FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA – UNIFOR

Heurísticas para Predição de Configurações de Custo Mínimo para Execução de Aplicações em Ambientes de Nuvem de Infraestrutura

Marcelo Canário Gonçalves

FORTALEZA

2014

Marcelo Canário Gonçalves

Heurísticas para Predição de Configurações de Custo Mínimo 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
Heurísticas para Predição de Configurações de
Custo Mínimo para Execução de Aplicações em Ambientes de Nuvem de Infraestrutura/ Marcelo Canário Gonçalves. – FORTALEZA, 2014-
35 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

Heurísticas para Predição de Configurações de Custo Mínimo 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

Segundo a **??**, 3.1-3.2), o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
abnTeX	ABsurdas Normas para TeX

Lista de símbolos

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

Sumário

1 Introdução

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. Ao comparar os serviços de provedores de computação em nuvem com a administração de um centro de dados próprio, a melhoria dos indicadores de desempenho e de custo, como redução de tempo de resposta, redução/otimização de custo de operação e melhores ferramentas com mais facilidades de gerenciamento, está entre os principais benefícios buscados a partir da adoção do ambiente de infraestrutura como serviço (IaaS – Infrastructure as a Service) (**????**).

Em geral, provedores de IaaS cobram um valor em função do tempo de ocupação de uma máquina virtual, 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 da 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. A aplicação deve manter ou superar na nuvem o nível de desempenho apresentado quando executada em centro de dados próprio e com indicadores de custo menores, a fim de que se justifique o investimento feito na migração de ambiente. Para se chegar a essa conclusão, faz-se necessário conhecer o comportamento da aplicação na nova implantação 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 exercida sobre a aplicação e as diversas possibilidades de variação de 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 (**??**). 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 saber se uma determinada configuração de recursos do provedor é capaz de atender a uma demanda específica, é preciso antes enumerar os indicadores de desempenho que mais interessam à aplicação e a partir daí estabelecer os valores aceitáveis para esses indicadores. Uma vez estabelecidos os valores aceitáveis, pode-se implantar a aplicação sob essa configuração de recursos e então aplicar diferentes níveis de carga de trabalho sobre a 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 muito diferente sob o mesmo nível de carga de trabalho a depender do momento em que sejam ativados (**??????**). 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 ensaios 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 (**??**). 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. Para evitar que sejam feitos testes com todas as combinações de provedores, configurações, horários, cargas de trabalho e métricas avaliadas, é possível lançar mão de técnicas de predição.

Através da predição, é possível estimar com razoável aproximação o desempenho que a aplicação apresentará ao ser executada em vários perfis de configuração diferentes, permitindo a determinação de qual configuração de menor custo capaz de executar a aplicação e sem a necessidade da realização completa dos testes. Como consequência, o custo dessa fase de ensaios pode ser reduzido sensivelmente.

Este trabalho propõe heurísticas de predição de custo mínimo para execução de aplicações em ambientes de nuvem de infraestrutura, bem como um arcabouço de programação que apoia a implementação dessas heurísticas. Além disso, o trabalho estuda os resultados apresentados pela aplicação das heurísticas propostas quanto ao custo total de execução da fase de ensaios para escolha da melhor configuração capaz de executar uma aplicação e quanto à acuidade dos resultados da predição em si.

2 Trabalhos Relacionados

Apresentamos neste capítulo alguns trabalhos cujos objetivos estejam alinhados com a ideia da avaliação de desempenho de aplicações executadas em ambientes de computação em nuvem. Fazemos então uma análise de seus objetivos e resultados alcançados, bem como traçamos as semelhanças com este trabalho, avaliando quais são nossas contribuições diferenciais.

2.1 cloud comparison services

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" (**??**), agrega dados de testes de desempenho realizados desde 2009 em mais de 60 provedores de nuvem. 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 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.

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

2.1.2 Cloud Ops

Este trabalho apresenta os resultados da medição do desempenho e da disponibilidade dos serviços oferecidos por nove provedores de nuvem (**??**). A coleta dos resultados foi realizada por um cliente *Javascript* que era executado ao final do carregamento de uma página web, aproveitando o tempo ocioso em que o usuário da página estava lendo o seu conteúdo. Esse *script* recebia uma lista de provedores de nuvem e de, forma randômica, escolhia o provedor que seria avaliado. Para cada provedor, eram enviados dois tipos de requisição: uma com 50 *bytes* e outra com 100 *Kbytes*. A requisição de menor tamanho era enviada duas vezes, com cada envio passando um parâmetro diferente, cujo conteúdo era definido randomicamente, de modo a evitar a utilização de *caches*. Essas requisições eram respondidas por uma aplicação hospedada em cada um dos provedores avaliados, permitindo a coleta de quatro métricas pelo *script* cliente: disponibilidade (o serviço está no ar), *HTTP Connect Time* (tempo necessário para estabelecer a conexão), *HTTP Response Time* (tempo necessário para receber a resposta da requisição) e *Throughput* (taxa de transferência da requisição).

Esse trabalho tem o mérito de conseguir mapear o desempenho de vários provedores de nuvem sob o ponto de vista de usuários espalhados por todo o mundo. Através dele, é possível escolher o melhor provedor para hospedar uma determinada aplicação, em termos das quatro métricas coletadas na época da realização dos experimentos, com base apenas na localização física dos clientes. Por outro, o trabalho não disponibiliza publicamente nenhum *benchmark* ou ferramenta de apoio, de modo que não é possível replicar os testes realizados em outros cenários ou com outros provedores.

2.2 trial-and-test tools

2.2.1 Expertus

Um outro trabalho recente foi descrito em (**??**), onde os autores propõem um ambiente para execução automática de testes de desempenho em nuvens IaaS. Nesse ambiente, chamado Expertus, os usuários implementam testes de desempenho através da geração e customização de *scripts* Shell a partir de *templates* especificados na forma de documentos XML (**??**).

2.2.2 CloudBench

(**??**) descrevem o CloudBench como um arcabouço para automação dos testes e avaliação do desempenho de ambientes de nuvem computacional sob o modelo IaaS. CloudBench prevê métricas que dizem respeito não só à aplicação, mas também ao provedor, como latência de provisionamento, bem como a execução dos testes e a coleta de dados

relativos às métricas de desempenho observadas, através da abstração de experimentos, aplicações e máquinas virtuais.

As abstrações criadas pelo CloudBench permitem que um experimento seja especificado através de uma lista de diretivas que descrevem os itens que compõem o experimento, definindo objetos como a aplicação e as instâncias de máquinas virtuais utilizadas.

CloudBench faz a implantação automática da aplicação a ser executada para efeito de testes, desde a criação da máquina virtual no ambiente até a coleta dos dados de desempenho e desligamento das máquinas, com suporte a diversos provedores.

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.

Neste sentido, o arcabouço que propomos com este trabalho se diferencia pelo fato de ser agnóstico em relação à aplicação que deverá ser testada, assim como quanto às métricas que a ela concernem, conferindo ao usuário da solução a oportunidade de avaliar o comportamento da aplicação de seu interesse implantada no ambiente pretendido e sob a perspectiva que lhe for mais conveniente, inclusive em termos de arquitetura de implantação.

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* (**??**).

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.

2.3 prediction tools

2.3.1 CloudProphet

Em (**??**) 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.

2.3.2 CloudCmp

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

Segundo **??**), 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.

Como essa solução compara serviços e recursos da nuvem através da análise de dados previamente coletados a partir da execução de diferentes *benchmarks*, a escolha dos

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

provedores e recursos mais apropriados para uma determinada aplicação só será eficaz se a aplicação utilizar os recursos da nuvem de forma semelhante à dos *benchmarks* avaliados. Além disso, a ferramenta não oferece suporte à execução de novos cenários de avaliação na nuvem, estando limitada àqueles previamente definidos para os respectivos *bechmarks*.

2.3.3 CloudAdvisor

(??)

2.3.4 CDOSim

(??)

2.4 Outros Trabalhos e Considerações

******

Caracteristicas gerais do Cloud Capacitor

Foco em minimização dos custos de execução dos testes

Proposta de análise do impacto da escalabilidade horizontal e/ou vertical nos resultados dos testes

Julgamento de qual tipo de máquina apresenta melhor custo x desempenho para cada faixa de workload

Flexibilidade de implementação da lógica de avaliação através do arcabouço de heurísticas

******

3 Formalização da Solução

A fim de que uma análise criteriosa possa ser feita, apresentamos a seguir 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.

Apresentamos também um formalismo que visa à generalização da metodologia utilizada neste trabalho e a permitir a melhor descrição do raciocínio lógico envolvido no desenvolvimento das heurísticas criadas.

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 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 as Heurísticas desenvolvidas e apresentadas saibam se a Aplicação é capaz de ser executada sobre 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.

As heurísticas desenvolvidas e apresentadas por este trabalho comparam implantações de diferentes Configurações em uma determinada camada da Aplicação sob Teste estudada. Isso permite 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

Damos 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).

Tomemos 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 suponhamos 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.

3.2 Formalismos

Dada uma Aplicação sob Teste A, precisamos identificar, dentre um conjunto de cenários de implantação e execução da aplicação em ambiente de nuvem computacional sob a modalidade de infraestrutura como serviço, sob quais desses cenários a aplicação é executada com sucesso e, dentre esses cenários, quais os de menor custo.

De modo a facilitar uma descrição formal do raciocínio lógico empregado na solução proposta, mostramos a seguir a notação que representa os conceitos descritos na seção anterior. Ao final, será apresentada uma descrição formal do modelo vislumbrado como solução para o problema apresentado.

3.2.1 Métricas de Desempenho

Seja $M(A)$ um conjunto de Métricas de Desempenho de interesse da Aplicação sob Teste A. Seja $|M(A)|$ a quantidade de Métricas de Desempenho $\in M(A)$ avaliadas para a Aplicação sob Teste A.

$M(A) = \{m_1, m_2, \ldots, m_{|P|}\} \mid m_n \in M, 1 \leq n \leq |M| \iff m_n$ é uma Métrica de Desempenho a ser avaliada para a Aplicação sob Teste A.

3.2.2 Provedor

Seja P um conjunto de Provedores. Seja $|P|$ a quantidade de Provedores pertencentes a P.

$P = \{p_1, p_2, \ldots, p_{|P|}\} \mid p_n \in P, 1 \leq n \leq |P| \iff p_n$ é um Provedor de infraestrutura como serviço.

3.2.3 Tipos de Máquinas Virtuais

Seja $T(P)$ um conjunto de Tipos de Máquinas Virtuais fornecidos pelo provedor P.

Seja $|T(P)|$ a quantidade de Tipos de Máquinas Virtuais pertencentes a $T(P)$.

$T(P) = \{t_1, t_2, \ldots, t_{|T(P)|}\} \mid t_n \in T(P), 1 \leq n \leq |T(P)| \iff t_n$ é um Tipo de Máquina Virtual fornecido pelo provedor P.

Definimos $i(t)$ como uma instância de Máquina Virtual do Tipo $t \in T(P)$.

Definimos $t(i)$ como o Tipo de Máquina Virtual $t \in T(P)$ usado para criar a instância i.

Definimos $\$(t)$ como o valor de custo unitário por hora de utilização de uma instância do Tipo t.

Definimos $cpu(t)$ como o tamanho da capacidade computacional do Tipo t.

Definimos $mem(t)$ como o tamanho da capacidade de memória do Tipo t.

Dados dois Tipos de Máquinas Virtuais t_1 e t_2:

$t_1 = t_2 \iff cpu(t_1) = cpu(t_2) \wedge mem(t_1) = mem(t_2)$.

$t_1 < t_2 \iff cpu(t_1) < cpu(t_2) \wedge mem(t_1) < mem(t_2)$.

$t_1 > t_2 \iff cpu(t_1) > cpu(t_2) \wedge mem(t_1) > mem(t_2)$.

$cpu(t_1) < cpu(t_2) \wedge mem(t_1) > mem(t_2) \iff$ relação indeterminada entre t_1 e t_2.

$cpu(t_1) > cpu(t_2) \wedge mem(t_1) < mem(t_2) \iff$ relação indeterminada entre t_1 e t_2.

3.2.4 Categorias de Máquinas Virtuais

Seja $Cat(P)$ um conjunto de Categorias de Máquinas Virtuais fornecidas pelo provedor P.

$Cat(P) = \{cat_1, cat_2, \ldots, cat_{|Cat(P)|}\} \mid cat_n \in Cat(P), 1 \leq n \leq |Cat(P)| \iff$ cat_n é uma Categoria de Máquina Virtual fornecida pelo provedor P.

$cat_n = \{T(P)_1, T(P)_2, \ldots T(P)_{|cat_n|}\} \mid T(P)_m \subset T(P), 1 \leq m \leq |cat_n| \iff$ $T(P)_m$ é um grupo de Tipos de Máquina Virtual fornecidos pelo provedor P.

Definimos $cat(t_n) \in Cat(P)$ como a Categoria atribuída ao Tipo $t_n \in T(P)$ pelo Provedor P.

3.2.5 Configurações

Seja c um grupo com um certo número de Máquinas Virtuais $i(t)$.

Dizemos que c é uma Configuração se $\forall\, i(t) \in c, t$ é constante.

Definimos $t(c)$ como o Tipo de Máquina Virtual usado para criar as instâncias que compõem c. Portanto, temos que $t(c) = t(i), \forall\, i(t) \in c$.

Definimos $\#(c)$ como o tamanho da Configuração c, ou seja, a quantidade de instâncias $i(t)$ usadas para formar c.

Definimos $\$(c)$ como o custo total da configuração, dado por $\$(t) \times \#(c)$.

O conjunto $C(P)$ é o conjunto de todas as Configurações c possíveis de serem formadas a partir dos Tipos $t \in T(P)$ e com um número arbitrário de instâncias $i(t)$.

Dadas duas Configurações c_1 e c_2:

$$c_1 = c_2 \iff t(c_1) = t(c_2) \wedge \#(c_1) = \#(c_2).$$

$$c_1 > c_2 \iff \begin{cases} t(c_1) = t(c_2) \wedge \#(c_1) > \#(c_2) \\ ou \\ t(c_1) > t(c_2) \wedge \#(c_1) = \#(c_2) \end{cases}$$

$$c_1 < c_2 \iff \begin{cases} t(c_1) = t(c_2) \wedge \#(c_1) < \#(c_2) \\ ou \\ t(c_1) < t(c_2) \wedge \#(c_1) = \#(c_2) \end{cases}$$

3.2.6 Espaço de Implantação

Seja um conjunto de Tipos de Máquinas Virtuais $T(DS_x^y) \subset T(P)$.

Seja um conjunto de Configurações C_x^y tal que:

$\forall\, t \in T(DS_x^y),\ \forall\, i \mid x \leq i \leq y,\ \exists\, c \in C_x^y \mid t(c) = t\ \wedge\ \#(c) = i.$

O Espaço de Implantação $DS_x^y(P)$ é um grafo cujo conjunto de vértices é dado por C_x^y e cujas arestas são dadas pelas seguintes operações:

1. Ordenar C_x^y pelo atributo *preço* de cada elemento do conjunto ($\$(c)$).
2. $preço_{\rightarrow} = \forall\, c_1, c_2 \in C_x^y,\ c_1 \rightarrow c_2 \iff \$(c_1) < \$(c_2) \wedge \nexists\, c_3 \mid \$(c_1) < \$(c_3) < \$(c_2)$
3. Ordenar C_x^y pelo atributo CPU de cada elemento do conjunto ($cpu(c)$).
4. $cpu_{\rightarrow} = \forall\, c_1, c_2 \in C_x^y,\ c_1 \rightarrow c_2 \iff cpu(c_1) < cpu(c_2) \wedge \nexists\, c_3 \mid cpu(c_1) < cpu(c_3) < cpu(c_2)$
5. Ordenar C_x^y pelo atributo $Memória$ de cada elemento do conjunto ($mem(c)$).
6. $mem_{\rightarrow} = \forall\, c_1, c_2 \in C_x^y,\ c_1 \rightarrow c_2 \iff mem(c_1) < mem(c_2) \wedge \nexists\, c_3 \mid mem(c_1) < mem(c_3) < mem(c_2)$

3.2.7 Cargas de Trabalho

Seja W um conjunto finito de valores que representam Cargas de Trabalho. Seja $|W|$ a quantidade de Cargas de Trabalho pertencentes a W.

$W = \{w_1, w_2, \ldots, w_{|W|}\} \mid w_n \in W, 1 \leq n \leq |W| \iff w_n$ é um valor válido como carga a ser submetida dentro do domínio de conhecimento da Aplicação sob Teste.

3.2.8 Resultados

Um Resultado r é uma quádrupla no formato $\{m, v, cpu, mem\}$ onde:

$m \in M(A)$ é a Métrica de Desempenho medida no Resultado;

v é o valor medido para a Métrica;

cpu é o percentual de CPU utilizada para a obtenção de v; e

mem é o percentual de memória RAM utilizada para a obtenção de v.

3.2.9 Execuções

Seja $E(A)$ um conjunto de Execuções da Aplicação sob Teste A. Seja $|E|$ a quantidade de Execuções pertencentes a $E(A)$.

$E(A) = \{e_1, e_2, \ldots, e_{|E|}\} \mid e_n \in E(A), 1 \leq n \leq |E(A)| \iff e_n$ é uma tripla no formato $\{c, w, r\} \mid c \in DS_x^y(P), w \in W$ e onde r é o Resultado obtido a partir da Execução da Aplicação na configuração c, submetida à Carga de Trabalho w.

3.2.10 Avaliação da Execução

Seja α um valor de referência definido como parâmetro de sucesso para uma Métrica de Desempenho quando da Execução de um teste da Aplicação.

Seja $e \in E(A)$ uma Execução da Aplicação A

Seja $r \in e$ o Resultado da Execução e

Seja $m \in r$ a Métrica de Desempenho avaliada no Resultado r

Seja $v \in r$ o valor medido para a Métrica m no Resultado r

Seja $atende(r, \alpha)$ uma função tal que:

$$m \text{ é } minimizável \iff atende(r, \alpha) = \begin{cases} \{1, \delta\}, v \leq \alpha \\ \{0, \delta\}, v > \alpha \end{cases}$$

$$m \text{ é } maximizável \iff atende(r, \alpha) = \begin{cases} \{0, \delta\}, v < \alpha \\ \{1, \delta\}, v \geq \alpha \end{cases}$$

Em ambos os casos, δ representa a distância entre o valor de Resultado obtido para a Métrica m na Execução e (ou seja, o valor $e[r[v]]$) e o Valor de Referência ou SLA α.

A representação dessa distância é dada por $\delta = \{grau\ de\ desvio,\ direção\ do\ desvio\}$, onde o *grau de desvio* indica se esse desvio é *alto*, *médio* ou *baixo*, e a *direção do desvio* indica se o resultado obtido está *abaixo* ou *acima* do SLA.

Os limites de comparação do Resultado com o SLA para a classificação do grau de desvio são parâmetros e podem ser modificados conforme a tolerância da Aplicação a flutuações de desempenho do ambiente de nuvem. Os valores desses parâmetros são dados em termos percentuais.

As informações trazidas por δ podem influenciar o modo como as Heurísticas de Avaliação de Capacidade caminham sobre o Espaço de Implantação na busca pelas melhores Configurações. Seu uso, porém, é opcional e fica a critério da própria Heurística.

Apresentamos a seguir uma especificação de funcionamento das Heurísticas, detalhando seu papel dentro deste trabalho, insumos de entrada e as operações que devem fornecer como interface para sua manipulação e controle.

4 Heurísticas de Avaliação de Capacidade

Dadas as definições e formalizações propostas no anteriormente, faz-se necessária a especificação de uma lógica de manipulação das entidades e operações descritas.

Passamos agora a descrever uma proposta de construção de estratégias de avaliação dos Resultados de Execuções a fim de buscar as Configurações de menor preço capazes de executar uma determinada Carga de Trabalho. A essas estratégias demos o nome de Heurísticas de Avaliação.

As Heurísticas de Avaliação de Capacidade, conforme propostas neste trabalho, são implementações de estratégias que buscam encontrar, a partir do Resultado de uma Execução da Aplicação sob Teste sob uma Carga de Trabalho inicial em uma Configuração inicial, se existe uma Configuração mais barata que seja capaz de executar a mesma Carga de Trabalho.

O funcionamento de uma Heurística é totalmente baseado no caminhamento sobre o Espaço de Implantação e sobre um a faixa de valores de Cargas de Trabalho. O tamanho de cada passo nesse caminho depende da lógica empregada pela Heurística e da avaliação do Resultado obtido. Cada passo dado pela Heuristica pode resultar em uma nova Execução dos testes ou em uma conclusão a respeito da capacidade da Configuração avaliada (e outras similares) de executar a Carga de Trabalho a um custo viável. Assim, a inteligência de uma Heurística está na forma como ela decide caminhar sobre o Espaço de Implantação e na faixa de Cargas de Trabalho apresentados.

Apresentamos a seguir o modelo de como uma Heurística deve funcionar, a especificação do arcabouço de implementação construído nesse trabalho, começando por descrever as operações que uma Heurística deve executar.

4.1 Operações Iniciais

Para que uma Heurística de Avaliação de Capacidade seja compatível no âmbito deste trabalho, deve apresentar um conjunto mínimo de operações esperadas para que a lógica da avaliação se complete e o resultado final obtido possa ser considerado válido e comparável com os resultados obtidos por outras Heurísticas.

Além disso, as operações constituem a interface pela qual o controlador das sessões de avaliação pode configurar as Heurísticas e informar-lhe os dados necessários ao controle da sua execução.

Apresentamos esse conjunto mínimo de operações nas subseções a seguir, que

representam o arcabouço necessário para a construção de uma Heurística de Avaliação de Capacidade.

4.1.1 Selecionar Carga de Trabalho Inicial

Este trabalho tem como premissa a necessidade de se identificar quais as Configurações mais baratas em um Provedor capazes de executar diversos níveis de Cargas de Trabalho, tendo como objetivo a otimização de custos para a execução de uma Aplicação so Teste.

Assim, pressupomos que exista uma faixa de valores para os níveis de Cargas de Trabalho a que a Aplicação é costumeiramente submetida e que seja de conhecimento prévio dos responsáveis pela Aplicação.

De posse dessa faixa de valores de Cargas de Trabalho, uma Heurística deve ser capaz de escolher, de acordo com sua estratégia de trabalho, um valor inicial de Carga de Trabalho a ser imposta sobre a Aplicação. A Carga de Trabalho escolhida deve ser retornada para o controlador da sessão, de forma que este possa coordenar a Execução dos testes.

4.1.2 Selecionar Configuração Inicial

Analogamente, a fim de que as atividades da sessão de avaliação possam ter início, é necessário que a Heurística de Avaliação de Capacidade usada selecione uma Configuração inicial.

A escolha da Configuração inicial é feita a partir das Configurações disponíveis no Espaço de Implantação previamente configurado pelo responsável pela avaliação. A Heurística deverá avaliar o conjunto de configurações disponíveis quanto ao seu preço, número de instâncias em cada, etc, de forma a escolher uma Configuração que considere mais adequada à sua estratégia para o início da sessão de avaliação.

4.2 Operações de Controle

Tendo em mãos uma Carga de Trabalho e uma Configuração iniciais, o controlador da sessão de avaliação de capacidade pode ordenar uma Execução de testes, onde serão coletados dados de desempenho relevantes para a Aplicação sob Teste.

Após a primeira Execução, um Resultado contendo os dados de desempenho colhidos é avaliado pelo controlador e, conforme sua decisão, novas Execuções podem se fazer necessárias. Neste caso, a Heurística deve ser novamente invocada, desta vez a selecionar uma nova Carga de Trabalho ou uma nova Configuração a partir do Espaço de

Implantação. Essa interação deve se repetir até que o controlador conclua os testes e dê por encerrada a sessão de avaliação.

Abaixo descrevemos as operações que a Heurística deve prover para que permita ao controlador a correta operação dos testes e da sessão.

4.2.1 Selecionar Nova Configuração

Depois da cada execução de testes, o controlador estará de posse de um Resultado, contendo os dados de desempenho da Aplicação executada sob a Carga de Trabalho e a Configuração selecionadaa. A depender dos dados desse Resultado, o controlador pode decidir executar novos testes em outra Configuração.

Para isso, a Heurística deve ser usada para selecionar a próxima Configuração a ser testada com a Aplicação. Com base no Resultado obtido pela Execução anterior, a Heurística usará sua lógica de navegação para determinar a distância a ser caminhada no Espaço de Implantação em busca da nova Configuração.

A ordem de caminhamento é dada conforme a necessidade identificada pelo controlador, que vai definir se precisa de uma Configuração mais ou menos potente. Porém, a Heurística é quem define, através de sua estratégia, qual será a próxima Configuração usada.

Assim, a Heurística deve prover ao controlador duas operações para escolha da próxima Configuração: uma para Elevar o Nível de Configuração, ou seja, escolher uma Configuração de capacidade superior, e outra para Reduzir o Nível de Configuração, isto é, escolher uma Configuração de capacidade inferior. Em ambos os casos, a Heurística deverá usar como dado de entrada o Resultado da última Execução.

As Heurísticas são livres para usar os dados do Resultado como melhor lhe aprouverem, desde que a saída seja uma Configuração que ainda não tenha sido usada nos testes anteriores.

4.2.2 Selecionar Nova Carga de Trabalho

De maneira similar à escolha de uma nova Configuração, o controlador pode usar a Heurística para selecionar uma nova Carga de Trabalho, de acordo com o resultado com a última Execução.

As Heurísticas devem, então, prover operações que permitam a navegação pela faixa de valores de Cargas de Trabalho estudada para a Aplicação sob Teste. Portanto, uma Heurística compatível deve fornecer duas operações de controle do nível de Carga de Trabalho: uma operação para que seja reduzido e outra operação para que seja elevado o nível de Carga de Trabalho.

Aqui também as Heurísticas são livres para criarem suas próprias lógicas de avaliação dos dados do Resultado e, a partir daí, definirem qual o tamanho do passo no caminhamento sobre a faixa de Cargas de Trabalho.

5 Esquema de Solução

***** DESCONSIDERAR POR ENQUANTO *****

O objetivo deste trabalho é estudar heurísticas de preenchimento da matriz P descrita no capítulo anterior sem necessariamente ter que executar de fato todos os testes necessários. Esse preenchimento deverá ser feito, assim, por meio de um motor de predições que executará uma ou mais estratégias para preencher a matriz P com resultados calculados a partir dos dados de entrada.

Consideraremos como dados de entrada os resultados de uma ou mais execuções reais para um ou mais cenários de teste. A precisão dos resultados obtidos pelo motor de predição vai depender da qualidade da estratégia escolhida e da quantidade de dados de entrada. Quanto mais diversificadas quanto aos cenários a que forem aplicadas e quanto maior o número de execuções reais usadas para alimentar inicialmente o motor de predições, mais preciso tenderá a ser o resultado da predição, porém mais caro se tornará o processo, uma vez que os testes executados no ambiente de nuvem incorrem em custo financeiro.

Apresentaremos um motor capaz de executar heurísticas de predição e um arcabouço de implementação dessas heurísticas, de forma que nova inteligência de predição possa ser agregada ao trabalho futuramente. Apresentaremos também, como forma de validar a proposta do arcabouço e do motor de execuções, duas heurísticas para geração da matriz P preenchida e sugestão da configuração de menor custo para executar a aplicação alvo em ambiente de nuvem de infraestrutura.