Alexandre Augusto Flores

UM MODELO PARA O SUPORTE DE TOMADA DE DECISÃO NO GERENCIAMENTO DE NUVENS COMPUTACIONAIS

Dissertação submetida ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de mestre em Ciência da Computação.
Orientador: Prof. Dr. Carlos Becker Westphall

Florianópolis
2016

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Flores, Alexandre Augusto
UM MODELO PARA O SUPORTE DE TOMADA DE DECISÃO NO GERENCIAMENTO DE NUVENS COMPUTACIONAIS / Alexandre Augusto Flores ; orientador, Carlos Becker Westphall - Florianópolis, SC, 2016.
98 p.

Dissertação (mestrado profissional) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Computação em Nuvem. 3. Teoria da Decisão. 4. Computação Autonômica. I. Westphall, Carlos Becker. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Alexandre Augusto Flores

UM MODELO PARA O SUPORTE DE TOMADA DE DECISÃO NO GERENCIAMENTO DE NUVENS COMPUTACIONAIS

Esta Dissertação foi julgada adequada para obtenção do Título de "Mestre em Ciência da Computação", e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, 10 de fevereiro de 2016.

_____
Profa. Carina Friedrich Dorneles, Dra.
Coordenadora do Curso

Banca Examinadora:

_____
Prof. Carlos Becker Westphall, Dr.
Orientador
Universidade Federal de Santa Catarina

_____
Prof.ª Alfredo Goldman vel Lejbman, Dr.
Universidade de São Paulo (Videoconferência)

_____
Prof. Carlos Barros Montez, Dr.
Universidade Federal de Santa Catarina

_____
Prof. Ricardo Alexandre Reinaldo De Moraes, Dr.
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus colegas de classe e aos meus queridos pais.

AGRADECIMENTOS

Por ter viabilizado o meu ingresso, ter me aceitado como seu orientando, ter me auxiliado durante o curso deste mestrado, agradeço ao meu orientador, professor Carlos B. Westphall.

A minha querida e atenciosa amiga Jaqueline L. Horbach, doutoranda em matemática, que além de ter me ajudado nos estudos para o PosComp, revisou e aperfeiçoou o modelo matemático proposto nessa dissertação.

Ao colega e amigo Rafael Mendes, agradeço pelo esbarrão nos corredores do INE que resultou na apresentação ao professor Carlos e posteriormente ao ingresso no mestrado. Também agradeço por todo tempo dispensado nas inúmeras horas de discussões e questionamentos.

Aos meus colegas do LRG que contribuíram em várias etapas deste mestrado, sempre solícitos, contribuindo e compartilhando seus conhecimentos.

Por ter tolerado as ausências, dado o suporte emocional e me acompanhado durante toda essa jornada, agradeço a minha amada namorada Bruna Carolina.

Aos amigos, Rafael Rauber, que me acompanhou nas disciplinas especiais e no PosComp, e Felipe Godói., meu sócio, que compreendeu e apoiou liberando-me de afazeres para que pudesse me dedicar a esta dissertação.

Também aos colegas e amigos da Eletrosul que me apoiaram e contribuíram em várias etapas deste mestrado.

A computer would deserve to be called
intelligent if it could deceive a human into
believing that it was human.
(Alan Turing, 1943)

RESUMO

Muito esforço tem sido feito visando se obter um gerenciamento cada vez mais eficiente do ambiente de nuvem computacional. Neste sentido surgem soluções autonômicas, auto gerenciáveis e que tomam decisões acerca do ambiente gerenciado. Estas soluções, apesar de se mostrarem eficientes nos seus propósitos, frequentemente focam em aspectos específicos da nuvem e, quando utilizadas em conjunto, podem ser contraditórias e ineficientes. Desta forma, é necessário um gerenciador holístico que possa efetuar um gerenciamento integrado, tomando decisões em todo o ambiente. Para tanto, o gerenciador precisa entender o ambiente em que ele está inserido, além de definir como e quando as decisões serão tomadas. Este trabalho propõe um modelo holístico para subsidiar a tomada de decisão de um gerenciador de nuvem. Este modelo utiliza o entendimento da nuvem como um jogo estocástico de múltiplos jogadores e permite ao gerenciador buscar equilíbrios ótimos. Para validar o modelo foi construído um simulador e foram conduzidos experimentos. O método utilizado consistiu de pesquisa bibliográfica exploratória; definição do modelo valendo-se de expressões matemáticas; especificação do simulador utilizando diagramas; e validação do modelo com dois experimentos.

Palavras-chave: Computação em nuvem. Teoria da decisão. Computação Autonômica.

ABSTRACT

Much effort has been done aiming to obtain a more efficient management of cloud computing environment. In this sense, arise autonomic self-managed solutions that make decisions about the managed environment. Although these solutions show efficiency in their purposes, often they focus on specific aspects of cloud and when are used together they may be contradictory and inefficient. Thus, a holistic manager that can make an integrated management making decisions throughout the environment is needed. Therefore, the manager must understand the environment in which it is inserted and also define how and when decisions are taken. This work proposes a holistic model to support the decision making of a cloud manager. This model uses the understanding of Cloud as a stochastic multiplayer game and allows the manager to search for an optimal equilibrium. To validate the model was built a simulator and experiments were conducted. The method used consisted of bibliographical research; model definition using mathematical expressions; simulator specification using diagrams and model validation with two experiments.

Keywords: Cloud Computing. Decision theory. Autonomic Computing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BLOs - Objetivos em nível de negócio (*Business-Level Objectives*)
ECA – Evento, Condição e Ação
IBM - *International Business Machines*
HTML – Linguagem de Marcação de Hipertexto (HyperText Markup Language)
IaaS - Infraestrutura como serviço (*Infrastructure as a Service)*
MAPE-K – *Framework* IBM (*Monitor-Analyze-Plan-Execute over a Knowledge*)
MDP – Processo de decisão Markoviano (*Markov Decision Process*)
MV – Máquina Virtual (*Virtual Machine*)
PaaS – Plataforma como um serviço (*Platform as a Service*)
QoS – Qualidade de Serviço *(Quality of Service)*
SaaS – *Software* como um serviço (*Software as a Service)*
SLO - Objetivos de Nível de Serviço (*Service Level Objective*)
SLA – Acordo de nível de serviço (*Service Level Agreement)*
SMG – Jogos estocásticos de múltiplos jogadores (*Stochastic multiplayer game*)

LISTA DE SÍMBOLOS

$\tilde{a}_t$– Ação de uma determinada parte interessada no tempo *t*
A_t – Ações no tempo *t*
$\tilde{A}_t$– Ações que uma parte interessada pode executar no tempo *t*
c_t – Função custo
d – Variável monitorada
D – Dimensões
D_t – Dimensões no tempo *t*
df – Função que retorna PD_t(dimensões filhas), dada uma dimensão pai.
dp – Função que recebe uma dimensão filha e retorna o pai
du_t– Função que relaciona partes interessadas e dimensões
dx – Função que relaciona D_t com X_{dt}
E_t – Eventos no tempo *t*
pa – Função que retorna as ações que uma parte interessada pode executar em um determinado tempo
p_t – Função que retorna a probabilidade de uma ação levar o sistema a um determinado estado
PD_t – Conjunto das partes que representa as dimensões filhas.
PS_{t+n} – Conjunto das partes que representa os possíveis estados futuros resultantes de uma ação
R_0 – Função que retorna as ações permitidas em um estado
R_1 – Função que retorna os eventos permitidos em um estado
s_t – Estado no tempo *t*
S_t – Conjunto com todos os estados possíveis no tempo *t*
S_{t+n} – Conjunto com todos os estados possíveis no tempo $t+n$
U – Partes interessadas no tempo *t*
v_t – Função de interesse
X_{dt} – Valores possíveis de uma dimensão no tempo *t*
$\mathcal{X}_{Dt}$ – Conjunto com todos X_{dt} no tempo *t*
γ – Interesse de uma parte interessada

SUMÁRIO

1 INTRODUÇÃO

A utilização generalizada de dispositivos de computação introduziu uma mudança drástica na maneira que a computação é produzida, distribuída e consumida. Escalabilidade, eficácia e custo-benefício são características cada vez mais demandadas e, para suprir essas necessidades, novas tecnologias e paradigmas foram criados recentemente (MUPPALA *et al.*, 2011).

Benefícios como a elasticidade e baixo custo operacional tornaram a Computação em Nuvem (*Cloud Computing*) uma forma popular para entrega de TI. O conceito de computação em nuvem se destaca como um dos paradigmas mais promissores da última década.

Hoje a computação em nuvem está presente, de maneira direta ou indireta, na realidade de quase todos os usuários da internet. Tamanha popularização resulta em *data centers* com milhares de servidores e estruturas altamente complexas (MASTELIC *et al.*, 2014).

Administradores de serviços que rodam em nuvens comerciais criam programas complexos buscando a orquestração e gerencia dos seus recursos (RANJAN *et al.*, 2015). Esses programas buscam por ótimos locais, de cada consumidor, ignorando assim a existência de outros consumidores que compartilham os mesmos recursos.

Devido a complexidade e buscando por nuvens cada vez mais eficientes, surgem as nuvens computacionais com funções autonômicas. Conforme (KEPHART; CHESS, 2003), sistemas autonômicos exibem a habilidade de autogerenciamento e na nuvem normalmente estes focam em elementos e fases específicas. Contudo, Corradi (2014) afirma que gerenciamento deveria estar presente em toda nuvem computacional.

Neste sentido, este trabalho busca desenvolver um modelo que suporte a tomada de decisão em nuvem, subsidiando a criação de um gerenciamento autonômico de todos os elementos. Acredita-se que esse tipo de gerenciamento resultará em nuvens mais eficientes e escaláveis.

Vários autores destacam a dificuldade da realização de testes em sistemas computacionais em nuvem de grande porte devido ao custo, à rigidez física da estrutura, e à dificuldade para reproduzir experimentos em condições controladas. Dito isto, a validação do modelo proposto se dará com o uso da simulação. As principais ferramentas utilizadas para realizar simulações de nuvens foram comparadas neste trabalho e optou-se pelo uso e extensão do *framework* Cloudsim (CALHEIROS *et al.*, 2011). Para se tirar um melhor proveito de relatórios, a ferramenta CloudReports (SÁ; SOARES; GOMES, 2011) também foi utilizada e extendida.

1.1 CONTEXTUALIZAÇÃO DO PROBLEMA

A dificuldade de administrar sistemas computacionais vai além de administrar softwares isolados e a necessidade de integrar ambientes heterogêneos pela internet introduz novos níveis de complexidade, superando os níveis da capacidade humana e resultando nos sistemas autonômicos (KEPHART; CHESS, 2003).

Apesar de vários trabalhos proporem o gerenciamento autonômico de nuvens, em geral, eles focam somente em partes ou em determinados aspectos da nuvem. Essas propostas muitas vezes têm objetivos distintos, sendo frequentemente contraditórias e também não integráveis.

Neste sentido, Beloglazov, Abawajy e Buya (2012) afirmam ser necessária uma solução holística, que envolva todos os elementos, para se ter um gerenciamento completo e eficaz.

1.2 HIPÓTESE

A aplicação de uma abordagem holística derivada da teoria dos jogos na criação de um modelo que suporte a tomada de decisão de um gerenciador autonômico de nuvens é adequada para se obter um gerenciamento multiobjetivo eficaz.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Desenvolver um modelo holístico de suporte à tomada de decisão que possa ser utilizado por um gerenciador autonômico de nuvens computacionais.

1.3.2 Objetivos Específicos

- Comparar trabalhos correlatos e elencar as suas principais contribuições visando caracterizar as necessidades do modelo proposto.
- Especificar com base na pesquisa realizada um modelo

que suporte a tomada de decisão em nuvens, definindo suas funções e características.
- Estender o simulador CloudSim implementando um gerenciador que utilize o modelo proposto de modo a validar a sua eficácia.
- Realizar simulações em cenários pequenos, médios e grandes e comparar os resultados obtidos com e sem o uso do modelo proposto.

1.4 MÉTODO

Em um primeiro momento foi feita uma pesquisa bibliográfica exploratória, definindo o estado da arte do gerenciamento autonômico de nuvens de computadores e os principais conceitos relacionados. Também por pesquisa bibliográfica exploratória foram elencados os trabalhos correlatos e suas características, de modo a subsidiar a construção do modelo proposto.

Para a definição do modelo, foram utilizadas definições matemáticas dos elementos e funções. Para a validação, o modelo foi implementado em JAVA e um simulador foi criado com o uso das ferramentas CloudSim e CloudReports.

Dois experimentos foram conduzidos visando validar o uso e a eficácia do modelo. Nos experimentos um gerenciador da nuvem utiliza o modelo para alocação e migração de MVs. No primeiro, as funções de interesse (especificadas adiante) consideram a afinidade entre as MV e buscam reduzir o número de servidores ligados. Já no segundo, as funções de interesse consideram critérios mais diversos. Neste o gerenciador é considerado como parte interessada e ajusta a frequência com que irá monitorar o ambiente.

1.5 ORGANIZAÇÃO DO TRABALHO

Esta dissertação está organizada como segue. A seção dois apresenta uma revisão das definições e conceitos relacionados com o tema, além da caracterização da tomada de decisão no gerenciamento de nuvens. Na terceira seção, os principais trabalhos correlatos são elencados, comparados e, com base nestes, as características desejadas para a construção do modelo são relacionadas. Em seguida, na seção quatro, o modelo holístico para subsidiar a tomada de decisão em nuvens computacionais autonômicas é definido. A seção cinco apresenta a validação do modelo, a qual passa por uma verificação da sua

aderência, avaliação das suas características e testes com a construção de um simulador. De forma conjunta, também são apresentados os experimentos e os resultados obtidos com o uso do modelo nas simulações. Por fim, são apresentadas as conclusões obtidas e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são revisados os principais assuntos necessários para a compreensão dos trabalhos correlatos e do modelo proposto. Primeiramente são revisados os conceitos de computação em nuvem. Em seguida a computação autonômica e o MAPE-K são abordados e por último assuntos relacionados a teoria da decisão e teoria dos jogos são revisados.

2.1 COMPUTAÇÃO EM NUVEM

A computação em nuvem é um paradigma que tem passado por várias definições, muitas vezes distintas, desde a sua concepção. Com o objetivo de apresentar a evolução das definições e subsidiar a visão utilizada neste trabalho, nesta seção será feita inicialmente uma revisão cronológica do conceito.

Foster et al. (2008) possuem a seguinte visão para computação em nuvem:

> Um paradigma de computação altamente distribuída dirigida pela economia de escala, na qual um *pool* abstrato, virtualizado, escalável dinamicamente e gerenciado de poder computacional, armazenamento, plataformas e serviços são entregues por demanda a consumidores externos pela internet.

A definição de Foster et al. é relevante principalmente por duas razões: primeiramente, define nuvem como um paradigma, e em segundo lugar, entende a influência da economia na nuvem.

Em Buyya et al. (2009), a definição de computação em nuvem está em consonância com a definição de Foster et al, porém mais completa. Buyya et al. correlacionam a computação em nuvem às *commodities* como energia e água. Definem também que a infraestrutura computacional é entregue para usuários de forma que aplicações possam ser acessadas de qualquer lugar e por demanda.

Buyya et al. não deixam de fora a importância econômica da computação em nuvem, reconhecendo que provedores de computação em nuvem são incentivados pelo lucro obtido da cobrança de seus usuários, enquanto os consumidores, como empresas, estão interessados em eliminar ou reduzir seus custos associados com a produção interna destes serviços.

Posteriormente, o *National Institute of Standards and Technology*

(NIST), (MELL; GRANCE, 2011), faz a seguinte definição de Computação em Nuvem:

> Computação em nuvem é um modelo para permitir de maneira ubíqua, conveniente e por demanda, o acesso, através de rede, a um *pool* compartilhado de recurso (exemplo, redes, servidores, armazenamento, aplicações e serviços) os quais podem ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou interação do provedor dos serviços. Este modelo de nuvem é composto de cinco características essenciais, três modelos de serviço e quatro modelos de *deployment*.

A definição do NIST vem no sentido de identificar um mínimo de padrão e arquitetura, definindo aspectos importantes e servindo para comparação e discussão de serviços de nuvem e estratégias de implantação. Desenvolvido para as agências do governo dos Estados Unidos, o resultado esperado era que estas pudessem criar, comprar e comparar serviços de nuvem com mais facilidade.

As principais características mencionadas na definição são:

- Serviço pela demanda: um consumidor pode unilateralmente provisionar mais recursos;
- Compartilhamento dos recursos: normalmente de processamento, armazenamento e rede;
- Amplo acesso à rede: serviço disponível pela rede de forma a poder ser utilizado por clientes em plataformas distintas;
- Rápida elasticidade: recursos podendo ser alocados e liberados rapidamente e;
- Serviço medido: automaticamente com algum nível de abstração, sendo reportado aos consumidores e provedores.

Como apresentado, a computação em nuvem mudou nos últimos anos de uma visão econômica para uma visão pragmática. A definição do NIST tem como objetivo a comparação de serviços, contudo, eles também afirmam que seus modelos de serviço e *deployment* não tem o objetivo de prescrever ou limitar nem um método específico.

Conforme será apresentado posteriormente, o gerenciamento em nuvem pode ser modelado como um jogo. Sabendo-se que a teoria dos jogos tem a sua origem na economia, a visão de computação em nuvem

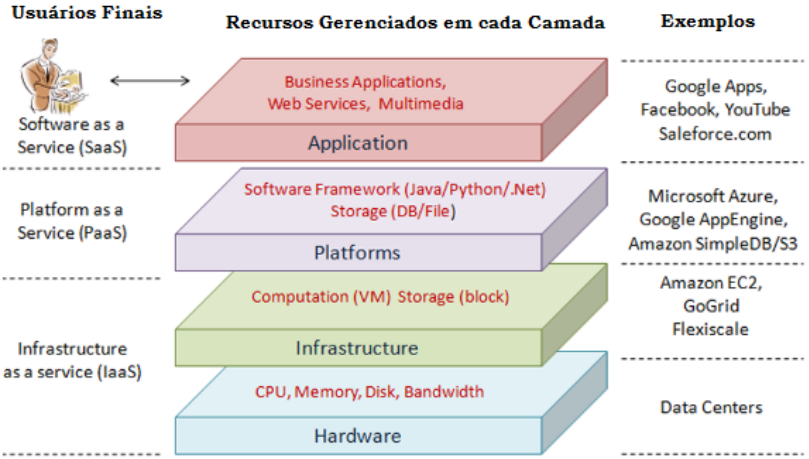
que inspira e norteia este trabalho é:

> Atividade econômica que foca na produção, distribuição e consumo em massa da computação. Esta computação tem recursos lógicos e físicos abstratos e fronteiras comerciais proeminentes entre partes interessadas que produzem e consomem esta (FLORES *et al.*, 2015).

2.1.1 Arquitetura e Modelo de Serviço

A arquitetura da nuvem computacional pode ser dividida em quatro camadas: *hardware*, infraestrutura, plataforma e aplicação. A Figura 1 ilustra essas camadas e as principais soluções existentes nas camadas acima da camada de *hardware*, sendo que nesta camada estão os recursos como servidores, elementos de rede e outros.

Figura 1 - Componentes da nuvem, adaptado de (ZHANG; CHENG; BOUTABA, 2010)

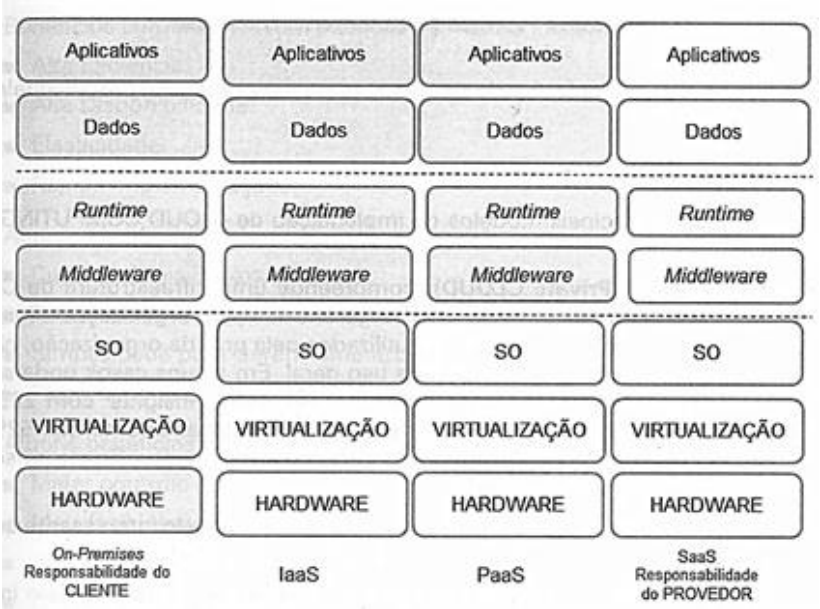


Ao observar a Figura 1, verifica-se que tipicamente ocorre um empilhamento das camadas conforme o modelo de serviço oferecido. No serviço de IaaS (Infraestrutura como serviço), máquinas virtuais e armazenamento são tipicamente oferecidos aos usuários. No PaaS, plataforma como um serviço, os usuários utilizam soluções na nuvem para disponibilizar serviços para outros usuários e os serviços de IaaS frequentemente são utilizados para prover os serviços de plataforma.

Quando analisamos o SaaS, software como um serviço, verificamos que ele pode ser implementado utilizando PaaS, o qual pode ser provido por IaaS. Contudo, cabe-se ressaltar que isso não é mandatório e que um SaaS pode ser implementado sem utilizar as camadas relacionadas.

Os elementos sobre os quais o usuário e o provedor do serviço possuem controle se alteram conforme o modelo de serviço. A Figura 2 apresenta os elementos que podem ser controlados conforme o modelo de serviço, em que os elementos em cinza são responsabilidades do usuário e em branco são responsabilidades do provedor. A primeira coluna apresenta a abordagem tradicional, em que não existe uma nuvem. Conforme as camadas são empilhadas, os provedores da nuvem vão tendo mais responsabilidades e os usuários menos, até o momento em que, no SaaS, os usuários só se preocupam em utilizar o software.

Figura 2 - Responsabilidades de controle (VERAS; TOZER, 2012)



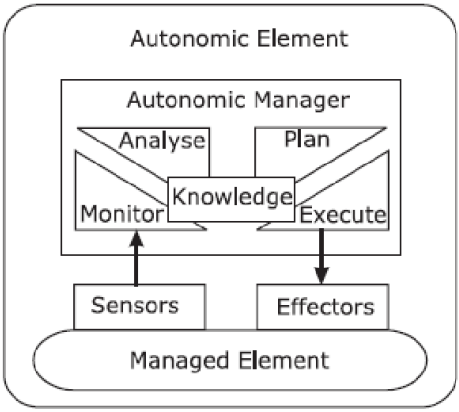
2.2 COMPUTAÇÃO AUTONÔMICA E MAPE-K

O conceito da computação autonômica é baseado no sistema nervoso humano que é autonômico e, por exemplo, coordena a frequência cardíaca e o corpo, liberando assim o cérebro consciente

deste fardo de lidar com esta e outras funções, ainda que vitais, de baixo nível (KEPHART; CHESS, 2003).

Neste sentido, a IBM sugeriu um modelo de referência para ciclos de controle autonômico, conhecido como MAPE-K (Figura 3 - MAPE-K, modelo de referência para controle autonômico (HUEBSCHER; MCCANN, 2008)). Este consiste dos elementos de Monitoramento, Análise, Plano, Execução e Conhecimento (KEPHART, 2003). Com as referidas fases, propõe-se obter as habilidades de autoconfiguração, auto-otimização, autoproteção e auto-reparo.

Figura 3 - MAPE-K, modelo de referência para controle autonômico (HUEBSCHER; MCCANN, 2008)



Utilizando como base Huebscher e Maccan (2008), os elementos do MAPE-K podem ser descritos da seguinte forma:

- *Managed Element* representa qualquer elemento a ser gerenciado, podendo ser um sistema, um hardware, um servidor web ou banco de dados o qual obtém comportamento autonômico sendo acoplado ao *Autonomic Manager*.
- *Sensors* coletam as informações sobre o *Managed Element*.
- *Effectors* realizam as alterações no *Managed Element*. Essas alterações podem ser, por exemplo, ajustes em parâmetros de configurações.
- *Autonomic Manager* utiliza as informações obtidas pelos *sensors* para monitorar e executar alterações no *Managed Element*. Idealmente, ele pode ser configurado utilizando objetivos de alto

nível. Com esses objetivos e conhecimento, normalmente da arquitetura, ele pode monitorar, analisar, planejar e executar mudanças, utilizando os *effectors*.

2.2.1 Gerenciador Autonômico

O gerenciador autonômico proposto pela IBM deve apresentar as habilidades de autoconfiguração, auto-otimização, autoproteção e auto-reparo. Para tanto, atua monitorando, analisando, planejando e executando, se valendo do conhecimento que possui. Essas ações são tão importantes que são denominadas por fases em um ciclo de controle (HUEBSCHER; MCCANN, 2008).

Monitoramento: essa fase compreende a captura de propriedades do sistema que são importantes para o autogerenciamento. A captura é realizada por sensores os quais podem ser software ou hardware. As propriedades monitoradas podem ser latência de rede, tempo de resposta de um serviço, disponibilidade ou qualquer critério relevante para o gerenciador autonômico.

Análise: a fase de análise compreende a transformação dos dados em informações. Com os dados recebidos do monitoramento e com o conhecimento sobre o sistema a análise identifica o estado atual do sistema e o que pode ser feito.

Planejamento: é nesta fase em que é decidido o que deve ser feito. O resultado desta fase é um plano com a prescrição de que ações tomar.

Execução: a responsabilidade desta fase é cumprir o plano, resultado da fase de planejamento.

2.2.2 Classificação

Visando classificar gerenciadores autonômicos, Huebscher e Mccann (2008) definem os quatro graus de autonomicidade (*autonomicity*) que servem para classificar o foco e a arquitetura aplicada. Os referidos graus são:

- *Support*: foca em um aspecto particular ou componente de uma arquitetura para ajudar na melhora do desempenho de toda arquitetura.
- *Core*: o autogerenciamento envolve o núcleo central da aplicação. É uma solução *full end-to-end* (atua em todas as fases do processo).

- *Autonomous*: também é uma solução *full end-to-end*, mas o sistema é mais inteligente e mais capaz de se adaptar ao ambiente.
- *Autonomic*: é o nível mais completo onde os interesses de alto nível, como SLAs, SLOs, objetivos de negócio são considerados no gerenciamento.

Outra classificação é proposta em (LALANDA; MCCAIN; DIACONESCU, 2013) para tomada de decisão em sistemas autonômicos:

- *Rule-based*: a base de conhecimento é implementada por regras de eventos-condições ações (ECA). Apesar de ser rápido, é muito simplista;
- *Model-based*: o sistema mantém um modelo dos elementos gerenciados e tem informações sobre o estado do sistema;
- *Goal-based*: existe um mecanismo explícito que expressa os objetivos do sistema.
- *Utility-based*: utiliza uma função de utilidade para comparar e escolher estados do sistema.

2.3 TEORIA DA DECISÃO

A teoria da decisão, conforme o próprio nome diz, está relacionada com o problema de se tomar decisões dentro de determinadas condições e tendo alguns conhecimentos prévios. Ela está presente em várias áreas como economia, matemática, estatística e computação.

O ponto central da teoria da decisão é a tomada de decisão sobre a incerteza. Esta utiliza o conhecimento estatístico para basear as decisões a serem tomadas, buscando sempre pela melhor decisão.

Um segundo tipo de conhecimento relevante para a tomada de decisão é a consequência de suas ações. Esse conhecimento frequentemente é traduzido em uma função. Essa função pode se chamar de função de perda, utilidade, ganho, a depender da área e do autor.

Em muitas decisões, calcular e comparar o retorno que uma ação pode gerar não é trivial. Para exemplificar, utilizamos o exemplo de (BERGER, 2013).

> Suponha que você tem a oportunidade de fazer uma tarefa que você não gosta por $100. Talvez, esse valor seja o suficiente para que você faça a tarefa. Contudo, se antes de ter essa oportunidade você ganhar um milhão de dólares, pouco

> provavelmente você se sujeitaria a fazer a mesma tarefa por $100.

O exemplo acima exemplifica a necessidade do conceito de utilidade. Tomar uma decisão está relacionado a um contexto e isso não pode ser desprezado. Por esse motivo, neste trabalho adota-se o termo função utilidade e não função de perda ou ganho.

Podemos dizer que a teoria da decisão está relacionada a tomada de decisão de agentes individuais. Quando se incrementam outros agentes e se preocupa como as decisões dos agentes estão relacionadas, temos a teoria dos jogos.

2.3.1 Teoria dos Jogos

A teoria dos jogos estuda fenômenos da tomada de decisão em modelos denominados jogos. Nestes, ocorrem jogadas sendo que as ações[1] de cada jogador pode afetar as decisões e resultados de outros jogadores. Para o estudo, podem ser utilizados modelos matemáticos, estatísticos e probabilísticos.

Uma importante classificação dos jogos diz respeito a soma de todas as recompensas recebidas pelos jogadores. Sendo zero, estes são classificados como *zero-sum games*, ou seja, os jogadores só fazem pagamentos entre si. Sendo diferente de zero, são chamados de *non-zero-sum games*, onde o comportamento dos jogadores irá alterar o total das recompensas (VON NEUMANN; MORGENSTERN, 2007).

Outra classificação relevante é a distinção entre jogos cooperativos e jogos competitivos, ou não cooperativos. O jogo pedra-papel-tesoura, por exemplo, é competitivo, pois seus jogadores somente buscam o resultado individual. Já cooperativos os jogadores podem combinar estratégias e obter um resultado melhor se colaborarem entre si. Um exemplo de jogo que pode ser cooperativo é o Dilema do Prisioneiro[2].

Vários princípios de otimização podem ser aplicado na teoria dos

[1] A literatura frequentemente chama tomada de decisão de ação (BERGER, 2013).

[2] No jogo do Dilema do prisioneiro dois suspeitos são interrogados pela policia em salas separadas. Se um confessa e o outro não, quem confessa recebe a liberdade e o outro é condenado. Se os dois confessam, os dois são condenados e se os dois não confessam, os dois são levemente punidos. Se o jogo for cooperativo os jogadores vão poder combinar resultados para então escolher não confessar.

jogos, os dois mais conhecidos são o equilíbrio de Nash e o ótimo a Pareto. Também é importante citar os jogos de Stackelberg, os quais são um tipo específico de jogos.

2.3.1.1 Jogos Estocásticos de Múltiplos Jogadores

Um critério importante na classificação dos jogos diz respeito ao número de jogadores. Muitos jogos consideram um número fixo de jogadores. Contudo, quando o número de jogadores é livre, os jogos são chamados de jogos de múltiplos jogadores. Uma classe de jogos muito estudada e relevante é a dos Jogos Estocásticos de Múltiplos Jogadores (SMG).

Os problemas de decisão podem ser determinísticos ou estocásticos. Em problemas determinísticos, a incerteza é considerada desprezível e não é considerada. Em problemas estocásticos, os efeitos da incerteza são modelados e estão presentes através de elementos e variáveis aleatórias (MURTHY; JACK, 2014).

2.3.1.2 Equilíbrio de Nash

Derivando do conceito de equilíbrio *Standard* da economia, John Nash (1950) propõe o equilíbrio de Nash. Este equilíbrio descreve uma situação nos jogos em que nenhum jogador consegue melhorar o seu resultado alterando unilateralmente a sua estratégia.

Exemplo, se um jogo envolve dois jogadores, X e Y, teremos o equilíbrio de Nash se a estratégia adotada por X é a melhor dada a estratégia adotada por Y e a estratégia adotada por Y é a melhor dada a estratégia adotada por X.

2.3.1.3 Ótimo a Pareto

O ótimo a Pareto ocorre quando pelo menos um jogador não consiga ser beneficiado sem que nenhum outro seja prejudicado. Trata-se de um conceito que busca uma eficiência social, relevante para todos os jogadores.

2.3.1.4 Stackelberg

O modelo de Stackelberg foi criado inicialmente para definir uma estrutura de competição entre duas empresas e foi definido inicialmente em (STACKELBERG, 1934).

Para a teoria dos jogos, se refere a um jogo estratégico em que existem dois jogadores, o líder e o seguidor. Os jogadores tomam suas ações em sequência e o líder toma a primeira ação sabendo que o seguidor irá conhecer a ação tomada.

2.3.2 Processo de decisão Markoviano

O processo de decisão Markoviano (*Markov Decision Process* – MDP) é um processo de controle estocástico comumente de tempo discreto. A modelagem do MDP prevê a tomada de decisão sequencial em que a incerteza está presente. Ele provê uma modelagem matemática utilizando épocas de decisão, ações, estados do sistema, funções de transição e funções de recompensa (PUTERMAN, 2014).

O estado do sistema é uma representação do ambiente em um determinado tempo *t*. Trata-se de uma representação estática, similar a uma fotografia, e é representado por uma tupla de valores de variáveis do sistema controlado.

Genericamente falando, o MDP codifica a interação entre um agente e o seu ambiente onde cada ação leva o sistema a um novo estado com certa probabilidade (determinada pelas funções de transição). A escolha de uma ação gera uma recompensa determinada pela função de recompensa.

A saída do processo de decisão são políticas. As políticas são uma prescrição de quais ações tomar em cada possível tempo de decisão futuro (sequências de ações). Sendo que os tomadores de decisão normalmente buscam por políticas que são ótimas em algum sentido.

2.3.3 A Decisão no gerenciamento de nuvem

Segundo Lozovanu e Pickl (2009), o gerenciamento de um ambiente de computação em nuvem pode ser classificado como um problema de controle multiobjetivo e multivariável em um sistema de tempo discreto. A dinâmica de um sistema de computação em nuvem permite que o controle possa ser assumido por vários atores onde cada um busca aperfeiçoar os seus resultados ao longo da trajetória. Essa trajetória é determinada por vetores de parâmetros de controle escolhidos por todos os envolvidos de forma coletiva.

Neste sentido, autores como Ardagna, Panicucci e Passacantando (2011) e Palmieri et al. (2013) implementam soluções em que a tomada

de decisão para o gerenciamento da nuvem está baseada na teoria dos jogos. O entendimento é que o resultado global é alterado conforme as decisões dos jogadores, resultando em um jogo *non-zero sum*. De forma conjunta, uma abordagem estatística pode ser utilizada resultando em um *Stochastic Multiplayer Game* (SMG). Nesta classe de problemas, os princípios de otimização de Nash, Pareto e Stackelberg são frequentemente utilizados com modelos da teoria dos jogos.

Outros autores, como Ksentini, Taleb, Chen (2014) e Mendes (2014), utilizam o MDP para a tomada de decisão no gerenciamento da nuvem. Nessa abordagem, é definido um estado da nuvem e o gerenciador do ambiente executa ações durante o tempo que podem levar para um ou mais estados. A grande diferente está no fato de que o planejamento é feito para várias ações a serem tomadas ao longo do tempo.

Por se tratar de um sistema discreto, o sistema possui tempos nos quais se realizam decisões. As decisões podem ser tomadas periodicamente ou quando é necessária uma ação por parte do gerenciador. Sendo importante ressaltar que não fazer nada também é considerado uma ação. As ações mais comuns no gerenciamento da nuvem, conforme Mendes et al. (2015), estão relacionadas com:

- Balanceamento de carga;
- Alocação de recursos;
- Dimensionamento;
- Clone e migração;
- Configurações específicas.

2.3.3.1 Estado da Nuvem

O estado da nuvem é uma representação do ambiente de computação em nuvem em um determinado tempo *t*. No MDP e na teoria do controle um estado é representado por valores de variáveis monitoradas *s*.

A representação do MDP puro não é suficiente para representar o ambiente de computação em nuvem, visto que novos elementos são inseridos durante e removidos no decorrer do tempo, resultando em diferentes conjuntos de possíveis estados *S*. É necessária então uma abordagem não estacionária, resultando em conjuntos de estados variáveis no tempo S_t.

2.3.3.2 Critérios para a tomada de decisão

O objetivo de um gerenciador da nuvem é otimizar as variáveis monitoradas diante de determinados critérios, ou seja, buscar levar o sistema ao estado ideal. Para tanto, o gerenciador precisa escolher as melhores ações e executá-las. A escolha é uma decisão que leva em consideração um ou mais critérios, mesmo que esse critério seja o aleatório.

No gerenciamento de nuvens, são várias as otimizações que podem ser almejadas e, consequentemente, são vários os critérios para a tomada de decisão. Frequentemente, encontram-se trabalhos que buscam a redução do consumo de energia, aumento do lucro, tempo de resposta, entre outros. Em Mendes et al. (2015) tem-se uma revisão que categoriza estes critérios em seis grupos, como seguem:

- Desempenho: este critério está relacionado a critérios de SLA, tempo de resposta, QoS e outros;
- Contabilidade: custo, preço e lucro são conceitos inseparáveis de serviços e do gerenciamento da nuvem;
- Falhas: vários trabalhos atuam na tolerância a falhas no gerenciamento da nuvem;
- Segurança: critério central e muitas vezes incorporado em sistemas de suporte;
- Emissão de CO2: o ponto central das nuvens verdes é o consumo de energia. Este pode ser dividido em custo, compreendido no critério de contabilidade e emissão de CO^2;
- Critérios especiais: outros critérios que não se enquadram nas categorias citadas podem ser considerados no gerenciamento de nuvem.

2.4 TEORIA DE CONTROLE

A teoria de controle utiliza a engenharia e a matemática para lidar com o comportamento de sistemas dinâmicos. O objetivo de um sistema de controle é fazer com que a saída Y se comporte de uma maneira desejada, controlando as entradas U da planta (sistema controlado) (SKOGESTAD; POSTLETHWAITE, 2007).

A teoria do controle frequentemente utiliza funções de transferência para representação, em termos esparsos ou com frequência temporal, da relação entre a entrada e a saída de um sistema linear. Por outro lado, para modelar sistemas mais complexos, como um sistema

multiobjetivo, a teoria de controle moderna frequentemente utiliza uma abordagem de estados ao invés de transformação. Sendo o estado de um sistema um conjunto finito de valores das variáveis deste.

3 TRABALHOS CORRELATOS E CARACTERÍSTICAS DESEJADAS

O autogerenciamento do ambiente de computação em nuvem passa pela computação autonômica. Neste campo de pesquisa, um dos mais importantes desafios da computação autonômica é definir as abstrações e modelos apropriados para o entendimento, controle e desenho do comportamento emergente. Isto porque sistemas autonômicos devem lidar com condições variáveis que só podem ser conhecidas em tempo de execução (KEPHART; CHESS, 2003).

Dito isto, nesta seção serão elencados trabalhos correlatos e estes são classificados dentro dos quatro graus de autonomicidade e tipo de tomada de decisão na etapa de planejamento. Além disso, características dos trabalhos correlatos que são relevantes para o modelo proposto também serão ressaltadas aqui.

Em Sharma (2013) um sistema para automatizar o processo de configuração e reconfiguração de nuvens híbridas, nomeado de SEAGULL, é desenhado e implementado. Seu sistema envolve elasticidade de uma nuvem privada focando na alocação de nós em nuvem pública, esta alocação se dá baseada em SLAs, SLOs, monitoramento e previsões.

Dado o foco em uma atividade específica que é a alocação de nós, o sistema em questão pode ser classificado como *Support* na classificação de Huebscher e Maccann (2008). Trabalhando com previsões, Sharma reconhece que estimar a capacidade de um sistema distribuído é um *hard challenge*, e que esse desafio é intensificado pelo fato de que componentes de softwares se comportam diferente em cada configuração de hardware. Assumindo que não se pode prever com exatidão como um software vai se comportar em um hardware específico, é mandatório que o sistema envolva incerteza. Desta forma, em Sharma (2013) se faz uso de estatística e probabilidade.

Em Beloglazov, Abawajy e Buyya (2012) um mecanismo autonômico consciente de energia é desenvolvido para autogerenciamento de mudanças de estados dos recursos para satisfazer SLAs e SLOs e se obter eficiência energética. Diferente de Sharma (2013), este trabalho tem um foco energético e introduz um modelo mais completo, envolvendo não somente servidores e máquinas virtuais,

mas estendendo-o com consumidores e uma entidade chamada *service allocator*, esta última faz a interface entre a infraestrutura da nuvem e os consumidores.

Apesar de não ter um modelo holístico, o trabalho de Beloglazov, Abawajy e Buyya (2012) reconhece essa necessidade baseado no fato de que as técnicas de otimização são muitas vezes contraditórias.

Palmieri et al. (2013) apresenta um *framework* que utiliza teoria dos jogos para agendar tarefas em máquinas em um ambiente de nuvem federada. O modelo proposto é de multiusuários que competem entre si em um modelo de jogo buscando o melhor resultado individual. O objetivo é que as interações entre os agentes resultem em um equilíbrio de Nash e os experimentos buscam avaliar o número de interações necessárias para que o equilíbrio seja obtido.

Gasior e Seredynsky (2014) propõem um algoritmo genético para fazer o agendamento e distribuição de cargas de trabalho com os objetivos de minimizar as chances de falha e o tempo para se completar as tarefas. Para se escolher a melhor estratégia a ser adotada, utiliza-se de um modelo da teoria dos jogos. A decisão é feita de forma distribuída em vários agentes autonômicos que competem entre si e pode-se buscar um equilíbrio de Nash.

Fitó et al. (2012) apresentam uma granularidade diferente, propondo um modelo autonômico gerido por objetivos em nível de negócio (*Business-Level Objectives* - BLOs). Neste o objetivo é obter um alinhamento entre negócio e TI, estendendo o modelo de *business-driven IT* (SAUVÉ *et al.*, 2006).

Em Kim (2011), os seguintes objetivos são utilizados para a alocação e realocação e são representados como casos de uso:

- Aceleração: caso de uso que explora como a nuvem pode ser utilizada para reduzir o tempo de execução de uma aplicação através, por exemplo, do uso de recursos para explorar paralelismo adicional;
- Conservação: caso de uso que investiga como as nuvem podem ser utilizadas para conservar alocações, com o tempo de execução apropriado e limitações de orçamento;
- Resiliência: investiga como as nuvens podem ser utilizadas para lidar melhor com situações inesperadas.

Sharma (2013) apresenta duas abordagens para o provisionamento dinâmico: *cloud provider centric* e *customer-centric*. A

primeira tenta maximizar a receita, enquanto a segunda tenta minimizar os custos para o consumidor. Tanto os objetivos apresentados por Sharma (2013) quanto os apresentados por Kim et al. (2011) são relativos. Este relativismo se refere ao escopo, tempo e perspectiva do usuário ou parte interessada. Outros objetivos de controle são relativos e diferentes em vários trabalhos, como Wei (2010), Corradi, Fanelli e Foshini (2014) e Loreti e Ciampolini (2014).

3.1 NECESSIDADES PARA O GERENCIAMENTO AUTONÔMICO DE NUVEM

Para se relacionar as características necessárias para o gerenciamento autonômico de nuvem apresenta-se a Tabela 1 - Comparativo dos trabalhos correlatos. A segunda coluna da referida tabela demonstra o grau de autonomicidade do trabalho relacionado, enquanto na terceira coluna tem o tipo de tomada de decisão. A última coluna apresenta qual característica mais relevante do referido trabalho que deve ser adotada no modelo a ser proposto.

Tabela 1 - Comparativo dos trabalhos correlatos.

	Grau de autonomicidade	**Tipo de tomada de decisão**	**Característica**
(SHARMA, 2013)	*Support*	*Goal-based*	Incerteza
(BELOGLAZOV; ABAWAJY; BUYYA, 2012)	*Autonomous*	*Rule-Based*	Modelo holístico
(PALMIERI *et al.*, 2013)	*Autonomous*	*Utility-based*	Multi-jogadores e tempo discreto.
(GASIOR; SEREDYNSKI, 2014)	*Autonomous*	*Goal-based*	Decisão com teoria dos jogos
(FITÓ *et al.*, 2012)	*Autonomic*	*Goal-based*	Objetivos alto nível
(KIM *et al.*, 2011)	*Autonomous*	*Goal-based*	Multi-objetivos
(WEI *et al.*, 2010)	*Support*	*Goal-based*	Objetivos relativos

Analisando os trabalhos apresentados quanto a tomada de decisão, esta é feita em várias formas diferentes em cada um dos

trabalhos. Dada a necessidade de uma abordagem holística e multiobjetiva, a tomada de decisão, utilizando uma função de utilidade, similar a de Palmieri, é a mais adequada.

O uso de função de utilidade também vai ao encontro da necessidade do modelo incorporar objetivos que sejam relativos as condições do momento e que possam considerar objetivos de alto nível. O resultado de uma função de utilidade varia conforme o tempo, as condições do ambiente e objetivos, além de poder considerar predições e especulações sobre tempos futuros.

Quanto ao grau de autonomicidade, verifica-se que Beloglazov, Abawajy e Buyya (2012), Palmieri et al. (2013), Gasior e Seredynski (2014) e Kim (2011) encontram-se no grau de autonomicidade, *Autonomous*. A principal melhoria para se atingir o grau mais alto é considerar objetivos de alto nível, assim como Fitó (2012) que considera objetivos de negócio e assim pode ser classificado como *Autonomic*. Acredita-se que para se ter um gerenciamento holístico como proposto, soluções no nível *Autonomous* já são suficientes, contudo, uma melhor orquestração será obtida no nível *Autonomic*.

Outra característica relevante para o modelo é a incerteza. Certamente a computação é uma ciência exata, contudo, considerar todos os fatores envolvidos na predição e na tomada de decisão não é algo computacionalmente e economicamente viável. Sendo assim, um sistema determinístico estaria frequentemente sujeito a erros. Desta forma é necessário considerar a incerteza e adotar um modelo não determinístico, reconhecendo que falhas e imprevistos podem acontecer.

4 MODELO PROPOSTO

O modelo a seguir foi desenvolvido com base nas necessidades apresentadas no Capítulo 3. Este foi apresentado em Flores et al. (2015) e tem como objetivo ser um modelo holístico para subsidiar a tomada de decisão de um gerenciador autonômico de nuvens computacionais.

Sendo assim, o gerenciador que utilizará o modelo não faz parte do escopo deste trabalho. Contudo, para que este possa ser validado, um gerenciador simplificado foi construído e será apresentado juntamente com o processo para validação, no Capítulo 5.

Em um primeiro momento os elementos que compõem o modelo são descritos por uma modelagem matemática, abordagem adotada em diversas áreas, assim como na teoria do controle. Posteriormente, em 4.2, será apresentada a dinâmica de funcionamento do modelo, apresentando o seu comportamento ao longo das fases do MAPE-K.

4.1 ESPECIFICAÇÃO

O modelo a ser apresentado será base para o processo de tomada de decisão em computação em nuvem e será utilizado nas fases do MAPE-K. Para se obter as características apontadas no Capítulo 3, propõe-se uma abordagem baseada em Lozovanu e Pickl (2009), criando-se um jogo com multijogadores, multiobjetivo e uma abordagem estocástica resultando em um SMG.

Os atores citados por Lozovanu e Pickl, são os jogadores do SMG e serão descritos neste modelo como Partes Interessadas. Na nuvem, as partes interessadas podem ser consumidores, o gerenciador do ambiente ou outros. A dinâmica proposta em forma de jogo entende que cada parte interessada faz as suas ações e o gerenciador, como uma parte interessada, escolhe que ações tomar com base neste modelo proposto. Com o uso do modelo o gerenciador entenderá de que forma a sua decisão irá afetar as outras partes interessadas.

A dinâmica proposta resulta em um jogo *non-zero sum*, em que a soma dos resultados dos jogadores se altera conforme a combinação de ações tomadas. Dependendo da estratégia escolhida para o gerenciador, o jogo poderá ser colaborativo, competitivo ou de Stackelberg. Nestas classes de problemas, os princípios de otimização de Nash, Pareto e Stackelberg são frequentemente utilizados.

Os principais elementos do modelo são: Estado, Partes Interessadas, Interesses, Ações, Eventos, Função de Custo e Elementos Hipotéticos. Estes serão descritos a seguir.

4.1.1 Estado da Nuvem, Dimensões e Índice Dimensional

Para atender as necessidades de gerenciamento da computação em nuvem, sugere-se uma abordagem derivada do MDP e da teoria do controle. As variáveis monitoradas, vindas da teoria do controle, neste modelo são apresentadas como dimensões d, sendo que os seus valores são representados por x_{dt}. Optou-se por alteração no nome pois este modelo traz uma abordagem de gerenciamento baseada em um SMG e não própria da teoria de controle.

Definimos também que D é o conjunto que contém todas as dimensões e, para que o modelo seja não estacionário, D_t é o um subconjunto de D que contém as dimensões no tempo t.

$$\begin{array}{l} Se\ D_t \supset d \\ d\ é\ uma\ variável\ monitorada,\ e \\ d \exists\ no\ tempo\ t \end{array} \qquad (1)$$
$$D \supset D_t$$

As dimensões podem possuir relações hierárquicas entre si. A hierarquia é definida com dimensão pai e dimensões filhas. Desta forma, define-se a função

$$dp: D_t \rightarrow D_t \qquad (2)$$

que, dada uma dimensão d qualquer, retorna a sua dimensão pai, caso exista. Também define-se a função

$$df: D_t \rightarrow PD_t \qquad (3)$$

a qual também recebe uma dimensão qualquer, porém retorna um elemento de PD_t[3] que representa as dimensões filhas, sendo que PD_t é o conjunto de todos os possíveis subconjuntos de D_t.

O índice dimensional X_{dt} representa todos os possíveis valores que x_{dt} pode assumir no tempo t, onde $X_{dt} \in \mathcal{X}_{Dt}$. A relação entre D_t e $\mathcal{X}_{Dt}$ é uma função bijetiva ($dx: D_t \rightarrow \mathcal{X}_{Dt}$). Sendo assim, $\mathcal{X}_{Dt}$ pode ser representado como um conjunto de conjuntos.

[3] Os conjuntos resultantes dos elementos de todos os possíveis subconjuntos de um conjunto são representados com a letra P maiúscula seguido pela representação do conjunto originário.

Definimos então s_t que é o estado na nuvem no tempo *t*, sendo que s_t é um conjunto com todos os valores das dimensões x_{dt} existentes no tempo *t*.

$$s_t = \{x_{dt} \mid x_{dt} \in X_{dt} \forall X_{dt} \exists \mathcal{X}_{Dt}\} \quad (4)$$

A título de representação matemática, podemos definir que o conjunto de todos os possíveis estados S_t como o produto cartesiano de cada conjunto X_{dt} em $\mathcal{X}_{Dt}$. A consequência é que cada elemento s_t em S_t é uma tupla $(x_1, x_2, \dots, x_n)$, onde x_1 é um elemento de X_{1t}, x_2 é um elemento de X_{2t} e assim sucessivamente. Sendo assim, S_t pode ser representado como:

$$S_t = \prod_{d_t \in D_t}^{d_t} X_{d_t} \in \mathcal{X}_{D_t}. \quad (5)$$

Neste momento fazemos uma ressalva não observada em Flores (2015). O produto cartesiano de todos os índices dimensionais pode resultar em estados que não fazem sentido no mundo real. Contudo, isso não afeta o uso do modelo, pois, conforme será explicado adiante, na dinâmica proposta a tomada de decisão se baseia no estado atual e nos possíveis estados futuros e estes são obtidos pela função pe_t que será definida adiante.

4.1.1.1 Exemplo

Para facilitar a visualização dos elementos definidos, dá-se este exemplo. Em uma nuvem hipotética que se possua os seguintes elementos:

- Datacenter 1
- Máquina física 1
- Máquina virtual 1
- Roteador 1

Supondo que os elementos monitorados e as dimensões correspondentes se deem conforme a Tabela 2.

Tabela 2 - Elementos monitorados e dimensões.

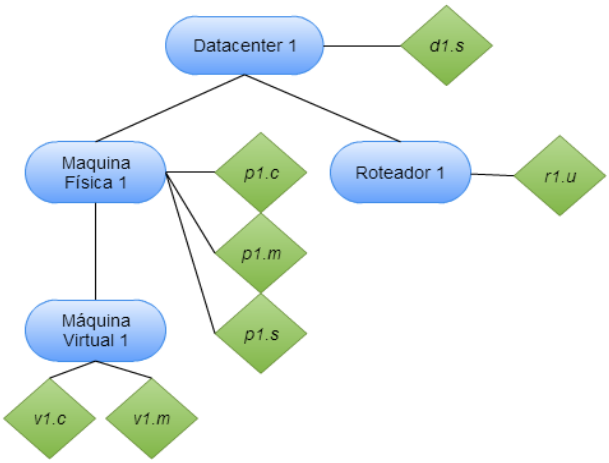
Componente monitorado	Dimensão
Estado geral do Datacenter 1	$d1.s$
CPU da máquina física 1	$p1.c$
Memória da máquina física 1	$p1.m$
Estado da máquina física 1	$p1.s$
CPU da máquina virtual 1	$v1.c$
Memória da máquina virtual 1	$v1.m$
Uso do roteador 1	$r1.u$

Para facilitar a organização das dimensões, pode se criar dimensões que representem os elementos reais. Acrescentamos então as dimensões *d1, p1, v1* e *r1* que representam respectivamente o datacenter, a máquina física, a máquina virtual e o roteador. Neste caso então D_t possuiria os seguintes valores

$$D_t = \{'d1','d1.s','p1','p1.c','p1.m','p1.s','v1','v1.c', 'v1.m','r1','r1.u'\}. \quad (6)$$

Neste exemplo, pode-se aplicar a hierarquia das dimensões conforme ilustra a Figura 4. Nesta as dimensões pais são representadas pelos quadrados azuis e os elementos monitorados são representados pelos losangos verdes.

Figura 4 - Hierarquia das dimensões



Para diminuir a possibilidade de estados, os valores assumidos podem ser aproximações. Sendo assim, temos os índices dimensionais conforme a Tabela 3.

Tabela 3 – Índices dimensionais.

Dimensão	Elemento	Valor
$p1.c$	X_{1t}	$\{'0\%','10\%','50\%','80\%','100\%'\}$
$p1.m$	X_{2t}	$\{'0\%','10\%','50\%','80\%','100\%'\}$
$p1.s$	X_{3t}	$\{'on','off','reboting','slepping'\}$
$v1.c$	X_{4t}	$\{'0\%','10\%','50\%','80\%','100\%'\}$
$v1.m$	X_{5t}	$\{'0\%','20\%',' 40\%',' 60\%',' 80\%',' 99\%'\}$
$r1.u$	X_{6t}	$\{'low',' high'\}$

Como resultado, temos

$$X_{Dt} = \begin{Bmatrix} '\{'0\%','10\%','50\%','80\%','100\%'\}', \\ '\{'0\%','20\%',' 40\%',' 60\%',' 80\%',' 99\%'\}', \\ '\{'on','off','reboting','slepping'\}', \\ '\{'0\%','10\%','50\%','80\%','100\%'\}', \\ '\{'0\%','20\%',' 40\%',' 60\%',' 80\%',' 99\%'\}', \\ '\{'low',' high'\}' \end{Bmatrix}. \quad (7)$$

Já o valor do estado da nuvem s_t em um momento de inatividade poderia ser representado como

$$s_t = \{'0\%,' 0\%,' 0\%','on',' 0\%',' low'\}. \quad (8)$$

4.1.2 Partes interessadas e Interesses

Segundo Beloglazov et al. (2012), conforme explicado no Capítulo 3, os objetivos de otimização no gerenciamento da nuvem muitas vezes são contraditórios. Para contornar essa situação, esse modelo adota a visão de Lozovanu e Pickl (2009), apresentada na seção 2.3.3.

Os atores propostos por Lozovanu e Pickl são apresentados neste modelo como partes interessadas u. Sendo assim, U é o conjunto de todas as partes interessadas existentes. Os membros do conjunto U podem variar conforme a abordagem adotada para implementação do modelo. Alguns possíveis membros de U são:

- Proprietário da nuvem: quem possui a nuvem e a disponibiliza a outras partes interessadas, normalmente

visando lucro.

- Consumidores: aqueles que utilizam os serviços da nuvem e normalmente possuem exigências de SLA a serem cumpridas.
- Usuários Finais: se a nuvem gerenciada for um IaaS ou um PaaS, consumidores da nuvem terão seus próprios usuários, os quais podem ser considerados no gerenciamento.

Ressalta-se que o modelo, como definido, não suporta a variação das partes interessadas no decorrer do tempo. Esse entendimento, além de viabilizar a implementação do modelo, entende que ao se inserirem outras partes interessadas, as premissas, estados, dimensões e funções anteriores não são mais válidas. Neste caso, quando uma nova parte interessada é inserida, o jogo tem um novo começo.

Cada parte interessada tem seus próprios interesses. Enquanto o proprietário da nuvem pode buscar maximizar o lucro, alguns consumidores podem prezar pela robustez e outros pelo desempenho. Para conciliar esses interesses distintos, problemas econômicos e jogos costumam ser modelados utilizando uma função de utilidade a qual representa a utilidade de uma ação para um jogador em um tempo específico. Extrapolando este conceito, é proposta uma função de interesse:

$$v_t: U \times S_t \rightarrow \gamma \tag{9}$$

que retorna o interesse γ de uma parte interessada em um particular estado no tempo t.

Como resultado, v_t retorna γ onde γ é um número real entre -1 e $+1$ ($\gamma \in \mathbb{R} \mid -1 \leq \gamma \leq 1$). Sendo assim, zero representa um interesse neutro, números positivos representam o grau de interesse e números negativos representam um grau de desinteresse.

Cada parte interessada terá um interesse diferente em um determinado estado e este interesse poderá mudar em tempos diferentes. Como exemplo, o objetivo de aceleração apresentado por Kim et al. (2011) poderia ser traduzido para o modelo proposto como um interesse de um consumidor em um estado em que ele teria mais MVs alocadas.

Algumas dimensões não podem ser alteradas por nenhuma parte interessada, pois estas são apenas para monitoramento e auxílio para a tomada de decisão. Contudo, outras são diretamente controladas por determinada parte interessada. Por esse motivo, se faz necessária a função

$$du_t: U \rightarrow D_t \quad (10)$$

a qual corelaciona todas as dimensões D_t que uma parte interessada pode alterar.

4.1.3 Ações e Eventos

A teoria do controle frequentemente utiliza a escolha de uma configuração para levar o sistema a um melhor estado. Já MDPs e SMGs frequentemente entendem que uma ação leva a um novo estado. Em Mendes (2014) tem-se uma boa aderência as necessidades de gerenciamento utilizando MDP e ações. Neste sentido, este modelo utiliza o conceito de ações o qual permite a interconexão entre partes interessadas, interesses e estado da nuvem.

As partes interessadas podem afetar e alterar o estado da nuvem de forma direta ou indireta. A forma indireta acontece quando o gerenciador utiliza como critério para a sua tomada de decisão o retorno da função de interesse v_t de outra parte interessada. Já a alteração direta acontece quando a parte interessada executa uma ação e esta ação altera o estado da nuvem.

Sendo A_t o conjunto de todas as ações possíveis no tempo *t*, cada parte interessada possui um subconjunto de ações $\tilde{A}_t \mid \tilde{a}_t \subset s_t R a_t$, que são as ações que uma determinada parte interessada pode tomar em um determinado estado. Para se obter $\tilde{A}_t$, tem-se a função

$$pa: U \times S_t \rightarrow \tilde{A}_t. \quad (11)$$

Ao refinar os modelos propostos por Mendes (2014) e por Palmieri (2013), conclui-se que o estado da nuvem pode ser alterado de uma maneira não prevista por conta de eventos E_t. Eventos são similares a ações e também podem alterar o estado da nuvem. A principal diferença entre eles é que eventos não são planejados e nem mesmo desejados pelas partes interessadas. Um evento pode ser um problema de hardware, uma falha em um software ou até mesmo uma falta de energia, por exemplo.

Os conjuntos das possíveis ações A_t e dos possíveis eventos E_t não são estacionários. Isto porque alguns destes só fazem sentido em estados particulares. Exemplo é a ação de ligar um servidor que só

existe se o servidor estiver desligado. O mesmo ocorre com eventos: uma falha em um software só pode acontecer se este estiver instalado e sendo executado. Sendo assim, definimos R_0 (12) como sendo a função que faz a relação entre estados e ações e R_1(13) a que faz a relação entre estados e eventos.

$$R_0: S_t \rightarrow A_t \quad (12)$$
$$R_1: S_t \rightarrow E_t \quad (13)$$

Também é importante definir a função

$$pe_t: S_t \times \tilde{A}_t \rightarrow PS_{t+n} \quad (14)$$

que recebe uma ação que pode ser executada por uma parte interessada e o estado em que essa ação seria executada. Por se tratar de um sistema não determinístico, o retorno é um elemento do conjunto PS_{t+n}. Sendo PS_{t+n} o conjunto de todos os subconjuntos de estados no tempo $t + n$.

As ações que as partes interessadas podem realizar podem levar o sistema a um novo estado desejado com certa probabilidade. Essa probabilidade é dada pela função

$$p_t: S_t \times \tilde{A}_t \times S_{t+n} \rightarrow \mathbb{R}_{[0,1]} \quad (15)$$

que recebe um estado S_t, normalmente o atual, a ação que será executada $\tilde{A}_t$ e um estado futuro S_{t+n}.

4.1.4 Função de custo

Toda ação tem um custo relacionado. O custo implica na redução de interesse de uma parte interessada em realizar determinada ação. A função custo pode ser definida como

$$c_t: U \times S_t \times \tilde{A}_t \rightarrow \mathbb{R} \quad (16)$$

A função custo pode ser utilizada em dois momentos distintos. O primeiro seria quando a parte interessada se tratar do gerenciador do sistema. Este precisa decidir quais ações tomar com base nos seus interesses e no custo destas ações. O segundo momento seria na predição, em que o gerenciador pode tentar prever que ações uma parte interessada irá tomar com base também nas funções (9), (11), (14) e (15).

Cabe-se ressaltar que as partes interessadas, excetuado o gerenciador, possuem tomada de decisão própria e por isso as ações tomadas por estes somente podem ser prevista pelas funções, mas não dependem destas.

4.1.5 Elementos hipotéticos

Os elementos hipotéticos são derivações imaginárias dos elementos já apresentados. O exemplo apresentado em Flores (2015) demonstra a sua necessidade.

> Um ambiente tem um provedor de nuvem com 10 servidores e 10 usuários. O servidor, em momentos de picos, utiliza todos os recursos disponíveis e todos os SLAs estão sendo satisfeitos com os usuários. Se a demanda aumentar, os SLAs poderão ser comprometidos, porém se diminuir haverá uma subutilização de recursos. A questão é: deve ser aumentada a capacidade do ambiente? Dado isto, o sistema pode inferir que se trata de um equilíbrio de Nash e não alocar mais nenhum recurso. Contudo, um gerente humano iria analisar todo o sistema e fazer predições sobre novos clientes e novas demandas.

Uma das vantagens de um gerenciador humano sobre algoritmos de gerenciamento autonômicos é a capacidade de um humano especular sobre o ambiente futuro. Sendo assim, para se escolher as melhores ações, é necessário se especular sobre o futuro, considerando não apenas os elementos e interesses atuais, mas também os interesses e elementos que poderão ser criados após a tomada das referidas ações.

Portanto, para que o gerenciador possa fazer as referidas predições, este modelo conta com elementos hipotéticos. Os elementos hipotéticos possuem a mesma modelagem dos elementos apresentados, porém não refletem o estado atual do sistema. Sendo assim, um gerenciador pode criar elementos hipotéticos de qualquer um dos tipos apresentados. Uma utilização comum de elementos hipotéticos é a suposição de estados futuros, que pode ser utilizado pelos gerenciadores para decidir quais rumos que se deseja seguir e qual estado se deseja alcançar.

4.2 DINÂMICA E USO DO MODELO

O modelo especificado na seção 4.1 descreve os elementos necessários para subsidiar a tomada de decisão de um gerenciador de nuvem computacional. Cabe-se ressaltar que o comportamento do gerenciador não é objeto deste trabalho. Contudo, para uma melhor visualização do funcionamento do modelo, nesta seção será demonstrado como um gerenciador poderia utilizar o modelo descrito.

Conforme especificado, a proposta é que a dinâmica da nuvem seja entendida como um jogo com múltiplos jogadores (partes interessadas) em que o gerenciador é um jogador o qual busca um resultado ótimo sob alguma perspectiva. Sendo assim, as jogadas do gerenciador serão pontos de tomada de decisão, ou épocas de decisão no MDP, nos quais o gerenciador precisa escolher qual ou quais ações devem ser tomadas. Os pontos de tomada de decisão podem ocorrer periodicamente, de tempos em tempos, ou após algum evento ocorrer e iniciarão o ciclo de controle MAPE-K. Este possui as fases de monitoramento, análise, planejamento e execução e os comportamentos nestas fases são descritos nos próximos parágrafos.

Na fase de monitoramento é que os dados do sistema são colhidos. O monitoramento pode ser passivo, quando um sensor externo notifica da modificação no sistema, ou ativo, caso em que o próprio agente gerenciador que busca a informação. Os dados obtidos são enviados para a Análise.

O processo de análise é o maior e mais complexo do gerenciamento de nuvens (MENDES *et al.*, 2015). Ele é responsável por receber os dados do monitoramento e preencher os componentes definidos no modelo, como:

- o estado atual (S_t);
- o conjunto de possíveis ações que podem ser executadas;
- as funções como pe_t, c_t, pa_t, p_t e v_t;

Terminada a análise, inicia-se o planejamento. É nesta fase que as decisões são tomadas. Para escolher as melhores decisões, várias estratégias podem ser adotadas, buscando equilíbrio de Nash ou um ótimo a Pareto, por exemplo. A saída desta fase será um plano com as ações que o gerenciador irá tomar.

Para elaborar o plano, é necessário escolher a melhor ação ou as melhores ações a serem tomadas. Neste sentido, o gerenciador utiliza a função pa_t passando como u_t ele próprio e o como s_t o estado atual e com isso obtêm as ações que ele pode tomar. Em seguida, se utiliza a

função pe_t para descobrir quais estados estas ações podem levar.

Tendo as ações que podem ser tomadas e os possíveis estados futuros, o gerenciador então pode tomar uma decisão considerando:

- o interesse das outras partes interessadas nos estados futuros através da função v_t;
- o custo em realizar cada ação com uso da função c_t; e
- a probabilidade das ações resultarem no estado desejado com a função p_t.

Por último então, a fase de execução recebe o plano pronto. Nesta fase é que ele é executado, produzindo efeitos no sistema e alterando o estado do sistema. Feito isso, o ciclo recomeça e as outras partes interessadas podem fazer as suas ações, até que ocorra um novo ponto de tomada de decisão do gerenciador.

5 VALIDAÇÃO

Para se iniciar a validação do modelo proposto para a tomada de decisão, devemos primeiramente retomar a hipótese levantada no início desta dissertação:

> "A aplicação de uma abordagem holística derivada da teoria dos jogos na criação de um modelo que suporte a tomada de decisão de um gerenciador autonômico de nuvens é adequada para se obter um gerenciamento multiobjetivo eficaz"

Verifica-se que o modelo ter uma abordagem ser holística derivada da teoria dos jogos, subsidiar a tomada de decisão do gerenciador autonômico e que com o uso do modelo possa se ter um gerenciamento eficaz. Cada uma destes atributos do modelo serão validados a seguir.

A primeira característica do modelo é ser holístico. Para validar essa característica, elencou-se todos os elementos descritos como ações, operações, elementos físicos e virtuais nos trabalhos correlatos apresentados no Capítulo 3. Com essa lista de elementos, verificou-se se estes poderiam ser representados com o modelo aqui proposto resultando na

Tabela 4.

Tabela 4 - Aderência do modelo.

	Número de elementos encontrados	Número de elementos mapeados[4]
(SHARMA, 2013)	32	32
(BELOGLAZOV; ABAWAJY; BUYYA, 2012)	19	19
(PALMIERI et al., 2013)	9	9
(GASIOR; SEREDYNSKI, 2014)	18	17
(FITÓ et al., 2012)	22	22
(KIM et al., 2011)	25	25
(WEI et al., 2010)	19	19

Percebe-se que quase todos os elementos apresentados nos principais trabalhos correlatos puderam ser mapeados com sucesso com os elementos propostos no modelo. O elemento faltante foi a definição de carga de trabalho proposta por Gasior e Seredynski (2014). Contudo, o efeito de uma carga de trabalho está representado nas dimensões e a carga em si pode ser modelada nos modelos de predições que serão utilizados pelas funções de interesse. Desta forma, esse elemento não invalida a característica holística do modelo.

A segunda característica que o modelo deve ter, conforme a hipótese, é subsidiar a tomada de decisão de um gerenciador autonômico. Para tanto, o modelo deve prover informações suficientes para que um gerenciador possa ter autoconfiguração, auto-otimização, autoproteção e auto-reparo.

Quanto a autoconfiguração, ela está relacionada a habilidade de se configurar de acordo com políticas de alto nível. Percebe-se que o modelo introduz o conceito de interesses e que estes são base para as decisões a serem tomadas pelo gerenciador, atendendo assim a este critério.

A auto-otimização também está presente, pois nos pontos de tomada de decisão, a decisão escolhida pode otimizar a nuvem ou o elemento gerenciador, visto que o gerenciador é modelado como uma parte interessada e seus interesses se refletem nas suas ações.

No caso em que alguma parte interessada tome uma ação danosa

[4] Elementos que fazem parte do modelo de predição ou que estabelecem regras ou objetivos são entendidos como mapeáveis através da função de interesse, as quais apenas são pertencem ao modelo, porém serão melhor detalhadas em trabalhos futuros.

ao sistema ou a outras partes, o sistema poderá se auto-proteger, considerando que aquela ação não é interessante. Já em caso de um ataque ao sistema, pressupõe-se que este resultará em eventos os quais permitirão que o gerenciador avalie o estado atual e as ações que pode tomar, podendo inclusive realizar o auto-reparo.

Além disso, o uso do modelo deve providenciar um gerenciamento multiobjetivo e eficaz. Para tanto, faz-se necessária a avaliação e a validação deste. Para a validação e avaliação vislumbrou-se a possibilidade de implementação do modelo, implantando em um ambiente de testes, assim como a possibilidade de se valer da simulação.

As seguintes características fizeram com que fosse optado pela simulação:

- Experimentos e resultados reprodutíveis;
- Menor complexidade, adequando-se ao prazo disponível;

Possibilidade de experimentar em ambientes diversos com facilidade.

Optado pela simulação, foram verificadas ferramentas e *frameworks* existentes que pudessem ser utilizados. Sakellari e Loukas (2013) apresentam em seu trabalho uma revisão das principais ferramentas para simulação de nuvens. Na Tabela 5 é exibida uma comparação das ferramentas avaliadas por eles. A segunda coluna se refere às soluções que suportaram avaliação da eficiência energética, enquanto a coluna QoS se refere àquelas que permitem comparações quanto a critérios de desempenho.

Tabela 5 - Comparativo de Simuladores (SAKELLARI; LOUKAS, 2013)

Software	Eficiência Energética	QoS	Linguagem de programação	Disponível na internet	Código aberto
CloudSim	X	X	Java	X	X
Duy et al.	X		Java		
CDOSim		X	Java		
TeachCloud			Java	X	X
NetworkCloudSim		X	Java		
Greencloud	X		C++		
MDCSim	X	X	Java		
iCanCloud		X	Java	X	X
SimGrid	X	X	C/Java/Lua/Ruby	X	X

O CloudSim e o SimGrid se destacam das demais ferramentas comparadas. Verificou-se que o CloudSim se mostrou mais popular na comunidade, tendo 46 publicações no ano de 2015, contra 10 do SimGrid. A qualidade de trabalhos que já utilizam o CloudSim para a

validação de simulações também pesou na decisão, visto que vários trabalhos relevantes a utilizam.

Dado o destaque do CloudSim e a aderência deste às necessidades para validação do modelo, este foi escolhido. Também foi utilizada nas simulações a ferramenta CloudReports (SÁ; SOARES; GOMES, 2011), que estende o CloudSim com facilitadores e relatórios.

5.1 CLOUDSIM

O *framework* CloudSim, desenvolvido pelo laboratório The CLOUDS Lab da universidade de Melbourne, busca facilitar e tornar mais rápida a criação de experimentos em nuvens computacionais. É uma ferramenta de código aberto e extensível que suporta a modelagem e simulação de nuvens computacionais de grande escala em uma única estação de trabalho (CALHEIROS *et al.*, 2011).

5.1.1 Arquitetura

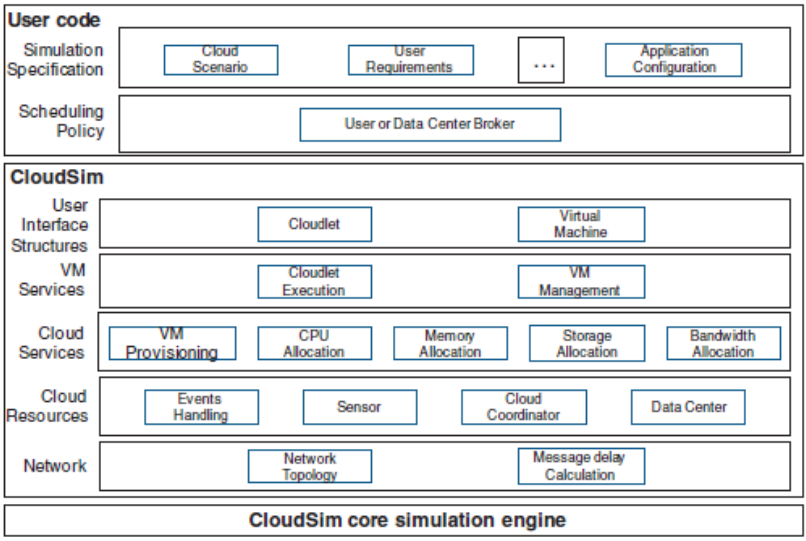
Conforme explicado, o CloudSim é um *Framework* escrito em Java. Para criar simulações, é necessário descrever todo o ambiente e eventos através da programação. Apesar de ser mais complexo do que uma simples parametrização, a abordagem adotada pelo CloudSim propicia flexibilidade e facilita a criação de extensões.

A sua arquitetura é composta por três camadas, conforme a Figura 5 - Arquitetura CloudSim (Calheiros et al, 2011). A primeira camada é a do motor de simulação, a qual trata de executar a simulação, enfileirar e processar os eventos, iniciar e encerrar a simulação. As camadas acima são:

- Camada CloudSim: esta camada provê suporte à modelagem e à simulação dos componentes de uma nuvem. Nesta camada estão definidos os elementos e interfaces da nuvem.

Camada de códigos usuário: essa camada frequentemente é utilizada para definir a própria simulação. Isto envolve definir os consumidores, infraestrutura da nuvem, carga de trabalho e outros.

Figura 5 - Arquitetura CloudSim (Calheiros et al, 2011)



Para dar prosseguimento ao trabalho se faz necessário definir alguns conceitos e entidades do CloudSim:

- VMs: São as máquinas virtuais que são solicitadas pelos consumidores.
- Hosts: São os servidores em que as máquinas virtuais são instanciadas.
- Datacenter: É a central de processamento de dados que que provê os recursos. Ele contém os Hosts, redes, armazenamento e outros elementos de simulação.
- Cloudlet: Representa uma carga de trabalho a ser processada por uma máquina virtual. Entre suas características estão: o tamanho inicial do arquivo, tamanho do arquivo após o processamento, quantidade de instruções necessárias, consumo de rede e quantidade de processadores.
- Datacenter Broker: Representa um agente que opera em nome do usuário, negociando recursos e distribuindo Cloudlets para as MVs.
- Vm Allocation Policy: Política de alocação de máquinas virtuais nas máquinas físicas. O *Framework* possui a classe abstrata *VmAllocationPolicy* a qual deve ser implementada definindo as regras para alocação.

- Custos: Os custos são descritos em uma unidade monetária genérica. É possível especificar os custos pelo uso dos elementos do Datacenter, como: processamento, memória, rede e armazenamento. Os custos são pela fração de recurso reservada, a memória é cobrada por megabyte, enquanto o processamento é cobrado por milhões de instruções.

5.1.2 Extensões CloudSim

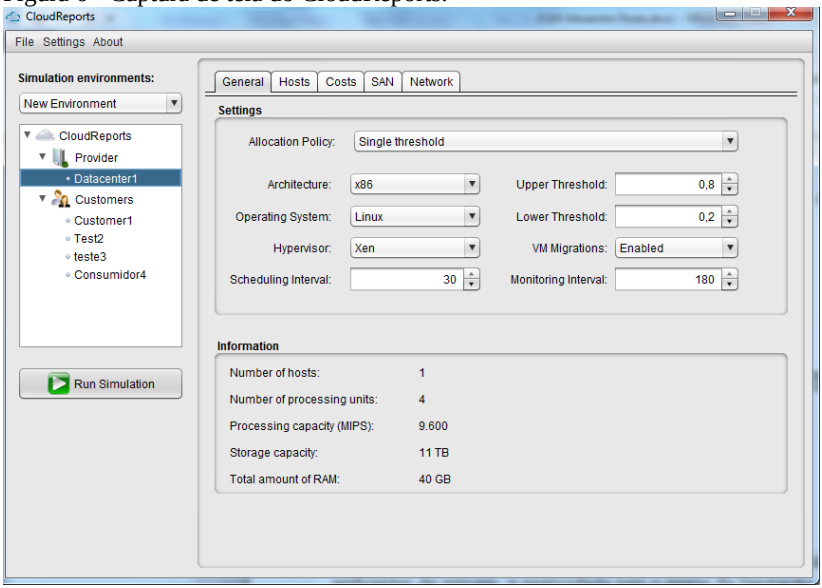
Tipicamente, o uso do CloudSim envolve a criação de códigos na camada de Códigos do usuário. Para realizar a validação do modelo, essa camada foi amplamente utilizada para definir os cenários da simulação. Contudo, para realizar a validação do modelo, também foram necessárias modificações na camada CloudSim.

A primeira modificação na camada CloudSim foi a criação de uma política de alocação. Foram criadas três políticas implementando a classe *VmAllocationPolicy*, duas para o experimento 1 (uma para o gerenciador e outra para o Round Robin) e outra para o experimento 2. Além disso, para o segundo experimento, foi necessário alterar classes do CloudSim para inserir chamadas para que o ciclo de controle fosse iniciado e o gerenciador avaliasse se deveria atuar ou não no sistema.

5.2 CLOUDREPORTS

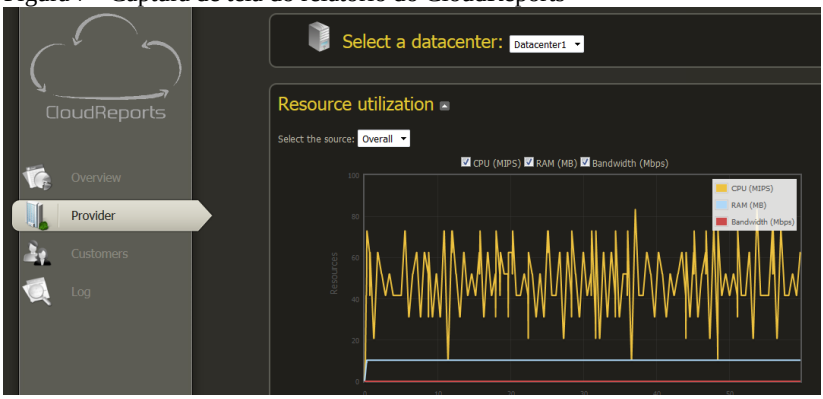
O CloudReports é uma ferramenta visual que simula ambientes de computação distribuída baseados no paradigma de computação em nuvem (SÁ; SOARES; GOMES, 2011). Sem qualquer extensão, o CloudReports permite a simulação de IaaS com *data centers*, servidores, MVs e cargas de trabalho personalizáveis. Como se pode observar na Figura 6, a interface de configuração é intuitiva e simples de utilizar.

Figura 6 - Captura de tela do CloudReports.



Para executar as simulações, o CloudReports utiliza o CloudSim. Após executar as simulações, o CloudReports gera um site HTML, contendo gráficos e resultados da simulação. Os gráficos são agrupados por provedor da nuvem e também por consumidores, mostrando dados como consumo de energia, custos e faturamento, tempo para execução de trabalhos. A Figura 7 é uma captura de tela do site gerado após uma simulação.

Figura 7 - Captura de tela do relatório do CloudReports



5.2.1 Extensões ao CloudReports

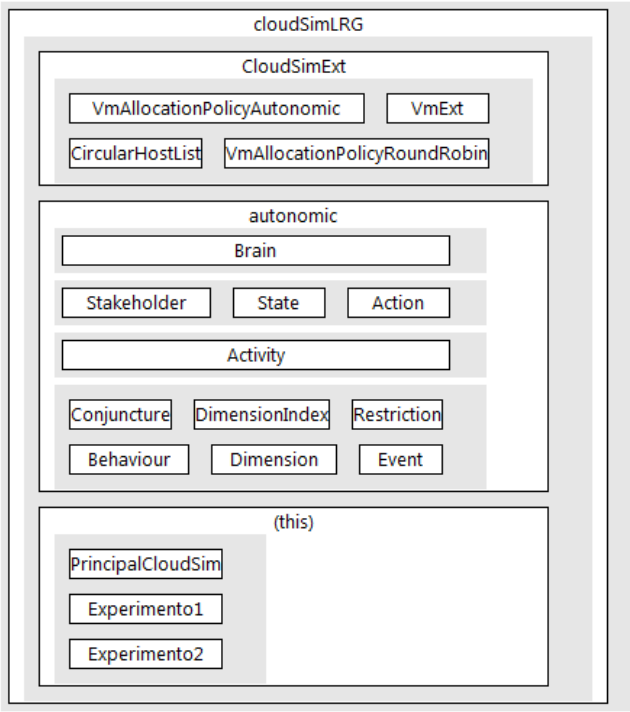
Apesar de ser uma ferramenta muito prática para a simulação de ambientes de nuvem, a praticidade tem o preço da limitação. Da maneira como está projetado, não é possível fazer simulações com os eventos em um tempo pré-definido ou até mesmo ordenar eventos. A única exceção cabe que as máquinas virtuais podem ter prioridades diferentes, porém não há ainda assim uma garantia da ordem e nem o agendamento.

Desta forma, o core do CloudReports foi modificado para suportar o agendamento de eventos. Além disso, como foram necessárias modificações no CloudSim para que o modelo proposto fosse suportado, também teve-se que alterar o CloudReports para utilizar a versão modificada.

5.3 DETALHES DO SIMULADOR

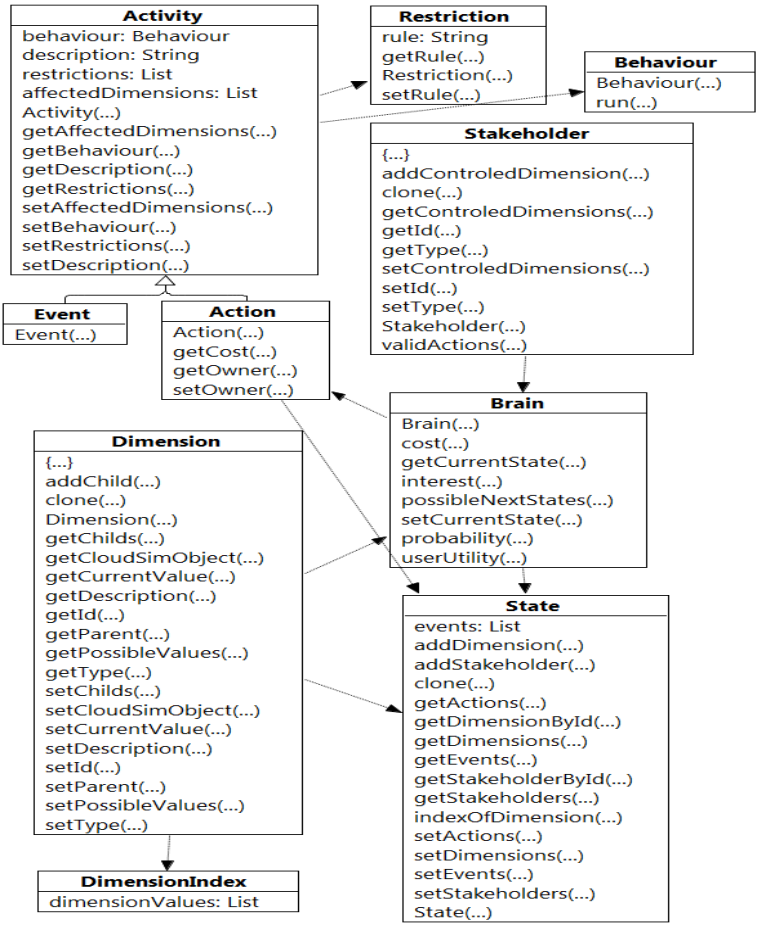
O modelo foi implementado em JAVA, resultando em várias classes para os elementos. A Figura 8 apresenta um diagrama de pacotes com as classes que foram criadas para o simulador. O pacote CloudSimLRG contém as classes relacionadas ao gerenciador e aos experimentos (PrincipalCloudSim, Experimento1 e Experimento2). Já o pacote ClouSimExt tem as classes novas que foram criadas para estender o CloudSim. No CloudReports, as extensões não precisaram de novas classes e por isso não estão no diagrama. Já no pacote autonomic estão as classes do modelo propriamente dito.

Figura 8 - Diagrama de Pacotes



O diagrama de classes mostrado na Figura 9 apresenta as classes que estão no pacote autonomic, ou seja, aquelas que são resultantes da implementação do modelo. No diagrama de classes, visando dar mais clareza e um melhor entendimento, somente os principais métodos, atributos e relações são exibidos. As classes apresentadas no diagrama são descritas nas próximas seções na ordem do Diagrama (da esquerda pra direita, de cima para baixo).

Figura 9 - Diagrama de Classes



5.3.1 Activity

A classe Activity é a classe que define os métodos que devem ser implementados pelas classes Action e Event, que representam as ações e dos eventos. Tanto ações como eventos afetam o estado da nuvem através de modificações nas dimensões. Desta forma, na classe Activity está definido o atributo affectedDimensions que é uma lista de dimensões afetadas pela ação ou evento.

5.3.2 Restriction

Conforme apresentado no modelo, nem todas as ações e eventos são possíveis em todos os estados. Essa classe define as restrições que irão possibilitar os resultados das funções R_0 e R_1.

5.3.3 Behaviour

Cada ação ou evento pode alterar dimensões da nuvem, afetando o estado. Essa classe encapsula este comportamento e é utilizada nas previsões de estados futuros.

5.3.4 Stakeholder

A classe Stakeholder representa as partes interessadas do modelo definido. Cada Stakeholder possui as dimensões que ele controla e, conforme o modelo matemático, essas dimensões são obtidas pela função du_t. Desta forma, essa classe implementa o método getControledDimensions, que retorna as dimensões controladas pelo Stakeholder.

Para identificar cada parte interessada, a classe Stakeholder possui o atributo ID. Verifica-se que existem tipos de partes interessadas e esse tipo pode determinar o retorno da função de interesse. Sendo assim, também foi adicionado o atributo Type.

Nem todas as ações são válidas em um determinado estado, somente um subconjunto delas. Neste sentido, no modelo está definida a função pa_t, a qual recebe uma parte interessada, um estado e retorna esse subconjunto de ações. O método validActions implementa esse comportamento.

5.3.5 Event

Os eventos E_t são representados na classe Event. Assim como a classe Action, a classe Event também é uma implementação da classe Activity. Os eventos alteram o estado da nuvem assim como as ações. Contudo, a principal diferença entre eventos e ações é que as ações possuem uma parte interessada que pode executá-la, enquanto um evento não.

5.3.6 Action

Action é uma implementação da classe Activity. Ela é responsável por representar as ações. As ações são realizadas por partes interessadas e podem modificar o estado do sistema. Neste sentido, cada ação possui uma parte interessada, a qual pode ser obtida pelo método getOwner.

5.3.7 Brain

Talvez a classe mais importante do sistema seja a classe Brain. Nesta classe é que estão algumas das principais funções definidas matematicamente no modelo:

- Função v_t: implementada com o método *interest*, a qual recebe um objeto da classe Stakeholder e outro da classe State está definida nessa classe;
- Função pe_t: que retorna os possíveis estados futuros que uma ação pode levar. É representada através do método *possibleNextStates* que recebe um objeto Action e um objeto State;
- Função p_t: codificada no método probability; e
- Função c_t: a qual retorna o custo de uma ação e tem seu comportamento encapsulado no método cost.

Além deste leque de funções, a classe Brain possui um objeto estático do tipo State e que representa o estado atual do sistema.

5.3.8 Dimension

A classe Dimension é o elemento Dimensão D_t do modelo. Conforme explicado, uma dimensão representa uma variável controlada. Cada dimensão possui uma descrição, um valor atual, uma dimensão pai, dimensões filhas e um índice dimensional (DimensionIndex).

Cada dimensão pode ser filha de uma outra dimensão e pai de N outras dimensões. A hierarquização de dimensões facilita a criação de um modelo de predição e também facilita a remoção de elementos da nuvem. A facilidade para a predição de resultados de ações está na navegação dos elementos, pois caso uma ação afete uma determinada MV, por exemplo, já se sabe as dimensões de CPU e memória são filhas dessa dimensão. Já para a remoção, ao remover uma dimensão, também se remove todas as dimensões filhas.

A função dx que relaciona uma dimensão ao seu índice foi

implementada nesta classe com o método *getPossibleValues* que retorna um objeto da classe DimensionIndex.

5.3.9 State

A classe State é o estado da nuvem. Por representar uma fotografia do ambiente, ela está relacionada com várias outras classes. Nas instâncias desta classe é que estão os valores das dimensões, as partes interessadas existentes no momento e os índices das dimensões.

O método clone, assim como em outras classes, é responsável por criar uma cópia de um estado. Esse método é muito utilizado para criar estados hipotéticos, para avaliar os resultados de uma ação. Com um objeto clone do estado atual, basta modificar os possíveis resultados de uma ação para então avaliar esse novo estado.

5.3.10 DimensionIndex

O índice dimensional X_{dt} definido no modelo resultou na classe DimensionIndex. Essa classe possui uma lista de valores que podem ser assumidos por uma referida dimensão.

5.4 GERENCIADOR

O objetivo do modelo é criar um SMG em que o gerenciador seja um jogador que tome as suas decisões com base no modelo proposto. Sendo assim, o cerne da simulação é o comportamento do gerenciador. O gerenciador deve escolher entre ações em tempos de tomada de decisão. Os tempos de tomada de decisão variam em cada um dos experimentos, porém o comportamento do gerenciador será sempre o mesmo.

Para escolher a melhor ação a ser tomada, o gerenciador terá um comportamento similar ao descrito na seção 4.2. Ele avaliará as ações que ele pode tomar, verificar em que estados essas ações poderiam levar e mensurar quão interessantes são estes estados para todas as partes interessadas e considerar o custo das ações.

Em mais detalhes, podemos dizer que cada possível estado será comparado por um valor i. Sendo c o custo de se tomar a ação que resultará no estado comparado, temos que

$$\text{Se } c \geq 1 \text{ então}$$
$$i = \frac{\sum_{u \in U} v_t(u)}{c}$$
$$\text{Se } c < 1 \text{ então} \qquad (17)$$
$$i = \sum_{u \in U} v_t(u) + 1 - c.$$

Definidos os valores de i, é possível definir como será a escolha da melhor ação. Os valores de i de cada estado serão multiplicados pela probabilidade das ações levarem para ele e o que retornar o maior valor terá a respectiva ação escolhida.

5.5 Experimentos e resultados

Para avaliar a eficácia do modelo, foram elaborados dois experimentos. O primeiro experimento simula a alocação e migração de MVs, utilizando o modelo e funções de interesses que consideram a afinidade entre essas MVs. Neste experimento as MVs de um consumidor são preferencialmente alocadas em um mesmo servidor, diminuindo o tráfego de dados pela rede. Os resultados são comparados com as políticas de *Round-Robin* (que faz a distribuição justa das MVs) e *First-Fit* (que aloca as MVs no primeiro servidor em que servirem).

Já o segundo experimento simula o funcionamento de uma nuvem por 60 minutos. Nesta simulação várias funções de interesse são consideradas para a tomada de decisão do gerenciador, o qual pode instanciar e migrar MVs, além de escolher o próximo período de avaliação de migração. O principal objetivo deste experimento é avaliar o comportamento do modelo e o tempo gasto com a tomada de decisão com funções de interesse mais elaboradas em uma simulação mais longa.

Ambos os experimentos foram executados em um computador com processador Intel i5 com quatro núcleos de 2,4 GHZ, 4096 megabytes de memória RAM e sistema operacional Windows 7 64 bits.

5.5.1 Experimento 1 – Alocação de MVs considerando afinidade

Esta seção descreve um experimento que utiliza o simulador desenvolvido com o objetivo de avaliar a viabilidade de uso do modelo proposto para subsidiar a tomada de decisão na alocação e migração de

MVs em servidores, considerando interesses simplistas de cada parte interessada. A simulação cria um jogo colaborativo em que os consumidores fazem as suas jogadas e o gerenciador escolhe pela ação que irá trazer o maior somatório de interesses para as partes, buscando um equilíbrio de Nash.

As partes interessadas são divididas entre consumidores e o provedor da nuvem. Os consumidores são clientes de uma nuvem IaaS e solicitam MVs ao longo do tempo. Eles têm interesse de ter suas solicitações atendidas e um interesse ainda maior de explorar a afinidade entre as suas MVs, agrupando-as em servidores. Já o provedor da nuvem tem o interesse de manter o menor número de servidores com MVs, podendo assim manter os outros servidores desligados.

Os resultados obtidos nas simulações com o uso do modelo para alocação das MVs são comparados com os das simulações, utilizando as políticas de alocação First-Fit e Round-Robin. Também são comparados os resultados das simulações com o uso do modelo para alocação e migração das MVs.

5.5.1.1 Cenários

O uso do modelo será avaliado para subsidiar a alocação e migração de MVs de dois tipos diferentes, uma que necessita de 2048 megabytes memória e 1 núcleo de processador (Tipo 1) e outra que necessita do dobro dessa capacidade (Tipo 2). As MVs serão alocadas em servidores, e a configuração destes recursos ocorre conforme mostrado na Tabela 6.

Tabela 6 - Configuração dos Servidores e MVs.

	Memória	Núcleos de CPU
Servidores	8192 mb	4
MVs Tipo 1	2048 mb	1
MVs Tipo 2	4096 mb	2

Cabe-se estabelecer algumas regras aplicáveis a estas simulações:

- A ordem de solicitação de MVs é aleatória;
- Cada consumidor irá solicitar somente uma MV por vez;
- Os servidores que terminarem a simulação sem uso serão desligados.

As simulações são divididas em dois momentos. O primeiro momento (T1) é o inicial, em que as primeiras solicitações são feitas e nenhuma MV está alocada. O segundo momento (T2) ocorre 5 segundos

após o atendimento das solicitações do período inicial, em que já existirão MVs alocadas nos servidores. Serão considerados três tipos de distribuições das solicitações das MVs durante as simulações, conforme a Tabela 7.

Tabela 7 - Distribuição das MVs.

	T1	T2
Caso 1	20 % das MVs	80% das MVs
Caso 2	80% das MVs	20% das MVs
Caso 3	50% das MVs	50% das MVs

Serão considerados três tipos de *data centers* nas simulações, denominados pequeno, médio e grande. O número de servidores em cada *data center* é apresentado na Tabela 8.

Tabela 8 - Tamanhos dos data centers.

Data center	Número de servidores
Pequeno	10
Médio	50
Grande	200

As demandas por MVs dos dois tipos serão feitas por 4 consumidores de maneira aleatória. O número total de solicitações por consumidor também será aleatório, obedecendo os intervalos relacionados aos tipos de *data centers*, de acordo com a Tabela 9.

Tabela 9 - MVs por consumidor.

Data center	Número de MVs por consumidor
Pequeno	2-5
Médio	10-20
Grande	40-50

5.5.1.2 Critérios para Tomada de decisão

O objetivo deste experimento é uma prova de conceito, em que se busca validar o uso do modelo para subsidiar a tomada de decisão de um gerenciador. Para tanto, as decisões que o gerenciador deverá tomar serão escolher os servidores para alocar as MVs e quais MVs migrar.

Alocar e migrar MVs não é uma tarefa trivial pois vários fatores podem ser considerados. Fatores estes como: economia de energia,

redução de custos, otimização de recursos, emissão de CO^2, entre outros. Contudo, dado o objetivo do experimento, para as simulações com o uso do modelo o gerenciador somente irá tomar decisões com base nas funções de interesse e custo.

As funções de interesse neste experimento consideraram a afinidade entre MVs e o interesse do provedor que é a redução do número de servidores ligados ao término da simulação.

5.5.1.3 Funções de Interesse e Custo

Ao todo, existem cinco partes interessadas mapeadas nestas simulações. Quatro delas são os consumidores e a outra é o provedor da nuvem. Os interesses de cada parte interessada podem ser descritos pela Tabela 10.

Tabela 10 - Interesses por parte interessada.

Parte interessada	Ação	Interesse
Consumidor 1	Solicitar MVs	MVs disponíveis e de preferência agrupadas em um mesmo servidor.
Consumidor 2	Solicitar MVs	MVs disponíveis e de preferência agrupadas em um mesmo servidor.
Comsumidor 3	Solicitar MVs	MVs disponíveis e de preferência agrupadas em um mesmo servidor.
Comsumidor 4	Solicitar MVs	MVs disponíveis e de preferência agrupadas em um mesmo servidor.
Provedor da nuvem	Criar as MVs solicitadas e migrar MVs	Ter o maior número possível de servidores desligados

MVs que são providas por um mesmo servidor possivelmente terão uma melhor comunicação, menor consumo de energia e menor utilização de recursos de rede. Neste sentido, podemos inferir que quanto menor o número de servidores que possuem MVs de um mesmo consumidor, mais este consumidor poderá se beneficiar destas vantagens. Por esse motivo que os consumidores, além de quererem suas MVs, irão preferir se elas estiverem em um servidor que já possui ao menos uma MV utilizada por ele.

Conforme apresentado na descrição do modelo, a função de interesse retorna γ recebendo um estado e uma parte interessada. Para os consumidores, γ tem os valores seguintes:

Caso a última MV solicitada esteja disponível,
$\gamma = 0$
Caso a última MV solicitada não esteja disponível,
$\gamma = -1$ (18)
Caso a última MV solicitada seja alocada em um servidor com outras MVs do mesmo consumidor,
$\gamma = 1$.

Para o provedor da nuvem, γ pode ser descrito conforme a equação (19), sendo que hd é o número de servidores desligados e h é numero total de servidores.

$$\gamma = -1 + \left(\frac{hd}{h}\right) * 2 \quad (19)$$

Conforme explicado, neste experimento a tomada de decisão pode ser a escolha de um servidor para se alocar uma MV ou então a migração de uma MV. Assume-se que não existe diferença de custo entre alocar uma MV no servidor A ou no servidor B. Sendo assim, a função custo irá retornar sempre 1 nestes casos. No mesmo sentido, para a migração, também assume-se que o custo será de 1. Nos pontos de tomada de decisão, uma opção é não executar nenhuma ação e nestes casos o custo será 0. Sendo c o custo de se executar uma ação, temos então que:

Se a ação for de alocação,
$c = 1$
Se a ação for uma migração, (20)
$c = 1$
Se a ação for não fazer nenhuma ação,
$c = 0$.

5.5.1.4 Pontos de tomada de decisão

A dinâmica proposta é que cada parte interessada execute suas ações na sua vez. Os consumidores executam as suas ações sem o

controle do gerenciador. O gerenciador irá executar as suas ações somente nos seus pontos de tomada de decisão.

Para as simulações em que somente são consideradas as alocações, os pontos de tomada de decisão do gerenciador ocorrerão logo após um consumidor solicitar uma MV. Já para as simulações com migração, além destes pontos definidos, também poderão ser tomadas decisões ao término de T1 e T2. Nestes casos, o simulador irá escolher se migra ou não cada uma das MVs.

5.5.1.5 Resultados

As simulações com a política Round-Robin fazem a alocação das MVs distribuindo-as por servidores. A primeira MV solicitada é alocada no servidor 1, a segunda no servidor 2 e assim por diante, até que todos os servidores possuam uma MV e então começa-se o processo novamente. Caso a MV não possa ser alocada no servidor da vez, o próximo é escolhido.

Já nas simulações com a política First-fit, as MVs são alocadas no primeiro servidor que servir a MV. Existe então uma lista de servidores e a cada solicitação os servidores são testados em ordem em busca de um servidor que a MV possa ser alocada.

Os resultados obtidos com uso do gerenciador e do modelo proposto somente para alocação e os resultados obtidos com o uso para alocação e migração das MV são comparados com os resultados obtidos com as políticas *Round Robin* e o *First-Fit*. Todas as simulações foram executadas 385 vezes para cada cenário e distribuição de MV, garantindo um intervalo de confiança de 95% e margem de erro de 5%. Na Tabela 11 são apresentados os percentuais médios de servidores desligados.

Tabela 11 - Percentual médio de servidores desligados – experimento 1.

Datacenter	Caso	Alocação e migração gerenciadas	Somente alocação gerenciada	Round Robin	First-fit
P	1	44,28%	42,10%	0,22%	39,28%
P	2	49,05%	40,68%	0,35%	41,71%
P	3	46,83%	41,20%	0,07%	40,71%
M	1	57,04%	56,45%	1,81%	54,57%
M	2	61,41%	55,87%	1,86%	54,01%
M	3	58,10%	55,88%	1,79%	53,57%
G	1	66,79%	64,57%	11,58%	62,79%
G	2	71,68%	64,74%	11,65%	64,50%
G	3	69,02%	64,13%	11,71%	63,81%

Conforme esperado, verificamos que o uso do modelo somente para a alocação trouxe um aumento significativo no número de servidores desligados quando comparado com o *Round Robin* em todos os casos. De fato, a política *Round Robin* pode ser vantajosa em casos como de balanceamento de carga, mas certamente não serve para agrupamento de MVs em servidores.

Diferentemente do que era esperado antes da realização do experimento, a política *First-fit* apresentou resultados muito próximos dos obtidos com o uso do gerenciador somente para a alocação das MVs. Após os resultados, foi concluído que como as funções de interesse são muito simples e não utilizam previsões, o gerenciador muitas vezes tomou decisões equivalentes às tomadas pela política *First-fit.*

O melhor desempenho foi apresentado com o uso do gerenciador para alocação e para a migração das MVs, apesar da ação de migração ser avaliada somente uma vez por MV em T1 e em T2. Verifica-se que para todos os tamanhos de data centers, o acoplamento foi maior no caso 2 de distribuição de MVs (80% em T1 e 20% em T2) e mediano para o caso 3. Isso já era esperado, pois no caso 2 80% das MVs já estão alocadas em T1 e são avaliadas quanto a migração duas vezes (ao término de T1 e de T2).

As funções de interesse dos consumidores dão preferência para que as suas MVs fiquem em um servidor que já possuem MV. Neste sentido, na Tabela 12 é apresentada a média da quantidade de servidores que possuem MVs de um mesmo consumidor.

Tabela 12 - Média de servidores por consumidor - experimento 1.

		Gerenciador alocação e migração	Gerenciador somente alocação	Round Robin	First-fit
P	1	2,40	2,70	2,95	2,76
P	2	2,45	2,61	2,89	2,66
P	3	2,41	2,69	3,10	2,93
M	1	10,01	10,98	13,34	11,01
M	2	9,98	10,57	13,29	11,21
M	3	10,11	11,04	13,35	11,09
G	1	31,98	33,66	44,19	34,57
G	2	26,02	33,71	44,00	35,34
G	3	29,77	34,00	43,89	35,11

Assim como na avaliação de número de servidores desligados, a política *Round-Robin* também teve os piores resultados em todos os cenários também no critério de servidores relacionados com consumidores. Como se esperava também, a alteração na distribuição das demandas não impactou em alteração significativa dos resultados obtidos com essa política.

Mais uma vez os resultados obtidos com a política *First-fit* foram muito próximos dos valores com o uso do gerenciador somente para alocação. Isto pois, conforme explicado, o gerenciador sem previsão tomou decisões simulares a do *Firs-fit*.

Quando avaliamos os resultados obtidos com o uso do gerenciador também para migrações, verificamos o real potencial do uso do modelo para tomada de decisão. Por conta do custo envolvido na migração, apenas as migrações mais vantajosas foram realizadas. O gerenciador também só avaliou cada MV uma vez quanto a migração e não decidiu quanto a sequência de migrações, o que possivelmente traria resultados ainda melhores.

Também como nos resultados de número de servidores desligados, o número de servidores por consumidor foi melhor no caso 2 de distribuição e médio no caso 3. Quando comparados com os resultados obtidos com o uso do *First-fit*, o uso do modelo propiciou uma melhora de até 26%, para os *data centers* grandes no caso 2.

5.5.2 Experimento 2 – Abordagem multicritério

O experimento descrito nesta seção simula o uso do modelo

proposto para a tomada de decisão de um gerenciador em um IaaS na alocação e migração de MVs e no ajuste do tempo de avaliação da migração. Diferente do primeiro experimento, neste a simulação irá considerar a carga de trabalho que os consumidores conseguem enviar, o consumo de energia, os intervalos para avaliação de migração e o tempo gasto para se tomar as decisões.

O principal objetivo deste experimento é avaliar o tempo gasto na tomada de decisão com o uso do modelo. Conforme explicado na seção 5.4, a tomada de decisão com o uso do modelo depende diretamente dos resultados da função de interesse e da função de custo. As funções consideradas no experimento anterior eram simplistas e consideravam poucos critérios.

Neste sentido, o experimento 2 irá considerar funções de interesse e custo multicritérios. De maneira similar ao experimento 1, neste o gerenciador também busca o equilíbrio de Nash, escolhendo sempre a decisão que possui o maior somatório de interesse de todas as partes interessadas e menor custo.

As funções de interesse e custo deste experimento envolvem previsões sobre o estado futuro que uma ação pode levar o sistema. Calcular com precisão o estado resultante de uma ação é uma tarefa complexa, custosa e sujeita a erros. Como o objetivo deste experimento é avaliar o uso do modelo proposto e não a capacidade de realizar previsões, serão utilizadas sub-simulações[5] para se verificar para qual estado da nuvem uma ação irá levar. O uso sub-simulações eliminará possíveis erros de predição e dispensa o uso de um modelo de predição.

5.5.2.1 Cenários

O experimento irá simular o funcionamento de uma nuvem durante 60 minutos. No primeiro momento, os consumidores solicitarão as suas máquinas virtuais. Após isso, eles irão enviar cargas de trabalho até o final da simulação.

Existirão dois tipos de carga de trabalho. Todas as cargas de trabalho irão processar um arquivo de 500 megabytes, porém a quantidade de processamento necessária para cada uma será distinta. As cargas do Tipo 1 utilizarão 500 milhões de instruções (MI), já as do tipo 2, utilizarão 50.000 MI.

Serão considerados neste experimento dois tipos de máquinas

[5] O tempo gasto para se processar as sub-simulações não é considerado na simulação principal.

virtuais e dois tipos de servidores. O primeiro tipo de MV necessita de 2048 megabytes memória e 1 núcleo de processador (Tipo 1) e o segundo tipo necessita do dobro dessa capacidade (Tipo 2).

O servidores possuem 8192 mb de memória e 4 núcleos de processadores. Os servidores do Tipo 1 possuem processadores que processam até 2400 milhões de instruções por segundo (MIPS) e os do tipo 2 processam 1200 MIPS. O consumo máximo de energia dos servidores também são distintos e a configuração dos servidores e MVs se dá conforme a Tabela 13.

Tabela 13 - Configuração dos Servidores e MVs.

	Memória	CPUs	MIPS/CPU	Consumo máximo de energia (w)
Servidores Tipo 1	8192 mb	4	2400	400
Servidores Tipo 2	8192 mb	4	1200	250
MVs Tipo 1	2048 mb	1	-	-
MVs Tipo 2	4096	2	-	-

Os servidores terão seus custos parametrizados de acordo com a Tabela 14.

Tabela 14 – Custos.

Custo por:	Servidores Tipo 1	Servidores Tipo 2
Segundo de processador	0,1	0,15
MB de memória Ram	0,05	0,05
MB de armazenamento	0,001	0,001
MB de uso de banda de rede	0,1	0,1

Similar ao primeiro experimento, serão considerados três tipos de *datacenters* nas simulações, denominados pequeno, médio e grande conforme a Tabela 15.

Tabela 15 - Tamanhos dos data centers.

Datacenter	Servidores Tipo 1	Servidores Tipo 2
Pequeno	5	5
Médio	25	25
Grande	100	100

O número de consumidores será fixo em 4, sendo que dois deles irão solicitar MVs do tipo 1 e outros dois MVs do tipo 2. A carga de trabalho também se dará de forma similar, conforme a Tabela 16. O número de MVs que cada consumidor irá solicitar irá variar por tamanho do *datacenter*, conforme a Tabela 17, e as solicitações serão feitas em ordem aleatória e uma por vez.

Tabela 16 – MVs e tipo de carga de trabalho por consumidor.

	Tipo de carga de trabalho	Tipo de MV
Consumidor 1	Tipo 1	Tipo 1
Consumidor 2	Tipo 2	Tipo 1
Consumidor 3	Tipo 1	Tipo 2
Consumidor 4	Tipo 2	Tipo 2

Tabela 17 - MVs por consumidor.

Data center	Número de MVs por consumidor
Pequeno	5
Médio	20
Grande	50

5.5.2.2 Pontos de tomada de decisão

A tomada de decisão pode ocorrer em vários momentos em um gerenciamento real, a depender sempre de quais são as responsabilidades atribuídas para a parte interessada em questão. Para o experimento aqui descrito, serão considerados os seguintes momentos para a tomada de decisão do gerenciador:

- Na alocação de uma MV em um servidor, tendo que decidir entre vários servidores;
- A cada n segundos, escolhendo entre migrar ou não MVs e auto-ajustando o valor de n;

Diferente o experimento 1, em que os momentos para migrações já estavam definidos, neste experimento ele será variável. O experimento se inicia já com o ajuste dessa periodicidade n. O valor de n poderá ser:

- 5 segundos
- 30 segundos
- 1 minuto
- 5 minutos
- 10 minutos

Apesar de o valor de n ser preciso, o momento em que se iniciará o ciclo de tomada de decisão pode variar. Isto porque, conforme explicado, o gerenciador só toma decisões na sua vez e quanto der o seu tempo, outras partes interessadas podem estar executando ações. Neste caso, o gerenciador entrará na fila e aguardará a sua vez.

5.5.2.3 Critérios para Tomada de decisão

Na seção 2.3.3.2, apresentou-se a classificação dos critérios de decisão proposta por Mendes (2015). Para o experimento proposto, optou-se por utilizar critérios de quatro das cinco categorias apresentadas previamente. Não foi escolhido nenhum critério de segurança, visto que este é um critério muito custoso de se mensurar e normalmente depende de ferramentas de terceiros.

A Tabela 18 apresenta os critérios que irão guiar as decisões do provedor da nuvem. O primeiro critério é do tipo falha e tem o objetivo de minimizar o número de MVs não atendidas. Por último, o critério do tipo CO^2, em que se objetiva minimizar a emissão de CO^2 vinda do consumo de energia elétrica.

Tabela 18 - Critérios para o provedor da nuvem.

Tipo	Critério	Objetivo
Falha	Número de MVs não atendidas	Minimizar
CO2	Emissão de CO^2 provinda da energia elétrica consumida no data center.	Minimizar

Em um ambiente real, cada consumidor pode ter critérios distintos para a tomada de decisão. Contudo, neste experimento todos serão guiados pelos mesmos critérios, conforme a Tabela 19. O primeiro é o tempo médio que o consumidor leva para ter as suas demandas atendidas. O segundo visa minimizar seus custos reduzindo o total devido ao servidor. O último é de falha, em se objetiva maximizar a disponibilidade.

Tabela 19 - Critérios para os consumidores.

Tipo	Critério	Objetivo
Desempenho	Tempo médio para atendimento de suas demandas	Minimizar
Contabilidade	Total devido ao provedor.	Minimizar
Falha	Disponibilidade	Maximizar

O gerenciador, como parte interessada, também terá critérios para a sua tomada de decisão conforme a Tabela 20. O primeiro é do tipo contabilidade, diz respeito a reduzir o custo com consumo de energia. O segundo é o tempo que ele irá levar para tomar uma decisão futura e que se objetiva minimizar.

Tabela 20 - Critérios para o gerenciador.

Tipo	Critério	Objetivo
Contabilidade	Custo com consumo de energia	Minimizar
Desempenho	Tempo de tomada de decisão futura	Minimizar

Os critérios apresentados estarão comtemplados nas funções de interesse e custo conforme a seção seguinte.

5.5.2.4 Funções de Interesse e Custo

Cada parte interessada possui a sua função de interesse e custo. Contudo, neste experimento, não serão feitas predições sobre quais ações as partes interessadas irão tomar no futuro. Sendo assim, todas as partes interessadas terão as suas funções de interesse definidas, porém somente o gerenciador terá também uma função de custo.

Dito isto, cabe então definir a primeira função de interesse que é a do proprietário da nuvem. Esta função nada mais é do que uma média simples das variáveis a e b

$$\gamma = \frac{a + b}{2}. \tag{21}$$

A variável a tem o seu comportamento relacionado ao critério de falha e varia de acordo a disponibilidade. Caso o número de MVs não atendidas no estado futuro di_{t+n} seja maior ou igual a atual di_t, b irá valer -1. Em outros casos b valerá 1, conforme a função

$$\begin{array}{c} Se\ di_t \leq\ di_{t+n} \\ a = -1 \\ Se\ di >\ di_{t+n} \\ a = 1 \end{array} \quad (22)$$

Quanto a variável b, ela está relacionada ao critério CO^2 e irá variar de acordo com o consumo de energia (kw/h). Caso total de energia que está sendo consumido no estado atual seja maior ou igual que este no estado futuro, b= 1. Em outros casos, $b = -1$.

$$\begin{array}{c} Se\ ce_t \geq\ n \\ b = 1 \\ Se\ ce <\ ce_{t+n} \\ b = -1 \end{array} \quad (23)$$

Os consumidores também terão a sua função de interesse. A função de interesse será uma média simples das variáveis d, e e f.

$$\gamma = \frac{d + e + f}{3} \quad (24)$$

A variável d está relacionada ao critério desempenho do consumidor e seu comportamento irá afetar o interesse do consumidor de acordo com o tempo médio para atendimento de suas demandas. Seu comportamento é de acordo com (25), sendo que tc_t representa o tempo médio no estado atual e tc_{t+n} no estado futuro.

$$\begin{array}{c} Se\ tc_t \leq\ tc_{t+n} \\ d = 1 \\ Se\ tc_t >\ tc_{t+n} \\ d = -1 \end{array} \quad (25)$$

A variável e é alterada de acordo a disponibilidade das MVs do consumidor. Sendo que mt_{t+n} é o total de máquinas virtuais solicitadas pelo consumidor no estado futuro e md_{t+n} é o subconjunto destas MVs que estão disponíveis, o valor de e é dado por

$$e = 2\frac{md_{t+n}}{mt_{t+n}} - 1. \quad (26)$$

Já a variável f está relacionada a minimização dos custos contraídos pelo consumidor. Sabendo que cc_t é o valor devido ao servidor atualmente e que cc_{t+n} é o total devido pelo consumidor no

estado futuro. Sabendo também que t é o tempo atual da simulação em segundos e que $t + n$ é o tempo futuro, temos que:

$$\begin{array}{l} Se\ \frac{cc_{t+n}}{t+n} \leq \frac{cc_t}{t} \\ f = 1 \\ Senão \\ f = -1 \end{array} \quad (27)$$

Por último, tratamos das funções do gerenciador. A sua função de interesse leva em conta o tempo que o gerenciador irá levar para tomar uma decisão no estado futuro. Desta forma, a função de interesse do gerenciador[6] é:

$$\begin{array}{c} Se\ td_t \leq 2 \\ \gamma = 1 - td_t \\ Se\ td_t > 2 \\ \gamma = -1 \end{array} \quad (28)$$

tal que td_t é o tempo estimado que o gerenciador irá levar para tomar a próxima decisão em segundos (sem considerar o tempo gasto para realizar as sub-simulações).

Conforme descrito na seção 4.1.5, as funções de custo podem ser utilizadas para o gerenciador ou na predição de quais ações podem ser tomadas por uma parte interessada. Essa segunda possibilidade não está contemplada pela simulação. Sendo assim, define-se a função custo das ações do gerenciador. Esta função será do critério contabilidade e ao mesmo tempo CO^2, pois irá considerar a energia gasta para se tomar a ação. A função custo para o gerenciador terá seu comportamento definido como:

$$c = 0{,}1\frac{e_a}{t_a} \quad (29)$$

em que e_a representa o total de energia consumido por todos os servidores, em kWh, durante a execução da ação, t_a representa o tempo de duração da ação em segundos. A multiplicação por 0,1 é uma

[6] Todas as decisões que levem mais de 2 segundos para serem tomadas terão o mesmo peso. Esse valor foi definido de forma empírica e não significa que seja o ideal.

ponderação para que o custo esteja compatível com a escala de interesse. Ter ou não um fator de ajuste depende da implementação e critérios da função e não do modelo.

5.5.2.5 Avaliação e comparação dos resultados

O primeiro critério de avaliação será o tempo total gasto na tomada de decisão com uso do modelo e o tempo médio por decisão. Conforme explicado, para a avaliação das ações a serem tomadas, serão utilizados valores obtidos com o uso de sub-simulações, ao invés de um modelo de predição. Desta forma, o tempo para a tomada de decisão a ser considerado exclui o tempo de configuração e execução das sub-simulações, visto que esses valores devem ser providos pelo modelo de predição.

Conforme explicado, as decisões no experimento 2 eram sobre a alocação de MVs e a migração destas. Ressalta-se que o tempo para se tomar uma decisão impacta mais no momento da alocação, caso em que a MV não está instanciada, do que na migração, caso em que o usuário continua sendo atendido. Sendo assim, estabeleceu-se 2 segundos por decisão como sendo razoável, visto que isso representa aproximadamente 0,05% do tempo total disponível.

Além disso, também são comparados os resultados obtidos com o uso do modelo para a tomada de decisão com os resultados obtidos sem gerência. Para as simulações sem gerência, o critério de alocação será o First-fit e sem migrações. Os critérios para comparação são os que seguem:

- Consumo de energia elétrica dos servidores do *data center*.
- Quantidade de Cloudlets processados por consumidor
- Total devido ao provedor

5.5.2.6 Resultados

O total de Cloudlets processados, o total de Cloudlets processados e o total devido ao provedor, resultantes da simulação sem o gerenciador estão dispostos na Tabela 21. Já a Tabela 22 apresenta os resultados obtidos com o uso do gerenciador e com migração de MVs.

Tabela 21 - Resultados experimento 2 sem gerenciador.

		Total de Cloudlets processados	Total devido ao provedor
P	Consumidor 1	2953	517
P	Consumidor 2	535	517
P	Consumidor 3	2953	1028,5
P	Consumidor 4	532	1028,5
M	Consumidor 1	15060	2068
M	Consumidor 2	2433	2068
M	Consumidor 3	15144	4116
M	Consumidor 4	2492	4116
G	Consumidor 1	44980	5170
G	Consumidor 2	7248	5170
G	Consumidor 3	45141	10290
G	Consumidor 4	7243	10290

Tabela 22 - Resultados experimento 2 com uso do gerenciador.

		Total de Cloudlets processados	Total devido ao provedor
P	Consumidor 1	2950	517
P	Consumidor 2	536	517
P	Consumidor 3	2952	1028,5
P	Consumidor 4	533	1028,5
M	Consumidor 1	15001	2068
M	Consumidor 2	2424	2068
M	Consumidor 3	15099	4116
M	Consumidor 4	2487	4116
G	Consumidor 1	44810	5170
G	Consumidor 2	7230	5170
G	Consumidor 3	45002	10290
G	Consumidor 4	7230	10290

Verifica-se que, conforme o esperado, o total de Cloudlets processados variou em todos os consumidores, tendo grande distinção dos consumidores com Cloudlets do tipo 1 para os do tipo 2. Neste critério, constata-se que para quase todos os consumidores houve uma redução mínima do total de Cloudlets processados quando comparados os resultados sem e com o uso do gerenciador. Isso se deve ao fato que as migrações realizadas consomem processamento e atrasam o processamento de alguns Cloudlets. Quanto ao total devido ao provedor, este variou conforme o tipo de MV solicitada por consumidor, não tendo distinção do uso do gerenciador.

A Tabela 23 apresenta o consumo de energia com e sem o uso do gerenciador. Podemos verificar que o uso do gerenciador trouxe uma redução significativa no consumo total. No cenário pequeno, a redução foi mais expressiva, chegando a quase 15%. Já no cenário grande, a redução foi relativamente menor, chegando a 9,4%

Tabela 23 - Consumo de energia.

	Consumo de energia pelos servidores sem gerenciador(kW/h)	Consumo de energia pelos servidores com gerenciador (kW/h)
P	684	583
M	2544	2364
G	4980	4512

Conforme explicado anteriormente, os resultados mais importantes deste experimento são os tempos gastos com a tomada de decisão. Ressalta-se que o tempo gasto para tomada de decisão irá variar de acordo com a estratégia do gerenciador, ambiente que está sendo executada a tomada de decisão, ações disponíveis e outros fatores. Os resultados obtidos são apresentados na Tabela 24.

Tabela 24 - Tempos gastos com a tomada de decisão e total de decisões.

	Tempo médio para a tomada de decisão[7]	Total de decisões realizadas
P	0,05s	200
M	0,23s	640
G	0,46s	1400

Quanto ao tempo médio gasto para a tomada de decisão, verificamos que em todos os cenários o tempo médio significantemente baixo. Também nota-se que quanto maior o cenário, maior o tempo médio por decisão. Isto era esperado pois um cenário maior aumenta exponencialmente o número possibilidades a serem testadas no gerenciador desenvolvido.

Já com relação ao total de decisões, verificamos que seu crescimento foi proporcional ao tamanho do cenário em que foi testado, visto que quanto maior o número de MVs e servidores, mais decisões o

[7] Os valores apresentados não consideram o tempo gasto com as sub-simulações utilizadas nas previsões.

gerenciador precisa tomar.

5.5.3 Conclusões sobre os experimentos

No primeiro experimento utilizou-se o modelo para o gerenciamento da alocação e migração de MVs na nuvem considerando a afinidade entre as MVs e a redução de servidores ligados. Verificou-se que os resultados do modelo sendo utilizado para a alocação foi levemente superior quando comparado à política *First-fit*.

O fator determinante para que o uso do modelo para alocação no primeiro experimento não fosse tão superior foi que não havia predição de comportamento no modelo. Por esse motivo, quando incluída a migração, os resultados foram relevantemente superiores.

Buscando avaliar o uso do modelo com funções de interesses e custo multicritérios, foi elaborado o segundo experimento. Como o modelo especificado deixa em aberto a implementação das funções de interesse, o experimento realizado utilizou sub-simulações para se obter resultados precisos dos estados futuros.

Assim como no primeiro experimento, o segundo fez uso de 3 tamanhos de datacenters, pequeno, médio e grande. Nestes cenários então foram avaliados os tempos gastos com a tomada de decisão, a quantidade de cargas de trabalho processadas (Cloudlets), o valor pago pelos consumidores e o consumo de energia elétrica pelos servidores.

Os resultados mostraram que o tempo médio de cada decisão chegou a 0,46 segundos e que, mesmo chegando a realizar 1400 decisões, o impacto na quantidade de Cloudlets processados foi mínimo. O total gasto por cada consumidor não teve alteração, demonstrando que todos os consumidores receberam exatamente o solicitado durante o período. Já o total de energia consumida teve uma redução quase 15% no cenário pequeno.

Ambos os experimentos demonstraram a viabilidade de uso do modelo por um gerenciador de modo a se obter um gerenciamento mais eficaz. Mesmo com funções de interesse pouco elaboradas, o experimento 1 obteve bons resultados. Já o segundo experimento demonstrou que mesmo com várias funções de interesse e com o cenário grande o tempo gasto para a tomada de decisão ainda trouxe ganhos significativos no consumo de energia e pouco impactou em outros critérios.

6 CONCLUSÕES E TRABALHOS FUTUROS

A necessidade de soluções autonômicas no gerenciamento da nuvem foi apresentada nessa dissertação e a necessidade de uma solução holística foi caracterizada. Também foram revisados os conceitos de computação em nuvem, computação autonômica, teoria do controle e teoria da decisão. Além disso, foi apresentada a dinâmica da nuvem e como ocorre a tomada de decisão no gerenciamento da nuvem, introduzindo o conceito de Estado da Nuvem e apresentando os principais critérios para a tomada de decisão.

Posteriormente, através da pesquisa bibliográfica exploratória os principais trabalhos correlatos foram elencados, descritos e comparados. A partir destes, as características necessárias para o gerenciamento autonômico do ambiente de nuvem foram levantadas e fundamentou-se a necessidade de um modelo holístico do ambiente de nuvem que suporte gerenciamento autonômico.

Com base na pesquisa desenvolvida e nos trabalhos correlatos foi proposto um modelo que descreve a nuvem computacional como um SMG, conceitua a tomada de decisão na nuvem e serve para subsidiar as decisões de um gerenciador da nuvem. O referido modelo foi especificado utilizando notações matemáticas para descrever os seus elementos, como Estado, Dimensões, Índice Dimensional, Partes Interessadas, Interesses, Ações, Eventos, Funções de Interesse e Funções de Custo.

Em seguida, uma avaliação do modelo foi desenvolvida. Esta se inicia verificando a aderência do modelo, utilizando-o para descrever os elementos descritos em trabalhos correlatos. Constata-se então que estes podem ser descritos com o modelo proposto e que, portanto, o modelo pode ser considerado holístico. Seguidamente, se dá uma argumentação no sentido demonstrar que as características autonômicas estão presentes no modelo.

No sentido de validar a eficácia do modelo proposto e avaliar os resultados que podem ser obtidos com o uso deste, optou-se por valer-se da simulação. As principais soluções que podem ser utilizadas para tanto são comparadas e o uso do *toolkit* CloudSim e do CloudReports foi considerado o mais adequado. Extensões para estes então foram desenvolvidas e o modelo foi implementado descrevendo-se as principais classes. Um gerenciador que toma decisões com uso do modelo também foi implementado e seu comportamento foi descrito.

Ao todo, dois tipos de experimentos foram desenvolvidos neste trabalho. O primeiro se dá com simulações de alocação e migração de

MVs sendo que o gerenciador que utiliza o modelo para a tomada de decisão deve decidir em qual servidor deve alocar uma MV ou quais devem ser migradas. Neste a tomada de decisão ocorreu em dois momentos, T1 e T2. Foram exibidos resultados com uso do gerenciador para alocação e migração e somente para alocação, além dos resultados obtidos com as políticas de alocação *Round-Robin* e *Firts-fit*. Esses resultados foram comparados demonstraram ospor ganhos obtidos com o uso do gerenciador.

Por fim, o segundo experimento foi desenvolvido inserindo novos critérios e novos pontos para a tomada de decisão. A tomada de decisão passou então a ocorrer em intervalos escolhidos pelo gerenciador em uma simulação de uma hora. Verificou-se que o uso do gerenciador resultou em uma redução significativa no consumo de energia e o tempo médio de cada decisão atingiu 0,46 segundos no pior cenário. A tomada de decisão teve um pequeno impacto negativo na quantidade de cargas de trabalho processadas.

6.1 PRINCIPAIS CONTRIBUIÇÕES

Em Mendes (2014) estão as primeiras contribuições deste trabalho, uma modelagem matemática para o planejamento de decisões em nuvens computacionais autonômicas. Neste modelos de teoria da decisão baseados no MDP foram adaptados para serem utilizados na fase de planejamento do MAPE-K.

Com base na pesquisa bibliográfica exploratória foi definido o estado da arte de nuvens autonômicas e da teoria da decisão aplicada no gerenciamento de nuvem. Isto somado a caracterização dos trabalhos correlatos e indicam rumos que devem ser seguidos pela comunidade científica no gerenciamento de nuvens.

A principal contribuição está no modelo desenvolvido e especificado que traz um processo para tomada de decisão de um gerenciador de nuvens autonômicas e foi apresentado em Flores (2015). Os experimentos realizados demonstraram a aplicabilidade do modelo e trouxe resultados promissores sendo que com funções de interesse e custo mais elaboradas, resultados ainda mais significantes podem ser obtidos.

6.2 TRABALHOS FUTUROS

Como trabalho futuro, propõe-se a continuidade da evolução do modelo: testando variações do modelo, criando-se novas funções de

interesse e custo, desenvolvendo modelos para predição de estados da nuvem, construindo novos gerenciadores com outras estratégias e realizando-se testes em ambientes reais.

O modelo foi implementado com base em um SMG. Contudo, outros modelos de jogos podem ser testados e seus resultados comparados com os obtidos. Facilmente pode-se adaptar o modelo para um jogo hierárquico, utilizando o modelo de Stackelberg ou ainda para um jogo *two-players non zero sum*, em que um jogador seria o gerenciador e o outro seria o acaso.

Novas funções de interesse e custo podem ser desenvolvidas, considerando mais critérios com pesos diferentes. As funções apresentadas neste trabalho, por não serem objetivo deste, não foram profundamente estudadas. Desta forma, possivelmente, melhores resultados podem ser obtidos somente realizando novos testes com outras funções.

A criação de um PaaS para o desenvolvimento das funções de interesse diretamente por seus consumidores também é um dos desafios a serem cumpridos em novas pesquisas. Um ambiente desse tipo daria escalabilidade para ambientes reais de grande porte.

Caracteriza-se também a necessidade da construção de modelos de predição eficientes a serem utilizados para alimentar as funções propostas no modelo. No primeiro experimento desenvolvido as decisões foram tomadas com o uso de funções simples sem predição. No segundo experimento, as funções careciam de resultados futuros e essa necessidade foi suprida com o uso de sub-simulações. Em um ambiente real, será necessário um modelo de predição que seja completo o suficiente para atender as necessidades do modelo de tomada de decisão e ao mesmo tempo simples o suficiente para que as decisões possam ser tomadas de maneira tempestiva.

O gerenciador construído para a simulação foi muito simplificado e muita pesquisa pode ser feita sobre o tema visando construir gerenciadores robustos, versáteis e eficientes. Além do equilíbrio de Nash, também podem ser testadas outras formas de equilíbrio para a tomada de decisão do gerenciador. Buscando resultados ainda melhores, também pode se implementar no gerenciador a criação de planos com sequências de ações como um MDP e a execução destes planos.

A continuidade dos trabalhos também passa pela adaptação do modelo a plataformas de gerenciamento de nuvem e a condução de testes rigorosos de modo a avaliar os resultados em um ambiente real. A utilização do modelo pode-se começar de maneira mais limitada, como na alocação de migração de MVs, podendo se estender por toda a

nuvem.

REFERÊNCIAS

ARDAGNA, D.; PANICUCCI, B.; PASSACANTANDO, M. **A game theoretic formulation of the service provisioning problem in cloud systems**. 20th international conference on World wide web. [S.l.]: ACM. 2011. p. 177-186.

BELOGLAZOV, A.; ABAWAJY, J.; BUYYA, R. Energy-aware resource allocation heuristicsfor efficient management of data centers for Cloudcomputing. **Future Generation Computer Systems**, v. 28, n. 5, p. 755-768, 2012.

BERGER, J. O. **Statistical decision theory and Bayesian analysis**. [S.l.]: Sprincer Sience & Business Media, 2013.

BUYYA, R. et al. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. **Future Generation computer systems**, 25, n. 6, 2009. 599-616.

CALHEIROS, R. N. et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. **Software: Practice and Experience**, 41, 2011. 23-50.

CORRADI, A.; FANELLI, M.; FOSCHINI, L. VM consolidation: a real case based on openstack cloud. **Future Generation Computer Systems**, p. 118-127, 2014.

CORRADI, A.; RANA, O. F. The management of cloud systems. **Future Generation Computer Systems**, v. 32, p. 24-26, 2014.

FITÓ, J. O. et al. Business-Driven IT Management for Cloud Computing Providers. **CloudCom**, p. 193-200, 2012.

FLORES, A. A. et al. **Decision-Theoretic Model to Support Autonomic Cloud Computing**. The Fourteenth International Conference on Networks. Barcelona: Iaaria. 2015.

FOSTER, I. et al. **Cloud Computing and Grid Computing 360-Degree Compared**. Grid Computing Environments Workshop. Austin, TX: [s.n.]. 2008. p. 1-10.

GASIOR, J.; SEREDYNSKI, F. A Game-Theoretic Approach to Multiobjective Job Scheduling in Cloud Computing Systems. In: _____ **Parallel &**

Distributed Processing Symposium Workshops (IPDPSW). [S.l.]: IEEE, 2014. p. 470-479.

HUEBSCHER, M. C.; MCCANN, J. A. A survey of autonomic computing - degrees, models, and applications. **ACM Computing Surveys (CSUR)**, 40, n. 3, 2008. 7.

KEPHART, J. An architectural blueprint for autonomic computing, 2003. Disponivel em: <http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf> . Acesso em: dez. 2015.

KEPHART, J.; CHESS, D. M. The vision of autonomic computing. **Computer**, 36, 2003. 41-50.

KIM, H. et al. Autonomic Management of Application Workflow on Hybrid Computing Infrastructure. **Scientific Programming**, v. 19, n. 2, p. 75-89, 2011.

KSENTINI, A.; TALEB, T.; CHEN, M. **Markov decision process-based service migration procedure for follow me cloud**. Communications (ICC), 2014 IEEE International Conference on. [S.l.]: IEEE. 2014. p. 1350-1354.

LALANDA, P.; MCCAIN, J. A.; DIACONESCU, A. **Autonomic Computing:** Principles, design and implementation. [S.l.]: Springer Science & Business Media, 2013.

LIU, Y.; JI, Y.; JIAO, R. J. A Stackelberg Solution to Joint Optimization Problems: A Case Study of Green Design. **Procedia Computer Science**, 16, 2013. 333-342.

LORETI, D.; CIAMPOLINI, A. **Policy for Distributed Self-Organizing Infrastructure Management in Cloud Datacenters**. [S.l.]: [s.n.]. 2014. p. 37-43.

LOZOVANU, D.; PICKL, S. **Optimization and Multiobjective Control of Time-Discrete Systems**. [S.l.]: Springer, 2009.

MASTELIC, T. et al. Cloud Computing: Survey on Energy Efficiency. **ACM Computing Surveys (CSUR)**, v. 47, p. 33, 2014. ISSN 2.

MELL, P.; GRANCE, T. **SP 800-145. The NIST Definition of Cloud Computing**. National Institute of Standards. Gaithersburg, MD, United States. 2011.

MENDES, R. et al. **Decision-Theoretic Planning for Cloud Computing**. The Thirteenth International Conference on Networks. Nice: Iaaria. 2014. p. 191-197.

MENDES, R. et al. Taxonomy, model and survey on Management of Computing in Clouds. **in proceedings**, 2015.

MUPPALA, J. K. et al. Dependable Systems \& Networks (DSN), 2011 IEEE/IFIP 41st International Conference on. In: _____ **Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on**. [S.l.]: IEEE, 2011. p. 590 - 591.

MURTHY, D. P.; JACK, N. Introduction to Stochastic Optimisation and Game Theory. In: _____ **Extended Warranties, Maintenance Service and Lease Contracts**. [S.l.]: Springer, 2014. p. 77-87.

NASH, J. **Non-cooperative games**. Annals of mathematics. [S.l.]: [s.n.]. 1951. p. 286-295.

NASH, J. F. **The Bargaining Problem**. [S.l.]: JSTOR, 1950.

PALMIERI, F. et al. A distributed scheduling framework based on selfish autonomous agents for federated cloud environments. **Future Generation Computer Systems**, v. 29, n. 6, p. 1461-1472, 2013.

PUTERMAN, M. L. **Markov decision processes:** discrete stochastic dynamic programming. [S.l.]: John Wiley & Sons, 2014.

SÁ, T. T.; SOARES, J.; GOMES, D. G. CloudReports: uma ferramenta gráfica para a simulaç ao de ambientes computacionais em nuvem baseada no framework CloudSim, 2011.

SAKELLARI, G.; LOUKAS, G. A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing. **Simulation Modelling Practice and Theory**, v. 39, p. 92-103, 2013.

SAUVÉ, J. et al. **An Introductory Overview and Survey of Business-driven**. Business-Driven IT Management, 2006. BDIM'06. The First IEEE/IFIP International Workshop on. [S.l.]: [s.n.]. 2006. p. 1-10.

SHARMA, U. **Elastic resource management in cloud computing platforms**. University of Massachusetts. [S.l.]. 2013.

SKOGESTAD, S.; POSTLETHWAITE, I. **Multivariable feedback control Analysis amd Design**. [S.l.]: [s.n.], v. 2, 2007.

STACKELBERG, H. V. **Marktform und gleichgewicht (market structure and equilibrium)**. [S.l.]: vienna, 1934.

STERRITT, R.; BUSTARD, D. Autonomic computing: a means of achieving dependability? In: _____ **10th IEEE International Conference and Workshop on the Engineering of Computer Based Systems**. [S.l.]: IEEE Computer Society, 2003. p. 247-251.

VERAS, M.; TOZER, R. **Cloud Computing:** nova arquitetura da TI. 1. ed. Rio de Janeiro: Brasport, v. 1, 2012.

VON NEUMANN, J.; MORGENSTERN, O. **Theory of games and economic behavior**. Woodstock, Reino Unido: Princeton university press, 2007.

WEI, G. et al. A game-theoretic method of fair resource allocation for cloud computing services. **The Journal of Supercomputing**, v. 54, n. 2, p. 252-269, 2010.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. **Journal of internet services and applications**, v. 1, n. Springer, p. 7-18, 2010. ISSN 1.