Gerenciamento de IaaS utilizando um Dashboard Inteligente baseado em Ontologia e Feedbacks

Allan R. Polachini[1], Gabriel T. Andreazi[1], Júlio C. Estrella[1], Luis H. V. Nakamura[2]

[1]Universidade de São Paulo (USP) – São Carlos, SP – Brasil

{allan.polachini, tomiatti}@usp.br, jcezar@icmc.usp.br

[2]Instituto Federal de São Paulo (IFSP) – Catanduva, SP – Brasil

nakamura@ifsp.edu.br

Abstract. *The accelerated growth in the number of cloud services has made their users develop higher expectations in relation to the quality of service acquired. Considering this fact, this paper purpose to improve the satisfaction of clients still dissatisfied with the service provider, developing a recommendation module based on user feedback. Thus, in this paper a Web platform (Dashboard) was developed that manipulates the IaaSOnt ontology, responsible for reflecting the entire context of the management of an infrastructure as a service (IaaS).*

Resumo. *O crescimento acelerado da quantidade de serviços em nuvem fez com que seus usuários desenvolvessem maiores expectativas em relação à qualidade do serviço adquirido. Considerando esse fato, a finalidade deste artigo é de aprimorar a satisfação dos clientes ainda insatisfeitos com o provedor de serviços, desenvolvendo um módulo recomendador baseado no feedback de usuário. Então, neste artigo foi desenvolvida uma plataforma Web (Dashboard) que manipula a ontologia IaaSOnt, responsável por refletir todo o contexto do gerenciamento de um infraestrutura como serviço (IaaS).*

1. Introdução

Devido ao rápido crescimento do número de serviços em nuvem, seus usuários estão tornando-se cada vez mais exigentes. O primeiro contato dos clientes com as plataformas em nuvem geralmente é realizada por uma interface denominada *Dashboard*. É por meio dessa interface que os clientes gerenciam as suas infraestruturas virtuais de IaaS (*Infrastructure as a Service*) configurando suas instâncias de máquinas virtuais, a capacidade de processamento de cada uma (grande, média ou pequena) e o método de pagamento pelo uso do serviço (retorno sobre investimento ou por demanda). Por esse motivo, propomos o desenvolvimento de um *Dashboard* mais inteligente que auxilie o usuário e identifique características de seu gerenciamento e ainda apoiado por uma ontologia que deve ser atualizada a fim de garantir a qualidade de serviço.

Essa plataforma *web* (*Dashboard*) manipula um banco de dados mapeado de uma ontologia chamada *IaasOnt* [Nakamura 2017], a qual expressa todo o contexto para gerenciamento de uma infraestrutura em nuvem visando ter informações suficientes para a garantia de qualidade de serviço. Além disso, a fim de aprimorar a satisfação dos clientes mais exigentes da plataforma, também foi elaborado um módulo de recomendação com base nos *feedbacks* desses usuários. Assim, o módulo pode recomendar novas configurações na infraestrutura. Esse Mecanismo Inteligente do *Dashboard* pode alterar o número de máquinas virtuais ou escolhe uma nova política financeira, com a finalidade de otimizar o uso de recursos computacionais do provedor ou minimizar os custos de utilização dos clientes. Isso é possível, devido ao retorno de *feedbacks* dos usuários ao mecanismo quando utilizam das funcionalidades administrativas do *Dashboard* e experimentam os resultados da infraestrutura.

2. Trabalhos Relacionados

Em [Fan et al. 2015] foi desenvolvido um novo *framework* de personalização integrado para sistemas em nuvem baseados em SaaS (*Software as a Service*). Esse *framework* melhora significativamente a qualidade, desempenho e interoperabilidade dos serviços. Há vários sistemas com diferentes tipos de requisitos, de modo que não é possível desenvolver uma solução única para todos os sistemas. Porém, esse artigo ofereceu um guia completo e flexível para um serviço de personalização (sendo apenas necessária a adaptação da solução para diferentes requisitos).

Em [Guo et al. 2015] foi realizada uma revisão das principais técnicas que conduzem os serviços de recomendação para serviços em nuvem (com o intuito de auxiliar na escolha do melhor serviço que corresponde com as preferências dos usuários). Sendo o escopo do trabalho a revisão da literatura, os autores focaram na implementação das soluções sugeridas. A dificuldade em definir parâmetros em serviço, devido a dispersão dos dados, foi destacada.

Em [Sun et al. 2016] é introduzido um *framework* para modelagem - ROAR (*Resource Optimization, Allocation and Recommendation System*) com o objetivo de simplificar, otimizar e automatizar a decisão da alocação de recursos em nuvem, visando garantir qualidade de serviço (QoS) para aplicações *web*, em diferentes grupos de servidores. Determinar o melhor grupo de recursos com custo efetivo, com foco em garantir requisitos mínimos de QoS para aplicações *web* sem encarecer o serviço foi um problema encontrado ao desenvolver o *framework*.

Em [Zheng et al. 2017] foi discutido um serviço de recomendação para nuvem baseado em qualidade de serviço. Os autores propuseram uma filtragem colaborativa usando o coeficiente de *Spearman* para recomendar serviços em nuvem. Uma questão que foi observada é que a filtragem colaborativa usando o coeficiente de *Pearson* pode atingir uma classificação mais precisa, porém pode incorrer num ranqueamento não confiável.

3. Mecanismo Inteligente para o Gerenciamento de Máquinas Virtuais

3.1. Descrição do Mecanismo

O Mecanismo Inteligente é uma aplicação que recebe requisições de usuários, processa essas informações e gerencia máquinas virtuais num provedor de serviços. Sua inteligência é baseada em um módulo avaliador que, dependendo da configuração atual da infraestrutura/máquinas virtuais, sugere mudanças para os usuários, visando otimizar recursos do provedor e diminuir gastos dos clientes (com base em políticas financeiras sugeridas por [Nakamura 2017]).

O processamento do mecanismo inteligente ocorre em diferentes módulos presentes na sua configuração. Por meio desses módulos é possível estabelecer um fluxo de dados: primeiro, o Cliente faz uma requisição para o mecanismo (por meio de um *Dashboard*). Depois o "Módulo do Cliente" é responsável por organizar essas informações e enviar para o "Módulo de Persistência dos Dados", que deverá manter a persistência das informações no Banco de Dados modelado para armazenar as informações do contexto da infraestrutura virtualizada. Em seguida, o "Módulo Gerenciador das Máquinas Virtuais" configura a nova infraestrutura do provedor de serviços (*cluster*). Por fim, o "Módulo de Avaliação" recebe a configuração atual do servidor, o histórico de uso para, através de um "Algoritmo Avaliador", recomendar mudanças nos atributos escolhidos pelo usuário, focando na otimização de recursos e consequentemente na redução de gastos financeiros, apenas se o *feedback* do cliente, provido pelo "Módulo de Feedback" for negativo. Caso contrário, considera-se que o cliente está satisfeito com a configuração atual do provedor, não necessitando de uma recomendação.

3.1.1. Diagrama de Componentes

O diagrama de componentes na Figura 1 ilustra o fluxo de dados, considerando inicialmente a requisição do cliente até a configuração da infraestrutura e recomendação do mecanismo inteligente. Uma breve explicação sobre cada um desses módulos é discutido a seguir.

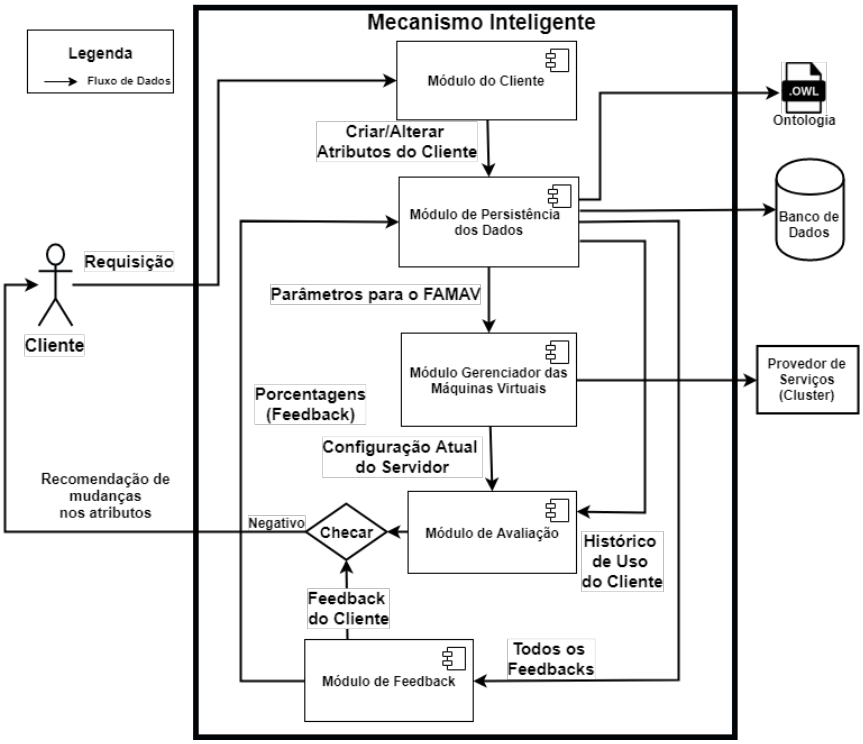


Figura 1. Diagrama de Componentes do Mecanismo Inteligente.

- **Módulo do Cliente**: Esse módulo é responsável pela criação/alteração de contas dos usuários. Além disso, esse módulo responsável por implementar funções, como a função carteira, onde se destinam os crédito para serem gastos na infraestrutura; atribuição de métricas para as políticas financeira;
- **Módulo de Persistência dos Dados**: Esse módulo é responsável por toda a parte de persistência no banco de dados dentro do *Dashboard*. Em colaboração ao projeto de doutorado [Nakamura 2017], o módulo de persistência de dados interage com a ontologia, a qual foi utilizada com os recursos da Web Semântica para realizar consultas, inferências e tomadas de decisões. O *Dashboard* e os demais módulos desenvolvidos neste artigo complementam o trabalho de doutorado que necessitava de uma interface de gerenciamento para a atualização da ontologia e para o controle mais transparente dos recursos virtualizados pelos clientes;
- **Módulo de Gerenciador de Máquinas Virtuais**: Esse módulo interage com a ferramenta FAMaV [Neves et al. 2014] que será mencionada posteriormente. Portanto, as funções desse módulo é o gerenciamento de máquinas virtuais com ações como: ligar, desligar e excluir máquinas virtuais da infraestrutura, etc.;
- **Módulo de Avaliação**: Esse módulo tem a função de analisar o histórico de uso do cliente sobre a infraestrutura e a partir dessa análise, propor uma nova configuração de infraestrutura a qual é baseada nos *feedbacks* do usuário.

3.1.2. Módulo de Avaliação

O Módulo de Avaliação recebe a configuração atual da infraestrutura, oriunda do Módulo Gerenciador das Máquinas Virtuais, e o histórico de uso do último cliente que fez a requisição na plataforma do *Dashboard*, vinda do Módulo de Persistência dos Dados. Ele recomenda para o cliente mudanças nos seus atributos, caso exista uma configuração que otimize o uso de recursos e diminua os gastos do cliente. Essa recomendação é baseada em um Algoritmo Avaliador que, através de uma Árvore Binária, elabora diversos casos de teste que se diferem por modificar ou não modificar determinadas configurações/parâmetros do cliente, para calcular quanto de recurso está sendo utilizado e qual o custo financeiro para o cliente em cada caso. Por fim, o algoritmo recomenda para o cliente a configuração que menos usa recursos do provedor e tem o menor custo financeiro quando o *feedback* do cliente for negativo.

3.1.3. Ontologia

A ontologia *IaaSOnt* [Nakamura 2017] deve ser atualizada a cada alteração na infraestrutura, pois a mesma pertence a outro projeto que visa garantir a qualidade de serviço da infraestrutura virtualizada. Dessa forma, a ontologia *IaaSOnt* é constituída por uma hierarquia de classes que são conectadas/relacionadas por propriedades denominadas Propriedades de Objetos (*Object Properties*). Assim, é possível "navegar" de uma classe para outra utilizando tais propriedades, pois essas propriedades também possuem um valor semântico. O Mecanismo Inteligente consegue acessar e atualizar os dados na ontologia, pois suas classes foram desenvolvidas utilizando a OWLAPI[1], permitindo a navegação e manipulação semântica. Todas essas informações foram mapeadas no banco de dados relacional MySQL e são manipuladas pelo Módulo de Persistência dos Dados.

3.1.4. Integração com o FAMaV

O FAMaV (Ferramenta de Administração de Máquinas Virtuais) [Neves et al. 2014] tem a função de gerenciar máquinas virtuais no provedor de serviços. Por meio da interface CLI (*Command Line Interface*) é possível administrar as VM's, a partir de uma biblioteca de acesso, ao qual os recursos sejam manipulados programaticamente. Dessa forma, essa biblioteca foi referenciada no projeto como o mecanismo responsável por gerenciar as máquina virtuais.

3.1.5. Geração de Carga de Trabalho

O termo *carga de trabalho* é usado para definir a capacidade exigida de um computador para executar uma aplicação. Para os propósitos deste projeto, a carga de trabalho foi gerada com o auxílio da ferramenta *JMeter*[2], a qual é utilizada para testes de carga em serviços oferecidos por sistemas computacionais. A ferramenta oferece possibilidades para manipular diversos tipos de requisições e controladores lógicos tais como o temporizador, que pode ser utilizado na construção de um planejamento de experimentos.

Foi necessário configurar um grupo de *threads* correspondente com o planejamento de experimento, variando o tempo de inicialização (em segundos). As requisições enviadas para o Mecanismo Inteligente foram do tipo *HTTP* e os controladores lógicos foram temporizador Uniforme e Gaussiano.

[1] `https://github.com/owlcs/owlapi/`
[2] `http://jmeter.apache.org/`

3.2. Fluxo do *Feedback*

3.2.1. Recebendo os *Feedbacks* dos Clientes

Para receber o *Feedback* dos clientes "fictícios" e testar o novo módulo foi elaborado um questionário com as perguntas demonstradas na Tabela 1. Neste primeiro teste não foram utilizadas métricas nas perguntas. O Banco de Dados foi alterado de modo a salvar os *Feedbacks* dos clientes "fictícios", as recomendações do algoritmo avaliador e as porcentagens dos *Feedbacks*.

Tabela 1. Questionário com as principais perguntas sobre satisfação.

Questões	Respostas
Tipo de Usuário?	Individual, Comercial
Quanto você usa a infraestrutura?	Pouco, Mais ou Menos, Muito
Satisfação em usar o Dashboard?	Ruim, Justo, Bom, Ótimo
Satisfação das VM's alocadas?	Ruim, Justo, Bom, Ótimo
Satisfação com as políticas financeiras?	Ruim, Justo, Bom, Ótimo
Satisfação com a interface do Dashboard?	Ruim, Justo, Bom, Ótimo

3.2.2. Cálculo das Porcentagens

Dentro do Módulo de *Feedback*, foi desenvolvido o método que recupera todos os *Feedbacks* dos usuários "fictícios" no Banco de Dados, com o objetivo de calcular as porcentagens de cada resposta, de modo a auxiliar o administrador do provedor de serviços a tomar conhecimento de quais módulos do Mecanismo não estão de acordo com as expectativas dos clientes. O calculo de cada porcentagem se baseia na soma de cada resposta similar (Ótimo, Bom, Justo ou Ruim) divido pelo numero total de respostas.

3.2.3. Algoritmo Avaliador

O Algoritmo Avaliador é um método que permanece em execução em paralelo ao Mecanismo, dentro do Módulo de Avaliação. Ele é responsável por receber a configuração atual do provedor de serviços, o histórico de uso da infraestrutura e o *Feedback* do cliente *online*. O algoritmo vai calcular o uso de recursos computacionais e custo financeiro de todos os casos de uma Árvore Binária. Se no *Feedback* do cliente a satisfação com as máquinas virtuais ou políticas financeiras for Ruim e algum dos casos calculados utilizar menos recursos ou ter custo menor para o cliente, então será criada uma recomendação de mudança dos atributos para este usuário.

4. Avaliação de Desempenho do Mecanismo

4.1. Desenho do Experimento

O planejamento de experimentos foi idealizado considerando as distribuições estatísticas: Uniforme e Gaussiana. Para simular os clientes, consideramos 30 *Threads*. Entretanto, diante de testes iniciais houve a necessidade de ajustar o tempo entre as requisições entre 0, 3 e 5 segundos para calibrar o comportamento das distribuições estatísticas. Por essas considerações que foi utilizada a carga sintética que permite uma melhor replicação e alteração de atributos dentro do experimento.

Com o objetivo de aumentar a confiança nos resultados, os experimentos foram repetidos 10 vezes considerando ambas as distribuições, é para todas variações dentro do escopo do planejamento de experimento. O intervalo de confiança obtido foi de 95%. O qual todo o planejamento de experimentos tem como foco na avaliação de desempenho foi baseado em [Jain 1991]. No planejamento de experimentos tem-se como principais objetivos, determinar as seguintes informações:

- O tempo de chegada das requisições, o qual é medido o tempo uma requisição será gerada pelo *JMeter*, e entregue ao *Dashboard*.
- O tempo de processamento individual de cada módulo do *Dashboard* em relação a requisição, cujo objetivo é identificar gargalos no sistema, apontando qual sub-módulo apresenta maior *overhead* de processamento.
- O tempo de resposta total das requisições, considerando desde a requisição feita pelo *Jmeter*, até a resposta HTTP de retorno, assim objetivamos medir o tempo de variação referentes ao número de *Threads*(clientes), pelos respectivos tempos de variação entre as requisições em ambas as distribuições.

4.2. Teste Funcional do Módulo de *Feedback* e Algoritmo Avaliador

A versão atual do módulo de *Feedback* está funcional. Para testar sua funcionalidade apenas foram realizados testes (Tabela 2), criando *Feedbacks* de usuários "fictícios" na base de dados, com o objetivo de analisar as porcentagens obtidas pelo Módulo de *Feedback* e checar a funcionalidade do mesmo. Para que seja avaliado o quanto esse módulo melhorou a eficácia da plataforma, novos testes com usuários reais serão efetuados, considerando a utilização da Plataforma Brasil, uma vez que é necessária a aprovação de um comitê de ética para testes com usuários reais. Além disso, foram realizados alguns testes funcionais no Algoritmo Avaliador, com clientes "fictícios" populados na base de dados pelo autor. Cada teste possui um número diferente de máquinas virtuais e capacidades que são baseadas nas instâncias de desempenho com capacidade da Amazon[3]:

- ***Small:*** VM com capacidade pequena, utilizando 1 vCPU e com 2Gb de RAM.
- ***Medium:*** VM com capacidade média, utilizando 2 vCPUs com 4Gb de RAM.
- ***Large:*** VM com capacidade grande, utilizando 4 vCPUs com 8Gb de RAM.

Tabela 2. Teste com *Feedbacks* de Clientes "Fictícios".

ID	Cliente	Categoria	Dashboard	Máquina Virtual	Política Financeira	Interface
1	Individual	Pouco	Ruim	Ruim	Ruim	Ruim
2	Comercial	Mais ou Menos	Ótimo	Ótimo	Ótimo	Ótimo
3	Individual	Muito	Ruim	Ruim	Ruim	Ruim
4	Individual	Pouco	Ótimo	Ótimo	Ótimo	Ótimo
5	Individual	Muito	Bom	Bom	Bom	Bom
6	Individual	Muito	Ruim	Ruim	Ruim	Ruim
7	Comercial	Pouco	Ótimo	Ótimo	Ótimo	Ótimo
8	Comercial	Muito	Justo	Justo	Justo	Justo
9	Comercial	Muito	Justo	Justo	Justo	Justo
10	Comercial	Pouco	Ruim	Ruim	Ruim	Ruim

4.3. Configuração do Ambiente

Para a execução dos experimentos com foco na avaliação de desempenho, considerou-se uma máquina virtual que hospeda o *Dashboard* com o mecanismo inteligente para receber as requisições geradas pelo *JMeter*. As configurações desse servidor são listadas na Tabela 3.

Tabela 3. Principais configurações de *Hardware* do Servidor *Dashboard*.

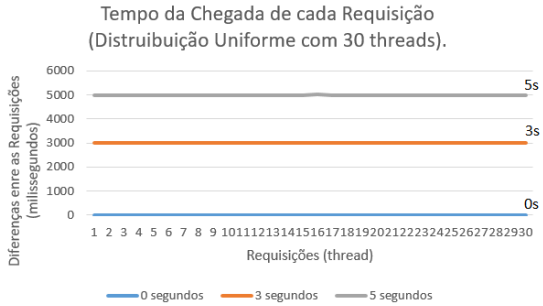
Sistema Operacional	Ubuntu 16.04.2
CPU 64 GB RAM DDR3 Kingston	Intel® Xeon™ CPU E5-2620 2.00GHz heightMemória
Switch	Switch 3Com 2920-SFP Plus
Java Heap Space	-Xms1024m -Xmx8192m
Número de Conexões permitidas MSQL	10000

Assim que as requisições chegam ao servidor são escalonadas para o Cluster Cosmos (vide infraestrutura no *website*)[4], que hospeda as VMs alocadas por meio do *Dashboard*.

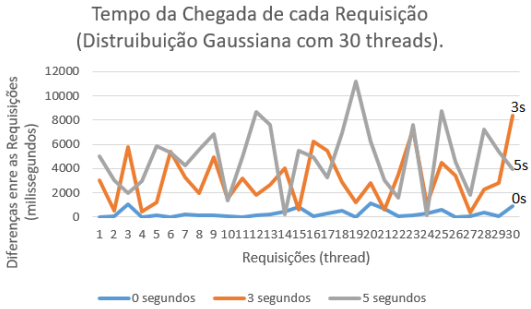
[3] `https://aws.amazon.com/pt/ec2/instance-types/`
[4] `http://infra.lasdpc.icmc.usp.br/cosmos/`

4.4. Análise dos Resultados

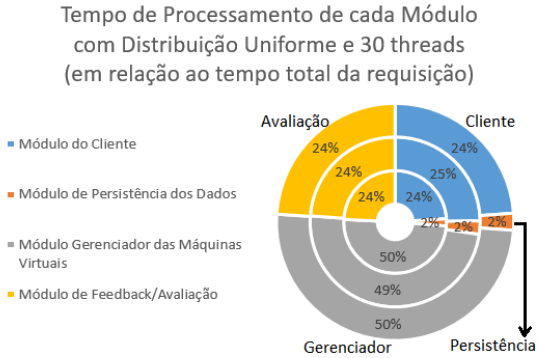
Os resultados foram coletados após a execução das 30 requisições (*threads*) e duas distribuições foram levadas em consideração nos experimentos: **Uniforme** e **Gaussiana**. Também foi alternado o intervalo de tempo entre as requisições (0, 3 e 5 segundos).



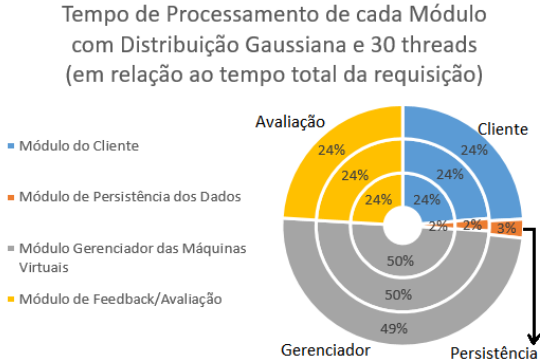
(a) Tempo da Chegada de cada Requisição.



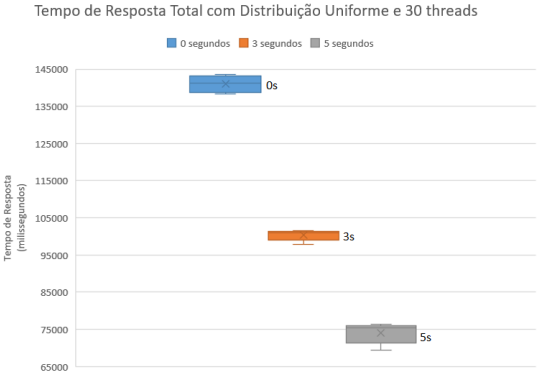
(b) Tempo da Chegada de cada Requisição.



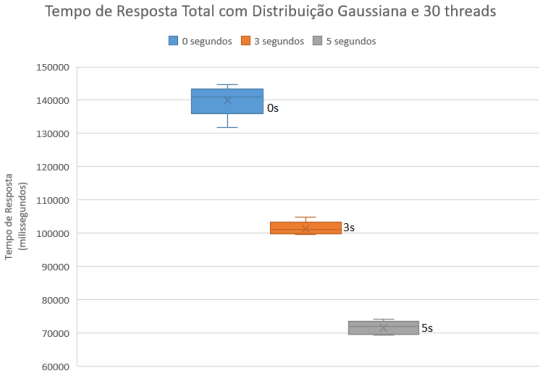
(c) Tempo de Processamento de cada Módulo.



(d) Tempo de Processamento de cada Módulo.



(e) Tempo de Resposta Total da Requisição.



(f) Tempo de Resposta Total da Requisição.

Figura 2. Resultados dos Experimentos de Desempenho.

Os resultados de todos os experimentos estão condensados na Figura 2, onde é possível notar que o tempo de resposta total demonstrado variou conforme o intervalo de tempo entre as requisições. Portanto, quanto maior o número de requisições ao mecanismo, dado um menor período de tempo, maior será o tempo de resposta. Outro fato observado durantes os testes foi que o módulo de alocação das máquinas virtuais, possui o maior gargalo entre os tempos de resposta dentre os módulos. O motivo disso se deve em razão as requisições dos usuários que quando chegam ao módulo, são sincronizadas a fim de evitar o acesso concorrente a região crítica, a qual consulta a disponibilidade de uma máquina virtual.

O Teste Funcional no Módulo de *Feedback* resultou em satisfações dos clientes que estão divididas em: 40% Ruim, 20% Justo, 10% Bom e 30% Ótimo, esses resultados foram calculados pelo Módulo de *Feedback* em relação ao uso do *Dashboard* (com base nos testes da Tabela 2).

5. Conclusão

Após um estudo acerca de outros trabalhos e plataformas que envolvem o gerenciamento de uma infraestrutura como um serviço em nuvem (IaaS), notou-se que há um diferencial do *Dashboard* com um Mecanismo Inteligente em relação aos apresentados na literatura, pois ele além de comunicar-se com uma ontologia, também é capaz de recomendar para os clientes novas configurações da infraestrutura, otimizando desta forma o uso de recursos computacionais e diminuindo os custos financeiros dos usuários.

Por meio dos resultados obtidos, concluímos que é possível diminuir o tempo médio de resposta das requisições ao mecanismo. Para trabalhos futuros, abrem-se possibilidades de novos estudos com o intuito de aperfeiçoar a plataforma proposta, reformular o módulo responsável por alocar as máquinas virtuais e a qualidade do *feedback* ao usuário.

AGRADECIMENTOS

Os autores agradecem a FAPESP (processos número: 11/12670-5 e 15/26504-0) pelo apoio financeiro concedido para a elaboração deste trabalho.

Referências

Fan, H., Hussain, F., Younas, M., and Hussain, O. (2015). An integrated personalization framework for saas-based cloud services. *Future Generation Computer Systems*, 53:157–173. cited By 7.

Guo, L., Zheng, X., Ding, C., Mu, D., and Li, Z. (2015). Cloud service recommendation: State of the art and research challenges. pages 761–764. cited By 4.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley professional computing. Wiley.

Nakamura, L. H. V. (2017). *Mecanismos de Autoconfiguração e Auto-otimização para arquiteturas virtualizadas que visam a provisão de qualidade de serviço*. Tese de doutorado, ICMC - USP - Universidade de São Paulo, São Carlos, SP.

Neves, Y. T., Nakamura, L. H. V., Prado, P. F., and Santana, M. J. (2014). Famav: Análise comparativa entre ferramentas de gerenciamento de máquinas virtuais. In *WSCAD 2014 - XV Simpósio em Sistemas Computacionais de Alto Desempenho*, volume 15, pages 254–259.

Sun, Y., White, J., Eade, S., and Schmidt, D. C. (2016). Roar: A qos-oriented modeling framework for automated cloud resource allocation and optimization. *Journal of Systems and Software*, 116(Supplement C):146 – 161.

Zheng, X., Xu, L. D., and Chai, S. (2017). Qos recommendation in cloud services. *IEEE Access*, 5:5171–5177.