Научная статья

УДК 004.75
DOI 10.25205/2541-9447-2024-19-2-5-14

Создание системы операторского контроля GARNET и распределенной системы управления на основе микросервисной архитектуры и применение на ускорителе ТИПр*

Михаил Станиславович Саратовских[1], Александр Николаевич Зимин[2]
Евгения Сергеевна Саратовских[3], Владимир Михайлович Гладков[4]
Андрей Юрьевич Орлов[5], Петр Алексеевич Федин[6]
Тимур Вячеславович Кулевой[7]

НИЦ «Курчатовский институт»
Москва, Россия

[1]saratovskikhms@gmail.com
[2]zimin.niitp@ya.com
[3]saratovskikh_evgenia@mail.ru
[4]gladkov3849@gmail.com
[5]orlov@itep.ru
[6]fedin-petr1991@yandex.ru
[7]kulevoy@itep.ru

Аннотация

В статье описаны основные принципы разработки распределенной системы управления (РСУ) и системы операторского контроля GARNET на основе микросервисной архитектуры в рамках работы на кластере высокой доступности. Описано применение системы операторского контроля в качестве компоненты РСУ. Приведены и описаны основные элементы программных компонент операторского контроля и РСУ, а также описан процесс конвейерной сборки и публикации программных средств в рабочую продуктовую среду, реализующий принцип непрерывной интеграции. Представлен механизм взаимодействия ключевых компонент между собой. Продемонстрирован механизм размещения сервисов управления при помощи системы контейнеризации Docker и оркестрации контейнеров Kubernetes. Также показаны примеры сервисов взаимодействия с пользователями в среде разрабатываемой системы операторского контроля GARNET, разделение пользователей по ролям и правам доступа, интеграция сервиса визуализации данных средствами Grafana, описан вектор дальнейшего развития РСУ и средств операторского управления, в частности, возможность использования практики разработки пользовательских web-интерфейсов, используя подход micro frontend. Представлены компоненты и результаты работы прототипа системы, разработанного для взаимодействия с измерительной инфраструктурой линейного ускорителя тяжелых ионов ТИПр (г. Москва, ККТЭФ).

Ключевые слова

ускоритель заряженных частиц, микросервисы, распределенные системы, системы управления

Благодарности

Работа выполнена на оборудовании Центра коллективного пользования КАМИКС (http://kamiks.itep.ru/) НИЦ «Курчатовский институт».

*Статья подготовлена по материалам конференции Russian Particle Accelerator Conference (RuPAC'23), Budker INP, 11–15 September 2023.

© Саратовских М. С., Зимин А. Н., Саратовских Е. С., Гладков В. М., Орлов А. Ю., Федин П. А., Кулевой Т. В., 2024

Для цитирования

Саратовских М. С., Зимин А. Н., Саратовских Е. С., Гладков В. М., Орлов А. Ю., Федин П. А., Кулевой Т. В. Создание системы операторского контроля GARNET и распределенной системы управления на основе микросервисной архитектуры и применение на ускорителе ТИПр // Сибирский физический журнал. 2024. Т. 19, № 2. С. 5–14. DOI 10.25205/2541-9447-2024-19-2-5-14

Management and Operator Control System based on Microservice Architecture and Application on the HIPr Accelerator

Mikhail S. Saratovskikh[1], Alexander N. Zimin[2]
Evgeniya S. Saratovskikh[3], Vladimir M. Gladkov[4],
Andrey Yu. Orlov[5], Petr A. Fedin[6],
Timur V. Kulevoy[7]

NRC “Kurchatov Institute”

[1]saratovskikhms@gmail.com
[2]zimin.niitp@ya.com
[3]saratovskikh_evgenia@mail.ru
[4]gladkov3849@gmail.com
[5]orlov@itep.ru
[6]fedin-petr1991@yandex.ru
[7]kulevoy@itep.ru

Abstract

The paper describes the basic principles of developing a distributed control system (DCS) and a GARNET operator control system based on a microservice architecture as part of a high-availability cluster. The application of the operator’s control system as a DCS component is described. The main elements of software components of operator control and DCS are presented and described. The process of conveyor assembly and publication of software tools into a working product environment, which implements the principle of continuous integration, is described. The mechanism of interaction of key components among themselves is presented. The mechanism for hosting management services using the Docker containerization system and Kubernetes container orchestration is demonstrated. Examples of services for interaction with users in the environment of the GARNET operator control system being developed, separation of users by roles and access rights, integration of the data visualization service using Grafana are shown. The vector of further development of DCS and operator control tools is described, in particular, the possibility of using the practice of developing user web interfaces using the micro frontend approach. The components and results of the operation of a prototype system designed to interact with the measurement infrastructure of the linear heavy ion accelerator HIPR are presented.

Keywords

charge particle accelerator, microservices, distributed systems, control systems

Acknowledgments

Work was performed using equipment of the KAMICS Center for Collective Use (http://kamiks.itep.ru/) of the National Research Centre “Kurchatov Institute”.

For citation

Saratovskikh M. S., Zimin A. N., Saratovskikh E. S., Gladkov V. M., Orlov A. Yu., Fedin P. A., Kulevoy T. V. Management and operator control system based on microservice architecture and application on the HIPr Accelerator. *Siberian Journal of Physics*, 2024, vol. 19, no. 2, pp. 5–14 (in Russ.). DOI 10.25205/2541-9447-2024-19-2-5-14

Введение

Современные ускорители заряженных частиц, как и прочие исследовательские или промышленные электрофизические установки, являют собой сложную совокупность комплексов устройств, разделенных по назначению, принципу работы и зачастую физически удаленных друг от друга. Данные особенности подсистем установок диктуют требования к системе управления и операторского контроля, которая, работая с распределенными подсистемами ускори-

теля или иного сложного устройства, должна сама отвечать требованиям распределенности и при этом обеспечивать единую точку доступа ко всем данным и к функциям управления электрофизической установкой для своевременного контроля состояния устройств и удобства работы с данными. Одно из существующих решений, к примеру, Tango Controls (https://www.tango-controls.org), основано на CORBA (common object request broker) (http://omniorb.sourceforge.net/omni42/omniORB/omniORB001.html) – протоколе вызова удаленных процедур, разработанном в 80-х гг. прошлого века. Из-за деталей реализации и общего устаревания данного подхода система управления, основанная на CORBA, обладает большим количеством критических мест и низким потенциалом горизонтального масштабирования. В рамках решения задачи создания гибкой системы управления и операторского контроля задействованы современные подходы к разработке программного обеспечения и его разворачивания в продуктовую среду выполнения. В работе представлены компоненты прототипа системы, разработанного для взаимодействия с измерительной инфраструктурой линейного ускорителя тяжелых ионов ТИПр [1].

Хранение данных

Прежде всего, необходимо решить задачу оперативного сохранения данных, а также задачу доступности архивных данных. Характер изменения объема получаемых данных во времени отображен на рис.1.

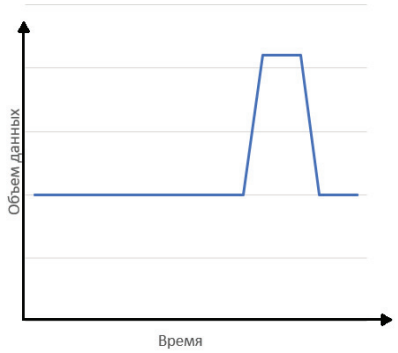

Рис. 1. Характер изменения объема данных
Fig. 1. Dynamic of changes in data volume

Приведенный примерный график соответствует ситуации с неким набором постоянно поступающей фоновой информации о состоянии установки (давление, температура узлов и т. п.) и резким увеличением объема информации при, например, включении импульсного источника электронов. При таком характере получения данных для их оперативного отображения оператору и для потоковой обработки требуется соответствующее хранилище. Традиционные SQL базы данных для этого не подходят, так как при большом количестве записей у подобных баз наблюдаются просадки в производительности при записи новых данных. Было принято решение использовать базы данных (БД) временных рядов, позволяющие сохранять темпы вставки данных вне зависимости от размера базы данных (рис. 2, 3) (https://habr.com/ru/companies/oleg-bunin/articles/464303).

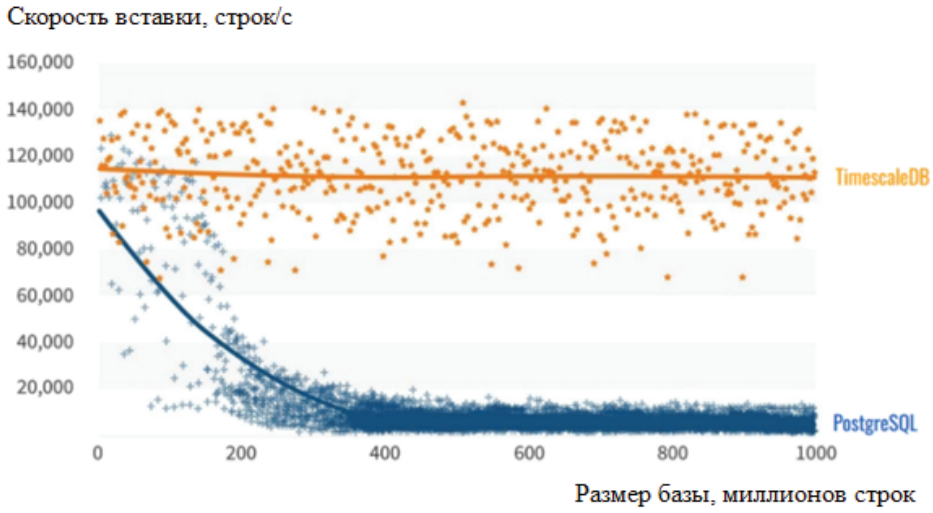

Рис. 2. Сравнение скорости вставки в SQL базу данных и в базу данных временных рядов
Fig. 2. Comparison of insert speed in SQL database and time series database

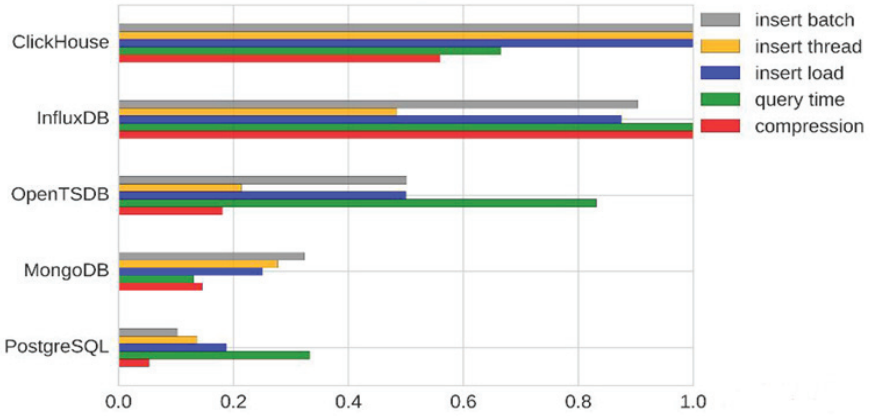

Рис. 3. Сравнение измеренных характеристик, нормированных на лучший результат в серии тестов с различными базами данных
Fig. 3. Comparison of measured characteristics normalized to the best result in a series of tests with different databases

Исходя из сравнительного анализа БД, в качестве базы для оперативного хранения данных предпочтение было отдано InfluxDB (https://www.influxdata.com/) как хранилищу с наиболее низкой задержкой вставки единичного значения (query time). ClickHouse (https://clickhouse.com/), которая является колоночной базой, в создаваемой системе предполагается использовать в качестве архивного хранилища для удобного доступа пользователей к произвольному набору собранных метрик. В то же время конфигурация электрофизической установки чувствительна к консистентности данных, поэтому все настройки устройств хранятся в реляционной БД.

Архитектура системы контроля

Программные средства для управления установками должны предоставлять следующие возможности:

1) прием данных с устройств;

2) обработка данных;
3) сохранение метрик о состоянии системы;
4) предоставление доступа к архивным данным;
5) обеспечение оператора наглядной и оперативной информацией о состоянии системы и ходе проводимых работ;
6) выдача команд для управления устройствами/удаленный вызов процедур;
7) выдача своевременных сигналов о неполадках и изменении состояния измерительной инфраструктуры.

На настоящий момент многие системы управления реализованы в виде приложений-монолитов, т. е. программ, осуществляющих весь набор операций с экспериментальными данными в рамках одной программы. Использование монолитного приложения или же групп подобных приложений небезопасно с точки зрения отказоустойчивости и сложностей в обслуживании, поскольку зачастую в подобных системах невозможно централизованное обновление компонентов. Система операторского контроля и управления GARNET (Generic Automated Research via Network Embedded Tools) создается на основе микросервисной архитектуры [2] (рис. 4). Данный подход позволяет создавать отказоустойчивую и широко масштабируемую систему, состоящую из множества микросервисов – набора программ, каждая из которых предназначена для решения узкоспециализированной задачи, к примеру – предоставление пользовательского интерфейса. Микросервисы запускаются в системе в контейнерах Docker (https://www.docker.com/resources/what-container/), что обеспечивает автономность, мультиплатформенность и возможность итеративного развития программного обеспечения. Помимо этого, система может быть развернута по частям или с неполным набором компонент для тестирования или для работы с ограниченным функционалом.

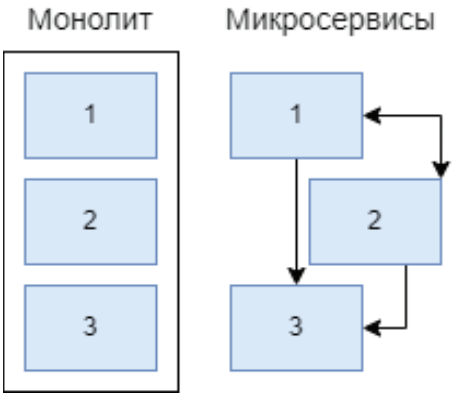

Рис. 4. Схематичное изображение модулей монолита и системы микросервисов, выполняющей те же задачи
Fig. 4. Schematic representation of the monolith modules and microservices system performing the same tasks

Для общения между собой микросервисы используют сетевой программный интерфейс (API – Application Program Interface) или же взаимодействуют посредством отправки сообщений в очереди сообщений или распределенный сетевой журнал. В GARNET для взаимодействия сервисов используется парадигма обмена состоянием (REST API) и запись в распределенный журнал Kafka (https://kafka.apache.org/documentation/#gettingStarted). Состав системы операторского контроля и управления представлен на настоящий момент следующими сервисами:

1) сервис обработки пакетов данных с устройств;
2) сервис отправки команд на устройства;

3) сервис настроек конфигурации установки;
4) подсистема анализа данных;
5) сервис предоставления пользовательского интерфейса;
6) сервис обновления значений на интерфейсе;
7) сервис обработки команд, передаваемых пользователем для управления;
8) сервис авторизации и предоставления разделенного в правах доступа к подсистемам;
9) сервис построения отчетов;
10) сервис выдачи уведомлений на интерфейс и целевые устройства;
11) сервис сбора логов подсистем.

Работа непосредственно с измерительной инфраструктурой вынесена в отдельную подсистему [3]. Эта подсистема разворачивается на подключаемых к устройствам машинах и взаимодействует с системой управления посредством отправки UDP-пакетов в сервис обработки пакетов и приемом UDP-пакетов от сервиса отправки команд на устройства. Вынесение работы непосредственно с драйверами устройств за пределы системы сервисов обеспечивает безопасное тестирование новых подключаемых устройств и отказоустойчивую работу уже подключенных средств измерения. В настоящий момент UDP-пакеты содержат информацию в формате JSON, в будущем планируется перейти на передачу по UDP-информации в виде Protobuf (https://protobuf.dev/) объектов для ускорения передачи информации по сети и уменьшения времени обработки пакетов. Таким образом, устройства измерительной инфраструктуры взаимодействуют с сервисами GARNET либо напрямую при наличии программируемого передающего устройства, либо посредством первоначального агрегирования/обработки данных на узловой машине с последующей передачей метрик в систему.

Функционирование в кластере kubernetes

На ускорителе ТИПр на средствах 3-х серверов развернут кластер высокой доступности при помощи программных средств Kubernetes (https://kubernetes.io/docs/tutorials/kubernetes-basics/). Предназначение кластера следующее:

1) управление и запуск контейнеров с сервисами (оркестрация контейнеров);
2) масштабирование сервисов при помощи разворачивания дополнительных экземпляров сервисов;
3) контроль состояния сервисов;
4) откат сервисов к предыдущим версиям при возникновении ошибок в работе;
5) распределение сетевой нагрузки между экземплярами сервисов, развернутых на разных серверах (балансировка нагрузки);
6) сбор сведений о состоянии серверов.

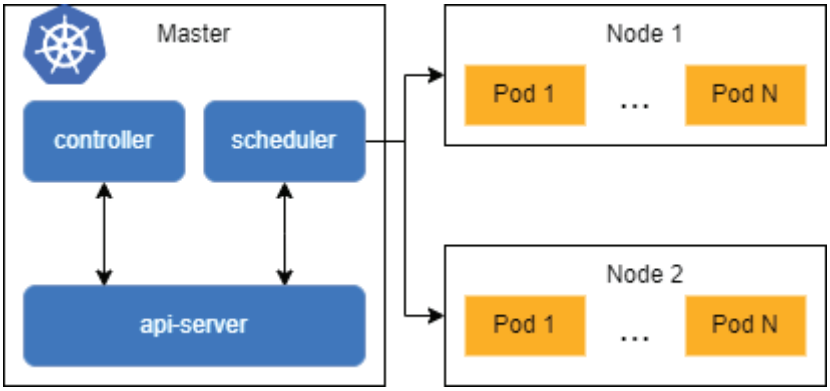

Рис. 5. Схема кластера Kubernetes, развернутого на серверах ускорителя ТИПр
Fig. 5. Kubernetes cluster schema, deployed on HIPR servers

Для работы на ускорителе ТИПр используется три сервера: один используется как управляющий (master) и два сервера – в качестве узлов кластера (node). На каждом из узлов (рис. 5) развернуты экземпляры сервисов в контейнерах (pod или «под»), управление развертыванием которых осуществляет планировщик (scheduler). Доступ к API кластера осуществляется при помощи специальной компоненты (api-server), а управление рабочими узлами и подами осуществляется при помощи набора утилит (controller) [4].

Для хранения данных на кластере также развернута файловая система Ceph (https://ceph.com/en/discover/) – программно определяемая распределенная файловая система с открытым исходным кодом, лишенная узких мест и единых точек отказа. Работу данной файловой системы можно сравнить с работой общеизвестного RAID (https://www2.eecs.berkeley.edu/Pubs/TechRpts/1987/CSD-87-391.pdf) массива, хранящего информацию распределенно сразу на множестве дисков, только в Ceph эти диски находятся на физически разнесенных машинах.

Непрерывное развертывание и интеграция

Как уже описывалось выше, одно из весомых преимуществ выбранной микросервисной архитектуры системы управления – возможность итерационно развивать подсистемы и встраивать новые сервисы. Данная возможность реализована при помощи средств автоматизации разворачивания в среде Kubernetes, а также внедрения культуры конвейерной разработки программного обеспечения, что гарантирует публикацию в продуктовую среду кластера протестированный в контейнере сервис. Процесс внедрения нового сервиса / новой версии сервиса состоит из следующих этапов:

1) разработка программного обеспечения на машине разработчика;
2) написание юнит-тестов для нового функционала;
3) локальное тестирование;
4) разворачивание и тестирование в локальном контейнере в тестовой среде;
5) упаковка приложения в контейнер, предназначенный для работы в кластере;
6) публикация контейнера в репозитории контейнеров, который может быть развернут, к примеру, на аппаратных средствах предприятия;
7) автоматическое или ручное внесение изменений в локальный конфигурационный файл, где содержатся данные о версиях контейнеров в кластере и настройке сетевого взаимодействия между сервисами;
8) публикация изменений конфигурации кластера в репозиторий;
9) вызов функции затягивания изменений конфигурации на управляющем сервере кластера, после чего, согласно указанным в конфигурационном файле настройкам, сервер сам подтянет из репозитория контейнеров нужные версии и заменит их в продуктовой среде кластера, не останавливая работу сервисов.

Важно отметить, что в момент обновления сервиса кластер сохраняет в рабочем состоянии предыдущую рабочую версию программы [4], разворачивая параллельно новую, затем балансировщик нагрузки переключает поток данных на новую версию и только после подтверждения работоспособности новой версии удаляет старый экземпляр. Таким образом осуществляется непрерывность развертывания и сохранение работоспособности системы с целостностью данных. Выглядит данный процесс довольно громоздко, однако при единоразовой настройке окружения на рабочей машине разработчика многие процессы происходят в автоматическом режиме. Поскольку, как уже отмечалось, разработка каждого сервиса проходит в автономном режиме, есть возможность применять современные средства и актуальные парадигмы при написании программного обеспечения, что позволяет создавать высокоэффективный цифровой продукт на актуализируемом стеке технологий, что также позволяет при необходимости расширять штат сотрудников, набираемых из общего рынка разработчиков, а не из узкого круга специалистов по конкретной системе управления, что в свою очередь, сильно увеличивает

жизненный цикл программного обеспечения и так называемый «фактор автобуса» (численный показатель, определяющий количество сотрудников, которых нужно исключить из процесса разработки для того, чтобы создаваемый продукт прекратил развитие/существование).

Взаимодействие с оператором

Оператор взаимодействует с системой управления посредством работы с интерфейсом web-приложения, которое развернуто в виде сервиса на кластере. Таким образом, рабочей станцией может являться любое устройство в локальной сети кластера, способное запустить браузер или специально разработанный клиент (являющийся по своей сути браузером, но с несколько большими возможностями в контексте работы с системой операторского контроля и управления). Предоставление web-интерфейса снижает расходы на обслуживание рабочих станций операторов, поскольку обновление клиентов, запущенных у операторов, происходит централизованно. Web-приложение реализовано с помощью библиотеки ReactJS (https://react.dev/learn) – современного средства разработки пользовательских интерфейсов, основанного на работе с virtual DOM (https://ru.legacy.reactjs.org/docs/faq-internals.html) – методе отрисовки только изменяющихся частей интерфейса, что повышает скорость отрисовки данных о метриках и повышает отзывчивость интерфейса. Пользовательское приложение позволяет работать с такими средствами, как:

1) интерактивная карта оборудования;
2) средства визуализации Grafana (https://grafana.com/grafana/);
3) набор интерфейсов для редактирования конфигурации оборудования;
4) набор интерфейсов для выдачи управляющих команд на устройства и узлы-агрегаторы устройств;
5) конструктор запросов к базе оперативных данных и множество более мелких интерфейсов для различных задач.

Конструктор запросов полезен при тестировании нового оборудования, конструировании собственных панелей отображения метрик, где могут быть представлены результаты измерений сразу со множества устройств на одном экране. При помощи описанных средств предоставляемый пользователям интерфейс может быть настроен для отображения подробных метрик в различных сочетаниях, что позволяет работать с единым интерфейсом всем операторам, задействованным при работах на различных узлах/подсистемах ускорителя или иной установки. При всем вышеперечисленном, при минимальном опыте работы с таким языком программирования, как, например, Python, имеется возможность разработать собственный сервис для, к примеру, обработки специфическим образом архивных данных, для чего в GARNET существует сервис электронной документации по API системы и разрабатывается сервис генерации шаблонов программного обеспечения для специфической обработки данных.

Дальнейшее развитие

Как было отмечено в начале данной статьи, одним из актуальных направлений развития создаваемой системы является внедрение полноценной поддержки Protobuf, что позволит большие массивы данных, такие как осциллограммы, передавать в сжатом формате, что снизит нагрузку на сеть при большом количестве подобных пакетов данных. Другим актуальным направлением является внедрение практики микрофронтенда (micro frontend) (https://martinfowler.com/articles/micro-frontends.html) – практики, при которой web-приложение для оператора создается не единой программой, а «слоями», отрисовываемыми в едином пространстве, данный подход призван обеспечить модульность в первую очередь разработки интерфейсов, так как для некоторых систем могут потребоваться очень специфичные инструменты отображения. Помимо этого, на данный момент ведутся работы по встраиванию колоночной базы данных ClickHouse,

которая ориентирована на выгрузку больших объемов данных за длительные сроки. Для более удобного разворачивания сервисов уже встроен инструмент Helm (https://helm.sh/docs/), позволяющий взаимодействовать с репозиториями сервисов, будто это привычные пакеты для приложений в Linux-системах или MacOS. Также в ближайшей перспективе – добавление поддержки стандартных протоколов встраиваемых систем, таких как MQTT (https://mqtt.org/mqtt-specification/) и XMPP (http://book.itep.ru/4/45/xmpp.htm), что позволит подключать подобные устройства напрямую в локальную сеть кластера «из коробки», т. е. без написания каких-либо программных обвязок. Для опытных пользователей уже добавлен сервис электронной документации, планируется в дальнейшем поддержка средств Jupyter (https://docs.jupyter.org/en/latest/) – Python фреймворка для анализа данных. Также ведется доработка сервиса построения отчетности сразу в формате для офисных программ для дальнейшего оформления отчетности о проделанной работе в ходе экспериментов.

Заключение

Прототип системы операторского управления и контроля GARNET работает и активно развивается, чему способствуют современные практики разработки программного обеспечения и актуальный набор технологий. Контейнеризация сервисов обеспечивает платформонезависимость и может запускаться на любой операционной системе, где есть поддержка виртуализации и контейнеров. Таковыми являются все современные операционные системы, применяемые на серверах и персональных компьютерах, включая AstraLinux (https://astralinux.ru/), что особенно актуально при работе в российских компаниях. Автономность сервисов позволяет в любой момент изменить реализацию и набор используемых средств любого из них без потери работоспособности системы. Автоматизация разворачивания системы делает комфортным и быстрым начало работы с GARNET: достаточно одного конфигурационного файла. Использование актуальных технологий позволяет расширять/набирать штат сотрудников из специалистов, представленных на общем рынке труда, а не искать редких разработчиков конкретной системы. Использование web-приложения для отображения интерфейса снижает накладные расходы на рабочие станции для операторов: в качестве рабочих станций мониторинга можно использовать даже мобильные устройства, подключенные к локальной сети кластера. Дальнейшее развитие системы призвано кратно увеличить функционал системы и номенклатуру устройств, работающих с создаваемой системой без предварительной настройки.

Список литературы

1. **Fedin P. A.** et al. Simulation Experiments at the Heavy Ion Accelerator HIPr // Physics of Atomic Nuclei. 2022. № 85, Suppl. 2. P. S50–S54.
2. **Ньюмен С.** Создание микросервисов. СПб.: Питер, 2016. 304 с.
3. **Ньюмен С.** От монолита к микросервисам. СПб.: БХВ-Петербург, 2021. 272 с.
4. **Ричардсон К.** Микросервисы. Паттерны разработки и рефакторинга. СПб.: Питер, 2019. 544 с.

References

1. **Fedin P. A.** et al. Simulation Experiments at the Heavy Ion Accelerator HIPr. *Physics of Atomic Nuclei*, 2022, vol. 85, suppl. 2, pp. S50–S54.
2. **Newman S.** Building Microservices. Saint Petersburg, Piter publ., 2016, 304 p.
3. **Newman S.** Monolith to Microservices. Saint Petersburg, BHV-Peterburg publ., 2021, 272 p.
4. **Richardson C.** Microservices Patterns. Saint Petersburg, Piter publ., 544 p.

Сведения об авторах

Саратовских Михаил Станиславович, инженер

Зимин Александр Николаевич, инженер-физик

Саратовских Евгения Сергеевна, инженер-физик

Гладков Владимир Михайлович, инженер-физик

Орлов Андрей Юрьевич, инженер-физик

Федин Петр Алексеевич, младший научный сотрудник

Кулевой Тимур Вячеславович, доктор технических наук

Information about the Authors

Mikhail S. Saratovskikh, Engineer

Alexander N. Zimin, Engineer-Physicist

Evgeniya S. Saratovskikh, Engineer-Physicist

Vladimir M. Gladkov, Engineer-Physicist

Andrey Yu. Orlov, Engineer-Physicist

Petr A. Fedin, Junior Researcher

Timur V. Kulevoy, Doctor of Technical Sciences

Статья поступила в редакцию 11.09.2023; одобрена после рецензирования 14.09.2023; принята к публикации 16.02.2024

The article was submitted 11.09.2023; approved after reviewing 14.09.2023; accepted for publication 16.02.2024