**UNIVERSIDADE FEDERAL DO MARANHÃO**
**CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA**
**CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

# ANÁLISE COMPARATIVA DE ABORDAGENS DE SISTEMAS TOLERANTES A FALHAS EM UMA NUVEM COMPUTACIONAL

**STEVE ATAKY TSHAM MPINDA**

São Luís
2013

**STEVE ATAKY TSHAM MPINDA**

# ANÁLISE COMPARATIVA DE ABORDAGENS DE SISTEMAS TOLERANTES A FALHAS EM UMA NUVEM COMPUTACIONAL

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

**Orientador: Zair Abdelouahab, Ph. D.**

São Luís
2013

Mpinda, Steve Ataky Tsham.

Análise comparativa de abordagens de sistemas tolerantes a falhas em uma nuvem computacional / Steve Ataky Tsham Mpinda. – São Luís, 2013.
91 f.

Orientador: Prof. PhD. Zair Abdelouahab
Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2013.

1. Computação em Nuvem. 2. Sistemas tolerantes a falhas. 3. Análise comparativa. I. Título.

CDU 004.738.2

**STEVE ATAKY TSHAM MPINDA**

# ANÁLISE COMPARATIVA DE ABORDAGENS DE SISTEMAS TOLERANTES A FALHAS EM UMA NUVEM COMPUTACIONAL

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em: 10 de Dezembro de 2013.

Zair Abdelouahab
Examinador

Aristófanes Corrêa Silva
Examinador

Carlos Eduardo Portela serra de Castro
Examinador

*Aos meus Pais, Guillaume Kisseng Ataky e Annie Kafuti, pelo amor que têm por mim e pela minha família, ainda pelo cuidado, carinho e ensino dos valores tanto morais como cristãos os quais caracterizam o que sou hoje. Je vous aime, chers parents.*

*Aos meus irmãos "Les ATAKYs", Ange Amy Ataky, Geoffrey Ataky Kisseng, Heritier Kiekubu Ataky, Jered Ataky, Patricia Matangila Ataky e Ezéchiel Ataky Kafuti por terem sido pessoas, amigos e companhia que qualquer pessoa gostaria de ter. Sem vocês o mundo seria um deserto aos meus olhos. Je vous aime tous de tout coeur.*

*Homenagem aos Ministérios das Relações exteriores (MRE) e da Educação (MEC) brasileiros pelo acolhimento e concessão de uma vaga na Universidade Federal do Maranhão (UFMA) no âmbito do Programa Estudante Convênio Graduação (PEC-G).*

# AGRADECIMENTOS

A Deus Altíssimo, Criador dos céus e da terra, por sua fidelidade e bondade que duram para sempre. Ele nunca desistiu de mim, apesar de mim. Sempre guiou os meus passos até os dias atuais da minha vida, renovando minhas forças, e por ter me sustentado e dado vitória apesar das tribulações e oposições.

A honra vai ao meu orientador professor Zair Abdelouahab. Ele me colocou em condições ideais para a realização deste trabalho e tem sido tão generoso em me transmitir a parte de seu grande conhecimento. Estou-lhe grato deveras.

A todos os membros da banca para o tempo que gastaram na avaliação do meu trabalho.

Aos professores Francisco Silva e Gentil Serra, pelas valiosas correções e adaptações feitas a este trabalho. Fico assaz felicitado.

Ao supremo *capela*, Leonardo Melo pelo provimento de diversão enquanto nos deslocávamos de volta às nossas respectivas residências, no ônibus.

Aos companheiros detentores do saber de Computação e eminentes letrados do vernáculo português, Artur Pinheiro, Sasha Nicolas, Mário Henrique, Higo Felipe, Jean Pablo, Wesley Lima, Daniel Souza, Thiago Pereira, Rafael Drummond, Lucas Carácas, Aitan Viegas, Ruberth André , Whesley Dantas, João Vítor, Dhully, Luiz Aurélio, Claudio, Jonathan, Samir Souza , William e as três irmãs Raianne, Patrícia e Rayssa.

Ao Marcus Vinicius e Fiston Sita (*maître Taw*), os quais nas últimas instâncias conseguiram detectar faltas aos meus olhos camufladas.

Ao casal querido Mário e Socorro e o caro Glécio Santos pela providência logística na concretização do meu deslocamento seguro às dependências da UFMA.

Aos insubstituíveis, Patrick Andjasubu, Mike Muya (*charcuterie*), Daniel Charles, Serge Nganga, Serge Makengo, Serge Lewula, Philippe Lukume, Fabrice Mutumbo (*grand baoba*), Elys Cunha, Diego Ngamuna, Patrick Masoka, Patrick Kayembe, Blanchard Paka, Adriano Kilala, Hans Cutrim, Helvécio Pereira, Euziel Lourenço, Michel Cantanhede, Lívia Fernanda, Olavo, Débora Castro e nosso pai Domingos, Junior Bibefu, Romildo Rocha, Everton Pinheiro Pereira, Alessandro sodré, Erickson Vinicius, Amanda Serra, o casal Flávio e Camila Ferrato, quando precisei de amigos, ganhei vocês como irmãos.

Aos colegas do PEC-G da Universidade Federal do Maranhão e de todo o Brasil.

Aos meus orientadores de iniciação cientifica e projetos de pesquisa e desenvolvimento. Carlos de Salles, você foi uma das primeiras pessoas que me

impressionaram tecnicamente na área de computação, especificamente em algoritmos. Denivaldo Lopes, você conseguiu me transmitir a sua serenidade diante de qualquer problema e projeto de pesquisa.

Ao pró-reitor de Pesquisa e Pós-Graduação, professor Fernando carvalho e toda equipe dessa Pró-reitoria (PPPG) pelo apoio moral e por terem sido mais chegados do que parentes.

Ao diretor e colegas do Departamento de Pesquisa (DPQ) da PPPG, Francisco Santos, Claudia, Érika, Nathana Valle, Marcos, Renato e Ramon pela alegria, brincadeiras e todos os bons momentos que passamos e ainda vamos passar juntos.

Aos confrades do LAWS, LESERC e LABSAC, e a grande turma de CP 2013.1. Foi um prazer trabalhar com vocês e espero continuar.

A todos os meus familiares por terem contribuído de alguma forma para a minha vida e por sempre acreditarem em mim desde que eu era pequeno até hoje.

À *dulcinée*, Laise Nayra, que com seu exemplo, apoio e carinho me mostrou como seria importante que eu concluísse esse trabalho.

Aos pastores, irmãos e amigos de Assemblée Chrétienne Chémin du Rocher e PIB São Luis, pelo apoio espiritual e pelas orações à minha vida e família direcionadas.

Ao professor Aristófanes Corrêa, pelos importantíssimos conselhos, direcionamentos e motivações tanto de ponto de vista acadêmico como social. Fico assaz felicitado.

Aos professores Rosário Girardi, Alexandre César, Geraldo Braz, Elivaldo Macedo, Carlos Portela e Anselmo de Paiva pelas motivações acadêmicas.

A todos os meus amigos da graduação pelo companheirismo nos momentos difíceis, pela exultante alegria nos momentos felizes e pelo bom humor de sempre que nos animou todos os dias desde que iniciamos o curso.

Aos meus professores do vernáculo português, José carlos, Marcos, Moises, Renata, Edirnelis (didi) e Karina Gaya. Tudo começou com vocês. Até a minha morte estarão sempre no meu coração.

Universidade Federal do Maranhão representada pelos professores do Departamento de Informática que, com suas qualidades e experiências, contribuíram para a minha formação acadêmica.

Por fim, a todos que direta ou indiretamente contribuíram para a realização deste objetivo.

*“εἰ γὰρ ἡ προθυμία πρόκειται, καθὸ ἐὰν ἔχῃ εὐπρόσδεκτος, οὐ καθὸ οὐκ ἔχει.”*
*Porque, se há prontidão de vontade, é aceitável segundo o que alguém tem, e não segundo o que não tem.*

*2 Coríntios 8:12*

# RESUMO

Nos últimos anos temos visto o desenvolvimento de Computação em Nuvem. O princípio fundamental é para deslocar o gerenciamento de serviços computacionais de empresas nos centros de hospedagens gerenciados por empresas de terceiros. Este deslocamento tem com principal vantagem a redução de custos para a empresa cliente , os meios necessários para o gerenciamento desses serviços são mutualizados entre clientes e gerenciados pela empresa de hospedagem destes serviços. A fiabilidade, disponibilidade na Computação em Nuvem são requisitos essenciais para garantir um funcionamento adequado e contínuo, mesmo na presença de falhas. O mecanismo de tolerância a falhas deve ser desenvolvido para resolver as deficiências que podem afetar as operações normais na Nuvem.

A tolerância a falhas é a capacidade de um sistema computacional de executar sua função, apesar da presença ou ocorrência de falhas, quer seja o dano de hardware, quer falhas de software, ataques maliciosos  ou erros na interação homem-máquina . Em outras palavras, a tolerância a falhas é um modelo de projeto que permite que um usuário gerencie falhas de nível de sistema ou de rede de uma forma graciosa, minimizando a degradação se sua experiência. Na ausência de tolerância a falhas, se um sistema falhar, isto pode causar a perda de dados ou outras alterações indesejadas. Agora, existem muitas técnicas para tolerância a falhas sendo investigadas e implementadas. Como um usuário poderia saber a qual das técnicas é melhor para a natureza do seu trabalho? Neste trabalho, apresenta-se uma visão geral da Computação em Nuvem, a tolerância a falhas e seu lugar dentro da confiabilidade, os modelos existentes, técnicas e abordagens das diferentes classes de tolerância a falhas são apresentados, assim como sua implementação, podendo ajudar usuários ingênuos para suas necessidades no que concerne a escolha de um sistema tolerante a falhas adequado para o seu negócio. As técnicas são ilustradas pela descrição das ferramentas utilizadas para a sua implementação, especificando suas políticas, os frameworks de desenvolvimento, os sistemas, os ambientes e jaez de falhas detectadas. Em seguida, foram apresentados e discutidos alguns elementos de métricas válidas para examinar os desafios da tolerância a falhas, comparando os modelos existentes através de tabelas e gráficos de tendências.

**Palavras-chave**: Computação em Nuvem, tolerância a falhas, técnicas de tolerância a falhas, modelos de sistema tolerante a falhas , análise comparativa.

# ABSTRACT

A few years back have seen the development of cloud computing. The main underlying principle is to deport or externalize the companies' IT services management in hosting centers which are managed by third party companies. The main advantage of the externalization resides on the fact that it allows saving costs for the client company, since the required resources for the management of these services are shared between clients and managed by the company hosting these services. Reliability, availability in the cloud computing are essential requirements to ensure proper and continuous operation even in the presence of faults. The fault tolerance mechanism must be developed to deal with failures that may affect the normal operations in the cloud.

Fault tolerance is the ability of a computer system to perform its function despite the presence or occurrence of faults, whether it is physical damage to the hardware, software faults, malicious attacks, or errors in man-machine interaction. In other words, fault tolerance is a design model that allows a user to manage system-level or network faults in a graceful way while minimizing degradation experience. In the absence of fault tolerance if one system fails, it can cause data loss or other undesirable changes. Actually, there are many techniques for fault tolerance that are being investigated and implemented. The question is how does a user know which technique will be best for his nature of work. In this work, we present an overview of cloud computing, the fault tolerance and its place within the dependability, existing models, techniques and approaches of the various classes of faults tolerance, as well as their implementation, which can help naive users regarding the choice of a suitable fault tolerant system for their business. These techniques are illustrated by the description of the tools used for their implementation by specifying their policies, development frameworks, systems, environment and detected fault. Later, were presented and discussed some valid metric elements for examining challenges to fault tolerance, comparing existing models through the table and trending graphs.



**Keywords**: cloud computing, virtualization, security in Cloud Computing, Cloud Computing administration, fault tolerance, fault tolerance technique, fault types, metrics, models of fault tolerant system, comparative analysis.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE SIGLAS

| | |
|---|---|
| SLA | *Service-Level Agreement* |
| SGBD | Sistema de gerenciamento de Banco de dados |
| IT | *Information Tachnology* |
| CC | *Cloud Computing* |
| IaaS | *Infrastructure as a Service* |
| SaaS | *Software as a Service* |
| PaaS | *Plateform as a Service* |
| API | *Application Programming Interface* |
| SMTP | *Simple Mail Transfer Protocol* |
| HTML | *Hyper-Text Markup Language* |
| HTTP | *Hypertext Transfer Protocol* |
| VM | *Virtual Machine* |
| OS | *Operational System* |
| VMM | *Virtual Machine Monitor* |
| IP | *Information Protocol* |
| UML | *User Model Linux* |
| SSL | *Secure Sockets Layers* |
| CPU | *Central Processing Unit* |
| TF | *Tolerância a Falhas* |
| TMR | *Triple Modular Redundancy* |
| IBD | *Internal Block Diagram* |
| SNR | State Machine Diagram |
| STR | Stochastic Reward Nets |

# SUMÁRIO

# 1 INTRODUÇÃO

A tecnologia de informação e comunicação está mudando e revolucionando a forma como vivemos e trabalhamos. A Computação em Nuvem ou computação virtual surgiu nos últimos anos como um novo modelo de gerenciamento e utilização dos sistemas de tecnologia da informação. O conceito é de delegar em servidores remotos os tratamentos e armazenamento habitualmente executadas localmente, a fim de acessar como um serviço.

Dependendo das necessidades, os serviços de Computação em Nuvem podem variar do simples fornecimento de máquinas virtuais, serviços IaaS (*Infrastructure as a Service*) aos serviços do tipo aplicações SaaS (*Software as a Service*). Vários modos de implantação destes serviços podem ser usados de acordo com o "Instituto Nacional de Padrões da Tecnologia" dos Estados Unidos (National Institute of Standards and Technology-NIST), entre os quais o modo público e modo privado.

Implantados em Nuvem pública, os serviços de Computação em Nuvem permitem oferecer aos usuários aplicações ou sistemas de computação como um serviço acessível através da Internet. Eles permitem que os usuários superem as tarefas de implantação e administração de sistemas computacionais complexos, realizadas localmente, proporcionando-lhes estes serviços seguindo um método de *pay-per-use*. No entanto, as ofertas de Nuvem pública estão sujeitas a controles tipo *vendor lock-in*[1] que impede que os usuários desfrutem da arquitetura. As questões de segurança e de rede também são fatores que limitam a adoção de serviços públicos de Computação em Nuvem. Além disso, é imprescindível que a Nuvem disponha de mecanismo tolerante a falhas cuja ausência pode causar a perda de dados ou outras alterações indesejadas nas aplicações, caso um sistema venha a falhar. O mecanismo de tolerância a falhas deve ser desenvolvido para resolver as deficiências que podem afetar as operações normais na Nuvem.

As principais vantagens da Computação em Nuvem são a escalabilidade, *pay-per-use*[2] e tolerância a falhas [68]. Desta forma, a tolerância a falhas é uma das questões-chave da Computação em Nuvem. Há muitas técnicas de tolerância a falhas em Computação em Nuvem [11]. A tolerância a falhas se preocupa com todas as técnicas necessárias para permitir que um sistema tolere falhas de software ou de hardware. Estas falhas podem ou não se

[1] Uma situação em que um cliente usando um produto ou serviço não pode facilmente fazer a transição para o produto ou serviço de um concorrente. Isto é geralmente o resultado de tecnologias proprietárias que são incompatíveis com os dos concorrentes.

[2] Um modelo de cobrança e custos de uso sob demanda. Em poucas palavras, significa pagar somente pelo que consumir.

manifestarem durante as operações de sistemas, mas quando o fazem, técnicas tolerantes a falhas de software devem fornecer os mecanismos para prevenir ocorrências de falha de sistema.

Portanto, partindo de sua importância, ir-se-á apresentar a tolerância a falhas e seu lugar dentro da confiabilidade, os modelos existentes, técnicas e abordagens das diferentes classes de tolerância a falhas, assim como sua implementação.

Por último, far-se-á uma análise comparativa das abordagens tolerantes a falhas com base nas métricas propostas no [78].

## 1. 1 Revisão Bibliográfica

Um grande número de técnicas de tolerância a falhas que estão estreitamente integrados com aplicações de software distribuídos durante o tempo de desenvolvimento têm sido propostas (por exemplo, [81], [82], [83]). Em [45], os autores apresentaram um *middleware*[3] de tolerância a falhas que usa a abordagem de replicação o *leader* ou *follower para* tolerar falhas no ambiente de Computação em Nuvem.

Uma interessante linha de pesquisa relevante para este artigo propôs uma abordagem para construir protocolos de tolerância a falhas, compondo microprotocols e combinando os em um sistema usando técnicas hierárquicas [84]. A implementação desta abordagem de protocolos de tolerância a falhas demonstra benefícios em termos de fácil personalização quando comparado com um sistema monolítico. Em [85], os autores utilizaram uma abordagem modular para desenvolver um quadro de tolerância a falhas proativa que pode adaptar um requisito *specific strategy*, e o demonstrou como sendo uma boa plataforma para o estudo de várias políticas de tolerância a falhas proativa.

Em [51] e [85], os autores apresentaram um mecanismo para sincronizar continuamente o estado da memória de um nó para os nós de *backup* usando o ponto de verificação. Quando a falha nó primário acontece, o nó de *backup* retoma o tempo de execução imediatamente, fornecendo aparência de não haver interrupção para o usuário final. Neste trabalho, faz-se uso da camada de virtualização para introduzir de forma transparente a tolerância a falhas nas aplicações implantadas.

Uma série de *framework*[4] de tolerância a falhas que permitem as aplicações

[3] Um programa de computação que faz mediação entre outros softwares, podendo mover informações entre programas, ocultar as diferenças de protocolos de comunicação, plataformas e dependências do Sistema Operacional.

[4] Em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

ganharem intrinsecamente a resistência contra falhas têm sido propostas. Em [17], os autores usam a tecnologia de replicação ativa para serviços web, e em [22] os autores propõem uma tecnologia para ganhar a tolerância a falhas bizantinas usando tecnologia de virtualização. Técnicas para a construção de aplicações tolerantes a falhas e eficientes para o EC2 da Amazon são fornecidos em [7]. Outra abordagem que utiliza *middleware* de tolerância a falhas que segue uma abordagem de replicação *leader* / *follower* para tolerar faltas de acidente tem sido proposta em [23]. No entanto, todas essas técnicas têm limitações mencionadas e ilustradas nos capítulos 3 e 4, e ou toleram apenas um tipo específico de falha ou fornecer um único método de resistência.

A revisão bibliográfica a respeito dos sistemas tolerantes a falhas deixou com muita clareza que muitas técnicas para tolerância a falhas são e estão sendo investigadas e implementadas. Sob este aspecto, é um tema em evidência que necessita ser levado em consideração pelos clientes e fornecedores da Nuvem, no intuito de que sempre se busca a otimização da fiabilidade, disponibilidade na Computação em Nuvem, uma vez que são requisitos essenciais para garantir um funcionamento adequado e contínuo, mesmo na presença de falhas.

## 1. 2 Motivação

Partindo da revisão bibliográfica, pode-se observar que hodiernamente existem muitas técnicas para tolerância a falhas sendo investigadas e implementadas. Neste contexto, a pergunta que merece preocupação é como o usuário poderia saber qual das técnicas escolher considerando a natureza do seu trabalho.

Percebe-se que dependendo do perfil da Nuvem que se deseja montar, e levando em consideração sempre o custo, pode-se escolher um modelo que satisfaça as métricas desejadas. Neste quadro, faz-se necessário apresentar uma visão geral da Computação em Nuvem, a tolerância a falhas e seu lugar dentro da confiabilidade, os modelos existentes, técnicas e abordagens das diferentes classes de tolerância a falhas, assim como sua implementação, podendo ajudar usuários ingênuos para suas necessidades no que concerne a escolha de um sistema tolerante a falha adequado para o seu negócio.

## 1. 3 Objetivos

O principal propósito desta pesquisa em um estudo detalhado sobre abordagens de diferentes sistemas tolerantes a falha em uma Nuvem computacional, as ferramentas utilizadas para a sua implementação, especificando suas políticas, os *frameworks*[5] de desenvolvimento, os sistemas, os ambientes e tipos de falhas detectadas. Em seguida, apresentar e discutir alguns elementos de métricas válidas para examinar os desafios da tolerância a falhas, comparando os modelos existentes através de tabelas e gráficos de tendências, para servir de base à escolha de um sistema tolerante a falhas em ambientes de Computação em Nuvem, que satisfaça à necessidade do usuário, sustentando a escalabilidade e flexibilidade existentes nas infraestruturas de Computação em Nuvem.

Para tal, é imprescindível que sejam analisados livros e artigos que relacionam os principais atributos da Computação em Nuvem, das estratégias de virtualização e das abordagens de sistemas tolerantes a falhas, almejando os seguintes objetivos:

- Estudar conceitos, princípios e aplicações de Computação em Nuvem, administração, virtualização e tolerância a falhas;
- Estabelecer um conhecimento em torno dos sistemas de tolerantes a falhas existentes;
- Aplicar as métricas para a tolerância a falha em Computação em Nuvem;
- Fazer uma comparação entre vários modelos baseado nos elementos de métricas;

Portanto, a partir destas etapas será possível fazer considerações conclusivas sobre a Computação em Nuvem, a virtualização e a abordagem de sistemas tolerantes a falhas em uma Nuvem computacional.

[5] Uma biblioteca de classes que captura padrões de interação entre os objetos. A estrutura consiste de um conjunto de classes concretas e abstratas, explicitamente projetados para serem usados em conjunto.

## 1. 4 Organização do trabalho

Esta monografia está estruturada da seguinte maneira:

- **Capítulo 1** – INTRODUÇÃO: É apresentada a introdução geral do trabalho. São evidenciados o contexto da Computação em Nuvem, a problemática da tolerância a falha neste contexto, sua importância.
- **Capítulo 2** – FUNDAMENTOS TEORICOS: Neste capítulo são apresentadas as principais características das infraestruturas de Computação em Nuvem. A Computação em Nuvem apresentada na forma de suas categorias de serviços e aplicações e modelos de implantação, descrevendo as nuvens privadas, comunitárias, públicas e híbridas, características da Computação em Nuvem, classificações, princípios, benefícios e administração, a computação entre a Computação em Nuvem e computação em grade, a virtualização, sistemas de tolerância a falhas e tolerância a falhas em Computação em Nuvem.
- **Capítulo 3** – ABORDAGENS DE SISTEMAS TOLERANTES A FALHAS EXISTENTES: São destacadas neste capítulo as principais técnicas de tolerância a falhas, os modelos de tolerância a falhas e ferramentas utilizadas para implementação de mecanismos tolerantes a falhas.
- **Capítulo 4 –** ANÁLISE COMPARATIVA: Aqui são definidos os elementos de métricas de tolerância a falhas. Tais métricas são úteis para a comparação entre os modelos existentes. Além disso, é feita uma avaliação de modelos baseando-se nas métricas.
- **Capítulo 5** – CONCLUSÕES: As conclusões são apresentadas a partir dos capítulos anteriores, levando em consideração o objetivo dessa pesquisa. Também são apresentadas observações e sugestões para trabalhos futuros relacionados ao tema desta monografia.

# 2 FUNDAMENTOS TEÓRICOS

Neste capítulo são apresentados conceitos gerais relacionados à computação em nuvem e à forma como ela se apresenta no contexto atual, a fim de expor alguns conceitos específicos sobre computação em nuvem, a virtualização e a visão geral de tolerância a falhas, para facilitar a compreensão dos capítulos posteriores.

## 2.1 Computação em Nuvem

A primeira pergunta é o que é a Computação em Nuvem?

A tradução literal de "Computação em Nuvem" propõe uma informática (computação) desmaterializada, podendo ser ofertada à demanda. Esta filosofia não é nova, pois relembra os conceitos de informática utilitárias propostas por John McCarthy em 1961. De onde se partiu para se chegar a esta Computação em Nuvem? De que modo a Computação em Nuvem é constituída? Quais são suas diferentes arquiteturas? O que ela vai trazer à computação de hoje e de amanhã?

### 2.1.1 Histórico

Não há uma data-chave a qual possamos atribuir o nascimento da Computação em Nuvem.

A noção de Computação em Nuvem se refere a uma Nuvem, tal que se tem o hábito de utilizá-lo em seus esquemas técnicas quando se quer representar a Internet. Uma rede como a Internet é constituída de vários sistemas fornecendo serviços e informações. A Computação em Nuvem é, nesta ordem de ideia, um conjunto de serviços e de dados consumíveis [1].

#### *2.1.1.1 Informática Utilitária de John MCCARTHY*

A noção de consumo foi proposta em 1961, durante uma conferência em Massachusetts Institute of Technology (MIT), por John McCarthy, também conhecido como um dos pioneiros da inteligência artificial e criador da linguagem do LISP em.

Neste discurso, John McCarthy sugerira que a tecnologia Informática compartilhada ("*Time sharing*") pudesse construir um belo futuro no qual o poder de cálculo, inclusive as aplicações específicas pudessem ser vendidas como um serviço público.

Esta ideia, demasiada popular nos anos 60, desapareceu em meados dos anos 70:

na época, as tecnologias de hardware, *software* e redes ainda não estavam prontas.

A Computação em Nuvem estabelece a ideia de informática utilitária do tipo serviço pública, proposta por John McCarthy. Esta pode ser comparada ao cluster de cálculo, no qual um grupo de computadores são conectados um ao outro para formar um computador virtual único, permitindo o cálculo de alto desempenho (HPC), também ao *Grid Computing* onde os computadores ligados e distribuídos geograficamente permitem a resolução de um problema comum.

#### *2.1.1.2 Serviços Bureau*

É nesta filosofia que desde os anos 70 foi inventada a noção de "s*ervices bureau*" para qualificar uma organização alugando linhas telefônicas, serviços informáticos e etc. Geralmente os clientes dos "*services bureau*" não têm experiência para integrar estes serviços na organização, por isso eles passam pelo provedor. A combinação de tecnologias, processos e experiências na área organizacional é o valor acrescentado dos "*services bureau*", como modelo econômico baseado na sua capacidade de produzir e implantar serviços em volume [1]. A própria IBM era um "services bureau" ao propor o conceito de "*on-demand*[6]".

Na época, o custo de compra e de uso de mainframes IBM era inestimável. Para isso, as soluções permitindo que as organizações pudessem usar estas tecnologias com um custo menor com o conceito de "pagamento para o consumo" foram propostas.

#### *2.1.1.3 Provedor de serviços de aplicações*

Os ASP, "*Application Service Provider*", também têm sua parte no histórico da Computação em Nuvem. Um ASP designa uma aplicação fornecida como um serviço, é o que se chama hoje SaaS para "*Software as a Service*", no contexto terminológico hodierno da Computação em Nuvem. Ao invés de instalar o software na máquina cliente assegurando as fases de instalações e de manutenção sobre cada uma, as aplicações disponibilizadas pelos ASP são hospedadas e centralizadas num servidor único e acessível a todos os clientes através de protocolo padrão. A título de exemplo, temos as aplicações web acessíveis via protocolo HTTP[7], as quais não impõem mais implantações ou manutenções na máquina do usuário,

[6] No contexto de TI, é uma instalação principal e característica dos serviços de Computação em Nuvem, que permitem que os usuários disponham de recursos de Nuvem em tempo de execução, quando e onde for necessário.

[7] É sigla de *HyperText Transfer Protocol* que em português significa "Protocolo de Transferência de Hipertexto". É um protocolo de comunicação entre sistemas de informação que permite a transferência de dados entre redes de computadores,

sendo que este apenas precisa de um simples *browser*. A implantação, a configuração, a manutenção e o *backup* são doravante responsabilidades do provedor do serviço, o cliente é então o consumidor.

#### *2.1.1.4 A Virtualização*

Como veremos mais adiante, a virtualização foi a primeira pedra para a era da Computação em Nuvem. De fato, este conceito permite um gerenciamento otimizado de recursos físicos a fim de poder executar vários sistemas "virtuais" num só recurso físico e fornecer uma camada suplementar de abstração do material (*hardware*). Os primeiros trabalhos podem ser atribuídos a IBM, que nos anos 60, já trabalhava sobre os mecanismos de virtualização desenvolvendo nos centros de pesquisas de Cambridge e de Grenoble, CMS (*Conversation Monitor System*), o primeiro *Hypervisor*[8].

Pode-se depreender do acima exposto, que a aproximadamente 50 anos a ideia da informática *on-demand* é presente nas mentes dos indivíduos, ainda que até então não existisse tecnologia adequada para se concretizar.

Com os diferentes progressos tecnológicos realizados nos últimos 50 anos, tanto da parte *hardware*, *software* e conceitual, nos avanços de mecanismos de segurança, à implantação de redes complexas mais padronizadas como Internet, e à experiência na edição e o gerenciamento de softwares, serviços, infraestrutura e armazenamento de dados, agora estamos prontos a entrarmos na era da Computação em Nuvem, tal como sonhava John McCarthy em 1961 [1].

### 2.1.2 Generalidade

Diante do aumento contínuo de custos de implantação e de manutenção de sistemas computacionais, as organizações externalizam cada vez mais seus serviços computacionais. Elas passam seu gerenciamento (de serviços computacionais) pelas organizações dedicadas ou especializadas (que chamamos provedores). O interesse principal desta estratégia reside no fato de que a organização só paga pelos serviços efetivamente consumidos.

De fato, um gerenciamento interno desses serviços pela própria organização não

---

principalmente na *World Wide Web* (Internet).

[8] É uma plataforma que permite aplicar diversas técnicas de controlo de virtualização para utilizar, ao mesmo tempo, diferentes sistemas operativos (sem modificar ou modificar-los em caso deparavirtualização) no mesmo computador (wikipedia).

seria completamente amortecido, em particular quando as necessidades de uso variam. O desenvolvimento desse modo de funcionamento foi favorecido por vários fatores, tais como a evolução e a generalização de acesso à Internet e o aumento do poder de computadores e de redes computacionais. A Computação em Nuvem se situa nesta recente orientação.

### 2.1.3 Definição

Como mencionado acima, a Computação em Nuvem é uma Nuvem de serviços e de dados. Mais precisamente, é um paradigma, e há falta de consenso sobre a definição de seu conceito. Entretanto, tomamos a definição proposta por CISCO [2]: "A Computação em Nuvem é uma plataforma para compartilhamento de dados, fornecendo às organizações serviços *on-demand* com a ilusão de recursos infinitos". Nesta definição, encontramos algumas semelhanças com as plataformas conhecidas como *Grid Computing* ou centros de hospedagem. Na literatura, a Computação em Nuvem apresenta-se como uma evolução destas infraestruturas de hospedagem compartilhadas. Como um exemplo, consideremos o método proposto por [3] (Figura 1), a qual é amplamente referida na literatura de desenvolvimento. De acordo com [3], a Computação em Nuvem é o resultado de uma evolução que pode ser apresentada em cinco fases:

- Ela começa com os Provedores de Acesso a Internet (PAI 1.0). Eles têm como objetivo estabelecer meios de telecomunicação para assegurar a conexão de pessoas e organizações para a Internet.
- A segunda fase é a orientação dos PAI para a hospedagem de páginas web (PAI 2.0). Esta fase marca um grande salto no desenvolvimento da Internet.
- A terceira fase (PAI 3.0) é a possibilidade oferecida pelos PAI de hospedar aplicativos de negócios organizacionais.
- Um conhecimento das necessidades de aplicações de organizações permite aos PAI de desenvolverem sua área de intervenção. Eles estabelecem plataformas de geração de aplicativos *on-demand*. Esta é a ASP (*Application Service Provider*) cujos "*Software as a Service*" (Seção 2.1.5.2) são derivados: o PAI 4.0.
- A generalização de práticas anteriores, levando em conta as novas práticas e a integração dos princípios que apresentamos nas seções seguintes dá à luz à Computação em Nuvem.

Seguindo essa tendência, apresenta-se na próxima seção uma perspectiva sobre a posição da Nuvem em relação às plataformas de hospedagem, tais como grades [5] (Grid5000 [6], Globus [7]).

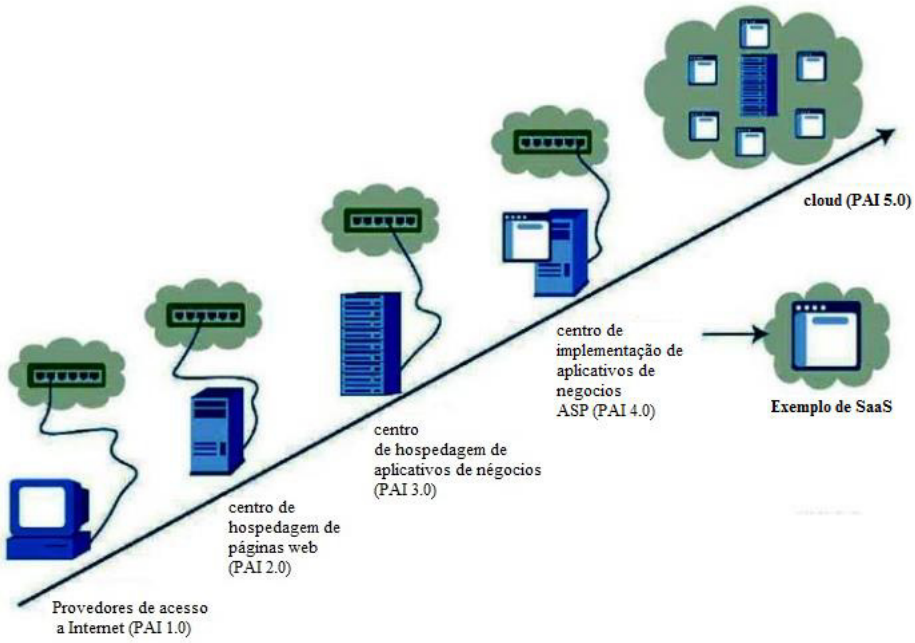



**Figura 1: Evolução para a Nuvem [15].**

A ideia de colocar aplicativos e dados em uma nuvem única, acessíveis por todos e distribuído em milhares de máquinas "abstratas" pode ser assustadora. Como veremos, os avanços em termos de segurança, tanto tecnológico quanto intelectual, nos permitem garantir uma confiança ótima.

A Computação em Nuvem não impõe nenhuma despesa de capital, uma vez que os serviços são pagos com base no uso. Isso permite que as organizações não se focalizem mais na gestão, manutenção e operação da infraestrutura ou serviços de aplicativos.

Os fortes avanços em virtualização tornaram possível a Computação em Nuvem. Esta virtualização permite otimizar os recursos de *hardware*, compartilhando-os entre vários ambientes ("*time-sharing*"). Do mesmo modo, é possível acoplar o aplicativo (e seu sistema operacional) e o equipamento (encapsulado na máquina virtual), proporcionando também um "*provisionning*", isto é, a capacidade de implantação de ambiente, de forma automática.

A Computação em Nuvem acoplada com as tecnologias de virtualização, permite o fornecimento de infraestrutura e plataforma *on-demand*. Mas a Computação em Nuvem não é apenas a infraestrutura (IaaS), ela perturba a plataforma de execução (PaaS) e as aplicações

(SaaS). Como veremos mais adiante, a Nuvem é tanto transversal quanto vertical.

Basicamente, a Nuvem propõe três camadas:

- A infraestrutura (IaaS: *Infrastructure as a Service*);
- A plataforma (PaaS: *Platform as a Service*);
- A aplicação (SaaS: *Software as a Service*).

Como mencionado no livro branco [4]: "Todo SaaS é um serviço Nuvem, mas toda Nuvem não é um serviço SaaS", podemos falar de Nuvem para uma aplicação SaaS, mas ele não para na camada aplicação, como vimos anteriormente.

Dado que a Computação em Nuvem é um conjunto de máquinas interconectadas e massivamente distribuídas, isto permite que seja tolerante a falhas de *hardware* e *software* (para mecanismos de redundância e replicação). Além disso, a Computação em Nuvem fornece elasticidade e flexibilidade. Assim, o poder de computação e a capacidade de armazenamento podem ser facilmente aumentados com a instalação de novos equipamentos dentro ou fora da Nuvem, de modo que a carga será distribuída de acordo com os novos equipamentos. Acontece o mesmo com as aplicações e os dados: eles podem ser acomodados em diferentes "lugares". Se houver aumento da carga, a Nuvem irá criar várias instâncias, a fim de distribuir a carga e garantir a disponibilidade máxima.

### 2.1.4 Computação em Nuvem vs Grid Computing

Introduzidas pela primeira vez em 90 anos, as *Grids* (grades) da Figura 2(esquerda) situam o compartilhamento mútuo (*pooling*) no coração de sua tecnologia. Da mesma forma, o compartilhamento está no coração da tecnologia Computação em Nuvem (Figura 2(direita)). A pergunta que fazemos: "Em que nível é a diferença entre a nuvem e as grades (se existir)?". Em outras palavras, o termo "Computação em Nuvem" não é outra forma de chamar a "*Grid Computing*"? Após consulta à literatura, foi encontrado apenas um estudo sério consagrado a esta comparação ([8]).

Do ponto de vista tecnológico, não se identifica a diferença real entre plataformas de grade e Nuvem. No entanto, a abertura de Nuvem aos usuários de diferentes níveis (nem sempre os profissionais de computação, como nas grades) e possivelmente seu caráter comercial, são potenciais diferentes que se identificam. Além disso, nos ambientes de grades,

as aplicações (chamadas normalmente de *job*[9]) se executam geralmente por um tempo limitado (embora possa ser ilimitado na Nuvem). Como se apresenta na seção 2.2.4, essas diferenças tornam a gestão e utilização de plataformas da Nuvem mais complexas do que nas grades.

Em suma, considera-se que a tecnologia de Computação em Nuvem usa a tecnologia de grade na qual associa os princípios de abertura ao público, faturamento e hospedagem sustentável. Na próxima seção, serão detalhados esses princípios.

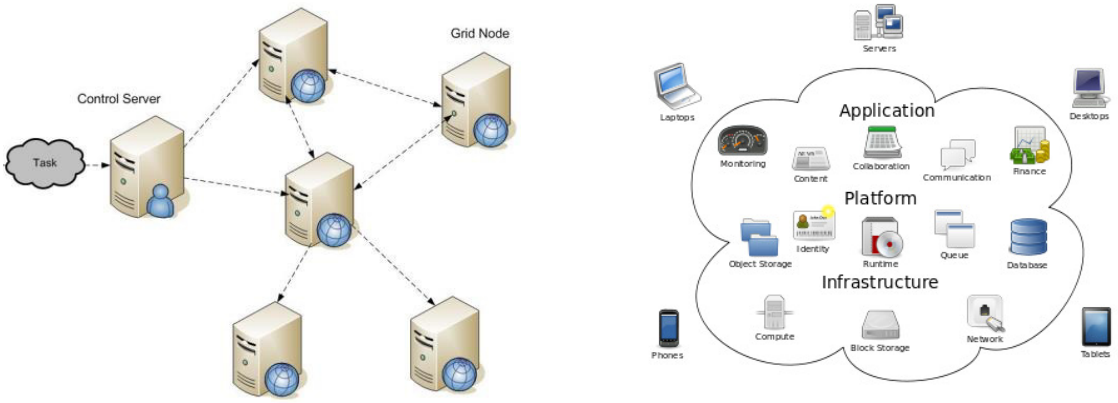



**Figura 2: Exemplo de Grid Computing (esquerda) e Computação em Nuvem(direita) [16].**

[9] O conceito básico num sistema de lotes, sendo constituído do programa a ser compilado e executado, acompanhado dos dados.

### 2.1.5 Características da Computação em Nuvem

As principais vantagens da Computação em Nuvem são a escalabilidade, o *pay-per-use,* elasticidade, confiabilidade, disponibilidade e a tolerância a falhas [68].

#### *2.1.5.1 Elasticidade*

Tal característica é um dos principais fatores para o sucesso da Nuvem como uma infraestrutura de TI [67]. Para um Sistema de Gerenciamento de Banco de Dados (SGBD) implantado em uma infraestrutura de Nuvem *pay-per-use*, um objetivo adicional é aperfeiçoar o custo operacional do sistema. Elasticidade, ou seja, a capacidade de lidar com variações de carga, adicionando mais recursos durante a alta carga ou consolidar os locatários para menos nós quando a carga diminui, isto tudo em um sistema a tempo real, sem interrupção do serviço é, portanto, fundamental para esses sistemas. Mesmo que a elasticidade seja frequentemente associada à escala do sistema, existe uma sutil diferença entre elasticidade e escalabilidade quando usada para expressar o comportamento de um sistema.

#### *2.1.5.2 Escalabilidade*

A escalabilidade é uma propriedade desejável de um sistema, o que indica a sua capacidade de lidar ou com quantidades crescentes de trabalho de uma forma harmoniosa ou com a sua capacidade para melhorar o rendimento quando os recursos adicionais (tipicamente *hardware*) são adicionados. Um sistema cujo desempenho melhora após a adição de *hardware*, proporcionalmente à capacidade acrescentada, é dito ser um sistema escalável.

#### *2.1.5.3 Confiabilidade*

Ao se falar da confiabilidade, vê-se a probabilidade de que um produto ou parte de um produto funciona corretamente durante um determinado período de tempo (*design life*), nas condições operacionais de projeto (tais como temperatura, volts, etc) sem falha [69]. O resultado do processo de medição é reproduzível, sendo semelhante ao resultado ao longo do tempo para algumas entradas diferentes e em muitas situações diferentes. A Nuvem recebe muitos pedidos ao mesmo tempo e também vai dar os resultados semelhantes para alguns pedidos em um período de tempo, assim as nuvens têm de ser confiáveis.

#### *2.1.5.4 Disponibilidade*

Neste caso, o serviço de Nuvem deve estar disponível o tempo máximo [7]. A natureza *on-demand*, elástica, escalável e personalizável da Nuvem devem ser consideradas ao implantar arquiteturas de Nuvem. Muitos clientes diferentes podem estar acessando os mesmos aplicativos de *back-end*, e muitos provedores de serviços de Nuvem têm a expectativa de que apenas suas aplicações serão devidamente entregues aos usuários. Na Computação em Nuvem é essencialmente necessário reunir as informações instantaneamente, sem fazer com que o usuário espere e que as informações coletadas sejam relacionadas entre si.

#### *2.1.5.5 Custo da Computação em Nuvem*

Permite que uma organização pague por hora de recursos computacionais, podendo levar à redução de custos, mesmo que a taxa horária para alugar uma máquina a partir de um provedor de Nuvem seja maior do que a taxa de possuir uma. Isto é preferível quando essencialmente procura-se um serviço que varia ao longo do tempo.

#### *2.1.5.6 Tolerância a Falhas*

A tolerância a falhas é uma das questões-chave da Computação em Nuvem. Há muitas técnicas de tolerância a falhas em computação paralela [11]. A tolerância a falhas se preocupa com todas as técnicas necessárias para permitir que um sistema tolere falhas de software. Estas falhas de *software* podem ou não se manifestarem durante as operações de sistemas, mas quando o fazem, técnicas tolerantes a falhas de *software* devem fornecer os mecanismos necessários para prevenir ocorrências de falha de sistema.

A Figura 3 ilustra as características da Nuvem detalhadas nesta seção.

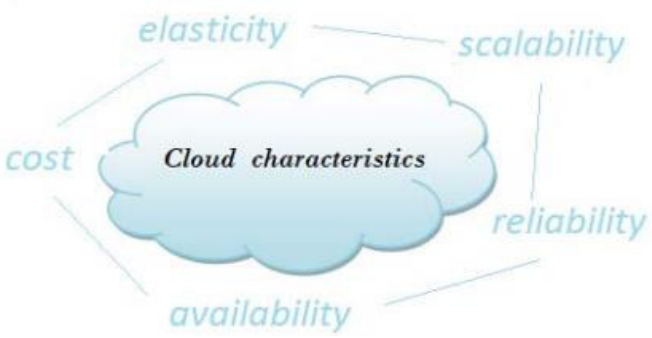



**Figura 3: Umas características da Computação em Nuvem [74].**

### 2.1.6 Princípios de Computação em Nuvem

Na Seção 2.1.5 foram enunciados alguns princípios fundamentais da Nuvem. Esses princípios permitem que a Nuvem se destaque das plataformas convencionais (centro de hospedagem, *datacenter*[10], grade). Os mais importantes dentre eles são o compartilhamento mútuo e a cobrança de uso de recursos[33][34]:

#### *2.1.6.1 Mutualização*

É a prática que consiste em partilhar o uso de um conjunto de recursos pelas organizações (ou entidade qualquer) mesmo não tendo nenhum laço entre si. Os recursos podem de natureza variada: *Software* ou hardware (máquina, equipamentos de rede, energia elétrica). Esta prática depende do desejo das organizações de deslocarem seus serviços computacionais para as infraestruturas de Nuvem.

Tendo como fundamento as tecnologias de grades, a Nuvem deve, no entanto, encarar os problemas ligados ao seu funcionamento e uso. Faz-se referencia aos problemas clássicos, como a segurança, a disponibilidade, a integridade, a fiabilidade e a uniformidade de acesso aos dados.

[10] Um ambiente projetado para abrigar servidores e outros componentes como sistemas de armazenamento de dados (*storages*) e ativos de rede (*switches*, roteadores).

#### *2.1.6.2 Alocação e cobrança on-demand*

Diferentemente dos centros de hospedagem web clássicos cujo pagamento é feito com antecedência, e com um valor fixo, a Nuvem propõe uma cobrança ao uso. Esta pode se ser realizada em duas formas. A primeira consiste em faturar à organização a duração do uso de um conjunto de recursos independente do uso efetivo. Por exemplo, seja *r* o conjunto de recursos reservados pela organização. Sejam $t_1$ e $t_2$ respectivamente o instante do inicio e do fim de uso de recursos. Seja $C_u$ o custo de uso de um recurso durante uma unidade de tempo. Desta forma, o custo total de uso do conjunto de recursos $r$ é: $Cr = (t_2 - t_1) * C_u * r$. Quanto à segunda forma, ela é muito mais fina do que a primeira. Ela consiste em faturar instantes durante os quais os recursos foram realmente usados. Usando os mesmo parâmetros que anteriormente, seja $T = \{t_{u1}, t_{u2}, \ldots, t_{un}\}\ com\ \ t_{ui} \in [t_1, t_2], 1 \leq i \leq n,$ o conjunto de unidades de tempo durante os quais os recursos foram realmente utilizados. Neste caso, o custo total de uso do conjunto de recursos r é: $Cr = Cu * r * \sum_{i=1}^{n} t_{u1}$.

À luz dessas práticas, pode-se depreender que o uso da Nuvem pode ser benéfico tanto para o provedor quanto para as organizações que o utilizam.

Os benefícios do provedor tão somente são devido ao fato do compartilhamento mútuo de recursos. Além do mais, o mesmo recurso pode ser objeto de várias cobranças.

Quanto à organização, ela é a primeira a ganhar desta tecnologia. Ela realiza benefícios em dinheiro e em flexibilidade na sua capacidade de crescer.

### 2.1.7 Classificações

Antes de se apresentar os diferentes tipos de Nuvem, estabelece-se em primeiro lugar alguns critérios de classificação:

- A razão de desenvolvimento (*business model*). Esta é a razão para a criação da plataforma. Ela pode ser comercial, científica ou comunitária.
- Nível de serviço. É a ambição da Nuvem para fornecer às organizações uma plataforma que atenda ou não às suas expectativas.
- Acessibilidade. A Nuvem pode ser acessível por todos (“Nuvem pública”) ou restrito a um público específico (“Nuvem privada”). Diferentemente dos dois primeiros, este não irá ser detalhado haja vista sua simplicidade de compreensão.

#### *2.1.7.1 Por razão de desenvolvimento*

O uso de Computação em Nuvem não se limita apenas às organizações em caráter comercial. Em função das razões de seu estabelecimento, distinguem-se quatro categorias de plataformas de Computação em Nuvem, a saber:

**Nuvem de Organizações**. Nesta categoria, encontramos as organizações de pequeno e médio porte cada uma pouco recurso e condição de manutenção de suas infraestruturas. Elas, portanto, se agrupam em torno de um projeto de Nuvem para reunir as suas capacidades. A plataforma resultante é privada, isto é, acessível apenas pelas entidades de diferentes organizações. Esta plataforma tem a vantagem de ser pequena e restrita a usuários conhecidos. Assim, os problemas de segurança são reduzidos e a administração pode ser especializada. Os grupos do Amazon EC2 [9] (através do "*Virtual Private Cloud*"), VMware[11] [11] ou VeePee[12] [10] oferecem essas soluções de Nuvens privadas.

Em comparação com as tecnologias existentes, esta categoria é idêntica aos clusters privados.

**Nuvem Governamental e Pesquisa Científica**. Por motivo de pesquisa e de desenvolvimento, os centros de pesquisas implantam os ambientes de Nuvem. Seu desenvolvimento é apoiado e financiado pelos governos. O acesso é exclusivamente permitido para os usuários da área de pesquisa, ou para pertencentes aos centros de pesquisa associados, ou tendo uma derrogação específica. A maioria destas plataformas é orientada a projeto. Somente os avanços científicos obtidos pelos grupos de pesquisa que as utilizam permitem valorizar a plataforma.

**Nuvem para o Provedor de Serviços**. Este é o modelo mais comum. Uma organização, chamada provedor, disponibiliza para outras (chamadas clientes) uma plataforma de execução de aplicações. Diz respeito de um modelo aberto a todo público e a caráter comercial. A plataforma hospeda todos os tipos de aplicações e o acesso a esses aplicativos é aberto a usuários externos. Os desafios de segurança e administração são importantes neste modelo. A plataforma de CC Amazon Elastic Compute Cloud (EC2) faz parte desta categoria.

---

[11] VMware é um software/máquina virtual que permite a instalação e utilização de um sistema operacional dentro de outro dando suporte real a software de outros sistemas operativos.

[12] Fundado em 2000, VeePee é um operador de serviço de IP que fornece uma gama de serviços em toda a cadeia de IP. Ele é ao mesmo tempo um operador da infra-estrutura (IaaS) e um provedor de serviços que oferece um pacote de serviços em todas as áreas críticas do sistema de gestão da informação.

### *2.1.7.2 Por nível de Serviço*

Em função do nível de abstração que oferece a Nuvem às organizações, na literatura, nós identificamos três categorias de plataformas de Computação em Nuvem (Figura 4).

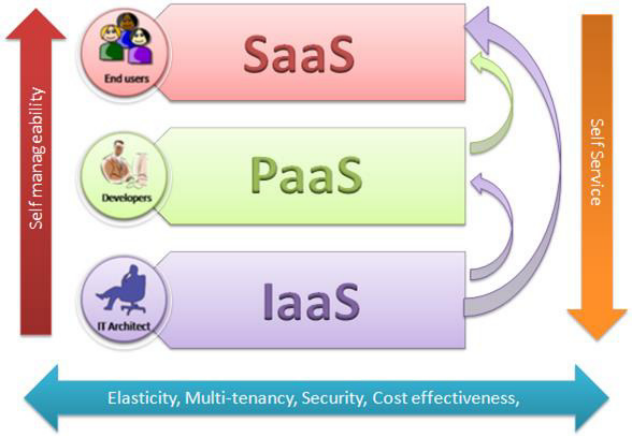



**Figura 4: Diferentes camadas de Computação em Nuvem[17].**

**Infrastructure as a Service (IaaS)**: A infraestrutura fornece as capacidades de tratamento e de armazenamento assim como uma conectividade em rede. Os servidores, sistemas de armazenamento, comunitários, roteadores e outros equipamentos, são postos à disposição para gerenciar uma carga de trabalho requerida pelas aplicações.

A infraestrutura como um serviço ou IaaS, permite dispor de uma infraestrutura on-demand, podendo hospedar e executar aplicativos, serviços ou ainda armazenar dados.

Concretamente, isso se caracteriza por uma infraestrutura física algumas vezes posta à disposição por um provedor de serviços. Encontrar-se-á uma solução de virtualização permitindo a criação de "*datacenters virtuais*".

Graças às soluções de virtualização existentes, torna-se possível criar muito facilmente máquinas virtuais conectadas em redes, sendo elas também virtuais, e que serão executadas nos "*hypervisors*" das máquinas físicas.

Esta virtualização nos fornece uma grande flexibilidade porque permite abstrair a camada física sobre a qual as aplicações que poderão ser distribuídas e redistribuídas sem serem ligadas a um servidor específico. A virtualização responde de forma dinâmica onde os servidores físicos fornecem um conjunto de recursos alocados segundo as necessidades, e onde a relação entre as aplicações e os recursos de tratamento, de armazenamentos e de rede,

poderão se adaptar de forma automática para responder à carga de trabalho e às exigências requeridas. Pode-se encontrar na Figura 5 a exemplificação de uma infraestrutura IaaS.

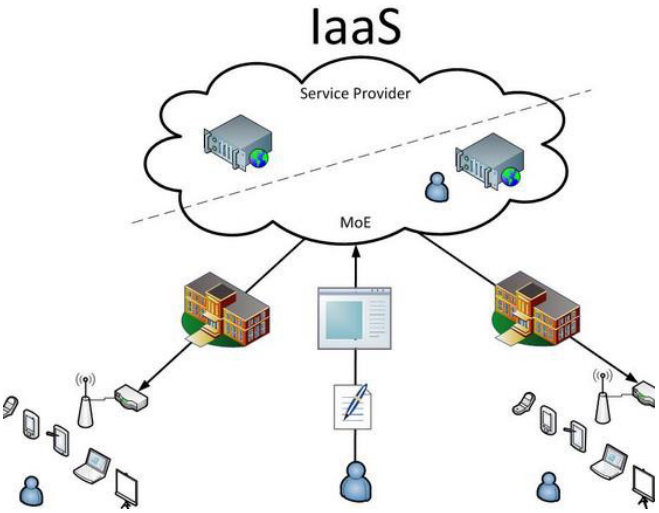



**Figura 5: Exemplo de Serviço Infrastructure as a Service(IaaS) [18].**

**Platform as a Service (PaaS):** A plataforma como serviço é a plataforma de execução, de distribuição e de desenvolvimento de aplicações. Refere-se a um nível de abstração acima de IaaS no qual a Nuvem fornece uma plataforma aplicativa de programação. Ela permite às organizações programar aplicações facilmente administráveis na Nuvem. Ela obriga a organização de um lado dominar a *Application Programming Interface* [13](API) do PaaS e de outro lado reimplementar suas aplicações segundo essa API. *Google App Engine*[12] e *Windows Azure*[13] são exemplos de PaaS.

Graças as PaaS, a distribuição de aplicações em diferentes ambientes pronto para usar é muito fácil (teste, pré-produção e produção sem se preocupar da infraestrutura e da plataforma nas quais irão se executar a aplicação ou o armazenamento de dados). O Exemplo desta plata forma pode ser visto na Figura 6.

[13] Um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços (wikipedia).

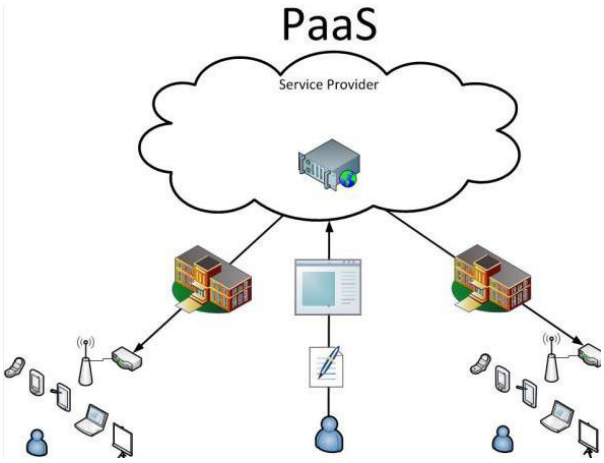



**Figura 6: Exemplo de Serviço Platform as a Service (PaaS) [18].**

**Software as a Service (SaaS):** Aqui, a Nuvem fornece diretamente as aplicações correspondentes às expectativas das organizações. As tarefas de administração são da responsabilidade da Nuvem (Figura 7); a organização não tem muita coisa que efetuar. Muito especializado (pela aplicação que ele fornece), esse tipo de Nuvem é o mais expandido. As nuvens para redes sociais e jogos são exemplo dessa categoria.

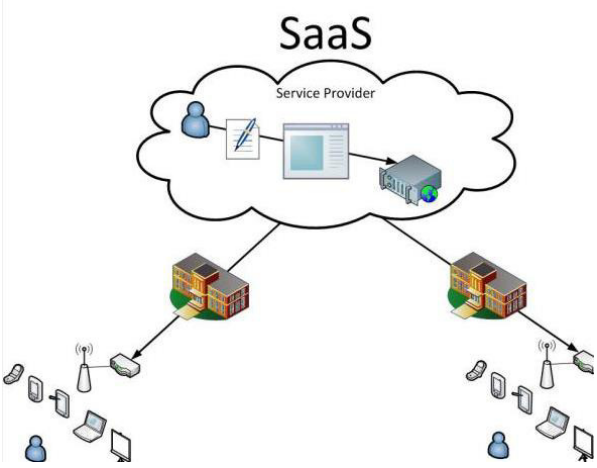



**Figura 7:Exemplo de Serviço Software as a Service (SaaS) [18].**

### 2.1.8 Modelos de implantação

Como vimos, a Nuvem é baseada em recursos físicos. A pergunta é “onde estão esses recursos físicos?” (Servidores, *switches*, roteadores, soluções de armazenamento, etc.).

A resposta "na Nuvem" não é realmente aceitável. De ponto de vista do consumidor, a abstração é tal que podemos determinar sobre quais recursos físicos a aplicação está hospedada. Por seu lado dinâmico, os recursos físicos que hospedam uma aplicação e dados em uma Nuvem nunca são fixos e mudam ao longo do tempo.

Em teoria, a Computação em Nuvem não impõe nenhuma despesa de imobilização. Usam-se geralmente os recursos físicos de um provedor de Nuvem. No entanto, esta tecnologia pode muito bem ser a infraestrutura física de uma organização: não sendo mais compartilhada, a Nuvem se torna privada: falar-se-á então de Nuvem pública, Nuvem privada, Nuvem comunitária e Nuvem híbrida.

#### *2.1.8.1 Nuvem privada*

Estes recursos físicos podem ser hospedados em infraestrutura própria de uma organização, e sendo sob seu controle, é, portanto, sua responsabilidade de controlar a implantação de aplicativos.

Voltaremos a esse conceito de Nuvem privada, mas já podemos nos perguntar se "uma Nuvem privada é realmente uma Nuvem?". Na verdade, no sentido de que, como dito acima, uma Nuvem não deve impor despesas em imobilização, a infraestrutura física em uma Nuvem privada é de responsabilidade da organização.

A Nuvem privada também pode designar uma Nuvem implantada em uma infraestrutura dedicada e disponibilizada para um provedor de serviços.

Assim, uma organização pode contratar de um provedor de serviços, um número significativo de servidores que são totalmente dedicados a ela, e sobre os quais a solução de Nuvem será implantada para gerenciar dinamicamente a aplicação, a plataforma ou a infraestrutura (virtual).

#### *2.1.8.2 Nuvem Pública*

Uma Nuvem pública é um serviço IaaS, PaaS ou SaaS proposto e oferecido por um terceiro.

Amazon, Google e Microsoft oferece uma Nuvem pública em que qualquer indivíduo ou organização pode hospedar suas aplicações, serviços e dados. Para os consumidores, não há nenhum investimento fixo inicial e nenhum limite de capacidade.

Os provedores de Nuvem pública cobram o uso e garantem a disponibilidade de serviços através contratos SLA: (documento que define a qualidade de serviço necessária entre um fornecedor e um cliente).

#### *2.1.8.3 Nuvem Comunitária*

Em uma Nuvem Comunitária a infraestrutura de Nuvem é controlada por vários clientes que se juntam para formar uma Nuvem que atende às suas necessidades específicas, particularmente do ponto de vista de controle, segurança e conformidade. Por exemplo, o Hospital *Mont-Sinaï* e outros 14 hospitais locais em Toronto estão trabalhando na criação de uma Nuvem comunitária para acessar um aplicativo para o ultrassom fetal e armazenar dados do paciente.

#### *2.1.8.4 Nuvem Híbrida*

Uma Nuvem Híbrida é a utilização de várias nuvens, públicas ou privadas. Assim, podemos delegar nossos aplicativos para uma Nuvem pública que vai consumir os dados armazenados e expostos em uma Nuvem privada, ou fazer interagir duas aplicações hospedadas em duas Nuvens privadas separadas, ou usar vários serviços hospedados em diferentes nuvens públicas. Em todos esses casos, estamos lidando com a noção de Nuvem híbrida.

### 2.1.9 Benefícios e Vantagens de Computação em Nuvem

Segue as umas vantagens relevantes:

- ✓ Nenhum investimento prévio
- ✓ Nenhum pré-requisito exigido

- ✓ Fácil implantação
- ✓ Gerenciamento simplificado
- ✓ Serviço de alta disponibilidade e adaptável
- ✓ Pagamento relativo ao consumo

Para o usuário, como uma *start-up*[14], por exemplo, a principal vantagem é a de beneficiar de um serviço personalizado, sem investimento e capaz de absorver qualquer pico de carga.

A organização Kobojo oferece jogos e aplicativos para redes sociais. Fundada em setembro de 2008, a organização tem, em menos de um mês, tido um crescimento de 5 000 de 700 000 usuários por dia. Hodiernamente, é mais de 2,5 terabytes baixados diariamente, com picos de carga muito importantes, entre 18 e 23 horas.

Difícil para uma pequena *start-up* investir em uma infraestrutura que pode crescer com tão pouco tempo para absorver tal carga. A Computação em Nuvem permitiu-lhes ter uma plataforma e uma infraestrutura para hospedar suas aplicações sem investimento financeiro, vantagem significativa na criação de uma estrutura deste tipo.

Além disso, eles foram capazes de responder dinamicamente à carga por meio da proposta de flexibilidade de Computação em Nuvem. Kobojo hoje tem mais de 50 milhões de usuários.

A Computação em Nuvem também elimina as restrições de gerenciamento de implantação e operação de tal arquitetura. *Start-ups* podem doravante se concentrar em seu negócio, Computação em Nuvem está ao seu lado para superar o problema de operar uma plataforma altamente disponível.

Claramente, a Computação em Nuvem pode reduzir os custos operacionais através da partilha de recursos físicos, garante alta disponibilidade de serviços de dados e infraestrutura técnica adapta-se ao volume de atividade.

### 2.1.10 Desafios

A Computação em Nuvem não é uma revolução tecnológica em si, mas constitui uma orientação para um modo de gerenciamento de infraestruturas de organizações. Todavia, a ideia de hospedar várias aplicações de diferentes usuários traz uns desafios que a Nuvem deve vencer. Trata-se do isolamento, a administração, interoperabilidade e a portabilidade de aplicações entre várias plataformas [15].

[14] Um negócio ou uma empresa que começou recentemente a operar.

#### *2.1.10.1 Isolamento*

A mutualização de recursos na Nuvem (como em toda infraestrutura) envolve a criação de vários mecanismos (segurança, contagem de recursos, conflitos de acesso, etc.) A mais importante delas é a gestão de conflitos / interferências de acesso entre os usuários. Uma solução ideal para o estabelecimento da mutualização é o isolamento. Reagrupam-se sob este termo vários tipos de isolamento, a saber: isolamento de recursos, isolamento do espaço usuários, isolamento de desempenho e isolamento de falhas.

- **Isolamento de recursos**: garante ao cliente a exclusividade de acesso a um conjunto de frações ("blocos") de recursos durante a sua presença na Nuvem (apesar da mutualização) [15]. O cliente tem a ilusão de ser o dono e vê todas as máquinas como um todo. Esse isolamento impede situações de inanição para aplicações cliente (uma situação em que um aplicativo aguarda indefinidamente por um recurso mantido por outro). Além disso, ele permite que o provedor da Nuvem identifique e conte os usos de recursos para cada usuário. Esta contagem será utilizada posteriormente na cobrança.
- **Isolamento do espaço usuários:** dá a cada cliente da Nuvem a ilusão de ser o único usuário. Nada deve deixa-lo perceber a presença de outros usuários ou aplicativos.
- **Isolamento de desempenho:** Permite que a Nuvem garanta que não haja o monopólio de recursos globais de Nuvem por um só cliente.
- **Isolamento de falhas:** permite garantir a não violação de espaços usuários na Nuvem. Ele também apresenta o desafio de segurança. Sendo centro de hospedagem de aplicações multiusuários, a Nuvem deve garantir a integridade de casa espaço usuário *vis-à-vis* dos outros. Assim, nenhuma ação maliciosa realizada por um cliente deve alterar nem o funcionamento da Nuvem, nem o de aplicações pertencentes a outros usuários. Este objetivo é de tanta importância quanto à maioria de usuários da Nuvem são não identificáveis. Refere-se aos usuários de serviços hospedados pelas organizações na Nuvem.

A análise deste desafio foi efetuada com a introdução da técnica virtualização na implementação de Nuvem.

#### *2.1.10.2 Administração*

Como exposto anteriormente, o uso de plataformas de Nuvem requer a intervenção de três tipos de usuários [39]: o administrador, as organizações e os usuários de aplicações corporativas. Os dois primeiros (administrador e organizações) são confrontados diariamente com várias tarefas de administração (a maioria repetitiva). O alívio dessas tarefas afeta a expansão da Nuvem nas organizações. De fato, para evitar a mesma observação feita quanto ao uso das grades (dedicadas aos científicos, ou seja usuários capacitados), as plataformas de Nuvem devem levar em consideração e facilitar as tarefas de administração de seus usuários.

#### *2.1.10.3 Interoperabilidade e Portabilidade*

A interoperabilidade é mais crucial do que nunca. Isso define como aplicativos ou sistemas comunicam e trocam. Este conceito tem sido fundamental para o desenvolvimento de redes de comunicação, como o telefone ou a Internet. Independentemente de qual tecnologia de servidor fornece um serviço de mensagens de um destinatário, ou qual software de e-mail o destinatário usa, a troca é feita através de padrão (por e-mail na Internet, este é o SMTP: *Simples mail Transfer Protocol*).

Os protocolos de comunicação e formatos de dados são padronizados para permitir a implementação de qualquer tecnologia, como com HTTP e HTML. Da perspectiva do usuário, se o *browser* desagrada, pode-se trocá-lo em favor de um que irá funcionar da mesma forma que os padrões da Internet.

No que concerne a Computação em Nuvem, dada à multiplicidade de plataformas Nuvem, os clientes podem ser enfrentados mais tarde, com duas opções: (1) a migração de uma aplicação a partir de uma Nuvem X para uma Nuvem Y e (2) o uso de múltiplas nuvens para hospedagem da mesma aplicação. A escolha (1) surge, por exemplo, quando a competição leva um cliente a partir da Nuvem que hospeda seu aplicativo a outro mais atraente. Isto pode também acontecer caso a plataforma inicial decide cortar seus serviços, o que condiciona o cliente de encontrar outra plataforma podendo acomodar seus aplicativos. Nestes dois casos, surge um problema portabilidade da Nuvem inicial para a Nuvem de destino. Quanto à escolha (2), acontece quando a Nuvem que hospeda o aplicativo encontra-se com falta de recursos. Neste caso, o cliente ou o fornecedor pode decidir de associar à Nuvem inicial recursos provenientes de outra plataforma. Assim, a mesma aplicação se executa em

dois ambientes de Nuvem diferentes pertencendo aos provedores distintos. O conjunto formado pelas duas plataformas constitui uma “Nuvem híbrida” [15]. Esta situação levanta o problema de interoperabilidade entre as plataformas da Computação em Nuvem.

## 2.2 Virtualização

Conforme apresentado na seção 2.1.9.1, o isolamento representa um dos desafios principais da implementação de plataformas de Nuvem. Existem várias formas de implantá-lo:

- O primeiro método consiste em alocar a totalidade do recurso físico a uma organização, mesmo que esta seja subscrita por apenas uma fração desse recurso. Este método permite apenas uma resolução parcial dos problemas de isolamento. De fato, alguns recursos como largura de banda de rede permanecem compartilhados entre as organizações na Nuvem. A menos que o fornecedor aloque exclusivamente para cada organização equipamentos e largura de banda de redes (o que não é razoável), é impossível com este método evitar situações de monopólio de recursos por organização. Deve ser complementada com uma solução de software.
- O segundo método consiste em deixar a responsabilidade para as organizações de implantar mecanismos de isolamento. Este método não é viável uma vez que a Nuvem não tem meios de introspecção de aplicações da organização a fim de garantir a implantação destes mecanismos.
- O último método é intermediário (*hardware* e *software*) aos dois primeiros. Ao implementar os mecanismos de isolamento, ele dá a ilusão de que a organização tenha acesso direto e exclusivo ao recurso físico. Enquanto isso, ele garante à Nuvem não acesso direto das organizações aos recursos físicos. Esta é a virtualização por isolamento.

### 2.2.1 Definição e princípios

A virtualização [19] se define como um conjunto de técnicas físicas e/ou *software* que permitem executar em uma máquina única, vários sistemas operacionais (chamados máquinas virtuais (VM), ou ainda OS convidado). Ela garante a independência e isolamento das VMs (o isolamento conforme apresentado na Seção 2.1.7.1). Em suma, ela permite obter a partir das VM o mesmo isolamento oferecido pelas máquinas reais.

A implementação de um sistema de virtualização é baseada em uma aplicação que se executa (funciona) entre o hardware e as máquinas virtuais: é a "*Virtual Machine Monitor* (VMM)". É ela que implementa os mecanismos de isolamento e de partilha de recursos físicas. A Figura 9 ilustra a arquitetura global de um sistema de uma máquina virtualizada (i.e. executando um sistema de virtualização). A VMM é capaz de iniciar simultaneamente várias máquinas virtuais de diferentes tipos (Windows, Mac ou ainda Linux) em um hardware. Vale lembrar que assim como em um sistema operacional normal, cada VM conserva seu funcionamento habitual. Em outras palavras, ela tem a ilusão de gerenciar os acessos de memórias, de discos, de redes, processador e outros componentes de seus processos. No lado externo, o usuário percebe o conjunto como um ambiente constituído por várias máquinas reais.

A VMM decide as alocações de tempo de processamento para máquinas virtuais. Quanto à comunicação com o lado externo, a VMM pode fornecer várias técnicas para tornar acessível ou não as VM. Ela pode realiza-la através de endereçamento IP e implementação de mecanismos de acesso rede às VM (por roteamento, filtragem de comunicação, etc).

Além de fornecer um sistema de isolamento de sistema operacional, um dos primeiros objetivos da virtualização é o de oferecer os desempenhos próximos daqueles das máquinas reais. Seus objetivos são apresentados na próxima seção. A Figura 8 apresenta uma visão geral de como é feita de alguns sistemas operacionais.

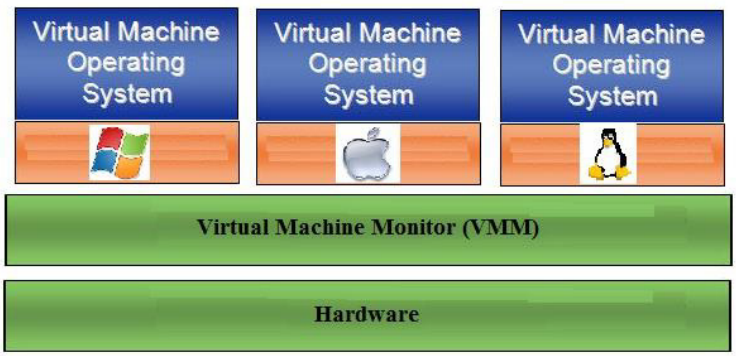



**Figura 8: Vista de sistemas virtualizados.**

### 2.2.2 Objetivos

Em geral, a implementação de um sistema de virtualização deve atender aos seguintes três objetivos [20]:

**Equivalência**: Toda execução de aplicação em um sistema virtualizado deve ser idêntico a um sistema em uma máquina real; com exceção do tempo de execução relacionado à disponibilidade de recursos.
**Eficácia**: a maioria de instruções da VM deve diretamente ser executada pelo processador sem intervenção de software de virtualização.
**Controle de recursos**: o conjunto de recursos é gerenciado de forma exclusiva (conforme mencionada na seção anterior) pelo *software* de virtualização. Isto permite garantir o isolamento de desempenho e de segurança.

### 2.2.3 Classificação

Diante da dificuldade de implementar o modelo de virtualização proposto originalmente (virtualização completo de hardware), devido a não compatibilidade de *drivers*[15] de *hardware*, muitas técnicas de virtualização se desenvolveram. Tendo sido melhoradas no decorrer de anos, as técnicas de implementação de sistemas virtuais podem ser agrupados em diferentes categorias de acordo com a aproximação / distância entre a VM e o *hardware*. Como podemos observar na Figura 9 onde tratamos as principais categorias. O sentido de setas na Figura representa esta evolução cronológica. Ressalta-se que alguns aqui usados podem ter outras descrições dependendo da literatura.

**Virtualização de sistemas de arquivo** (**Figura** 9(b)). É um método que se baseia em um sistema operacional pré-instalado (sistema *host*). Este permite construir espaços usuários (neste caso a VM) cujos sistemas de arquivo são completamente isolados. Cada VM possui um sistema de arquivo de árvore exclusivamente sua [15]. Outros recursos tais que a memória, as placas de rede, o processador são diretamente acessíveis pelas VM. Não existe nenhum isolamento para esses recursos. Nesta categoria, encontramos as ferramentas ***chroot*** ou UML (*User Mode Linux*).

[15] Um software que permite que o computador se comunique com o hardware ou com os dispositivos.

**A emulação** (**Figura** 9(a)). Uma vez que a categoria precedente não permite a instalação d'OS, foi desenvolvida uma tecnologia baseada na emulação. Esta técnica fornece uma aplicação (virtualizador), que é executado acima do sistema host em espaço do usuário. Esta aplicação (que corresponde à VMM) tem como missão emular o hardware e permite iniciar vários sistemas operacionais reais no espaço do usuário. Esta técnica de virtualização induz uma sobrecarga significativa em máquinas virtuais em execução [15]. Na verdade, cada operação na VM é interceptada e interpretada pela VMM. O sistema de virtualização VirtualBox [21] baseia-se nesta técnica.

**Para-virtualização** (**Figura** 9(c)). Esta técnica foi desenvolvida para resolver os problemas da categoria anterior. Trata-se de modificar os Os das VM para que eles estejam cientes de seu estado (de VM). Dessa forma, o hardware é mapeado para a VM e, portanto, acessível diretamente sob o controlo do VMM (*hypervisor* na Figura 9 (c)). Esta se executa diretamente acima do hardware. Ela substitui o sistema operacional host, que é considerada uma VM. A restrição de modificação d'OS das VM é um limite para esta técnica. Na verdade, ela não permite a execução de VM equipada com sistema operacional proprietário (como o Windows). A plataforma Xen [21] é o sistema de Paravirtualização mais expandido graças ao nível de desempenho que fornece [24].

**Virtualização assistido pelo hardware** (**Figura** 9(d)). A evolução hodierna de *drivers* de *hardware* e processadores (Intel VT tecnologia [Corporação]) para levar em conta a virtualização permite o desenvolvimento de uma nova técnica de virtualização. Assim, a intervenção da VMM sobre as ações das VM é limitada e estas não precisam ser modificadas. Novas implementações de Xen ou VMware [VMware.org] permitem usar esta técnica quando o **hardware** suporta.

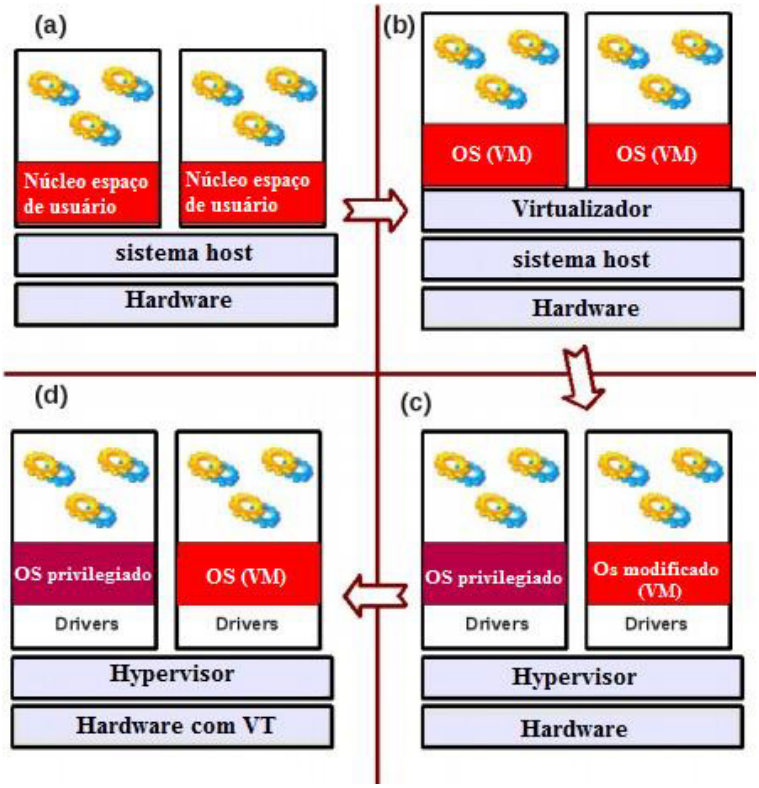



**Figura 9: Técnica de virtualização [15].**

Depois de abordar esta parte da virtualização, constata-se que ela responde perfeitamente ao desafio de isolamento com o qual a Nuvem se depara. Em particular, ela amplifica a prática da mutualização de recursos, que é o coração da Nuvem. A sua introdução na Nuvem envolve a modificação do modo de gerenciamento de recursos e tarefas administrativas em geral. No que se refere ao gerenciamento de recursos, a unidade de alocação na Nuvem vai da máquina real à máquina virtual.

## 2.3 Segurança em Computação em Nuvem

A segurança permite garantir a confidencialidade, a integridade, a autenticidade e a disponibilidade de informações. A evolução de tecnologias relacionadas e a normalização destas colocam à nossa disposição um conjunto de algoritmos e protocolos para tratar dessas questões.

### 2.3.1 A confidencialidade

A Confidencialidade garante que os dados de um cliente possam ser acessados apenas por pessoas autorizadas. Diferentes soluções de Computação em Nuvem incluem mecanismos de privacidade, como gerenciamento de identidade e acesso, isolamento ou criptografia.

Os controles de acesso mais seguros não são nenhuma proteção contra um atacante que tenha acesso a credenciais ou chaves. Assim, as informações de credenciais ou gerenciamento de chaves são elos fundamentais na concepção de segurança.

A maioria de trocas internas ou externas são encapsuladas em SSL (*Secure Sockets Layer*) e autenticadas com um certificado gerado pelo cliente. Este certificado não está ligada a qualquer autoridade de certificação raiz confiável, mas auto assinado pelo próprio cliente. Como ele assume o controle da chave privada, o mecanismo permite um alto grau de certeza: somente os clientes autorizados e tendo esta chave podem acessar aspectos específicos do serviço.

Outro fator existente é que a Computação em Nuvem é baseada na virtualização. Um ponto crítico é o isolamento da máquina virtual com as outras máquinas virtuais, bem como o isolamento das máquinas virtuais *vis-à-vis* do *hypervisor*. Hoje, os provedores de Computação em Nuvem se baseiam nas tecnologias Microsoft e VMware, garantino um alto isolamento de sistemas convidados graças à forte experiência de produtos de virtualização dos seguintes editores (*Virtual* PC, *Virtual Server* e, mais recentemente, Microsoft Hyper-V, VMware *Workstation*, ESX, ESXi da VMware).

Além disso, as tecnologias de Computação em Nuvem integram analisadores de pacotes de rede assegurando que as máquinas virtuais não confiáveis não podem gerar tráfego falsificado, ou receber tráfego a eles não destinados, ou redirecionar o tráfego protegido para outros pontos de terminal e nem podem enviar ou receber tráfego em "*broadcast*[16]" para todas

[16] Uma forma de envio de sinal.

as máquinas da Nuvem. Mesmo as conexões entre diferentes instâncias de um aplicativo dentro da Nuvem são consideradas como conexões de Internet, portanto, inseguras.

Finalmente, a criptografia de dados para armazenamento e transmissão se alinha sobre as melhores práticas para garantir a confidencialidade e integridade de dados. Como mencionado acima, as comunicações internas são protegidas com SSL. Dependendo do cliente, as plataformas de desenvolvimento oferecem bibliotecas que permitem aos desenvolvedores integrar protocolos e algoritmos de criptografia em suas aplicações.

### 2.3.2 A integridade

Os clientes que procuram terceirizar seus dados podem, obviamente, esperar serem protegidos contra alterações não autorizadas. Os sistemas nas nuvens fornecem uma série de mecanismos para proteger a integridade dos dados.

### 2.3.3 A disponibilidade

Uma das principais vantagens proporcionadas pelas plataformas de Computação em Nuvem baseia-se na disponibilidade de redundância robusta alcançada com tecnologias de virtualização. Windows Azure [25], por exemplo, oferece vários níveis de redundância proporcionando uma disponibilidade máxima de dados e aplicativos. Os dados são replicados em três nós distintos do Windows Azure para minimizar o impacto de falhas de *hardware*. Além disso, os clientes podem criar funções personalizadas para replicar e sincronizar dados entre instalações da Microsoft. Eles também podem criar funções personalizadas para escrever dados de armazenamento para backups em um local privado.

Os agentes rodando em máquinas virtuais convidadas monitoram o estado da referida máquina. Se o agente não responde, o controlador reinicia a máquina virtual. Os clientes poderão optar por executar os processos de vigilância mais sofisticados do estado, e adaptados à sua política de continuidade. Em caso de falha do equipamento, o controlador desloca a instância da função para um novo nó e reprograma a configuração de rede para instâncias desta função, a fim de restaurar a disponibilidade total de serviço.

Os controladores aderem ao mesmo princípio da disponibilidade através de redundância e balanço automático para garantir a disponibilidade contínua de capacidade de gerenciamento de controladores.

## 2.4 Administração de Nuvem

Concebidas como uma evolução de plataformas partilhadas, as nuvens requerem tarefas de administração encontradas em grades e ambientes distribuídos em geral. A estes se acrescentam as relacionadas com a introdução de virtualização. Dizem respeito a diferentes protagonistas na Nuvem. Como já mostrado na Seção 2.1.6.2, distinguimos três tipos de usuários na Nuvem (Figura 10): o fornecedor (administra a Nuvem), as organizações clientes (usam os recursos da Nuvem e gerenciam suas aplicações) e usuários finais (aqueles que utilizam os serviços prestados pelos aplicativos corporativos). Ao contrário dos dois primeiros usuários, este último não está confronta nenhuma tarefa de administração. Em geral, os dois gerenciadores dos dois níveis de administração realizam as tarefas de [39]:

- **Implantação**: a implantação de VM a nível IaaS e implantação de softwares para a organização cliente;
- **Monitoramento**: monitoramento de recursos materiais e VM a nível IaaS (fornecido pelo elemento "*MonitoringController*" na Figura 11) e monitoramento de software ao nível organização cliente (fornecido pelo elemento "*AppMonitoringController*" na Figura 11);
- **Gerenciamento de recursos**: alocação eficaz de recursos físico a nível IaaS (fornecido pelo elemento "*ResourceController*" na Figura 11) e reserva eficaz de VM ao nível organização cliente (fornecido pelo elemento "*AppResourceController*" na Figura 11);
- **Reconfiguração**: reconfiguração dos VM ao nível da IaaS (fornecido pelo elemento "*AppMonitoringController*" na Figura 11) e reconfiguração de softwares ao nível organização cliente (fornecido pelo elemento "*AppScheduler*" na Figura 11).

Nesta Seção, apresenta-se em detalhes e separadamente por um lado a administração como realizada na IaaS e por outro lado as operações administrativas realizadas pela organização.

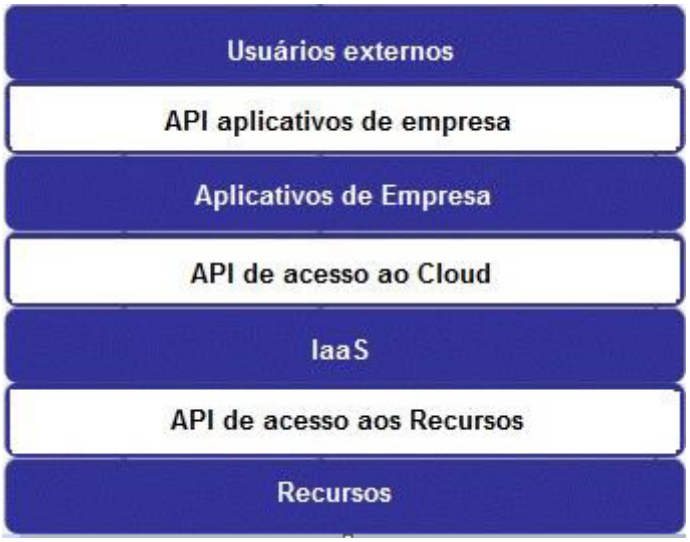



**Figura 10: Arquitetura simplificada da Nuvem.**

### 2.4.1 Administração em nível de IaaS

Diretamente relacionado com o ambiente físico, a administração em nível de IaaS se resume ao gerenciamento de máquinas virtuais (para o uso eficiente de recursos) e de seus servidores de gerenciamento (*scheduler*, servidores de armazenamento, etc.). Não podendo ser realizadas com antecedência, certas tarefas de administração de IaaS são causadas pelas tarefas de organizações (que serão interpretadas como operações de reconfiguração). Entre as operações de administração a nível IaaS, as mais comuns são: a alocação de recursos, implantação de servidores e máquinas virtuais, a configuração e a inicialização ( das VM e seus servidores). Quanto às outras, elas são causados por eventos externo para algumas (consolidação, reparação) e regular para outras (monitoramento).

#### *2.4.1.1 Alocação de recursos*

É a primeira operação realizada na Nuvem [15]. Ela atribui à organização cliente sua porção de recursos exploráveis. Ao se falar de recursos, agrupa-se ao mesmo tempo a memória, o processador, o espaço de armazenamento, a largura de banda e os equipamentos computacionais. Mutualizados entre todos os usuários, os recursos representam o ponto de rentabilidade para o provedor. Desta forma, a elaboração e implementação de políticas de alocação na Nuvem dependem da estratégia de negócio do provedor.

Note-se que a alocação é causada entre outras por uma reserva da organização. O provedor pode, portanto, oferecer várias formas de reserva:

1. Reserva para um tempo indeterminado: neste modo, um contrato é estabelecido com a organização por um período infinito e continuo.
2. Reserva para uso futuro e por um tempo limitado: neste modo, a dificuldade reside no gerenciamento de planos de reserva. Encontra-se o problema bem conhecido e complexo que é o planejamento.
3. Reserva para uso imediato e limitado no tempo: neste caso, os recursos requeridos devem estar imediatamente disponíveis.

A consideração desses modos de reserva pode tornar complexa a alocação na Nuvem. Em particular, pode ser levada a usar filas de espera, com o conceito de prioridade. Assim, como exemplo, os recursos obtidos pelo último modo de reserva podem ser considerados com prioridade menor do que os obtidos através dos dois primeiros. Neste caso, a Nuvem deve ser capaz de identificar os recursos a disponibilizar caso houver a necessidade de uma aplicação de prioridade maior.

#### *2.4.1.2 Implantação*

A implantação na IaaS concerne principalmente os sistemas de arquivos das máquinas virtuais. É realizada em duas ocasiões. Primeiramente (seta (2) da Figura 11) durante o armazenamento na Nuvem dos sistemas de arquivos d'OS (também chamados imagens) e dos dados da organização. Para fazer isso, a Nuvem utiliza um servidor de armazenamento, chamado *RepositoryController* (Figura 11), distinto de lugares de execução das VM. A segunda implantação (seta (4) da Figura 11) ocorre na execução desta. Na verdade, a imagem utilizada para a execução de uma VM na Nuvem é uma cópia da imagem original. Isto é devido a duas razões. A primeira é a possibilidade de inacessibilidade de máquinas host (hospedando as VM) ao servidor de armazenamento. Muitas vezes, por razões de eficiência, o formato de armazenamento dos dados (incluindo sistemas de arquivos) pelo servidor de armazenamento pode ser diferente do esperado pelo sistema de virtualização que irá executar a VM. A segunda razão é a possibilidade de utilização da mesma imagem para executar várias VM. Além disso, a organização pode querer recuperar a imagem original após a execução da VM.

#### *2.4.1.3 ConFiguração e Inicialização*

A operação de inicialização de máquinas virtuais exige a sua configuração prévia. Este último depende tanto das demandas da organização e também da política de alocação de recursos na Nuvem. Durante a fase de reserva, uma organização subscreve para VM cujas características ela fornece à Nuvem. Essas características incluem: a quantidade de memória, número de processadores, o lugar geométrico da execução da VM e o sistema de arquivos representando o OS da VM. Quanto às restrições de configuração de IaaS , isto concerne as configurações de rede. De fato, a IaaS pode implementar várias configurações de acesso a redes : a alocação de uma rede virtual (VLAN) para VM na mesma organização ou o uso de uma VLAN comum para todas as VM (isto é apenas exemplos). Ele também deve configurar os controles de rede (*firewall*[17] ou quotas de uso de rede) com base nas assinaturas da organização.

#### *2.4.1.4 Reconfiguração*

Como mencionado no preâmbulo desta seção, a maioria das tarefas de administração em nível da IaaS podem ser interpretadas como operações de reconfiguração (realizadas pelo "*VMController*" na Figura 11). Na verdade, eles operam durante a execução da IaaS e altera sua composição. Nesta seção, apresentamos algumas tarefas de reconfigurações específicas, principalmente relacionadas com o gerenciamento de VM: a consolidação e a reparação [15].

**A Consolidação de VM**

Lembrando-se de plataformas de hospedagem onde as máquinas como todo eram dedicadas a um cliente, nessas plataformas, nenhuma tarefa de administração da parte do provedor não era possível uma vez a máquina atribuída ao cliente (com o risco de violar o uso exclusivo da máquina pelo cliente). Em contraste, as nuvens baseadas em virtualização fornecem uma margem ao administrador da IaaS para a intervenção em máquinas físicas. Esta possibilidade é oferecida pela natureza isolante de virtualização. Ela permite, entre outros, ao provedor implementar diferente políticas de alocação ou redistribuição de recursos, a fim de lucrar ou atender a um contrato de cliente.

[17] Uma barreira de proteção que ajuda a bloquear o acesso de conteúdo malicioso.

**A Reparação de falhas**

Dado o tamanho da Nuvem e da multidão de usuários e o número de clientes que recebe, o risco de ocorrência de falhas é importante. A ocorrência de uma disfunção deve ser rapidamente identificada e tratada pelo administrador a fim de evitar a penalização das organizações.

Uma das falhas mais críticas nas IaaS é a falha de uma máquina física ou virtual. Na verdade, isso afeta aplicativos de negócios. Devido à sua não intromissão, IaaS não tem conhecimento do software em execução em VMs que hospeda. Portanto, a IaaS não pode efetivamente resolver uma falha sem a cooperação da organização em causa.

Apesar desta limitação, IaaS pode desfrutar dos benefícios da virtualização e , assim, oferecer várias políticas de reparação. Estes podem variar desde o mais simples (reiniciar a VM em questão) aos mais sofisticados (reiniciar o último ponto de verificação) . A aplicação dessas políticas pode depender dos termos do contrato assinado pela organização. Em todos os casos, a implementação dessas políticas depende do sistema de monitoramento instalado nas IaaS.

#### *2.4.1.5 Monitoramento*

As seções anteriores introduziram o conceito de monitoramento. Na verdade, todas as tarefas de administração na Nuvem dependem da informação obtida pelo monitoramento de ambientes de hardware, virtuais e softwares (feito por "*MonitoringController*" na Figura 11). Entre as informações de monitoramento que interessadas, pode-de citar a taxa de uso da CPU, de discos, da rede, da memória, etc. Todavia, a arquitetura específica da Nuvem e de aplicações que são executadas constitui um fator limitante para o monitoramento. De fato, reconhecido como uma tarefa complexa em ambientes distribuídos formados por máquinas reais, o monitoramento na Nuvem tornou-se um verdadeiro desafio.

Uma vez que os recursos são virtualizados na Nuvem, não é evidente fornecer imagem que possa refletir as máquinas de estado reais. Na verdade, podem-se distinguir dois níveis de observação e análise dos recursos: a máquina virtual e a máquina física. Por um lado, a IaaS deve ser capaz de separar os recursos consumidos por cada nível , a fim de fornecer aos clientes informações sobre apenas o seu consumo. Por outro lado, a IaaS deve

fornecer as informações do administrador sobre tanto a VM como a máquina host na qual se executa. Em ambos os casos, a informação deve ser compreensível e utilizável.

Até então, fala-se de informações locais para cada máquina física ou virtual. Mas a Nuvem é um sistema distribuído e heterogêneo. Na maioria dos casos, o administrador está interessado em informações agregadas obtidas e cálculos agrupados por observações locais. Por exemplo, este cálculo pode ser feito por combinação de informações vinda de um conjunto de máquinas pertencentes a mesma área geográfica ou na mesma organização.

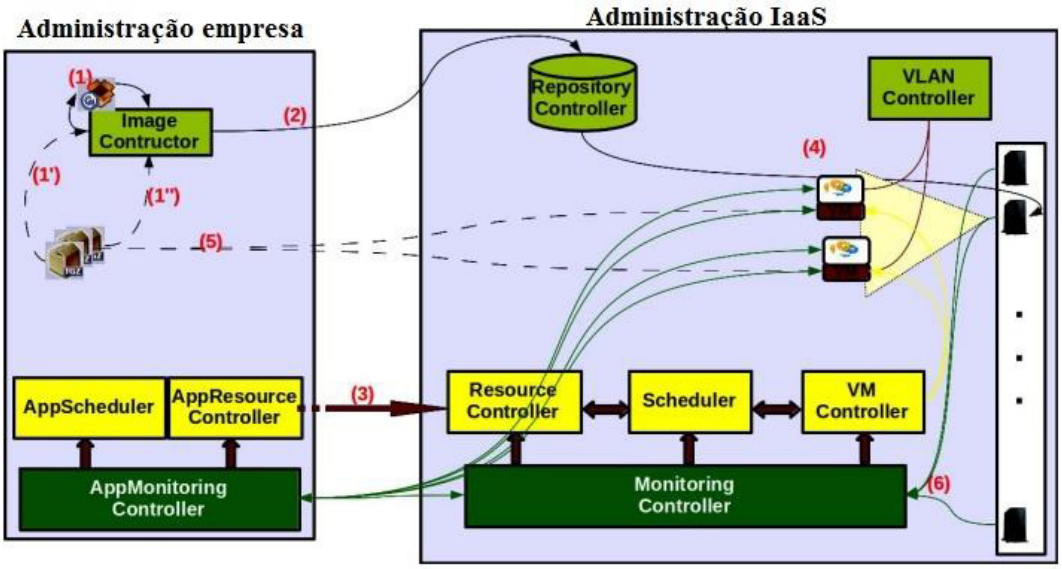



**Figura 11: Administração na Nuvem [15].**

### 2.4.2 Administração a nível Organização

A ideia de as organizações confiarem seus dados (essencial para a concorrência) para uma empresa externa ainda não está totalmente aceita nas organizações. Diante dessa ideia, a Nuvem responde com mais desenvolvimento de segurança e confidencialidade. Como internamente a administração de aplicativos na Nuvem é uma tarefa tediosa para a empresa. Além disso, no que tange as Nuvens virtualizadas, o gerenciamento das VM é uma operação adicional. Em suma, a administração de aplicativos em uma Nuvem virtualizada inclui as seguintes tarefas [15]:

- Construção de sistemas operacional (ou VM);
- Reserva/Alocação de recursos na Nuvem;
- Instalação e inicialização de softwares;
- Acompanhamento da execução de softwares e das VM;
- Reconfiguração/Otimização (Escalabilidade, atualização de *softwares*, reparação, etc.).

#### *2.4.2.1 Construção de VM e Implantação*

A execução de qualquer aplicativo de uma organização na Nuvem acontece em uma VM. A implementação deste último exige a presença de sua imagem no seu servidor de armazenamento na Nuvem. Para a organização, a implantação pode ocorrer em duas ocasiões: durante o download do sistema de arquivos das VM (seta (2) na Figura 11) e durante a cópia (da organização para a VM) (seta (5) na Figura 11).

#### *2.4.2.2 Alocação de recursos*

A alocação de recursos para a organização pode constituir na reserva de local de armazenamento ou ainda de máquinas. A primeira permite armazenar as imagens do sistema operacional construído ou dados utilizáveis em seguida pelas VM. Lembrando-se que estas não são vistas pela organização como VM. Para ela, são máquinas físicas disponibilizadas pela Nuvem e que possui. Isso é feito na forma de contratos com a Nuvem. A empresa assina vários recursos de capacidades idênticas ou não (em termos de CPU, memória, a largura de banda, etc.). Esta reserva é traduzida em nível da IaaS pela inicialização das VM

correspondentes e pela transmissão de informações de conexão da organização (seta (3) na Figura 11). O fim do contrato envolve a interrupção das VM e a liberação dos recursos que lhes foram atribuídas.

#### *2.4.2.3 Configuração e inicialização*

Uma vez que a VM inicializada, o administrador da organização planeja a configuração e a inicialização de software. Estas operações requerem acessos múltiplos às VM através do endereço IP ou DNS da VM. Eles dependem do software fornecido. Em alguns casos, a inicialização pode requerer a configuração de laços de comunicação entre pode requerer ligações de comunicação entre softwares localizados em VMs diferentes. Devido ao grande número de softwares, estas operações muitas vezes são fontes de erros.

#### *2.4.2.4 ReconFiguração*

Assim como na IaaS, inúmeras são as possíveis operações de reconfiguração pela organização (realizadas por "*Scheduler* de aplicações" na Figura 11). Entretanto, nesta seção apresentar-se-á apenas duas delas, mas muito frequentes na administração de aplicações distribuídas:

**Escalabilidade**

Uma vez a aplicação iniciada, a organização deve ser capaz de acompanhar o estado de diferentes softwares a fim de intervir caso necessário. Esta prática é comumente chamada de ***princípio de crescimento à demanda***.

**Reparação**

Assim como na parte interna, dois tipos de falhas podem surgir durante a execução da aplicação:

1. Uma falha de máquina, no caso VM;
2. Uma falha de software.

Não tendo nenhum plano de reparação nas suas VM, a organização pode assinar por um contrato de reparação no caso (1). Neste caso, a organização pode ser levada a reimplantar e iniciar os *softwares* que se executavam na VM antes da falha. Se a IaaS implementa o backup regular das VM, a organização não terá nenhuma reparação a efetuar.

Quanto à falha do tipo (2), suas reparações dependem de aplicação. Em certas aplicações, uma falha do tipo (1) ou (2) pode causar a reconfiguração inteira da aplicação.

## 2.5 Visão geral de tolerância a falhas

A tolerância a falhas é a capacidade de um sistema de computacional de cumprir a sua função, apesar da presença ou a ocorrência de falhas [26], quer se tratem de danos físicos de hardware, falhas de softwares, ataques maliciosos, os erros de interação homem-máquina. Ela aparenta uma forma garantir a segurança de funcionamento.

A primeira seção apresenta os conceitos gerais de segurança de funcionamento. A segunda seção aborda os diferentes métodos de tolerância a falhas de uma maneira muito geral. A terceira Seção apresenta o modelo de aplicação distribuída e apresenta técnicas de tolerância a falhas específicas para esta área.

### 2.5.1 Segurança de funcionamento e tolerância a falhas

A fiabilidade de funcionamento de um sistema de computacional é a sua capacidade de oferecer um serviço em que se pode ter uma confiança justificada. [27] A fiabilidade de funcionamento pode ser apresentada em torno de três conceitos descritos na Figura 12 extraído de [26]: os seus atributos, barreiras e meios.

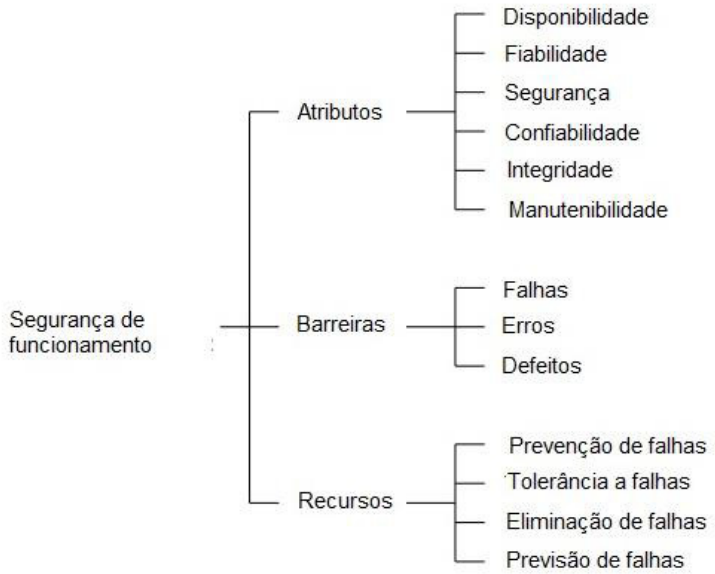



**Figura 12: A arvore de fiabilidade de funcionamento [26]**

Os atributos da fiabilidade de funcionamento correspondem às propriedades que deve verificar um sistema. Estes atributos são usados para avaliar a qualidade do serviço fornecido pelo sistema. Seis atributos de segurança são definidos em [27]:

- **Disponibilidade** é o fato de ser pronto para utilização;
- **Confiabilidade** corresponde à continuidade do serviço;
- **Segurança (*safety*)** é a ausência de consequências catastróficas causadas por falhas;
- **Confidencialidade** corresponde à ausência de divulgação não autorizada da informação;
- **Integridade** corresponde à ausência de alterações inapropriadas da informação;
- **Manutenibilidade** corresponde à capacidade relaciona as reparações e evoluções.

A importância de cada atributo pode ser diferente, dependendo da aplicação e das necessidades do sistema computacional a que se destina. Para aplicações paralelas a longo prazo, os principais atributos serão a fiabilidade e manutenibilidade.

#### *2.5.1.1 Barreiras à segurança de funcionamento*

Barreiras à segurança são de três tipos: [26] falhas, erros e defeitos. Falhas, erros e defeitos são ligados por relações causais mostrado na Figura 13.

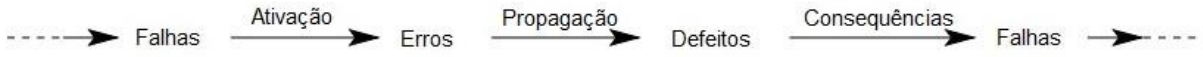



**Figura 13: A cadeia fundamental das barreiras à segurança de funcionamento [26]: As setas indicam as relações de causalidade entre falhas, erros e defeitos.**

Um defeito é o evento que ocorre quando o comportamento do sistema se desvie da sua função. O erro é a parte do estado do sistema, que é susceptível de provocar um defeito. O defeito ocorre quando o erro afeita o serviço fornecido ao usuário. A falha é definida como sendo a causa atribuída ou suposta do erro.

As falhas são classificadas de acordo com oito critérios: fase de criação ou de ocorrência, os limites do sistema, causa fenomenológica, dimensão, intenção, capacidade e persistência. A combinação relevante desses critérios dá uma classificação abrangente de todos os tipos de falhas. Por questões de simplicidade, podem ser agrupadas em três

categorias, não exclusivas:

- As falhas de desenvolvimento que trazem erros que podem ocorrer durante o desenvolvimento;
- As falhas físicas que trazem as falhas que afetam o hardware;
- As falhas de interações que trazem falhas externas, isto é, aquelas que se situam fora dos limites do sistema e que propagam erros dentro do sistema por interação de interferência.

#### *2.5.1.2 Meios de garantir a segurança (security) de funcionamento*

Os meios para garantir a segurança de funcionamento são definidos como métodos usados para garantir essa propriedade. Quatro métodos principais [27] podem ser distinguidos.

- A **prevenção de falhas** visa prevenir a ocorrência ou introdução de falhas no sistema. Baseia-se em regras de desenvolvimento (modularização, utilização de linguagem fortemente tipada, prova formal, etc.).
- A **eliminação de falhas** procura reduzir a presença (número, gravidade) de falhas. Este método funciona tanto no desenvolvimento (confira condições, teste de regressões, injeção de falhas, etc.) ou durante a utilização (Manutenção).
- A **previsão de falhas** procura estimar (qualitativamente e quantitativamente) a ocorrência e as consequências das faltas. Ela é realizada por modelagem e a avaliação de sistemas.
- A **tolerância a falhas** tenta mascarar a ocorrência de falhas e continuar a fornecer o serviço requerido apesar de sua aparição. Na próxima seção apresentar-se-á diferentes abordagens para alcançar a tolerância a falhas.

### 2.5.2 Tolerância a falhas

O objetivo da tolerância a falhas é evitar defeitos do sistema, apesar da presença de falhas. Isto equivale a quebrar a cadeia descrita na Figura 13, que leva da falha ao defeito. A tolerância a falhas é implementada pela detecção de erros e recuperação do sistema.

#### *2.5.2.1 Detecção de erro*

A detecção de erros pode ser realizada durante uma suspensão de serviço. Diz-se então que ela é preventiva. Em contrapartida, diz-se que ela é concomitante quando ela é realizada durante a execução normal do serviço.

Técnicas de detecção concomitante utilizam a redundância em nível de informação ou componente, ou a redundância temporal ou algorítmica. As formas mais são usadas são as seguintes.

- O código de detecção de erro: eles introduzem redundância na representação de informação [28].
- A duplicidade e a comparação: as unidades de processamento são duplicadas e os seus resultados são comparados.
- Os controles e execução temporal: um controle de "cão de guarda" (*watchdog*[18]) controla o tempo de resposta ou o progresso da execução.
- Os controles de plausibilidade ou de dados estruturados: afirmações são inseridas no código para verificar tipos, índices, valores, etc.

#### *2.5.2.2 Recuperação do sistema*

A recuperação do sistema visa transformar um estado errôneo em estado livre de erro e falha. O tratamento da falha se faz identificando o componente em falho e excluindo-o. O tratamento de erro pode ser feito por três técnicas: a recuperação, a continuidade e a compensação [26].

A **recuperação** é a técnica mais comumente utilizada. O estado do sistema é guardado regularmente. Quando um erro é detectado, o sistema é trago de volta a um estado antes da ocorrência do erro. Este estado salvo é chamado de pontos de restauração.

A **continuidade** consiste em buscar um novo estado livre de erros. Isto pode, por exemplo, ser realizado através da combinação de um tratamento especial, quando um erro é detectado. O objetivo deste tratamento é corrigir o estado errado.

A **compensação** exige que o estado do sistema tenha redundância suficiente para permitir a

[18] Um temporizador que dispara um reset ao sistema se o programa principal, devido a alguma condição de erro.

sua transformação em um estado livre de erro. É transparente vis-à-vis da aplicação, pois não necessita de executar novamente uma parte da aplicação (recuperação), nem um procedimento específico (continuação). Ele pode, por exemplo, ser realizado através da replicação de componentes e efetuando uma votação por maioria dos resultados. Outra abordagem é a utilização de códigos de correção de erro [28] ou em geral algoritmos tolerantes a falhas [30, 29, 31, 32].

Pode notar-se que o método de compensação não necessita de detecção de erros específico já que ela mesma efetua a detecção de erro. Um método de compensação pode servir como o detector de erro, enquanto que o inverso não é verdadeiro. De fato, a compensação requer maior redundância para corrigir o erro [13]. Por exemplo, em termos de componentes, dois componentes são suficientes para detectar um erro, mas pelo menos três serão necessárias para corrigir o problema.

No restante desta seção, vai-se concentrar na tolerância a falhas numa Computação em Nuvem.

## 2.6 Tolerância a falhas para sistemas de Computação em Nuvem

Devido a certos empecilhos de manter uma tecnologia de infraestrutura interna e os custos crescentes a ela associados, a maior partes das organizações está cada vez mais exteriorizando seus serviços de TI que são, portanto, gerenciados por organizações especializadas (chamados, na literatura, de provedores). Este processo levou ao surgimento da chamada abordagem de Computação em Nuvem.

Anteriormente foi apresentado um fundamento teórico sobre a Computação em Nuvem. Como síntese, a Computação em Nuvem é uma solução abrangente para o fornecimento de TI como um serviço. É uma solução de computação baseada na Internet, onde os recursos compartilhados são fornecidos como energia elétrica. Computação em Nuvem é uma forma de processamento e/ou tratamento onde o serviço é prestado através da Internet utilizando-se modelos e níveis de abstração [35].

Encontram-se muitas pesquisas abordando a Nuvem, tais como tolerância a falhas, segurança, etc. A tolerância a falhas é uma importante questão chave na Nuvem, que se preocupa com todas as técnicas necessárias para permitir que um sistema tolere falhas inesperadas de software ou de hardware. Para alcançar robustez e confiabilidade na

Computação em Nuvem, a falha deve ser avaliada e tratada de forma eficaz. Os principais benefícios da implementação de tolerância a falhas em Computação em Nuvem incluem a recuperação de falhas, menor custo, melhor desempenho, etc. [36].

Quando várias instâncias de um aplicativo estão executando em várias máquinas virtuais e um dos servidores cair, pode-se depreender que existe ali uma falha e faz-se necessário ser implementado um mecanismo tolerante a falhas.

De acordo com [37], uma tolerância a falha numa Computação em Nuvem é uma instalação ou configuração que impede que um computador ou dispositivo de rede venha a falhar no caso de um erro ou problema inesperado. Para tornar um computador ou rede tolerante a falhas, isto requer que o usuário ou a organização preveja como um computador ou dispositivo de rede pode falhar e tomar as medidas que ajudam a evitar esse tipo de falha. O caminho de geração de erro é ilustrado na Figura 14 da seção anterior. Neste contexto, cumpre mencionar que:

- Um sistema é dito falho quando ele não cumpre suas exigências.
- Um erro é parte do estado do sistema, que pode levar a uma falha.
- A causa de um erro é uma falha [38]. Transitórios, falhas intermitentes ou permanentes, falhas de projeto ou falhas operacionais.

### 2.6.1 Tipos de Tolerância a falhas

Há várias falhas que podem ocorrer em Computação em Nuvem. Com base nas políticas de tolerância a falhas podem ser usadas várias técnicas tolerantes a falhas que podem ser tanto em nível de tarefa ou em nível de fluxo de trabalho. Com base nas políticas e técnicas de tolerância a falhas pode-se classificar esta técnica em dois tipos: proativas e reativas.

#### *2.6.1.1 Tolerância a falhas reativa*

Políticas de tolerância a falhas reativas reduzem o efeito de falhas na execução da aplicação quando a falha ocorre de forma eficaz. Encontram-se algumas técnicas que são baseadas nessas políticas, como por exemplo, *Checkpoint/Restart*, *Replay* e *Retry* , etc, detalhadas no Capítulo 3.

#### *2.6.1.2 Tolerância a Falhas Proative*

O princípio das políticas de tolerância a falhas proativas é evitar a ocorrência de falhas, erros e defeitos, prevendo-os e proativamente substituir os componentes suspeitos por outros componentes de trabalho. Umas das técnicas baseadas nessas políticas são *Preemptive migration*, *Software Rejuvenation*, etc, também encontradas no Capítulo 3.

Estes podem ser classificados em duas subtécnicas de processamento de erros e tratamento de falhas. Processamento de erro visa eliminar erros do estado computacional. Tratamento de falhas visa prevenir falhas de ser reativado [46] [47].

### 2.6.2 Desafios da Implementação de Tolerância a Falhas em Computação em Nuvem

Para fornecer a tolerância a falhas, é preciso uma consideração e análise cuidadosa devido à sua complexidade, interdependência e pelas seguintes razões:

- Existe a necessidade de implementar a técnica de tolerância a falhas autônomo para várias instâncias de um aplicativo em execução em várias máquinas virtuais [40].
- Diferentes tecnologias de fornecedores concorrentes de infraestrutura de Nuvem precisam ser integradas para o estabelecimento de um sistema confiável [41].
- A nova abordagem tem de ser desenvolvida para integrar essas técnicas de tolerância a com algoritmos de programação de fluxo de trabalho falhas existentes [42].
- Um método baseado em referência pode ser desenvolvido em ambiente de Nuvem para avaliar o desempenho dos componentes de tolerância a falhas em comparação com os similares [43].
- Para garantir a alta confiabilidade e disponibilidade de vários provedores de computação em nuvens com pilhas de software independentes deveriam ser usados [44] [45].
- Tolerância a falhas autônoma deve reagir a sincronização entre várias nuvens [41].

# 3 SISTEMAS TOLERANTES A FALHA: ABORDAGENS EXISTENTES

Conforme ilustrado na Figura 14, as três camadas são identificáveis em uma plataforma de Nuvem: recursos, VMs e aplicações. Cada um deles se preocupa com as falhas. Portanto, identificam-se três tipos de falha em uma plataforma de Nuvem: falha de *hardware*, falha de VM e falha de aplicação. A estratégia tolerância a falhas (TF) inclui duas fases distintas: fase de detecção e fase de reparo.


**Figura 14: da Computação em Nuvem.**

As técnicas de tolerância a falhas são usadas para superar os defeitos de forma que o sistema funcione sem problemas durante todo o processo. Há inúmeros métodos para lidar com falhas que são métodos de redundância, métodos de recuperação e balanceamento de carga [54].

A redundância de hardware é uma técnica de redundância estrutural que mascara completamente falhas dentro de um conjunto de módulos redundantes. Neste caso, um número de módulos idênticos executa as mesmas funções (Figura 15), e as suas saídas são votadas para remover erros criados por um módulo defeituoso [52]. Redundância Modular Tripla (TMR) é um comumente usado a partir de mascaramento de falhas em que o circuito é triplicado e considerado conforme ilustrado na Figura 15. O circuito de votação também pode ser triplicado para que falhas de eleitores individuais também possam ser corrigidas pelo processo de votação. Um sistema TMR falha sempre que dois módulos em um trio redundante criam erros de sorte que o voto não é mais válido.

Por sua vez, o método de redundância de software utiliza dois programas ou algoritmos diferentes, além do que é requerido para executar uma determinada função, para detectar e possivelmente tolerar falhas. Neste processo, ela produz de forma independente duas ou mais versões do software, na esperança de que as diferentes versões não falharão com a mesma entrada. Estas formas diversificadas de projeto garantirão que nem todas as cópias falharão no mesmo conjunto de dados de entrada [53].

Além disso, temos o balanceamento de carga, esse é basicamente o fato de trabalhar em modelagem de tráfego, ele pode gerenciar tráfego de entrada ou saída em qualquer.

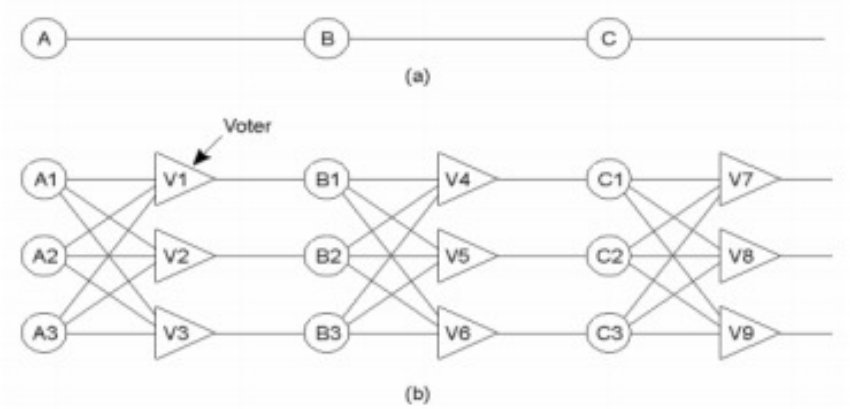



**Figura 15: Triple Modular Redundancy [54].**

Podem-se encontrar várias técnicas tolerantes a falhas em [46] [48] [49] [50] [51]:

## 3.1 Técnicas tolerantes a falhas Na Computação em Nuvem

Existem várias falhas que podem ocorrer em Computação em Nuvem. Com base nas políticas de tolerância a falhas podem ser usadas várias técnicas de tolerância a falhas que podem ser tanto nível de tarefa ou nível de fluxo de trabalho.

### *3.1.1 Check Pointing*

*Check pointing* dá a uma aplicação, ou um sistema, a possibilidade de salvar seu estado, e tolerar falhas, permitindo que o estado falhado se recupere a partir de um estado seguro mais recente. Nesse cenário, depois de fazer todas as alterações no sistema, um ponto de restauração é marcado. Quando uma tarefa falha, é permitida ser reiniciada a partir do estado recentemente marcado, em vez de reiniciar desde o início. É um nível de tarefa de técnica de tolerância a falha eficiente para a execução de aplicações maiores e de longa duração [50].

Ideias chaves conforme a Figura 16:

- Salva o estado executivo;

- Fornece mecanismo de recuperação, na presença de uma falha;
- Pode permitir a tolerância de qualquer falha não grave;
- Fornece mecanismo de migração de processos em sistemas distribuídos, por razões de tolerância a falhas ou balanceamento de carga.

Quando um ponto de verificação é executado, um *snapshot*[19] de todo o estado do programa é salvo em alguma máquina não volátil, máquina de acesso médio.

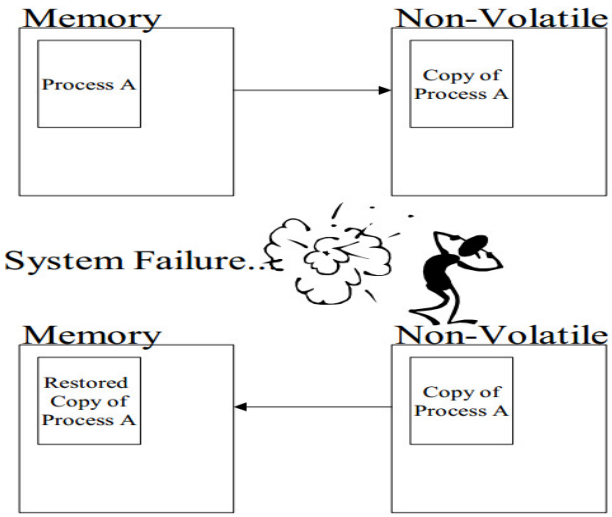



**Figura 16: Checkpoint Recovery[73].**

### *3.1.2 Job Migration*

Às vezes, pode acontecer que devido a uma razão qualquer um *job* não pode ser completamente executado em uma máquina específica. No momento da falha de qualquer tarefa, essa tarefa pode ser migrada para outra máquina. Esta técnica pode ser implementada usando HAProxy.

HAProxy significa *High Availability Proxy* e é usado por empresas para balanceamento de carga e *failover*[20] de servidor na Nuvem. Em HAProxy, normalmente, há um balanceador de carga para distribuir a carga entre um *pool* de servidores web. Ele também funciona como um sistema tolerante a falhas. Sempre que um servidor cair é levado para fora do *pool* até que esteja novamente pronto para lidar com solicitações. HAProxy tem a capacidade de executar esta tarefa, fazendo exames de estado periódicos em todos os

[19] Permite ao usuário criar imagens de um sistema de arquivos especificado.
[20] A capacidade de determinado sistema/serviço migrar automaticamente para um outro servidor, sistema ou rede redundante ou que está em standby quando da ocorrência de falha ou término anormal do servidor, do sistema que estava ativo até aquele instante (littleoak).

servidores em um *cluster*[21]. Mesmo que um dos servidores de aplicativos não esteja funcionando, os usuários ainda terão a disponibilidade para a aplicação. HAProxy irá tratar adequadamente solicitações dos usuários, redirecionando-os para o segundo servidor, dando a impressão de que não há falhas [64].

### *3.1.3 Replication*

Replicação significa nada mais e nada menos que cópia. Várias tarefas são replicadas e elas são executadas em diferentes recursos, para a execução bem-sucedida e para obter o resultado desejado (Figura 17). Ela pode ser implementada usando ferramentas como HAProxy, Hadoop e AmazonEc2 etc.

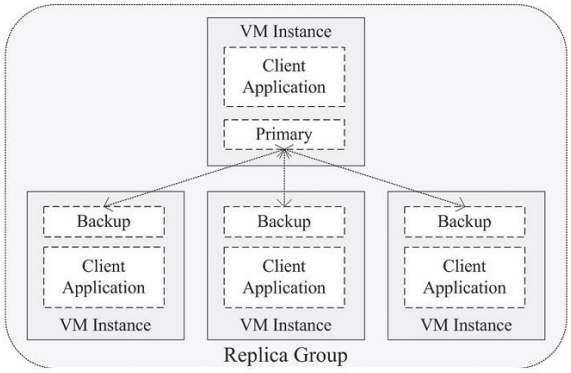



**Figura 17: de replicação de recursos [80].**

### 3.1.4 Outras técnicas

Além das técnicas acima, existem na literatura outras técnicas menos detalhadas, contudo importante citá-las. A Tabela 1 mostra cada técnica e suas características.

**Tabela 1: Outras técnicas e suas características**

| Técnicas | Características |
|---|---|
| ***Self- Healing*** | Neste caso, uma grande tarefa pode ser dividida em partes. Esta Multiplicação é feita para um melhor desempenho. Quando várias instâncias de uma aplicação estiverem executando em várias máquinas virtuais, ele lida automaticamente com falha de instâncias |

[21] É formado por um conjunto de computadores, que utiliza um tipo especial de sistema operacional classificado como sistema distribuído.

| | |
|---|---|
| | de aplicação. |
| ***Safety-bag checks*** | Neste processo, é feito o bloqueio de comandos que não satisfazem as características de segurança [46]. |
| ***S-Guard*** | É menos turbulento para o processamento normal de fluxo e torna mais recursos disponíveis. SGuard é baseada na recuperação de reversão (*rollback*) [51] e pode ser implementado em Hadoop, Amazon EC2. |
| ***Retry*** | É a técnica mais simples de nível de tarefa. Neste caso, repete-se a tarefa que falhou no mesmo recurso da Nuvem [55]. |
| ***Task Resubmission*** | É a técnica mais utilizada de tolerância a falhas em sistemas de fluxos de trabalhos científicos hodiernos. Sempre que uma falha de tarefa é detectada, ela é reenviada quer para o mesmo ou para um recurso diferente para execução. |
| ***User defined exception handling*** | Neste caso usuário especifica o tratamento particular de uma falha da tarefa para fluxos de trabalho. |
| ***Timing Check*** | Isto é feito por cão de guarda (*watch dog*). Esta é uma técnica de controlo da função com o tempo crítico [4]. |
| ***Rescue workflow*** | Esta técnica permite que o fluxo de trabalho persista, mesmo se a tarefa falhar, até que se torne inimaginável avançar sem a restauração da tarefa que falhou. |
| ***Software Rejuvenation*** | É uma técnica que projeta o sistema para reboots periódicos. Ele reinicia o sistema com estado limpo e contribui para um novo começo [63]. |
| ***Preemptive Migration*** | A migração preemptiva conta com um mecanismo de controle de loop de *feedback*. A aplicação é constantemente monitorada e analisada. |
| ***Masking*** | Após o uso de recuperação de erro, o novo estado precisa ser identificado como um estado transformado. Ora, se este processo aplicado sistematicamente, mesmo na ausência de erro efetivo proporciona o erro do usuário de mascaramento [49]. |
| ***Reconfiguration*** | Neste procedimento, elimina-se o componente defeituoso do sistema. |
| ***Resource Co-allocation*** | Este é o processo de alocação de recursos para a execução de outras tarefas. |

## 3.2 Modelos de tolerância a falhas

Com base nos tipos de técnicas descritas na Seção 3.1, vários modelos são implementados. Esses modelos são baseados na proteção contra o tipo de falha, e procedimento.

### 3.2.1 Modelo 1: *A Fault Tolerance for Real Time Cloud Computing* (AFTRC)

A AFTRC é um modelo de tolerância a falhas para a Computação em Nuvem em tempo real baseada no fato de que um sistema de tempo real pode aproveitar tanto a capacidade computacional como o ambiente virtualizado escaláveis de Computação em Nuvem para melhor implementar aplicações em tempo real. Neste modelo, o sistema proposto tolera a falha de forma proativa (Seção 2.6) e faz a dicção com base na fiabilidade dos nós de processamento [56].

### 3.2.2 Modelo 2: *Low Latency Fault Tolerance* (LLFT)

É um modelo proposto que contém um *middleware* de tolerância a falha de baixa latência para fornecer a tolerância a falhas para aplicações distribuídas implantadas no ambiente de Computação em Nuvem como um serviço oferecido pelos proprietários da Nuvem. Este modelo baseia-se no fato de que um dos principais desafios da Computação em Nuvem é o de assegurar que a aplicação está se executando na Nuvem sem interrupção no serviço fornecido para o usuário. Este *middleware* replica aplicação pelo uso de replicação semiativa ou processo de replicação semipassiva para proteger a aplicação contra vários tipos de falhas [57].

### 3.2.3 Modelo 3: *Fault Tolerance Work Flow* (FTWF)

É um modelo proposto que contém um algoritmo de tolerante a falhas de agendamento de fluxo de trabalho para fornecer tolerância a falhas usando a replicação e a resubmissão de tarefas com base na prioridade das tarefas em uma heurística matricial. Este modelo baseia-se no fato de que o fluxo de trabalho é um conjunto de tarefas processadas em alguma ordem com base em dados e dependência de controle. Agendando o fluxo de trabalho levando em consideração a falha de uma tarefa em um ambiente de Nuvem é um grande

desafio. FTWS replica e agenda as tarefas para cumprir o prazo [58].

### 3.2.4 Modelo 4: *Fault Tolerance Manager* (FTM)

É um modelo proposto para superar a limitação das metodologias existentes do serviço *on-demand*. Para alcançar a confiabilidade e resistência que propõem uma perspectiva inovadora na criação e gestão de tolerância a falhas. Por esta metodologia específica, o usuário pode especificar e aplicar o nível de tolerância a falhas de seu desejo sem necessidade de qualquer conhecimento sobre a sua implementação. A arquitetura FTM, sobretudo pode ser visto como um conjunto de vários componentes de serviços web (*web services*), cada um com uma funcionalidade específica [59].

A FTM deve abordar a questão da heterogeneidade em recursos computacionais, cumprir a meta de fornecer transparentemente o suporte da tolerância a falhas para aplicações do usuário contra falhas de nós, e satisfazer as metas de escalabilidade e interoperabilidade [80]. A Figura 18 fornece uma visão desse modelo.

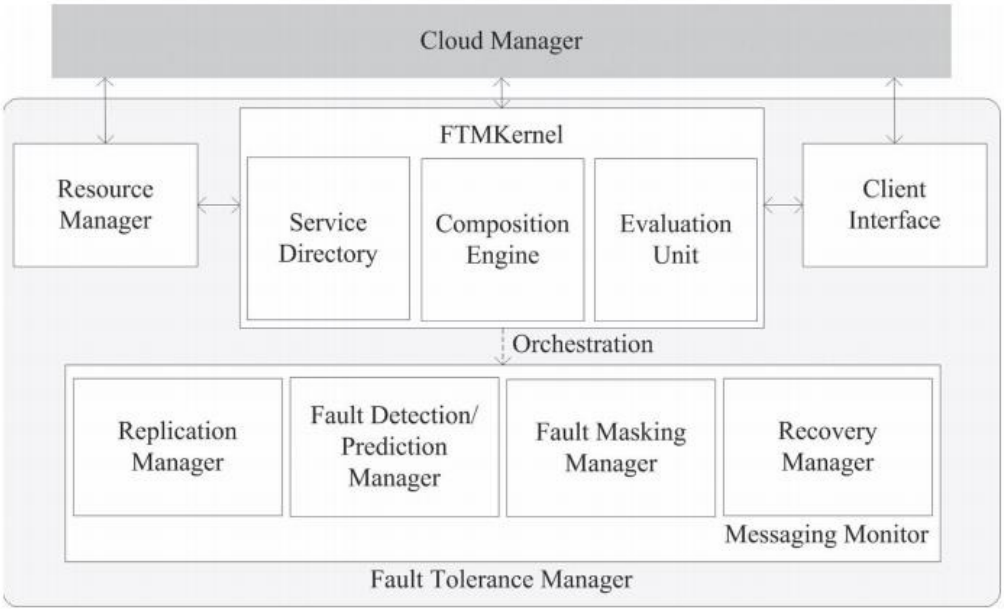



**Figura 18: Visão arquitetural do TFM [80].**

### 3.2.5 Modelo 5: Candy

É um *framework* de modelagem de componentes baseado na disponibilidade, que constrói um modelo de disponibilidade abrangente semi automaticamente a partir de

especificação do sistema descrito pela linguagem de modelagem de sistemas. Este modelo baseia-se no fato de que a garantia de alta disponibilidade de serviços em Nuvem é uma das principais características do serviço de Nuvem e também uma das principais questões críticas e desafiadoras para provedor de serviços de Nuvem [59].

### 3.2.6 Modelo 6: Veja-warden

É um sistema de gerenciamento de usuários uniforme que fornece um espaço global de usuários de diferentes serviços de infraestrutura virtual e de serviço de aplicação em ambiente de Computação em Nuvem. Esse modelo é construído para *cluster* virtual baseado no ambiente da Computação em Nuvem para superar os dois problemas: usabilidade e segurança que surge da partilha de infraestrutura [60].

### 3.2.7 Modelo 7: *FT-Cloud*

É um *framework* baseada na classificação de componentes e sua arquitetura para a construção de aplicações em Nuvem. *FT-Cloud* emprega a estrutura de invocação de componentes e frequência para identificar o componente. Existe um algoritmo para determinar automaticamente a tolerância a falhas imponente [61].

### 3.2.8 Modelo 8: *Magi-Cube*

É uma arquitetura de armazenamento de baixa redundância e alta confiabilidade para a Computação em Nuvem. Constrói-se o sistema na parte superior do HDFS e usa-o como um sistema de armazenamento para arquivos de leitura / gravação e gerenciamento de meta dados. Também foi construído um *script*[22] de arquivo e componente de reparo para trabalhar no *background* de forma independente.

Este modelo baseia-se no fato que a alta fiabilidade e desempenho e baixo custo (espaço) são os três componentes de conflito do sistema de armazenamento. Para oferecer essas facilidades a um determinado modelo, *Magi-cube* foi proposta em [62].

A tabela a seguir ilustra a comparação entre os modelos citados baseados na proteção contra o tipo de falha, e o procedimento a se aplicado.

[22] Na informática são programas usados para facilitar tarefas rotineiras.

**Tabela 2: Comparação entre os vários modelos baseados na proteção contra o tipo de falha e os procedimentos aplicados.**

| Número de Modelo | Proteção contra o tipo de falha | Procedimento aplicado para tolerar a falha |
|---|---|---|
| **1** | *Reliability* | • Excluir o nó dependendo de sua confiabilidade;<br>• Recuperação *Backwork*[23] com a ajuda de Check pointing. |
| **2** | *Crash-cost, trimming fault* | • Replicação |
| **3** | *Dead line of work flow* | • Replicação e resubmissão de Jobs |
| **4** | *Reliability, availability, on demand service* | • Replicação de aplicação dos usuários e se houver falha na replicação pode-se usar algoritmo como *gossip based protocol.* |
| **M5** | *Availability* | • Reúne os componentes do modelo gerado a partir de IBD e STM de acordo com a notação de alocação.<br>• Em seguida, a atividade SNR é sincronizada com o sistema SNR identificando o relacionamento entre a ação da atividade SNR e transição de estado no sistema SRN. |
| **M6** | *Usability, security, scaling* | • A autenticação de duas camadas e solução técnica padrão para a aplicação. |
| **M7** | *Reliability, crash and value fault* | • Componente significativo é determinado com base no ranking.<br>• A ótima técnica de tolerância a falhas é determinada. |
| **M8** | *Performance, reliability, low storage cost* | • O arquivo de origem é codificado em divisões para salvar como um cluster. |

[23] Trabalhos recentemente realizados.

| | | |
|---|---|---|
| | | • Procedimento de recuperação do arquivo é acionado se o arquivo original é perdido. |

### 3.3 Ferramentas utilizadas para implementação de mecanismos tolerantes a falhas

Os desafios e técnicas de tolerância a falhas foram implementados usando várias ferramentas. Na tabela abaixo (Tabela 3)[48] compara-se essas ferramentas com base no seu *framework* de programação, ambiente e tipo de aplicação, juntamente com diferentes técnicas de tolerância a falhas. O HAProxy é usado para o servidor *failover* na Nuvem [75]. Shelp [40] é um sistema de execução leve que pode sobreviver a falhas de software no contexto de máquinas virtuais. Ele também pode trabalhar em ambiente de Nuvem para a implementação de *check pointing*. Quando ao ASSURE [76], ele apresenta pontos de resgate para lidar com falhas previstas pelo programador. Hadoop [77] é usado para aplicações intensivas de dados, mas também pode ser usado para implementar técnicas de tolerância a falhas em ambiente Nuvem. Já o Amazon Elastic Compute Cloud (EC2) [78] fornece um ambiente de computação virtual para executar aplicativos baseados em Linux para tolerância a falhas.

**Tabela 3: Ferramentas usadas para implementar técnicas de tolerância a falhas existentes**

| **Técnicas** | **Políticas** | **Framework de desenvolvimento** | **Sistema** | **Ambiente** | **Falha detectada** |
|---|---|---|---|---|---|
| *Check pointing* | Reativa | SQL,JAVA | SHelp[12] | Máquina virtual | Falha de aplicação |
| *Self Healing, Job Migration, Replication* | Reativa / Proativa | Java | HAProxy[13] | Máquina virtual | Falhas de processo /nó |
| *Check pointing, Retry, Self Healing* | Reativa / Proativa | JAVA | Assure[9] | Máquina virtual | Falha de Host / rede |
| *Job Migration,Replication,Sguard,Resc* | Reativa / Proativa | Java,HTML,CSS | Hadoop[7] | Ambiente Nuvem | Falhas de processo /nó |

| | | | | | |
|---|---|---|---|---|---|
| *Replication, Sguard,Task Resubmission* | Reativa / Proativa | Amazon Machine Image, Amazon Map | AmazonEC2[8] | Ambiente Nuvem | Falhas de processo /nó |

# 4 ANÁLISE COMPARATIVA

Conforme descrito no Capítulo 3, as técnicas de tolerância a falhas existentes na Computação em Nuvem consideram vários parâmetros. Tais parâmetros são como seu tipo de tolerância a falhas (proativa, reativa e adaptativa), desempenho, tempo de resposta, escalabilidade, rendimento, confiabilidade, disponibilidade, usabilidade, segurança e sobrecarga associada. Estes elementos sim podem ser usados como métricas.

A métrica é usada para medir e entender o comportamento do software. Métricas para Computação em Nuvem podem ser usadas para medir o comportamento de Nuvem, que utiliza os recursos de computadores, como um computador virtual coletivo, onde as aplicações podem rodar de forma independente a partir de determinadas conFigurações de computador ou servidor [65]. A Nuvem oferece seus serviços através da internet e fornece a funcionalidade total do usuário de um aplicativo de software pelos sites que oferecem software como serviço. Sites dinâmicos fornecem regularmente diversas informações aos usuários e utilizam páginas geradas dinamicamente e mantém dados para exibição em um banco de dados [66]. A Nuvem, de forma nativa, tem o papel de entregar as aplicações sob demanda. As métricas em Nuvem devem seguir algumas características que ajudam a avaliar a Nuvem em cada e todos os parâmetros necessários para uma Nuvem de boa qualidade, de modo que um cliente possa contar com ela para escolher a melhor Nuvem.

Neste capítulo far-se-á, uma análise comparativa entre os modelos tolerantes a falhas descritos no Capítulo 3 com base nas métricas a seguir.

- ***Performance***: Geralmente, desempenho está ligado à capacidade de uma aplicação na infraestrutura de Nuvem. Isto é usado para verificar a eficiência do sistema. Ele tem de ser melhorada, por exemplo, a um custo razoável, reduzir o tempo de resposta, mantendo atrasos aceitáveis .
- ***Response Time***: é a quantidade de tempo que demora a responder através de um algoritmo específico. Este parâmetro deve ser minimizado.
- ***Scalability***: esta é a capacidade de um algoritmo para realizar a tolerância a falhas de um sistema com qualquer número finito de nós. Essa métrica deve ser melhorada.
- ***Throughput***: isto é usado para calcular o numero de tarefas cuja execução tenha sido concluída. Deve ser elevada para melhorar o desempenho do sistema.
- ***Reliability***: este aspecto tem como objetivo dar resultado correto ou aceitável

dentro de um ambiente limitado tempo.

- ***Availability***: a probabilidade de que um item irá operar satisfatoriamente em um determinado ponto no tempo, com usado sob condições estabelecidas. Disponibilidade de um sistema é tipicamente medida como um fator de sua confiabilidade e assim como a confiabilidade aumenta, o mesmo acontece com a disponibilidade.
- ***Usability***: na medida em que um produto pode ser usado por um usuário para alcançar objetivos com eficácia, eficiência e satisfação.
- ***Overhead Associated***: determina a quantidade de sobrecarga envolvida durante a implementação de um algoritmo de tolerância a falhas. Ele é composto de sobrecarga devido ao movimento de tarefas, internamente ao processador e comunicação entre processos. Isto deve ser minimizado de modo que uma técnica de tolerância a falhas pode funcionar de forma eficiente.
- ***Cost effectiveness***: aqui o custo só é definido como um custo monitorial.

A Tabela 4 ilustra comparação entre os modelos definidos no Capítulo 3 com base nos elementos de métricas em densidade [79] aqui definidos. Nessa tabela, avalia-se as métricas em intensidade, isto é, alto, médio e baixo, 3, 2 e 1 sucessivamente.

**Tabela 4: Comparação entre os modelos com base nos elementos de métricas [80] [79].**

| Modelo (M) | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 |
|---|---|---|---|---|---|---|---|---|
| ***Performance*** | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 3 |
| ***Response time*** | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 |
| ***Scalability*** | 3 | 3 | 1 | 1 | 3 | 3 | 3 | 3 |
| ***Throughghput*** | 3 | 2 | 1 | 3 | 3 | 3 | 3 | 3 |
| ***Reliability*** | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 |
| ***Availability*** | 3 | 3 | 2 | 3 | 3 | 3 | 2 | 2 |
| ***Usability*** | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
| ***Overhead Associated*** | 2 | 1 | 3 | 1 | 1 | 3 | 3 | 2 |
| ***Cost effectiveness*** | 2 | 1 | 3 | 1 | 1 | 1 | 3 | 3 |

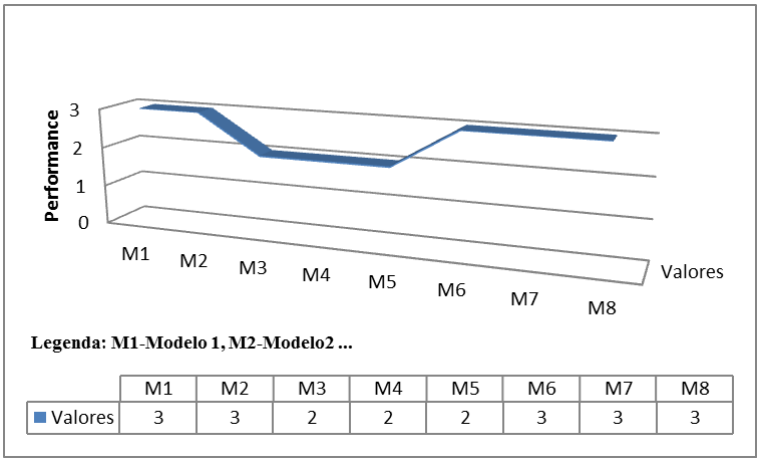
Observando a tabela 4, percebe-se que dependendo do perfil da Nuvem que se deseja montar, e levando em consideração sempre o custo, pode-se escolher um modelo que satisfaça as métricas desejadas. Por exemplo, ao querer maior desempenho pode-se optar

pelos modelos Modelo 1, Modelo 2, Modelo 6, Modelo 7 ou Modelo 8. No entanto, ao querer uma combinação de métricas, tem de haver uma avaliação de cada modelo cujas métricas escolhidas sejam satisfatórias. Por exemplo, se deseja-se maior performance*, availability e throughput*, neste caso teremos apenas Modelo 1 e Modelo 6.

**Avaliação dos modelos**

Considerando as métricas, escolheu-se a representação gráfica de linha para mostrar a variação (tendência) de cada métrica com relação os diferentes modelos.

***Performance*:**




| | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 |
|---|---|---|---|---|---|---|---|---|
| Valores | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 3 |

**Figura 19: Gráfico 1- Densidade de performance em diferentes modelos.**

Esse gráfico representa a densidade de performance em diferentes modelos. Observe que ao desejar-se implantar um sistema tolerantes a falha onde o principal requisito é a performance, os modelos 1, 6, 7 e 8 apresentam os melhores resultados.

***Response Time:***

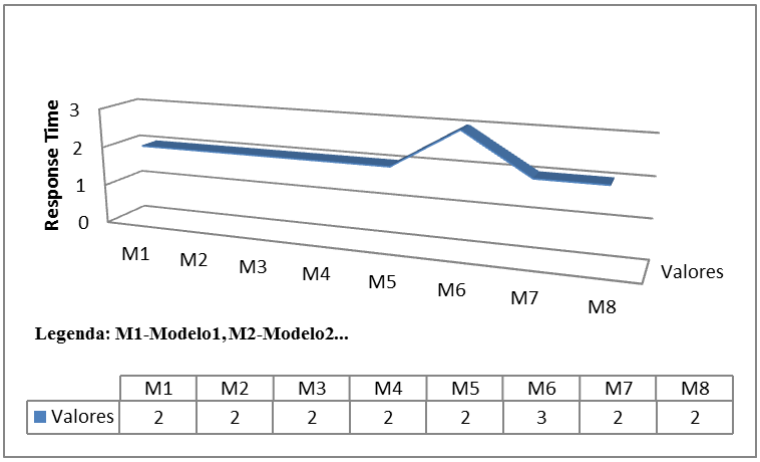



**Figura 20: Gráfico 2- Relação dos modelos com relação a *Response Time.***

Neste caso, apenas o modelo M6 apresenta um bom resultado.

***Scalability***:

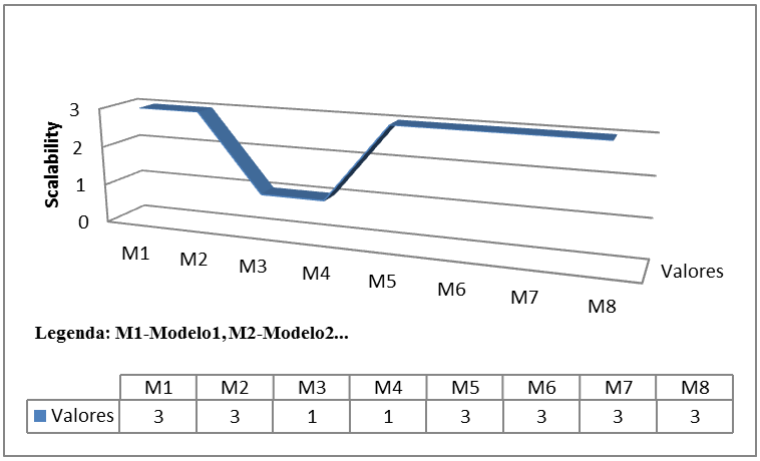
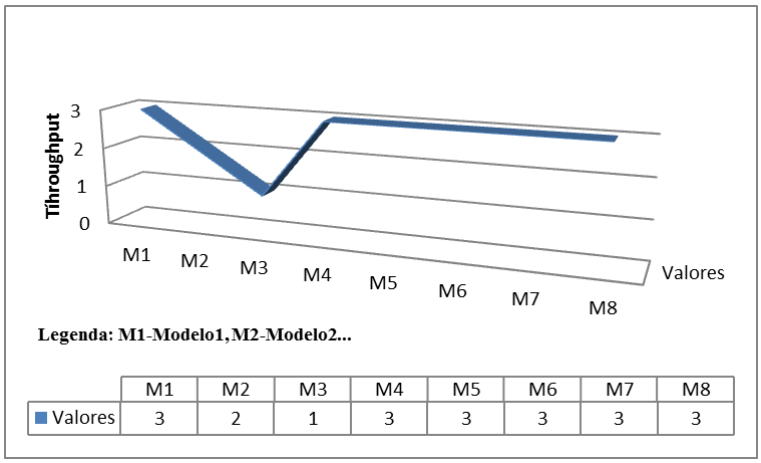



**Figura 20: Gráfico 3- Relação dos modelos com relação a *Scalability.***

Para o caso de escalabilidade, os modelos 1, 2, 5, 6, 7, 8 apresentaram resultados melhores.

***Throughput***:

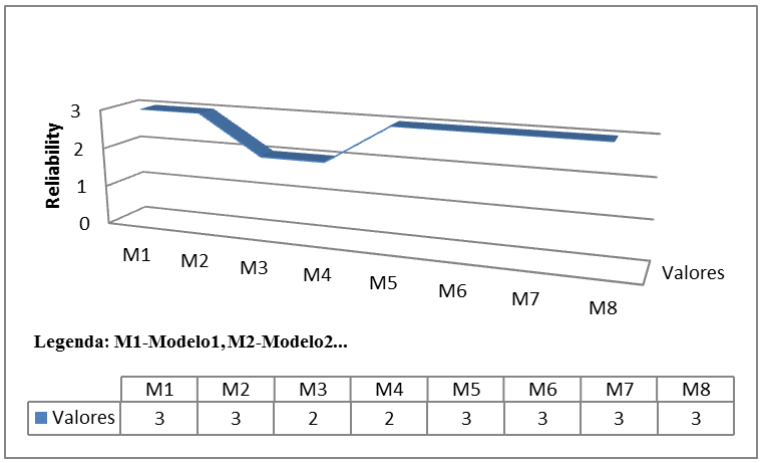



| | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 |
|---|---|---|---|---|---|---|---|---|
| Valores | 3 | 2 | 1 | 3 | 3 | 3 | 3 | 3 |

**Figura 21: Gráfico 4- Relação dos modelos com relação a *Throughput.***

Coforme pode ser visto, os modelos 1, 4, 5, 6, 7 e 8 são os que tem melhores resultados.

***Reliability***:

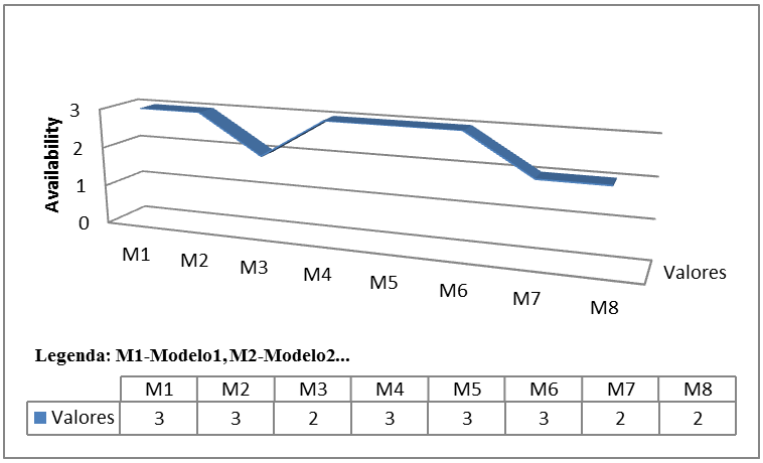



| | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 |
|---|---|---|---|---|---|---|---|---|
| Valores | 3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 |

**Figura 22: Gráfico 5- Relação dos modelos com relação a *Reliability.***

Levando-se em consideração a fiabilidade, pode-se concluir que os modelos 1, 2, 5, 6, 7 e 8 apresentaram os melhores resultados.

***Availability*:**

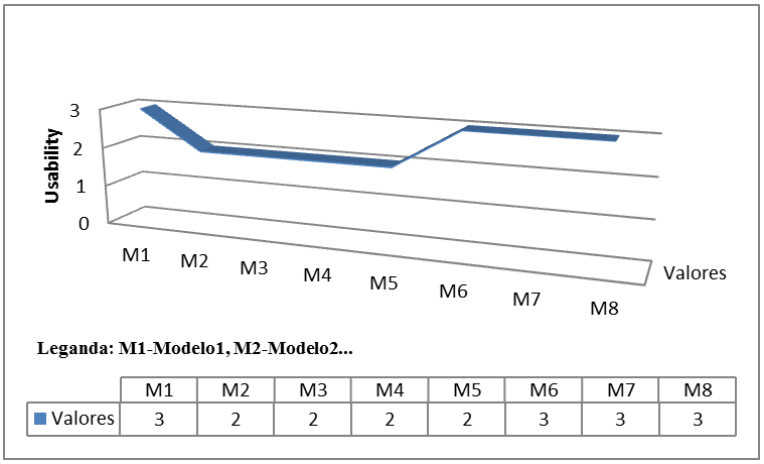



| | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 |
|---|---|---|---|---|---|---|---|---|
| Valores | 3 | 3 | 2 | 3 | 3 | 3 | 2 | 2 |

**Figura 23: Gráfico 6- Relação dos modelos com relação a *Availability.***

Preferindo tal requisito, os modelos 1, 2, 4, 5 e 6 são os melhores.

***Usability***:




| | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 |
|---|---|---|---|---|---|---|---|---|
| Valores | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |

**Figura 24: Gráfico 7- Relação dos modelos com relação a Usability.**

Para o fator de usabilidade, os modelos 1, 6, 7 e 8 apresentaram resultados melhores.

***Overhead Associated***:

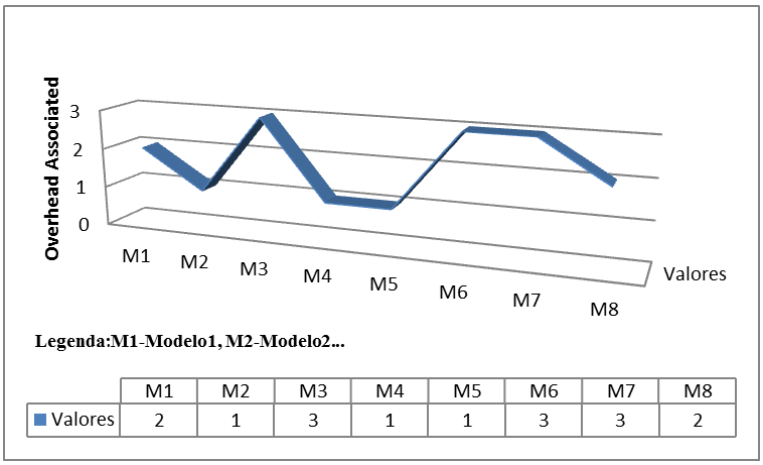



**Figura 25: Gráfico 8- Relação dos modelos com relação a Overhead Associated.**

De acordo com a observação, é evidente dizer que preferindo tal requisito, os modelos 3, 6 e 7 são os melhores visto seus resultados.

***Cost effectiveness***:

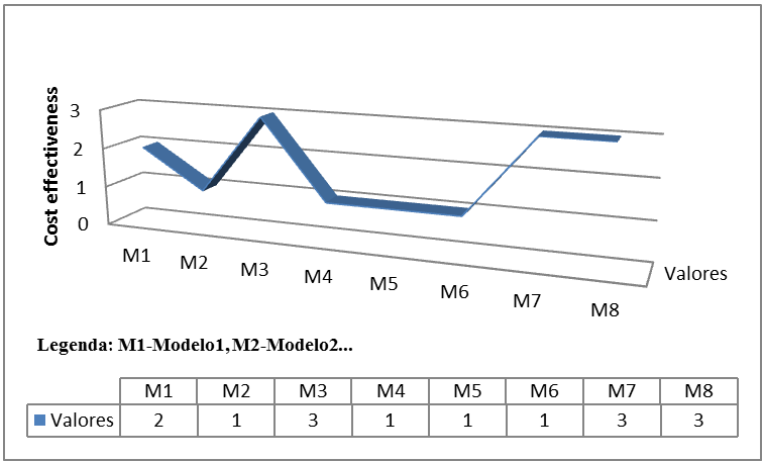



**Figura 26: Gráfico 9-Relação dos modelos com relação a Cost effectiveness.**

Para o requisite custo-efetividade, o trio 3, 7 e 8 são preferíveis aos demais modelos.

**Observações 1:** ao analisar os gráficos de variação "métricas-modelos", pode-se notar que o modelo M6 só não apresentou um bom resultado na métrica *Availability*.

**Observação 2:** observando os gráficos nas Figuras 28, 29, 30 e 31 percebe-se que algumas métricas são "quase" proporcionais, direta ou inversamente entre elas. Por exemplo, quando a performance aumenta há aumento da disponibilidade. Ou ainda, quando c*ost effectiveness* aumenta, *usability* diminue.

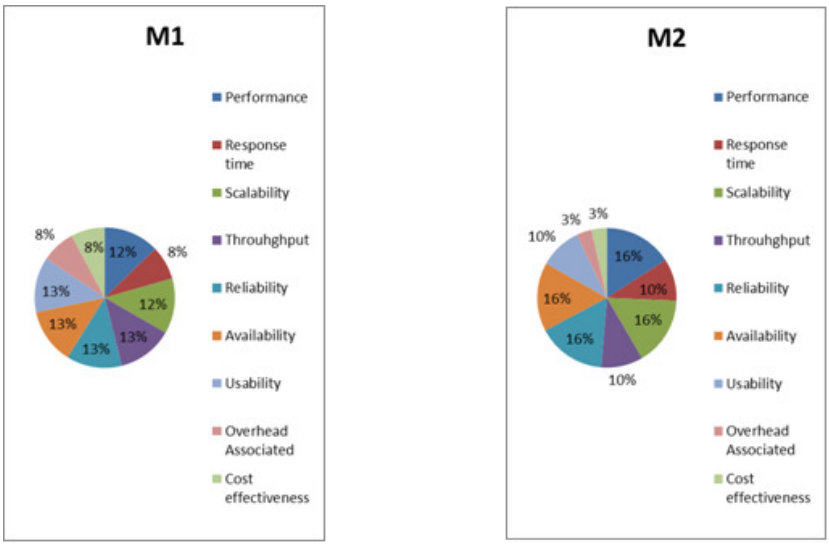



**Figura 28: Contribuição das métricas nos modelos M3 e M4.**

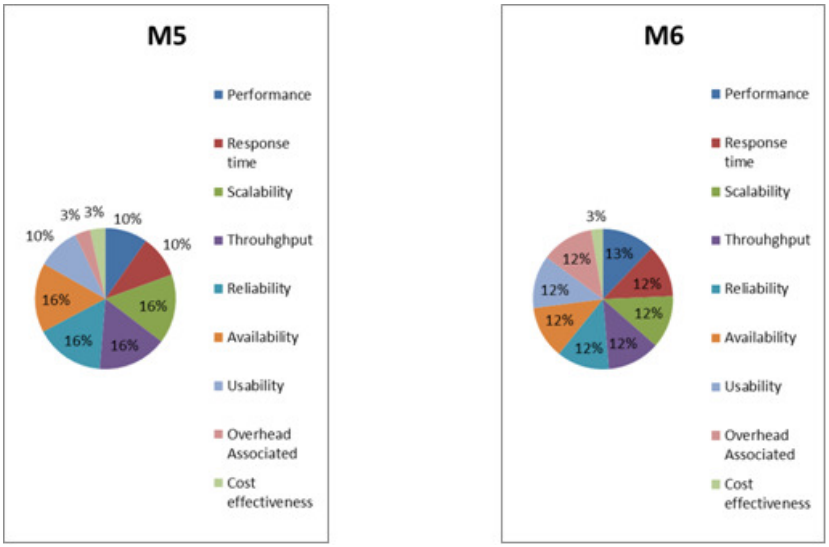



**Figura 27: Contribuição das métricas nos modelos M5 e M6.**

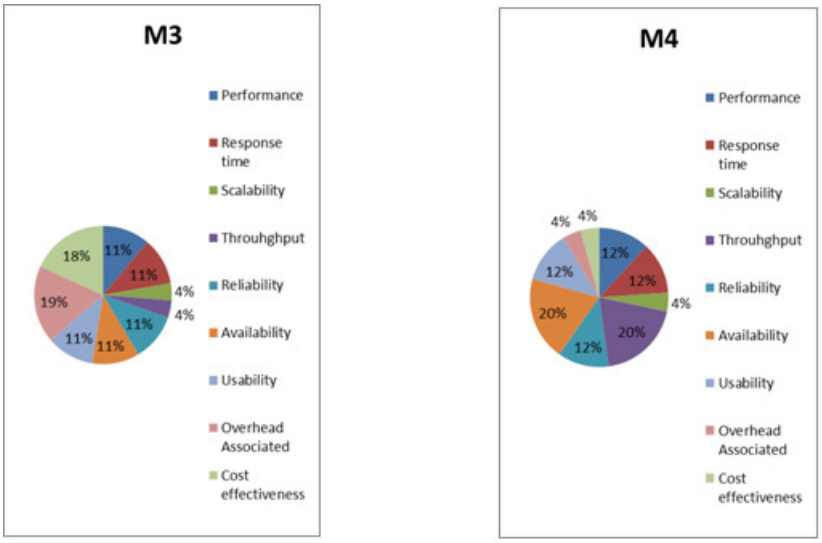



**Figura 30: Figura 20- Contribuição das métricas nos modelos M3 e M4.**

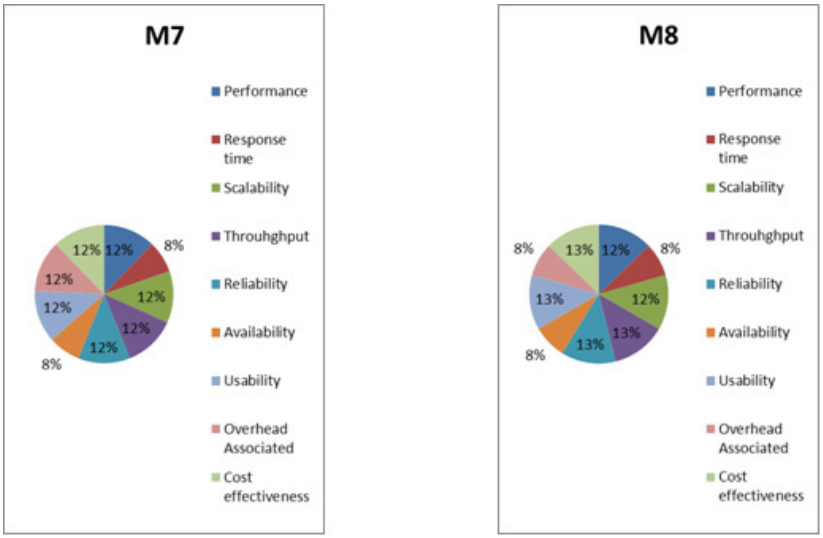



**Figura 29: Contribuição das métricas nos modelos M7 e M8.**

# 5 CONCLUSÃO

Os sistemas tolerantes a falhas se preocupa com todas as técnicas necessárias para permitir que um sistema tolere falhas de software ou hardware durante ao longo de seu funcionamento.

Métodos de tolerância a falhas entram em jogo no momento em que uma falha entra nos limites do sistema. Dessa forma, teoricamente, técnicas de tolerância a falhas são usadas para prever estas falhas e tomar uma ação apropriada antes que elas ocorram. O presente trabalho apresentou a taxonomia de falhas, as falhas em uma Nuvem computacional e as técnicas de tolerância a falhas. Vários modelos propostos para a tolerância a falhas foram apresentados e, por fim, comparados com base nas métricas para tolerância a falhas em Nuvem.

Atualmente, há inúmeros modelos de tolerância a falhas que fornecem diferentes mecanismos tolerantes a falhas para melhorar o sistema. Todavia, ainda existem alguns desafios que precisam de alguma preocupação para cada *framework* ou modelo. Pôde-se perceber que há algum problema em razão de que nenhum deles pode preencher integralmente todos os aspectos de falhas. Outrossim, ao analisar os gráficos de variação "métricas-modelos", pode-se notar que o modelo *Veja-Warden*(Modelo 6) só não apresentou um bom resultado na métrica *Availability*. Portanto, há uma possibilidade de superar as desvantagens de todos os modelos anteriores e tentar fazer um modelo compacto que irá cobrir ao máximo o aspecto de tolerância a falhas. Este modelo pode ser a melhoria do modelo M6 que apenas implicará no aumento da métrica *Cost effectiveness.*

Este trabalho incentiva pesquisadores a contribuir no desenvolvimento de algoritmos mais eficientes. Para isso, a implementação de um sistema capaz de satisfazer a todas as métricas é prevista como trabalho futuro.

**Bibliografia**

[1] Wygwam TM, “Le Computação em Nuvem: Réelle revolution ou simple evolution?” , Bureau d’expertise technologique, laboratoire de recherche xBrainLab, ano, France.

[2] Bakshi, K. (2009). Cisco Cloud Computing- data center strategy, architecture, and solutions point of view white paper for u.s. public sector 1st edition.
http ://www.cisco.com/web/strategy/docs/gov/CiscoCloudComputing WP.pdf

[3] Forester Research James Staten, with Simon Yates, F. E. G. (2008). Is Cloud Computing ready for the enterprise. http ://www.forrester.com/rb/Research/is Cloud Computingreadyfor enterprise/q/id/44229/t/2 ?src=46613pdf.

[4] François Tonic, “Stratégie et révolution de l'infrastructure informatique, de la manière de concevoir les applications et leur consommation dans le nuage sous forme de services”, France, septembre 2009.

[5] Kesselman, C. and Foster, I. (1998). The Grid : Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers.

[6] Bolze, R., Cappello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., Jégou,Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P., Quetier,B., Richard, O., Talbi, E.-G., and Touche, I. (2006). Grid'5000 : A Large ScaleAnd Highly Recon gurable Experimental Grid Testbed. Int. J. High Perform. Comput. Appl., 20(4) :481-494.

[7] Wankar, R. (2008). Grid computing with globus : An overview and research challenges. International Journal of Computer Science & Applications, 5 :56-69.

[8] Foster, I. T., Zhao, Y., Raicu, I., and Lu, S. (2009). Cloud Computing and grid computing 360-degree compared. CoRR, abs/0901.0131.

[9] Inc, A. (2008). Amazon Elastic Compute Cloud (Amazon EC2). Amazon Inc., http ://aws.amazon.com/ec2/#pricing.

[10] VeePee. Veepee : Operateur de services ip. http ://www.veepee.com/.

[11] VMware.org. http ://www.vmware.com.

[12] Engineers, G. S. (2009). Getting started : Java - google app engine.

[13] Microsoft. Windows azure : Microsoft's cloud services platform. http ://www.microsoft.com/windowsazure/.

[14] T. J. Velte, . A. T. Velte e R. Elsenpeter, Cloud Computing, a Practical Approach, 1ª ed., New York, NY: McGraw-Hill, Inc., 2010.

[15] A. B. Tchana, Système d'Administration Autonome Adaptable: application au Cloud, these en vue de l’obtention du doctorat de l’université de Toulouse, Toulouse, 2011.

[16] Difference Between Grid Computing and Cloud Computing: Grid Computing vs Computing. http://www.differencebetween.co.in/technology/difference-between-grid-computing-and-cloud-computing/.

[17] Cloud Computing: Acronyms (IaaS, PaaS and SaaS). http://mahameeditpro.blogspot.com.br/2012/03/saas-software-as-service-essentially.html.

[18] Edutech Associates: Cloud Watching #1 - Cloud 101. http://edutechassociates.net/2011/02/23/cloud-watching-1-cloud-101/.

[19] Goth, G. (2007). Virtualization : Old technology offers huge new potential. IEEE Distributed Systems Online, 8(2).

[20] Popek, G. J. and Goldberg, R. P. (1974). Formal requirements for virtualizable third generation architectures. In Commun. ACM, volume 17, pages 412-421.

[21] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Wareld, A. (2003). Xen and the art of virtualization. In ACM symposium on Operating systems principles (SOSP'03), pages 164-177, New York,NY, USA. ACM.

[22] The user-mode linux kernel. Disponível em: <http ://user-modelinux.sourceforge.net/index.html.>. Acesso em: 01 novembro 2013.

[21] VirtualBox.org. Virtualbox. Disponível em:<http ://www.virtualbox.org.>. Acesso em: 29 outubro 2013.

[24] Braham, P., Dragovic, B., Fraser, K., Hand, S., Haris, T., Alex, Neugebauer, R.,Pratt, I., and War eld, A. (2003). Xen and the art of virtualization. SOSP '03 :Proceedings of the nineteenth ACM symposium on Operating systems principles,2 :14.

[25] Windows Azure: Disponível em: <http://www.microsoft.com/windowsazure/whitepapers/default.aspx.>. Acesso em: 25 outubro 2013.

[26] Jean Arlat, Yves Crouzet, Yves Deswarte, Jean-Charles Fabre, JeanClaude Laprie et David Powell : Encyclopédie de l'Informatique et des Systèmes d'Information, chapitre Tolérance aux fautes, pages 241–270. Vuibert, 2006.

[27] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell et Carl E. Landwehr : Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing, 1(1):11–33, 2004.

[28] John Wakerly : Error detecting codes, self-checking circuits and applications. Computer Design and Architecture Series. Elsevier, 1978.

[29] Manuel Blum, Michael Luby et Ronitt Rubinfeld : Self-testing/correcting with applications to numerical problems. Journal of Computer and System Sciences, 47:549–595,

1990.

[30] Kuang-Hua Huang et Jacob A. Abraham : Algorithm-based fault tolerance for matrix operations. IEEE Transactions on Computers, 33(6):518–528, 1984.

[31] Hal Wasserman et Manuel Blum : Software reliability via run-time resultchecking. Journal of the ACM, 44:826–849, 1994.

[32] Thomas Roche : Dimensionnement et intégration d'un chiffre symétrique dans le contexte d'un système d'information distribué de grande taille. Thèse de doctorat, Laboratoire d'Informatique de Grenoble, France, 2010.

[33] Sun Microsystems, Inc. "Introduction to Cloud ComputingArchitecture" White Paper 1st Edition, June 2009.

[34] Mladen A. Vouk, "Cloud Computing – Issues, Research and Implementations", Department of Computer Science, North Carolina State University, Raleigh, North Carolina, USA,Journal of Computing and Information Technology - CIT 16, 2008, 4, 235–246doi:10.2498 /cit.1001391

[35] L. Arockiam, S. Monikandan& G. Parthasarathy," Cloud Computer: A Survey, International Journal of Internet Computing (IJIC), and ISSN No: 2231 – 6965, Volume-1, Issue-2, 2011.

[36] Y.M. Teo, B.L. Luong, Y. Song, T. Nam,"CostPerformance of Fault Tolerance in Cloud Computing, International Conference on Advanced Computing and Applications, (Special Issue of Journal of Science and Technology, Vol. 49(4A), pp. 61-73), Ho Chi Minh, Vietnam, October 19-21, 2011.

[37] Ravi Jhawar, Vincenzo Piuri, Marco Santambrogioy," A Comprehensive Conceptual System-Level approach to Fault Tolerance in Cloud Computing,DOl 10.1109/SysCon.2012.6189503.

[38] A. Christy Persya,Sr.Lecturer, T.R.Gopalakrishnan Nair," Fault tolerant real time systems, International Conference on Managing Next Generation Software Application (MNGSA-08), Coimbatore, 2008.

[39] A.Tchana, L. Broto, D. Hagimont: Fault Tolerant Approaches in Cloud ComputingInfrastructures. Institut de Recherche en Informatique de Toulouse (IRIT) Toulouse, France. ICAS 2012: The Eighth International Conference on Autonomic and Autonomous Systems

[40] Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zhou, Weizhong Qiang, Gang Hu, "SHelp: Automatic Selfhealing for Multiple Application Instances in a Virtual Machine Environment", IEEE International Conference on Cluster Computing, 2010.

[41] Imad M. Abbadi, “Self-Managed Services Conceptual Model in Trustworthy Clouds' Infrastructure”, 2010.

[42]Yang Zhang1, Anirban Mandal2, Charles Koelbel1 and Keith Cooper,” Combined Fault Tolerance and Scheduling Techniques for Workflow Applications on Computational Grids “in 9th IEEE/ACM international symposium on clustering and grid, 2010.

[43]S. Hwang, C. Kesselman, “Grid Workflow: A Flexible Failure Handling Framework for the Grid”,12th IEEE international Symposium on Zigh Performance Distributed Computing (HPDC’03), Seattle, Washington,USA., IEEE CS Press, Los Alamitos, CA, USA, June 22 - 24, 2003.

[44] Michael Armbrust, Armando Fox,Rean Griffith, “ Above the Clouds: A Berkeley View of Computação em Nuvem”, Electrical Engineering and Computer Sciences University of California at Berkeley, 2009.

[45] Wenbing Zhao, P. M. Melliar-Smith and L. E. Moser,” Fault Tolerance Middleware for Computação em Nuvem”, 2010 IEEE 3rd International Conference on Cloud Computer

[46] Benjamin Lussier, Alexandre Lampe, Raja Chatila, JérémieGuiochet, Félix Ingrand, Marc-Olivier Killijian, David Powell, “Fault Tolerance in Autonomous Systems: How and How Much?” LAAS-CNRS 7 Avenue du Colonel Roche, F-31077 Toulouse Cedex 04, France {firstname.lastname}@laas.fr.

[47] P.LatchoumyandP.Sheik AbdulKhader,”Survey on fault tolerance in grid computing” IJCSI International Journal of Computer Science Issues, Vol 2, No 4, November 2011.

[48] AnjuBala, InderveerChana,” Fault Tolerance Challenges, Techniques and implementation in Computação em Nuvem” IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012 ISSN (Online): 1694-0814 www.IJCSI.org

[49] Jean-clandeLaprie “Dependable computing and fault tolerance: concepts and terminology” LAAS-CNRS 7 Avenue du Colonel Roche, 31400 Toulouse, France.

[50] GolamMoktaderNayeem , Mohammad Jahangir Alam,” Analysis of Different Software Fault Tolerance Techniques”, 2006.

[51] G. Vallee, K. Charoenpornwattana, C. Engelmann, A. Tikotekar, Stephen L. Scott, “ A Framework for Proactive Fault Tolerance”.

[52] Avizienis, (Ed), “Dependable Computing and FaultTolerant Systems Vol. 1”, The Evolution of Fault-Tolerant Computing, Vienna: Springer-Verlag.

[53] TimothyTsai,” Fault tolerance via N-Modular Software Redundancy, FTCS`98 Munich Germany, 23-25,06,1998.

[54] G. Singh, S. Kinger," A Survey On Fault Tolerance Techniques And Methods In Computação em Nuvem", International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 6, June – 2013.

[55] Elvin Sindrilaru,,Alexandru Costan,, Valentin Cristea," Fault Tolerance and Recovery in Grid Workflow Management Systems", 2010 International Conference on Complex, intelligent and Software Intensive Systems.

[56] Sheheryar MalikandFabriceHuet "Adaptive Fault Tolerance in Real Time Cloud Computing" 2011 IEEE World Congress on Service.

[57] Wenbing Zhao, P.M. Melliar and L.E. Mose" Fault Tolerance Middleware for Computação em Nuvem" 2010 IEEE 3rd International Conference on Cloud Computer.

[58] Jayadivya S K, JayaNirmala S, Mary SairaBhanus"Fault Tolerance Workflow Scheduling Based on Replication and Resubmission of Tasks in Cloud Computing" International Journal on Computer Science and Engineering (IJCSE).

[59] Ravi Jhawar, Vincenzo Piuri and Marco Santambrogio"A Comprehensive Conceptual System level Approach to Fault Tolerance in Computação em Nuvem" IEEE.

[60] Jianlin, Xiaoyi Lu, Lin Yu, YongqiangZou and Li Zha"Vega Warden: A Uniform User Management System for Cloud Applications "2010 Fifth IEEE International Conference on Networking, Architecture, and Storage.

[61] ZibinZheng, Tom Chao Zhou, Michel R. Lyu, and Irwin king "FT-Cloud: A Component Ranking Framework for Fault-Tolerant Cloud Applications "2010 IEEE 21st International Symposium on Software Reliability Engineering.

[62] QingqingFeng, Jizhong Han, Yun Gao, Dan Meng"Magicube: High Reliability and Low Redundancy Storage Architecture for Computação em Nuvem" 2012 IEEE Seventh International Conference on Networking, Architecture, and Storage.

[63] M.Armbrust, A.Fox, R. Griffit,et al., "A view of Cloud Computing", Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[64] [Online] Disponível em: http://www.aproxy.1wt.eu/download/1.3/doc/configuration.txt

[65] Gurdev Singh, Shanu Sood, Amit Sharma CM-Measurement Facets for Cloud Performance. International Journal of Computer Applications (0975 – 8887) Volume 23– No.3, June 2011

[66] David Cleary "Web Based Development and Functional Size Measurement" IFPUG Annual conference.

[67] Team Sardes, Inria Rhône-Alpes, Elasticity in Cloud Computer, June 23, 2011.

[68] M. Armbrust et al. Above the clouds: A berkeley view of Cloud Computer. Technical Report UCB/EECS-2009-28, 2009.

[69] Kareim M. Sobhe, Ahmed Sameh “Multi-Channel Clustered Web App lication Server”.

[70] “Making Cloud Service Continuity a Reality” NetPrecept Software Ltd.

[71] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen and Zhenghu Gong, “The characteristics of Computação em Nuvem”.

[72] Disponível em :<https://www.usenix.org/legacy/events/osdi99/full_papers/castro/castro_html/node6.html.>. Acesso em : 26 outubro 2013.

[73] Disponível em:< http://www.ece.cmu.edu/~koopman/des_s99/checkpoint/presentation.pdf.>. Acesso em: 26 outubro 2013.

[74] Eljona P, Ilia N, “Analysis and Strategy for the Performance Testing in Cloud Computing”, Global Journal of Computer Science and Technology Cloud & Distributed, Volume 12 Issue 10 Version 1.0 July 2012, Publisher: Global Journals Inc. (USA).

[75] Disponível em: <http://haproxy.1wt.eu/download/1.3/doc/conFiguration.txt.>. Acesso em: 27 outubro 2013.

[76] S. Sidiroglou, O. Laadan, C. Perez, N. Viennot, J. Nieh, and A. D. Keromytis, “ASSURE: Automatic Software Self-healing Using REscue points”, Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’09), ACM Press, March 7-11, 2009, Washington, DC, USA, pp.37-48.

[77] HadoopMapReduceTutorial.http://hadoop.apache.org/core/docs/current/mapred tutorial.html.

[78] AmazonElasticComputeCloud(EC2) , Disponível em: <http://www.amazon.com/ec2/>;. Acesso em: 05 novembro 2013.

[79] P. Kumar, H. Singh, G. Singh, “Fault Tolerance Techniques and Comparative Implementation in Cloud Computing”, International Journal of Computer Applications (0975 – 8887), Volume 64– No.14, February 2013.

[80] R. Jhawar, V. Piuri, M. Santambrogio, “Fault Tolerance Management in Cloud Computing: A System-Level Perspective”, © 2012 IEEE.

[81] M. Castro and B. Liskov, “Practical Byzantine fault tolerance,” in Proc. 3rd Symp. Operating Syst. Design Implementation, 1999, pp. 173–186.

[82] P. Narasimhan, K. Kihlstrom, L. Moser, and P. Melliar-Smith, “Providing support for survivable CORBA applications with the immune system,” in Proc. 19th IEEE Int. Conf. Distributed Comput. Syst., May 1999, pp. 507–516.

[83] N. Ayari, D. Barbaron, L. Lefevre, and P. Primet, “Fault tolerance for highly available internet services: Concepts, approaches, and issues,” IEEE Commun. Surveys Tutorials, vol. 10, no. 2, pp. 34–46, Apr.–Jun. 2008.

[84] M. Hiltunen and R. Schlichting, “An approach to constructing modular fault-tolerant protocols,” in Proc. 12th Symp. Reliable Distributed Syst.1993, pp. 105–114.

[85] Y. Tamura, K. Sato, S. Kihara, and S. Moriai, “Kemari: Virtual machine synchronization for fault tolerance,” in Proc. USENIX Annu. Tech. Conf. (Poster Session), 2008.