidade. A computação de cluster, conforme mostra a figura 1, geralmente, é usada para programação paralela na qual o único programa, intensivo em computação, é executado em paralelo em várias máquinas. Sendo que, cada cluster consiste em um conjunto e nós de computação controlados e acessados por meio de um único nó mestre. Segundo os autores, as tarefas típicas do nó mestre são manipular a alocação de nós a um determinado programa paralelo, manter uma fila de *jobs* apresentados e proporcionar uma interface para os usuários do sistema.

Figura 1 - Sistema de Computação em Cluster



Fonte: Instituto de Computação, 2009

A Figura 1 apresenta o exemplo de um sistema de computação em cluster, onde há um nó (nodo) mestre (node01) que gerencia os demais nós (node03-node08), todos estes interligados por um elemento de interconexão (switch), possibilitando a distribuição do sistema, viabilizando o ambiente necessário para a replicação de dados em componentes de hardware distintos.

2.1.2 Alta Disponibilidade e replicação

Na alta disponibilidade parte-se da premissa de que um sistema esteja operando corretamente e esteja disponível para executar seu serviço no instante em que for solicitado. Conforme Ahrendt (2010), a disponibilidade é medida com o percentual de tempo de atividade dividido pelo tempo total. O termo "alta disponibilidade" tem muitas definições mas, na maioria das vezes, refere-se a um sistema que está disponível na maior parte do tempo, por exemplo, um sistema com indisponibilidade mínima, como uma indisponibilidade programada, necessária para atualizações ou manutenção. A disponibilidade é medida com o percentual de tempo de atividade dividido pelo tempo total.

Sob esta perspectiva, em uma perspectiva conceitual, um sistema pode ser altamente disponível mesmo que apresente períodos frequentes de inoperabilidade, desde que estes períodos sejam significativamente curtos, ou seja, que o percentual de tempo em que o sistema fica indisponível seja insignificante.

Segundo Pasin (2003), alta disponibilidade pode ser implementada em sistemas computacionais por réplicas de componentes de *software* e *hardware*, sendo que o uso de réplicas permite que componentes operacionais falhos – os que apresentem inoperabilidade - sejam substituídos por outros componentes operacionais. Além disso, o uso de réplicas possibilita o *failover* que é uma das propriedades mais desejadas e mais difícil de ser assegurada em sistemas distribuídos. Basicamente, um mecanismo de *failover* faz com que a réplica assuma as funções do componente se tornou inoperante, de forma transparente para o usuário do serviço.

Para Pasin (2003), a implementação para alta disponibilidade depende da estrutura do sistema distribuído disponível, sendo que a forma mais usada é a replicação por *software* ou por *hardware*. Embora existam estas duas possibilidades, a réplica por *software* é a tendência atual, pois apresenta baixo custo de desenvolvimento em relação ao *hardware*, que requer componentes específicos.

Sobre replicação para alta disponibilidade, Pasin (2003) afirma:

> A natureza isolada dos nodos em um ambiente distribuído pode ser facilmente aproveitada para implementar alta disponibilidade através da replicação através do conceito de grupos. A replicação permite alta disponibilidade, quando uma réplica localizada em um nodo falho é substituída por outra, e alto desempenho principalmente através de acessos concorrentes para leitura em réplicas. (PASIN, 2003)

Portanto, conforme exposto acima, a replicação é utilizada para possibilitar a alta disponibilidade quando ocorrer falha em um nodo, sendo que esse nodo será substituído por outro.

A replicação não traz apenas benefícios, mas também desafios. Em um ambiente com replicação, quando um objeto é alterado, suas réplicas também devem ser atualizadas para que mantenham um estado distribuído consistente.

A replicação em sistemas de banco de dados pode ser assíncrona ou síncrona. Enquanto a replicação assíncrona realiza cópias dos objetos em intervalos de tempo pré-estabelecidos, a replicação síncrona requer servidores secundários ativos (*online backups*), os quais são atualizados à medida que as operações vão sendo processadas no banco de dados primários. Portanto, sistemas de banco de dados replicados com requisitos rígidos de alta disponibilidade demandam técnicas síncronas.

Como afirma Pasin (2003), sistemas de banco de dados síncronos tradicionalmente implementam alta disponibilidade usando técnicas de replicações centralizadas para propagar as atualizações de uma transação.

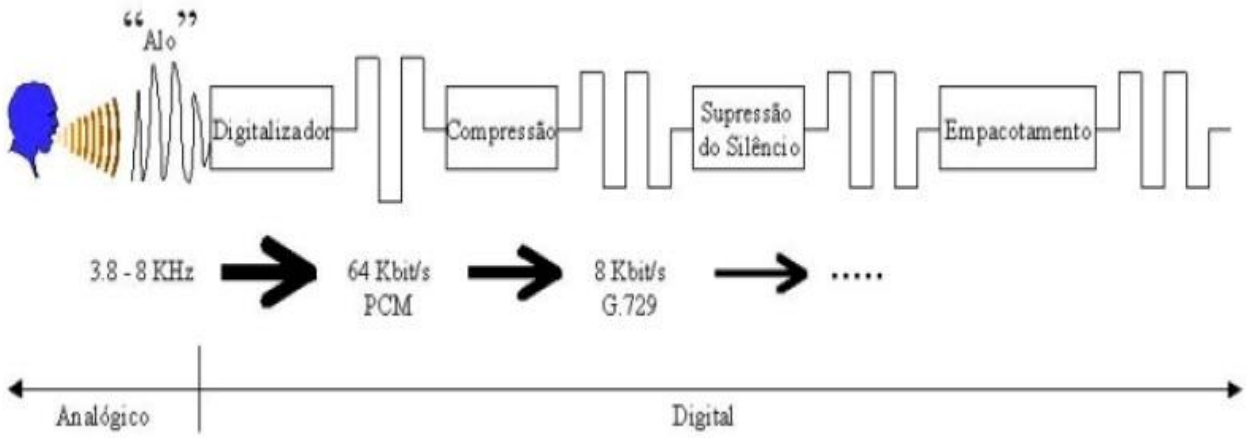
2.2 Sistemas VoIP

Sistemas VoIP (também chamados de aplicações VoIP) são sistemas de conversação que utilizam tecnologias de transmissão de voz sobre o protocolo IP. O uso de tecnologias de transmissão de voz sobre IP (VoIP -Voice over IP) está se expandindo cada vez mais e tende a alterar o perfil das operadoras de telecomunicações, sendo que a demanda por sistemas de voz sobre IP tem crescido em comparação aos sistemas de telefonia convencionais, atraindo a atenção inclusive das grandes operadoras. Isso acontece porque esse tipo de serviço oferece suporte para a comunicação de voz utilizando o Protocolo da Internet (IP) com preços atrativos, em função do

uso compartilhado dos links de comunicação por meio da multiplexação estatística que ocorre nas comunicações por comutação de pacotes (KUROSE; ROSS, 2006).

Em Sistemas VoIP, a voz humana é digitalizada, codificada e enviada na forma de fluxo de pacotes através de uma rede IP, como mostra a Figura 2:

Figura 2 - Funcionamento do Sistema VoIP



Fonte: Sitolino, 2001,p.21

Segundo Pedroso et al. (2008), dispositivos ou programas denominados *codecs* (*coder-decoder*) realizam a codificação e decodificação de um fluxo de dados de voz. Como já dito anteriormente, um sistema VoIP é caracterizado pelo transporte da voz através de uma infraestrutura baseada em IP, sendo que, usualmente, se refere à conversão de voz tradicional (sinal analógico) em sinais digitais a serem transportados em pacotes IP. Esta conversão conhecida como codificação da voz, segundo Pedroso, Caldeira e Fonseca (2006) pode ser realizada diretamente no próprio aparelho telefônico (telefonia IP), no PABX ou em *gateways* específicos antes de serem encaminhados para roteadores IP.

O processo inicia-se quando o utilizador retira o telefone IP do gancho, sendo emitido um sinal para a aplicação sinalizadora do roteador de "telefone fora do gancho", então a parte de aplicação emite um sinal de discagem. Após, o utilizador digita o número de destino, cujos dígitos são acumulados e armazenados pela aplicação da sessão. Em seguida, os *gateways* comparam os dígitos acumulados com os números programados, sendo que quando há uma coincidência ele mapeia o endereço discado com o IP do *gateway* de destino. Dessa forma,a

aplicação de sessão roda o protocolo de sessão sobre o IP para estabelecer um canal de transmissão e recepção para cada direção através da rede IP. No entanto, se a ligação estiver sendo realizada por um PABX, o *gateway* troca a sinalização analógica digital com o PABX, informando o estado da ligação. Sendo que, se o número de destino atender a ligação, é estabelecido um fluxo RTP sobre UDP entre o *gateway* de origem e destino, tornando a conversação possível.

Devido ao volume de dados gerados por uma aplicação VoIP, esta tecnologia se encontra em funcionamento principalmente em redes corporativas privadas. Mas conforme afirma Sitolino (2001), se a rede base para o transporte desta aplicação for a Internet, não deveria ser utilizada para fins profissionais, em virtude do TCP/IP não oferecer padrões de QoS (Qualidade de Serviço), comprometendo desta forma a qualidade da voz. Assim sendo, segundo este autor, o uso do VoIP, muitas vezes, se restringe a redes corporativas privadas, nas quais é relativamente simples e menos oneroso a disponibilização de amplos recursos em termos de largura de banda passante. Portanto, a capacidade de administrar a rede é fundamental para o sucesso da comunicação, já que a qualidade da voz fica dependente da intensidade do tráfego de dados existente no momento da conversa.

Ainda segundo Sitolino (2001), a transmissão de voz tem que garantir dois aspectos principais: um atraso mínimo para permitir a sensação de presença e um bom entendimento da conversa e até mesmo poder reconhecer a voz da pessoa com quem se esteja conversando.

Segundo Pedroso et al. (2008) os equipamentos que operam com VoIP devem suportar no mínimo o codec ITU G.711 baseado na técnica PCM (Pulse Code Modulation). No entanto, os dispositivos podem implementar *codecs* mais complexos, como por exemplo G.723 e G.729, o que impacta o perfil de tráfego gerado. Por exemplo, os *codecs* que obtém melhores taxas de compressão tendem a utilizar pacotes maiores transmitidos a intervalos de tempo maiores se comparados ao G.711.

A voz humana pode ser modelada como uma sequência de períodos de atividade e silêncio, conhecida como padrão ON-OFF. Porém, um dos problemas enfrentados na implementação dessa nova tecnologia é o correto dimensionamento do sistema, sendo que o tráfego gerado por sistemas VoIP não possui as mesmas características apresentadas pelo sistema de telefonia convencional.

2.3 Bancos de Dados

Banco de dados pode ser definido genericamente como coleção de dados relacionados (ELMARSI; NAVATHE, 2011). Contudo, sua definição pode ser pensada de forma ainda mais ampla e abrangente, possibilitando inúmeros benefícios e facilidades que pode trazer para as empresas e órgãos públicos. Isto porque o compartilhamento de um banco de dados permite que diversos usuários e programas acessem-no simultaneamente, ou seja, na abordagem de banco de dados, um único repositório mantém dados que são definidos uma vez e depois acessados por vários usuários.

Bancos de dados podem ser manuais ou computadorizados (ELMARSI; NAVATHE, 2011), sendo que um banco de dados computadorizado pode ser criado e mantido por um grupo de programas de aplicação escritos especificamente para essa tarefa ou por um sistema gerenciador de banco de dados.

Segundo Elmarsi e Navathe (2011), um sistema gerenciador de banco de dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados, sendo o SGBD um sistema de *software* de uso geral que facilita o processo de definição, construção, manipulação e compartilhamento de dados entre diversos usuários e aplicações. Estes autores afirmam, ainda, que não é necessário utilizar *software* de SGBD de uso geral para implementar um banco de dados computadorizado, podendo-se escrever um conjunto próprio de programas para criar e manter o banco de dados, caracterizado como um *softwar*e de SGBD próprio de uso especial. Alternativamente, pode-se trabalhar na perspectiva de modificar um SGBD existente, ou integrá-lo a outras ferramentas, para que este passe a atender as necessidades de uma aplicação.

O banco de dados e o catálogo do SGBD, habitualmente são armazenados em disco, sendo que o acesso ao disco é controlado, em especial, pelo sistema operacional (SO). Além disso, muitos SGBDs possuem o próprio módulo de gerenciamento de buffer para planejar a leitura/escrita em disco, obtendo um efeito considerável sobre o desempenho.

Conforme Elmarsi e Navathe (2011), SGBD Distribuído (SGBDD) pode ter o banco de dados real e o *software* de SGBD distribuídos por vários locais, conectados por uma rede de computadores. Os SGBDDs podem ser classificados como homogêneos e heterogêneos, sendo que os homogêneos usam o mesmo software de SGBD em todos os locais, enquanto os heterogêneos podem usar um software de SGBD diferente em cada local.

Atualmente, tem-se SGBDs de código aberto (gratuito), como MySQL e PostgreSQL, que tem suporte de fornecedores terceirizados com serviços adicionais. Em particular, em uma das versões disponíveis do MySQL, são disponibilizadas funcionalidades no contexto de alta disponibilidade.

2.3.1 MySQL Cluster

De acordo com Oracle (Oracle Corporation, 2013), MySQL Cluster é uma tecnologia que permite o agrupamento de bancos de dados compartilhados na memória de um sistema. A arquitetura compartilhada permite ao sistema trabalhar com *hardware* básico, e com um mínimo de requisitos específicos de *hardware* ou *software*. MySQL Cluster é projetado para não ter nenhum ponto de falha. Em um sistema compartilhado, cada componente deverá ter sua própria memória e disco. Já os mecanismos de armazenamento são compartilhados. Basicamente, o MySQL Cluster integra o servidor MySQL padrão com um mecanismo de armazenamento de cluster em memória chamado NDB (que significa "Network Data Base"). Segundo a Oracle, tal termo refere-se à parte da instalação que é específico para o mecanismo de armazenamento, enquanto que MySQL Cluster refere-se à combinação de um ou mais servidores MySQL com o NDB que é o mecanismo de armazenamento.

Do ponto de vista de componentes, o MySQL Cluster é composto por um conjunto de computadores, onde cada um deles executa um ou mais processos, sendo que tais processos são chamados de nós, caracterizados como segue:

***Data Node*:** É o nó responsável pelo armazenamento dos dados. Apenas um *Data Node* já é suficiente para o armazenamento dos dados. Entretando, é recomendado ter dois ou mais *Data Nodes* pois assim ocorrerá replicação de tais dados e, consequentemente, a possibilidade de oferecer uma solução de alta disponibilidade.

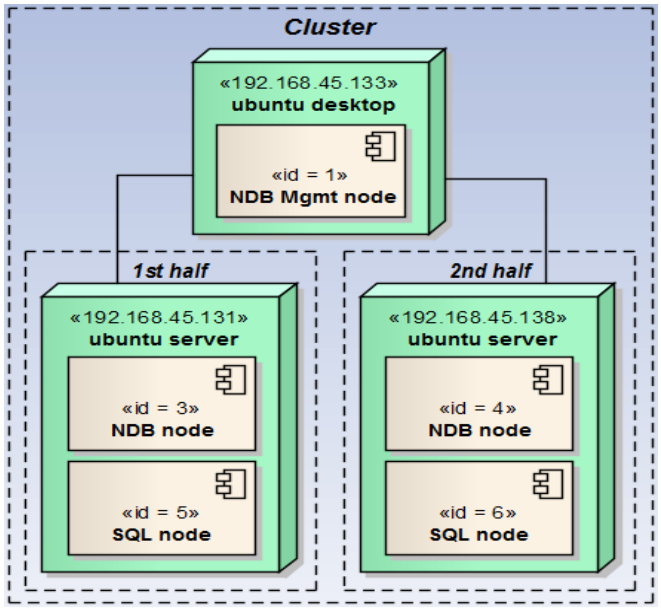
Management Node: Trata-se do nó gerenciador, que possui a função de gerenciar os outros nós dentro do MySQL Cluster, realizando funções como fornecimento de dados de configuração, iniciar e parar nós e fazer backup. Uma vez que esse tipo de nó controla a configuração dos outros nós, tal nó deve ser iniciado antes de qualquer outro.

SQL *Node:* Tal nó executa uma versão especializada da ferramenta MySQL Server (Oracle Corporation, 2013), a qual adota uma API complementar para interfaceamento com a

ferramenta de armazenamento NDBCLUSTER (Oracle Corporation, 2013), sendo responsável pelo acesso aos dados do cluster a partir de consultas (*queries*) feitas pela aplicação.

A arquitetura de um ambiente MySQL Cluster e o gerenciamento dos nós pode ser melhor entendido observando-se a Figura 3.

Figura 3 - MySQL Cluster



Fonte: Wargandi, 2013

Os nós apresentados na figura 3 trabalham juntos para formar um MySQL Cluster, sendo que quando os dados armazenados no Data Node são atualizados, todos os outros servidores que consultam estes dados podem ver a mudança imediatamente. No entanto, um servidor MySQL que não está ligado a um MySQL Cluster não pode usar o mecanismo NDB de armazenamento e não pode acessar quaisquer dados do MySQL Cluster.

O ambiente MySQL Cluster pode ser implementado em um host físico separado, em VM ou instância em nuvem. Sendo que, um cluster implementado em um host físico tem uma estrutura física como lógica, que consiste em um conjunto de computadores (cluster hosts) ou hosts. Enquanto que os Clusters virtuais possuem uma estrutura virtual e são criados com VMs instaladas em servidores distribuídos a partir de um ou mais clusters de máquinas físicas. Sendo que, as VMs em um cluster virtual são logicamente conectadas por uma rede virtual em várias redes físicas, ou seja, cada cluster virtual é formado com máquinas físicas ou uma máquina

virtual hospedada por vários clusters físicos. Além disso, pode-se dizer que possuem certas vantagens sobre clusters de máquinas físicas em termos de velocidade, flexibilidade e armazenamento. Já o Cluster em nuvem possui as mesmas características do Cluster físico, porém, pode ser acessado de forma remota.

3 PROPOSTA: SISTEMA VOIP COM ALTA DISPONIBILIDADE DE DADOS

Segundo Tanenbaum (2007), disponibilidade é definida como a propriedade de um sistema estar pronto para ser usado imediatamente, sendo que em um sistema com alta disponibilidade, os dados precisam estar disponíveis sempre que acessados. Por isso, em geral, refere-se à probabilidade de o sistema estar funcionando corretamente em qualquer momento determinado e estar disponível para executar suas funções em nome de seus usuários. Em números pode-se dizer que um sistema com alta disponibilidade estará disponível em 99,999% das vezes consultadas.

Para conseguir tal disponibilidade, as aplicações de banco de dados tornam-se um campo com necessidades de tolerância a falhas, visando a integridade dos mesmos. Sendo assim, ressalta-se a importância da implementação do Sistema VoIP com alta disponibilidade de dados, visando manter a integridade destes durante a operação normal, bem como após a recuperação ocasionada por uma falha. No entanto, isso implica em medidas extras de segurança na forma de verificação de consistência e redundância, preservando informações importantes que poderiam ser perdidas em caso de falhas.

Tendo como base Tanenbaum (2007), acredita-se que falhas como as descritas na tabela 1, podem interromper o bom funcionamento do Sistema VoIP, sendo prejudicial aos usuários.

Tabela 1 - Tipos de falha

Tipo de falha	Descrição
Falha por queda	O servidor para de funcionar, mas estava funcionando corretamente até parar.
Falha por omissão *Omissão de recebimento* *Omissão de envio*	O servidor não consegue responder a requisições que chegam O servidor não consegue responder mensagens que chegam O servidor não consegue enviar mensagens
Falha de temporização	A resposta do servidor se encontra fora do intervalo de tempo
Falha de resposta *Falha de valor* *Falha de transição de estado*	A resposta do servidor está incorreta O valor da resposta está errada O servidor se desvia do fluxo de controle correto
Falha arbitrária	Um servidor pode produzir respostas arbitrárias em momentos arbitrários

Fonte: Dados primários

Além das falhas serem prejudiciais aos usuários do sistema, como já exposto anteriormente, podem ocasionar falta de confiança no sistema fornecido. Sendo que uma falha por queda ocorre quando um servidor para prematuramente, mas estava funcionando corretamente até parar. Um exemplo típico de uma falha por queda é um sistema operacional que para de repente, havendo uma única solução: reinicializá-lo.

Uma falha por omissão ocorre quando um servidor deixa de responder a uma requisição. No caso de uma falha por omissão de recebimento, é possível que o servidor nunca tenha recebido a requisição. A falha de omissão de recebimento em geral não afeta o estado corrente do servidor, porque o servidor não fica ciente de que qualquer mensagem foi enviada a ele. Já a falha por omissão de envio ocorre quando o servidor fez seu trabalho, mas, de algum modo, deixa de enviar uma resposta. Sendo que, ao contrário de uma falha por omissão de recebimento, agora o servidor pode estar em um estado que reflete que ele acabou de concluir um serviço para aquele cliente.

As falhas de temporização ocorrem quando a resposta se encontra fora de um intervalo de tempo real especificado, podendo fornecer dados muito cedo, o que pode causar problemas para um receptor se não houver espaço de buffer suficiente para conter todos os dados que chegam. Entretanto, o mais comum, é que um servidor responda tarde demais, quando então se diz que ocorreu uma falha de desempenho.

Na falha de resposta, a resposta do servidor é simplesmente incorreta, podendo ocorrer dois tipos de falhas de respostas. Sendo, a falha de valor, onde um servidor simplesmente fornece a resposta errada a uma requisição. Já o outro tipo de falha de resposta é a falha de transição de estado, sendo que essa ocorre quando o servidor reage inesperadamente a uma requisição que chega.

As falhas arbitrárias, tidas por mais sérias são também conhecidas como falhas bizantinas. Essa pode acontecer quando um servidor está produzindo saídas que nunca deveria ter produzido, mas que não podem ser detectadas como incorretas, ou ainda, um servidor faltoso pode estar trabalhando maliciosamente em conjunto com outros servidores para produzir respostas erradas intencionalmente.

Como falhas são inevitáveis, várias técnicas são empregadas buscando garantir a disponibilidade destes serviços, mesmo em caso de erros conforme Pereira (2004). Segundo o referido autor, estas técnicas podem ser tanto no nível do software como do hardware: através de

hardwares redundantes tolerantes a falhas, a falha de um componente é compensada pela utilização de outro. Devido ao alto custo de tal solução, clusters são montados e configurados de modo a atingir um comportamento similar: a falha de um nó é compensada pela migração dos recursos comprometidos para outro nó operante.

Porém, é possível tornar um sistema tolerante a falhas por meio de mascaramento por redundância, sendo eles: redundância de informação, redundância de tempo e redundância física. Com redundância de informação, são adicionados bits extras para permitir recuperação de bits deteriorados.

Com redundância de tempo, uma ação é realizada e, então, se for preciso, ela é executada novamente. Se uma transação for abortada, ela pode ser refeita sem causar nenhum dano, sendo que tal redundância tem especial utilidade quando as falhas são transientes ou intermitentes.

Com redundância física, são adicionados equipamentos ou processos extras para possibilitar que o sistema como um todo tolere a perda ou o mau funcionamento de alguns componentes.

Outra forma de promover tolerância a falhas é a replicação, sendo que ter um grupo de processos idênticos nos permite mascarar um ou mais processos faltosos naquele grupo. Pode-se replicar processos para um processo vulnerável substituir um processo tolerante a falha. No entanto, há dois modos de abordar tal replicação: por meio de protocolos baseados em primários ou por meio de protocolos de escrita replicada.

Em casos de tolerância a falha a replicação baseada em primários aparece sob a forma de um protocolo de primário e backup. Nesse caso, um grupo de processos é organizado de modo hierárquico no qual um servidor primário coordena todas as operações de escrita. Na prática, o servidor primário é fixo, embora seu papel possa ser assumido por um dos backups, se for necessário. Na verdade, quando um servidor primário cai, os backups executam algum algoritmo de eleição para escolher um novo servidor primário.

Já em casos de protocolos de escrita replicada são usados sob a forma de replicação ativa, bem como por meio de protocolos baseados em quórum. Essas soluções correspondem a organizar o conjunto de processos idênticos em um grupo simples.

Tendo em vista possíveis falhas em um sistema, o presente trabalho apresenta como proposta um modelo de Sistema VoIP com Alta Disponibilidade de dados amparado por ferramentas de computação em Cluster.

4 VALIDAÇÃO

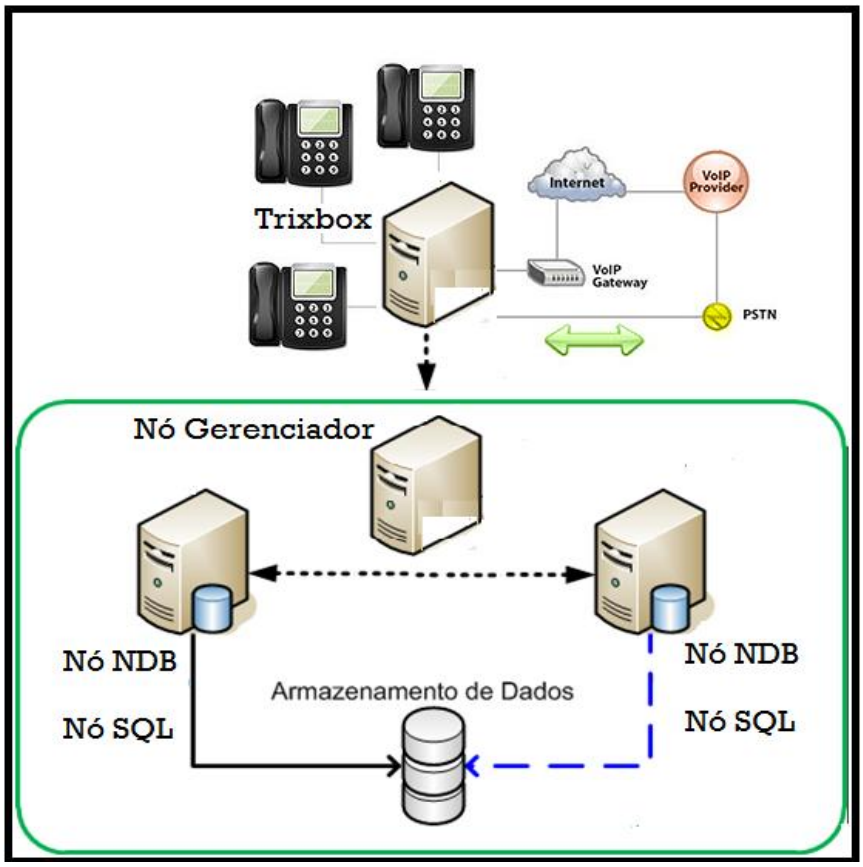
Neste capítulo é apresentada a metodologia de validação da proposta apresentada no presente trabalho, expondo sua implementação, descrevendo o ambiente experimental utilizado e apresentando os resultados obtidos. Porém, é importante ressaltar que, embora a proposta de um sistema de alta disponibilidade envolva o tratamento de diferentes falhas expostas no Capítulo 3, o trabalho experimental desenvolvido foca no problema de falha por queda, partindo do princípio de que ela é mais comum.

4.1 Implementação

A implementação de uma ambiente para validação da proposta apresentada no Capítulo 3 foi concebida com o intuito de demonstrar que é possível prover um aumento da disponibilidade de dados do Sistema VoIP, proporcionando, consequentemente, uma maior satisfação dos clientes e usuários do sistema. Em particular, cabe destacar que a proposta tem como foco tornar o sistema tolerante a falha de queda, agregando as funcionalidades do SGBD MySQL Cluster à Plataforma Trixbox.

Para tanto, foi montado um cluster com quatro máquinas virtuais (VMs), sendo a primeira composta pelo sistema VoIP (TRIXBOX), a segunda pelo nó gerenciador e as outras duas por um nó NDB e um nó SQL cada uma delas, ambas com o Sistema Operacional CentOS 5.5. A partir da replicação do banco de dados, é possível ter alta disponibilidade nos mesmos, sendo que, além da máquina gerenciadora, tem-se uma máquina primária e outra secundária sincronizando os dados. Portanto, se acontecer uma falha de queda na máquina primária, por exemplo, a gerenciadora dá o comando para a secundária assumir o serviço de banco de dados, sendo que os dados são obtidos através do sistema VoIP (TRIXBOX) como apresenta a Figura 4. Desta forma, caso aconteça alguma falha no *hardware*, o nó afetado será substituído por outro dando continuidade a demanda do serviço. Além disso, é importante ressaltar que o sistema aqui implementado é síncrono, sendo que os dados são atualizados à medida que são processados no banco de dados.

Figura 4 - Inclusão do MySQL Cluster na Plataforma Trixbox



Fonte: Dados primários

4.2 Ambiente experimental de avaliação

Para a realização de tal estudo, foi criado inicialmente, com a utilização da ferramenta Virtual Box (FONALITY), um ambiente virtual com o cenário de quatro máquinas virtuais, sendo uma máquina com o sistema VoIP com a plataforma TRIXBOX, tendo como Sistema Operacional padrão o CentOS 5.5 (32 Bits), uma com o nó gerenciador e duas máquinas de bancos de dados com o Sistema Operacional Ubuntu 10.04 (32 Bits). Para a primeira máquina foram disponibilizados 512 Megabytes (MB) de memória RAM e 10 gigabytes (GB) para armazenamento do sistema operacional. Para a segunda máquina foram disponibilizados 512 Megabytes (MB) de RAM e 10 gigabytes (GB) para armazenamento do sistema operacional. Já para a terceira e quarta máquina foram disponibilizados, para cada uma, 2048 MB de RAM e 50 GB para armazenamento de dados. As máquinas de bancos de dados foram replicadas e,

consequentemente, sincronizadas, através da criação de um MySQL Cluster com um nó gerenciador, dois nós de dados e dois nós SQL.

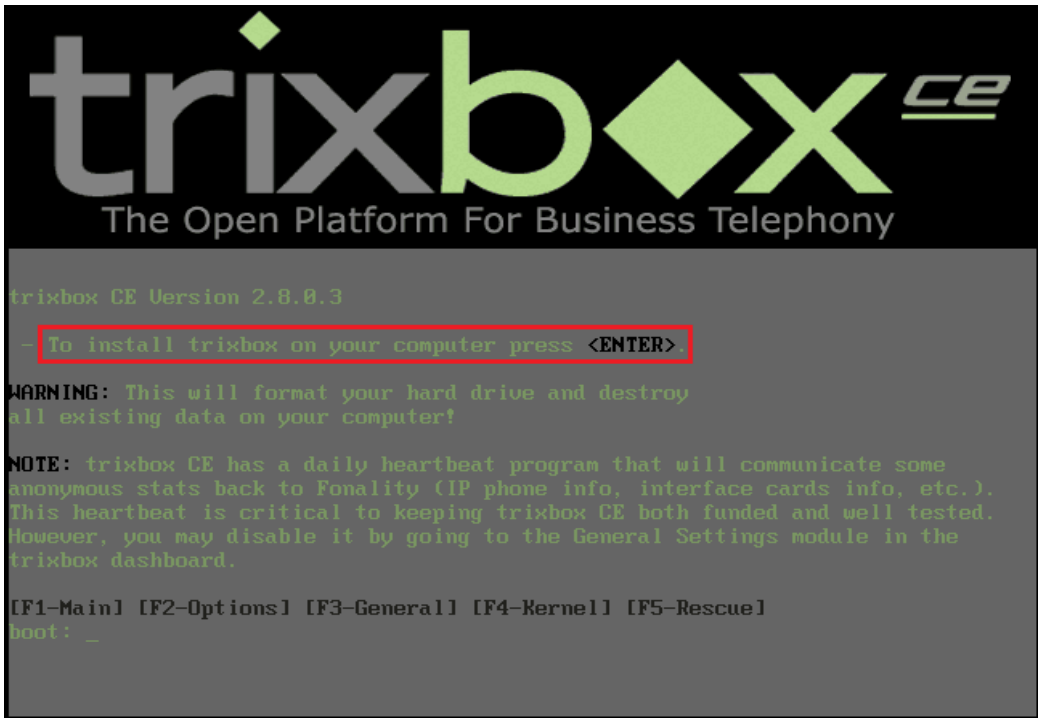
4.3 Processo de instalação e adaptação do Trixbox ao MySQL Cluster

Tal processo ocorreu da seguinte maneira: Primeiramente, foi instalado o Trixbox em uma máquina virtual, depois foi instalado o MySQL Cluster em três máquinas, e, após foi realizada a integração do Trixbox ao MySQL Cluster.

4.3.1 Instalação Trixbox

O Trixbox é uma distribuição que facilita a instalação e a configuração do Asterisk, que é um software que utiliza protocolos abertos como SIP, MGCP e IAX para realizar a sinalização das chamadas telefônicas na rede IP. Sendo que, a plataforma Trixbox pode ser instalada em apenas 20 minutos. Para iniciar sua instalação basta inserir o disco no computador, sendo que, depois é necessário apenas dar enter na tela inicial conforme figura 5 e seguir os passos do processo de instalação como, por exemplo: escolha de teclado, criação de usuário e senha.

Figura 5 - Instalação do Trixbox



Fonte: Dados primários

Após concluído o processo de instalação é exibida a tela onde será necessário efetuar o login para poder

4.3.2 Instalação MySQL Cluster

A instalação do MySQL Cluster foi feita seguindo o tutorial disponibilizado por Wargandi (2013). Porém, o binário utilizado foi o seguinte: mysql-cluster-gpl-7.2.10-Linux2.6-x86_64.tar.gz.

Primeiramente em uma das máquinas, foi instalado o nó de gerenciamento do Cluster, sendo que para isso é essencial a criação do arquivo config.ini como mostra a figura 6:

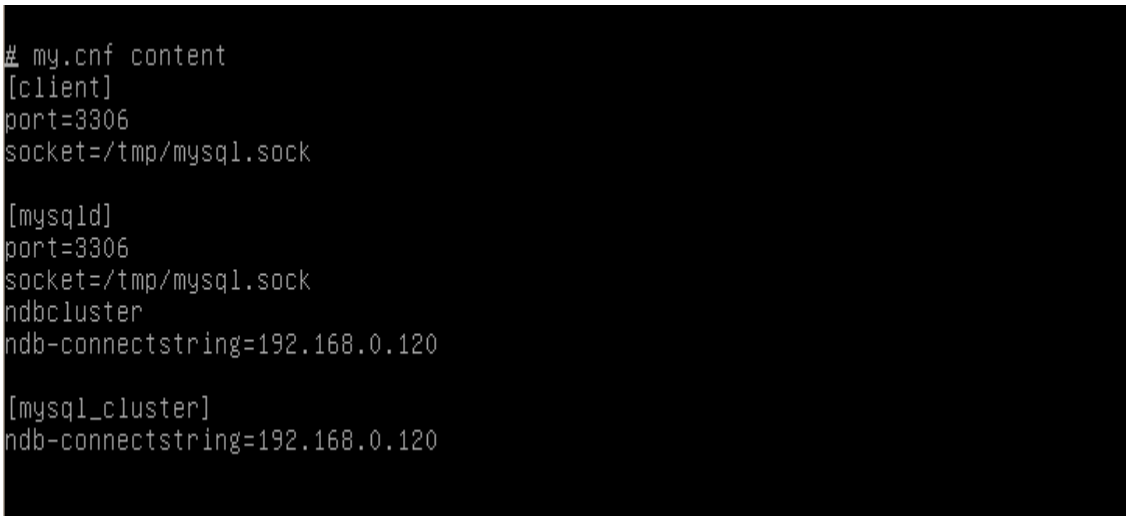
Figura 6 - Arquivo config.ini

```
[ndb_mgmd]
NodeId=1
HostName=192.168.0.120
DataDir=/var/lib/mysql-cluster

[ndbd default]
DataDir=/var/lib/mysql-cluster
NoOfReplicas=2
MaxNoOfConcurrentOperations=32000
MaxNoOfAttributes = 10000
MaxNoOfOrderedIndexes=512
DataMemory=1G
IndexMemory=500M

[ndbd]
NodeId=3
HostName=192.168.0.121

[ndbd]
NodeId=4
HostName=192.168.0.122

[mysqld]
HostName=192.168.0.121

[mysqld]
HostName=192.168.0.122
```

Fonte: Dados primários

Depois de instalado o nó de gerenciamento, em outra máquina foi instalado o nó NDB, sendo que para isso, é necessário criar o arquivo my.cnf conforme apresentado na figura 7:

Figura 7 - Arquivo my.cnf



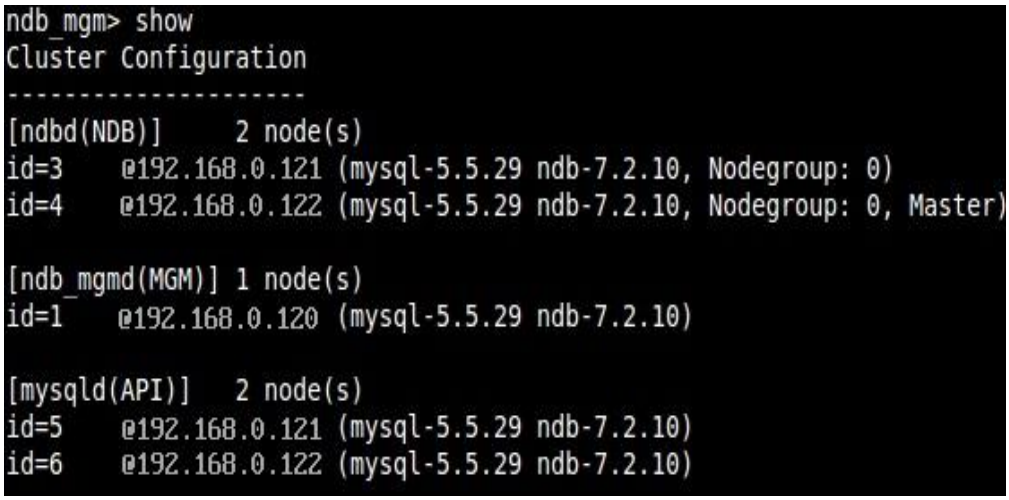
Fonte: Dados primários

Ainda na mesma máquina, após instalado o nó NDB, foi feita a instalação do nó SQL, que depende do arquivo my.cnf criado na instalação anterior.

O processo de instalação do nó NDB e nó SQL é realizado, novamente, em outra máquina para que, posteriormente, possa ser realizada a replicação dos dados.

Obtendo-se êxito no processo de instalação dos nós e estes conectados visualiza-se a figura 8.

Figura 8 - Nós conectados

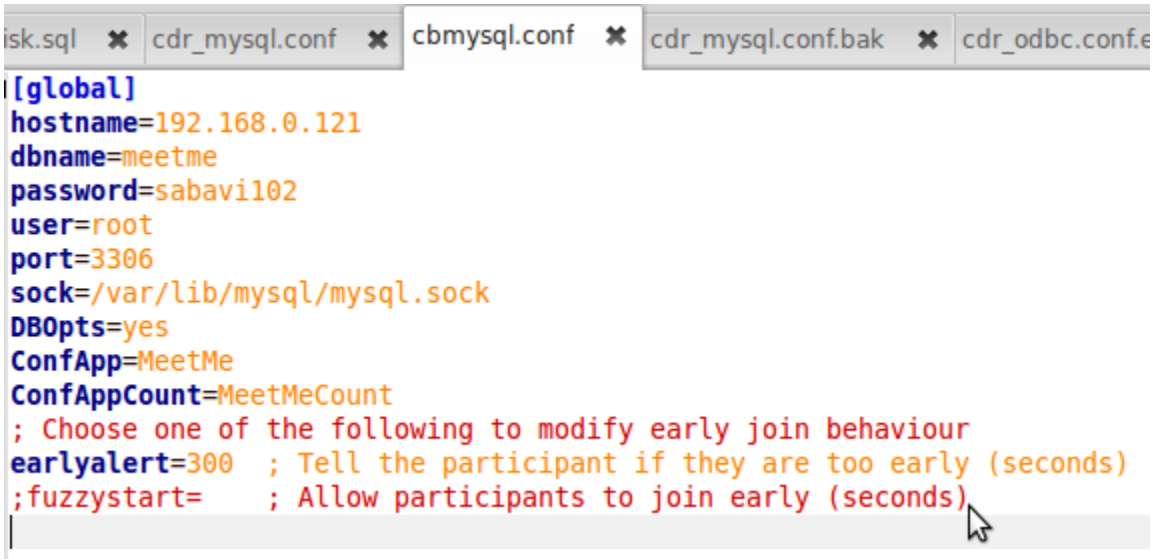


Fonte: Dados primários

4.3.3 Integraçao Trixbox e MySQL Cluster

Algumas alterações foram necessárias para a integração de dois serviços, ou seja, o Trixbox acessando o ambiente MySQL Cluster com a replicação dos dados em todos os servidores que compõem o ambiente. Para tanto foram realizadas algumas alterações nos arquivos de configuração de conexão com o banco de dados que o ambiente do Trixbox utiliza como pode ser visto na figura 9.

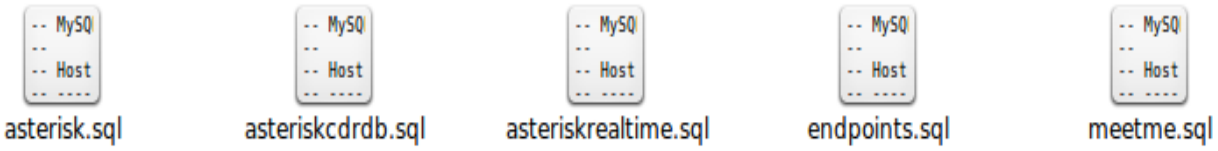
Figura 9 - Integração Trixbox e MySQL Cluster



Fonte: Dados primários

Após as alterações de configuração, foi realizado a exportação da estrutura da base de dados do Trixbox e criado no ambiente MySQL Cluster. Desta forma, a base de dados foi replicada para o segundo nó que compõe o cluster, conforme mostra figura 10.

Figura 10 – Exportação de base de dados



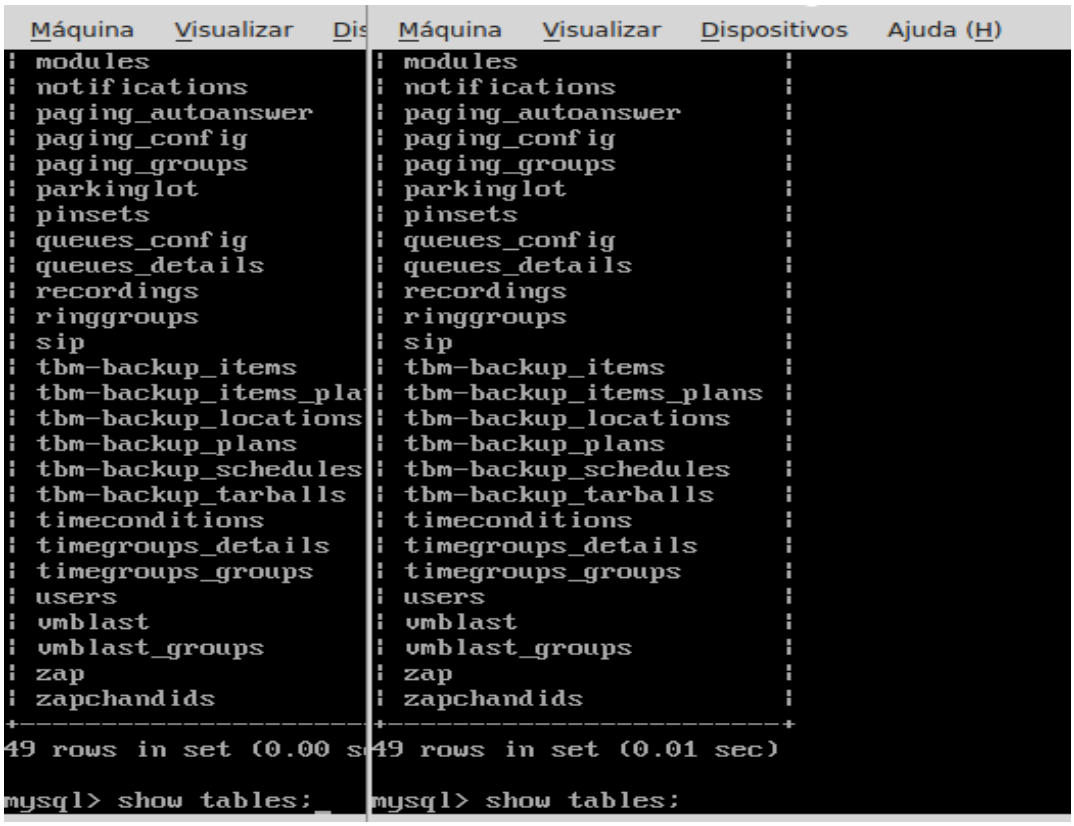
Fonte: Dados primários

A partir desse momento, todos os registros salvos através do Trixbox são armazenados na base de dados remota, e, esta por sua vez, replicada para o outro servidor.

4.4 Apresentação dos resultados

Após a criação do Cluster, foram exportadas as tabelas da base de dados do Asterisk, que, posteriormente, foram inseridas em um dos nós de dados, sendo replicadas automaticamente para o outro nó de dados como pode-se observar na figura 11:

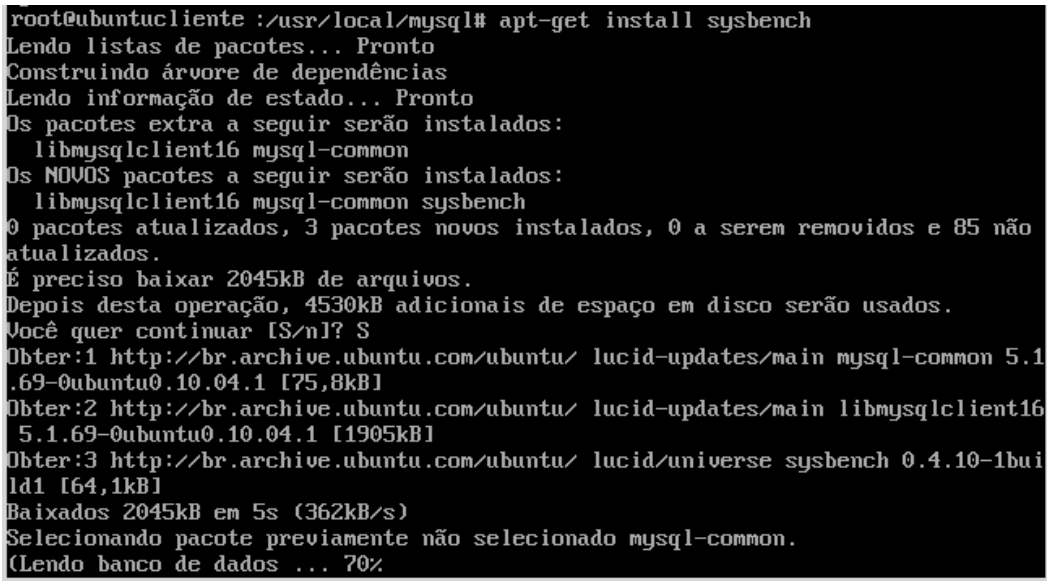
Figura 11 - Replicação de Dados



Fonte: Dados primários

Visto que a proposta do presente trabalho era implementar um Sistema VoIP com Alta Disponibilidade, após a criação do Cluster e replicação dos dados contidos no Trixbox, foi instalada a ferramenta Sysbench (Howtoforge, 2013) conforme figura 12 em uma das máquinas do banco de dados composta pelos nós *Data Node* e SQL *Node*.

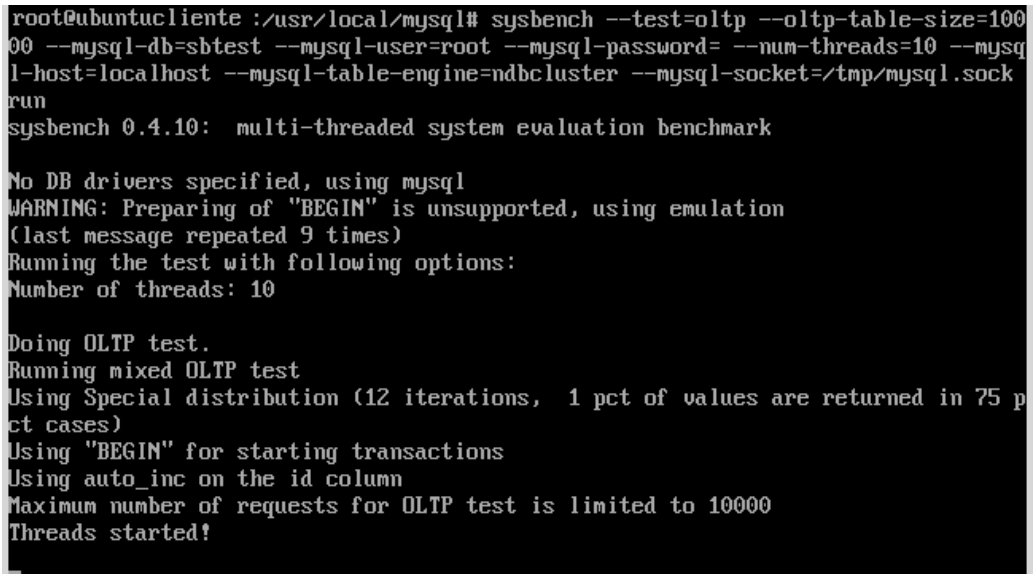
Figura 12 - Instalação do Sysbench



Fonte: Dados primários

Após a instalação do Sysbench, foram realizados testes usando como bechmark um processo de inserção de 1000 registros para avaliar a performance do MySQL Cluster, conforme figura 13.

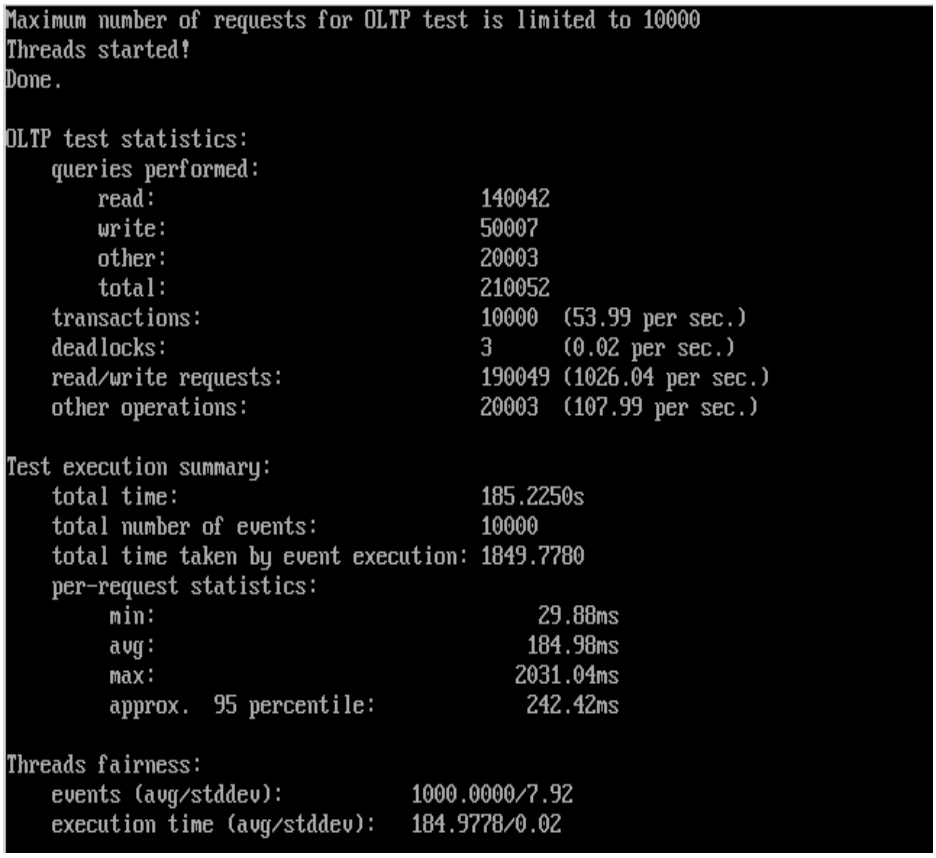
Figura 13 - Benchmark do Sistema



Fonte: Dados primários

O comando da figura 13 foi executado com os dois nodos de dados replicados em pleno funcionamento, obtendo-se como resultado a figura 14:

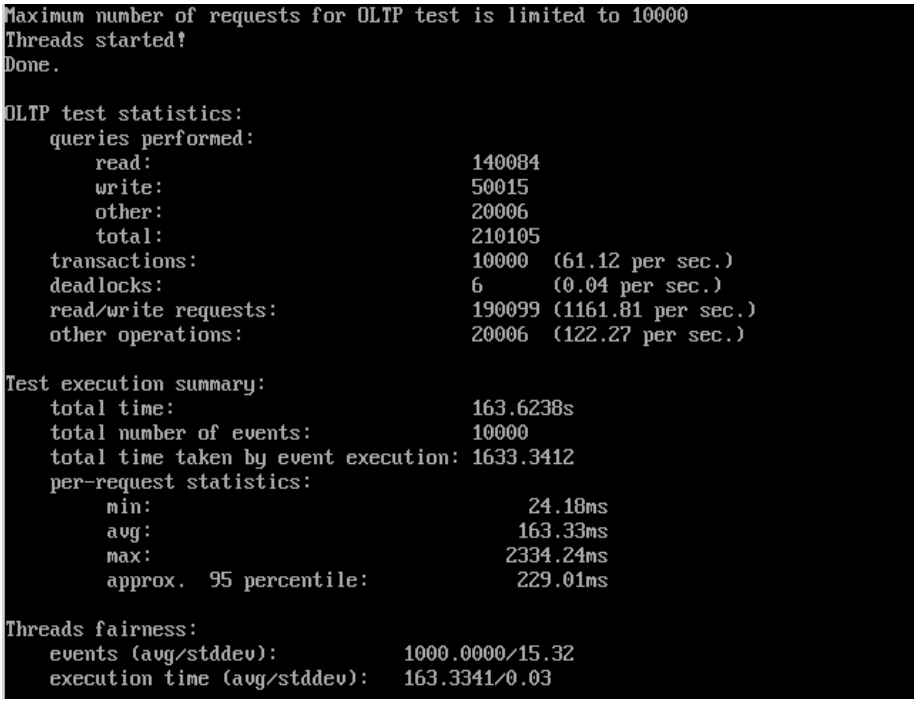
Figura 14 - Benchmark de dois nós



Fonte: Dados primários

Posteriormente foi simulada uma “Falha por queda” em um dos nós de dados e executado o benchmark em apenas um nó, conforme pode-se visualizar na figura 15.

Figura 15 - Benchmark de apenas um nó



```
Maximum number of requests for OLTP test is limited to 10000
Threads started!
Done.

OLTP test statistics:
    queries performed:
        read:                            140084
        write:                           50015
        other:                           20006
        total:                           210105
    transactions:                        10000  (61.12 per sec.)
    deadlocks:                           6      (0.04 per sec.)
    read/write requests:                 190099 (1161.81 per sec.)
    other operations:                    20006  (122.27 per sec.)

Test execution summary:
    total time:                          163.6238s
    total number of events:              10000
    total time taken by event execution: 1633.3412
    per-request statistics:
         min:                                  24.18ms
         avg:                                 163.33ms
         max:                                2334.24ms
         approx.  95 percentile:              229.01ms

Threads fairness:
    events (avg/stddev):           1000.0000/15.32
    execution time (avg/stddev):   163.3341/0.03
```

Fonte: Dados primários

A partir da análise dos resultados obtidos com o referido benchmark chegou-se à seguinte tabela:

Tabela 2 - Resultados do Benchmark

	DOIS NÓS	UM NÓ
Transações	53,99 por segundo	61,12 por segundo
Tempo médio p/ requisição	184,98 ms	163,33 ms

Fonte: Dados primários

Conforme os números apresentados pela tabela 2, o desempenho do sistema contendo dois nós ativos realizando replicação consome mais tempo do que o sistema que utiliza apenas um nó.

Além do teste de performance de sistema, foi realizado um novo teste de desligamento proposital de um dos nós de dados, visando verificar a consistência do sistema ao se reestabelecer o serviço. A partir deste teste foi possível comprovar que a replicação dos dados do

Sistema VoIP com a ferramenta MySQL Cluster efetivamente permite transformar o Trixbox em uma plataforma com alta disponibilidade de dados.

5 CONCLUSÃO

Por meio do presente estudo foi possível constatar que através de uma estrutura de MySQL Cluster pode-se sustentar a distribuição de dados através de réplicas dos nós de dados, o que resulta assim em um sistema de alta disponibilidade de dados aplicável a uma sistema VoIP. Além disso, a distribuição e replicação dos dados torna o sistema tolerante à falha de queda, tornando-o mais consistente, e consequentemente, mais confiável.

Com a implementação de tal proposta pode-se minimizar a perda ou danos dos dados de em consequência de uma falha de queda. Pois, verificou-se que após uma queda repentina, o sistema é capaz de reestabelecer o serviço automaticamente, característica básica de uma sistema com alta disponibilidade.

Os testes realizados também deixaram evidente que a performance do sistema contendo dois nós ativos sincronizados, realizando replicação, consome mais tempo do que o sistema que utiliza apenas um nó, embora esta diferença, no ambiente experimental construído para a validação da proposta, tenha sido da ordem de dezenas de milisegundos.

Como trabalhos futuros, pretende-se ampliar os experimentos simulando outros tipos de falhas e utilizando novos *benchmarks* que incorporem características mais específicas da aplicação, bem como avaliar o comportamento do sistema em diferentes cenários de interligação entre as instâncias (links com diferentes latências e vazão). Complementarmente, existe a intenção de avaliar a solução em um ambiente com o sistema Trixbox distribuído, de modo a agregar alta disponibilidade não apenas no banco de dados do sistema mas também na aplicação. Assim, se houver uma falha em uma das instâncias da aplicação, o sistema não precisaria parar momentaneamente até que fosse reestabelecido, evitando a indisponibilidade de comunicação na empresa.

REFERÊNCIAS

AHRENDT, Chris. **Construindo uma infraestrutura de eventos altamente disponível do WebSphere Business Events.** 2010. Disponível em: <http://www.ibm.com/developerworks/br/websphere/library/techarticles/1006_ahrendt/1006_ahrendt.html>. Acesso em: 26 jul. 2013.

COULOURIS, George. **Sistemas Distribuídos: Conceitos e Projeto.** Porto Alegre: Bookman, 4 Edição, 2007.

ELMARSI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados.** São Paulo: Pearson, 6 Edição, 2011.

FONALITY. **Fonality trixbox | The Open Platform for Business Telephony**. Disponível em: <http://www.trixbox.com>. Acesso em: 27 jul. 2013.

HOWTOFORGE. **Sysbench**. Disponível em: <http://www.howtoforge.com/how-to-benchmark-your-system-cpu-file-io-mysql-with-sysbench>. Acesso em: 15 jul. 2013.

KUROSE, James F,; ROSS, Keith W. **Redes de computadores e a Internet : uma abordagem top-down**. São Paulo: Pearson Addison Wesley, 2006.

ORACLE CORPORATION. **MySQL Cluster.** Disponível em: <http://www.mysql.com/products/cluster/>. Acesso em: 15 jul. 2013.

PASIN, Marcia. **Réplicas para Alta Disponibilidade em Arquiteturas Orientadas a Componentes com Suporte de Comunicação de Grupo.** Porto Alegre, 2003. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/4165/000397539.pdf?sequence=1>. Acesso em: 19 jun. 2013.

PEDROSO, Carlos Marcelo et. al. **Análise da Evolução de Características de Tráfego VoIP.** In: XXVI Simpósio Brasileiro de Telecomunicações. Rio de Janeiro, 2008. Disponível em: <http://www.eletrica.ufpr.br/pedroso/Artigos/sbrt2008-final.pdf>. Acesso em: 2 jul. 2013.

PEDROSO, Carlos Marcelo; CALDEIRA, Jefferson; FONSECA, Keiko. **Caracterização de tráfego VoIP.** Curitiba, 2006. Disponível em: <http://www.eletrica.ufpr.br/pedroso/Artigos/ModelagemVoIPv3.pdf>. Acesso em: 19 jun. 2013.

PEREIRA, Nélio Alves. **Serviços de Pertinência para Clusters de Alta Disponibilidade.** São Paulo, 2004. Disponível em: <renderfarm.googlecode.com/files/dissertacao2.pdf>. Acesso em: 15 jul. 2013.

SITOLINO, Claudio Luis. **VoIP: Um Estudo Experimental.** Porto Alegre, 2001. Disponível em: <https://www.repositorioceme.ufrgs.br/bitstream/handle/10183/3182/000333405.pdf?sequence=1>. Acesso em: 14 jul. 2013.

TANEMBAUM, Andrew S; STEEN, Maarten V. **Sistemas Distribuídos: Princípios e Paradigmas.** São Paulo: Pearson, 2 Edição, 2007.

WARGANDI, Bromo K. A. My SQL Cluster: parte 1, 2 e 3. Disponível em: <http://bromokun.wordpress.com/2012/07/07/installing-mysql-cluster-management-node-in-ubuntu-server-v-12-04/>. Acesso: 28 Mai. 2013.