PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIAS

MESTRADO PROFISSIONAL EM GESTÃO DE REDES DE TELECOMUNICAÇÕES

MARCOS JESUS DOS SANTOS

UMA PROPOSTA DE MODELO DE CONTROLADOR PARA COMPUTAÇÃO EM NUVEM UTILIZANDO MÁQUINA DE ESTADOS NEBULOSOS

CAMPINAS
2012

Marcos Jesus dos Santos

UMA PROPOSTA DE MODELO DE CONTROLADOR PARA COMPUTAÇÃO EM NUVEM UTILIZANDO MÁQUINA DE ESTADOS NEBULOSOS

Dissertação apresentada ao Curso de Mestrado Profissional em Gestão de Redes de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito parcial para obtenção do título de Mestre em Gestão de Redes de Telecomunicações.
Área de Concentração: Gestão de Redes e Serviços.

Orientador: Prof. Dr. Eric Alberto de Mello Fagotto.

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

t005.74 Santos, Marcos Jesus dos.
S237p Uma proposta de modelo de controlador para computação em nuvem utilizando máquina de estados nebulosos / Marcos Jesus dos Santos. - Campinas: PUC-Campinas, 2013.
98p.

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

1. Sistemas de recuperação da informação. 2. Redes de computação. 3. Inovações técnológicas. 4. Gerenciamento da informação. I. Fagotto, Eric Alberto de Mello. II. Pontifícia Universidade Católica de Campinas. Centro de Ciências Exatas, Ambientais e de Tecnologias. Pós-Graduação em Engenharia Elétrica. III. Título.

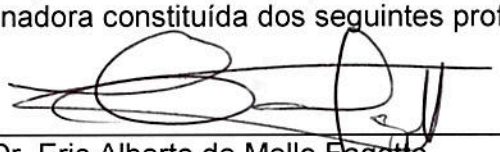
22.ed.CDD – t005.74

MARCOS JESUS DOS SANTOS

UMA PROPOSTA DE MODELO DE CONTROLADOR PARA COMPUTAÇÃO EM NUVEM UTILIZANDO MÁQUINA DE ESTADOS NEBULOSOS

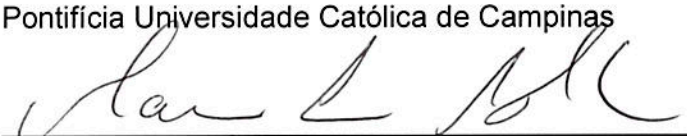
Dissertação apresentada ao Curso de Mestrado Profissional em Gestão de Redes de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito parcial para obtenção do título de Mestre em Gestão de Redes de Telecomunicações.
Área de Concentração: Gestão de Redes e Serviços.
Orientador: Prof. Dr. Eric Alberto de Mello Fagotto.

Dissertação defendida e aprovada em 18 de dezembro de 2012 pela Comissão Examinadora constituída dos seguintes professores:



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

Prof. Dr. Alexandre de Assis Mota
Pontifícia Universidade Católica de Campinas



Prof. Dr. Mauricio Ferreira Magalhães
Universidade Estadual de Campinas

A minha Mãe, in memoriam. A minha Esposa e a minha Filha pelo apoio e compreensão para execução deste trabalho.

AGRADECIMENTOS

A Deus,

Pela vida e pelas oportunidades que me trouxeram até aqui

Ao Prof. Dr. Eric Alberto de Mello Fagotto

Pela Orientação, com paciência, disponibilidade e dedicação.

Ao Diretor de Operações da BT Latam ltda. Ricardo de Carvalho Brissac,

Pelo incentivo, paciência e compreensão nos períodos de maior dificuldade.

"Os grandes feitos são conseguidos não pela força, mas pela perseverança."
Samuel Johnson
(1709 – 1784)

RESUMO

SANTOS, Marcos Jesus dos. Uma Proposta de Modelo de Controlador para Computação em Nuvem Utilizando Máquina de Estados Nebulosos. Dissertação (Mestrado em Gestão de Redes de Telecomunicações) – Pós-Graduação em Engenharia Elétrica, Centro de Ciências Exatas, Ambientais e de Tecnologias, Pontifícia Universidade Católica de Campinas. Campinas, 2012.

A utilização de sistemas de Computação em Nuvem tem se tornado quase que obrigatória na disponibilização de soluções e serviços, seja para pequenas ou grandes corporações, ou mesmo para usuários finais. Tal modelo de consumo de recursos abre toda uma gama de oportunidades de negócio que está sendo cada vez mais explorada. Em especial, o uso de Nuvens Híbridas, nas quais os recursos da própria organização usuária são combinados com recursos de provedores externos, representa um novo desafio para as ferramentas de gerenciamento. Estas ferramentas devem permitir a alocação dos recursos de forma eficiente e possibilitar ao gestor a visualização do estado do sistema em tempo real. Com tal cenário em vista, neste trabalho apresenta-se e investiga-se um modelo de controlador para Computação em Nuvem baseado em uma máquina de estados nebulosos. Esta máquina de estados opera de acordo com critérios definidos nos contratos de nível de acordo (service level agreement – SLA) e com a qualidade de experiência do usuário (quality of experience – QoE). Comparando-se o desempenho desta ferramenta com o de um controlador típico, baseado em Álgebra Booleana, obteve-se uma economia entre 3% até 50% de recursos, isto para um sistema operando com servidor único, dependendo das solicitações de demanda. Finalmente, observa-se que, até o presente momento, a abordagem desenvolvida neste trabalho é inédita na literatura.

Termos de indexação: Computação em Nuvem, Lógica Nebulosa, Gerência de redes.

ABSTRACT

SANTOS, Marcos Jesus dos. A Proposal for a Cloud Computing controller based on a Fuzzy Finite-State Machine. Dissertação (Mestrado em Gestão de Redes de Telecomunicações) – Pós-Graduação em Engenharia Elétrica, Centro de Ciências Exatas, Ambientais e de Tecnologias, Pontifícia Universidade Católica de Campinas. Campinas, 2012.

Over the last years, Cloud Computing has brought about a paradigm shift regarding the deployment of computing services for corporations, small business and even end-users. This model of resource utilization has originated a whole new set of business options and business opportunities yet to be explored. The Hybrid Cloud model, which refers to the situation when the cloud is built using both internal and external resources, or by the combination of resources provided by more than one provider, presents a new challenge for the network management tools. The main difficult is on an efficient resource allocation that allows the network management to follow the system performance in real-time. In this work, it is proposed a Fuzzy Logic controller based on a Fuzzy Finite-State Machine as a tool to manage Hybrid Cloud deployments. This Fuzzy Finite-State Machine operates as in accordance with the service level agreements (SLA) and the quality of experience of the user. By comparing the results obtained by using the proposed controller with typical controllers based on Boolean Logic, savings ranging from 3% to up to 50% where achieved, depending on the number of servers and the demand. The prior use of Fuzzy Finite-State Machines to manage Cloud Computing systems has not been found in the literature until the present proposal.

Index terms: Cloud Computing, Network Management, Fuzzy Logic.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	= Application Programming Interface
CRM	= Customer Relationship Management
DHCP	= Dynamic Host Control Protocol
DNS	= Domain Name System
DoS	= Denial of Service
DDoS	= Distributed Denial of Service
HTTP	= Hypertext Transfer Protocol
IaaS	= Infra-Structure as a Service
IBM	= International Business Machines
IDS	= Intrusion Detection System
IEC	= International Electrotechnical Commission
IP	= Internet Protocol
IPS	= Intrusion Prevention System
ISO	= International Organization for Standardization
ITU	= International Telecommunication Union
LAN	= Rede local do Inglês: Local Area Network
LDAP	= Lightweight Directory Access Protocol
LOG	= Arquivo de Registro de Informações
MPLS	= Multiprotocol Label Switching
NIST	= National Institute of Standards and Technology
PaaS	= Platform as a Service
QoE	= Quality of Experience
QoS	= Quality of Service
RAM	= Random Access Memory
RDP	= Remote Desktop Control
RFC	= Request for Comment
SaaS	= Software as a Service
SLA	= Service Level Agreement
SNMP	= Simple Network Management Protocol
SQL	= Structured Query Language
SSH	=Secure Shell
TCP-IP	= Transmission Control Protocol- Internet Protocol
TI	= Tecnologia da Informação
TS	= Terminal Server
UDP	= User Datagram Protocol
VCPU	= CPU Virtual
VM	= Virtual Machine – Máquina Virtual
VPN	= Virtual Private Network

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

A utilização de computação como serviço, chamada comumente de Computação em Nuvem ou "*Cloud Computing*", tem se tornado praticamente uma norma para implementação de soluções em Tecnologia da Informação. Esta solução é especialmente interessante para desenvolvedores de serviços e soluções para Internet como destacado por[1]. Este modo de uso de recursos e sistemas utiliza vários modelos de serviços e métodos de criação de sistemas a serem disponibilizados. O tema tem despertado tanto interesse, que associações e entidades têm criado grupos específicos a respeito de Computação em Nuvem, como o Portal do *IEEE Cloud Computing Comunity*[2], ou o *Focus Group* do ITU[3].

A definição mais aceita para Computação em Nuvem é a do NIST[4] que define a Computação em Nuvem como um modelo de acesso sob demanda de recursos computacionais compartilhados, utilizando rede, os quais podem ser aumentados ou diminuídos com mínima ou nenhuma interação entre o usuário e o provedor .

As aplicações da Computação em Nuvem, compreendem serviços de correio eletrônico(por exemplo o Gmail [5] e o Microsoft Live mail[6]), discos virtuais para armazenamento de arquivos como o Dropbox [7], e sistemas de relacionamento com o cliente, como o Salesforce[8].

Normalmente estes serviços são contratados de provedores de serviços como Amazon[9] e Google[10], sendo que o usuário possui muito pouco ou nenhum conhecimento sobre os recursos utilizados pelo provedor para o fornecimento dos serviços. Neste tipo de ambiente, o usuário final tem uma capacidade limitada de acompanhamento dos serviços providos, principalmente em relação ao acesso a ferramentas de depuração de problemas, quando o desempenho obtido não equivale ao esperado ou necessário ao negócio fim.

Não é raro que, devido a requisitos de negócio, ou políticas da empresa que, por exemplo, não se permita que determinado tipo de informação seja hospedada em sistemas sob controle de terceiros. Desta frorma, faz-se necessária a utilização de soluções híbridas, nas quais recursos internos sejam combinados com recursos externos. Várias pesquisas tem abordado o gerenciamento deste tipo de solução como [11] que demonstra as vantagens de se utilizar um modelo preditivo para o consumo de recursos de rede em implementações de nuvens híbridas.

Um tema recorrente na relação entre o cliente ou usuário e o provedor de nuvem é a monitoração da Qualidade de Serviço (QoS) e do cumprimento dos acordos de níveis de serviços(SLA) contratados. Destaca-se também a oferta da monitoração de QoS como serviço, utilizando o conceito de Computação em Nuvem para monitoração de Qos. Em [12] é demonstrado a implementação de uma solução de monitoração de QoS para soluções de nuvem, que é provida no

modelo de nuvem. A solução nomeada QoS-MONaaS [12], permite a geração de indicadores a partir das SLAs que são monitorados contra violação.

O systema DynaQoS descrito em [13] é um controlador de QoS ,baseado em lógica nebulosa, para aplicações em nuvem. O controlador proposto é auto ajustável e controla os recursos em um ambiente utilizando máquinas virtuais. O objetivo é garantir que as políticas de QoS sejam atendidas na alocação de máquinas virtuais.

Um tema correlato à gestão de QoS e SLA para serviços em nuvem, é a gestão do QoE. Em [14] é feita uma análise dos desafios do tratamento do QoE para as aplicações corporativas, que apesar de estarem cobertas em contratos de SLA , que governam o desempenho do serviço contratado e entregue aos usuários , sua tradução para o QoE não é direta. Para viabilizar o gerenciamento de QoE para soluções de Nuvem , em [15], é proposta uma arquitetura de solução que permita a monitoração do QoE , sua interligação com o QoS de rede, de Computação, os requisitos e métricas de Negócio.

Estão documentadas na literatura aplicações de lógica nebulosa[16] em soluções de gerência de redes e serviços de datacenter. Em[17] é proposto um controlador baseado em lógica nebulosa para monitorar e controlar o cumprimento de acordos de níveis de serviço em ambientes com redes virtuais, propondo a um controle que pune as redes que violam os SLAs acordados. O grau da violação, e o grau utilização da rede são considerados em conjunto para estabelecer as punições. Por sua vez em [18] é proposta um gerenciador baseado em políticas, associado a um controlador Nebuloso, para os nós de rede no escopo do sistema de gerência. Este trabalho aborda exclusivamente o gerenciamento de QoS na rede, através de DiffServ[19] , mas os conceitos e métodos podem ser transladados para a parte de rede que suporta o consumo de uma solução em nuvem.

O uso da lógica nebulosa para criação de um processo de negociação de SLAs entre o cliente e o provedor é proposto em [20]. Nesta proposta o processo de negociação para contratação do serviço é feito considerando-se os requisitos do cliente em comparação com as ofertas feitas pelo provedor. Para a comparação e a seleção das opções de contratação é utilizado a lógica nebulosa.

Em [21], se descreve o uso de controladores baseados em Lógica Nebulosa para a alocação de recursos em um centro de dados, de maneira virtual, tratando tanto localmente a carga alocada e os recursos disponíveis como considerando as políticas globais de gestão de carga.

Em [22] são apresentadas métricas de QoS, para soluções desenvolvidas segundo arquitetura orientada a serviços. Estas métricas se adéquam ao tratamento de QoS para as soluções baseadas em Nuvem. Em[23] é discutido o uso de um controlador Fuzzy para a alocação de recursos em ambientes de Computação em *Grid*.

.

1.2 Objetivos do trabalho

O objetivo deste trabalho é desenvolver um sistema de gerência, baseado em Lógica Nebulosa, para sistemas de Computação em Nuvem. Este sistema opera de acordo com critérios definidos nos SLAs e no QoE. Trabalhos como [17], [18], [21], [24], são adequados para o provedor de serviço ou para o usuário final que pretende implementar sua nuvem privada. O provedor da solução em nuvem, seja ele externo ou interno, tem a administração dos equipamentos utilizados, podendo coletar dados de gerência obtidos diretamente dos dispositivos físicos. O usuário final tem somente acesso aos servidores e serviços virtuais, com um acesso bem limitado a informações de desempenho, alarmes e erros ocorridos na plataforma. As soluções de gerência disponibilizadas ao usuário final normalmente se limitam a ferramentas de contabilidade de uso, que lhe permitem obter relatórios de uso de recursos para validar a fatura que recebe.

Outros trabalhos, como [20] [25] [26] são focados na seleção dos serviços de nuvem segundo critérios de SLAs, de custos ou de desempenho. Os métodos e ferramentas propostas atuam na fase pré-ativação das máquinas virtuais que compõe o serviço utilizado.

Considerando-se estes aspectos este trabalho propõe uma ferramenta a ser utilizada pelo usuário final, para por um lado avaliar os SLAs que se obtém da solução de nuvem que se utiliza, e por outro lado tomar decisões de aumento, ou migração de recursos. Isto é feito através do uso da lógica nebulosa de maneira similar a [18].

A diferenciação deste trabalho se dá pela utilização de máquinas de estados nebulosos[27] que permite considerar a sequência de eventos e a temporização no sistema de gerenciamento de Computação em Nuvem.

1.3 Organização da dissertação

No Capitulo 2 é feita uma apresentação das definições e tecnologias utilizadas para a montagem de soluções de Computação em Nuvem diferenciando os vários modelos e sua relação com o trabalho realizado.

No Capitulo 3 é apresentado o controlador nebuloso caracterizando sua aplicabilidade.

No Capitulo 4 é apresentado uma introdução a máquinas de estados nebulosos.

No Capitulo 5 é apresentado o modelo proposto incluindo as variáveis utilizadas para a gerência.

No Capitulo 6 é apresentada o método utilizado na experimentação.

No Capitulo 7 são apresentados os resultados experimentais e a discussão dos mesmos.

No Capitulo 8 são apresentadas as conclusões a respeito deste trabalho.

2 Computação em Nuvem

Neste capítulo apresenta-se uma revisão dos modelos existentes para sistemas de Computação em Nuvem.

2.1 Definição da Computação em Nuvem

A definição do NIST [4] estabelece, cinco características, três modelos de serviço e quatro modelos de provimento para que o serviço possa ser classificado como Computação em Nuvem. As características que o serviço deve possuir para enquadrar-se como um sistema de Computação em Nuvem são as seguintes:

- Permitir alteração dos recursos utilizados sob demanda, com a mínima interação do usuário com o provedor, normalmente valendo-se de interfaces de programação de aplicativos(APIs) ou portais para interagir com o provedor.

- Fazer uso de recursos compartilhados.

- Permitir à rápida variação na quantidade de recursos alocados, aumentando ou diminuído a quantidades de recursos utilizados.

- Cobrado por uso, o usuário paga somente a quantidade de recursos efetivamente utilizados.

- Ser facilmente acessível pela rede, através do uso de clientes e protocolos padrões.

Este serviços podem, também de acordo com a definição do NIST [4] ser classificados pelos seguintes modelos de acordo com o recurso que é fornecido:

- Infraestrutura como serviço (Infrastructure as a Service - IaaS):. Modalidade na qual o serviço fornecido equivale a um recurso físico como, espaço em disco, ou um servidor de capacidade de computação.

- Plataforma como serviço (Platform as a Service - PaaS) :- Permite ao usuário montar suas aplicações com os recursos fornecidos pelo provedor usando ferramentas ou linguagens suportadas pela plataforma oferecida.

- Software como Serviço (Software as a Service - SaaS): A aplicação final é oferecida ao usuário, sendo que o provedor ser encarrega de manter e operar toda a plataforma.

Um fato que se destaca é que normalmente estas plataformas são utilizadas de maneira hierárquica[28], por exemplo, o SaaS é montado sobre uma plataforma PaaS que por sua vez é montada sobre uma plataforma de de IaaS..

2.2 Implementação

A definição do NIST[4] aborda também como os recursos são providos, classificando a Nuvem nos seguintes modelos:

- Nuvem Privada: Os recursos utilizados, computadores, sistemas de armazenamento de dados em disco, são de uso exclusivo de uma empresa ou usuário.

- Nuvem Comunitária: Quando são utilizados por um grupo de empresas ou entidades que tenham fins comuns.

- Nuvem Pública: Os recursos utilizados são compartilhados entre varias empresas usuárias sem nenhuma relação entre si.

- Nuvem Híbrida: Para disponibilização dos serviços são utilizados recursos de duas ou mais nuvens distintas dos modelos anteriores.

Os modelos Privados e Comunitários não definem a propriedade ou a localização dos recursos, que podem estar dentro ou fora da organização, mas somente a alocação dos recursos físicos utilizados pela nuvem.

De especial interesse é a utilização de nuvens híbridas. As mesmas podem ser oriundas da necessidade de utilização de diferentes componentes na disponibilização de uma solução mais complexa, ou da necessidade de prover redundância. Outra aplicação é como forma de prover capacidade extra a uma solução estabelecida. Estão documentados na literatura estudos com o objetivo de permitir a gerência de nuvens híbridas, como a proposta em [29], que aborda especificamente a criação de uma interface padronizada e uma camada de abstração que permita a interação com a API de diversos fornecedores. Já existem várias implementações para a interface com a gerência de nuvens híbridas e de diferentes provedores como o Jclouds[30], que implementa uma interface padrão para várias opções de software de controle de Nuvens. A interface é apresentada como uma API em Java, que conta com módulos de tradução para as interfaces nativas de cada sistema de Computação em Nuvem. Outro exemplo de implementação de sistema de gerência de nuvens híbridas é o Eucalyptus[31][32] permite a gestão de nuvens privadas e públicas com interface compatível com a API da Amazon EC2[9] .

Na Figura 1 tem-se uma representação de uma nuvem híbrida, com uma nuvem privada e duas nuvens públicas.

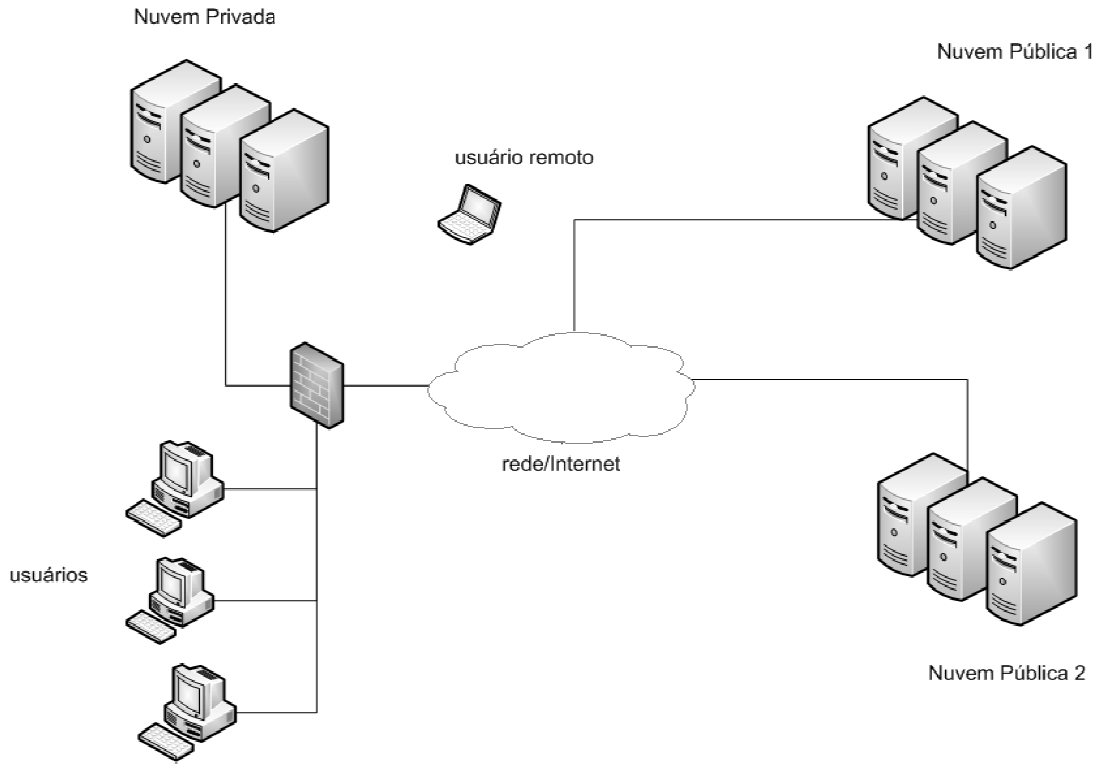


Figura 1 Nuvem Híbrida

As nuvens privadas, e públicas podem servir de alternativa uma a outra, com diferentes condições de operação e custo. Em uma solução complexa, formada por ofertas de diferentes provedores, a importância da solução de gerência aumenta como forma de garantir os parâmetros de qualidade, desempenho e segurança da solução final.

2.3 Tecnologias

A Tecnologia chave para a Computação em Nuvem reside na utilização de recursos virtuais[33], tanto de rede, computação e infraestrutura, como espaço em disco. Isto é obtido através da transformação de um servidor com hardware mais potente em servidores virtuais menores, que aparecem para o usuário final como o próprio hardware, pelo uso de uma camada de software especifica para este fim.

2.3.1 Máquina virtual

O Uso de máquinas virtuais é o grande impulsionador da Computação em Nuvem[1], principalmente no modelo de IaaS. Em alguns casos, é possível o uso do hardware físico[34], no modo chamado “bare metal”. Neste modo o equipamento é mantido desligado e ao ser acionado faz a carga de uma imagem de sistema operacional especifica, comportando-se para todos os aspectos da Computação em Nuvem da mesma maneira que uma máquina virtual. Para aplicações onde os requisitos de Hardware ou de licenciamento requeiram o uso de uma máquina física, o suporte a “*bare metal*” permite que o usuário continue se beneficiando das facilidades da Computação em Nuvem.

Para a execução de uma máquina virtual é carregado na máquina física, denominada hospedeira, um sistema operacional básico que permite que os recursos sejam compartilhados, incluindo, CPU, memória e dispositivos de entrada e saída, sejam estes últimos de armazenamento ou de comunicação. Este sistema, chamado monitor[35], realiza o controle dos recursos, compartilhando os ciclos de processamento da CPU e garantindo a isolação entres as máquinas virtuais que podem ter requisitos de segurança diferentes, ou mesmo pertencer a clientes diferentes como no caso das nuvens públicas. Na Figura 2 tem-se um diagrama básico da estrutura de uma máquina hospedeira. A camada de software que implementa o monitor pode funcionar com o sistema operacional nativo da máquina física ou, então, substituí-lo por completo.

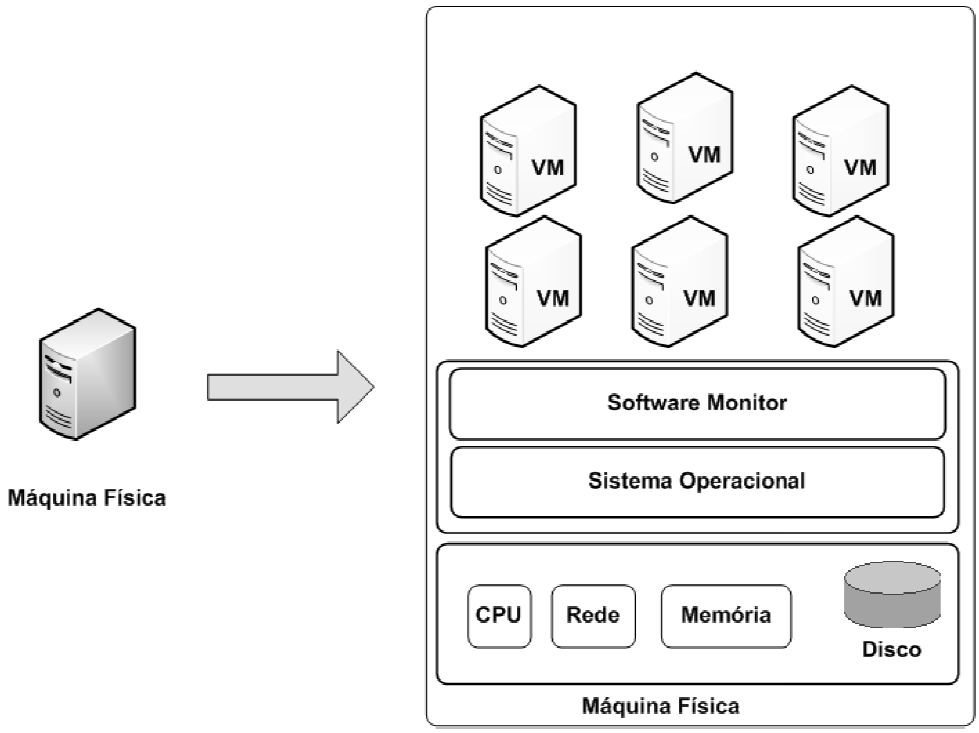


Figura 2 - Máquina Virtual

Os softwares de gestão de máquinas virtuais mais utilizados são os de código aberto como,por exemplo, XEN[36][37] , KVM[38][39], VirtualBox [40] e os proprietários VMware[41] e o Hiper-V da Microsoft [42]. Estes softwares transformam PCs comuns ou servidores padrão Intel, em máquinas hospedeiras de servidores virtuais. Os softwares de código aberto apresentam a vantagem óbvia de não terem custos de licença, enquanto os softwares proprietários, contam com uma estrutura de suporte mais robusta.

Um dos pontos abordados na literatura [43] é o da segurança que um usuário final pode conseguir em um ambiente de nuvem pública, sendo que apesar de ter pleno controle da máquina virtual, a máquina hospedeira é administrada pelo provedor, e como tal este pode ter pleno acesso a todos os recursos suportados por esta. Isto, em princípio, permitiria ao provedor acessar os dados confidenciais de um cliente, sem a autorização ou conhecimento deste.

Uma das possibilidades para proteger os dados seria criptografa-los, o que pode ser feito em seu armazenamento no disco ou em sua transmissão através de uma VPN. Uma vez na memória da máquina virtual, para processamento, os mesmo voltariam a estar acessíveis. Em [44] é proposta uma solução para o problema de confidencialidade e integridade dos dados, com a implementação de uma solução de nuvem segura através entidades externas de certificação. Este modelo baseia-se em uma entidade certificadora que atesta que o hardware e o software, da máquina hospedeira do provedor, são íntegros e atendem os controles necessários a garantir a confidencialidade dos dados do cliente.

2.3.2 Rede virtual

As redes que as máquinas virtuais utilizam para a comunicação entre cada instância dentro do hospedeiro e com o mundo externo, também são disponibilizadas de maneira virtual. Vários estudos [45], [46],[47] endereçam as consequências desta característica na capacidade de gerenciamento deste modelo de implementação de recursos de rede. Em [45] são discutidas as opções de monitoração apresentadas ao usuário final para monitorar as interfaces de rede que lhe são visíveis.

Na comunicação entre as máquinas virtuais dentro de um mesmo servidor hospedeiro, os pacotes são roteados em uma rede virtual implementada pelo software monitor [37]. Isto tem grande impacto nas medidas que se consegue fazer nestes dispositivos e na interpretação de seus resultados, como detalhado em [46]. Nos servidores representados na Figura 3 todo trafego, entre as máquinas virtuais representadas, é processado por software internamente ao servidor.

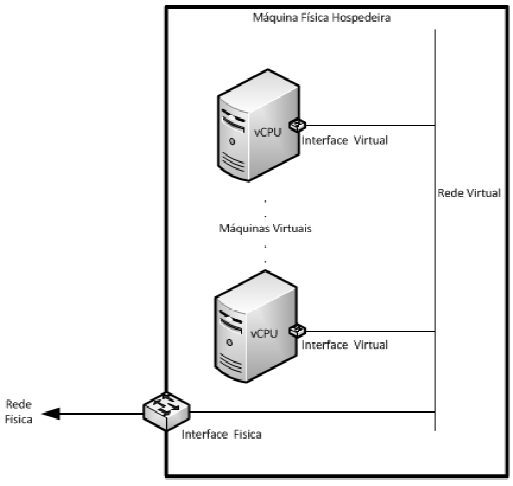


Figura 3 Interface de rede Virtual

Posto que grande parte do processamento é realizado pela CPU da máquina hospedeira, o desempenho de rede pode exercer grande influência no desempenho do sistema, como detalhado em [48]. Isto evidencia a importância

em se utilizar um sistema de gerenciamento adequado às características de um sistema de Computação em Nuvem.

2.3.3 Software de orquestração de Nuvem

Para controlar todos os dispositivos, o monitor de máquinas virtuais, e fazer a interface com usuários, é utilizada uma camada de software [49] que integra todos os componentes e oferece uma interface para que o usuário final possa controlar os recursos contratados. Esta camada de software é composta por uma plataforma de implementação de sistemas de Computação em Nuvem.

Esta plataforma de implementação de sistemas de computação em nuvem, representada na Figura 4, integra todos os dispositivos, “switches”, roteadores, servidores e sistema de armazenamento controlando-os, para que trabalhem em conjunto para a montagem do sistema de Computação em Nuvem.

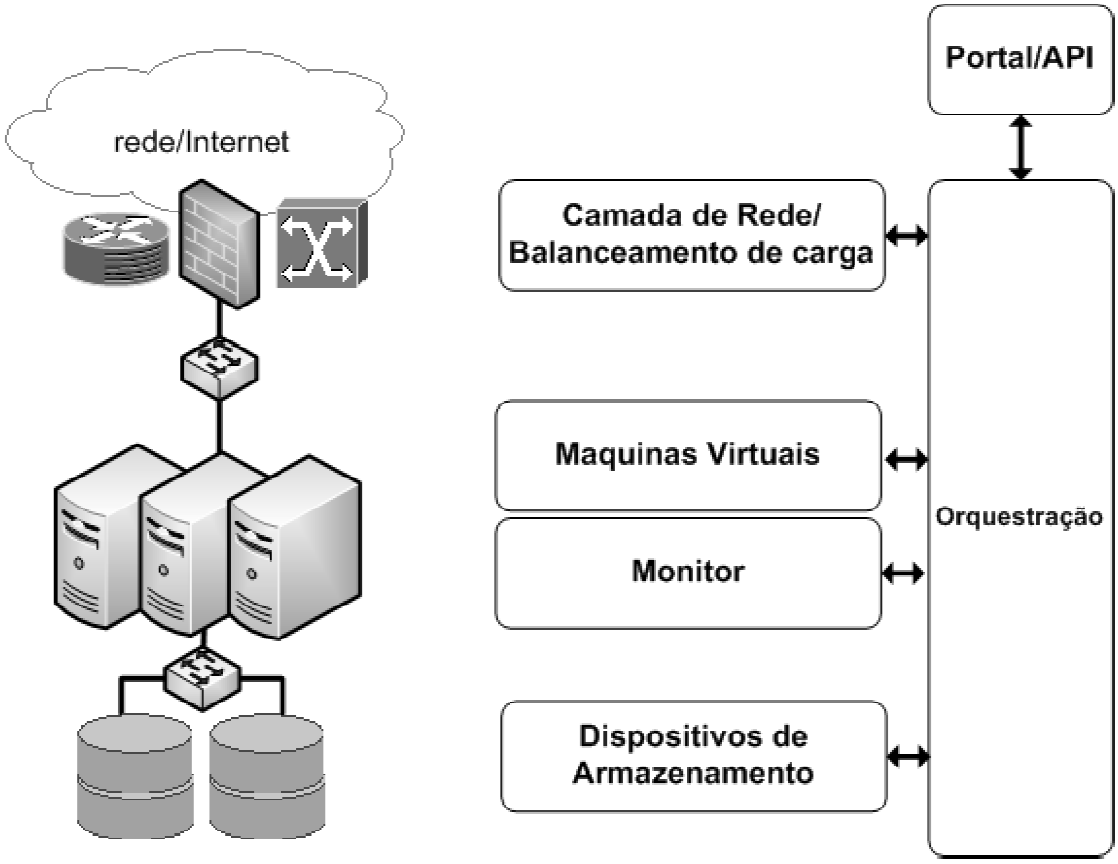


Figura 4 Sistema de Gerência de Nuvem

Estão disponíveis vários conjuntos de softwares para a implementação de nuvens, como os de código aberto Openstack [50], Cloudstack[51] e OpenNebula [52], existindo também vários sistemas proprietários, como Vcloud Director[53], OnApp[54].

2.4 Custos

A cobrança dos serviços em Nuvem por definição é feita por uso considerando-se a quantidade de recursos utilizados por período de tempo. De acordo com o modelo IaaS, o custo seria dado pela soma do custo de uso de máquina virtual, do custo de uso de espaço de armazenamento e do custo de tráfego de rede [55] [56][57]. Alguns provedores cobram o tráfego de rede por banda alocada e não por bytes trafegados[58].

O primeiro componente de custos a ser considerado é o custo de uso de máquina virtual, $vmCost$ Este custo é cobrado por período de tempo de uso. Normalmente os provedores definem este tempo como 1 (uma) hora [55] [56][57], ou por mês [58] [59]. Para os propósitos dos testes a unidade monetária será definida como unidade monetária $. Também foi considerado que somente há um modelo de máquina virtual disponível. O custo de uso de máquinas virtuais, $VMUseCost$, é dado pela somatória de horas utilizadas, *Hs*, para cada máquina virtual ativada, VMi , multiplicado pelo custo horário, $vmCost$ desta máquina .

$$VMUseCost = \sum_{i=1}^{n} (vmCost_{VMi} \cdot Hs) \quad (1)$$

Considerando $bwCost$, como o custo por unidade de dados transferidos, $bytesIn$ no sentido rede para servidores, e $bytesOut$ no sentido servidores para rede, normalmente dado em valor monetário, $, dividido por gigabytes ou megabytes transferidos em ambas as direções, na interface do provedor com a rede de acesso, normalmente Internet. O Custo de rede, $NetworkCost$, é dado por (2).

$$NetworkCost = (bytesIn + bytesOut) \cdot bwCost \quad (2)$$

Custo por unidade de dados armazenados, $stCost$, normalmente dado em valor monetário por Gigabyte armazenado por período de tempo. O espaço de dados pode ter mais que um tipo de classificação, podendo ter opções mais caras segundo o desempenho em números de transações de entrada e saída suportados, sua perenidade, ou segurança implementadas.

Considerando opção única de armazenamento o custo total, $StorageCost$ é dado pelos bytes armazenados, $bytesArmazenados$, pelo número de horas que estes dados ficaram armazenados.

$$StorageCost = bytesArmazenados \cdot Hs \cdot stCost \quad (3)$$

O custo total de uso da nuvem por um dado período de tempo, TotalCost, é dado pela soma dos três custos, de utilização de máquinas virtuais, $VmUseCost$, trafego de dados, $NetworkCost$, e de dados armazenados, $StorageCost$, como representado em (4).

$$TotalCost = VmUseCost + NetworkCost + StorageCost \quad (4)$$

Outro componente desta parametrização é a parte de receitas da solução. Este aspecto da solução normalmente não recebe o tratamento adequado.

No modelo transacional pode-se atribuir uma receita, *Rtransac*, por transação individual. A receita $Rtotal$, por período de tempo é dada por (5). A taxa de transações por período de tempo, Transac, responderia pela receita por período de tempo em comparação ao custo por período.

$$Rtotal = \frac{Transac}{s} \cdot Rtransac \quad (5)$$

2.5 Desafios

Uma característica importante para o gerenciamento de uma infraestrutura computacional são os direitos administrativos sobre a mesma. Em uma implementação que utiliza o modelo IaaS, o cliente possui completa gestão sobre as máquinas virtuais podendo instalá-las e configurá-las. O cliente pode, a qualquer momento iniciar, ou desligar suas máquinas virtuais, trocar configurações, instalar ou desinstalar softwares. O provedor de nuvem normalmente não tem qualquer direito de acesso à máquina virtual do cliente, limitando-se a monitorá-la com as ferramentas e APIs que o software monitor da máquina hospedeira lhe fornecer. O cliente também tem acesso ao sistema de gerência de nuvem, normalmente através de um portal, mas sua visibilidade de informações limita-se ao que for definido pelo provedor da nuvem. Diferentemente de uma máquina física alocada para o cliente, o mesmo não consegue acesso a nenhum parâmetro da máquina hospedeira.

Dado as opções de gerenciamento disponíveis para o usuário final, as capacidades de monitoração de identificação de problemas de rede e alocação de recursos são limitadas. Normalmente o usuário tem visibilidade somente da interface de rede virtual da máquina virtual que está executando, e não consegue

identificar problemas na interface física da máquina hospedeira, tais como congestionamento ou perda de pacotes.

O caso torna-se ainda mais complexo quando mais do que um provedor for utilizado, posto que as diferentes nuvens podem utilizar interfaces de programação de gerência com capacidades distintas e diferentes políticas de acesso. As diferenças nos parâmetros de cobrança e acordos de nível de serviço podem adicionar complexidade à gestão da nuvem, que na proposta deste trabalho é tratada através da utilização de lógica nebulosa.

3 Controlador Nebuloso

Um Controlador Nebuloso utiliza a Lógica Nebulosa[16][60] [61] para tratamento das variáveis medidas, do processo ou sistemas que se deseja controlar, e para geração das variáveis de saída, que atuarão neste sistema ou processo.

Especificamente no caso da gerência de redes, pode-se analisar a percepção do desempenho da rede a partir da perspectiva do usuário e como ele o descreve no caso da necessidade de abertura de um chamado técnico. Considerando que uma aplicação se comporta adequadamente para a maioria os usuários com um tempo de resposta de 200ms por transação, se a mesma em um dado momento estiver apresentando 300ms ou 100ms por transação, diferentes usuários terão diferentes opiniões sobre o desempenho da mesma aplicação. Neste aspecto alguns podem descrever a aplicação como lenta, para outros pode estar muito lenta, e para outros, normal. Então embora o SLA esperado da aplicação seja que ela demore 200ms por transação, pequenos desvios são tolerados sem impactar a percepção do usuário final.

3.1 Variáveis e Conjuntos Nebulosos

As variáveis nebulosas representam, através de valores linguísticos, os parâmetros diretamente medidos do sistema em seus valores numéricos e valores derivados. Estes valores linguísticos representam os conjuntos nebulosos e a associação dos valores numéricos medidos a estes conjuntos é feito através do processo de Fuzzificação.

Os conjuntos nebulosos representam os valores linguísticos atribuídos para a variável nebulosa. Os conjuntos são definidos por termos linguísticos, como Alto, Médio, Baixo, acrescido ou não de qualificadores como Pouco Alto ou Muito Baixo. Na Figura 5 apresenta-se uma representação gráfica dos conjuntos nebulosos atribuídos à alocação de consumo de CPU de uma máquina Linux. Considerando-se a alocação de CPU pode-se definir os conjuntos nebulosos, muito baixo (VERYLOW), baixo (LOW), médio (MEDIUM), alto (HIGH), muito alto (VERYHIGH). A representação gráfica destes conjuntos pode ser verificada na Figura 5.

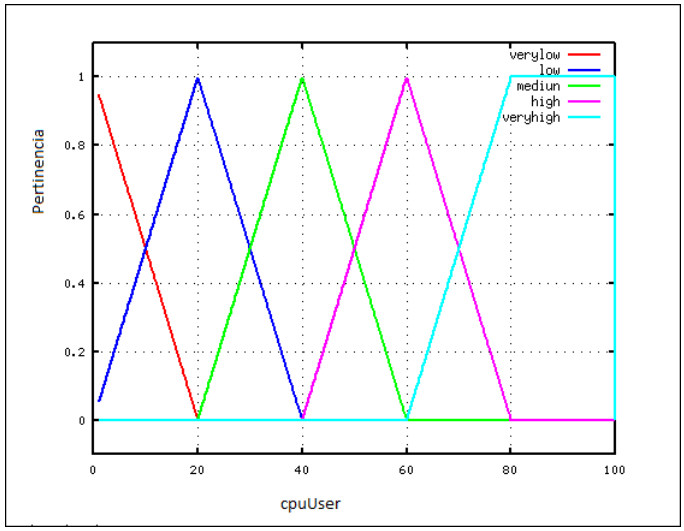


Figura 5 Conjuntos Nebulosos da alocação de CPU

Matematicamente o conjunto nebuloso e definido como uma coleção de pares, $(\mu_A(x),x)$ onde $\mu_A(x)$ é a função de pertinência que indica em qual grau x pertence ao conjunto A, dentro de um universo U. Na lógica booleana a pertinência $\mu_A(x)$ é dada por (6) , assumindo os valores 0 ou 1;

$$\mu_A(x)\begin{cases}0\ se\ x \notin A\\1\ se\ x \in A\end{cases} \qquad (6)$$

Na lógica nebulosa $\mu_A(x)$ pode assumir qualquer valor entre 0 e 1 de acordo com o grau em que *x* pertence ou não ao conjunto A. Na Figura 6 esta representada uma função de pertinência, sendo xi o valor numérico da variável x e $\mu_A(x_i)$ a pertinência deste valor ao conjunto nebuloso A.

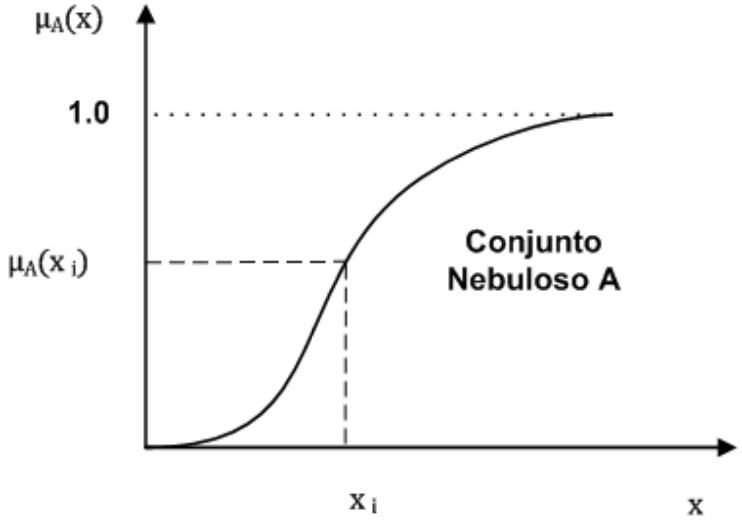


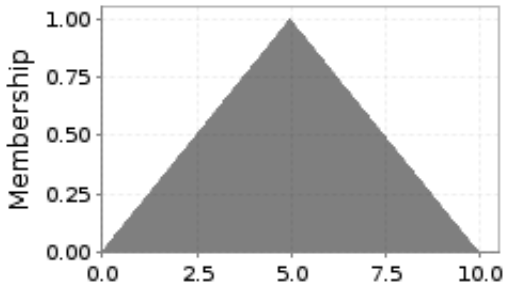
Figura 6 Função de pertinência

As funções de pertinência podem ser representadas por formas padronizadas como a triangular, a trapezoidal, a gaussiana e o conjunto unitário. A pertinência

pode ser definida por qualquer função matemática que represente o conjunto nebuloso e retorne os valores de pertinência a este conjunto no intervalo de 0 a 1. A seguir são apresentados alguns conjuntos nebulosos e suas formulas de calculo do valor de pertinência.

3.1.1 Função de pertinência triangular

A função triangular, representada graficamente na Figura 7 é definida pelo valor numérico no eixo x para os três vértices do triangulo,(a,b,c). Neste exemplo o conjunto seria definido pela tripla (0.0, 5.0, 10.0).

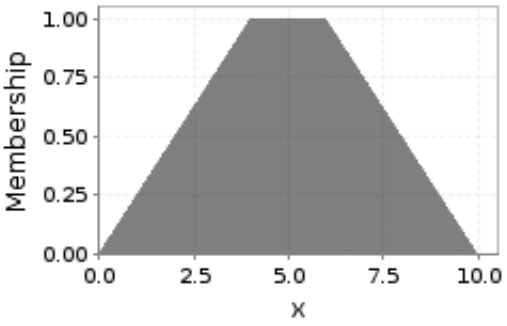


$$\mu(x) = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{b-a}, & a \leq x \leq b \\ \dfrac{c-x}{x-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases}$$

Figura 7 Função de pertinência triangular.

3.1.2 Função de pertinência trapezoidal

A função de pertinência trapezoidal, representada na Figura 8, seria definida pelos valores de x dos quatro vértices, do trapézio, dado pelos valores (a,b,c,d). No Exemplo da Figura 8, seria definido pela quádrupla, (0.0, 3.0, 6.0, 10.0).

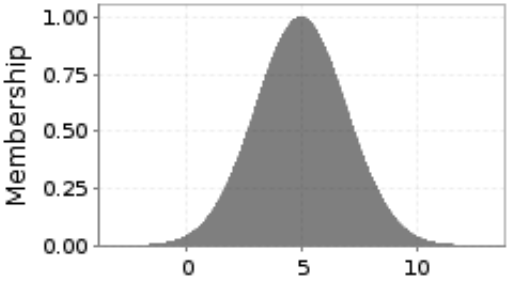


$$\mu(x) = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \dfrac{d-x}{x-c}, & c \leq x \leq d \\ 0, & d \leq x \end{cases}$$

Figura 8 Função de pertinência trapezoidal

3.1.3 Função de pertinência Gaussiana

A função gaussiana, representada graficamente na Figura 9, é definida pela média, m e pelo desvio padrão, σ. Neste exemplo é definido pela média com valor 5 e desvio padrão igual a 2.

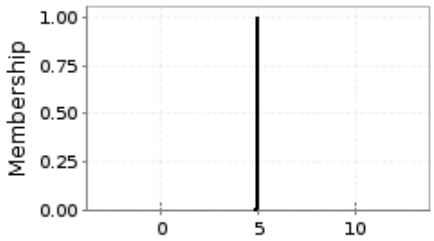


$$\mu(x) = e^{\frac{-(x-m)^2}{2\sigma^2}}$$

Figura 9 Função de pertinência gaussiana.

3.1.4 Conjunto Unitário.

A função de pertinência Conjunto unitário, representada na Figura 10, é definida por um único valor no eixo x,



$$\mu(x) = \begin{cases} 0, & x \neq a \\ 1, & x = a \end{cases}$$

Figura 10 Conjunto unitário.

3.2 Inferência de regras e Defuzzificação

O Processo de inferência de regras consiste na avaliação do conjunto de regras definidas para o controlador. As regras tem o formato IF *premissa* THEN *consequência*, e realizam a implicação dos conjuntos nebulosos de entrada para os conjuntos nebulosos de saída.

No processo de inferência todas as regras são avaliadas em paralelo e para controladores nebulosos não há encadeamento , ou seja o consequente de uma regra não é utilizado como antecedente de outra[62].

Através do processo de defuzzificação[63] o a variável resultante do processo de implicação é transformada do conjunto nebuloso resultante novamente para um valor exato que pode ser aplicado no processo ou sistema controlado pelo controlador nebuloso. Na Figura 11 esta representado o resultado de um processo de implicação de regras através da pertinência a cada conjunto da variável de saída. A variável usada é a variável “tip”, do programa exemplo TestTipper.java da biblioteca JFuzzyLogic[64].

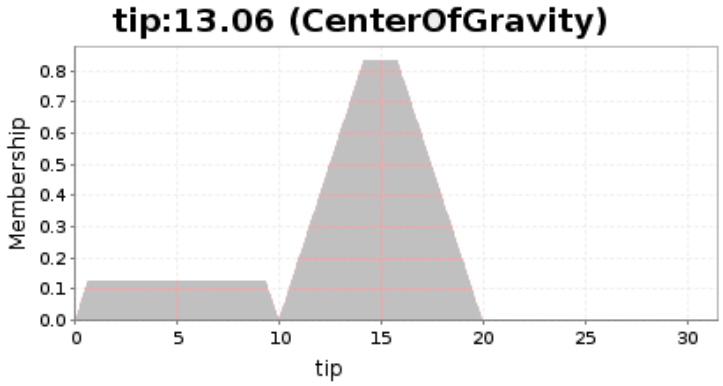


Figura 11 Defuzzificação (adaptado do exemplo TestTipper.java)

Os metodos utilizados para defuzzificação pode variar dependendo da aplicação esta documentada na literatura varios estudos para seleção do metodo adequado [65][63], [66]. Na Tabela 1estão listados dois métodos comumente utilizados e suas respectivas formulas de calculo.

Tabela 1 Métodos de defuzzificação.

Centro de Gravidade (Center of Gravity –COG)	$Y = \frac{\int_{min}^{max} \mu(x) \cdot x dx}{\int_{min}^{max} \mu(x) dx}$
Centro de Gravidade para conjuntos unitários (Center of Gravity Singletons – COGS)	$Y = \frac{\sum_{i=1}^{n} \mu_i(x) \cdot x}{\sum_{i=1}^{n} \mu_i(x)}$

3.3 Controlador baseado em lógica nebulosa

A lógica nebulosa é amplamente empregada para criação de controladores de processos[67][62]. Como descrito em [27], a lógica nebulosa permite agregar o conhecimento humano na solução de controle. O controlador nebuloso pode atuar diretamente sobre o processo o sistema sendo controlado, como representado no diagrama da Figura 12.

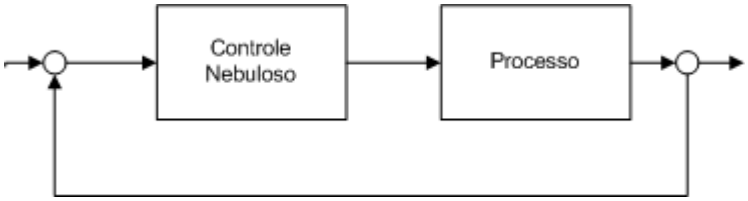


Figura 12 Controle direto de processos

O sistema de controle funciona através da coleta de variáveis do sistema controlado, que passam pelo processo de Fuzzificação pelo qual os valores exatos são convertidos para os valores linguísticos dos conjuntos nebulosos. A estes valores são aplicados às regras do sistema de inferência. Na saída destes sistemas as variáveis de controle resultantes passam pelo processo de Defuzzificação através do qual, as variáveis nebulosas são convertidas para valores exatos que são alimentados ao sistema ou processo controlado. O diagrama da Figura 13, contém a representação de um controlador nebuloso, que alimentado com parâmetros de um ambiente de computação, e tem como saídas os indicadores de condição de operação.

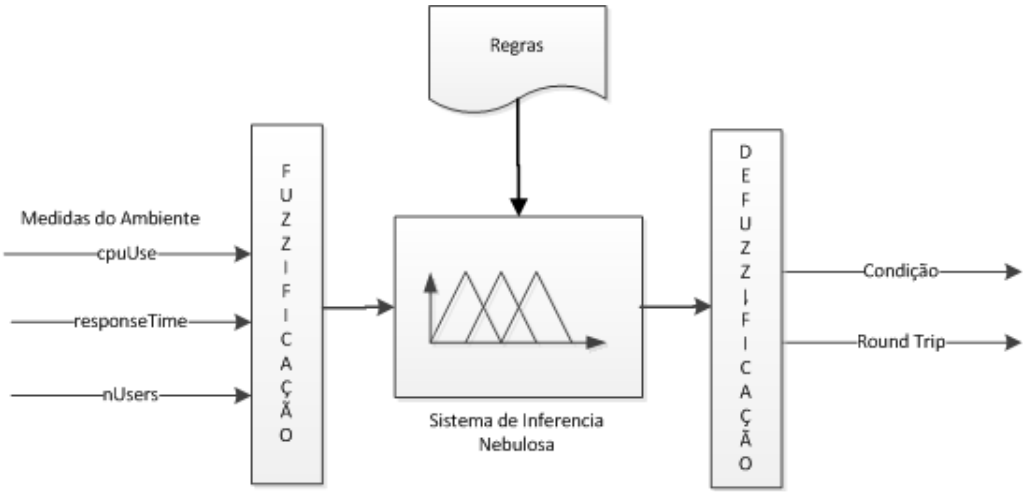


Figura 13 Sistema de Controle Nebuloso

4 Máquina de Estados Nebulosos

Além do gerenciamento das dos parâmetros instantâneos medidos de um sistema em nuvem, como tempo de resposta, consumo de CPU e desempenho de rede, é necessário o gerenciamento das sequencias de eventos associados a operação do sistema..

As máquinas virtuais providas através do modelo IaaS tem um ciclo de vida [68] que incluem vários estados e sequências específicas de ativação ou eliminação de instâncias. A máquina virtual a se iniciada em uma nuvem, passa pelo estado de inicialização , que pode ou não incluir transferência da imagem da máquina virtual para a plataforma que provê a nuvem. A máquina após o processo de inicialização passa ao estado em que esta executando as aplicações do cliente. A Figura 14 traz uma representação desta sequência de estados, adaptada do ciclo de vida de máquina virtual apresentado em [68] .

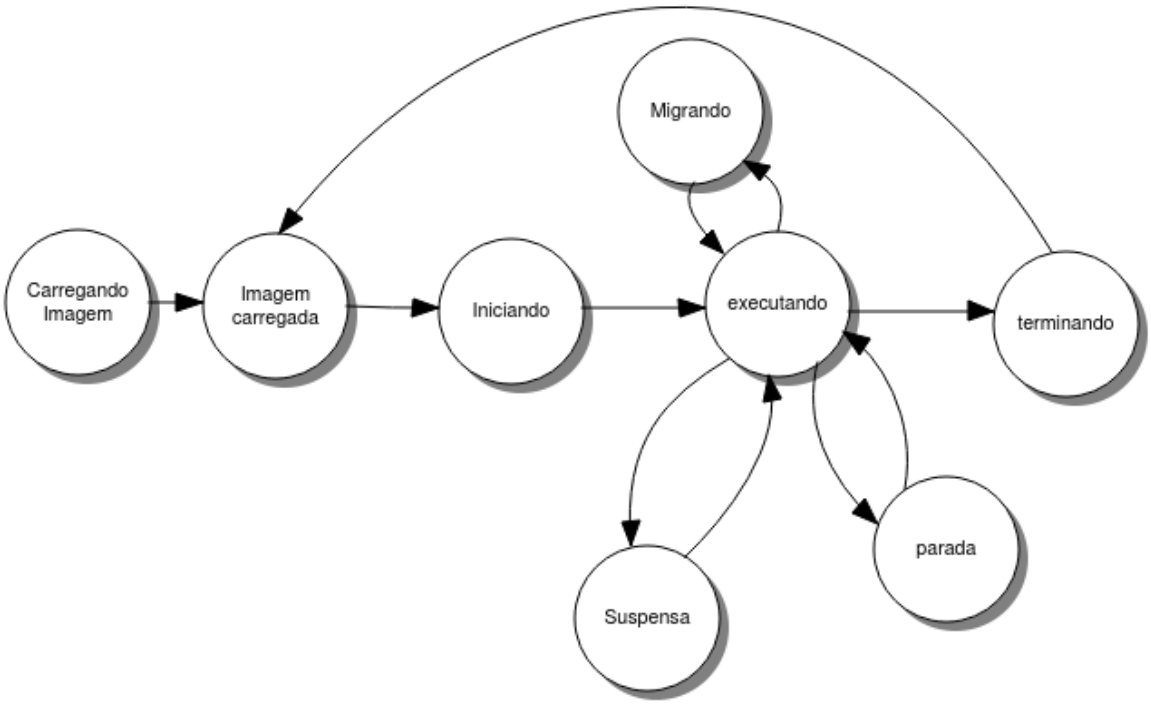


Figura 14 Ciclo de vida de uma máquina virtual

Uma Máquina de Estados Finitos[69], é dado pela quíntupla $FSM=(S,I,O,t,g)$ sendo S,I,O, respectivamente, os conjunto de estados, entradas e saídas. Sendo que t é a função de transição entre estados e g é a função geradora das saídas.

4.1 Máquina de Estados Finitos Nebulosos

Uma Máquina de Estados Finitos Nebulosos[27], difere de uma máquina de estados tradicional pela característica de que cada estado tem um grau de pertinência continuo no intervalo de 0 a 1. Desta forma a máquina pode estar parcialmente em mais que um estado ao mesmo tempo.

As máquinas de Estados Nebulosos são aplicadas na modelagem de emoção em personagens de jogos eletrônicos conforme demonstrado em [70] [71]. Através da máquina nebulosa, mais de um estado pode estar ativo em um dado momento e com diferentes graus pertinência.

Como definido em [72], a Máquina de Estados Nebulosos pode ser definida como $FFSM = (I,O,S,T,g)$ sendo I, O e S são, respectivamente, os conjuntos não vazios de entradas, saídas e estados , T é o conjunto de transição e g a função que gera as saídas. Na Figura 15 pode-se verificar a representação de uma máquina de estados nebulosos, com dois estados Si e Sj, sendo $\boldsymbol{t_i^j}$, a transição do estado i para o estado j e $\boldsymbol{t_j^i}$ a transição do estado j para o estado i. As transições de um estado para outro são governadas por regras nebulosas, de acordo com as entradas e a pertinência do estado origem.

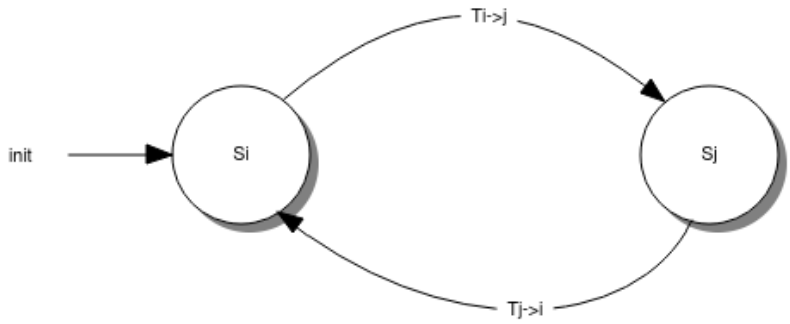


Figura 15 Máquina de Estados Nebulosos

Considerando o estado S_i, a função $\mu(S_i)$ representa a pertinência da máquina com relação a este estado. Algumas restrições devem ser atendidas, sendo a primeira delas que a somatória da pertinência de todos os estados em um dado instante de tempo *t* seja igual a 1, conforme expressado por (7). Esta condição é satisfeita para uma máquina de estados finitos na qual em um dado momento somente um estado esteja ativo, tendo, portanto pertinência igual a 1. Todos os demais estados estarão inativos e, tendo, portanto, pertinência igual a 0.

$$\sum_i \mu(S_i) = 1 \qquad (7)$$

Da mesma forma o somatório da atividade de todas as transições, t_i^j, pertencente ao universo de transições T,a partir dos estados S_i, também será 1, como expressado em (8) .

$$\sum_{t_i^j \in T} \mu(t_i^j) = 1 \qquad (8)$$

Esta restrição, se aplicada ao caso particular de uma máquina de Estados Finitos, é plenamente atendida, posto que a pertinência de uma transição é 1, e as demais 0.

Em [70], para atender (8), utiliza-se uma normalização dos valores de transição. Para o caso no qual a somatória de todas as atividades de transição ultrapasse o valor 1, tem-se que a pertinência normalizada de cada transição, $\mu'(t_i^j)$, é dada por (9).

$$\mu'(t_i^j) = \frac{\mu(t_i^j)}{\sum_{t_i^n \in T_j} \mu(t_i^n)} \qquad (9)$$

Para o caso no qual a somatória das pertinências de todas as transições t_j^k a partir de um estado S_j é menor que 1, o valor remanescente deve continuar no estado originador das transições. Desta forma em[70] define-se que a pertinência do estado no instante seguinte, $\mu'(S_j)$, é dada por (10).

$$\mu'(S_j) = \mu(S_j) \cdot \left(1 - \sum_{t_j^k \in T_j} \mu(t_j^k)\right) + \sum_{t_i^n \in T_j} \mu(t_i^j) \cdot \mu(S_i) \qquad (10)$$

Na equação (10) tem-se a contribuição para geração da nova pertinência do estado S_j, expressada por $\mu'(S_j)$, somando-se as contribuições das transições de outros estados S_i para S_j assim como a contribuição do remanescente das transições originadas no estado S_j .

Para o caso de a pertinência combinada de todas as transições exceder 1 aplica-se (9) para a normalização das transições em (10). Desta forma a expressão para cálculo da pertinência, $\mu'(S_j)$é calculada por (11).

$$\mu'(S_j) = \mu(S_j) \cdot \left(1 - \sum_{t_j^k \in T_j} \frac{\mu(t_j^k)}{\sum_{t_i^n \in T_j} \cdot \mu(t_i^n)}\right) + \frac{\sum_{t_i^n \in T_j} \cdot \mu(t_i^j) \cdot \mu(S_i)}{\sum_{t_i^n \in T_j} \cdot \mu(t_i^n)} \quad (11)$$

A partir de (10) e (11), criou-se o método de cálculo de pertinência para cada estado no instante de tempo t+Δt , sendo Δt o período no qual a máquina de estados nebulosos, tem suas transições avaliadas.

Apesar de não estar representada no diagrama Figura 15 a transição do estado S_i para ele próprio, fica implícita uma transição de cada estado para ele mesmo, quando a somatória das pertinências de todas as transições deste estado para outro for igual a 0..

5 Modelo de Controlador Nebuloso

Neste capitulo é discutido o modelo de controlador proposto para tratamento das variáveis medidas e o tratamento aplicado às mesmas. O modelo considera que as variáveis coletadas passam pelo tratamento de um controlador Nebuloso que gera como saída um ou mais indicadores de condição de operação da nuvem monitorada.

Este controlador é formado por um modulo de coleta de dados, um módulo de pré tratamento , uma máquina de estados nebulosos e uma base de regras.

O modelo aplica-se a uma solução de Computação em Nuvem contratada na modalidade infraestrutura como serviço. Embora seja proposto que os controles possam ser extrapolados para modalidades de plataforma como serviço e software como serviço, as mesmas ficaram fora do escopo deste trabalho.

Para gerenciar o sistema no modelo IaaS, foi proposto um controlador composto por uma máquina de inferência e uma máquina de estados nebulosos representado na Figura 16. Este controlador, a partir das entradas medidas do sistema gerenciado e de uma base de regras, tem como saídas alarmes, indicadores e ações através da API de controle da nuvem.

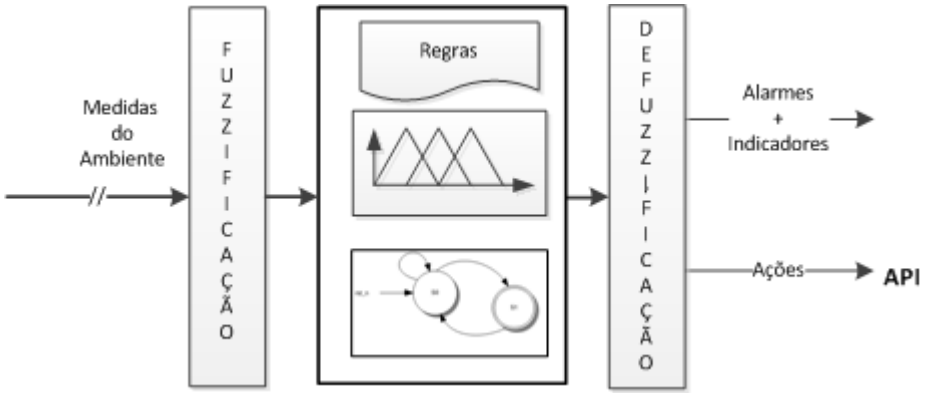


Figura 16 Controlador Nebuloso com Máquina de Estados Nebulosos

O modelo de controlador possui um conjunto de variáveis de entrada *I*, um conjunto de estados *S*, com um conjunto de transições *T* e um conjunto de ações *A*. Cada transição t_i^j possui um conjunto de regras *R*. As ações do conjunto *A* interagem diretamente com a interface de programação da solução de implementação da nuvem.

A nuvem será modelada pelos parâmetros utilizados para contratação do sistema junto ao provedor de Computação em Nuvem. No Caso de uma solução IaaS, estes parâmetros são dados por número de máquinas virtuais em execução por período de tempo, dados trafegados total, e dados armazenados por período de tempo.

Para controle do sistema foi proposto um controlador nebuloso baseado em máquina de estados nebulosos. Este controlador é composto por múltiplos módulos. O primeiro módulo, a partir das entradas do sistema, pode acionar alarmes e mesmo ações corretivas e de ajuste no ambiente. Este módulo realiza um pré-tratamento das variáveis de entrada. Outro módulo contém uma máquina de estados Nebulosos, que permite além de controlar a sequencia dos estados a incorporação de temporizadores também aderentes ao conceito nebuloso. O controlador foi escrito em linguagem Java utilizando a biblioteca jFuzzyLogic [73][64]. Através do uso desta biblioteca toda descrição, das variáveis, dos conjuntos nebulosos e as regras de inferência podem ser descritos na linguagem Fuzzy Control Language ou FCL[74], permitindo a rápida reconfiguração do sistema e habilitando a interface com outros sistemas, aplicativos ou bibliotecas que usem este padrão.

5.1 Variáveis

As variáveis utilizadas para gerência da solução em Nuvem, em sua maioria são coletadas por SNMP[75], ou por medida direta da aplicação. Utilizou-se somente informações que possam ser obtidas pelo usuário final sem a necessidade de nenhuma configuração específica do lado do provedor de Computação em Nuvem. Informações relacionadas a chamados técnicos por falha ou lentidão são obtidos de sistema de Atendimento ou relatório gerado a partir deste. Foi considerado que estes dados serão diretamente acessíveis de um banco de dados SQL que contenha as informações em tempo real. As variáveis e todos os conjuntos nebulosos foram representados pelos termos linguísticos em inglês.

5.1.1 Variável Latência

Média da latência da rede medida entre o roteador de saída da rede corporativa onde estão os usuários e o roteador de borda do provedor de solução em nuvem ou a interface externa do conjunto de servidores utilizado pela aplicação.

A latência foi dividida em três conjuntos BAIXO, MÉDIA e ALTA, que estão representados na Figura 17.

Tabela 2 Conjuntos nebulosos da variável *rtt.*

Conjunto	Função	valores
LOW	Trapézio	(0 , 0 , 10 , 30)
HIGH	Triangulo	(20 , 30 , 100, 110)
VERYHIGH	Trapézio	(100 , 120 , 250 , 250)

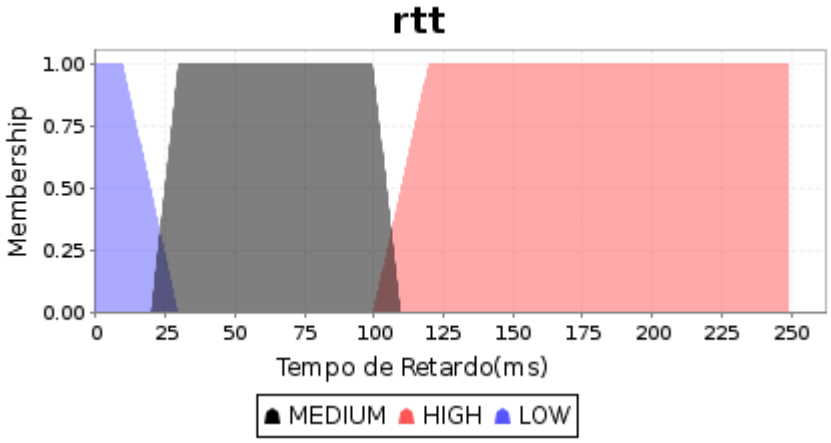


Figura 17 Conjuntos do Tempo de Retardo

Nos experimentos a latência foi emulada em uma máquina virtual Linux atuando como roteador, através do pacote NETEM [76][77],sendo aplicados valores de atraso simétricos nos pacotes saindo das duas interfaces do roteador. O tempo de retardo *rTT* é composto pelo retardo original da rede *rTTorg* somado ao dobro do atraso *Tatraso* somado a cada interface do roteador acrescido de uma variação randômica *Tvar*.

$$rTT = rTTorg + 2 \cdot Tatraso \pm Tvar \quad (12)$$

Desta forma é a latência é controlada para fins de experimentação.

Para aplicações altamente iterativas baseadas no estabelecimento de sessão a alta latência pode ter impacto significativo no desempenho experimentado pelo usuário final.

5.1.2 Variável Tempo de Resposta

O tempo de resposta é medido diretamente pela aplicação. Este tempo é definido como o período decorrido entre a máquina cliente enviar uma requisição ao servidor e a chegada da resposta a esta requisição.

O SLA definido para o Tempo de Resposta estabelece que a média deste dever ser inferior a um certo intervalo de tempo por transação. Em função disso, foram estabelecidos os conjuntos Baixo (LOW), Alto (HIGH) e Muito Alto (VERYHIGH) representados na Figura 18.

Tabela 3 Conjuntos nebulosos da variável *respTime*

Conjunto	Função	valores
FEW	Trapézio	(0 , 0 , 3 , 4)
MANY	Triangulo	(3 , 5 , 7)
TOOMANY	Trapézio	(6 , 7 , 10 , 10)

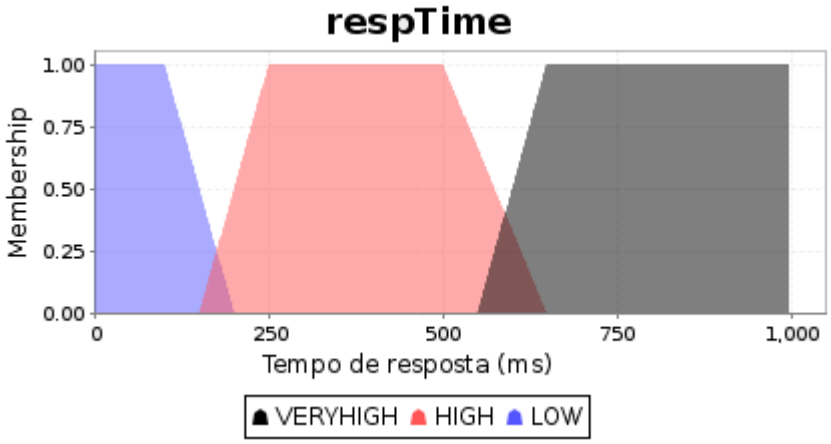


Figura 18 Tempo de Resposta de Aplicação.

O tempo *respTime* é dado pela soma dos tempo de tráfego dos pacotes da requisição, de processamento pelo servidor , e de tráfego dos pacotes da resposta. Em uma aplicação que utiliza somente recursos um único servidor o tempo de resposta, *respTime*, é dado por (13), onde *rtt* indica o tempo de ida e volta de um pacote, entre a máquina cliente e o servidor, e *Tproc*, representa o tempo de processamento da requisição.

$$respTime = rtt + Tproc \tag{13}$$

Em soluções complexas nas quais o servidor de aplicação depende dos resultados de outros servidores, como, por exemplo, quando um servidor de aplicação precisa consultar um servidor de banco de dados, que por sua vez utiliza serviços de um servidor de arquivos, o tempo total é dado por (14), que inclui a contribuição de cada componente de retardo rtt_i e de processamento $Tproc_i$.

$$respTime = \sum_{i=1}^{n} rtt_i + Tproc_i \tag{14}$$

O tempo de resposta é um parâmetro de projeto da solução disponibilizada, a sua alteração durante a operação do sistema é resultado de

alterações em outras variáveis tais como o tempo de retardo entre a rede onde estão localizados os clientes e a rede do servidor. O aumento de carga no servidor, que leva ao aumento do tempo de processamento *Tproc,* também altera o tempo de resposta, conforme descrito por 13. A variação de *respTime*, deve ser analisada em conjunto com outros parâmetros para determinar a causa raiz da piora do desempenho da aplicação e definir uma ação correta de mitigação.

5.1.3 Variável Consumo de CPU

A variável cpuUser representa o consumo de CPU realizado pela aplicação e suas funções de suporte. Porcentagem do tempo da *vCPU* alocado à aplicação do usuário. Este valor representa a taxa de ocupação de CPU de cada máquina, incluindo nesta alocação também todo processamento do Sistema Operacional dedicado a servir à aplicação fim.

Tabela 4 Conjuntos nebulosos da variável c*puUser.*

Conjunto	Função	valores
VL	Triangulo	(0 , 0 ,20)
L	Triangulo	(0 , 20 , 40)
M	Triangulo	(20 , 40 , 60)
H	Triangulo	(40 , 60 , 80)
VH	Trapézio	(60 , 80 , 100 , 100)

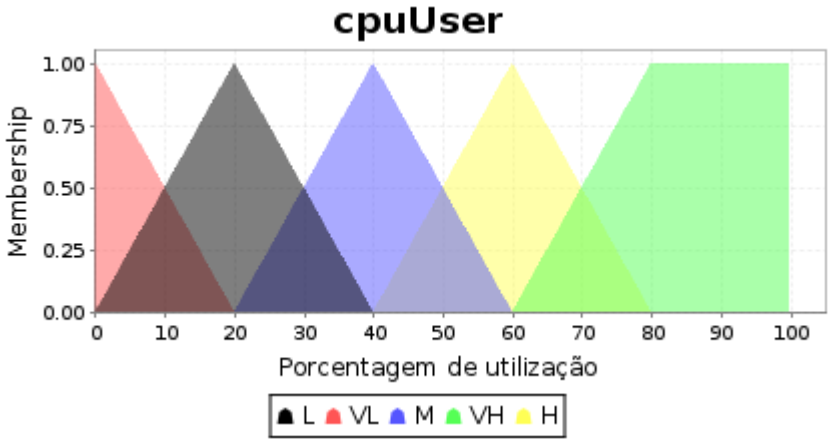


Figura 19 - Consumo de CPU.

Este parâmetro mensura a ocupação de cada máquina individual.

Uma alternativa ao uso da alocação de CPU seria uma métrica que inclua o consumo de outros recursos, tais como memória e redes. Em [78] propõe-se uma métrica multi dimensional dada por

$$Vol = \frac{1}{(1-cpu)} \cdot \frac{1}{(1-net)} \cdot \frac{1}{(1-mem)} \quad (15)$$

Em [79] os autores propõem o uso de um vetor de fração de carga, $NodeLoadFracVec$ dado por (16), para cada elemento da rede u. Sendo $cpuU$ a quantidade ciclos de CPU utilizados por período, a ser dividida por $cpuCap$ que indica a quantidade de ciclos disponíveis no mesmo período . A variável $memU$ indica a quantidade de memória utilizada , a ser dividida por $memCap$ que indica a capacidade de memória total disponível no sistema. A variável $netU$ indica a quantidade de dados trafegados a ser relacionado com a banda total da interface $netCap$. A variável ioU representa o número de operações de acesso ao espaço de armazenamento,a ser dividido pelo número máximo de operações de entrada e saída suportadas pelos sistema analisado, dado por $ioCap$.

Esta proposta cobre não somente os servidores, mas também os nós de rede. Para cada elemento considerado é criado um vetor de frações que assumem valores entre 0 e 1.

$$NodeLoadFracVec(u) = \left[\frac{cpuU}{cpuCap}, \frac{memU}{memCap}, \frac{netU}{netCap}, \frac{ioU}{ioCap}\right] \quad (16)$$

Para a mensuração de carga de equipamentos de armazenamento, utiliza-se (17). Adicionalmente aos valores de número de operações de entrada e saída, ioU e $ioCap$, são considerados os valores representativos de espaço de armazenamento utilizado $spaceU$ e espaço total disponível $spaceCap$.

$$NodeLoadFracVec(u) = \left[\frac{spaceU}{spaceCap}, \frac{ioU}{ioCap}\right] \quad (17)$$

No trabalho apresentado nesta dissertação, também foi utilizado, o mesmo método para dimensionar a carga do servidor. Entretanto cada elemento do vetor é considerado em separado, mediante o uso de lógica nebulosa.

5.1.4 Variável Chamados de Lentidão

Casos abertos no sistema de chamados de suporte da aplicação que têm como motivo de abertura a lentidão da aplicação. Estes casos servem como parâmetro para se conhecer a percepção do usuário indicando que as métricas estabelecidas não estão ajustadas.

Tabela 5 Conjuntos nebulosos da variável *slowTT.*

Conjunto	Função	valores
FEW	Triangulo	(0,0,3)
MANY	Trapézio	(2,4,5,7)
TOOMANY	Trapézio	(6,8,25,25)

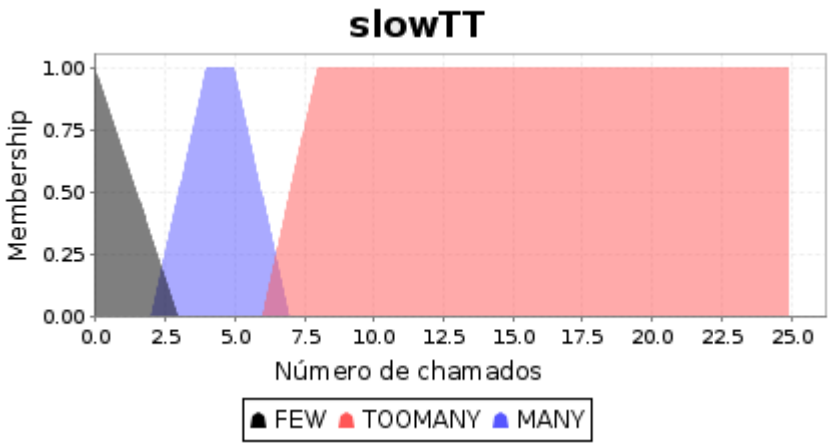


Figura 20 Chamados com causa Lentidão.

Esta métrica pode servir como identificação de deterioração do desempenho da aplicação, posto que uma vez acostumado com desempenho melhor o usuário identifica piora deste como lentidão.

5.1.5 Variável Condição da Nuvem

Definida como um valor indicativo da condição de uso da nuvem onde as máquinas virtuais estão hospedadas, que pode ser usado para definição dos estados a serem utilizados no controle da solução. Os conjuntos nebulosos desta variável, definidos na Tabela 6 e representados na Figura 21, foram definidos como, Confortável (CONFORTABLE), Aceitável (ACCEPTABLE) e Inaceitável (UNACCEPTABLE).

Tabela 6 Conjuntos nebulosos para condição de nuvem .

Conjunto	Função	valores
CONFORTABLE	Trapézio	(0 , 0 , 0.25 , 0.5)
ACCEPTABLE	Triangulo	(0.25 , 0.5 , 0.75)
UNACCEPTABLE	Trapézio	(0.5 , 0.75 , 1 , 1)

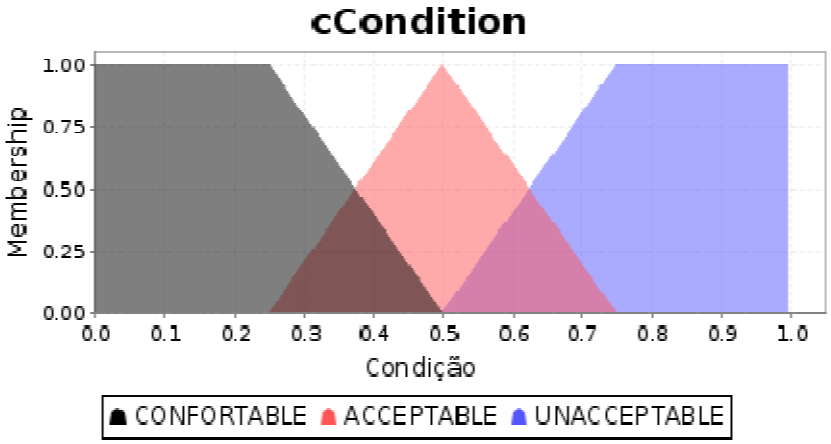


Figura 21 Condição da Nuvem

Esta variável pode ser múltipla de acordo com diferentes aspectos que se deseja controlar. Em uma aplicação sensível a consumo de energia pode-se utilizar a variável relativa a eficiência energética, que leve em conta o gasto de energia utilizado pela solução. Esta medida pode ser obtida agregando as variáveis de entrada medidas como consumo de energia e temperatura do centro de dados onde está hospedada a Nuvem Privada.

5.1.6 Variável Número de máquinas virtuais ativas.

A variável *nVms* indica o número de máquinas virtuais ativas na nuvem em um determinado instante de tempo. Os conjuntos nebulosos para esta variável, com valores definidos na Tabela 7 e representados na Figura 22, foram definidos como poucas (FEW), muitas (MANY), demasiadas (TOOMANY).

Tabela 7 Conjuntos nebulosos da variável *nVms.*

Conjunto	Função	valores
FEW	Trapézio	(0,0,3,4)
MANY	Triangulo	(3,5,7)
TOOMANY	Trapézio	(6,7,10,10)

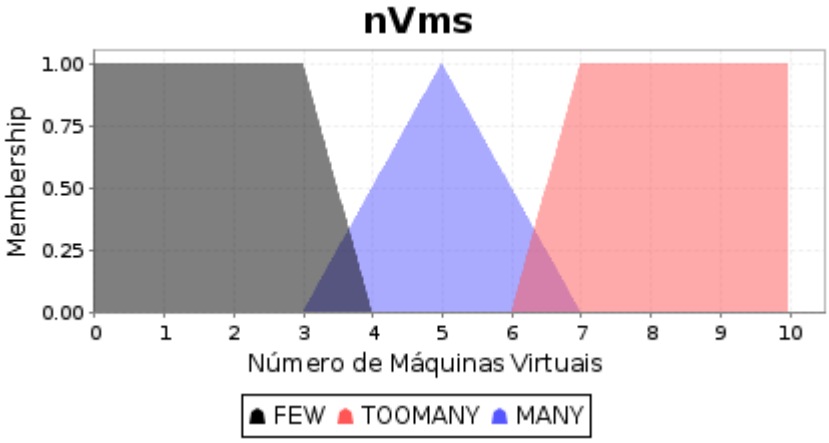


Figura 22 Conjuntos nebulosos da variável número de máquinas virtuais.

No modelo discutido foram utilizadas até 10 máquinas virtuaismas este valor e os conjuntos correspondentes devem se definidos de acordo com o número de máquinas virtuais necessárias para o suporte da base de usuários da aplicação.

5.1.7 Variável *users*

A variável *users* representa o número de usuários utilizando simultaneamente a aplicação em nuvem em um dado instante de tempo *t*. Os conjuntos nebulosos definidos para esta variável, listados na tabela 8 e representados na Figura 25, foram definidos em Poucos(FEW), Muitos(MANY) e Demasiados(TOOMAY).

Tabela 8 Conjuntos nebulosos da variável users.

Conjunto	Função	valores
FEW	Trapézio	(0,0,90,120)
MANY	Triangulo	(90,150,210)
TOOMANY	Trapézio	(180,210,300,300)

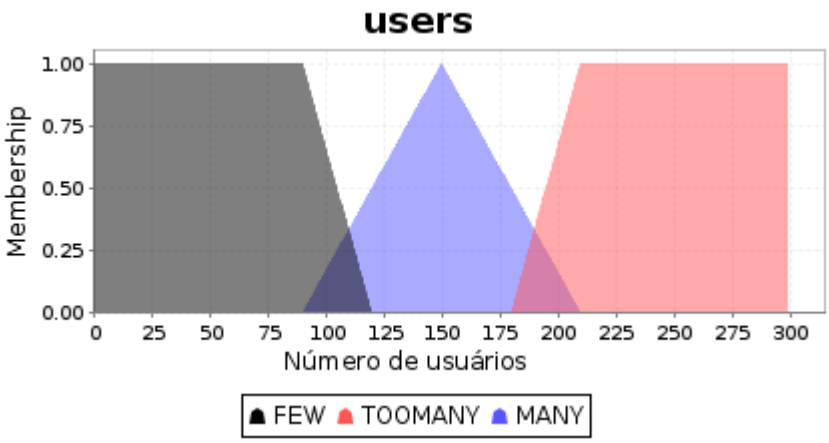


Figura 23 Conjuntos nebulosos da variável *action*

A variável *users*, guarda relação com o número de máquinas virtuais, *nVms*, onde neste caso uma máquina virtual pode suportar adequadamente 30 usuários concorrentes.

5.2 Variáveis auxiliares

As variáveis auxiliares são utilizadas pelo modelo para controlar as transições entre os estados.

5.2.1 Variável *deltaT*

A variável *deltaT* representa o tempo decorrido entre a ocorrência de um evento no sistema e o instante de tempo atual. Esta variável é independente para cada estado o tempo inicial é dado pelo instante no qual a pertinência ao estado sai do valor 0. O tempo é medido em segundos e os conjuntos, definidos na Tabela 9, estão representados na Figura 24.

Tabela 9 Conjuntos nebulosos da variável deltaT.

Conjunto	Função	valores
LOW	Trapézio	(0,0,10,20)
HIGH	Triangulo	(15,25,35)
VERYHIGH	Trapézio	(30,50,100,100)

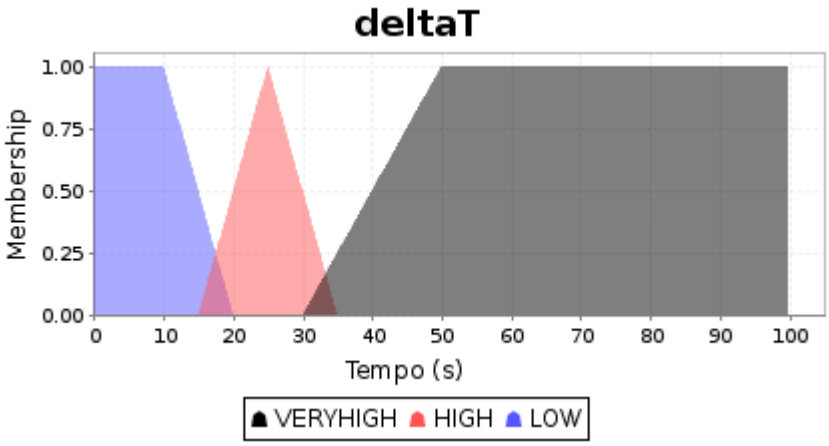


Figura 24 Conjuntos nebulosos da variável deltaT

A implementação realizada neste trabalho, utiliza somente uma definição de conjuntos nebulosos para as variáveis *deltaT*, comum para todas as transições. Posto que cada transição, utiliza um sistema de inferência nebuloso próprio, é possível a atribuição de diferentes conjuntos nebulosos para cada transição.

5.2.2 Variável *action*

A variável *action* representa a ação de transição. Os conjuntos nebulosos definidos para esta variável, representados na Figura 25, são dois “singletons”, um em 0 e outro em 1, representado respectivamente as ações de permanência no estado atual, e transição para o próximo estado.

Figura 25 Conjuntos nebulosos da variável *action*

Como método de defuzzificação foi utilizado o Centro de Gravidade de Conjuntos Unitários.

5.2.3 Alocação de Recursos

A variável *resourceAlloc*, dada com um valor numérico de 0.0 a 1.0 representa a alocação de recursos na nuvem. Os conjuntos foram definidos como Subprovisionado (UNDERPROVISIONED), Adequado (ADEQUATE) e Superprovisionado (OVERPROVISIONED). Definidos na Tabela 10 e representados na Figura 26.

Tabela 10 Conjuntos nebulosos da variável resourceAlloc.

Conjunto	Função	valores
LOW	Trapézio	(0 , 0 , 0.25 , 0.5)
HIGH	Triangulo	(0.25 , 0.5 , 0.75)
VERYHIGH	Trapézio	(0.5 , 0.75 , 1 , 1)

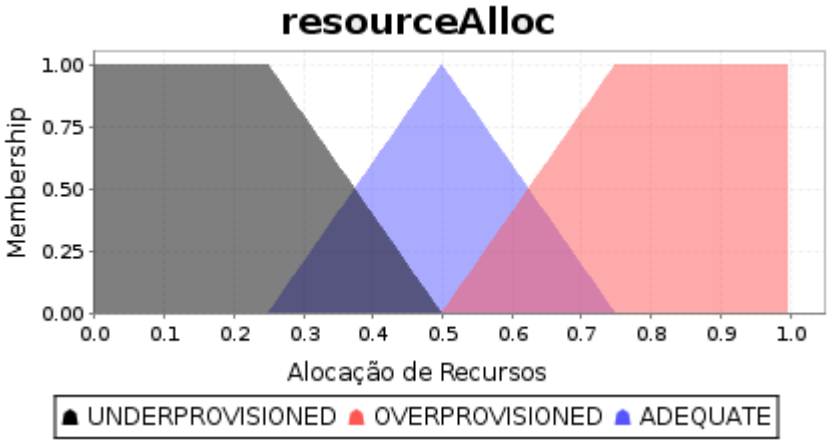


Figura 26 Conjutos nebulos para alocaçao de recursos

A variável *resouceAlloc* foi definida como um valor real, variando de 0 a 1, mas para melhor representatividade pode ser associado a uma relação entre os recursos alocados, por exemplo máquinas virtuais, e usuários concorrentes. Esta opção não foi investigada neste trabalho.

5.3 Definição de Regras e conjuntos

As regras e os conjuntos nebulosos de cada variável são derivados do contrato de SLA estabelecido para o serviço entregue aos usuários e não somente do SLA formal definido no contrato de serviço do provedor externo.

O gestor de tecnologia responsável pela contratação dos serviços em nuvem é o responsável perante os usuários finais do desempenho do serviço final que é entregue a estes, atuando como seu provedor de serviço interno[80] sendo a interface entre eles e o provedor de serviço externo. Este processo pode ter sua complexidade aumentada quando o serviço final é composto por soluções de

mais de um provedor. Normalmente quando um provedor combina solução de outros provedores para ofertar uma solução a um cliente final, ele é a única interface que este cliente possui, se responsabilizando perante este por toda violação de serviço ocorrida, independente da parte causadora. No caso do serviço de nuvem ser implementado pelo próprio usuário final, utilizando, por exemplo, uma plataforma PaaS de um provedor e plataforma IaaS de outro , ou mesmo utilizando diferentes modalidades de IaaS, toda responsabilidade sobre a operação do serviço fim a fim recai sobre o time interno do próprio cliente.

As SLA definidas em contrato normalmente expressam os valores limites para uma determinada variável. Para a gestão é necessário que o sistema utilize pontos de aviso anteriores a degradação completa do sistema que não permita atuação para correção.

Tabela 11 Descrição do SLA contratado

Estado	Descrição
Disponibilidade	Disponibilidade de 99.9%
Tempo de resposta	Abaixo de 200ms .
Tempo de resposta Máximo	Abaixo de 800ms .
Tempo de Recuperação após Falhas	2 horas
Banda Servidor x Cliente	2M
Banda Cliente x Servidor	2M

5.4 Coleta de Variáveis

Para coleta das variáveis foi desenvolvido módulos em Java que realizam a coleta de dados através de SNMP, e ou HTTP. Estes módulos permitem a customização da coleta de dados para se adequar aos experimentos a serem realizados.

Tabela 12 Coleta de Variáveis

Variável	método	Fonte
cpuUser	SNMP	Medido diretamente da máquina virtual.
Mem	SNMP	Medido diretamente da máquina virtual
respTime	HTTP	Obtido da aplicação “in-band” ou através de aplicação de teste.
slowTT	SQL	Requisição SQL diretamente na base de dados do sistema de atenção ao Usuário
nVMs	HTTP	Obtido da API de controle da nuvem
nUsers	SQL/HTTP	Obtido da API ou do sistema de aplicação

Todos os dados são armazenados em uma base de dados SQL, utilizando um banco de dados java Apache Derby [81]. Todos os dados medidos juntamente com as variáveis processadas são armazenados no banco de dados para posterior análise e utilização como fonte de dados para simulações de cenários.

5.5 Máquina de Estados Nebulosos

Para tratamento da sequência temporal foi utilizado uma máquina de estados finitos nebulosos, conforme apresentado no capitulo 3. Esta máquina representada na Figura 27 define o comportamento do controlador seguindo uma sequência definida. Esta máquina utiliza como entradas as variáveis medidas do ambiente e tem como saídas, indicadores de estado e ações corretivas. A lógica de ativação desta máquina representa uma prova de conceito e não busca ser exaustiva no estabelecimento de estados e regras aplicáveis.

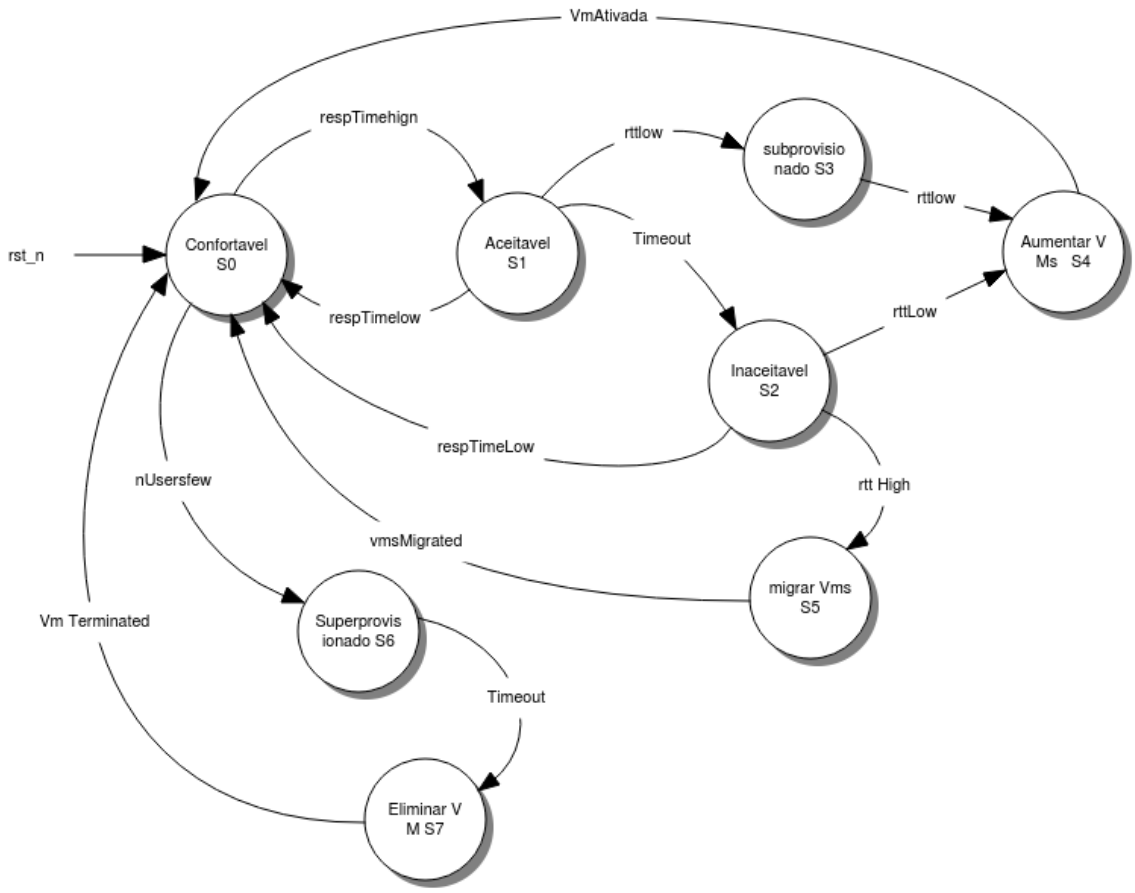


Figura 27 Máquina de Estados Nebulosos

A máquina de estados nebulosos baseia-se nos seguintes requisitos de operação descritos de forma textual:

Em condições normais de operação a máquina de estados se encontra no estado Confortável.

Caso o número de máquinas virtuais seja superior ao necessário para desempenho confortável, a máquina transitará para um estado de alocação excessiva de recursos e se permanecer por muito tempo neste estado deverá reduzir o número de máquinas virtuais utilizadas.

Caso o tempo de resposta da aplicação aumente para valores fora do SLA , a máquina de estados deve transitar para o estado Aceitável e neste estado permanecerá, dependendo do tempo que a deterioração durar, e da intensidade do aumento no tempo de resposta.

Para remediar o alto tempo de resposta dependendo do número de usuários e máquinas virtuais alocadas, e do retardo entre cliente e servidores de nuvem, o controlador pode decidir por aumentar os recursos ou realizar migração para provedor alternativo.

Estes requisitos foram traduzidos na máquina de estados da Figura 27 contendo os estados, Confortável, Aceitável, Inaceitável, Sub provisionado, Super provisionado , e os estados responsáveis por ação aumentar máquina virtuais (Aumentar VM) , migrar máquinas virtuais (Migrar VM) , e eliminar máquinas virtuais (Eliminar VM) . Os estados estão descritos na Tabela 13.

Tabela 13 Descrição dos Estados

Estado		Descrição
Confortável	S0	Estado de operação em regime da solução com performance e alocação de recursos nominais.
Aceitável	S1	Estado com violação de parâmetro de performance, mas ainda dentro de padrões aceitáveis.
Inaceitável	S2	Estado com violação continuada de parâmetros de performance da solução.
Sub provisionado	S3	Estado em que a alocação de recursos corrente da infraestrutura não permite atingir as níveis de performance esperado da solução.
Aumentar VM	S4	Estado no qual o sistema de gerência realiza o aumento dos recursos alocados a solução.
Migrar VM	S5	Estado no qual o sistema realiza a migração dos recursos do provedor de nuvem corrente para o provedor alternativo.
Super provisionado	S6	Estado no qual os recursos alocados no ambiente excedem a necessidade apresentada pelo conjunto de usuários.
Eliminar VM	S7	Estado no qual a gerência realiza a diminuição dos recursos alocados junto ao provedor de nuvem.

O estado inicial do sistema de gerência é o estado S0 – Aceitável, que no instante t = t0 apresenta pertinência igual a 1, sendo que os estados de S1 a S7 apresentam pertinência igual a 0.

A regra de transição entre o estado S0 Confortável e Estado S1 Aceitável se dá pelo aumento do tempo de resposta da aplicação com consequente piora da performance da mesma, para os usuários finais. Esta transição é ativa quando o tempo de resposta apresenta pertinência aos conjuntos, Tempo de Resposta Alto (HIGH) ou o conjunto Tempo de Resposta Muito Alto (VERYHIGH).

Implícito nesta transição está a transição do estado S0 para o próprio estado S0 dado pela regra RULE1 do código FCL da Figura 28 através da regra RULE 1, e a ação STAY.

```
; Transição do estado 0 para o estado 1
RULEBLOCK S01
    AND : MIN;
    ACT : MIN;
    ACCU : MAX;
    RULE 1  : IF respTime is LOW  and currState is PERT  then action is STAY;
    RULE 2  : IF respTime is HIGH or respTime is VERYHIGH then action is GO ;
END_RULEBLOCK
```

Figura 28 Transição S0 para S1

No Sentido oposto há a transição entre S1 e S0, através do código da Figura 29, ativada pelo retorno do tempo de resposta a valores confortáveis, dados pelo conjunto nebuloso BAIXO. Nesta transição a máquina de estados nebulosos retorna a condição de regime aceitável.

```
; Transição do estado 1 para o estado  0
RULEBLOCK S1S0
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF respTime IS LOW  AND currState is PERT THEN action IS GO;
        RULE 2  : IF respTime IS HIGH  OR respTime IS VERYHIGH THEN action IS STAY ;
END_RULEBLOCK
```

Figura 29 Transição S1 para S0

A transição entre o estado S1 e S2, código da Figura 30, ocorre ou por aumento do tempo de resposta para valores muito altos, dado pelo conjunto nebuloso VERYHIGH da variável respTime ou pela permanência por muito tempo

no estado S1 dado pela variável deltaT e seu conjunto VERYHIGH. O código encontra-se na Figura 30.

```
; Transição do estado S1 para o estado  S2
RULEBLOCK S1S2
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF respTime IS LOW  OR respTime IS HIGH THEN  action IS STAY;
    RULE 2  : IF respTime IS VERYHIGH AND currState IS PERT THEN  action IS GO ;
        RULE 3 : IF deltaT IS VERYHIGH THEN  action IS GO;
END_RULEBLOCK
```

Figura 30 Transição S1 para S2

A transição do estado S1 para o estado S3, código da Figura 31, se dá pela monitoração dos recursos alocados, considerando o número de usuários ativos em comparação ao número de máquinas virtuais disponibilizadas.

```
; Transição do estado 1 para o estado  3
RULEBLOCK S1S3
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF users IS MANY   AND currState IS PERT THEN action IS STAY;
         RULE 2  : IF rtt IS LOW AND currState is PERT THEN action IS GO;
        RULE 3  : IF users IS  FEW OR users IS MEDIUN THEN action is GO ;
END_RULEBLOCK
```

Figura 31 Transição S1 para S3

A transição do estado S2 para o estado S0 ocorre pelo retorno do tempo de resposta a valores baixos, conforme código da Figura 32. Os valores de tempo de resposta são caracterizados pelo conjunto nebuloso LOW da variável respTime.

```
; Transição do estado 2 para o estado  0
RULEBLOCK S2S0
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF respTime IS LOW AND currState is PERT   THEN action IS GO;
        RULE 2  : IF respTime IS  HIGH OR respTime IS VERYHIGH THEN action is STAY ;
END_RULEBLOCK
```

Figura 32 Transição S2 para S0

A transição do estado S2 para o estado S5 controlada pela monitoração do tempo de retardo entre o servidor e a rede das máquinas clientes. Se o tempo de retardo é considerado alto, a máquina realiza a migração das máquinas virtuais do provedor atual para o provedor alternativo. A regras desta transição levam em conta o o grau do retardo rtt, do tempo de resposta, respTime e o tempo de permanência no estado S2.

```
; Transição do estado 2 para o estado  5
RULEBLOCK S2S5
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF respTime IS LOW  OR respTime IS HIGH THEN  action IS STAY;
        RULE 2  : IFrespTime IS VERYHIGH AND rtt IS HIGH THEN  action IS GO ;
        RULE 3 : IF deltaT IS HIGH AND rtt IS HIGH THEN  action IS GO;
END_RULEBLOCK
```

Figura 33 Transição S2 para S5

A transição do estado S2 para o estado S4 é condicionada o baixos valores de retardo associados a alto tempos de resposta , expressados, pela regra 2 do código FCL na figura x. O tempo de permanência do estado atua de maneira alternativa forçando a transição para o estado S4.

```
; Transição do estado 2 para o estado  4
RULEBLOCK S2S4
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF respTime IS LOW  OR respTime IS HIGH THEN  action IS STAY;
        RULE 2  : IF respTime IS VERYHIGH AND rtt IS LOW THEN  action IS GO ;
        RULE 3 : IF deltaT IS VERYHIGH THEN  action IS GO;
END_RULEBLOCK
```

Figura 34 Transição S2 para S4

A transição do estado S3 para o estado S4 é ativada pelo temporizador , quando este atinge pertinência ao conjunto HIGH.

```
; Transição do estado 3 para o estado  4
RULEBLOCK S3S4
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF deltaT is LOW THEN action IS STAY ;
         RULE 2  : IF deltaT is HIGH THEN action IS GO;
END_RULEBLOCK
```

Figura 35 Transição S3 para S4

A transição do estado S4 para o estado S0 é disparada pela ativação da máquina virtual, ação ativada por este estado. No código FCL, Figura 36, não está descrito a condição de máquina virtual ativada, que é considerada pelo tempo decorrido na variável deltaT.

```
; Transição do estado 4 para o estado  0
RULEBLOCK S4S0
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF users IS MANY   THEN action IS STAY;
        RULE 2  : IF deltaT is THEN action IS GO;
END_RULEBLOCK
```

Figura 36 Transição S4 para S0

A transição do estado S5 para o estado S0, código da Figura 37, é acionada pela migração das máquinas virtuais do provedor A para o provedor B.

```
; Transição do estado 5 para o estado  0
RULEBLOCK S5S0
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF users IS THEN action IS STAY;
        RULE 2  : IF deltaT is HIGH THEN action IS GO;
END_RULEBLOCK
```

Figura 37 Transição S5 para S0

A transição do estado S0 para o estado S6, código da Figura 38, é controlada pela monitoração dos recursos alocados, quando o número de máquinas virtuais excede a necessidade, para o número de usuários ativos.

```
; Transição do estado 0 para o estado  6
RULEBLOCK S0S6
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF users IS MANY   THEN action IS STAY;
        RULE 2  : IF rtt IS LOW THEN action IS GO;
        RULE 3  : IF users IS  FEW OR users IS MEDIUN THEN action is GO ;
END_RULEBLOCK
```

Figura 38 Transição S0 para S6

A transição do estado S6 para o estado S7, código na Figura 39, se da pela monitoração dos recursos alocados e pelo incremento do temporizador *deltaT*.

```
; Transição do estado 6 para o estado  7
RULEBLOCK S6S7
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF users IS MANY   THEN action IS STAY;
        RULE 2  : IF users IS  FEW OR users IS MEDIUN THEN action is GO ;
        RULE3: IF deltaT is HIGH THEN action IS GO;
END_RULEBLOCK
```

Figura 39 Transição S6 para S7

A transição do estado S7 para o estado S0, código da Figura 40, ocorre pela ação de eliminação de uma máquina virtual excedente. Nas regras este evento está expresso pela expiração do temporizador *deltaT*.

```
; Transição do estado 7 para o estado  0
RULEBLOCK S7S0
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        RULE 1  : IF deltaT IS HIGH  AND THEN action IS GO;
        RULE 2  : IF deltaT  IS LOW THEN action IS STAY;
END_RULEBLOCK
```

Figura 40 Transição S7 para S0

5.6 Implementação do Modelo

O modelo foi implementado em um conjunto de classes em linguagem Java. Este conjunto de classes esta dividido em dois grupos, um que implementa a coleta de dados juntos ao sistema de Computação em Nuvem , e outro que implementa o controlador e a máquina de estados nebulosos.

A Máquina de Estados Nebulosos foi implementada utilizando-se 4 classes, FuState, FuTransition, FuStateAction e FuFSM. A classe FuState implementa os estados aos quais a máquina pode assumir, sendo FuTransition a classe que implementa as transições entre os estados. A classe FuTransition é a que contém as máquinas de inferência responsáveis por avaliar as variáveis frente as regras nebulosas. A classe FuStateAction, implementa as ações executadas por cada estado, e foi implementada em classe separada para permitir uma maior flexibilidade no tipo de ação executada. A classe FuFSM implementa a máquina de estado, utilizando as classes anteriores, esta classe é responsável pelo processo de execução das máquinas de inferência de cada transição.

Na Figura 41 esta representado o diagrama de classes da implementação da Máquina de Estados Nebuloso, mostrando os principais atributos e os métodos.

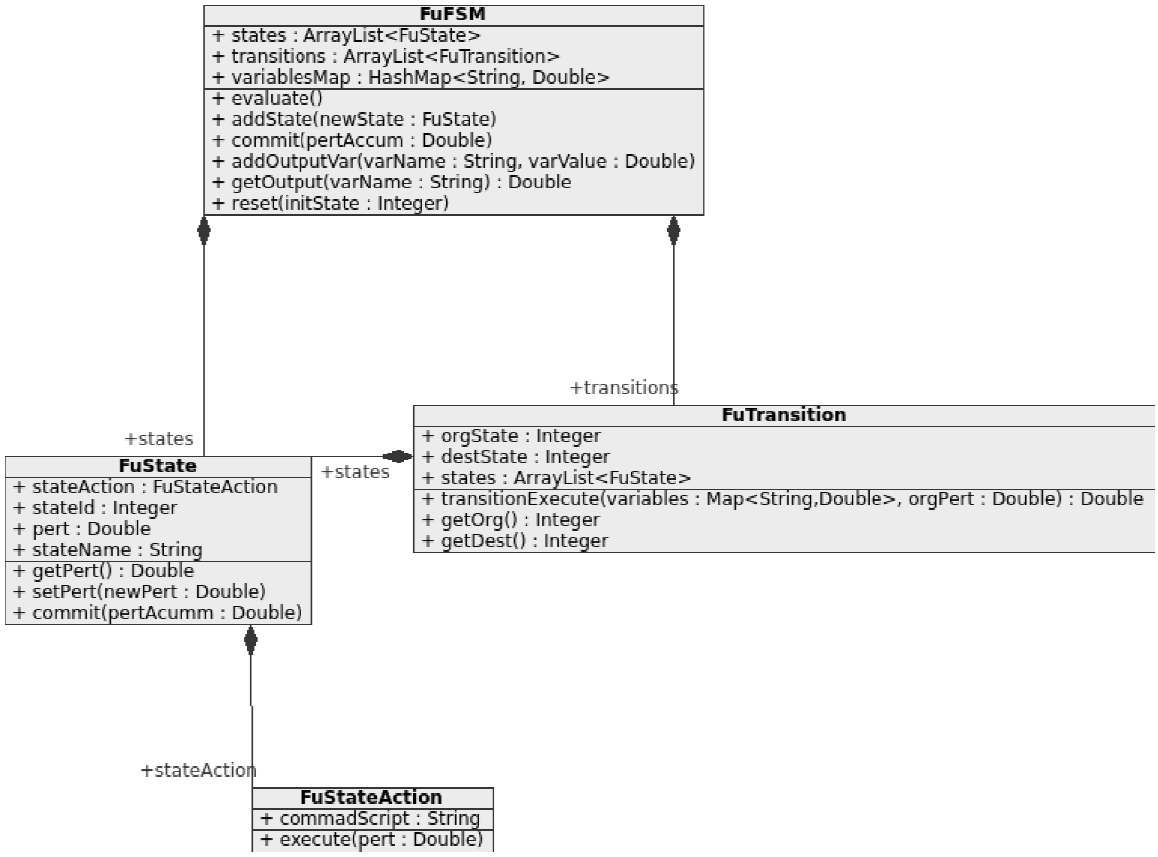


Figura 41 Diagrama de Classes da Máquina de Estados Nebulosos

O código fonte das classes que implementam a máquina de estados nebulosos esta listado no Anexo C.

6 Sistema Experimental

Para verificação do modelo proposto foram testados cenários de aplicações utilizando Computação em Nuvem, com variação do tempo de retardo, rtt , e carga de processamento na máquina virtual, userCpu, nas máquinas virtuais adjacentes e na máquina hospedeira. Nas seções a seguir é descrito o sistema experimental utilizado para investigação do modelo proposto.

6.1 Sistema de Teste

Para a montagem do laboratório de testes, utilizou-se exclusivamente máquinas operando com Linux, distribuição Ubuntu 12.04 LTS[82], versões "Server" e "Desktop". O sistema operacional Ubuntu foi escolhido por apresentar pacotes bem atualizados, amplo suporte ao uso de máquinas virtuais, tanto para operação como máquina hospedeira como para uso em imagens, sendo amplamente suportados pelos provedores de Nuvem como a Amazon[9] e Google[10]. Na Tabela 14, estão as descrições das máquinas utilizadas para montagem sistema experimental.

Tabela 14 Máquinas Utilizadas

Máquina	Configuração
1	CPU Core i7 870 16G RAM, 500G HD Ubuntu Linux 12-04
2	CPU Core2 Duo 2G RAM 80G HD Ubuntu 12-04
3	CPU Core 2 Duo 4G RAM 160G HD Ubuntu 12-04
4	2 Xeon 6c HT 56G RAM 500G HD Ubuntu Server 12-04
5	1 Xeon 4C HT 32G RAM 300G HD Ubuntu Server 12-04
6	AMD Athlon X2 2G RAM 80G HD Ubuntu Linux 12-04

. Adicionou-se a instalação básica do Ubuntu, pacotes, openssh-server[83], snmpd[84], mongodb[85] e Apache2[86] com suporte a PHP[87], em suas versões mais atualizadas para a distribuição utilizada.

Tabela 15 -Máquinas Virtuais

Máquina	Configuração
Servidores	1 VCPU 512M RAM, 5G HD Ubuntu Linux Server
Roteadores	1 VCPU 256 RAM 4G HD Ubuntu Server
Clientes	1 ou 2 VCPU 1G RAM 8G HD Ubuntu Server
Gerência	1 VCPU 1G RAM 12G HD Ubuntu Server

Todos os aplicativos para teste foram escritos em linguagem Java, compilados com a versão 1.7u9[88], utilizando as bibliotecas SNMP4J[89] na versão 2.0.3, JFuzzyLogic [64], na versão 2.1a.

6.1.1 Condições de Contorno

Algumas premissas são consideradas em relação ao ambiente a ser utilizado para testes, seja para simulação ou testes empíricos. As principais características assumidas são:

- Os servidores virtuais, quando dedicados a uma mesma aplicação, têm configurações idênticas.
- A aplicação apresenta consumo intensivo de capacidade de processamento e baixos requisitos de operações de leitura e escrita em disco.
- Os usuários se distribuem de maneira homogênea pelos servidores virtuais disponíveis, em cada instante de tempo.
- Os testes não são realizados em ambiente de produção, mas mediante simulação ou em ambientes controlados de laboratório.
- A aplicação cliente e a aplicação de gerência coletam o tempo de resposta, *respTime*, entre a requisição e a resposta do servidor.

6.2 Topologia do ambiente de testes

A topologia utilizada para os testes inclui o uso de três máquinas físicas uma emulando a rede na qual as estações clientes estariam operando , outra emulando um provedor de nuvem e outro emulando um segundo provedor de nuvem. Estas máquinas estão conectadas por um *switch Fast Ethernet*.

A máquina virtual de gerência foi instanciada no hospedeiro, junto às máquinas virtuais clientes. Com esta distribuição, procura-se evitar, que os testes em cenários que gerem saturação de processamento nas máquinas servidoras, possam impactar a coleta dos dados pela máquina de gerência.

Além da máquina virtual de gerência, foi criada uma máquina virtual de igual configuração que hospeda o banco de dados SQL no qual são armazenadas as medidas e a configuração da solução de gerência.

A topologia da montagem de testes está representada na Figura 42. As máquinas clientes além dos scripts para teste da plataforma, utilizam o pacote de software YCSB[90] que produz relatórios de desempenho do servidor de nuvem utilizando cargas de trabalho específicas. As máquinas clientes executam um

programa em Java que emula um cliente fazendo requisições por HTTP, sendo que o programa cliente armazena os tempos de resposta obtidos, em uma base de dados, que posteriormente serão utilizados para definir o tempo médio de resposta da aplicação, *resptTime*.

O suíte YCSB[90] foi utilizado para fazer o perfil de uma aplicação simulando o uso de uma base de dados mongodb[85]. Este perfil estabelece o número máximo de transações por segundo que a máquina seria capaz de suportar. Este número serve de parâmetro para dimensionar as receitas que a máquina seria capaz de gerar com carga máxima de usuários concorrentes.

Nas máquinas servidoras, além do mongodb, utiliza-se programas em PHP, para simular a aplicação acessada pelas máquinas clientes. Para simulação de alto consumo de CPU utilizam-se programas em linguagem C.

Para conexão das máquinas virtuais com a rede física, utilizou-se servidores com Linux, com duas interfaces de redes virtuais. Para comunicação entre as maquinas virtuais de diferentes máquinas físicas foi utilizado rotas estáticas configuradas em cada servidor. A topologia do ambiente de testes esta documentada na Figura 42.

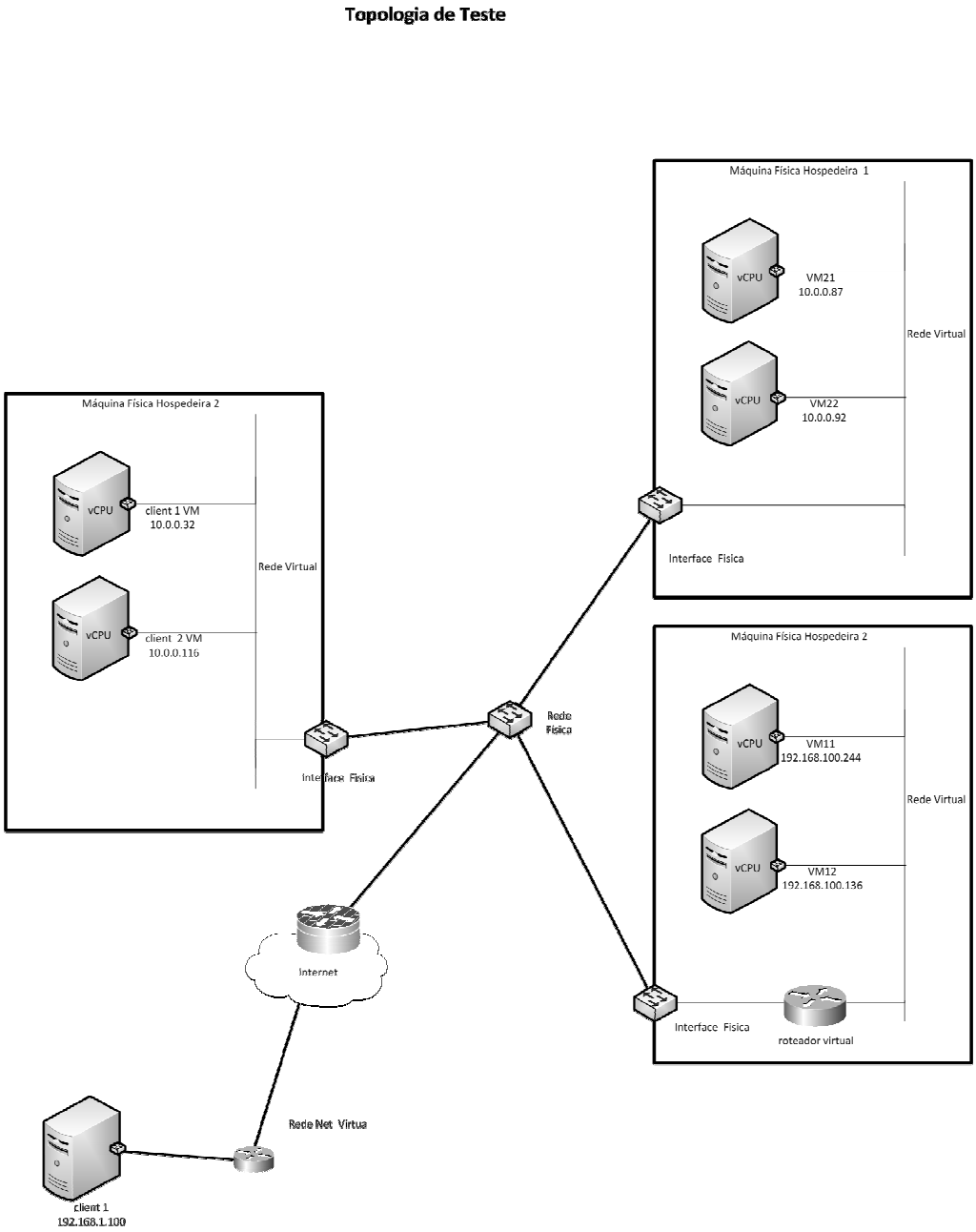


Figura 42 Topologia de Testes

Os dados coletados são posteriormente transferidos para arquivos de texto, que são utilizados como fonte de dados para as simulações com as máquinas de estados nebulosos.

Para este trabalho foram utilizados valores fictícios para o número de tickets de lentidão (*slowTT*) , como forma de validar o funcionamento dos algoritmos. Esta variável é definida como uma porcentagem do universo de clientes em operação da solução.

7 Resultados

Para verificação do modelo do controlador proposto, foram testados diferentes cenários de aplicações utilizando-se de um sistema de Computação em Nuvem. Ao longo do experimento variou-se o tempo de retardo, o número de usuários conectados e a carga de processamento na máquina virtual. A carga de processamento foi variada nas máquinas virtuais e na máquina hospedeira..

O controlador utilizado nos expetimento tem seu diagrama de estados descrito na Figura 43. Está máquina de estados nebulosos inclui estados representativos de degradação do desempenho da solução gerenciada e dos cenários de alocação excessiva de recursos.

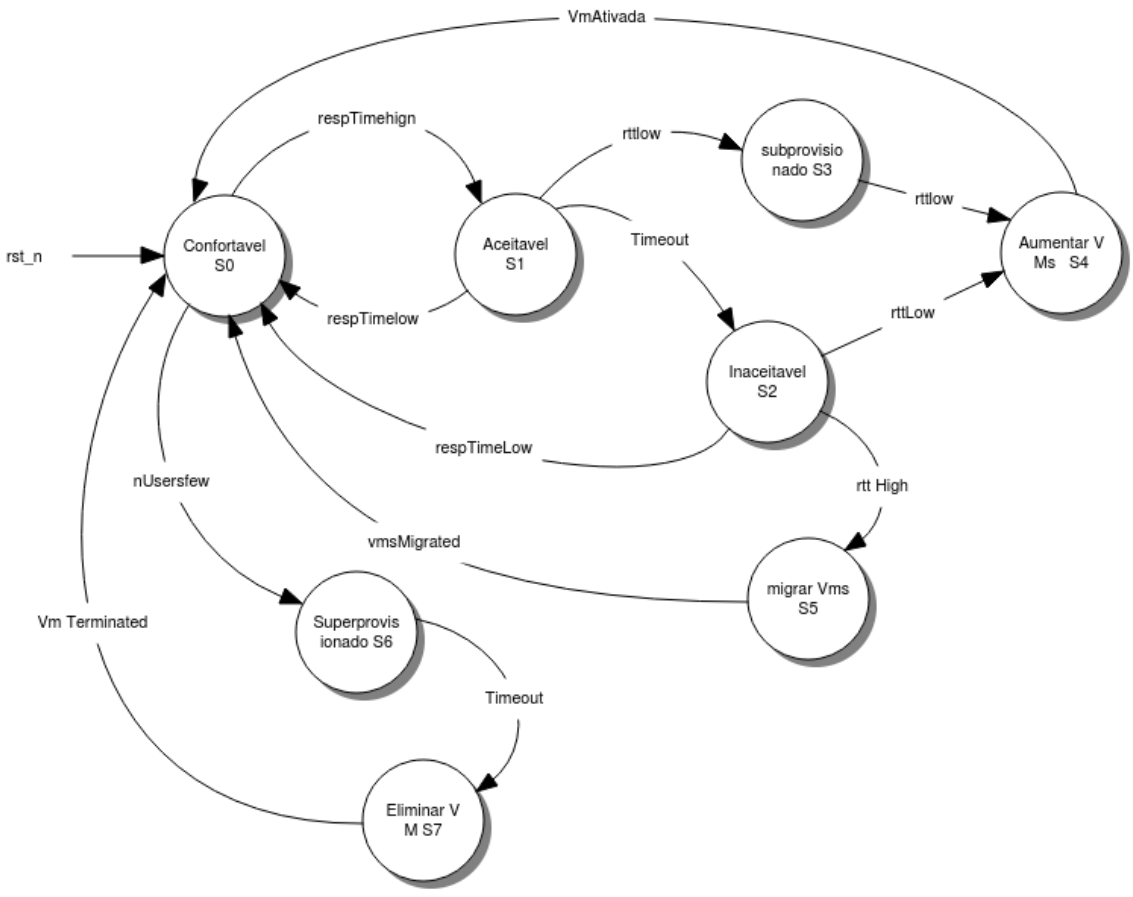


Figura 43 Máquina De Estatos Testada

Esta máquina de estado implementa três ações: aumento de recursos de máquinas virtuais, migração de recursos para uma nuvem alternativa e diminuição de recursos alocados. O aumento de recursos ocorre quando o desempenho cai devido ao aumento do número de usuários, aumento do consumo médio de recursos ou pelo aumento do tempo de resposta de rede. A diminuição de recursos ocorre quando o número de máquinas virtuais em operação excede o número necessário para atender a demanda dos usuários ativos. A Migração de

máquinas virtuais de uma nuvem para outra ocorre pela degradação do tempo de resposta de rede ou pelo degradação do desempenho das máquinas virtuais.

A descrição de cada estado esta na Tabela 13 da seção 5.5 na página 52.

7.1 Cenário com aumento de tempo de resposta

A máquina de estados foi alimentada com um conjunto de dados que correspondem a um aumento no tempo de resposta de uma aplicação, e que teria como causa o aumento do tempo de latência da rede, *rtt*. Este aumento pode ser causado pela alteração das características do meio ou pela migração da máquina virtual, por parte do provedor, para outra zona de disponibilidade. O aumento simulado modifica o valor médio do tempo de latência, rtt, de 10 ms para 200 ms. Como resultado, o tempo de resposta da aplicação, respTime, é degradado para valores acima do SLA esperado. A alteração aplicada no instante, t = 200s inicialmente tem duração de 1000s, conforme pode ser verificado na Figura 44

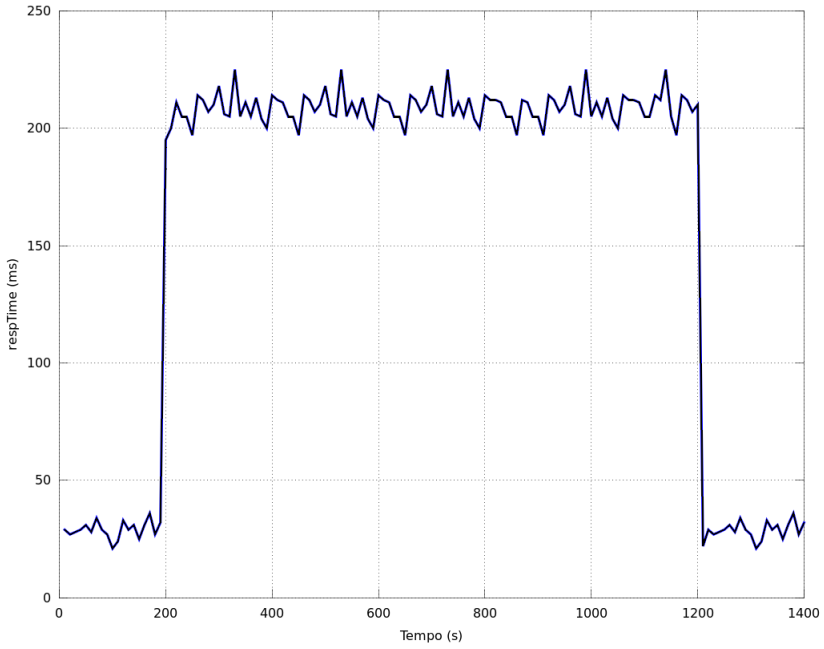


Figura 44 Tempo de resposta degradado

Como condição de contorno, é considerado que nenhum outro parâmetro é alterado ao longo do período de degradação do tempo de resposta, mantendo-se constantes em seus valores de operação normal.

Na Figura 45 tem-se uma representação da pertinência dos estados ao longo do tempo. Nas ordenadas representa-se o tempo de simulação.Cada curva representa um estado do controlador ou variável controlada com valores variando de 0 a 1. Para facilitar a visualização, as variáveis Tempo de Resposta(*respTime*) e Latência, *rtt*, foram normalizadas para se ter como valor máximo 1.0, ficando na mesma escala das pertinências dos estados da Máquina de Estados Nebulosos.

A partir do gráfico da Figura 45 pode-se verificar, em *t* = 200s, a transição do estado confortável (S0) para o estado Aceitável (S1). Nesta transição a pertinência do estado Confortável passa a 0 e a pertinência do estado Aceitável passa a 1.0. Em *t* = 520 s inicia-se a transição do estado Aceitável para o estado Inaceitável (S2).

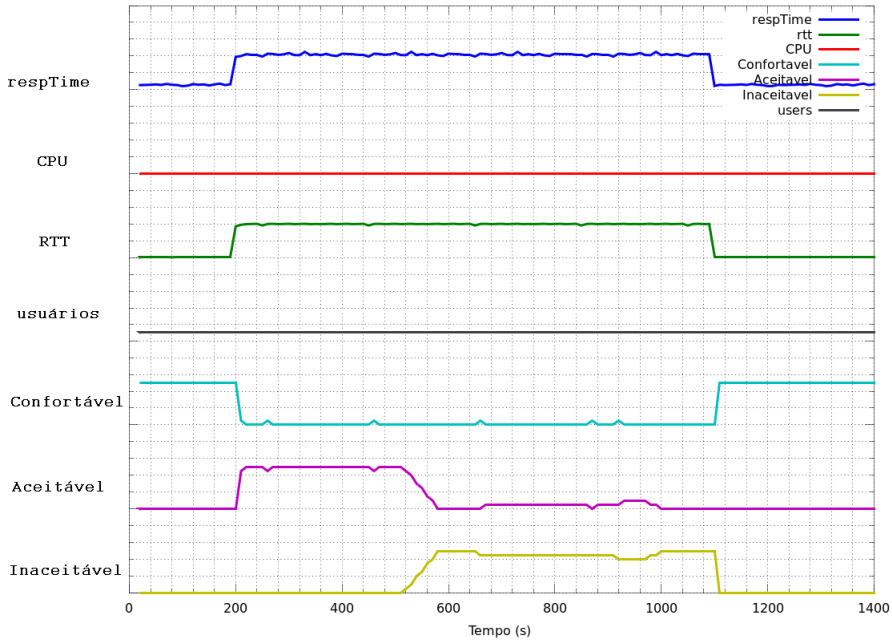


Figura 45 Resposta ao aumento de rTT

A fim de testar o funcionamento da Máquina de Estados Nebulosos, foi realizado um experimento onde a duração do período de degradação no valor de rtt, inicialmente configurada em 1000 segundos, teve seu valor diminuído, O período de degradação de rtt, foi diminuído em 100s a cada experimento. Para valores abaixo de 400s, não é verificada a transição do estado Aceitável para o estado Inaceitável. Estes resultados evidenciam o comportamento esperado para máquina de estados que não deve acionar medidas corretivas para degradações moderadas por curto espaço de tempo.

A duração da degradação pelo período de 400s, com aumento de rtt para 200ms, faz com que a máquina de estados faça a transição do estado Aceitável para o estado inaceitável. Como pode ser verificado na Figura 46, o período de alteração é suficiente para disparar um evento corretivo, como a migração da máquina virtual de infraestrutura.

Para verificação do efeito da intensidade da degradação, diferentes valores de retardo, utilizando os mesmos períodos de degradação, foram testados. O valor médio de rtt, foi variado de 100 ms, até 550 ms. Desta forma testa-se a resposta da máquina de estados a degradações de alta e baixa intensidade com diferentes durações.

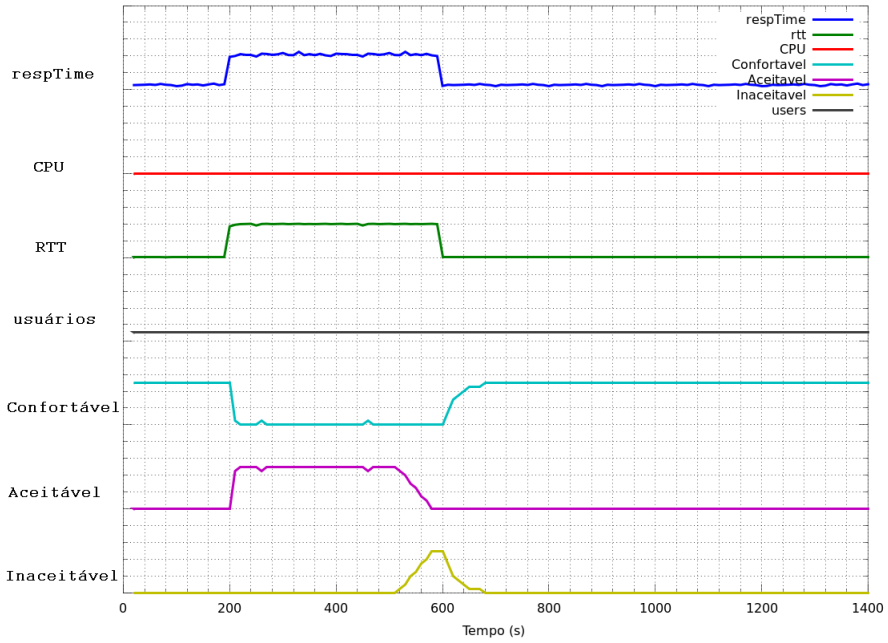


Figura 46 Degradação de tempo de resposta por 400s

Aumentando-se a intensidade da degradação, a transição de estados ocorre segundo e o diagrama de estados mostrado anteriormente na Figura 43. Observa-se que uma alteração de maior intensidade é tolerada por menos tempo. Isto pode ser verificado na Figura 47, na qual a degradação do tempo de resposta, *respTime* atinge valores em torno de 550ms. A resposta da máquina de estados, se altera e a transição entre estados é praticamente instantânea.

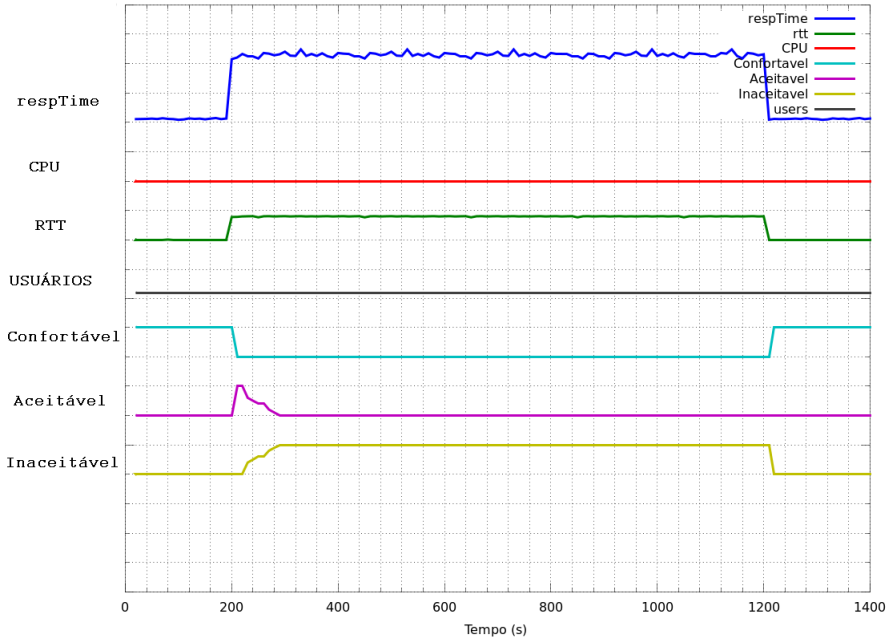


Figura 47 Tempo de Resposta degradado para 550ms

A degradação com intensidade de 550 ms , registrada no gráfico da Figura 47, tem pronta resposta. Realizando a transição de estados em intervalos de tempo inferiores ao da transição realizada para uma degradação de 200 ms, registrada na Figura 45. A máquina que realiza a transição do estado Confortável para Aceitável e deste para Inaceitável. . A influência da intensidade também pode ser observada no gráfico da Figura 48, quando em comparação ao gráfico da Figura 45 a transição do estado Aceitável , para o estado Inaceitável inicia-se já no ciclo seguinte à entrada neste estado. Neste gráfico fica evidenciada a característica de transição gradual de uma Máquina de Estados Nebulosos

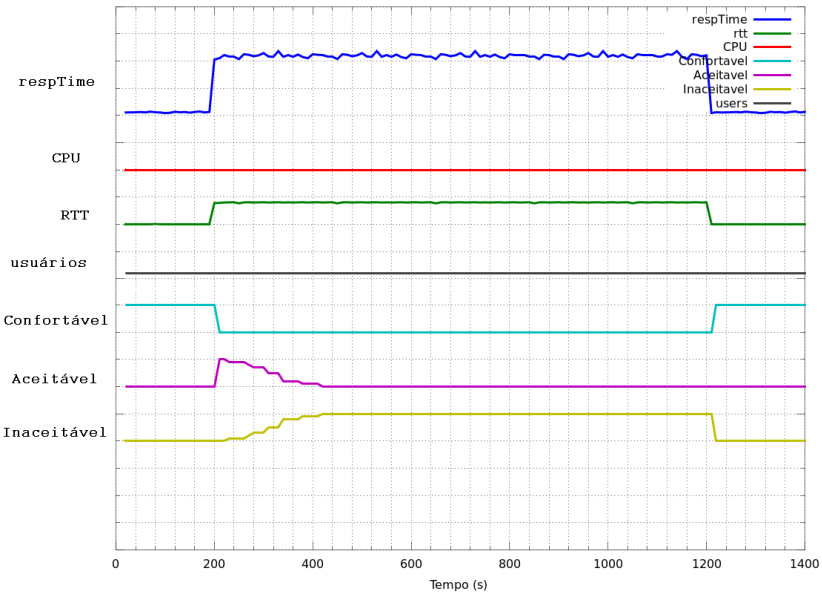


Figura 48 Degradação a 525 ms

Para comparar o efeito da intensidade da degradação na transição entre os estados, foram medidos os tempos nos quais se inicia a transição entre os estados e o intervalo de tempo necessário para que o estado atinja pertinência máxima, caso isto ocorra. Para tanto foi determinado o instante inicial no qual a pertinência do estado S0, ou Confortável, deixa de ser 1 e o estado S1, Aceitável, passa a ter pertinência diferente de 0.0. Os mesmos critérios foram aplicados a transição entre o estado S1 e o estado S2, Inaceitável. Os resultados destas medidas estão na Tabela 16. A partir dela pode-se observar que para tempos baixos, até 100ms , nenhuma transição é verificada. Para tempos entre 150ms e 175 ms, ocorre a transferência de pertinência, mas a pertinência máxima não é atingida por nenhum dos estados destinos. Para tempos superiores a 200ms a transição entre estados se completa , com a pertinência se transferindo completamente para o estado destino.

Tabela 16 Tempos de Transições de Estados

Degradação (ms)	$\mu(S1) \neq 0$ (s)	$\mu(S1) = 1$ (s)	$\mu(S2) \neq 1$ (s)	$\mu(S2) = 1$ (s)
100 ms	-	-	-	-
150	30	-	-	-
175	10	-	320	-
200	10	20	320	380
250	10	10	330	390
500	10	10	340	380
525	10	10	30	220
550	10	10	30	90

A característica de ser possível ter pertinência a mais que um estado fica evidente analisando-se o comportamento da máquina de estados para um aumento de tempo de resposta para um valor equivalente a 175ms. Para este valor de atraso a Máquina de Estados Nebulosos, não consegue completar a transição de um estado para o outro.

Com o pode ser observado na Figura 49, a máquina de estados apresenta pertinência parcial aos três estados, não completando a transição durante a duração da degradação. Nesta condição o sistema não consegue atuar para execução de ações corretivas.

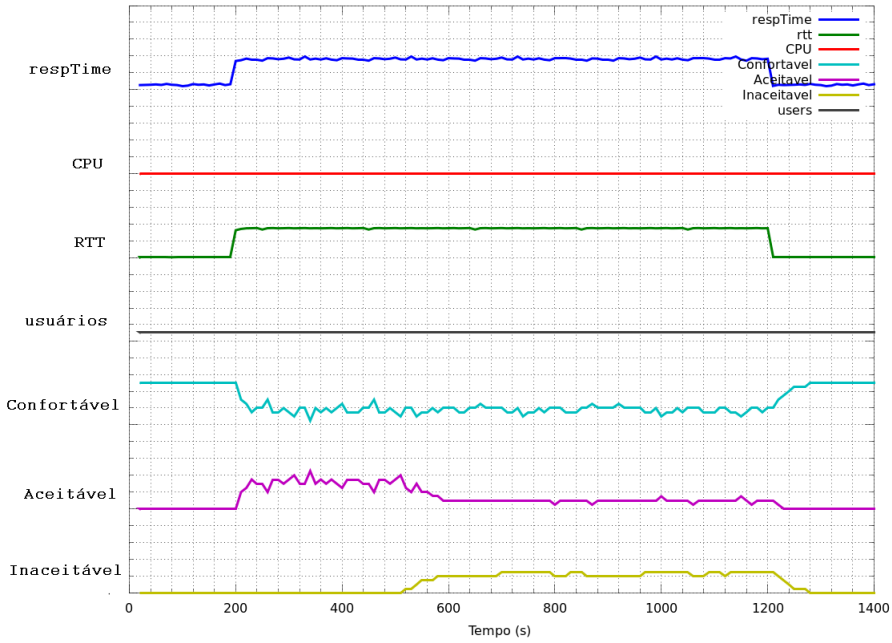


Figura 49 Pertinência parcial em três estados.

7.2 Cenário com aumento de consumo de CPU

Para testar a resposta do controlador nebuloso a aumento de consumo de CPU e sub dimensionamento de recursos, o fator originador da degradação de tempo é alterado. O aumento de tempo de retardo, é substituído pelo aumento de consumo de CPU e sobrecarga da máquina virtual servidora.O gráfico da Figura 50 representando o diagrama de transição de estados, em resposta ao aumento de consumo de CPU. Este gráfico inclui os estados referentes a alocação de recursos na nuvem.

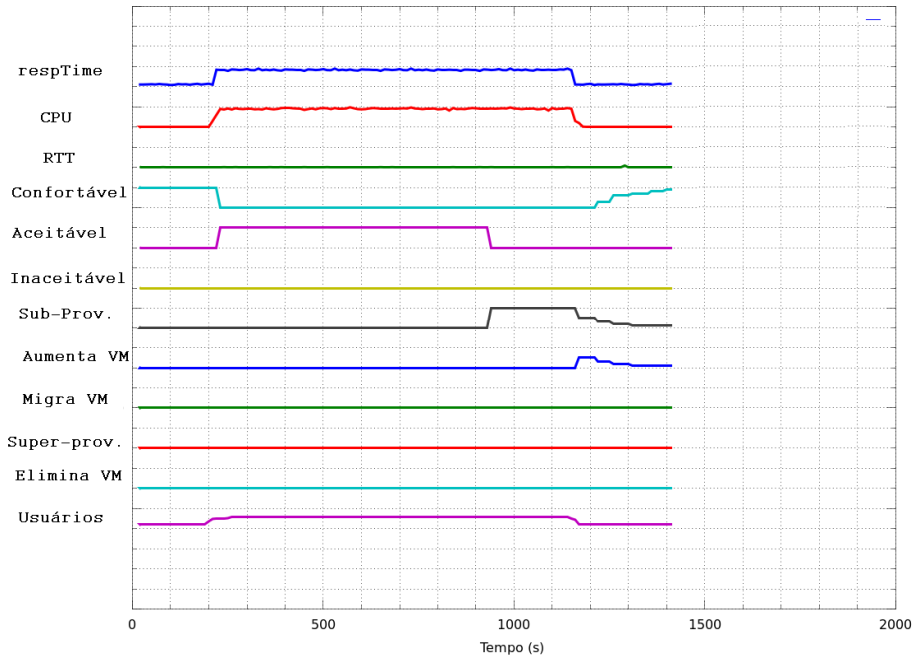


Figura 50 Aumento de consumo de CPU

No cenário registrado na Figura 50 a alteração de tempo e CPU muda do estado Aceitável para o estado sub-provisionado.[

7.3 Instabilidade no sistema de nuvem

O ajuste do limiar de tempo do temporizador implementado pela variável *deltaT* pode ser usado para alterar o ponto de reação a alterações como tempo de resposta de aplicação, *resptTime*, inconstante. Este cenário seria caracterizado por curtos períodos de degradação, com duração inferior ao limiar necessário para a transição do estado S1 para o estado S2.

Neste cenário a transição de S1 para S0 ocorre imediatamente após ao retorno das variáveis que dispararam a transição de S0 para S1 retornarem aos valores de regime, com o sistema operando no estado S0, confortavel. Isto pode ser verificado através do gráfico da Figura 51, onde a transição do estado Aceitavel para o estado Confortavel, ocorre na sequência do retorno a valores normais de operação da variável *respTime.*

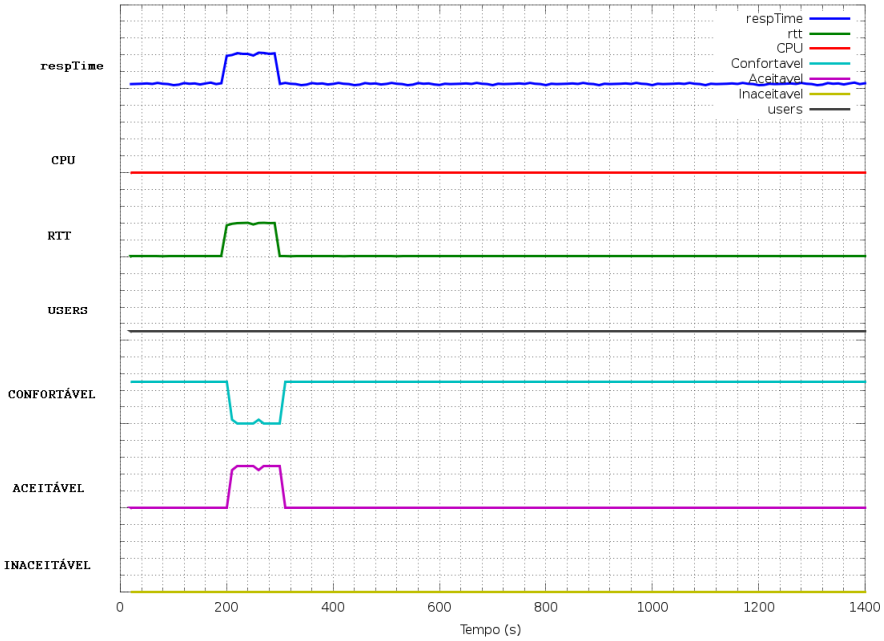


Figura 51 Transição curta de estados S1 para S0

. Na ocorrência de uma nova degradação dos parâmetros da nuvem ocorre nova transição de estados com reinicio dos temporizadores. Isto pode ser verificado na Figura 52, na qual uma nova degradação em rtt causa a transição do estado Confortavel para o Aceitável..

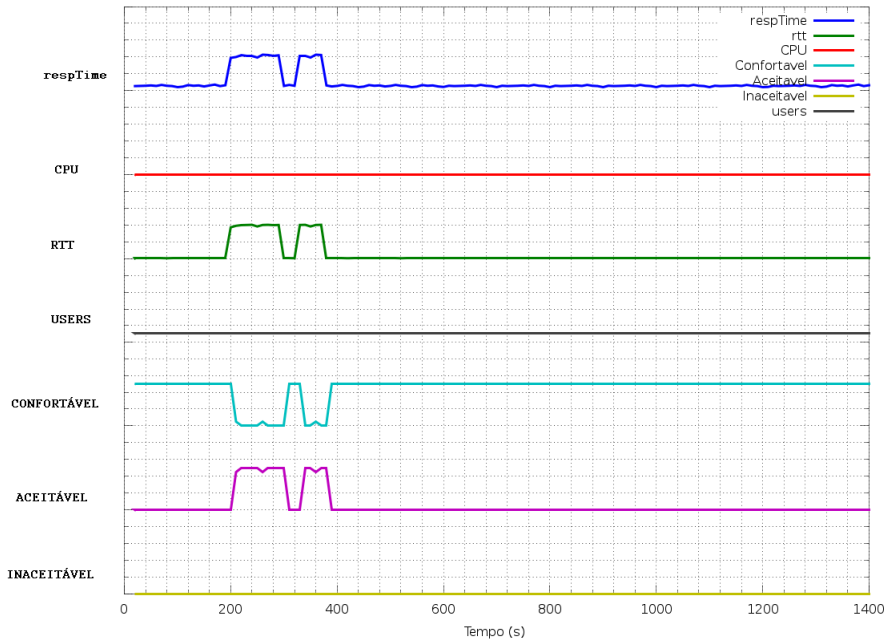


Figura 52 Deterioração seguida de tempo de resposta.

Para tratar este tipo de ocorrência é adicionado o tratamento do temporizador deltaT às regras da transição de S1 para S0. Desta forma mesmo com o retorno do tempo de resposta a valores de regime, a máquina de estado permanece no estado S1, somente retornando após um intervalo de tempo, que é dependente da intensidade do sinal e do valores dos conjuntos nebulosos da variável deltaT.

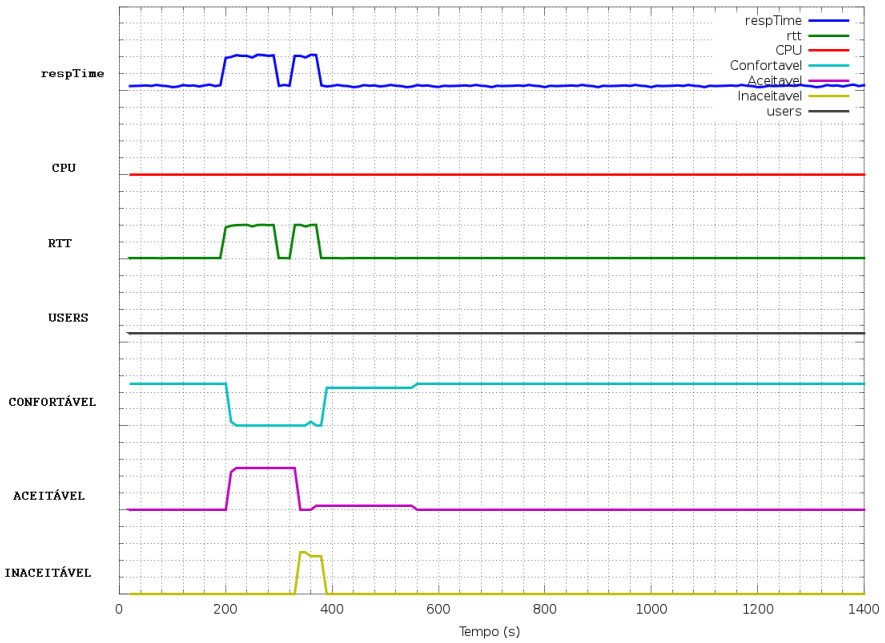


Figura 53 Deterioração seguida do tempo de resposta com temporizador de estado

Com o ajuste do temporizador, em violações seguidas do nível de acordo de serviço, a máquina transita do estado S1 para o S2.

7.4 Custo de uso

Considerando-se a métrica do custo de uso da solução uma comparação é feita entre a adoção de metodologias exatas e metodologias com lógica difusa. Uma das ações suportadas é a mudança da aplicação de uma nuvem para outra baseado na violação do tempo de resposta, ou em outros condicionadores da nuvem em utilização. A nuvem de produção pode ser privada ou publica. Em ambos os casos é atribuído um custo para comparação.

Neste caso foram utilizadas as seguintes condições de contorno:

Considerando-se $\mathbf{C_A}$, custo de Hora de máquina na nuvem A, $\mathbf{C_B}$, custo de Hora de máquina na nuvem B , $C_B > C_A$; No caso foi arbitrado o custo da nuvem A para $100,00 por período de uso e a da nuvem B para $125,00.

O tempo de Comutação da nuvem A para B , $\mathbf{T_{A \rightarrow B}}$ é o mesmo de B para A , $\mathbf{T_{B \rightarrow A}}$, e não existem penalidades ou custos adicionais pela transição

O tempo de uso mínimo da Nuvem é de uma unidade de tempo $\mathbf{T_{use}}$ abreviado nos testes para 10 segundos para facilitar a simulação.

7.5 Custo e receita

Considerando o aumento simultâneo de tempo de resposta da aplicação , e tempo de retardo, obtido através da introdução no ambiente de testes de um tempo de retardo de 200 ms, o que ocasiona um incremento do tempo de resposta da aplicação sem alteração dos outros parâmetros como CPU.

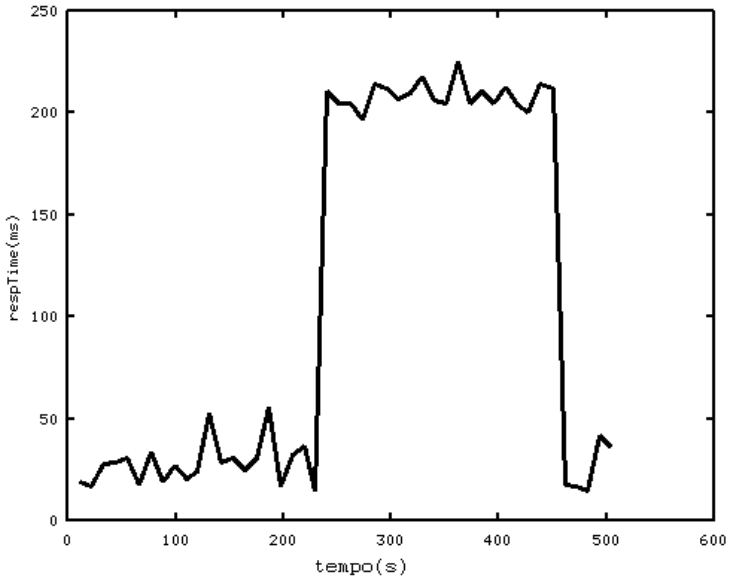


Figura 54 Aumento de tempo de Resposta

Este aumento de tempo é processado controlador o que gera a ativação de uma máquina na nuvem alternativa.

No caso em que o impacto da degradação do tempo de resposta da aplicação não é diretamente associado à diminuição de receita, a atuação do controlador nebuloso pode representar uma economia , nos testes realizados entre 3% e 50% na alocação de recursos. No primeiro caso considerando um ambiente que apresenta demanda marginalmente superior a capacidade alocada por até uma hora diariamente. O segundo cenário considera que somente uma máquina virtual esta sendo utilizada e uma segunda máquina seria ativada como resposta à instabilidade momentânea que origina a degradação no tempo de resposta.

O custo de uso desta solução representado na Figura 55, inclui o custo de operação concomitante das máquinas quando da troca de nuvem, e a economia gerada quando da não ativação da máquina alternativa.

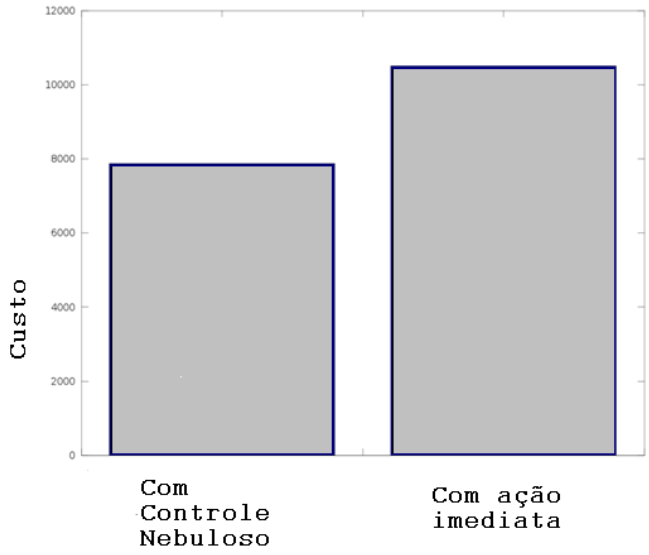


Figura 55 Custo de uso com operação simultanea da máquina virtual alternativa.

Após o retorno do retardo da solução normal a valores de regime a solução é alterada novamente para condição inicial desativando a máquina da nuvem alternativa.

8 Conclusão

Neste trabalho desenvolveu-se e testou-se um modelo de controlador para sistemas de computação em nuvem, utilizando-se de uma máquina de estados nebulosos que, pode contribuir para a gerência destes sistemas. Devidos às características de gerenciamento compartilhadas destes sistemas nos quais o usuário tem uma visibilidade bastante limitada do estado dos ambientes em que seus aplicativos e serviços estão hospedados, a Lógica Nebulosa permite implementar políticas de controle e otimização de uso dos recursos que atendam aos critérios estabelecidos nos contratos de acordo de nível de serviço. Além disso, o controlador também leva em conta a qualidade de experiência do usuário. Comparando-se o desempenho desta ferramenta com o de um controlador típico, baseado em Álgebra Booleana, obteve-se uma economia entre 3% até 50% de recursos, isto para um sistema operando com servidor único, dependendo das solicitações de demanda. Em complementação as estes fatores, a Máquina de Estados Nebulosos permite o adequado tratamento das sequências de eventos de operação de uma implementação de Computação em Nuvem.

Observa-se que não foi encontrada na literatura referência à utilização de máquinas de estados nebulosos para o tratamento de sequência de eventos em controle de sistemas de computação em nuvem.

A integração com outros sistemas de controle, como os sistemas de energia e refrigeração também contribuem para validade do uso das técnicas baseadas em Lógica Nebulosa, posto que muitos destes sistemas, já são controlados por controladores baseados nesta tecnologia. O módulo de controle nebuloso pode ser integrado também a ferramenta de gerência de redes como o Nagios[91] e o OpenNMS[92].

Finalmente, o controlador discutido nesta dissertação poderia ser aplicado à interligação de redes de sensores e equipamentos por rede IP, sendo que a capacidade de computação, rede e armazenamento para estes dispositivos poderiam ser disponibilizados na forma de um sistema em nuvem..

Enquanto toda lógica nebulosa está definida em arquivos de configuração, a configuração para coleta dos parâmetros medidos e todo tratamento aplicado a estes valores está definido no código do sistema, um trabalho posterior poderia migrar este tratamento para código XML como utilizado em [93].

.

9 Referências

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica e M. Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing,” Electrical Engineering and Computer Sciences, University Of California at Berkeley , 2009.

[2] “IEEE Cloud Computing Community,” IEEE, April 2012. [Online]. Available: http://cloudcomputing.ieee.org/about. [Acesso em 1 10 2012].

[3] “ITU Focus Group on Cloud Computing,” ITU, 12 2011. [Online]. Available: http://www.itu.int/en/ITU-T/focusgroups/cloud/Pages/default.aspx. [Acesso em 1 10 2012].

[4] P. Mell e T. Grance, “The NIST Definition of Cloud Computing,” National Institute of Standars and Technology, 2009.

[5] “Gmail,” Google, [Online]. Available: mail.google.com. [Acesso em 1 10 2012].

[6] “Mail.live.com,” Microsoft, [Online]. Available: http://mail.live.com. [Acesso em 1 10 2012].

[7] “Dropbox,” Dropbox, [Online]. Available: https://www.dropbox.com/. [Acesso em 1 10 2012].

[8] “Salesforce,” Salesforce, [Online]. Available: http://www.salesforce.com. [Acesso em 1 10 2012].

[9] “Amazon Web Services,” [Online]. Available: http://aws.amazon.com. [Acesso em 01 Outubro 2012].

[10] Google, “Google Apps,” [Online]. Available: http://www.google.com/apps. [Acesso em 01 Outubro 2012].

[11] Ramasivakarthik, K. Annapureddy e Y. Raivio, “Efficient and Dynamic Resource Management of Telecom Components in Hybrid Cloud.,” em *International Software Technology Exchange Workshop* , 2011.

[12] L. Romano, D. D. Mari, Z. Jerzak e C. Fetzer, “A novel Approach to QoS montoring in The Cloud,” em *First International Conference on Data Compression , Communications and Processing* , 2011.

[13] J. Rao, Y. Wei, J. Gong e C. Z. Xu, “DynaQoS: Model-free self-tuning fuzzy control of virtualized resources for QoS provisioning.,” em *Quality of Service (IWQoS), 2011 IEEE 19th International Workshop on. IEEE, 2011.*, 2011.

[14] T. HoBfeld, R. Schatz, M. Varela e C. Timmerer, “Challenges of QoE Management for Cloud Applications,” *IEEE Communications Magazine ,* pp. 28-36, Abril 2012.

[15] E. Kagetzakis, H. Koumaras, M. A. Kourtis e V. Koumaras, “QoE4CLOUD: A

QoE-driven Multidimensional Framework for Cloud Environments," em *International Conference on Telecomunications and Multimedia (TEMU)*, 2012.

[16] L. A. Zadeh, "Fuzzy Logic," *Computer Magazine,* pp. 83-93, April 1988.

[17] H. E. T. Carvalho, N. C. Fernandes e O. C. M. B. Duarte, "Um Controlador Robusto de Acordo de Nível de Serviços para Redes Virtuais Baseado em Lógica Nebulosa," em *XXIX SBRC Simpósio Brasileiro de Redes de Computadore e Sistemas Distribuidos*, 2011.

[18] M. P. Fernandez, A. d. C. P. Pedroza e J. F. Rezende, "Implementação de Políticas de Gerenciamento através de lógica Fuzzy visando melhoria de Qualidade de Serviço (QoS)," em *XXI Simpósio Brasilieiro de Redes de Computadores*, 2012.

[19] D. Grossman, "New terminology and clarification for Diffserv RFC3260," IETF, 2002.

[20] D. Bacciu, A. Botta e H. Melgratti, "A fuzzy approach for negotiating quality of services.," em *Trustworthy Global Computing (2007)*, 2007.

[21] J. Xu, Zhao, Ming, Fortes, José, R. Carpenter e M. Yousif, "Autonomic resource managemetn in virtualiza data center using fuzzy logic-based aproaches," em *Cluster Computing*, 2008.

[22] Y. Lee, "QoS metrics for service level measurement for SOA environment," em *6th International Conference on Advanced Information Management and Service (IMS), 2010* , 2010.

[23] Z. Chen, Y.-H. Zhu e M. Y. Yuan, "Towards Self-Optimization in Utility Computing using Fuzzy Logic Controller," em *EEE International Conference on Service Operations and Logistics, and Informatics*, Beijing, China. , 2005..

[24] M. Tarighi, S.A.Motamedi e S. Sharifian, "A new model for virtual machine migration in virtualized cluster server based on Fuzzy Decision Making," *Journal of Telecommunications ,* vol. 1, n. 1, pp. 40-51, 2010.

[25] A. Andrzejak, D. Kondo e S. Yi, "Decision model for cloud computing under sla constraints," em *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems* , 2010.

[26] I. Breskovic, M. Maurer, V. C. Emeakaroha, I. Brandic e S. Dustdar, "Cost-efficient utilization of public SLA templates in autonomic cloud markets," em *Fourth IEEE International Conference on Utility and Cloud Computing* , 2011.

[27] L. M. Reyneri, "An introduction to fuzzy state automata," *Biological and Artificial Computation: From Neuroscience to Technology,* pp. 273--283, 1997.

[28] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger e D. Leaf, "Nist Cloud Computing Reference Architecture," NIST - National Institute of Standards and Technology , 2011.

[29] R.Dodda, M. C. e S. A. van, "An architecture for Cross-Cloud system management. Contemporary Computing," University of Newcastle upon Tyne, 2009.

[30] "jClouds," [Online]. Available: http://www.jclpouds.org. [Acesso em 1 10 2012].

[31] B. Sotomayor, R. S. Montero, I. M. Llorente e a. I. Foster, "An Open Source Solution for Virtual Infrastructure Management in Private and Hybrid Clouds," *IEEE INTERNET COMPUTING, SPECIAL ISSUE ON CLOUD COMPUTING,* vol. 13, n. 5, pp. 14-22, 7 7 2009.

[32] “Eucalyptus,” Eucalyptus, [Online]. Available: http://www.eucalyptus.com/. [Acesso em 1 10 2012].

[33] J. Dejun, P. Guillaumeand e C. Chi-Hung, “EC2 performance analysis for resource provisioning of service-oriented applications,” em *Service-Oriented Computing.*, Berlin/Heidelberg, 2010.

[34] *OpenStack Compute Administration Manual,* Openstack LLC, 2012.

[35] M. Rosenblum, “Virtual Machine Monitors: Current Technologi and Future Trends,” *Computer,* vol. 38, n. 5, pp. 39-47, 2005.

[36] “XEN,” [Online]. Available: http://www.xen.org. [Acesso em 1 10 2012].

[37] P. Barham, B. Dragovic, K. Fraser, T. H. S. Harrisn, A. Ho, R. Neugebauer, I. Pratt e A. Warfield, “Xen and the Art of Virtualization,” em *SOSP*, 2003.

[38] “KVM - Kernel Based Virtual Machine,” [Online]. Available: http://www.linux-kvm.org. [Acesso em 1 10 2012].

[39] A. Kivity, Y. Kamay, D. Laor, U. Lublin e A. Liguori, “kvm: the Linux Virtual Machine Monitor,” em *Proceedings of Linux Symposium*, Ottawa, Ontario - Canada , 2007.

[40] “Oracle VM VirtualBox,” Oracle, [Online]. Available: http://www.virtualbox.org. [Acesso em 1 10 2012].

[41] “Vmware,” [Online]. Available: https://www.vmware.com/. [Acesso em 1 10 2012].

[42] “Hyper-V,” Microsoft, [Online]. Available: http://www.microsoft.com/en-us/server-cloud/ws2012/default.aspx. [Acesso em 1 10 2012].

[43] Q. Zhang, L. Cheng e R. Boutaba, “Cloud Computing: state-of-the-art and research challenges,” *Journal of Internet Services and Applications,* vol. 1, n. 1, pp. 7-18, 2010.

[44] N. Santos, K. P. Gummadi e R. Rodrigues, “Towards trusted cloud computing,” em *2009 Conference on Hot Topics in cloud computing*, 2009.

[45] A. Ciuffoletti, “Monitoring virtual network infrastructure,” *ACM SIGCOMM Computer Communication Review ,* vol. 40, n. 5, pp. 47-52, 2010.

[46] F. N. ,. G. S. ,. K. O. ,. W. D. Battré D., “Evaluation of Network Topology Inference in Opaque Compute Clouds Through End-to-End Measurements,” em *IEEE CLOUD 2011*, 2011.

[47] L. Cherkasova e R. Gardner, “Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor,” em *USENIX annual technical conference*, 2005.

[48] G. Wang e T. E. Ng, “The impact of virtualization on network performance of Amazon EC2 data center,” em *INFOCOM Proceeding IEEE*, 2010.

[49] G. v. Laszewski, J. Diaz e G. C. F. Fugang Wang, “Comparison of Multiple Cloud Frameworks,” em *IEEE Fifth International Conference on Cloud Computing*, 2012.

[50] “Openstack Project,” [Online]. Available: http://www.openstack.org . [Acesso em 1 10 2012].

[51] “Apache Cloudstack,” [Online]. Available: www.cloudstack.org . [Acesso em 1 10 2012].

[52] “OpenNebula,” [Online]. Available: http://www.opennebula.org. [Acesso em 1 10 2012].

[53] “Vcloud Director,” Vmware, [Online]. Available: http://www.vmware.com/products/vcloud-director/overview.html. [Acesso em 07 01 2013].

[54] OnApp, “Onapp Cloud Plataform,” [Online]. Available: http://onapp.com/. [Acesso em 07 01 2013].

[55] “AWS pricing,” Amazon, [Online]. Available: http://aws.amazon.com/ec2/pricing/. [Acesso em 07 01 2013].

[56] “Google Compute Engine - pricing,” Google, [Online]. Available: https://cloud.google.com/pricing/compute-engine. [Acesso em 07 01 2013].

[57] “Rackspace Cloud Pricing,” Rackspace, [Online]. Available: http://www.rackspace.com/cloud/servers/pricing/. [Acesso em 07 01 2013].

[58] “Uol Cloud Computing,” UOL, [Online]. Available: http://www.uolhost.com.br/uol-cloud-computing.html. [Acesso em 07 01 2013].

[59] “Localweb,” Localweb, [Online]. Available: http://www.locaweb.com.br/produtos/cloud-server/precos.html. [Acesso em 07 01 2013].

[60] L. A. Zadeh, “Fuzzy Logic = Computing with Words,” *IEEE TRANSACTIONS ON FUZZY SYSTEMS,* vol. 4, n. 2, pp. 103-111, 1996.

[61] L. A. Zadeh, “From Computing with Numbers to Computing with Words-from manipulation of measurements to manipulation of perceptions,” *Int. J Appl Math. Comut Sci,* vol. 12, n. 3, pp. 307-324, 2002.

[62] C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controler - Part II,” *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS,* vol. 20, n. 2, pp. 419-435, 1990.

[63] T. A. Runkler, “Selection of Appropriate Defuzzification Methods Using Application Specific Properties,” *IEEE Transactions on Fuzzy Systems ,* vol. 5, n. 1, pp. 72-79, 1997.

[64] “JFuzzyLogic,” [Online]. Available: http://www.javafuzzylogic.org. [Acesso em 02 02 2012].

[65] D. H. Rao e S. S. Saraf, “Study of defuzzification methods of fuzzy logic controller for speed control of a DC motor,” em *IEEE International Conference on Power Electronics, Drives and Energy Systems for Industrial Growth*, 1996.

[66] S. Naaz, A. Alam e R. Biswas, “Effect of different defuzzification methods in a fuzzy baed load balancing application,” *International Journal of Computer Science Issues ,* vol. 8, n. 5, pp. 261-267, 2011.

[67] C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part I,” *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS,* vol. 20, n. 2, pp. 404-418, 1990.

[68] R. S. Montero, “Building Clouds with OpenNebula 1.4,” em *CESGA*, Santiago de Compostela, 2010.

[69] D. Lee e M. Yannakakis, “PRINCIPLES AND METHODS OF TESTING FINITE STATE MACHINES - A SURVEY,” *Proceedings of the IEEE,* vol. 84, n. 8, pp. 1090-1123, 1996.

[70] L. G. M. Alvim e A. J. d. O. Cruz, “A Fuzzy State Machine Applied to an Emotion Model for Electronic Game Characters,” em *IEEE International Conference on Fuzzy Systems* , 2008.

[71] J. Li, Z. Wang e Y. Zhang, “An Implementation of Artficial Emotion Based on Fuzzy State Machine,” em *Third Internationa Conference on Intelligent Human-Machine System and Cybernetics*, 2011.

[72] F. A. Unal e E. Khan, “A fuzzy finite state machine implementation based on a neural fuzzy system,” em *IEEE World Congress on Computational Intelligence*, 1994.

[73] P. Cingolani e J. Alcalá-Fedz, “jFuzzyLogic: A Robust and Flexible Fuzzy-Logic Inference System Language Implementation,” em *IEEE World Congress on Computational Inteligence*, Brisbane , Australia, 2012.

[74] *International Electrotechnical Commission technical committee industrial process measurement and control. IEC 61131 - Programmable Controllers - Part 7: Fuzzy control programming.,* IEC, 2000.

[75] W. Stallings, SNMP,SNMPv2,SNMPv3 and RMON 1 and 2, Addison-Wesley, 1999.

[76] “Netem - Network Emulation for Linux,” Linux Foundation , [Online]. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem. [Acesso em 1 10 2012].

[77] S. Hemminger, “Network Emulation with NetEm,” em *Linux Conf Au*, 2005.

[78] T. Wood, P. Shenoy, A. Venkataramani e M. Yousif, “Black-box and Gray-box Strategies for Virtual Machine Migration,” em *USENIX Symposium on Networked Systems Design USENIX Association & Implementation*, 2007.

[79] A. Singh, M. Korupolu e D. Mohapatra, “Server-Storage Virtualization: Integration and Load Balancing in Data Centers,” em *ACM/IEEE conference on Supercomputing*, 2008.

[80] “SLA Management Handbook: Volume 2 - Concepts and Principles,” TMForum , 2005.

[81] “Apache Derby,” Apache Foundation, [Online]. Available: http://db.apache.org/derby/. [Acesso em 1 10 2012].

[82] “Ubuntu,” Canonical, [Online]. Available: http://www.ubuntu.com . [Acesso em 1 10 2012].

[83] “Open SSH,” OpneBSD, [Online]. Available: http://www.openssh.com/. [Acesso em 01 10 2012].

[84] “Net-SNMP,” [Online]. Available: http://www.net-snmp.org/. [Acesso em 01 10 2012].

[85] “mongo DB,” mongodb ORG, [Online]. Available: http://www.mongodb.org/. [Acesso em 1 10 2012].

[86] “Apache HTTP Server,” Apache Foundation, [Online]. Available: http://projects.apache.org/projects/http_server.html. [Acesso em 01 10 2012].

[87] “PHP,” The PHP Group, [Online]. Available: http://php.net/. [Acesso em 01 10 2012].

[88] “Java SE,” Oracle , [Online]. Available: http://www.oracle.com/technetwork/java/javase/overview/index.html.

[89] “SNMP4J,” SNMP4J.org, [Online]. Available: http://www.snmp4j.org/. [Acesso em 01 06 2012].

[90] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan e R. Sears, “ Benchmarking cloud serving systems with YCSB,” em *Proceedings of the 1st*

ACM symposium on Cloud Computing, Indianapolis, 2010.

[91] “Nagios,” [Online]. Available: http://www.nagios.org. [Acesso em 01 10 2012].

[92] “OpenNMS,” [Online]. Available: http://www.opennms.org/. [Acesso em 01 10 2012].

[93] T. Emir, G. Pujolle, E. Jamhour e M. C. Penna, “An XML-based Model for SLA Definition with Quality and Performance Indicators,” em *Proc. of IEEE International Workshop on IP Operations and Management*, San Jos´e, USA, 2007.

[94] P. Cingolani e J. Alcala Fdez, “jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation,” em *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2012*, Montreal, Canada, 2012.

[95] C. A. Yfoulis e A. Gounaris, “Honoring SLAs on cloud computing services: a control perspective,” em *EUCA/IEEE European Control Conference*, 2009.

[96] J. O. Fitó, Í. Goiri e J. Guitart, “SLA-Driven Elastic Cloud Hosting Provider,” em *Parallel, Distributed and Network-Based Processing (PDP), 2010, 18th Euromicro International Conference on. IEEE, 2010.*, 2010.

10 Bibliografias Consultadas

GEORGE J. KLIR AND BO YUAN, “Fuzzy Sets and Fuzzy Logic Theory and Applications ”, Prentice Hall . 1995

PEDRYCZ, WITOLD AND GOMIDE, FERNANDO, “An Introduction to Fuzzy Sets” , The MIT Press 1998.

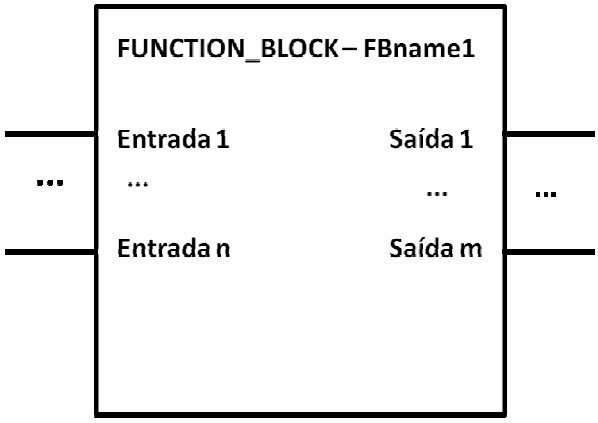
SILER, WILLIAN AND BUCKLEY, JAMES J , “Fuzzy Expert Systems and Fuzzy Reasoning” , John Wiley & Sons Inc. 2005.

E. A. LEE AND S. A. SESHIA, Introduction to Embedded Systems - A Cyber-Physical Systems Approach, http://LeeSeshia.org, 2011.

11 Anexos

Anexo A – Estrutura de um arquivo FCL

A linguagem FCL, “Fuzzy Control Language”, é dedicada a especificação de controladores Fuzzy. construção básica do arquivo FCL conforme definido em [74] é um bloco funcional, que define um controlador com entradas e saídas. Na figura abaixo encontra-se uma representação de bloco do controlador com n entradas em saídas.



O blocos de função em FCL podem ser armazenados em arquivos texto, normalmente com extensão “.fcl”. O blocos de código componentes do arquivo são :

- Cabeçalho de declaração

  FUNCTION_BLOCK FBname1

Declaração das variáveis de entrada

```
VAR_INPUT
   Entrada1 : varType;
   Entrada n : varitype;
END_VAR
```

Declaração das variáveis de saída

```
VAR_OUTPUT
   outputVarName   : varType;
END_VAR
```

Declaração dos conjuntos nebulosos e funções de pertinência associadas as variáveis de entrada

```
FUZZIFY Entrada1
  TERM terName := (0,1)  (2000,1);
END_FUZZIFY
```

Declaração dos conjuntos nebulosos e funções de pertinência associadas as variáveis de entrada

```
DEFUZZIFY  Saida1
   TERM   CONFORTABLE := (0,1) (0.25 ,1) (0.5, 0);
   METHOD :  metodo;
   DEFAULT := default value;
END_DEFUZZIFY
```

Declaração de regras

```
RULEBLOCK ruleBlockName
    RULE 1 : IF respTime IS LOW THEN cCondition IS CONFORTABLE;
    RULE 2 : IF respTime IS HIGH THEN cCondition IS ACCEPTABLE;
END_RULEBLOCK
```

Fecho de arquivo.

```
END_FUNCTION_BLOCK
```

Dentro de cada bloco , as declarações devem ser finalizadas com “;”. Maiores detalhes sobre a sintaxe e as opções da linguagem FCL podem ser encontradas na norma IEC 61131 - Programmable Controllers - Part 7: Fuzzy control programming.

Anexo B –Código FCL para definição das variáveis

```
FUNCTION_BLOCK cloudCondition

VAR_INPUT
   respTime : REAL;
   cpuUser   : REAL;
   slowTT    : REAL;
   users : REAL;
   rtt  : REAL;
   nVms : REAL;
   T    : REAL;
   deltaT : REAL;
END_VAR

VAR_OUTPUT
   cCondition   : REAL;
   resourceAlloc : REAL;
END_VAR

FUZZIFY T
  TERM VALID := (0,1)  (2000,1);
END_FUZZIFY

FUZZIFY T
  TERM SMALL:= (0,1)  (2000,1);
  TERM HIGH := (1500,0) (30000,1);
END_FUZZIFY

FUZZIFY nVms
   TERM FEW := (0,1) (3,0);
   TERM MANY := (2,0) (4,1) (6,0);
   TERM TOOMANY := (5,0) (7,1) (10,1);
END_FUZZIFY

FUZZIFY slowTT
   TERM FEW := (0,1) (3,0);
   TERM MANY := (2,0) (4,1) (5, 1) (7,0);
   TERM TOOMANY := (6,0) (8,1) (25,1);
END_FUZZIFY

FUZZIFY respTime
    TERM LOW :=(0,1) (20,1) (25,0);
    TERM HIGH := (20,0) (30,1) (50,1) (60,0);
    TERM VERYHIGH := (55,0) (65,1) (1000,1);
END_FUZZIFY

FUZZIFY rtt
    TERM LOW := (0,1) (10,1) (30,0);
    TERM MEDIUN := (20,0) (30,1) (100,1) (110,0);
    TERM HIGH := (100,0) (120,1) (200,1);
END_FUZZIFY

FUZZIFY cpuUser
    TERM VL := (0,1) (20,0);
    TERM L  := (0,0) (20,1) (40,0);
    TERM M := (20,0) (40,1) (60,0);
    TERM H  := (40,0) (60,1) (80,0);
    TERM VH  := (60,0) (80,1) (100,1);
```

```
END_FUZZIFY

FUZZIFY users
   TERM FEW := (0, 1) (5, 1) (10,0) ;
   TERM MEDIUM := (7.5,0) (13,1) (17.5,0);
   TERM MANY := (15, 0) (20,1) (25,1);
END_FUZZIFY

DEFUZZIFY  cCondition
   TERM   CONFORTABLE := (0,1) (0.25 ,1) (0.5, 0);
   TERM   ACCEPTABLE := (0.25,0)  (0.5,1) (0.75,0) ;
   TERM   UNACCEPTABLE :=  (0.5 , 0)  (0.75,1) (1 , 1);
   METHOD : COG;
   DEFAULT := 0;
END_DEFUZZIFY

DEFUZZIFY resourceAlloc
   TERM   UNDERPROVISIONED := (0,1) (0.25 ,  1) (0.5, 0);
   TERM   ADEQUATE :=  (0.25,0)  (0.5,1) (0.75,0) ;
   TERM   OVERPROVISIONED :=  ( 0.5, 0)  (0.75,1) (1,1);
   METHOD : COA;
   DEFAULT := 0;
END_DEFUZZIFY


RULEBLOCK RBL1
    RULE 1 : IF respTime IS LOW THEN cCondition IS CONFORTABLE;
    RULE 2 : IF respTime IS HIGH THEN cCondition IS ACCEPTABLE;
    RULE 3 : IF respTime IS VERYHIGH THEN cCondition IS UNACCEPTABLE;
    RULE 4 : IF users IS FEW AND nVms IS  FEW then resourceAlloc IS ADEQUATE;
    RULE 5 : IF (nVms IS MANY  OR nVms IS TOOMANY)  AND users IS FEW THEN resourceAlloc IS
OVERPROVISIONED;
    RULE 6 : IF (users IS MEDIUM AND nVms  IS FEW) AND respTime IS LOW THEN resourceAlloc IS
ADEQUATE;
    RULE 7 : IF (users IS MANY AND nVms  IS FEW) AND respTime IS LOW THEN resourceAlloc IS
ADEQUATE;
    RULE 8 : IF (nVms IS FEW AND respTime IS HIGH) AND rtt IS LOW THEN resourceAlloc IS
UNDERPROVISIONED;
    RULE 9 : IF (respTime IS LOW OR respTime is HIGH) AND slowTT IS MANY THEN cCondition IS
UNACCEPTABLE;
    RULE 10 : IF (respTime IS LOW OR respTime is HIGH) AND slowTT IS TOOMANY THEN
cCondition IS UNACCEPTABLE;

END_RULEBLOCK

END_FUNCTION_BLOCK
```

Anexo C –Código Fonte da Maquina de Estados Nebulosos

Aquivo FuFSM.java

```
import java.util.ArrayList;
import java.util.HashMap;

public class FuSM implements Runnable{


        private ArrayList<FuState>  stateArray;
        private ArrayList<FuTransition> transitionArray;
        private HashMap<String,Double> variableMap;
      private HashMap<String,Double> outputValuesMap;

      private  int period ;  // delay time between evaluations, effectively the


        public FuSM(){

                this.stateArray = new ArrayList<FuState>();
                this.transitionArray = new ArrayList<FuTransition>();
                this.variableMap = new HashMap<String,Double>();
                this.period = 5000;    // standard evaluation time of  5 minutes
                this.outputValuesMap = new HashMap<String,Double>();


        }

       public     FuSM(ArrayList<FuState>     states,     ArrayList     <FuTransition>
transitions, Integer  delay ){

                this.stateArray =  states;
                this.transitionArray = transitions;
                this.variableMap = new HashMap<String,Double>();
                this.period = delay;
              this.outputValuesMap = new HashMap<String,Double>();

        }

   @Override
   public void run(){

                while(true){
                        try{
                                Thread.sleep(500);
                        }
                        catch (Exception e){
                                System.err.println("Erro no sleep n");
                        }
```

```
                        this.evaluate();
                }


        }


        private void evaluate(){
                int i = 0;
                Long deltaT;
                Double pertAccum = 0.0;

                while ( i < transitionArray.size()){
                        Double                          orgPert                 =
stateArray.get(transitionArray.get(i).getOrg()).getPert();

                        deltaT                                                  =
stateArray.get(transitionArray.get(i).getOrg()).getInitTime();
                    Double                              destPert                =
transitionArray.get(i).transitionExecute(variableMap, orgPert,deltaT);
                            pertAccum = pertAccum + destPert;
                            stateArray.get(
transitionArray.get(i).getOrg()).outPertAdd(destPert);
                        stateArray.get(transitionArray.get(i).getDest()).
inPertAdd(orgPert*destPert);

                        i++;

           this.commit( pertAccum);
        }
}

   public void addState( FuState newState){
                this.stateArray.add(newState);
        }

    public void addTransition(FuTransition newTransition){
                this.transitionArray.add(newTransition);
         }

         private void commit( Double pertAccum){
                int i = 0;

                        pertAccum = 1.0;

                while(i < this.stateArray.size()){
                        this.stateArray.get(i).commit(pertAccum);
                        i++;
                }
                i =0;
```

```
                while(i < this.stateArray.size()){
                        this.stateArray.get(i).actionExecute();
                        i++;
                }
        }

        public void setVariable(String varName, Double varValue){
                this.variableMap.put(varName,varValue);
        }

        public void setPeriod(int delay){
                this.period  = delay;
        }

        public void addOutputVar(String name,Double varValue){
                this.outputValuesMap.put(name,varValue);
        }

        public Double getOutput(String name){
                return(this.outputValuesMap.get(name));
        }

        public void processOutputs(){



        }

        public void  reset( Integer initState){
                int i = 0;
                while ( i < stateArray.size()){
                        stateArray.get(i).reset();
                }

        }

}
```

Anexo C –Código fonte dos Estados Nebulosos

```
import net.sourceforge.jFuzzyLogic.FIS;
import java.util.*;

public class FuState{
        private  Integer stateId; // sateId attributed at creation
        private Double initialPert;
        private  Double   pert;
        private  Double   nextPert;// pertinency to be attributed in the next commit
        private  Double   outTpertSum;  // Sum of outgoing Transition pertinencies
        private  Double   inTpertSum; // Sum of incoming Trasition pertinencies
        private  String stateName;
        private  String  stateDescription;
        private  Long initTime;
      private  FuStateAction stateAction;
        private   static final Long STATE_TIMEOUT= 1000L;   // does not belong
here
     private boolean varActionExecuted;

        public FuState(){

        }

        public FuState(String name,Integer id, Double pertOrg, String description){
                this.stateId = id;
                this.pert = pertOrg;
                this.initialPert = pertOrg;
                this.nextPert = pertOrg;
                this.stateName = name;
                this.stateDescription = description;
                //this.initTime =  System.currentTimeMillis();
                this.initTime = 0L;
                this.inTpertSum= 0.0;
                this.outTpertSum=0.0;
                this.stateAction = null;
                this.varActionExecuted = false;
        }

        public FuState(String name,Integer id, Double pertOrg, String description,
FuStateAction action){
                this.stateId = id;
                this.pert = pertOrg;
                this.initialPert = pertOrg;
                this.nextPert = pertOrg;
                this.stateName = name;
                this.stateDescription = description;
                //this.initTime =  System.currentTimeMillis();
                this.initTime = 0L;
```

```
                this.inTpertSum= 0.0;
                this.outTpertSum=0.0;
                this.stateAction =  action;
                this.varActionExecuted = false;
        }

        public boolean actionExecuted(){
                return(this.varActionExecuted);
        }

    public void setPert(Double newpert){
                this.nextPert = newpert;
        }

        public void inPertAdd(Double inTpert){              //inTpert is the Transition
pertinency times the source State pertinency
             this.inTpertSum = inTpertSum + inTpert;
        }

        public void outPertAdd(Double outTpert){
                this.outTpertSum = outTpertSum+outTpert;  // outTpert contains only
the transition pertinency
        }
        public Double getPert(){
                return(this.pert);
        }

        public void setName(String name){
                this.stateName = name;
        }

        public String getName(){
                 return(this.stateName);
        }

        public void setDescription(String desc){
                this.stateDescription = desc;
        }

        public String getDescription(){
                return(this.stateDescription);
                }

        public void  setAction(String command){
                 this.stateAction = new FuStateAction(command);
        }

        public void commit( Double pertAccum){

                   if (pertAccum > 1){
```

```
                        this.pert=this.pert*(1-
this.outTpertSum/pertAccum)+this.inTpertSum/pertAccum;
                }
                else{

                        this.pert = this.pert * (1- this.outTpertSum)
+this.inTpertSum;
                }

                if(this.pert == 0.0 ){
                        this.initTime =  0L;  //resets  timer

                }
                else
                        this.initTime = this.initTime+1L;

                this.inTpertSum = 0.0;
                this.outTpertSum = 0.0;

        }

        public String toString(){
                return ( "State: "+this.stateName+"  id: "+this.stateId+" pert:
"+this.pert);
                }
        public Long getInitTime(){
                 return this.initTime;
        }
        public Long getDeltaT(){

                return(this.initTime);
        }

        public void timerRestart(){
                this.initTime= 0L;
        }
        public void reset(){
                 this.pert = initialPert;
                this.nextPert = initialPert;
                this.initTime = 0L;
                this.inTpertSum= 0.0;
                this.outTpertSum=0.0;
        }
                public void actionExecute(){
                if( this.stateAction != null)
                        this.varActionExecuted = this.stateAction.execute(this.pert);
        }

}
```

Anexo d –Código fonte dos Transições Nebulosas

```
import net.sourceforge.jFuzzyLogic.FIS;
import java.util.ArrayList;
import java.util.Map;
import net.sourceforge.jFuzzyLogic.plot.JDialogFis;
import net.sourceforge.jFuzzyLogic.rule.Rule;

public class FuTransition{

        private Integer orgState;
        private Integer destState;
        private FIS  transitionFis;
        private ArrayList<String> variables;
        private ArrayList<FuState> states;
      JDialogFis jdialog ;


        public  FuTransition(){
                this.orgState = null;
                this.destState = null;
                this.transitionFis = null;
                this.variables = new ArrayList<String>();
               this.states = new ArrayList<FuState>();
               jdialog = null;

        }

        public FuTransition(Integer orgState, Integer destState, String[]
myvariables, ArrayList<FuState> myStates,String fisDescriptiom){
                this.orgState = orgState;
                this.destState = destState;
                try{
                        this.transitionFis                                        =
FIS.createFromString(fisDescriptiom,false);


                }
                catch(Exception e){
                        System.out.println("Deu erro na criacao FuTransition \n");
                         System.out.println("Message "+e.getMessage());
                        //do nothing
                }

                this.variables = new ArrayList<String>();
                int i = 0;
                while (i < myvariables.length )
                        this.variables.add(myvariables[i++]);
```

```
                this.states = myStates;
               }

    public FuTransition(Integer orgState, Integer destState, String[]
myvariables,ArrayList<FuState> myStates, FIS fis){
                this.orgState = orgState;
                this.destState = destState;
                this.transitionFis = fis;

                this.variables = new ArrayList<String>();
                int i = 0;
                while (i < myvariables.length )
                        this.variables.add(myvariables[i++]);
                 this.states = myStates;
               }

      public Double transitionExecute(Map<String,Double> variablesValues,
Double orgPert){
            int i = 0;
            transitionFis.setVariable("currState",orgPert);
                while (i < this.variables.size() ){

transitionFis.setVariable(variables.get(i),variablesValues.get(variables.get(i)));
                        i++;
                }
                if(states.get(orgState).actionExecuted())
                        transitionFis.setVariable("activated",1);
                else
                        transitionFis.setVariable("activated",0);
                transitionFis.setVariable("timer",0);
            transitionFis.evaluate();
            return(transitionFis.getVariable("action").getValue());
        }

       public Double transitionExecute(Map<String,Double> variablesValues,
Double orgPert, Long deltaT){
            int i = 0;
            transitionFis.setVariable("currState",orgPert);
                while (i < this.variables.size() ){

transitionFis.setVariable(variables.get(i),variablesValues.get(variables.get(i)));
                        i++;
                }
                 if(states.get(orgState).actionExecuted())
                        transitionFis.setVariable("activated",1);
                else
                        transitionFis.setVariable("activated",0);
                transitionFis.setVariable("deltaT", deltaT);
            transitionFis.evaluate();
```

```
                return(transitionFis.getVariable("action").getValue());
        }

        public Integer getOrg(){
                        return(this.orgState);
        }

    public Integer getDest(){
                return(this.destState);
    }


}
```