Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 169 (2020) 773–779

Procedia Computer Science

www.elsevier.com/locate/procedia

Postproceedings of the 10th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2019 (Tenth Annual Meeting of the BICA Society)

Using the machine learning methods for resource management of high availability broadcasting containerized system

Aleksandr Orlov[a,*], Mikhail Rovnyagin[a], Anastasiia Aminova[b], Anna Guminskaia[a], Fedor Chernilin[a] and Alexander Hrapov[a]

[a]*Department of Computer Systems and Technologies, National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Kashirskoe shosse 31, Moscow, 115409, Russian Federation*

[b]*Department of Applied Mathematics, National Research University HSE (Higher School of Economics), Myasnitskaya street 20, Moscow, 101000, Russian Federation*

Abstract

Most of today applications are built on a micro-service architecture, where a large application is divided into different functional parts that can be deployed on many containers that enable good load balancing. Container management tools need system load forecasting means to timely balance system load. It is an important problem for systems with direct streams of popular events which periodically have large splashes of a load. In this paper, we propose the load prediction method for such systems in two cases: usual and broadcasting workload. Also, we propose an architecture of adaptive infrastructure using our load forecasting method.

Keywords: Neural networks; Machine learning; Docker; Distributed systems; Containerization; Load prediction; Resource management

1. Introduction

Computing clusters are actively used for processing large amounts of data in high-loaded broadcasting systems. The

* Corresponding author. Tel.: +7-977-617-17-08.
E-mail address: alexandrorlovne@yandex.ru

10.1016/j.procs.2020.02.166

number of systems operating on the principle of microservice architecture is steadily growing. To increase cluster performance, horizontal scaling is used: if necessary, new computing servers are added to the cluster. However, with an increase in the number of users the overhead costs of maintaining the system also increase. To increase the profitability of the system and maintain a short response time on the client side, it is essential that, on the one hand, computing servers are not idle, and on the other, they are not overloaded.

For the early identification and solution of problems in such systems, monitoring and orchestration systems are used [7]. Monitoring and orchestration systems have a problem of timely deployment of additional facilities [10]. In the case of late deployment of the facilities, the response time on the client side increases, and the probability of failure of the nodes also increases [8]. But if there is always a large number of available facilities, the cost of system maintenance increases.

To determine the required number of capacities, it is necessary to predict the load on the system as a whole and on its nodes [11]. Therefore, modern container orchestration systems use various methods to predict the load on a broadcasting system (random forest [6], time-series analysis [1], users' activity analysis). However, there is a number of events in which it is extremely difficult to predict the load on the system by existing methods. For example, in the case of a broadcast of a socially significant event (such as a popular boxing or football match), the load on the broadcasting system will increase dramatically just before the start of the broadcast. Although the beginning of the broadcast is known in advance, traditional methods of forecasting the load will not allow forecasting the load on the system.

This paper proposes a new approach to forecast the load on a broadcasting system with applying data analysis from social networks based on machine learning, as well as the new linearly scalable architecture of the system for monitoring and orchestrating containers with a load prediction module using this approach.

The authors [1] propose to predict the load on the server using log analysis. This article says that existing methods focus on the relationships between workload and time, which gives rise to many unrealistic hypotheses. The model in this article is based on events (user actions) that actually cause the load of the system. Our article proposes to use this approach to forecast the load on the system in the case when no socially significant events are planned.

The authors [2] offer an analytical workflow for data analytics projects. This workflow saves the time needed for the implementation of the Data Science project by splitting the project into 6 phases: Discovery, Data preparation, Model planning, Model building, Communicate results, Operationalize. Also, the basic concepts of using the main predictive models, except neural networks, are described in this book.

In [3], the author states the basic concepts of using neural networks for the classification of text comments. The book describes the types of layers of neural networks and their scope of application. Also, the concepts of classification of textual data are described in this book. These concepts were used in this article.

This work is partly a continuation of research from [4], which proposes the distributed containerized system architecture that allows authors to collect metrics from nodes and generate control actions based on the collected metrics.

2. Research method

Our approach focuses on predicting the load on the system and its components using ML algorithms in order to distribute services to network nodes better for reducing latency and maintenance costs and for increasing system throughput.

Our methodology uses a combination of two methods for predicting the load on the system. The first method is the traditional method (analysis of logs, user activity and system load at previous periods of time), and it is supposed to be used in standard operating modes of the system. However, if there is a planned broadcast of a certain popular event by the system, this method is not suitable for forecasting the load, since the load on the system, in this case, depends primarily on how many people will be watching this event. In this case, it is proposed to use our new method. This method is based on forecasting the load through the analysis of data collected from social networks since it is possible to understand from this data how popular an event is in society.

The first method as a whole is described in [1] and other articles, so we will not consider it in detail. The main purposes of this article are to describe the second method, since this method is new, and to describe the architecture of the system.

The second method is the creation of a model that predicts the load on the basis of data from social networks. To build and implement the model, it is necessary to perform several stages: Discovery, Data preparation, Model planning, Model building, Communicate results, Operationalize. At stage 1, a review of available data sources is performed, and data available on social networks and within the system is examined. At the second stage, data is collected from social networks and from the system. The main goal is to collect data related to the time periods before and during the broadcasts with a large number of views. Also on this stage cleaning and transformation of data is required. At the third stage, a review of potentially suitable for load forecasting types of models and their preliminary testing take place. The most appropriate type of model is selected. At the 4th stage, the model is trained and tested. At stage 5, a discussion of the results takes place, with a possible return to the previous steps if the load prediction model does not meet the requirements. At stage 6, a system architecture is developed. This architecture must allow collecting data from social networks and, if necessary, rebuilding the model [14].

The architecture of our system consists of the following components:

1) Container Launch Manager on the basis of Docker Swarm.
2) Container Distribution Manager.
3) Node monitoring manager based on the collection of metrics and KPIs of nodes and containers.
4) ML-based metrics and KPI analysis node manager.
5) Manager of collecting and analyzing data from social networks based on Machine Learning (ML).

At the beginning, we want to define the architecture of such a system, which allows us to effectively manage the cluster and use data from nodes and from social networks to predict the load. It is suggested in this article to use modernized architecture proposed in [5]. The modernized architecture of this system is presented in Fig. 1.

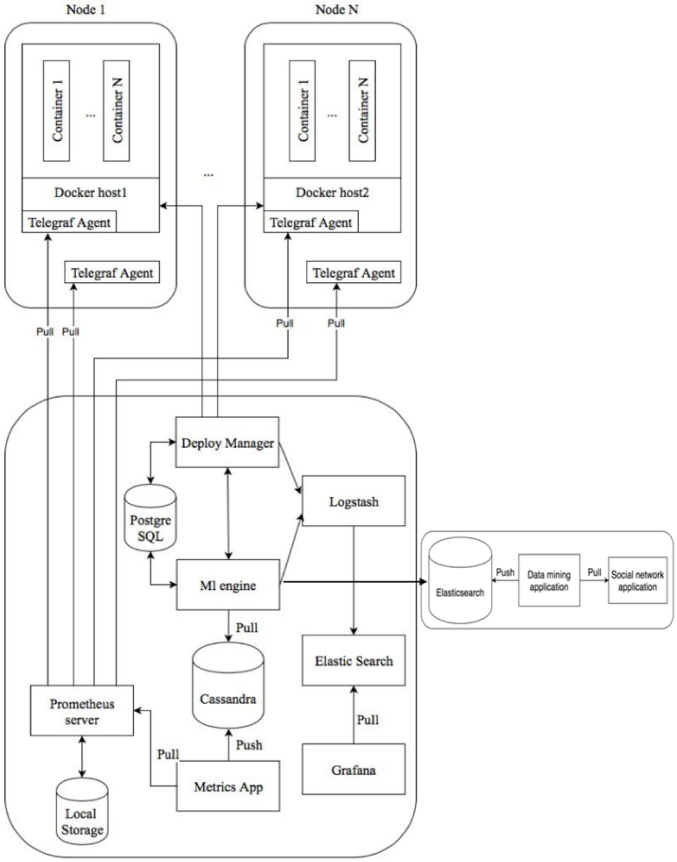

Fig. 1. The architecture of the monitoring and orchestration system

The architecture has the following form. On Node 1, ..., Node N nodes, there are server applications that serve clients, packaged in Docker containers. The main metrics are read from these nodes (CPU, I / O, RAM, number of threads and so forth) and sent to the Prometheus Server. Prometheus stores metrics in Local Storage. Local Storage is the temporary storage of metrics. The Metrics App is the application that periodically forwards metrics from the Local Storage (via Prometheus) to Cassandra, and, if necessary, removes needless data from the sample and adds new calculated columns. ML engine analyzes metrics collected from system and data from social networks and determines the need to transfer the container. Next, the ML engine requests a container transfer from the deployment manager subsystem (Deploy Manager). The Deploy Manager is directly responsible for the deployment and transfer of containers. Information about the location of containers on the nodes, as well as the success of the deployment, is stored in the PostgreSQL.

Fig. 2 shows the scheme of the load forecasting subsystem. ML Engine collects data from three different sources in order to be able to use both system and non-system metrics. Non-system metrics are stored in Elasticsearch.

To analyze the load and build a model for the mode of operation, when broadcasting a socially significant event, the following occurs. Data mining application searches for event-related posts in social networks by tags. These posts are checked for compliance with the theme of the event using the classification algorithm so that the sample does not include posts that do not actually relate to the event, although they have the requested tags. Next, additional information is extracted from the filtered entries (number of likes, number of views, number of reposts, comments and so forth). Further Data mining application writes the filtered posts and additional information into the Elasticsearch repository, from where, if necessary, they can be loaded into the ML Engine to train the model or forecast the load on the system. Based on the predictions, the ML engine sends a control action to the Deploy Manager in order to increase or decrease the number of containers at a certain point in time.

This scheme of the subsystem work allows to create forecasts based on a combination of data from different sources, which increases the prediction accuracy in different modes of operation.

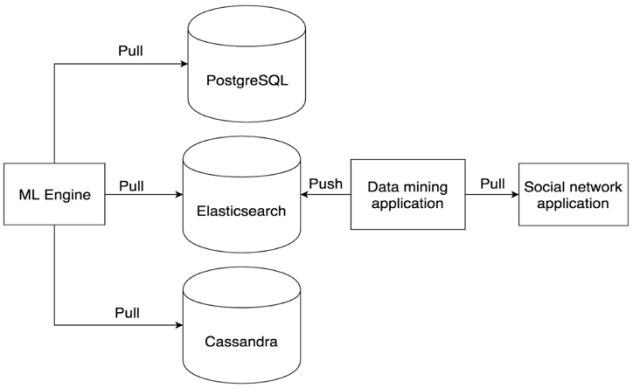

Fig. 2. Scheme of the load forecasting subsystem

3. Results and discussions

In this paper, experimental development of the cluster load prediction subsystem for the mode of broadcasting a socially significant event was carried out. As the predicted value, the number of views (that actually represents the load on the system) of the Mixed martial arts (MMA) fights videos in one of the groups in the most popular Russian social network was chosen. As input data on the basis of which the prediction was made, we selected posts for two months before each fight in this group. The initial hypothesis was the linear dependence of the number of views on certain metrics reflecting users' interest in the fight, such as: the total number of likes to all posts with certain tag, the average number of likes to these posts, the total number of views of the posts, the total number of comments to the posts. The metric of the total number of comments showed the smallest variance for predicting the number of views using the linear regression model. The dependence graph of the number of views on the number of comments is

presented in Fig. 3. This graph also shows a straight line constructed on the basis of the predictions of the created regression model. The graph shows that the relationship is truly linear. The equation of the constructed linear model: $y = 703.9 * x + 185704$. Standard deviation: 168302 views. R2 (coefficient of determination) regression score = 0.93, that indicates that there is a correlation between predicted and target values.

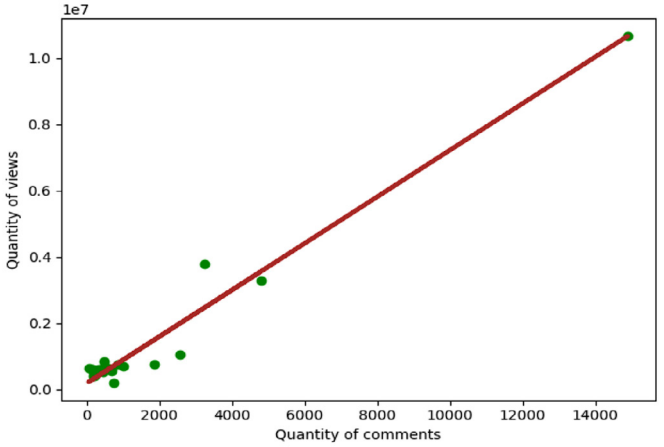

Fig. 3. Views by comments

Since the search for posts related to the topic is done by tags, then non-topic related messages can potentially get into the training sample and the prediction sample. For instance, when predicting the number of views of the MMA fight there can be 2 athletes with the same last name - Ivanov, one of them is a skier, and the second is a MMA fighter. Then it is needed to consider only comments to those posts with the #Ivanov tag that are related to mixed martial arts. To exclude such situations, it is necessary to filter the posts by the topics to which the event relates [9].

For this purpose, a neural network model has been developed [18]. The neural network classifies data into 14 classes of posts. The neural network model is presented in Fig. 4.

```
from keras import models
from keras import layers
from keras import regularizers
model = models.Sequential()
model.add(layers.Dense(64, activation='relu',
                       input_shape = (MAX_NUM_WORDS, )))
layers.Dropout(0.5)
model.add(layers.Dense(64, activation='relu'))
layers.Dropout(0.5)
model.add(layers.Dense(14, activation='softmax'))
```

```
model.compile(optimizer='rmsprop',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])
```

Fig. 4. Keras neural network model architecture

The topology of the neural network includes 2 hidden layers of the Dense type with 64 neurons and Relu activation function. The output layer is of the Dense type and has Softmax activation function because thus it is possible to predict the probability that the comment belongs to a particular class [12][13][17]. The loss function is categorical_crossentropy, the optimizer is rmsprop. To reduce overfitting, dropout was added in the form of Dropout layers [15]. The key idea of the dropout is to randomly drop units (along with their connections) from the neural network during training [16], which prevents units from co-adapting too much. The basic idea is that introducing noise

to output values can break randomly created patterns that do not have much value. Thus, the neural network reduces overfitting.

The training graph of the model is presented in Fig. 5. It is seen on the graph that 4 epochs are enough for quality training (this is the minimum of loss function). After the 25th epoch, the network begins overfitting, and the result worsens.

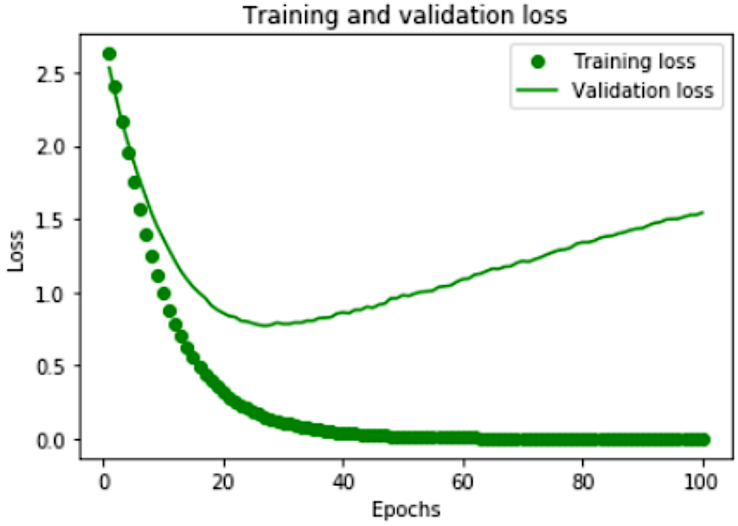

Fig. 5. Epochs vs validation loss

The quality of random prediction is less than 10 percent, and the quality of prediction of the resulting model after its training for 25 epochs is 85%, which is significantly higher than the accuracy of random prediction.

In the absence of a planned event, the load on the system can be predicted using the method presented in [1].

Thus, the combination of the new method and the formerly developed method allows forecasting quite accurately the load on the system in the standard mode of operation and in the mode of broadcasting a socially significant event. This way, it is possible to define the necessary number of nodes at a certain time for processing requests.

4. Conclusion

Thus, in this article, a new method of forecasting the load on the system during the broadcast of a socially significant event has been created. Also, a distributed architecture of the system, which allows to collect and analyze the metrics and logs of nodes and also to collect and analyze non-system data has been developed. Also, this article proposes to combine existing methods and the developed method of forecasting the system load in different modes of operation. The use of this architecture, coupled with a combination of described forecasting approaches, makes it possible to reduce the response time on the client and decrease the number of servers in the cluster due to the efficient load distribution. As further research, it is proposed to increase the number of parameters by which the prediction is made, and it is also proposed to optimize the data collection process for forecasting.

References

[1] Zheng Huang, Jiajun Peng, Huijuan Lian, Jie Guo, and Weidong Qiu. (2017) “Deep Recurrent Model for Server Load and Performance Prediction in Data Center,” *Complexity*

[2] David Dietrich, Barry Heller. (2015) “Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data”, *Wiley Publishing*

[3] Francois Chollet (2017) “Deep Learning with Python”, *Manning Publications Co.*

[4] M. M. Rovnyagin, A. V. Guminskaia, A. A. Plyukhin, A. P. Orlov, F. N. Chernilin and A. S. Hrapov. (2018) "Using the ML-based architecture for adaptive containerized system creation," *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering*: 358-360.

[5] Puder, Arno, Römer, Kay, Pilhofer, Frank. (2006) "Distributed systems architecture - a middleware approach"

[6] R. Cao, Z. Yu, T. Marbach, J. Li, G. Wang and X. Liu. (2018) "Load Prediction for Data Centers Based on Database Service," *2018 IEEE 42nd Annual Computer Software and Applications Conference*: 728-737.

[7] M. M. Rovnyagin, S. S. Varykhanov, Y. V. Maslov, I. S. Riakhovskaia and O. V. Myltsyn. (2018) "NFV chaining technology in hybrid computing clouds," *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering*: 109-113.

[8] D. A. Shulga, A. A. Kapustin, A. A. Kozlov, A. A. Kozyrev and M. M. Rovnyagin. (2016) "The scheduling based on machine learning for heterogeneous CPU/GPU systems," *2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference*: 345-348.

[9] Chiroma, Fatima, Liu, Han, Cocea, Mihaela. (2018) "Text Classification For Suicide Related Tweets"

[10] M. M. Rovnyagin, V. V. Odintsev, D. Y. Fedin and A. V. Kuzmin. (2018) "Cloud computing architecture for high-volume monitoring processing, "*2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering*: 361-365.

[11] Wu, Shuangyan, Zhang, Dong, Yan, Bingheng, Guo, Feng, Sheng, Dongpu. (2018) "Auto-Scaling Web Application in Docker Based on Gray Prediction"

[12] Kim, Jaeyoung, Jang, Sion, Choi, Sungchul, Park, Eunjeong. (2018) "Text Classification using Capsules"

[13] Li, Yue, Wang, Xutao, Xu, Pengjian. (2018) "Chinese Text Classification Model Based on Deep Learning", *Future Internet*

[14] Willi Richert, Luis Pedro Coelho. (2013) "Building Machine Learning Systems with Python", *Packt Publishing*

[15] Yao, Liang, Mao, Chengsheng, Luo, Yuan. (2018) "Graph Convolutional Networks for Text Classification"

[16] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, Salakhutdinov, Ruslan. (2018) "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*: 1929-1958.

[17] Berna Altınel, Murat Can Ganiz. (2018) "Semantic text classification: A survey of past and recent advances", *Information Processing & Management*: 1129-1153.

[18] Jochen Hartmann, Juliana Huppertz, Christina Schamp, Mark Heitmann. (2018) "Comparing automated text classification methods", *International Journal of Research in Marketing*