Uma Abordagem para Avaliação de Topologias de Aplicações em Nuvem Baseado em TOSCA

Tiago Rolim[1], **Américo Sampaio**[1], **Nabor Mendonça**[1], **Matheus Cunha**[1]

[1]Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321, Edson Queiroz, CEP 60811-905 Fortaleza, CE

{tiago.rol,mathcunha}@gmail.com, {americo.sampaio,nabor}@unifor.br

Abstract. *Provisioning cloud applications usually is a complex task as it involves the deployment and configuration of several components (e.g., load balancer, application server, database) and cloud services (computing, storage, CDN, etc.) also known as application blueprints or topologies. The Topology and Orchestration Specification for Cloud Applications (TOSCA) is a recent standard that has focused on standardizing the way cloud applications are structured and managed to favor interoperability. In this paper we describe an approach that facilitates the evaluation of different application topologies by enabling cloud users to easily provision and evaluate different TOSCA topology options based on performance and cost metrics. We show the technical feasibility of the approach based on a case study with the WordPress blogging application where various topologies were automatically provisioned and evaluated in order to gain insights into the best (w.r.t. performance) options.*

Resumo. *Provisionamento de aplicações em nuvem geralmente é uma tarefa complexa, uma vez que envolve a implantação e configuração de vários componentes (ex: balanceador de carga, servidor de aplicações, banco de dados) e serviços de nuvem (computação, armazenamento, CDN, etc.), também conhecido como blueprints ou topologias. Topology and Orchestration Specification for Cloud Applications (TOSCA) é um padrão recente que tem se concentrado em padronizar a forma com que as aplicações em nuvem são estruturadas e gerenciadas para favorecer interoperabilidade. Neste artigo descrevemos uma abordagem que facilita a avaliação de diferentes topologias de aplicações, permitindo que os usuários de nuvem facilmente provisionem e avaliem diferentes opções de topologias TOSCA baseado em métricas de desempenho e de custo. Nós mostramos a viabilidade técnica da abordagem com base em um estudo de caso com a aplicação de blog WordPress onde várias topologias foram provisionadas automaticamente e avaliadas, a fim de obter informações sobre as melhores opções.*

1. Introdução

A computação em nuvem foi rapidamente adotada por um crescente número de organizações que perceberam a nuvem como uma forma mais escalável e de baixo custo, em comparação com aquisição e manutenção de sua própria infraestrutura local [Armbrust et al. 2010]. Os provedores de nuvem, como Amazon, Rackspace, Goo-

gle e Microsoft oferecem uma ampla gama de recursos de computação virtual que variam em capacidade e preço, bem como uma gama de modelos de preços (por exemplo, pay-as-you-go, instâncias reservadas, instâncias spot). Estes recursos podem ser facilmente adquiridos ou liberados pelos usuários da nuvem, conforme a demanda cresce ou diminui, possibilitando controlar mais eficazmente a utilização de recursos e os custos. No entanto, apesar do fato destes recursos computacionais poderem ser facilmente gerenciados ainda é comum que usuários da nuvem desperdicem recursos mesmo quando utilizam serviços de controle automático de elasticidade (autoscaling) [Almeida Morais et al. 2013][Gonçalves et al. 2015].

Serviços de Autoscaling em nuvem geralmente se concentram em adição ou remoção de máquinas virtuais (também conhecido como escalabilidade horizontal) com base em limiares definidos pelo usuário sobre o consumo de recursos virtuais, como CPU e memória. Apesar disto ser um mecanismo útil para gerenciar uma gama de aplicações, especialmente aquelas com demandas mais imprevisíveis, muitos trabalhos já mostraram a importância de se provisionar e testar o desempenho de aplicações para descobrir as melhores opções de recursos virtuais (i.e., tipos de instância do provedor de nuvem) para níveis particulares de demanda [Jung et al. 2013][Jayasinghe et al. 2012][Cunha et al. 2013][Silva et al. 2013]. A seleção adequada de recursos virtuais é crítica para se atingir um uso efetivo da nuvem e ainda mais relevante caso a aplicação tenha uma demanda mais previsível.

Apesar dos muitos benefícios oferecidos pelas soluções de computação em nuvem existentes, e a relativa simplicidade de aquisição e liberação de recursos na nuvem, muitos usuários ainda tem dificuldades na seleção dos recursos e serviços que melhor atendam às necessidades das suas aplicações [Frey et al. 2013][Saripalli and Pingali 2011][Gonçalves Junior et al. 2015]. Por exemplo, estudos anteriores [Cunha et al. 2013][Borhani et al.] mostraram que variações na quantidade e tipos de máquinas virtuais (VMs) utilizadas para implantar uma aplicação web multicamadas na nuvem da Amazon podem acarretar diferenças significativas de desempenho da aplicação. Particularmente para algumas camadas específicas da aplicação (ex: camada web e de aplicação) arquiteturas de implantação (topologias) baseadas em clusters contendo múltiplas VMs de baixa capacidade podem significativamente melhorar o desempenho e reduzir os custos da aplicação quando comparada à topologias baseadas em uma única VM de grande capacidade [Cunha et al. 2013]. Portanto, as decisões de arquiteturas durante a implantação de aplicações em nuvem podem ter um impacto direto não apenas nos custos operacionais das aplicações, mas também em muitos outros atributos importantes, como performance, escalabilidade, uso de recursos, segurança e assim por diante [Gonçalves Junior et al. 2015][Chung et al. 2013].

A solução dos problemas relacionados à escolha de topologias adequadas requer o tratamento adequado de alguns desafios ainda em aberto. Primeiramente é importante que se tenha uma solução para a implantação e configuração automatizada dos componentes da aplicação (ex: servidores de aplicação, balanceadores de carga, banco de dados) além de prover os recursos necessários (ex: máquinas virtuais, armazenamento) para o provisionamento da aplicação no ambiente de nuvem [Breitenbucher et al. 2014]. Além disso, também é importante que se tenha uma maneira sistemática para testar diferentes topologias da mesma aplicação para investigar quais topologias são mais eficazes sob a ótica de

algum critério (ex: desempenho e custo).

Neste artigo apresentamos uma abordagem que se foca em apresentar soluções para os desafios supracitados. A abordagem baseia-se na Topology and Orchestration Specification for Cloud Applications (TOSCA) [OASIS 2013b][OASIS 2013a], um padrão para especificar e automatizar a avaliação de diferentes opções de topologias de aplicação. Como exemplo, considere uma aplicação web LAMP, tais como WordPress, que é composto de um servidor de aplicação e um servidor de banco de dados. A nossa abordagem permite especificar e automatizar a implantação, configuração e provisionamento de múltiplas variações de topologia da aplicação (por exemplo, todos os componentes na mesma VM, servidor de aplicação e banco de dados em diferentes VMs, cluster de servidores de aplicação que executam em diferentes VMs e assim por diante), utilizando TOSCA como o formalismo subjacente. Além disso, a abordagem permite aos usuários especificarem diferentes tipos de recursos a serem testados (por exemplo, uma gama de diferentes tipos de instância), bem como os parâmetros de desempenho a serem medidos (ex: tempo de resposta). Em suma, permitirá automatizar a execução de uma variedade de testes de desempenho para múltiplas variações da topologia de aplicação em combinação com um gama de seleção de tipos de recursos. Para o melhor de nosso conhecimentos nossa abordagem é a primeira tentativa de combinar esses duas características de uma perspectiva baseada no padrão TOSCA.

O restante deste artigo é descrito da seguinte forma. A Seção 2 apresenta os principais conceitos e tecnologias relevantes para o nosso trabalho. A Seção 3 descreve a abordagem proposta e sua implementação. A Seção 4 mostra a avaliação da abordagem através de um estudo de caso baseado na aplicação de blog WordPress. A seção 5 compara a nossa abordagem com os trabalhos relacionados. Finalmente, a Secção 6 conclui o artigo e oferece sugestões para trabalhos futuros.

2. Background

Esta seção apresenta o padrão TOSCA, bem como duas tecnologias que são utilizadas por nossa abordagem: Cloudify [Gigaspaces] e Cloud Crawler [Cunha et al. 2013].

2.1. TOSCA

Topology and Orchestration Specification for Cloud Applications (TOSCA) é um padrão criado pela OASIS (Organization for the Advancement of Structured Information Standards) para descrever componentes de aplicações em nuvem e seus relacionamentos [OASIS 2013b][OASIS 2013a]. TOSCA define um Topology Template (também referido como o modelo da topologia de um serviço) que é um metamodelo para a definição de aplicações. O Topology Template define os componentes da aplicação, bem como uma forma de gerenciá-la de forma independente do provedor de nuvem. Diferenças significativas entre as APIs de diferentes provedores de nuvem estão encapsuladas por trás da implementação TOSCA.

A Figura 1 mostra os principais elementos de um *Topology Template*: *Node Template* e *Relationship Template*. O primeiro descreve os componentes da aplicação em nuvem e o último os seus relacionamentos. Juntos, *Node Template* e *Relationship Template* definem uma topologia da aplicação como um grafo. Um *Node Template* especifica a ocorrência de um *Node Type* como um componente de um serviço. Um *Node Type*

define as propriedades dos componentes da aplicação e as operações disponíveis para manipular esses componentes. Essa abordagem permite que *Node Types* sejam reutilizados por outros *Node Templates*. *Relationship Templates* especificam a ocorrência de um relacionamento entre os nós em um *Topology Template*. Cada *Relationship Template* refere-se a um *Relationship Type* que define a semântica e propriedades do relacionamento.

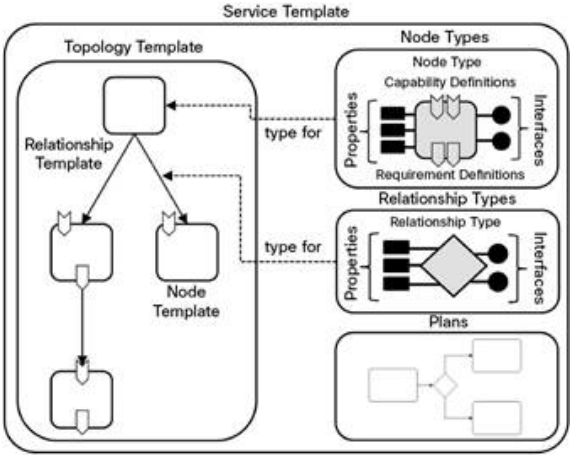

Figura 1. TOSCA Service Template [OASIS 2013b]

2.2. Cloudify

Cloudify [Gigaspaces] é uma plataforma de orquestração open-source baseada em TOSCA. Ela automatiza o processo de implantação, configuração e gerenciamento da aplicação, e foi concebida para apoiar qualquer tipo de aplicação em diversos provedores de nuvem. Usuários Cloudify definem a aplicação usando uma linguagem baseada em TOSCA. Esses arquivos de definição, chamados *blueprints*, descrevem a topologia da aplicação que inclui os detalhes necessários para instalar e gerenciar a aplicação em nuvem.

2.3. Cloud Crawler

O ambiente Cloud Crawler [Cunha et al. 2013] suporta usuários de nuvem IaaS para avaliar automaticamente o desempenho de suas aplicações sob uma variedade de perfis de configuração e cargas de trabalho. O ambiente permite aos usuários da nuvem especificar os testes de desempenho a serem executados na nuvem de forma declarativa, em um nível mais alto de abstração. Para este fim, Cloud Crawler fornece uma linguagem específica de domínio (DSL) declarativa, denominada Crawl, que permite a descrição de uma rica variedade de avaliações de desempenho de aplicações na nuvem. Crawl também oferece um recurso para controlar o ciclo de vida dos recursos de nuvem (por exemplo, algumas máquinas virtuais podem ser configuradas para serem desligadas no final de cada teste individual, enquanto outras podem ser deixadas em execução até que o último teste seja executado), permitindo assim que os usuários da nuvem tenham um controle mais refinado sobre os seus custos de nuvem. O ambiente também inclui um mecanismo de execução, chamado Crawler, que é o serviço responsável pelo pré-processamento, a análise, validação, execução, monitoramento e coleta dos resultados das avaliações de desempenho das aplicações descritas no Crawl.

Cloud Crawler foi desenvolvido por nosso próprio grupo de pesquisa e durante os testes foi responsável por gerar a carga de trabalho exigida, a implantação, execução e monitoramento do desempenho da aplicação de destino. É importante mencionar que, no contexto deste trabalho Crawler é chamado por um serviço do Cloudify após a aplicação baseada em TOSCA ser implantada e configurada pelo Cloudify conforme detalhado na a próxima seção.

3. A Abordagem Proposta

A Figura 2 mostra a abordagem proposta em detalhes. A fim de auxiliar a compreensão da abordagem ilustramos os passos com base na aplicação WordPress, que foi utilizado para a avaliação descrita na seção 4.

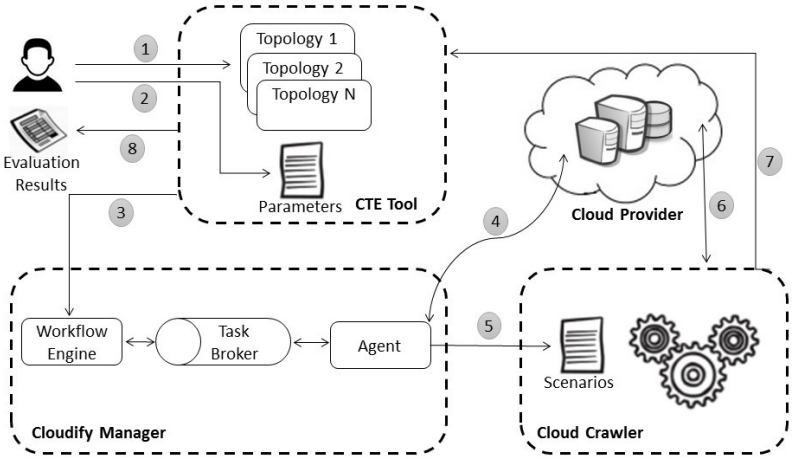

Figura 2. A abordagem proposta

No passo 1 os usuários definem um conjunto de especificações de topologia para uma aplicação específica. Cada especificação de topologia tem um conjunto de arquivos que descrevem os componentes da aplicação, e suas propriedades de configuração. É importante destacar que o apoio atual do framework Cloud Topology Evaluation Tool (CTE) não permite editar a especificação da topologia ou gerá-la a partir de um modelo visual. Esses arquivos são criados em YAML baseado no padrão TOSCA e podem ser reutilizados e extendidos a partir de arquivos criados anteriormente como blueprints do Cloudify. A idéia principal do Passo 1 é facilitar a seleção de várias topologias que mais tarde serão avaliadas pela execução de um conjunto de testes de desempenho configurados com os parâmetros fornecidos pelo usuário (por exemplo, cargas de trabalho, tipos de máquinas usadas, bem como parâmetros de provedor de nuvem) e capturados pelo CTE. No futuro pretendemos melhorar a criação de topologias usando uma abordagem baseada em modelos.

A Figura 3 mostra um exemplo que descreve uma das topologias utilizadas durante a nossa avaliação do WordPress (wordpress-blueprint.yaml). Este arquivo foi originalmente reutilizado do Cloudify e complementado com a criação de outros tipos, bem como parâmetros fornecidos como entrada pelo usuário no Passo 2. As linhas 8-12 mostram os parâmetros de entrada capturados pela nossa ferramenta, tais como a imagem a ser utilizada, o tamanho (neste caso um tipo de instância na Amazon) e agente de usuário (usuário que o cloudify utiliza para se conectar à VM via SSH). As linhas 14-40 definem

os *node templates* que representam os componentes da aplicação. O *host* (linhas 16-20) representa uma máquina virtual na Amazon, cujo tipo de instância será configurado com base no parâmetro de entrada size e imagem fornecidos anteriormente. Esta máquina virtual será o host do banco de dados MySQL e da aplicação WordPress (veja seção relações definidas nas linhas 26-27; e as linhas 37-38). Para cada componente foi definido um tipo de nó específico, para o banco de dados (linha 22) e para o Wordpress (linha 30), tal como sugerido pelo padrão TOSCA.

```
1  tosca_definitions_version: cloudify_dsl_1_1
2
3  imports:
4    - http://www.getcloudify.org/.../3.2.1/types.yaml
5    - http://www.getcloudify.org/.../1.2.1/plugin.yaml
6    - types/wordpress-types.yaml
7
8  inputs:
9
10   image:
11   size:
12   agent_user:
13
14 node_templates:
15
16   host:
17     type: cloudify.aws.nodes.Instance
18     properties:
19       image_id: { get_input: image }
20       instance type: { get input: size }
21   mysql:
22     type: wordpress.nodes.MySQLDatabase
23     properties:
24       port: 3306
25     relationships:
26       - type: cloudify.relationships.contained_in
27         target: host
28
29   wordpress:
30     type: wordpress.nodes.WordpressApplicationModule
31     properties:
32       port: 80
33     interfaces:
34       cloudify.interfaces.monitoring:
35         start: scripts/crawler/start-monitoring.sh
36     relationships:
37       - type: cloudify.relationships.contained_in
38         target: host
39       - type: node_connected_to_mysql
40         target: mysql
```

Figura 3. Exemplo de um arquivo de topologia (wordpress-blueprint.yaml)

A Figura 4 mostra os tipos de nó e tipos de relacionamentos utilizados no modelo de topologia anterior. *MySQLDatabase* (linhas 3-13) e *WordpressApplicationModule* (linhas 15-25) são os dois tipos de nós definidos. Estes tipos estendem tipos previamente definidos e definem as propriedades porta, bem como um conjunto de interfaces, que representam fases do ciclo de vida (criar, iniciar, parar) que referenciam scripts que são chamados quando esses nós são implantados pelo Cloudify. As linhas 28- 33 mostram a definição do relacionamento que conecta o WordPress ao banco de dados. A propriedade postconfigure referencia um script que será chamado durante a implantação, a fim de definir o IP do banco de dados em um arquivo de configuração do WordPress.

```
1  node_types:
2
3    wordpress.nodes.MySQLDatabase:
4      derived_from: cloudify.nodes.DBMS
5      properties:
6        port:
7          description: MySQL port
8          type: integer
9      interfaces:
10       cloudify.interfaces.lifecycle:
11         create: scripts/mysql/install-mysql.sh
12         start: scripts/mysql/start-mysql.sh
13         stop: scripts/mysql/stop-mysql.sh
14
15   wordpress.nodes.WordpressApplicationModule:
16     derived_from: cloudify.nodes.ApplicationModule
17
18     properties:
19       port:
20         description: Web application port
21         type: integer
22     interfaces:
23       cloudify.interfaces.lifecycle:
24         create: scripts/wordpress/install-wordpress.sh
25         start: scripts/wordpress/start-apache.sh
26         stop: scripts/wordpress/stop-apache.sh
27
28 relationships:
29
30   node_connected_to_mysql:
31     derived_from: cloudify.relationships.connected_to
32     target_interfaces:
33       cloudify.interfaces.relationship_lifecycle:
34         postconfigure: scripts/wordpress/set-mysql-url.sh
```

Figura 4. Tipos de Nós e Relacionamentos definido para WordPress

Depois de concluir essas definições, o usuário aciona a ferramenta CTE que por sua vez chama um serviço do Cloudify passando os arquivos com a especificação da topologia (arquivos YAML definidos anteriormente), bem como a os parâmetros de configuração que serão utilizados posteriormente durante os testes com Cloud Crawler. Isto é o que é feito no Passo 3. Dentro do Cloudify Manager o *Workflow Engine*

será responsável por orquestrar as tarefas para implantação da aplicação, bem como a sua configuração (por meio do *task broker*). O agente Cloudify é responsável pela comunicação com a API do provedor de nuvem (neste exemplo Amazon AWS) e realiza a implantação e configuração da toda a pilha da topologia descrita anteriormente (Passo 4).

Uma vez que toda a aplicação é completamente implantada e configurada na nuvem, a ferramenta Cloudify chama a API do Cloud Crawler (Passo 5) passando os parâmetros de testes coletados durante o Passo 1. Esses parâmetros, tais como: lista de tipos de máquina virtual a ser testado; cargas de trabalho a ser executada (por exemplo, 100, 300, 500, ... usuários simultâneos) e número de repeticões de testes que são usados pelo Crawler para configurar seus cenários de testes. Cloud Crawler irá então usar seu gerador de carga para testar a aplicação (neste caso Wordpress) com cada um dos os tipos de máquinas virtuais selecionadas a partir da lista e cada carga de trabalho (Passo 6). Uma vez que os testes forem concluídos, Crawler retorna os dados coletados (Passo 7) para a ferramenta CTE.

É importante destacar que as etapas anteriores serão exercidas de forma repetitiva para cada modelo de topologia a ser avaliada. Uma vez que todas as topologias são testadas, o usuário terá dados suficientes para fazer a seleção da configuração das topologias mais apropriadas para uma determinada carga de trabalho. Esse argumento será reforçado na próxima sessão.

4. Avaliação

A fim de avaliar a utilização da nossa abordagem nós realizaramos um estudo para executar o provisionamento automatizado e testes de diferentes topologias da aplicação WordPress.

4.1. Questões de Pesquisa

Este estudo centrou-se em responder as seguintes questões de pesquisa:
QP1. Como diferentes topologias de aplicações se comportam em termos de desempenho e custo?
QP2. Como os diferentes tipos de máquinas provisionadas afetam o desempenho de uma determinada topologia?
QP3. Qual é o impacto do uso de TOSCA para provisionar as diferentes topologias?

4.2. Projeto do Estudo

Nós selecionamos o WordPress [Borhani et al.], uma aplicação de web blogging real, como a nossa aplicação de referência. Essa seleção deveu-se a popularidade desta aplicação e a sua arquitetura de componentes escalável que são ideais para implantação em nuvem. WordPress segue uma típica arquitetura multi-camadas, compreendendo duas camadas: aplicação e dados. Em cada uma dessas camadas implantamos os seguinte componentes: o servidor de aplicação Apache e o MySQL servidor de banco de dados relacional, respectivamente. Para avaliarmos o desempenho do WordPress sob diferentes condições de carga nós usamos o framework de testes de carga Gatling [Gatling]. Gatling apresenta uma DSL amigável que permite a definição de uma variedade de operações de usuários e estatísticas associadas, permitindo testar facilmente o desempenho e coletar os resultados de aplicações web interativas.

O estudo considerou diferentes variações de topologia da aplicação WordPress como mostra a Figura 5. Nós basicamente definimos 5 topologias diferentes e provisionamos cada topologia em diferentes tipos de instâncias na nuvem da Amazon, conforme descrito abaixo. Nós definimos três tipos diferentes de instâncias (m3.medium, m3.large e m3.xlarge) para cada topologia definida e executamos testes para diferentes cargas de trabalho (100, 300, 500, 700 e 900 usuários simultâneos). Cada teste foi repetido três vezes baseados na seguinte sequência de requisições: logar; adicionar um novo post; procurar o novo post; atualizar o novo post; procurar um post anteriormente existente por palavra-chave; atualizar o post já existente; e por fim, sair.

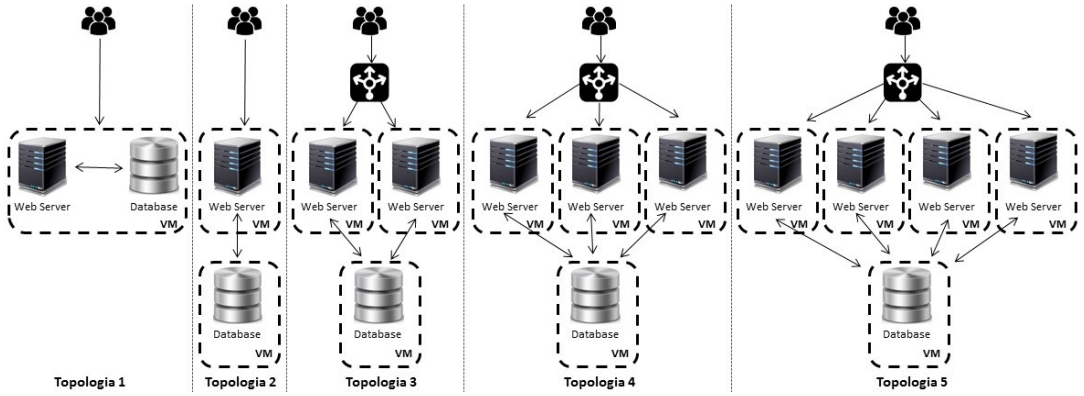

Figura 5. Diferentes topologias para WordPress

A topologia 1 é a mais simples contendo apenas uma máquina virtual que inclui o WordPress (Servidor Apache) e o banco de dados (MySQL). Como descrito acima, essa topologia irá variar quanto ao tipo de instância (m3.medium, m3.large e m3.xlarge) e vai ser testada 3 vezes para cada uma das cargas de trabalho anteriores.

A topologia 2 separa a camada web (Servidor Apache) e banco de dados (MySQL) em duas máquinas virtuais diferentes. Neste caso, só variamos o tipo da VM (m3.medium, m3.large e m3.xlarge) do servidor de aplicação e o banco de dados foi fixado como uma única instância do tipo m3.xlarge como recomendada por outros trabalhos [Gonçalves Junior et al. 2015]. Novamente nós repetimos o teste três vezes.

As topologias 3, 4 e 5 são baseados em uma ligeira variação da topologia 2 onde o servidor de aplicação (Apache) é replicado, respectivamente, em dois, três e quatro VMs e um Balenceador de Carga (Nginx) é adicionado para receber a entrada e distribuir requisições para um dos grupos de servidores do Apache. Os testes também são executados três vezes para as mesmas cargas de trabalho.

4.3. Execução do Estudo

A fim de avaliar as topologias que se aplicam a abordagem definida na seção 3, nós primeiramente definimos a especificação TOSCA para cada uma das topologias definidas anteriormente como mostrado na Figura 6 e na Figura 7. A Figura 6 mostra a definição da topologia T2 e a Figura 7 mostra a especificação das topologias T3, T4 e T5. A topologia T1 já foi mostrada na Figura 3.

T2 é uma variação da especificação T1 onde dois diferentes hosts são definidos como máquinas virtuais da Amazon, sendo um para o banco de dados MySQL e outro

```
node_templates:

  mysql_host:
    type: cloudify.aws.nodes.Instance
    properties:
      image_id: { get_input: image }
      instance_type: { get_input: size }

  mysql:
    type: wordpress.nodes.MySQLDatabase
    properties:
      port: 3306
    relationships:
      - type: cloudify.relationships.contained_in
        target: mysql_host

  wordpress_host:
    type: cloudify.aws.nodes.Instance
    properties:
      image_id: { get_input: image }
      instance_type: { get_input: size_wordpress }

  wordpress:
    type: wordpress.nodes.WordpressApplicationModule
    properties:
      port: 80
    relationships:
      - type: cloudify.relationships.contained_in
        target: wordpress_host
      - type: node_connected_to_mysql
        target: mysql
```

Figura 6. Especificação TOSCA para topologia T2

```
node_templates:

  mysql_host:
    ...

  mysql:
    ...

  wordpress_host:
    type: cloudify.aws.nodes.Instance
    instances:
      deploy: ${instances_deploy}
    ...

  wordpress:
    ...

  nginx_host:
    type: cloudify.aws.nodes.Instance
    properties:
      image_id: { get_input: image }
      instance_type: { get_input: size }

  nginx:
    type: wordpress.nodes.Nginx
    relationships:
      - type: cloudify.relationships.contained_in
        target: nginx_host
      - type: nginx_connected_to_vm
        target: wordpress_host
```

Figura 7. Especificação TOSCA para topologias T3, T4, T5

para a aplicação WordPress.

T3, T4 e T5 compartilham a mesma definição YAML onde um parâmetro (instances) é configurado durante o tempo de implantação para receber um número maior de instâncias como definido pelo usuário. Isto representa um cluster de Apaches utilizado para a camada de negócio do WordPress.

Depois de descrever as topologias, o usuário fornece as entradas necessárias para os testes de desempenho para a ferramenta de CTE. Os parâmetros passados serão: credenciais da nuvem; lista de tipos de máquinas a serem testadas e lista de cargas de trabalho a serem executadas. Estes parâmetros serão usados pela abordagem para configurar os testes no Cloud Crawler como discutido na seção 3. Após todos os testes serem executados para todas as topologias com todas os tipos de máquinas e cargas de trabalho a abordagem é finalizada e os dados são analisados.

4.4. Resultados e Discussões

A Figura 8 mostra os resultados dos testes de desempenho para as topologias anteriores.

Os dados representam os tempos médios de resposta (em milissegundos) para as cargas de trabalho estudadas (100, 300, 500, 700 e 900) para todas as topologias exceto T4 e T5. A razão pela qual omitimos essas duas topologias foi para melhorar a legibilidade. Nós avaliamos os dados com relação às questões de pesquisa anteriores.

QP1. *Como diferentes topologias de aplicações se comportam de termos de desempenho e custo?*

A fim de avaliar os dados é também importante salientar que os tempos correspondem a uma série de operações e que avaliar o que é bom ou ruim dependerá das restrições

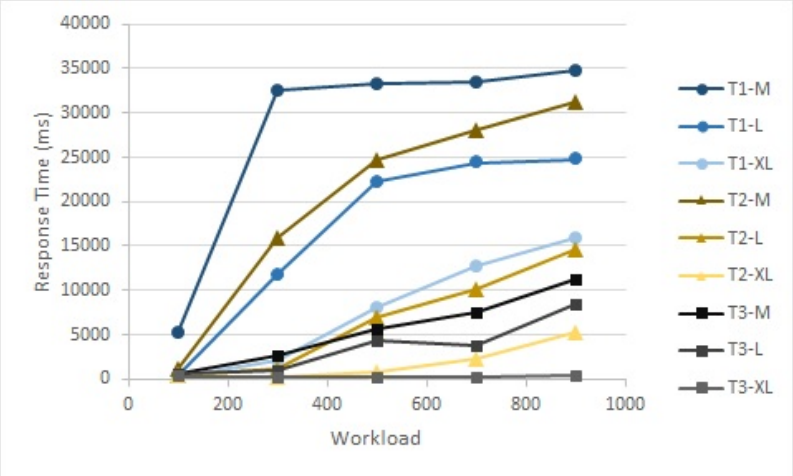

Figura 8. Resultados de desempenho para topologias WordPress

do usuário e do nível de qualidade esperado. Para esta combinação de operações, por exemplo, um tempo de resposta inferior a 10s (10.000ms) é considerado muito bom.

Considerando a carga de trabalho de 100 usuários pode-se ver que todas as topologias apresentam tempos médios de resposta muito baixos. O maior seria T1-M (Topologia T1 com uma máquina virtual do tipo m3.medium), com cerca de 5s que ainda seria bom. T1-M no entanto, não se mostra uma configuração muito confiável quando há 300 usuários simultâneos, pois o seu tempo de resposta aumentou para cerca de 30s em média, o que pode ser considerado não adequado para usuários da aplicação.

Separar o banco de dados do servidor de aplicação (T2) mostra uma grande melhoria nos tempos de resposta para todos os tipos de máquina selecionados. Por exemplo, para 300 usuários T2-M dá um tempo de resposta de cerca de metade do tempo de T1-M (15s quando comparado a T1-M que é em torno de 30s).

Adicionando um balanceador de carga e réplicas do servidor de aplicação (T3) melhora o tempo de resposta significativamente especialmente para cargas de trabalho mais elevadas. Por exemplo, a melhor configuração de T2 (T2-XL) é consideravelmente superado por T3-XL (por exemplo, para 900 usuários médias T3-XL 403ms enquanto as médias T2-XL cerca de 5000ms).

Este tipo de análise também é muito importante quando considerado os custos de utilização de recursos. Os valores de custo para cada tipo de instância (on demand no Oeste da Virginia) são: m3.medium = $0,067; m3.large = $0,133; m3.xlarge = $0,266). Por exemplo, se os usuários considerarem o tempo de resposta de 10 segundos como aceitável, a seguinte seleção poderia ser aplicada:

100 usuários: T1-M seria o mais barato, uma vez que utiliza uma única VM do tipo medium.

300 usuários: para esta carga de trabalho duas opções são as melhores em termos de custo: T1-XL ou T2-L. A primeira usa uma única VM (m3.xlarge) com custo de $0,266 e o segunda usaria duas VMs (m3.large) dando o mesmo custo. Por escalabilidade melhor, o usuário pode decidir usar a segunda opção.

500 usuários: para essa carga de trabalho o mesmo raciocínio da carga de trabalho 300 pode ser aplicada. A única observação aqui é que, neste caso o tempo médio de resposta de T1-XL é mais perto dos limite dos 10s o que poderia reforçar a seleção de T2-L.

700 usuários: neste caso, T2-M é a melhor opção em termos de custo, mas o seu desempenho é muito perto do limite. Neste caso, o T2-XL dá um melhor desempenho com o tempo de resposta médio de cerca de 2.3s mas o custo para a VM do Apache seria o dobro (de $0,133 para $0,266). Outra opção seria T3-M que teria um desempenho intermediário (cerca de 7s), mas teríamos que acrescentar uma outra máquina para o balanceador de carga e outra para o servidor web Apache.

900 usuários: para essa carga de trabalho a melhor escolha seria T2-XL, uma vez que não teríamos a máquina do balanceador de carga e teríamos uma uma máquina virtual a menos para o Apache, em seguida, T3-L.

QP2. *Como os diferentes tipos de máquinas provisionadas afetam o desempenho de uma determinada topologia?*

O aumento do tamanho da máquina virtual para qualquer topologia (a.k.a escalabilidade vertical) teve um impacto direto sobre os resultados. Praticamente todas as topologias testadas tiveram uma melhora de desempenho quando foram usados tamanhos maiores de VM. Contudo, o que os usuários precisam ter em mente é que às vezes é melhor aumentar o número de instâncias de máquinas virtuais (escalabilidade horizontal), ao invés de apenas aumentar a capacidade de VM. Por exemplo, a topologia T3, que se baseia em um cluster de Apaches supera significativamente as outras topologias especialmente para cargas mais elevadas. A topologias T4 e T5 (não mostradas no gráfico) melhoraram ainda mais estes resultados. Portanto, se os usuários precisam de um limiar de desempenho muito rigoroso, então é aconselhável escolher as topologias T3, T4 ou T5.

QP3. *Qual é o impacto do uso de modelos TOSCA para provisionamento as diferentes topologias?*

A resposta para esta pergunta é melhor compreendida analisando-se as Figuras 3, 6 e 7. Pode ser visto que, uma vez que o usuário definiu uma topologia tal como a topologia T1 mostrado na Figura 3 é muito fácil estender os arquivos YAML para criar as outras topologias. Na Figura 6, o usuário só precisa alterar a especificação anterior, separando os dois componentes (WordPress e MySQL) para usar dois diferentes hosts definidos na especificação YAML. Para as topologias T3, T4 e T5 (Figura 7), a reutilização é ainda melhor, uma vez que podem compartilhar o mesmo arquivo de especificação. Neste caso outro host precisava ser definido para o balanceador de carga Nginx, bem como seu tipo de nó. Um parâmetro (instances) é configurado durante tempo de implantação para receber um número maior de instâncias conforme definido pelo usuário. Isto irá representar um cluster de servidores Apache utilizado para a camada de aplicação WordPress.

Além de reutilizar as especificações de topologias, usando TOSCA temos a vantagem de facilitar a portabilidade de implantação de aplicativos para outras nuvens e infraestruturas locais que são suportados por uma implementação baseada em TOSCA, tais como Cloudify e outras ferramentas. No caso do nosso trabalho isso representa a capacidade de implantar com facilidade, configurar e testar o desempenho das diferentes topologias

para diferentes nuvens, como Cloud Crawler também suporta múltiplas implementações de nuvem [Cunha et al. 2013].

5. Trabalhos Relacionados

Provisionamento automático de aplicações na nuvem é assunto de vários trabalhos [Breitenbucher et al. 2014][Lu et al. 2013][Qasha et al. 2015][Binz et al. 2014]. Esses trabalhos concentram-se na questão do fornecimento de um caminho para especificar a topologia do aplicativo em nuvem baseado em TOSCA [Breitenbucher et al. 2014][Qasha et al. 2015][Binz et al. 2014] ou em uma linguagem específica própria [Lu et al. 2013]. Binz et al. [Breitenbucher et al. 2014] apresenta um meio para combinar os dois sabores de planos de provisionamento, declarativas e imperativas, que são gerados com base em modelos baseados na topologia TOSCA. Estes planos de provisionamento são fluxos de trabalho que podem ser executados automaticamente e podem ser personalizados por desenvolvedores de aplicação. Lu et al. [Lu et al. 2013] implementa um serviço de implantação que é semelhante à abordagem anterior, no qual define a estrutura da aplicação por meio de um modelo proprietário de topologia que é usado para provisionamento. A abordagem pesquisa maneiras diferentes de provionsar cada componente contido na topologia e executar suas operações diretamente sem gerar um plano explícito de execução. Além disso, essa abordagem baseia-se em uma DSL proprietária em vez de TOSCA o que limita a sua aplicação no que se refere à portabilidade. Qasha et ai. [Qasha et al. 2015] mostra como TOSCA pode ser usado para especificar os componentes e gerenciar o ciclo de vida de workflows científicos por mapear os elementos básicos de uma carga de trabalho real para entidades especificadas pelo TOSCA. Binz et al. [Binz et al. 2014] mostra a implementação de um conjunto de ferramentas de suporte ao pacote TOSCA, assim como o implantação e gerenciamento. Essas abordagens anteriores são complementares ao nosso trabalho, e vão incidir principalmente sobre provisionamento, pois não abordam avaliação de desempenho das aplicações.

Em relação à avaliação de desempenho de aplicações em nuvem, podemos citar vários trabalhos anteriores [Jung et al. 2013][Jayasinghe et al. 2012][Silva et al. 2013], incluindo o nosso próprio [Cunha et al. 2013]. Estes trabalhos focam-se basicamente em oferecer suporte para executar testes de desempenho de aplicações na nuvem. Nenhum destes trabalhos se baseiam na especificação TOSCA e podem ser entendidos como complementares ao trabalho deste artigo inclusive sendo utilizados como ferramenta de execução como fizemos com o Cloud Crawler.

Por isso, nós afirmamos que, com o melhor do nosso conhecimento, que o trabalho apresentado neste artigo é a melhor forma de combinar provisionamento de topologias TOSCA com a capacidade de realizar avaliações de desempenho reais para diferentes variações de topologia.

6. Conclusão e Trabalhos Futuros

A seleção de um conjunto apropriado de tipos de máquinas virtuais e configurações de implantação de uma determinada aplicação é fundamental para um uso eficaz de recursos de nuvem. Isto é particularmente verdadeiro para aplicações multi-camadas com níveis de carga mais previsíveis, para os quais soluções de reconfiguração automática existentes, como autoscaling, podem resultar em super-provisionamento de recursos. Este artigo apresentou uma abordagem que utiliza um padrão existente para

especificação e implantação de topologias de aplicações em nuvem (TOSCA) e ferramentas de implantação e avaliação de desempenho em nuvem existentes (Cloudify e Cloud Crawler, respectivamente) para facilitar a seleção das melhores topologias de implantação para uma dada aplicação em nuvem, em termos de certos atributos de qualidade. A viabilidade e a utilidade da abordagem foram ilustrados por meio de um estudo de caso realizado com a aplicação de blog WordPress na nuvem Amazon AWS, onde foram especificadas cinco topologias WordPress diferentes, implantadas, testadas e analisadas no que diz respeito ao seu custo operacional e desempenho considerando diferentes tipos de máquinas virtuais e configurações de implantação.

Como trabalho futuro, pretendemos estender a nossa abordagem e apoiar o desenvimento de um conjunto de ferramentas para analisar outros atributos de qualidade da aplicação, tais como a segurança e escalabilidade. Nós também planejamos conduzir mais experimentos para investigar se e como a nossa abordagem poderia ser generalizada para outros domínios de aplicação e plataformas em nuvem, com um foco especial em nuvens híbridas e cenários de implantação multi nuvem.

Referências

Almeida Morais, F. J., Vilar Brasileiro, F., Vigolvino Lopes, R., Araujo Santos, R., Satterfield, W., and Rosa, L. (2013). Autoflex: Service agnostic auto-scaling framework for iaas deployment models. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 42–49. IEEE.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

Binz, T., Breitenbücher, U., Kopp, O., and Leymann, F. (2014). Tosca: portable automated deployment and management of cloud applications. In *Advanced Web Services*, pages 527–549. Springer.

Borhani, A., Leitner, P., Lee, B.-S., Li, X., and Hung, T. Wpress: An application-driven performance benchmark for cloud virtual machines. *IEEE EDOC 2014*.

Breitenbucher, U., Binz, T., Képes, K., Kopp, O., Leymann, F., and Wettinger, J. (2014). Combining declarative and imperative cloud application provisioning based on tosca. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 87–96. IEEE.

Chung, L., Hill, T., Legunsen, O., Sun, Z., Dsouza, A., and Supakkul, S. (2013). A goal-oriented simulation approach for obtaining good private cloud-based system architectures. *Journal of Systems and Software*, 86(9):2242–2262.

Cunha, M., Mendonca, N., and Sampaio, A. (2013). A declarative environment for automatic performance evaluation in iaas clouds. In *2013 IEEE Sixth International Conference on Cloud Computing*, pages 285–292. IEEE.

Frey, S., Fittkau, F., and Hasselbring, W. (2013). Search-based genetic optimization for deployment and reconfiguration of software in the cloud. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 512–521. IEEE Press.

Gatling. Gatling project.

Gigaspaces. Cloudify website.

Gonçalves, M., Cunha, M., Mendonca, N. C., and Sampaio, A. (2015). Performance inference: A novel approach for planning the capacity of iaas cloud applications. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 813–820. IEEE.

Gonçalves Junior, R., Rolim, T., Sampaio, A., and Mendonça, N. C. (2015). A multi-criteria approach for assessing cloud deployment options based on non-functional requirements. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1383–1389. ACM.

Jayasinghe, D., Swint, G., Malkowski, S., Li, J., Wang, Q., Park, J., and Pu, C. (2012). Expertus: A generator approach to automate performance testing in iaas clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 115–122. IEEE.

Jung, G., Mukherjee, T., Kunde, S., Kim, H., Sharma, N., and Goetz, F. (2013). Cloudadvisor: A recommendation-as-a-service platform for cloud configuration and pricing. In *Services (SERVICES), 2013 IEEE Ninth World Congress on*, pages 456–463. IEEE.

Lu, H., Shtern, M., Simmons, B., Smit, M., and Litoiu, M. (2013). Pattern-based deployment service for next generation clouds. In *Services (SERVICES), 2013 IEEE Ninth World Congress on*, pages 464–471. IEEE.

OASIS (2013a). Topology and orchestration specification for cloud applications primer version 1.0.

OASIS (2013b). Topology and orchestration specification for cloud applications version 1.0.

Qasha, R., Cala, J., and Watson, P. (2015). Towards automated workflow deployment in the cloud using tosca. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 1037–1040. IEEE.

Saripalli, P. and Pingali, G. (2011). Madmac: Multiple attribute decision methodology for adoption of clouds. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 316–323. IEEE.

Silva, M., Hines, M. R., Gallo, D., Liu, Q., Ryu, K. D., and Da Silva, D. (2013). Cloudbench: experiment automation for cloud environments. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 302–311. IEEE.