

GitOps-УПРАВЛЯЕМОЕ РАСПРЕДЕЛЕННОЕ ХРАНИЛИЩЕ ДАННЫХ dCache ДЛЯ НЕЙТРИННОГО ЭКСПЕРИМЕНТА JUNO В ИНФРАСТРУКТУРЕ ОИЯИ

А. В. Баранов^{1,}, Н. А. Балашов¹, А. Н. Махалкин¹,
Р. Н. Семенов¹, И. А. Кашунин¹*

¹ Объединенный институт ядерных исследований, Дубна, Россия

Представлена распределенная система хранения данных, развернутая в Объединенном институте ядерных исследований для поддержки участия в нейтринном эксперименте JUNO. Архитектура системы хранения основана на промежуточном программном обеспечении dCache. Управление конфигурацией системы хранения реализовано по методологии GitOps: декларативное описание конфигураций, хранение в Git, контроль версий и регламентированное внедрение через GitLab CI/CD. Автоматизированное развертывание и классификация узлов осуществляется с помощью Puppet и Foreman. Система мониторинга основана на стеке Prometheus, Grafana, Loki/Alloy и Alertmanager. Представлены результаты текущей работы системы.

The scientific paper presents a distributed data storage system deployed at the Joint Institute for Nuclear Research (JINR) to support participation in the neutrino experiment (JUNO). The storage architecture is based on the dCache middleware. Storage management is implemented using the GitOps methodology: declarative configuration descriptions, storage in Git, version control, and regulated deployment via GitLab CI/CD. Automated deployment and node classification are performed using Puppet and Foreman. The monitoring stack is built on Prometheus, Grafana, Loki/Alloy, and Alertmanager. The results of the system's current operation are provided.

PACS: 89.20.Ff; 07.05.Kf

ВВЕДЕНИЕ

В международном эксперименте Jiangmen Underground Neutrino Observatory (JUNO) [1] генерируются большие объемы данных, требующие надежного хранения и доступности для участников коллаборации, распределенных по всему миру. Объединенный институт ядерных исследований (ОИЯИ) выступает одним из региональных центров обработки данных, обеспечивая прием, долговременное хранение и устойчивый

* E-mail: baranov@jinr.ru

доступ к части экспериментальных данных через несколько протоколов передачи.

Для этих целей в ОИЯИ развернуто хранилище на базе системы dCache [2], широко применяемой в научных инфраструктурах для организации распределенного доступа к данным. Архитектура dCache включает множество специализированных сервисов: управляющие узлы, шлюзы по различным протоколам, пулы хранения, базы данных и сервисы координации. Модульность обеспечивает гибкость и масштабируемость, но усложняет сопровождение и контроль изменений конфигурации.

Для упрощения администрирования внедрен подход GitOps [3], при котором инфраструктура описывается как код, а конфигурация хранится в системе контроля версий и применяется через процессы CI/CD с использованием Puppet [4] и Foreman [5]. Это обеспечивает централизованное управление, воспроизводимость конфигураций и прозрачный контроль изменений.

Настоящая работа описывает опыт эксплуатации хранилища данных JUNO в ОИЯИ с акцентом на архитектуру и организацию управления конфигурациями по практикам GitOps.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ И МЕТОДЫ

Архитектура распределенного хранилища. Система хранения обрабатывает большие объемы операций с данными (чтение и запись) и многочисленные одновременные сетевые подключения, что приводит к неоднородным нагрузкам на сервисы dCache. Опыт эксплуатации показал, что управляющие сервисы, как правило, испытывают меньшую нагрузку, чем серверы, на которых размещаются данные (пулы). Для стабильной работы управляющему сервису не требуется тот же объем ресурсов, что и пулу. Таким образом, в архитектуре системы хранения номинально выделены два класса компонентов: управляющие сервисы и пулы, для которых применяются различные способы развертывания.

Сервисы управления размещены в облачной среде [6] на восьми виртуальных машинах, что позволяет при необходимости оперативно масштабировать ресурсы центрального процессора, оперативной памяти и дисков. Список управляющих сервисов включает:

- gPlazma (выполняет аутентификацию и авторизацию доступа пользователей);
- Namespace/Chimera (публикует пространство имен и экспортирует NFS);
- Doors (реализуют сетевые конечные точки протоколов WebDAV, XRootD и GridFTP);
- SRM (управляет очередями передачи данных и резервированием дискового пространства);
- ZooKeeper (координирует состояние доменов и повышает отказоустойчивость системы).

Метаданные и учет операций пользователей хранятся в двух экземплярах базы данных PostgreSQL (Chimera и Billing).

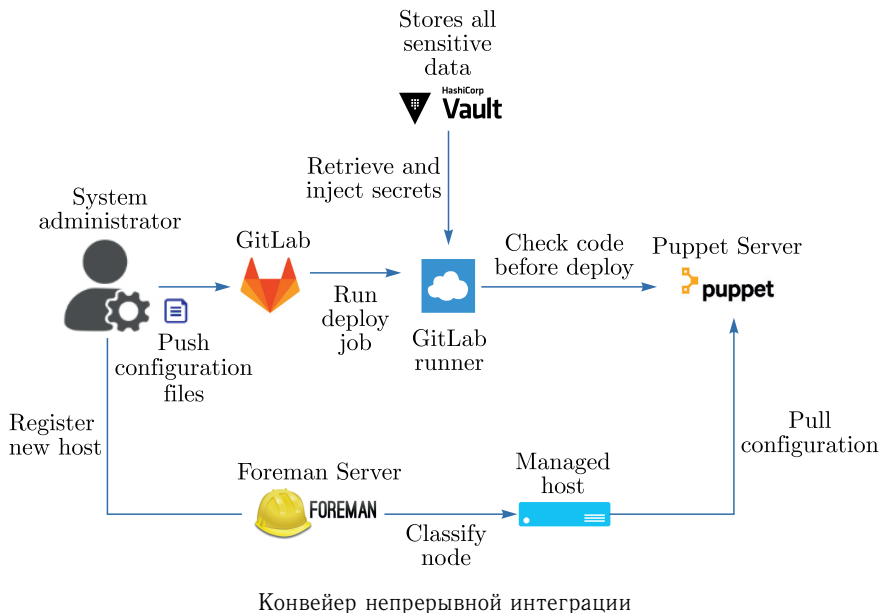
Второй класс архитектуры образуют пулы. Они размещены на физических серверах, поскольку работа под высокой нагрузкой сопровождается интенсивными дисковыми операциями ввода-вывода. Время выполнения каждой операции с данными напрямую зависит от совокупной производительности программного обеспечения и дисковой подсистемы. Поэтому пул (сервис обработки данных) должен размещаться непосредственно на физическом сервере. Управление конфигурацией каждого сервера включает установку и настройку программных RAID-массивов: RAID-1 для системного раздела и RAID-6 с двойным контролем четности для пользовательских данных (mdadm). Средний объем одного пула составляет около 162 ТБ, а суммарная эффективная емкость системы хранения достигает 1,4 ПБ.

Такое раздельное размещение компонентов позволяет горизонтально и вертикально масштабировать оба класса без простоев, сохраняя управляемость сложности конфигурации и межсервисных зависимостей благодаря применению единой методологии «инфраструктура как код» (GitOps).

Методология GitOps, CI/CD и управление конфигурацией. Для упрощения управления системой dCache применен подход GitOps — методология администрирования, при которой конфигурация инфраструктуры хранится в Git и автоматически применяется через механизмы непрерывной интеграции и доставки (CI/CD). Такой подход автоматизирует значительную часть ручных операций, обеспечивает воспроизводимость и прослеживаемость изменений, снижая риск неконтролируемых модификаций и повышая стабильность.

Основная ветка Git-репозитория (main) защищена от прямых изменений и содержит конфигурацию, применяемую в эксплуатационной среде; разработка ведется в отдельных ветках. Изменения следуют установленному регламенту: постановка задачи (Issue) → разработка в изолированном окружении и создание запроса на слияние (Merge Request) → экспертное рецензирование → слияние с защищенной веткой main. Перед слиянием автоматически запускается конвейер CI (рисунок), который в изолированной среде выполняет проверки синтаксиса и стиля манифестов Puppet и базовые тесты корректности. Во время работы конвейера конфиденциальные данные — сертификаты, пароли, SSH-ключи — подставляются из внешнего защищенного хранилища и не хранятся в Git-репозитории.

Конфигурации сервисов задаются декларативно в Puppet. Управление включает Puppet Server (источник эталонной конфигурации) и Puppet Agent, который периодически сверяет фактическое состояние узла с эталоном и автоматически устраняет расхождения. Для логической классификации узлов используется Foreman как внешний классификатор



(External Node Classifier, ENC), назначающий роли (например, «узел-доступ» или door) и профили (наборы конфигураций подсистем для данной роли). Параметры для различных ролей централизованно управляются с использованием Hiera.

Такой регламент и GitOps-подход обеспечивают воспроизводимость среды, сквозную трассируемость изменений от Issue до примененных настроек, снижение риска отказов за счет экспертного и автоматического контроля, контролируруемую работу с секретами, а также предсказуемое масштабирование и ввод новых узлов.

Мониторинг и оповещение. При эксплуатации сложных комплексов, таких как dCache, наличие системы мониторинга критично. Она позволяет выявлять сбои на ранних стадиях и оперативно обнаруживать проблемы.

Для мониторинга хранилища был применен набор широко распространенных технических решений, таких как Prometheus [7], для сбора и хранения метрик (Node Exporter — для публикации метрик на узле, где он размещен). Для агрегирования и централизованного хранения сообщений журналов применяются Loki [8] и Alloy [9]. Визуализация метрик осуществляется в Grafana, уведомления доставляет Alertmanager [10]. С учетом специфики dCache адаптирован компонент dCache-exporter для AlmaLinux 9.6 и разработаны собственные панели мониторинга. Конфигурация всей подсистемы мониторинга управляется по принципам GitOps, обеспечивая воспроизводимость и согласованность.

ЗАКЛЮЧЕНИЕ

Представлена организация управления dCache-хранилищем по принципам GitOps, целевое состояние хранится в Git-репозитории и автоматически применяется. Подход дает воспроизводимость конфигураций, трассируемость вносимых изменений и повышенную устойчивость всей системы к расхождению конфигураций. Опыт эксплуатации с применением описанного подхода показал, что он ведет к повышению стабильности и надежности работы системы.

СПИСОК ЛИТЕРАТУРЫ

1. *Abusleme A. et al. (JUNO Collab.)*. JUNO Physics and Detector // Prog. Part. Nucl. Phys. 2022. V. 123. 103927; <https://doi.org/10.1016/j.pnpnp.2021.103927>; arXiv:2104.02565 [hep-ex].
2. *Mkrtchyan T. et al.* dCache: Inter-Disciplinary Storage System // Eur. Phys. J. Web Conf. 2021. V. 251. 02010; <https://doi.org/10.1051/epjconf/202125102010>.
3. *Chittibala D.* GitOps: Revolutionizing Configuration Management in DevOps // Intern. J. Sci. Res. 2021. V. 10, No. 12. P. 1456–1462.
4. Puppet. <https://puppet.com> (accessed 17.09.2025).
5. Foreman. <https://theforeman.org> (accessed 17.09.2025).
6. *Baranov A. V., Balashov N. A., Kutovskiy N. A. et al.* JINR Cloud Infrastructure Evolution // Phys. Part. Nucl. Lett. 2016. V. 13. P. 672–675; <https://doi.org/10.1134/S1547477116050071>.
7. Prometheus. Monitoring System & Time Series Database. <https://prometheus.io> (accessed 17.09.2025).
8. Grafana Loki. <https://grafana.com/docs/loki/latest/> (accessed 17.09.2025).
9. Grafana Alloy. <https://grafana.com/docs/alloy/latest/> (accessed 17.09.2025).
10. Alertmanager. <https://prometheus.io/docs/alerting/alertmanager> (accessed 17.09.2025).