LATIN.SCIENCE XVI Congresso Latino-americano de Software Livre e Tecnologias Abertas LATINOWARE 2019

A Recommender for Resource Allocation in Compute Clouds Using Genetic Algorithms and SVR

Thiago Nelson Faria dos Reis1, Mário Antonio Meireles Teixeira1,
João Dallyson Sousa de Almeida1, Anselmo Cardoso de Paiva1
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal do Maranhão - UFMA
Av. dos Portugueses, 1966 - Campus do Bacanga - CEP 65080-805 – São Luís - MA
thiagonelson@gmail.com, mario@deinf.ufma.br, jdallyson@gmail.com, paiva@deinf.ufma.br

***Resumo*—Resource allocation in Cloud Computing has been done reactively, hindering service guarantees and generating unnecessary charging of idle resources. In order to mitigate these problems, this work proposes and evaluates a predictive resource allocation approach, implemented as a Configuration Recommender, based on Support Vector Regression (SVR) and Genetic Algorithms (GA). This combination is used to estimate application runtime and recommends a viable and valid configuration of resources in the cloud, regarding execution time and monetary costs. As a case study, machine learning applications based on the Weka tool are chosen. The results show that predicted times were very close to actual ones, achieving an efficient estimation of time and cost and their consequent reduction.**

***Keywords*-Computer networks, Platform virtualization, Web services, Genetic algorithms, Predictive models.**

I. Introdução

A redução de custos é um desafio constante, tanto para os provedores quanto para os usuários da computação em nuvem. Uma questão a ser respondida é: como determinar quais recursos serão necessários para executar uma tarefa, e por quanto tempo? De acordo com [1], a maioria dos serviços de análise é reativa, ou seja, somente após a demanda se mostrar necessária é que os recursos serão provisionados, normalmente em até 5 minutos, e leva-se ainda um certo tempo até que seja identificado que os recursos alocados não mais estão sendo usados (em média, 15 minutos). Considere também o tempo necessário para os recursos estarem efetivamente disponíveis. Durante este tempo de análise e disponibilização dos recursos, existe o risco da qualidade de serviço (QoS) e do nível de serviço (SLA) contratados pelo cliente não serem garantidos, além da cobrança ou alocação desnecessária de recursos.

Já [2] demonstra que analisar as informações de carga de trabalho das máquinas e definir o melhor momento para realizar a migração reduz o tempo e o impacto desse procedimento no ambiente, confirmando a eficácia na mudança da análise reativa para uma análise preditiva ou pró-ativa, a qual permite estimar quais recursos são necessários antes da execução da tarefa, alocando-os previamente. Como ferramentas para análise e estimação do uso de recursos, destacam-se as Redes Neurais, Regressão Linear e Máquina de Vetores de Suporte. Na literatura, pode-se identificar alguns trabalhos que utilizam Aprendizado de Máquina e Inteligência Computacional, com esta finalidade, em ambientes em nuvem.

Em [1], é apresentada uma estratégia de alocação de recursos pró-ativa baseada na predição de demanda de uso de recursos, utilizando *Support Vector Regression* (SVR), juntamente com *Particle Swarm Optimization* (PSO), com o objetivo de melhorar a acurácia da predição. O resultado daquela pesquisa comprovou que a predição foi mais eficiente e precisa que a utilização de métodos tradicionais de SVR e Regressão Linear (LR).

Em [3], demonstra-se a eficácia da utilização de técnicas de predição na migração on-line de máquinas virtuais. O comparativo foi realizado entre métodos estatísticos de probabilidade para definição das páginas de memória que deveriam ser migradas, usando o algoritmo ARIMA (*Autoregressive Integrated Moving Average*). O segundo estudo utilizou aprendizado de máquina baseado em regressão usando o modelo SVR. O modelo ARIMA obteve uma acurácia de 91,74%, enquanto o modelo SVR alcançou 94,61% na predição de páginas de memórias sujas, constatando-se a superioridade com relação ao ARIMA.

O aprendizado de máquina também vem sendo usado em diversas outras áreas. Em [4] é utilizado na predição de recursos de computação em nuvem para aplicações multimídia, com o objetivo de atingir um maior equilíbrio no uso dos recursos computacionais do ambiente em nuvem e para garantir a QoS. Para tanto, foi desenvolvida uma solução de alocação de recursos dinâmicos baseada em aprendizado de máquina de modo que, para cada tarefa, fosse feita uma predição dos recursos de acordo com seu histórico de uso. Os experimentos consideraram a utilização de processador e alcançaram um uso mais eficiente se comparados aos métodos convencionais de alocação.

Em [5], é realizado um estudo usando Redes Neurais (RN), por meio de um algoritmo usando autorregressão linear e RN para prever a carga de rede das aplicações no serviço de

nuvem. O modelo desenvolvido combina predição de carga de rede, modelos estocásticos e alocação de recursos, para minimizar o consumo de energia e manter o desempenho requerido. Outra linha que emprega predição e computação em nuvem trata sobre a diminuição da intervenção humana. Isso pode ser observado em [6], [7] e [8], que utilizam aprendizado de máquina através do modelo SVM (*Support Vector Machine*) para alocação de recursos, configuração de aplicações e geração de estimativas de custos com alta precisão, a fim de garantir a QoS exigida, resultando no uso mais eficaz e eficiente dos recursos e redução dos custos, tanto operacionais quanto de manutenção. Os trabalhos de [1], [3], [4], [5], [6], [7] e [8] demonstram a eficiência do emprego de SVR para estimativa com séries históricas.

A partir dos trabalhos analisados, é apresentado neste artigo um Recomendador de configurações de máquinas virtuais em nuvem, que toma como base dados históricos de aplicações, com emprego de SVR e Algoritmos Genéticos. Utilizando-se de métodos preditivos, o recomendador visa sugerir uma configuração ideal de instâncias de VMs na nuvem, que tem por objetivo reduzir o tempo e custo, predizendo os mesmos antes das execuções. Como estudo de caso, utiliza-se o software Weka [9], sem prejuízo de generalização da solução apresentada para outros cenários. Diferindo assim, dos outros trabalhos que avaliam e realizam ajustes durante a execução da atividade, este modelo têm sua ação antes do início da execução, de forma que o processo possua o melhor contexto possível desde o início.

II. Arquitetura do Recomendador de Configuração da Nuvem

Na Fig. 1, tem-se o diagrama principal do Recomendador, onde o cliente submete o arquivo de *dataset* a ser utilizado pelo Weka à interface com o usuário, a qual pode ser um *Web service* ou uma aplicação disponibilizada como SaaS. As informações de execução são enviadas ao Algoritmo Genético, que faz a análise propriamente dita, gerando as possíveis configurações de VMs na nuvem para execução da tarefa submetida.

O Regressor SVR possui a responsabilidade de predizer o tempo de execução da tarefa levando em consideração a base histórica de execuções, o arquivo de *dataset* e as configurações geradas pelo AG. Esta associação permite ao AG, através da sua função *fitness*, receber o tempo estimado pelo regressor e calcular o custo/tempo da execução, na busca da melhor configuração na nuvem para a aplicação submetida pelo usuário, conforme o objetivo colocado.

Em seguida, já com o resultado da configuração ideal de VMs fornecido pelo AG, o Recomendador (no nível IaaS do provedor de nuvem) instanciará as máquinas virtuais de acordo com a configuração sugerida e submeterá o arquivo de *dataset* para execução no ambiente Weka em nuvem. Ao término da execução da tarefa, o cliente receberá o resultado e a configu-ração utilizada, bem como o tempo e custo reais da execução, a fim de alimentar a base histórica de execuções Weka. Tal informação será utilizada futuramente pelo Regressor, no intuito de melhorar suas predições.

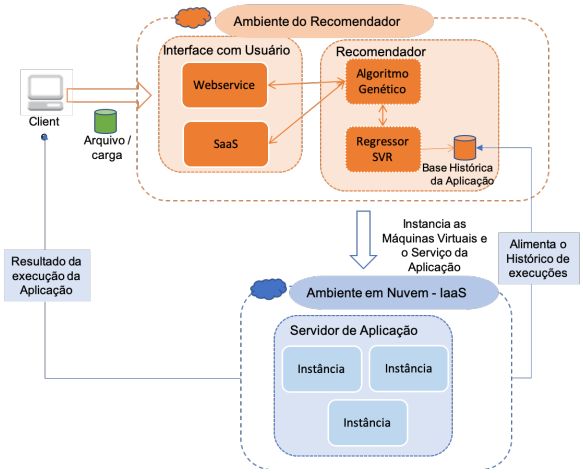

Figura 1. Visão geral da arquitetura do Recomendador.

A. Nuvem Privada

Para implementação do protótipo e geração dos dados históricos de entrada para o Regressor SVR, descritas neste trabalho, foi utilizada a solução de nuvem privada Eucalyptus [10], juntamente com a ferramenta Weka [9] com o módulo Weka Server habilitado [11], a fim de permitir a execução das tarefas de forma distribuída, maximizando o aproveitamento dos recursos em nuvem.

Dados os recursos disponíveis na nuvem Eucalyptus, foi possível alocar até 16 núcleos de processamento e 16 GB de RAM por máquina virtual. A elasticidade empregada para os recursos em nuvem foi do tipo horizontal, ou seja, se necessário serão adicionadas novas instâncias do mesmo tipo lado a lado. As instâncias são definidas com o tamanho de memória e disco fixos, havendo somente variação na quantidade de processadores, dado que as aplicações Weka são mais dependentes de processamento do que de memória ou disco.

Embora se tenha optado por uma infraestrutura de nuvem privada, esta pode ser estendida sem muito esforço para um modelo híbrido, de preferência lançando-se mão do provedor de nuvem *Amazon Web Services* (AWS), uma vez que a nuvem Eucalyptus integra-se de forma transparente com a AWS, podendo intercambiar até mesmo imagens de VMs (*AMI – Amazon Machine Images*). Assim, os recursos colocados à disposição da solução aqui apresentada tornam-se potencialmente ilimitados.

Considerando as características de execução do *Weka Server*, conforme [9], optou-se por utilizar modelos preditivos para o escalonamento das aplicações, em particular análise de séries temporais [12]. Desta forma, é possível realizar a análise e estimação dos custos antes de se iniciar a execução

da aplicação, fazendo uma alocação prévia dos recursos.

B. Máquinas Virtuais (Instâncias)

Para fins de execução do software Weka no ambiente em nuvem e consequente criação dos dados históricos, foram criadas várias instâncias (de duas a cinco) a fim permitir execuções em paralelo. Todas as instâncias possuíam a mesma configuração, diferindo apenas na quantidade de núcleos de processamento. A capacidade de memória das instâncias foi suficiente para execução das tarefas submetidas ao Weka, não sendo necessária a utilização de paginação de memória em nenhum momento.

Foi feita uma estimativa do custo de se manter uma infraestrutura similar em uma nuvem pública. Para tanto, utilizou-se como referência os valores praticados pela Amazon AWS [13], conforme mostrado na Tabela I (em dólares americanos).

Tabela I
CUSTOS DE INSTÂNCIAS NA AMAZON AWS.

Instância	Qtd Núcleos	Estados Unidos	
		Custo Hora	Custo Seg
t2.small	1	0,023	0,000006389
t2.medium	2	0,047	0,000013056
t2.xlarge	4	0,188	0,000052222
m4.2xlarge	8	0,4	0,000111111

C. Weka

A execução do software Weka de forma distribuída exige que um dos nós realize a função de controlador, sendo responsável por receber as requisições e distribuí-las para os demais. O controlador também executa as tarefas, assim como os escravos. Nos experimentos, foram utilizados três arquivos de dados distintos, obtidos a partir da mesma fonte de dados, os quais continham informações para detecção de massas em imagens mamográficas, cedidas por [14], porém com complexidades distintas na quantidade de atributos e na quantidade de instâncias, a fim de testar diferentes cargas de dados. Desta forma, os arquivos foram divididos em baixa, média e alta complexidade, sendo obtidos de forma aleatória, a partir da mesma origem.

A infraestrutura de nuvem privada implementada na plataforma Eucalyptus limitou o uso de cinco máquinas virtuais para os experimentos.

As execuções foram realizadas empregando-se 14 configurações diferentes na nuvem, detalhadas na Tabela II. Foram utilizadas entre 2 a 5 máquinas virtuais e um total de 4, 5, 7 e 10 núcleos disponíveis no ambiente, distribuídos nas instâncias. O Servidor e os Nós na tabela correspondem às VMs criadas e cada configuração (as linhas) indica a distribuição dos núcleos de processamento entre as VMs. Em todas as execuções, foram capturadas as informações de consumo de memória, processamento e tempo de execução.

Tabela II
CONFIGURAÇÕES UTILIZADAS NA NUVEM EUCALYPTUS.

	Qtd de Núcleos				
	Servidor	**Nó 1**	**Nó 2**	**Nó 3**	**Nó 4**
Config 1	2	4	4		
Config 2	2	2	2	4	
Config 3	2	8			
Config 4	2	2	2	2	2
Config 5	1	2	2	2	
Config 6	1	2	4		
Config 7	1	1	1	2	2
Config 8	1	2	2		
Config 9	1	4			
Config 10	1	1	1	1	1
Config 11	1	1	1	2	
Config 12	1	1	1	1	
Config 13	1	1	2		
Config 14	2	2			

A partir dos experimentos realizados, percebeu-se que o uso de memória está diretamente relacionado à carga de dados, não variando de uma execução para outra, da mesma origem. Sofre influência, somente, da complexidade e quantidade dos dados analisados. Uma particularidade observada na ferramenta Weka é que cada tarefa utiliza somente um núcleo de processamento quando em execução, podendo chegar até 100% de utilização deste. O Weka, portanto, não conseguiu dividir uma tarefa entre vários núcleos, o que levou este trabalho a considerar apenas o tempo de execução (em um único núcleo) como variável de estimativa para o regressor.

D. Regressor SVR

A partir dos dados gerados pelas execuções do Weka foram obtidas as séries históricas, ou seja, valores que são utilizados pelo regressor para realizar a predição do tempo de execução para uma determinada configuração, dado um arquivo específico.

O Regressor foi implementado utilizando-se a biblioteca *Sklearn* [15], voltada a aprendizado de máquina na linguagem Python, de forma nativa. No regressor, empregou-se o método *GridSearchCV*, que realiza pesquisa exaustiva sobre valores de parâmetros especificados para um estimador, implementando um método de ajuste (fit) e pontuação (score), que permite ir ajustando valores dos parâmetros e classificando o resultado da predição através de validação cruzada. Com este método, é possível testar vários parâmetros para o estimador de acordo com a base de dados submetida.

Esta etapa consistiu em validar e melhorar a acurácia do regressor, aumentando os intervalos dos parâmetros, chegando a uma combinação de 20 milhões de possibilidades na última configuração. Na Tabela III, têm-se as cinco configurações executadas do GridSearchCV com os parâmetros obtidos e os

resultados de acurácia calculados. Para todas as configurações, foram utilizados treinamento (70%), teste (30%) e validação cruzada.

Observa-se que desde a primeira validação, SVR 1, o regressor obteve uma acurácia de 94,35% na predição, confirmando a viabilidade do modelo e na última execução, SVR 5, obteve uma acurácia ainda melhor, de 95,09%, como consequência do maior intervalo de combinações possíveis e maior tempo dispendido no cálculo dos parâmetros.

Tabela III
RESULTADO DAS CONFIGURAÇÕES DO SVR.

Config	epsilon	gamma	cost	Variação Máxima %	Acurácia %
SVR 1	1,0	0,25	32768,0	5,65	94,35
SVR 2	0	0,125	16384,0	6,06	93,94
SVR 3	1,0	0,5	8192,0	5,81	94,19
SVR 4	1,0	1,0	32768,0	5,84	94,16
SVR 5	1,0	0,03125	8388608,0	4,91	95,09

E. Algoritmo Genético

Concluída a etapa de treinamento do regressor, passou-se para a construção do algoritmo genético, cujo objetivo é gerar as possíveis configurações da nuvem, para então encontrar as melhores soluções [16]. Os critérios são especificados na Função de Avaliação (*fitness*), que define se as soluções são válidas e o quanto uma solução é mais viável ou eficaz que outra, de acordo com [17]. Para tanto, foi criada uma função que avalia o menor tempo de execução e, em seguida, considera o menor custo ou, ainda, o menor tempo de execução até um limite de custo, conforme representado na Fig. 2.

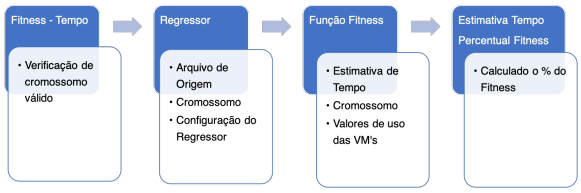

Figura 2. *fitness* segundo o tempo de execução.

Observe que, sempre que a função *fitness* é chamada, o regressor também é convocado para fazer a estimativa do tempo, utilizando-se como parâmetro de entrada o arquivo de origem dos dados Weka e o cromossomo que possui a configuração da nuvem.

O cromossomo adotado possui dois tipos de genes, uma vez que para a execução do Weka em ambiente em nuvem é necessário um Servidor (que, além de gerenciar as tarefas, também processará as requisições), além dos Escravos, representados por N1, N2, N4 e N8, que irão somente processar as requisições. Sendo assim, o cromossomo é dividido em duas partes, conforme visualizado na Fig. 3, onde são representados alguns indivíduos da população, sendo a primeira coluna correspondente à identificação do gene:

Indiv.	Servidor	Escravos			
		N1	N2	N4	N8
1	2	0	4	0	0
2	2	0	2	1	0
3	1	0	0	0	1
4	2	0	0	0	1
5	2	1	1	0	0

Figura 3. Exemplos de indivíduos da população.

O custo de uso das máquinas é calculado levando em consideração o tempo de execução e as máquinas usadas:

Custo = Tempo * (N1 * V1 + N2 * V2 + N4 * V4 + N8 * V8)

onde V1, V2, V4 e V8 são os custos por segundo de uso das máquinas e N1, N2, N4 e N8, a quantidade de máquinas usadas com 1, 2, 4 e 8 núcleos respectivamente, conforme representado na Fig. 4.

Indiv.	Servidor	Escravos				Tempo Estimado (s)	Custo Estimado (US$)
		N1	N2	N4	N8		
1	2	0	4	0	0	1122	0,0879
2	2	0	2	1	0	1468	0,1533
3	1	0	0	0	1	1472	0,1631
4	2	0	0	0	1	1191	0,1555
5	2	1	1	0	0	1299	0,0592

Figura 4. Estimativas de tempo e custo de indivíduos da população.

As mutações e cruzamentos podem ocorrer nas duas partes do cromossomo, ou seja, na região que representa o servidor e na dos escravos, exclusivamente. Não são permitidas essas operações entre servidores e escravos.

Foram realizadas execuções do AG com taxas de cruzamento segundo as recomendações de [18], que sugere entre 0,6 e 0,99, Como o algoritmo proposto visa explorar a maior quantidade de possíveis soluções, adotou-se uma taxa de 0,8.

Já para as taxas de mutação e tamanho da população, foram realizadas execuções do AG com taxas de 0,1%, 1% e 10%, com tamanhos variando entre 25 e 150, com incremento de 25 indivíduos. Para cada combinação dos parâmetros acima, foram realizadas 200 execuções e obtidos os valores de resultado de configuração e tempo para a melhor solução encontrada, para o *fitness* Tempo.

Na Fig. 5, tem-se o diagrama principal do processo aqui utilizado, onde submete-se ao AG tanto o arquivo a ser usado

pelo Weka, representado por X(i), quanto os parâmetros de configuração do AG, denotados por N(i), além do tipo de *fitness*, tamanho da população e taxa de mutação. Os indivíduos da população inicial são gerados aleatoriamente e os cromossomos são, então, validados. Caso seja gerado um cromossomo inválido, é gerado um novo. Somente os cromossomos válidos são submetidos ao regressor (SVR), que recebe como parâmetro o cromossomo c(i) e os dados do arquivo Weka, X(i). O regressor tem como resultado o Tempo Estimado, t(i), que retorna ao AG para o cálculo do Custo, C(i), juntamente com a configuração c(i). Todos os cromossomos gerados, que sofrerão cruzamento de um ponto e/ou mutação, são submetidos novamente ao regressor para a estimativa de tempo, chamado na função *fitness* do AG.

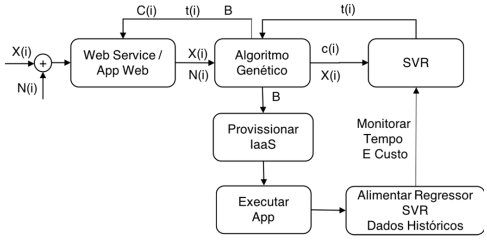

Figura 5. Diagrama do Processo Proposto.

Para a seleção dos cromossomos, utilizou-se o critério de Roleta Simples associado a Seleção Elitista, sendo assim, após as gerações criadas são definidos os melhores indivíduos de cada geração e o AG, baseado na função *fitness*, seleciona o melhor indivíduo de todas, retornando como B a configuração recomendada, o tempo estimado e o custo estimado para o Web Service/Aplicação Web e para o módulo de IaaS da nuvem.

Como forma de explorar o máximo possível de soluções, justamente por estar usando também a seleção elitista, adotou-se como critério de parada a centésima geração criada, com o objetivo de analisar e identificar o comportamento dos indivíduos e também quando ocorre a convergência das soluções, sendo que o mesmo foi configurado para ser executado cinco vezes nos experimentos, para cada validação.

Com a configuração definida, o serviço de nuvem provisionará, então, as máquinas e serviços necessários e a aplicação Weka será, por fim, executada. Ao término da execução da aplicação, as máquinas instanciadas podem ser desligadas, liberando os recursos, a fim de minimizar os custos de utilização da nuvem, dado que os recursos somente são provisionados e tarifados durante a execução da aplicação.

No processo principal, a execução da aplicação Weka é monitorada e comparada com os dados previstos, gerando-se estatísticas. Ademais, os dados (históricos) das execuções são utilizados para alimentar o regressor SVR, de modo que a predição possa ter sua acurácia aumentada com o passar do tempo.

III. VALIDAÇÃO DA ARQUITETURA

Conforme discutido na seção II-C, os dados da série histórica foram gerados com base em três modelos de arquivos de dados e catorze configurações diferentes do ambiente em nuvem, sendo que todas elas foram executadas em uma nuvem privada utilizando a solução Eucalyptus. Os dados capturados foram utilizados para o treinamento, teste e validação do regressor SVR (seção II-D), realizados também no ambiente em nuvem.

Todas as instâncias criadas possuem os mesmos softwares e versões de bibliotecas, além disso suas configurações e custos foram padronizados com relação aos da Amazon AWS, conforme descrito na seção II-A.

A. Regressor SVR

A Tabela IV apresenta os resultados dos testes, definidos na seção II-D, realizados com o regressor SVR para a predição do tempo em segundos, demonstrando sua viabilidade, com erro máximo de 5,65%, chegando a uma acurácia de 94,35%. A coluna *Tempo Real(s)* contém os valores obtidos na execução da aplicação Weka em nuvem, através da captura do tempo de execução. A coluna *Tempo Estimado(s)* contém os tempos previstos pelo regressor, enquanto as colunas *Diferença(s)* e *var %* são o erro em segundos e percentual, correspondentes à subtração do tempo estimado pelo tempo real de execução.

Tabela IV
RESULTADOS INICIAIS DO SVR.

#	Tempo Real(s)	Tempo Estimado(s)	Diferença(s)	%	var %
1	455	480,6976	25,6976	105,65%	5,65%
2	4025	4007,1128	17,8872	99,56%	-0,44%
3	3175	3119,9928	55,0072	98,27%	-1,73%
4	870	890,2641	20,2641	102,33%	2,33%
5	2974	3025,8053	51,8053	101,74%	1,74%
6	2143	2143,6356	0,6356	100,03%	0,03%
7	3677	3680,2360	3,2360	100,09%	0,09%
8	2527	2521,8074	5,1926	99,79%	-0,21%
9	2042	2143,6356	101,6356	104,98%	4,98%
10	854	862,5274	8,5274	101,00%	1,00%
11	459	480,6976	21,6976	104,73%	4,73%
12	448	449,7849	1,7849	100,40%	0,40%
13	1197	1190,1406	6,8594	99,43%	-0,57%
14	2242	2201,2420	40,7580	98,18%	-1,82%
15	4065	4041,3855	23,6145	99,42%	-0,58%
16	3337	3308,8268	28,1732	99,16%	-0,84%
17	2231	2249,8334	18,8334	100,84%	0,84%
18	1584	1575,9550	8,0450	99,49%	-0,51%
19	1466	1526,5028	60,5028	104,13%	4,13%
20	881	846,9477	34,0523	96,13%	-3,87%

B. Algoritmo Genético

Os desafios com relação ao Algoritmo Genético foram de encontrar os parâmetros viáveis e validar a sua eficácia na resolução do problema. Para tanto, realizou-se 200 execuções para cada combinação dos parâmetros, definidos na Seção II-E, totalizando cerca de 18.000 monitoramentos. Após a análise dos dados capturados, observou-se uma melhor convergência dos valores das funções de avaliação para as taxas de mutação de 1% e para os tamanhos de população de 125 e 150 indivíduos. Desta forma, a população de 125 indivíduos tornou-se a mais eficaz, pois produz valores equivalentes à de 150, com a vantagem de ter um custo computacional menor. Por conseguinte, adotou-se uma população de 125 indivíduos, 100 gerações, taxa de mutação de 0,01 e cruzamento de 0,8.

C. Recomendador de Configurações

O Recomendador de Configurações (AG + SVR) produz como saída uma determinada configuração da nuvem, definindo a quantidade de núcleos para o servidor e a quantidade de máquinas escravas (com o número de núcleos de cada uma delas). Tal configuração visa permitir a execução da aplicação Weka no menor tempo possível e com o menor custo.

Foram realizados testes comparativos da configuração recomendada com o tempo previsto de execução e a execução da aplicação em ambiente de produção, em 10 recomendações aleatórias. Foram instanciadas as máquinas virtuais na infraestrutura de nuvem privada de acordo com as recomendações sugeridas pelo modelo e executada a aplicação. Os resultados são apresentados na Tabela V e visualizados na Fig. 6 , tendo sido obtido um erro máximo de 5,29% e erro médio de 3,07%, o que demonstra a eficácia do processo proposto neste trabalho.

Tais diferenças entre o tempo real e estimado, e os respectivos percentuais de erro são devidas as variações presentes em qualquer ambiente, como tráfego na rede, link do cliente com o provedor de nuvem, rota de acesso, demanda no provedor de IaaS, dentre outros fatores, de modo que esta variação foi considerada aceitável para o modelo proposto. Os custos não estão representados na tabela, pois apresentam a mesma variação percentual para essas execuções, uma vez que o cálculo é diretamente proporcional à configuração e ao tempo de execução.

Tabela V
RESULTADOS DE VALIDAÇÃO DOS TEMPOS DE EXECUÇÃO

Rodada	Tempo Real (s)	Tempo Previsto (s)	Real / Previsto (%)	Erro(%)
1	1118	1108	99,11%	0,89%
2	1054	1016	96,39%	3,61%
3	1107	1062	95,93%	4,07%
4	1111	1067	96,04%	3,96%
5	435	412	94,71%	5,29%
6	400	412	103,00%	-3,00%
7	415	412	99,28%	0,72%
8	407	412	101,23%	-1,23%
9	1383	1330	96,17%	3,83%
10	1386	1314	94,81%	5,19%
Média	**881,6**	**854,5**	**96,93%**	**3,07%**

Os gráficos da Fig. 7 apresentam as superfícies de execução dos tempos reais (a) e as calculadas pelo Recomendador (b). Verificou-se que as duas são qualitativamente similares, sendo que o Recomendador possui a vantagem de permitir simulações de comportamentos de acordo com a necessidade do usuário, de forma rápida e confiável, sem a necessidade de alocar recursos na nuvem.

Finalmente, executou-se novamente as mesmas aplicações Weka iniciais, desta vez conforme a configuração em nuvem sugerida pelo Recomendador, foi possível alcançar uma redução de tempo de 38,8% e de 45,62% no custo da execução. A configuração de referência inicial possuía um servidor de 2 núcleos e 2 máquinas de 4 núcleos (Tabela II), enquanto a configuração vencedora, sugerida pelo Recomendador era composta por um servidor com 1 núcleo, 3 máquinas de 2 núcleos e 1 máquina de 4 núcleos.

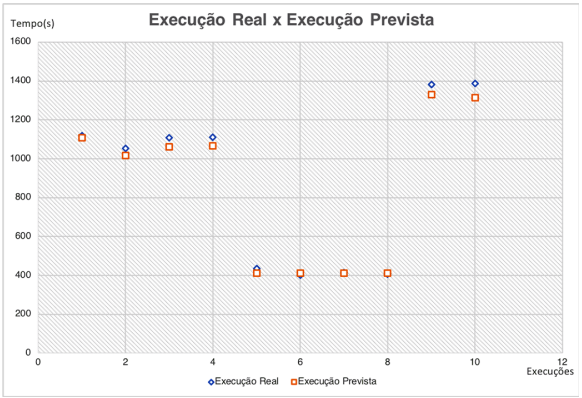

Figura 6. Comparativo de Execução Real x Prevista.

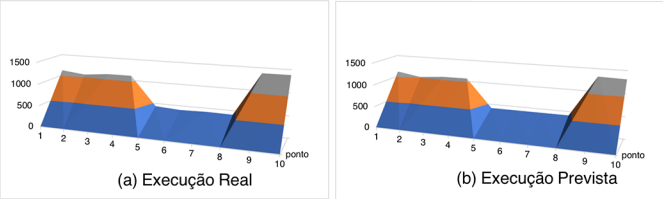

Figura 7. Superfície de resposta.

IV. SERVIÇO DE RECOMENDAÇÃO

Como o Recomendador foi desenvolvido em linguagem Python, de forma modular e orientada a objetos, sua integração com outras soluções e ferramentas é facilitada, podendo ser implementado diretamente na camada de IaaS e/ou disponibilizado como uma aplicação do tipo SaaS, de acordo com [19] [12].

A. *Módulo de Administração da Nuvem*

Um dos objetivos da Computação em Nuvem é provisionar recursos de forma automatizada. Assim, o Recomendador pode facilmente se integrar à nuvem, principalmente se o provedor suportar as APIs da AWS, utilizadas pela Amazon e Eucalyptus, sendo possível, desta forma, utilizá-lo em nuvens privadas, públicas ou híbridas [20] [21]. Esta integração está representada pela funcionalidade *Provisionar IaaS* da Fig. 5 (seção II-E), que permite automatizar a criação das máquinas virtuais e a consequente execução da aplicação Weka, sem intervenção humana.

É importante salientar que o Recomendador aqui descrito é independente dessa integração com o provedor de nuvem, porém é altamente sugerido que esta seja realizada, a fim de melhor aproveitar os recursos computacionais a serem provi-sionados.

B. *Recomendação de Configurações como um Serviço (SaaS)*

Neste trabalho, também foi implementada uma interface de comunicação com o Recomendador por meio de um Serviço Web do tipo REST, possibilitando, assim, sua ampla utilização pelas mais diversas aplicações, que podem consumir o serviço de forma simples e eficiente. O serviço possui três parâmetros obrigatórios: a quantidade de linhas do arquivo de origem de dados, a quantidade de colunas e o limite da função *fitness*, que são passados na URI.

O retorno do serviço web é mostrado a seguir, em formato JSON, onde são fornecidas informações da execução, tais como: tipo de regressor, tipo de *fitness* usado, a configuração de núcleos recomendada para o Servidor e Escravos, e a estimativa de Tempo e Custo para a chamada *http://recomendador.saas.api/Tempo/5620/65/0* :

{
linhas: 5620,
colunas: 65,
tipo_regressor: N3,
tipo_*fitness*: Tempo,
limite: 0,
Populacao: 125,
Mutacao: 1,
Tempo: 49.81017331752264,
Custo: 0.010722984681276321,
fitness: 0.010722984681276321,
*fitness*Percento: 0.038373161309446215,
Servidor: 2,
Host_1N: 1,
Host_2N: 0,
Host_4N: 1,
Host_8N: 1
}

V. CONCLUSÃO

Provisionar recursos computacionais de forma eficaz é uma tarefa importante, porém difícil, principalmente devido ao fato de que o uso desses recursos tem natureza e características muito distintas, e comportamentos muitas vezes imprevisíveis, tanto para os provedores de infraestrutura quanto para os usuários. Por outro lado, o uso de dados históricos de utilização dos mesmos permite mapear seu uso e o comportamento das execuções.

Este trabalho descreve um protótipo de um Recomendador de Configurações em Nuvem utilizando um Regressor SVR, que alimenta um Algoritmo Genético, de modo a permitir a estimativa do tempo e custo de execução de uma aplicação Weka, bem como a sugestão de uma configuração viável de máquinas virtuais na nuvem.

O experimento de melhor resultado mostra que uma base de conhecimento ampla e bem treinada consegue predizer o tempo de execução das aplicações com precisão, de forma rápida e eficaz, com erros próximos de 5% em comparação com o ambiente real. Além disso, é possível alcançar, em alguns casos, uma economia de até 38,8% e 45,62% no tempo e custo de execução, respectivamente, em relação a uma seleção aleatória de configuração na nuvem.

Como trabalhos futuros, sugere-se que a arquitetura proposta seja generalizada para outros tipos de aplicações e avaliada, onde neste trabalho foi utilizado o software Weka como estudo de caso, sendo necessário ter acesso a uma base histórica de utilização e a realização do treinamento do regressor. Além disso, o Recomendador pode ser implementado como um serviço da própria nuvem, ficando à disposição dos usuários.

REFERÊNCIAS

[1] Z. Zhu, J. Peng, Z. Zhou, X. Zhang, and Z. Huang, "Pso-svr-based resource demand prediction in cloud computing," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 20, no. 2, pp. 324–331, 2016.

[2] A. Baruchi, E. T. Midorikawa, and L. M. Sato, "Reducing virtual machine live migration overhead via workload analysis," *IEEE Latin America Transactions*, vol. 13, no. 4, pp. 1178–1186, 2015.

[3] M. Patel, S. Chaudhary, and S. Garg, "Machine learning based statistical prediction model for improving performance of live virtual machine migration," *Journal of Engineering*, vol. 2016, 2016.

[4] K. Sembiring and A. Beyer, "Dynamic resource allocation for cloud-based media processing," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2013, pp. 49–54.

[5] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *System of Systems Engineering (SoSE), 2011 6th International Conference on*. IEEE, 2011, pp. 276–281.

[6] O. Niehorster, A. Krieger, J. Simon, and A. Brinkmann, "Autonomic resource management with support vector machines," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*. IEEE Computer Society, 2011, pp. 157–164.

[7] A. A. Bankole and S. A. Ajila, "Predicting cloud resource provisioning using machine learning techniques," in *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on*. IEEE, 2013, pp. 1–4.

[8] E. Hormozi, H. Hormozi, M. K. Akbari, and M. S. Javan, "Using of machine learning into cloud environment (a survey): managing and scheduling of resources in cloud systems," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on*. IEEE, 2012, pp. 363–368.

[9] Weka. (2017) The university of waikato. weka 3: Data mining software in java. Disponível em: http://www.cs.waikato.ac.nz/ml/weka. Acesso em: Janeiro 2017.

[10] HP. (2016) Hp eucalyptus. Disponível em: http://hphelion.com. Acesso em: Junho 2016.

[11] Pentaho. (2017) Pentaho weka server datamining. Disponível em: http://wiki.pentaho.com/display/DATAMINING/Weka+Server. Acesso em: Janeiro 2017.

[12] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10723-014-9314-7

[13] Amazon. (2016) Amazon web service. Disponível em: http://amazon.com. Acesso em: Junho 2016.

[14] F. Silva, João; Vaz, "Modelo de implantação de nuvem privada em infraestruturas de computação em nuvem baseadas no sistema eucalyptus," *ERBASE 2013*, 2013.

[15] sklearn. (2017) scikit-learn - machine learning in python. Disponível em: http://scikit-learn.org/stable/. Acesso em: Setembro 2017.

[16] C. Sheppard, *Genetic Algorithms with Python*, 1st ed. CreateSpace Independent Publishing Platform, 2016.

[17] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2011.

[18] S. O. Rezende, *Sistemas Inteligentes: Fundamentos e Aplicações*. Manole, 2003.

[19] P. J. Jonathan Kupferman, Jeff Silverman, "Scaling into the cloud," *ADVANCED OPERATING SYSTEMS*, pp. 1–8, 2016.

[20] C. R. Cunha, E. P. Morais, J. P. Sousa, and J. P. Gomes, "The role of cloud computing in the development of information systems for smes," *IBIMA Publishing*, pp. 1–7, 2017. [Online]. Available: http://hdl.handle.net/10198/14061

[21] S. M. Parikh, "A survey on cloud computing resource allocation techniques," in *Engineering (NUiCONE), 2013 Nirma University International Conference on*. IEEE, 2013, pp. 1–5.