PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGICAS.

LUIS PAULO GONÇALVES PIRES

ALTA DISPONIBILIDADE: UMA ABORDAGEM COM DNS E PROXY REVERSO EM MULTI-CLOUD

CAMPINAS

2016

LUIS PAULO GONÇALVES PIRES

ALTA DISPONIBILIDADE: UMA ABORDAGEM COM DNS E PROXY REVERSO EM MULTI-CLOUD

Dissertação apresentada ao Programa de Pós Graduação Stricto Sensu em Engenharia Elétrica do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Eric Alberto de Mello Fagotto

PUC-CAMPINAS

2016

Ficha Catalográfica
Elaborada pelo Sistema de Bibliotecas e
Informação - SBI - PUC-Campinas

t303.4833
P667a

Pires, Luis Paulo Gonçalves.
Alta disponibilidade: uma abordagem com DNS e Proxy reverso em Multi-Cloud / Luis Paulo Gonçalves Pires. - Campinas: PUC-Campinas, 2016.
85p.

Orientador: Eric Alberto de Mello Fagotto.
Dissertação (mestrado) – Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Pós-Graduação em Engenharia Elétrica.
Inclui bibliografia.

1. Tecnologia - Serviços de informação. 2. Serviços da Web. 3. Computação em nuvem. 4. Nomes de domínio na Internet. 5. Redes de informação. 6. Processamento eletrônico de dados. I. Fagotto, Eric Alberto de Mello. II. Pontifícia Universidade Católica de Campinas. Centro de Ciências Exatas, Ambientais e de Tecnologias. Pós-Graduação em Engenharia Elétrica. III. Título.

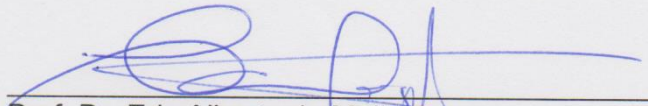
20.ed. CDD – t303.4833

LUIS PAULO GONÇALVES PIRES

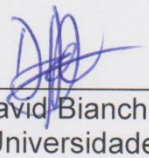
ALTA DISPONIBILIDADE: UMA ABORDAGEM COM DNS E PROXY REVERSO EM MULTI-CLOUD

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito parcial para obtenção do título de Mestre em Gestão de Redes de Telecomunicações.
Área de Concentração: Engenharia Elétrica.
Orientador: Prof. Dr. Eric Alberto de Mello Fagotto

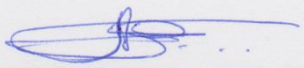
Dissertação defendida e aprovada em 15 de dezembro de 2016 pela Comissão Examinadora constituída dos seguintes professores:



Prof. Dr. Eric Alberto de Mello Fagotto
Orientador da Dissertação e Presidente da Comissão Examinadora
Pontifícia Universidade Católica de Campinas



Prof. Dr. David Bianchini
Pontifícia Universidade Católica de Campinas



Prof. Dr. Luiz Henrique Bonani do Nascimento
Universidade Federal do ABC

Dedico este trabalho primeiramente a Deus e a minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida.

AGRADECIMENTOS

A Deus,
Pelas oportunidades, coragem e determinação a mim concedidas.

Ao Prof. Dr. Eric Alberto de Mello Fagotto.
Pela paciência na orientação e incentivo que tornaram possível a realização desse trabalho.

À Pontifícia Universidade Católica de Campinas.
Pela concessão integral da bolsa no Programa de Mestrado.

Aos professores Marcelo Abbade, Omar Branquinho, David Bianchini, Eric Alberto e Marcius Fabius.
Pelas aulas ministradas.

Aos colegas de turma.
Pelo apoio, convívio, companheirismo e troca de conhecimentos.

“Só há duas maneiras de viver a vida: a primeira é vivê-la como se os milagres não existissem. A segunda é vivê-la como se tudo fosse milagre”.
Albert Einstein

RESUMO

PIRES, Luis Paulo Gonçalves. **Alta Disponibilidade: Uma abordagem com DNS e *Proxy* Reverso em *Multi-Cloud***, 2016. Dissertação (Mestrado Profissional em Gestão de Redes de Telecomunicações) - Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Campinas, 2016.

A despeito de haver considerável entusiasmo quanto à migração de *data-centers on-primese* para serviços de *Cloud Computing*, ainda existe certo receio no que se refere à disponibilidade destes mesmos serviços. Isso se deve, por exemplo, a incidentes históricos como o ocorrido em 2011, quando uma falha nos servidores da Amazon fez com que sites de vários de seus clientes ficassem fora do ar por quase 36 horas. Em vista disso, torna-se necessário desenvolver estratégias para garantir a disponibilidade oferecida pelos provedores. No presente trabalho, descreve-se uma solução que implementa alta disponibilidade em ambientes *Multi-Cloud*, mediante a distribuição de acesso por DNS e a utilização de *proxy* reverso. Realizou-se também uma análise financeira, levando-se em conta valores de mercado em serviços de *Cloud Computing*, o que mostrou que a solução proposta pode ser mesmo vantajosa com a relação à solução tradicional. Especificamente, um sistema *Multi-Cloud*, composto por duas *Clouds* com disponibilidade de 99,90%, que provê disponibilidade total de 99,999%, custa 34% menos do que uma única *Cloud* com disponibilidade de 99,95%. Os resultados de simulação, obtidos em ambiente virtualizado, utilizando-se duas *Clouds*, com disponibilidades de 99,49% e 99,43%, alcançaram disponibilidade 99,9971%. Desta forma, utilizando-se sistemas *Multi-Cloud* é possível se obter sistemas de alta disponibilidade, de acordo necessidade do usuário, a partir de *Clouds* de mais baixa disponibilidade, além de ser possível economizar com os custos dos serviços do provedor.

Termos de indexação: Gestão de Redes e Serviços, Alta Disponibilidade, *Cloud Computing*, *Multi-Cloud*, *Proxy* Reverso e Sistema de nomes de domínio.

ABSTRACT

PIRES, Luis Paulo Gonçalves. **Alta Disponibilidade: Uma abordagem com DNS e *Proxy* Reverso em *Multi-Cloud***, 2016. Dissertação (Mestrado Profissional em Gestão de Redes de Telecomunicações) - Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Campinas, 2016.

While there is considerable enthusiasm for the migration of on-premise data centers to cloud computing services, there is still some concern about the availability of these same services. This is due, for example, to historical incidents such as that in 2011, when a crash on Amazon's servers caused sites of several of its customers to go down for almost 36 hours. In view of this, it becomes necessary to develop strategies to guarantee the availability offered by the providers. In the present work, a solution is proposed, which implements high availability in Multi-Cloud environments, through the distribution of DNS access and the use of reverse proxy. A financial analysis was also carried out, taking into account market values in Cloud Computing services, which showed that the proposed solution may even be advantageous with respect to the traditional one. Specifically, a Multi-Cloud system, consisting of two Clouds with 99.90% availability each, provides total availability of 99.999%, and it costs 34% less than a single Cloud with 99.95% availability. The simulation results, obtained in a virtualized environment, using two Clouds, with availability of 99.49% and 99.43%, showed a system availability of 99.9971%. In this way, using Multi-Cloud systems it is possible to obtain high availability systems, from lower availability Clouds, according to user’s needs, besides saving with provider services costs.

Indexing Terms: Management of Networks and Services, High Availability, Cloud Computing, Multi-Cloud, Reverse Proxy, and Domain Name System.

LISTA DE FIGURAS

Fig. 1 – Representação de Redundância – Ativo-Ativo / Ativo-Passivo 5

Fig. 2 – Estimativa de falhas em componentes do ambiente estudado 10

Fig. 3 – Custo monetário (US$) X diferentes soluções de armazenamento 11

Fig. 4 - Arquitetura em camadas – Virtualização de Hardware 12

Fig. 5 – Representação de Cloud Computing. 14

Fig. 6 – Percentual de empresas x Adoção de estratégias Multi-Cloud. 16

Fig. 7 – Planejamento e Implantação de sistema de alta disponibilidade. 19

Fig. 8 - Representação esquemática de SLA. 20

Fig. 9 – Exemplos de nomes traduzidos IP. 21

Fig. 10 – DNS Arquitetura em Árvore Invertida. 22

Fig. 11 – Fluxo de requisição de endereço IP. 23

Fig. 12 – Proxy de encaminhamento. 24

Fig. 13 – Proxy Reverso como Frontend. 25

Fig. 14 – Distribuição de acessos com *proxy* simples em *Multi-Cloud*. 26

Fig. 15 – Diagrama geral da proposta - Distribuição de acessos por DNS e Proxy Reverso. 28

Fig. 16 – Cenário de Indisponibilidade da *Cloud* 1. 30

Fig. 17 – Cenário de Indisponibilidade do Servidor *Proxy*. 31

Fig. 18 – Cenário de Indisponibilidade do Servidor Web. 33

Fig. 19 - Configuração de múltiplas entradas para o mesmo nome 35

Fig. 20 – Servidores Web - Participação de mercado nos principais sites 37

Fig. 21 – Painel de administração do Oracle Virtualbox. 39

Fig. 22 - Exemplo de Monitoramento do Nginx com Munin 40

Fig. 23 – NETEM: Comando para inserção de latência. 41

Fig. 24 – NETEM: Comando para inserção de perda de pacotes. 41

Fig. 25 – NETEM: Comando para inserção de percentual de pacotes duplicados. 41

Fig. 26 – NETEM: Comando para inserção de reordenação de pacotes. 41
Fig. 27 – Comando para execução o Apache Benchmark. 42
Fig. 28 – Exemplos de gráficos gerados com AB + GNUPlot. 43
Fig. 29 – Diagrama de Rede – Utilizado para o testabilidade da proposta. 44
Fig. 30 - Resultado do Benchmark – *Cloud*1. 47
Fig. 31 - Resultado do Benchmark – *Cloud*2. 47
Fig. 32 - Servidores de Aplicação – Cloud1 e 2 – 24 horas. 49
Fig. 33 - Log do Script gerador de Interrupção - *Cloud2*. 50
Fig. 34 - Log do Script gerador de Interrupção - *Cloud1.* 51
Fig. 35 – Variação da Disponibilidade por número de *Clouds* de 99,95%. 54
Fig. 36 – Variação da Disponibilidade por número de *Clouds* de 99,90%. 55
Fig. 37 – Comparativo: Custo por Provedor *Multi-Cloud*. 56
Fig. 38 – Combinações de *Clouds* x Custo. 58

LISTA DE TABELAS

Tabela 1: Percentual de Disponibilidade x Tempo de Parada 17

Tabela 2: Ordenação de respostas DNS 36

Tabela 3: Relação de Máquinas Virtuais. 45

Tabela 4 - Disponibilidade do ambiente de testes – *Clouds* Individuais x *Multi-Cloud*. 52

Tabela 5: Disponibilidade x Custo 53

Tabela 6: Comparativo % de disponibilidade da simulação 55

Tabela 7: Combinações de Diferentes D para DTmin de 99,999% 56

LISTA DE ABREVIATURAS E SIGLAS

CC	=	Cloud Computing
IBM	=	International Business Machine
VM	=	Virtual Machive
SLA	=	Service Level Agreement
EBS	=	Elastic Block Storge
CRM	=	Customer Relationship Management
PaaS	=	Plataform as a Service
P2P	=	Peer to Peer
OS	=	Operational System
DB	=	Data Base
PUF	=	Ponto Único de Falha
VMM	=	Virtual Machine Monitor
ROI	=	Return Over Investiment
TI	=	Tecnologia da Informação
EC2	=	Elastic Compute Cloud
S3	=	Simple Storage Service
NIST	=	National Institute of Standards and Tecnology
XaaS	=	X as a Service
SaaS	=	Software as a Service
IaaS	=	Infrastructure as a Service
TMForum	=	Telemanagement Forum
DNS	=	Domain Name System
IP	=	Internet Protocol
BIND	=	Berkley Internet Name Domain
TTL	=	Time to Live
Ts	=	Tempo de Serviço

Tp	=	Tempo de parada
DT	=	Disponibilidade Total
WWW	=	*World Wide Web*
HTTP	=	*Hypertext Transfer Protocol*
WWW	=	Hypertext Transfer Protocol Secure
NCSA	=	National Center for Supercomputing Activies
BSD	=	Berkeley Software Distribution
NETEM	=	Network Emulator
AB	=	Apache Benchmark
LAN	=	Local Area Network
WAN	=	Wide Area Network
NS	=	Name Server

SUMÁRIO

1 INTRODUÇÃO 1
1.1 - Proposta 5
1.2 - Organização 6

2 TRABALHOS RELACIONADOS 7

3 COMPUTAÇÃO EM NUVEM 12
3.1 Virtualização 12
3.1.1 – Hypervisores 12
3.2 – Centro de Dados 13
3.3 – Definição: *Cloud Computing* 13
3.3.1 – Modelos de Serviços 14
3.3.2 – Modelos de Oferta 15

4 DEFINIÇÃO DE *MULTI-CLOUD* 16
4.1 - Disponibilidade 16
4.2 – *Service Layer Agreement* 19

5 MODELO DE ALTA DISPONIBILIDADE EM *MULTI-CLOUD* 21
5.1 - DNS 21
5.1.2 - DNS – Suporte para Balanceamento de Carga 23
5.2 – Proxy 23
5.2.1 - Proxy de encaminhamento 24
5.2.2 - Proxy Reverso 24
5.3 – Alta Disponibilidade em *Multi-Cloud* 25
5.3.1 – Ponto Único de Falha 26
5.3.2 – Abordagem com DNS e *Proxy* Reverso em *Multi-Cloud* 26
5.4 – Expressão para cálculo de disponibilidade 33
5.5 – Expressão proposta para cálculo de disponibilidade em *Multi-Cloud* 34

6 SISTEMA EXPERIMENTAL 35
6.1 – Ferramentas utilizadas 35
6.1.1 – BERKELEY INTERNET NANE DOMAIN – BIND 35
6.1.2 – BIND ROUND ROBIN 35
6.1.3 – Servidores Web 36
6.1.3.1 – Apache2 38
6.1.3.2 – Nginx 38
6.2 – Ambiente de Virtualização 38
6.3 – Sistema Operacional Base 39
6.4 – Software de Monitoramento 39
6.5 – Emulação de rede 40
6.5.1 – Network Emulator (NETEM) 40
6.6 – Ferramenta de Benchmark 42
6.6.1 – Apache Benchmark (AB) 42
6.7 – Funcionamento do Ambiente 43

7 TESTES DE DESEMPENHO E DISPONIBILIDADE 46

8 ANÁLISE DE FATORES DE CUSTO ASSOCIADOS A DIFERENTES COMBINAÇÕES DE CLOUD PARA PROVIMENTO DE IAAS 53

9 CONCLUSÃO.....59
9.1 - Comentários Gerais.....59
9.2 - Trabalhos Futuros.....60

10 REFERÊNCIAS.....61

11 APÊNDICES.....67

1 INTRODUÇÃO

A utilização de serviços de tecnologia da informação estruturados em nuvem, conhecidos como *Cloud Computing*, vem ganhando nos últimos anos, cada vez mais adeptos ao redor do mundo [1]. No ano de 2010, o Brasil superou a média mundial na adoção de *Cloud Computing*, segundo um estudo realizado pela Cisco [2]. No "*Cisco Connected World Report*", relatório divulgado em oito de Dezembro de 2010, que foi realizado com base em dados de 13 países, o Brasil apareceu empatado com a Alemanha, com 27% das companhias do país já utilizavam aplicações baseadas em *Cloud*, número bem acima da média mundial (18%), ficando a frente de países como Índia (26%), Estados Unidos (23%) e México (22%) [3]. Outro dado importante divulgado na pesquisa foi que 88% dos entrevistados disseram que utilizariam alguma tecnologia baseada em *Cloud* nos anos seguintes. Já em 2015 [4], o mercado mundial de serviços em nuvem superou a casa dos US$ 175 bilhões em receitas. Estima-se que ao final de 2016 o valor supere os US$ 204 bilhões, e para 2017, a expectativa do mercado é que haja um aumento da ordem de 17,3% em relação às receitas de 2016 envolvendo serviços em nuvem.

Historicamente, o conceito de Computação em Nuvem pode ser comparado ao modelo que foi idealizado na década de 1950, com a utilização compartilhada dos servidores mainframes por meio de "terminais burros" [5]. Estes, mais eram do que estações sem nenhum poder de processamento que possibilitavam que seus usuários se conectassem ao servidor. Já os mainframes apresentavam enormes infraestrutura de hardware dispostos em salas que ficaram conhecidas como "*server room*". Devido à sua estrutura e altos custos envolvidos em seu gerenciamento, não seria viável a disponibilização de um mainframe para cara usuário [6]. Assim, a solução foi compartilhar os recursos computacionais entre os usuários, levando desta forma, a um melhor retorno sobre o investimento.

Na década de 1970, a IBM expandiu a utilização de aplicações compartilhadas nos mainframes para múltiplos sistemas virtuais por meio System/370 Mainframe. Criava-se o conceito de *Virtual Machine*, ou virtualização, considerada um dos maiores avanços na área de computação [5].

As tecnologias de virtualização associada à evolução nos sistemas de telecomunicações ocorridos desde a década de 1990 que tornaram a internet mais acessível permitiram criar um novo paradigma de forma que os limites da computação seriam definidos não mais por critérios técnicos, mas sim por fatores financeiros [6].

Atualmente são milhares as aplicações em *Cloud Computing* que envolvem o cotidiano de um usuário final, e vão desde a utilização de *softwares* para edição de texto e planilhas eletrônicas até Redes Sociais.

Diversos são os benefícios na utilização de serviços em *Cloud Computing* que podem ser sumarizados conforme abaixo [7]:

- **Acessibilidade**
  Sistemas e arquivos disponíveis em nuvem podem ser acessados a qualquer tempo por qualquer dispositivo compatível, seja por computadores ou dispositivos móveis.

- **Colaboração**
  Os dados armazenados em nuvem podem ser compartilhados entre diferentes usuários simultaneamente formando grupos de trabalho.

- **Escalabilidade**
  Não há necessidade de investimento em infraestrutura para aumentar os recursos, bastando reconfigurar e alocar conforme a necessidade.

- **Redução de custo**
  Diversos tipos de plataformas em nuvem cobram pelo uso efetivo, não sendo desperdiçados recursos financeiros.

Contudo, problemas relacionados à indisponibilidade em provedores que acabam comprometendo o acesso aos serviços de *Cloud Computing* são comuns. Em 2011, uma falha nos servidores de armazenamento em nuvem do provedor de *Cloud Computing* Amazon fez com que sites de vários de seus clientes ficassem fora do ar

por quase de 36 horas [8], entre eles Foursqure e New York Times. Neste mesmo episódio, a falha ocorrida gerou mais problemas do que sites fora do ar, chegando a perder dados que estavam armazenados na solução Amazon EBS Volumes. Alguns dias após a falha, a Amazon enviou um comunicado aos clientes afetados com o seguinte conteúdo [9]:

> *"Hello,*
>
> *A few days ago we sent you an email letting you know that we were working on recovering an inconsistent data snapshot of one or more of your Amazon EBS volumes. We are very sorry, but ultimately our efforts to manually recover your volume were unsuccessful. The hardware failed in such a way that we could not forensically restore the data.*
>
> *What we were able to recover has been made available via a snapshot, although the data is in such a state that it may have little to no utility...*
>
> *If you have no need for this snapshot, please delete it to avoid incurring storage charges.*
>
> *We apologize for this volume loss and any impact to your business.*
>
> *Sincerely,*
>
> *Amazon Web Services, EBS Support"*

Devido a este tipo de acontecimento, empresas ainda são relutantes em utilizar as tecnologias de *Cloud Computing* em seus Ambientes de Produção. Em [1], mostrou-se que 91% das 58 empresas Europeias consultadas, com experiências bem sucedidas em *Cloud*, migraram somente serviços de ambientes "*non-productions*". Ainda assim, mesmo no caso que sistemas de produção foram migrados, estes envolviam apenas serviços de baixa criticidade, tais como sistemas de CRM (*Customer Relationship Management*) e ferramentas de escritório para a exploração dos benefícios de flexibilidade e mobilidade oferecidos por *Cloud Computing*. Desta forma, garantindo-se não afetar a continuidade dos negócios em casos de falha.

Pode-se dizer que, a despeito do entusiasmo global com *Cloud Computing* a adesão à tecnologia enfrenta problemas de confiabilidade semelhantes aos encontrados pelos sistemas de *OutSourcing* na década de 1990 [10]. Em face de tal cenário, prover alta disponibilidade é, portanto, uma tarefa desafiadora.

O termo disponibilidade, em computação, refere-se ao período de tempo em que um determinado sistema fica disponível para utilização, por exemplo: 16 horas por dia, 6 dias por semana [11] ou percentuais de disponibilidade mensal, por exemplo 99,95% [12]. Este nível de serviço é definido entre o provedor e o cliente em contrato. Caracteriza-se por alta disponibilidade, um sistema que fora desenhado para ter redundância de componentes, para que em caso de falhas, os serviços continuem em funcionamento de forma transparente para o usuário, ou minimize-se o tempo de parada (tempo em que o sistema fica indisponível para o usuário) do sistema como um todo [11].

Soluções de alta disponibilidade podem ser implantadas em sistemas em datacenters locais (*On-Premise DataCenter*), para recuperação de desastres são utilizadas soluções de alta disponibilidade geograficamente distribuídas de forma que suportem por exemplo, problemas de natureza física, como inundações ou falhas de rede regional. Em ambos os casos, o principal mecanismo é a redundância que pode ser dividida em dois níveis [13].

- Redundância Ativo-Ativo
  Refere-se a soluções com duas ou mais instâncias ativas do sistema de forma que todos possam lidar com pedidos de forma simultânea (Fig. 1a). Também pode ser utilizada para melhor a escalabilidade.

- Redundância Ativo-Passivo
  Refere-se a soluções em que uma instância do sistema fica ativa, atendendo as solicitações do usuário, e uma instância passiva, que fica em estado de espera, de forma que esta somente fica ativa caso a instancia principal falhe (Fig. 1b). Neste cenário, existe um monitoramento conhecido como *heartbeat* do servidor que está ativo, de forma que o servidor passivo possa conhecer o estado do sistema.

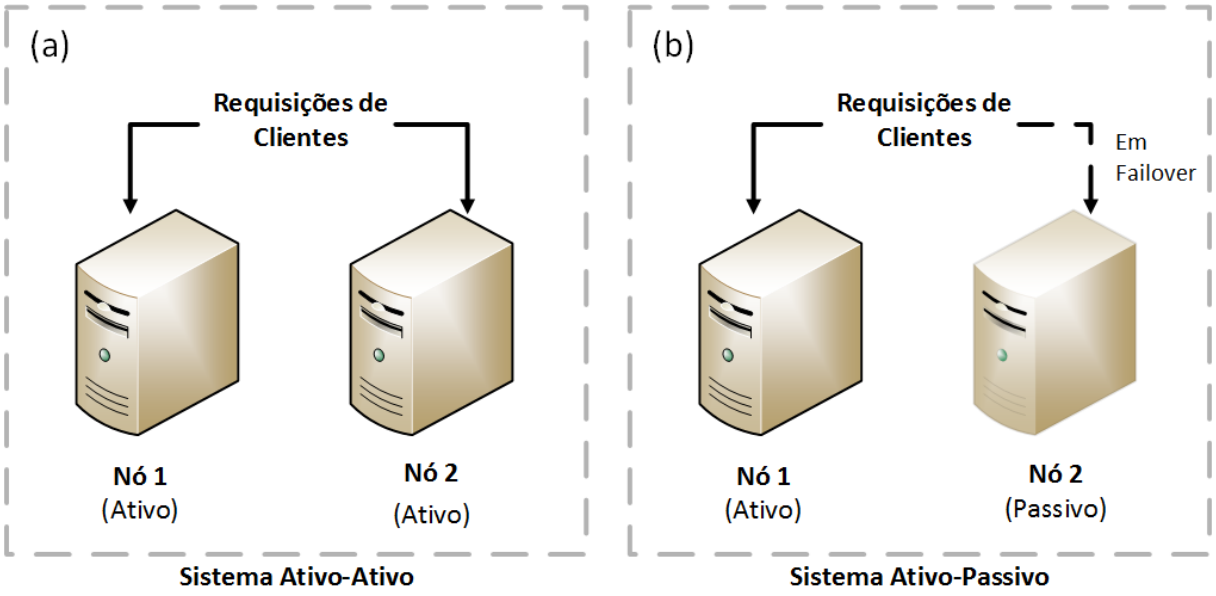


Fig. 1 – Representação de Redundância – Ativo-Ativo / Ativo-Passivo (adaptado de [12]).

Assim, tendo em vista este contexto, cabem estudos que busquem soluções para os problemas aqui apresentados de forma que, principalmente empresas que ainda são relutantes ao conceito, possam utilizar-se das tecnologias de *Cloud Computing* sem que necessariamente fiquem atrelados com um único provedor de serviços e à mercê da possibilidade de ficar sem seus sistemas em caso de falha, ou pior, perder dados hospedados em *Cloud*.

1.1 - Proposta

Neste trabalho, propõe-se e testa-se, utilizando-se de simulações, uma proposta para o provimento de alta disponibilidade, mediante a distribuição de acessos em *Multi-Cloud*[1] [14], o que permite ofertar, de acordo com a necessidade do usuário, por exemplo, até da ordem de 99,999% [15]. Para tanto, em conjunção com um ambiente *Multi-Cloud*, são utilizadas técnicas de distribuição de acessos por *Domain Name System* (DNS) e conexão de *Proxy* reverso [16].

[1] Utiliza-se neste estudo o termo *Multi-Cloud* sem tradução, pois sua tradução literal "Múltiplas Nuvens" não é um termo conhecido e não foi encontrado em nenhum estudo. Apensar de o termo "computação em nuvem" ser conhecido em português, optou-se por utilizar os termos "*Cloud*" e "*Multi-Cloud*".

1.2 - Organização

O presente estudo organiza-se da seguinte forma: A seção II apresenta os trabalhos relacionados. Em III, apresentam-se os conceitos e definições sobre Computação em Nuvem. Em IV define-se *Multi-Cloud* e apresentam-se conceitos de disponibilidade e de Acordo de Nível de Serviço. Em V, apresenta-se e discute-se o modelo proposto, além da topologia da solução. Em VI, apresenta-se o ambiente de experimentação para o teste do método de Alta Disponibilidade. Na sessão VII, são apresentados os resultados dos testes de desempenho e disponibilidade do ambiente. Em VIII é apresentada uma análise financeira e, finalmente, em IX, tem-se a conclusão. As referências são apresentadas na sessão X, e em XI expõe-se os apêndices.

2 TRABALHOS RELACIONADOS

Neste capítulo são identificados alguns estudos que apresentam conceitos e/ou tecnologias que são relevantes para as abordagens apresentadas neste estudo.

Multi-Cloud: Expectations and Current Approaches [17] - O estudo visa identificar o estado da arte em *Multi-Cloud* e apresentar melhorias nas soluções para as necessidades de desenvolvedores. O estudo foi dividido em três partes: 1º Firma-se a necessidade de *Multi-Cloud*; 2º Posiciona-se *Multi-Cloud* versus múltiplos modelos de nuvem; 3º Revisa-se softwares *Multi-Cloud* comerciais e acadêmicos disponíveis.

São apresentadas dez razões pelas quais produtos e serviços em *Multi-Cloud* são necessários, conforme abaixo:

1 – Lidar com picos de solicitações de serviços e recursos de clientes externos baseados em demanda;

2 – Otimizar custos ou melhorar a qualidade de serviços;

3 – Reagir às mudanças por parte dos provedores;

4 – Seguir as restrições de locais ou leis;

5 - Replicação de aplicações ou serviços a partir de diferentes provedores de *Cloud Computing* e assim, garantir alta disponibilidade;

6 – Evitar a dependência de um único provedor.

7 – Garantir redundância para lidar com inatividades programadas ou catástrofes;

8 – Agir como intermediário;

9 – Melhorar os próprios recursos de serviços em *Cloud* com base em acordos com outros provedores.

10 – Consumir diferentes serviços com suas particularidades fornecidas por diferentes provedores.

Existem muitos termos relativos ao assunto e apresentam-se alguns como: *Multi-Cloud* (Múltiplas *Clouds*), *Cloud Federation* (*Clouds* Federadas), *Inter-Cloud, Hybrid Cloud (Clouds* Hibridas*), Cloud-of-Clouds (Clouds* para *Clouds), Sky Computing (*Computação no céu*), Aggregated Clouds (Clouds* Agregadas*) , Multi-tier Clouds (Clouds* com Múltiplas Camadas*), Cross-Cloud (Cloud Cruzada), Cloud Blueprint*

(Cloud Modelo*), Cloud Merge (Cloud* Mesclada*), Fog Computing (*Computação em Névoa*), Hi-erarchical Clouds (Clouds* Hierárquica)*, Distributed Clouds* (*Clouds* distribuídas) entre outras. Essa variedade de termos ocorre porque não existe uma taxonomia bem estabelecida, porém, [18] considera um limite entre *Multi-Cloud* e outros modelos de *Cloud.*

Apresentam-se dois tipos de entrega de modelos de múltiplas *Clouds*: *Federated Cloud* (Nuvens Federadas) e *Multi-Cloud (*Múltiplas Nuvens). A diferença essencial entre os dois tipos apresentados [19], é que no modelo *Federated Cloud* existe colaboração entre as *Clouds* envolvidas, o que não ocorre no modelo *Multi-Cloud*. Os modelos podem ser classificados como *Volunteer* (Voluntário), em *Federated Cloud*, e como *Non-Volunteer* (Não Voluntário) em *Multi-Cloud*. Seguindo a classificação proposta em [20] o termo *Multi-Cloud* indica o uso de *Clouds* múltiplas e independentes por um cliente ou serviço.
Uma funcionalidade importante em ambientes *Multi-Cloud* é o gerenciamento em diferentes *Clouds*. Assim, são apresentadas algumas ferramentas voltadas para gerenciamento *Multi-Cloud*, que foram classificadas em *Library-based* (Baseado em Bibliotecas) e *Service-based* (Baseado em serviços) [20]. Sumariza-se abaixo:

***Library-based*:**

- Jclouds – Ferramenta de código aberto para suportar portabilidade de aplicações Java.
- Libcloud – Biblioteca *Python* que abstrai as diferenças entre interfaces de programação oferecidas.
- Simple Cloud – Biblioteca em PHP que oferece interfaces uniformes para armazenamento de arquivos e documentos.

Service-based:

- RightScale – Plataforma de gerenciamento para controle e administração de desenvolvimento em diferentes *Clouds*.
- enStratus – Permite gerenciamento, monitoração, automação e governança do consumo de recursos em *Multi-Cloud.*
- Kaavo – Permite o gerenciamento de aplicações distribuídas em várias *Clouds.*

Concluiu-se que seguindo os múltiplos cenários atuais de *Clouds*, não existe um conceito concreto sobe *Multi-Cloud*, porém, o estudo foi considerado útil no que diz respeito à identificação de lacunas para se chegar ao conceito em um futuro próximo, uma vez que os requisitos de *Multi-Cloud* apresentados são considerados ponto de partida para novas abordagens.

A Federated Multi-Cloud PaaS Infrastructure [21] - Segundo o autor, *Cloud Computing* é uma das principais tendências de pesquisa em sistemas escaláveis e distribuídos.
O estudo apresenta a aplicação de uma Plataforma como serviço (PaaS) em Nuvens Federadas, utilizando onze Provedores de *Cloud Computing*. A arquitetura utilizada para provimento da PaaS foi dividida em três fundamentos:

i) Modelo de Serviço Aberto:

Promove a ideia de montagem de componentes de software em uma rede serviços acoplados.

ii) Arquitetura Configurável:

Baseia-se em um *Kernel* (núcleo) configurável, especializado para ajustar as características do ambiente em nuvem.

iii) Serviços de Infraestrutura:

Refere-se a serviços comuns que são fornecidos para facilitar o desenvolvimento de aplicativos em *Cloud* utilizando a infraestrutura proposta.

Para validação da Infraestrutura *Multi-Cloud* para PaaS, foram utilizadas três aplicações SaaS sendo um de Monitoramento de Rede P2P (Ponto-a-Ponto) e dois *softwares* de desenvolvimento próprio.

Concluiu-se, o estudo abriu novas perspectivas para ambientes *Multi-Cloud*, como por exemplo, a elasticidade neste tipo de ambiente, devido à possibilidade de mais *Clouds* serem inseridas no ambiente conforme necessário. A diversidade de estratégias comerciais e preços oferecidos por provedores de soluções em nuvem torna a abordagem relevante do ponto de vista financeiro. Essa é uma abordagem proposta para investigação futura.

Em parte, este assunto proposto foi abortado em nosso estudo, uma vez que no capitulo 8 apresentamos uma análise de viabilidade financeira da solução de utilização de *Multi-Cloud* proposta, com diferentes combinações de *Clouds* com disponibilidades individuais menores, buscando unir alta disponibilidade a melhor custo x benefício.

Dependability Modeling Framework: A test procedure for High Availability in Cloud Operating Systems [15], utilizando-se o *OpenStack* [22], software para controle de recursos em *datacenter*, propôs-se um método para o teste de disponibilidade em *Cloud*. Foram instalados componentes em um nó simples, gerenciados pelo *OpenStack* e, por meio de *scripts*, promoveram-se interrupções individuais nestes componentes. Como resultado, apresentaram-se os valores típicos de indisponibilidade (Fig. 2) dos componentes obtidos nos testes e o impacto que cada um gera sobre a disponibilidade do serviço. Concluiu-se que o método proposto permite mensurar a disponibilidade em arquiteturas de *Cloud Computing,* evidenciando-se a relação de dependência dos componentes envolvidos no sistema.

Component	Availability Level	Avg. Recovery Time	Outage Risk per Day
Hardware of OpenStack installation	99.99999 %	600	0.00144 %
OS of OpenStack installation	99.99999 %	600	0.00144 %
Apache	99.9 %	1’200	7.2 %
Ceilometer	99.9 %	1’200	7.2 %
Cinder	99.9 %	1’200	7.2 %
VM internal Connection DB	99.8 %	1’200	14.4 %
VM internal Connection Management	99.8 %	1’200	14.4 %
Glance	99.9 %	1’200	7.2 %
Horizon	99.9 %	1’200	7.2 %
Keystone	99.9 %	1’200	7.2 %
VM internal Node Location Detection	99.8 %	1’200	14.4 %
MySQL	99.9 %	1’200	7.2 %
Nova	99.9 %	1’200	7.2 %
VM internal Operating System	99.8 %	1’200	14.4 %
VM internal Password DB	99.8 %	1’200	14.4 %
Quantum	99.9 %	1’200	7.2 %
RabbitMQ	99.9 %	1’200	7.2 %
VM internal Password Management	99.8 %	1’200	14.4 %
Swift	99.9 %	1’200	7.2 %
VM internal User DB	99.8 %	1’200	14.4 %
VM internal Ceilometer Plugin	99.8 %	1’200	14.4 %

Fig. 2 – Estimativa de falhas em componentes do ambiente estudado [14].

O conceito de geração de falhas no ambiente de experimentação de forma independente e aleatória por meio da utilização de *scripts* para testes de disponibilidade apresentados neste estudo nortearam a forma com que se realizaram as interrupções no ambiente de experimentação do presente estudo, a fim de comprovar a disponibilidade da solução proposta.

CHARM: A Cost-Efficient Multi-Cloud Data Hosting Scheme with High Availability [14], no âmbito de armazenamento de dados propõem-se algoritmos para a distribuição e o armazenamento de dados em *Multi-Cloud* com foco em alta disponibilidade e eficiência no custo de armazenamento. A implementação do protótipo foi realizada utilizando-se de *Clouds* comerciais. Como resultados foram apresentados dados comparativos entre o custo de armazenamento por meio da distribuição de dados proposta no estudo, e outros softwares semelhantes, conforme vê-se na Fig.3.

Availability	RepRa	RepGr	EraRa	EraGr	CHARM
99.99%	383.18	239.38	285.55	277.71	210.09
99.999%	377.38	239.38	281.31	275.25	212.64
99.9999%	283.68	239.38	270.14	275.25	217.19
99.99999%	453.29	288.71	269.67	275.25	222.08

Fig. 3 – Custo monetário (US$) X diferentes soluções de armazenamento [13].

A forma de alocação dos dados que foi proposta, empregou um servidor *proxy* distribuindo os arquivos entre *Multi-Cloud*, que se assemelha ao modelo de distribuição de acessos utilizado em nossa proposta. Contudo, neste estudo, utilizou apenas um servidor *proxy* para distribuição de dados, o que configura um problema conhecido por “PUF” (Ponto Único de Falha) [23]. Este problema em nossa implementação foi resolvido com a utilização do conceito de N+1 *proxies*, sendo N é o número de *Clouds* utilizadas na solução proposta. Este assunto será abordado em detalhes no capítulo 5.

3 COMPUTAÇÃO EM NUVEM

Neste capítulo são apresentados os conceitos e definições sobre Computação em Nuvem, assim como as seus modelos de serviços de entrega. Apresentam-se também os conceitos de Virtualização, que dão suporte e possibilitam o paradigma da Computação em Nuvem.

3.1 Virtualização

O conceito de virtualização remete à emulação de ambientes isolados, nos quais pode-se executar diferentes sistemas operacionais dentro de uma mesma arquitetura de hardware [24]. Em uma arquitetura em camadas (Fig. 4), um sistema operacional instanciado é conhecido como VM *Guest (*Máquina Virtual convidada*)* e é executado sobre uma camada de software que fornece a abstração necessária para a execução concorrente de diversos sistemas operacionais em um mesmo computador. Esses são conhecidos como *Hypervisores* ou VMM *(Virtual Machine Monitor)* [25].

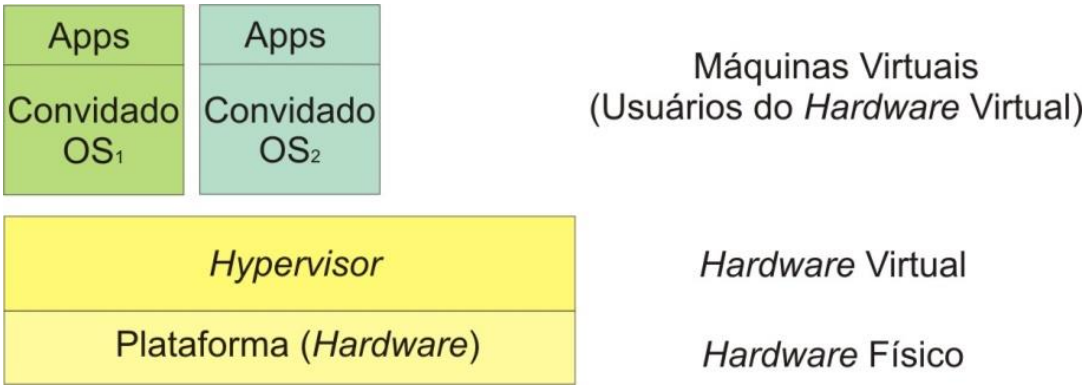


Fig. 4 - Arquitetura em camadas – Virtualização de Hardware (Adaptado de [21]).

3.1.1 – Hypervisores

Os *Hypervisores* fornecem para os sistemas operacionais plataformas de hardwares virtuais, isoladas, para que sejam executados. Neste contexto, o

hypervisor é a camada que abstrai o hardware físico e fornece o cenário virtual para a execução da VM [26].

3.2 – Centro de Dados

Em 2009 [27], as infraestruturas computacionais existentes estruturadas em centros de dados modernos apresentavam-se subutilizadas, sendo que em sua maioria, a taxa de utilização ficava entre 5% a 10% de sua capacidade. Neste contexto o retorno sobre o investimento (ROI) torna-se menor, e em tempos em que o termo sustentabilidade é bastante difundido, esta subutilização de recursos computacionais, não só de processamento, mas de consumo energético, de refrigeração, espaço físico e outros, chamou a atenção de grandes empresas do setor de TI [27].

A Amazon, após a modernização de seus *datacenters*, foi a precursora de um novo paradigma da computação em nuvem, oferecendo as soluções *Amazon Elastic Compute Cloud* (EC2) e *Simple Storage* Service (S3) [6]. Posteriormente às novas tecnologias oferecidas pela Amazon, outras empresas passaram a oferecer tecnologias de *Cloud Computing*, como por exemplo, o Google que popularizou a modalidade de serviços em *Cloud* para usuários finais por meio do lançamento do Google Docs [28]. Grandes nomes do segmento de TI também investiram em *Cloud Computing*, como a Apple, Microsoft, IBM entre outras.

3.3 – Definição: *Cloud Computing*

De acordo com o *National Institute of Standards and Tecnology*, NIST, o conceito de *Cloud Computing* pode ser definido como um modelo que permite acesso compartilhado à um conjunto de recursos computacionais sob demanda, que podem ser rapidamente fornecidos e liberados com esforço mínimo de gerenciamento do usuário ou por interação do provedor do serviço [7].

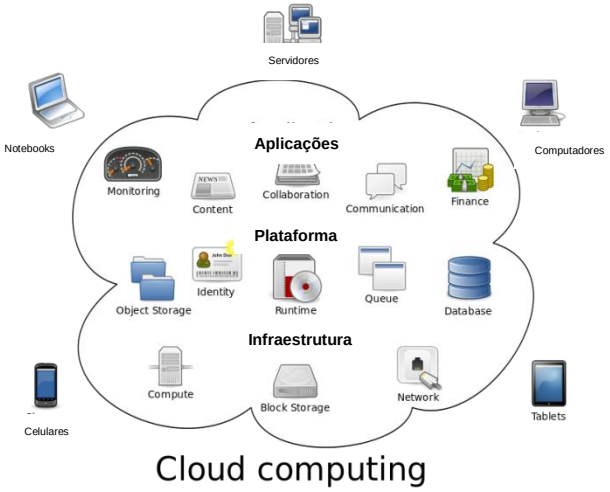


Fig. 5 – Representação de *Cloud Computing.*

Provedores de *Cloud Computing* oferecem diversos serviços conhecidos como *"X as a Service"* [29], ou "tudo como serviço" que permitem uma ampla gama de possibilidades, que vão desde a utilização de um simples *software*, até complexas infraestruturas de rede como serviço. Pode-se dividir o padrão de *Cloud Computing* conhecido hoje em três *Service Models (*Modelos de Serviço) e quatro *Deployment Models* (Modelos de Oferta) [7].

3.3.1 – Modelos de Serviços

- **SaaS – Software as a Service (*Software* como Serviço)**
  Permite que o usuário utilize aplicações oferecidas pelo provedor. Ex: Google Drive, Microsoft Office 365.
  Neste modelo, o usuário não exerce nenhum tipo de gerenciamento em nível de redes, servidores e sistemas operacionais sob a infraestrutura de Cloud Computing.

- **PaaS – Plataform as a Service (Plataforma como Serviço)**
  Permite ao usuário implantar ou criar aplicações através de linguagens de programação, bibliotecas ou serviços que são suportados pelo Provedor em sua infraestrutura de *Cloud Computing.*
  Ex: Google App Engine, Microsoft Azure.

Neste modelo, o usuário também não exerce nenhum tipo de gerenciamento em nível de redes, servidores e sistemas operacionais sob a infraestrutura de Cloud Computing, como no modelo SaaS.

- **IaaS – Infrastructure as a Service (Infraestrutura como Serviço)**
  Permite ao usuário o provisionamento e gerenciamento de recursos necessários à sua demanda, como processamento, armazenamento, redes e sistemas operacionais.
  Ex: Amazon AWS, Google Cloud.

3.3.2 – Modelos de Oferta

- **Nuvem Privada**
  A infraestrutura neste modelo é provisionada de modo exclusivo para uma única organização. Sua operação e gerenciamento podem ficar por conta da própria organização ou repassados a terceiros.

- **Nuvem Comunitária**
  A infraestrutura neste modelo é provisionada exclusivamente para uma comunidade de usuários de organizações que compartilham informações. Ex: Polícia, Governos, entre outros. Sua operação e gerenciamento podem ficar por conta de uma ou mais organizações ou serem repassados a terceiros.

- **Nuvem Publica**
  A infraestrutura neste modelo é provisionada de forma aberta ao público em geral. Sua operação e gerenciamento podem ficar por conta de Empresas de TI, Organizações de Governos, Acadêmicos, entre outros.

- **Nuvem Hibrida**
  A infraestrutura neste modelo é composta por dois ou mais Modelo de Oferta em *Cloud Computing.*

4 Definição de *MULTI-CLOUD*

A utilização simultânea de duas ou mais *Clouds* como estrutura para um mesmo sistema de TI ou a um conjunto de aplicações de uma mesma organização é conhecida como *Multi-Cloud* [17]. Em relação à utilização de múltiplas *Clouds*, podem-se combinar diferentes *Deployment Models* para compor o ambiente *Multi-Cloud.* Por exemplo, uma organização que possua um *Data-Center On-Primese* que ofereça serviços de *Cloud Computing* a seus usuários pode também utilizar simultaneamente um Modelo Oferta de Nuvem Pública. Desta maneira, forma-se uma Nuvem Hibrida cujos recursos entregues aos usuários estão distribuídos em diferentes estruturas formando um ambiente *Multi-Cloud*, melhorando-se diversos atributos do ambiente como um todo, como poder de processamento, redundância de estrutura, o que leva consequentemente a um aumento na disponibilidade, entre outros. Segundo [30] a maioria das organizações tem adotado estratégias em *Multi-Cloud*, conforme apresentado na Fig.6.

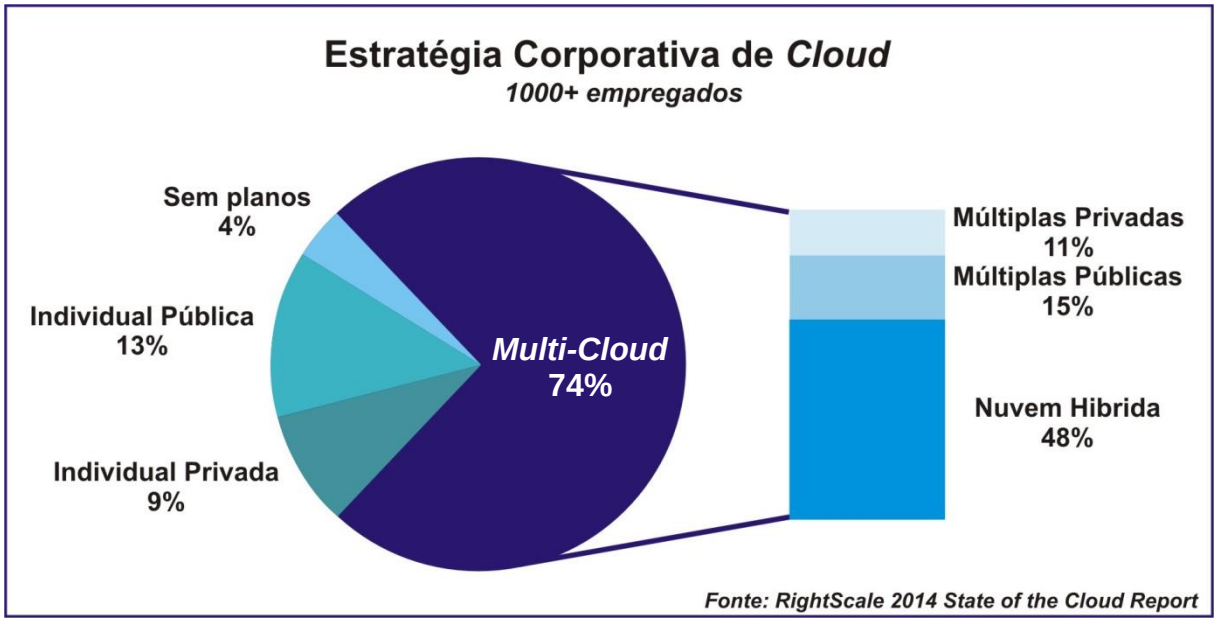


Fig. 6 – Percentual de empresas x Adoção de estratégias Multi-Cloud [25].

4.1 - Disponibilidade

Trata da relação entre o tempo de atividade de um sistema pelo seu tempo de parada, seja ela corretiva ou programada. Neste cenário, disponibilidade é um dos

parâmetros que apresentam maior impacto em quaisquer acordos de nível de serviço. Na esfera empresarial, este é um dos grandes desafios enfrentados quando se trata de serviços em *Cloud Computing* [12].

Tipicamente, provedores de *Cloud Computing* anunciam percentuais de disponibilidade de 99,95% [12]. Isso, por muitas vezes, leva os usuários a desenvolverem uma expectativa equivocada quanto a ser praticamente nula a indisponibilidade (*downtime*) dos serviços oferecidos por provedores de *Cloud Computing*. Tal equívoco, a dificuldade de se quantificar os atributos que compõem o SLA (*Service Layer Agreement*), destaca-se como uma das principais causas da baixa aderência aos recursos de *Cloud Computing* em ambientes de produção por parte de usuários corporativos [1].

Na tabela 1, apresentam-se os percentuais de disponibilidade tipicamente visualizados em se tratando de serviços em *Cloud Computing*, bem como os valores calculados de tempo de parada anual e mensal com base no percentual de disponibilidade.

Tabela 1: Percentual de Disponibilidade x Tempo de Parada

Disponibilidade (%)	Tempo de Parada - Anual		Tempo de Parada Mensal	
99,9999999%	0,03	Seg.	0,003	Seg.
99,999999%	0,32	Seg.	0,026	Seg.
99,99999%	3,15	Seg.	0,259	Seg.
99,9999%	31,54	Seg.	2,592	Seg.
99,9995%	2,63	Min.	12,960	Seg
99,999%	5,26	Min.	25,920	Seg
99,995%	26,28	Min.	2,160	Min.
99,99%	52,56	Min.	4,320	Min.
99,95%	4,38	Horas	21,600	Min.
99,9%	8,76	Horas	43,200	Min.
99,8%	17,52	Horas	1,440	Horas
99,7%	26,28	Horas	2,160	Horas
99,6%	35,04	Horas	2,880	Horas
99,5%	43,80	Horas	3,600	Horas
99,4%	52,56	Horas	4,320	Horas
99,3%	61,32	Horas	5,040	Horas
99,2%	70,08	Horas	5,760	Horas
99,1%	78,84	Horas	6,480	Horas

Em princípio, observando-se a disponibilidade típica de 99,99% oferecida por diversos provedores de *Cloud Computing*, pode-se ter a impressão de que a possibilidade de ficar com o serviço fora por no máximo 21 minutos no mês não faz grande diferença, porém, para algumas empresas, esse tempo sem operar remete a um grande prejuízo. Em agosto de 2013, uma falha deixou o portal de comércio eletrônico da Amazon.com fora do ar por aproximadamente 30 minutos. Baseados nas vendas do ano anterior no mesmo período, calculou-se o custo dessa falha, chegando-se ao valor de cerca de 2 milhões de dólares, ou seja, a Amazon.com deixou de vender cerca de US$ 66.240,00 por minuto em decorrência da falha [31].

Desta forma, empresas que pretendem estabelecer um sistema de alta disponibilidade para serviços de missão crítica devem realizar uma análise aprofundada, baseado nas regras de negócios, conforme modelo proposto em [32]:

- Análise de impacto nos negócios;
- Identificar e categorizar processos de negócios com requisitos de disponibilidade;
- Formular o custo de inatividade;
- Estabelecer o objetivo de tempo de recuperação (RTO) e do ponto de recuperação;
    - RTO – *Recovery Time Objective*: Definição de quanto tempo um sistema de TI pode ficar indisponível antes da organização sofrer consequências inaceitáveis.
- Compreender os objetivos de gerenciamento e custo total de propriedade (TCO) e retorno sobre o investimento (ROI);
    - TCO – *Total Cost of Ownership* – Refere-se a todos os custos incluídos pelo projeto.
    - ROI – *Return on Investiment* – Indicador de retorno financeiro obtido com um investimento realizado.

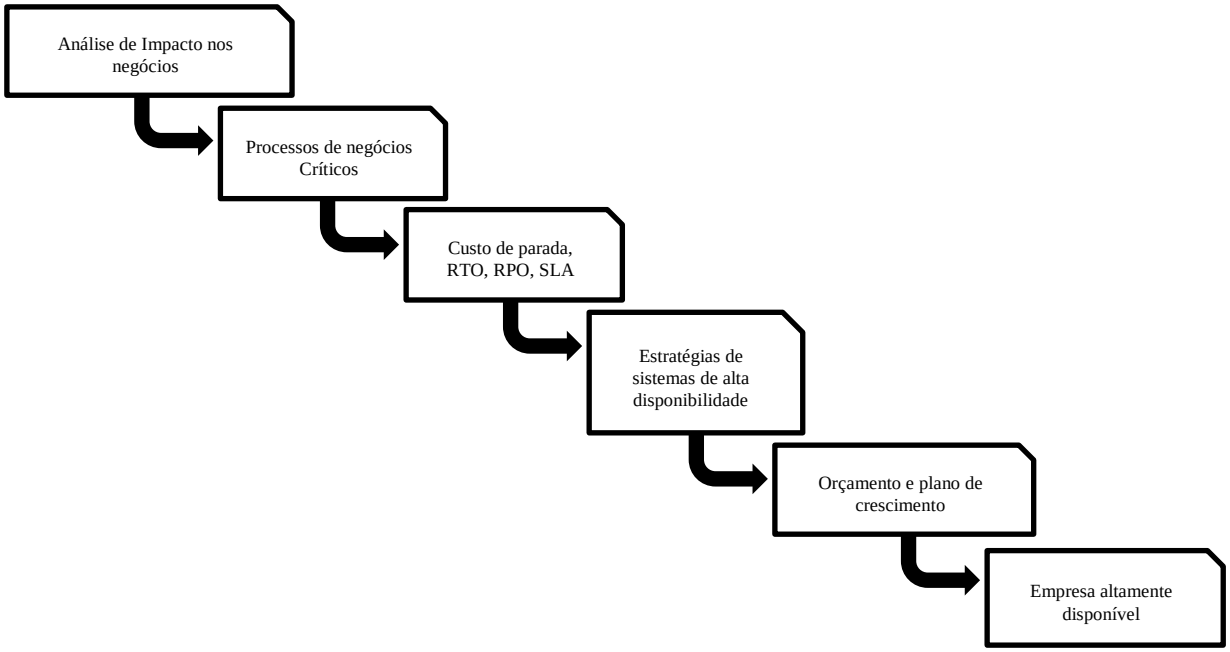


Fig. 7 – Planejamento e Implantação de sistema de alta disponibilidade [27].

Na Fig.7 vê-se o fluxograma do modelo de análise proposto por [32] para planejamento e implantação de sistema de alta disponibilidade em uma empresa.

4.2 – *Service Layer Agreement*

Entende-se por um acordo de nível de serviço (SLA), um contrato firmado entre o fornecedor de um serviço e seu usuário, no qual são formalizados os níveis de serviço sob determinadas condições, que serão vigentes durante o tempo do contrato [33].

O TeleManagement Forum (TMForum), consórcio internacional de prestadores e fornecedores de serviços de comunicação [33], define SLA como:

> *"A formal negotiated agreement between two parties, sometimes called a Service Level Guarantee. It is a contract (or part of one) that exists between the service provider and the customer, designed to create a common understanding about services, priorities, responsibilities, etc."* [33]

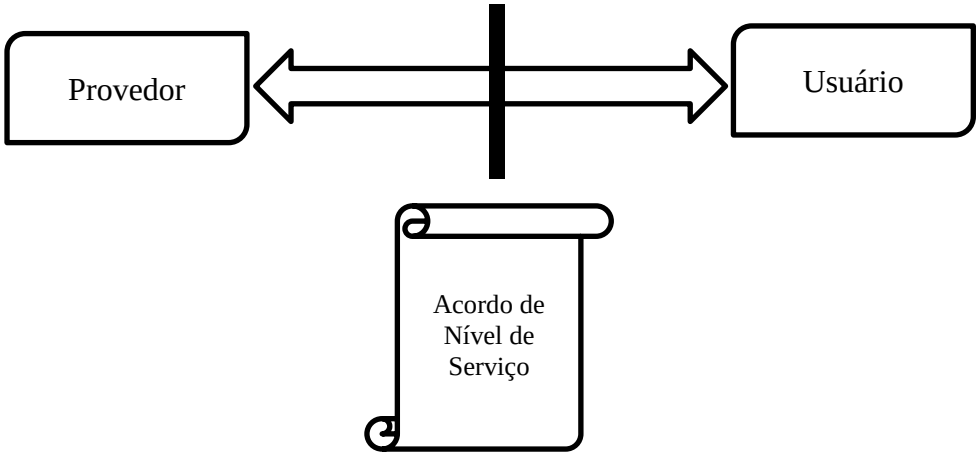


Fig. 8 - Representação esquemática de SLA – Adaptado [28].

Como exemplos de atributos excluídos dos cálculos de SLA por parte de provedores de *Cloud Computing*, podemos citar: *i*) a indisponibilidade nos serviços de *Cloud*, causada por manutenções programadas, que são informadas aos usuários com antecedência, e *ii*) falhas na rede externa, as quais, na maioria das vezes, são de responsabilidade de empresas de telecomunicações terceiras. Contudo, estes fatores afetam a taxa de disponibilidade para os usuários da *Cloud* [12]. Relatos apresentados em [34] mostram que, por diversas vezes, falhas em serviços de *Cloud* levaram a grandes períodos de inatividade e, em alguns casos, a perda de dados de seus usuários.

5 MODELO DE ALTA DISPONIBILIDADE EM *MULTI-CLOUD*

Neste capítulo, apresenta-se um modelo para provimento de alta disponibilidade baseado em ambiente *Multi-Cloud* por meio da distribuição de acessos via DNS e *Proxy* Reverso. Apresentam-se também breves revisões do Sistema de Nomes de Domínio (*Domain Name System* – DNS) e Sistemas *Proxies (*reverso e de encaminhamento), componentes necessários para construção do modelo.

5.1 - DNS

O DNS constitui-se de um banco de dados distribuído, de forma que os dados de cada segmento desta base estejam disponíveis mediante um esquema cliente-servidor. O principal propósito do DNS é a resolução de nomes de domínio em endereço IP [35].

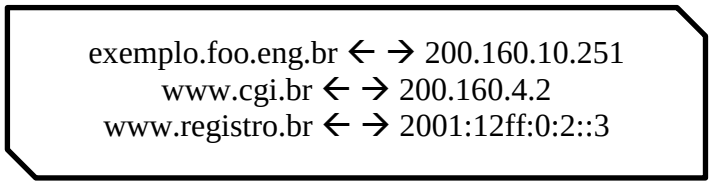


Fig. 9 – Exemplos de nomes traduzidos IP – Adaptado [30].

A base de dados de DNS armazena dados que são associados aos nomes de domínio por meio de Registros de Recursos (Fig.9), os quais são divididos em classes e tipos. Os tipos mais comuns de Registros [35] são:

- **SOA** - Indica onde começa a autoridade da zona;
- **NS** - Indica um servidor de nomes para da zona;
- **A** - Mapeamento de nome a endereço (IPv4);
- **AAAA** - Mapeamento de nome a endereço (IPv6);
- **MX** - Indica um mail *exchanger* para um nome (servidor de e-Mail);
- **CNAME** - Mapeia um nome alternativo (apelido);
- **PTR** - Mapeamento de endereço a nome;

Desta forma, quando uma solicitação de resolução de nome ocorrer para um endereço de e-Mail, o Resolver DNS buscará um Registro do tipo **MX.** Entretanto, quando a solicitação ocorrer apenas com o nome de domínio, a busca se fará com um Registro do tipo **A**.

O DNS possui uma arquitetura hierárquica de forma que os dados são dispostos em uma árvore invertida. Conforme apresentado na Fig.10, a tradução de nomes em IPs começa pela do fim para o inicio. Assim, cada “nó” (ponto) constante no nome buscado corresponde a um segmento da base de dados.

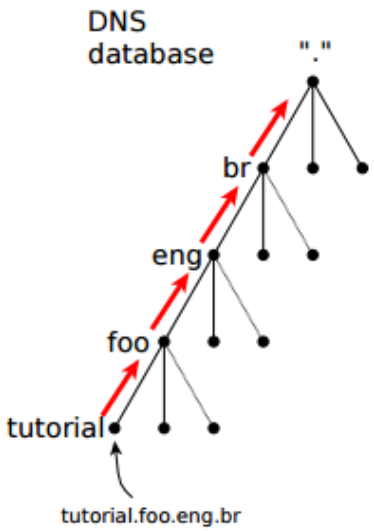


Fig. 10 – DNS Arquitetura em Árvore Invertida [30].

O fluxo de uma consulta DNS, para resolução do nome “exemplo.foo.eng.br” em seu IP “200.160.10.251” apresentado na Fig. 11, inicia-se por meio de uma consulta do “*resolver*” ao servidor DNS recursivo. Este por sua vez irá consultar o Servidor de DNS Autoritativo “.”, que irá encaminhar as referências de servidores Autoritativos responsáveis por domínios “.br”. Seguindo o fluxo da busca, o servidor Autoritativo de domínios “.br” envia ao servidor DNS recursivo, as referências de servidores responsáveis pelo domínio “foo.eng.br”. Desta forma, o DNS recursivo consegue chegar ao servidor de DNS Autoritativo que efetivamente possui a referência do *hostname* consultado. Assim, a resposta com o endereço IP é enviada ao resolver.

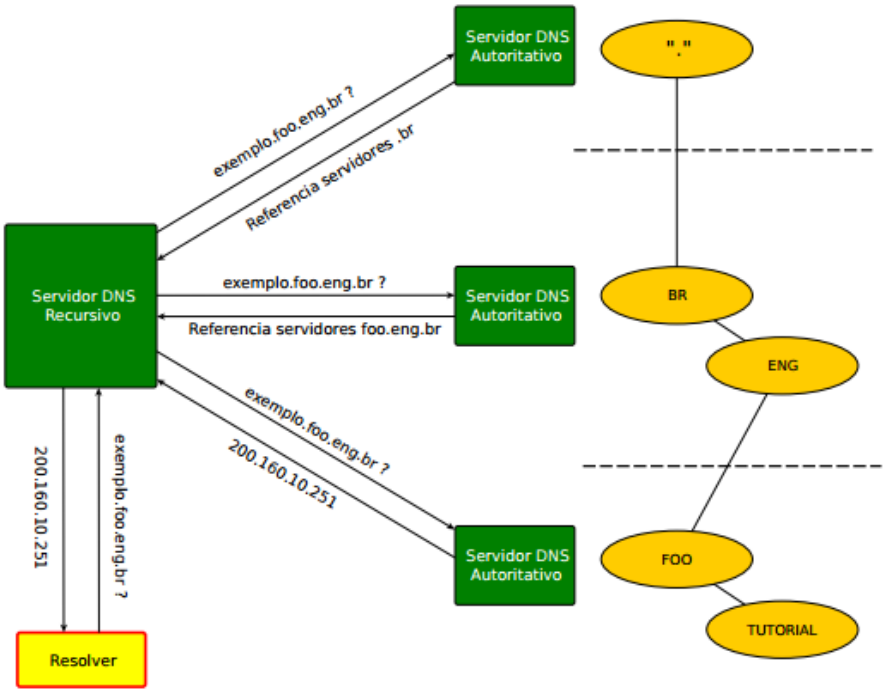


Fig. 11 – Fluxo de requisição de endereço IP [30].

5.1.2 - DNS – Suporte para Balanceamento de Carga

Apesar do principal propósito do DNS ser a resolução de nomes, outras funções podem ser executadas por este serviço. De acordo com a RFC 1794 [36], pode-se implementar no servidor DNS um sistema para o balanceamento de carga. Inicialmente, esta funcionalidade destinava-se a resolver para um mesmo nome diferentes endereços IPs de máquinas que trabalhavam *Cluster* e continham funcionalidades semelhantes. Por volta de 1986 uma série de diferentes funcionalidades para a resolução de um mesmo nome para diferentes endereços IP permitindo o balanceamento de carga foram implementadas no Servidor de DNS Berkeley Internet Name Domain (BIND), por meio do código de Marshall Rose, conhecido como “*Round Robin*” [36].

5.2 – Proxy

Um servidor *proxy* é utilizado principalmente para interligar redes de diferentes dispositivos com diferentes características. Para realizar as adequações entre as redes às quais está conectado, suas funções devem ser projetadas de

acordo com as características de cada uma delas (endereçamento IP, taxa de transferência, latência, entre outras) sejam em redes de sensores ou em redes com e sem fios [37]. Os servidores *proxies* podem sem classificados em duas categorias, conforme apresentado nos itens 5.2.1 e 5.2.2.

5.2.1 - Proxy de encaminhamento

É um sistema intermediário que permite a um cliente se conectar a um servidor em uma rede remota, a qual ele normalmente não teria acesso (Fig.12). Um *proxy* de encaminhamento também pode ser utilizado para a redução a carga do servidor e o consumo de banda entre o servidor de conteúdo e o servidor *proxy*, uma vez que pode armazenar dados localmente [38].

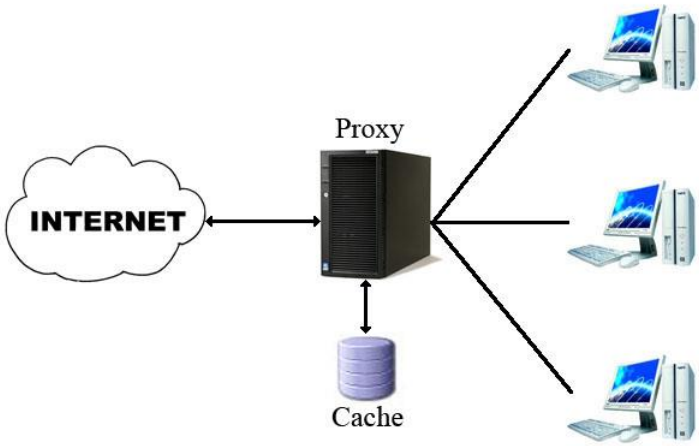


Fig. 12 – Proxy de encaminhamento.

5.2.2 - Proxy Reverso

É um sistema capaz de servir conteúdos provenientes de outros servidores, de forma que para o cliente, essa ação seja totalmente transparente como se o conteúdo acessado estivesse hospedado no próprio servidor *proxy.*

Pode ser utilizado como cache de servidores *backend,* com configurações menores, além de permitir técnicas avançadas de gestão de URL. Sistemas de *proxy* reverso também são utilizados como *Frontend* para distribuição de acessos a diversos servidores *backends* [38], conforme apresentado na Fig.13.

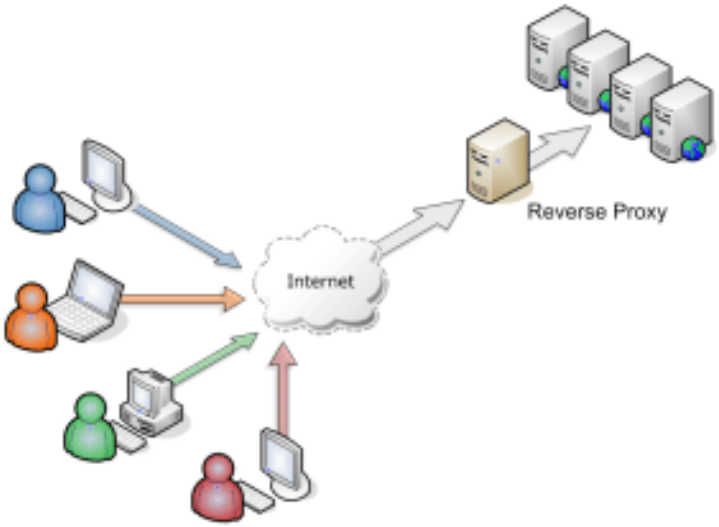


Fig. 13 – *Proxy* Reverso como *Frontend*.

5.3 – Alta Disponibilidade em *Multi-Cloud*

A utilização simultânea de duas ou mais *Clouds* como estrutura para um mesmo sistema de TI é conhecida como *Multi-Cloud* [39]. Esta solução pode ser implementada para sistemas que demandem, dentre outros quesitos, disponibilidade elevadas, muitas vezes não atingíveis se se utilizasse apenas um provedor de serviços em *Cloud*.

Sistemas *Multi-Cloud* podem tolerar falhas simultâneas em *N Clouds*, desde que o conteúdo seja replicado em *N+1 Clouds* distintas [14]. Existem estudos [40]-[41] que apresentam soluções para o espelhamento de conteúdo em diferentes C*louds*, o que torna possível a utilização desse tipo de estrutura.

A utilização de *Proxy* para a distribuição de acessos entre *Clouds* (Fig.14) utilizada em [14] é a forma mais comum em sistemas *Multi-Cloud.* Contudo, essa abordagem apresenta um inconveniente denominado como “ponto único de falha” [42].

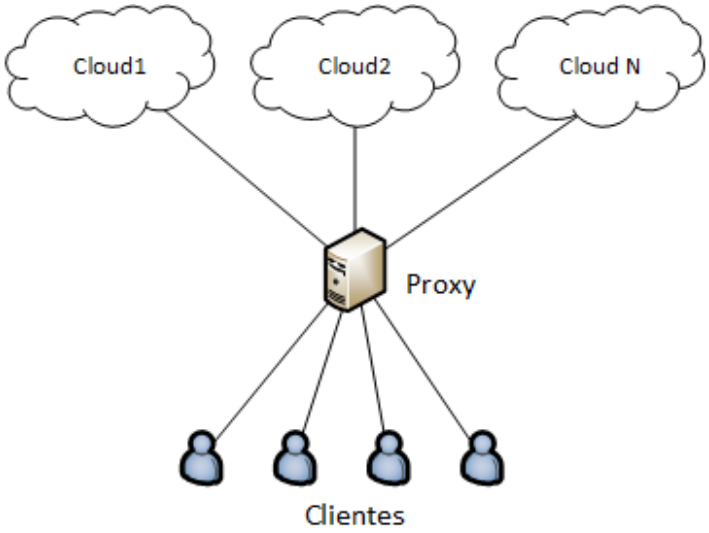


Fig. 14 – Distribuição de acessos com *proxy* simples em *Multi-Cloud*.

5.3.1 – Ponto Único de Falha

Um componente de um determinado sistema é considerado “ponto único de falha” quando em caso de mau funcionamento, afeta o sistema como um todo. Conhecidos também como “PUF” (Ponto Único de Falha), são considerados problemas de arquitetura da solução ao qual estão empregados. Este tipo de problema pode ser proveniente tanto de hardware quanto software, e afetam diretamente disponibilidade da solução em caso de falhas, além de muitas vezes serem a causa de gargalos em infraestruturas de rede [23].

Devido à sua topologia, todos os acessos passam obrigatoriamente por um único *proxy*, que distribui efetivamente os acessos entre os servidores de conteúdo. Neste cenário, em caso de falha desse componente, toda a estrutura de *Multi-Cloud* ficaria indisponível, o que neste caso, o caracteriza como um ponto único de falha.

5.3.2 – Abordagem com DNS e *Proxy* Reverso em *Multi-Cloud*

De modo a resolver o inconveniente do ponto único de falha, aumentando assim a disponibilidade total do sistema, a proposta baseia-se em um dos principais

mecanismos responsáveis pelo aumento da disponibilidade de um ambiente, a redundância de componentes [13]. Neste cenário, o serviço de *Cloud Computing* torna-se um componente da solução e não mais uma base única de suporte a um sistema.

Além da redundância de *Clouds,* conta-se com diversos servidores (*Proxy* e de conteúdo) junto às *Clouds* utilizadas de forma que todos os componentes do ambiente possuam redundância.

A redundância de componentes empregada nesta solução classifica-se no nível Ativo-Ativo, uma vez que todas as instâncias dos componentes são utilizadas de forma simultânea.

Neste ambiente, cada servidor *proxy* possui conexão direta com todos os servidores de conteúdo hospedados nos diferentes provedores de *Cloud,* formando conexões cruzadas (Fig.15). Assim, mesmo que *N Proxies* apresentem falhas, *N+1 Proxies* continuarão com a distribuição do acesso aos servidores de conteúdo, conforme apresentado em [14].

A Fig.15 mostra o diagrama geral da solução de alta disponibilidade proposta neste trabalho para um ambiente de Infraestrutura como Serviço (IaaS) em *Multi-Cloud* com múltiplos *proxies*.

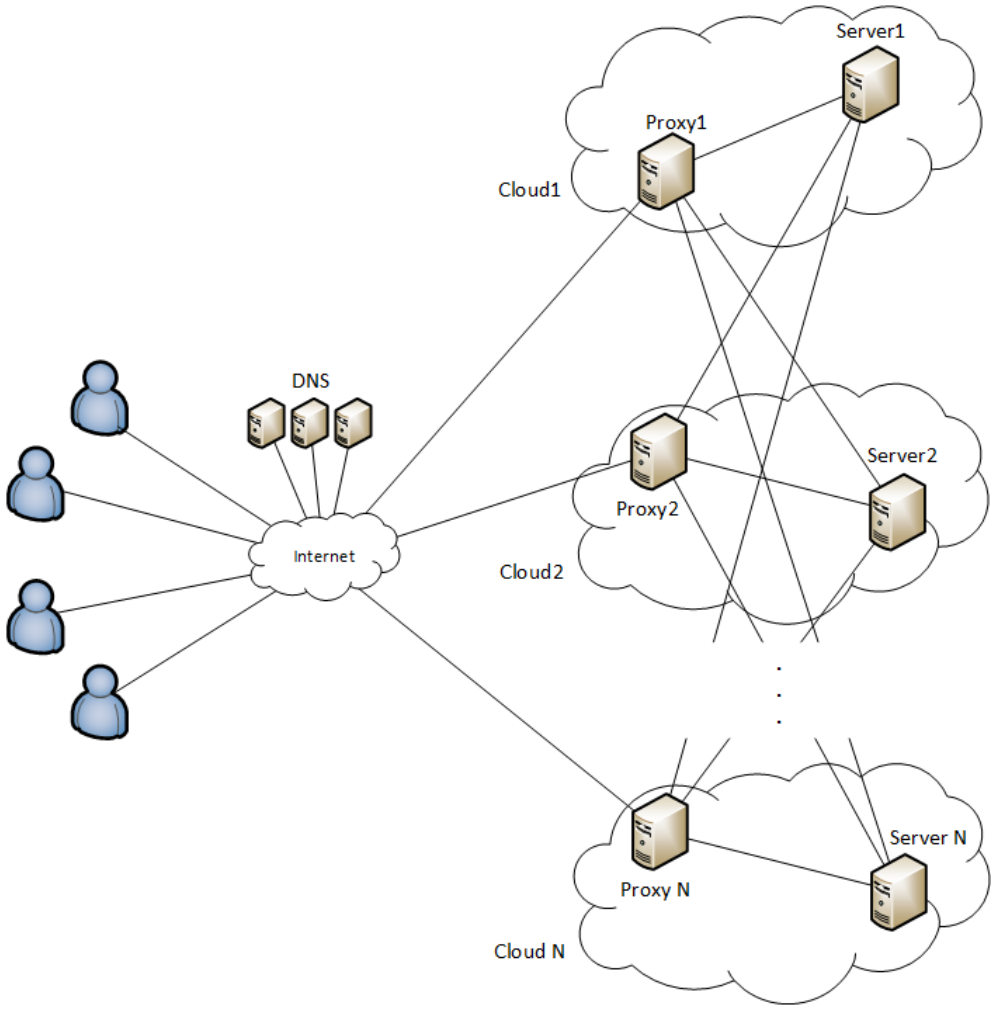


Fig. 15 – Diagrama geral da proposta. – Distribuição de acessos por DNS e Proxy Reverso.

No modelo proposto, são utilizados *N Proxies* reversos, correspondendo *N* ao número de *Clouds* utilizadas no sistema. Cada *Cloud* conta com o seu próprio *Proxy* reverso e com o seu servidor de conteúdo, de forma que todas as *Clouds* envolvidas possuam autonomia de componentes para o seu funcionamento.

A requisição de acesso do usuário passa inicialmente pelo Sistema de Nomes de Domínio (DNS), que é responsável por resolver o nome consultado pelo IP do servidor, neste caso, o IP do servidor *Proxy*. É possível, mediante funcionalidade do servidor de DNS, entregar para o mesmo nome de domínio, uma lista com diferentes endereços IPs, de forma que os registros dessa lista sejam reordenados a cada consulta [43], conforme detalhamento apresentado no item 6.1.2.

No lado do cliente, o navegador de Internet, por exemplo, recebe a lista de IPs do domínio consultado, com os endereços dos servidores *Proxies*. Caso o *Proxy* referente ao primeiro IP da lista não esteja acessível, automaticamente, o navegador tentará acessar o domínio pelo próximo IP listado na consulta ao DNS. Desta forma, faz-se com que o acesso chegue a todos os servidores *Proxies* de forma distribuída, de acordo com a lista de IPs que fora reordenada e entregue pelos servidores de DNS.

Uma vez que a requisição de acesso chega ao servidor *proxy*, ele acessa o servidor de Web e então responde ao usuário com o conteúdo desejado. Cada servidor *proxy* possui uma lista contendo todos os servidores de conteúdo do ambiente. O acesso a cada um deles ocorre de maneira sequencial, do primeiro para o último da lista e, desta forma, caso um servidor de conteúdo esteja indisponível, automaticamente o servidor *proxy* buscará o conteúdo no próximo servidor da lista.

Nos casos em que o serviço de *Cloud* de determinado provedor esteja indisponível, seja por falha ou por manutenção programada, o acesso ao conteúdo por parte do usuário continua garantido por meio de outros serviços de *Cloud* que compõem o ambiente *Multi-Cloud*.

Dado o funcionamento do sistema de alta disponibilidade proposto, pode-se elencar os possíveis cenários de falha e o comportamento da solução em cada cenário:

- **Serviço de *Cloud* indisponível**
  O cliente ao fazer uma requisição recebe incialmente a lista com os endereços IP que respondem ao domínio solicitado. Uma vez que o acesso ao primeiro endereço que corresponde ao IP do servidor *proxy* de uma *Cloud* que está indisponível, o navegador de internet do cliente realiza automaticamente o acesso pelo próximo IP da lista, conforme fluxos (a) e (b) apresentados na Fig.16.

  - Fluxo do cliente (a): O Cliente recebe a lista com os IPs referentes ao domínio que fora consultado. Por sua vez, o navegador de internet tenta estabelecer uma conexão com a *Cloud*1, por se tratar do primeiro IP da lista. Sem receber resposta da *Cloud*1, uma vez que ela está

indisponível, o navegador de internet tenta acessar o conteúdo por meio da *Cloud*2, cujo endereço é o próximo IP da lista. Desta forma, o navegador chega até o servidor *Proxy* reverso da *Cloud*2. Este servidor busca o conteúdo no servidor Web que está ao seu alcance e o envia ao cliente.

- Fluxo do cliente (b): Ao receber a lista de IPs do domínio que fora consultado, o cliente tenta fazer a conexão com a *Cloud*2, uma vez que seu endereço é o primeiro da lista que recebida da consulta do DNS. Neste caso, a conexão ocorre normalmente.

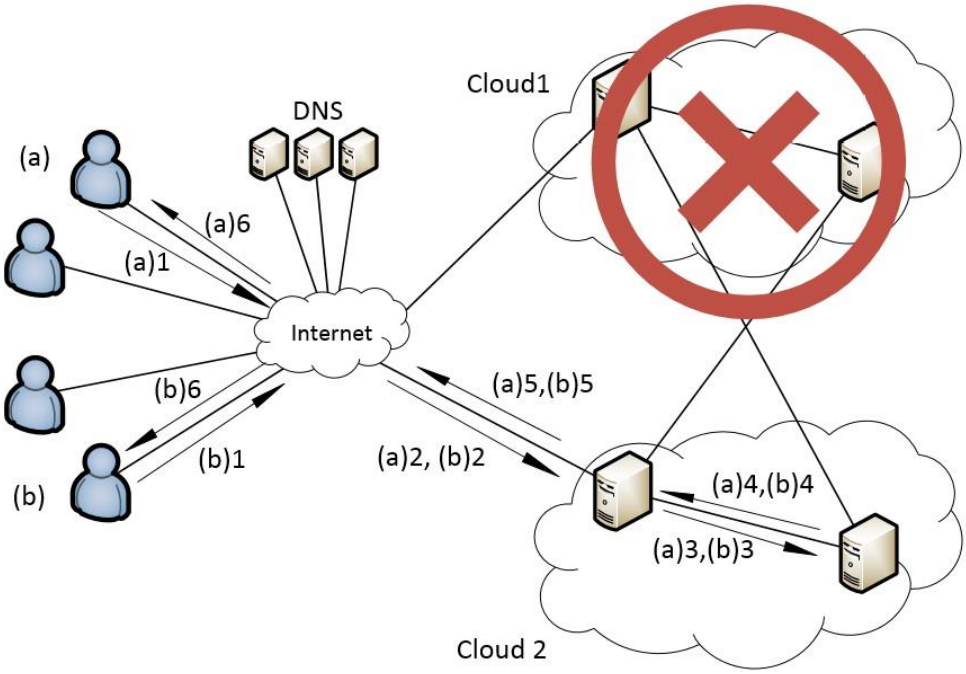


Fig. 16 – Cenário de Indisponibilidade da *Cloud* 1.

- **Servidor *Proxy* Reverso indisponível**

  Neste cenário, o processo é parecido com o que ocorre quando a *Cloud* está indisponível, uma vez que para o cliente o servidor *proxy* é a porta de entrada da *Cloud*, exceto pelo fato que o servidor de conteúdo localizado nesta *Cloud* continua atendendo requisições de outros servidores *proxies* localizados em outras *Cloud.* Este processo é ilustrado na Fig.17.

- Fluxo do cliente (a): O Cliente recebe a lista com os IPs referentes ao domínio que fora consultado. Por sua vez, o navegador de internet tenta estabelecer uma conexão com a *Cloud*1, por se tratar do primeiro IP da lista. Sem receber resposta da *Cloud*1, uma vez que o servidor *proxy* reverso que é a porta de entrada da *Cloud* está indisponível, o navegador de internet tenta acessar o conteúdo por meio da *Cloud*2, cujo endereço é o próximo IP da lista. Desta forma, o navegador chega até o servidor *Proxy* reverso da *Cloud*2. Este servidor busca o conteúdo no servidor Web que está ao seu alcance e o envia ao cliente.

- Fluxo do cliente (b): Ao receber a lista de IPs do domínio que fora consultado, o cliente tenta fazer a conexão com a *Cloud*2, uma vez que seu endereço é o primeiro da lista que recebida da consulta do DNS. A conexão é realizada com o servidor *proxy* reverso da *Cloud*2, e o mesmo busca o conteúdo desejado nos servidores Web ao seu alcance. Neste caso, o conteúdo foi obtido com o servidor *Web* da *Cloud*1, que estava disponível ao servidor *proxy* reverso da *Cloud*2.

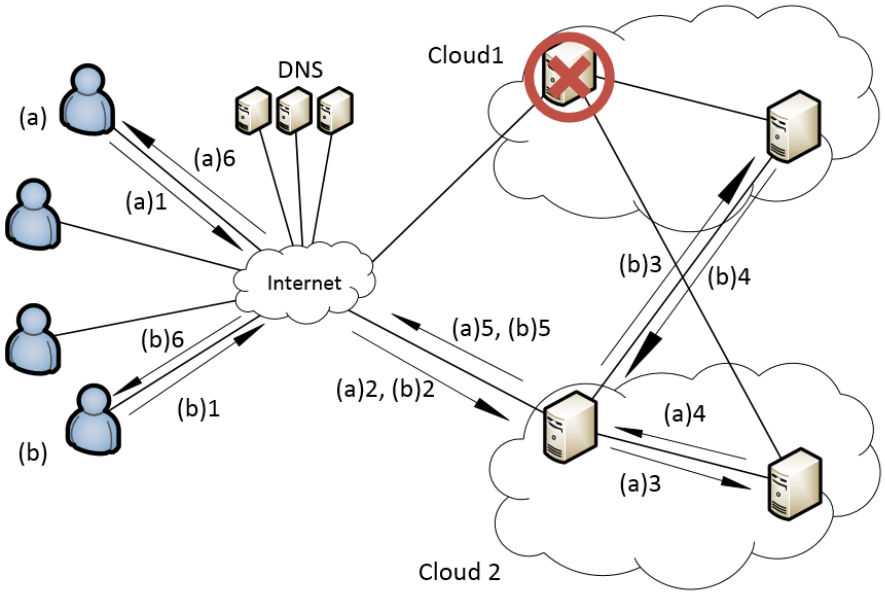


Fig. 17 – Cenário de Indisponibilidade do Servidor *Proxy*.

- **Servidor de conteúdo indisponível**

  Caso este componente apresente falha, tanto o servidor proxy localizado na mesma *Cloud*, quanto os servidores proxies localizados em outras *Clouds*, continuarão atendendo as requisições por meio dos outros servidores de conteúdos que estejam disponíveis, uma vez que possuem acesso direto a eles, conforme apresentado na Fig.18.

  - Fluxo do cliente (a): O Cliente recebe a lista com os IPs referentes ao domínio que fora consultado. Por sua vez, o navegador de internet tenta estabelecer uma conexão com a *Cloud*1, por se tratar do primeiro IP da lista. Desta forma, o navegador chega até o servidor *Proxy* reverso da *Cloud*1. Este servidor busca o conteúdo no servidor *Web* que está ao seu alcance e o envia ao cliente. Neste cenário, o servidor de conteúdo da *Cloud*1 está indisponível, então o *proxy* reverso busca o conteúdo no servidor de conteúdo da *Cloud*2 para então responder à requisição do cliente.

  - Fluxo do cliente (b): Ao receber a lista de IPs do domínio que fora consultado, o cliente tenta fazer a conexão com a *Cloud*2, uma vez que seu endereço é o primeiro da lista que recebida da consulta do DNS. Desta forma, o navegador chega até o servidor *Proxy* reverso da *Cloud*2 e o mesmo busca o conteúdo no servidor *Web* que está ao seu alcance para assim atender a requisição do cliente.

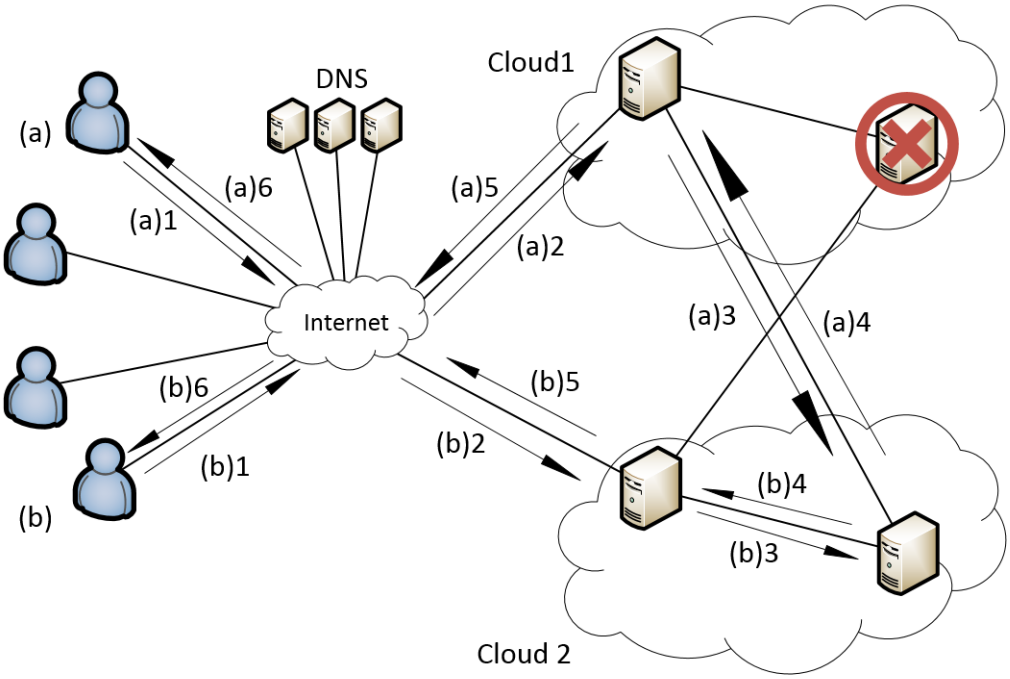


Fig. 18 – Cenário de Indisponibilidade do Servidor Web.

No que diz respeito à escalabilidade, que trata da facilidade com a qual um sistema ou componente pode ser modificada para se ajustar à demanda necessária [44], a solução mostra-se compatível ao conceito, de forma que é possível adicionar novas estruturas de *Cloud* conforme demanda, bastando-se adicionar entradas nos servidores de DNS e inserindo os servidores de conteúdo na listagem dos servidores *proxies.*

5.4 – Expressão para cálculo de disponibilidade

A disponibilidade, que trata da relação entre o tempo de atividade e o tempo de operação de um sistema de TI, pode ser calculada por meio de (1), conforme [15].

$$D_j = \frac{Ts - Tp}{Ts} \qquad (1)$$

Sendo D_j, Ts, Tp, a disponibilidade total da j-ésima *Cloud*, o tempo de serviço e o tempo de parada, respectivamente.

5.5 – Expressão proposta para cálculo de disponibilidade em *Multi-Cloud*

Para o cálculo da disponibilidade total mínima em *Multi-Cloud*, propõe-se a seguinte expressão:

$$D_{T\min} = 1 - \left(\prod_{j=1}^{N} 1 - D_j \right) \qquad (2)$$

Sendo: $D_{T\min}$ a disponibilidade total mínima do sistema, *N* o número de *Clouds,* D_j a disponibilidade individual da j-ésima *Cloud*.

Na equação (2), tem-se um produto relacionado às disponibilidades das *N* Clouds, que compõem o ambiente *Multi-Cloud.* Para o cálculo da D_T supõe-se que as D_j das *Clouds* $j = \{1,2,3,...\}$ sejam independentes entre si.

A equação para calculo de disponibilidade proposta destina-se à projetistas de soluções, como ferramenta para obtenção de valores mínimos de disponibilidade em ambientes *Multi-Cloud.*

A despeito da utilização de sistemas *Multi-Cloud* já ser uma realidade [39], no melhor de nosso conhecimento esta é a primeira vez que este tipo de solução, conjugada com DNS e *Proxy* reverso*,* é utilizada para o provimento de alta-disponibilidade em *Cloud Computing*.

6 SISTEMA EXPERIMENTAL

A fim de se testar a presente proposta, projetou-se um sistema que provesse disponibilidade DT > 99.99% a partir de *Clouds* com menor disponibilidade. Para tanto, emularam-se duas *Clouds* com disponibilidade individual máxima de 99,95%, um valor típico oferecido por provedores [12].

6.1 – Ferramentas utilizadas

Para a implementação da proposta apresentada no capítulo 5, foram utilizados softwares de licenças livre e de código aberto, de modo que, para um possível usuário da solução, não fosse acrescido nenhum tipo de custo em sua infraestrutura para funcionamento do ambiente *Multi-Cloud*, exceto o custo mensal das *Clouds* propriamente ditas, envolvidas no ambiente.

6.1.1 – BERKELEY INTERNET NANE DOMAIN – BIND

O BIND, também conhecido como "*named*" é uma implementação dos protocolos de DNS por meio de um software Cliente-Servidor de código aberto. Fornece uma plataforma estável para aplicações de sistemas distribuídos e resolução de nomes de domínio [45].

6.1.2 – BIND ROUND ROBIN

A implementação do servidor DNS Bind, oferece uma funcionalidade básica para o balanceamento de carga que pode ser utilizada com múltiplos registros *(Resource Record Data)* do mesmo tipo (**A** por exemplo) para o mesmo nome (www), conforme demonstrado na Fig.19.

Name	TTL	CLASS	TYPE	Resource Record (RR) Data
www	600	IN	A	10.0.0.1
	600	IN	A	10.0.0.2
	600	IN	A	10.0.0.3

Fig. 19 - Configuração de múltiplas entradas para o mesmo nome [33].

Quando uma requisição de resolução do nome “www” for solicitada, o servidor rotacionará os registros e responderá à solicitação com os endereços IPs de forma alternada. No caso dos registros do exemplo da Fig.19, seriam entregues nas seguintes ordens, para diferentes requisições, conforme disposto na tabela 2.

Tabela 2: Ordenação de respostas DNS

Requisição	**Ordem**
1º	1, 2, 3
2º	2, 3, 1
3º	3, 2, 1
4º	3, 1, 2

Esta ordenação é realizada por meio do algoritmo conhecido como Round Robin, e executado pelo Bind9 por meio da chamada da seguinte função [43]:

```
rrset-order {
        class IN type A name "host.example.com" order random;
        order cyclic; };
```

Nesta abordagem, o termo balanceamento de carga não é utilizado, uma vez que esta função do DNS não leva em consideração a carga dos servidores envolvidos para fazer a ordenação dos endereços IP, na resolução dos nomes. Portanto, utilizamos o termo “distribuição de acessos” que melhor se enquadra na abordagem realizada no presente estudo.

6.1.3 – Servidores Web

Segundo o NIST, em “*Guidelines on Securing Public Web Servers*” [46], a World Wide Web (www), de modo básico pode ser dividida em dois componentes principais:

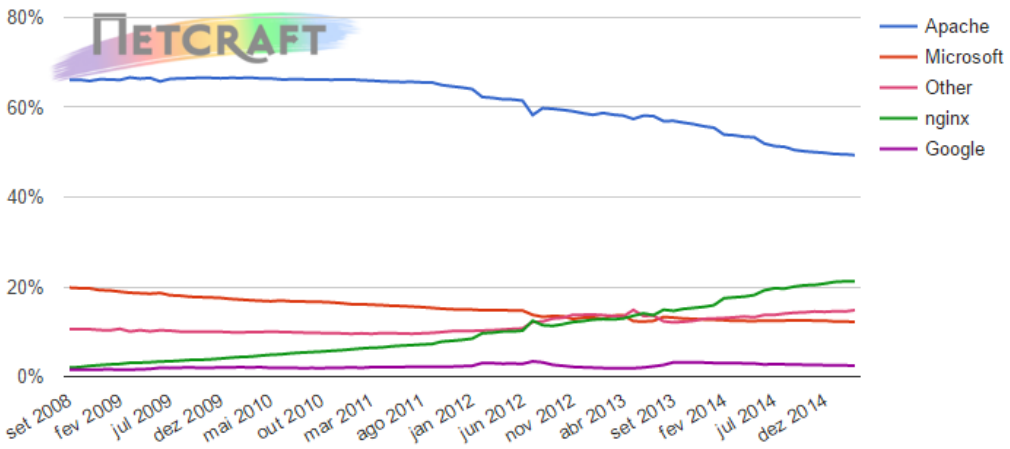
- Servidores Web

  Aplicações que permitem que a informação seja disponibilizada na internet (essencialmente publicam as informações), por meio do Protocolo de Transferência de Hipertexto (HTTP) ou pelo Protocolo de Transferência de Hipertexto Seguro (HTTPS).
- Navegadores Web

  São aplicações clientes utilizadas para que usuários possam acessar e visualizar as informações armazenadas em Servidores Web.

Existem no mercado diversos Servidores Web com licenças livres, como o Apache2 [47], Nginx [48], Lighttpd [49], Cherokee [50], e Servidores Web pagos, como o Microsoft IIS [51], IBM HTTP Server [52], entre outros.

Em 2015, [53] divulgou dados de uma pesquisa sobre os servidores web utilizados em quase um milhão de sites mais acessados. O resultado percentual de cada desenvolvedor é exibido no gráfico da Fig.20. Destaca-se neste levantamento, o quão mais utilizado é o Apache WebServer, que detém um percentual de utilização maior do que a soma de todos os seus concorrentes.



Developer	February 2015	Percent	March 2015	Percent	Change
Apache	495,321	49.53%	493,463	49.35%	-0.19
nginx	212,141	21.21%	212,151	21.22%	0.00
Microsoft	122,540	12.25%	122,069	12.21%	-0.05
Google	24,918	2.49%	24,434	2.44%	-0.05

Fig. 20 – Servidores Web – Participação de mercado nos principais sites [42].

Desta forma, foram utilizados no ambiente experimentação da proposta, dois servidores web de licença livre, que também são os mais utilizados no mercado, apresentados a seguir:

6.1.3.1 – Apache2

O projeto do apache foi iniciado em 1992 por um grupo de oito pessoas que queriam acrescentar funcionalidades e corrigir problemas existentes no código de outro projeto da NCSA (*Nacional Center for Supercomputing Activies*) chamado NCSA HTTPD [54]. Randy Terbush, um os integrantes do grupo, escreveu o *Apache Software License* baseado no *BSD Software License*, garantindo que o Apache *Server* seria livre e de código aberto. Em 1995, a versão do Apache Server 0.6.2 foi liberada [54]. Segundo [53] em Dezembro de 2014, o Apache2 era utilizado como servidor em 49% dos quase um milhão de sites mais acessados.

6.1.3.2 – Nginx

O Servidor Nginx é uma combinação de Servidor *Web*, *Proxy* com cacheamento e balanceador de carga para sites web [55]. O código do Nginx foi escrito por Igor Sysoev, engenheiro de *software* Russo. Em 2004, Igor passou a oferecer o Nginx para a comunidade de código aberto. Segundo [53], em dezembro de 2014, o Servidor Nginx era utilizado como servidor em 21% dos um milhão de *sites* mais acessados.

6.2 – Ambiente de Virtualização

O ambiente de experimentação da proposta foi executado sob uma plataforma de virtualização, de modo que todos os componentes necessários para a realização dos testes puderam ser executados nos hardware disponíveis. Assim, utilizou-se o software de virtualização Oracle Virtualbox [56], cujo painel de administração é exibido na Fig.21.

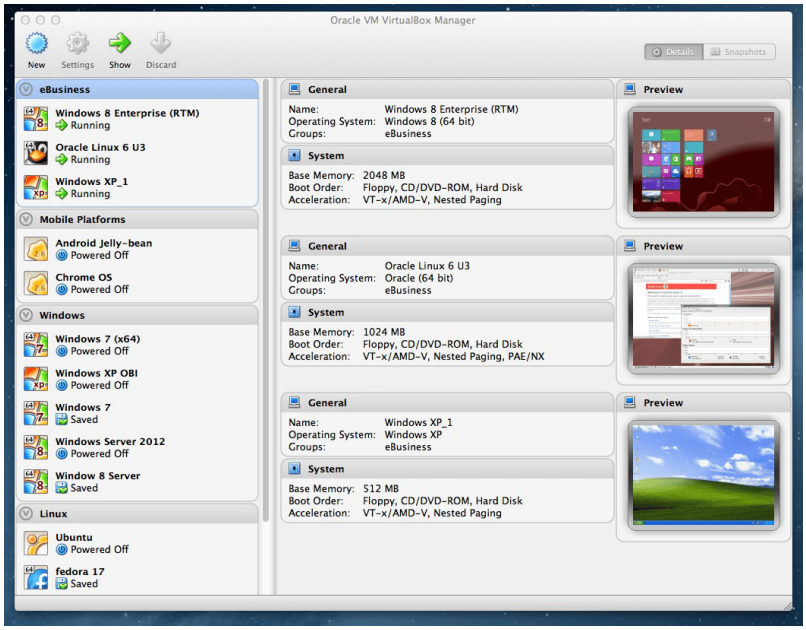


Fig. 21 – Painel de administração do Oracle Virtualbox.

6.3 – Sistema Operacional Base

Sistema operacional pode ser definido como coleções de programas computacionais, que coordenam a operação do *hardware* e *software* do computador [57]. O programa responsável pela coordenação mencionada é chamado de *Kernel*, considerado o coração do sistema operacional. Ele é responsável, dentre outros, por fazer a interface entre os programas e os dispositivos físicos, controlar a execução de processos, alocar recursos de memória para execução dos processos, entre outros [57].

Tanto os servidores virtuais quanto os computadores hospedeiros executavam como sistema operacional a distribuição Ubuntu [58], com Kernel V3.13.0-32.

6.4 – Software de Monitoramento

Para monitoramento dos sistemas envolvidos nos testes, utilizou-se a ferramenta de monitoramento Munin [59]. A ferramenta produz gráficos (Fig.22) por meio do RRDTool [60], a partir de informações coletadas via rede ou com a utilização de *plug-ins* para monitoramento de softwares específicos, como é o caso

dos servidores *web*, onde foram necessário instalações de agentes para coleta de informações de acesso às paginas hospedadas.

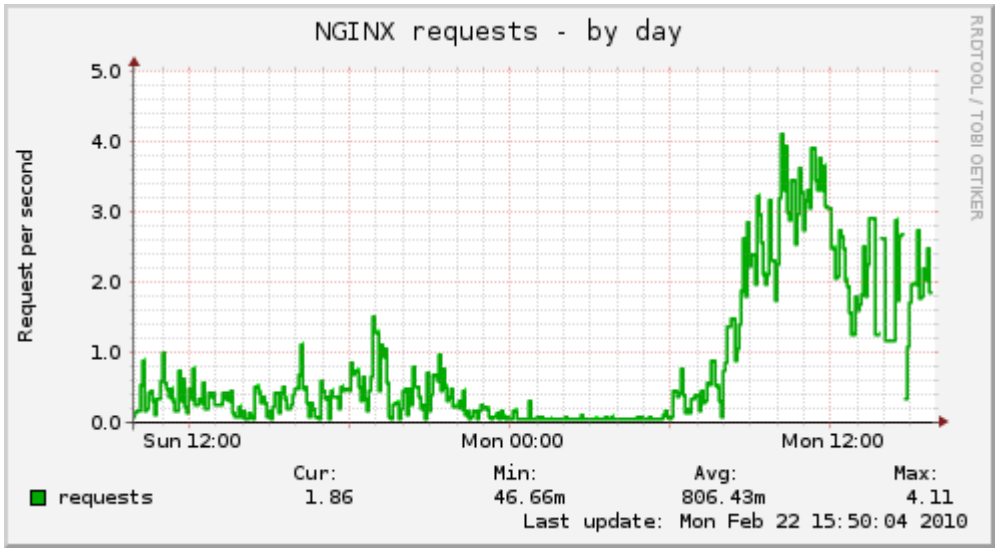


Fig. 22 - Exemplo de Monitoramento do Nginx com Munin

6.5 – Emulação de rede

No ambiente de experimentação foi necessário emular propriedades de redes de longa distância, uma vez que em um cenário real de *Multi-Cloud*, existem latências diferentes no acesso às diferentes *Clouds* que compõem o sistema. Dessa foram, foi utilizado o Network Emulator (NETEM) [61].

6.5.1 – Network Emulator (NETEM)

Netem provê emulação de funcionalidades para testes emulando propriedades de redes de longa distância, como [61]:

- **Variação de latência**

  Refere-se ao tempo que o primeiro bit leva para ir do cliente ao servidor em uma comunicação via rede [62].

  A Fig. 23 mostra o comando do NETEM para emulação da latência.

```
# tc qdisc add dev eth0 root netem delay 100ms
```

Fig. 23 – NETEM: Comando para inserção de latência.

- **Perda de pacotes**

  Refere-se ao percentual de pacotes perdidos em trânsito entre o cliente e o servido.

  A Fig. 24 mostra o comando NETEM para inserção percentual de perda de pacotes em uma comunicação de rede.

```
# tc qdisc change dev eth0 root netem loss 0.1%
```

Fig. 24 – NETEM: Comando para inserção de perda de pacotes.

- **Duplicação de pacotes**

  Refere-se ao percentual de pacotes duplicados em uma comunicação de redes.

  A Fig.25 mostra o comando NETEM para inserção percentual de pacotes duplicados.

```
# tc qdisc change dev eth0 root netem duplicate 1%
```

Fig. 25 – NETEM: Comando para inserção de percentual de pacotes duplicados.

- **Reordenação de pacotes:**

  Refere-se ao percentual de pacotes que serão enviados de forma reordenada.

  A Fig.26 mostra o comando NETEM para inserção percentual de reordenação de pacotes.

```
# tc qdisc change dev eth0 root netem delay 10ms reorder 25% 50%
```

Fig. 26 – NETEM: Comando para inserção de reordenação de pacotes.

O *Network Emulator* é uma característica implementada no *Kernel* do Linux desde sua versão 2.6 e pode ser controlada por linha de comando através do comado “tc”, conforme exemplos são visualizados nas imagens 23, 24, 25 e 26.

6.6 – Ferramenta de Benchmark

Para averiguar se os componentes inseridos em nosso ambiente de *multi-cloud* gerariam *overload* na experiência de uso dos serviços providos pelo nosso ambiente, foram realizados testes de benchmark afim de medir o tempo de resposta dos serviços com e sem a distribuição por DNS e *proxy* reverso utilizados na estrutura proposta. Para tanto, utilizou-se o Apache Benchmark (AB).

6.6.1 – Apache Benchmark (AB)

A ferramenta de *benckmark* desenvolvida pela Apache Foundation [63], é utilizada para aferição de servidores HTTP. Pode-se gerar requisições simultâneas afim de testar-se qual o limite de requisições máximo o servidor web é capaz de responder [63].

```
# ab -n 100 -c 10 http://www.ha.lab.local
```

Fig. 27 – Comando para execução o Apache Benchmark.

Por meio do comando visualizado na Fig.27, testa-se um servidor web que responde pelo nome de www.ha.lab.local. Os parâmetros dos testes estão dispostos abaixo:

- ab: comando binário.
- -n *requests*: número de requisições que serão executadas na sessão.
- -c *concurrency:* número de requisições que serão executadas ao mesmo tempo.

Adicionando-se ao comando o parâmetro “-g” a execução do *Benchmark* gera um arquivo de saída no padrão aceito pelo *software* GNUPlot, de modo que podem ser gerados gráficos dos resultados obtidos, conforme exemplo da Fig.28.

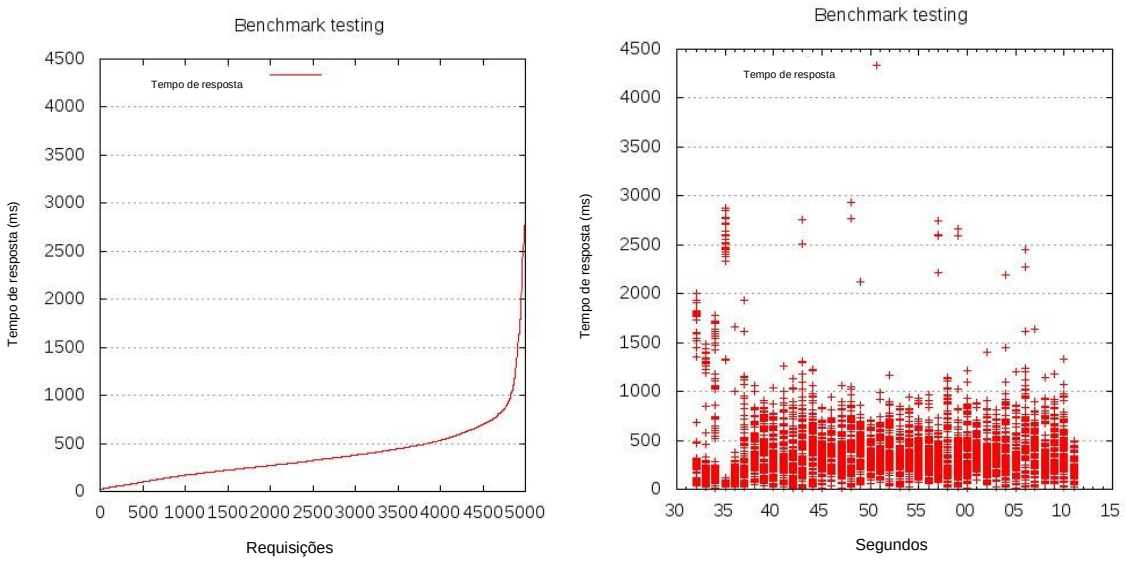


Fig. 28 – Exemplos de gráficos gerados com AB + GNUPlot.

6.7 – Funcionamento do Ambiente

De forma a distribuir o acesso entre as *Clouds*, utilizou-se nos testes uma função do sistema de nomes de domínio (DNS - *Domain Name System*) conhecida como *Round Robin* [36]-[64] implementada no Servidor *DNS Bind*, que responde a cada solicitação de consulta com uma lista de endereços IP, em ordem aleatória, para determinado nome de domínio. Desta forma, um cliente ao realizar uma consulta ao DNS receberá uma lista de IPs correspondentes e, a cada nova consulta e, novo cliente, essa lista será entregue com os endereços IPs colocados de forma aleatória. Permite-se, assim, que os acessos sejam distribuídos a diversos servidores de conteúdo, evitando-se a limitação imposta pelo ponto único de falha presente em outros sistemas.

A implementação do ambiente de experimentação do método, cujo diagrama é mostrado na Fig.29, foi realizada utilizando-se a ferramenta de virtualização Oracle Virtualbox [56]. Com esta, foi possível se emular todos os componentes de uma instância IaaS, em um microcomputador com sistema operacional Linux *Based Kernel 3.16.0-31* [58].

Uma vez configurado o servidor *Bind*, com a função *Round Robin* (DNS-RR), distribui-se para consultas ao nome de domínio utilizado (no caso específico deste trabalho, *www.ha.lab.local*) dois endereços IP, que remetem aos servidores *Proxy1 e Proxy2,* localizados um em cada *Cloud,* que executavam o Servidor Nginx [48], configurados como *Proxy* reverso. A função do *Proxy* é a de receber a solicitação do cliente e repassá-la aos servidores de conteúdo *Server1* e *Server2*, que executam o Servidor *Web* Apache2 [47]. Cada servidor *Proxy* foi ligado diretamente, via LAN *(Local Area Network)*, a um servidor de conteúdo localizado na mesma *Cloud*, e, via WAN *(Wide Area Network)*, a outro servidor de conteúdo localizado em outra *Cloud*. Desta forma, cada *Proxy* consegue direcionar as requisições de acesso dos clientes aos seus servidores, formando uma conexão cruzada.

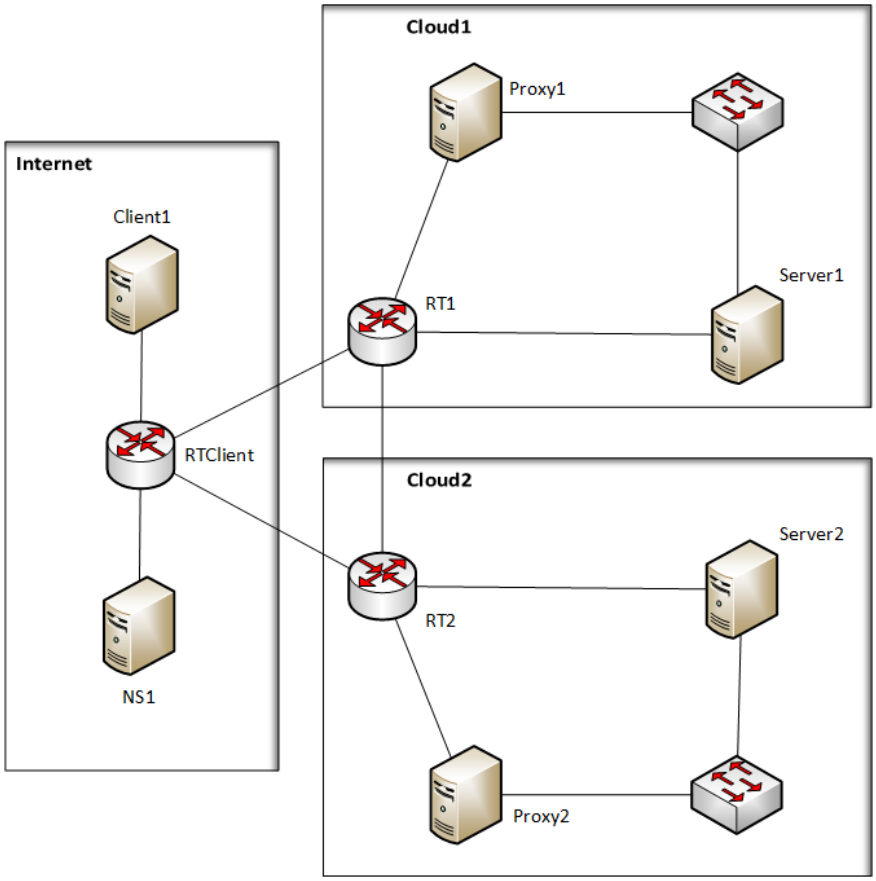


Fig. 29 – Diagrama de Rede – Utilizado para o testabilidade da proposta.

Emularam-se no ambiente de experimentação duas *Clouds*, sendo que a *Cloud1* tinha disponibilidade de 99,49% e a *Cloud2* de 99,43%. Utilizando-se a equação (2), calculou-se que esse sistema *Multi-Cloud* proveria uma $D_{T\min}$ = 99,9971%.

Em nosso experimento, utilizaram-se as máquinas virtuais (VM), conforme Tabela 3:

Tabela 3: Relação de Máquinas Virtuais.

VM	Função	Em execução
Server1	Servidor de aplicação – *Cloud1*	Apache Server V2.4.7
Server2	Servidor de aplicação – *Cloud2*	Apache Server V2.4.7
Proxy1	Servidor Proxy Reverso – *Cloud1*	Nginx V1.4.6
Proxy2	Servidor Proxy Reverso – *Cloud2*	Nginx V1.4.6
RT1	Roteador *Cloud1*	Roteamento do Kernel Linux V3.13.0-32, NetEm
RT2	Roteador *Cloud2*	Roteamento do Kernel Linux V3.13.0-32, NetEm
NS1	Sistema de Nomes de Domínio (DNS)	Bind V9 com DNS-RR
RTClient	Roteador de acesso cliente	Roteamento do Kernel Linux V3.13.0-32, Netem
Client1	Cliente gerador de requisições	Apache Benchmark V2.4.7

7 TESTES DE DESEMPENHO E DISPONIBILIDADE

Para simular o acesso a diferentes serviços de *Cloud*, foram utilizados recursos do NetEm [61], implementados no *Kernel* do Linux, com o qual torna-se possível inserir atrasos, perda de pacotes, *jitter* e outras características típicas de um sistema de redes de telecomunicação [61]. Neste caso, no roteador que permite acesso à *Cloud*1, foi inserido um atraso na conexão da ordem de 24 ms e, no roteador de acesso à *Cloud*2, um atraso de 140 ms. Tais atrasos foram baseados na ferramenta *Cloudping* [65], que testa a latência da conexão a diversos serviços de *Cloud Computing*. Os tempos de atraso utilizados remetem ao tempo de resposta de uma *Cloud* localizada a 100 km de onde o acesso ao sistema foi originado e de outra localizada a 7500 km, emulando um sistema distribuído entre diferentes países.

De forma a medir o tempo de resposta dos servidores, nas diferentes *Clouds*, utilizou-se a ferramenta Apache Benchmark para servidores *Web* [63].

Na Fig. 30, apresentam-se os resultados de um teste do Apache Benchmark realizado sobre o *Proxy* da *Cloud*1. Pode-se observar a diferença entre os tempos de resposta sem e com a distribuição por *Proxy reverso* entre as *Clouds*. Em média, baseando-se nos resultados dos testes do Apache Benchmark, distribuindo-se os acessos, o tempo médio de resposta foi 18% menor e o tempo total da conexão (tempo de processamento + tempo de espera) foi 33% menor. Esta melhora no tempo de resposta não foi observada realizando-se o mesmo teste na *Cloud*2, que apresenta atraso na conexão de 140ms. Neste caso, em média, praticamente não houve diferença no tempo de acesso com o *Proxy* distribuindo as requisições entre as *Clouds*, conforme se mostra na Fig.31. Observa-se que o fato de não haver diminuição no tempo de resposta para a *Cloud*2, ao contrário do que aconteceu com a *Cloud*1, deve-se à maior latência na conexão, não interferindo na disponibilidade. Buscou-se com esta parte do experimento, mostrar que a distribuição de acessos não penaliza o tempo de respostas das conexões com relação ao tempo típico de resposta das *Clouds.* Na verdade, em alguns casos, com o da *Cloud1*, pode até melhorá-lo.

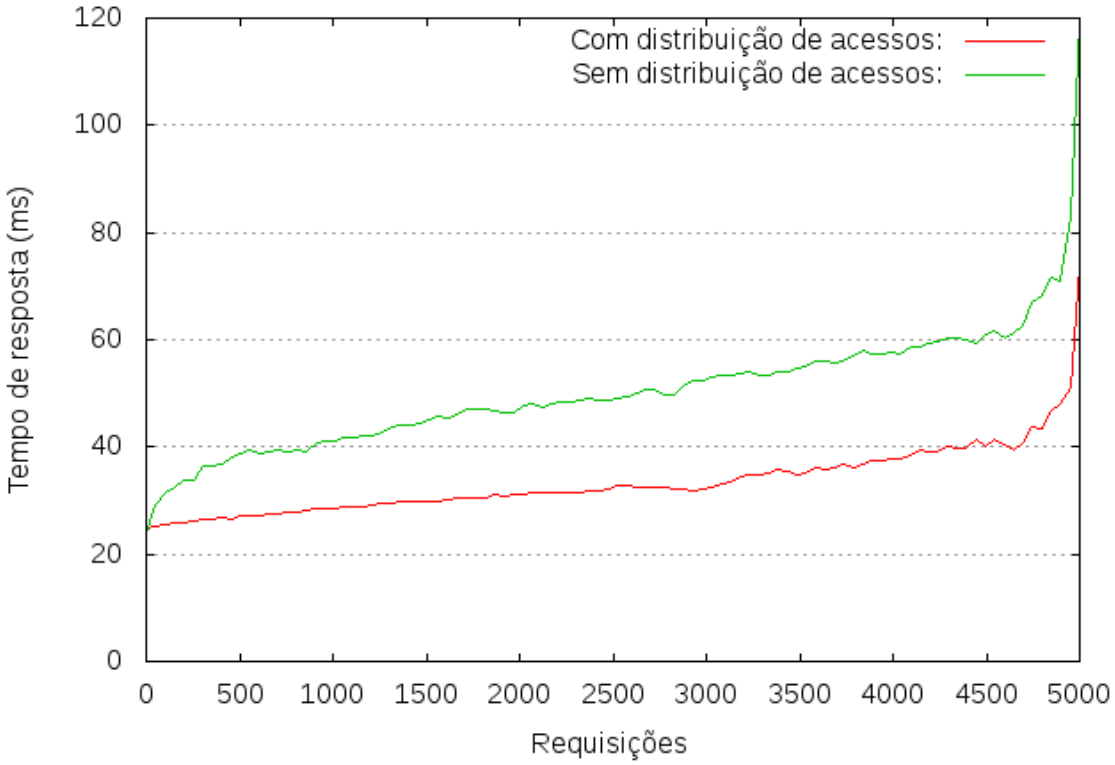


Fig. 30 - Resultado do Benchmark – *Cloud*1.

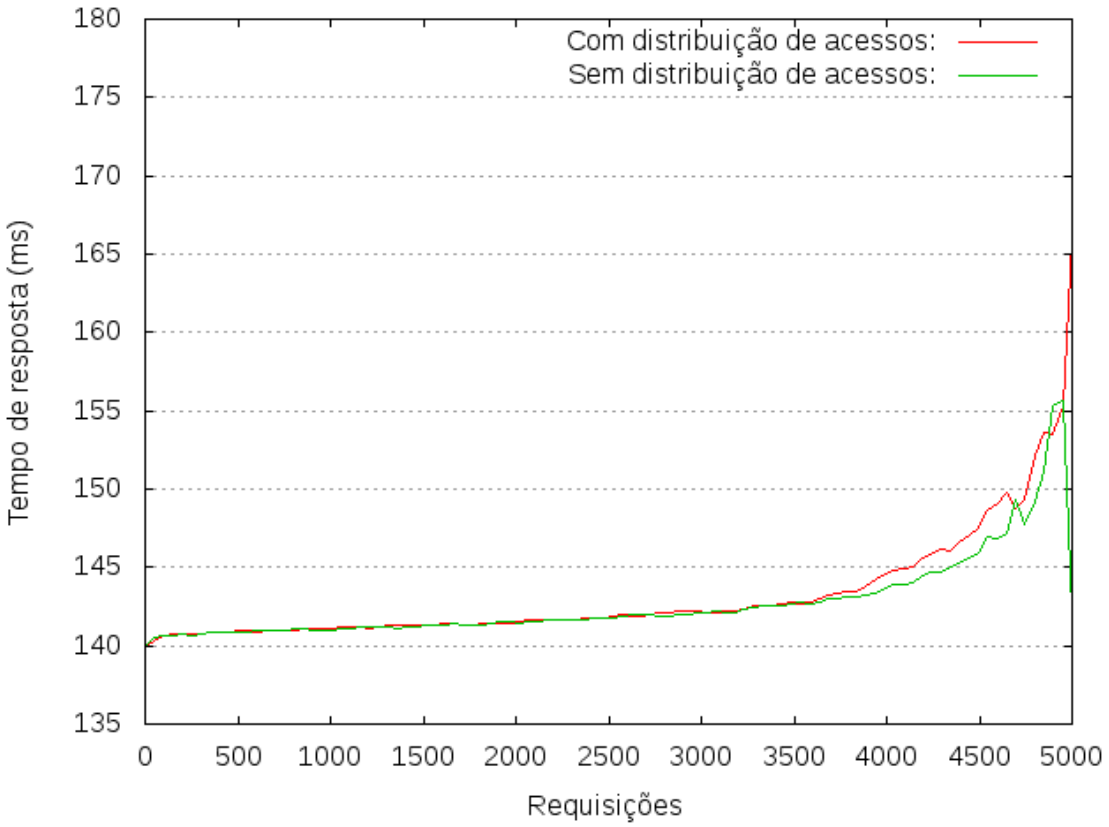


Fig. 31 - Resultado do Benchmark – *Cloud*2.

Na sequência, realizou-se um teste de 24 horas, ao longo do qual geraram-se requisições aleatórias aos servidores mediante o Apache Benchmark, de acordo com o **Algoritmo1**. Os parâmetros referentes ao número de conexões totais e o número de requisições simultâneas foram determinados aleatoriamente, utilizando-se a função RANDOM do *Kernel* do Linux [66], para simular o tráfego do sistema.

Algoritmo 1. GeradorRequisições

```
1 inicio
2   enquanto verdadeiro; faça
3       execute benchmark com número de concorrência
        aleatória entre 30 e 150 com conexões totais de
        200 a 15000 em http://www.ha.lab.local
4 aguarde por um tempo aleatório entre 1 e 3 segundos
5 fim
```

Utilizando-se do **Algoritmo 2,** emularam-se falhas de forma independente e aleatória, que causaram interrupções nos serviços, tanto nos servidores *Proxies* como nos servidores de conteúdo.

Algoritmo 2. GeradorInterrupção

```
1 início
2   aguarde tempo aleatório entre 3600 e 86400
        segundos.
3     se o serviço http estiver em execução então
4       pare o serviço http
5         aguarde um tempo entre 330 e 600 segundos
6       inicie o serviço http
7    se não
8       saia
9 fim
```

Durante o período de testes, coletaram-se os dados de acessos aos servidores de conteúdo, por meio do *software* de monitoramento *Munin Monitoring* [59], com os resultados mostrados nas Fig. 31(a) e Fig. 31(b), respectivamente, para as *Clouds* 1 e 2.

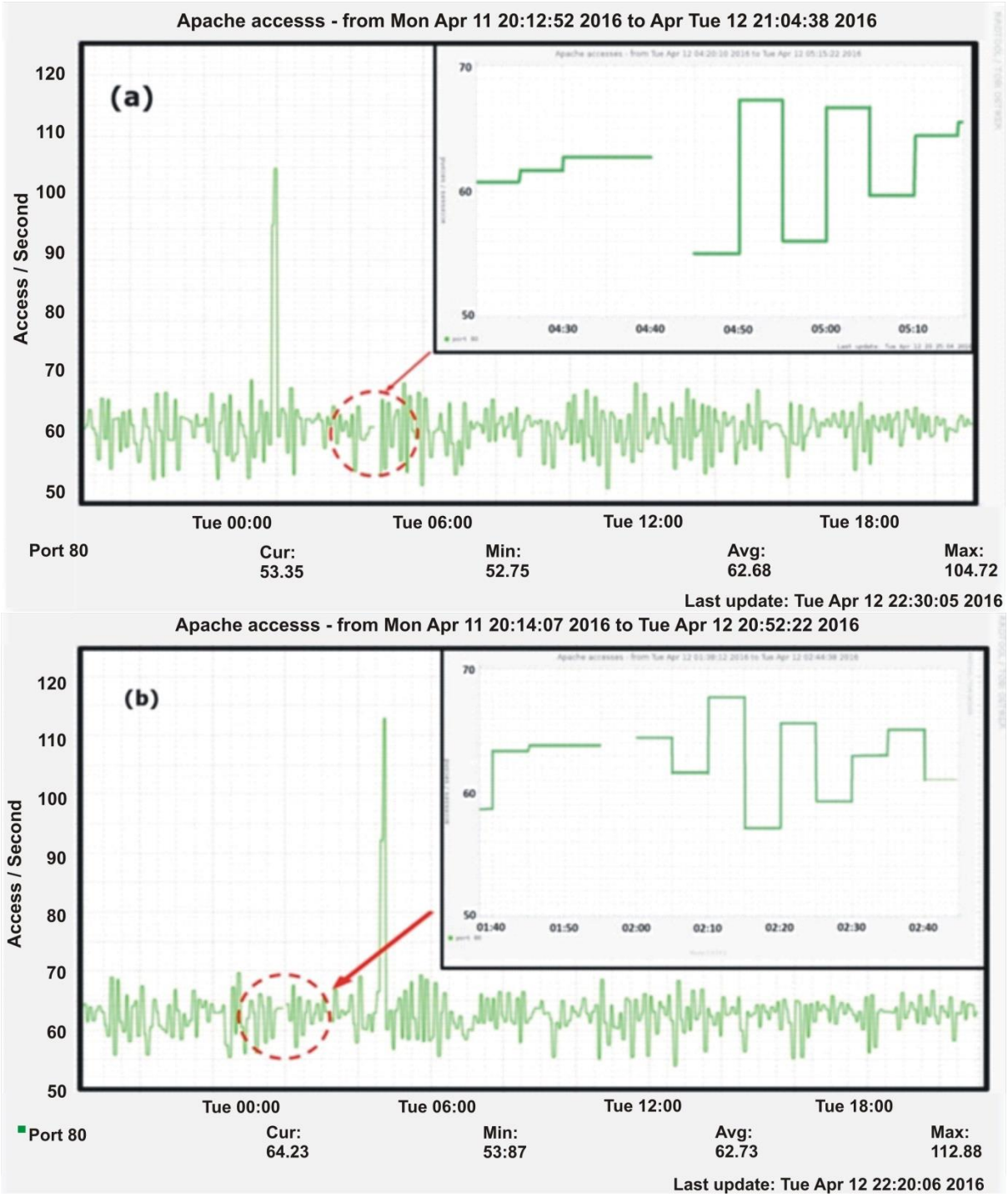


Fig. 32 - Servidores de Aplicação – Cloud1 e 2 – 24 horas.

A variação no número de acessos por segundo ao servidor da *Cloud1* (Fig. 32(a)), manteve-se entre 55 e 70 acessos, exceto às 01h55m, quando se nota um pico, correspondendo a 105 acessos.

Cenário semelhante foi evidenciado no monitoramento da *Cloud2* (Fig. 32(b)), sendo que o pico de acesso aconteceu às 04h40m, chegando a 112 acessos.

Desta forma, destacam-se dois momentos: *i*) às 01h55m registrou-se um aumento nos acessos ao Servidor de Aplicação da *Cloud*1, concomitantemente, à interrupção de acesso ao servidor de aplicação da *Cloud*2, (detalhe da Fig. 32(b)). Pode-se verificar por meio do arquivo de log gerado que neste período foi promovida uma falha no serviço que executa o Apache Web Server na *Cloud*2, deixando-o indisponível por 08:12 minutos (Fig. 33).

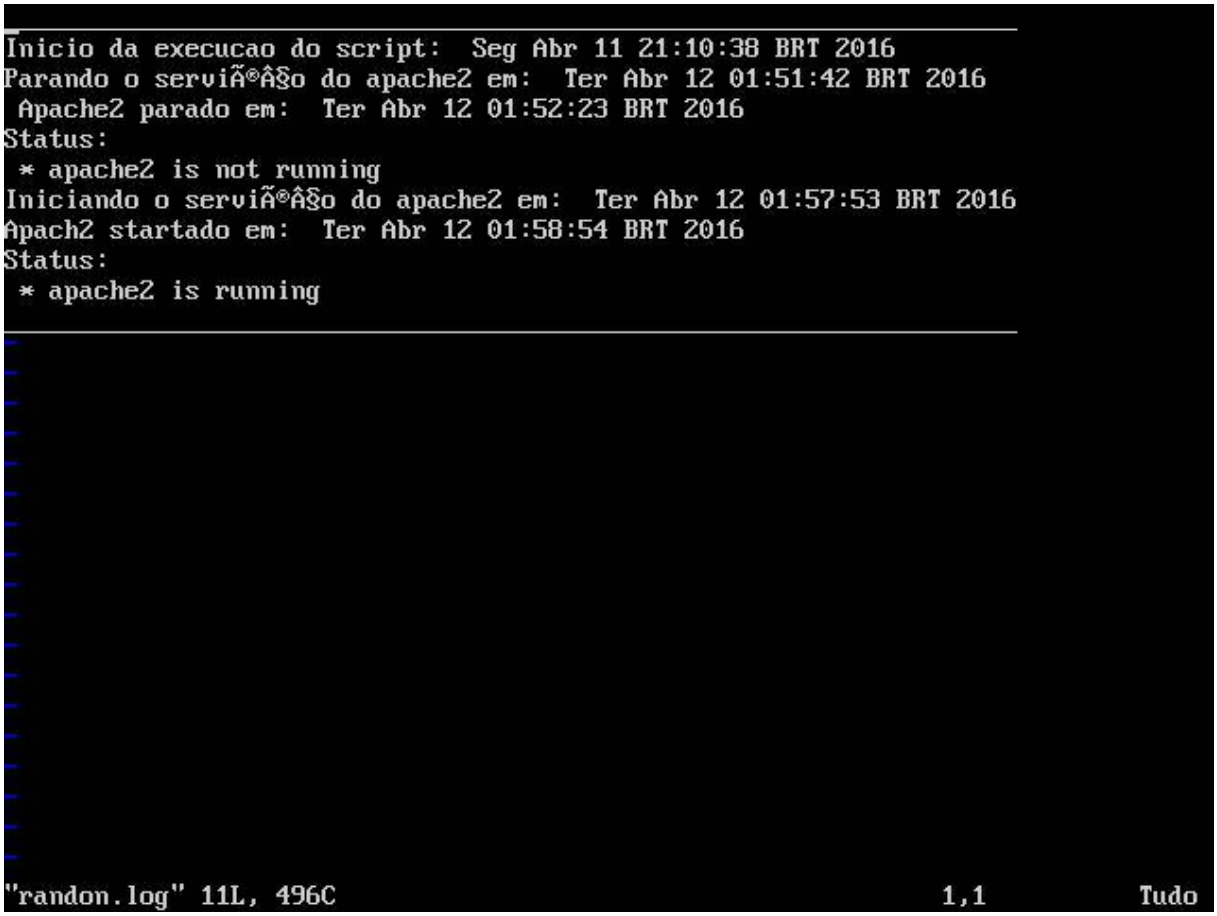


Fig. 33 - Log do *Script* gerador de Interrupção - *Cloud2*.

ii) às 04h40m o serviço *web* do Servidor de aplicação da *Cloud*1 falhou devido à ação do script gerador de interrupção (Fig. 34) ficando fora do ar por 07:02 minutos (detalhe da Fig. 32(a)), aumentando o número de acessos ao servidor de aplicação da *Cloud*2.

Fig. 34 - Log do *Script* gerador de Interrupção - *Cloud1.*

A despeito das interrupções dos servidores, observa-se que jamais o acesso do usuário ao conteúdo ficou indisponível. Isso ocorreu devido ao uso da distribuição de acessos por DNS aos servidores *Proxies*, que encaminharam as solicitações aos servidores de aplicação disponíveis. Assim, qualquer que seja a origem das falhas em componentes isolados, ou mesmo da *Cloud* como um todo, o acesso ao conteúdo por parte do usuário continua de forma transparente. Desta forma, obteve-se DT = 100%, que é maior do que $D_{T\min}$ = 99,9971%, indicando que a estratégia de projeto mostrou-se bem sucedida neste experimento.

Em relação à DT = 100% observada no experimento realizado com espaço amostral de 24 horas, vê-se um aumento de 0,51% em relação à disponibilidade da melhor *Cloud* individual (*Cloud1* – 99,49%). Assim, para o usuário do sistema, tem-se a diminuição do tempo de inatividade de 7,2 minutos da *Cloud1* e 8,2 minutos da *Cloud2* para 0 (zero) minutos em ambiente *Multi-Cloud*.

Comparando-se $D_{T\min}$ com a melhor *Cloud* individual, nota-se uma melhora de 0,5% no índice de disponibilidade. Neste cenário, pode-se calcular o tempo máximo de inatividade do sistema que poderia ser experimentado pelo usuário. Assim, das 24 horas de experimentação da solução, a indisponibilidade seria da ordem de 3 segundos.

Comparativamente para o período de um ano, mantendo-se os mesmos índices, ter-se-ia a diminuição do tempo de inatividade de 44,68 horas apresentado pela *Cloud1* e 49,93 horas apresentado pela *Cloud2*, para um tempo máximo de indisponibilidade (de acordo com $D_{T\min}$) de 15,24 minutos no ambiente *Multi-Cloud* combinada com a estratégia de DNS e *proxy* reverso, conforme dados apresentados na tabela 4.

Tabela 4 - Disponibilidade do ambiente de testes – *Clouds* Individuais x *Multi-Cloud*.

Disponibilidade (%)	Indisponibilidade Anual		Indisponibilidade Mensal	
Cloud 1: 99,49%	44,68	horas	3,672	Horas
Cloud 2: 99,43%	49,93	horas	4,104	Horas
Multi-Cloud: 99,9971%	15,24	Min.	1,253	Min.

8 ANÁLISE DE FATORES DE CUSTO ASSOCIADOS A DIFERENTES COMBINAÇÕES DE CLOUD PARA PROVIMENTO DE IAAS

Com base na solução apresentada neste trabalho, busca-se não somente apresentar uma abordagem de alta-disponibilidade tecnicamente viável, mas também factível do ponto de vista financeiro, uma vez que, de acordo com [5] atualmente, as maiores limitações costumam ser são critérios financeiros, e não técnicos. Desta forma, apresenta-se a seguir, uma análise baseada em diferentes combinações de provedores de *Cloud Computing*, baseando-se no percentual de disponibilidade e custo apresentados.

Conforme mostrado no capítulo 6, é possível por meio do modelo proposto, utilizar diferentes serviços de *Cloud Computing* para aumentar a disponibilidade de um sistema a níveis conhecidos na literatura como alta-disponibilidade (99,999%) utilizando-se ambientes *Multi-Cloud* baseados em *Clouds* com disponibilidades individuais menores, como por exemplo: 99,95%; 99,90%; entre outros. Na revisão da literatura realizada, não foram encontrados provedores de *Cloud Computing* que, de forma individual, ofereçam em contrato disponibilidades da ordem de 99,999%.

Como cenário para comparação, foram utilizados dados de diferentes provedores, que apresentam o valor mensal a ser pago por um cliente que eventualmente adquira o serviço, e também o percentual de disponibilidade mensal mínimo oferecido em contrato. Existem ainda provedores que oferecem descontos no valor da fatura nos casos em que dentro do período mensal, os serviços não atinjam o percentual mínimo de disponibilidade acordado em contrato [67]. Estes dados foram utilizados para calcular os valores dos percentuais de disponibilidade conforme disposto na tabela 5.

Tabela 5: Disponibilidade x Custo [67].

Disponibilidade (%)	**Valor (R$)**
99,95	319,00
99,90	110,00
99,00	101,20
95,00	93,50
90,00	82,50
89,90	66,00

Destaca-se que os valores apresentados na tabela 5 foram obtidos no ano de 2016 com o valor do dólar cotado a R$ 3,38. Alterações nos valores podem ser encontradas em detrimento da variação cambial e por reajustes da parte dos provedores.

Inicialmente, para obter o percentual de disponibilidade classificado como “alta-disponibilidade” [68], ou seja, 99,999% podemos utilizar dois provedores que ofereçam disponibilidade mínima de 99,95%, conforme apresentado na Fig.35.

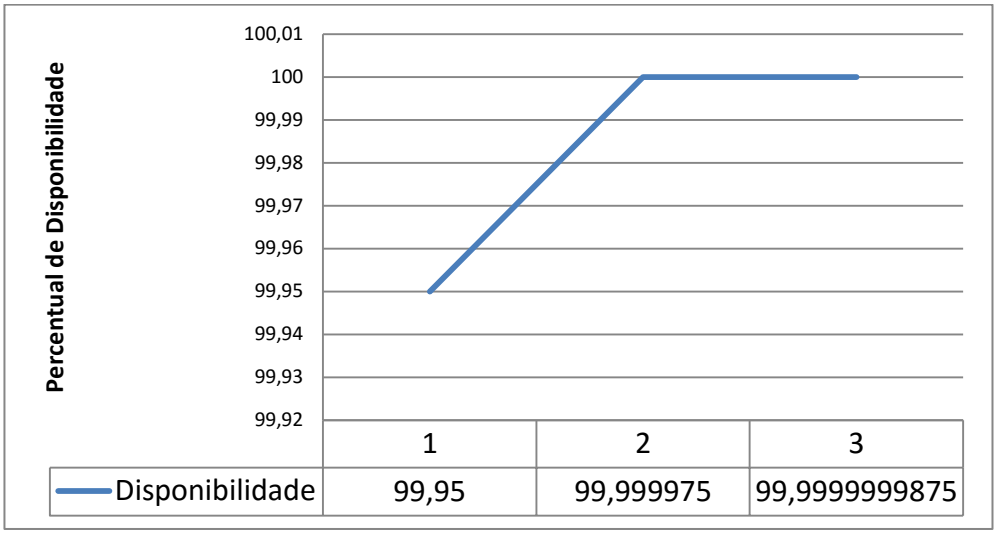


Fig. 35 – Variação da Disponibilidade por número de *Clouds* de 99,95%.

Conforme visualizado, o percentual de 99,999% pode ser atingido se utilizados dois provedores de IaaS com disponibilidade 99,95%.

O valor médio mensal por máquina virtual praticado por provedores com este índice de disponibilidade é de R$ 319,00 [67]. Transportado este cenário ao nosso ambiente de experimentação, no qual cada *Cloud* utilizou três servidores virtuais: Servidor de DNS, Servidor *Proxy* e Servidor de aplicação, teríamos o custo total de R$ 1914,00 mensais para seis servidores virtuais divididos em duas *Clouds.*

Utilizando o mesmo ambiente, porém, em provedores de IaaS que ofereçam percentuais mínimos de disponibilidade da ordem de 99,90%, obtém-se o percentual de alta-disponibilidade 99,999% utilizando dois provedores, conforme Fig. 36.

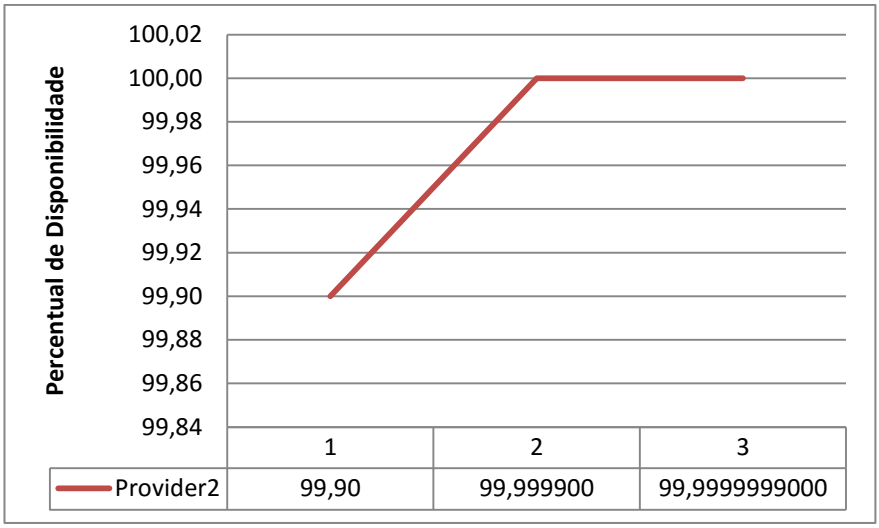


Fig. 36 – Variação da Disponibilidade por número de *Clouds* de 99,90%.

Neste cenário, o custo mensal seria de R$ 660,00 para manter-se a mesma estrutura proposta no ambiente de experimentação, com 6 servidores (3 em cada *cloud*).

Em relação ao percentual de disponibilidade, nota-se que ao se utilizar um sistema *Multi-Cloud* com dois provedores que oferecem disponibilidades de 99,95%, obtém-se 99,999975%, e com dois provedores com disponibilidade de 99,90%, obtém-se 99,999900%. Na tabela 6 são exibidas as diferenças no tempo de parada mensal e anual em cada percentual de disponibilidade calculado nesta simulação de ambiente *Multi-Cloud*:

Tabela 6: Comparativo % de disponibilidade da simulação.

Disponibilidade (%)	Indisponibilidade Anual (s)	Indisponibilidade Mensal (s)
Multi-Cloud 1: 99,999975	7,88	0,648
Multi-Cloud 2: 99,999900	31,54	2,592

Ambos os valores de disponibilidade atingidos remetem ao conceito de alta-disponibilidade por atingirem o mínimo de 99,999%, porém, nota-se claramente a diferença no quesito custo, conforme vê-se na Fig.37.

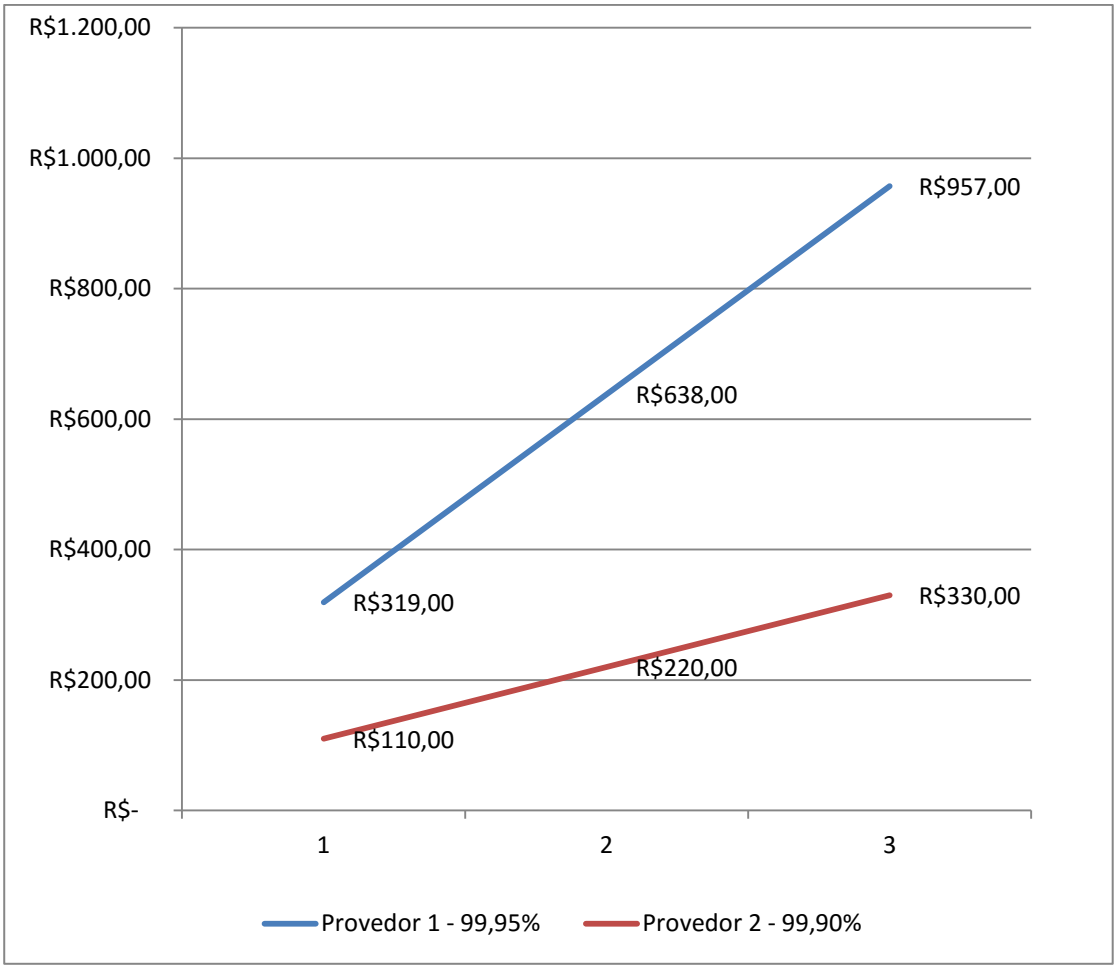


Fig. 37 – Comparativo: Custo por Provedor *Multi-Cloud*.

Desta forma, buscando-se o encontrar a melhor relação custo/benefício, calcularam-se os valores em diferentes combinações de *Clouds* para DT_{min} = 99,999% conforme apresentado na tabela 7.

Tabela 7: Combinações de Diferentes D para DTmin de 99,999%

Base	Variável (%)	DT (2 Clouds)	Valor (R$)
0,9995	0,9990	99,99995	429,00
	0,9900	99,9995	420,20
	0,9500	99,9975	412,50
	0,9000	99,995	401,50
	0,8990	99,99495	385,00

Base	Variável (%)	DT (2 Clouds)	Valor (R$)
0,9990	0,9995	99,99995	429,00
	0,9900	**99,999**	**211,20**
	0,9500	99,995	203,50
	0,9000	99,99	192,50
	0,8990	99,9899	176,00

Base	Variável (%)	DT (2 Clouds)	Valor (R$)
0,9900	0,9995	99,9995	420,20
	0,9990	99,999	211,20
	0,9500	99,95	194,70
	0,9000	99,9	183,70
	0,8990	99,899	167,20

Base	Variável (%)	DT (2 Clouds)	DT (3 Clouds)	Valor (R$)
0,9500	0,9995	99,9975	99,99999875	731,50
	0,9990	99,995	99,999995	313,50
	0,9900	99,95	99,9995	295,90
	0,9000	99,5	99,95	258,50
	0,8990	99,495	99,948995	225,50

Base	Variável (%)	DT (2 Clouds)	DT (3 Clouds)	Valor (R$)
0,9000	0,9995	99,995	99,9999975	720,50
	0,9990	99,99	99,99999	302,50
	0,9900	99,9	99,999	284,90
	0,9500	99,5	99,975	269,50
	0,8990	98,99	99,89799	214,50

Base	Variável (%)	DT (2 Clouds)	DT (3 Clouds)	Valor (R$)
0,8990	0,9995	99,99495	99,99999748	704,00
	0,9990	99,9899	99,9999899	286,00
	0,9900	99,899	99,99899	268,40
	0,9500	99,495	99,97475	253,00
	0,9000	98,99	99,899	231,00

Utilizando-se como base uma *Cloud* com disponibilidade individual de 99,95%, 99,90% ou 99,00%, obtém-se a disponibilidade de 99,999% utilizando-se a base e mais uma *Cloud*. Para as *Clouds* com disponibilidade individual de 95,00%, 90,00%

e 89,90%, atinge-se a disponibilidade de 99,999% utilizando-se a base mais duas *Clouds*.

Dentre as combinações exibidas na Fig. 38, o melhor cenário entre disponibilidade e custo, consiste na utilização de duas *Clouds,* em destaque na tabela 7, sendo uma com 99,90% ao custo de R$ 110,00 e outra com 99,00% ao custo de R$ 101,20. Assim, configurando-se um ambiente *Multi-Cloud*, atinge-se a disponibilidade desejada de 99,999% ao custo mensal de R$ 211,20. Transportando-se os servidores utilizados em nosso ambiente de teste para este cenário, o custo mensal da solução seria de R$ 633,60. Melhor valor possível para a disponibilidade desejada de 99,999%

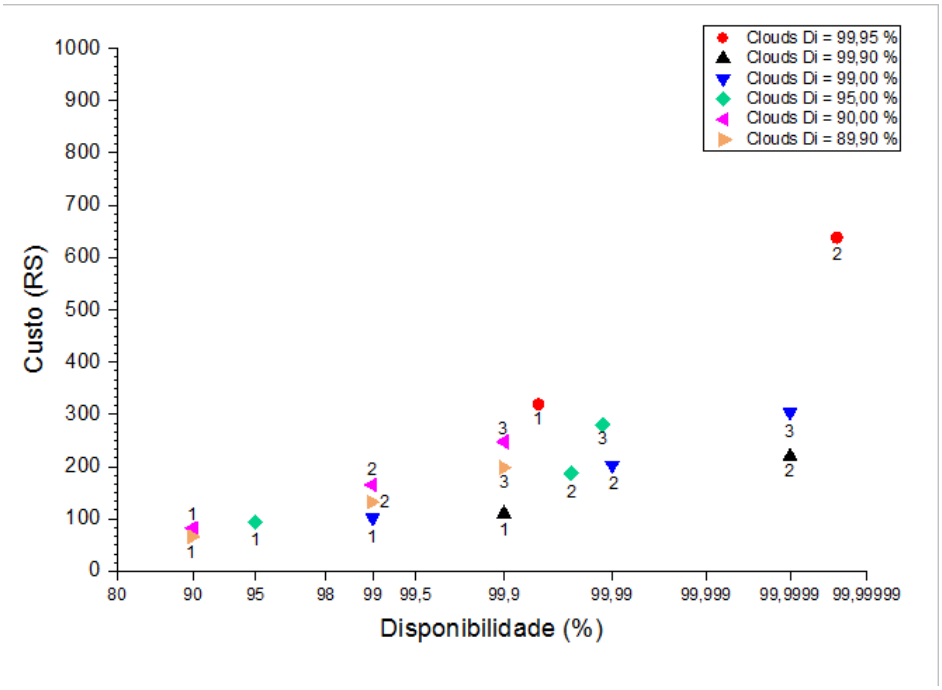


Fig. 38 – Combinações de *Clouds* x Custo.

Destaca-se nesta abordagem, que o ambiente *Multi-Cloud* com disponibilidade DT_{min} = 99,999% formado por duas *Clouds* de 99,90% e 99,90%, apresenta um custo mensal 34% menor que um ambiente *Single-Cloud* de 99,95%. Assim mostra-se a solução viável não somente em quesitos técnicos, como também em aspecto financeiro.

9 CONCLUSÃO

9.1 - Comentários Gerais

Neste trabalho, foi apresentada e testada uma estratégia para provimento de alta-disponibilidade para serviços em *Cloud*, utilizando-se de ambientes *Multi-Cloud,* conjugada com a distribuição de acessos por DNS e *Proxy* reverso. A motivação deste estudo se deu pelo fato de que empresas ainda são relutantes na aderência a soluções *de Cloud Computing* em seus ambientes de produção, devido a indisponibilidades históricas [31]-[9] que fizeram com que clientes de computação em nuvem ficassem fora do ar, e até mesmo perdessem dados [8]. A solução proposta permite que empresas utilizem serviços de diferentes *Clouds* simultaneamente, de forma que não fiquem atreladas a um único provedor de serviços, aumento assim a disponibilidade de suas aplicações.

Propôs-se também neste trabalho, uma expressão para calculo de disponibilidade mínima em ambientes *Multi-Cloud*, baseada na expressão de cálculo de disponibilidade conhecida na literatura [15].

Nossos testes foram realizados mediante simulações com duas *Clouds* emuladas, com disponibilidades de 99,49% e 99,43%, que permitiram alcançar uma disponibilidade total mínima de 99,9971.

O aumento de 0,5% no índice de disponibilidade, inserido no ambiente de testes pelo sistema proposto, corresponde a uma diminuição da ordem de 120 vezes do tempo indisponibilidade do sistema *Multi-Cloud* na janela de um ano.

No que se refere à analise financeira baseada em valores de serviços fornecidos por provedores de *Cloud*, mostrou-se que a solução de *Multi-Cloud* não só é economicamente viável, mas também pode ser um fator utilizado na redução de custos, uma vez que a combinação de *Clouds* com menor disponibilidade mostrou-se 34% mais barata do que uma única *Cloud* que oferece disponibilidade típica de 99,95%.

Em [21], mostrou-se uma arquitetura de PaaS em *Clouds* federadas para a qual foram realizadas avaliações utilizando-se de três cenários (SCA – *Service Component Architecture*) propostos. Os resultados foram organizados em uma tabela, na qual são exibidos o tempo médio de execução e a média de sobrecarga inserida pela execução de cada SCA. Observa-se que, em seu melhor resultado, a sobrecarga inserida pelo sistema foi de 17,37%, o que representou um acréscimo de pouco mais de 21 s ao tempo de execução.

Comparativamente, na proposta de *Multi-Cloud* (não federadas) utilizando-se distribuição por DNS e *proxy* reverso de forma conjugada, foram realizados testes para avaliação de desempenho por meio do Apache Benchmark. Os resultados obtidos determinaram que a variação no tempo de resposta se dá pela latência no acesso de cada *Cloud,* sendo que os componentes utilizados para provimento da solução não promoveram sobrecarga no ambiente. Na verdade, em alguns casos evidenciou-se melhora no tempo de resposta, uma vez que o tempo de acesso à *Cloud* com latência de 24ms mostrou-se 18% menor em comparação ao teste realizado sem a utilização de *proxy* reverso. No caso da *Cloud* com maior latência (140ms), não foi evidenciada alteração no tempo de resposta na execução dos testes.

9.2 - Trabalhos Futuros

Sugere-se, para trabalho futuros, a implementação do cenário de Infraestrutura como Serviço proposto nesta dissertação, em um ambiente *Multi-Cloud*, por meio da utilização de *Clouds* Públicas ou Hibridas. Também deve ser avaliada a testabilidade, com diferentes configurações de critérios, para a distribuição de acesso por parte dos servidores de *proxy* reverso. Por fim, sugere-se também estudar em trabalhos futuros diferentes combinações de configuração de *hardware* virtual para os diferentes tipos de servidores utilizados no estudo, no intuito de se obter custo/benefício para o ambiente em nível de *hardware* virtual oferecidos por provedores de *Cloud Computing*.

10 REFERÊNCIAS

[1] N. Sfondrini, G. Motta, and L. You, “Service Level Agreement (SLA) in Public Cloud Environments : A Survey on the current enterprises adoption.,” *2015 5th Int. Conf. Inf. Sci. Technol.*, 2015.

[2] “Cisco,” 2016. [Online]. Available: http://www.cisco.com/. [Accessed: 10-Sep-2016].

[3] Cisco, “Brasil supera média mundial na adoção de cloud computing e virtualização,” 2010. [Online]. Available: http://www.marketwired.com/press-release/brasil-supera-media-mundial-na-adocao-de-cloud-computing-e-virtualizacao-nasdaq-webx-1366967.htm. [Accessed: 08-Sep-2016].

[4] CompTIA, “Trends in Cloud Computing,” 2016. [Online]. Available: https://www.comptia.org/resources/trends-in-cloud-computing-2016. [Accessed: 14-Nov-2016].

[5] J. Steddum, “A Brief History of Cloud Computing,” 2013. [Online]. Available: http://blog.softlayer.com/2013/virtual-magic-the-cloud. [Accessed: 11-Oct-2016].

[6] N. Sinha and L. Khreisat, “Cloud Computing Security , Data , And Performance Issues,” *Wirel. Opt. Commun. Conf. -WOCC 2014*, pp. 1–6, 2014.

[7] P. Mell and T. Grance, “The NIST definition of cloud computing,” *NIST Spec. Publ.*, vol. 145, p. 7, 2011.

[8] “Falha em serviço da Amazon faz clientes perderem dados permanentemente,” 2011. [Online]. Available: http://tecnologia.uol.com.br/ultimas-noticias/redacao/2011/04/29/falha-em-servico-da-amazon-faz-clientes-perderem-dados-permanentemente.jhtm. [Accessed: 05-Sep-2016].

[9] H. Blodget, “Amazon’s Cloud Crash Disaster Permanently Destroyed Many Customers' Data,” 2011. [Online]. Available: http://www.businessinsider.com/amazon-lost-data-2011-4. [Accessed: 05-Sep-2016].

[10] M. J. Earl, “The Risk of Outsourcing IT,” *Sloan Manage. Rev.*, vol. 37, no. 3, pp. 26–32, 1996.

[11] P. S. W. J. Pisoni, *Clusters for High Availability : A Primer of HP Solutions 2nd*, Edition: 2. .

[12] X. Yuan, Y. Li, T. Jia, T. Liu, and Z. Wu, “An Analysis on Availability Commitment and Penalty in Cloud SLA,” *Proc. - Int. Comput. Softw. Appl. Conf.*, vol. 2, no. 61232005, pp. 914–919, 2015.

[13] H. A. Guide, “Oracle ® Application Server,” vol. 3, no. February, 2006.

[14] Q. Zhang, S. Li, Z. Li, Y. Xing, Z. Yang, and Y. Dai, “CHARM: A Cost-Efficient Multi-Cloud Data Hosting Scheme with High Availability,” *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 372–386, 2015.

[15] K. Benz and T. Bohnert, “Dependability modeling framework: A test procedure for high availability in cloud operating systems,” *IEEE Veh. Technol. Conf.*, 2013.

[16] B. Ciciani, F. Quaglia, P. Romano, and D. Dias, “Analysis of Design Alternatives for Reverse Proxy Cache Providers,” *11TH IEEE/ACM Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, 2003.

[17] D. Petcu, “Multi-Cloud : Expectations and Current Approaches,” *2013 Int. Work. Multi-cloud Appl. Fed. clouds*, pp. 1–6, 2013.

[18] F. Moscato, B. Di Martino, and V. Munteanu, “An Analysis of mOSAIC ontology for Cloud Resources annotation,” *Fed. Conf. Comput. Sci. Inf. Syst.*, pp. 973–980, 2011.

[19] A. J. Ferrer, F. Hern, J. Tordsson, E. Elmroth, K. Djemame, W. Ziegler, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forg, T. Sharif, and C. Sheridan, “OPTIMIS : a Holistic Approach to Cloud Service Provisioning,” 2012.

[20] N. Grozev and R. Buyya, “Inter-Cloud architectures and application brokering : taxonomy and survey,” *Wiley Online Libr.*, no. December 2012, pp. 369–390, 2014.

[21] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier, “2012 IEEE Fifth International Conference on Cloud Computing A Federated Multi-Cloud PaaS Infrastructure,” 2012.

[22] Openstack.org, “Openstack,” 2016. [Online]. Available: www.openstack.org. [Accessed: 26-May-2016].

[23] K. Ranjithprabhu and D. Sasirega, “Eliminating Single Point of Failure and Data Loss in Cloud Computing,” vol. 3, no. 4, pp. 335–337, 2014.

[24] R. Y. Ameen, “Survey of Server Virtualization,” *Int. J. Comput. Sci. Inf. Secur. Vol.11, No. 3, 2013*, 2013.

[25] T. Jones, “Anatomia de um Hypervisor Linux.” [Online]. Available: https://www.ibm.com/developerworks/br/library/l-hypervisor/. [Accessed: 11-Oct-2016].

[26] Vmware Paper, “Virtualization Overview 1,” *VMWARE*, 2006. [Online]. Available: https://www.vmware.com/pdf/virtualization.pdf. [Accessed: 15-Jul-2016].

[27] R. Talaber, V. M. Ware, L. Lamers, and V. M. Ware, “USING VIRTUALIZATION TO IMPROVE DATA CENTER EFFICIENCY,” *GREEN GRID*, pp. 1–22, 2009.

[28] G. Inc, “Google Docs,” 2016. [Online]. Available: https://www.google.com/docs/about/. [Accessed: 10-Aug-2016].

[29] Cognizant, “Optimizing XaaS,” *Cognizant.*, 2013. [Online]. Available: https://www.cognizant.com/industries-resources/technology/Optimizing-XaaS.pdf. [Accessed: 08-Sep-2016].

[30] Vmware Paper, “MULTI-CLOUD ENVIRONMENTS ARE BECOMING THE NEW NORMAL FOR IT,” *Dimensional Research*, 2016. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/vrealize/vmware-managing-the-multi-cloud-environment.pdf. [Accessed: 01-Jun-2016].

[31] K. Clay, “Amazon.com Goes Down, Loses $66,240 Per Minute,” *Forbes*, 2013. [Online]. Available: http://www.forbes.com/sites/kellyclay/2013/08/19/amazon-com-goes-down-loses-66240-per-minute/#678fb1783c2a. [Accessed: 05-Oct-2016].

[32] P. Authors, V. Beecher, V. Schupmann, J. Stern, C. A. Abbas, L. Ashdown, A. Babb, T. Bednar, P. Belknap, J. Blowney, L. Carpenter, I. Chan, T. Chien, D. Cooksey, T. Das, M. Dilman, R. Dutcher, R. Exley, C. Foch, A. Kini, F. Kobylanski, B. Llewellyn, B. Lundhild, R. Mau, P. Mcelroy, J. Meeks, M. Michalewicz, V. Moore, D. Norris, M. Nowak, A. Ray, M. Scardina, V. Schupmann, J. Shi, M. T. Smith, V. Srihari, H. Sun, L. To, D. Utzig, J. Viscusi, T. Wang, and S. Yamaguchi, “Oracle ® Database - High Availability Overview,” vol. 2, no. July, 2013.

[33] *SLA Management Handbook*, vol. 4, no. October. 2004.

[34] C. Pham, P. Cao, Z. Kalbarczyk, and R. K. Iyer, “Toward a High Availability Cloud : Techniques and Challenges,” *IEEE*, 2012.

[35] W. R. David Roberto, Rafael Dantas, “Tutorial DNS Objetivos,” *3º PTT Fórum*, pp. 1–135, 2009.

[36] T. Brisco, “Request for Comments: 1794 - DNS Support for Load Balancing,” 1995. [Online]. Available: https://tools.ietf.org/html/rfc1794. [Accessed: 01-May-2016].

[37] M. Endler, R. C. A. Rocha, M. Endler, H. Rubinsztejn, R. C. A. Rocha, and V. Sacramento, “Proxy-based Adaptation for Mobile Computing Proxy-based Adaptation for Mobile Computing *,” *Monogr. em Ciência da Comput. No. 24/05*, no. June, 2005.

[38] P. Security, “Forward and Reverse Proxy,” *Siemens*, 2003. [Online]. Available: ftp://www.3gpp.org/tsg_sa/WG3_Security/TSGS3_31_Munich/Docs/PDF/S3-030744.pdf. [Accessed: 23-Jul-2016].

[39] M. A. Alzain, E. Pardede, B. Soh, and J. A. Thom, “Cloud Computing Security : From Single to Multi-Clouds,” *2012 45th Hawaii Int. Conf. Syst. Sci.*, 2012.

[40] L. M. Pham and T. M. Pham, “Autonomic fine-grained migration and replication of component-based applications across multi-clouds,” *Inf. Comput. Sci. (NICS), 2015 2nd Natl. Found. Sci. Technol. Dev. Conf.*, pp. 5–10, 2015.

[41] A. Abouzamazem and P. Ezhilchelvan, “Efficient inter-cloud replication for high-availability services,” *Proc. IEEE Int. Conf. Cloud Eng. IC2E 2013*, pp. 132–139, 2013.

[42] W. Wu, K. Wang, R. Jan, and C. Huang, “A Fast Failure Detection and Failover Scheme for SIP High Availability Networks 1,” *13th IEEE Int. Symp. Pacific Rim Dependable Comput.*, pp. 187–190, 2007.

[43] I. S. Consortium, “BIND 9 Administrator Reference Manual,” 2013.

[44] A. Khare, Y. Huang, H. Doan, and M. S. Kanwal, “A Fresh Graduate’s Guide to Software Development Tools and Technologies.”

[45] S. S. Nathan, S. S. Mohan, A. R. Harudas, and K. Nisar, “BERKELEY I NTERNET NAME DOMAIN (BIND),” vol. 1, no. 1, pp. 1–10, 2012.

[46] T. Winograd, M. Tracy, and W. Jansen, “Guidelines on Securing Public Web Servers Recommendations of the National Institute of Standards and Technology.”

[47] “Apache2,” 2016. [Online]. Available: httpd.apache.org. [Accessed: 01-Jan-

2016].

[48] “Nginx,” 2016. [Online]. Available: www.nginx.com. [Accessed: 09-Mar-2016].

[49] “lighttpd.” [Online]. Available: https://www.lighttpd.net/. [Accessed: 29-Jun-2016].

[50] “cherokee web server.” [Online]. Available: http://cherokee-project.com/. [Accessed: 30-Jul-2016].

[51] “Microsoft IIS.” [Online]. Available: https://www.iis.net/. [Accessed: 30-Jul-2016].

[52] “IBM http server.” [Online]. Available: http://www-03.ibm.com/software/products/pt/http-servers. [Accessed: 30-Jul-2016].

[53] Netcraft, “March 2015 Web Server Survey.” [Online]. Available: https://news.netcraft.com/archives/2015/03/19/march-2015-web-server-survey.html.

[54] R. Bowen and C. Mcgregor, “Introduction to the Apache Web Server,” 2005.

[55] T. Nginx, “What is NGINX?,” 2012. [Online]. Available: https://www.nginx.com/resources/glossary/nginx/. [Accessed: 12-May-2016].

[56] Oracle, “VirtuaBox,” 2015. [Online]. Available: www.virtualbox.org. [Accessed: 01-Mar-2015].

[57] A. S. Tanenbaum, *Sistemas Operacionais Modernos*, 2ª ed. 2005.

[58] “Ubuntu,” 2016. [Online]. Available: http://www.ubuntu.com/desktop. [Accessed: 15-May-2016].

[59] “Munin Monitoring,” 2016. [Online]. Available: http://munin-monitoring.org/. [Accessed: 10-Mar-2016].

[60] T. Oetiker, “About RRDtool.” [Online]. Available: http://oss.oetiker.ch/rrdtool/. [Accessed: 02-Aug-2016].

[61] Linux Foundation, “NetEm,” 2009. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem. [Accessed: 01-May-2016].

[62] A. S. Tanenbaum, *Redes de computadores*, 4th Editio. Amsterdam, Holand.

[63] Apache.org, “Apache Benchmark,” 2015. [Online]. Available: httpd.apache.org/docs/2.4/programs/ab.html. [Accessed: 01-Mar-2015].

[64] Y. S. Hong, J. H. No, and S. Y. Kim, “DNS-based load balancing in distributed web-server systems,” *Proc. - Fourth IEEE Work. Softw. Technol. Futur. Embed. Ubiquitous Syst., SEUS 2006 andthe Second Int. Work. Collab. Comput. Integr., Assur. WCCIA 2006*, vol. 2006, pp. 251–254, 2006.

[65] “Cloudping,” 2015. [Online]. Available: http://www.cloudping.info/. [Accessed: 10-Dec-2015].

[66] Michael Kerrisk, “Man-pages RANDOM,” 2015. [Online]. Available: http://man7.org/linux/man-pages/man4/random.4.html. [Accessed: 26-May-2016].

[67] Embratel, “Embratel - Cloud Server,” 2016. [Online]. Available: http://portal.embratel.com.br/cloud/cloud-server/. [Accessed: 31-Aug-2016].

[68] M. A. Sharkh, A. Shami, P. Ohlen, A. Ouda, and A. Kanso, “Simulating High Availability Scenarios in Cloud Data Centers: A Closer Look,” *2015 IEEE 7th Int. Conf. Cloud Comput. Technol. Sci.*, pp. 617–622, 2015.

11 APÊNDICES

Apêndice 1 – Script “gera_interrupcao.sh”

```
#!/bin/bash
# Autor: Luis Paulo
# Script gera interrupção na execução do apache2 de forma aleatória, por um tempo
        aleatório.
# A saida dos comandos fica registrada no arquivo "randon.log".
echo "_____" >> randon.log
echo "Inicio da execucao do script: " $(date) >> randon.log
sleep $((RANDOM%86400+3600))
echo "Parando o serviÃ§o do apache2 em: " $(date) >> randon.log
service apache2 stop
echo " Apache2 parado em: " $(date) >> randon.log
echo "Status: " >> randon.log
service apache2 status >> randon.log
sleep $((RANDON%600+330))
echo "Iniciando o serviÃ§o do apache2 em: " $(date) >> randon.log
service apache2 start
echo "Apach2 startado em: " $(date) >> randon.log
echo "Status: " >> randon.log
service apache2 status >> randon.log
echo "_____" >> randon.log
```

Apêndice 2 – Arquivo de configuração – Nginx *Proxy* Reverso 1

```
upstream backend  {
        # Para acesso ao server1 via LAN
        server 192.168.38.100;
        # Para acesso ao server2 via WAN
        server 200.210.15.100;
        # O server2 nao pode ser acessado via LAN pois esta em outra Cloud.
}

server {
        listen 80 default_server;
        listen [::]:80 default_server ipv6only=on;

        root /usr/share/nginx/html;
        index index.html index.htm;

        # Make site accessible from http://proxy1/
        server_name proxy1;

location / {
              try_files $uri $uri/ =404;
              access_log off;
            proxy_pass  http://backend;
}
}
```

Apêndice 3 – Arquivo de configuração – Nginx *Proxy* Reverso 2

```
upstream backend  {
        # Para acesso ao server2 via LAN
        server 192.168.39.200;
        # Para acesso ao server1 via WAN
        server 187.10.10.100;
        # O server1 nao pode ser balanceado via LAN pois esta em outra cloud
}

server {
        listen 80 default_server;
        listen [::]:80 default_server ipv6only=on;

        root /usr/share/nginx/html;
        index index.html index.htm;

        # Make site accessible from http://proxy2/
        server_name proxy2;

location / {
            try_files $uri $uri/ =404;
            access_log off;
            proxy_pass  http://backend;
}
}
```

Apêndice 4 – Arquivo de configuração – Bind9

```
$ttl 38400
ha.lab.local.      IN      SOA     Bind9. lpgonline.gmail.com. (
                   1454930522
                   10800
                   3600
                   604800
                   38400 )
ha.lab.local.      IN      NS      Bind9.
www.ha.lab.local.  IN      A       192.0.0.10
www.ha.lab.local.  IN      A       172.0.0.20
```

luispaulo@Bind9:~$ nslookup www.ha.lab.local 127.0.0.1

```
Server:         127.0.0.1
Address:        127.0.0.1#53

Name:           www.ha.lab.local
Address: 172.0.0.20
Name:           www.ha.lab.local
Address: 192.0.0.10
```

luispaulo@Bind9:~$ nslookup www.ha.lab.local 127.0.0.1

```
Server:         127.0.0.1
Address:        127.0.0.1#53

Name:           www.ha.lab.local
Address: 192.0.0.10
Name:           www.ha.lab.local
Address: 172.0.0.20
```