Predição de Cargas de Trabalho de Nós de Computação em Nuvem Usando Modelos ARIMA e Redes Neurais Recorrentes Tipo LSTM

Danilo da Silva Santos*, Allan Edgard Silva Freitas†
Programa de Pós-Graduação em Engenharia de Sistemas e Produtos (PPGESP)
Instituto Federal da Bahia, Campus de Salvador, Bahia, Brasil

*danilo.silva@ifba.edu.br, †allan@ifba.edu.br

***Resumo*—Manter ambientes de computação em nuvem sempre disponíveis e cumprindo os níveis mínimos de serviço é uma tarefa que exige um gerenciamento eficiente dos recursos computacionais. Prever cargas de trabalho computacionais é uma estratégia que pode trazer ganhos ao possibilitar a tomada de ações proativas no uso eficiente dos recursos, no entanto, fazer tais predições em ambientes heterogêneos e dinâmicos é um desafio. Neste sentido, este trabalho demonstra a aplicação de técnicas para predizer valores de séries temporais utilizando modelos ARIMA e Redes Neurais Recorrentes (RNR) tipo LSTM ao analisar cargas de trabalho de nós de computação em nuvem. Ao final, os resultados demonstram que os modelos ARIMA e as Redes Neurais Recorrentes apresentam capacidade preditiva bem próximas entre si considerando o cenário proposto, sendo o ARIMA ligeiramente melhor em todos os conjuntos de dados do experimento.**

***Palavras-chave*—computação em nuvem, cargas de trabalho, predição, séries temporais, ARIMA, redes neurais recorrentes, LSTM.**

***Abstract*—Keeping cloud environments always available and meeting service agreement levels is a task that requires efficient management of computing resources. Predicting computational workloads is a strategy that can bring gains by enabling proactive actions to be taken in the efficient use of resources, however, making such predictions in heterogeneous and dynamic environments is a challenge. In this sense, this work demonstrates the application of techniques to predict time series values using ARIMA models and LSTM-type Recurrent Neural Networks (RNN) when analyzing workloads of cloud data nodes. Our results demonstrate that the ARIMA models and the Recurrent Neural Networks present a predictive capacity very close to each other considering the proposed scenario, with ARIMA being slightly better in all datasets of the experiment.**

***Index Terms*—cloud computing, workloads, prediction, time series, ARIMA, recurrent neural networks, LSTM.**

I. Introdução

A computação em nuvem já é uma realidade estabelecida em todo o mundo, principalmente em países que possuem um amplo setor de serviços [1]. A facilidade de disponibilização compartilhada de recursos computacionais, que podem ser rapidamente provisionados e com o mínimo de esforço [2], favorece a adoção deste modelo de computação e permite entregar ótimas experiências para os seus usuários, no entanto, a infraestrutura necessária para prover tais serviços com uma disponibilidade satisfatória gera custos para os provedores do ambiente em nuvem.

Como os recursos de computação em nuvem podem ser provisionados sob demanda, a camada de infraestrutura precisa ter a quantidade suficiente de recursos computacionais para atender a um pico de demandas. Devido a isto, há uma consequência negativa para os provedores ao manter estes recursos computacionais sempre ligados, pois, há estimativas de que o custo com energia elétrica e resfriamento representam cerca de 53% das despesas operacionais totais dos *data centers* [3].

Soluções tecnológicas podem ser utilizadas para monitorar a utilização de recursos e permitir o acionamento ou desligamento dos nós, conforme a necessidade do ambiente [4], no entanto, ao atuar de maneira reativa de acordo com dados do momento, podem ser surpreendidos com o aumento ou redução repentina de demandas de recursos computacionais, ocasionando ociosidades ou violações de Acordos de Nível de Serviço - SLA (do inglês, *Service Level Agreement*).

Ao utilizar os dados dos registros históricos da carga de trabalho (i.e. processador, memória RAM, etc.) destes *clusters*, é possível aplicar técnicas de regressão ou de aprendizado de máquina para prever a necessidade computacional destes *clusters* em períodos futuros.

Desta forma, se faz necessário incrementar modelos preditivos nestas soluções tecnológicas de maneira que a automação dos recursos da infraestrutura seja baseada na predição de cargas de trabalho futuras, ao invés de utilizar apenas as informações da carga do momento.

A predição de informações futuras é uma atividade que pode trazer inúmeros benefícios, desde a antecipação de soluções de problemas, bem como a otimização do uso de recursos. Segundo [5] a predição de cargas de trabalho em *clusters* de computação em nuvem possui diversos desafios, tais como:

- Adaptabilidade com o comportamento das aplicações hospedadas.
- Proatividade relacionada a provisionamentos e migrações de máquinas virtuais.
- Dados históricos insuficientes inicialmente.

- Para serem eficientes, os modelos não devem ser complexos.
- Dificuldade em identificar quais recursos serão monitorados devido à granularidade de dados.
- Identificação de padrões em um ambiente heterogêneo com aplicações distintas.

Este trabalho demonstra a aplicação de técnicas para prever valores de séries temporais utilizando modelos ARIMA e Redes Neurais Recursivas (RNR) com LSTM ao analisar cargas de trabalho de nós de computação em nuvem.

Além desta introdução, as demais partes do artigo estão organizadas como segue. A Seção II apresenta alguns conceitos sobre as técnicas preditivas utilizadas neste trabalho. Tratamos de alguns trabalhos relacionados na Seção III. Em seguida começamos a abordar aspectos relacionados ao conjunto de dados do experimento e as suas características na Seção IV. Realizamos as predições utilizando modelos ARIMA e Redes Neurais Recorrentes com LSTM na Seção V. Descrevemos o método de validação dos modelos na Seção VI, apresentamos os resultados na Seção VII e, por fim, na Seção VIII tecemos nossas conclusões sobre este trabalho.

II. Técnicas de Predição de Séries Temporais

A predição de séries temporais é uma importante área de pesquisa e aplicação que tem tido esforços no desenvolvimento e melhoria de modelos de predição nas últimas décadas [6].

As técnicas de predição de séries temporais relacionadas a cargas de trabalho podem ser classificadas em diversos esquemas [5], tais como os baseados em regressão linear ou em classificadores que incluem as redes neurais, com base estocástica, bem como outros baseados em predição de cinza, *clustering* de autocorrelação, caos, modelo de filtro de Kalman, wavelet, filtragem colaborativa, conjunto, além de outros esquemas híbridos com a combinação destes citados.

Neste trabalho as abordagens baseadas em regressão estatística (AR, MA, ARMA e ARIMA) e em inteligência artificial (Redes Neurais) foram escolhidas por serem duas das principais técnicas para predição de cargas de trabalho vistas na literatura [5].

Neste trabalho realizaremos previsões de séries temporais utilizando modelos Autorregressivos Integrados de Médias Móveis (ARIMA) e Redes Neurais Recorrentes (RNR) com LSTM. O método ARIMA, através de ajustes nos seus parâmetros, permite aplicar as outras abordagens de regressão citadas no parágrafo acima.

A. Modelos regressivos ARIMA

Os modelos lineares ARIMA de Box-Jenkins [7] são muito utilizados na predição de séries temporais. Em geral, uma série temporal pode ser modelada como uma combinação de parâmetros que representam como serão utilizados os valores do passado. O modelo ARIMA é expresso conforme a equação 1:

$$X_t = \theta_0 + \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + \cdots + \varphi_p X_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \cdots - \theta_q e_{t-q} \quad (1)$$

Onde X_t e e_t são os valores reais e o erro aleatório no tempo t, respectivamente. $\varphi_i (i = 1, 2, \ldots, p)$ e $\theta_j (j = 1, 2, \ldots, q)$ são parâmetros do modelo. p e q são as ordens dos polinômios autorregressivos e de média móvel, respectivamente [8].

B. Redes Neurais Recorrentes

Rede Neural Recorrente (RNR) é uma das arquiteturas básicas de rede neural. Muitas das arquiteturas avançadas de hoje são inspiradas por RNRs. A principal característica de uma RNR é que a rede tem conexões de *feedback*, ao contrário de uma rede neural *feed-forward* tradicional [9].

Essas redes em *loop* são chamadas recorrentes porque fazem as mesmas operações e cálculos para cada elemento em uma sequência de dados de entrada. RNRs têm memória, o que auxilia na obtenção de informações de sequências anteriores [10].

Uma RNR oferece um método no qual a saídas das previsões anteriores são usadas como uma entrada adicional para a próxima iteração [11]. No entanto, a RNR se mostra ineficaz em longos intervalos, pois, algoritmos de aprendizagem baseados em gradiente enfrentam um problema cada vez mais difícil à medida que aumenta a duração das dependências a serem capturadas, e com isto, há um problema de desaparecimento ou explosão do gradiente, resultando em falhas de treinamento [12].

C. Redes LSTM

As redes de Memória Longa de Curto Prazo - LSTM (do inglês, *Long Short-Term Memory*) são projetadas explicitamente para evitar o problema de dependência de longo prazo. O seu comportamento padrão é lembrar de informações por longos períodos [13].

A Figura 1 demonstra o funcionamento de uma RNR com células LSTM proposto por [14].

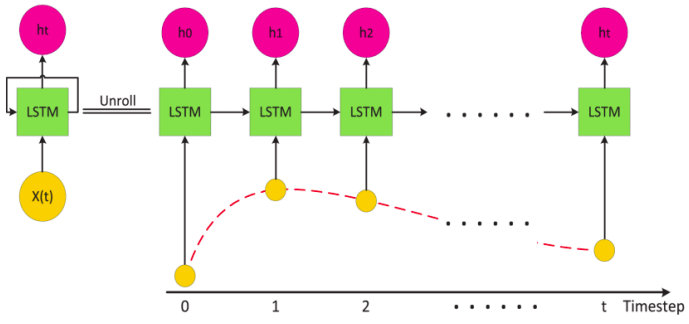

Figura 1. Rede LSTM usando séries temporais como entrada [15]

Internamente, cada célula LSTM usa uma porta de entrada, uma de esquecimento e uma de saída e estas portas definem se determinados dados podem passar ou não a depender da sua prioridade. Além disto, as portas também permitem que a

rede aprenda o que salvar, o que esquecer, o que lembrar, o que prestar atenção e o que imprimir [16].

III. Trabalhos Relacionados

Em [17] são apresentadas as vantagens e desvantagens de três técnicas de predição de cargas de trabalho usualmente aplicadas no contexto da computação em nuvem (ARIMA, MLP e GRU). Para a definição dos hiperparâmetros de cada modelo são utilizadas algumas técnicas como *Grid Search* e *Tree of Parzen Estimators* (TPE). Os resultados demonstram que os algoritmos possuem precisões equivalentes.

No entanto, o trabalho não define um *baseline* que sirva de parâmetro para saber se os modelos utilizados de fato são melhores que esta *baseline*. Em nosso trabalho entendemos que o limiar é o RMSE (do inglês, *root mean squared error*) obtido a partir da diferença da raiz quadrática média entre um conjunto dos valores estimados ingenuamente e o conjunto observado.

Os autores de [18] propõem um modelo preditivo baseado em redes neurais composto pelas etapas de preprocessamento que compreende a agregação da carga de trabalho e a normalização dos dados, seguido pelo treino da rede neural, retorno dos valores da predição e avaliação da acurácia. Com os resultados, os autores concluem que o modelo proposto supera o modo de predição baseado em propagação reversa.

Os autores utilizam dois conjuntos de dados, no entanto, não apresentam as características destes dados brutos, além disto, os experimentos são realizados no *software* MATLAB que exige licenças para utilização, dificultando a replicação do experimento por outros usuários.

Em nosso trabalho buscamos detalhar a etapa de preparação dos dados antes do preprocessamento visando identificar características do conjuntos de dados que permitissem escolher as melhores abordagens na aplicação dos modelos. Além disto, nosso experimento foi disponibilizado em código aberto utilizando linguagem de programação em *software* livre com as instruções detalhadas permitindo que qualquer interessado possa replicar o experimento.

Em [19] é proposta uma abordagem denominada REAP, que integra a seleção de recursos e técnicas de predição de uso de recursos para obter alto desempenho. A eficácia da proposta é avaliada em um ambiente de nuvem real, através de experimentos que demonstram que a abordagem supera os modelos existentes, melhorando significativamente a taxa de precisão e reduzindo o tempo de execução.

Como o conjunto de dados não foi disponibilizado, entendemos que não é possível replicar o experimento e avaliar outras possibilidades bem como a comparação com outras técnicas. Em nosso trabalho buscamos viabilizar a abertura dos dados e todas as etapas necessárias para a repetição dos nossos resultados.

Em [20], os autores desenvolveram uma estratégia de provisionamento de recursos baseada em previsões usando Rede Neural e Regressão Linear antecipando demandas futuras de recursos. Os resultados dos experimentos demonstraram que a técnica oferece um gerenciamento de recursos autoadaptativo para os aplicativos hospedados na nuvem.

Os dados utilizados no experimento são provenientes de uma ferramenta de *benchmark* para testar sites de comércio eletrônico, no entanto, entendemos que tal aplicação não reflete o comportamento de usuários finais e consequentemente os registros de cargas de trabalho podem não refletir valores de ambientes reais. Em nosso trabalho buscamos a utilização de conjuntos de dados de servidores do Grupo Alibaba, contendo dados de cargas de trabalho reais.

Em [21] é proposto um modelo de predição integrado, denominado BG-LSTM, composto por um modelo Bidirecional LSTM e um modelo *Grid* LSTM. Estes dois modelos complementam a capacidade de modelagem implícita de LSTMs, por isto, o BG-LSTM supera variantes típicas de LSTMs com a mesma configuração de parâmetros.

Nosso trabalho se diferencia destes ao avaliar de forma comparativa uma abordagem RNR baseada em LTSM com a técnica de predição ARIMA.

IV. Predição de Cargas de Trabalho

Diversas técnicas podem ser empregadas para encontrar predições de cargas de trabalho futuras, desde os modelos de regressão clássicos como o ARIMA, bem como técnicas mais atuais de aprendizado de máquina com as Redes Neurais Recorrentes LSTM. Ambas serão apresentadas e comparadas neste trabalho, mas antes disto, analisaremos o conjunto de dados que contém as cargas de trabalho de alguns nós de um *cluster*.

O conjunto de dados utilizado nas predições deste trabalho foi obtido do Programa de Rastreamento de *Cluster* do Grupo Alibaba [22]. Neste repositório existem dados de cargas de trabalho de 4.023 nós coletados no ano de 2018 durante oito dias consecutivos.

Os recursos monitorados dos nós incluem valores de CPU, RAM, disco e rede que foram disponibilizados em um arquivo no formato CSV, contendo o identificador do nó do recurso monitorado e os carimbos de tempo. Estes dados estão organizados de forma cronológica, o que caracteriza o conjunto como uma série temporal [23] e possibilita a aplicação de técnicas preditivas.

Nos experimentos deste trabalho faremos previsões do atributo CPU dos nós, mas é possível replicar o experimento e testar outros atributos, pois disponibilizamos o código-fonte no link que colocamos na Seção VII.

A. Extração, Carga e Preparação dos Dados

Os dados de cargas de trabalho dos 4.023 nós estavam disponibilizados em um arquivo de 9,0 GB, por isto, criamos um *script* para ler cada linha e separar as informações de cada nó em arquivos distintos.

Esta separação nos permitiu realizar as análises com um melhor desempenho e ajudou a identificar as características de cada nó. Com isto, constatamos que boa parte dos *hosts* possuíam registros faltantes em excesso, com intervalos grandes de ausência de dados que poderiam enviesar os resultados

da predição, devido a isto, identificamos os *hosts* que possuíam a maior quantidade de registros históricos.

Desta forma, utilizamos os dados das máquinas identificadas por m_103, m_694 e m_3330 devido a estas possuírem maiores quantidades de registros em relação às outras.

A partir desta identificação, fizemos a carga dos arquivos utilizando a biblioteca Pandas do Python e realizamos alguns ajustes com os recursos desta ferramenta.

Foi necessário converter o formato dos carimbos de tempo que estava como número inteiro sendo modificado para data e hora. Assim, observamos que os intervalos entre cada registro eram aleatórios, variando de segundos até minutos, devido a isto, fizemos a padronização fixando em intervalos de 5 minutos utilizando a função *resample* da biblioteca Pandas para modificar o intervalo das observações a caso houvessem dados faltantes a função *interpolate* se encarregava de preencher as lacunas com base nos valores vizinhos.

Ao fim da preparação dos dados, conforme Figura 2, mantivemos apenas a coluna que continha os dados históricos do percentual de uso de CPU referente aos últimos 3 dias.

time_stamp	cpu
1970-01-06 00:00:00	17.0
1970-01-06 00:05:00	15.0
1970-01-06 00:10:00	26.0
1970-01-06 00:15:00	14.0
1970-01-06 00:20:00	21.0
...	...
1970-01-08 23:35:00	14.0
1970-01-08 23:40:00	16.0
1970-01-08 23:45:00	12.0
1970-01-08 23:50:00	13.0
1970-01-08 23:55:00	21.0

Figura 2. Amostra dos conjuntos de dados usados no experimento

B. Análise da Série Temporal

Antes de submeter o conjunto de dados às técnicas preditivas, foram realizadas algumas análises para identificar características da série como a estacionariedade.

Um processo estacionário é aquele cujas propriedades estatísticas, como a média, não mudam com o tempo [24]. Esta característica pode ser observada na Figura 3 com exceção dos registros finais da máquina m_694.

Outra forma de identificar a estacionariedade da série é através do teste *Augmented Dickey-Fuller* (ADF). O teste apresenta alguns resultados, dentre eles, o Valor-P que indica estacionariedade caso seja inferior a 5% [25]. Conforme Tabela I apenas a máquina m_103 apresentou estacionariedade observando esta técnica.

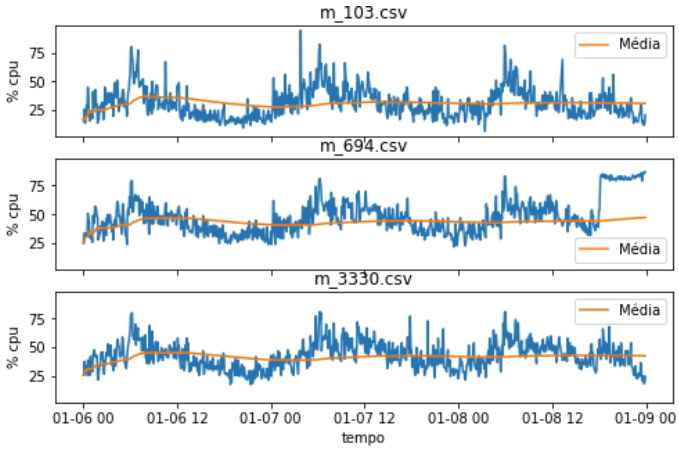

Figura 3. Média dos valores de CPU dos nós

Tabela I
RESULTADO DO TESTE ADF

dados	adf	pvalue
m_103.csv	-3.54	0.01
m_694.csv	-1.51	0.53
m_3330.csv	-2.73	0.07

C. Separação dos Dados para Treino e Teste

Por uma questão de desempenho durante o processamento dos modelos, utilizaremos uma porção dos dados equivalente ao período dos últimos três dias, sendo dois dias para a base de treino e um dia para os testes. Desta forma, tendo em vista que o conjunto de dados foi ajustado com intervalos de 5 minutos, então o conjunto de treino possuirá 576 registros, enquanto o de testes terá 288.

A escolha de uso de apenas 3 dias foi motivada por questões de desempenho no experimento, dado que cada execução de busca de parâmetros para as redes neurais utilizando os dados completos durava dias, portanto, com esta escolha os resultados de busca em cada conjunto de dados passaram a ser obtidos entre 24 e 45 horas após o início da execução.

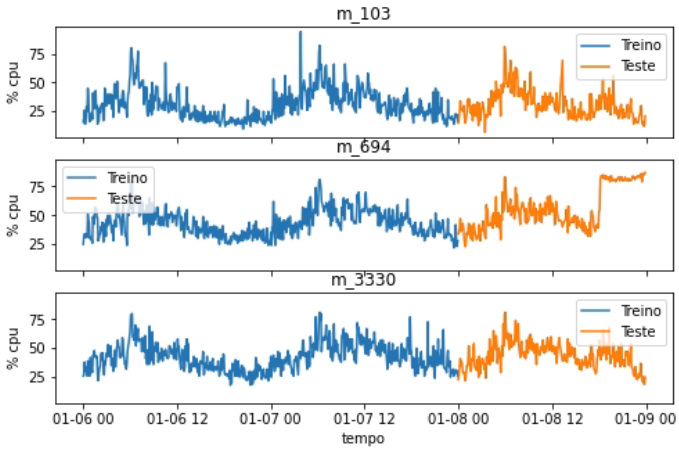

Figura 4. Dados separados em treino e teste

O conjunto de treino, como o nome já diz, será utilizado para treinar os modelos e gerar as previsões, enquanto o conjunto de teste será utilizado para comparar com as previsões obtidas em cada modelo que veremos a seguir.

V. Experimentos com Modelos Preditivos

A partir daqui iniciaremos apresentando os modelos através de experimentos aplicados nos conjuntos de dados e comparando os resultados provenientes de cada técnica.

A. Modelos Ingênuos

Antes de fazer a análise dos modelos preditivos ARIMA e da Rede Neural, faremos uma predição simples utilizando um método ingênuo (*Naïve*), de modo a utilizar esta predição como o nosso *baseline*, servindo como um patamar para que os demais modelos apresentem resultados melhores do que este modelo ingênuo.

Os modelos ingênuos podem ser simples previsões baseadas na média móvel, definindo a duração da média móvel como 1 ou até mesmo a repetição do último registro sendo a previsão do tempo t igual ao tempo t - 1 [26].

Neste experimento utilizaremos o valor do último registro como predição, para tal, replicaremos a coluna CPU, aplicaremos um atraso de um passo na nova coluna e teremos uma predição ingênua, conforme a Figura 5.

time_stamp	cpu	predict
1970-01-08 00:05:00	20.0	15.0
1970-01-08 00:10:00	25.0	20.0
1970-01-08 00:15:00	34.0	25.0
1970-01-08 00:20:00	27.0	34.0
1970-01-08 00:25:00	27.0	27.0
...	...	...
1970-01-08 23:35:00	14.0	16.0
1970-01-08 23:40:00	16.0	14.0
1970-01-08 23:45:00	12.0	16.0
1970-01-08 23:50:00	13.0	12.0
1970-01-08 23:55:00	21.0	13.0

Figura 5. Conjunto de dados com predição ingênua

B. Predição com Modelos ARIMA

Para realizar a predição com modelos ARIMA foi utilizado o *software* Statsmodels, uma biblioteca da linguagem de programação Python que fornece classes e funções para a estimativa de modelos estatísticos, além de permitir a realização de testes e exploração de dados estatísticos [27].

O modelo ARIMA possui três parâmetros em sua sintaxe ARIMA(p, d, q), conforme abaixo [28]:

- O p representa o componente autorregressivo.
- O d é a quantidade de diferenças para tornar a série temporal estacionária.
- O q representa o componente de média móvel.

Desta forma, tendo em vista que a série temporal da máquina m_103 é estacionária, segundo a técnica ADF, conforme explicado na subseção IV-B, foi definido que o parâmetro d (diferença) terá valor zero e os parâmetros p (autorregressivo) e q (média móvel) serão avaliados com diversos valores para identificar quais trazem uma predição mais precisa. Nas máquinas m_694 e m_3330 serão avaliadas a utilização do parâmetro d com valores 0 e 1.

Para possibilitar a validação de diversos modelos com parâmetros distintos, utilizamos a técnica *Grid Search* [29] que permite a identificação dos melhores hiperparâmetros ao avaliar todas as combinações possíveis dos parâmetros p, d, q considerando valores definidos em uma lista e utilizamos uma quantidade de espaços busca dentro das limitações de tempo e *hardware* disponíveis para o experimento. Ao fim da execução do *script* são exibidos os parâmetros dos modelos que apresentaram os menores erros conforme Tabela II.

Tabela II
Melhores parâmetros de cada modelo ARIMA por conjunto de dado

	Parâmetros		
dados	p	d	q
m_103.csv	2	0	1
m_694.csv	3	1	0
m_3330.csv	5	0	4

Uma vez identificados os melhores parâmetros, os valores foram utilizados na execução do modelo ARIMA no conjunto de treino das três máquinas visando obter as previsões e possibilitar a comparação com o conjunto de teste, conforme Figura 6.

C. Predição com Redes Neurais Recorrentes

Nesta categoria de rede neural, a cada passo da série temporal, a RNR recebe uma entrada, atualiza a camada oculta e apresenta a predição [30].

Para a aplicação desta técnica, utilizamos a biblioteca Keras do Python, que oferece *interfaces* de programação consistentes e simples para a criação de modelos de aprendizado de máquina [31].

Utilizamos o componente LSTM do Keras para que não seja necessário criar toda a estrutura da rede, tendo em vista que esta ferramenta possui uma estrutura pré-pronta e configurada [32].

Para identificar os melhores parâmetros de cada conjunto de dados, realizamos algumas combinações de hiperparâmetros e avaliamos todas as possibilidades possíveis dentro de alguns limites utilizando a técnica de *Grid Search* para modelos de aprendizado profundo [33]. Utilizamos uma quantidade de espaços busca dentro das limitações de tempo e *hardware* disponíveis para o experimento.

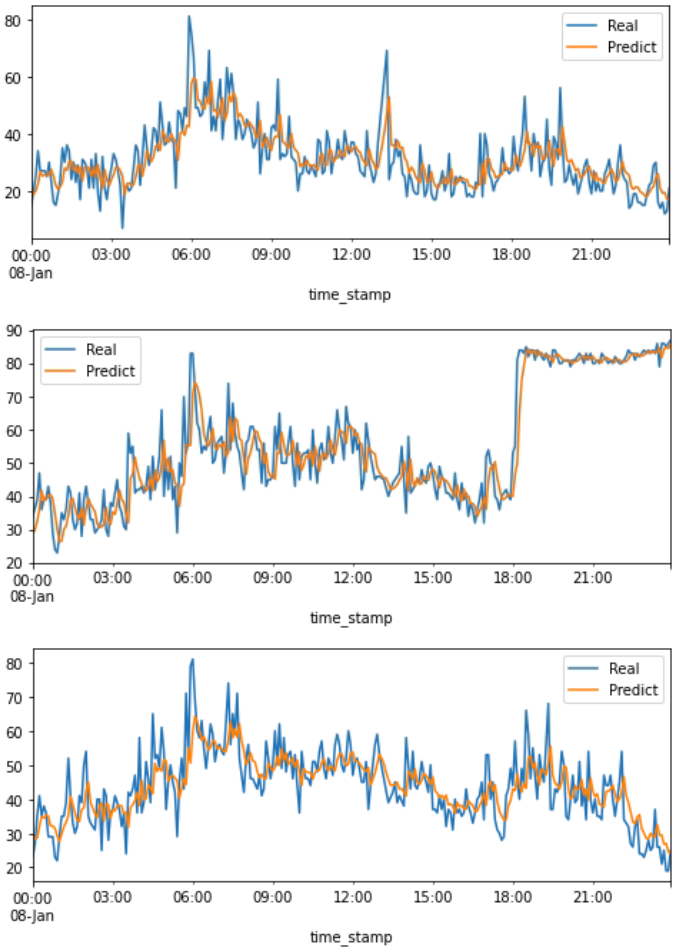

Figura 6. Comparação Entre os Valores Reais e as Predições do ARIMA

A Tabela III apresenta os melhores parâmetros identificados das redes neurais com LSTM utilizados em cada conjunto de dados.

Tabela III
PARÂMETROS DA REDE NEURAL COM LSTM

	Hiperparâmetros do LSTM			
dados	entradas	nós LSTM	épocas	batch
m_103	12	16	300	60
m_694	12	16	150	30
m_3330	12	12	300	60

O treinamento das redes utilizando estes parâmetros resultou em previsões que foram comparadas com os respectivos conjuntos de teste das três máquinas, conforme Figura 7.

VI. VALIDAÇÃO DAS TÉCNICAS PREDITIVAS

Diante das técnicas apresentadas, visualmente é possível observar a coerência entre os valores previstos e os valores reais do conjunto de testes.

Apesar desta percepção visual da capacidade preditiva dos modelos, se faz necessária a utilização de uma métrica para avaliar esta precisão e permitir a comparação da acurácia dos modelos.

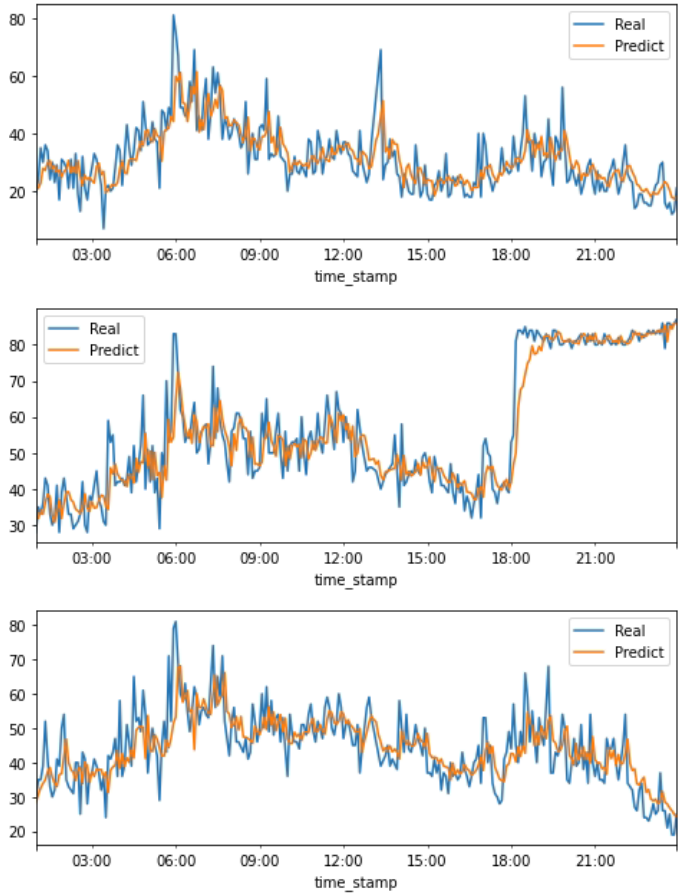

Figura 7. Comparação Entre os Valores Reais e as Predições do LSTM

Neste sentido, a métrica adotada para fazer a validação dos modelos foi a raiz do erro quadrático médio (*Root Mean Squared Error* - RMSE), uma das medidas mais comumente usadas para validar previsões numéricas [34], que consiste na raiz quadrada da variância média entre os valores reais e os valores preditos [19], representada pela equação 2:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(V_i - \hat{V}_i)^2}{n}} \quad (2)$$

Onde V é o valor real do conjunto de testes e $\hat{V}$ é a previsão gerada pelo modelo.

Com esta métrica, quanto menor o valor do RMSE melhor é a acurácia do modelo preditivo.

VII. RESULTADOS

As técnicas preditivas ARIMA e RNR LTSM foram aplicadas sobre os dados de alguns nós conforme apresentados na Tabela IV que apresenta o RMSE associado a cada modelo. Os parâmetros dos melhores modelos ARIMA apresentados foram obtidos *Grid Search* com as seguintes possibilidades:

- p: 0, 1, 2, 3, 4 e 5
- d: 0 e 1
- q: 0, 1, 2, 3, 4 e 5

Já o modelo de RNR foi executado com as seguintes possibilidades de parâmetros:

- Entradas: 12
- Nós LSTM: 10, 12, 14, 15, 16 e 30
- Épocas: 150, 200, 300
- Batch: 11, 20, 25, 30, 40, 50, 60, 70

Tabela IV
RMSE DOS MODELOS EM CADA CONJUNTO DE DADOS

dados	Naïve	ARIMA	RNR LSTM
m_103	9.0798	7.9295	7.9810
m_694	7.4392	6.8903	7.1733
m_3330	8.5146	7.3421	7.4131

Diante dos resultados apresentados na Tabela IV os modelos ARIMA e as Redes LSTM demonstraram capacidade preditiva bem próximas entre si, com o ARIMA ligeiramente melhor em todos os casos. Além disto, as duas técnicas preditivas foram melhores que o modelo ingênuo (*Naive*), utilizado como limiar. Cabe ressaltar que outros conjuntos de dados podem ser utilizados e apresentarem previsibilidades diferentes, além dos hiperparâmetros que podem ser ajustados visando melhorar ainda mais a acurácia dos modelos.

Visando a replicabilidade do experimento, disponibilizamos o código-fonte e as orientações em um repositório do Git Hub: https://github.com/dssantos/cloud-predict.

VIII. CONSIDERAÇÕES FINAIS

A aplicação de modelos preditivos para identificar cargas de trabalho futuras em *clusters* computacionais é uma abordagem interessante que permite viabilizar um gerenciamento inteligente dos estados dos nós visando o consumo eficiente de energia elétrica e predição de violações de SLA.

Com a predição de cargas de trabalho em *clusters* é possível desenvolver *scripts* ou aplicações mais elaboradas que utilizem esta capacidade preditiva para gerenciar melhor o uso dos recursos computacionais dos *datacenters*.

Portanto, ferramentas de gerenciamento de instâncias e *hosts* computacionais como [35] e [4] podem ser aperfeiçoados para atuar com os componentes preditivos implementados neste trabalho e oferecer um gerenciamento mais eficiente dos recursos, baseando-se em previsões de cargas de trabalho e atuando com proatividade.

Neste trabalho, ao aplicarmos os modelos ARIMA e as Redes Neurais Recorrentes com LTSM para predição sob um conjunto de dados reais do Programa de Rastreamento de *Cluster* do Grupo Alibaba, observamos capacidade preditiva bem próximas entre si considerando o cenário proposto, com o ARIMA ligeiramente melhor que o RNR LTSM em todos os conjuntos de dados do experimento, e ambas, como esperado apresentam-se melhor posicionadas que a estratégia *Naïve*.

Os resultados preliminares ora apresentados serão utilizados como base para subsidiar o desenvolvimento de módulo preditor de carga de trabalho para melhorar a eficiência de uma estratégia de gerenciamento de nós em nuvem computacional.

REFERÊNCIAS

[1] K. Vu, K. Hartley, and A. Kankanhalli, "Predictors of cloud computing adoption: A cross-country study," *Telematics and Informatics*, vol. 52, p. 101426, 2020.

[2] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing," *NIST Special Publication 800-145*, 2011.

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[4] D. da Silva Santos and A. E. S. Freitas, "Um protótipo para experimentos de eficiência energética em nuvem openstack," in *Anais Estendidos do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC, 2019, pp. 1–8.

[5] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Computing*, vol. 23, no. 4, pp. 2399–2424, 2020.

[6] Y. Chen, B. Yang, and J. Dong, "Time-series prediction using a local linear wavelet neural network," *Neurocomputing*, vol. 69, no. 4-6, pp. 449–465, 2006.

[7] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[8] R. S. A. Alsudani and J. Liu, "The use of some of the information criterion in determining the best model for forecasting of thalassemia cases depending on iraqi patient data using arima model," *Journal of Applied Mathematics and Physics*, vol. 5, no. 3, pp. 667–679, 2017.

[9] S. Hiriyannaiah, A. Srinivas, G. K. Shetty, G. Siddesh, and K. Srinivasa, "A computationally intelligent agent for detecting fake news using generative adversarial networks," *Hybrid Computational Intelligence: Challenges and Applications*, p. 69, 2020.

[10] A. Subasi, *Practical Machine Learning for Data Analysis Using Python*. Academic Press, 2020.

[11] L. Nashold and R. Krishnan, "Using lstm and sarima models to forecast cluster cpu usage," *arXiv preprint arXiv:2007.08092*, 2020.

[12] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[13] C. Olah, "Understanding lstm networks," 2015.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] H. Zhou, Y. Zhang, L. Yang, Q. Liu, K. Yan, and Y. Du, "Short-term photovoltaic power forecasting based on long short term memory neural network and attention mechanism," *IEEE Access*, vol. 7, pp. 78 063–78 074, 2019.

[16] P. T. Yamak, L. Yujian, and P. K. Gadosey, "A comparison between arima, lstm, and gru for time series forecasting," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, 2019, pp. 49–55.

[17] D. F. Kirchoff *et al.*, "Avaliação de técnicas de aprendizado de máquina para previsão de cargas de trabalho aplicadas para otimizar o provisionamento de recursos em nuvens computacionais," Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2019.

[18] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Generation Computer Systems*, vol. 81, pp. 41–52, 2018.

[19] G. Kaur, A. Bala, and I. Chana, "An intelligent regressive ensemble approach for predicting resource usage in cloud computing," *Journal of Parallel and Distributed Computing*, vol. 123, pp. 1–12, 2019.

[20] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[21] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, vol. 424, pp. 35–48, 2021.

[22] Alibaba, "Alibaba cluster trace program: cluster data collected from production clusters in alibaba for cluster management research," https://github.com/alibaba/clusterdata, 2018, (Accessed on 09/08/2020).

[23] G. Milone, *Estatística: geral e aplicada*. Pioneira Thomson Learning, 2004.

[24] G. P. Nason, "Stationary and non-stationary time series," *Statistics in volcanology*, vol. 60, 2006.

[25] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.

[26] V. M. D. Santos, “Previsão de vendas: como elaborar a sua?” https://www.fm2s.com.br/previsao-de-vendas-manual/, 2021, (Accessed on 18/08/2021).
[27] Statsmodels, “Statistical models, hypothesis tests, and data exploration,” https://www.statsmodels.org, 2021, (Accessed on 16/06/2021).
[28] S.-L. Ho, M. Xie, and T. N. Goh, “A comparative study of neural network and box-jenkins arima modeling in time series prediction,” *Computers & Industrial Engineering*, vol. 42, no. 2-4, pp. 371–375, 2002.
[29] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization.” *Journal of machine learning research*, vol. 13, no. 2, 2012.
[30] I. Sutskever, J. Martens, and G. E. Hinton, “Generating text with recurrent neural networks,” in *ICML*, 2011.
[31] Keras, “Keras. simple. flexible. powerful.” https://keras.io/, 2021, (Accessed on 24/06/2021).
[32] F. Feltrin, *Redes Neurais Artificiais*. Fernando Feltrin, 2020.
[33] J. Brownlee, *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.
[34] N. Gayatri, S. Nickolas, A. Reddy, S. Reddy, and A. Nickolas, “Feature selection using decision tree induction in class level metrics dataset for software defect predictions,” in *Proceedings of the world congress on engineering and computer science*, vol. 1. Citeseer, 2010, pp. 124–129.
[35] P. S. Nascimento and A. E. S. Freitas, “Cloudgi, gerenciando instâncias de um serviço replicado em uma plataforma de computaç ao em nuvem,” *Anais Estendidos do XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2015.