UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E DA TECNOLOGIA
CIÊNCIA DA COMPUTAÇÃO

BRUNO EMER

Implementação de alta disponibilidade em uma empresa prestadora de serviços para Internet

André Luis Martinotto
Orientador

Caxias do Sul
Abril de 2016

Implementação de alta disponibilidade em uma empresa prestadora de serviços para Internet

por

Bruno Emer

Projeto de Diplomação submetido ao curso de Bacharelado em Ciência da Computação do Centro de Ciências Exatas e da Tecnologia da Universidade de Caxias do Sul, como requisito obrigatório para graduação.

Projeto de Diplomação

Orientador: André Luis Martinotto
Banca examinadora:

Maria de Fatima Webber do Prado Lima
CCTI/UCS
Ricardo Vargas Dorneles
CCTI/UCS

Projeto de Diplomação apresentado em
x de x de 2016

Daniel Luís Notari
Coordenador

SUMÁRIO

LISTA DE ACRÔNIMOS

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Palavras-chave: .

1 INTRODUÇÃO

O crescente avanço tecnológico e o desenvolvimento da internet, provocou um aumento no número de aplicações ou serviços que dependem da infraestrutura de TI. Além disso, percebe-se um aumento significativo no número de operações e negócios *on-line* que são realizados, tanto por organizações públicas ou privadas, quanto por grande parte da sociedade.

Desta forma, a sociedade está cada vez mais dependente da tecnologia, computadores e sistemas. De fato, pode-se observar sistemas computacionais desde em uma farmácia, até em uma grande indústria. Sendo assim, a estabilidade e confiabilidade destes sistemas tem grande importância em nosso dia-a-dia, pois um número significante de atividades cotidianas dependem destes.

Uma interrupção imprevista em um ambiente computacional poderá causar um prejuízo financeiro para a empresa que fornece o serviço, além de interferir na vida de pessoas que dependem de forma direta ou indireta deste serviço. Essa interrupção terá maior relevância para corporações cujo o serviço ou produto final é fornecido através da internet, como por exemplo, comércio eletrônico, *web sites*, sistemas corporativos, entre outros. Em um ambiente extremo, pode-se imaginar o caos e o possível risco de perda de vidas que ocorreria em caso de uma falha em algum sistema de controle aéreo (COSTA, 2009).

Para essas empresas um plano de contingência é fundamental para garantir uma boa qualidade de serviço, otimizar o desempenho das atividades, bem como para uma prevenção de falhas e uma recuperação rápida caso essas ocorram (COSTA, 2009). De fato, hoje em dia a confiabilidade de um serviço ou de um sistema é um grande diferencial para a empresa fornecedora deste serviço, sendo que a alta disponibilidade é fundamental para atingir esse objetivo.

A alta disponibilidade consiste em manter um sistema disponível por meio da tolerância a falhas, isto é, utilizando mecanismos que fazem a detecção, mascaramento e a recuperação de falhas, sendo que esses mecanismos podem ser implementados a nível de *software* ou de *hardware* (REIS, 2009). Para que um sistema seja altamente disponível ele deve ser tolerante a falhas, ou seja, utilizando redundância. Redundância é feita através da duplicação de componentes, caso ocorra falha em um dos componentes evita-se a interrupção do sistema (BATISTA, 2007).

Neste trabalho será realizado um estudo sobre a implementação de um sistema de alta disponibilidade em uma empresa de hospedagens. Essa oferece serviços pela internet, como por exemplo hospedagens de sites, *e-mail*, sistemas de gestão, *e-mail marketing*, entre outros. A empresa possui aproximadamente 55 servidores físicos e virtuais, e aproximadamente 9000 clientes, sendo que em períodos de pico atende em torno de 1000 requisições por segundo.

Atualmente, essa empresa possui redundância de conexões de acesso a internet, refrigeração e energia, com *nobreaks* e geradores. Porém, essa empresa não possui nenhuma redundância dos serviços que estão sendo executados nos servidores. Desta forma, caso ocorra uma falha de software ou hardware os serviços ficarão indisponíveis. Neste trabalho será realizada uma análise dos serviços oferecidos pela empresa, sendo que mecanismos de alta disponibilidade serão desenvolvidos para os serviços mais críticos. Para a redução dos custos serão utilizadas ferramentas gratuitas e de código aberto.

1.1 Objetivos

Atualmente a empresa não possui nenhuma solução de alta disponibilidade para seus serviços críticos. Desta forma, neste trabalho será desenvolvida uma solução de alta disponibilidade para estes serviços, sendo que essa solução será baseada no uso de ferramentas de código aberto e de baixo custo.

Para que o objetivo geral seja atendido os seguintes objetivos específicos deverão ser realizados:

- Identificar os serviços críticos a serem integrados ao ambiente de alta disponibilidade;
- Definir as ferramentas a serem utilizadas para implementar tolerância a falhas;
- Realizar testes para a validação do sistema de alta disponibilidade que foi desenvolvido.

2 ALTA DISPONIBILIDADE

2.1 Definição da alta disponibilidade

Alta disponibilidade é bastante conhecida, sendo cada vez mais empregada nos ambientes computacionais. O objetivo de promover alta disponibilidade resume-se em um serviço estar sempre a disposição quando o cliente solicitar ou acessar (COSTA, 2009). Pode-se definir alta disponibilidade como a redundância de *hardware* ou *software* para que o serviço fique mais tempo disponível. Quanto maior for a disponibilidade desejada maior deverá ser a redundância no ambiente, assim reduzindo os pontos únicos de falha, em inglês *Single Point Of Failure* (SPOF).

A alta disponibilidade está diretamente relacionada a confiabilidade, disponibilidade, dependabilidade e tolerância a falhas.

A confiabilidade, é o mais importante atributo, transmite a ideia de continuidade de serviço (PANKAJ, 1994). Confiabilidade refere-se a probabilidade de um serviço funcionar corretamente durante um dado intervalo de tempo. Já a disponibilidade é a probabilidade de um serviço estar operacional no instante em que for solicitado (COSTA, 2009).

Por sua vez dependabilidade indica a qualidade do serviço fornecido e a confiança depositada nele. A dependabilidade envolve vários atributos como confiabilidade, disponibilidade, segurança, entre outros. Os atributos mais relevantes são confiabilidade e disponibilidade, definidos anteriormente (WEBER, 2002).

A tolerância a falhas fornece disponibilidade de um serviço utilizando mecanismos capazes de detectar, mascarar e recuperar falhas, e seu objetivo é alcançar a dependabilidade, assim indicando uma boa qualidade de serviço (COSTA, 2009). Uma das principais palavras-chave da alta disponibilidade é a tolerância a falhas, que será melhor detalhada na próxima seção.

2.2 Tolerância a falhas

Sabe-se que o *hardware* tende a falhar por isso utiliza-se métodos como prevenção de falhas e tolerância a falhas. A abordagem prevenção de falhas melhora a disponibilidade e a confiabilidade de um serviço porém, não resolverá todas as possíveis falhas. Sendo assim, a tolerância a falhas fornece disponibilidade de um serviço mesmo com presença de falhas (PANKAJ, 1994). O objetivo da tolerância a falhas é aumentar a disponibilidade de um sistema, isto é, aumentar o tempo que os serviços fornecidos aos clientes ou usuários ficam disponíveis. Um sistema é dito tolerante a falhas se ele pode mascarar a presença de falhas ou recuperar-se de uma falha, frenquentemente a tolerância a falhas é implementada utilizando redundância

detalhada na próxima seção.

A tolerancia a falhas pode ser dividida em duas classes:

- Mascaramento
- Detecção, localização e reconfiguração

Na primeira, mascaramento, as falhas são tratadas na origem e manifesta-se na forma de erro. Um exemplo são os códigos de correção de erros, em inglês *error correction code* (ECC), utilizados em memórias para detecção e correção de erros. Na segunda, geralmente necessita de menor redundância, e consiste em detectar, localizar e reconfigurar o *software* ou *hardware* e por sim resolver a falha (WEBER, 2002).

–Colocar detalhado fases tolerancia a falhas?: detecção, .. recuperação de erros? detalhar? Pag 39 pankaj1994

2.3 Redundância

Redundância pode ser feita através da replicação de componentes, para garantir o mascaramento de falhas. Na prática se um componente falhar, ele deve ser reparado ou substituido por um novo sem que haja uma interrupção no serviço. Também pode ser através do envio de sinais ou *bits* de controle junto aos dados, servindo assim para detecção de erros e até para correção (WEBER, 2002).

Segundo (NøRVåG, 2000) existem quatro tipos diferentes de redundância que são:

- *Hardware*: utiliza-se replicação de componentes, sendo que caso um falhe outro possa assumir seu lugar. Para fazer a detecção de erros a saída de cada componente é constantemente monitorada e comparada à saída de outros componentes;
- Informação: quando uma informação extra é enviada ou armazenada para possibilitar a detecção e correção de erros;
- *Software*: são todos os *softwares* ou instruções utilizadas para suporte a tolerância a falhas. Podem ser implementados de várias formas, desde um processo que fica monitorando o serviço para verificar se ele esta funcionando corretamente, até um programa que efetua uma verificação nos resultados para saber se está operando da forma desejada;
- Tempo: esta é feita através da execução de instruções várias vezes no mesmo componente, assim assim detectando falho caso ocorra. Necessita tempo adicional, e é utilizado onde o tempo não crítico. Por exemplo um cão de guarda (*watchdog timer*), ele recebe um sinal do programa ou serviço monitorado e caso este sinal não seja recebido, devido alguma falha, o *watchdog* irá fazer alguma ação de reinicialização do serviço. Diferentemente de redundância de *hardware* e informação ela não requer um *hardware* extra para sua implementação (COSTA, 2009).

COLOCAR EXEMPLOS

Tabela 2.1: Níveis de alta disponibilidade

Nível	Uptime	Downtime por ano	Downtime por semana
1	90%	36.5 dias	16 horas e 51 minutos
2	98%	7.3 dias	3 horas e 22 minutos
3	99%	3.65 dias	1 hora e 41 minutos
4	99.8%	17 horas e 30 minutos	20 minutos e 10 segundos
5	99.9%	8 horas e 45 minutos	10 minutos e 5 segundos
6	99.99%	52.5 minutos	1 minuto
7	99.999%	5.25 minutos	6 segundos
8	99.9999%	31.5 minutos	0.6 segundos

Tabela 2.2: Exemplos de sistemas

Nível	Nome
1	computadores pessoais
3	sistemas de acesso
5	provedores de internet
6	CPD, sistemas de negócios
7	sistemas de telefonia; sistemas de saúde; sistemas bancários
8	sistemas de defesa militar

2.4 Cálculo da alta disponibilidade

Um ponto importante sobre alta diponibilidade é como medi-la. Para isso são utilizados os valores de *uptime* e *downtime*, que são respectivamente o tempo que os serviços esta funcionando normalmente e o tempo que não estão funcionando. Outra forma de expressar a alta disponibilidade é pela quantidade de "noves", isto é, se um serviço possui 4 noves de disponibilidade este possui uma disponibilidade de 99,99% (FILHO, 2004). A tabela 2.1 possui alguns níveis de disponibilidade enumerados. Já a Tabela 2.2 possui alguns exemplos de serviços relacionados ao nível de disponibilidade. DETALHAR TABELA

Podemos calcular a disponibilidade através da equação

$$d = \frac{MTBF}{(MTBF + MTTR)} \tag{2.1}$$

onde d é a porcentagem de disponibilidade. O *Mean Time Between Failures* ($MTBF$) é o tempo médio entre falhas, correspondente ao tempo médio entre as paradas dos serviços. E o *Mean Time To Repair* ($MTTR$) é o tempo médio de recuperação, isto é, o tempo entre a queda e a recuperação de um serviço (GONçALVES, 2009).

A alta disponibilidade é um dos principais fatores para garantir a confiabilidade de clientes ou usuários, principalmente para empresa que fornece serviços *on-line*. Por isso existe o *Service Level Agreement*(SLA), acordo de nível de serviço, que garante que o serviço fornecido atenda as expectativas (SMITH, 2010). DETALHAR, ONDE?

3 VIRTUALIZAÇÃO

O conceito virtualização surgiu na década de 60, sendo que um dos principais motivos foi a necessidade de um grande servidor, conhecido como *mainframe*, executar uma variedade de *softwares*. Isso ocorreu pois cada *mainframe* possuía um próprio sistema operacional necessitando assim a criação de máquinas virtuais.

Segundo (CARISSIMI, 2008) falar sobre vantagens virtualizacao

Virtualização defini-se como uma camada entre o *hardware* e o sistema operacional assim possibilitando a divisão e proteção dos recursos físicos (SMITH; NAIR, 2005).

REFERÊNCIAS

BATISTA, A. C. **Estudo teórico sobre cluster linux**. 2007. Pós-Graduação(Administração em Redes Linux) — Universidade Federal de Lavras, Minas Gerais.

CARISSIMI, A. Virtualização: da teoria a soluções. In: **Minicursos do Simpósio Brasileiro de Redes de Computadores**. Porto Alegre: XXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2008.

COSTA, H. L. A. **Alta disponibilidade e balanceamento de carga para melhoria de sistemas computacionais críticos usando software livre**: um estudo de caso. 2009. Pós-Graduação em Ciência da Computação — Universidade Federal de Viçosa, Minas Gerais.

FILHO, N. A. P. **Serviço de pertinência para clusters de alta disponibilidade**. 2004. Dissertação para Mestrado em Ciência da Computação — Universidade de São Paulo, São Paulo.

GONçALVES, E. M. **Implementação de Alta disponibilidade em máquinas virtuais utilizando Software Livre**. 2009. Trabalho de Conclusão (Curso de Engenharia da Computação) — Faculdade de Tecnologia e Ciências Sociais Aplicadas, Brasilia.

NøRVåG, K. **An Introduction to Fault-Tolerant Systems**. 2000. IDI Technical Report 6/99 — Norwegian University of Science and Technology, Trondheim, Noruega.

PANKAJ, J. **Fault tolerance in distributed system**. Nova Jérsei, Estados Unidos: P T R Prentice Hall, 1994.

REIS, W. S. dos. **Virtualização de serviços baseado em contêineres**: uma proposta para alta disponibilidade de serviços em redes linux de pequeno porte. 2009. Monografia Pós-Graduação(Administração em Redes Linux) — Apresentada ao Departamento de Ciência da Computação, Minas Gerais.

SMITH, J. E.; NAIR, R. The architecture of virtual machines. **IEEE Computer**, [S.l.], v.38, p.32–38, 2005.

SMITH, R. **Gerenciamento de Nível de Serviço**. <Disponível em: `http://blogs.technet.com/b/ronaldosjr/archive/2010/05/25/gerenciamento-de-n-237-vel-de-servi-231-o.aspx/`>. Acesso em 25 de março de 2016.

WEBER, T. S. **Um roteiro para exploração dos conceitos básicos de tolerância a falhas**. 2002. Curso de Especialização em Redes e Sistemas Distribuídos — UFRGS, Rio Grande do Sul.