FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA – UNIFOR

# Processo para Inferência de Desempenho de Configurações para Execução de Aplicações em Ambientes de Nuvem de Infraestrutura

**Marcelo Canário Gonçalves**

**FORTALEZA**

**2014**

Marcelo Canário Gonçalves

# Processo para Inferência de Desempenho de Configurações para Execução de Aplicações em Ambientes de Nuvem de Infraestrutura

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada (PPGIA) da Universidade de Fortaleza como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática Aplicada.

Orientador: Prof. Dr. Américo Tadeu Falcone Sampaio

Coorientador: Prof. Dr. Nabor das Chagas Mendonça

Universidade de Fortaleza

Programa de Pós-Graduação em Informática Aplicada (PPGIA)

FORTALEZA

2014

Marcelo Canário Gonçalves
Processo para Inferência de Desempenho de
Configurações para Execução de Aplicações em
Ambientes de Nuvem de Infraestrutura/ Marcelo Canário Gonçalves. – FORTALEZA, 2014-
65 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Américo Tadeu Falcone Sampaio

Dissertação (Mestrado) – Universidade de Fortaleza
Programa de Pós-Graduação em Informática Aplicada (PPGIA), 2014.

1. Cloud computing. 2. Heurísticas. I. Orientador. II. Universidade de Fortaleza. III. PPGIA. IV. Título

CDU 02:141:005.7

Marcelo Canário Gonçalves

# Processo para Inferência de Desempenho de Configurações para Execução de Aplicações em Ambientes de Nuvem de Infraestrutura

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada (PPGIA) da Universidade de Fortaleza como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática Aplicada.

Trabalho aprovado. FORTALEZA, 24 de novembro de 2012:

---

**Prof. Dr. Américo Tadeu Falcone Sampaio**
Orientador

---

**Professor**
Convidado 1

---

**Professor**
Convidado 2

FORTALEZA
2014

*À minha esposa, meu Anjo e minha luz, Isabela,*
*e à minha filha e razão de viver, Melisa.*

# Agradecimentos

Deus, Isabela e Mel, Mãe(orações de longe)

Chagas, Bento, – liberação e suporte

Jaime Gama, Paulo Benicio, José Maria – cartas de recomendacao

Matheus, – colaboração

Américo e Nabor,

Julio e Ronaldo, – momentos de dificuldade e horas de laboratorio

*"Não vos amoldeis às estruturas deste mundo,*
*mas transformai-vos pela renovação da mente,*
*a fim de distinguir qual é a vontade de Deus:*
*o que é bom, o que Lhe é agradável, o que é perfeito.*
*(Bíblia Sagrada, Romanos 12, 2)*

# Resumo

Um dos principais desafios enfrentados pelos usuários de nuvens que oferecem infraestrutura-como-serviço (IaaS) é planejar adequadamente a capacidade dos recursos da nuvem necessários às suas aplicações. Este trabalho propõe uma nova abordagem para apoiar o planejamento da capacidade de aplicações em nuvens IaaS. A nova a abordagem tem como premissa a definição de uma relação de capacidade entre as diferentes configurações de recursos oferecidas por um provedor de nuvem, com a qual é possível prever (ou "inferir"), com alto grau de precisão, o desempenho esperado de uma aplicação para determinadas configurações de recursos. A predição é realizada com base no desempenho observado para outras configurações de recursos do mesmo provedor. Dessa forma, a abordagem consegue reduzir, de forma significativa, o número total de configurações que precisam ser de fato testadas na nuvem (resultados preliminares, obtidos em um ambiente real de nuvem, mostram uma redução de mais de 80% no número total de configurações avaliadas), implicando em menores custo e tempo para o processo de planejamento.

**Palavras-chaves**: Computação em Nuvem. Planejamento de Capacidade. Inferência de Desempenho.

# Abstract

One of the main challenges faced by users of infrastructure-as-a-service (IaaS) clouds is to correctly plan the resource capacity required for their applications' needs. This work proposes a new approach to support application capacity planning in IaaS clouds. This new approach is based on the definition of a capacity relation between different resource configurations offered by a cloud provider which enables to predict (or "infer"), with a high level of accuracy, the expected performance of an application for certain resource configurations. The prediction is made based upon the observed performance for other resource configurations within the same provider. The approach significantly reduces the total number of configurations effectively tested in the cloud (preliminary results show reductions of over 80% on the number of total tested configurations) resulting in lower costs and time for the capacity planning process.

**Key-words**: Cloud Computing. Capacity Planning. Performance Inference.

# Lista de ilustrações

# Lista de tabelas

# Sumário

# 1 Introdução

## 1.1 Motivação

A computação em nuvem é um paradigma computacional que está transformando a forma de desenvolver e gerenciar aplicações e serviços de tecnologia da informação. Diversas organizações passaram a adotar este paradigma atraídas pelo seu modelo de negócios onde recursos computacionais (ex: computação, armazenamento e transferência de dados) podem ser consumidos, sob-demanda, como um serviço e pagos de acordo com o consumo efetuado. Um dos principais desafios enfrentados pelos usuários de nuvens que oferecem infraestrutura-como-serviço (IaaS) é planejar adequadamente a capacidade dos recursos da nuvem necessários às suas aplicações.

O processo de decisão pela migração de aplicações para o ambiente de nuvens computacionais envolve uma série de análises que buscam, entre outras coisas, identificar que vantagens a mudança trará de fato (LI et al., 2011; RODERO-MERINO et al., 2010), além de tentar descobrir a melhor maneira de configurar a aplicação com os vários recursos oferecidos pelo provedor de nuvem.

Um dos principais recursos computacionais oferecidos por provedores de nuvem IaaS, que normalmente representam a parte mais significativa dos gastos dos clientes, é o serviço de máquina virtual. Em geral, provedores de IaaS cobram um valor em função do tempo de utilização deste recurso, normalmente medido em horas, e esse valor unitário varia conforme o tamanho da máquina virtual (capacidade de processamento, memória e espaço de armazenamento). Dessa forma, a apuração do custo de operação de uma aplicação em um determinado período de tempo leva em conta a quantidade de máquinas virtuais utilizadas bem como seu perfil, ou seja, o tamanho e quantidade de recursos usados em cada uma.

Para prever o custo de operação de uma aplicação na nuvem, é preciso estimar ou medir como a aplicação responderá à demanda submetida em termos de indicadores de desempenho. Para se chegar a essa conclusão, faz-se necessário conhecer o comportamento da aplicação no ambiente de nuvem para que se identifiquem quais perfis de máquinas virtuais oferecidos pelo provedor são capazes de executar a aplicação com níveis satisfatórios de desempenho. Somem-se a isso as variações da demanda exercidas sobre a aplicação e as diversas possibilidades de variação da arquitetura de implantação por meio de procedimentos de escalabilidade.

Ao se tomarem procedimentos de escalabilidade vertical (variando-se a quantidade de recursos de cada máquina) e/ou de escalabilidade horizontal (variando-se a quantidade

de máquinas em uma ou mais camadas da aplicação, como dados, apresentação e negócio) chega-se a níveis de desempenho e de custo muito diversos. A variação da demanda exige que a aplicação também varie em tamanho da implantação, vertical ou horizontalmente, conforme a carga aplicada. Quanto mais acentuadas e mais frequentes as variações na demanda, mais variações de custo e desempenho serão observadas.

Assim, o custo apresenta-se entre os mais difíceis de prever, uma vez que depende necessariamente do tamanho da demanda exercida sobre a aplicação além do desempenho oferecido e preços cobrados pelo provedor de nuvem de infraestrutura contratado (CUNHA, 2012). Estrategicamente, torna-se interessante identificar, entre as possíveis composições de máquinas virtuais ofertadas em um ou vários provedores, quais são as configurações de menor custo capazes de executar a aplicação mantendo-se os níveis satisfatórios para os indicadores de desempenho.

Para facilitar o entendimento do problema suponha o seguinte exemplo. Uma determinada organização quer utilizar um provedor de nuvem para implantar uma determinada aplicação web multicamadas. Para saber se uma determinada configuração de recursos do provedor (ex: uma máquina virtual de tamanho pequeno para a camada de aplicação e outra para a a camada de dados) é capaz de atender a uma demanda específica (ex: carga imposta por 100 usuários concorrentes), é preciso antes enumerar os indicadores de desempenho que mais interessam à aplicação (ex: tempo de resposta) e a partir daí estabelecer os valores aceitáveis para esses indicadores (ex: tempo de resposta abaixo de 10 segundos para uma determinado conjunto de operações com 100 usuários). Uma vez estabelecidos os valores aceitáveis, pode-se implantar a aplicação sob essa configuração de recursos no ambiente da nuvem e então testar o desempenho da aplicação. Ao comparar a resposta da aplicação com os valores dados como aceitáveis para os indicadores, é possível determinar se aquela configuração de recursos escolhida é capaz de executar a aplicação a contento e ainda calcular o custo mensal dessa implantação.

Porém, partindo-se do pressuposto de que o desempenho da aplicação foi satisfatório, o que se tem até agora é o custo de uma única configuração capaz de executar a aplicação estudada sob um único nível de carga de trabalho. No entanto, cargas de trabalho costumam variar em função do tempo em implantações reais, fazendo-se necessário, portanto, que esse efeito seja contemplado nos testes por meio da medição do desempenho da aplicação submetida a diferentes níveis de carga de trabalho.

Analogamente, as diversas configurações de máquinas e recursos, ainda que no mesmo provedor, podem responder de maneira diferente sob o mesmo nível de carga de trabalho a depender do momento em que sejam ativados (CUNHA; MENDONÇA; SAMPAIO, 2011; IOSUP; YIGITBASI; EPEMA, 2011; JAYASINGHE et al., 2011). Independente do motivo que leve a esse comportamento de certa forma imprevisível, é preciso levar em conta nos ensaios de avaliação de desempenho essa variabilidade e isso

pode ser alcançado através da repetição dos cenários de teste em horários e dias diferentes.

Um grande problema começa a se desenhar ao seguir essa abordagem: a fase de experimentação pode atingir patamares elevados de custo, a depender das necessidades de variação da demanda, da arquitetura de implantação e das configurações utilizadas em cada arquitetura implantada (SILVA et al., 2013). Ainda que certos provedores IaaS ofereçam descontos ou pacotes de horas grátis para novos clientes, em geral esses incentivos são suficientes para custear apenas um mês de utilização de uma única máquina virtual muito pequena, provavelmente incapaz de suportar a carga de uma aplicação real em produção. Assim, executar uma aplicação real, tipicamente implantada em arquitetura de várias camadas, em máquinas virtuais de tamanho considerável e por longos períodos de tempo apenas para estudar o seu comportamento, pode se traduzir em um custo alto que inviabilize o próprio projeto de migração dessa aplicação para a nuvem.

Com o intuito de resolver este problema algumas abordagens (ref XXX) já foram desenvolvidas e se organizam em duas categorias: preditiva e empírica. Normalmente baseadas em simuladores, estas abordagens apresentam uma baixa eficiência (Colocar referencia XXX) com relação a sua capacidade de prever as melhores configurações dos recursos da nuvem para variadas cargas impostas na aplicação. Isto se deve normalmente a fatores como: diferenças conceituais entre o modelo de simulação e o modelo real de comportamento da aplicação e instabilidade dos serviços de nuvem. Uma vantagem das abordagens preditivas é não onerar os clientes com os testes pois não exigem implantação real no ambiente da nuvem.

As abordagens empíricas são baseadas na implantação das aplicações alvo na nuvem e em testes de carga. As abordagens normalmente oferecem suporte aos usuários para automatizar as taferas de instalar, configurar e testar as aplicações no ambiente da nuvem eliminando uma série de trabalhos manuais custosos. Por executarem no ambuiente de nuvem real elas conseguem ajudar a atingir resultados significativamente melhores no que diz respeito a seleção de recursos computacionais para cargas de trabalho específicas. No entanto, um problema ainda existente nessas abordagens é a necessidade de se testar exaustivamente uma grande quantidade de configurações de recursos e cargas de trabalho implicando em altos custos durante a fase de experimentação.

Com o intuito de combinar as vantagens das duas abordagens, este trabalho propõe uma nova maneira de apoiar o planejamento da capacidade de aplicações em nuvens IaaS. A nova abordagem tem como premissa a definição de uma relação de capacidade entre as diferentes configurações de recursos oferecidas por um provedor de nuvem, com a qual é possível prever (ou "inferir"), com alto grau de precisão, o desempenho esperado de uma aplicação para determinadas configurações de recursos. A predição é realizada com base no desempenho observado (em testes na nuvem) para outras configurações de recursos do mesmo provedor.

## 1.2 Objetivos

Este trabalho tem como principal objetivo o de propor uma abordagem de planejamento de capacidade baseado nas relações de capacidade existentes em recursos da nuvem e também em um processo de inferência de desempenho que utiliza como insumo resultados de execuções reais na nuvem. Os objetivos específicos do trabalho são:

- Definir uma forma de representar as relações de capacidade entre configurações de recurso de um provedor de nuvem;
- Propor um processo de planejamento de capacidade que seja capaz de prever as melhores configurações de recurso capazes de suportar variadas cargas de trabalho;
- Fornecer uma implementação concreta do processo proposto;
- Avaliar a eficiência e efetividade do processo através de um estudo baseado em uma aplicação concreta em um provedor de nuvem real.

## 1.3 Contribuições

Com o intuito de resolver a problemática do planejamento de capacidade na nuvem atendendo aos objetivos propostos algumas contribuições podem ser destacadas deste trabalho.

Primeiramente propusemos um modelo que representa as relações de capacidade de configurações de uma nuvem. Este modelo é chamado de Espaço de Implantação e consiste de uma estrutura em grafo que representa as relações entre os Tipos de Máquinas Virtuais oferecidos pelo Provedor de nuvem. O grafo é construído de forma a indicar as relações de capacidade entre os tipos de instância indicando quais tipos são maiores, menores ou não relacionados com outros.

Outra contribuição significativa é o processo de inferencia de desempenho .. falar das inovacoes, heuristicas,

capacitor –> falar das inovacoes,das

## 1.4 Estrutura da Dissertação

# 2 Trabalhos Relacionados

Apresentamos neste capítulo alguns trabalhos cujos objetivos estão alinhados com a ideia da avaliação de desempenho de aplicações executadas em ambientes de computação em nuvem. Esses trabalhos estão agrupados de acordo com a abordagem utilizada para a realização da avaliação de desempenho. São duas abordagens, a primeira é a abordagem preditiva, nesta, os trabalhos não executam diretamente a aplicação alvo no ambiente onde se deseja implantá-la. Já a segunda abordagem é a empírica, nesta as aplicações alvo são implantadas na nuvem e então submetidas a testes de carga.

Para apoiar a análise crítica de cada trabalho e suas abordagens, definimos um conjunto de critérios a partir dos quais poderemos tanto descrever, quanto comparar as soluções existentes de apoio a avaliação de desempenho do ponto de vista do usuário. Seguem os critérios:

1. **Completude da solução**
    a) **Definição da aplicação** — flexibilidade da solução para definir a aplicação a ser avaliada;
    b) **Definição da demanda** — flexibilidade da solução para definir os níveis de carga de trabalho sobre os quais a aplicação será submetida;
    c) **Definição dos recursos da nuvem** — flexibilidade da solução para definir as configurações de recursos do provedor sobre as quais a aplicação será executada;
    d) **Definição do acordo de nível de serviço (SLA)** — flexibilidade da solução para definir o SLA desejado.
2. **Efetividade da solução**
    a) **Eficiência** — tempo e custo necessários para a solução resolver o problema;
    b) **Acurácia** — confiabilidade das respostas oferecidas pela solução;
    c) **Complexidade** — grau de complexidade/esforço exigido do usuário da solução.

Cada abordagem será apresentada em uma subseção, bem como os diversos trabalhos da área que serão resumidos separadamente. Ao final da subseção, será apresentada uma análise crítica que será baseada nos critérios descritos acima.

## 2.1 Ferramentas com Abordagem Preditiva

As ferramentas que serão apresentadas nesta seção têm em comum o fato de indicarem a configuração de implantação na nuvem sem executarem avaliações de desempenho diretamente na aplicação que será migrada para a nuvem. Dessa forma, elas prevêem a configuração de implantação, por isso estão sendo chamadas de abordagens preditivas. Muitas das soluções de abordagem preditiva realizam as suas predições após a caracterização da performance dos recurso da nuvem, o que é realizado através da execução de *benchmarks*. Já o *CloudProphet*, apresentado em (LI et al., 2011), coleta como a aplicação alvo faz uso dos recursos computacionais em um ambiente controlado e então repete essa utilização em uma nuvem candidata. Dessa forma, não necessita caracterizar a performance dos recursos da nuvem.

A seguir, apresentaremos seis trabalhos que se destacam na avaliação de desempenho de aplicações na nuvem usando uma abordagem preditiva, são eles: (CLOUDHARMONY, 2014), (MALKOWSKI et al., 2010), (LI et al., 2010), (JUNG et al., 2013), (FITTKAU; FREY; HASSELBRING, 2012) e (LI et al., 2011).

### 2.1.1 Cloud Harmony

O projeto *CloudHarmony*, cujo objetivo é "tornar-se a principal fonte independente, imparcial e útil de métricas de desempenho de provedores de nuvem" (CLOUDHARMONY, 2014), agrega dados de testes de desempenho realizados desde 2009 em mais de 60 provedores de nuvem. Conforme descrito em (CUNHA, 2012), além do histórico das avaliações, o *CloudHarmony* disponibiliza uma ferramenta para executar novas avaliações de desempenho a qualquer momento, denominada *Cloud Speed Test*,[1], a qual permite realizar quatro tipos de teste:

***Download a few large files*** — objetiva determinar o melhor provedor para descarregar arquivos grandes, sendo útil para aplicações como *video streaming*;

***Download many small files*** — objetiva determinar o melhor provedor para descarregar arquivos pequenos, podendo ser útil para hospedar uma página web, por exemplo;

***Upload*** — útil para avaliar serviços que serão utilizados para envio de arquivos;

***Test network latency*** — a latência afeta o tempo de resposta da aplicação e geralmente está relacionada com a região de onde o teste está partindo.

Os resultados disponibilizados pelo *CloudHarmony* têm como pontos fortes a grande quantidade de dados de testes de desempenho disponíveis, além da possibilidade do cliente

[1] <http://cloudharmony.com/speedtest>

da nuvem poder executar novos testes a qualquer tempo. Por outro lado, os testes estão limitados àqueles implementados pela ferramenta de teste, não podendo ser facilmente modificados para contemplar novas métricas ou cenários de avaliação.

### 2.1.2 *CloudXplor*

*CloudXplor* (MALKOWSKI et al., 2010) é uma ferramenta para planejamento de configuração de recursos da nuvem baseada em dados empíricos. A ferramenta foi desenvolvida tomando como base um modelo de planejamento de configuração de recursos de Tecnologia da Informação (TI), com foco explícito em aspectos econômicos. Por essa razão, a ferramenta se utiliza de acordos de nível de serviço baseados na relação do custo da infraestrutura de TI com o valor dos recursos do provedor do serviço. Esse valor será maior quando o tempo de resposta da aplicação for plenamente atendido pelo provedor do serviço, e vai diminuindo à medida em que esse tempo de resposta não é alcançado.

Os dados empíricos precisam ser coletados, previamente, através da execução de diversos experimentos de avaliação de desempenho. Esses dados são compostos por métricas de sistema (uso de CPU, memória utilizada, tráfego na rede e E/S de disco) e métricas de mais alto nível (tempo de resposta e *throughput*). Após a coleta, os dados dos experimentos são submetidos e analisados pela ferramenta, utilizando um de seus quatro módulos: análise de tempo de resposta, análise de *throughput*, análise do valor agregado e do custo, e análise do lucro. Cada um desses módulos filtra os dados, fazendo uso apenas das informações necessárias para a execução da sua análise. Após a análise dos dados, a ferramenta pode ser utilizada para produzir gráficos que ilustram o comportamento da aplicação ao se variar parâmetros como carga de trabalho e configuração dos componentes da aplicação.

### 2.1.3 CloudCmp

Li et al. (2010) apresentam uma ferramenta para apoiar a avaliação e a comparação do desempenho e do custo dos recursos e serviços de diversos provedores de nuvem pública, de modo a auxiliar o cliente da nuvem a escolher o provedor mais adequado para a sua aplicação. Essa ferramenta, denominada *CloudCmp*, analisa os serviços de elasticidade, persistência de dados e rede oferecidos pelos provedores de interesse, com base em resultados previamente coletados a partir da execução de diversos *benchmarks*: uma versão modificada do *SPECjvm2008* (SPEC, 2008) para avaliar a característica de elasticidade do provedor; um cliente Java para avaliar os serviços de armazenamento e persistência de dados; e as ferramentas *iperf*[2] e *ping* para avaliar os serviços de rede. Após a fase inicial da coleta dos dados, a ferramenta pode ser utilizada para gerar gráficos que auxiliem o cliente da nuvem a comparar o desempenho dos recursos de cada um dos provedores nos quais as avaliações

[2] http://iperf.sourceforge.net.

foram realizadas, que assim poderá escolher o provedor e os recursos mais apropriados para as necessidades e demandas específicas de suas aplicações.

Segundo Li et al. (2010), até a época do trabalho não houve nenhum provedor de nuvem que se destacasse com relação aos demais. Outra constatação foi de que os resultados obtidos a partir da execução dos *benchmarks* em cada provedor apenas refletiam o momento em que foram coletados, uma vez que a estrutura utilizada pelos provedores para hospedar seus serviços sofre frequentes modificações e a demanda por seus recursos computacionais é bastante variável.

### 2.1.4 CloudAdvisor

O trabalho apresentado em (JUNG et al., 2013) "introduz uma nova plataforma de recomendação de nuvem, chamada *CloudAdvisor*". Essa plataforma destina-se a auxiliar o seu usuário na tarefa de capturar as implicações monetárias e financeiras das configurações de implantação das suas aplicações. Para recomendar a configuração, a platforma recebe como entrada parâmetros de configuração de alto nível como, orçamento, expectativa de performance e economia de energia, os quais estão limitados a uma escala discreta que vai de 0 até 10, onde 0 significa baixa influência e 10 significa alta influência. Uma vez informados os parâmetros de configuração, o *CloudAdvisor* irá caracterizar a performance da aplicação alvo em termos de uso dos recursos computacionais e em seguida executará o *benchmark CloudMeter*, (JUNG et al., 2013), nas nuvens candidatas, a fim de caracterizar a performance dos recursos dessas nuvens.

Para ilustrar o uso do *CloudAdvisor*, os autores implantaram a solução em servidores locais e em três provedores de nuvem pública, foram eles: Windows Azure (AZURE, 2012), Rackspace (RACKSPACE, 2012), and Amazon EC2 (EC2, 2012). Como resultado, foi observado que a taxa de erro da configuração para uma determinada carga foi de 10 %. No entanto, quando o usuário do ambiente escolhe parâmetros de configuração extremos (por exemplo, configurar o máximo de orçamento, de economia de energia e de nível de performance), essa taxa de erro elevou para 18 %.

Os autores concluem que o usuário da solução pode explorar diversas opções de configuração da sua aplicação na nuvem utilizando uma interface amigável e sem a necessidade de informar detalhes específicos de configuração. Além disso, mostraram que é possível utilizar uma técnica de caracterização de performance, para uma dada carga, baseada na execução de *benchmarks* em nuvens candidatas.

### 2.1.5 CDOSim

Uma solução para o desafio da escolha da configuração de implantação de uma aplicação na nuvem foi apresentada em (FITTKAU; FREY; HASSELBRING, 2012) com o

nome de *CDOSim*. Essa solução auxilia o usuário no processo de escolha do que os autores chamaram de opção de implantação na nuvem — do inglês *Cloud Deployment Option* (CDO)—, uma vez que a análise manual das "potenciais CDOs é intratável, custosa e consome tempo, devido à heterogeneidade dos ambientes de nuvem", (FITTKAU; FREY; HASSELBRING, 2012). Para a escolha das CDOs, são realizadas simulações que se baseiam no custo e nas propriedades de performance de cada CDO. O custo é informado pelo provedor de nuvem, já a caracterização da performance é realizada através da execução de um *benchmark* para medir a quantidade de *mega integer plus instructions per second* (MIPIPS) (FITTKAU; FREY; HASSELBRING, 2012), por opção de configuração. O código desse *benchmark* deve ser gerado para cada linguagem de programação utilizada pela aplicação alvo. Esta, por sua vez, deve passar por um processo de engenharia reversa para modelos KDM, que é descrito em (PÉREZ-CASTILLO; GUZMAN; PIATTINI, 2011), para que o simulator consiga escolher a opção de configuração mais adequada.

Para ilustrar o uso da solução, os autores executaram três tipos de experimentos, um para validar o uso de MIPIPS para caracterizar a performance das opções de configuração, outro para comparar os resultados da simulação com dados reais, e um último experimento para verificar a possibilidade da predição da performance de um provedor de nuvem com base nos dados de outro provedor de nuvem. Esses experimentos foram conduzidos em dois ambientes de nuvem, sendo uma pública, a Amazon EC2, e outra privada. Na nuvem pública foi evidenciado que os valores de MIPIPS dependem da região onde o *benchmark* foi realizado e da carga sobre a máquina física que hospeda a máquina virtual. Já na comparação dos resultados da simulação com os dados reais, os autores mostraram que a taxa de error da utilização de CPU simulada com a utilização de CPU medida, chegou a 30,86 %. Contudo, a taxa de erro médio global foi abaixo de 22,75 %, portanto abaixo do limiar estabelecido pelos autores que era de 30 %. Já a predição da performance de uma instância da Amazon EC2 a partir da performance de uma instância da nuvem privada gerou 15,76 % como taxa de erro global.

Finalmente, os autores concluem que a simulação pode auxiliar no processo de escolha da opção de implantação com maior performance e menor custo e que os resultados da simulação são razoavelmente próximos dos valores reais.

### 2.1.6 CloudProphet

Em (LI et al., 2011) os autores apresentam o *CloudProphet*, um sistema de predição de desempenho de aplicações em ambiente de nuvem computacional baseado na metodologia de "rastrear e reproduzir" (*trace and replay*).

O *CloudProphet* não testa a aplicação do cliente de fato no ambiente de nuvem. De modo contrário, ele injeta na implantação original da aplicação um módulo que registra um rastreamento detalhado dos eventos de utilização de recursos de CPU, armazenamento

e rede em cada componente da aplicação durante um período de execução habitual em seu ambiente de produção.

Em um passo seguinte, outro módulo faz uma extração das relações de dependência entre os eventos coletados, ordenando as transações executadas nos diversos componentes.

O terceiro passo é a reprodução dos eventos coletados durante a fase de rastreamento. Essa reprodução consiste fazer com que o ambiente de nuvem computacional que se deseja avaliar execute as transações representadas nos dados do rastreamento a partir de requisições que partem de clientes simulados.

O objetivo do *CloudBench*, segundo os autores é eliminar o custo e o trabalho envolvidos na migração da aplicação real para a nuvem para a execução de testes antes que seja de fato tomada a decisão em favor dessa migração.

Os autores argumentam que a simples implantação da aplicação no ambiente de um serviço de nuvem computacional já incorre em custos, que podem ser altos a depender do tamanho ou da arquitetura da aplicação. Além disso, a tarefa de migração pode ser bastante trabalhosa conforme o número e a diversidade dos componentes da aplicação, que podem acarretar dificuldades de configuração e compatibilidade no novo ambiente.

## 2.2 Ferramentas com Abordagem Empírica

As ferramentas que serão apresentadas nesta seção têm em comum o fato de utilizarem como aplicação alvo a pória aplicação que se deseja implantar na nuvem. Portanto, é preciso inicialmente realizar uma implantação na nuvem para que seja dado início ao processo de análise de desempenho. Por isso, cada ferramenta oferece um mecanismo para que o seu usuário possa definir como a aplicação deve ser implantada e configurada. Além da definição da aplicação e dos recursos da nuvem que serão utilizados, essas ferramentas também permitem que sejam definidas a demanda que será imposta a cada aplicação e o acordo de nível de serviço. Dessa forma, é possível definir, por exemplo, o número de usuários simultâneos e o tempo de resposta esperado para uma transação.

Uma vez que que as ferramentas que realizam a abordagem empírica possibilitam ao usuário muita liberdade na definição da aplicação, demanda, recurso da nuvem e SLA, essas soluções têm o mais alto grau de completude. Além disso, como faz-se uso da própria aplicação alvo para a avaliação de desempenho, a acurácia dos resultados apresentados pelas ferramentas é a mais elevada. A seguir, apresentaremos quatro trabalhos que se destacam na avaliação de desempenho de aplicações na nuvem usando a abordagem empírica, são eles: (JAYASINGHE et al., 2012), (SILVA et al., 2013), (CUNHA; MENDONÇA; SAMPAIO, 2013) e (SCHEUNER et al., 2014).

### 2.2.1 Expertus

Devido à complexidade e às implicações da escolha da configuração para a implantação de uma aplicação na nuvem, em (JAYASINGHE et al., 2012) os autores apresentam o *Expertus*, que é descrito como "um framework flexível de geração de código para automatizar testes de performance de aplicações distribuídas em nuvem de infraestrutura". Essa geração automática de código é realizada a partir de *templates* especificados na forma de documentos XML (JAYASINGHE et al., 2012). Os templates utilizados nas avaliações de desempenho devem ser escritos pelo usuário e servem de entrada para o ambiente, que realiza diversas transformações nessa entrada até a forma de *shell scripts*. Esses *scripts*, por fim, possuem os comandos para a configuração da avaliação de desempenho na aplicação alvo.

Como demonstração da usabilidade da ferramenta, os autores apresentaram em (JAYASINGHE et al., 2012) resultados de experimentos realizados com duas aplicações alvo. Cada uma das aplicações foi avaliada com duas opções de sistemas de gerenciamento de bancos de dados, o que demonstrou também como diferentes opções de configuração poderiam ser utilizadas nas aplicações. Além da demonstração da usabilidade, os autores realizaram experimentos para evidenciar a magniture e tipos de *scripts* que podem ser gerados pela ferramenta. Como exemplo da magnitute, para a realização de experimentos com 48 nós, o total de linhas de scripts geradas pelo *Expertus* girou em torno de 15 mil. Por fim, os autores demonstraram o que chamaram de "riqueza da ferramenta", que foi comprovada através da execução de experimentos em 5 nuvens (por exemplo, Amazon EC2 e Open Cirrus (AVETISYAN et al., 2010)).

Dessa forma, pode-se concluir que a ferramenta apresentada minimiza a ocorrência de falhas humanas na avaliação de desempenho de uma aplicação implantada em diversos nós. Além disso, os mesmo experimentos podem ser repetidos em diferentes provedores de nuvem pública. De modo que mais cenários de implantação podem ser considerados para a escolha do mais adequado para a aplicação.

### 2.2.2 CloudBench

(SILVA et al., 2013) descreve o *CloudBench* como um arcabouço para automação da avaliação de desempenho de ambientes de nuvem computacional sob o modelo IaaS. As abstrações apresentadas neste trabalho permitem que um experimento seja especificado através de uma lista de diretivas as quais descrevem os itens que compõem o experimento. São exemplos desses itens, objetos como a aplicação alvo, as instâncias de máquinas virtuais utilizadas, e as métricas de desempenho que são tanto relativas à aplicação alvo, quanto ao serviço do provedor de nuvem (por exemplo, latência de provisionamento).

Para a realização dos experimentos, o *CloudBench* faz a implantação automática da

aplicação a ser executada para efeito de testes. Portanto, o acompanhamento é realizado desde a criação da máquina virtual no ambiente até a coleta dos dados de desempenho e desligamento das máquinas. Essas características fazem do CloudBench uma ferramenta muito poderosa para a automação de testes e coleta de dados para análise das execuções. Suas ferramentas de monitoramento fornecem informações com grandes níveis de detalhamento a respeito de cada componente implantado e usado nos testes, proporcionando excelente embasamento para a tomada de decisão.

Entretanto, embora o CloudBench tenha um escopo de solução muito mais amplo, voltado para a avaliação de desempenho tanto da aplicação do cliente como do provisionamento de máquinas pelo provedor, seu alvo no momento da execução de testes está restrito a *benchmarks* pré-definidos, não permitindo a execução de uma aplicação real no ambiente testado.

### 2.2.3 Cloud Crawler

Este trabalho apresenta um ambiente programável para apoiar os usuários de nuvens IaaS na realização de testes automáticos de desempenho de aplicações na nuvem. As principais contribuições do ambiente são: a linguagem declarativa *Crawl*, com a qual os usuários podem especificar, através de uma notação simples e de alto nível de abstração, uma grande variedade de cenários de avaliação de desempenho de uma aplicação na nuvem; e o motor de execução *Crawler*, que automaticamente executa e coleta os resultados dos cenários descritos em *Crawl* em um ou mais provedores. Essas duas ferramentas são denominadas conjuntamente de *Cloud Crawler* (CUNHA; MENDONÇA; SAMPAIO, 2013).

Para iniciar os testes de desempenho de uma aplicação através do ambiente *Cloud Crawler*, os componentes dessa aplicação precisam ser declarados em um *script* da linguagem *Crawl*. Compõem esse *script Crawl*, por exemplo, o provedor de nuvem, os tipos de máquinas virtuais e as máquinas virtuais que serão utilizadas nas avaliações, além disso, métricas de desempenho e a demanda imposta à aplicação também irão compor o cenário de avaliação que é declarado no *script Crawl*. Finalizada essa etapa de declaração, o usuário do ambiente irá submeter o *script crawl* para o motor de execução *Crawler*. Esse motor irá iniciar todas as máquinas virtuais, caso seja necessário, irá proceder com a modificação do tipo de máquina virtual, de acordo com o que estiver declarado. Após a inicialização de cada máquina virtual, o motor pode executar alguma configuração nessa máquina, por exemplo, a configuração do endereço ip de um banco de dados, ou a configuração do total de memória utilizado por uma máquina virtual java. Todas as configurações necessárias para a aplicação executar na nuvem devem estar declaradas no *script Crawl* que foi submetido para o motor. Quando a última máquina virtual é configurada, o motor *Crawler* executa um por um os cenários de avaliação, com suas respectivas demandas, e ao mesmo tempo

coleta as métricas de desempenho especificadas. As métricas de desempenho podem ser tanto métricas de sistema, como percentual de CPU utilizado e de memória RAM, quanto métricas de aplicação, como o tempo de resposta de uma aplicação WEB.

A fase de mapeamento dos componentes da aplicação é realizada apenas uma vez, enquanto que a submissão para o motor de execução pode ser repetida ao critério do usuário. Ambientes como o *Cloud Crawler* permitem que os seus usuários testem suas aplicaçãoes em difirentes cenário de implantação e possibilitam que o mesmo entenda o comportamento da sua aplicação à medida em que ela é submetida a diferentes demandas e implantada em diferentes configurações, porém, a qualidade da avaliação de desempenho dependerá da qualidade dos cenários de testes que os usuários declararem, uma vez que o ambiente não decide qual será a nova configuração testada. O ambiente apenas segue aquilo que foi declarado pelo usuário.

### 2.2.4 Cloud WorkBench

Uma vez que a escolha da infraestrutura computacional ótima para hospedar uma determinada aplicação na nuvem se trata de uma tarefa que "exige a avaliação de custos e performances de diferentes combinações de configurações" (SCHEUNER et al., 2014). Onde os autores propõem uma arquitetura e uma implementação concreta dessa arquitetura para automatizar a realização de avaliações em serviços da nuvem. O *Cloud WorkBench*, nome dado à solução apresentada neste trabalho, adota noções de Infraestrutura como Código, do inglês *Infrastructure-as-Code* (IaC) (HÜTTERMANN, 2012), para a realização dessas avalições. Dessa forma, as ações necessárias para o provisionamento dos recursos utilizados pela aplicação encontram-se todas codificadas.

Para ilustrar o uso do *Cloud WorkBench*, foi realizado um pequeno experimento para avaliar a velocidade de escrita sequencial em disco de recursos da nuvem. Nesse experimento, foram utilizados três perfis de recursos computacionais da Amazon EC2 na região Irlanda (*eu-west-1*), em servidores utilizando o sistema operacional Ubuntu 14.04. Para cada perfil de recurso, foram realizadas entre 8 e 12 execuções do benchmark FIO[3] 2.1.10. Conforme descrito em (SCHEUNER et al., 2014), o experimento evidenciou que há diferenças na performance dos perfis de recursos utilizados. Esse diferença poderia se refletir na performance de uma aplicação que fizesse muita escrita em disco.

Após a análise dos resultados, os autores concluem que o *Cloud WorkBench* suporta a realização de experimentos em nuvens de infraestrutura, e que toda a complexidade da configuração do ambiente pode ficar codificada. O que diminui a ocorrência de erros decorrentes de eventuais intervenções manuais.

[3] http://git.kernel.org/cgit/linux/kernel/git/axboe/fio.git

## 2.3 Análise Crítica

### 2.3.1 Abordagem Preditiva

As soluções de abordagem preditiva possuem desempenho bastante variados nos critérios de desempenho estabelecidos no início deste capítulo, porém elas acabam convergindo na efetividade, que comproente eficiência, acurácia e complexidade. No critério de eficência, as soluções se destacam, possuem alta eficiência, uma vez que não requerem a alocação de recursos de nuvem para realizarem as predições. No entanto, o *CloudProphet* é a única solução de abordagem preditiva com baixa eficiência, pois requer a alocações de recursos da nuvem para avaliar todas as demandas e configurações. Já com relação à acurácia, as soluções tem um desempenho moderado, com distinção para a solução apresentada em (FITTKAU; FREY; HASSELBRING, 2012), que possui baixa acurácia, conforme fica evidenciado nos resultados apresentados na subseção 2.1.5. Finalmente, no que diz respeito à complexidade, a solução com menor complexidade, portanto com maior destaque, é a *CloudHarmony* a qual permite que os testes sejam iniciados e que as pesquisas de resultados anteriores sejam realizadas através de uma interface amigável, sem a necessidade de intervenções do usuário. Já as demais possuem complexidade moderada.

Ainda com base nos critérios de avaliação, as soluções apresentadas em (MALKOWSKI et al., 2010; CLOUDHARMONY, 2014) possuem baixa completude, pois não permitem que sejam definidos aplicação, demanda, recurso da nuvem e SLA. Já o *CloudAdvisor* apresentando em (JUNG et al., 2013), permite a definição da aplicação, e da carga, porém não permite a escolha do SLA e não faz referência a respeito do uso de nuvens públicas diferentes das apresentadas nos resultados. Dessa forma, com relação à completude, o *CloudAdvisor* tem um desempenho moderado. Das abordagens preditivas, o *CloudProphet* é a solução que se destaque no critério da completude, pois permite que o usuário defina aplicação, demanda, recurso da nuvem e SLA desejado.

### 2.3.2 Abordagem Empírica

As soluções de abordagem empírica, por sua vez, possuem desempenhos muito parecidos tanto na completude, quanto na efetividade. Com relação à completude, que abrange a definição da aplicação, da demanda, dos recursos da nuvem e do SLA, as soluções possuem alto desempenho. Pois permitem muita liberdade nessas definições. Por exemplo, na solução apresentada em (CUNHA; MENDONÇA; SAMPAIO, 2013), o usuário pode definir toda a pilha de componentes da aplicação alvo, todos cenários utilizados na avaliação de desempenho, as demandas que serão submetidas a cada um dos cenários e o critério que define se o cenário suportou a demanda, ou seja, o SLA. Da mesma forma, as soluções apresentadas em (JAYASINGHE et al., 2012; SILVA et al., 2013; SCHEUNER et al., 2014), possuem os seus mecanismos para a realização dessas definições.

Apesar das soluções de abordagem empírica terem destaque na completude, no que diz respeito à efetividade, que abrange eficiência, acurácia e complexidade, essas soluções possuem desempenho moderado. No critério da eficiência, essas ferramentas deveriam fazer uso de resultados anteriores para evitar a execução de testes que claramente poderiam ser evitados. Por exemplo, em uma situação na qual uma demanda é submetida à aplicação que está sendo executada em uma máquina virtual com baixo poder computacional, seria coerente afirmar que essa mesma demanda pode ser atendida por máquinas com perfis computacionais mais robustos. Devido à essa necessidade de executar os testes em cada uma das configurações, sem fazer uso de resultados anteriores, as soluções de abordagem empírica possuem baixa eficiência. Por outro lado, como irão executar de fato a aplicação e submetê-la à demanda especificada, essas soluções possuem alta acurácia. Já no que diz respeito à complexidade, o desempenho é considerado moderado. Uma vez que cada trabalho faz uso de uma estratégia de uso particular, e que as experiências anteriores do usuário irão ser determinantes na percepção da complexidade. Esse usuário precisa se adaptar à sintaxe de cada solução, e eventualmente, configurar imagens contendo os componentes da aplicação que será avaliada.

# 3 Avaliação de Capacidade

Este capítulo apresenta uma proposta de processo de avaliação de capacidade que visa a buscar as Configurações de menor preço capazes de executar uma determinada Carga de Trabalho. Esse processo foi implementado como parte de um sistema computacional ao qual demos o nome de *Cloud Capacitor* e que descreveremos no capítulo seguinte.

Antes de descrever o processo, é preciso que seja apresentado um conjunto de definições e terminologias que baseiam o entendimento da construção do trabalho e também a avaliação dos resultados, bem como a eficiência e eficácia da solução proposta. Os conceitos aqui explicitados são tomados de forma a permitir um estudo agnóstico quanto a aplicações, plataformas e provedores utilizados durante a execução das ferramentas desenvolvidas neste trabalho.

## 3.1 Definições e Terminologias

Apresentamos a seguir as definições que permeiam o conhecimento necessário para a análise dos problemas estudados e soluções propostas. Mostramos também as terminologias ou nomenclaturas que criamos para designar esses conceitos a fim de facilitar a comunicação e o entendimento por parte do leitor.

### 3.1.1 Aplicação sob Teste

A Aplicação sob Teste é um sistema computacional, possivelmente implementado em arquitetura multicamadas, para o qual se deseja observar o comportamento em um ambiente de computação em nuvem e ao qual estão ligadas uma ou mais Métricas de Desempenho.

### 3.1.2 Métrica de Desempenho

Uma característica ou comportamento mensurável de forma automatizada e comparável a um Valor de Referência capaz de indicar o grau de sucesso de uma execução da Aplicação. Ex. tempo de resposta, quadros por segundo, etc. Métricas podem ser minimizáveis ou maximizáveis, a depender do objetivo da métrica quanto ao resultado desejado. Por exemplo, “tempo de resposta” é uma métrica minimizável, uma vez que geralmente se deseja que uma Aplicação responda a uma requisição com o menor tempo de resposta possível nos resultados. Contrariamente, uma métrica “quadros renderizados por segundo”, no domínio da computação gráfica, é uma métrica maximizável, pois quanto

mais quadros são renderizados por unidade de tempo, maior a qualidade percebida pelo usuário.

### 3.1.3 Valor de Referência de Desempenho ou SLA

Um valor predefinido como minimamente aceitável como resultado apresentado por uma Métrica após uma Execução da Aplicação sob Teste. Este valor, também referenciado neste trabalho como SLA (Service Level Agreement), serve como base de comparação para que se classifique a Aplicação como capaz de ser executada em um determinado arranjo de máquinas virtuais e sob uma determinada Carga de Trabalho a ela imposta.

### 3.1.4 Provedor

Consideramos neste trabalho a figura do provedor como representando uma empresa que fornece infraestrutura computacional como serviço cobrado financeiramente por fração de tempo de utilização. Alguns provedores fornecem conjuntamente a modalidade de plataforma como serviço. Nós, porém, não estamos considerando essa modalidade neste trabalho, interessando-nos apenas os serviços de infraestrutura, notadamente a disponibilização de máquinas virtuais.

### 3.1.5 Tipos de Máquinas Virtuais

Provedores costumam classificar as máquinas virtuais fornecidas conforme suas características, de modo a manter uma linha de produtos discreta e finita. Normalmente essa classificação se dá em termos de quantidade de memória RAM, quantidade de espaço em disco e capacidade computacional, neste caso, quer seja em termos relativos a um valor padrão tomado como base, quer seja em termos absolutos, como número de CPUs virtuais.

### 3.1.6 Categorias de Máquinas Virtuais

Tipos de Máquinas Virtuais podem ser agrupados em Categorias, conforme suas características físicas, plataforma e/ou arquitetura de hardware e a natureza do uso a que se destinam. Dentro de uma mesma Categoria, os Tipos de Máquinas Virtuais variam apenas na quantidade de cada um dos recursos especificados para a Categoria e no preço cobrado pelo uso das máquinas virtuais.

Como exemplo, podemos citar uma Categoria de máquinas destinadas a armazenamento de arquivos, onde as máquinas devem privilegiar o espaço de armazenamento em massa. Dentro dessa categoria, a principal diferença entre os Tipos de Máquinas Virtuais se dá em função da quantidade de espaço em disco disponibilizado, enquanto características como memória RAM e CPU teriam pequenas variações. Outras Categorias podem enfatizar o consumo de banda de rede ou processamento paralelo de alto desempenho.

### 3.1.7 Configurações

Chamamos de Configuração um conjunto de máquinas virtuais pertencentes ao mesmo Tipo de Máquinas Virtuais e, portanto, de uma mesma Categoria. Configurações são usadas para implantar uma camada arquitetural da Aplicação sob Teste (apresentação, negócio, persistência, etc.) e representam o estado de uma determinada camada da aplicação quanto à sua escalabilidade, vertical ou horizontal.

Por exemplo, suponhamos avaliação do comportamento de uma Aplicação cuja camada de negócios está implantada em arquitetura de cluster de servidores de aplicação. Variando a quantidade de máquinas que compõem esse cluster, obtemos diferentes níveis de escalabilidade horizontal para os quais podemos avaliar o desempenho da Aplicação. Agora, suponhamos que podemos usar uma, duas, três ou quatro máquinas em paralelo como componentes do cluster da camada de negócios da Aplicação sob Teste. Assim, teríamos então quatro Configurações diferentes, a primeira com uma instância na camada de negócio, a segunda Configuração com duas instâncias, a terceira com três e a quarta com quatro instâncias de máquinas do mesmo Tipo de Máquina Virtual. A Aplicação sob Teste seria executada quatro vezes, cada uma das quais utilizando uma dessas Configurações. Os resultados dessas Execuções refletem o efeito da escalabilidade horizontal no desempenho geral da Aplicação.

Analogamente, poderiam ser usadas Configurações criadas a partir de Tipos de Máquinas Virtuais diferentes, umas mais potentes que as outras. A comparação dos resultados obtidos nesse cenário nos dão insumos para avaliar o efeito da escalabilidade vertical sobre o desempenho da Aplicação.

Os experimentos desenvolvidos e apresentados neste trabalho comparam implantações de diferentes Configurações em uma determinada camada da Aplicação sob Teste estudada. Isso permite, por exemplo, que sejam feitas avaliações como a viabilidade financeira da escalabilidade vertical face ao desempenho possivelmente obtido com a escalabilidade horizontal.

### 3.1.8 Espaço de Implantação

Chamamos de Espaço de Implantação o conjunto limitado de Configurações tomadas para execução da Aplicação sob Teste em uma sessão de avaliação.

Idealmente, uma Aplicação deveria ser testada sob todos os Tipos de Máquinas Virtuais fornecidos pelo Provedor (cobrindo todo o espaço de escalabilidade vertical) com o maior número possível de combinações de quantidade de instâncias (cobrindo o espaço de escalabilidade horizontal). Porém, se muitos Tipos de Máquinas Virtuais forem necessários e se o intervalo de número de instâncias solicitado for muito grande, o tempo de duração da sessão e o custo da muitas execuções podem se tornar proibitivos.

Assim, o processo de especificação de um Espaço de Implantação consiste em selecionar uma lista de Tipos de Máquinas Virtuais entre os oferecidos pelo Provedor e designar o melhor valor para o número máximo de instâncias que serão usadas na criação das Configurações. Isso faz com que ambos os espaços de escalabilidade vertical e horizontal sejam limitados, de forma a controlar melhor os custos e permitir que sejam executados testes mais objetivos e de acordo com a meta de Carga de Trabalho a ser atendida pela Aplicação.

### 3.1.9 Carga de Trabalho

A Carga de Trabalho, ou Workload, representa o tamanho da demanda que será imposta à Aplicação sob Teste em uma execução. A unidade de medida da Carga é dependente do domínio da Aplicação, como a duração do vídeo em uma aplicação de transcodificação de arquivos multimídia ou o tamanho do arquivos de entrada para uma aplicação de compactação de arquivos. Entretanto, para efeito deste trabalho, essa unidade de medida da Carga é irrelevante, uma vez que a responsabilidade pela execução dos testes e, por conseguinte, pela geração da carga, é delegada a um módulo à parte dentro do sistema de avaliação, como um software de *benchmarking*.

### 3.1.10 Execução

Dá-se o nome de Execução ao evento de utilização de uma Configuração para executar a Aplicação sob Teste submetida a uma determinada Carga de Trabalho. Dessa forma, a avaliação dos Resultados de uma Execução nos dará uma ideia de como a Aplicação responderá às requisições de certo número de usuários (Workload) após ser implantada num ambiente de nuvem com certo grau de escalabilidade horizontal (número de máquinas virtuais usadas).

Tome-se como exemplo uma aplicação web muito comum, um blog. São requisições comuns a um blog o acesso à página principal, uma consulta às postagens de uma categoria e o acesso a uma postagem específica. Agora, suponha-se uma Configuração composta de três instâncias de máquinas virtuais do Provedor Rackspace, do Tipo "Performance 1", executando uma instalação do blog Wordpress, muito usado hoje em dia. Uma Execução dessa aplicação seria a imposição da Carga de Trabalho correspondente a um conjunto de requisições disparadas por 100 usuários simultâneos sobre essa Configuração.

Introduzidas as definições e formalizações, torna-se possível agora a especificação de uma lógica de manipulaçãoo das entidades e operações descritas. As próximas seções descrevem a proposta de um Processo de Avaliação de Capacidade por Inferência de Desempenho cujo objetivo é identificar as Configurações capazes de executar uma Aplicação respeitando um certo SLA definido.

## 3.2 Visão Geral do Processo

O processo (Figura 1) prevê um conjunto de dados de entrada, ao menos uma execução da Aplicação sob Teste no ambiente de nuvem de infraestrutura almejado para hospedá-la e a análise do desempenho obtido pela Aplicação a partir de suas execuções. Com base nos dados de desempenho, o processo passa por diversos pontos de decisão que podem levar a novas execuções da Aplicação em diferentes cenários. Ao final do processo, é fornecida como saída uma lista de Configurações, ordenadas por preço, capazes de executar a Aplicação sob cada uma das Cargas de Trabalho fornecidas como parte dos dados de entrada.

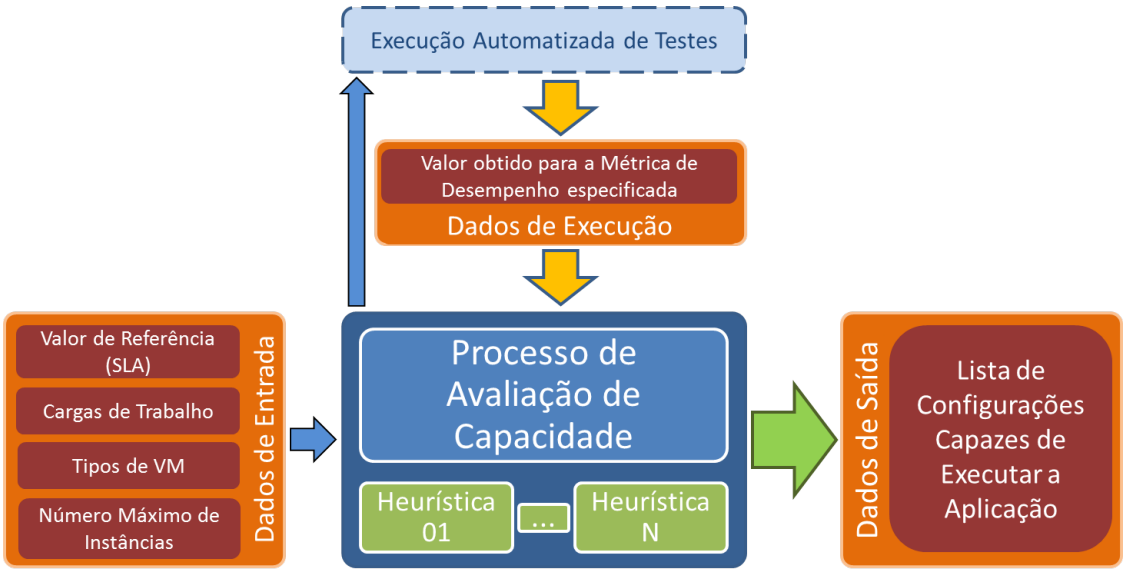



Figura 1 – Visão geral do Processo de Avaliação de Capacidade

Este capítulo aborda em detalhes todas as fases do processo proposto, explicando quais são os dados de entrada necessários, as operações executadas pelo processo e quais as decisões pelas quais o processo tem que passar até determinar quais são as Configurações de menor custo capazes de executar a Aplicação.

### 3.2.1 Dados de Entrada

O principal parâmetro esperado pelo processo de avaliação de capacidade é o Valor de Referência de Desempenho, ao qual também nos referimos como SLA (*Service Level Agreement*). Esse valor será usado para determinar se a Aplicação atingiu os requisitos mínimos de desempenho exigidos, conforme veremos na descrição do funcionamento do processo, mais adiante.

Além do SLA, o processo precisa também conhecer quais são as Cargas de Trabalho para as quais o desempenho da Aplicação sob Teste deverá ser avaliado. Porém, nem todas as Cargas de Trabalho serão impostas de fato à Aplicação. Isso vai depender do

conjunto de decisões tomadas pelo processo com base na comparação do resultado obtido pela Aplicação com o SLA. Ainda assim, graças à sua característica de inferência de desempenho, o processo mostra resultados para todas as Cargas de Trabalhado informadas como parâmetro de entrada.

Para que o desempenho da Aplicação seja avaliado, é preciso que o processo conheça quais são as Configurações disponibilizadas no Provedor de nuvem para esse fim. Para isso, o processo deve ser alimentado com uma lista de Tipos de Máquinas Virtuais que serão utilizadas na execução da Aplicação, bem como a quantidade máxima de instâncias usadas para compor cada Configuração. Através desses dados o processo passa a conhecer então o Espaço de Implantação disponível para os testes de desempenho, composto por uma lista de Configurações geradas a partir da lista de Tipos de Máquinas Virtuais disponíveis e do número máximo de instâncias.

### 3.2.2 Atividades Customizáveis

O processo de avaliação de capacidade proposto é um processo extensível, do qual fazem parte atividades customizáveis para as quais são delegadas funções de cunho mais específico, como a comunicação com o Provedor de nuvem e a Aplicação sob Teste para fins de orquestração do teste de desempenho, e também funções para as quais é desejado um certo grau de flexibilidade a fim de tornar o processo mais adaptável, como a escolha das Cargas de Trabalho e Configurações que serão usadas na execução da Aplicação.

#### 3.2.2.1 Execução dos Testes de Desempenho

Todas as atividades ligadas à rotina de execução da Aplicação, desde sua implantação, passando pela criação e configuração das máquinas virtuais no ambiente do Provedor de nuvem, bem como pelo controle de inicialização e finalização dessas instâncias, serviços subjacentes como bancos de dados e filas, e a própria parametrização da execução e parada da Aplicação em si não fazem parte do escopo do processo. Este, por sua vez, presume que os dados de resultado para cada execução estarão disponíveis quando necessários. A maneira como esses dados serão de fato obtidos é encapsulada pela implementação concreta desta atividade do processo.

Portanto, o processo prevê a customização da atividade de Execução dos Testes, que será responsável pelas ações necessárias à execução da Aplicação sob Teste no ambiente alvo. A atividade customizada deverá conhecer os detalhes inerentes à comunicação com o Provedor e com a Aplicação sob Teste e, assim, ser capaz de ordenar a sua execução e coletar como resposta os dados de desempenho esperados pelo processo. Esse é um dos pontos de extensibilidade oferecidos pelo processo e sua implementação concreta está fora do escopo deste trabalho, cujo foco não está na automação de execução de testes de qualquer natureza, mas na análise dos dados resultantes dessa execução.

#### 3.2.2.2 Estratégias e Heurísticas

De modo análogo à abordagem adotada em relação às atividades de execução dos testes, as operações de seleção da Configuração sobre as quais a Aplicação sob Teste será executada, bem como a seleção das Cargas de Trabalho a que ela será submetida durante sua execução, não são executadas diretamente pelo processo. Nesse caso, são delegadas a uma atividade customizada que chamamos de Estratégia de Avaliação ou, simplesmente, Estratégia. Seu objetivo é permitir a aplicação de diferentes métodos para a escolha da melhor Configuração e/ou Carga de Trabalho mais adequada aos objetivos da avaliação de capacidade em curso e também ao perfil da Aplicação.

Como veremos na seção 3.3, em diversas oportunidades durante a execução do processo se faz necessária a seleção de uma Configuração de maior ou menor capacidade. Do mesmo modo, em certos momentos o processo precisa que uma Carga de Trabalho menor ou maior seja selecionada. A partir dessas escolhas o processo é capaz de navegar no Espaço de Implantação submetendo a Aplicação sob Teste a diferentes cenários de Cargas de Trabalho em diversas condições de capacidade computacional.

Um problema relacionado à execução de testes de desempenho em ambientes de nuvem é que o próprio teste implica num custo financeiro que pode ser bastante elevado caso o Espaço de Implantação definido seja muito extenso. O mesmo se dá com relação à lista de Cargas de Trabalho. A fim de minimizar esse problema, este trabalho propõe a técnica de inferência de desempenho explicada detalhadamente na seção 3.3.1.1, e através da qual será possível eliminar grande parte das execuções reais da Aplicação durante os testes, reduzindo assim o custo total da avaliação.

Porém, outro problema enfrentado na busca pela melhor Configuração capaz de executar uma Aplicação está justamente no momento de selecionar, dentre um conjunto de Configurações possíveis, qual a mais promissora a ser usada para uma execução real, dado que não é conhecido previamente o potencial computacional dessas Configurações.

A fim de solucionar esse problema, este trabalho introduz o conceito das Heurísticas de Seleção, que são abordagens a serem observadas no momento em que a Estratégia de Avaliação deve escolher a próxima Configuração ou a próxima Carga de Trabalho.

Foi definido um conjunto de 3 abordagens aplicáveis ao Espaço de Implantação, ou seja, à escolha da próxima Configuração a ser testada, e outras 3 abordagens aplicáveis à lista de Cargas de Trabalho. A combinação dessas abordagens dá origem a 9 Heurísticas de Seleção, a saber:

**OO - Otimista/Otimista**

Visa Configurações menores e Cargas de Trabalho maiores

**OC - Otimista/Conservadora**

Visa Configurações menores e Cargas de Trabalho intermediárias

**OP - Otimista/Pessimista**
Visa Configurações e Cargas de Trabalho menores

**CO - Conservadora/Otimista**
Visa Configurações intermediárias e Cargas de Trabalho maiores

**CC - Conservadora/Conservadora**
Visa Configurações e Cargas de Trabalho intermediárias

**CP - Conservadora/Pessimista**
Visa Configurações intermediárias e Cargas de Trabalho menores

**PO - Pessimista/Otimista**
Visa Configurações e Cargas de Trabalho maiores

**PC - Pessimista/Conservadora**
Visa Configurações maiores e Cargas de Trabalho intermediárias

**PP - Pessimista/Pessimista**
Visa Configurações maiores e Cargas de Trabalho menores

Para um melhor entendimento de como as Heurísticas influenciam a navegação do processo entre as Configurações que compõem o Espaço de Implantação a ser explorado e também entre as Cargas de Trabalho a serem impostas sobre a Aplicação, convém observar a Figura 2.

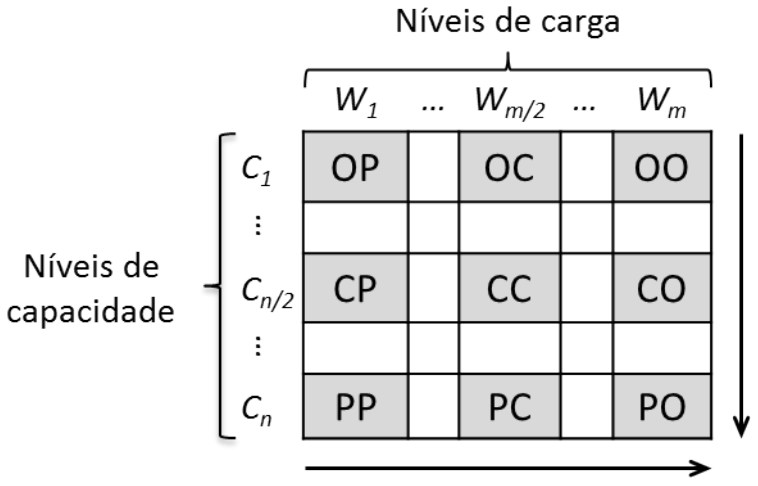



Figura 2 – Diagrama de Funcionamento das Heurísticas de Seleção

A imagem ilustra dois conjuntos dispostos em forma de matriz, um conjunto na vertical, formando as linhas, composto de $n$ Configurações $C_1 \ldots C_{n/2} \ldots C_n$ e um conjunto

composto de $m$ Cargas de Trabalho $W_1 \dots W_{m/2} \dots W_m$, formando as colunas. As células dessa matriz mostram as Heurísticas que selecionariam o par formado pela Configuração e pela Carga de Trabalho referentes à linha e coluna da célula. Nessa representação das Heurísticas, a primeira letra refere-se à abordagem usada para a escolha da Configuração e a segunda letra refere-se à abordagem usada na escolha da Carga de Trabalho.

Podemos ver, então, que as Heurísticas com abordagem Otimista escolherão Configurações mais próximas a $C_1$ e Cargas de Trabalho mais próximas a $W_m$. Por serem otimistas, essas abordagens consideram que máquinas menores são capazes de executar sob cargas mais severas.

Ainda com base na mesma imagem, vemos que as abordagens conservadoras se concentram nas células intermediárias, conforme a descrição das Heurísticas. A Heurística OC, otimista para Configurações e conservadora para Cargas, se concentra na primeira linha, ou seja, mais próxima de $C_1$, e nas colunas do centro, mais próximas de $W_{n/2}$. Observação similar se faz para a Heurística PO, pessimista para Configurações e otimista para Cargas, concentrando-se nas últimas linhas e últimas colunas, ou seja, Configurações e Cargas maiores.

Voltando a tratar das Estratégias de Avaliação, estas serão responsáveis por efetuar a escolha de Configurações e Cargas de Trabalho implementando a lógica prevista pelas Heurísticas. Enquanto as Heurísticas são lógicas que indicam as proximidades onde deve ser feita a escolha de Configurações e Cargas, a Estratégia implementa de fato um algoritmo que reflita o comportamento esperado pela ideia da Heurística.

A aplicação das Heurísticas de Seleção, através da implementação de Estratégias de Avaliação, está intrinsecamente ligada aos objetivos deste trabalho de estudar os efeitos da inferência de desempenho na eficiência do processo de avaliação de capacidade para aplicações em ambientes de nuvem de infraestrutura. A inteligência das Heurísticas propostas, ou seja, sua capacidade de escolher corretamente as Configurações e Cargas de Trabalho, é determinante para o sucesso do processo e da técnica de inferência. A eficácia e efetividade da aplicação das heurísticas é analisada no capítulo 5.

## 3.3 Funcionamento do Processo

Para efeito de entendimento do funcionamento geral do processo de avaliação de capacidade ora proposto, podemos abstrair temporariamente o comportamento das Heurísticas. Esta é, aliás, outra vantagem da abordagem adotada de delegação de funções específicas a atividades customizadas: além da flexibilidade e adaptabilidade, a abstração dessas operações torna mais fácil o entendimento, a descrição e a implementação concreta do processo.

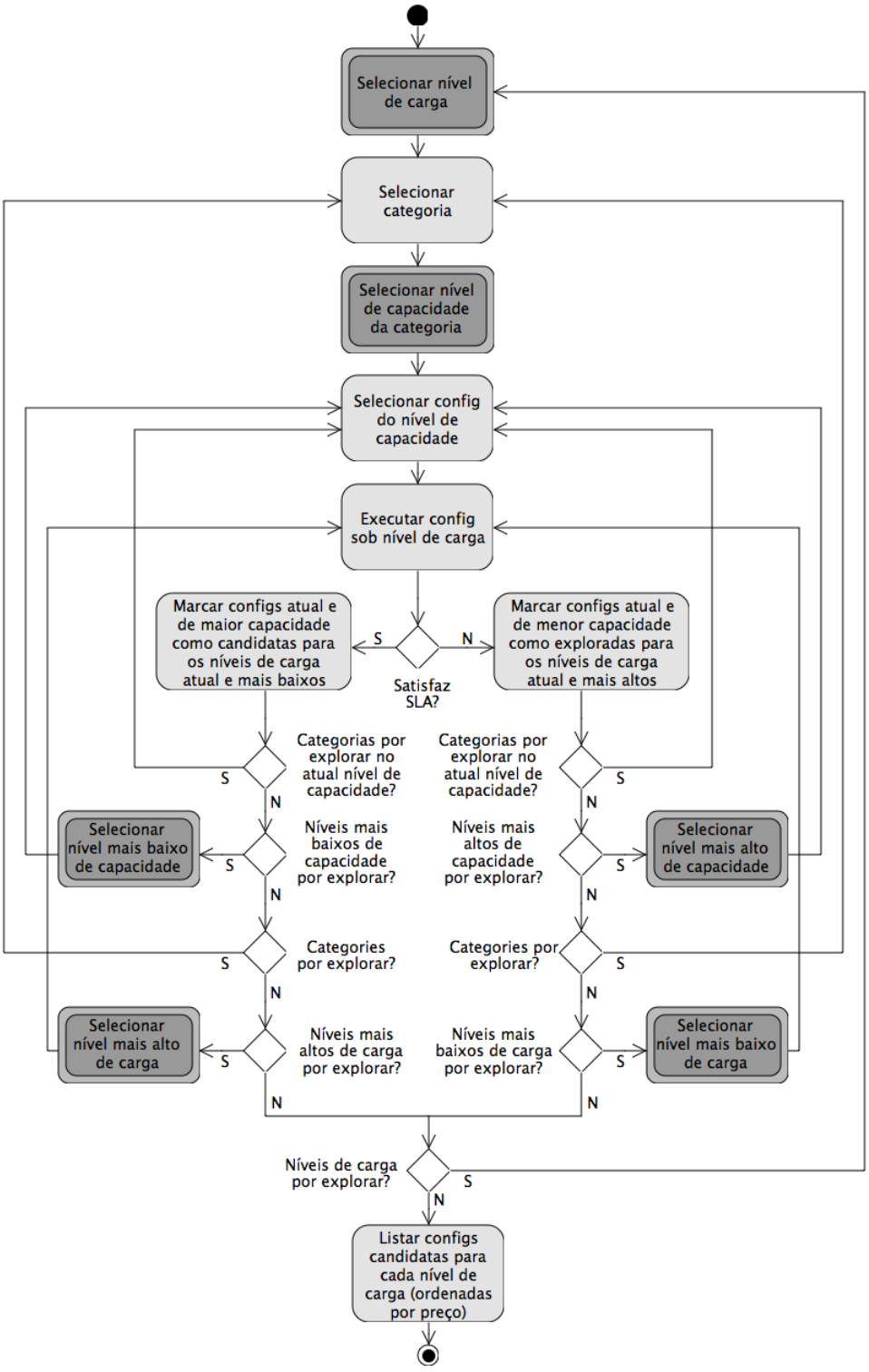



Figura 3 – Diagrama de Funcionamento do Processo de Avaliação de Capacidade

Na Figura 3, os blocos em destaque representam as operações que o processo espera que sejam executadas por uma Estratégia de Avaliação que implemente as Heurísticas propostas na seção anterior. Os outros blocos referem-se a ações comuns do próprio processo, executadas de maneira idêntica independentemente de qual seja a Aplicação sob Teste ou de qual seja a Estratégia de Avaliação usada. A delegação de funções para a atividade de Execução de Testes não está destacada no diagrama e se dá no passo "*Executar config sob nível de carga*", situado aproximadamente no centro da figura.

### 3.3.1 Operações iniciais

Uma vez tendo recebido os dados de entrada, o Processo tem seu início com o momento de escolher por onde começar a execução dos testes.

A primeira atividade desempenhada pelo Processo é a escolha da Carga de trabalho inicial. A Estratégia é solicitada para realizar esta escolha que deve seguir a orientação dada pela lógica da Heurística selecionada para a avaliação. Assim, será escolhido um volume de carga maior se a Heurística for Otimista ou um volume menor se a Heurística para Cargas de Trabalho for Pessimista.

Depois de selecionar o nível de carga, o processo segue para selecionar a Categoria. Conforme as definições apresentadas no Capítulo **??**, os Tipos de Máquinas Virtuais oferecidos pelo Provedor são normalmente agrupados por Categorias, que reunem máquinas de propósito e atributos semelhantes. Dessa forma, o Espaço de Implantação sobre o qual a avaliação de capacidade se dará está dividido em Categorias. O número de Categorias envolvidas na avaliação depende do conjunto de Tipos de Máquinas Virtuais selecionados pelo usuário e passados como parte dos dados de entrada do Processo.

O Processo de Avaliação seleciona a primeira Categoria de máquinas a ser explorada. O processo não especifica a ordem ou método dessa escolha, pois essa ordem não é importante uma vez que todas as Categorias presentes no Espaço de Implantação serão avaliadas. Conforme veremos no Capítulo 4, a implementação de referência do Processo desenvolvida neste trabalho escolhe a primeira Categoria em ordem alfabética pelo nome. Outras implementações do processo podem optar por outros métodos de escolha.

#### 3.3.1.1 Níveis de Capacidade

Dando continuidade à sequência de operações iniciais dentro do Processo de Avaliação de Capacidade, o próximo passo prevê que a Estratégia de Avaliação deve selecionar um Nível de Capacidade inicial.

Níveis de Capacidade são um conceito criado para ajudar a estabelecer uma hierarquia sobre as relações de capacidade de processamento entre as diversas Configurações.

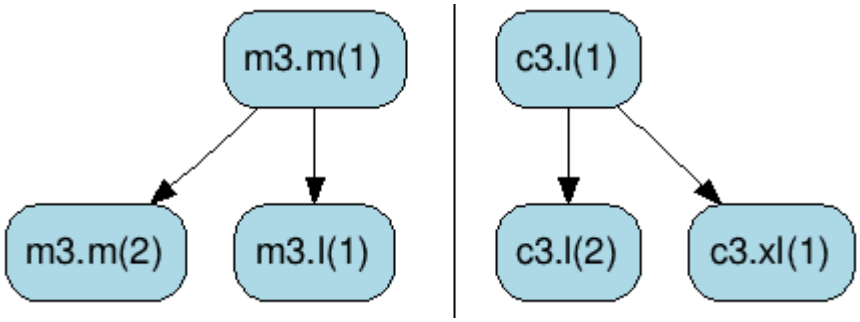



Figura 4 – Agrupamento de Configurações por Níveis de Capacidade

Essa hierarquia de capacidade é válida apenas entre Configurações de uma mesma Categoria de Máquinas Virtuais.

Para cada Categoria, o Nível "um"de Capacidade é composto apenas pela Configuração de menor preço dentro da Categoria. Um novo nível é criado com todas as Configurações para as quais é verdadeira a relação "maior que", conforme a definição descrita no seção **??**. Esse, então, passa a ser o Nível "dois"e a lógica se repete daí em diante, tomando-se, para cada Configuração desse nível, as imediatamente maiores, formando um novo Nível. O procedimento continua até que todas as Configurações estejam devidamente classificadas em Níveis de Capacidade.

A Figura 4 mostra um pequeno exemplo, onde 6 Configurações, pertencentes a duas Categorias distintas, foram classificadas em dois Níveis de Capacidade dentro de cada Categoria. Os retângulos representam as Configurações, com o texto indicando o nome do Tipo de Máquina Virtual utilizado e o número entre parênteses representando a quantidade de instâncias que compõem a Configuração. As setas que ligam as Configurações indicam a existência da relação de capacidade entre elas apontando da menor para a maior. A ausência de seta entre duas Configurações implica a impossibilidade de se afirmar uma relação de capacidade entre elas.

Assim, observando o Espaço de Implantação organizado por Categorias e classificado hierarquicamente, a Estratégia deve selecionar, com base em uma das Heurísticas propostas na seção **??**, um Nível de Capacidade inicial. As Configurações que fazem parte do Nível inicial escolhido são disponibilizadas para que a Avaliação proceda com a execução dos testes da Aplicação.

### 3.3.2 Execução do Teste de Desempenho

Após a escolha da Carga de Trabalho inicial e do primeiro Nível de Capacidade a ser avaliado, uma Configuração deve ser tomada a partir do Nível de Capacidade atual. Essa seleção não segue nenhuma regra específica, uma vez que todas as Configurações do Nível de Capacidade devem ser avaliados, ainda que por meio da técnica de Inferência de

Desempenho, vista adiante.

Executa-se, então, a Aplicação sob Teste, impondo-se a ela a Carga de Trabalho selecionada, e analisa-se o Resultado (ver seção **??**) dessa execução. Nesse ponto o Processo se bifurca, atingindo seu primeiro ponto de decisão.

Esse é o momento em que a técnica de Inferência, conforme propomos neste trabalho, será aplicada. A análise do Resultado obtido, mais especificamente do valor atribuído à Métrica de Desempenho avaliada, comparado ao parâmetro do Valor de Referência, ou SLA, define se a Aplicação é ou não capaz de atender à demanda imposta pela Carga de Trabalho. Vejamos a seguir uma explanação mais detalhada a respeito das inferências que sucedem essa análise.

#### 3.3.2.1 Inferência de Desempenho

O processo de Inferência de Desempenho acontece logo após a análise comparativa do Resultado, conseguinte a uma execução real da Aplicação sob Teste em uma Configuração, que foi tomada a partir de um Nível de Capacidade previamente selecionado. Durante essa execução, foi imposta sobre a Aplicação uma Carga de Trabalho, também previamente selecionada.

Observando o diagrama da Figura 3, podemos ver, bem ao centro, o ponto de decisão que define o sucesso ou o fracasso da Aplicação em atingir o SLA exigido.

Se o desempenho da Aplicação satisfaz o SLA proposto, o Processo considera que a Configuração é capaz de executar sob a Carga de Trabalho imposta e diz que a Configuração atual (sobre a qual a Aplicação acabou de ser executada) deve ser assinalada como uma Configuração Candidata.

Neste ponto, a técnica de Inferência de Desempenho é aplicada e, como vemos no texto do diagrama do Processo, todas as Configurações maiores que a atual também são assinaladas como Candidatas. Ora, se identificamos que uma certa Configuração consegue executar a Aplicação sob uma certa Carga de Trabalho, é intuitivo o pensamento de que qualquer Configuração que possua maior poder computacional também seja capaz de executar a mesma Aplicação sob a mesma Carga de Trabalho.

Assim, usando a representação do Espaço de Implantação conforme descrito na Seção 3.3.1.1, o Processo assinala como candidatas todas as Configurações para as quais, direta ou indiretamente, a Configuração atual aponte uma seta, ou seja, todas as Configurações "maiores que" a Configuração atual. Considera-se que uma Configuração $C_1$ é "maior que" uma outra Configuração $C_2$ se:

**a)** ambas são formadas pelo mesmo Tipo de Máquina Virtual e $C_1$ possui número maior de instâncias do que $C_2$; ou

**b)** ambas são formadas pelo mesmo número de instâncias e o Tipo de Máquina de $C_1$ possui mais CPU e mais memória que o Tipo de Máquina de $C_2$.

Mas a técnica de Inferência ainda vai mais longe. Sabendo que as Configurações que acabaram de ser assinaladas como Candidatas são capazes de executar a Aplicação sob a Carga de Trabalho atual, também é intuitivo concluir que Cargas de Trabalho menores, ou mais brandas, serão naturalmente atendidas.

Então, prosseguindo com a aplicação da Inferência de Desempenho, o Processo marca a Configuração atual e todas as maiores que ela como Candidatas não só para a Carga de Trabalho atual, mas também para todas as Cargas inferiores à atual.

De volta ao ponto de decisão da análise do Resultado em relação ao SLA, caso o desempenho da Aplicação não satisfaça o SLA, o Processo considera que a Configuração não é capaz de executar sob a Carga de Trabalho atual. Assim, essa Configuração é marcada como Rejeitada para tal Carga.

De forma coerente, a lógica de Inferência de Desempenho entende que, se dada Configuração não consegue executar uma Aplicação a contento sob uma certa Carga, intuitivamente, as Configurações menores tampouco conseguirão. Assim, o processo indica a marcação de todas as Configurações "menores que" a atual como Rejeitadas para a Carga de Trabalho atual.

Ainda seguindo a Inferência de Desempenho, quando uma Configuração não consegue atender a demanda de uma Carga de Trabalho, presume-se que também não consiga atender a demandas maiores, ou mais severas. O Processo de Avaliação manda, então, que a Configuração atual e todas as Configurações menores sejam também marcadas como Rejeitadas para todas as Cargas de Trabalho maiores que a Carga atual.

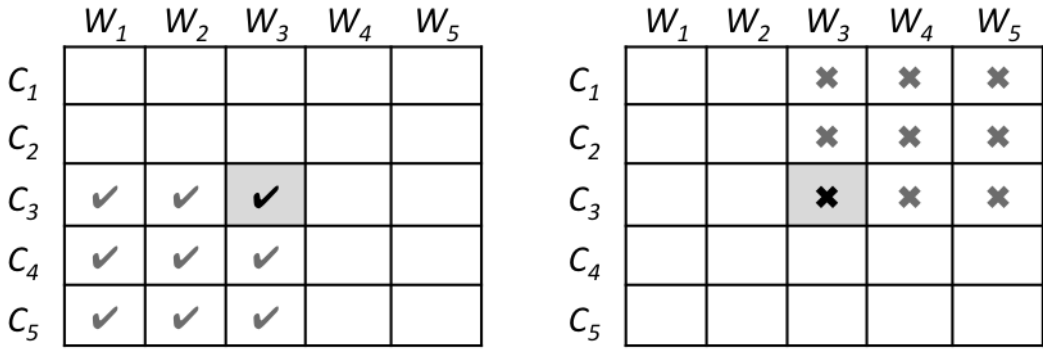



Figura 5 – Inferência de Desempenho: Marcação de Configurações Candidatas e Rejeitadas

O efeito da aplicação da técnica de Inferência de Desempenho pode ser melhor visualizado através da Figura 5. A figura mostra duas situações: a da esquerda para o caso em que o SLA é satisfeito e a da direita caso o SLA não seja satisfeito. Em ambos os casos

vemos uma célula central realçada, indicando que a Aplicação sob Teste foi executada na Configuração $C_3$ sob a Carga de Trabalho $W_3$. As células marcadas com $\surd$ mostram as Configurações marcadas como Candidatas para as Cargas de Trabalho correspondentes. As células marcadas com $\times$ representam as Configurações assinaladas como Rejeitadas.

Essa representação serve para demonstrar a contribuição da aplicação da técnica de Inferência na aplicação de testes de desempenho. Nesse exemplo, foram explorados 9 cenários diferentes, dados pelas combinações de Configurações e Cargas de Trabalho, mas apenas 1 execução real foi conduzida. Isso significa não só que o tempo total gasto na Avaliação de Capacidade é reduzido, mas também o custo financeiro envolvido nessa atividade.

Mais adiante neste capítulo, expandiremos essa representação para mostrar a ação da Inferência de Desempenho após várias iterações do laço compreendido pelo Processo de Avaliação. Por hora, continuemos com a descrição dos passos seguintes.

### 3.3.3 Seleção dos Próximos Cenários

Passada a fase de Inferência de Desempenho, o processo segue seu caminho, tomando as decisões que levarão à seleção dos cenários seguintes a serem avaliados quanto à sua capacidade.

O ponto de decisão que sucede a marcação das Configurações Candidatas ou Rejeitadas checa se existem Configurações pertencentes ao atual Nível de Capacidade cujas execuções ainda não tenham sido avaliadas para a Carga de Trabalho atual. Se existirem, o Processo volta ao passo de seleção de uma Configuração a partir do Nível de Capacidade e uma nova execução é solicitada.

Se não existirem Configurações inexploradas para a Carga de Trabalho atual no Nível de Capacidade corrente, o Processo buscará por um Nível de Capacidade que ainda não tenha sido completamente explorado. Se a Aplicação satisfez o SLA na execução anterior, a Estratégia deverá selecionar um Nível de Capacidade menor. Se o SLA não tiver sido atingido, a Estratégia tentará selecionar um Nível de Capacidade maior. Depois de selecionado o próximo Nível de Capacidade, o laço do Processo retorna ao ponto de seleção da próxima Configuração e outra execução acontece.

Novo ponto de decisão surge quando a Estratégia não encontra um Nível de Capacidade a ser explorado. Nessa situação, o Processo procura uma outra Categoria dentro do Espaço de Implantação que ainda possua pelo menos uma Configuração ainda não explorada para a Carga de Trabalho atual. Se houver, o Processo retorna ao passo de seleção de um Nível de Capacidade dentro da Categoria selecionada e nova execução será realizada.

Caso não haja uma outra Categoria com Configurações não avaliadas, atingimos

então um outro ponto de decisão, onde a Estratégia deve buscar uma Carga de Trabalho que não tenha sido avaliada. Se a execução anterior atingiu o SLA, uma Carga de Trabalho maior será buscada. Caso contrário, a Estratégia tentará uma Carga menor que a atualmente selecionada. Se a Estratégia obtiver sucesso nessa escolha, o Processo dispara uma nova execução da Aplicação com a Configuração corrente e sob a Carga de Trabalho que acabou de ser selecionada.

Porém, se a Estratégia não conseguir fornecer uma Carga de Trabalho segundo as restrições do Processo segundo a comparação do Resultado com o SLA, o Processo voltará ao passo inicial de seleção de Carga de Trabalho, que não tem qualquer restrição quanto a essa seleção. Tentará assim a escolha de uma Carga inexplorada qualquer. Caso não seja encontrada nenhuma Carga de Trabalho inexplorada, significa que não há mais nada a ser testado e o Processo é encerrado.

### 3.3.4 Finalização da Avaliação

Não tendo mais Configurações a serem testadas sob nenhuma Carga de Trabalho, o Processo é dado por concluído e sua finalização se efetiva pela apresentação de uma lista que contém, para cada Carga de Trabalho, as Configurações capazes de executar a Aplicação sob Teste, em ordem crescente de preço.

Por "capazes de executar", entenda-se como as Configurações para as quais o valor obtido para a Métrica de Desempenho durante a execução da Aplicação sob determinada Carga de Trabalho é menor ou igual ao valor definido para o SLA ou Valor de Referência como um dos dados de entrada no início do Processo de Avaliação de Capacidade.

Embora essa seja a resposta final do processo, a contribuição real deste trabalho está na maneira como chegamos a essa resposta, com redução de custo e tempo, e na precisão da resposta, ou seja, o nível de acerto atingido pelo Processo ao apontar as Configurações Candidatas e Rejeitadas. Dados reais que comprovam a eficácia do Processo e da técnica de Inferência de Desempenho propostos são apresentados no Capítulo 5, que trata dos experimentos realizados e analisa criticamente os resultados obtidos.

Para deixar mais clara essa contribuição, é explicado a seguir como o Processo constrói a resposta final, demonstrando o rastreamento dos vários momentos em que a técnica de Inferência de Desempenho é empregada e evidenciando a influência que a utilização de diferentes Heurísticas exerce sobre a eficiência do Processo.

A Figura 6 mostra, nos quadros da parte inferior, o rastro de todas as iterações da execução do Processo de Avaliação com 3 Heurísticas de Seleção distintas: Otimista/Otimista, Conservadora/Conservadora e Pessimista/Pessimista. O quadro da parte superior da imagem mostra o que seria o resultado da execução real da aplicação em todos os cenários analisados, ou seja, com todas as combinações de Configurações e Cargas de

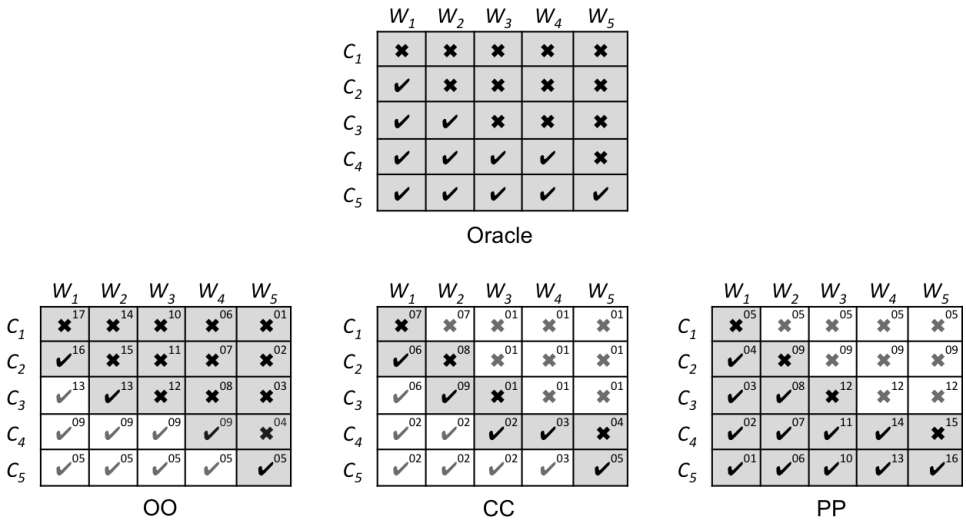



Figura 6 – Rastreamamento da Execução das Heurísticas e Inferência de Desempenho

Trabalho. A essa representação damos o nome de Oráculo, por conter toda a informação real de desempenho da Aplicação sob Teste considerada em cada cenário.

O Oráculo serve-nos de base de referência para a apuração da eficácia do Processo e sua Inferência de Desempenho, ou seja, a precisão de acerto quanto a quais são de fato as Configurações capazes de executar a Aplicação com desempenho que atenda ao SLA. Além disso, com base no Oráculo, podemos também aferir qual a eficiência da Heurística utilizada na Avaliação, medindo a economia em número de execuções reais e, por conseguinte, em tempo e custo financeiro da Avaliação.

As execuções reais são representadas nas figuras pelas células destacadas, mais escuras. As células marcadas com um $\surd$ indicam que a Aplicação atingiu um desempenho que cumpre o SLA exigido para aquela combinação de Configuração e Carga de Trabalho, ou seja, a Configuração é considerada Candidata para aquela Carga. As células marcadas com um $\times$ mostram onde a Aplicação não conseguiu cumprir o SLA, ou seja, as Configurações Rejeitadas.

Dentro de cada célula nos quadros que mostram o comportamento das Heurísticas há um número que indica em qual passo da iteração do processo a Configuração foi marcada como Candidata ou Rejeitada. Alguns números são repetidos em várias células e isso serve para mostrar a Inferência de Desempenho atuando. As células brancas foram marcadas como Candidatas ou Rejeitadas por meio da técnica de Inferência no passo denotado pelo pequeno número no canto da célula. Note-se que para as células destacadas os números não se repetem e demonstram a ordem de escolha das Configurações pela Heurística para execução real da Aplicação.

Os exemplos apresentados na figura 6 mostram uma eficácia de 100%, ou seja, usando a Inferência, o Processo indicou com correção todas as Configurações que, caso

fossem testadas de fato, com uma execução real da Aplicação, teriam um desempenho dentro do SLA esperado. A imagem mostra, ainda, que, para a Aplicação fictícia considerada, a Heurística Conservadora/Conservadora foi a mais eficiente, com apenas 9 execuções reais contra um potencial total de 25 execuções. Supondo que o tempo de execução dos testes de desempenho da Aplicação é o mesmo em cada iteração, isso representa uma economia de 64% no tempo de execução da Avaliação de Capacidade em comparação com a execução de todos os cenários.

Outra contribuição trazida pelo emprego das Heurísticas de Seleção de Configurações é a automação da decisão de por onde começar a execução dos testes. Quando se faz necessária a análise do comportamento de uma aplicação qualquer em um determinado ambiente, com dezenas ou até centenas de variações tanto do ambiente como dos volumes de demanda, a escolha de uma combinação inicial dessas variáveis é um problema de difícil solução.

O emprego das Heurísticas de Seleção ajudam não só na escolha da Configuração e da Carga de Trabalho iniciais, mas também de qual a próxima combinação a ser testada e sempre visando ao menor custo de execução dessa avaliação.

O Capítulo 5 irá mostrar os resultados obtidos com o Processo de Avaliação proposto e descrito neste trabalho ao avaliar uma Aplicação real em um ambiente também real de nuvem de infraestrutura. Serão apresentados resultados de eficiência e eficácia do emprego da técnica de Inferência de Desempenho e das Heurísticas de Seleção, comparando-as sob diversos SLAs.

## 3.4 Resumo

Este capítulo apresentou uma proposta de processo de avaliação de capacidade cuja ideia chave é a inferência de capacidade de um recurso a partir dos dados reais de desempenho obtidos por um outro recurso semelhante, mas de capacidade presumidamente menor ou maior.

Essa proposta de processo se apoia na hipótese de que é possível estabelecer uma relação de capacidade entre os recursos disponibilizados por um provedor de nuvem de infraestrutura. Com base nessa hipótese, definiu-se uma sequência de passos que visam a identificar quais recursos são capazes de executar uma determinada aplicação sob determinados volumes de carga de trabalho com o menor número possível de execuções reais da aplicação. Como resultado, o processo busca a redução de custo e tempo normalmente envolvidos na atividade de avaliação de capacidade dos recursos disponíveis na nuvem.

O Capítulo 4 a seguir apresentará uma visão mais concreta, com a descrição do arcabouço de software desenvolvido neste trabalho como implementação de referência para

o processo proposto. Além disso, será utilizada uma aplicação web para ilustrar o uso do processo de avaliação de capacidade baseado em inferência de desempenho.

# 4 Implementação do Processo

Este capítulo apresenta a implementação do Processo de Avaliação de Capacidade descrito no capítulo anterior, bem como da técnica de Inferência de Desempenho e das Heurísticas de Seleção que dão suporte a esse Processo.

A implementação está estruturada na forma de uma biblioteca extensível e de um sistema computacional que demonstra seu funcionamento.Foi dado o nome de CloudCapacitor à biblioteca, implementada como uma *gem* (i.e., um pacote) da linguagem Ruby (MATSUMOTO, 2014) e desenvolvido o sistema computacional Capacitor Web para ser uma interface visual para a utilização do CloudCapacitor, usando o *framework* Ruby on Rails (HANSON, 2014).

A seguir são descritos os detalhes da implementação de cada um destes módulos e como ambos se relacionam para oferecer ao usuário a experiência da avaliação de capacidade de baixo custo e alta precisão prevista pelo Processo proposto, com uma interface amigável e de fácil utilização.

## 4.1 CloudCapacitor

CloudCapacitor é uma biblioteca para criação de sistemas de avaliação de capacidade em ambientes de nuvem de infraestrutura como serviço. É a implementação completa da especificação do Processo de Avaliação de Capacidade do Capítulo 3, permitindo que sejam customizadas as atividades definidas pelo Processo como pontos de extensão, como as Estratégias de Avaliação e o disparo e controle da execução da Aplicação sob Teste.

A apresentação do CloudCapacitor inicia-se pelas classes que compõem a biblioteca e suas responsabilidades. Em seguida, será mostrado como CloudCapacitor auxilia desenvolvedores de software na criação de sistemas de avaliação de capacidade, mostrando o fluxo de utilização da biblioteca através de sua interface de programação. Depois, serão explicados alguns detalhes de implementação da biblioteca, como a solução para representação do Espaço de Implantação e seu papel na execução do Processo de Avaliação de Capacidade. Por fim, serão apresentados os pontos de extensão da biblioteca, notadamente como implementar um Executor, a classe responsável pelo controle de execução da Aplicação sob Teste, e como sobrescrever a Estratégia de Avaliação fornecida pela biblioteca a fim de alterar seu comportamento padrão.

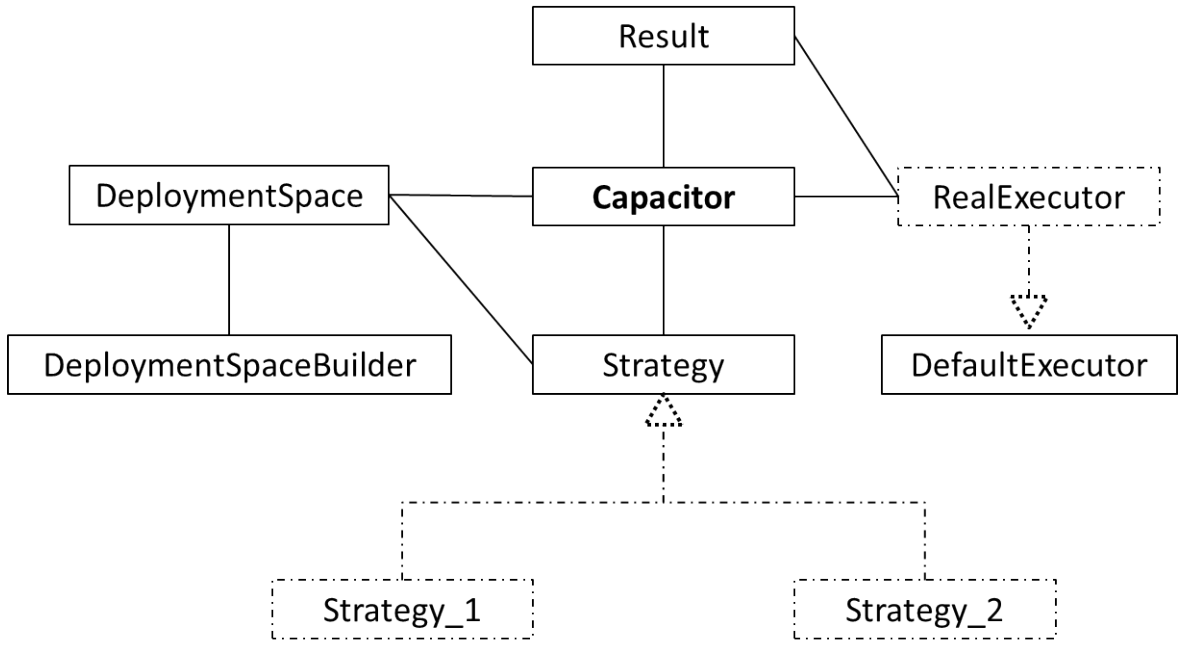
Para concluir a apresentação do CloudCapacitor, será mostrada a saída de dados fornecida pela biblioteca com as Configurações capazes de executar a Aplicação sob Teste em cada uma das Cargas de Trabalho respeitando o SLA definido.

### 4.1.1 Classes e Responsabilidades

CloudCapacitor é formado por um conjunto de classes que representam os componentes envolvidos na avaliação de capacidade de aplicações em ambientes de nuvem de acordo com os conceitos e o Processo definidos anteriormente, nos Capítulos **??** e 3. A Figura 7 mostra as principais classes cujas responsabilidades e cooperação levam ao resultado final de uma Avaliação.

Ao utilizar a biblioteca CloudCapacitor na construção de um software para avaliação de capacidade, o desenvolvedor tem à sua disposição uma classe principal, chamada *Capacitor*. Essa classe fornece o fluxo principal do Processo de Avaliação, com todos os seus pontos de decisão e de extensão.

Figura 7 – Principais classes que compõem o CloudCapacitor e suas relações




Entre os pontos de extensão, destaca-se o uso da classe *Strategy*. Como pode ser visto na Figura 7, essa classe é fornecida pela biblioteca e a notação de linhas pontilhadas denota a sua possibilidade de especialização, onde, sobrescrevendo alguns de seus métodos, é possível criar Estratégias com comportamento diversificado.

A classe *DefaultExecutor* é o outro ponto de extensão oferecido pela biblioteca. Porém, neste caso, o desenvolvedor deve mandatoriamente implementar uma subclasse que forneça a lógica necessária ao controle da execução do teste de desempenho. Na figura, a classe a ser implementada pelo desenvolvedor é representada com o nome de *RealExecutor*.

Para que a Avaliação de Capacidade possa ser efetuada, a classe *Capacitor* deve conhecer o resultado de cada execução da Aplicação sob Teste, de modo que possa tomar as

decisões corretas na indicação das Configurações Candidatas e Rejeitadas. Esses resultados são encapsulados na classe *Result*, cujos objetos são fornecidos pela subclasse responsável pela execução dos testes de desempenho.

E, finalmente, durante a execução da Avaliação de Capacidade, a classe *Capacitor* precisa ter conhecimento do Espaço de Implantação disponibilizado. Essa é a responsabilidade da classe *DeploymentSpace*, que implementa uma estrutura de dados em memória para representar os diversos Níveis de Capacidade formados entre as Configurações. A construção dessa estrutura é responsabilidade da classe *DeploymentSpaceBuilder*, que contém os algoritmos necessários à preparação do grafo usado para navegação pelos Níveis de Capacidade.

### 4.1.2 Fluxo de Utilização da Biblioteca

CloudCapacitor foi criada para ser uma ferramenta de auxílio na construção de aplicações destinadas à condução de testes para avaliação de capacidade, visando basicamente a reusabilidade e a simplicidade na composição dessas aplicações.

Embora tenha sido desenvolvida objetivando sua utilização na construção de aplicações web baseadas no framework Ruby on Rails, CloudCapacitor pode facilmente ser usada em todo tipo de aplicação escrita em Ruby, desde scripts puros até aplicações baseadas em outros frameworks.

Em suma, o fluxo de utilização da biblioteca é bastante simples e pode ser descrito por:

1. Configurar parâmetros de criação do Espaço de Implantação
2. Identificar Tipos de Máquinas Virtuais para o Espaço de Implantação
3. Instanciar um objeto *Capacitor*
4. Atribuir ao *Capacitor* um objeto *DefaultExecutor*
5. Atribuir ao *Capacitor* um objeto *Strategy*
6. Executar o método *run_for* do *Capacitor*

Antes que a aplicação que está sendo desenvolvida possa de fato dar início ao uso da biblioteca, é preciso que o desenvolvedor a configure previamente.

O primeiro passo é configurar os limites de tamanho do Espaço de Implantação e isso deve ser feito através de um arquivo em formato YAML (YAML, 2009). A localização e o nome do arquivo de parâmetros para a criação do Espaço de Implantação dependem de como a aplicação está sendo implementada: se for uma aplicação Ruby on Rails, deverá

haver um arquivo chamado *capacitor.yml* na pasta *config*, que fica dentro da pasta raiz da aplicação; caso contrário, deverá haver um arquivo chamado *capacitor_settings.yml* já na pasta raiz da aplicação ou na mesma pasta em que está o script que invoca a biblioteca.

A Figura 8 mostra um exemplo de conteúdo do arquivo que configura os limites do Espaço de Implantação. São apenas dois parâmetros:

**max_price**
O custo máximo que uma Configuração pode atingir

**max_num_instances**
Número máximo de instâncias usadas em uma Configuração

Figura 8 – Parâmetros de Configuração para o CloudCapacitor

```
1 deployment_space:
2   #The maximum price for a whole Configuration.
3   #This refers to the individual VM_Type price multiplied
4   #by the number of instances that make up the Configuration
5   max_price: 7.0
6
7   #The maximum number of instances in a Configuration.
8   #This is for horizontal scaling.
9   max_num_instances: 4
```

No exemplo acima, serão criadas Configurações com 1, 2, 3 e 4 instâncias para cada Tipo de Máquina Virtual especificada, desde que o custo total da Configuração não ultrapasse o valor de 7,00 unidades monetárias.

O próximo passo na utilização do CloudCapacitor é especificar quais serão os Tipos de Máquinas Virtuais utilizados na geração do Espaço de Implantação. Essa especificação é feita em outro arquivo em formato YAML, discriminando as características de CPU, Memória e custo de cada Tipo de Máquina para que o CloudCapacitor possa gerar os Níveis de Capacidade usados para inferência de desempenho. O caminho onde esse arquivo deve ser entrado precisa ser passado como parâmetro na inicialização do objeto Capacitor. Caso não seja informado, o CloudCapacitor procurará um arquivo pelo nome *deployment_space.yml*, na pasta *config*, se for uma aplicação Ruby on Rails, ou na pasta raiz do script para outro tipo de aplicação.

A Figura 9 mostra a especificação de um conjunto de Tipos de Máquinas Virtuais oferecidos pelo serviço EC2 do Provedor Amazon Web Services (EC2, 2012). São disponibilizadas para o CloudCapacitor máquinas *m2.xlarge*, *c1.xlarge*, *m1.xlarge* e *m2.4xlarge* e, para cada uma, podem ser vistas as características de CPU, memória RAM e preço, conforme informados pelo Provedor. Esses são os dados que serão usados na criação do

Figura 9 – Especificação de Tipos de Máquinas para o Espaço de Implantação

```
1  ---
2  - !ruby/object:CloudCapacitor::VMType
3    name: m2.xlarge
4    cpu: 6.5
5    mem: 17.1
6    price: 0.41
7    category: m2
8  - !ruby/object:CloudCapacitor::VMType
9    name: c1.xlarge
10   cpu: 20
11   mem: 7.0
12   price: 0.58
13   category: c1
14 - !ruby/object:CloudCapacitor::VMType
15   name: m1.xlarge
16   cpu: 8
17   mem: 15.0
18   price: 0.48
19   category: m1
20 - !ruby/object:CloudCapacitor::VMType
21   name: m2.4xlarge
22   cpu: 26
23   mem: 68.4
24   price: 1.64
25   category: m2
```

Espaço de Implantação, atendendo às restrições de tamanho e custo impostas no passo anterior, de forma que os testes sejam executados conforme o proposto no Processo de Avaliação de Capacidade.

Feitas as configurações necessárias, o desenvolvedor pode, assim, criar um objeto a partir da classe *Capacitor* e, então atribuir a ele um objeto instanciado a partir de uma subclasse de *DefaultExecutor*, subclasse esta de sua própria implementação e que forneça os meios necessários para administração da execução dos testes de desempenho da Aplicação sob Teste.

Na Figura 10 vê-se a instanciação do Capacitor, na linha 1. Na linha seguinte, um objeto da classe *DummyExecutor*, que é apenas didática, é atribuído ao *Capacitor*.

A partir da linha 3, vêem-se os passos seguintes da utilização do CloudCapacitor, com a definição de uma Estratégia de Avaliação, configurada com uma Heurística de Seleção do tipo OC (v. Seção **??**). A execução da Avaliação de Capacidade acontece de fato na chamada que ocorre na linha 8, onde são passados valores que representam uma lista de grandezas de Cargas de Trabalho que serão impostas à Aplicação sob Teste rodando

Figura 10 – Código Ruby para execução do CloudCapacitor

```
1   capacitor = Capacitor.new
2   capacitor.executor = Executors::DummyExecutor.new
3   capacitor.strategy = Strategies::Strategy.new
4
5   capacitor.strategy.approach workload: :optimistic,
6                               config:   :conservative
7
8   candidates = capacitor.run_for [100,200,300,400,500]
9
10  total_cost = capacitor.run_cost
11  total_executions = capacitor.executions
```

sobre as Configurações geradas a partir do Espaço de Implantação.

Ao final da Avaliação, o Capacitor retorna a lista de Configurações Candidatas para cada valor de Carga de Trabalho passado como parâmetro. Adicionalmente, o desenvolvedor tem à sua disposição alguns dados a respeito da própria Avaliação, como o número total de execuções da Aplicação sob Teste no ambiente de nuvem e o custo total acarretado por essas execuções.

Agora, já com uma visão geral dos componente e da utilização do CloudCapacitor, segue-se uma descrição um pouco mais detalhada do que acontece durante a atividade de Avaliação de Capacidade.

### 4.1.3 Funcionamento Interno

Uma vez demonstrado o modelo de utilização da biblioteca e explicadas as relações entre as diversas classes que a compõem, é possível detalhar alguns aspectos da implementação que ajudarão a esclarecer melhor como todas essas partes agem em conjunto para executar, da forma como proposto no Capítulo 3, o Processo de Avaliação de Capacidade.

Nesta seção são mostradas inicialmente as linhas gerais da implementação da classe responsável pelo controle da execução dos testes de desempenho. Mostra-se em seguida como a biblioteca representa o Espaço de Implantação de forma que os Níveis de Capacidade sejam usados para navegar entre Configurações e como a Inferência de Desempenho atua sobre esses dados. Explica-se ainda como pode ser feita a customização de uma Estratégia de Avaliação a fim de alterar o comportamento de uma Heurística. Será também apresentado um exemplo de saída do resultado retornado pela classe Capacitor ao final da execução de uma Avaliação.

#### 4.1.3.1 Controle da Execução

O controle da execução da Aplicação sob Teste no ambiente de nuvem é feito por meio de um objeto instanciado de uma classe que deve ser criada pelo desenvolvedor que está utilizando o CloudCapacitor na implementação de sua ferramenta de Avaliação de Capacidade.

Para que tudo funcione a contento, a classe criada deve ser extendida a partir da classe *DefaultExecutor* fornecida pelo CloudCapacitor. A única exigência feita para essa subclasse é que ela implemente um método:

- Com a assinatura *run(configuration, workload)*; e
- Que retorne um objeto da classe *Result*

Ou seja, a classe criada pelo desenvolvedor deve pelo menos apresentar um método *run* que receba como parâmetros uma Configuração e um valor para a Carga de Trabalho que será imposta sobre a Aplicação. O retorno desse método deve ser um objeto da classe *Result*, que deve ser inicializado com apenas 3 atributos:

***value***
O valor obtido pela Aplicação para a Métrica de Desempenho estudada

***cpu***
O percentual de CPU consumido pela Aplicação durante o teste

***mem***
O percentual de Memória RAM utilizada pela Aplicação durante o teste

Essas informações devem ser obtidas pelo desenvolvedor através de chamadas a uma ferramenta de execução de testes, como (CUNHA, 2012), (JAYASINGHE et al., 2012) ou (SILVA et al., 2013). A implementação dessa comunicação com a ferramenta de automação de testes escolhida está fora do escopo deste trabalho e deve ficar a cargo do desenvolvedor prover o código que controle a execução dos testes e obtenha os valores de desempenho e consumo de CPU e memória.

O método *run*, implementado na subclasse criada a partir de *DefaultExecutor*, é chamado pelo objeto *Capacitor* nos momentos em que novas execuções são necessárias, conforme descrito na Seção 3.3.2. O objeto *Result* retornado é usado na comparação com o SLA especificado pelo usuário do sistema como Valor de Referência para indicar se a Aplicação atendeu ou não ao requisito de desempenho.

De posse dessa resposta, o *Capacitor* tem condições de prosseguir no seu processamento, marcando as Configurações como Candidatas ou Rejeitadas e escolhendo as

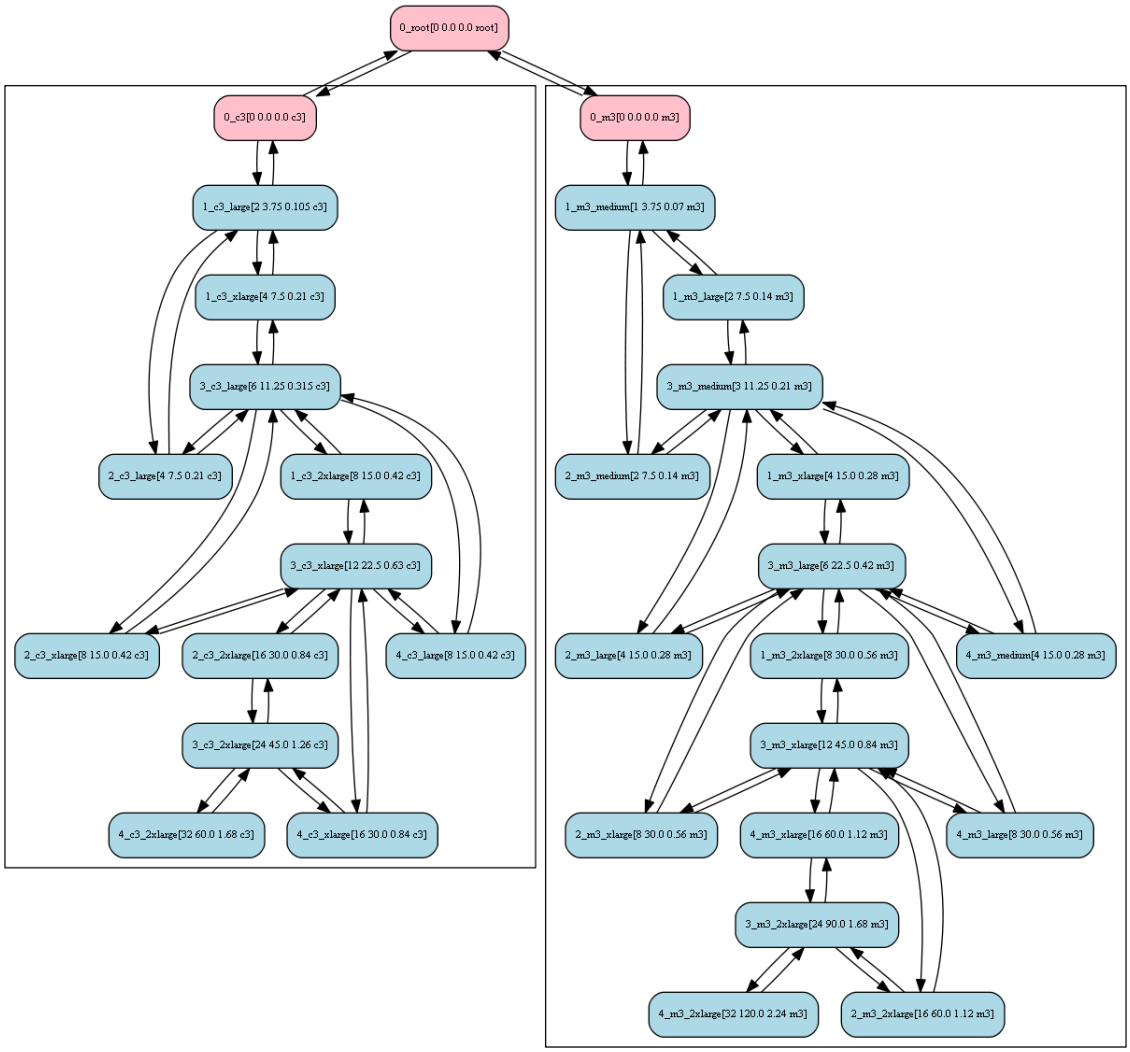
próximas Configurações a serem executadas. Essas ações se baseiam no caminhamento sobre o Espaço de Implantação, cuja implementação está descrita a seguir.

#### 4.1.3.2 Espaço de Implantação

O uso do CloudCapacitor começa com as configurações de como a biblioteca vai montar o Espaço de Implantação, apontando-se quais as restrições e quais os Tipos de Máquinas disponíveis para realização dos testes.

No momento em que o desenvolvedor instancia um objeto da classe *Capacitor*, uma das primeiras tarefas da biblioteca é exatamente proceder à montagem de uma estrutura que represente o Espaço de Implantação de forma que as Heurísticas possam navegar entre as Configurações e que a rotina que implementa o Processo de Avaliação possa identificar, por meio da técnica de Inferência de Desempenho, quais são as Configurações Candidatas e Rejeitadas.

Figura 11 – Representação interna de um Espaço de Implantação no CloudCapacitor



Como é possível ver na Figura 11, o Espaço de Implantação é representado internamente por um grafo direcionado onde arestas ligam duas Configurações para identificar as relações imediatas "maior que" e "menor que", conforme as definições da Seção **??**.

CloudCapacitor separa o Espaço de Implantação em subgrafos, agrupando as Configurações por Categoria de Máquinas Virtuais. Cada subgrafo é destinado às Configurações cujo Tipo de Máquina pertença a uma determinada Categoria. O primeiro nó, observado no topo do grafo, é o nó raiz e serve apenas para estabelecer a separação e permitir o trânsito entre as Categorias.

As classes *DeploymentSpace* e *DeploymentSpaceBuilder* implementam a geração do Espaço de Implantação com base na relação de capacidade ou na relação de preço entre as Configurações. O desenvolvedor pode optar pelo grafo por capacidade ou por preço no momento da instanciação do objeto *Capacitor*. Embora esse não seja o foco desta pesquisa, isso foi feito para que pudesse ser estudada a influência da formação do Espaço de Implantação na precisão e eficiência de custo das Heurísticas de Seleção. Esses resultados serão discutidos no Capítulo 5.

O Espaço de Implantação é a estrutura que dá suporte à navegação das Estratégias sobre os diversos Níveis de Capacidade no momento em que devem ser escolhidas as próximas Configurações a serem testadas. Suporta também a técnica de Inferência de Desempenho na identificação das relações de tamanho e/ou preço entre as Configurações nos pontos onde o Processo marca as Candidatas ou Rejeitadas, mostrando-se como um ponto chave para o sucesso do Processo de Avaliação.

A próxima seção descreve a implementação e o funcionamento das Estratégias de Avaliação e como o desenvolvedor pode criar sua própria Estratégia que melhor supra suas necessidades de seleção de Configurações.

#### 4.1.3.3 Estratégias de Avaliação

A Estratégia de Avaliação é responsável por aplicar uma Heurística de Seleção de Configurações nos momentos em que o Processo precisa navegar pelo Espaço de Implantação. Assim, a cada momento em que um novo Nível de Capacidade deve ser escolhido, segundo o fluxo de ações previsto no Processo de Avaliação proposto, a Estratégia é invocada e age conforme a implementação da Heurística selecionada para a Avaliação (v. Seção 3.3.3).

CloudCapacitor fornece uma classe chamada *Strategy* como uma implementação padrão para as 9 Heurísticas descritas no Capítulo 3. As Figuras 12 e 13 mostram o código fonte dos métodos de seleção de Carga de Trabalho e Nível de Capacidade, respectivamente.

Ambas as implementações são bastante simples. Para a seleção da Carga de Trabalho, o método recebe como parâmetro a lista ordenada de Cargas informada à classe

Figura 12 – Seleção de Carga de Trabalho na classe *Strategy*

```
1   def select_workload( workload_list )
2     case @wkl_approach
3       when :pessimistic
4         workload_list.first
5       when :optimistic
6         workload_list.last
7       when :conservative
8         workload_list[ workload_list.size / 2 ]
9     end
10  end
```

Figura 13 – Seleção de Nível de Capacidade na classe *Strategy*

```
1   def take_a_capacity_level_from( unexplored_levels )
2     levels = unexplored_levels.keys
3     return [] if levels.empty?
4     case @cfg_approach
5       when :pessimistic
6         unexplored_levels.assoc( levels[-1] )
7       when :optimistic
8         unexplored_levels.assoc( levels[0] )
9       when :conservative
10        unexplored_levels.assoc( levels[levels.size / 2] )
11    end
12  end
```

*Capacitor*, como mostrado na linha 8 da Figura 10. O método então considera a abordagem especificada para a seleção de Cargas de Trabalho (Otimista, Conservadora ou Pessimista). Se a abordagem especificada é a Pessimista, o método retorna a primeira Carga da lista, que é a menor. Se a abordagem for a Otimista, retorna a última Carga, que é a maior. Caso a abordagem seja a Conservadora, será retornada uma Carga do meio da lista.

A implementação da seleção do próximo Nível de Capacidade é bem similar. O método recebe uma lista de Níveis de Capacidade onde cada item é uma lista das Configurações que pertencem ao Nível. O método então analisa a abordagem especificada para a seleção de Configurações e, se a abordagem for a Pessimista, retorna a lista de Configurações do maior e último Nível de Capacidade. Se a abordagem for a Otimista, retorna as Configurações do primeiro Nivel, ou seja, o menor deles. Caso a abordagem seja Conservadora, as Configurações retornadas serão as que compõem o Nível de Capacidade do meio da lista de Níveis.

A combinação das abordagens de seleção para Cargas de Trabalho e Níveis de Capacidade nos métodos descritos corresponde às 9 Heurísticas apresentadas na Seção **??**.

Os experimentos realizados neste trabalho e apresentados no Capítulo 5 foram todos executados com a implementação mostrada acima que, como será analisado, demonstra grande eficiência na predição de desempenho para a Aplicação sob Teste avaliada.

Porém, caso se faça necessário atender especificidades da Aplicação sob Teste, é possível substituir essa implementação padrão trazida pela classe *Strategy* por uma mais adequada aos objetivos do usuário. O relacionamento entre a classe *Capacitor* e a Estratégia de Avaliação se dá conforme o Padrão de Projeto (*Design Pattern*) *Strategy* (GAMMA et al., 1994), o que justifica a escolha do nome da classe e do conceito criado para a implementação das Heurísticas.

Assim, para que uma nova lógica de seleção de Cargas de Trabalho seja fornecida ao CloudCapacitor, basta que o desenvolvedor implemente uma nova classe que extenda *Strategy* e sobrescreva o método *select_workload*. Analogamente, para uma nova lógica de seleção de Níveis de Capacidade, a nova classe deve sobrescrever o método *take_a_capacity_level_from*.

Essa flexibilidade oferecida pelo CloudCapacitor permite que seu uso seja customizado para uma ampla gama de Aplicações. Aliada essa característica à implementação de uma subclasse de *DefaultExecutor*, customizando assim o controle da execução da Aplicação, CloudCapacitor se torna adaptável também ao ambiente alvo dos testes, permitindo testes dos mais diversos.

Dando continuidade à descrição da implementação concreta do Processo de Avaliação, a seção seguinte apresenta o retorno dos resultados obtidos pelo CloudCapacitor após a execução da Avaliação.

Figura 14 – Representação do retorno do resultado da execução do CloudCapacitor

```
1  - 100
2      m3.large
3      c3.xlarge
4      m3.2xlarge
5  - 200
6      c3.xlarge
7      m3.2xlarge
8  - 300
9      c3.xlarge
10     m3.2xlarge
11 - 400
12     m3.2xlarge
```

#### 4.1.3.4 Resultado da Avaliação

Durante toda a execução da Avaliação, CloudCapacitor segue colecionando as Configurações marcadas como Candidatas, segundo a técnica de Inferência de Desempenho, para cada uma das Cargas de Trabalho especificadas como parâmetro de entrada.

Ao final da execução, CloudCapacitor dispõe de uma lista de Cargas de Trabalho associadas cada qual a uma lista de Configurações Candidatas. Essa estrutura de lista de listas é devolvida como retorno do método *run_for* da classe *Capacitor* e uma representação conceitual de exemplo pode ser vista na Figura 14.

Esses dados podem então ser usados como base para escolha, por parte do usuário, da Configuração mais barata que atende às diversas demandas esperadas para a sua aplicação implantada na nuvem.

Conclui-se assim a apresentação da implementação da biblioteca CloudCapacitor. A seção a seguir mostra a implementação de um sistema de avaliação de capacidade que faz uso do CloudCapacitor e que foi usado neste trabalho para realizar os experimentos e analisar os seus resultados.

## 4.2 Capacitor Web

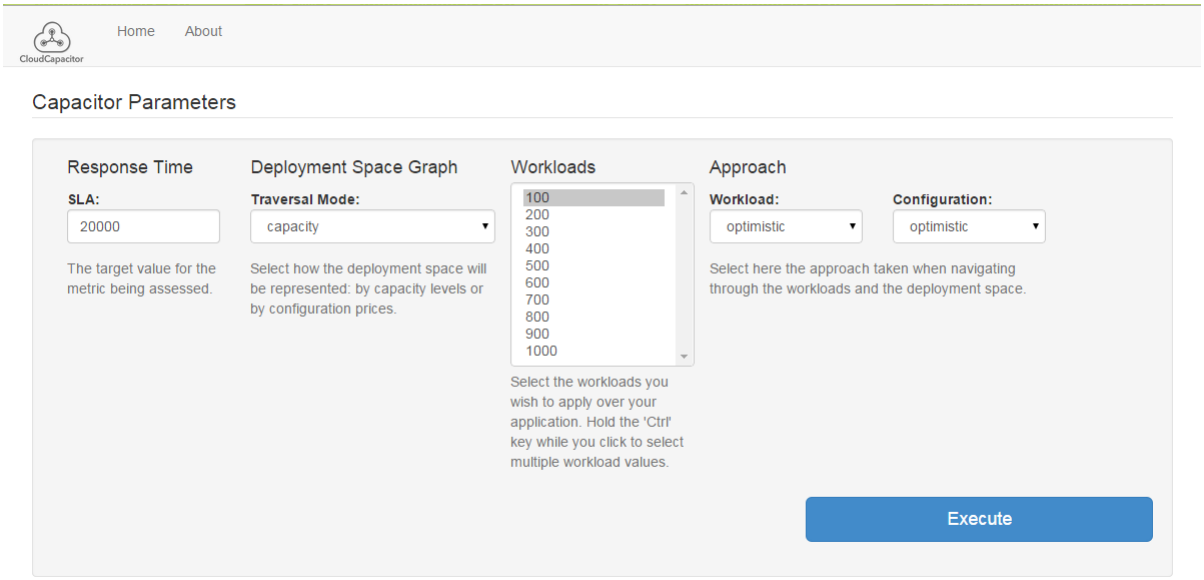
A fim de validar a implementação concreta do Processo através da biblioteca CloudCapacitor, foi construída uma aplicação web que faz uso das funcionalidades oferecidas para a realização de testes de Avaliação de Capacidade em ambientes de nuvem. A aplicação recebeu o nome de Capacitor Web e foi utilizada também para a execução dos experimentos que fazem parte deste trabalho.

A Figura 15 mostra a tela inicial do sistema, onde o usuário tem a oportunidade de parametrizar a execução da Avaliação. O primeiro campo permite a especificação do SLA que deve ser respeitado pela Aplicação para que uma execução seja considerada bem sucedida. Em seguida apresenta-se o campo para seleção da representação do Espaço de Implantação, com opções de representação por capacidade ou por preço.

Mais ao centro da tela está uma lista de valores para que o usuário selecione quais serão repassados ao CloudCapacitor como a lista de Cargas de Trabalho a serem impostas sobre a Aplicação sob Teste. O usuário pode selecionar valores em qualquer ordem e mesmo não adjacentes na lista.

Os últimos dois campos apresentam as opções de abordagens a serem usadas na seleção de Cargas de Trabalho e Níveis de Capacidade. A combinação dessas duas abordagens forma a Heurística de Seleção que a Estratégia de Avaliação usará nos momentos apropriados. As opções aqui são "*Optimistic*", "*Conservative*" e "*Pessimistic*" para ambos os campos. Depois de selecionados os parâmetros, o usuário ordena a execução ao pressionar

Figura 15 – Página inicial do Capacitor Web




o botão presente na parte inferior da tela.

Uma vez concluída a execução da Avaliação, o sistema apresenta ao usuário a tela cuja imagem está na Figura 16. Essa tela se divide em duas grandes áreas, uma superior, fixa, e uma inferior, que, por sua vez, se subdivide em três partes, separadas conforme as abas que dividem a parte superior da inferior.

A parte superior mostra os dados passados como parâmetros pelo usuário na tela inicial, para facilitar a análise dos resultados e sua comparação futura. Assim, são exibidos o valor do SLA, as abordagens para Cargas de Trabalho e Níveis de Capacidade, formando a Heurística de Seleção utilizada, e o atributo usado para geração do grafo que representa o Espaço de Implantação, isto é, capacidade ou preço. Abaixo, são exibidos o número de execuções reais realizadas no ambiente de nuvem alvo da Avaliação e o custo total por hora dessas execuções, com base no preço definido pelo Provedor para as Configurações efetivamente utilizadas no teste.

As abas da parte inferior ativam a visualização das informações complementares resultantes da execução. A primeira, com título “Results”, exibe a saída retornada pela chamada ao método *run_for* da classe *Capacitor* da biblioteca, conforme descrito na seção anterior. Essa saída é formada por um conjunto de valores de Cargas de Trabalho e, abaixo de cada um, uma lista com os nomes das Configurações consideradas como Candidatas para executar com sucesso a Aplicação sob aquela Carga.

A segunda aba da parte inferior da tela de resultados apresenta o título “Exec Trace” e ativa a exibição dos dados de rastreamento da execução. Essa visualização permite que o usuário analise a sequência de execuções reais da Aplicação sob Teste, apontando

Figura 16 – Página de Resultados do Capacitor Web

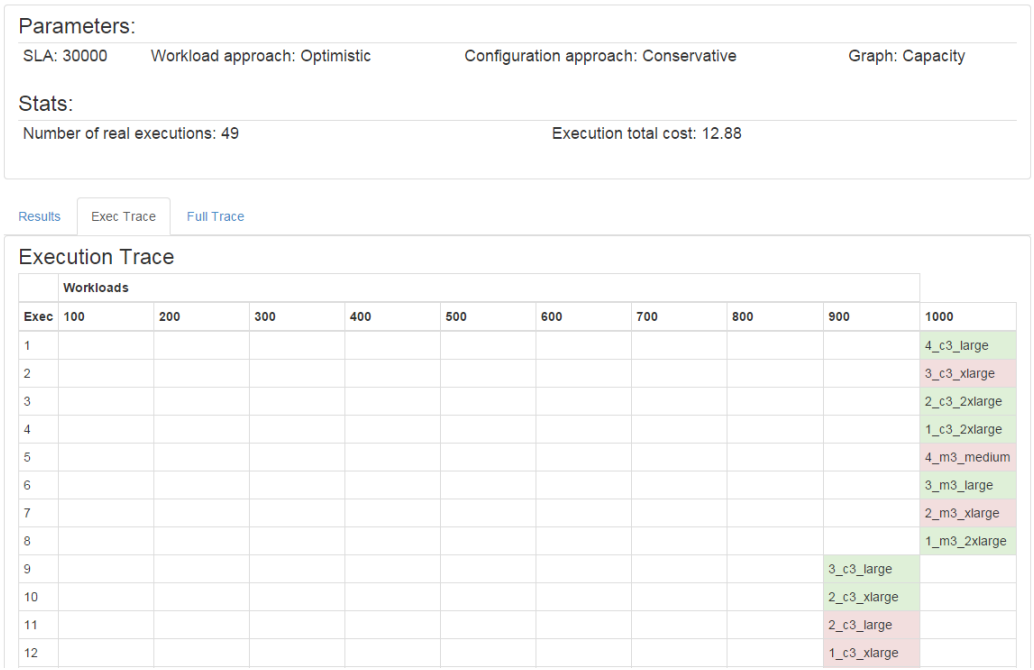

em que momento cada Carga de Trabalho foi aplicada e qual Configuração foi usada para aferir o desempenho da Aplicação. A Figura 17 mostra uma imagem dessa tela.

Os dados são exibidos em uma tabela cujas linhas apresentam a sequencia das execuções, identificadas pelo número presente na primeira coluna. Cada coluna da tabela representa uma Carga de Trabalho e as células, que são a intercessão entre uma sequencia de execução e uma Carga, trazem um texto com o nome da Configuração utilizada naquela execução daquela Carga de Trabalho. As células que aparecem na cor esverdeada indicam as execuções onde o SLA foi satisfeito. De modo contrário, as células na cor avermelhada indicam as execuções onde a Aplicação não conseguiu desempenho suficiente para satisfazer o SLA.

Por fim, a terceira aba, com o título “Full Trace”, traz também uma tabela, esta com as colunas representando as Cargas de Trabalho e as linhas representando as Configurações, como pode ser visto na Figura 18. As células representam as execuções de cada Configuração em cada Carga de Trabalho e sua cor indica o sucesso ou fracasso em atender ao SLA.

Cada célula dessa tabela traz um número que identifica a execução real que levou ao resultado representado pela cor da célula. As células dessa tabela podem aparecer pintadas em quatro cores diferentes:

Figura 17 – Rastro de Execução do Capacitor Web



Parameters:

SLA: 30000 Workload approach: Optimistic Configuration approach: Conservative Graph: Capacity

Stats:

Number of real executions: 49 Execution total cost: 12.88

Results | Exec Trace | Full Trace

Execution Trace

| | Workloads | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Exec | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
| 1 | | | | | | | | | | 4_c3_large |
| 2 | | | | | | | | | | 3_c3_xlarge |
| 3 | | | | | | | | | | 2_c3_2xlarge |
| 4 | | | | | | | | | | 1_c3_2xlarge |
| 5 | | | | | | | | | | 4_m3_medium |
| 6 | | | | | | | | | | 3_m3_large |
| 7 | | | | | | | | | | 2_m3_xlarge |
| 8 | | | | | | | | | | 1_m3_2xlarge |
| 9 | | | | | | | | | 3_c3_large | |
| 10 | | | | | | | | | 2_c3_xlarge | |
| 11 | | | | | | | | | 2_c3_large | |
| 12 | | | | | | | | | 1_c3_xlarge | |

**Verde Escuro**

Execução real que satisfez o SLA

**Verde Claro**

Predição de Configuração Candidata

**Vermelho Escuro**

Execução real que não satisfez o SLA

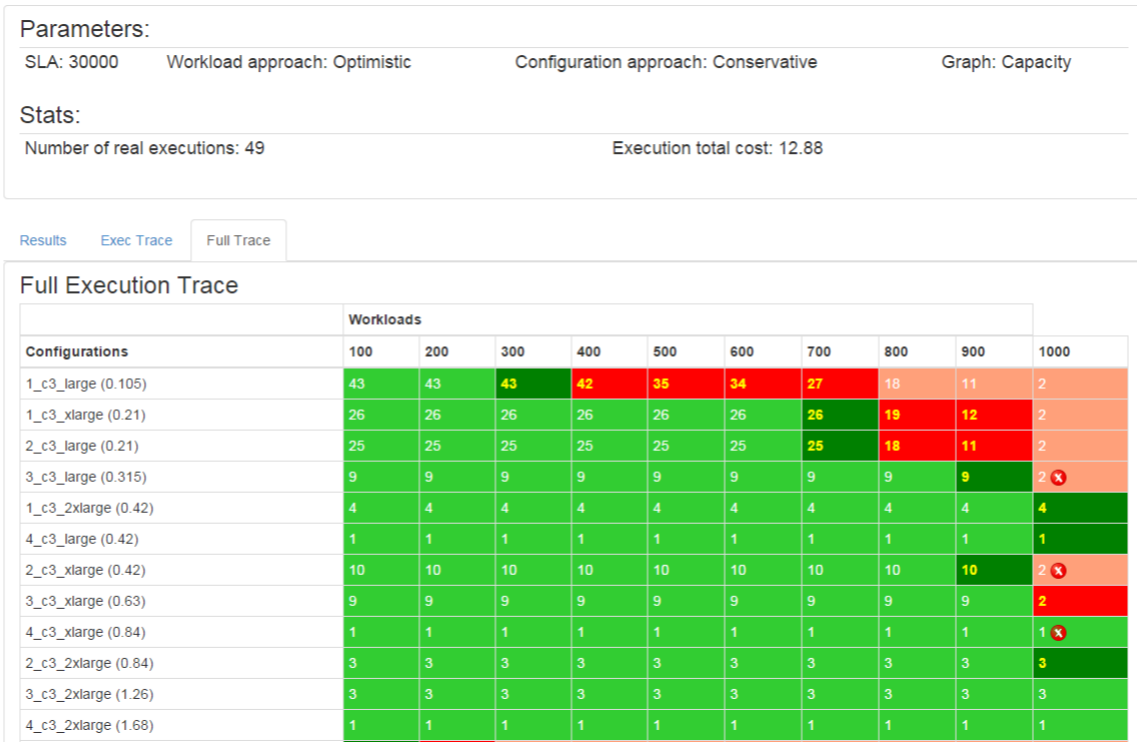
**Vermelho Claro**

Predição de Configuração Rejeitada

As células em cores claras são as Configurações para as quais não houve uma execução real, mas cujo desempenho foi inferido. O número constante da célula indica qual execução real levou o CloudCapacitor a marcar a Configuração como Candidata ou Rejeitada para a Carga de Trabalho em questão.

Quando são feitas essas predições de desempenho, em que o sucesso ou fracasso da Configuração em satisfazer o SLA é inferido com base no desempenho de outra Configuração, há a possibilidade de que a predição não tenha sido feita corretamente. Quando essa situação acontece, o sistema faz uma marcação na célula com um pequeno círculo vermelho com um × branco inscrito. A imagem da Figura 18 mostra essa marcação de erro de predição na coluna referente à Carga de Trabalho de valor 1000 e linhas referentes às Configurações “3_c3_large” e “2_c3_xlarge”, onde a predição afirma erroneamente

Figura 18 – Rastro Completo da Execução do Capacitor Web



Parameters:

SLA: 30000 Workload approach: Optimistic Configuration approach: Conservative Graph: Capacity

Stats:

Number of real executions: 49 Execution total cost: 12.88

Results Exec Trace Full Trace

Full Execution Trace

| | Workloads | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Configurations** | **100** | **200** | **300** | **400** | **500** | **600** | **700** | **800** | **900** | **1000** |
| 1_c3_large (0.105) | 43 | 43 | 43 | 42 | 35 | 34 | 27 | 18 | 11 | 2 |
| 1_c3_xlarge (0.21) | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 19 | 12 | 2 |
| 2_c3_large (0.21) | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 18 | 11 | 2 |
| 3_c3_large (0.315) | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 2 |
| 1_c3_2xlarge (0.42) | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 4_c3_large (0.42) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2_c3_xlarge (0.42) | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 2 |
| 3_c3_xlarge (0.63) | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 2 |
| 4_c3_xlarge (0.84) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2_c3_2xlarge (0.84) | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 3_c3_2xlarge (1.26) | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 4_c3_2xlarge (1.68) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

que o SLA não seria satisfeito, e "4_c3_xlarge", onde a predição errou ao inferir que a Configuração conseguiria satisfazer o SLA. A explicação de como o Capacitor Web é capaz de saber se uma predição está correta ou não é dada no Capítulo 5, que detalha os experimentos realizados.

O rastreamento completo apresentado nessa tela permite a visualização da eficiência atingida pela aplicação da Heurística nos testes de desempenho. Quanto maior a quantidade de células pintadas em cor clara (quer sejam verdes ou vermelhas), maior o número de predições, o que significa menor custo e menor tempo gasto na Avaliação. Do mesmo modo, quanto menor a presença dos círculos vermelhos indicadores de erro de predição, maior a precisão alcançada pela técnica de Inferência de Desempenho ao prever quais Configurações são capazes de atender cada Carga de Trabalho.

## 4.3 Resumo

Este capítulo apresentou a implementação concreta do Processo de Avaliação proposto no Capítulo 3 sob a forma de uma biblioteca a ser usada na criação de sistemas mais completos destinados à avaliação de desempenho e capacidade de nuvens de infraestrutura. Foram explicados os detalhes de programação necessários ao entendimento de como os conceitos vistos foram concretizados em software, ao tempo em que foram descritos os

passos envolvidos na customização das tarefas mais específicas ou que conferem flexibilidade ao Processo.

Foi mostrado ainda o sistema web criado para validação da implementação da biblioteca e para apoiar a execução dos experimentos realizados neste trabalho, com a descrição de sua interface de entrada de dados e apresentação de resultados.

O Capítulo 5 a seguir explica em detalhes os experimentos realizados, detalhando a metodologia utilizada, a Aplicação sob Teste escolhida, como foi implantada e como foram realizadas as execuções para coleta de dados de desempenho. Além disso, serão analisados os resultados obtidos pela aplicação das 9 Heurísticas, comprovando e quantificando a eficiência do Processo de Avaliação de Capacidade proposto e sua técnica de Inferência de Desempenho.

# 5 Experimentos e Resultados

Este capítulo vem apresentar os experimentos realizados como forma de verificação do Processo de Avaliação de Capacidade por Inferência de Desempenho proposto no Capítulo 3. Inicialmente é apresentada a metodologia utilizada para construção dos experimentos, com a descrição da Aplicação sob Teste escolhida, como foi implantada e como foram realizadas as execuções para coleta de dados de desempenho. Depois são apresentados os resultados obtidos por cada uma das 9 Heurísticas ao fazer a Avaliação de Capacidade da Aplicação. Esses resultados são usados para uma comparação qualitativa das Heusrísticas entre si e para atestar a eficiência do Processo de Avaliação de Capacidade proposto e sua técnica de Inferência de Desempenho tanto quanto à economia de tempo e custo como quanto à precisão de acerto de seuas predições.

## 5.1 Metodologia

A fim de validar a eficiência da Inferência de Desempenho no apoio ao Planejamento de Capacidade, foram realizadas seções de avaliação de capacidade de uma aplicação implantada em um provedor de nuvem de infraestrutura como serviço.

A aplicação escolhida foi o WordPress (WORDPRESS, 2014), um motor de construção e administração de *blogs*. Sua escolha foi motivada por ser uma aplicação bem conhecida, de utilização via web, ideal para implantação em ambiente de nuvem, e com componentes arquiteturais escaláveis. Além disso, o fluxo de utilização típico apresenta características bem diversificadas quanto ao uso de recursos de CPU e memória, rede, sistema de arquivos e banco de dados.

O Provedor escolhido foi a Amazon, com seu serviço de infraestrutura AWS EC2 (EC2, 2012), onde o WordPress foi implantado em duas camadas: uma para o banco de dados MySQL, e outra para a aplicação, executada pelo servidor Apache HTTPD. Como balanceador de carga, foi utilizada uma máquina executando o servidor web Nginx. A Figura 19 mostra um panorama geral dessa implantação.

Devido a restrições de custo e tempo, os experimentos foram limitados de forma a variar apenas a camada de aplicação, usando de 1 a 4 servidores Apache executando o WordPress. Na imagem, essa variação está representada pela suavização de cor das máquinas dessa camada, no sentido de que podem não estar presentes em certos cenários.

A execução dos testes foi orquestrada pelo Cloud Crawler (CUNHA, 2012), que automatizou as tarefas de iniciar e parar as todas as instâncias, configurar o balanceador de carga de acordo com o número de instâncias testadas na camada de aplicação, iniciar e

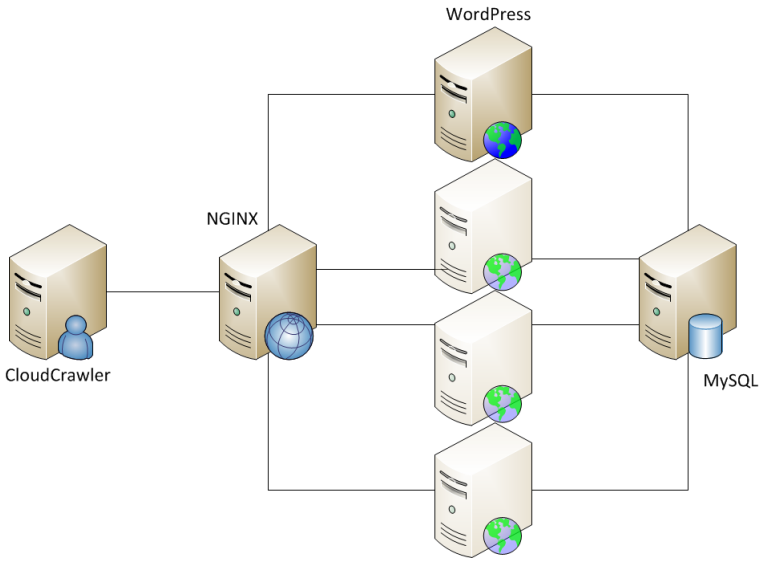



Figura 19 – Implantação do WordPress na AWS EC2 para Avaliação de Capacidade

parar a execução dos testes, controlando as Cargas de Trabalho impostas à Aplicação sob Teste (o WordPress) e coletando os dados de desempenho para cada execução.

De forma a viabilizar uma *baseline* para validação e verificação das predições de desempenho inferidas pelo Processo de Avaliação implementado pela biblioteca CloudCapacitor, foram efetivamente executados testes de desempenho para todas as combinações de Configurações e Cargas de Trabalho. A esse conjunto de dados reais de execução foi dado o nome de "oráculo" e esse procedimento de teste de todas as possibilidades foi considerado como uma Heurística de "Força Bruta". As 9 Heurísticas propostas no Capítulo 3 foram comparadas entre si e com a Heurística de Força Bruta, que não faz qualquer inferência de desempenho. O resultado dessas comparações serão analisados nas seções seguintes.

Para compor o Espaço de Implantação do experimento desenvolvido para este trabalho, foram escolhidos 7 Tipos de Máquinas Virtuais oferecidos pelo serviço EC2:

- m3_medium
- m3_large
- m3_xlarge
- m3_2xlarge
- c3_large
- c3_xlarge
- c3_2xlarge

Para cada um desses Tipos de Máquinas, foram criadas Configurações com 1, 2, 3 e 4 instâncias, levando a um total de 28 Configurações diferentes no Espaço de Implantação, divididas em duas Categorias distintas, "C3" e "M3". As Cargas de Trabalho para este experimento foram quantificadas em número de usuários fazendo requisições ao WordPress. Foram criadas 10 Cargas de Trabalho representando 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000 usuários. Com isso, foram coletados dados de desempenho para 280 cenários diferentes, ou seja, foram testadas as 28 Configurações em cada uma das 10 Cargas de Trabalho especificadas para a Avaliação de Capacidade do WordPress na nuvem.

O teste de desempenho consistiu em fazer com que a Aplicação sob Teste WordPress atendesse à demanda imposta pelo acesso de tantos usuários quanto especificados na Carga de Trabalho no período de 1 hora. Cada usuário disparava a seguinte sequência de requisições:

1. Efetuar *logon*
2. Inserir uma postagem
3. Visitar uma postagem específica
4. Alterar uma postagem
5. Efetuar pesquisa por palavra-chave
6. Alterar uma postagem
7. Efetuar *logoff*

A Métrica de Desempenho usada no experimento foi "Tempo de Resposta Total", ou seja, o tempo total decorrido entre o envio da primeira requisição da sequência acima e o momento em que o cliente recebeu a resposta para última requisição da sequência. Assim, para ser considerada como Candidata, uma Configuração devia ser capaz de atender 90% dos conjuntos de requisições em um tempo total abaixo do tempo informado na entrada do parâmetro SLA.

A seguir serão discutidos os resultados obtidos com o Processo de Inferência de Desempenho e suas Heurísticas na indicação das Configurações capazes de executar a Aplicação sob Teste sob vários níveis de demanda. Foram avaliadas a precisão do Processo na predição de capacidade e a economia de tempo e custo na execução dos testes.

## 5.2 Avaliação dos Resultados

A partir da execução do Processo de Inferência de Desempenho implementado pela biblioteca CloudCapacitor, usada no desenvolvimento do Capacitor Web, foram realizadas

Avaliações de Capacidade em 280 cenários de implantação do WordPress no serviço AWS EC2.

Com base nos resultados desses testes, foram avaliados os seguintes aspectos em relação ao desempenho do Processo:

**Acurácia**

Relação entre acertos e erros na predição de capacidade das Configurações

**Eficiência**

Grau de redução do número de execuções reais e, consequentemente, do custo e do tempo total gasto nos testes de desempenho.

### 5.2.1 Acurácia

A avaliação da Acurácia das predições realizadas pelo Processo de Inferência visa a ratificar a expectativa de que, uma vez identificada uma relação de capacidade entre diferentes Configurações oferecidas por um Provedor de infraestrutura na nuvem, é possível inferir o desempenho de uma Aplicação executando em uma Configuração com base no desempenho observado em uma execução real da Aplicação em uma Configuração distinta.

| Heurísticas | SLA (segundos) | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | | | 20 | | | 30 | | | 40 | | | 50 | | |
| | P | R | F | P | R | F | P | R | F | P | R | F | P | R | F |
| CC | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,98** | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| CO | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,99** | 1,00 | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| CP | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,98** | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| OC | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,99** | **0,99** | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| OO | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,99** | 1,00 | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| OP | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,98** | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| PC | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,98** | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| PO | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,99** | 1,00 | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| PP | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,98** | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |

Tabela 1 – Acurácia da Inferência de Desempenho no Grafo por Capacidade

| Heurísticas | SLA (segundos) | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | | | 20 | | | 30 | | | 40 | | | 50 | | |
| | P | R | F | P | R | F | P | R | F | P | R | F | P | R | F |
| CC | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,97** | **0,98** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| CO | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,98** | 1,00 | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| CP | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,97** | **0,98** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| OC | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,97** | **0,98** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| OO | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,98** | 1,00 | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| OP | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,97** | **0,98** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| PC | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,99** | **0,97** | **0,98** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| PO | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,98** | 1,00 | **0,99** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| PP | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | **0,99** | **0,97** | **0,98** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |

Tabela 2 – Acurácia da Inferência de Desempenho no Grafo por Preço

Assim, este trabalho submeteu o Processo à verificação de sua acurácia ao realizar predições para o desempenho do WordPress executando em máquinas virtuais do provedor Amazon Web Services. O cálculo de Precision and Recall (POWERS, 2011) sobre os resultados dos experimentos realizados mostra que o Processo de Inferência de Desempenho é capaz de obter índices de precisão muito próximos de 100%, como pode ser observado nas Tabelas 1 e 2.

A Tabela 1 apresenta os dados de acurácia das predições realizadas usando-se o Espaço de Implantação estruturado pelas relações de capacidade entre as Configurações. Já a Tabela 2 mostra a acurácia das predições feitas utilizando-se o Espaço de Implantação organizado conforme a relação de preço entre as Configurações. Em ambas, as linhas exibem as Heurísticas usadas na execução do Processo e as colunas, agrupadas segundo os SLAs aplicados nos testes, de 10 segundos até 50 segundos, os resultados de acurácia calculados para *Precision*, *Recall* e *F-Measure*.

Foram medidos os valores de *Precision*, *Recall* e *F-Measure* para as predições realizadas em todas as combinações de Níveis de Carga de Trabalho e SLAs. Cada célula nessas tabelas apresenta a média obtida pela Heurística nas predições para cada SLA em questão. Considerando os 10 Níveis de Carga de Trabalho utilizados no experimento, de 100 a 1000 usuários variando em intervalos de 100, as fórmulas para o cálculo das médias são as seguintes:

$$\textbf{\textit{Precision}:}\quad \frac{\sum_{i=100}^{1000} \frac{true\ positives}{true\ positives\ +\ false\ positives}}{10}$$

$$\textbf{\textit{Recall}:}\quad \frac{\sum_{i=100}^{1000} \frac{true\ positives}{true\ positives\ +\ false\ negatives}}{10}$$

$$\textbf{\textit{F-Measure}:}\quad \frac{\sum_{i=100}^{1000} 2 \cdot \frac{Precision\ \times\ Recall}{Precision\ +\ Recall}}{10}$$

Observando as tabelas, nota-se que, dentre os 5 níveis de SLA avaliados, em apenas um deles o Processo deixou de obter 100% de acurácia nas predições. Durante a execução dos experimentos foram observados momentos de flutuação no desempenho das máquinas virtuais disponibilizadas pelo Provedor. Essa flutuação afetou os testes para o SLA igual a 30 segundos, refletindo em erros de predição. Essa é na verdade uma limitação constatada do Processo de Avaliação de Capacidade por Inferência de Desempenho, que está sujeito às flutuações do Provedor por fazer uso, ainda que pouco frequente, de execuções reais. De fato, oscilações no desempenho da infraestrutura do Provedor são comuns e já foram observados por (IOSUP; YIGITBASI; EPEMA, 2011) e (CUNHA; MENDONCA; SAMPAIO, 2011). O impacto dessas flutuações, porém, é mínimo, reduzindo a acurácia do Processo de Inferência em, no máximo, 3%.

Embora afetado pela oscilação do desempenho no Provedor, o Processo de Inferência de Desempenho ainda se mostra bastante preciso quando observados os demais cenários e SLAs avaliados. Espera-se, também, que abordagens de avaliação de capacidade baseadas puramente em simulação devam ser igualmente ou ainda mais afetados, justamente por não levarem flutuações de desempenho em conta nas simulações, uma vez que é difícil prever em que situações uma flutuação pode ocorrer e, principalmente, qual a intensidade dessa flutuação e seu impacto sobre o desempenho geral de diversos tipos de Configurações.

Com base na acurácia constatada, a seção seguinte analisa a eficiência do Processo de Inferência de Desempenho, aferindo o desempenho de economia de tempo e custo trazido pela aplicação das diversas Heurísticas de seleção de Configurações.

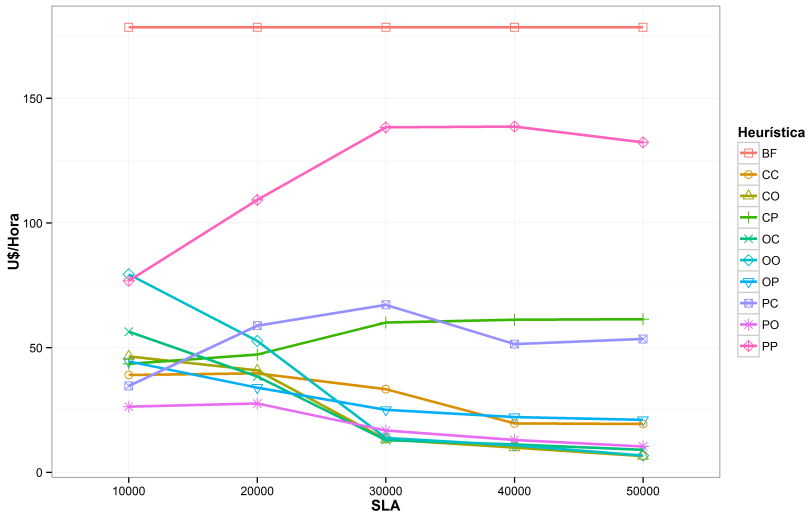
### 5.2.2 Eficiência

Esta seção apresenta os resultados de eficiência atingidos pelas Heurísticas usadas pelo Processo de Inferência de Desempenho sob dois aspectos distintos: o custo total da Avaliação e a quantidade de execuções realizadas pelas Heurísticas. Esse custo foi calculado somando-se o preço da hora de utilização, conforme a tabela de preços do provedor na data realização dos testes, para cada uma das Configurações para as quais foram realizadas execuções reais na nuvem.

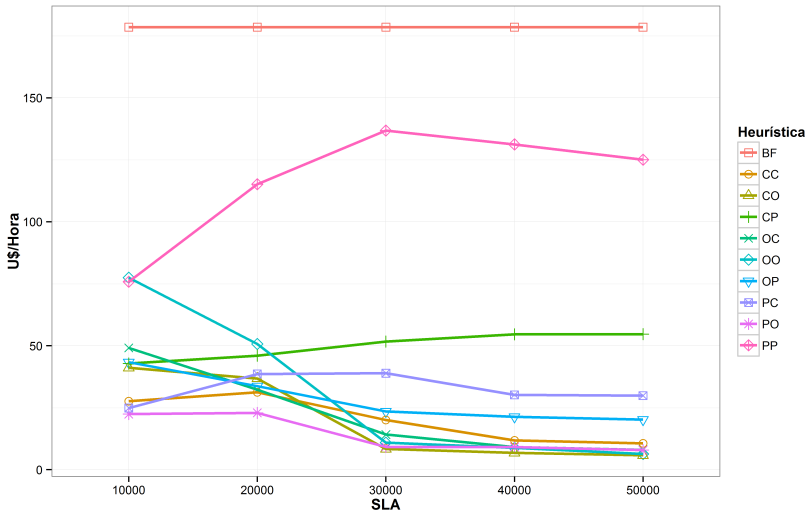
Para cada aspecto analisado, custo e número de execuções, foram realizadas duas baterias iguais de testes, uma utilizando o Espaço de Implantação gerado com base nas relações de poder computacional entre as Configurações e outra com o Espaço de Implantação baseado nas relações de preço. Essa variação foi feita para que fosse possível estudar o efeito da regra de formação do Espaço de Implantação sobre o desempenho geral do Processo de Inferência de Desempenho. Embora haja alguma diferença entre os resultados obtidos com ambas as representações, a percepção geral da eficiência das Heurísticas não muda.

A Figura 20 mostra os gráficos dos resultados obtidos pelas 9 Heurísticas em termos de custo total dos testes de Avaliação considerando SLAs entre 10 e 50 segundos. O primeiro gráfico apresenta os resultados quando os testes foram executados com a representação do Espaço de Implantação baseado na relação de capacidade entre as Configurações. O segundo gráfico considera o grafo da relação de preço das Configurações usado para representar o Espaço de Implantação.

No topo de ambos os gráficos vê-se um linha horizontal que representa a *baseline* de comparação, que é o custo total da Heurística de Força Bruta (*Brute Force - BF*). Como o custo total para execução da Aplicação em todas as combinações de Configurações e Cargas de Trabalho é sempre o mesmo, independente do SLA requerido, a linha do gráfico para essa Heurística sempre é uma linha horizontal.




(a) Grafo por Capacidade

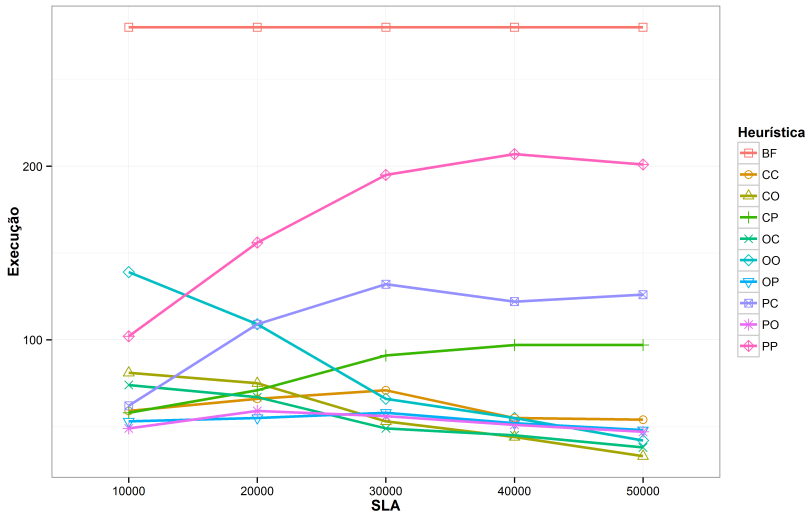



(b) Grafo por Preço

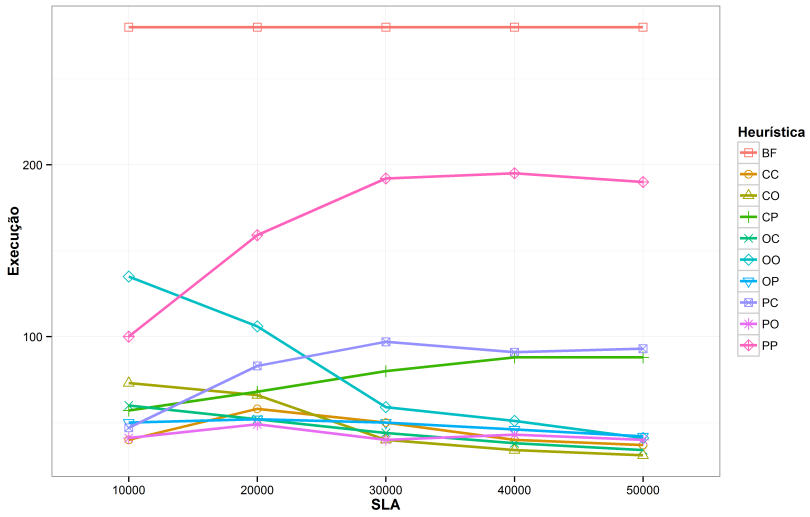
Figura 20 – Avaliação da Eficiência de Custo das Heurísticas

A análise das imagens dos gráficos mostra que mesmo a Heurística com o pior desempenho no que se refere ao custo total da Avaliação já apresenta uma redução em relação à Força Bruta. Por outro lado, as melhores Heurísticas chegam a representar uma economia da ordem de 96% em comparação com o que seria gasto com a execução de todas as combinações de Configurações e Cargas de Trabalho.

Embora o comportamento das Heurísticas varie em função do SLA, é possível notar que quando a exigência do SLA é mais moderada o comportamento de todas as Heurísticas se estabiliza, tornando possível identificar que algumas delas tendem a ser mais econômicas que as outras. Ainda que não seja possível afirmar que uma só Heurística seja a melhor em todas as situações, pode-se considerar que a Heurística Pessimista Otimista se mostra como a mais econômica em geral. A Heurística Conservadora Otimista merece atenção para os SLAs mais brandos, com os menores custos absolutos nessas circunstâncias.




(a) Grafo por Capacidade




(b) Grafo por Preço

Figura 21 – Avaliação da Eficiência do Número de Execuções das Heurísticas

A Figura 21 mostra o desempenho das Heurísticas sob o aspecto da quantidade de execuções reais da Aplicação sob Teste no ambiente de nuvem alvo da Avaliação. Assim como a Figura 20, essa imagem apresenta dois gráficos, que retratam o comportamento do Processo de Inferência ao usar as representações do Espaço de Implantação com grafos das relações de Capacidade e Preço entre as Configurações.

A análise desses gráficos faz notar um número de execuções até 88% menor em relação à Heurística Força Bruta. Isso se reflete em menor tempo gasto no planejamento de capacidade e, consequentemente, menores custos, não só como já visto no que diz respeito à economia de horas de máquina, mas também de outros custos envolvidos em um projeto de software.

Tanto no aspecto custo como no aspecto quantidade de execuções, nota-se que

a Hedurística Pessimista Pessimista tem um desempenho muito ruim, exigindo muitas execuções e, por isso, elevando muito o custo do planejamento de capacidade. As Heurísticas Pessimista Conservadora e Conservadora Pessimista ainda aparecem um pouco descoladas do desempenho das outras Heurísticas, embora com uma redução em torno de 65% no número de execuções.

Os menores números de execuçõs são atingidos pelas Heurísticas Otimista Conservadora e Conservadora Otimista, sob os SLAs mais brandos. Porém, como não se saem tão bem sob SLAs mais rígidos, como 10 segundos, a Heurística Pessimista Otimista ganha destaque por ter comportamento mais estável, figurando entre as mais econômicas no aspecto de quantidade de execuções sob a maioria dos SLAs avaliados.

## 5.3 Resumo

Este capítulo descreveu os experimentos realizados neste trabalho como forma de comprovar a hipótese de que é possível estabelecer uma relação de capacidade entre as diversas configurações de máquinas usadas na implantação de uma aplicação e, partindo dessa relação, é possível inferir o desempenho da aplicação em uma configuração com base no desempenho observado em outra.

A execução de um sistema de avaliação desenvolvido com o uso da biblioteca CloudCapacitor, que implementa o Processo de Avaliação de Capacidade por Inferência de Desempenho proposto no Capítulo 3, demonstrou que a Inferência de Desempenho é viável como técnica de suporte ao planejamento de capacidade, uma vez que a precisão de acertos das predições realizadas nos testes se aproxima muito dos 100%.

Além disso, os experimentos mostram também que a aplicação do Processo de Inferência de Desempenho pode trazer até 96% de economia financeira e até 88% de economia de tempo na realização das atividades de testes de desempenho.

| Métricas | Heurísticas | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | CC | CO | CP | OC | OO | OP | PC | PO | PP |
| Custo (US$) | 25,23 | 21,55 | 52,31 | 24,05 | 31,74 | 28,85 | 42,79 | **16,54** | 117,95 |
| Tempo (h) | 50,10 | 80,30 | 50,60 | 53,00 | 53,00 | 79,50 | 96,20 | **47,50** | 169,70 |
| Acurácia | 0,997 | 0,998 | 0,997 | 0,997 | 0,998 | 0,997 | 0,997 | **0,998** | 0,997 |

Tabela 3 – Resultados Consolidados da Inferência de Desempenho

A Tabela 3 apresenta uma consolidação geral dos resultados obtidos por todas as Heurísticas definidas pelo Processo de Avaliação de Capacidade por Inferência de Desempenho. Como pode ser observado, a Heurística Pessimista-Otimista atinge os melhores resultados, predizendo com uma precisão de 99,8% as melhores Configurações para executar a Aplicação sob Teste e gastando menos de US$ 17,00 e menos de 48 horas na média global para completar a Avaliação de Capacidade.

Destacam-se ainda outras Heurísticas como Conservadora-Conservadora e Otimista-Conservadora, que, na média global, obtiveram resultados não muito acima do atingido pela Pessimista-Otimista. A Heurística Pessimista-Pessimista, como já mostrado na análise dos gráficos das Figuras 20 e 21, apresentou resultados muito ruins quanto ao custo e ao tempo gastos para completar a Avaliação.

Esses dados consolidados comprovam ainda a grande precisão com que a predição pela Inferência de Desempenho aponta as melhores Configurações para executar a Aplicação sob Teste sob as mais variadas Cargas de Trabalho.

# 6 Conclusão

# Referências

AVETISYAN, A. I. et al. Open cirrus: A global cloud computing testbed. *Computer*, Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, 17 th Fl New York NY 10016-5997 United States, v. 43, n. 4, p. 35–43, 2010. Citado na página 11.

AZURE. Windows azure platform. <http://www.windowsazure.com/>. 2012. Citado na página 8.

CLOUDHARMONY. Cloudharmony benchmarking the cloud. <http://cloudharmony.com/benchmarks>. 2014. Citado 2 vezes nas páginas 6 e 14.

CUNHA, M.; MENDONCA, N.; SAMPAIO, A. Avaliação do custo por usuário de uma aplicação de rede social na amazon ec2. In: *WCGA-XI Workshop em Clouds, Grids e Aplicações (SBRC-2011)*. [S.l.: s.n.], 2011. Citado na página 56.

CUNHA, M.; MENDONÇA, N.; SAMPAIO, A. Investigating the impact of deployment configuration and user demand on a social network application in the amazon ec2 cloud. In: IEEE. *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. [S.l.], 2011. p. 746–751. Citado na página 2.

CUNHA, M.; MENDONÇA, N.; SAMPAIO, A. Cloud crawler: Um ambiente programável para avaliar o desempenho de aplicaç oes em nuvens de infraestrutura. *31o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC 2013*, 2013. Citado 3 vezes nas páginas 10, 12 e 14.

CUNHA, M. C. C. *Um Ambiente Programável para Avaliar o Desempenho de Aplicaçoes em Nuvens de Infraestrutura*. Dissertação (Mestrado) — Universidade de Fortaleza, 2012. Citado 4 vezes nas páginas 2, 6, 41 e 52.

EC2. Amazon elastic compute cloud. <http://aws.amazon.com/ec2>. 2012. Citado 3 vezes nas páginas 8, 38 e 52.

FITTKAU, F.; FREY, S.; HASSELBRING, W. Cdosim: Simulating cloud deployment options for software migration support. In: IEEE. *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the*. [S.l.], 2012. p. 37–46. Citado 4 vezes nas páginas 6, 8, 9 e 14.

GAMMA, E. et al. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Pearson Education, 1994. Citado na página 45.

HANSON, D. H. The ruby on rails framework. <https://www.rubyonrails.org>. 2014. Citado na página 35.

HÜTTERMANN, M. *DevOps for developers*. [S.l.]: Springer, 2012. Citado na página 13.

IOSUP, A.; YIGITBASI, N.; EPEMA, D. On the performance variability of production cloud services. In: IEEE. *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*. [S.l.], 2011. p. 104–113. Citado 2 vezes nas páginas 2 e 56.

JAYASINGHE, D. et al. Variations in performance and scalability when migrating n-tier applications to different clouds. In: IEEE. *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. [S.l.], 2011. p. 73–80. Citado na página 2.

JAYASINGHE, D. et al. Expertus: A Generator Approach to Automate Performance Testing in IaaS Clouds. In: IEEE COMPUTER SOCIETY. *Cloud Computing (CLOUD), 2012 IEEE International Conference on*. [S.l.], 2012. p. 73–80. Citado 4 vezes nas páginas 10, 11, 14 e 41.

JUNG, G. et al. Cloudadvisor: A recommendation-as-a-service platform for cloud configuration and pricing. In: IEEE. *Services (SERVICES), 203 IEEE Ninth World Congress on*. [S.l.], 2013. p. 456–463. Citado 3 vezes nas páginas 6, 8 e 14.

LI, A. et al. CloudCmp: Comparing Public Cloud Providers. In: *Internet Measurement Conference*. [S.l.: s.n.], 2010. Citado 3 vezes nas páginas 6, 7 e 8.

LI, A. et al. Cloudprophet: predicting web application performance in the cloud. *ACM SIGCOMM Poster*, 2011. Citado 3 vezes nas páginas 1, 6 e 9.

MALKOWSKI, S. et al. Cloudxplor: A tool for configuration planning in clouds based on empirical data. In: ACM. *Proceedings of the 2010 ACM Symposium on Applied Computing*. [S.l.], 2010. p. 391–398. Citado 3 vezes nas páginas 6, 7 e 14.

MATSUMOTO, Y. The ruby language. <https://www.ruby-lang.org>. 2014. Citado na página 35.

PÉREZ-CASTILLO, R.; GUZMAN, I. G.-R. D.; PIATTINI, M. Knowledge discovery metamodel-iso/iec 19506: A standard to modernize legacy systems. *Computer Standards & Interfaces*, Elsevier, v. 33, n. 6, p. 519–532, 2011. Citado na página 9.

POWERS, D. M. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies, 2:1-37-63*, 2011. Citado na página 56.

RACKSPACE. The rackspace cloud. <http://www.rackspacecloud.com/>. 2012. Citado na página 8.

RODERO-MERINO, L. et al. From infrastructure delivery to service management in clouds. *Future Generation Computer Systems*, Elsevier, v. 26, n. 8, p. 1226–1240, 2010. Citado na página 1.

SCHEUNER, J. et al. Cloud workbench-infrastructure-as-code based cloud benchmarking. *arXiv preprint arXiv:1408.4565*, 2014. Citado 3 vezes nas páginas 10, 13 e 14.

SILVA, M. et al. Cloudbench: Experiment automation for cloud environments. In: IEEE. *Cloud Engineering (IC2E), 2013 IEEE International Conference on*. [S.l.], 2013. p. 302–311. Citado 5 vezes nas páginas 3, 10, 11, 14 e 41.

SPEC. Java virtual machine benchmark 2008. <http://www.spec.org/jvm2008/>. 2008. Citado na página 7.

WORDPRESS. Wordpress: Blog tool, publishing platform, and cms. <https://wordpress.org/>. 2014. Citado na página 52.

YAML. Yaml specification. <http://www.yaml.org/spec/1.2/spec.html>. 2009. Citado na página 37.