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

T. Reis, M. Teixeira, J. Almeida, and A. Paiva

***Abstract*—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.**

***Index Terms*—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). Considera-se também o tempo necessário para os recursos estarem efetivamente disponíveis. Durante este tempo de análise e disponibilização dos recursos, existe o risco da qualidade do serviço (QoS) e do nível de serviço (SLA) contratado pelo cliente não serem garantidos, além da cobrança ou alocação desnecessária de recursos.

Em [2], demonstra-se uma abordagem mais eficiente de migração entre máquinas virtuais, mas ainda assim a abordagem reativa apresenta um custo computacional maior.

T. N. F. Reis is with the Universidade Federal do Maranhão, São Luís, Maranhão, Brazil (e-mail: thiagonelson@gmail.com).

M. M. Teixeira is with the Universidade Federal do Maranhão, São Luís, Maranhão, Brazil (e-mail: mario@deinf.ufma.br).

J. D. S. Almeida is with the Universidade Federal do Maranhão, São Luís, Maranhão, Brazil (e-mail: jdallyson@gmail.com).

A. C. Paiva is with the Universidade Federal do Maranhão, São Luís, Maranhão, Brazil (e-mail: paiva@deinf.ufma.br).

Uma forma de resolver este problema é passar dessa abordagem de análise reativa para uma análise *preditiva* ou *pro-ativa*, a qual permite estimar quais recursos serão necessários antes da execução da tarefa, alocando-os previamente. Como ferramentas para análise e estimativa do uso de recursos, destacam-se as Redes Neurais, Regressão Linear e Máquina de Vetores de Suporte. Na literatura, pode-se identificar alguns trabalhos com esta finalidade 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 pro-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 do 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 utilizando métodos estatísticos de probabilidade para definição das páginas de memória que deveriam ser migradas, usando o algoritmo *Autoregressive Integrated Moving Average* (ARIMA). 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 a sua 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. Em [23], a predição é utilizada com o objetivo de garantir o SLA contratado. 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 obtiveram um uso mais eficiente se comparados aos métodos convencionais de alocação.

Como em [24] foi demonstrada a necessidade de analisar vários critérios para projetos que utilizam nuvem, também foram empregados outros critérios que podem ser adotados, tais como melhor tempos, menor custo ou melhor relação tempo / custo para a execução destes projetos.

Em [5], foi feito um estudo usando Redes Neurais (RN), por meio de um algoritmo utilizando 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 da diminuição da intervenção humana. Isso pode ser observado em [6], [7] e [8], que utilizam aprendizado de máquina através do modelo SVM 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.

A predição do uso de recursos de memória e de processadores é abordada em [26], [27], [28] e [29] que utilizam técnicas com modelos híbridos de auto regressão e através de Redes Neurais, confirmando a viabilidade da predição e a necessidade da melhor utilização de recursos computacionais e financeiros.

Os trabalhos de [1], [3], [4], [5], [6], [7] e [8] demonstram a eficiência do emprego de SVR para estimativa com séries históricas. Ao passo que o trabalho [25] utiliza Algoritmos Genéticos (AG) para determinar a expansão e otimização de redes de transmissão, cujos objetivos são explorar de forma eficiente soluções possíveis e determinar os melhores resultados.

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, valendo-se do emprego de SVR e Algoritmos Genéticos. Tal Recomendador visa sugerir uma configuração ideal de instâncias de VMs na nuvem, mediante o objetivo colocado, predizendo o tempo e custo das execuções. Como estudo de caso, utiliza-se o software Weka [9], sem prejuízo de generalização da solução apresentada para outros cenários.

Em consequência, o usuário da nuvem será capaz de selecionar de forma mais eficiente os recursos da nuvem que melhor atendam a requisitos pré-definidos de preço e tempo de execução. Isto representa uma inovação em relação à forma tradicional de escolher os recursos da nuvem, com resultados diretos em economia de tempo e dinheiro para o usuário, seja ele da indústria ou da academia.

II. Arquitetura do Recomendador de Configurações 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 tem 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.

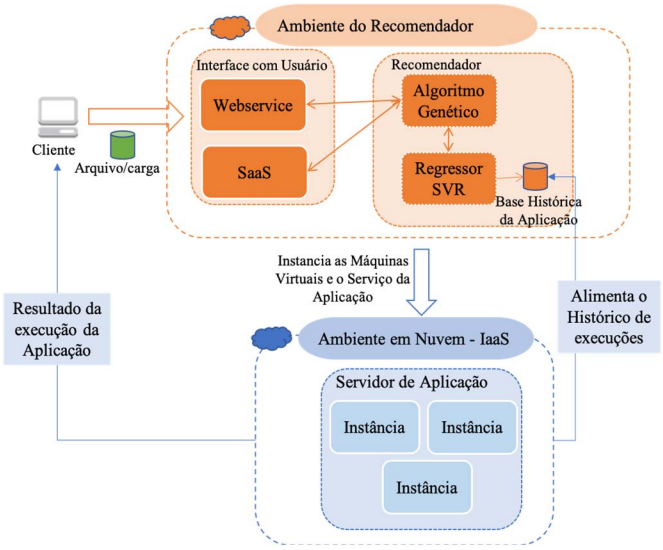

Fig. 1. Visão geral da arquitetura do Recomendador.

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 configuração 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.

A. *Nuvem Privada*

Para implementação do protótipo e geração dos dados de entrada para o Regressor SVR, descritos 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 para a 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 seriam 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 de provedores de nuvem como *Amazon Web Services* (AWS). A nuvem Eucalyptus integra-se de forma transparente com a AWS, podendo intercambiar até mesmo imagens de VMs (*Amazon Machine Images* – AMI). Assim, os recursos colocados à disposição da solução aqui apresentada tornam-se

potencialmente ilimitados. Como consequência da escolha dessa infraestrutura, este modelo permite somente execução em tempo real, não havendo a possibilidade de execução em lote ou agendamento.

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, 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 *AWS* [13], conforme mostrado na Tabela I (em dólares americanos).

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.

TABELA I
CUSTO DE INSTÂNCIAS NA AWS

Tipos de Instâncias	Qtde. Núcleos	Custo/hora (US$)
t2.small	1	0,023
t2.medium	2	0,047
t2.xlarge	4	0,188
m4.2xlarge	8	0,400

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. Tais configurações foram as possíveis de instanciar na infraestrutura computacional que estava disponível para a pesquisa, as quais acreditamos refletir usos comuns de configurações de máquinas virtuais em ambiente de nuvem. O Servidor e os Escravos 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 de tempo de execução.

TABELA II
CONFIGURAÇÕES UTILIZADAS NA NUVEM EUCALYPTUS

Nome	Qtd de Núcleos				
	Servidor	Escravo 1	Escravo 2	Escravo 3	Escravo 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, por parte 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 podem ser 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 o aprendizado de máquina na linguagem Python, de forma nativa. No Regressor, empregou-se o método *GridSearchCV* [22], 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 atingiu uma acurácia de 94,35% na predição, confirmando a viabilidade do modelo e na última execução, SVR 5, alcançou uma acurácia ainda melhor, de 95,09%, como consequência do maior intervalo de combinações possíveis e maior tempo despendido no cálculo dos parâmetros.

TABELA III
RESULTADO DAS CONFIGURAÇÕES DO SVR

Config.	epsilon	gamma	cost	Variação Máx. %	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, conforme [17]. Para tanto, foi criada uma função que avalia o menor tempo de execução e, em seguida, foi considerado o menor custo ou, ainda, o menor tempo de execução até um limite de custo, conforme representado na Fig. 2. Aqui, as configurações do Regressor são qual o tipo de *Fitness* (tempo, custo e tempo/custo) e/ou o valor mínimo ou máximo para estes, sendo que estes não são obrigatórios.

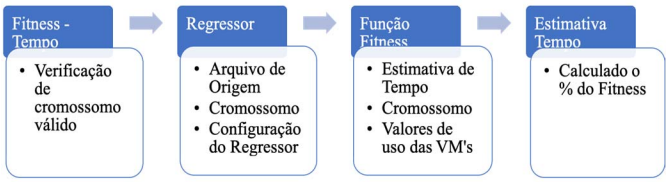

Fig. 2. Fitness segundo o tempo de execução.

Observe-se 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, na qual são representados alguns indivíduos da população, sendo a primeira coluna correspondente à identificação do gene.

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			
		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

Fig. 3. Exemplos de indivíduos da população.

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

Fig. 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], sugeridas 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 com relação ao *Fitness* Tempo.

F. Fluxo de Execução do Recomendador

A execução desse modelo pode ocorrer tanto em nuvem privada quanto híbrida ou pública. Na Fig. 5, tem-se o diagrama principal do processo aqui utilizado, onde se submete 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), que são o tipo de *Fitness*, tamanho da população e taxa de mutação, sendo estes dois últimos pré-estabelecidos. 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.

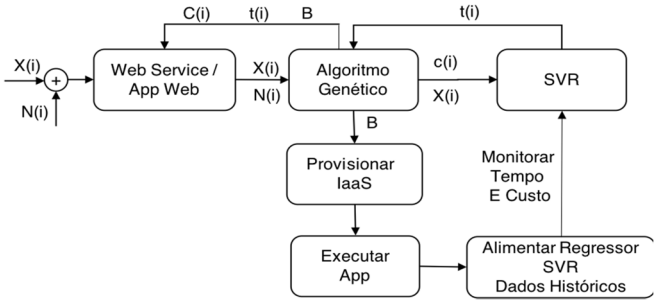

Fig. 5. Diagrama do processo proposto.

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.

O AppWeb retorna ao usuário as estimativas, e envia ao IaaS a configuração recomendada para provisionamento dos recursos e máquinas virtuais, e subsequente execução da aplicação no ambiente.

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, existe a possibilidade de que os dados (históricos) das execuções sejam utilizados para alimentar o Regressor SVR, de modo que a predição possa ter sua acurácia aumentada e as suas aplicações ampliadas, embora esse recurso não tenha sido utilizado para estes experimentos.

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 usados 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 software e versões de bibliotecas, além disso suas configurações e custos foram padronizados com relação aos da AWS, conforme descrito na seção II-A.

A. Regressor SVR

A Tabela IV apresenta os resultados das execuções com os dados de testes, definidos na seção II-D, realizados com o regressor SVR para a predição do tempo em segundos, ficando demonstrada sua viabilidade, com erro máximo de 5,65% e 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. Vale lembrar que para cada caso de teste, executou-se 10 vezes, calculando a diferença entre os tempos e a seguir os valores médios para cada configuração.

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	3680,236	3,236	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	2201,242	40,758	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	1575,955	8,045	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 são encontrar os parâmetros viáveis e validar a sua eficácia na resolução do problema. Para tanto, foram realizadas 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 torna-se mais eficaz, pois produz valores equivalentes à de 150 indivíduos, 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. O ambiente de produção foi configurado na plataforma HPE Eucalyptus, definido na seção II-A, onde todas as máquinas virtuais possuíam o mesmo sistema operacional CentOS e bibliotecas. Foram instanciadas as máquinas virtuais na infraestrutura de nuvem privada, de acordo com as recomendações sugeridas pelo modelo e executada nas mesmas a aplicação Weka. Cada linha da Tabela V representa a média do resultado de 10 execuções de cada uma das configurações recomendadas, com um intervalo de confiança de 95%, 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 às 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 V, 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%

Os gráficos da Fig. 7 apresentam as superfícies de execução dos tempos reais (a) e as calculadas pelo Recomendador (b). Verifica-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, executando-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%, além 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 com melhor desempenho, sugerida pelo Recomendador era composta por um servidor com 1 núcleo, 3 máquinas de 2 núcleos e 1 máquina de 4 núcleos.

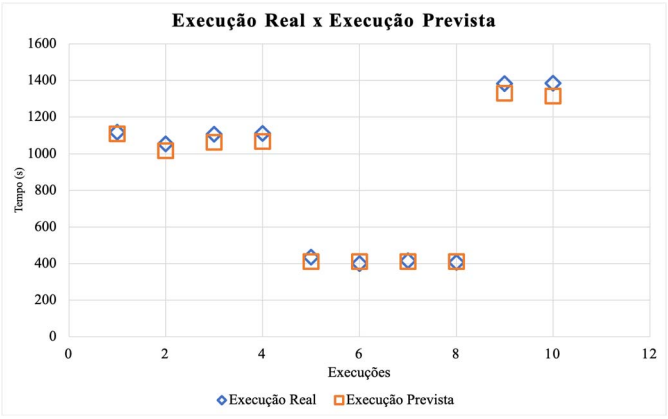

Fig. 6. Comparativo de Execução Real x Prevista.

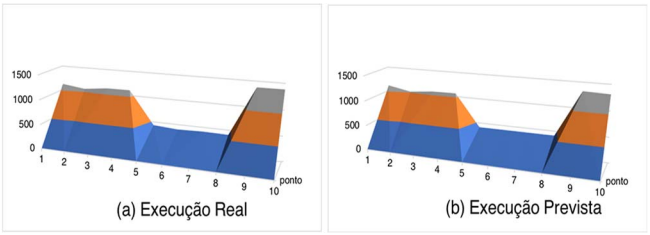

Fig. 7. Superfície de resposta.

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 executa de forma independente da integração com o provedor de nuvem, porém é altamente sugerido que esta seja realizada, a fim de melhor aproveitar os recursos computacionais a serem provisionados.

O Recomendador possui alguns parâmetros de entrada e ou configuração, que são o arquivo de origem que será submetido ao Weka (neste caso é obtido ou informada a quantidade de linhas e colunas), e qual o tipo da função *fitness* que será adotado, sendo, como já mencionado: Tempo, Custo ou Tempo/Custo. Além destes, existe um opcional que é o limite mínimo ou máximo aceito pela função *fitness*, detalhado a seguir.

Os dados obtidos de novas execuções podem ser utilizados para o treinamento do Regressor, permitindo assim, a inclusão de novos modelos de execuções, e melhorando a acurácia do sistema. Recomenda-se esse novo treinamento quando a eficiência do modelo diminuir, principalmente pela adição de novos modelos de execução, para que estes possam ser incorporados à predição.

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,
    FitnessPercento: 0.038373161309446215,
    Servidor: 2,
    Host_1N: 1,
    Host_2N: 0,
    Host_4N: 1,
    Host_8N: 1
}
```

V. CONCLUSÃO

Este trabalho descreveu 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% no tempo e de 45,62% no custo de execução, respectivamente, em relação a uma seleção aleatória de configuração na nuvem.

Nessa perspectiva, o impacto no custo e no tempo da execução demonstra os benefícios do modelo, permitindo assim aos pesquisadores e usuários um melhor gerenciamento dos seus recursos. Esses benefícios também são estendidos caso esteja sendo utilizado um modelo de nuvem privada ou híbrida, pois assim, permite um melhor aproveitamento do parque tecnológico, por meio do compartilhamento mais eficaz dos recursos.

Cabe salientar que 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.

Apesar deste modelo utilizar algumas técnicas mencionadas em trabalhos correlacionados, este se diferencia principalmente pelo momento em que é utilizado, ou seja, antes do início da execução das tarefas, garantindo assim a melhor configuração possível, a priori. Em outras abordagens encontradas na literatura, as recomendações ocorrem durante a execução, necessitando-se de tempo e recursos para adaptação e efetivação da nova configuração. Neste aspecto, nossa proposta mostrou-se mais eficaz.

Por outro lado, embora tenham sido utilizadas tarefas baseadas no software Weka, como estudo de caso, a arquitetura proposta pode ser generalizada para outros tipos de aplicações, desde que se tenha acesso a uma base histórica de utilização. 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," *J. of Adv. Comp. Intell. and Intell. Inform.*, 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 Lat. Am. Trans.*, 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," *J. of Eng.*, vol. 2016, 2016.

[4] K. Sembiring and A. Beyer, "Dynamic resource allocation for cloud-based media processing," in *Proc. of the 23rd ACM Works. on Netw. and Oper. Syst. Supp. for Dig. Audio and Video*. Oslo, Norway, Feb. 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 *6th Intern. Conf. on Syst. of Syst. Eng.*, Albuquerque, New Mexico, USA, Jun. 27-30, 2011, pp. 276–281.

[6] O. Niehorster, A. Krieger, J. Simon, and A. Brinkmann, "Autonomic resource management with support vector machines," in *Proc. of the 2011 IEEE/ACM 12th Intern. Conf. on Grid Comp.*, Washington, DC, USA, 2011, pp. 157–164.

[7] A. A. Bankole and S. A. Ajila, "Predicting cloud resource provisioning using machine learning techniques," in *26th Ann. IEEE Can. Conf. on Elect. and Comp. Eng.*, Regina, Canada, 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 *7th Intern. Conf. on P2P, Par., Grid, Cloud and Int. Comp.*, Victoria, British Columbia, Canada, Nov. 12-14, 2012, pp. 363–368.

[9] Weka. The university of waikato. weka 3: Data mining software in java, Jan. 2017. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka.

[10] HP. Hp eucalyptus, Jun. 2016. [Online]. Available: http://hphelion.com.

[11] Pentaho (2017). Pentaho weka server datamining, Jan. 2017. [Online]. Available: http://wiki.pentaho.com/display/DATAMINING/Weka+Server.

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

[13] Amazon. Amazon web service, Jun. 2016. [Online]. Available: http://amazon.com.

[14] O. P. d. Silva Neto, et al, “Detecção automática de massas em imagens mamográficas usando particle swarm optimization (pso) e índice de diversidade funcional”, M. S. thesis, Depart. de Pós-grad. em C. da Comp., Univ. Fed. do Mar., Maranhão, Brasil, 2016.

[15] Scikit-learn: Machine Learning in Python, Pedregosa et al., *JMLR 12*, pp. 2825-2830, 2011. (http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html).

[16] C. Sheppard, Genetic Algorithms with Python. 1st ed., Create Space 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 Publ.*, pp. 1–7, 2017. [Online]. Available: http://hdl.handle.net/10198/14061.

[21] S. M. Parikh, “A survey on cloud computing resource allocation techni- ques,” in Engineering (NUiCONE), in *Nirma Univ. Intern. Conf. on. Eng*., Ahmedabad, India, 2013, pp. 1–5.

[22] GridSearchCV. scikit-learn - machine learning in python, Sep. 2017. [Online]. Available: http://scikit-learn.org/stable/.

[23] L. H. V. Nakamura et al., “An Analysis of Optimization Algorithms designed to fully comply with SLA in Cloud Computing,” *IEEE Lat. Am. Trans.*, vol. 15, no. 8, pp. 1497-1505, 2017.

[24] A. Souza Ribeiro and D. Bianchini, "The deployment of Systems in Cloud Computing environment: A Methodology to Select and Prioritize projects," *IEEE Lat. Am. Trans.*, vol. 15, no. 3, pp. 557-562, Mar. 2017.

[25] J. T. Jiménez, J. L. Guardado, N. G. Cabrera, J. R. Rodriguez and F. Figueroa, “Transmission expansion planning systems using algorithm genetic with multi-objective criterion,” *IEEE Lat. Am. Trans.*, vol. 15, no. 3, pp. 563-568, Mar. 2017.

[26] Duggan, Martin, et al. "Predicting host CPU utilization in cloud computing using recurrent neural networks." *International Conference for Internet Technology and Secured Transactions (ICITST).* IEEE, 2017.

[27] SONG, Binbin et al. “Host load prediction with long short-term memory in cloud computing.” *The Journal of Supercomputing*, v. 74, n. 12, p. 6554-6568, 2018.

[28] Chen, Jing, and Yinglong Wang. "A hybrid method for short-term host utilization prediction in cloud computing." *Journal of Electrical and Computer Engineering* , 2019.

[29] Kumar, Jitendra, and Ashutosh Kumar Singh. "Workload prediction in cloud using artificial neural network and adaptive differential evolution." *Future Generation Computer Systems*, no 81, pp 41-52, 2018.

Thiago Nelson Faria dos Reis Possui graduação em Ciência da Computação (2004) e Mestrado em Ciência da Computação (2018) pela Fundação Universidade Federal do Maranhão. Especialização em Análise e Projeto de Sistemas, Especialização em Redes de Computadores, MBA em Gerenciamento de Projetos e Gerente de Projetos com certificação Project Management Professional, PMP e Professional Scrum Master - PSM. Possui experiência em desenvolvimento de software, banco de dados, aplicações web e desktop, Segurança, Perícia Forense, Gerenciamento de Projetos, BI - Business Intelligence, Cloud Computing e tem interesse nos temas de Computação em nuvem, Aprendizado de Máquina, Inteligência Artificial e Banco de Dados.

Mário Antonio Meireles Teixeira possui graduação em Ciência da Computação pela Universidade Federal do Maranhão (1992), mestrado (1997) e doutorado (2004) em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo. Atualmente é Professor Associado da Universidade Federal do Maranhão e docente permanente dos Mestrados em Ciência da Computação e Design da mesma Universidade. Tem experiência na área de Ciência da Computação, atuando principalmente nos seguintes temas: sistemas distribuídos, arquitetura orientada a serviços e computação em nuvem.

João Dallyson Sousa de Almeida possui graduação em Ciência da computação pela Universidade Federal do Maranhão (2007), mestrado em Engenharia de Eletricidade pela Universidade Federal do Maranhão (2010) e doutorado em Engenharia de Eletricidade pela Universidade Federal do Maranhão (2013). Atualmente é professor Adjunto III da Universidade Federal do Maranhão. Tem experiência na área de Ciência da Computação, atuando principalmente nos seguintes temas: processamento de imagens, reconhecimento de padrões, aprendizado de máquina, telemedicina e imagens médicas oftalmológicas. Atualmente coordena pesquisas com a finalidade de diagnosticar patologias da visão em imagens e vídeos.

Anselmo Cardoso de Paiva possui graduação em Engenharia Civil pela Universidade Estadual do Maranhão (1990), mestrado em Engenharia Civil - Estruturas pela Pontifícia Universidade Católica do Rio de Janeiro (1993) e doutorado em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (2001). Atualmente é Professor Titular da Universidade Federal do Maranhão. Sendo coordenador do Núcleo de Computação Aplicada NCA-UFMA. Tem experiência na área de Ciência da Computação, com ênfase em Processamento Gráfico (Graphics), atuando principalmente nos seguintes temas: Realidade Virtual e Aumentada, Computação Gráfica, GIS, Processamento de Imagens Médicas e Visualização Volumétrica. É membro da SBC (Sociedade Brasileira de Computação) e da ACM (Association for Computing Machinery).