# ТЕХНОЛОГИЯ ГИБРИДНЫХ КОНТЕЙНЕРИЗИРОВАННЫХ ВЫЧИСЛЕНИЙ ДЛЯ ОРГАНИЗАЦИИ ВЫСОКОПРОИЗВОДИТЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ В КЛАСТЕРНЫХ СИСТЕМАХ

***Ровнягин М.М.[1], Чугунков И.В.[2], Савченко Н.А.[3]***

*Значительное количество современных распределенных приложений развертываются в кластерном окружении. Наибольшей надежности работы таких приложений можно достичь путем декомпозиции бизнес-логики на отдельные независимые друг от друга функциональные компоненты так называемые микро-сервисы. Они могут представлять собой как бинарные исполняемые файлы, так и виртуализированные контейнеры на основе технологий гипервизора виртуальных машин, либо Docker. В статье на примере докеризации ML-моделей (в том числе гибридных) предлагается архитектура для создания многоузловых приложений контейнеризированного типа. Формулируются требования к коду модели. Рассматривается структура программного каркаса для выполнения экспериментов с предложенной архитектурой. Особое внимание уделяется исследованию производительности схемы авторизации доступа к Docker-контейнерам. Приводится описание процесса моделирования в виде сети массового обслуживания. Производится анализ результатов экспериментов, в частности, даются оценки зависимости интенсивности запроса к модели от вероятности устаревания токена, а также интенсивности запросов нового токена от времени жизни токена. Делается вывод об эффективности предложенного подхода. Отмечается, что наличие схемы авторизации на основе Bearer token не приводит к значительным потерям производительности.*

**Ключевые слова:** *HPC, ML, Docker, микро-сервисы, распределенные системы, контейнеризация.*

**DOI:** *10.21681/2311-3456-2019-3-39-44*

**Введение**

В современных приложениях все чаще применяются модели машинного обучения в самых разных целях. Для примера можно привести финансовые модели, модели распознавания изображений, модели прогнозирования временных рядов и многие другие. Если раньше модели редко встраивались непосредственно в вычислительные процессы (использовалось в основном пакетное исполнение с выгрузкой результатов работы модели в файл и последующим их импортом в вычислительное приложение), то сегодня ситуация изменилась. Все чаще требуется построение online прогнозов, где обращение к модели происходит непосредственно во время выполнения пользовательских запросов. Подобная организация систем требует наличия не только самой модели и данных, которые она обрабатывает, но также и среды запуска и исполнения, инфраструктуры авторизации и выполнения запросов.

**Связанные работы**

В настоящий момент большое количество работ [1,2,3] посвящено тематике контейнеризации программных модулей. Речь может идти как об изолированном запуске программных компонент в рамках технологии виртуализации сетевых функций [4,5], так и о контейнеризации IoT-программ [6]. Контейнеризация ML-моделей не является исключением [7,8]. Наиболее значимой практической работой по данному направлению является создание корпорацией Microsoft сервера для запуска R/python-моделей[4]. Также большой вклад в разработку методик виртуализации запуска внесли компании Facebook, Google and Uber[5]. Сравнение подходов к виртуализации (гипервизор против контейнеров) представлены в статьях [7,9].

**Архитектура**

Основываясь на вышеупомянутых исследованиях, в настоящей статье предлагается архитектура для запуска ML-моделей, построенная на основе технологий контейнеризации. Каждая модель машинного обучения упаковывается в отдельный контейнер. Помимо python/R кода самой модели в контейнер помещаются файлы с коэффициентами модели, зависимости, конфигурационные файлы и т.д.

К коду модели выставляются определенные требования, например, основная функция *predict* (точка

1 Ровнягин Михаил Михайлович, кандидат технических наук, ПАО Сбербанк, г. Москва, m.rovnyagin.2015@ieee.org

2 Чугунков Илья Владимирович, кандидат технических наук, доцент, ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», Национальный исследовательский ядерный университет «МИФИ», г. Москва, ivchugunkov@mephi.ru

3 Савченко Наталья Александровна, старший преподаватель ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», г. Москва, savchenko.n@gubkin.ru

4 Microsoft Machine Learning Server: / https://docs.microsoft.com/en-us/machine-learning-server/what-is-machine-learning-server

5 Introducing MLflow: an Open Source Machine Learning Platform: https://databricks.com/blog/2018/06/05/introducing-mlflow-an-open-source-machine-learning-platform.html

входа) должна иметь единственный аргумент-строку и единственное возвращаемое значение-строку. Разработчик модели должен предоставить также функцию инициализации *init*, в которой размещается код единоразовой загрузки библиотек (в том числе для работы с гибридной GPGPU-инфраструктурой) и файлов-зависимостей ML-модели. Модель поставляется в виде zip-файла со всеми перечисленными элементами.

Версия модели формируется как версия кода + версия коэффициентов (версия обучающей выборки). Архитектура предполагает обертывание предоставленных разработчиком модели функций *predict* и *init* в REST-сервис с последующим упаковыванием в контейнер с использованием технологии Docker [10]. Для развертывания модели может быть использована любая технология кластеризации Docker-контейнеров, например Docker Swarm, Kubernetes, OpenShift [11].

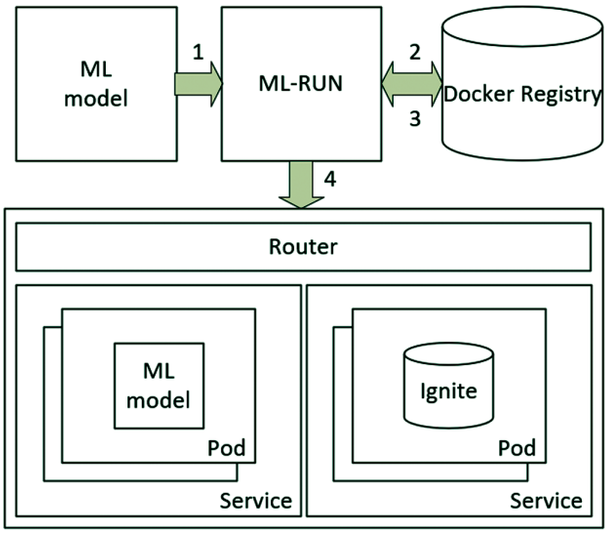


*Рис. 1. Архитектура системы*

Для выполнения экспериментов с предлагаемой архитектурой был разработан программный каркас ML-RUN (рис. 1).

Жизненный цикл модели в ML-RUN:

1. Загрузка zip-файла с моделью в программный каркас;
2. Формирование модуля-обертки и Dockerfile;
3. Сборка Docker-образа и публикация его в Docker registry
4. Развертывание модели в OpenShift.

Созданный Docker-контейнер опубликовывается в кластере OpenShift. Помимо Pods с ML-моделью в кластере могут быть также размещены контейнеры с экземплярами Apache Ignite, Apache Spark [12,13] и любыми другими технологиями для организации параллельных вычислений в памяти. При этом данные ресурсы будут доступны из кода развернутых в контейнерах ML-моделей.

Предлагаемая схема авторизации основывается на механизме Bearer token (рис. 2). Согласно данному

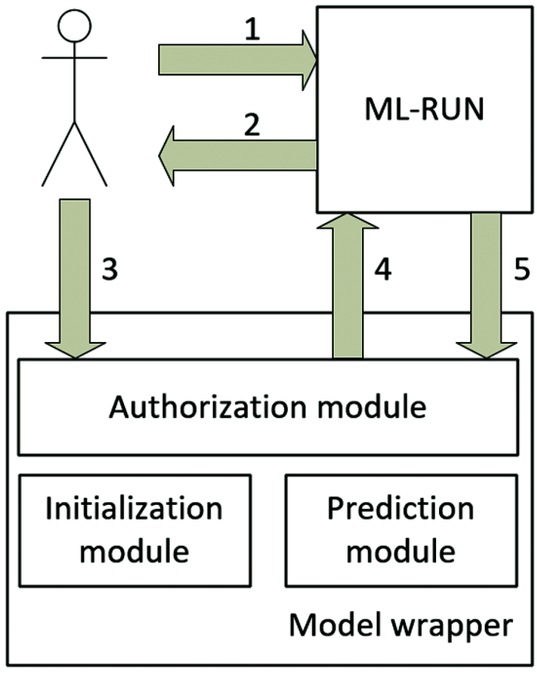


*Рис. 2. Схема авторизации доступа к моделям*

подходу пользователь обращается (1) к программному каркасу с использованием своих логин/пароль (либо посредством использования технологии LDAP и пр.). В ответ пользователь получает token, который действует ограниченное время. Все свои запросы к модели пользователь сопровождает полученным токеном (3). Модуль авторизации доступа к ML-модели проверяет наличие представленного пользователем token в кеше и его время жизни. В случае отсутствия записи в кеше модуль авторизации запрашивает информацию о токене в ML-RUN. Далее модуль обновляет информацию в кеше и пропускает (или не пропускает) пользовательский запрос к методу predict.

### Моделирование в виде сети массового обслуживания

Описанную выше архитектуру легко представить в виде разомкнутой сети массового обслуживания. Для этого введем следующие обозначения:

- $I$ – интенсивность входящего потока заявок в систему;
- $I_w$ – интенсивность входящего потока заявок в модуль-обертку ML-модели;
- $\lambda_i$ – интенсивность входящего потока заявок в подсистему *i*;
- $p_{ij}$ – вероятность передачи заявки из подсистемы *i* в подсистему *j*;
- $p_{i0}$ – вероятность передачи заявки из подсистемы *i* во внешнюю среду;
- $S_c$ – подсистема проверки токена авторизации;
- $S_t$ – подсистема получения токена авторизации;
- $S_p$ – подсистема обработки запроса в ML-модели;
- $S_a$ – подсистема авторизации;
- $S_r$ – подсистема маршрутизации;
- $K$ – количество экземпляров модуля-обертки ML-модели;

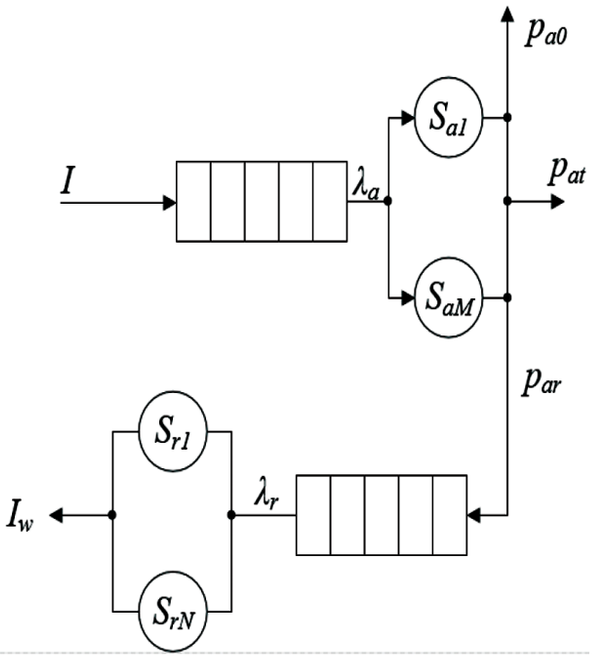


*Рис. 3. Представление архитектуры в виде сети массового обслуживания*

- M – количество экземпляров подсистемы авторизации;
- N– количество экземпляров подсистемы маршрутизации.

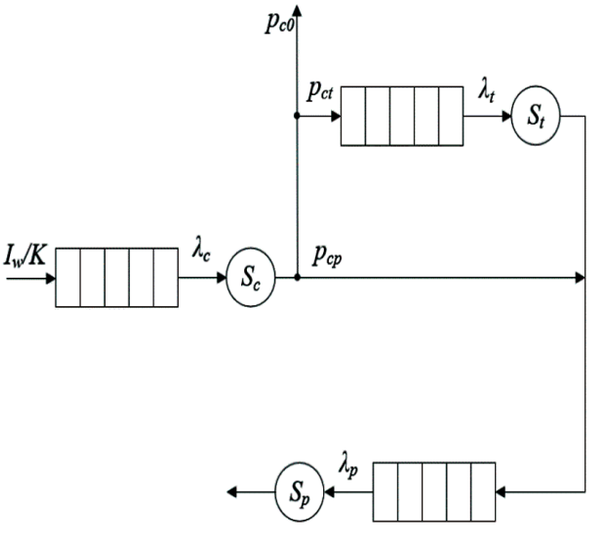


*Рис. 4. Модуль-обертка модели как сеть массового обслуживания*

Соответствующие обозначениям схемы сетей массового обслуживания представлены на рис. 3 и рис. 4. В рамках данной работы моделируется размещение программным каркасом ML-модели одного вида с числом реплик K.

Согласно теории массового обслуживания и основываясь на том, что поток заявок к ML-моделям пуассоновский, можно описать представленные выше системы соотношениями (1) и (2):

$$\begin{cases} \lambda_r = p_{ar}\mathrm{I} = (1 - p_{a0} - p_{at})\mathrm{I} \\ \lambda'_a = \frac{\lambda_a}{M} \\ \lambda'_r = \frac{\lambda_r}{M} \\ \mathrm{I} = I_t + I_i \\ I_w = \lambda_r \end{cases} \quad (1)$$

$$\begin{cases} \lambda_p = p_{cp}\lambda_c + \lambda_t \\ I_t = \lambda_t = p_{ct}\lambda_c \\ \lambda_c = I_w \\ p_{co} = 1 - p_{ct} - p_{cp} \end{cases} \quad (2)$$

**Экспериментальная часть.**

В ходе проведения экспериментов основной измеряемой характеристикой являлась интенсивность потока заявок. Это случайная величина, поэтому ее можно охарактеризовать такими параметрами как математическое ожидание (оценкой является среднее арифметическое по выборке) и дисперсия (оценкой является выборочная дисперсия). Согласно теории организации эксперимента, для экспериментального исследования некоторой величины необходимо выполнить серию экспериментов определенной длины при установленных показателях точности и доверительной вероятности. Среднее арифметическое значение выборки определяется по формуле:

$$\bar{x}_V = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (3)$$

где n-количество элементов в выборке (количество экспериментов), x_i i-е значение выборки (значение измеряемого параметра в i-м эксперименте).

Выборочная дисперсия является смещенной оценкой генеральной дисперсии, поэтому при проведении эксперимента используют стандартное отклонение по выборке, выражающееся через дисперсию по формуле:

$$s_V = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_V)^2} \quad (4)$$

Для большого числа экспериментов (n>100) для оценки доверительной вероятности используется функция Лапласа:

$$P\{\bar{x}_V - \delta < x < \bar{x}_V + \delta\} \approx 2\Phi(t) = p \quad (5)$$

где Ф(t)-функция Лапласа, δ-точность, $\{\bar{x}_V - \delta < x < \bar{x}_V + \delta\}$ – доверительный интервал, p-доверительная вероятность.

Для нормального распределения точность выражается формулой:

$$\delta = \frac{t s_V}{\sqrt{n}} \quad (6)$$

Соотношения (1) и (2) составлены в предположении, что потоки являются пуассоновскими. При большом n распределение Пуассона можно аппроксимировать распределением Гаусса с $\sigma \approx \sqrt{\lambda}$ [129]. Таким образом число экспериментов определяется неравенством:

$$n \geq \left(\frac{ts_V}{\delta}\right)^2 \quad (7)$$

Таким образом последовательность действий при проведении серии экспериментов (с учетом оценки количества измерений) можно задать следующим образом:

1) Проводится предварительная серия измерений ($n \geq 100$), определяется среднее значение по выборке $\bar{x}_V$ и стандартное отклонение $s_V$;
2) Задается доверительная вероятность *p*, в соответствии с формулой (5) определяется значение параметра *t*;
3) Задается точность *δ*;
4) Определяется количество экспериментов по формуле (7).

Согласно методике, описанной выше, проведены серии экспериментов. Усредненные значения для входного потока запросов с интенсивностью 10000 запросов в секунду, доверительной вероятностью 0,95 и точностью 0,0001 приведены на рис. 5 и рис. 6.

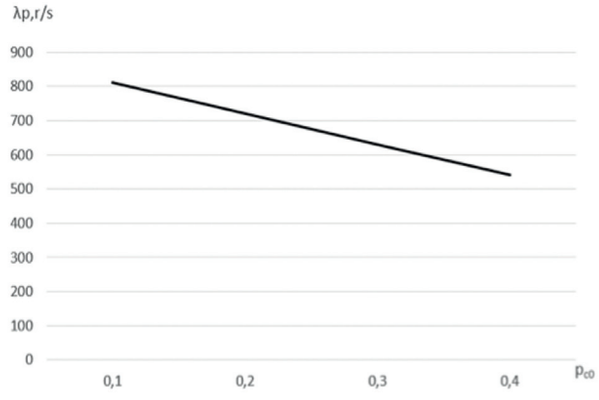


*Рис. 5. Зависимость интенсивности запроса к модели от вероятности устаревания токена*

Интенсивность запросов непосредственно к ML-модели от вероятности устаревания (либо невалидности) токена авторизации на этапе проверки зависит линейно.

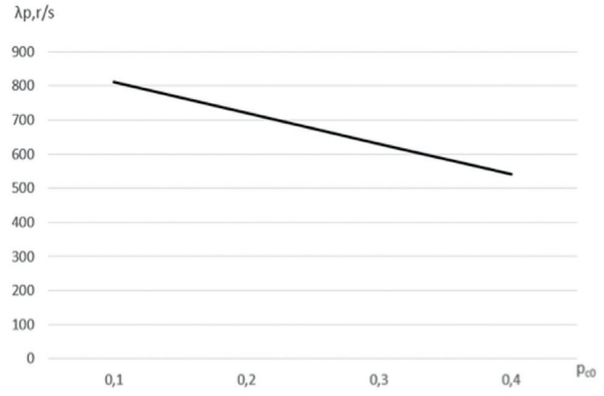


*Рис. 6. Зависимость интенсивности запросов нового токена от времени жизни токена*

Интенсивность поступления запросов от модуля-обертки в подсистему проверки/выдачи токенов ML-RUN экспоненциально зависит от времени жизни токена, однако остается низкой, если токен живет около секунды. При большом количестве экземпляров ML-моделей подсистема авторизации нуждается в масштабировании согласно представленной зависимости.

**Выводы.**

Таким образом, в настоящей статье представлена архитектура для изолированного запуска ML-моделей в распределенной среде исполнения. Преимуществами предлагаемого подхода являются: высокая отказоустойчивость и масштабируемость (обеспечиваются программным каркасом для распределенного запуска контейнеров), поддержка версионности моделей, наличие схемы авторизации доступа к моделям. Стоит отметить, что согласно теоретическим и экспериментальным оценкам, наличие схемы авторизации на основе Bearer token не приводит к значительным потерям производительности.

**Литература**

1. S. Kumar Pentyala, «Emergency communication system with Docker containers, OSM and Rsync,» 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bangalore, 2017, pp. 1064-1069.
2. O. Sallou and C. Monjeaud, «GO-Docker: A Batch Scheduling System with Docker Containers,» 2015 IEEE International Conference on Cluster Computing, Chicago, IL, 2015, pp. 514-515.
3. Qiang Liu; Wei Zheng ; Ming Zhang ; Yuxing Wang ; Kexun Yu "Docker-Based Automatic Deployment for Nuclear Fusion Experimental Data Archive Cluster", IEEE Transactions on Plasma Science, 2018 , vol. 46, pp. 1281 - 1284
4. Rovnyagin, M. M., Sergey S. Varykhanov, Yury V. Maslov, Iuliia S. Riakhovskaia and Oleg V. Myltsyn. "NFV chaining technology in hybrid computing clouds." 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus) (2018): 109-113.
5. Cziva, Richard and Dimitrios P. Pezaros. "Container Network Functions: Bringing NFV to the Network Edge." IEEE Communications Magazine 55 (2017): 24-31.
6. Morabito, Roberto, Ivan Farris, Antonio Iera and Tarik Taleb. "Evaluating Performance of Containerized IoT Services for Clustered Devices at the Network Edge." IEEE Internet of Things Journal 4 (2017): 1019-1030.
7. Rovnyagin, M.M., Guminskaia, A.V., Plyukhin, A.A., Orlov, A., Chernilin, F.N., & Hrapov, A.S. (2018). Using the ML-based

architecture for adaptive containerized system creation. 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 358-360.
8. Al-Rakhami, Mabrook, Mohammed A. Alsahli, Mohammad Mehedi Hassan, Atif Alamri, Antonio Guerrieri and Giancarlo Fortino. “Cost Efficient Edge Intelligence Framework Using Docker Containers.” 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech) (2018): 800-807.
9. Walter Blair, Aspen Olmsted, Paul Anderson, « Docker vs. KVM: Apache spark application performance and ease of use» 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), 2017, pp. 199-201.
10. Sarita and S. Sebastian, «Transform Monolith into Microservices using Docker,» 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, 2017, pp. 1-5.
11. OpenShift: Container Application Platform by Red Hat, Built on Docker and Kubernetes: [Online] / https://www.openshift.com
12. Meng, Xiangrui, Joseph K. Bradley, Burak Yavuz, Evan R. Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Bosagh Zadeh, Matei Zaharia and Ameet S. Talwalkar. “MLlib: Machine Learning in Apache Spark.” Journal of Machine Learning Research 17 (2016): 34:1-34:7.
13. Shanahan, James G. and Laing Dai. “Large Scale Distributed Data Science using Apache Spark.” KDD (2015).

# HYBRID CONTAINERIZED COMPUTING TECHNOLOGY FOR HIGH-PERFORMANCE DATA PROCESSING IN CLUSTER SYSTEMS

***Rovnyagin M.M.[6], Chugunkov I.V.[7], Savchenko N.A.[8]***

*A significant number of modern distributed applications are deployed in a cluster environment. The highest reliability of such applications can be achieved by decomposing business logic into separate independent functional components, the so-called micro-services. They can be either binary executables or virtualized containers based on virtual machine hypervisor technologies, or Docker. Using the example of the dockerization of ML models (including hybrid ones), the authors offer an architecture for creating multisite containerized applications. The model code requirements are formulated. The framework structure for performing experiments with proposed architecture is considered. Particular attention is paid to studying the performance of the scheme for authorizing access to Docker containers. The modeling process in the form of a queuing network is described. The experiments’ findings are analyzed, in particular, estimates are given of the dependence of model request intensity on the likelihood of token ageing, as well as the intensity of a new token request on the token lifetime. The conclusion is made about the effectiveness of the proposed approach. It is noted that the presence of the Bearer token-based authorization scheme the does not result in significant performance losses.*

***Keywords:*** *HPC, ML, Docker, micro services, distributed systems, containerization.*

**References**

1. S. Kumar Pentyala, «Emergency communication system with Docker containers, OSM and Rsync,» 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bangalore, 2017, pp. 1064-1069.
2. O. Sallou and C. Monjeaud, «GO-Docker: A Batch Scheduling System with Docker Containers,» 2015 IEEE International Conference on Cluster Computing, Chicago, IL, 2015, pp. 514-515.
3. Qiang Liu; Wei Zheng ; Ming Zhang ; Yuxing Wang ; Kexun Yu “Docker-Based Automatic Deployment for Nuclear Fusion Experimental Data Archive Cluster”, IEEE Transactions on Plasma Science, 2018 , vol. 46, pp. 1281 - 1284
4. Rovnyagin, M. M., Sergey S. Varykhanov, Yury V. Maslov, Iuliia S. Riakhovskaia and Oleg V. Myltsyn. “NFV chaining technology in hybrid computing clouds.” 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (2018): 109-113.
5. Cziva, Richard and Dimitrios P. Pezaros. “Container Network Functions: Bringing NFV to the Network Edge.” IEEE Communications Magazine 55 (2017): 24-31.
6. Morabito, Roberto, Ivan Farris, Antonio Iera and Tarik Taleb. “Evaluating Performance of Containerized IoT Services for Clustered Devices at the Network Edge.” IEEE Internet of Things Journal 4 (2017): 1019-1030.

6 Rovnyagin Mikhail Mikhailovich, PhD, PAO Sberbank, Moscow, m.rovnyagin.2015@ieee.org
7 Chugunkov Iliya Vladimirovich, PhD, Associate Professor, Gubkin Russian State University of Oil and Gas (National Research University), National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, ivchugunkov@mephi.ru
8 Savchenko Natalya Aleksandrovna, Gubkin Russian State University of Oil and Gas (National Research University), savchenko.n@gubkin.ru

7. Rovnyagin, M.M., Guminskaia, A.V., Plyukhin, A.A., Orlov, A., Chernilin, F.N., & Hrapov, A.S. (2018). Using the ML-based architecture for adaptive containerized system creation. 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 358-360.
8. Al-Rakhami, Mabrook, Mohammed A. Alsahli, Mohammad Mehedi Hassan, Atif Alamri, Antonio Guerrieri and Giancarlo Fortino. “Cost Efficient Edge Intelligence Framework Using Docker Containers.” 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech) (2018): 800-807.
9. Walter Blair, Aspen Olmsted, Paul Anderson, « Docker vs. KVM: Apache spark application performance and ease of use» 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), 2017, pp. 199-201.
10. Sarita and S. Sebastian, «Transform Monolith into Microservices using Docker,» 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, 2017, pp. 1-5.
11. OpenShift: Container Application Platform by Red Hat, Built on Docker and Kubernetes: [Online] / https://www.openshift.com
12. Meng, Xiangrui, Joseph K. Bradley, Burak Yavuz, Evan R. Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Bosagh Zadeh, Matei Zaharia and Ameet S. Talwalkar. “MLlib: Machine Learning in Apache Spark.” Journal of Machine Learning Research 17 (2016): 34:1-34:7.
13. Shanahan, James G. and Laing Dai. “Large Scale Distributed Data Science using Apache Spark.” KDD (2015).