ФИЗИКА ЭЛЕМЕНТАРНЫХ ЧАСТИЦ И АТОМНОГО ЯДРА
2024. Т. 55, вып. 3. С. 553–558

ПРОТОТИП ПАКЕТНОЙ ОБРАБОТКИ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ КОНТЕЙНЕРОВ

А. В. Баранов *

Объединенный институт ядерных исследований, Дубна

В успешно эксплуатируемом дата-центре МИВК Лаборатории информационных технологий им. М. Г. Мещерякова ОИЯИ существует постоянный запрос на новые вычислительные ресурсы как со стороны экспериментов ускорительного комплекса NICA, так и со стороны установок, базирующихся на LHC (ЦЕРН). Для пакетной обработки данных используется система SLURM. В традиционной практике выполнение задачи осуществляется на физическом сервере (вычислительном узле) в установленной и настроенной операционной системе, а также в промежуточном программном обеспечении. Рассматривается реализованный прототип батч-системы с использованием контейнеризованной среды. С помощью контейнеров можно применять более одного типа ресурсов для запуска задач, а также минимизировать операционные усилия по подготовке вычислительных узлов.

The successfully operated data center of the Meshcheryakov Laboratory of Information Technologies JINR experiences a continuous demand for new computational resources, both from the experiments of the NICA accelerator complex and from facilities at the LHC (CERN). The SLURM system is used for batch data processing. In traditional practice, task execution takes place on a physical server (compute node) with an installed and configured operating system, as well as middleware. The article considers the implemented prototype of a batch system using a containerized environment. The use of containers can allow for the utilization of more than one type of resources for task execution and minimize operational efforts for preparing compute nodes.

PACS: 07.05.Kf

ВВЕДЕНИЕ

Одна из задач, возложенных на Многофункциональный информационно-вычислительный комплекс (МИВК) ЛИТ ОИЯИ [1], — обеспечение вычислительной мощностью процесса обработки данных от экспериментов, базирующихся на ускорительном комплексе NICA и LHC. В эту задачу входит распределение ресурсов, управление очередью заданий, доступ к вычислительным ресурсам и мониторинг. Для этого используется батч-система управления задачами на базе SLURM [2]. Основными

* E-mail: baranov@jinr.ru

пользователями системы являются виртуальные организации, такие как NICA, CMS, ATLAS и др. Полученные данные обрабатываются на более чем 30 000 ЦПУ-ядер, базирующихся на 950 физических серверах.

Успешная обработка данных зависит от многих факторов, один из них — качественное техническое обслуживание оборудования, включающее монтажные работы, замену неисправных комплектующих, обновление и конфигурацию программного обеспечения (ПО). Сервис LCMS [3] используется для конфигурации физических серверов, но большая часть задач выполняется персоналом. В условиях ограниченного бюджета и растущего спроса на ресурсы оптимизация использования вычислительных ресурсов и масштабируемость применяемых решений в их управлении становятся важными задачами.

Исследование направлено на создание экономичной, гибкой операционной модели, основанной на применении контейнерных технологий [4]. Такая модель может позволить исключить часть повторяющихся операций, что упростит конфигурацию ПО, и задействовать дополнительные ресурсы облачного сервиса, работающего по модели IAAS.

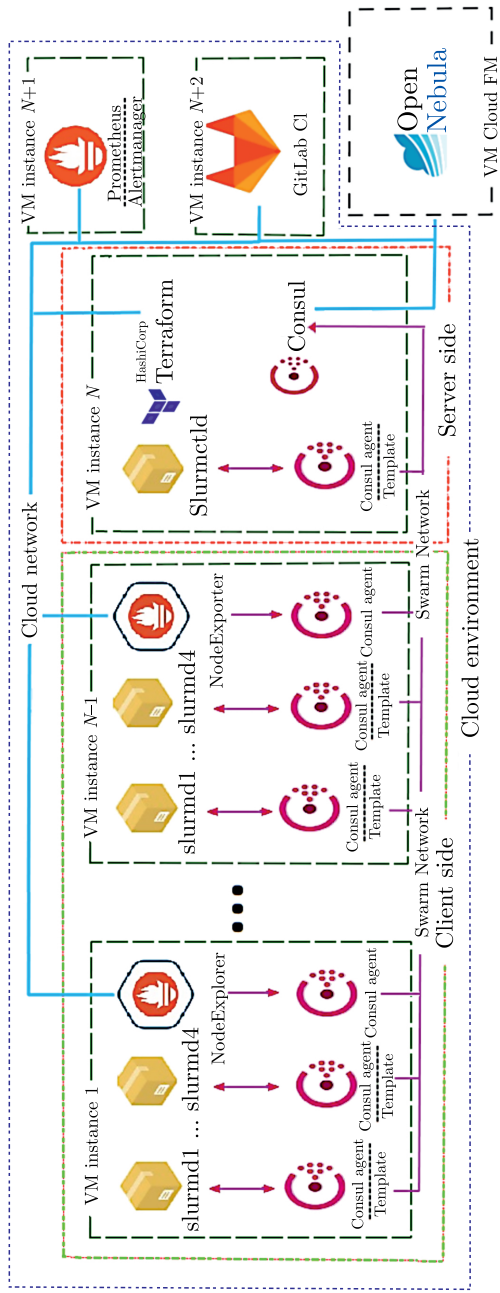
ОПИСАНИЕ ПРОТОТИПА

Разработка прототипа велась на ресурсах облачного сервиса, работающего на базе промежуточного ПО OpenNebula [5]. Прототип представляет собой набор различных сервисов, созданных на основе открытого промежуточного ПО (ППО). В перечень задействованного ППО входят:

- GitLab [6] — репозиторий для ведения разработки скриптов и конфигурационных файлов, а также для хранения собранных контейнеров с помощью механизма непрерывной сборки (CI);
- Terraform [7] — ППО, реализующее модель «инфраструктура как код», работающее на основе манифестов, описывающих запрашиваемые ресурсы в терминах провайдера;
- Prometheus [8] / Alertmanager [9] — система мониторинга и оповещения;
- HashiCorp Consul [10] — комплекс технологий, предназначенных для решения различных задач, связанных с обслуживанием сетевых сервисов.

В данной работе задействована часть функционала программы, связанного с регистрацией сервисов, необходимых для организации кластера SLURM, а также выполнения динамической конфигурации Prometheus.

Изначально в качестве контейнерной технологии был выбран Docker [11]. Выбор сделан в том числе с учетом популярности его применения во многих IT-сообществах. Docker позволяет изолировать пользовательскую среду, а использование кластеризации на основе



Структурная схема прототипа

технологии Swarm [12] позволяет объединять работающие контейнеры Docker в разных сегментах информационной сети.

Выбор перечисленного ППО основан на опыте работы автора.

Прототип разделяется на два типа конфигурации сервисов: сервер и клиент (рисунок).

Для простоты развертывания прототипа серверная и клиентская части были реализованы на двух виртуальных машинах (ВМ), работающих на облачном сервисе ОИЯИ.

В качестве операционной системы (ОС) на всех ВМ используется Scientific Linux 7.9.

Разработка серверной части ведется на ВМ, обладающей следующими характеристиками: 2 ЦПУ, 3 ГБ ОЗУ и сетевой интерфейс 10 Гбит/с. На данной машине сконфигурированы и работают сервисы Terraform, Consul, Swarm. Сервисы GitLab, Prometheus и Alertmanager ко времени создания прототипа уже находились в эксплуатации и были задействованы в разработке.

Клиентская часть представляет собой контейнер, в котором работает ряд сервисов, необходимых для авторизации и приема задач от контроллера SLURM, а также механизм регистрации и регулярной публикации статуса сервисов в Consul. Контейнер работает в составе ВМ, обладающей такими характеристиками: 5 ЦПУ, 5 ГБ ОЗУ и сетевой интерфейс 10 Гбит/с. На одной машине предполагается запуск четырех контейнеров. Ресурсы выделены из расчета 1 ЦПУ, 1 ГБ ОЗУ на один контейнер. В среде, где выполняется процесс, обслуживающий контейнер Docker, потребление памяти и ЦПУ не лимитировано по умолчанию. Потенциально может возникнуть ситуация, в которой контейнеры потребляют больше ресурсов, что может привести к снижению производительности системы в целом. Для избежания подобной ситуации указывается лимит доступных ресурсов при запуске контейнера.

СТАДИИ РАБОТЫ ПРОТОТИПА

Весь жизненный цикл контейнеризированного вычислительного узла разбит на следующие стадии: создание, обнаружение, обновление и прекращение приема новых задач.

Создание вычислительного узла. Процесс создания начинается с внесения изменений в манифест Terraform. Манифест описывает среду, в которой будет работать клиентская часть, включая сеть, диск, ЦПУ, ОЗУ и контекстуализацию ВМ. Также указывается количество экземпляров, запрашиваемых ВМ. После добавления изменений запускается задача CI в GitLab, отправляющая запрос в OpenNebula для создания ВМ.

После загрузки ОС в ВМ выполняется скрипт контекстуализации. В нем ВМ присоединяется к кластеру Swarm для доступа к виртуальной

сети overlay. Затем ВМ получает контейнер из GitLab, содержащий необходимые компоненты, SLURM и Prometheus.

Запуск каждого контейнера происходит с указанием предела доступных для него ресурсов, лимитирование выполняется для стабильной работы всех работающих контейнеров в одной ВМ. Далее следует запуск агента Prometheus для сборки метрик выполнения мониторинга, включая статус «здоровья» среды. Это важно для операции drain — остановки приема новых задач на вычислительном узле.

Обнаружение вычислительного узла. Центральной составляющей в прототипе является Consul. Он выполняет роль базы данных. Сервисы, работающие на серверной и клиентской частях, могут регистрироваться и публиковать свои статусы на основе регулярных проверок с помощью consul agent. Кроме того, в Consul записываются идентификаторы контейнеров вычислительного узла. Эти публикуемые данные применяются для внесения изменений в конфигурационный файл SLURM на всех контейнерах в кластере. Для этой цели задействован механизм consul template. Таким образом, автоматически в режиме реального времени происходит регулярное обновление количества подключенных и доступных узлов в конфигурации SLURM. Для авторизации вычислительного узла в SLURM используется стандартный сервис Munge.

С использованием Consul реализовано самостоятельное добавление агентов в конфигурацию Prometheus с помощью встроенного функционала «Обнаружение сервисов».

Обновление. В настоящее время обновление SLURM выполняется через создание и публикацию новых контейнеров для серверной и клиентской частей в GitLab. Однако этот метод имеет недостатки, такие как необходимость остановки кластера SLURM во время обновления контроллера и выполнение пересоздания вычислительных узлов в контейнерах на каждой ВМ. Рассмотрена альтернативная технология DUCC [13] для обновления через репозиторий CVMFS с использованием docker-graphdriver, но она оказалась нестабильной с текущей версией Docker из-за прекращения развития проекта docker-graphdriver.

Остановка приема новых задач (drain). В процессе работы вычислительного узла периодически выполняется операция drain. Она необходима для проведения обновления программного обеспечения и в случае обнаружения неисправности оборудования. Операцию выполняет администратор при плановых работах.

В аварийном режиме начинает действовать механизм прекращения приема новых задач на всех вычислительных узлах, работающих на «аварийном» оборудовании. Агент Prometheus использует метрику container для публикации состояния. Значения метрики 0 и 1 обозначают штатное и аварийное состояния соответственно. Метрика также содержит идентификаторы вычислительного узла, передаваемые через метки. Alertmanager отправляет сообщения с идентификаторами контейнеров

для операции drain. Обработку сообщения осуществляет специальное приложение webhook.

Скрипт обрабатывает сообщение и выполняет drain каждому узлу из полученного списка. В качестве детектора обнаружения проблем на оборудовании предлагается использовать инструмент node_problem_detector [14].

ЗАКЛЮЧЕНИЕ

В данной работе описан разработанный прототип батч-системы. Прототип основан на контейнеризации, что позволяет обеспечивать создание, мониторинг и обновление и выполнять операцию drain. Дальнейшие перспективы развития включают исследование альтернативных технологий, таких как Apptainer, и изучение возможности внедрения Bare Metal для оптимизации использования ресурсов. Применение этих технологий позволит более гибко и эффективно управлять вычислительными узлами кластера.

СПИСОК ЛИТЕРАТУРЫ

1. *Baginyan A. et al.* Current Status of the MICC: An Overview // CEUR Workshop Proc. 2021. V. 3041. P. 1–9.
2. SLURM Home Page. https://slurm.schedmd.com/documentation.html (accessed 28.08.2023).
3. *Baranov A. et al.* Lifecycle Management Service for the Compute Nodes of Tier1, Tier2 Sites (JINR) // CEUR Workshop Proc. 2021. V. 3041. P. 429–433; doi: 10.54546/MLIT.2021.38.11.001.
4. *Pahl C., Brogi A., Soldani J., Jamshidi P. et al.* Cloud Container Technologies: A State-of-the-Art Review // IEEE Trans. Cloud Comput. 2019. V. 7. P. 677–692.
5. *Baranov A. V., Balashov N. A., Kutovskiy N. A. et al.* JINR Cloud Infrastructure Evolution // Phys. Part. Nucl. Lett. 2016. V. 13, No. 672; doi: 10.1134/S1547477116050071.
6. GitLab CI Home Page. https://docs.gitlab.com/ee/ci/ (accessed 28.08.2023).
7. HashiCorp, Terraform. https://www.terraform.io (accessed 28.08.2023).
8. Prometheus Authors, Prometheus Monitoring System & Time Series Database. https://prometheus.io (accessed 28.08.2023).
9. Alertmanager. https://prometheus.io/docs/alerting/alertmanager/ (accessed 28.08.2023).
10. HashiCorp, Consul by HashiCorp. https://www.consul.io (accessed 28.08.2023).
11. Docker Home page. https://www.docker.com/ (accessed 28.08.2023).
12. Swarm, Documentation. https://docs.docker.com/engine/swarm/ (accessed 28.08.2023).
13. DUCC, Documentation. https://cvmfs.readthedocs.io/en/stable/cpt-ducc.html (accessed 28.08.2023).
14. Node Problem Detector. https://github.com/kubernetes/node-problem-detector (accessed 28.08.2023).