Uma arquitetura de balanceamento de carga web escalável para nuvens Eucalyptus

Pedro Roger Magalhães Vasconcelos [1]
Gisele Azevedo de Araújo Freitas [1]

Resumo: Computação em nuvem provê acesso a um conjunto de recursos como máquinas virtuais, armazenamento e rede como serviços. Neste contexto, o balanceamento de carga é importante para computação em nuvem, pois, permite a distribuição de carga de trabalho entre vários nós de processamento que podem ser provisionados sob demanda. Este artigo descreve um modelo para o balanceamento de carga web dinâmico com base em limites sobre o estado de execução dos nós, monitorados e escalados dinamicamente em função dos recursos: uso do processador, uso de memória e número de conexões. O trabalho demonstrou os benefícios do balanceamento de carga na Nuvem que é capaz de lidar com cargas repentinas, entregando recursos sob demanda e mantendo melhor utilização de recursos e infra-estrutura.

***Palavras-chave*:** Balanceamento de carga. Computação em nuvem. Escalabilidade.

Abstract: *Cloud computing provides access to a set of resources such as virtual machines, storage and network as services. In this context, load balancing is important for cloud computing because it allows the distribution of workload across multiple processing nodes that can be provisioned on demand. This article describes a model for dynamic web load balancing based on limits of the state of execution of the nodes, monitored and dynamically scaled according to resources: processor usage, memory usage and number of connections. The work has demonstrated the benefits of load balancing in the Cloud that is able to handle sudden loads, delivering on-demand resources and maintaining better resource and infrastructure utilization.*

***Keywords*:** *Cloud computing. Elasticity. Scalability.*

1 Introdução

A computação em nuvem emergiu como um novo paradigma de computação distribuída em larga escala que possibilitou transferir o poder de computação e os dados dos desktops e dispositivos móveis para os grandes datacenters [1]. Possui a capacidade de aproveitar o poder da Internet para a utilização de recursos disponíveis remotamente, provendo soluções rentáveis para a maioria dos requisitos reais [2]. É um modelo para permitir um acesso ubíquo, prático e sob demanda por rede a um conjunto compartilhado de de recursos computacionais (como redes, servidores, armazenamento e serviços) que podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços.

A computação em nuvem é composta por cinco características essenciais, três modelos de serviço e quatro modelos de implantação [3]. Uma das características é a elasticidade, na qual os recursos são provisionados em várias formas e quantidades provendo escalabilidade aos sistemas. Aos usuários, os recursos aparentam serem ilimitados.

O balanceamento de carga é um mecanismo que distribui o excesso de carga de trabalho uniformemente entre todos os nós. É utilizada para alcançar uma alta satisfação de usuários e melhor taxa de utilização dos recursos, certificando-se que nenhum nó esteja sobrecarregado e contribuindo para a melhora do desempenho global do sistema [4]. Técnicas de balanceamento de carga podem prover utilização ótima dos recursos disponíveis, proteção a falhas, escalabilidade, redução tempo de resposta, além de evitar gargalos e excesso de provisionamento [5].

[1]Programa de Pós-Graduação em Engenharia Elétrica e de Computação), UFC, Campus Sobral - Sobral (CE) - Brasil
{pedro.roger@alu.ufc.br, gisele@lia.ufc.br}

A Computação utilitária pode ser definida como o provisionamento de recursos de computação e armazenamento como um serviço medido, semelhantes aos serviços públicos tradicionais como eletricidade, telefonia e água [6]. A cobrança por tais serviços é feita baseada no total de recursos utilizados ao invés de uma taxa fixa. Ao adotar tais modelos de provimento de serviços, também chamados pay-as-you-go, as empresas são capazes de evitar um grande investimento inicial, pagando apenas o custo de operação de suas funcionalidades [7].

Este trabalho propõe uma arquitetura de balanceamento de carga de aplicações web executada em uma nuvem privada Eucalyptus, na qual os estados das instâncias servidoras de aplicações e a utilização de recursos são constantemente monitorados. Este monitoramento é feito por um sofware que baseado nos dados obtidos pode otimizar o sistema iniciando novas instâncias, caso as instâncias servidoras estejam sobrecarregadas, ou removendo instâncias que estejam ociosas.

O Eucalyptus [8] (um acrônimo para Elastic Utility Computing Architecture for Linking Your Program To Useful Systems) é um framework de código aberto para a implantação de nuvens privadas de infraestrutura. Ele permite que os seus usuários executem e controlem máquinas virtuais utilizando uma dada infraestrutura física. Eucalyptus provê uma API baseada na Amazon EC2 [9] e Amazon S3 [10], permitindo compatibilidade com esses importantes serviços [11].

A plataforma Eucalyptus possui uma arquitetura modular e hierárquica, onde cada módulo é implementado como um serviço web a fim de alcançar mais compatibilidade e segurança. Os seis módulos que compõem a plataforma Eucalyptus são: Cloud Controller (CLC), Scalable Object Storage (SOS), Cluster Controller (CC), Storage Controller (SC), Node Controller (NC) e um módulo opcional chamado VMware Broker (VB) [12].

O monitoramento das instâncias é realizado através do sistema de monitoramento de código aberto Zabbix [13]. Essa ferramenta é um sistema de nível coorporativo designado para monitorar a disponibilidade e o desempenho de infraestruturas computacionais e de telecomunicações. Um agente Zabbix é instalado em cada instância e realiza o monitoramento dos recursos de modo passivo. Através de uma API o software controlador desenvolvido resgata o estado de utilização dos recursos provisionados a cada instância.

O balanceamento de carga das requisições é realizado com o servidor web Nginx [14], que é um servidor web leve e flexível que também pode atuar como proxy reverso, balanceador de carga e servidor de cache.

O software controlador mede constantemente o uso de três recursos das instâncias: utilização do processador, consumo de memória e quantidade de conexões.

Testes de desempenho foram realizados com o software Apache JMeter [15], nos quais uma carga de trabalho foi gerada para uma aplicação PHP servida na nuvem. Durante o teste, o sistema de monitoramento detectou uma sobrecarga nas instâncias iniciais e baseado nas métricas de monitoramento foi ajustando os recursos alocados para o processamento das requisições. O trabalho analisa o desempenho total do cluster quando monitorado sob cada uma das métricas citadas.

Este trabalho está organizado da maneira a seguir: A seção 2 apresenta uma fundamentação teórica necessária ao entendimento do trabalho. A seção 3 descreve trabalhos relacionados com o tema. A seção 4 apresenta e detalha a arquitetura de balanceamento de carga proposta. A seção 5 descreve os experimentos realizados e a análise dos resultados. Por fim, a conclusão do trabalho e as perspectivas futuras são descritas na seção 6.

2 Balanceamento de Carga e Computação em Nuvem

O balanceamento de carga é o processo de melhorar o desempenho de um sistema paralelo ou distribuído através da redistribuição de carga entre os processadores [16].

Alguns dos objetivos de um algoritmo de balanceamento de carga são [17]:

- Alcançar uma melhoria no desempenho global do sistema a um custo razoável.
- Tratar todas as requisições ao sistema de forma igualitária independente de sua origem.
- Possuir tolerância a falhas.

- Possuir a habilidade de modificar-se de acordo com qualquer alterações ou expandir a configuração do sistema distribuído.
- Manter a estabilidade do sistema.

O balanceamento de carga dinâmico pode ser realizado em abordagens diferentes: distribuído e não distribuído. Na abordagem distribuída, o algoritmo de balanceamento é executado em todos os nós presentes no sistema e a tarefa de balanceamento é distribuída entre eles [18]. Este tipo de balanceamento geralmente utiliza mais mensagens que os não distribuídos, pelo fato de que cada nó do sistema necessita interagir com todos os outros nós. Um benefício dessa abordagem é que mesmo que um ou mais nós do sistema fiquem indisponíveis, não ocorrerá o comprometimento total do processo de balanceamento, porém, poderá haver degradação de desempenho. O balanceamento não distribuído pode possuir duas formas: centralizado e não centralizado. Na primeira, o algoritmo de balanceamento de carga é executado em um único nó do sistema (nó central) e requer menos mensagens para alcançar a decisão de balanceamento. Isso acontece pelo fato de que os outros nós no sistema não interagem uns com os outros, eles apenas interagem com o nó central. Por outro lado, algoritmos de balancemento centralizado colocam em perigo o desempenho do sistema caso ocorra indisponibilidade do nó central. Há também a possibilidade de que esse nó cause um gargalo se ele for congestionado com mensagens de todos os outros nós do sistema[18]. Um estudo realizado por tem demonstrado que o balanceamento de carga centralizado é mais adequado a sistema de pequeno porta (menos de 100 nós) do que qualquer outro método de controle.

Nuvens de Infraestrutura como Serviço (IaaS) devem provisionar dinamicamente seus recursos de modo a aumentar a escalabilidade dos sistemas em momentos de pico e diminuir a escalabilidade nos momentos de subutilização, a fim de maximizar a eficiência de uso de recursos e/ou minimizar os custos associados. A maioria dos autores concorda que uma nuvem consiste de aglomerados de computadores distribuídos que fornecem recursos ou serviços por demanda a uma rede com a escala e confiabilidade de um centro de dados [19]. Noções de virtualização de recursos e grade de computação podem ser utilizadas para que estes aglomerados forneçam instâncias por demanda.

Atualmente, combinar a tecnologia de virtualização com a nuvem computacional tem sido uma tendência. Um plataforma em nuvem é um enorme recipiente que pode mesclar diferentes tecnologias de virtualização e aplicações em conjunto através da Internet. Considerando o elevado custo na gerência de computadores físicos ou servidores, cada vez mais pessoas e empresas voltam sua atenção para os serviços em nuvem que podem ser oferecidos através da Internet. Os usuários finais tendem a usar recursos virtuais para processamento de dados em larga escala, computação, renderização de animações e armazenamento de dados. Estes serviços dependem de redes de altas taxas de transmissão, grande capacidade de processamento e armazenamento, projetos de interações mais adequados e alta disponibilidade de serviços. Prover serviços de segurança consistentes em nuvens de infraestrutura é de fundamental importância devido à natureza multi-locatário e multi-fornecedor das plataformas em nuvem [20].

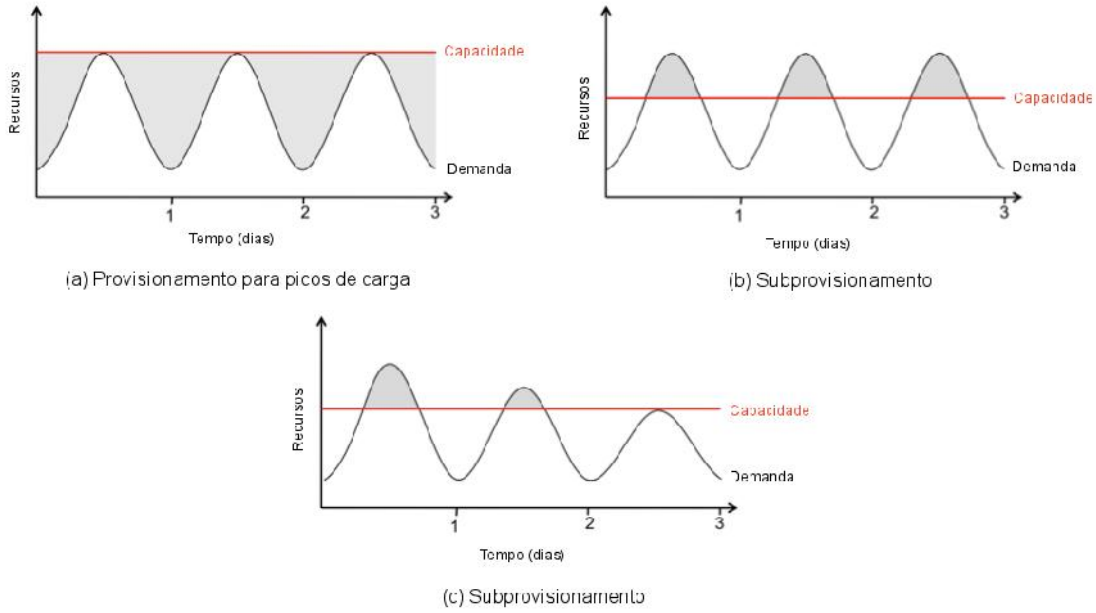
Portanto, a combinação de plataforma de infraestrutura de nuvem e virtualização é um ponto-chave na prestação de serviços de clientes. Com o objetivo de oferecer aos clientes uma experiência eficiente e confiável quando eles exigem uma máquina virtual a partir da internet, precisam do apoio de instalações de hardware para oferecer uma infraestrutura essencial, uma velocidade alta e ambiente de rede segura para construir uma conexão entre clientes e serviços em nuvem, um conjunto de recursos disponíveis para fornecer recursos virtuais reais.

Várias organizações estão migrando para serviços de computação em nuvem para diminuição de riscos e melhor disponibilidade dos negócios. Em acessos sob demanda os usuários requisitam serviços de infraestrutura para acesso imediato e por um curto intervalo de tempo, a cobrança dos serviços é feita baseada nessa duração. As requisições são servidas pela alocação de máquinas virtuais servidoras de aplicações que são alocadas no hardware subjacente [21].

A escalabilidade permite que os usuários provisionem recursos suficientes para os períodos de picos de carga e evitem a subutilização de recursos em momentos de não pico [1]. A Figura 1(a) ilustra que mesmo com a antecipação do provisionamento para a carga de pico, sem a elasticidade há desperdício de recursos (área sombreada) durante períodos de não pico. A Figura 1(b) ilustra o efeito do subprovisionamento quando o sistema é submetido a cargas de pico e a Figura 1(c) ilustra outro efeito do subprovisionamento de recursos que é o comprometimento do experiência dos usuários ao acessarem os sistemas. Os usuários abandonam o site permanentemente

após experenciarem mau serviço [1].

Figura 1: Provisionamento para picos de carga e subprovisionamento de recursos [1]



3 Trabalhos relacionados

Balanceamento de carga é um tema de frequentes trabalhos de pesquisa que têm como objetivo uma melhor distribuição dos recursos de forma a melhor atender suas cargas de trabalho.

As técnicas de balanceamento de carga em nuvens existentes, consideram vários parâmetros como desempenho, tempo de resposta, escalabilidade, utilização de recursos, tolerância a falhas, tempo de migração, sobrecargas, consumo de energia e emissão de carbono[5].

Katyal [22] discutiu vários esquemas de balanceamento de carga, concluindo que o balanceamento de carga estático possui ambientes de simulação e monitoramento mais simples mas falham na modelagem da natureza heterogênea da nuvem. Por outro lado, o balanceamento de carga dinâmico é difícil de simular mas é melhor adaptado aos ambientes heterogêneos de computação em nuvem.

Em [23] os autores apresentam uma estratégia de balanceamento de carga dada por um algoritmo genético que mantém um histórico do estado de execução das instâncias, conseguindo uma predição da influência que a ação de escalonamento terá no sistema, escolhendo a ação que alcance o melhor balanceamento de carga e reduza a flutuação do escalonamento. Essa estratégia resolveu o problema de desbalanceamento de carga e alto custo de migração dos algoritmos tradicionais.

[24] propuseram uma arquitetura de escalabilidade baseada em um monitoramento simples do estado das máquinas virtuais através de túneis SSH, a fim de obter um modelo de criação de instâncias de custos otimizados. Eles desenvolveram um servidor de monitoramento que possui dois componentes, um componente central que coleta o estado de execução das instâncias e realiza comandos de escalabilidade no controlador da nuvem, além de um front-end web para interação do administrador e geração de gráficos.

Em [25] foi proposta uma arquitetura que utiliza computação em nuvem em conjunto com princípios de computação autonômica para o provimento de um ambiente elástico. Nos experimentos foram utilizadas uma nuvem privada OpenNebula, cargas de trabalho com o software HTTPERF e um sistema de multiplicação de matrizes em Java. A métrica utilizada para disparar ações de elasticidade foi a média do percentual de utilização dos processadores das máquinas virtuais.

Os autores de [21] propuseram um modelo de alocação eficiente de máquinas virtuais a fim de diminuir o tempo de alocação e otimizar a utilização de recursos para acesso sob demanda em uma nuvem OpenNebula. Os seus experimentos mostraram que o algoritmo proposto pode melhorar a utilização de recursos para servir um maior número de requisições de recursos sem comprometer o tempo de alocação. Quando uma requisição de máquina virtual chega ao agendador, ele calcula a razão entre a especificação da máquina virtual requisitada pela especificação dos host físico. Esse valor é calculado para cada um dos hosts físicos que estão organizados em uma árvore de pesquisa binária começando do nó raiz até os nós folha, até que se encontre o melhor ajuste.

4 Arquitetura do modelo de balanceamento de carga

Eucalyptus possui uma arquitetura modular e hierárquica onde cada módulo é implementado como o serviço web a fim de alcançar mais compatibilidade e segurança. Os seis módulos que compõem a plataforma Eucalyptus são: Cloud Controller (CLC), Scalable Object Storage (SOS), Cluster Controller (CC), Storage Controller (SC), Node Controller (NC) e um módulo opcional chamado VMware Broker (VB) (Figura 2) [12].

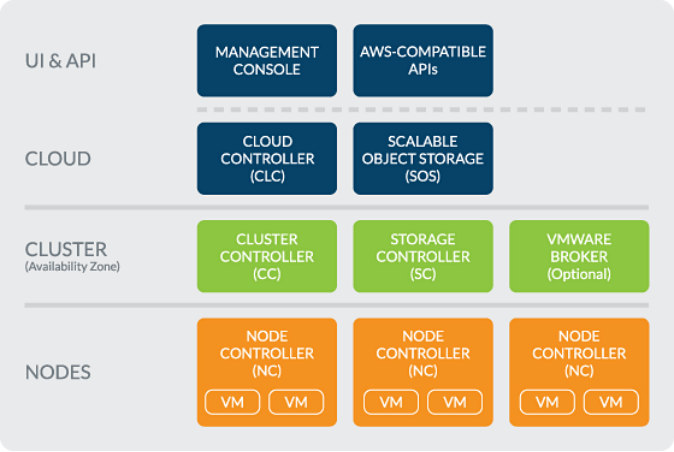


Figura 2: Componentes Eucalyptus.

- **Cloud Controler:** O Cloud Controler (CLC) é o ponto de entrada da nuvem para os seus administradores, desenvolvedores e usuários finais. O CLC consulta informações de outros componentes, realiza decisões de agendamento e realiza requisições ao Cluster Controler. Como uma interface para a plataforma de gerenciamento, o CLC é responsável por expor e gerenciar os recursos virtualizados subjacentes (servidores, rede e armazenamento). Apenas um CLC pode existir por nuvem.

- **Scalable Object Storage:** É um serviço Eucalyptus similar ao AWS Simple Storage Service (S3). O Scalable Object Storage (SOS) é um serviço que permite aos administradores da infraestrutura terem a flexibilidade de implementar armazenamento escalável para nuvens de larga escala. Eucalyptus também provê um sistema de armazenamento básico, chamado Walrus, mais adequado para avaliações e nuvens de pequena escala.

- **Cluster Controller:** Um cluster é equivalente a uma zona de disponibilidade AWS, e um uma nuvem Eucalyptus pode possuir vários clusters. O Cluster Controller (CC) age como um *front end* para um cluster dentro de uma nuvem Eucalyptus e se comunica com o Storage Controller e o Node Controller. O CC gerencia a execução de instâncias e Acordos de Níveis de Serviço (SLA) por cluster.

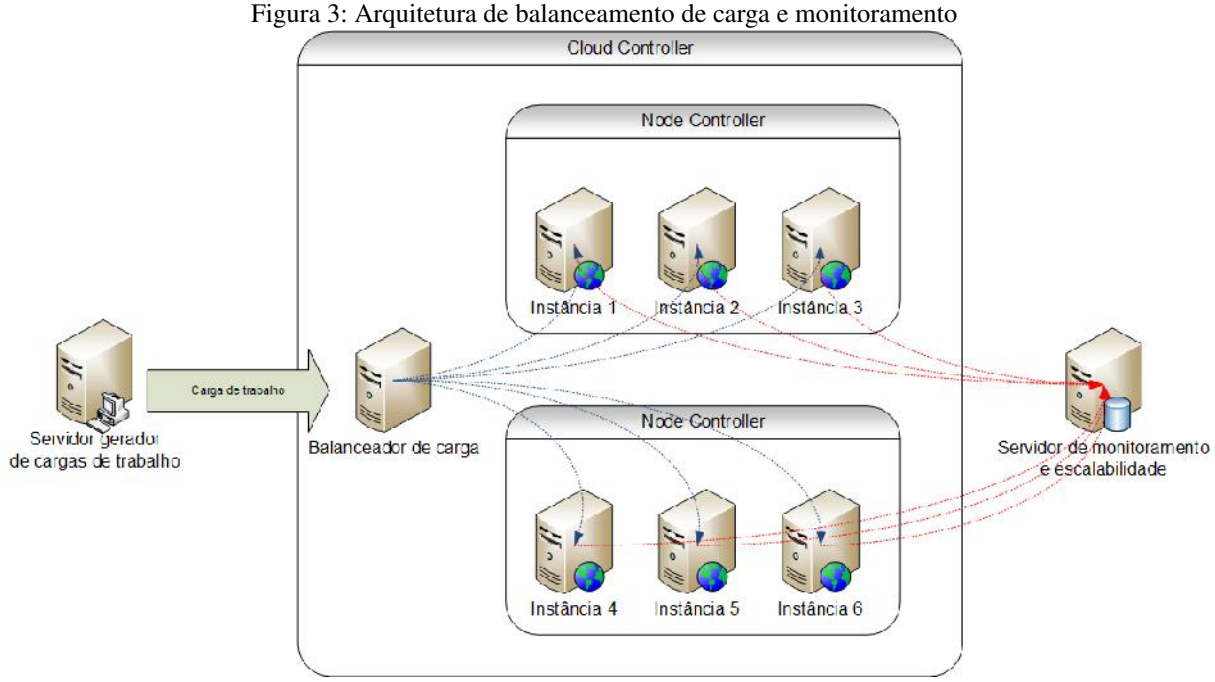
- **Storage Controller:** É um componente escrito em java e é equivalente ao AWS Elastic Block Store (EBS). O Storage Controller (SC) realiza comunicações com o Cluster Controller e Node Controller e gerencia volumes e *snapshots* de dispositivos de bloco para as instâncias dentro de um cluster. O SC realiza interfaces com vários sistemas de armazenamento, como NFS, iSCSI e SAN.

- **VMware Broker:** É um módulo opcional que permite ao Eucalyptus lançar máquinas virtuais em uma infraestrutura VMware. O VMware Broker (VB) media toda a comunicação entre o Cluster Controler e os hipervisores VMware (ESX e ESXi).
- **Node Controller:** O Node Controller (NC) é executado nas máquinas que hospedam máquinas virtuais. O NC controla atividades das máquinas virtuais como execução, inspeção e término de instâncias. Ele também recupera e mantém um cache local de imagens de instâncias.

Uma instância é lançada com a finalidade de prover o balanceamento de carga das requisições. Essa instância executa o servidor Nginx que recebe as conexões vindas do exterior da nuvem e balanceia as requisições de forma igualitária, utilizando um algoritmo Round-robin, entre as instâncias servidoras.

O diagrama apresentado na Figura 3 ilustra a arquitetura de balanceamento.

Figura 3: Arquitetura de balanceamento de carga e monitoramento



Um script controlador foi desenvolvido e executado no servidor de monitoramento. Tal script utiliza conexões seguras, através da autenticação por chaves do protocolo SSH, para comunicar-se com o Cloud Controller. As instâncias são manipuladas e os dados obtidos através da execução de comandos do pacote de ferramentas eucatools, como os comandos euca-describe-instances, euca-run-instances e euca-terminate-instances. Inicialmente o controlador recupera a lista de instâncias servidoras em execução. A seguir passa a monitorar a utilização do recurso selecionado em tais instâncias e caso a média de utilização deste recurso ultrapasse determinado limiar superior, a ordem de iniciar uma nova instância é emitida ao Cloud Controller. Caso a utilização do recurso esteja abaixo de um limite inferior, significa a subutilização dos recursos provisionados e uma ordem de remoção de uma instância é emitida. Se a utilização destes recursos se encontrar entre os dois limites, significa que a carga total do sistema está aceitável e que ainda possui uma margem de recursos livres.

5 Experimentos e análise dos resultados

Para avaliar a arquitetura proposta um experimento foi realizado utilizando uma nuvem na plataforma Eucalyptus versão 3.4.2 executada em um sistema operacional CentOS Linux 6.5 64 bits. Os servidores utilizados foram duas IBM BladeCenter HS21 com dois processadores Intel Xeon E5-2620 de 2GHz (6 núcleos cada), 15MB

de cache L2 por núcleo e 48GB de RAM, conectadas através de uma rede Gigabit Ethernet. As instâcias utilizadas foram criadas a partir de imagens EMI do Debian 7 64 bits distribuídas pelo aplicativo eucastore do próprio Eucalyptus. Todas as instâncias são do tipo "m1.small"que aloca 1 VCPU, 256 MB de memória e 5 GB de disco. Uma aplicação PHP foi desenvolvida e instalada nas instâncias servidoras, nas quais rotinas são executadas a fim de fazer uso exaustivo de processador e memória quando sob alta concorrência. A aplicação calcula uma série de Fibonacci de tamanho variável e aloca uma quantidade de memória durante sua execução. A carga de trabalho é gerada por um servidor externo à nuvem, simulando clientes reais acessando o sistema através da Internet. Utilizamos o servidor web Nginx v1.6.0 como balanceador de carga e o aplicativo Apache JMeter v2.11 para simular uma concorrência de 100 usuários realizando 40 iterações cada, totalizando 4000 requisições ao script. O sistema de monitoramento Zabbix v2.0.5 e o software controlador estão instalados em uma instância na mesma Zona de Disponibilidade que as outras.

O software controlador da escalabilidade monitora continuamente o consumo dos recursos das instâncias e toma as decições baseadas nos dados obtidos conforme a Tabela 1.

Tabela 1: Métricas de escalabilidade

Métrica	Escala para cima	Escala para baixo
Utilização de CPU	$> 80\%$	$< 30\%$
Utilização de memória	$> 60\%$	$< 40\%$
Número de conexões	> 80	< 30

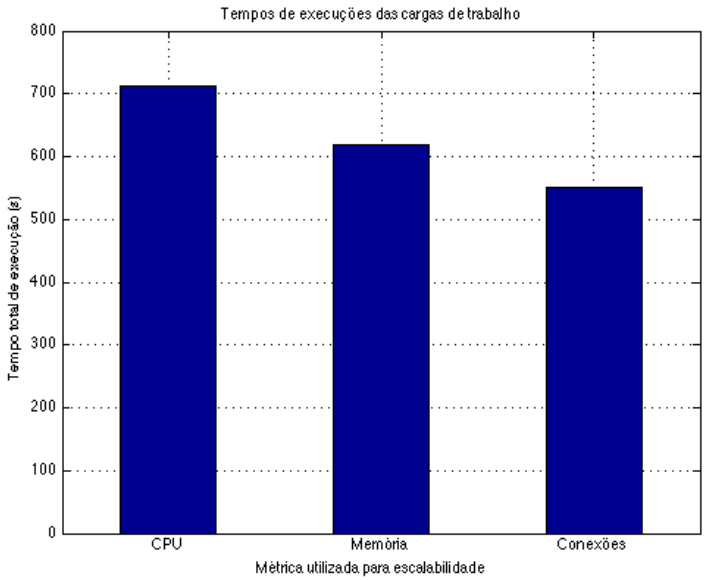
O intervalo entre as coletas de dados de monitoramento foi de 5 segundos e o período de espera antes de se enviar requisições a uma nova instância foi de 60 segundos. Esse intervalo é necessário devido ao tempo gasto para o provisionamento dos recursos da instância bem como seu processo de boot.

5.1 Análise dos resultados

O objetivo do experimento é observar o desempenho do cluster quando monitorado sob diferentes métricas. Assim, pode-se observar se a criação e destruição de novas instâncias ocorre com mais brevidade e menos flutuações quando sob efeito do monitoramento de uma ou outra métrica.

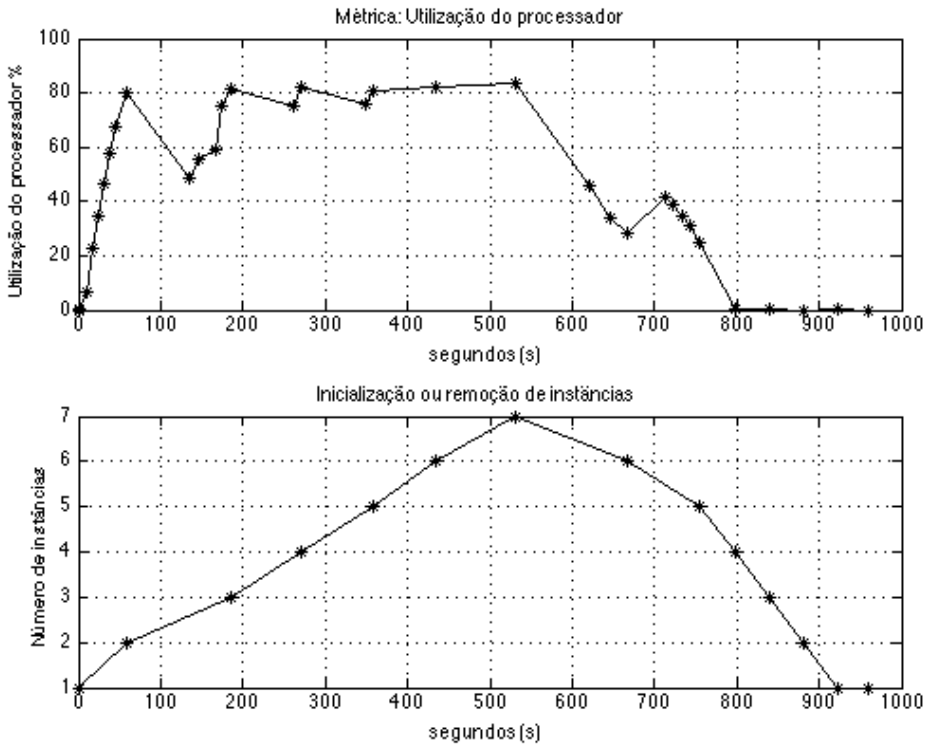
A Figura 4 demonstra o tempo total de execução das requisições sob cada métrica de escalabilidade.

Figura 4: Tempo de execução dos testes por cada métrica de escalabilidade



A Figura 5 demonstra a média de utilização do processador de todas as instâncias em execução e o comportamento da escalabilidade.

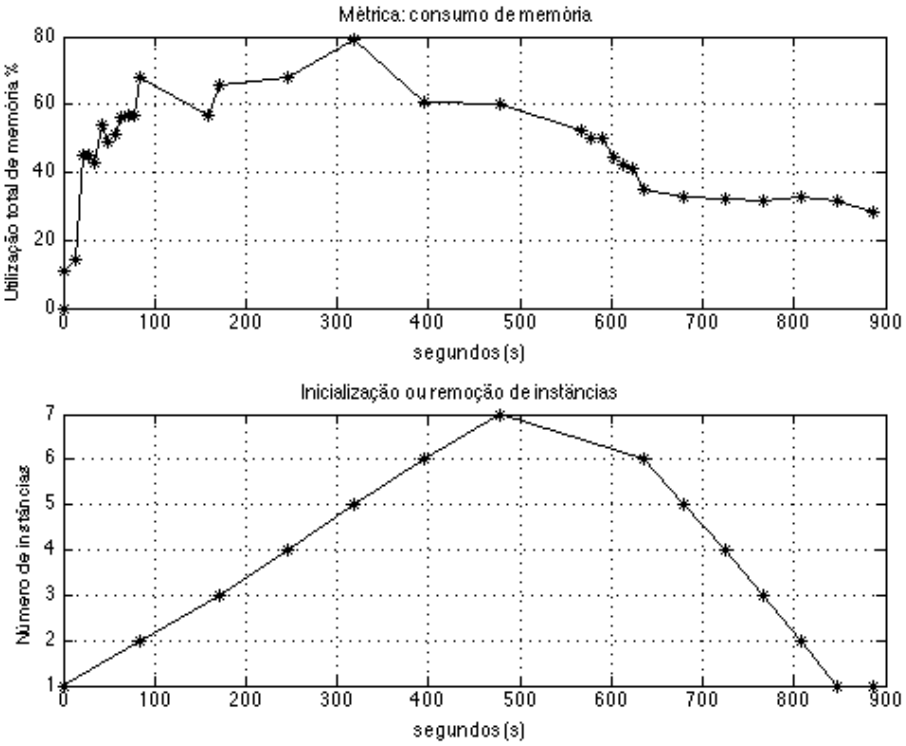
Figura 5: Utilização do processador



A carga do processador aumente a medida que novas requisições são enviadas às instâncias pelo balanceador de carga. Quando essa carga ultrapassa o limite superior uma ordem para a execução de uma nova instância é emitida pelo controlador para o Cloud Controller. Quando a nova instância entra em execução e novas requisições são enviadas a ela a utilização global de CPU tende a diminuir. Conforme a concorrência diminui, e consequentemente o uso do processador, o controlador ordena o término de instâncias a fim de manter o seu número compatível com a carga de trabalho corrente.

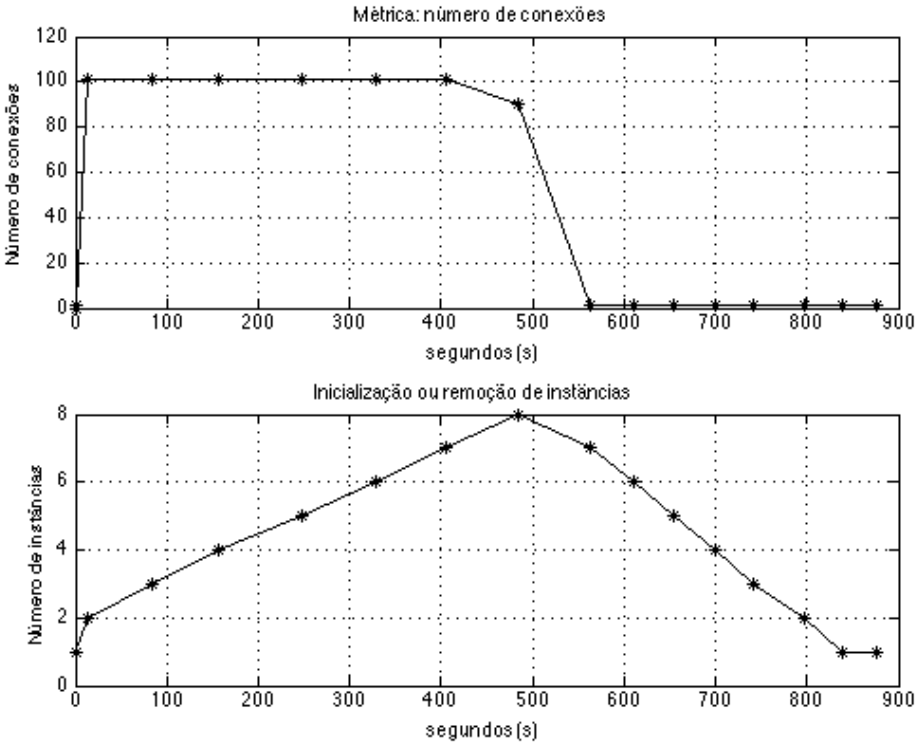
Tais comportamentos são válidos também quando utilizamos as outras métricas de balanceamento. A Figura 6 demostra o comportamento da escalabilidade baseada no consumo de memória e a Figura 7 relaciona a escalabilidade pela quantidade de conexões que ingressam no balanceador de carga.

Figura 6: Consumo de memória



O gráfico da escalabilidade por número de conexões, Figura 7, mostrou o menor tempo de execução, a maior quantidade de instâncias executadas e o menor tempo de inicialização entre instâncias. Esse fato ocorreu pela característica do teste utilizado, utilizamos a concorrência de 100 conexões e o limite superior da escalabilidade foi configurado como 80. Dessa forma, assim que o balanceador encaminha as 100 primeiras conexões à única instância inicial o gatilho superior de 80 conexões é disparado e o sistema escala logo nos primeiros segundos do teste, o que justifica o seu melhor desempenho. De forma equivalente, os gatilhos para os parâmetros processador e memória só são disparados depois de alguns ciclos de monitoramento, devido ao comportamento do consumo de tais recursos ser mais progressivo.

Figura 7: Número de conexões



Como efeito do balanceamento de carga e da escalabilidade, nas Figura 8, 9 e 10, observamos que a quantidade de transações processadas por segundo aumenta à medida que mais instâncias são adicionadas ao sistema.

Figura 8: Transações por segundo - CPU

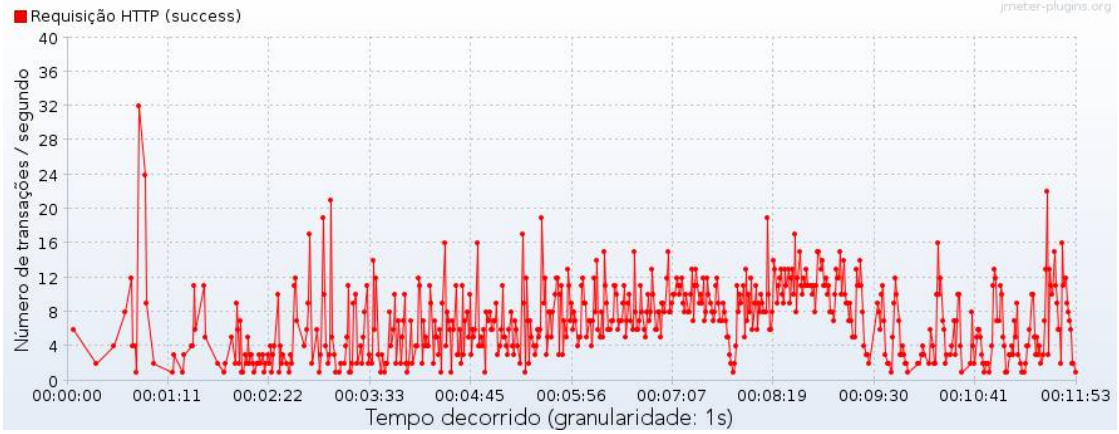


Figura 9: Transações por segundo - Memória

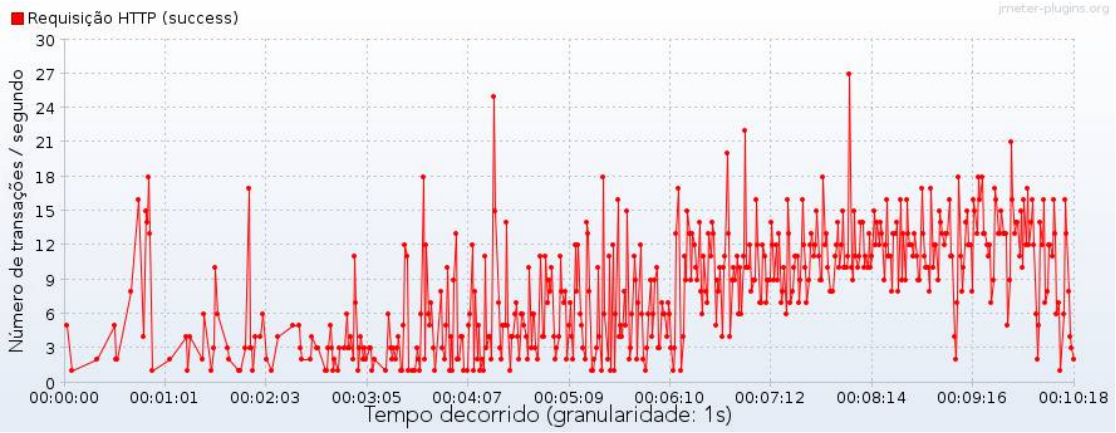
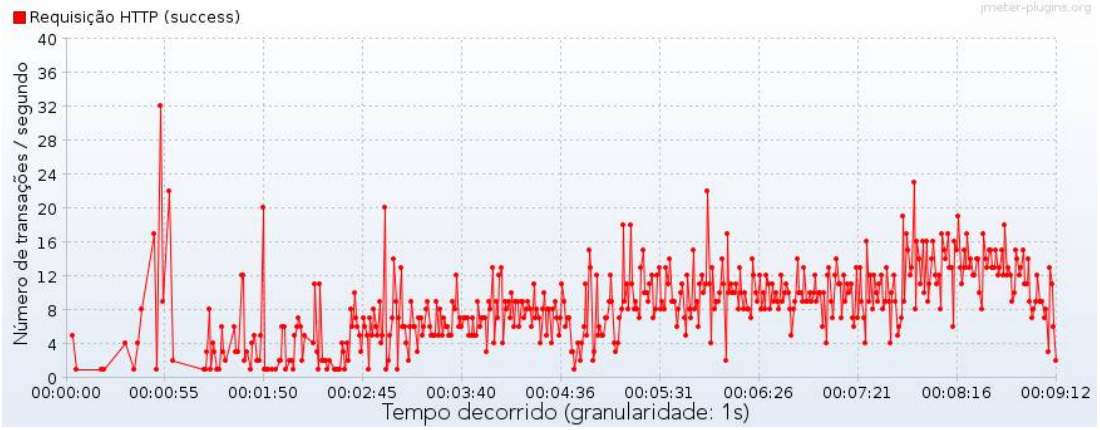


Figura 10: Transações por segundo - Número de conexões



6 Conclusão

Este trabalho apresenta um arquitetura de balanceamento de carga web para uma nuvem privada Eucalyptus que provê escalabilidade sob demanda baseada no estado das instâncias. Abordamos e discutimos os resultados da escalabilidade da nuvem em função do estado de três métricas: utilização dos processadores, consumo de memória e número de conexões.

O trabalho demonstrou alguns benefícios das plataformas de computação em nuvem, que são capazes de lidar com cargas repentinas, entregando recursos sob demanda para os usuários e mantendo maior utilização de recursos e infraestrutura, reduzindo assim, custos de gestão.

A maior contribuição do trabalho foi apresentar uma proposta de balanceamento de carga dinâmica que pode evitar problemas de subprovisionamento de recursos, e possíveis problemas de sobrecarga de sistemas e provisionamento para cargas de pico, o que poderia gerar recursos ociosos.

O software escalonador permite configurar limites superiores e inferiores que devem ser trabalhados baseados no tipo de aplicação servida na nuvem, se esta fizer uso intensivo de processamento ou memória. Ou ainda, se haverá um grande número de conexões ou tráfego em rajadas. Nos experimentos, a escalabilidade servida conseguiu provisionar os recursos necessários ao tratamento da carga de trabalho evitando o subprovisionamento de recursos e terminando instâncias à medida que a carga total do sistema diminuía. Técnicas melhoradas de predição e testes de desempenho com outras soluções de balanceamento de carga serão objetos de pesquisas futuras.

Agradecimentos

Gostaríamos de prestar agradecimentos ao Núcleo de Tecnologia da Informação da Universidade Estadual Vale do Acaraú (UVA) por permitir a utilização de sua infraestrutura computacional para a implantação, desenvolvimento e testes da arquitetura apresentada.

Referências

[1] ARMBRUST, M. et al. *Above the Clouds: A Berkeley View of Cloud Computing*. [S.l.], 2009. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf>.

[2] BHADANI, A.; CHAUDHARY, S. Performance evaluation of web servers using central load balancing policy over virtual machines on cloud. In: *Proceedings of the Third Annual ACM Bangalore Conference*. New York, NY, USA: ACM, 2010. (COMPUTE '10), p. 16:1–16:4. ISBN 978-1-4503-0001-8. Disponível em: <http://doi.acm.org/10.1145/1754288.1754304>.

[3] MELL, P.; GRANCE, T. *The NIST Definition of Cloud Computing*. [S.l.], 2011.

[4] LAHA, J.; SATPATHY, R.; DEV, K. Load balancing techniques : Major challenges in cloud computing - a systematic review. In: *IJCSN International Journal of Computer Science and Network*. [S.l.: s.n.].

[5] KANSAL, N. J.; CHANA, I. Cloud load balancing techniques : A step towards green computing. In: *IJCSI International Journal of Computer Science Issues,*. [S.l.: s.n.].

[6] CLOUD Computing Implementation, Management and Security. [S.l.]: CRC Press, 2010.

[7] IRAKOZE, I. *Cloud-Based Mobile Applications*. B.S. Thesis, 2013.

[8] EUCALYPTUS. *Eucalyptus | Open Source Private Cloud Software*. 2014. <http://www.eucalyptus.com/>.

[9] EC2, A. *AWS | Amazon Elastic Compute Cloud (EC2)*. 2014. <http://aws.amazon.com/pt/ec2/>.

[10] S3, A. *AWS | Amazon Simple Storage Service (S3)*. 2014. <http://aws.amazon.com/pt/s3/>.

[11] FOLCH, A. *Interface development for Eucalyptus based cloud*. Dissertação (Mestrado) — Vilnius Gediminas Techinical University, 2011.

[12] EUCALYPTUS. *Eucalyptus Cloud Computing Architecture*. 2014. Disponível em: <https://www.eucalyptus.com/eucalyptus-cloud/iaas/architecture>.

[13] ZABBIX. *Homepage of Zabbix :: An Enterprise-class Open Source Distributed Monitoring Solution*. 2014. <http://www.zabbix.com/>.

[14] NGINX. *NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy*. 2014. <http://nginx.com/>.

[15] JMETER, A. *Apache JMeter*. 2014. <http://jmeter.apache.org/>.

[16] SHARMA, S.; SINGH, S.; SHARMA, M. Performance analysis of load balancing algorithms. In: *World Academy of Science, Engineering and Technology*. [S.l.: s.n.].

[17] MADHU, K. M.; SHAH, S. M. Study on dynamic load balancing in distributed system. In: *International Journal of Engineering Research & Technology (IJERT)*. [S.l.: s.n.].

[18] ALAKEEL, A. M. A guide to dynamic load balancing in distributed computer systems. In: *IJCSNS International Journal of Computer Science and Network Security*. [S.l.: s.n.].

[19] GROSSMAN, R. L. The case for cloud computing. *IT Professional*, IEEE Computer Society, Los Alamitos, CA, USA, v. 11, n. 2, p. 23–27, 2009. ISSN 1520-9202. Disponível em: <http://www.computer.org/csdl/mags/it/2009/02/mit2009020023-abs.html>.

[20] DEMCHENKO, Y. et al. Security infrastructure for on-demand provisioned cloud infrastructure services. In: LAMBRINOUDAKIS, C.; RIZOMILIOTIS, P.; WLODARCZYK, T. W. (Ed.). *CloudCom*. IEEE, 2011. p. 255–263. ISBN 978-1-4673-0090-2. Disponível em: <http://dblp.uni-trier.de/db/conf/cloudcom/cloudcom2011.html>.

[21] MANDAL, S. K.; KHILAR, P. M. Efficient virtual machine placement for on-demand access to infrastructure resourcesin cloud computing. *International Journal of Computer Applications*, v. 68, n. 12, p. 6–11, April 2013. Full text available. Disponível em: <http://research.ijcaonline.org/volume68/number12/pxc3887101.pdf>.

[22] KATYAL, M.; MISHRA, A. A comparative study of load balancing algorithms in cloud computing environment. *International Journal of Distributed and Cloud Computing*, 2013.

[23] HU, J. et al. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In: *Proceedings of the 2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming*. Washington, DC, USA: IEEE Computer Society, 2010. (PAAP '10), p. 89–96. ISBN 978-0-7695-4312-3. Disponível em: <http://dx.doi.org/10.1109/PAAP.2010.65>.

[24] KARMAKAR, K.; MANDAL, A. Cost optimized virtual machine deployment in eucalyptus for autonomous systems. *International Journal of Innovations in Engineering and Technology (IJIET)*, v. 3, n. 4, p. 6–11, April 2014. Disponível em: <http://ijiet.com/issues/volume-3-issue-4-april-2014/>.

[25] COUTINHO, E.; GOMES, D. G.; SOUZA, J. de. Uma proposta de arquitetura autonômica para elasticidade em computação em nuvem. In: *Proceedings of 4º Workshop de Sistemas Distribuídos Autonômicos*. Santa Catarina, Brasil: [s.n.], 2014. (WoSiDA 2014), p. 3–6. ISSN 2177-496X. Disponível em: <http://sbrc2014.ufsc.br/anais/files/wosida/anaisWoSiDA2014.pdf>.