

РАЗРАБОТКА АРХИТЕКТУРЫ МИКРОСЕРВИСОВ ДЛЯ ОНЛАЙН-МОНИТОРИНГА ДАННЫХ ЭКСПЕРИМЕНТА MPD НА КОЛЛАЙДЕРЕ NICA

*А. В. Крылов^{1,*}, В. А. Крылов¹, О. В. Рогачевский¹,
С. Гнатич¹, Я. Буша¹, А. А. Мошкин¹, А. В. Бычков¹*

¹ Объединенный институт ядерных исследований, Дубна, Россия

Современные эксперименты в физике высоких энергий характеризуются большими объемами данных, требующими эффективных методов онлайн-обработки и анализа. Эксперимент MPD (Multi-Purpose Detector) на коллайдере NICA (Nuclotron-based Ion Collider fAcility) является современным международным экспериментом физики высоких энергий в Объединенном институте ядерных исследований (ОИЯИ) и готовится к запуску в 2026 г. Для работы эксперимента возникает необходимость в разработке системы мониторинга для контроля данных всех подсистем детектора (дисплей событий). Для работы дисплея событий требуется производить предварительную обработку потока данных с детекторов в режиме онлайн. Реализация данной задачи требует использования современных методов параллельных и распределенных вычислений. Представлена реализация подобной системы на основе микросервисной архитектуры для эксперимента MPD.

Modern high-energy physics experiments are characterized by large dataflow that requires efficient methods of online processing and analysis. The MPD (Multi-Purpose Detector) experiment at the NICA (Nuclotron-based Ion Collider fAcility) is a state-of-the-art international high-energy physics project at the Joint Institute for Nuclear Research (JINR) and is scheduled to start in 2026. For the operation of the experiment, it is necessary to develop a monitoring system to control the data from all detector subsystems (the event display). The event display requires preliminary processing of detector data streams in real time. The implementation of this task demands the use of modern parallel and distributed computing methods. The realization of such a system based on a microservice architecture for the MPD experiment is presented.

PACS: 07.05.Rm; 29.40.Gx

ВВЕДЕНИЕ

Центральную роль в регистрации треков заряженных частиц играет ТРС (Time Projection Chamber) — детектор, обеспечивающий трехмерную реконструкцию траекторий с высоким пространственным разреше-

* E-mail: avkrylov@jinr.ru

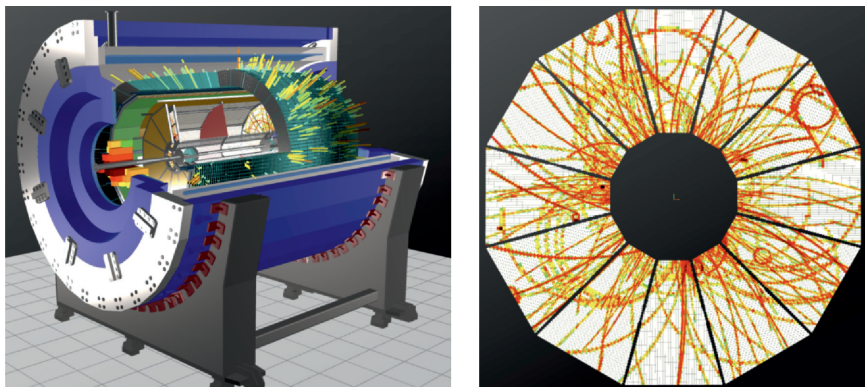


Рис. 1. Общий вид детектора MPD в дисплее событий (слева) и расположение секторов детектора TPC (справа)

нием. Система съема данных детектора TPC состоит из 24 независимых секторов (рис. 1, справа), каждый из которых включает в себя более 4000 активных сенсоров и имеет собственные модули считывания. Электроника обеспечивает дискретизацию сигналов с частотой до 10 МГц и разрядностью АЦП 10 бит, что вместе с шагом временного профиля, равным 100 нс, позволяет оценить размер данных одного события как 40 МБ [1]. В совокупности с частотой набора данных событий (~ 7 кГц согласно MPD CDR [2]) задача обработки и визуализации данных детектора TPC накладывает необходимость использования как современных технологий и библиотек, так и методов распределенных вычислений и параллелизации.

В последние годы в научных вычислениях все шире применяется подход, основанный на контейнеризации и микросервисной архитектуре. Он обеспечивает воспроизводимость, гибкость и масштабируемость вычислительных сервисов, что особенно важно для экспериментов, работающих с большими потоками данных [3]. В эксперименте ATLAS реализуется переход к использованию Kubernetes для оркестрации вычислительных узлов системы TDAQ (Trigger and Data Acquisition). В рамках подготовки к фазе Run 4 показано, что Kubernetes обеспечивает гибкое управление ресурсами и масштабируемость при обработке миллионов событий в секунду [5]. Показательным примером перехода крупной НЕР-инфраструктуры на контейнеризованную архитектуру является миграция сервисного кластера CMSWEB на Kubernetes [6], что позволило повысить устойчивость и автоматизировать процессы развертывания и обновления сервисов. Таким образом, опыт ATLAS и CMS демонстрирует, что использование Docker и Kubernetes становится стандартом для современных систем онлайн-обработки данных в физике высоких энергий.

В контексте эксперимента MPD такой подход открывает возможность распределенной обработки сигналов ТРС по независимым секторам, параллельного выполнения алгоритмов кластеризации для каждого ряда отдельно взятого сектора и агрегации результатов в единую базу данных для последующей визуализации и анализа.

ПРОБЛЕМАТИКА И ТРЕБОВАНИЯ К СИСТЕМЕ

Согласно технической документации MPD (CDR) ожидаемая скорость набора событий составляет от 1 до 7 кГц в зависимости от режима работы и интенсивности пучка [2]. При этом каждое событие детектора ТРС может содержать до 40 МБ данных. При полной загрузке системы и скорости 5–7 кГц общий поток данных может достигать десятков гигабайт в секунду, что предъявляет жесткие требования к архитектуре обработки. Традиционные монолитные системы, применявшиеся ранее в ряде экспериментов, сталкиваются с рядом ограничений:

- низкая масштабируемость: увеличение числа событий или каналов требует полной переработки архитектуры;
- сложность обновления: внедрение новых алгоритмов или модулей требует пересборки всей системы;
- отсутствие отказоустойчивости: сбой одного компонента может привести к остановке всей обработки;
- слабая адаптивность: невозможность гибко перераспределять ресурсы между задачами в зависимости от нагрузки.

В условиях реального эксперимента, где требуется параллельная обработка данных детектора, становится очевидной необходимость перехода к распределенной микросервисной архитектуре. Такой подход обеспечивает такие параметры, как:

- модульность — возможность независимой разработки, тестирования и обновления компонентов;
- масштабируемость — способность системы увеличивать производительность за счет горизонтального или вертикального расширения ресурсов;
- устойчивость к сбоям — локализация ошибок внутри отдельных сервисов, автоматический перезапуск отказавших компонентов и, как следствие, сохранение работоспособности всей системы в целом.

АРХИТЕКТУРА МИКРОСЕРВИСОВ

Для обеспечения онлайн-обработки данных ТРС в эксперименте MPD разработана распределенная микросервисная архитектура, реализующая параллельную кластеризацию сигналов по секторам и рядам. В системе используется технология контейнеризации компонентов при помощи Docker и их оркестрации инструментом Docker Compose.

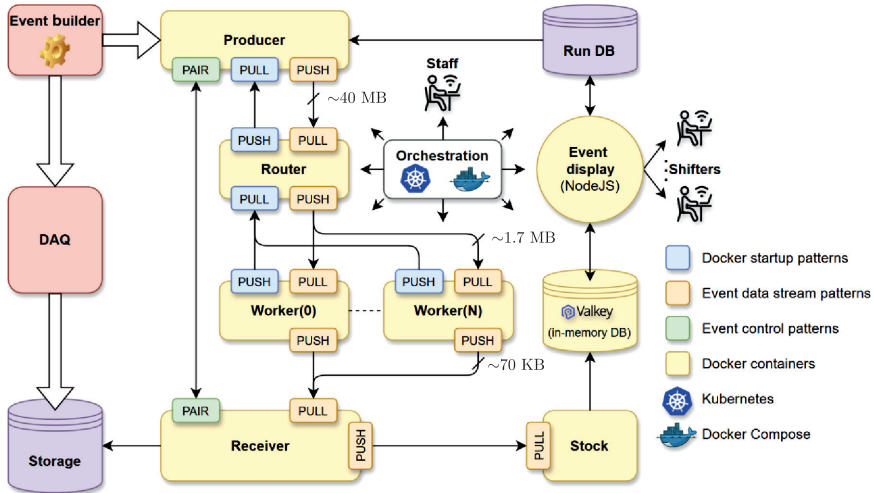


Рис. 2. Схема обмена данными между различными компонентами системы онлайн-обработки данных для детектора TPC

На рис. 2 представлена схема архитектуры, отражающая поток данных и взаимодействие компонентов.

Все компоненты реализованы как Docker-контейнеры, что обеспечивает изоляцию, воспроизводимость и удобство тестирования. Управление контейнерами осуществляется через оркестратор Docker Compose, который обеспечивает автоматическое масштабирование компонентов при росте нагрузки, перезапуск при сбоях, а также балансировку сетевых потоков между микросервисами.

Для передачи данных и иной информации между микросервисами системы используется высокопроизводительная библиотека ZeroMQ. Ее применение обусловлено необходимостью высокоскоростной передачи сообщений между различными компонентами системы с минимальными задержками (паттерн обмена PUSH/PULL), в то время как асинхронная модель передачи сообщений и встроенные методы буферизации гарантируют стабильность работы всей системы онлайн-обработки данных детектора TPC.

Для хранения промежуточных результатов используется высокопроизводительная база данных Valkey, обеспечивающая быстрый доступ к данным и поддержку ключевых операций (GET/SET, PUB/SUB). Она также служит буфером между онлайн-обработкой и системами визуализации или анализа.

Далее будут рассмотрены отдельные сервисы разработанной архитектуры.

1. Producer. Компонент Producer получает данные события после задачи Event Builder группы DAQ и выделяет сигналы, относящиеся к

различным детекторам*. Он реализован в виде Docker-контейнера на языке C++. Объем одного события может достигать ~ 40 МБ, что требует эффективной маршрутизации.

2. Router. Компонент Router принимает данные от Producer по каналу PUSH/PULL и распределяет полученные данные 24 независимых секторов по различным Workers, обеспечивая балансировку нагрузки при необходимости.

3. Workers. Компонент Worker получает данные одного сектора от Router и производит их обработку. Обработка включает в себя алгоритм быстрой кластеризации данных, позволяющий объединить сигналы физических кластеров, а также накопление необходимой статистической информации по каждому событию. В методе быстрой кластеризации реализована многопоточность по 53 независимым рядам, что позволяет обрабатывать данные параллельно. Объем выходных данных — порядка 70 кБ на событие. Каждый Worker работает как независимый Docker-контейнер, управляемый оркестратором, что обеспечивает масштабируемость и отказоустойчивость.

4. Receiver. Компонент Receiver агрегирует результат обработки данных от всех Worker-узлов. В его задачу входит синхронизация частичных данных, полученных из 24 независимых секторов TPC, и формирование целостного представления обработанного события. Для этого реализован метод очереди событий на этапе обработки: событие в очереди считается обработанным только при условии получения всех данных для 24 секторов от Worker-компонентов. После этого обработанное событие отправляется на компонент Stock.

5. Stock. Компонент Stock отвечает за передачу обработанных событий, полученных от Receiver, в высокопроизводительную базу данных Valkey и контроль за ее заполнением. Valkey используется для временного хранения событий, маршрутизации по ключам (run/event ID) и передачи данных в систему визуализации. Stock также может взаимодействовать с долговременным хранилищем (например, PostgreSQL или файловой системой).

После сохранения обработанных данных события в базе данных Valkey они становятся доступны для компонента Event Display, который извлекает события по ключам и выполняет их визуализацию [7].

ЗАКЛЮЧЕНИЕ

Разработанная система онлайн-обработки данных детектора TPC эксперимента MPD была протестирована на моделированных данных, сформированных без применения процедур фильтрации сигналов. В ходе тестов удалось воспроизвести полный цикл работы микросервисной ар-

* На текущем этапе разработки система работает только с данными детектора TPC.

хитектуры: прием событий, маршрутизацию по секторам, параллельную кластеризацию сигналов и агрегацию результатов в базе данных. Измерения показали, что средняя скорость обработки одного события в режиме Debug составляет менее 20 мс, что на сегодня позволяет подтвердить возможность использования данной распределенной архитектуры микросервисов в эксперименте MPD.

Реализация на основе Docker-контейнеров, использование ZeroMQ для обмена сообщениями, Valkey для хранения промежуточных данных, а также Docker Compose для оркестрации обеспечивают гибкость, масштабируемость и устойчивость системы. Данный подход создает основу для дальнейшей интеграции алгоритмов анализа и визуализации в единую инфраструктуру эксперимента MPD на комплексе NICA.

В дальнейших планах предусмотрена реализация полноценного мониторинга работоспособности представленной системы, включающего централизованное логирование, сбор и накопление метрик производительности. Это позволит отслеживать состояние микросервисов, своевременно выявлять отклонения в работе, анализировать нагрузку и оптимизировать параметры обработки данных по мере развития инфраструктуры эксперимента.

Финансирование. Данная работа финансировалась за счет средств бюджета института (учреждения, организации). Никаких дополнительных грантов на проведение или руководство данным конкретным исследованием получено не было.

Конфликт интересов. Авторы данной работы заявляют, что у них нет конфликта интересов.

СПИСОК ЛИТЕРАТУРЫ

1. *Averyanov A. et al.* Time Projection Chamber for the Multi-Purpose Detector at NICA: Technical Design Report. Dubna: JINR, 2019. Version: 16 Jan. 174 p.
2. *MPD Collab.* Conceptual Design Report for the Multi-Purpose Detector at NICA. Dubna: JINR, 2016. 168 p.
3. *Šimko T. et al.* Scalable Declarative HEP Analysis Workflows for Containerised Compute Clouds // Front. Big Data. 2021. V. 4. 661501.
4. *Albrecht J. et al.* A Roadmap for HEP Software and Computing R&D for the 2020s // Comput. Software Big Sci. 2019. V. 3, No. 7.
5. *ATLAS Collab.* Evolution of the ATLAS TDAQ Online Software Framework Towards Phase-II Upgrade: Use of Kubernetes // CHEP 2024 Proc. 2024.
6. *Imran M., Kuznetsov V., Dziedziewicz-Wojcik K.M. et al.* Migration of CMSWEB Cluster at CERN to Kubernetes: A Comprehensive Study. Cluster Comput. 2021. V. 24. P. 3085–3099; DOI: 10.1007/s10586-021-03325-0.
7. *Krylov A. V., Rogachevsky O. V., Krylov V. A., Bychkov A. V.* Web Based Event Display Server for MPD/NICA Experiment // Proc. 9th Intern. Conf. "Distributed Computing and Grid Technologies in Science and Education" (GRID 2021). CEUR Workshop Proc. 2021. V. 3041. P. 562–567.