Разработка системы мониторинга для серверного приложения

А. Ю. А. Альфара, Д. В. Королев, К.С. Зайцев, М.Е. Дунаев

***Аннотация*. Целью данной работы является исследование подхода для создания подсистемы мониторинга серверного приложения медицинской информационной системы для анализа снимков УЗИ щитовидной железы. Для достижения поставленной цели была спроектирована, реализована и исследована архитектура подсистемы анализа работоспособности серверного приложения. Основой ее является связка инструментов Prometheus, Grafana, Clickhouse и движок Apache Spark. Prometheus выступает в качестве средства сбора метрик с узлов анализируемой системы, Grafana - в качестве средства визуализации данных и оповещения о сбоях внутри медицинской системы, Clickhouse – для хранения данных. Для получения информации о работе серверного приложения медицинской системы используются авторские решения для создания экспортеров метрик, которые позволяют, как внедрять их в узлы приложения, так и оформлять в виде отдельных контейнеров. Такой подход позволяет быстро перестраивать систему мониторинга под изменения серверного приложения. Поиск аномалий в метриках контролируемых параметров в реальном времени выполняется с помощью Apache Spark и методов машинного обучения. Предложенное решение для создания подсистемы мониторинга серверного приложения показало свою эффективность при тестировании работы медицинской системы.**

Ключевые слова – мониторинг, серверное приложение, анализ данных, метрики, docker контейнеры, prometheus, grafana, Clickhouse, nginx, cadvisor, apache spark.

I. ВВЕДЕНИЕ

Современные программные системы часто представляют собой многокомпонентные комплексы, эффективность работы которых в сильной степени зависит от правильного функционирования отдельных компонентов.

Существует большое количество факторов, способных влиять на исправность работы системы, начиная от ошибок внутри самих программ и заканчивая внешними атаками на систему.

Статья получена 24 марта 2023.

Альфара Амир Юсеф Али, Национальный Исследовательский Ядерный Университет МИФИ, бакалавр, aay005@campus.mephi.ru

Королев Денис Вячеславович, Национальный Исследовательский Ядерный Университет МИФИ, бакалавр, kdv024@ campus.mephi.ru

Зайцев Константин Сергеевич, Национальный Исследовательский Ядерный Университет МИФИ, профессор, KSZajtsev@mephi.ru; Дунаев Максим Евгеньевич, Национальный Исследовательский Ядерный Университет МИФИ, аспирант, MEDunaev@mephi.ru

Для повышения надежности работы программных систем используют мониторинг с последующим анализом его результатов. Контроль функционирования отдельных компонентов системы позволяет своевременно определять и устранять угрозы корректной работе системы.

Среди разных решений для создания подсистемы мониторинга можно выделить такие связки инструментов как «Elasticsearch, Logstash и Kibana (ELK)», или вариации связок инструментов «Prometheus, Clickhouse, InfluxDB, Graphite и Grafana».

Стек инструментов ELK [1, 2, 3] применяется, когда предпочтение при мониторинге отдается анализу логов системы. В этом случае Elasticsearch играет роль нереляционного хранилища логов, Logstash используется для сбора, фильтрации и нормализации логов, Kibana представляет собой панель визуализации данных логов, хранящихся в кластере Elasticsearch. Если кроме логов необходимо выполнить также визуализацию метрик программной системы, то к стеку ELK необходимо добавлять инструмент Grafana.

Нам для мониторинга серверного приложения требуется хранить и анализировать значения метрик, а не логов. Поэтому в качестве решения для анализа значений метрик системы был выбран набор инструментов «Clickhouse, Prometheus и Grafana (CPG)» [4, 5, 6]. Здесь Clickhouse отведена роль хранилища данных. Значения метрик снимаются и попадают в хранилище с помощью инструмента Prometheus. Для визуализации метрик используется Grafana, позволяющая отображать данные в виде деловой графики. Возможно также подключение системы агрегации логов Grafana Loki для реализации отображения данных и из логов системы.

При формировании стека для анализа метрик рассматривались различные варианты инструментов, входящих в него. Так, в качестве решения для хранения значений метрик рассматривалось TimescaleDB — расширение PostgreSQL для высокопроизводительной работы с временными рядами [7]. В отличие от колоночной архитектуры Clickhouse, в TimescaleDB хранение данных реализовано в виде гипертаблиц, т.е. специально построенных наборов реляционных таблиц и, естественно,

обработка запросов происходит на языке PostgreSQL, в то время как в Clickhouse используется свой язык запросов, во многом использующий конструкции стандарта SQL. Проведенное сравнение производительности этих двух решений [8] показало, что Clickhouse обходит TimescaleDB по производительности при работе с большими данными и сложными запросами, но уступает при работе с небольшими наборами данных и более простыми запросами.

Кроме этого Clickhouse в отличие от TimescaleDB обладает набором встроенного инструментария для анализа данных. Поэтому по совокупности показателей предпочтение было отдано Clickhouse.

Схожие с Prometheus решения были детально проанализированы в статье [9]. По итогам сравнения предпочтение было отдано Prometheus, за счет точности и возможности к модификации. Не последнюю роль сыграла и популярность этого инструмента, благодаря которой существует большое количество готовых решений для экспорта значений метрик в формате Prometheus.

В ходе мониторинга программных систем необходимо практически непрерывно анализировать значения метрик, чтобы выявлять аномалии функционирования программных компонентов и стараться предсказать появление ошибок. Существуют разные подходы к определению аномальных значений [10], среди них очень популярно использование алгоритмов классификации и кластеризации [11].

Целью данной работы является исследование возможностей повышения надежности работы серверной части медицинской программной системы, с помощью встраивания в нее подсистемы мониторинга для анализа метрик в реальном времени.

Создание встраиваемой подсистемы мониторинга подразумевает знание состава серверного приложения, чтобы можно было определить инструменты, необходимые для работы с метриками при их визуальном контроле и выявления аномальных значений.

II. АРХИТЕКТУРА СЕРВЕРНОГО ПРИЛОЖЕНИЯ

Состав и структура серверного приложения определяют набор инструментов подсистемы мониторинга данных по его работе. Поэтому перед началом ее разработки рассмотрим архитектуру исходного серверного приложения, куда ее необходимо встроить.

На рисунке 1, представлена исходная (без подсистемы мониторинга) архитектура серверного приложения. Это приложение работает с нейросетевыми моделями анализа медицинских изображений (блок «ML-server»), поэтому наряду с обычными процессорами (CPU) использует также графические (GPU).

Как видно запросы на «ML-server» поступают с модуля web приложений «Django-server» через брокер сообщений Redis и асинхронный сервер отложенной отправки запросов «Celery». Так же в серверном приложении используется база данных PostgreSQL [12], с которой взаимодействует модуль web приложений Django. Для обеспечения надежной работы серверного приложения при таком взаимодействии необходимо отслеживать количество и частоту запросов к базе данных.

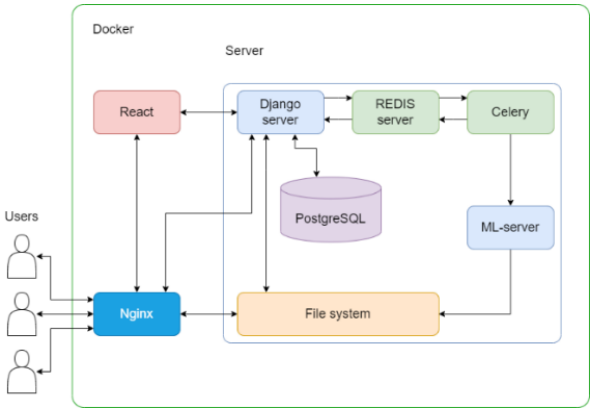

Рисунок 1 – Архитектура серверного приложения без модулей по сборке метрик.

Помимо взаимодействия с базой данных, модуль «Django-server» обрабатывает пользовательские запросы. Интерфейс между клиентской и серверной частями реализован посредствам RestAPI и библиотеки django rest framework [13].

В архитектуре веб-сервера используется обратный прокси сервер Nginx [14], который предоставляет информацию о поступающих на сервер пользовательских запросах. Данные с Nginx также можно проверять в процессе мониторинга и использовать в дальнейшем для анализа работоспособности системы.

Блок «React», как инструмент для создания пользовательских интерфейсов поддерживает взаимодействие с клиентской частью веб-сервиса. В этом модуле расположены статические файлы (html, css, js), которые редко изменяются и используются для создания графического пользовательского интерфейса клиентской части. Доступ к статическим файлам осуществляется через модуль «Nginx».

III. АНАЛИЗ ИНСТРУМЕНТОВ ДЛЯ СБОРА МЕТРИК

Проведенный анализ архитектуры серверного приложения позволяет перейти к этапу определения подходов и инструментов, с помощью которых можно экспортировать метрики о работе компонентов системы.

Для получения информативных метрик о работоспособности системы было решено экспортировать не только данные, связанные с выполнением основных задач каждого из компонентов системы, но и получать значения о потреблении ресурсов системы каждым из контейнеров.

Получение данных о работе каждого компонента системы требует специальных инструментов для экспорта метрик [15], которые определяются используемыми технологиями.

С помощью библиотеки Django-prometheus [16] удается получать данные о работе сразу трех контейнеров (Django server, React, PostgreSQL). Подключение экспортера для Django происходит в виде применения библиотеки django_prometheus и внедрения в код процессов, считывающих и отправляющих метрики по адресу {Django_address}/metrics. Этот экспортер позволяет считывать метрики о http и ajax запросах, которые обрабатывал Django server. Так же появился доступ к информации о взаимодействии с базой данных.

Сведения о поступающих на сервер запросах, удается получать благодаря Nginx-prometheus-exporter [17]. С его помощью данные о запросах и подключениях к Nginx серверу, которые доступны для экспортера по адресу "{Nginx_address}\stub_status", переводятся в формат, который доступен для считывания с помощью Prometheus.

В ходе контейнерной реализации экспортера, необходимо уделять особое внимание доступу к серверу Nginx. При полном отключении ограничений доступа к Nginx серверу по адресу {nginx address}/stub status и настройкой адреса для экспортера по имени контейнера с Nginx сервером можно беспрепятственно настроить соединение между экспортером и Nginx.

Мониторинг ресурсов системы проводится на двух уровнях. Сбор общих данных о потреблении ресурсов системы и данные о потреблении ресурсов каждым из контейнеров.

Анализ потребления ресурсов системы на общем уровне был реализован с помощью программы, написанной на языке Python. Программа считывает данные о загрузке процессора, оперативной памяти и видеокарты. Для анализа загрузки видеокарты была использована специальная библиотека pynvml [18], с помощью которой можно получить доступ к информации о ресурсах видеокарты на чипсете NVIDIA. После получения данных, метрики передаются в Prometheus, с помощью библиотеки prometheus_client. Для запуска этого решения в докер контейнере понадобилось установить надстройку над докером - nvidia-docker и убедиться в наличии актуальных драйверов для видеокарты.

Данные о потреблении ресурсов системы отдельными Docker контейнерами удается получать с помощью Cadvisor [19]. Этот инструмент позволяет считывать данные об использовании отдельными контейнерами ресурсов процессора, оперативной памяти и файловой системы.

IV. СОЗДАНИЕ АРХИТЕКТУРЫ ДЛЯ СБОРА МЕТРИК

Для реализации встроенного мониторинга в архитектуру серверного приложения были добавлены следующие программные модули по сбору метрик:

Django-exporter – собирает информацию о работе Django сервера и о запросах к СУБД PostgreSQL;

Cadvisor – собирает информацию об использовании CPU, GPU и ОЗУ каждым отдельным контейнером;

System metrics – собирает общую информацию об использованных CPU, GPU и ОЗУ серверного приложения;

Nginx stub status – собирает информацию об используемых API эндпоинтах;

Prometheus – агрегирует информацию от указанных выше сборщиков метрик.

Также были добавлены модули Carbon-CH и Clickhouse [20] для сохранения истории изменения метрик. Чтобы корректно получать данные от Prometheus и Clickhouse с того же URL, на котором размещен сам веб-сервер, был настроен доступ через обратный прокси сервер на Nginx [21]. Через этот прокси сервер пользователи получают доступ к инструментам сервиса, а разработчики могут считывать данные в Grafana [22] и анализировать их посредством Apache Spark.

На Рисунок 2 представлена архитектура серверного приложения с добавленными компонентами агрегации метрик.

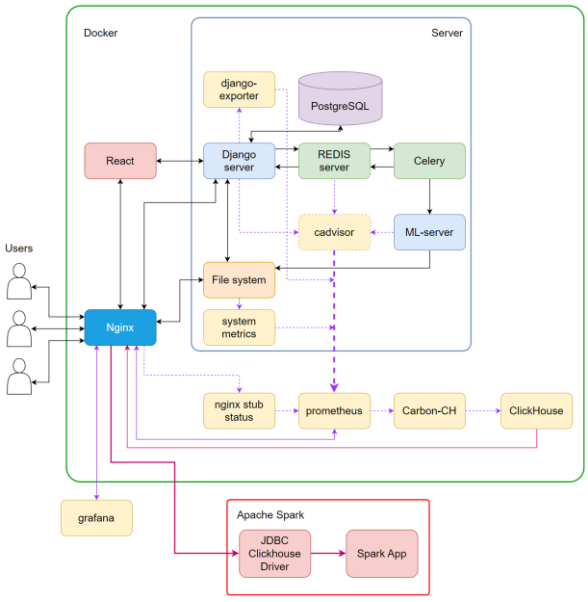

Рисунок 2 – Архитектура серверного приложения с модулями сборки метрик.

Первоначально был разработан вариант со сборкой метрик в Prometheus, расположенный вне сервиса мониторинга. При такой конфигурации Prometheus собирал метрики с различных эндпоинтов сервиса, что повышает общее

количество запросов к веб-серверу. Также усложняется настройка Prometheus перед началом мониторинга всех метрик сервера. В последующем отказались от такого решения.

Все данные, полученные с помощью ранее перечисленных инструментов сбора информации, считываются в Prometheus, откуда в дальнейшем передаются и записываются в Clickhouse. Для анализа данных метрики считываются с помощью JDBC драйвера и передаются в Spark приложение. Отображение экспортированных данных в виде набора графиков производится с помощью Grafana [23]. Графики с полученными данными об использовании ресурсов процессора, оперативной памяти и видеокарты системы представлены на рисунках 3-5.

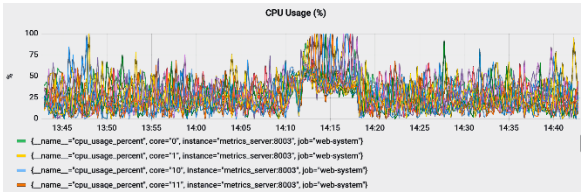

Рисунок 3 – Загруженность ЦП в процентах

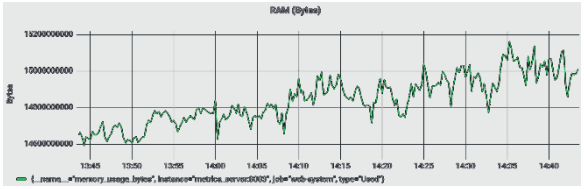

Рисунок 4 – Загруженность ОЗУ в байтах

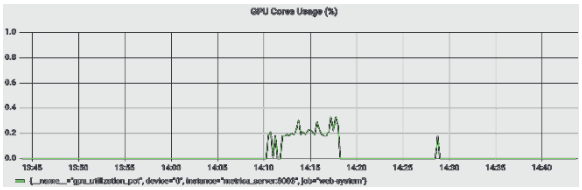

Рисунок 5 – Загруженность ГП в процентах

V. МОНИТОРИНГ И ОПОВЕЩЕНИЯ

Для визуализации результатов мониторинга была произведена настройка дэшбордов Grafana, позволяющих отображать принимаемые метрики. Графики с данными о POST/GET запросах и частотой http запросов полученных с Django экспортера представлены на рисунках 6, 7.

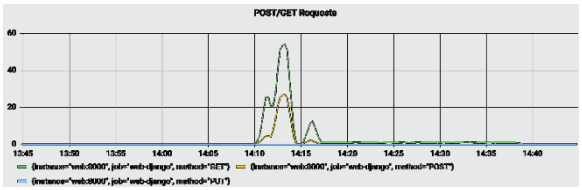

Рисунок 6 – Количество POST/GET запросов

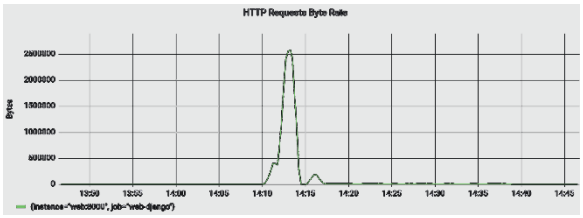

Рисунок 7 – Частота HTTP запросов

Для отслеживания нагрузки на систему от отдельных узлов серверного приложения были добавлены отдельные графики с информацией о потреблении ресурсов процессора и оперативной памяти. Графики с информацией о загрузке отдельных контейнеров представлены на рисунках 8 и 9.

С помощью встроенного в Grafana модуля настроены оповещения об ошибках и сбоях в подсистеме. Оповещения об ошибке и её состоянии отправляются в указанный в Grafana telegram чат. На рисунке 10 приведен пример оповещения об отсутствии соединения с сервером.

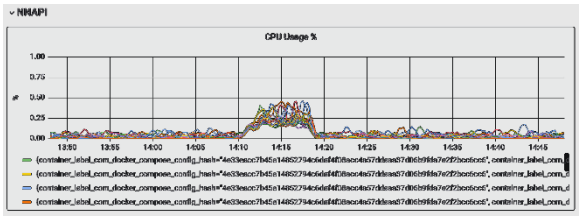

Рисунок 8 – Нагрузка на ЦП узлом с нейросетью

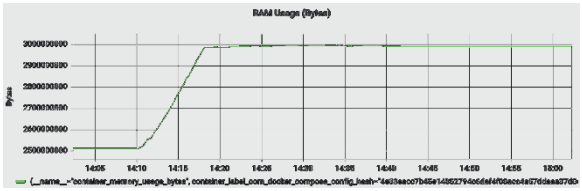

Рисунок 9 – Нагрузка на ОЗУ узлом с нейросетью

состоянии отправляются в указанный в Grafana telegram чат. На рисунке 10 приведен пример оповещения об отсутствии соединения с сервером.

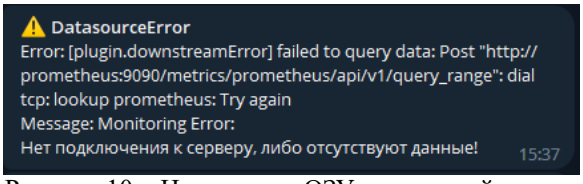

Рисунок 10 – Нагрузка на ОЗУ узлом с нейросетью

VI. ОБСУЖДЕНИЕ

Существует большое количество готовых решений для мониторинга программных систем, но вопрос универсального решения в данной области актуален и в настоящее время. По результатам разработки подсистемы мониторинга и её бесперебойной работы во время эксплуатации, можно сделать вывод, что разработанная подсистема является гибким и эффективным решением, благодаря

использованию связки инструментов Prometheus, Grafana и Clickhouse. Эта подсистема мониторинга может быть легко модифицирована и подстроена под анализируемую программную систему. Полученные авторами выводы нашли подтверждение или опровержение в других работах, посвященных созданию систем мониторинга.

Например, результаты в исследовании [4] подтверждают выводы, сделанные в ходе данного исследования. В приведенной статье рассматриваются различные подходы и инструменты для реализации системы анализа работоспособности интернет-магазина. Полученные авторами выводы свидетельствуют о том, что стек технологий Prometheus, Grafana, Clickhouse являются одним из наиболее универсальных и эффективных решений в данной области.

В статье [5] авторы рассказывают об инструментах и методах, которые применялись при создании облачного сервиса для мониторинга программных систем. В ходе разработки системы авторами были использованы такие технологии как Clickhouse и Grafana. В системе было реализовано представление данных в формате Prometheus, вследствие удобства и популярности данного решения. Однако, авторы столкнулись с проблемой реализации оповещений об ошибках из-за очень большого количества анализируемых данных.

В работе [24] проводится сравнение точности измерений при мониторинге программных систем с использованием Prometheus вместе с Kubernetes и Docker контейнерами. Это исследование скорее доказывает, что точность измерений при реализации системы мониторинга с помощью Kubernetes возрастает на 3,5%. Этот вывод свидетельствует о том, что разработанная в ходе данной работы система мониторинга может быть улучшена, при реализации системы с использованием Kubernetes и отказе от Docker контейнеров.

VII. ЗАКЛЮЧЕНИЕ

В ходе проделанной работы были проанализированы инструменты для получения информации о работоспособности узлов серверного приложения непрерывно работающей медицинской системы. В первоначальную архитектуру серверного приложения были успешно внедрены программные модули по сборке метрик. Получаемые метрики в последующем собирались с помощью Prometheus и сохранялись в Clickhouse для дальнейшего анализа. С помощью Grafana удалось настроить удобную визуализацию полученных метрик в виде наборов графиков.

Созданная подсистема мониторинга работы серверного приложения была протестирована, и продемонстрировала безотказную работу во время опытной эксплуатации серверного приложения для анализа снимков УЗИ щитовидной железы. Таким образом, созданная подсистема мониторинга доказала свою эффективность и, благодаря гибкой архитектуре, может служить отправной точкой для создания новых систем мониторинга работы серверных приложений.

БЛАГОДАРНОСТИ

Авторы выражают благодарность Высшей инжиниринговой школе НИЯУ МИФИ за помощь в возможности опубликовать результаты выполненной работы.

БИБЛИОГРАФИЯ

[1] R. J., "Data Discoverability in Science Gateways at Scale using Elasticsearch Cluster Architecture," *Practice and Experience in Advanced Research Computing,* pp. 1-3, 2022.

[2] G. D. E. Sankar P., "Social media monitoring using ELK Stack," *2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES),* vol. 1, pp. 231-235, 2022.

[3] M. O. K. V. Tokar D., "The IoT Applications Productivity: Data Management Model and ELK Tool Based Monitoring and Research," *2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET),* pp. 162-167, 2022.

[4] B. K. A. Kalyaeva A. V., "Architecture development of a combined system for monitoring the health and business processes of an online store," *Scientific and technical seminar of the Department of MOEM,* p. 56., 2021.

[5] W. H. e. al, "An Microservices-Based OpenStack Monitoring System," *2022 11th International Conference on Educational and Information Technology (ICEIT). – IEEE,* pp. 232-236, 2022.

[6] T. Leppänen, "Data visualization and monitoring with Grafana and Prometheus," 2021.

[7] T. M. L. Ivanova E. V., "Overview of modern time series processing systems," *Bulletin of the South Ural State University. Series: Computational Mathematics and Informatics,* vol. 9, no. 4, pp. 79-97, 2020.

[8] R. Booz, "What is ClickHouse, how does it compare to PostgreSQL and TimescaleDB, and how does it perform for time-series data?," Timescale, 21 10 2021. [Online]. Available: https://www.timescale.com/blog/what-is-clickhouse-how-does-it-compare-to-postgresql-and-timescaledb-and-how-does-it-perform-for-time-series-data/. [Accessed 2 5 2023].

[9] B. M. N. B. C., "Commercial and Open Source

Cloud Monitoring Tools: A Review.," in *Advances in Decision Sciences, Image Processing, Security and Computer Vision*, 2020, pp. 480-490.

[10] C. S. B. C. Qian Ma, "A novel model for anomaly detection in network traffic based on kernel support vector machine," in *hindawi*, 2021.

[11] V. B. D. N. Sarvani A., Anomaly Detection Using K-means Approach and Outliers Detection Technique, Springer Nature Singapore Pte Ltd., 2019.

[12] "PostgreSQL: The World's Most Advanced Open Source Relational Database," The PostgreSQL Global Development Group, [Online]. Available: https://www.postgresql.org/. [Accessed 23 01 2023].

[13] "Django REST framework," Encode OSS Ltd, [Online]. Available: https://www.django-rest-framework.org/. [Accessed 23 01 2023].

[14] "Nginx," F5, Inc, [Online]. Available: https://www.nginx.com/. [Accessed 23 01 2023].

[15] ITSumma, "Monitoring starts with metrics. Part 2: server software," *Habr,* 2022.

[16] "django-prometheus. Export Django monitoring metrics for Prometheus.io," [Online]. Available: https://github.com/korfuri/django-prometheus. [Accessed 23 01 2023].

[17] "NGINX Prometheus Exporter for NGINX and NGINX Plus," NGINX, Inc., [Online]. Available: https://github.com/nginxinc/nginx-prometheus-exporter. [Accessed 23 01 2023].

[18] "pyNVML. Python bindings to the NVIDIA Management Library," NVIDIA Corporation, [Online]. Available: https://pythonhosted.org/nvidia-ml-py/. [Accessed 23 01 2023].

[19] "Cadvisor. Analyzes resource usage and performance characteristics of running containers.," google, [Online]. Available: https://github.com/google/cadvisor. [Accessed 23 01 2023].

[20] "Serverless. Simple. ClickHouse Cloud.," ClickHouse, Inc., [Online]. Available: https://clickhouse.com/. [Accessed 23 01 2023].

[21] B. A., "How we built monitoring on Prometheus, Clickhouse and ELK," *Habr,* 2019.

[22] "Grafana. Operational dashboards for your data here, there, or anywhere," Grafana Labs, [Online]. Available: https://grafana.com/. [Accessed 23 01 2023].

[23] U. S., "Monitoring Basics (Review of Prometheus and Grafana)," *Habr,* 2023.

[24] Malathi K., "Multi Cluster Monitoring for Fault Detection Using Novel Kubernetes with Prometheus over Docker Container," *Journal of Pharmaceutical Negative Results,* pp. 1548-1555, 2022.

[25] T. J., "Elasticsearch Index and Query Design for Joining Full Text and Time Series Data," *Available at SSRN 4325567,* 2022.

[26] S. M. V., "Elastic Search Engine Analysis and Its Contribution to Quicker Data Retrieval Solutions for Numerous Issues," *International Journal of Engineering Research,* vol. 2, no. 1, pp. 05-08, 2022.

Development of a monitoring system for a server application

A. Yu. A. Alfara, D. V. Korolev, K. S. Zaytsev, M. E. Dunaev

Annotation. **The purpose of this work is to study an approach for creating a monitoring subsystem for a server application of a medical information system for analyzing thyroid ultrasound images. To achieve this goal, the architecture of the subsystem for analyzing the health of the server application was designed, implemented and studied. It is based on a bunch of Prometheus, Grafana, Clickhouse tools and the Apache Spark engine. Prometheus acts as a means of collecting metrics from the nodes of the analyzed system, Grafana - as a means of visualizing data and alerting about failures within the medical system, Clickhouse - for data storage. To obtain information about the operation of the server application of the medical system, author's solutions are used to create exporters of metrics that allow you to both embed them in the application nodes and arrange them in the form of separate containers. This approach allows you to quickly rebuild the monitoring system for changes in the server application. The search for anomalies in the metrics of controlled parameters in real time is performed using Apache Spark and machine learning methods. The proposed solution for creating a monitoring subsystem for a server application has shown its effectiveness in testing the operation of a medical system.**

Keywords – monitoring, server application, data analysis, metrics, docker containers, prometheus, grafana, Clickhouse, nginx, cadvisor, apache spark.

REFERENCES

[1] R. J., "Data Discoverability in Science Gateways at Scale using Elasticsearch Cluster Architecture," *Practice and Experience in Advanced Research Computing,* pp. 1-3, 2022.

[2] G. D. E. Sankar P., "Social media monitoring using ELK Stack," *2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES),* vol. 1, pp. 231-235, 2022.

[3] M. O. K. V. Tokar D., "The IoT Applications Productivity: Data Management Model and ELK Tool Based Monitoring and Research," *2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET),* pp. 162-167, 2022.

[4] B. K. A. Kalyaeva A. V., "Architecture development of a combined system for monitoring the health and business processes of an online store," *Scientific and technical seminar of the Department of MOEM,* p. 56., 2021.

[5] W. H. e. al, "An Microservices-Based OpenStack Monitoring System," *2022 11th International Conference on Educational and Information Technology (ICEIT). – IEEE,* pp. 232-236, 2022.

[6] T. Leppänen, "Data visualization and monitoring with Grafana and Prometheus," 2021.

[7] T. M. L. Ivanova E. V., "Overview of modern time series processing systems," *Bulletin of the South Ural State University. Series: Computational Mathematics and Informatics,* vol. 9, no. 4, pp. 79-97, 2020.

[8] R. Booz, "What is ClickHouse, how does it compare to PostgreSQL and TimescaleDB, and how does it perform for time-series data?," Timescale, 21 10 2021. [Online]. Available: https://www.timescale.com/blog/what-is-clickhouse-how-does-it-compare-to-postgresql-and-timescaledb-and-how-does-it-perform-for-time-series-data/. [Accessed 2 5 2023].

[9] B. M. N. B. C., "Commercial and Open Source Cloud Monitoring Tools: A Review.," in *Advances in Decision Sciences, Image Processing, Security and Computer Vision*, 2020, pp. 480-490.

[10] C. S. B. C. Qian Ma, "A novel model for anomaly detection in network traffic based on kernel support vector machine," in *hindawi*, 2021.

[11] V. B. D. N. Sarvani A., Anomaly Detection Using K-means Approach and Outliers Detection Technique, Springer Nature Singapore Pte Ltd., 2019.

[12] "PostgreSQL: The World's Most Advanced Open Source Relational Database," The PostgreSQL Global Development Group, [Online]. Available: https://www.postgresql.org/. [Accessed 23 01 2023].

[13] "Django REST framework," Encode OSS Ltd, [Online]. Available: https://www.django-rest-framework.org/. [Accessed 23 01 2023].

[14] "Nginx," F5, Inc, [Online]. Available: https://www.nginx.com/. [Accessed 23 01 2023].

[15] ITSumma, "Monitoring starts with metrics. Part 2: server software," *Habr,* 2022.

[16] "django-prometheus. Export Django monitoring metrics for Prometheus.io," [Online]. Available: https://github.com/korfuri/django-prometheus.

[Accessed 23 01 2023].

[17] "NGINX Prometheus Exporter for NGINX and NGINX Plus," NGINX, Inc., [Online]. Available: https://github.com/nginxinc/nginx-prometheus-exporter. [Accessed 23 01 2023].

[18] "pyNVML. Python bindings to the NVIDIA Management Library," NVIDIA Corporation, [Online]. Available: https://pythonhosted.org/nvidia-ml-py/. [Accessed 23 01 2023].

[19] "Cadvisor. Analyzes resource usage and performance characteristics of running containers.," google, [Online]. Available: https://github.com/google/cadvisor. [Accessed 23 01 2023].

[20] "Serverless. Simple. ClickHouse Cloud.," ClickHouse, Inc., [Online]. Available: https://clickhouse.com/. [Accessed 23 01 2023].

[21] B. A., "How we built monitoring on Prometheus, Clickhouse and ELK," *Habr,* 2019.

[22] "Grafana. Operational dashboards for your data here, there, or anywhere," Grafana Labs, [Online]. Available: https://grafana.com/. [Accessed 23 01 2023].

[23] U. S., "Monitoring Basics (Review of Prometheus and Grafana)," *Habr,* 2023.

[24] Malathi K., "Multi Cluster Monitoring for Fault Detection Using Novel Kubernetes with Prometheus over Docker Container," *Journal of Pharmaceutical Negative Results,* pp. 1548-1555, 2022.

[25] T. J., "Elasticsearch Index and Query Design for Joining Full Text and Time Series Data," *Available at SSRN 4325567,* 2022.

[26] S. M. V., "Elastic Search Engine Analysis and Its Contribution to Quicker Data Retrieval Solutions for Numerous Issues," *International Journal of Engineering Research,* vol. 2, no. 1, pp. 05-08, 2022.