UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

GABRIEL ARAUJO SICCARDI MOREIRA

VertElastic: Um Módulo de decisão para explorando elasticidade vertical no Autoelastic

São Leopoldo
2018

GABRIEL ARAUJO SICCARDI MOREIRA

VERTELASTIC: UM MÓDULO DE DECISÃO PARA EXPLORANDO ELASTICIDADE VERTICAL NO AUTOELASTIC

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos, como requisito parcial para obtenção do título de Mestre em Educação.

Orientador: Prof. Dr. Rodrigo Rosa Righi

Linha de Pesquisa: Internet das Coisas e Aplicações Distribuídas.

São Leopoldo

2018

M838v Moreira, Gabriel Araujo Siccardi.
VertElastic: um módulo de decisão para explorando elasticidade vertical no Autoelastic / Gabriel Araujo Siccardi Moreira. – 2018.
75 f.: il.; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2018.
"Prof. Dr. Rodrigo Rosa Righi (Orientador). ”

1. Computação em nuvem. 2. Computação de alto desempenho. 3. Computação. I. Título.
CDU 004

Dados Internacionais de Catalogação na Publicação (CIP)
(Bibliotecário: Flávio Nunes – CRB 10/1298)

Gabriel Araujo Siccardi Moreira

VertElastic: Um Módulo de decisão para explorando elasticidade vertical no autoelastic

Dissertação apresentada à Universidade do Vale do Rio dos Sinos – Unisinos, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 25 de setembro de 2018

BANCA EXAMINADORA

Prof. Dr. Rodrigo Rosa Righi - UNISINOS

Prof. Dr. Adenauer Correa Yamin - UFPEL

Prof. Dr. Cristiano André da Costa - UNISINOS

Prof. Dr. Rodrigo Rosa Righi (Orientador)

Visto e permitida a impressão
São Leopoldo,

Prof. Dr. Rodrigo Rosa, Righi
Coordenador PPG em Computação Aplicada

Mãe, a saudade é imensa, não imaginas a falta que me fazes. Este mestrado é dedicado a ti. Te amo.

AGRADECIMENTOS

A meu pai Jesus Soares Moreira pelo exemplo de pessoa a ser copiado, por sempre me apoiar e proporcionar o necessário para que pudesse ir em busca dos meus sonhos.

À minha mãe Rosadilis Araujo Siccardi Moreira (in memoria), minha eterna gratidão a agradecimento por tudo, este mestrado é dedicado a ti, mãe.

A meu irmão Raphael pela amizade verdadeira, companheirismo e alegrias.

A meu orientador, Prof. Dr. Rodrigo Rosa Righi, pelo rumo, pelas dicas valiosas e salvadoras, mas principalmente por acreditar nesta jornada.

À Vinicius Facco Rodrigues autor do AutoElastic, pela cordialidade, paciência e boa vontade em mais de uma oportunidade sentar e conversar sobre o projeto.

A meus colegas de trabalho que nesta reta final me proporcionaram a calmaria dentro do caos do trabalho para que pudesse focar na pesquisa, especialmente ao Gustavo, pela ajuda nos códigos, as conversas, as trocas de ideias que sempre engrandecem.

E preciso agradecer especialmente a minha esposa, pela cumplicidade, companheirismo, paciência e carinho. Pelo cuidadosa revisão, dicas e sugestões, certamente sem o teu empenho nesta jornada seria muito complicado vencer, não tenho como expressar a gratidão a tua dedicação Gabriela!

RESUMO

O conceito de elasticidade está muito ligado à computação em nuvens, pois consiste na capacidade de contrair recursos computacionais de maneira dinâmica e em tempo real. Usualmente, em computação de alto desempenho (HPC), as aplicações são modeladas para serem utilizadas com técnica de balanceamento de carga, fazendo uso da tecnologia de máquinas virtual. A computação paralela há muito tem sido utilizada para resolver questões computacionais que envolvem a execução de muitos processos simultaneamente e demanda quantidade grande de cálculos, cuja premissa é que um grande trecho de código a ser processado pode ser quebrado em menores e, assim, o problema como um todo dividido e resolvido de forma mais rápida. HPC é um típico caso de uso de paralelismo computacional que tem como seu protocolo de comunicação mais comum o Message Passing Interface (MPI), porém quando estamos tratando de aplicações em MPI, o aproveitamento máximo da elasticidade se dá de forma trabalhosa, com a necessidade de reescrita de código, de conhecimento profundo do comportamento da aplicação, além de serem inevitáveis algumas interrupções na aplicação para recompilar novas e pô-la em produção. A fim de evitar a reescrita de código e o aproveitamento total dos hardwares que estão cada vez mais robustos propõe-se na pesquisa desta dissertação a possibilidade de implementação de elasticidade vertical para trabalhar com aplicação de alto desempenho. Um modulo de decisão chamado VertElastic, é incorporado ao framework AutoElastic permitindo assim que se expanda a possibilidade para as duas formas de elasticidade – vertical e horizontal, podendo ainda ser feita de forma fixa com a indicação de threholds ou com predição os valores sejam calculados automaticamente. Trabalhos abordam a elasticidade vertical com threshold, já outros se valem da elasticidade horizontal de forma proativa e/ou reativa, porém não se encontrou pesquisas que permitissem a flexibilidade de se utilizar elasticidade vertical ou horizontal conforme a necessidade de forma proativa ou reativa, para isso o VertElastic se utiliza da elasticidade assíncrona, proporcionando que a aplicação não seja bloqueada enquanto a elasticidade acontece, seja ela para aumentar ou diminuir o recurso computacional. O VertElastic demostra sua viabilidade em uma rotina de testes executados na ferramenta open source *OpenNebula*. A execução de uma aplicação *CPU-Bound* demostra que o VertElastic se mostrou entre 13% e 38% mais eficaz que a não utilização de nenhuma técnica de elasticidade. Os testes ainda mostraram que quanto maior o threshold utilizado menor é o ganho no consumo de recursos computacionais e maior o tempo de execução da aplicação.

Palavras-chave: Computação em Nuvem. Elasticidade. Computação de Alto Desempenho.

ABSTRACT

The concept of elasticity is closely linked to cloud computing because it consists of the ability to contract computational resources dynamically and in real time. Usually, in high performance computing (HPC), applications are modeled for use with load balancing technology, making use of virtual machine technology. Parallel computing has long been used to solve computational issues involving the execution of many processes simultaneously and demand large amounts of computations whose premise is that a large piece of code to be processed can be broken into smaller ones and thus the problem as a whole divided and resolved more quickly. HPC is a typical case of use of computer parallelism that has as its most common communication protocol Message Passing Interface (MPI), but when we are dealing with applications in MPI, the maximum use of elasticity occurs in a laborious way, with the need code rewriting, deep knowledge of application behavior, and some interruptions in the application to recompile new ones and put it into production are inevitable. In order to avoid the rewriting of code and the total use of hardwares that are increasingly robust, it is proposed in the research of this dissertation the possibility of implementing vertical elasticity to work with high performance application. A decision module called VertElastic is incorporated into the AutoElastic framework, thus allowing the possibility for both forms of elasticity - vertical and horizontal - to be expanded, and can be done in a fixed way with the indication of threholds or with prediction values are calculated automatically. Studies deal with vertical elasticity with threshold, while others use proactive and / or reactive horizontal elasticity, but no research was found that allowed the flexibility to use vertical or horizontal elasticity as needed proactively or reactively, for this the VertElastic uses the asynchronous elasticity, providing that the application is not blocked while the elasticity happens, be it to increase or decrease the computational resource. VertElastic demonstrates its feasibility in a testing routine run on the open source OpenNebula tool. The execution of a CPU-Bound application showed that VertElastic was 13% to 38% more effective than the non-use of any elasticity technique. The tests also showed that the higher the threshold used the lower the gain in the consumption of computational resources and the longer the execution time of the application.

Keywords: Cloud Computing. Elasticity. High Performance Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	Application Program Interface
CLI	Command Line Interface
HPC	High Performance Computing
IaaS	Infrastructure as a Service
ICLB	Interconnect Cloud Load Balance
MPI	Message Passing Interface
MV	Máquina Virtual
NFS	Network File System
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
SaaS	Software as a Service
SLA	Service Level Agreement
SLO	Service Level Objectives

SUMÁRIO

1 INTRODUÇÃO

O conceito de Elasticidade, embora originalmente desenvolvido nas áreas da Física e da Economia, tem sido empregado em larga escala na área da Computação — e, mais especificamente, na Computação em Nuvem que consiste basicamente numa espécie de *utility computing* (pagamento pelo uso e adaptação à demanda), ou seja, um sistema em que usuários pagam pelo processamento, armazenamento e transferência de dados de acordo com o que efetivamente é utilizado.

A associação do conceito de elasticidade à computação em nuvem se deve ao fato de o primeiro estar relacionado à capacidade de contrair recursos computacionais de maneira dinâmica e em tempo real. Se seguirmos a definição adotada por Sharma et al. (2011), diríamos que elasticidade consiste na capacidade de adicionar e remover recursos de forma automática, de acordo com a carga de trabalho, sem interrupções, e utilizando os recursos de forma otimizada. Em outras palavras, elasticidade na computação diz respeito à adaptação automática de um sistema à variação da carga de trabalho de forma praticamente instantânea.

Para implementar a elasticidade, as principais formas utilizadas são técnicas de replicação e redimensionamento de recursos. Com a utilização da computação em nuvem, a percepção que se tem é de que os recursos são ilimitados, e isso é muito bem explorado pela elasticidade.

Outro conceito amplamente utilizado na computação e que é comumente relacionado à elasticidade é o conceito de escalabilidade. Em determinados contextos, ambos chegam a ser confundidos. No entanto, elasticidade e escalabilidade possuem uma sutil diferença que, para os objetivos desta pesquisa, tem significativa relevância.

A escalabilidade satisfaz a necessidade de aumentar a capacidade de expansão linear de acordo com a carga de trabalho. Não se busca verificar se existe uma plena utilização desses recursos adicionados — e normalmente não existe uma remoção deles, ocasionando desperdício de recurso e, por consequência, desperdício de energia e aumento do custo operacional.

Já a elasticidade busca justamente fazer o uso eficiente e otimizado dos recursos. Assim, as aplicações são monitoradas para que sejam alocados mais recursos quando for necessário um maior poder de processamento e mantém-se a vigilância para que os recursos sejam deslocados tão logo não estejam mais sendo úteis, a fim de evitar o gasto desnecessário e priorizando a eficiência e economia. Essa dinamicidade é uma das qualidades mais destacadas da elasticidade atualmente, junto com a forma transparente de lidar com toda a camada mais baixa do hardware.

Em aplicações paralelas, em geral, a escalabilidade necessita de uma adaptação da aplicação para que consiga tratar de forma linear as mudanças de hardware. O middleware AutoElastic vem a tornar muito mais prática, fácil e dinâmica a exploração da escalabilidade horizontal, que consiste em adicionar novos nós computacionais a uma arquitetura. A definição de Ferreira (2010) sintetiza essa prática que consiste em adicionar um novo servidor e um sistema de software que permita a distribuição do trabalho entre múltiplas máquinas, e o AutoElastic otimiza essa ação de forma automática e simplificada.

1.1 Motivação

Vários trabalhos abordam com eficiência a utilização de elasticidade horizontal em aplicações de alto desempenho, entretanto, ainda existe outra forma de se beneficiar da elasticidade. Trata-se de aumentar o poder computacional do nó aumentando a quantidade de recursos, normalmente CPU e memória, buscando que, com isso, seja possível atender a demanda crescente de poder computacional sem a necessidade de aumentar a quantidade de nós que estão disponíveis para executar a tarefa e abordar esta lacuna foi um dos principais motivadores para a abordagem desta dissertação. Atualmente, o AutoElastic trabalha exclusivamente alocando e duplicando a quantidade de nós, seguindo sempre um template pré-definido de recursos exercendo, portanto, apenas elasticidade horizontal.

Conforme destacado por Rodrigues, em 2015, o AutoElastic introduz o conceito de elasticidade assíncrona: reorganização de recursos e processos na perspectiva dos usuários, não bloqueando nem finalizando a execução da aplicação durante as operações de alocação ou deslocação de recursos. Sendo

assim, o AutoElastic se constitui em um framework que possui um controlador que, por sua vez, atua de forma transparente, gerenciando as operações de elasticidade horizontal, não sendo necessárias adaptações ou modificações na aplicação.

1.2 Questão de Pesquisa

Partindo dessas considerações iniciais, começou a se delinear a proposta que foi executada na pesquisa aqui descrita. Seu objeto de investigação configurou-se na implementação de um módulo de tomada que permitisse ao AutoElastic se utilizar também dos benefícios da elasticidade vertical sempre que possível, tornando o AutoElastic ainda mais flexível em uma nova versão que foi denominada VertElastic.

Para realizar o estudo, partiu-se da seguinte questão de pesquisa que disparou e permeou todo o processo investigativo: *Como se configura um modelo de elasticidade para aplicações de alto desempenho, baseado no AutoElastic, que permite apenas elasticidade horizontal, que possibilite a implementação de elasticidade vertical, permanecendo totalmente transparente a necessidade da reescrita de código?*

Neste trabalho, o objetivo geral centrou-se no esforço de criar um módulo de decisão que permita ao AutoElastic aplicar elasticidade de forma vertical, sem interferência de usuários e sem prejuízo na execução da aplicação de HPC. Os objetivos específicos definidos para auxiliar na busca do objetivo geral foram os seguintes: promover um amplo entendimento a respeito da implementação do AutoElastic; realizar um levantamento bibliográfico para dar suporte aos conceitos de elasticidade e computação em nuvem; elaborar, através de outros trabalhos relacionados, um mini estado da arte sobre a aplicabilidade de elasticidade e da computação em nuvens para HPC; compreender os conceitos levantados pelos trabalhos pesquisados que tratam de elasticidade e computação em nuvem; modelar um protótipo para testes; realizar testes comparativos entre desempenho de uma aplicação utilizando AutoElastic com elasticidade horizontal e com o módulo que propõe elasticidade vertical; analisar todos resultados obtidos nos testes; proporcionar um estudo de base para a

realização de trabalhos futuros envolvendo o mesmo objeto, contribuindo assim para a pesquisa nesse campo de conhecimento.

O presente trabalho se encontra apresentado em seis capítulos, a saber: num primeiro momento, foi abordada a fundamentação teórica que dá suporte ao estudo aqui desenvolvido, com levantamento de todas as teorias envolvidas, HPC, elasticidade, AutoElastic e computação em nuvem (capítulo 2). Em seguida, foram explorados diversos trabalhos e pesquisas já realizados que demonstraram possuir alguma relação com esta investigação (capítulo 3). No quarto capítulo, é apresentado o conceito novo de VertElastic para aplicar elasticidade vertical nos moldes do modelo AutoElastic — que trabalha com elasticidade na nuvem utilizando elasticidade horizontal. Em seguida (capítulo 5), foi explicitado tipo de pesquisa realizada e a metodologia adotada na elaboração de um modelo do módulo para AutoElastic implementar elasticidade vertical. No sexto capítulo, foram descritas as avaliações a partir das ações e testagens desenvolvidas, bem como as análises dos resultados que serviram de base para a escrita desta dissertação. Por fim, no capítulo 7, foram tecidas as considerações finais acerca do estudo, quais contribuições ele pode vir a oferecer para o campo de pesquisa abordado e para trabalhos futuros relacionados à temática aqui explorada.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo visa fornecer uma revisão da literatura sobre os conceitos fundamentais que balizam este trabalho, buscando assim fornecer suporte necessário aos leitores e pesquisadores para uma melhor compreensão desta investigação.

2.1 COMPUTAÇÃO EM NUVEM

O termo “computação em nuvem” surgiu em 2006, em uma palestra proferida por Eric Schmidt, da Google, a respeito da maneira como sua empresa gerenciava seus data centers (TAURION, 2009). De lá para cá, a computação em nuvem vem revolucionando o ciberespaço, configurando-se como o esteio de um movimento de profundas transformações no mundo da tecnologia e na globalização.

A definição do *National Institute of Standards and Technology* (NIST) para computação em nuvem é a seguinte:

> A computação em nuvem é um modelo para acesso conveniente, sob demanda, e de qualquer lugar, a uma rede compartilhada de recursos de computação (isto é, redes, servidores, armazenamento, aplicativos e serviços) que possam ser prontamente disponibilizados e liberados com um esforço mínimo de gestão ou de interação com o provedor de serviços (NIST, 2011).

Para se chegar a uma definição universal sobre computação em nuvem, de acordo com Chirigati (2009), é necessário considerar três conceitos que estão diretamente associados (e implicados) a ela: virtualização, escalabilidade e modelo *pay-per-use*. Esses conceitos serão explicados ao longo do capítulo.

Do mesmo modo como foi mencionado anteriormente haver uma sutil diferença entre os conceitos de Elasticidade e Escalabilidade, também é necessário que se faça distinção entre os conceitos de Virtualização e Computação em Nuvem, uma vez que comumente há uma tendência em confundi-los e tratá-los como similares.

Para Daniels (2009), a virtualização consistiria numa forma de se otimizar o uso de servidores físicos, fazendo com que vários servidores virtuais, sob o controle de um hipervisor (monitor das máquinas virtuais), possam rodar sobre o mesmo hardware. Já a computação em nuvem vai mais além disso, sendo um conceito mais amplo. A confusão se dá pelo fato de a computação em nuvem utilizar a virtualização, ou seja, “em cada máquina física do provedor de serviços de nuvem podem ser criadas várias máquinas virtuais que são alocadas ou liberadas de acordo com a necessidade” (OPUS SOFTWARE, 2015, p. 33). Resumindo: na computação em nuvem, a unidade básica de processamento é uma máquina física, e as máquinas virtuais são alocadas e liberadas pelo usuário de acordo com a demanda.

Outra definição para computação em nuvem vem de Armbrust (2009) que refere tratar-se de um conjunto de serviços de rede ativados, proporcionando escalabilidade, qualidade de serviço, infraestrutura barata de computação sob demanda e que pode ser acessada de uma forma simples e pervasiva.

Vergara (2017, p. 5 apud VAQUERO et al., 2008) fez uma avaliação de mais de 20 definições a respeito da computação em nuvem e sugeriu uma proposta unificada que pode ser traduzida como:

> Um grande conjunto de recursos virtualizados e compartilhados (hardware, plataformas de desenvolvimento ou software) que podem ser facilmente acessados e utilizados. Esses recursos podem ser dinamicamente reconfigurados para se ajustarem a uma carga variável de trabalho, permitindo 5 usos otimizados dos mesmos. Este conjunto de recursos é tipicamente explorado por um modelo de pagamento por utilização chamado de *pay-per-use* em que as garantias são oferecidas pelo provedor de infraestrutura por meio de contratos de serviço personalizados (Service Level Agreement – SLA) (VERGARA, 2017, p. 5).

Segundo NIST (MELL; GRANCE, 2011), esse modelo de computação promove disponibilidade e é composto por:

- Cinco características essenciais: Autoatendimento sob demanda; Acesso amplo à rede; Agrupamento de recursos; Rápida elasticidade; e Medição de serviço;
- Três modelos de serviço: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS);

- Quatro modelos de implantação: Nuvem Privada, Nuvem Pública, Nuvem Híbrida e Nuvem Comunitária.

2.1.1 Características essenciais

Seguindo a linha de NIST, as características essenciais que um ambiente de nuvem deve prover são as seguintes:

1. Autoatendimento sob demanda: Provisiona ao consumidor, de forma transparente e direta, recursos computacionais — por exemplo, tempo de processamento no servidor ou armazenamento na rede — na medida em que estes são necessários, sem a necessidade de interação humana com a administração do provedor;
2. Amplo acesso à rede: Os recursos estão disponíveis para acesso à rede de maneira padronizada e com possibilidade de acesso via multi-dispositivos. Segundo Mell et al. (2009), a interface de acesso à nuvem não impõe aos usuários que mudem suas condições e seus ambientes de trabalho;
3. Compartilhamento de recursos: Os recursos computacionais são compartilhados entre múltiplos clientes, atribuindo ou retribuindo recursos virtuais conforme demanda. A esse modelo denomina-se *multi-tenant* em que uma única instância roda no servidor, permitindo atender a uma diversidade de requisições de diferentes usuários. Existe ainda uma independência sobre local e não existe controle sobre os recursos, apenas informações mais gerais são fornecidas ao cliente. Em alguns casos, o cliente pode optar por contratar o recurso de determinada região ou país, variando conforme o provedor;
4. Elasticidade: Consiste na possibilidade de aumentar ou diminuir recursos operacionais, conforme a necessidade do cliente, de forma dinâmica. Em certos momentos, temos a percepção de que os recursos são infinitos. Esta é uma das características mais marcantes da computação na nuvem e um dos temas focais deste trabalho;
5. Serviços Mensuráveis: Os serviços na nuvem são monitorados e contabilizados de forma automática, otimizando sua utilização. Existem

diferentes níveis de cobrança, permitindo que o cliente seja tarifado conforme política previamente definida, o que facilita o controle do custo.

2.1.2 Modelos de Serviço

A computação em nuvem distribui os seus recursos sob a forma de modelos de serviços. Esses modelos são classificados em três categorias: Software como Serviço (SaaS), Infraestrutura como Serviço (IaaS) e Plataforma como Serviço (PaaS).

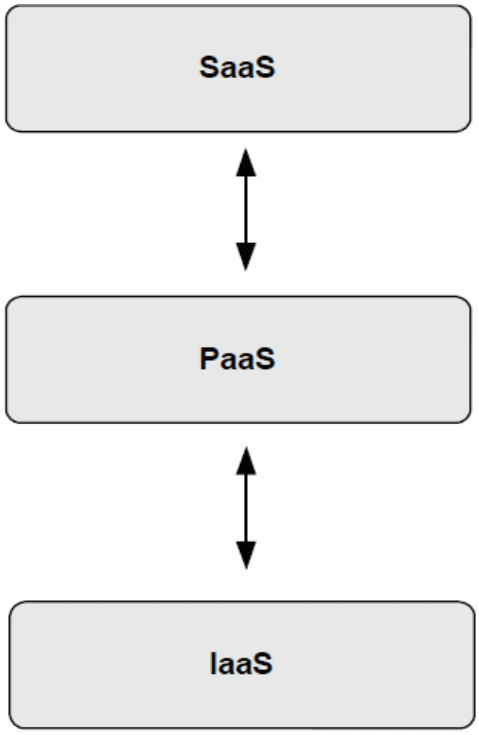


Figura 1- Modelos de serviço.
Fonte: Adaptado de Vaquero, 2009.

SaaS – Software-as-a-Service – Esse modelo se baseia na entrega de *software* como um serviço. É a categoria mais madura e mais conhecida, pois diz respeito aos serviços de mais alto nível disponibilizado em uma nuvem. O cliente pagará pelo *software* que utilizar e não precisa arcar com o custo da licença ou de ter que instalar e manter o software. A interface do software é acessível, normalmente via uma web browser. A monitoração da aplicação, a garantia de performance e o gerenciamento centralizado são de

responsabilidade do provedor de serviços. Alguns exemplos de SaaS são os *softwares* de aplicação, como processadores de texto e sistemas de bancos de dados.

PaaS – Plataform-as-a-Service – Diferentemente das funções oferecidas pelo modelo SaaS, é um modelo mais recente. O PaaS provê uma plataforma de software sobre a qual os clientes podem desenvolver suas próprias aplicações e hospedá-las utilizando a infraestrutura do provedor de serviços. A plataforma de software é utilizada como um framework para construção das aplicações. O provedor disponibiliza ao cliente plataformas de banco de dados e servidores de componentes e serviços para serem utilizados pelas aplicações. O cliente não tem a preocupação com a elasticidade, escalabilidade ou com a plataforma (sistema operacional e hardware), ou qualquer custo referente à infraestrutura. Por enquanto, o cliente se vê obrigado a utilizar as APIs (Application Programming Interface) do provedor de serviços, o que levanta a preocupação com a portabilidade das aplicações.

IaaS – Infrastructure-as-a-Service – Nesse modelo, o cliente utiliza a infraestrutura de TI necessária para suas operações como equipamentos de hardware, storage, rede e servidores que são mantidos pelo provedor de serviço. O cliente pode instalar sistemas operacionais e aplicações sobre essa infraestrutura e pagar pelo uso dos recursos. O cliente também pode instalar o sistema operacional e a aplicação que desejar sobre a infraestrutura disponibilizada. O IaaS é o que oferece o maior nível de controle comparado ao SaaS e PaaS, porque nesse caso, a escalabilidade e elasticidade são responsabilidade do cliente, e não do provedor de serviços.

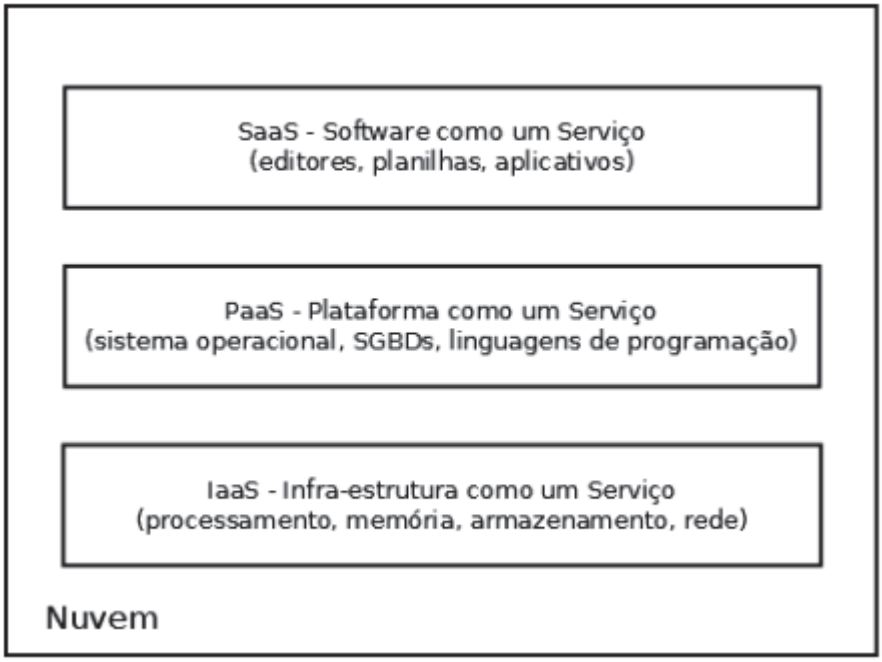


Figura 2- Arquitetura de Serviços na Nuvem
Fonte: Adaptado de Vaquero, 2009

2.1.3 Modelos de Implantação

A forma de implantação de uma nuvem está diretamente relacionada à natureza das necessidades das aplicações que deverão ser implementadas. Poderá haver alguma restrição ou abertura de acesso, dependendo de vários fatores, tais como: o tipo de informação a ser compartilhada, qual o nível de visão que se deseja que os usuários tenham acesso, como se processarão os negócios etc. Assim, de acordo com o modelo de implantação da nuvem, pode haver restrição de acesso ao ambiente de computação em nuvem ou na utilização de determinados recursos nesse ambiente.

Em Puthal (2015), temos a classificação dos modelos de implantação de nuvem como sendo: Nuvem Privada; Nuvem Pública; Nuvem Comunitária; e Nuvem Híbrida que varia conforme a disponibilidade infraestrutura é disponibilizada e mantida.

— Nuvem Privada – É o modelo no qual toda a infraestrutura contratada é de uso exclusivo de um contratante. Consiste no modelo adequado para empresas que pretendem migrar sua estrutura computacional para a computação em nuvem, mas preferem manter todo o controle da

estrutura. De acordo com Chirigati (2009), uma nuvem privada é, em geral, construída sobre um *data center* privado.

— Nuvem Compartilhada – Neste modelo, o provedor compartilha os serviços contratados para um grupo de empresas normalmente com interesses em comum. Segundo Mell et al. (2011), este modelo pode ser administrado por organizações ou por um terceiro, e pode existir localmente ou remotamente. Modelo também conhecido como Comunidade.

— Nuvem Pública – É aquela executada por terceiros. Trata-se do modelo de nuvem em que os serviços contratados são ofertados para qualquer pessoa ou empresa que possa pagar, não existindo exclusividade quanto à oferta de serviço.

— Nuvem Híbrida – Conforme o próprio nome sugere, trata-se de uma combinação entre dois ou mais modelos de implantação de nuvens. Comumente, resulta da composição entre nuvens públicas e privadas. Segundo Chirigati (2009), o termo “computação em ondas” é, em geral, utilizado quando se refere às nuvens híbridas.

2.2 ELASTICIDADE

Como já foi referido anteriormente, o conceito de elasticidade está intimamente ligado à computação em nuvem, uma vez que se configura como a capacidade de contrair recursos computacionais de maneira dinâmica e em tempo real. Além da já mencionada definição adotada por Sharma et al. (2011), encontrou-se em Herbst, Kounev e Reussner (2013) o conceito de elasticidade como sendo o grau em que um sistema é capaz de adaptar-se às mudanças de carga de trabalho, provisionando e desprovisionando recursos de forma automática, de tal modo que em cada ponto no tempo, os recursos disponíveis correspondam à demanda atual, tanto quanto possível.

Righi e Galante (2016) afirmam que elasticidade consiste na capacidade de um sistema de modificar dinamicamente os recursos computacionais utilizados por uma aplicação. Ainda de acordo com esses mesmos autores:

> Além de abstrair detalhes de configuração da infraestrutura e de escalonamento, a elasticidade pode ser usada para encontrar os recursos mais apropriada para a execução de aplicações cujos requisitos não podem ser determinados antecipadamente com exatidão devido a mudanças nas cargas de trabalho, ou ainda, devido a mudanças na estrutura da aplicação (RIGHI e GALANTE, 2016, p. 1).

Para se compreender de forma mais ampla o conceito de elasticidade, é necessário fazer uma revisão do conceito de escalabilidade, visto que este último é um pré-requisito para podermos falar do primeiro. Um sistema só pode ser elástico se for escalável.

Escalabilidade diz respeito à habilidade que um sistema possui de suportar cargas crescentes de trabalho com o desempenho adequado, à medida que novos recursos computacionais forem sendo adicionados. Portanto, seria a capacidade de sustentação do aumento de cargas de trabalho em um sistema (HERBST, KOUNEV e REUSSNER, 2013). Para que isso ocorra, o sistema precisa ser elástico. Nesse caso, a elasticidade consiste na capacidade de um sistema de “esticar” e “encolher” em termos de recursos utilizados em função da demanda.

Uma relação equivocada entre os conceitos de escalabilidade e elasticidade no que tange à computação em nuvem se baseia no pensamento comumente difundido de que esta última, por suas características, oferece uma infraestrutura que permite a construção de sistemas escaláveis. No entanto, essa não é uma condição *sine qua non* da computação em nuvem. Para que um sistema seja escalável, ele precisa ser construído de forma a utilizar a capacidade que o ambiente de nuvem oferece de alocar e liberar recursos dinamicamente. Para tanto, pode haver a necessidade de um processo de adaptação específico — o que pode incluir uma etapa manual. Sem um processo de adaptação definido, um sistema escalável não pode se comportar de maneira elástica.

Na imagem que será apresentada a seguir (Figura 3), à esquerda, é possível verificar um exemplo de escalabilidade em que, em vermelho, observamos gargalo por falta de recurso. Tendo um sistema escalável, a solução para tal problema seria crescer a capacidade computacional até atingir um nível

acima do pico em vermelho que representa a falta de recurso. Essa solução se encontra demostrada em amarelo.

Na mesma figura, porém, como também é possível observar, após o crescimento, a capacidade computacional maior irá permanecer perene, embora a utilização, demostrada na figura em azul, seja totalmente variável. E conforme demostra a imagem, teríamos uma capacidade de atender a demanda mesmo em pico, mas para isso, teríamos uma ociosidade grande de recursos fora dos horários de pico. À direita da imagem, podemos observar ainda que a capacidade computacional acompanha a demanda, evitando desperdício e ociosidade, gerando assim economia e uma gestão mais consistente dos recursos.

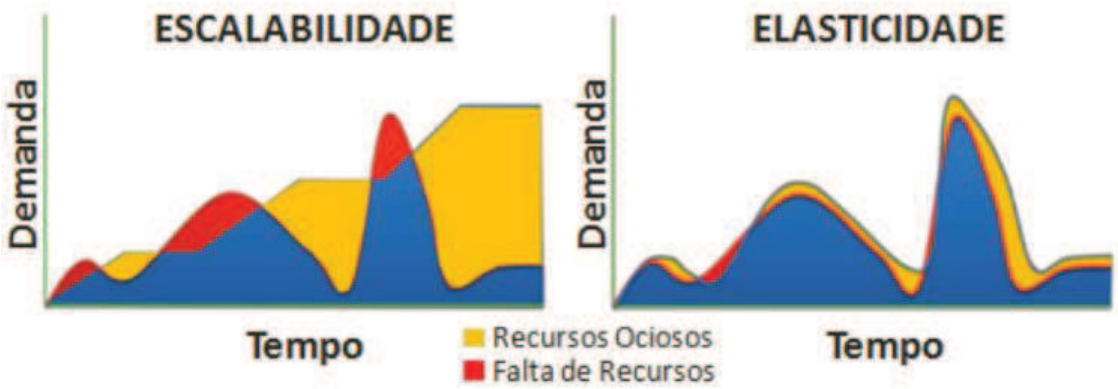


Figura 3- Escalabilidade versus Elasticidade
Fonte: Wendel, 2015.

Uma gestão dinâmica de recursos diminui em muito a ociosidade média dos servidores (em torno de 85%) e acelera a velocidade com que esses recursos são provisionados para seus usuários (TAURION, 2012).

Segundo Coutinho, a classificação do mecanismo de elasticidade pode ser verificada como é apresentada na Figura 4, a seguir.

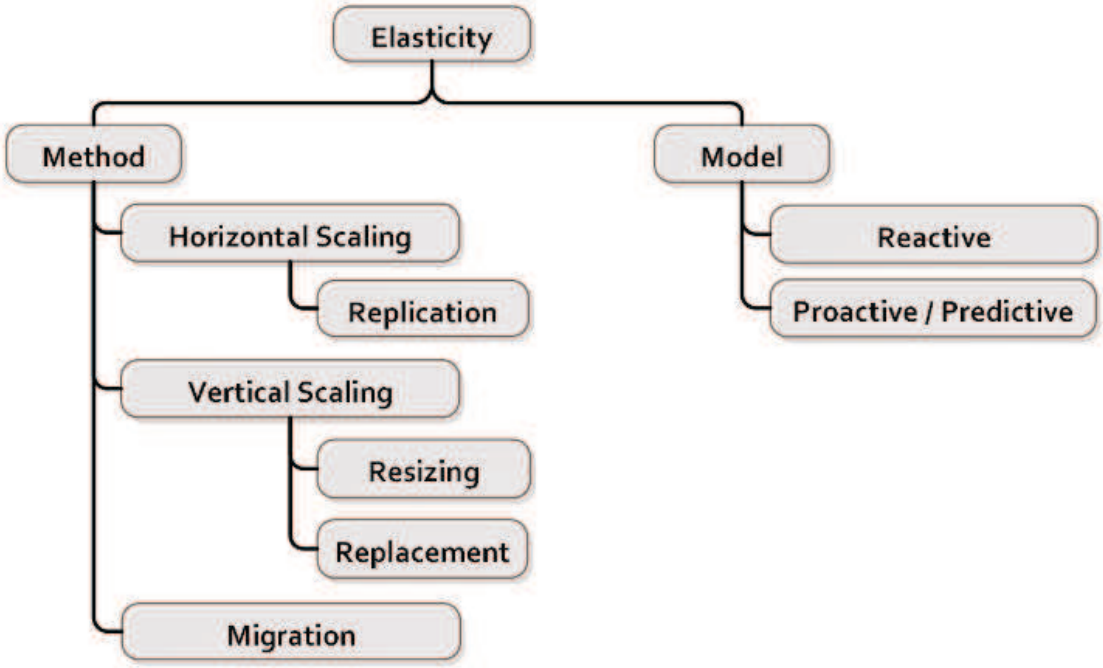


Figura 4- Classificação do mecanismo de elasticidade
Fonte: Ferreira, 2010.

2.2.1 Métodos

Conforme Galante e Bona, existem três métodos diferentes para oferecer a elasticidade: horizontal, vertical e migração, que serão apresentados a seguir.

Uma nuvem com elasticidade horizontal possui a capacidade de conectar e permitir a interação entre múltiplas máquinas virtuais (MVs). Entretanto, conforme Righi e Galante (2016), “uma nuvem com elasticidade horizontal possibilita apenas a adição ou a remoção dinâmica de MVs da plataforma de computação alocada pelo usuário” (p. 4). Ainda de acordo com Righi, na elasticidade horizontal, o intuito é adicionar ou remover nós computacionais (MVs) ou sua migração para novos nós de processamento. Atualmente, o método mais utilizado para oferecer a elasticidade horizontal é através da replicação de instâncias (GALANTE e BONA, 2012b).

Com relação à elasticidade vertical, Righi e Galante (2016) afirmam sua caracterização consiste na “possibilidade de se alterar a capacidade de MVs em execução (p. 4). Assim, na elasticidade vertical, o processo considerado é de adição ou remoção de recursos diretamente nas instâncias (MVs) que estão em utilização.

Ainda dentro da elasticidade vertical, podemos observar duas técnicas para operacionalizar essa elasticidade: redimensionamento ou reposição. Sendo a técnica mais utilizada atualmente, o redimensionamento altera os parâmetros de hardware (CPU, memória, disco, interfaces de rede) da máquina virtual. Essa solução é comum em sistemas operacionais baseados em Unix sistemas, uma vez que eles suportam *on-the-fly* (sem reiniciar) (COUTINHO et al., 2015, p. 20). Já na técnica de reposição, o recurso é substituído por outro com maior ou menor capacidade computacional, sendo que, neste caso, o servidor de máquinas virtuais pode ser substituído por outro, dependendo da necessidade (GALANTE e BONA, 2012b; COUTINHO et al., 2014).

Ainda é apresentada a técnica de migração que consiste em transferir uma máquina virtual ou um aplicativo que está sendo executado em um servidor físico para outro. Máquinas virtuais podem ser migradas de um servidor para outro para fins de consolidação ou balanceamento de carga. As aplicações também podem ser migradas (COUTINHO et al., 2015, p. 20).

2.2.2 Modelos

Segundo Rodrigues (2015, p. 29), os diversos métodos de elasticidade podem ser realizados de maneiras diferentes. Galante e Bona definem políticas que apontam dois modelos de elasticidade diferentes: manual e automático. Esses modelos são classificados conforme as interações necessárias dos usuários para a execução das operações de elasticidade (GALANTE e BONA, 2012b).

No modelo manual, o administrador da infraestrutura precisa tomar as decisões de elasticidade e executar as ações por intermédio do ferramental disponibilizado pelo sistema/provedor que está sendo utilizado. As formas mais comuns de ferramentas disponíveis são: interface gráfica; interface de linha de comando; API's (*Application Programming Interface).*

No modelo automático, existem ferramentas de monitoração do comportamento da nuvem que interagem com algoritmos de avaliação das métricas previamente definidas pelo administrador da infraestrutura. Assim, ações de elasticidades são disparadas. Portanto, embora seja automático, ainda

existe a necessidade de interação na definição das métricas e algoritmos a serem utilizados para a tomada de decisão (GALANTE e BONA, 2012b).

Ainda quanto às ações do modelo automático, temos uma divisão em ações reativas e proativas, que são definidas pelo tipo de algoritmo empregado. Na forma reativa, as métricas coletadas são comparadas com valores mínimos e máximos predefinidos, e caso os valores coletados não fiquem dentro do esperado, ações de alocação de recurso ou deslocação de recurso são disparadas. Na forma proativa, ainda se utilizam os dados coletados, porém eles são utilizados como fonte de dados para uso de previsões de demandas futuras. Técnicas de previsão de carga dos recursos são comumente utilizadas para a reorganização proativa dos recursos (GALANTE e BONA, 2012b; COUTINHO et al., 2014).

2.3 SÉRIES TEMPORAIS

De acordo com Latorre e Cardoso (2001), série temporal é um conjunto de observações realizadas de forma sequencial ao longo de um período definido, tendo como principal característica a dependência entre os dados vizinhos. Essa dependência é o foco principal da análise e modelagem da série de dados onde a ordem das observações é relevante e crucial.

Ainda em Latorre e Cardoso (2001), uma série temporal é definida como uma sequência de dados impetrados em instantes de tempo regulares com uma duração definida, sendo também denominados série histórica. As autoras acrescentam que em análises de série temporal, o objetivo primário é modelar o fenômeno a ser estudado e, a partir daí, conseguir descrever o comportamento em série, fazendo estimativas e avaliando quais comportamentos foram influenciados para o comportamento da série. Busca-se, com isso, definir quais são as relações de causa e efeito entre um conjunto de séries, tudo baseado em um completo conjunto de técnicas estatísticas que varia com base num modelo definido.

Ehlers (2012) menciona como principal característica das séries temporais a importância da ordem cronológica das observações, uma vez que observações vizinhas são dependentes uma das outras, diferentemente de modelos de regressão. Nestes modelos, a ordem cronológica não importa.

Segundo Dias (2015, p. 31), dados provindos de séries temporais surgem de diversos campos do conhecimento, tais como: economia (bolsa de valores), indústria (controle de processo de produção), meteorologia (temperatura diária), entre outros.

Com relação à análise de tendência, temos dois métodos mais amplamente utilizados: no primeiro caso, temos uma análise comportamental da série em torno de um ponto estimando a tendência, e neste ponto que o algoritmo do VertElastic se embasa para estimar a tendência de consumo no ponto futuro com base nos dados coletados pelos logs das aplicações ao longo de sua execução; a outra apoia-se em um ajuste de uma função polinomial de tempo.

2.3.1 Modelos de regressão polinomial

Nestes modelos, temos os valores da série expressos como valores dependentes (Y) e os tempos dos estudos com valores independentes (X). Faz-se inicialmente a dispersão Zt (Y) em relação ao tempo, buscando identificar uma função adequada ao processo, podendo ser ela uma parábola, exponencial, linear, e assim por diante.

O uso da variável período-centralizada (subtraído o ponto médio da série histórica), a partir da transformação da variável do período, é recomendado para evitar, assim, a correlação serial entre os termos da equação de regressão.

2.3.2 Sazonalidade

Em Latorre e Cardoso (2001), a sazonalidade é definida como “um fenômeno [...] como aquele que ocorre regularmente em períodos fixos de tempo”, sendo de difícil estimativa, visto que depende da interpelação de questões estatísticas com questões conceituais do fenômeno estudado. As autoras (2001) explicam que na sazonalidade dita determinística, podem ser utilizados modelos de regressão que incorporem funções do tipo seno ou cosseno à variável tempo.

Para se descobrir se existe sazonalidade na série de valores e verificar

qual o seu ritmo, Latorre e Cardoso recomendam realizar uma análise espectral. Com isso, torna-se possível determinar o padrão sazonal. A média móvel centrada no período de repetição ou a diferença entre a série original e o polinômio estimado podem ser utilizadas para retirar o efeito sazonal.

2.3.3 Modelos auto regressivos

Quando não temos ocorrências aleatórias em meio a uma média constante, temos uma série estacionária. Este modelo se configura no mais simples de uma série histórica estacionária, assim definida por Latorre e Cardoso (2001):

> Esta série é consequência da variação aleatória do ruído branco ao redor de uma grande média, ao longo do tempo. Ela é escrita como a combinação aleatória dos valores anteriores da Z_t ($Z_t = b_1 Z_{t-1} + b_2 Z_{t-2} + ... + b_p Z_{t-p}$) e, por isso, a série toda pode ser função do ruído branco. Essa classe de modelos é conhecida como modelos auto-regressivos-AR (LATORRE e CARDOSO, 2001, p. 150).

O modelo ARMA é a incorporação do processo de diferenças com um modelo estacionário homogêneo. Já o ARIMA é a inclusão integrada de médias móveis a um modelo auto regressivo), sendo *d* a diferença necessária para remover a tendência da série.

Latorre e Cardoso (2001) explicam que:

> Há duas situações em que a série pode ser considerada não estacionária: 1) quando durante um período os pontos oscilam ao redor de uma média e, depois, mudam de patamar (neste caso, basta tomar uma diferença da série); e 2) quando a série é não estacionária em relação à tendência (geralmente, para torná-las estacionárias, é necessário tomar a segunda diferença). Os modelos ARIMA podem dar conta da sazonalidade quando há *lags* de baixa ordem (LATORRE e CARDOSO, 2001, p. 150).

Múltiplos períodos de sazonalidade precisam ser tratados em um modelo que utilize a sazonalidade estocástica. Neste caso, o modelo utilizado dever ser o SARIMA que consiste na utilização de funções trigonométricas em um modelo ARIMA.

2.3.4 Modelos lineares generalizados

Este modelo utiliza a dependência ou a resposta da variável (Y) como contador e as demais variáveis independentes são candidatas a explicar o comportamento ao longo do tempo.

2.4 AUTOELASTIC

De acordo com Rodrigues (2016, p. 53), AutoElastic é um *middleware* que opera no nível de PaaS, permitindo a aplicações paralelas não elásticas tirarem vantagem da elasticidade em nuvem sem nenhuma alteração em seu código fonte. O *middleware* trabalha provendo elasticidade horizontal, adicionando máquinas virtuais de forma reativa, ou seja, baseado em parâmetros mínimos e máximos previamente definidos (chamados de *thresholds*), utilizados para confrontar os níveis de serviços acordados SLA (Service Level Agreement) e, assim, definir se existe algum parâmetro fora do esperado.

Dois conceitos devem ser definidos dentro do escopo do AutoElastic: nuvem AutoElastic e elasticidade assíncrona.

Nuvem AutoElastic foi definida por Rodrigues (2016) como:

> [...] uma nuvem modelada com *f* recursos computacionais homogêneos e distribuídos, em que no mínimo um deles (Nó 0) está sempre ativo. Essa máquina física é encarregada de executar uma máquina virtual com o processo mestre e outras *n* máquinas virtuais com um processo escravo cada, em que *n* representa o número de núcleos de processamento dentro de um nó em particular. O grão de elasticidade para cada ação se refere a uma simples máquina física. Por fim, a qualquer momento, o número de máquinas virtuais executando processos escravos é igual a $v = n \times f$ (RODRIGUES, 2016, p. 54).

Por sua vez, Elasticidade assíncrona foi definida da seguinte forma:

> [...] é uma maneira de assincronamente notificar uma aplicação que está executando em nuvem sobre mudanças na disponibilidade de recursos do ambiente, tais como o número de instâncias de máquinas virtuais. Por exemplo, a aplicação é

> notificada assim que uma nova instância de uma máquina virtual está disponível no ambiente, sem prejudicar seu fluxo de execução normal (RIGHI et al., 2016 apud RODRIGUES, 2016, p. 56).

No AutoElastic, as ações de elasticidade sempre acontecem com a adição ou subtração de maquinas virtuais de forma horizontal. Rodrigues (2016, p. 58) demostra como a elasticidade horizontal assíncrona é viabilizada (Fig. 5):

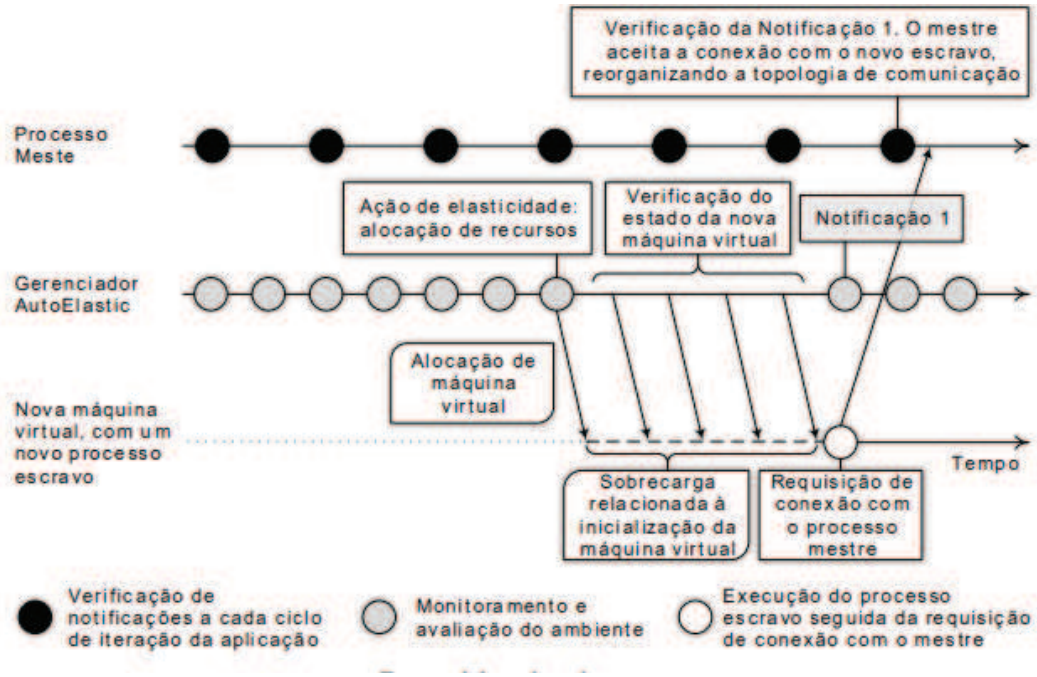


Figura 5 - Funcionamento do processo mestre, um novo processo escravo e o Gerenciador
Fonte: Rodrigues, 2016

2.5 CONSIDERAÇÕES PARCIAIS

Este capítulo abordou os principais conceitos utilizados neste estudo, apresentando as definições, propriedades e características de computação em nuvem, elasticidade e AutoElastic para facilitar a contextualização em que se deu esta pesquisa. Como a proposta consistiu em trabalhar com métodos proativos, foi abordado o tema de séries temporais, visto que se trata de previsão de valores.

3 TRABALHOS RELACIONADOS

As iniciativas que impulsionam a pesquisa acadêmica possuem, entre diversas motivações, a busca por preencher lacunas deixadas em investigações já realizadas ou aprofundar abordagens já existentes a respeito da temática adotada pelo pesquisador. No caso particular deste estudo, pretendeu-se fazer um levantamento acerca do conceito de elasticidade vertical e sua relação com a computação em nuvem. Para tanto, foi desenvolvida uma pesquisa bibliográfica em bancos de teses e dissertações e em sites de revistas científicas e artigos acadêmicos. Foram utilizados os seguintes descritores: elasticidade vertical; elasticidade horizontal; computação em nuvem. Nesse sentido, apresento alguns trabalhos selecionados para análise.

Roloff et al. (2017) fazem uma extensa avaliação das principais nuvens públicas comerciais, partindo para uma avaliação aprofundada do desempenho e da eficiência de custos de aplicações paralelas na nuvem. Com os estudos desenvolvidos, os autores obtiveram resultados que mostram que a degradação ocorrida devido à virtualização e outras despesas gerais da nuvem é insignificante. No entanto, concluíram que a interconexão de rede na nuvem ainda continua a ser um grande estrangulamento para o desempenho da aplicação HPC.

Wu et al. (2016) apresentam o HybridScaler, um método de escala automático preciso e econômico, que combina escalabilidade horizontal preditiva de longo prazo e oportuna Escala vertical reativa. O referido método é baseado em um *resource-pressure* que afirma conseguir oferecer uma quantidade adequada de recursos que correspondem à mudança de carga de trabalho.

Os resultados obtidos pelos supracitados autores foram avaliados quanto à eficácia e eficiência, comparando-se com métodos de dimensionamento automático. De acordo com os resultados apontados, o método de escalabilidade horizontal preditiva diminuiu o tempo médio de resposta em torno de 16-39% e a taxa de violação SLO entre 34-50%, tomando como ponto de partida os limites estáticos e o esquema baseado em previsão.

Sortiríeis et al. (2016), em seu trabalho, apresentam os resultados de uma pesquisa utilizando a elasticidade inter-nuvem que se concentra no balanceamento de carga da nuvem, com base em reconfiguração da máquina

virtual dinâmica, quando variações em carga ou no volume de pedidos do usuário são observados. Foi desenvolvido um sistema de reconfiguração dinâmica, denominado balanceador de carga entre nuvens (ICLB), que demostrou ser capaz de expandir ou diminuir os recursos virtuais de forma automatizada, eliminando assim o tempo de indisponibilidade e possíveis falhas de comunicação. As reconfigurações foram dinâmicas e em tempo real, baseadas em requisições de um servidor WEB.

Farokhi et al. (2015), em seu artigo, descrevem o desenvolvimento de um controlador de elasticidade vertical para aplicativos baseados em nuvem. Para isso, usaram a teoria de controle principal para garantir requisitos de desempenho de alocação de memória com um botão de controle.

A novidade do artigo dos autores mencionados acima reside na aplicação de uma técnica de síntese de controlador para garantir a robustez e a estabilidade do sistema controlado, sendo o tempo de resposta do aplicativo um critério de tomada de decisão. Os resultados experimentais que obtiveram com seus estudos revelam que o controlador pode economizar pelo menos 47% de uso de memória, mantendo assim uma experiência aceitável para o usuário.

O Elastack (2012) consiste em um sistema adaptativo genérico desenvolvido para trabalhar em conjunto com as principais estruturas de IaaS, promovendo elasticidade horizontal de forma automática. Dessa forma, possibilita a qualquer provedor de serviço nas nuvens implementar a monitoração que serve de base para as adaptações automatizadas.

Os trabalhos elencados acima fazem parte de uma seleção de textos acadêmicos e artigos científicos encontrados nos principais repositórios e periódicos sobre o tema, tais como *IEEE.or e ACM*, por exemplo. Esses trabalhos foram aqui apresentados em virtude de possuírem relação com os principais aspectos da pesquisa realizada, além de terem sido publicados há menos de três anos — o que implica dizer que se trata de estudos recentes, alguns ainda em fase experimental. Além deles, outros trabalhos relevantes também foram encontrados e devidamente lidos, contribuindo para embasamento teórico a respeito do tema.

Uma análise comparativa entre os trabalhos relacionados foi descrita na Tabela 1 (a seguir). Nela, percebe-se que embora tenham semelhanças com o objeto desta investigação por também trabalharem com elasticidade, nenhuma

das pesquisas elencadas aborda diretamente o tema deste estudo, a possibilidade de escolha entre elasticidade vertical ou horizontal de forma automática ou pré-definida.

Roloff et al. apresentam trabalhos em que a elasticidade acontece apenas de forma horizontal, de forma automática ou manual, o que diferencia o VertElastic, que além de oferecer elasticidade horizontal como os trabalhos citados se diferencia por oferecer a possibilidade da elasticidade vertical. Já sortiríeis et al., Farokhi et al. e o Elastack oferecem apenas a possibilidade de elasticidade vertical de forma automática ou com threshold pré-definidos, mas nenhum trabalha com elasticidade horizontal. O Wu et al. é o que se aproxima mais do VertElastic, pois trata as duas formas de elasticidade, porém não é possível pré-configurar threshold opção contemplada no VertElastic, sendo assim um incremento deste trabalho que o diferencia de todos trabalhos abordados.

Indo mais além, é necessário afirmar que a abordagem da elasticidade vertical para aplicações de alto desempenho não é tão utilizada em pesquisas com computação em nuvem, onde os principais trabalhos tendem a abordam preferencialmente o conceito de elasticidade horizontal, sendo esta uma das lacunas identificadas.

Nesse sentido, o modelo proposto neste estudo diferencia-se principalmente por possuir um controlador que oferece a possibilidade de efetuar tanto a elasticidade horizontal quanto a vertical de forma automática, e também prefixando indicadores de performance que devem ser respeitados. O modulo de decisão do VertElastic, será abordada mais detalhadamente no capítulo 4, buscas justamente se encaixar nesta lacuna.

Tabela 1 - Trabalhos relacionados ao tema Elasticidade

Paper	Tipo de elasticidade	Localização	Mecanismo de controle	Ação de elasticidade	Oferta de recursos
Roloff et al.	Horizontal	Infraestrutura	Manual	Elasticidade Horizontal	Fixa
Wu et al.	Horizontal e Vertical	Infraestrutura	Automático	Elasticidade Horizontal e Vertical	Fixa e dinâmica
Sortiríeis et al.	Vertical	Infraestrutura	Automático	Elasticidade Vertical	Fixa

Fonte: Autoria própria

4 MÓDULO DE DECISÃO PARA EXPLORAÇÃO DA ELASTICIDADE VERTICAL NO AUTOELASTIC

Neste capítulo, apresento o conceito de VertElastic para aplicar elasticidade vertical nos moldes do modelo AutoElastic que foca em elasticidade na nuvem para aplicações de alto desempenho, utilizando elasticidade horizontal.

As próximas seções são organizadas da seguinte forma: Na Seção 4.1 será apresentada a decisão do projeto. Na Seção 4.2, discorre-se sobre a arquitetura do modelo do VertElastic definindo, assim, sua forma de trabalhar para a implementação de elasticidade vertical. Por fim, na Seção 4.3 serão apresentados os mecanismos de elasticidade que definirão a forma com que o modelo deverá alocar recurso computacional.

4.1 DECISÃO DO PROJETO

A elasticidade é uma das características mais proeminentes quando se está trabalhando em computação em nuvem. A possibilidade de a aplicação dinamicamente adquirir ou liberar recurso computacional permite uma nova visão sobre a alocação de recursos e a demanda necessitada por uma aplicação.

Do ponto de vista do usuário, o ideal da aplicação é que o ciclo computacional desejado seja finalizado no menor tempo possível e com o recurso de hardware otimizado ao máximo. Entretanto, a estimativa de recurso

necessário para uma execução mais otimizada é um tema cada vez mais estudado, visto que não é uma tarefa simples de ser solucionada.

Na abordagem tradicional dos principais fornecedores de PAAS — como é o caso do Windows Azure e Amazon WS —, torna-se necessária a intervenção do usuário para que seja configurado previamente um conjunto de regras. E com base nestas regras, é preciso que sejam tomadas ações de elasticidade, sendo que tais ações também consistem numa pré-configuração no ambiente, conforme demostrado na Figura 6.

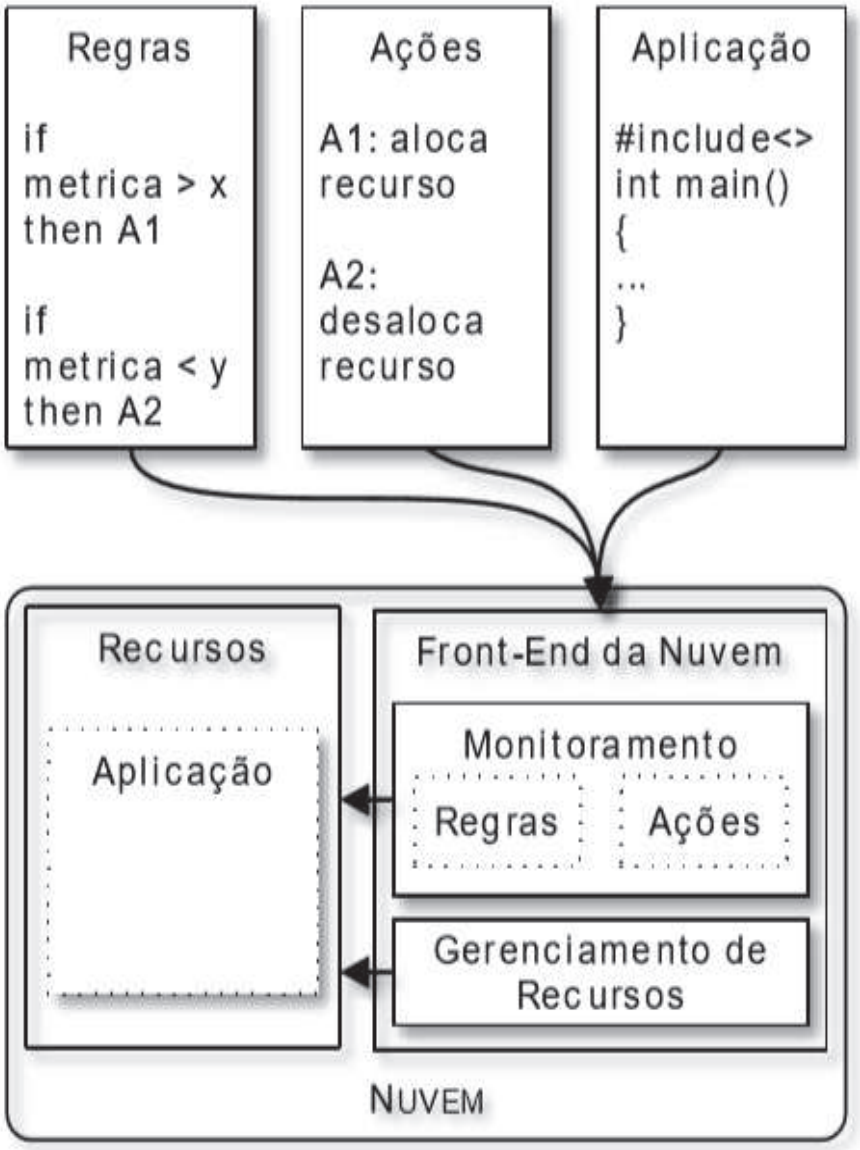


Figura 6 - Abordagem padrão de elasticidade
Fonte: Righi et al., 2015.

De acordo com Righi et al. (2015, p. 47), o AutoElastic fornece elasticidade horizontal e reativa de forma transparente para aplicações paralelas, sem a necessidade de intervenção do usuário para a definição de regras e ações ou modificações no código fonte da aplicação.

Como foi abordado anteriormente, no AutoElastic, a ação de elasticidade se dá com o aumento ou a diminuição da quantidade de máquinas virtuais, conforme necessidade da aplicação, visando com isso garantir que sua execução permaneça dentro de um tempo esperado. A Figura 7, a seguir, demostra a ideia abordada pelo AutoElastic.

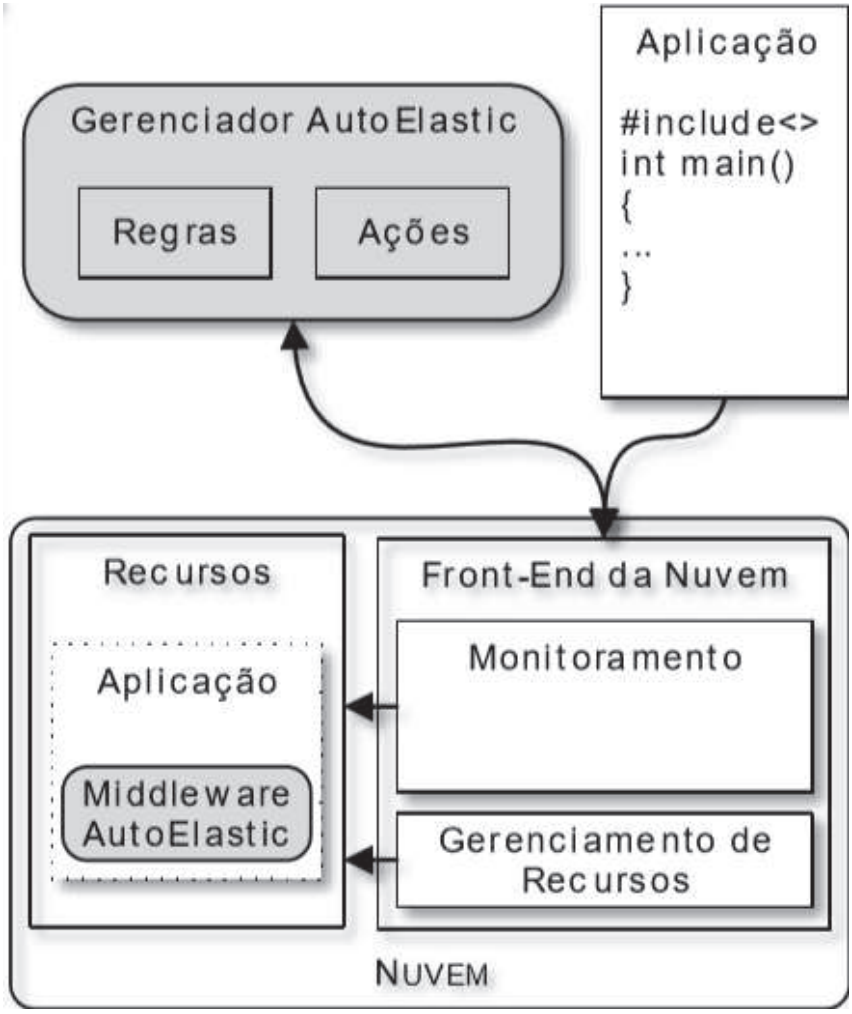


Figura 7- Elasticidade do modelo AutoElastic.
Fonte: Rodrigues, 2016

A abordagem aqui proposta consisti incrementar ao modelo do AutoElastic um módulo decisório onde após executar uma série de testes para saber se existe a disponibilidade de hardware necessária para que a elasticidade vertical seja executada, deve-se levar em consideração que a capacidade total

de computação do nó hospedeiro, ela é uma restrição para a implementação de elasticidade vertical, por isto o VertElastic antes de executar a elasticidade vertical verifica a disponibilidade de hardware e caso seja insuficiente o modulo de decisão entende a limitação e permite que o fluxo de elasticidade horizontal seja iniciado. Sendo possível, o modulo inicia a execução da elasticidade vertical a fim de que a quantidade de recurso necessária para que uma aplicação finalize seu ciclo de execução dentro de um SLA esperado, seja alcançada com a utilização da elasticidade vertical, sempre que o nó computacional permitir. Na Figura 8, apresento a interação do módulo do VertElastic com o gerenciamento e *middleware* do AutoElastic.

Segundo Younge et al. (apud RIGHI, 2013, p. 5), essa estratégia é pertinente para a economia de energia elétrica. Tais autores mostram em gráficos que a alocação de uma máquina virtual no mesmo nó de computação é menos custosa em termos de potência (Watts) do que alocar um novo nó e ali lançar uma máquina virtual.

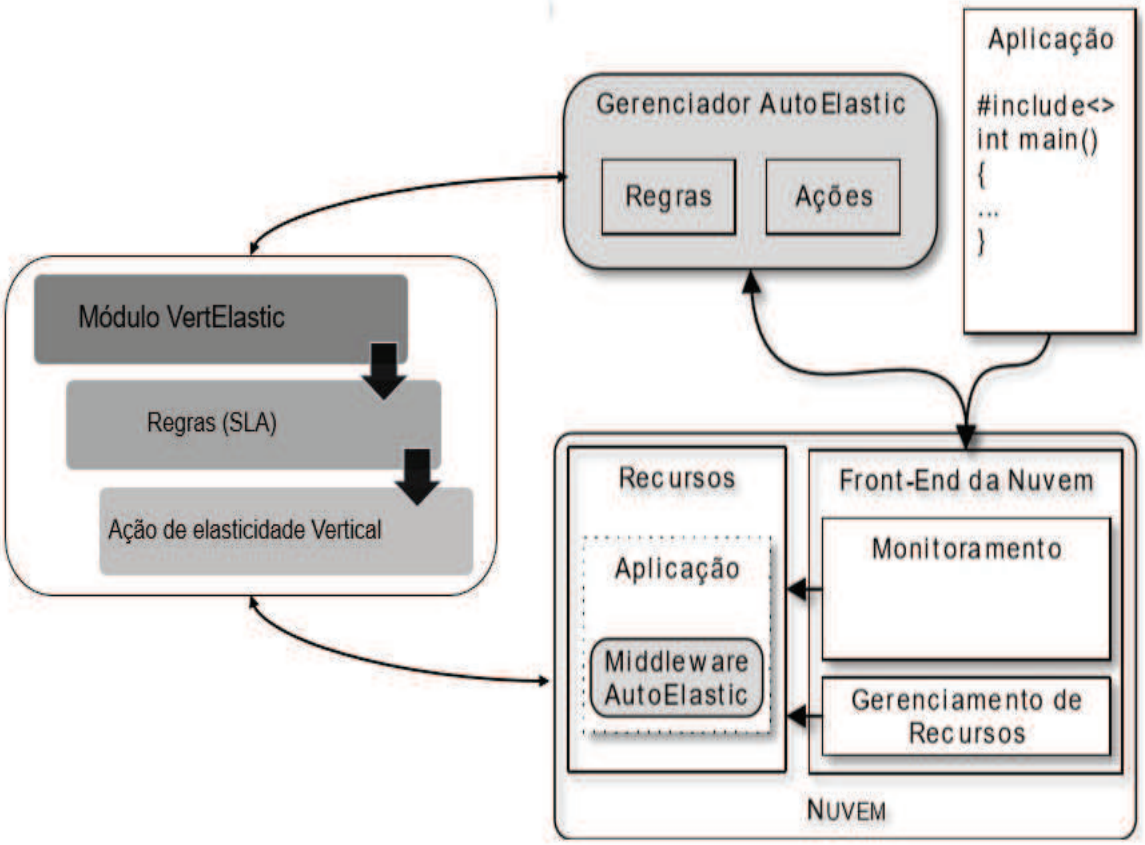


Figura 8 -Interação do Módulo VertElastic com o AutoElastic.
Fonte: Adaptado de Rodrigues (2016)

No modelo proposto a lacuna de apenas elasticidade horizontal ser comtemplada pelo modelo do AutoElastic é contornada, conseguindo assim transpor a barreira de crescer apenas em número de nós, como faz a elasticidade horizontal mas também explorar a cada vez maior capacidade de processamento e quantidade de memória, extraindo assim com elasticidade vertical todo a capacidade computacional das máquinas e diminuir a ociosidade dos nós, além disso propõe uma resposta à questão central da pesquisa — como se configura um modelo de elasticidade para aplicações de alto desempenho baseado no AutoElastic que possibilite a implementação de elasticidade vertical, permanecendo totalmente transparente a necessidade da reescrita de código? —, busquei verificar alguns desdobramentos decorrentes dessa questão, tais como:

É eficiente prover elasticidade nas nuvens para aplicações HPC baseada em elasticidade vertical, tendo em vista a possibilidade de limitação de hardware e da literatura indicar normalmente a utilização de elasticidade horizontal para aplicações de HPC; Verificar a viabilidade manter o AutoElastic e o VertElastic provendo elasticidade em nuvem para aplicações HPC sem a necessidade de interação no código fonte da aplicação; existem realmente ganho de performance no uso de elasticidade vertical em plataformas que não permitem auto *scalling* em tempo de execução e se as aplicações HPC que se beneficiam com elasticidade na nuvem promovida pelo AutoElastic terão um melhor desempenho com o módulo que implementa elasticidade vertical justificando a utilização do VertElastic.

4.2 ARQUITETURA

O VertElastic apresenta um protótipo de um modelo baseado em fila M/M/m, tomando como fundamento na teoria de filas (PENTA et al., 2001). Partindo da arquitetura apresentada pelo AutoElastic, busca-se adaptar o protótipo desenvolvido para que a elasticidade aconteça de forma vertical, prioritariamente e dentro das possibilidades dos recursos computacionais disponíveis. Em outras palavras, procura-se expandir ou reduzir o recurso das máquinas virtuais conforme a necessidade, de maneira proativa, usando formas de predição, sempre respeitando os níveis de serviços pré-definidos.

Sendo assim, é possível tentar antever a necessidade de recursos necessários para efetuar a tarefa definida para o próximo período de tempo, sempre buscando a otimização do recurso e a melhor performance possível, encontrando um ponto de equilíbrio entre o recurso necessário para a execução da tarefa e o mínimo custo e desperdício de recurso computacional.

Atualmente, a arquitetura do framework do AutoElastic é baseada em um gerenciador da nuvem AutoElastic que é responsável pela monitoração dos recursos e também por iniciar as ações de elasticidades quando necessário, sendo essas ações operadas simultaneamente. Como esse modelo foi projetado para trabalhar de forma reativa, os *thresholds* contendo a quantidade mínima e máxima de máquinas virtuais podem ser informados. Ou, na falta dessa informação, será considerada a quantidade de máquinas no início da aplicação (RODRIGUES, 2015).

O VertElastic explora a nuvem do AutoElastic, baseando-se em sua arquitetura. Implementa-se a elasticidade vertical tomando como ponto de partida a elasticidade horizontal, utilizando a técnica de replicação que consiste em alocar ou remover recursos das máquinas virtuais por intermédio de uma imagem ou um *template* (RIGHI, 2016). Em conjunto com a técnica de replicação, utiliza-se também a técnica de redimensionamento, uma vez que a nova máquina virtual disponível — que foi criada a partir do evento de elasticidade —, embora siga as principais definições do *template* original, deverá sofrer uma alteração em sua configuração, podendo ocasionar o acréscimo ou a diminuição de alguns de seus parâmetros, sendo os mais comuns a CPU e a memória.

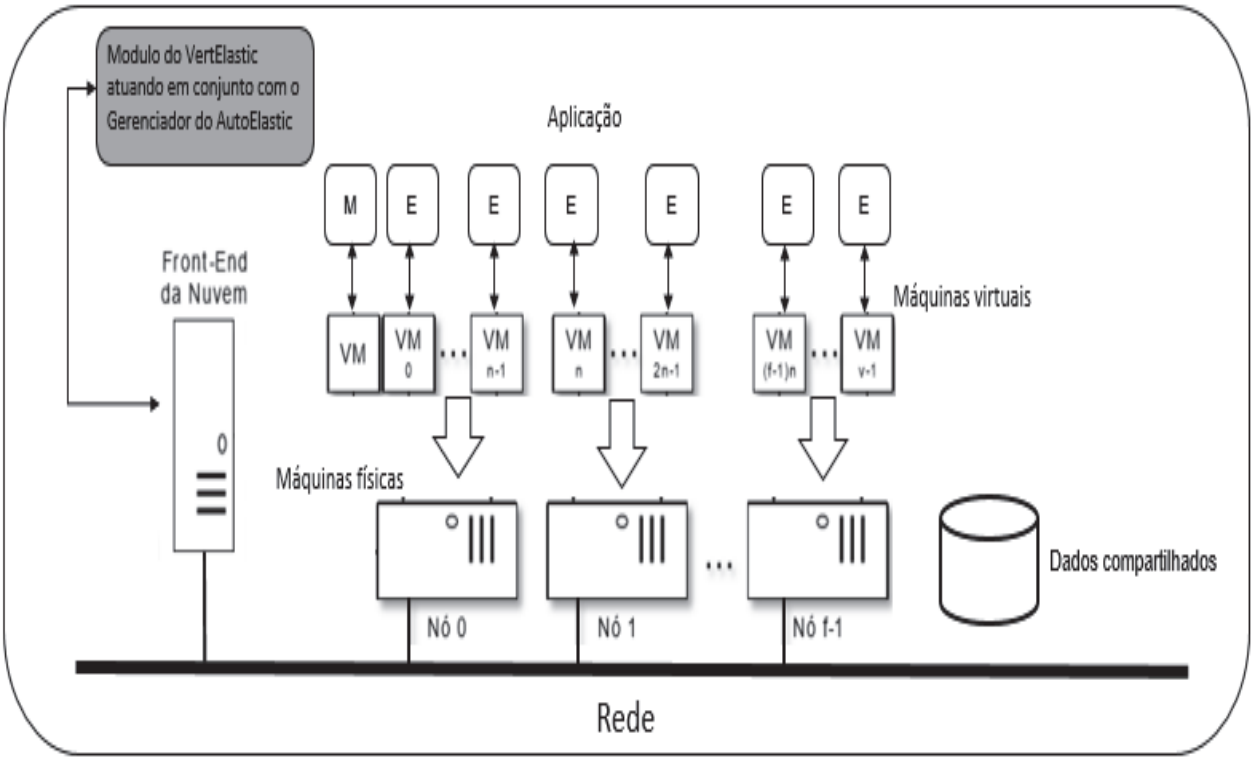


Figura 9 - Arquitetura do VertElastic.
Fonte: baseado em Righi et al., 2015.

O VertElastic deve trabalhar da seguinte maneira: uma máquina virtual A em que uma aplicação com determinado QoS está sendo executada, para atender a essa necessidade da aplicação, executa uma ação de elasticidade. Essa ação de elasticidade cria uma nova máquina virtual (máquina virtual B) com mais recurso. Esse é um processo que demanda tempo (inicialização de uma nova máquina), porém a máquina virtual A continua computando, enquanto a nova máquina virtual (B) está em processo de inicialização. Quando ambas as máquinas virtuais estiverem prontas, a nova máquina virtual (B), criada a partir da ação de elasticidade da máquina virtual original (A), assumirá a computação da aplicação.

No momento em que se inicia a ação de elasticidade, são gerados logs da máquina A. Assim que a nova máquina estiver disponível, é feito um comparativo entre ambas (máquinas A e B), sendo esta diferença adicionada à máquina B que assume o lugar da primeira máquina, já com a nova especificação de hardware (CPU e Memória). Técnicas de *live migration* de máquinas virtuais podem ser implementadas para auxiliar nessa etapa. O AutoElastic já propicia isso, utilizando um *loop* em que conecta e desconecta

máquinas virtuais, segundo Rodrigues (2015, p. 54-55). Durante esse *loop*, é feita a troca das máquinas, sem que seja afetada a aplicação.

De fato, aplicações baseadas em *loops* são convenientes para a implementação de elasticidade em nuvem devido ao fato de haver uma fácil reconfiguração da quantidade de recursos no início de cada interação, sem alterar a semântica da aplicação (SPINNER et al., 2014). Além disso, o *loop* de distribuição garante o estado de consistência global da aplicação.

Outro importante fator que possibilita a elasticidade no VertElastic é o compartilhamento de uma área de memória utilizada pela nuvem do AutoElastic, área esta que possibilita a interação entre as máquinas virtuais e o módulo de gerenciamento de elasticidade. A área de compartilhamento serve para a interação entre as máquinas virtuais. O tema relacionado ao uso de uma área compartilhada tem sido reiteradamente abordado por autores que falam sobre computação em nuvem privadas. Nesta pesquisa, entrou-se em contato com os trabalhos de vários autores (CAI et al.; MILOJICIC; LLORENTE; MONTERO, 2011; WEN et al., 2012).

A área privada de compartilhamento de memória entre as máquinas virtuais e o gerenciador de AutoElastic disponibiliza três tipos de notificações: (1) informação feita pelo gerenciador AutoElastic de que novos recursos foram disponibilizados para aplicação; (2) comunicação feita pelo mestre sobre os processos a serem consolidados pelos escravos; e (3) notificação de consolidação de novos recursos, feita também pelo mestre.

O papel dessa área compartilhada é primordial para se manter um estado global consistente (RODRIGUES, 2015). É nessa área que o VertElastic consulta as notificações novas que estão sendo gravadas pelo gerenciador de AutoElastic e pelo processo mestre. É nela também que o VertElastic consulta a notificação de que a máquina escrava A finalizou sua demanda de processamento e que no próximo *loop*, a máquina escrava B está apta para assumir seu lugar.

Como foi trocada a máquina virtual no processo de elasticidade, devemos garantir que na quarta notificação, o novo IP da nova máquina virtual seja informado.

A elasticidade vertical para HPC é recomendada por Spinner et al. (2014), pelo fato de ocorrer uma grande carga nas operações quando temos uma reorganização de recursos. As ações de elasticidade ocorrem em consonância

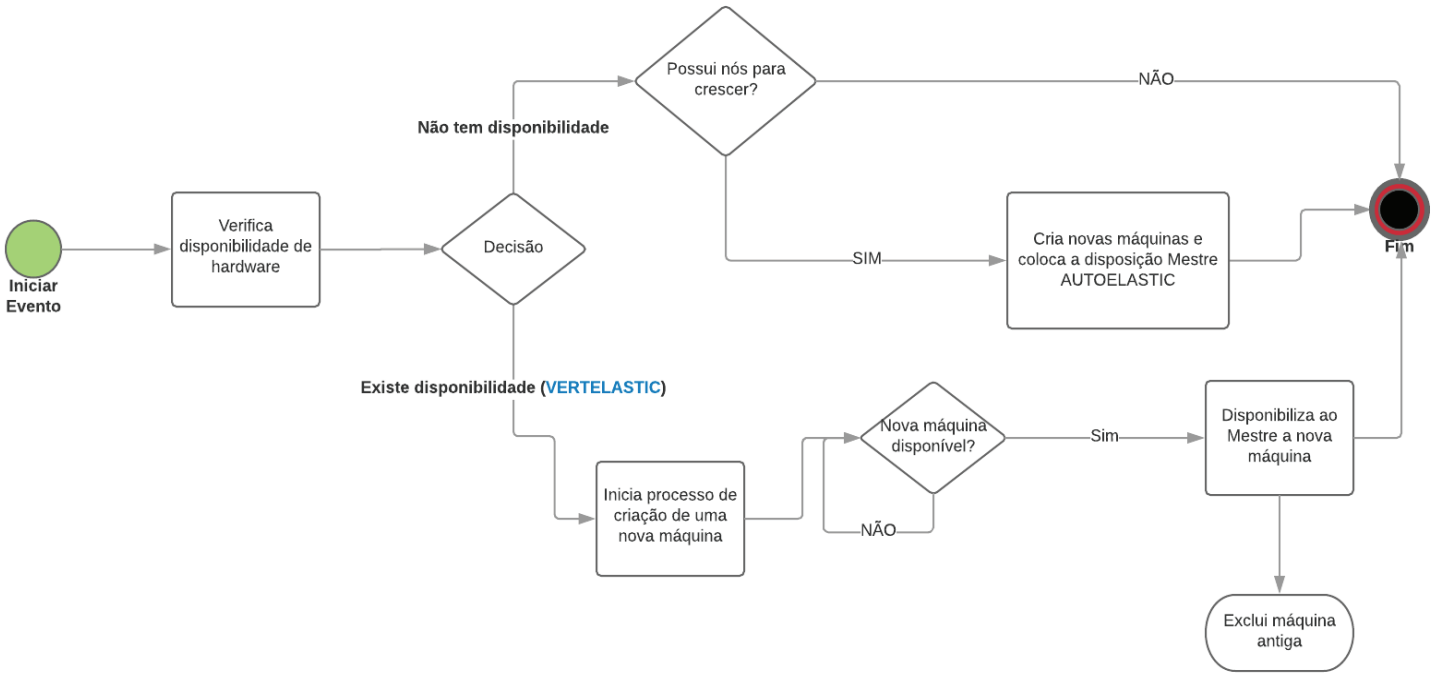
com a execução da aplicação. No caso do VertElastic, embora seja uma ação de elasticidade vertical — o que pressupõe apenas a reconfiguração de uma máquina existente, e não a criação de uma nova máquina —, é necessário instanciar uma nova máquina reconfigurada, pois não é possível, na plataforma OpenNebula, a alteração de parâmetros CPU e memória com a máquina ligada.

Esse processo de criação de uma nova máquina demanda tempo para que a nova MV esteja disponível (tempo de *boot*), visto que o sistema não permite, durante o tempo de execução, o redimensionamento da MV. O tempo de preparo necessário para deixar a nova MV já redimensionada e apta para consumir as notificações encaminhadas pelo mestre é uma questão conhecida. Busca-se superar essa limitação, sem afetar a aplicação, utilizando elasticidade assíncrona — que, em conjunto com a área de compartilhamento de memória, possibilita a execução adequada da aplicação.

> Elasticidade Assíncrona é uma maneira de assincronamente notificar uma aplicação que está executando em nuvem sobre mudanças na disponibilidade de recursos do ambiente, tais como o número de instâncias de máquinas virtuais. Por exemplo, a aplicação é notificada assim que uma nova instância de uma máquina virtual está disponível no ambiente, sem prejudicar seu fluxo de execução normal (RIGHI et al., 2015).

4.3 Mecanismo De Elasticidade

A quantidade de recursos que deve ser alocada configura-se num ponto crucial para o bom desempenho das aplicações. Um dos grandes problemas a ser solucionado consiste em encontrar um número mínimo, porém satisfatório, de servidores para atender a demanda esperada no menor tempo possível.



.

Figura 10 – Mecanismo de elasticidade do VertElastic.
Fonte: elaborado pelo autor

Messias (2016) propôs um algoritmo de *autoscaling* que foi estudado antes da implementação do algoritmo do VertElastic. Segundo esse autor (2016, p. 45), na equação 1.1, ρ é a utilização do sistema, λ é a taxa de chegada de requisições, μ é a taxa de processamento do servidor e *m* é o número de servidores.

$$\rho = \frac{\lambda}{m\mu} \tag{1.1}$$

Para o sistema ser estável, ρ precisa ser menor do que 1 (GROSS et al., 2013). Nosso objetivo é encontrar o menor valor para *m* que mantenha o sistema estável.

$$m = \lceil \frac{\lambda}{\rho\mu} \rceil \tag{1.2}$$

Escrevendo a equação 1.1 em função de *m*, temos a equação 1.2, apresentada logo acima.

Messias (2016, p. 2) explica ainda que para encontrar um valor adequado para ρ, é necessário considerar o valor de μ e o tempo de resposta de uma requisição no servidor. O tempo de resposta do servidor pode ser calculado como mostra a equação 1.3 (GROSS et al., 2013):

$$R = \frac{\frac{1}{\mu}}{1 - \rho} \tag{1.3}$$

Colocando a equação 1.4 em função de ρ, obtemos a seguinte equação:

$$\rho = 1 - \frac{1}{R\mu} \tag{1.4}$$

Juntando as equações 1.3 e 1.4, obtemos a equação 1.5 que representa o número de servidores, *m*, cada um capaz de processar μ requisições por segundo, necessárias para atender uma taxa de chegada de λ requisições por segundo, com tempo de resposta máximo de *R* segundos (MENASCÉ; ALMEIDA; DOWDY, 2002 apud MESSIAS, 2016, p. 45).

$$m = \lceil \frac{R\lambda}{R\mu - 1} \rceil \tag{1.5}$$

A seguir, a solução proposta segundo adaptação do algoritmo usando soma ponderada (MESSIAS, 2016, p. 36).

```
Entrada: log de requisição μ, R, w1, w2
Saída: Número de servidores virtuais (recursos) a serem alocados (m)
    Inicio
        μ <- atribuir 0 para valor de recurso subprovisionados de cada modelo
        o <- atribuir 0 para valor de recurso superprovisionados de cada modelo
            para cada hora faça
                μ <- atualizar o número de recursos subprovisionado para cada modelo
                o <- atualizar o número de recursos superprovisionado para cada modelo
                x <- resover a equação 4.3 sujeito as restrições 4.4 e 4.5
                ts <-criar série temporal a partor dos dados extraídos do log
                    para cada modelo de previsão i faça
                        funcaoi <- estimar uma função matemática usando ts e o modelo i
                        pi <- prever a demanda para o próximo periodo usando a funcaoi
                    fim
                    pf <-∑ 6
                           i=1   pixi
                    m <- calcula número de recursos usado μ, R, pf por meio da equação 4.18
                    aloca m recursos junto ao provedor de infraestrutura na nuvem
            fim
    fim
```

Figura 11 - Algoritmo usando soma ponderada.
Fonte: Adaptado de Messias, 2016

As equações 4.3, 4.4 e 4.5, acima apresentadas, correspondem às fórmulas a seguir:

$$Z = minimizar \sum_{i=1}^{2} w_i Z_i \qquad (4.3)$$

$$x_i \in [0,1] \qquad (4.4)$$

$$\sum_{i=1}^{N} x_i = 1 \qquad (4.5)$$

$$m = \lceil \frac{R\lambda}{R\mu - 1} \rceil \qquad (4.18)$$

A respeito do algoritmo de *autoscaling*, Messias (2016) explica:

> O algoritmo 1 ilustra o processo de alocação de recursos usando o método da soma ponderada. Ele recebe como entrada o log de requisições, a taxa de processamento de cada servidor a ser alocado na nuvem (μ), o tempo de resposta máximo acordado com o cliente (R), e o peso de cada um dos objetivos (w1, w2). A saída será a quantidade de recursos a ser alocada 4.2. Autoscaling proposto 37 para o próximo intervalo de tempo (hora), de acordo com a previsão de demanda. Inicialmente, para cada modelo de previsão, é atribuído o valor zero para a quantidade de recursos sub e superprovisionados. Na sequencia o algoritmo entra em um laço onde, a cada intervalo de tempo

> de uma hora, irá: atualizar o número de recursos sub e superprovisionados de acordo com as previsões passadas de cada modelo; definir o peso de cada modelo na previsão resolvendo o sistema de equações lineares das equações 4.3, 4.4 e 4.5; combinar a previsão futura de cada modelo, usando os pesos calculados no passo anterior; e, por fim, calcular e alocar a quantidade necessária de recursos para a próxima hora, de acordo com a previsão. A atualização da quantidade de recursos sub e superprovisionados para cada modelo de previsão é feita da seguinte forma: supondo que o modelo A previu que, no próximo intervalo de tempo, seriam necessários 10 recursos (servidores virtuais) para atender a demanda, e o modelo B previu 5. No entanto, ao final do intervalo, foi verificado que eram necessários 8 recursos. Assim sendo, o modelo A irá acumular, em sua conta, 2 recursos superprovisionados. Já o modelo B irá acumular 3 recursos subprovisionados. Essa conta é atualizada a cada hora (MESSIAS, 2016, p. 36-37).

O VertElastic implementa um algoritmo que leva em consideração a média do consumo histórico do período *t* — em que esse consumo foi superior ao *threshold* (nosso SLA pré-definido) —, junto com o consumo no momento da elasticidade. Esse cálculo é utilizado para definir o tamanho da nova máquina virtual que será instanciada no método *def create_new_vm*, chegando assim em uma configuração que atenda a demanda crescente da aplicação, levando em conta o histórico, buscando reduzir ao máximo as ações desnecessárias. Por isso, o VertElastic implementa a elasticidade apenas após extrapolar, tanto para cima quanto para baixo, um número pré-definido de vezes em que o *threshold* mínimo ou máximo é violado.

Definida a necessidade de elasticidade e calculado o recurso que deve ser provisionado na nova máquina virtual, havendo disponibilidade desse recurso no nó em que a nova máquina virtual deverá ser criada, dá-se início ao processo de criação da máquina que será dimensionada e realocada no lugar da máquina antiga que, por sua vez, será excluída assim que a nova máquina esteja apta para assumir seu lugar.

```
1  Entrada: Total de vms, threshold maximo cpu, threshold maximo memoria, threshold minimo cpu, threshold minimo memoria, vezes que violou limite
2  maximo de cpu/memoria para elasticidade, invervalo de check em segundos, templaste original, templaste original elasticio
3  Saída : ação de elasticidade vertical, elasticidade horizontal
4      Inicio
5          vms_encontradas  <- guarda as vms monitoradas pelo serviço
6          vms_com_metricas <- guarda lista de vms com metrica
7              para cada periodo de tempo faça
8                  MVM, LogC, Thld <- meticas vms, logs consumo, threshold
9                      se threshold violado dispara elasticidade
10                         chama metodos criação vm, status vm e remove vm;
11                     senão faz elasticidade horizontal
12                     fim
13             fim
14     fim
```

Figura 12 -Pseudo código do deamon do VertElastic
Fonte:Elaborado pelo Autor

5 METODOLOGIA DE AVALIAÇÃO

Neste capítulo, será descrito o percurso empreendido no estudo desenvolvido até chegarmos aos seus resultados finais. Mas antes dessa descrição, faz-se necessário mencionar que tipo de pesquisa foi realizada, qual seu objeto, questão a ser respondida, objetivos geral e específicos, instrumentos e procedimentos empregados.

Trata-se de uma pesquisa de abordagem quantitativa, de natureza aplicada, visto que é dirigida à solução de um problema específico. Quanto aos objetivos, consiste em uma pesquisa explicativa, uma vez que se preocupou em identificar os fatores que determinam ou contribuem para a ocorrência dos fenômenos que foram investigados (GIL, 2007). Por fim, quanto aos procedimentos, o estudo se constituiu em uma pesquisa experimental.

De acordo com Gil (2007), “a pesquisa experimental consiste em determinar um objeto de estudo, selecionar as variáveis que seriam capazes de influenciá-lo, definir as formas de controle e de observação dos efeitos que a variável produz no objeto” (GIL, 2007, p. 47). Nesse tipo de estudo, o pesquisador é um agente ativo, e não um observador passivo. É da ação dele que dependem os resultados a serem alcançados.

Na pesquisa experimental, a elaboração de instrumentos para a coleta de dados é submetida a testes para verificar sua empregabilidade e assegurar sua eficácia em medir aquilo que a pesquisa se propôs a medir. Para tanto, ela pode ser desenvolvida em laboratório ou no campo (GERHARDT e SILVEIRA, 2009). No caso específico deste estudo, o contexto se deu em laboratório.

O objeto de investigação desta pesquisa foi a elasticidade vertical, girando em torno da implementação de um módulo de tomada que permitisse ao AutoElastic se utilizar também dos benefícios da elasticidade vertical sempre que possível, tornando o AutoElastic ainda mais flexível em uma nova versão que foi denominada, neste estudo, VertElastic.

Conforme já anunciado, a questão de pesquisa que disparou e permeou todo o processo investigativo foi a seguinte: **Como se configura um modelo de elasticidade para aplicações de alto desempenho baseado no AutoElastic que possibilite a implementação de elasticidade vertical, permanecendo totalmente transparente a necessidade da reescrita de código?**

Os desdobramentos advindos da questão de pesquisa inicial, e que já foram explicitados em capítulo anterior[1].

O contexto físico do estudo foi a Universidade do Vale do Rio dos Sinos - UNISINOS. A infraestrutura de nuvem para teste foi o laboratório C01 413 do Programa de Pós-Graduação em Computação Aplicada dessa Universidade. Nesse laboratório, foi utilizada a plataforma de nuvem OpenNebula existente. Convém ressaltar que consiste em no mesmo contexto, infraestrutura e plataforma de nuvem utilizados para a avaliação do modelo AutoElastic que serviu de base para esta pesquisa.

Como procedimentos de investigação, temos as seguintes ações: revisão de literatura acerca do tema investigado; levantamento bibliográfico de material pertinente ao objeto de estudo; análise documental do referido material no sentido de proceder a um mini estado da arte sobre o tema; dois encontros com o pesquisador desenvolvedor do AutoElastic; desenvolvimento de um novo protótipo denominado VertElastic; verificação dos resultados e análises finais; reuniões (presenciais e virtuais) com orientador; registro em texto do percurso investigativo.

5.1 PROTÓTIPO

Como já foi supramencionado, para fins de avaliação e comparação do modelo proposto de elasticidade vertical, foram utilizados o mesmo contexto, a mesma infraestrutura de nuvem para teste, a mesma plataforma de nuvem e a mesma aplicação interativa empregada na avaliação do modelo AutoElastic. Com isso, objetivou-se comparar os resultados da elasticidade vertical proativa, que é a proposta deste estudo, com os valores obtidos na proposta original de elasticidade horizontal reativa.

A fim de integrar o modelo de decisão para elasticidade vertical do VertElastic ao modelo já existente do AutoElastic, foi desenvolvido um protótipo do VertElastic utilizando a linguagem de programação *Ruby* que foi utilizada em virtude de contar com uma extensa documentação e uma API bem completa de

[1] Cf. subcapítulo 4.1 DECISÃO DO PROJETO.

integração com o *middleware* OpenNebula, sendo ainda uma linguagem com que este pesquisador tem maior identificação e familiaridade.

A API do *middleware* OpenNebula em *Ruby* fornece a flexibilidade necessária para obter as informações sobre o consumo e os recursos da nuvem computacional. Além disso, permite que seja efetuado o gerenciamento das máquinas de maneira bastante flexível, sendo possível, inclusive, instanciar e remover máquinas virtuais da nuvem computacional implementada.

O OpenNebula trabalha com *template*. Sendo assim, dois *templates* diferentes foram criados para podermos distinguir o processo mestre do processo escravo. Outros parâmetros também podem ser informados antes do início da execução do protótipo, como: o *threshold* máximo e mínimo; o intervalo de checagem; e a quantidade de vezes em que o *threshold* pode ser ultrapassado antes de executar o processo de elasticidade.

Foi passado por parâmetro ao OpenNebula o arquivo XML com os *templates* iniciais que contêm as informações mínimas e máximas esperadas para a execução da aplicação. Na nuvem do OpenNebula, foi utilizado um espaço no NFS, conforme já mencionado anteriormente[2]. Nessa área se dá a troca de notificações e a comunicação entre o mestre e o escravo.

5.2 APLICAÇÃO PARALELA DE TESTE

A escolha da aplicação que calcula a aproximação para a integral do polinômio *f (x)* em intervalo fechado e que foi implementada com o método de Newton-Cotes — conhecida como Regra do Trapézio Repetida, apresentada por Comanescu (2012) — deveu-se ao fato de ter sido desenvolvida e utilizada como aplicação de teste de desempenho na avaliação de performance do AutoElastic.

A integral *f (x)* consiste na soma das áreas do trapézio contidas no intervalo [a,b], onde o processo mestre distribui as equações entre os processos escravos, não sendo necessariamente um processo equânime de divisões, visto que a equação *(s+1)* pode não ser divisível pela quantidade exata de escravos.

Rodrigues (2016) destaca que sendo *s* a definição da quantidade de subintervalos e, consequentemente, a quantidade de equações para computar a

[2] Cf. subcapitulo 4.2 ARQUITETURA.

integral, quanto maior for esse parâmetro, maior é a carga computacional para atingir o resultado final para uma equação em particular. A equação define a integral *f (x)* como a soma das áreas do trapézio no intervalo [a.b] em que A_i = área do trapézio *i* com *i* = 0, 1, 2, 3, ..., s-1.

$$\int_a^b f(x)\,dx \approx A_0 + A_1 + A_2 + A_3 + \ldots + A_{s-1} \tag{4.9}$$

A aplicação permite utilizar quatro padrões de carga: decrescente, crescente, constante e onda. Em todos eles, o processo é o mesmo. A carga de trabalho é gerida pelo mestre que encaminha para os escravos uma lista com equações e seus parâmetros e intervalo, sendo o retorno uma quantidade igual de valores de números. Cada processo enviado pelo mestre aos escravos uma equação é selecionada. A aplicação tem como foco final a variação de processamento, e não o resultado final da integral.

A Figura 12, adaptada de Righi et al. (2015) representa graficamente a padronização das cargas, em que o eixo *x* representa as interações (equação a ser calculada) e o eixo *y*, por sua vez, representa a carga de processamento propriamente dita. Os intervalos entre [a,b] determinam a quantidade de equações a serem calculadas por cada escravo. Logo, quando maior o subintervalo, maior será a carga de processamento que cada escravo irá executar.

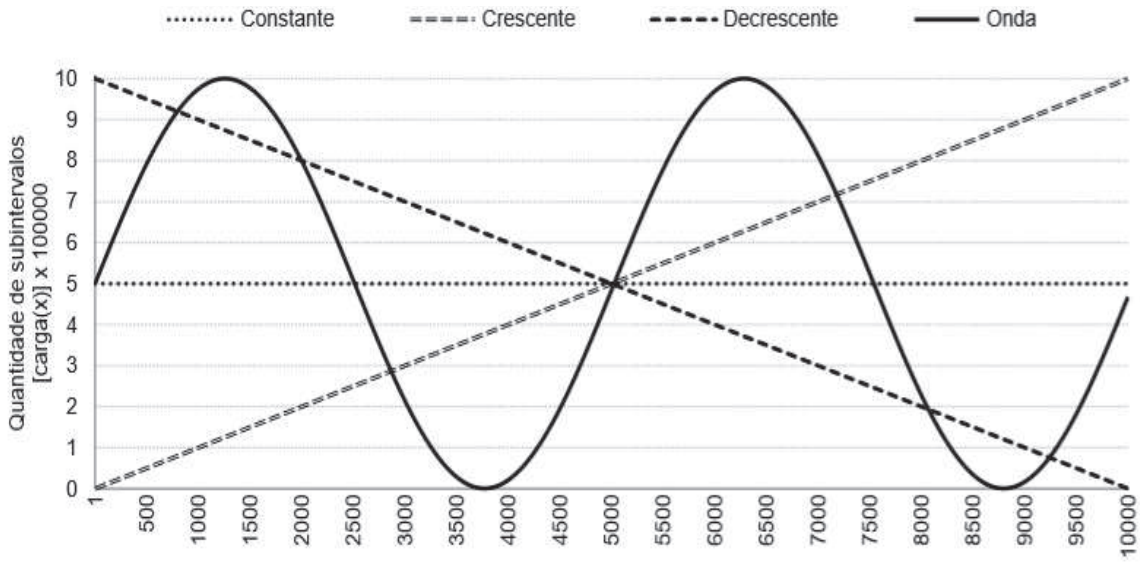


Figura 13 - Padrão das cargas
Fonte: Righi et al., 2015.

5.3 CENÁRIOS DE AVALIAÇÃO

Foram efetuados testes em alguns cenários pré-definidos com o intuito de coletar resultados para comparação dos diferentes métodos aplicados. A aplicação foi executada no *cluster* OpenNebula sem elasticidade, utilizando o método de elasticidade horizontal — modelo padrão implementado pelo AutoElastic — e com elasticidade vertical, modelo proposto pelo VertElastic. Dessa forma, foram obtidos diversos dados para a elaboração de comparação entre os resultados apresentados por Rodrigues (2016) — que realizou a avaliação em um mesmo cenário e com a mesma aplicação de teste, porém implementando o AutoElastic e o método de elasticidade horizontal — e os resultados obtidos com o VertElastic.

Assim, foram elencados os cenários dos testes a serem executados:

- C.E1: Execução da aplicação de cálculo de integrais com apenas uma máquina física, habilitada para proporcionar elasticidade vertical com o *threshold* fixo habilitado (tanto o *threshold* mínimo quanto o *threshold* máximo) gerando, assim, diferentes cenários de execução;
- C.E2: Execução da aplicação de cálculo de integrais com apenas uma máquina física habilitada para proporcionar elasticidade vertical, sem a definição do *threshold* fixo habilitado, gerando diferentes cenários de execução.

Com o comportamento da execução de ambos os cenários, se obtêm as métricas da aplicação de teste que foram utilizadas para confrontar os resultados das execuções entre VertElastic e AutoElastic. Para que os testes tivessem um ambiente igual ao utilizado por Rodrigues em 2016 — e assim, garantir a comparação o mais fidedignamente possível —, adotou-se o mesmo intervalo de observação de monitoramento, 15 segundos, além de serem utilizadas as mesmas métricas de avaliação, que serão delineadas no subcapítulo a seguir.

5.4 MÉTRICAS PARA AVALIAÇÃO

Foram elencadas métricas para analisar o desempenho da aplicação, a eficiência de alocação de recursos, o consumo de recurso e o custo estimado baseado no total de recurso consumido. Para avaliação da aplicação quanto ao tempo gasto de execução, foi utilizado o Speedup (S), representado na equação (4.6), em que a função entre número de processos *p* resulta do tempo de execução sequencial sobre a execução em paralelo *t(1)* e *t(p)*.

$$S(p) = \frac{t(1)}{t(p)} \tag{4.6}$$

Já na equação (4.7), Rodrigues (2016) explica que a eficiência de um sistema paralelo pode ser descrita na fração de tempo que é utilizada pelos processadores *p* em uma dada computação.

$$E(p) = \frac{S(p)}{p} \tag{4.7}$$

Para a medição de energia, foi utilizada a abordagem apresentada por Rodrigues (2016). A fim de fornecer uma maneira mais fácil de medir a energia consumida durante a execução da aplicação, foi definida uma métrica de avaliação calculada empiricamente, chamada de Energia, sendo calculada pela equação 4.8.

$$Energia(i, s) = \sum_{j=i}^{s} (j \times pt_e(j)) \tag{4.8}$$

Quanto ao custo, foi adotada uma medida de Rodrigues (2016) em que a noção relativa a essa métrica foi adaptada do conceito de custo referente a uma computação paralela, conforme abordada por Wilkinson e Allen (2005 apud

RODRIGUES, 2016), porém, no caso específico desta pesquisa, com suporte a ambientes elásticos.

5.4 METODOLOGIA DE AVALIAÇÃO

Ao final do Capítulo 3, foram avaliados diversos trabalhos que oferecem estratégias distintas de elasticidade. Em sua maioria, abordam a elasticidade de forma horizontal e com a necessidade de configurações prévias pelo administrador do ambiente.

Da mesma forma, a maioria das pesquisas encontradas utiliza a CPU como métrica para efetuar as avaliações, em que níveis de consumo de CPU são previamente definidos como *threshold*, ou seja, valores limiares que quando alcançados, são gatilhos para influenciar uma tomada de decisão baseada nos algoritmos e definições dos gerenciadores de elasticidade.

Os valores de *threshold* mais comumente observados foram: 30%, 35%, 40%, 45%, 50%, 75%, 80%, 90% e 95%. Por estarem sendo amplamente utilizados, esses valores foram levados em considerações no teste.

Assumindo os *thresholds* observados em diversos trabalhos, a pesquisa delimitou-se a utilizar aqueles que empiricamente parecem estar em consonância com a sua utilização fora do mundo acadêmico, mas que, ainda assim, permitem que esta pesquisa mantenha sua relevância científica, visto que uma granularidade tão elevada nos *thresholds* mínimos e máximos, embora possuam valor cientifico, não seriam de fato utilizados fora do ambiente de pesquisa.

Sendo assim, foram adotados os *thresholds* inferiores de 30% e 35%, levando em consideração o trabalho de Al-Haidari, Sqalli e Salah (2013) que observaram o desempenho da elasticidade em nuvem à luz da teoria de filas. Já para a seleção dos *thresholds* superiores, a escolha apoiou-se nas pesquisas de Suleiman (2012); Al-Haidari, Sqalli e Salah (2013); Beernaert et al. (2012), que utilizam 80% e 90% em suas pesquisas.

Portanto, totalizam-se quatro execuções de cargas, conforme foi representado na tabela a seguir:

Tabela 2 - Apresentação das combinações possíveis de thresholds fixos

Thresholds (%)		Superior	
		80	90
Inferior	30	80/30	90/30
	35	80/35	90/35

Fonte: Autoria própria.

5.5 CONSIDERAÇÕES PARCIAIS

A metodologia desenvolvida para fundamentar a pesquisa e os procedimentos realizados para apresentar o protótipo do VertElastic, bem como testá-lo, foram abordados neste capítulo.

O protótipo do VertElastic foi desenvolvido em *Ruby*, e os testes foram conduzidos em um ambiente de nuvem baseado no *middleware open source* OpenNebula que fornece uma API bastante completa para a linguagem escolhida para o desenvolvimento do referido protótipo. Com isso, as tarefas necessárias de gerenciamento, monitoração e organização dos recursos no *OpenNebula* foram todas executadas pelo VertElastic. A aplicação de cálculo de integrais utilizada foi desenvolvida em *Java* e apresentada por Rodrigues (2016).

O modelo desenvolvido nesta pesquisa foi posto à prova em diversos testes com cargas de processamento parametrizadas, de acordo com a função a ser calculada pela aplicação. Para que os resultados fossem comparados diretamente, foram definidos dois cenários para essa testagem:

- C.E1: aplicação sendo executada com elasticidade e *thresholds* fixos;
- C.E2: aplicação sendo executada com elasticidade sem a definição de *thresholds*;

As avaliações dos resultados desta pesquisa são apresentadas no próximo capítulo.

6 RESULTADOS E ANÁLISES

Neste capítulo, são apresentados os resultados obtidos através dos dois cenários descritos no capítulo anterior, comparando seu desempenho com os testes efetuados na apresentação do AutoElastic, em Rodrigues (2016), que serão disponibilizados em anexo neste trabalho. Primeiramente, apresenta-se o resultado relacionado à métrica de Tempo, Energia, Custo, *Speedup* e Eficiência elástica.

6.1 AVALIAÇÃO DAS MÉTRICAS TEMPO, ENERGIA ECUSTO

A Tabela 3 expõe a os resultados referentes às métricas adotadas — Tempo, Energia e Custo — para avaliar o protótipo VertElastic, considerando os dois cenários proposto no subcapítulo 5.5[3], quais sejam: C.E1 - aplicação sendo executada com elasticidade e *thresholds* fixos; e C.E2 - aplicação sendo executada com elasticidade sem a definição de *thresholds*.

É possível observar que no primeiro cenário, temos uma variação maior da métrica Tempo quando são utilizados *thresholds* superiores mais elevados, como visto na figura 14. O que faz total sentido, visto que quanto antes forem feitas as ações de elasticidade, mais rapidamente são alocados recursos extras possibilitando, assim, que a carga total de trabalho seja executada mais

[3] Cf. subcapítulo 5.5 CONSIDERAÇÕES PARCIAIS.

rapidamente. Quanto mais tempo levar para que os *thresholds* superiores sejam

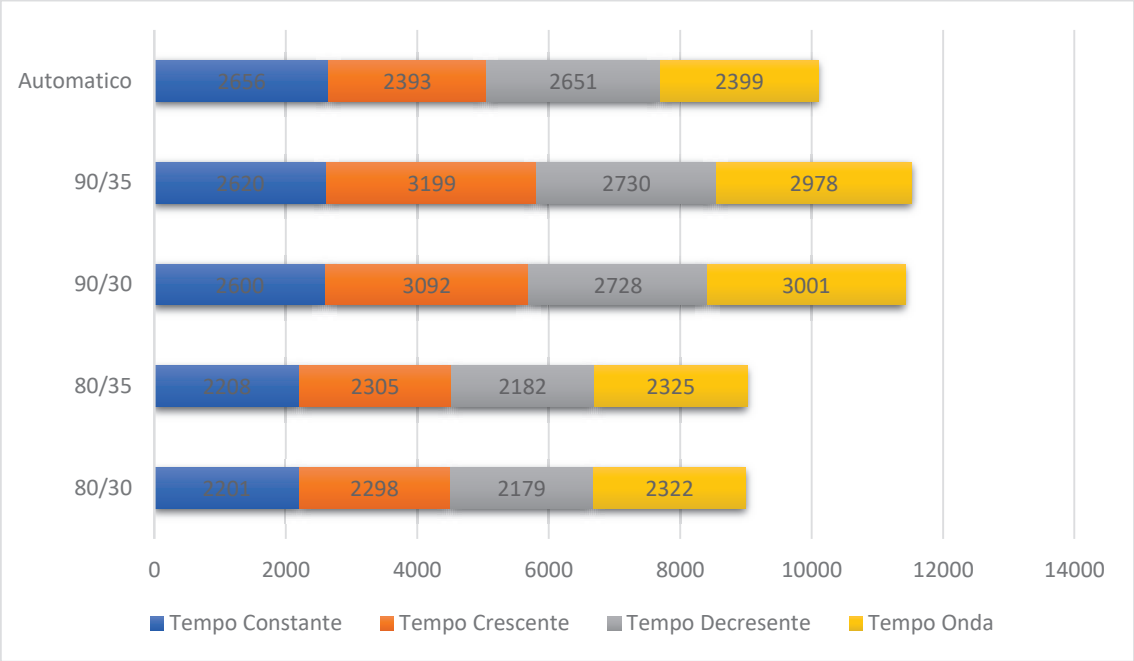


Figura 14 - Métricas de Tempo os comportamentos de cargas da aplicação nos cenários com thresholds fixos e sem thresholds definidos

violados, mais tempo a aplicação será executada com a carga alta e, consequentemente, mais tempo levará para finalizar todo processo.

A melhor métrica Tempo foi obtida com *thresholds* superiores em 80% e a pior, em 90%. Todavia, a métrica Energia se portou de maneira oposta — o que já era esperado, visto que para executar a tarefa de forma mais rápida, foi necessário alocar recurso extra mais rápido, como visto na figura 15. Isso fez com que se amenizasse a carga das máquinas, porém implicou instanciar máquinas mais robustas que consomem mais. Assim, embora tenhamos obtido um ganho no tempo total de execução, esse ganho não foi suficiente para garantir o consumo equivalente. A melhor métrica Energia foi obtida com *thresholds* superiores em 90% e a pior, em 80%.

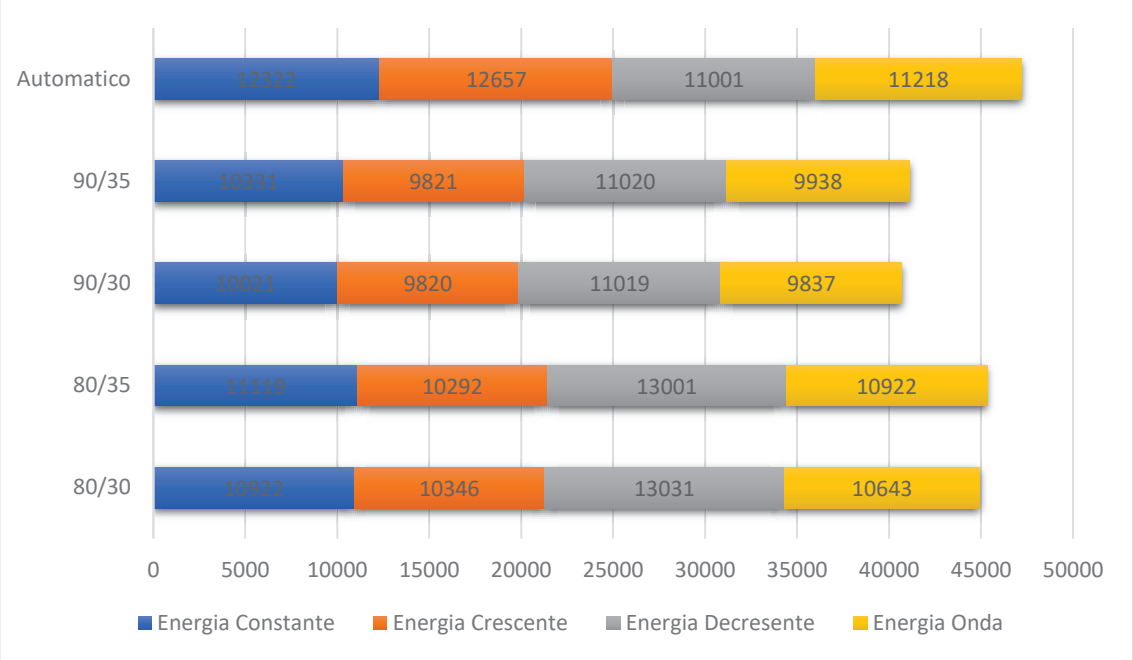


Figura 15 - Métricas de energia os comportamentos de cargas da aplicação nos cenários com Thresholds fixos e sem thresholds definidos

Ainda levando em conta o índice de energia, podemos observar, por meio de experimentos, que quando a carga tem um comportamento crescente, um *threshold* superior (T_S) tem melhor desempenho, uma vez que quanto mais próximo ao limite máximo de processamento for o *threshold* superior, menores serão as possibilidades de ações de elasticidades, como demostrado na figura 15. Já para obtermos um melhor resultado com relação à medida de energia, precisamos ter a diminuição de recurso computacional, o que acontece quando temos *thresholds* inferiores (T_i) mais elevados, como acontece quando a carga possui um comportamento crescente.

Por outro lado, quando a carga de processamento é reduzida com o tempo, menos computação é necessária e os recursos disponíveis são mais bem aproveitados, fato este que demostra porque o ato de demover recurso não tem impacto significativo em cargas decrescentes. Em contrapartida, a alocação de recurso é bastante custosa e impacta fortemente em aplicação com cargas crescentes. Na Tabela 3 (a seguir), demonstrativa dos dados ora descritos, é possível perceber um valor melhor de energia quando temos (T_S) em 90% na carga crescente e um valor melhor para (T_i) em 35% na carga decrescente.

Na métrica Custo, não houve um comportamento inversamente proporcional como o que foi apresentado pelo Tempo e pela Energia. O padrão

mais próximo que pode ser traçado é que em praticamente todas as formas de execução, o custo está profundamente ligado ao melhor índice de tempo. O mesmo *threshold* superior e *threshold* inferior obtiveram o melhor índice de custo e tempo, na carga constante e na carga crescente, conforme demostrado abaixo na figura 16. Já na carga decrescente e em onda, onde isto não aconteceu, não se observa diferença significativa no índice de tempo — menos de 1% de diferença.

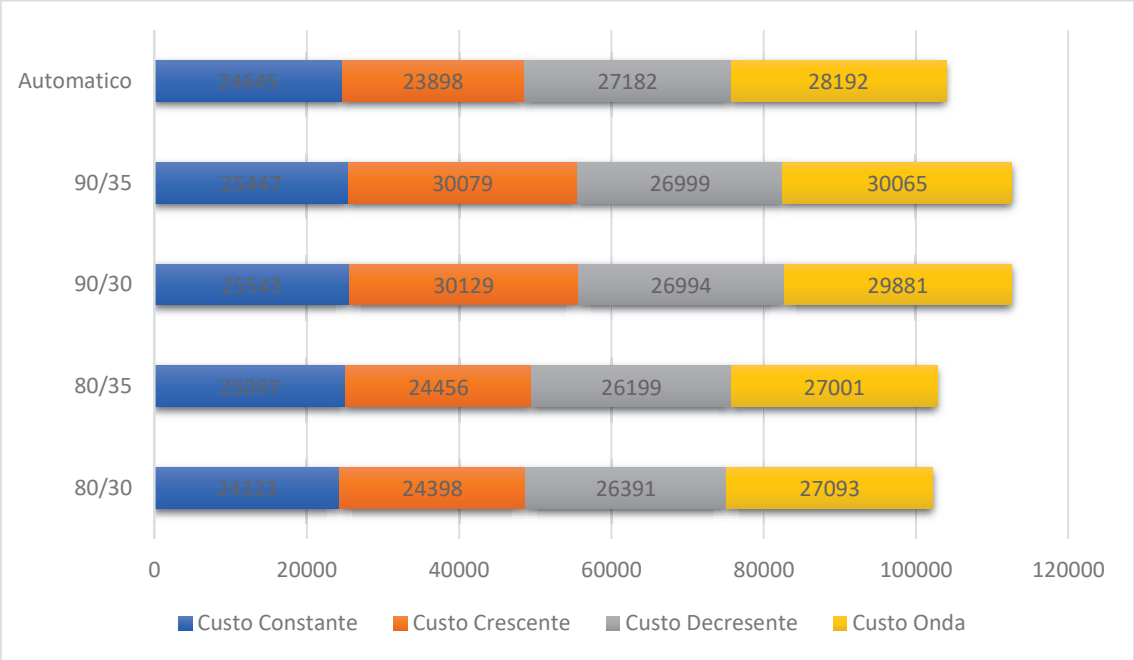


Figura 16 - Métricas de custo os comportamentos de cargas da aplicação nos cenários com thresholds fixos e sem thresholds definidos

Tabela 3 - Métricas de Tempo, Energia e Custo para os comportamentos de cargas da aplicação nos cenários com *thresholds* fixos (C. E1) e sem *thresholds* definidos (C. E2)

Cenário	Parâmetro		Constante			Crescente			Decrescente			Onda		
	T_s	T_i	Tempo	Energia	Custo	Tempo	Energia	Custo	Tempo	Energia	Custo	Tempo	Energia	Custo
C.E1	80	30	2201	10922	24323	2298	10346	24398	2179	13031	26391	2322	10643	27093
	80	35	2208	11119	25097	2305	10292	24456	2182	13001	26199	2325	10922	27001
	90	30	2600	10021	25543	3092	9820	30129	2728	11019	26994	3001	9837	29881
	90	35	2620	10331	25447	3199	9821	30079	2730	11020	26999	2978	9938	30065
C.E2	-	-	2656	12322	24645	2393	12657	23898	2651	11001	27182	2399	11218	28192

Fonte: Autoria própria.

Comparando-se os resultados obtidos nos testes com o VertElastic com os resultados apresentados por Rodrigues (2016) para execução das mesmas cargas sem elasticidade, observou-se que houve um ganho evidente em termos de tempo e custo. Isso pode ser explicado pelo fato de os testes sem elasticidade sobrecarregarem os recursos computacionais por um largo período de tempo, aumentando o tempo de execução. Em decorrência disso, o custo é consideravelmente penalizado, embora o consumo de energia seja o melhor em relação a qualquer cenário de elasticidade, o que evidencia o ganho significativo que a elasticidade vertical pode proporcionar para a computação de alto desempenho.

Comparando os melhores resultados do C. E1 e o C. E2 para a métrica Tempo, notou-se que com a carga constante, o tempo é 17,1% maior. Já com carga crescente, mostrou-se 3,9% mais lento, e com a carga decrescente, o tempo apresentou-se 17,8% mais lento. Por fim, com carga em onda, foi 3,2% mais lento.

Ainda com relação à métrica Tempo, se levarmos em consideração o C. E2 com carga em onda, onde o VertElastic sem *thresholds* definidos obteve o seu melhor resultado para a referida métrica, e comparamos com o valor apresentado por Rodrigues (2016) nos testes do AutoElastic com *Live Thresholding*, o valor do primeiro foi 9,7% mais lento. Isso evidencia que o algoritmo *Live Thresholding* se demonstra mais ajustado no cenário de predição. O aprofundamento deste item será destacado como um dos estudos futuros a serem desenvolvidos.

6.2 AVALIAÇÃO DAS MÉTRICAS RECURSO, SPEEDUP E EFICIÊNCIA ELÁSTICA

O resultado obtido para as métricas Recurso, *Speedup* e Eficiência elástica nos quatro padrões de carga efetuados pela aplicação de teste são apresentados na Tabela 4 a seguir, em que RE se refere à métrica Recurso, SP corresponde a *Speedup* e EE diz respeito à Eficiência elástica.

Tabela 4 - Métricas de Recurso, *Speedup* e Eficiência elástica para os comportamentos de cargas da aplicação nos cenários com *thresholds* fixos (C. E1) e sem *thresholds* definidos (C. E2)

Cenário	Parâmetro		Constante			Crescente			Decrescente			Onda		
	T_s	T_i	RE	SP	EE	RE	SP	EE	RE	SP	EE	RE	SP	EE
C.E1	80	30	5,37	1.78	0,77	4,8	1,97	0,83	6,6	2,21	0,71	5,01	1,88	0,81
	80	35	5,37	1,79	0,77	4,6	1,95	0,83	6,61	2,19	0,74	5,01	1,9	0,81
	90	30	3,8	2,01	0,81	4	1,7	0,89	4	1,8	0,83	3,8	1,69	0,9
	90	35	3,7	2,02	0,81	4,1	1,72	0,9	4	1,81	0,84	3,82	1,69	0,91
C.E2	-	-	6,01	1,62	0,5	6,42	1,77	0,59	6,63	1,97	0,75	6,55	1,98	0,78

Fonte: Autoria própria.

O valor representado pela Eficiência elástica diz respeito ao índice da quantidade de máquinas virtuais utilizadas para a execução de uma aplicação. Ele é calculado com a utilização da métrica Recurso (RE) e tem um comportamento similar à métrica Energia, métrica esta apresentada por Rodrigues (2016). No C. E1, levando em conta os *thresholds* que obtiveram os melhores resultados, como visto na figura 17, bem como os *thresholds* que obtiveram os piores resultados em relação às métricas *Speedup* e Eficiência elástica, é possível traçar uma relação intrínseca entre elas e as métricas Tempo e Energia, vistas na Tabela 3 e. Energia e Eficiência elástica demostraram a mesma relação que *Speedup* e Eficiência elástica apresentam com a métrica Tempo, com a peculiaridade de que nas cargas crescente e constante, elas

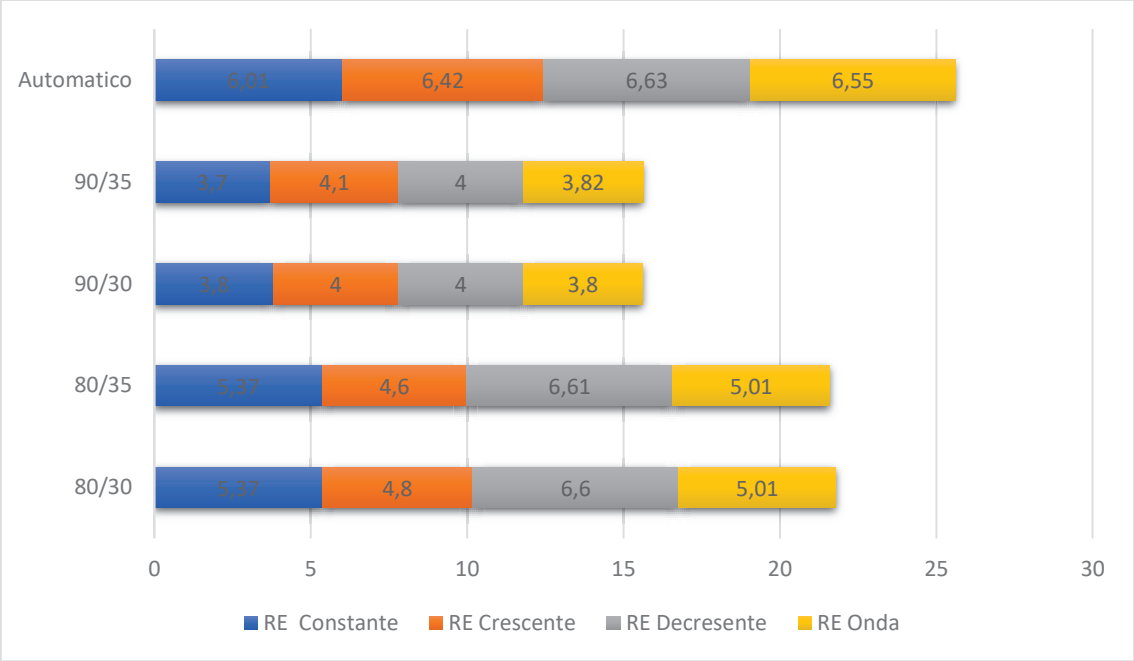


Figura 17 - Métricas recurso os comportamentos de cargas da aplicação nos cenários com thresholds fixos e sem thresholds definidos

obtiveram resultados praticamente semelhantes, representado nas figuras 18 e 19.

Sempre que novos recursos são adicionados na elasticidade horizontal, a tendência é de que a carga comece a ser espalhada e os recursos comecem a ser menos utilizados. Esse fenômeno ocorreu também na elasticidade vertical. Embora se proporcionasse mais recurso para a máquina virtual, existiu uma limitação imposta pela capacidade de crescimento do nó.

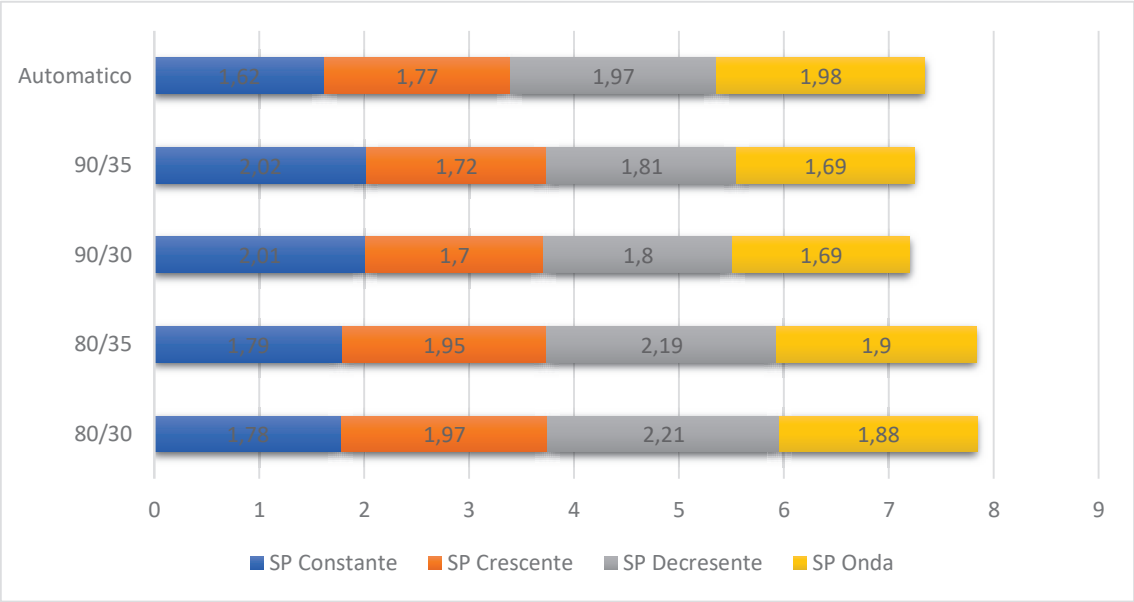


Figura 18 - Métricas speedup os comportamentos de cargas da aplicação nos cenários com thresholds fixos e sem thresholds definidos

O crescimento vertical acontecia primeiramente em uma máquina, mas acabava replicando para o outro par, sendo que a quantidade de máquinas sempre foi configurada para ser, no mínimo, duas máquinas por nó. Como a aplicação utilizada é intensa quando ao consumo de CPU, um melhor desempenho determina uma penalização se a busca for por eficiência, sendo o inverso também verdadeiro.

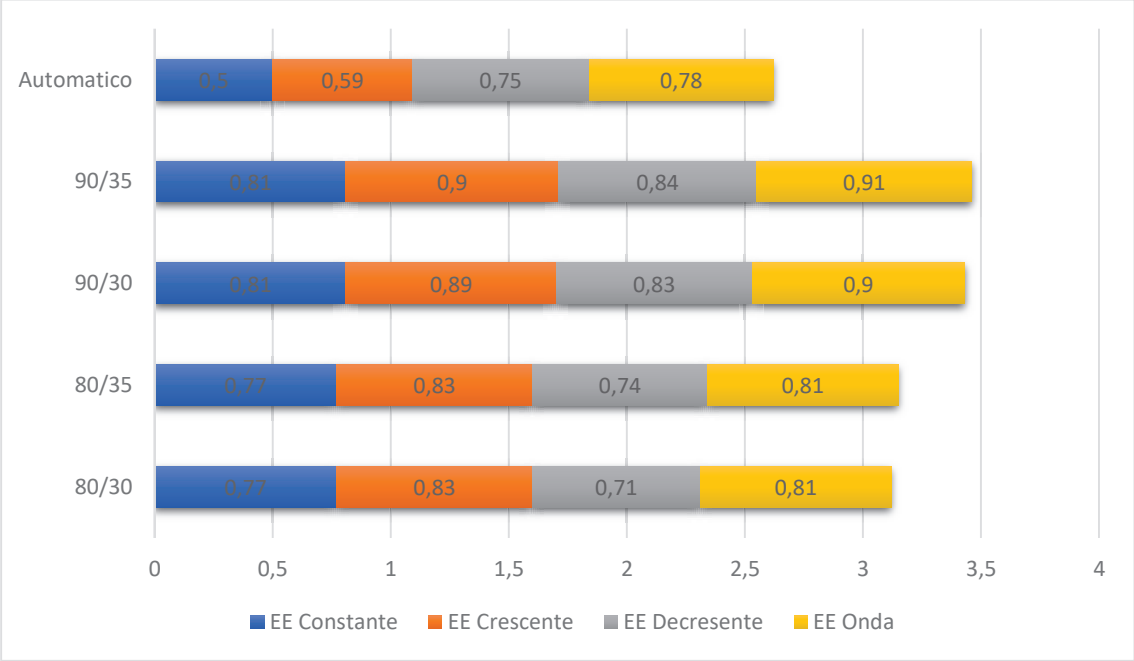


Figura 19 - Métricas eficiência os comportamentos de cargas da aplicação nos cenários com thresholds fixos e sem thresholds definidos

6.3 CONSIDERAÇÕES DO CAPÍTULO

Considerado os testes efetuados em que foi posto à prova o protótipo de elasticidade vertical VertElastic, é possível afirmar que o protótipo obteve um ganho em relação a cenários sem elasticidade em todos os diferentes testes e abordagens explorados. Fica evidenciada uma relação estreita entre desempenho de uma aplicação e o consumo de recurso de quem a executa. O desempenho sempre será melhor quando o consumo de recurso for proporcionalmente maior também. Caso se prime pelo consumo de recursos, é preciso ter em mente que a aplicação irá ser penalizada.

A possibilidade de distribuir as tarefas entre recursos faz com que todo o processo da aplicação seja agilizado. Galante e Bona (2015) alertam que uma aplicação cientifica é diferente de uma aplicação baseada em servidores. As

aplicações cientificas tendem a consumir o recurso disponível em sua totalidade, independentemente da quantidade. Já as aplicações baseadas em servidores consomem o recurso mediante a solicitação de demanda de carga.

Ficou evidente que os piores cenários acontecem quando os *thresholds* superiores são definidos próximos a 100%, ao passo que *thresholds* baixos possibilitam que a ação de elasticidade aconteça antes.

A limitação do hardware que hospeda a máquina virtual e a impossibilidade de alterar parâmetros computacionais em tempo de execução são as lacunas em que vale pena aprofundar a pesquisa.

O tempo de espera para uma nova máquina estar disponível e pronta para começar as atividades ficou em média 172,5 segundos.

Em nenhum dos resultados, o cenário sem elasticidade conseguiu superar os tempos obtidos pelos testes com ações de elasticidade, sendo que os melhores resultados foram obtidos com o trabalho de *thresholds* pré-definidos.

7 CONCLUSÃO

Como já foi amplamente mencionado neste estudo, a elasticidade vertical consiste em adicionar ou remover recursos computacionais de uma instância em execução. Em sua grande maioria, os trabalhos relacionados ao tema de aplicações HPC, usam elasticidade vertical, sendo possível ratificar este fato pela grande maioria dos trabalhos encontrados na área, usarem preferencialmente de elasticidade vertical, em virtude da sobrecarga na reorganização dos recursos.

Utilizando-se do conceito de elasticidade assíncrona, nesta dissertação foi apresentado um modulo de decisão que trabalho em conjunto com o middleware AutoElastic que propôs a utilização de elasticidade vertical e horizontal de forma automática ou com a utilização de parâmetros para atender aplicações de HPC, a este módulo deu-se o nome de VertElastic.

Com isso, vislumbrou-se a possibilidade de combinar a elasticidade vertical com a elasticidade assíncrona, diferenciando o VertElastic de todos os demais trabalhos relacionados até o presente momento.

Tomando como ponto de partida esse modelo podemos evidenciar via teste que é possível efetuar a elasticidade vertical, sem interferência de usuários e sem prejuízo na execução da aplicação de HPC. Após sua implementação e testagem, conclui-se que o módulo aqui desenvolvido conseguiu responder à principal pergunta que permeava todo o trabalho: Como se configura um modelo de elasticidade para aplicações de alto desempenho baseado no AutoElastic que possibilite a implementação de elasticidade vertical, permanecendo totalmente transparente a necessidade da reescrita de código? Para tanto, basta utilizar o modelo do VertElastic que dá conta da demanda explicitada na referida questão.

O algoritmo do VertElastic constituiu-se na principal contribuição científica deste trabalho. O algoritmo que foi desenvolvido para ser incorporado ao VertElastic mostrou, nos testes realizados, ser capaz de garantir um melhor desempenho se comparado à não utilização de elasticidade. Seus ganhos se situam entre 13% e 38%. Portanto, fica evidente o benefício da utilização de elasticidade vertical em aplicações para computações de alto desempenho.

Por outro lado, esperava-se que seu desempenho fosse melhor nas testagens realizadas com *thresholds* sem definição, o que não aconteceu.

Observou-se que o VertElastic se mostrou mais efetivo em relação ao cenário onde ele trabalhou com *thresholds* definidos. Tal ocorrência sugere uma revisão no algoritmo de predição para que o VertElastic alcance resultados mais satisfatórios, trabalhando de forma proativa.

Como campo de ações futuras, pode-se elencar o refinamento do algoritmo do VertElastic, a fim de garantir resultados ainda melhores, tanto trabalhando com a definição de *thresholds*, quanto não. Com a adoção cada vez maior de contêineres, uma provável lacuna tente a ser a viabilidade e se existirá a necessidade de utilização de elasticidade para ferramentas que oferecem o encapsulamento das aplicações e do sistema operacional em pequenos executáveis totalmente gerenciáveis e dinâmicos. Outra possibilidade de aprofundamento futuro consiste em testar o VertElastic em um ambiente onde a alocação de recurso possa acontecer em tempo de execução, sem a necessidade de iniciar uma nova instancia e sim permitir o redimensionamento *on-the-fly* das características computacionais, como já é possível, hoje em dia, em nuvens privadas, como é o caso do VMWARE.

REFERÊNCIAS

ABRANCHES, Marcelo Cerqueira de. **Um Mecanismo de Auto Elasticidade com base no Tempo de Resposta para Ambientes de Computação em Nuvem baseados em Containers.** 2017. 120 f. Dissertação (Mestrado) - Curso de Instituto de Ciências Exatas, Ciência da Computação, Universidade de Brasília, Brasília, 2017.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R. H.; KONWINSKI, A.; LEE, G.; PATTERSON, D. A.; RABKIN, A.; STOICA, I.; ZAHARIA, M. **Above the clouds**: A berkeley view of cloud computing. Technical report, EECS Department, University of California, Berkeley, 2009.

BEERNAERT, L.; MATOS, M.; VILAÇA, R.; OLIVEIRA, R. Automatic elasticity in openstack. In: Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management (2012), **Proceedings** …ACM, p. 2.

CHIRIGATI, Fernando Seabra. Computação em Nuvem**.** Rio de Janeiro, **Anais eletrônicos...**Rio de Janeiro: UFRJ, 2009. Disponível em: <http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2009_2/seabra/>. Acesso em: 30 maio 2017.

CHIRIGATI, FERNANDO; OLIVEIRA, DANIEL DE; OGASAWARA, EDUARDO. Exploring many task computing in scientific workflows. In MTAGS '09 Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, 2009. **Proceedings** …*IEEE.* Article N°. 2

COUTINHO, Emanuel Ferreira; SOUSA, Flávio R. C.; REGO, Paulo Antonio Leal; GOMES, Danielo Goncalves; SOUZA, José Neuman de. Elasticity in cloud computing: a survey. **Annales des Télécommunications** 70 (2015): 289-309.

COUTINHO, E. F.; PAILLARD, G.; SOUZA, J. N. de. Performance analysis on scientific computing and cloud computing environments. In: euro american conference on telematics and information systems, 7., 2014, New York, NY, USA. **Proceedings...** ACM, 2014. P. 5:1–5:6. (eatis ’14).

D. Milojicic, I. M. Llorente, R. S. Montero, "Opennebula: A cloud management tool". In: Internet Comput. **Proceedings** …*IEEE*, vol. 15, pp. 11-14, Mar./Apr. 2011.

EHLERS, Ricardo. Análise de séries temporais. Curitiba **Anais eletrônicos...**Curitiba: UFPR, 2017. Disponível em: <http://www.each.usp.br/rvicente/AnaliseDeSeriesTemporais.pdf>. Acesso em: 21 maio 2017.

E. Roloff, F. Birck, M. Diener, A. Carissimi, P. Navaux, "Evaluating high performance computing on the Windows Azure platform"**.** In**:** *5th Int. Conf. Cloud Comput.* **Proceedings** *…IEEE,* pp. 803-810, 2012.

FAROKHI, Soodeh et al. Performance-Based Vertical Memory Elasticity. In: IEEE International Conference On Autonomic Computing**,** [s.l.], p.1-2, jul. 2015. **Proceedings** *…IEEE*. Disponível em: <http://dx.doi.org/10.1109/icac.2015.51>. Acesso em: 30 maio 2018.

FERREIRA, Edmar. Escolhendo entre escalabilidade horizontal e escalabilidade vertical. 2010. Disponível em: <http://escalabilidade.com/2010/09/21/ escolhendo-entre-escalabilidade-horizontal-eescalabilidade-vertical/>. Acesso em: 16 julho 2017.

GALANTE, Guilherme. **Explorando a elasticidade em nível de programação no desenvolvimento e na execução de aplicações científicas.** 2014. 115 f. Tese (Doutorado) - Curso de Ciências Exatas, Programa de Pós-graduação em Informática, Universidade Federal do Paraná, Curitiba, 2014.

GALANTE, G.; BONA, L. C. E. d. A Survey on Cloud Computing Elasticity. In: IEEE/ACM FIFTH INTERNATIONAL CONFERENCE ON UTILITY AND CLOUD COMPUTING, 2012., 2012, Washington, DC, USA. **Proceedings. . .** IEEE Computer Society, 2012. p. 263–270. (UCC '12).

GALANTE, G.; BONA, L. de. A Survey on Cloud Computing Elasticity. In: UTILITY AND CLOUD COMPUTING (UCC), 2012 IEEE FIFTH INTERNATIONAL CONFERENCE ON, 2012. **Proceedings** *…IEEE*, 2012. p. 263–270.

GERHARDT, Tatiana Engel e SILVEIRA, Denise Tolfo (orgs.). Métodos de pesquisa. **UAB/UFRGS. Curso de Graduação Tecnológica – Planejamento e Gestão para o Desenvolvimento Rural da SEAD/UFRGS**. – Porto Alegre: Editora da UFRGS, 2009.

GIL, A. C. **Como elaborar projetos de pesquisa.** 4. ed. São Paulo: Atlas, 2007.

HERBST, Nikolas Roman; KOUNEV, Samuel; REUSSNER, Ralf. Elasticity in Cloud Computing**:** What It Is, and What It Is Not. In: ICAC 2013: 10th International Conference on Autonomic Computing (2013). **Proceedings. . .**. ICAC , Jun 26-28, 2013, San Jose-CA. Disponível em: <https://sdqweb.ipd.kit.edu/publications/pdfs/HeKoRe2013-ICAC-Elasticity.pdf>. Acesso em: 13 julho 2017.

KONSTANTELI, K.; CUCINOTTA, T.; PSYCHAS, K.; VARVARIGOU, T. Admission control for elastic cloud services. In: *IEEE* 5th International Conference on (2012), Cloud Computing (CLOUD), 2012. **Proceedings** …, IEEE, p. 41–48.

LATORRE, Maria do Rosário Dias de Oliveira e CARDOSO, Maria Regina Alves. Análise de séries temporais em epidemiologia: uma introdução sobre os aspectos metodológicos. **Rev. bras. epidemiol.** [online]. 2001, vol.4, n.3, pp.145-152. ISSN 1415-790X. Disponível em: <http://dx.doi.org/10.1590/S1415-790X2001000300002>. Acesso em: 16 julho 2018.

M. Comanescu, **"**Implementation of time-varying observers used in direct field orientation of motor drives by trapezoidal integration". In: **.** *6th IET Int. Conf. Power Electron.* **Proceedings. .**. *Mach. Drives*, pp. 1-6, 2012.

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

MESSIAS, Valter Rogério. **Combinação de modelos de previsão de séries temporais por meio de otimização multiobjetivo para alocação eficiente de recursos na nuvem.** 2016. 2017 f. Tese (Doutorado) - Curso de Instituto de Ciências Matemáticas e de Computação, Programa de Pós-graduação em Ciências de Computação e Matemática Computacional, Universidade de São Paulo, São Carlos, 2016.

NIST Definition of Cloud Computing**.** United States: National Institute of Standards & Technology Created December 01, 2016, Updated October 06, 2017. Disponível em: <http://csrc.nist.gov/groups/SNS/cloud-computing/>. Acesso em: 16 julho 2016.

OPUS SOFTWARE. **O Que Você Realmente Precisa Saber Sobre Computação em Nuvem.** 1. ed. São Paulo - SP - Brasil. Edição realizada por Opus Software Com. e Repr. Ltda. 2015.

PORTO, Ivens Oliveira. **Padrões e diretrizes arquiteturais para escalabilidade de sistemas.** Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação. Uberlândia-MG, 2009.

RIGHI, Rodrigo & GALANTE, Guilherme. Explorando a Elasticidade em Nuvem para a Computação de Aplicações Paralelas**.** São Leopoldo. **Anais eletrônicos**... São Leopoldo: UNISINOS. ERAD, 2016. Disponível em: <https://www.researchgate.net/publication/311589613_Explorando_a_Elasticidade_em_Nuvem_para_a_Computacao_de_Aplicacoes_Paralelas>.

RIGHI, Rodrigo da Rosa et al. AutoElastic: Automatic Resource Elasticity for High Performance Applications in the Cloud. In: Transactions On Cloud Computing, [s.l.], v. 4, n. 1, p.6-19, 1 jan. 2016. **Proceedings** *…IEEE* (. Disponível em: <http://dx.doi.org/10.1109/tcc.2015.2424876>.

RIGHI, R.; RODRIGUES, V.; COSTA, C. A, da; GALANTE, G.; BONA, L. C. E. de; FERRETO, T. AutoElastic: automatic resource elasticity for high performance applications in the cloud. In: Cloud Computing, IEEE Transactions **Proceedings. . .** New York, NY, USA, v. PP, n. 99, p. 1–1, 2015.

RODRIGUES, Vinicius Facco. **AutoElastic:** Explorando a Elasticidade de Recursos em Nuvem para Aplicações de Alto Desempenho Interativas. 2015. 99 f. Dissertação (Mestrado) - Curso de Computação Aplicada, Programa de Pós-graduação em Computação Aplicada, Universidade do Vale do Rio dos Sinos, São Leopoldo, 2016.

ROLOFF, Eduardo et al. HPC Application Performance and Cost Efficiency in the Cloud. In: 25th Euromicro International Conference On Parallel, Distributed And Network-based Processing (pdp)**,** [s.l.], p.2-3, 2017. **Proceedings** *…IEEE*. Disponível em: <http://dx.doi.org/10.1109/pdp.2017.59>. Acesso em: 16 julho 2016.

SHARMA, U.; SHENOY, P.; SAHU, S.; SHAIKH, A. A Cost-Aware Elasticity Provisioning System for the Cloud. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 2011, Washington, DC, USA. **Proceedings** *…IEEE* Computer Society, 2011. p. 559–570. (ICDCS ’11).

SOTIRIADIS, Stelios et al. Vertical and horizontal elasticity for dynamic virtual machine reconfiguration. In**:**Transactions On Services Computing**,** [s.l.], p.1-1, 2016. **Proceedings** *…IEEE*. Disponível em: <http://dx.doi.org/10.1109/tsc.2016.2634024>. Acesso em: 16 março 2017.

SPINNER, S.; KOUNEV, S.; ZHU, X.; LU, L.; UYSAL, M.; HOLLER, A.; GRIFFITH, R. Runtime Vertical Scaling of Virtualized Applications via Online Model Estimation. In: IEEE 8TH INTERNATIONAL CONFERENCE ON SELF-ADAPTIVE AND SELF-ORGANIZING SYSTEMS (SASO), 2014. **Proceedings** *…IEEE*, 2014.

TANENBAUM, A. **Computer Networks**. 4th. ed. Upper Saddle River, New Jersey: Prentice Hall PTR, 2003. 912 p.

TAURION, Cezar. **Cloud Computing:** Computação em Nuvem: Transformando o mundo da tecnologia da informação. Rio de Janeiro: Brasport, 2009.

VAQUERO, L. M.; MERINO-RODERO, L.; CACERES, J.; LINDNER, M. A Break in the Clouds: Towards a Cloud Definition. In: ACM SIGCOMM Computer Communication Review, 39(1): 50-55, janeiro. **Proceedings** *…ACM*, 2014 2009.

VERGARA, Guilherme Fay. **Arquitetura de um Controlador de Elasticidade para Nuvens Federadas.** 2017. 87 f. Dissertação (Mestrado) - Curso de Instituto de Ciências Exatas, Ciência da Computação, Universidade de Brasília, Brasília, 2017.

WU, Yingjun; TAN, Kian-lee. ChronoStream: Elastic stateful stream computation in the cloud. In: 31st International Conference On Data Engineering , [s.l.], p.0-22. 2015. **Proceedings** *…IEEE.* Disponível em: <http://dx.doi.org/10.1109/icde.2015.7113328>. Acesso em: 16 dezembro 2018.

WU, Song et al. HybridScaler: Handling Bursting Workload for Multi-Tier Web Applications in Cloud. In: 15th International Symposium On Parallel And Distributed Computing (ispdc)**,** [s.l.], p.3-6, 2016. **Proceedings** *…IEEE*,. Disponível em: <http://dx.doi.org/10.1109/ispdc.2016.26>.

ANEXO A -

Tabela 12: Analisando as métricas de Tempo, Energia e Custo para os quatro comportamentos de carga considerando os seguintes cenários: (C1) sem elasticidade; (C2) utilizando *thresholds* fixos; e (C3) utilizando a técnica Live Thresholding. Os valores em verde e vermelho representam, respectivamente, o melhor e o pior resultado com *thresholds* fixos.

Cenários	Parâmetros		Constante			Crescente			Decrescente			Onda		
	T_s	T_i	Tempo	Energia	Custo	Tempo	Energia	Custo	Tempo	Energia	Custo	Tempo	Energia	Custo
C1		-	4283	8542	36585	4319	8618	37221	4410	8798	38799	4363	8700	37958
		30	1888	12382	23377	1818	11936	21700	1891	14056	26580	2286	12696	28566
		35	1903	12444	23681	1873	11824	22146	1906	13882	26459	2304	12256	28238
	70	40	1863	12294	22904	1840	11718	21561	1892	13196	24967	2288	12010	27479
		45	1898	12264	23277	1833	12058	22102	1889	13092	24731	2301	11814	27184
		50	1913	12546	24000	1825	11874	21670	1880	12746	23962	2296	11784	27056
		30	1890	12388	23413	2011	11350	22825	1986	13020	25858	2327	11982	27882
		35	2112	11280	23823	1959	11258	22054	1976	12540	24779	2300	12212	28088
	75	40	2109	11280	23790	1954	11294	22068	1967	12314	24222	2323	12036	27960
		45	2123	11370	24139	2007	11040	22157	1984	12194	24193	2340	11638	27233
		50	2119	11268	23877	2041	11302	23067	1958	11916	23332	2352	11662	27429
		30	2101	11238	23611	2201	10512	23137	1998	12888	25750	2325	11642	27068
		35	2110	11292	23826	2234	10488	23430	1971	12570	24775	2372	11362	26951
C2	80	40	2105	11284	23753	2201	10632	23401	1965	12294	24158	2355	11442	26946
		45	2103	11228	23612	2271	10470	23777	1971	12164	23975	2322	11222	26057
		50	2118	11348	24035	2211	10642	23529	1972	11982	23629	2383	11046	26323
		30	2652	10066	26695	2521	10026	25276	2170	11370	24673	2469	10808	26685
		35	2129	11450	24377	2463	10080	24827	2195	11400	25023	2540	10680	27127
	85	40	2603	9900	25770	2582	10080	26027	2161	11226	24259	2460	10590	26051
		45	2603	9900	25770	2610	10040	26204	2165	11160	24161	2503	10852	27163
		50	2620	9960	26095	2604	10070	26222	2191	11134	24395	2559	10568	27044
		30	2625	9886	25951	2891	9450	29210	2667	10110	26963	2911	9750	28382
		35	2610	9840	25682	2923	9470	27681	2667	10092	26915	2946	9842	28905
	90	40	2649	9900	26225	3015	9494	28624	2663	10036	26726	2922	9724	28414
		45	2626	9834	25824	2945	9480	27919	2643	9900	26166	2914	9750	28412
		50	2653	9954	26408	3000	9540	28620	2638	9840	25958	2904	9600	27878
C3		-	1932	12828	24781	1769	12064	21345	2000	13088	26176	2165	13408	29028

Fonte: elaborado pelo autor.

ANEXO B -

Tabela 13: Analisando as métricas de Recursos (RE), *Speedup* Elástico (ES) e Eficiência Elástica (EE) para os quatro comportamentos de carga considerando os seguintes cenários: (C1) sem elasticidade; (C2) utilizando *thresholds* fixos; e (C3) utilizando a técnica Live Thresholding. Os valores em verde e vermelho representam, respectivamente, o melhor e o pior resultado com *thresholds* fixos.

Cenários	Parâmetros		Constante			Crescente			Decrescente			Onda		
	T_s	T_i	RE	SE	EE	RE	SE	EE	RE	SE	EE	RE	SE	EE
C1		-	2,00	1,00	1,00	2,00	1,00	1,00	2,00	1,00	1,00	2,00	1,00	1,00
		30	6,56	2,27	0,69	6,57	2,36	0,72	7,43	2,26	0,61	5,47	1,87	0,68
		35	6,54	2,25	0,69	6,31	2,29	0,73	7,28	2,25	0,62	5,32	1,86	0,70
	70	40	6,60	2,30	0,70	6,37	2,33	0,73	6,97	2,26	0,65	5,25	1,87	0,71
		45	6,46	2,26	0,70	6,58	2,34	0,71	6,93	2,27	0,66	5,13	1,86	0,73
		50	6,56	2,24	0,68	6,51	2,35	0,72	6,78	2,28	0,67	5,13	1,87	0,73
		30	6,55	2,27	0,69	5,64	2,13	0,76	6,56	2,16	0,66	5,15	1,84	0,71
		35	5,34	2,03	0,76	5,75	2,19	0,76	6,35	2,17	0,68	5,31	1,86	0,70
	75	40	5,35	2,03	0,76	5,78	2,19	0,76	6,26	2,18	0,70	5,18	1,84	0,71
		45	5,36	2,02	0,75	5,50	2,13	0,77	6,15	2,16	0,70	4,97	1,83	0,74
		50	5,32	2,02	0,76	5,54	2,10	0,76	6,09	2,19	0,72	4,96	1,82	0,73
		30	5,35	2,04	0,76	4,78	1,95	0,82	6,45	2,14	0,66	5,01	1,84	0,73
		35	5,35	2,03	0,76	4,69	1,92	0,82	6,38	2,17	0,68	4,79	1,81	0,76
C2	80	40	5,36	2,03	0,76	4,83	1,95	0,81	6,26	2,18	0,70	4,86	1,82	0,75
		45	5,34	2,04	0,76	4,61	1,89	0,82	6,17	2,17	0,70	4,83	1,84	0,76
		50	5,36	2,02	0,75	4,81	1,94	0,81	6,08	2,17	0,71	4,64	1,80	0,78
		30	3,80	1,62	0,85	3,98	1,70	0,85	5,24	1,97	0,75	4,38	1,73	0,79
		35	5,38	2,01	0,75	4,09	1,74	0,85	5,19	1,95	0,75	4,20	1,69	0,80
	85	40	3,80	1,65	0,87	3,90	1,66	0,85	5,19	1,98	0,76	4,30	1,74	0,81
		45	3,80	1,65	0,87	3,85	1,64	0,85	5,15	1,98	0,77	4,24	1,71	0,79
		50	3,80	1,63	0,86	3,87	1,64	0,85	5,08	1,95	0,77	4,13	1,67	0,81
		30	3,77	1,63	0,86	3,06	1,39	0,91	3,79	1,61	0,85	3,35	1,47	0,88
		35	3,77	1,64	0,87	3,24	1,47	0,91	3,78	1,61	0,85	3,34	1,45	0,87
	90	40	3,74	1,62	0,87	3,15	1,42	0,90	3,77	1,61	0,85	3,33	1,47	0,88
		45	3,74	1,63	0,87	3,22	1,45	0,90	3,75	1,62	0,86	3,35	1,47	0,88
		50	3,75	1,61	0,86	3,18	1,43	0,90	3,73	1,62	0,87	3,31	1,47	0,89
C3		-	6,64	2,22	0,67	6,82	2,44	0,72	6,54	2,21	0,67	6,19	2,02	0,65

Fonte: elaborado pelo autor.