НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЦЕНТР
«КУРЧАТОВСКИЙ ИНСТИТУТ»
Институт физики высоких энергий имени А.А. Логунова
Национального исследовательского центра
«Курчатовский институт»

Препринт 2022–11

А.А. Котляр[1], В.В. Котляр[1]

Применение технологии контейнеризации для вычислений на центральном Linux кластере НИЦ «Курчатовский институт» - ИФВЭ

[1]НИЦ «Курчатовский институт» - ИФВЭ

Протвино 2022

УДК 539.1.074.3 М-24

Аннотация

Котляр А.А., Котляр В.В. Применение технологии контейнеризации для вычислений на центральном Linux кластере НИЦ «Курчатовский институт» - ИФВЭ: Препринт НИЦ «Курчатовский институт» – ИФВЭ 2022–11. – Протвино, 2022. – 12 с., 5 рис., библиогр.: 14.

Технология контейнеризации для систем Linux появилась много лет назад с системами OpenVZ и LCX. Но более широкое применение пришло вместе с появлением Docker системы и добавлением новых механизмов поддержки контейнеров в само ядро Linux. Зарождаясь как система для серверов, технология контейнеризации постепенно перешла в пользовательское пространство, как универсальный механизм распространения и запуска сложного программного обеспечения, в том числе для проведения расчётов и анализа данных на вычислительных кластерах или облачных сервисах. В работе описаны основные принципы и особенности применения данной технологии для организации вычислений на центральном Linux кластере НИЦ «Курчатовский институт» - ИФВЭ.

Abstract

Kotliar A.A., Kotliar V.V. Containerization technology practice for computing on the NRC «Kurchatov Institute» – IHEP central Linux cluster: NRC «Kurchatov Institute» – IHEP Preprint 2022–11. – Protvino, 2022. – p. 12, fig. 5, refs.: 14.

Containerization technology for Linux systems appeared many years ago with OpenVZ and LCX systems. However, wider use came with the advent of the Docker system and the addition of new container support mechanisms to the Linux kernel itself. Originating as a system for servers, containerization technology has gradually moved into the user space as a universal mechanism for distributing and running complex software, including calculations and analyzing data on computing clusters or cloud services. The paper describes the main principles and features of the application of this technology for organizing calculations on the IHEP central Linux cluster.

1. Введение

Технология контейнеризации для систем Linux появилась много лет назад с системами OpenVZ [1] и LCX [2]. Но более широкое применение пришло вместе с появлением Docker системы [3] с её сообществом по обмену готовыми решениями контейнеров. Благодаря развитию этих систем, в основное ядро Linux были добавлены новые механизмы поддержки контейнеров, намного облегчающие создание и контроль за их выполнением, а также обеспечивающие безопасность всей системы в целом. Зародившись как технология изоляции программного обеспечения (ПО) для серверов, контейнеризация постепенно распространилась на задачи создания, тестирования и распространения ПО для DevOps методики [4], парадигмы построения микро сервисной архитектуры на базе систем управления контейнерами, такими как Kubernetes [5], а также на пользовательское пространство, как универсальный механизм распространения и запуска сложного программного обеспечения. Данная технология начала применятся для проведения расчётов и анализа данных на вычислительных кластерах или облачных сервисах и поддерживается большинством систем организации цепочек вычислений на базе стандарта общего языка рабочих процессов (CWL - The Common Workflow Language) [6].

В связи с тем, что архитектура и используемое системное и прикладное ПО на каждом вычислительном кластере уникальны, существует проблема адаптации процесса запуска контейнеров под определенное окружение. Так для центрального Linux кластера НИЦ «Курчатовский институт» - ИФВЭ проделана работа по адаптации нескольких систем запуска контейнеров. Это позволяет осуществлять базовые операции по созданию и запуску контейнеров в пространстве имен пользователей для применяемой на кластере системы запуска задач. Основная специфика работы кластера, это совместное использование вычислительных узлов для расчётов в системе Грид для Большого Адронного Коллайдера (БАК) и экспери-

ментов Института в области Физики Высоких Энергий (ФВЭ). В данном случае на первое место выходит один из основополагающих принципов использования контейнеров по их изоляции от другого программного обеспечения и друг от друга без снижения производительности.

2. Вычислительный кластер НИЦ «Курчатовский институт» - ИФВЭ

В настоящий момент Linux кластер НИЦ «Курчатовский институт» - ИФВЭ представляет собой вычислительный кластер общего назначения для распределённых вычислений. Компьютерные мощности через систему очередей используются совместно для Грид БАК и локальных экспериментов ФВЭ. Распределение нагрузки происходит на программном уровне, и на одном и том же счётном узле запускаются задачи различных типов. Обычно распределённый характер вычислений основан на разделении обрабатываемых данных на логические блоки, что позволяет упростить распределенную обработку и максимально эффективно задействовать доступные ресурсы. Адаптированная под контейнеры архитектура кластера представлена на рис.1

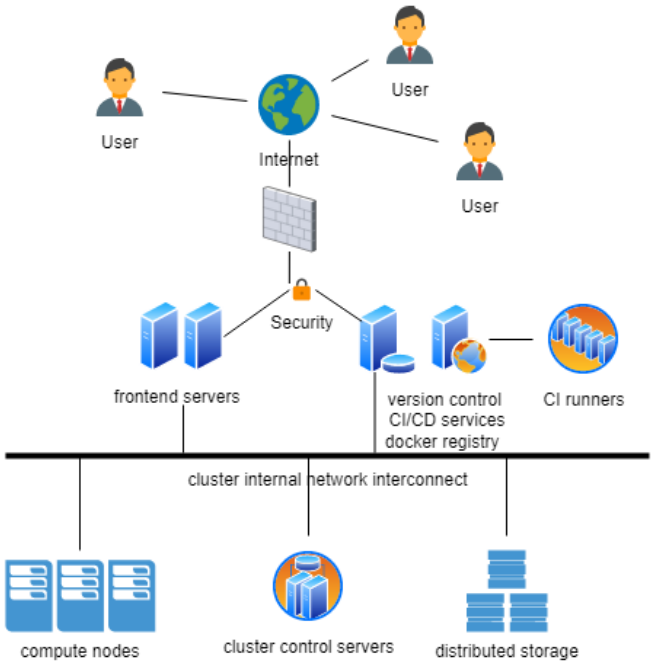

Рис. 1. Архитектура вычислительного кластера НИЦ «Курчатовский институт» - ИФВЭ для работы с контейнерами.

Из системы Интернет пользователи имеют доступ на интерактивные серверы работы с кластером (frontend), а также на систему управления версиями ПО, создания цепочек для непрерывной интеграции (Continuous Integration – CI) и непрерывного развертывания (Continuous Deployment – CD). Отдельно обеспечивается доступ к хранилищу контейнеров в формате Docker, который является одним из стандартных способов их представления и использования. CI/CD представлены отдельным набором виртуальных машин, доступных всем пользователям (shared runners). Для вычислений пользователям доступны ресурсы в составе: около 150 расчётных серверов (~ 3000 процессорных потоков), распределённой отказоустойчивой системы хранения объемом 300ТБ, сетевыми подключениями Интернет с пропускной способностью 2x10Гб/с. Базовая операционная система для вычислительных узлов CentOS7 x86_64. Также доступны расчёты на графических картах, ПО Ansys и Mathematica. Для работы в Грид БАК установлено промежуточное ПО UMD4 [7].

Функционирование инфраструктуры обеспечивается в режиме 24x7 с минимальным временем простоя из-за применяемых техник повышенной отказоустойчивости отдельных сервисов.

3. Базовые принципы использования контейнеров для вычислений

Очень часто сложное программное обеспечение, которое создаётся для выполнения вычислений, анализа и обработки данных исследований, имеет общий набор проблем, вытекающих из специфики его создателей – это прежде всего учёные, а на втором месте программисты. Главное для учёного - закончить быстро одно исследование, написать статью и идти дальше к следующему исследованию. Обычно такому ПО присущи следующие проблемы: наличие багов, слабая документированность кода, сложность сборки и запуска на других компьютерах, неустойчивость к изменениям в коде или среде выполнения, медленное или полное отсутствие обновлений и правок кода, а также очень часто пренебрежение принципами информационной безопасности. Порой практически невозможно повторить расчёты по статьям, используя приложенный программный код. Справиться с частью проблем помогает применение техник из DevOps [4] таких как:

- Управление версиями ПО (Version Control) через центральный репозиторий кода;

- Непрерывное развертывание (Continuous Integration) с непрерывным слиянием разрабатываемого программного кода с центральным репозиторием;
- Автоматическое тестирование (Automated Testing) изменений в программном коде на регулярной основе для раннего обнаружения проблем;
- Непрерывное развертывание (Continuous Delivery) в автоматическом режиме готового к применению программного обеспечения, которое прошло тестирование в предыдущем шаге;
- Тщательное описание инфраструктуры с применением методик управления версиями на языке какой-либо из систем управления вычислительной инфраструктурой (Infrastructure as Code), что должно позволять выполнять разработанное программное обеспечение.

Для решения оставшихся проблем подходит технология контейнеризации, которая тесно переплетается с DevOps техниками, описанными выше. Контейнеры концептуально напоминают виртуальные машины с меньшей изоляцией и меньшими накладными расходами в плане использования вычислительных ресурсов. Для применения в вычислениях они помогают обеспечить:

- Инкапсуляцию всего необходимого программного обеспечения для эксперимента в целом, исключая какие-то скрытые не описанные зависимости от внешнего ПО;
- Установку уже подготовленного программного стека на вычислительном узле;
- Изоляцию выполняемого кода от воздействия других задач и процессов на одном вычислительном узле;
- Распределение подготовленного окружения на любое число вычислительных узлов, а также в облачных средах или других вычислительных кластерах;
- Управление вычислением отдельных задач в контейнерах через системы планирования запуска контейнеров, системы запуска распределённых задач или системы создания и запуска потоков задач.

В настоящий момент повсеместно распространённым стандартом промышленного использования контейнеров de-facto является система Docker [3]. Это проект с открытым программным обеспечением для создания, распространения и запуска любых приложений как легковесных контейнеров. Для изоляции процессов Docker использует следующие техноло-

гии Linux: пространство имён (namespaces), контрольные группы (cgroups) и объединяющие файловые системы (unioning file systems - UFS). Образы контейнеров напоминают виртуальные машины, но они совместно используют одно ядро Linux с основной системой. Это позволяет легко устанавливать и запускать контейнеры, во много раз быстрее чем виртуальные машины, а также запускать несколько сотен таких контейнеров на обычном компьютере. Дополнительно механизм cgroups делает возможным изоляцию сетевого доступа на уровне контейнеров. Для работы с файлами Docker применяет файловую систему, состоящую из нескольких слоёв, построенную на базе одной из UFS. Каждый последующий слой файловой системы подключается поверх предыдущих. Самый первый слой, получивший название основного образа (base image), обычно представляет собой набор файлов и папок, составленный из базового набора одного из Linux дистрибутивов (RHEL, Debian, Fedora и др.) не обязательно совпадающем с основной операционной системой компьютера. Каждый слой, кроме самого верхнего, можно только читать без возможности записи. Особенностью файловых слоёв является хранение только отличий в файлах верхнего слоя по сравнению с предыдущим нижним слоем. Все слои индексируются по хэш-функции sha256 [8] и могут использоваться совместно разными контейнерами, что позволяет уменьшить требования к необходимому для запуска дисковому пространству и увеличить скорость запуска задач.

Учитывая всё вышесказанное, весь процесс подготовки к запуску и непосредственно запуск контейнера можно разбить на следующие базовые шаги (далее, для упрощения, используется нотация в Docker командах):

- Необходимо найти и выбрать базовый образ для контейнера, обычно доступный в одном из мест размещения образов готовых контейнеров, например, в Docker Hub [9] - ‘docker search ubuntu’;
- Скопировать базовый образ на локальную систему - ‘docker pull ubuntu`;
- Модифицировать базовый образ для создания своего, если необходимо
    - запустить контейнер – ‘docker run –it ubuntu bash’;
    - выполнить изменения внутри контейнера – ‘apt install vim’
    - остановить контейнер – ‘exit’
    - выполнить операцию закрытия слоя контейнера ‘docker commit HASH ubuntu/with_vim’

- Запустить контейнер в интерактивном или асинхронном (detached) режиме – ‘docker run –d ubuntu’;
- Если контейнер запущен в detached режиме, к его консоли всегда можно подключиться командой `docker attach HASH’.

Шаги по созданию необходимого образа можно выполнить с помощью механизма использования Dockerfile [10], когда в файле специального формата указывается последовательность команд, по созданию образа контейнера и выполняется команда ‘docker build’, которая его читает и выполняет.

4. Применение CI механизма для создания образов контейнеров для Linux кластера НИЦ «Курчатовский институт» - ИФВЭ

Как видно, процесс создания необходимого образа контейнера является одной из самых фундаментальных задач, на которой основан непосредственно процесс запуска контейнерных вычислений. Для решения данной задачи на Linux кластере НИЦ «Курчатовский институт» - ИФВЭ предполагается использовать один из сервисов ИВЦ, предназначенный для организации совместной работы над программным обеспечением и удовлетворяющий всем современным требованиям к такой разработке. В основу создания образа контейнера взят один из принципов DevOps – принцип непрерывной интеграции (CI). Сам процесс состоит не в создании образа контейнера, как в конечном результате, а в описании инструкций и подготовки необходимого окружения, итогом которого является готовый образ контейнера в Docker формате. Пример такого проекта представлен на рис. 2. и доступен пользователям ИФВЭ [10]. Он представляет собой базовый набор для такого рода вычислений:

- Существует какая-то программа для вычислений (Pi_count.c);
- Для сборки программы существует набор инструкций (Makefile);
- Для задействования CI создается файл с описанием состояний (.gitlab-ci.yml);
- Для создания образа контейнера подготавливается файл сценария (Dockerfile);
- В качестве документации к проекту по его использованию создаётся файл с описанием (README.md)

Name	Last commit	Last update
.gitlab-ci.yml	Add support docker image build for CI	1 week ago
Dockerfile	Add support docker image build for CI	1 week ago
Makefile	Add support docker image build for CI	1 week ago
Pi_count.c	Add support docker image build for CI	1 week ago
README.md	Update README.md	1 week ago

Рис. 2. Проект для организации контейнерных вычислений.

Как видно, в данной парадигме присутствует ориентированность на описание процесса, что позволяет избежать многих ошибок при создании и поддержке такого рода проектов. Описанная в CI логика представляет собой файл состояний на языке YAML [11], с очень простым алгоритмом (рис. 3)

Рис. 3. Описание CI для создания образа контейнера для вычислений.

CI настраивается так, чтобы он запускался после любого изменения программного обеспечения. В итоге, при отсутствии ошибок в сборке, в системе всегда имеется готовый к запуску образ контейнера, к которому можно применять разные схемы имён – выставления тэгов для облегчения его использования на кластере.

5. Запуск вычислительных контейнеров на Linux кластере

Запуск вычислительных контейнеров является специфической задачей для каждой компьютерной инфраструктуры и зависит от многих факторов: используемых операционных систем, используемых систем управления вычислительными узлами, используемых систем хранения, сетевой связности инфраструктуры, а также применяемых политик безопасности к выполняемому программному обеспечению. На вычислительном кластере НИЦ «Курчатовский институт» - ИФВЭ внедрены две системы управления контейнерами Apptainer/Singularity [12] и enroot [13]. Обе системы примерно одного класса со своими особенностями, которые могут влиять на выбор пользователей. Так Apptainer/Singularity широко используется в Грид БАК и в настоящий момент является основным стандартом запуска вычислений в распределённых системах на базе контейнеров в Физике Высоких Энергий (ФВЭ). Enroot является специализированной системой запуска контейнеров для вычислений на больших и сверхбольших кластерах с прямой поддержкой доступа к современным графическим ускорителям и прямой поддержкой широко распространённой системой управления кластерами - Slurm [14]. С точки зрения обычного пользователя Linux кластера НИЦ «Курчатовский институт» - ИФВЭ там, где нужен быстрый запуск большого числа задач, рекомендуется использовать enroot; там, где необходима универсальность и совместимость с другими контейнерами ФВЭ - Apptainer/Singularity.

Обе системы на кластере НИЦ «Курчатовский институт» - ИФВЭ работают в пользовательском пространстве и могут использоваться для запуска ранее подготовленного Docker контейнера. Процедура запуска в таком случае выглядит одинаково и похожа на общую процедуру запуска любого контейнера:

- Выполняется чтение образа контейнера из репозитория и преобразование его в свой промежуточный формат sif для Apptainer/Singularity, squashfs для enroot;
- Выполняется непосредственный запуск контейнера для вычислений, используя параметры по умолчанию или с дополнительными параметрами (обычно параметры внутрь контейнеров передают через переменные окружения);
- Если необходимо выполнить одновременный запуск большого числа контейнеров, то используются стандартные средства распределённого запуска задач на кластер, где в качестве программы указывается контейнер.

Пример работы на кластере для Apptainer/Singularity приведён на рис. 4.

Рис. 4. Пример запуска контейнера на Linux кластере НИЦ «Курчатовский институт» - ИФВЭ.

На двух интерактивных серверах вычислительного кластера НИЦ «Курчатовский институт» - ИФВЭ проведены исследования времени запуска простого контейнера. Они показали неоспоримый выигрыш в скорости работы системы поддержки контейнеров enroot (рис. 5). В экспериментах среднее значение времени запуска контейнера зависит от загрузки и особенностей самого сервера, где запускаются контейнеры, но даже с учётом таких особенностей разница между enroot и Apptainer/Singularity составляет значимое время, которое необходимо учитывать при планировании запуска большого числа короткоживущих контейнеров.

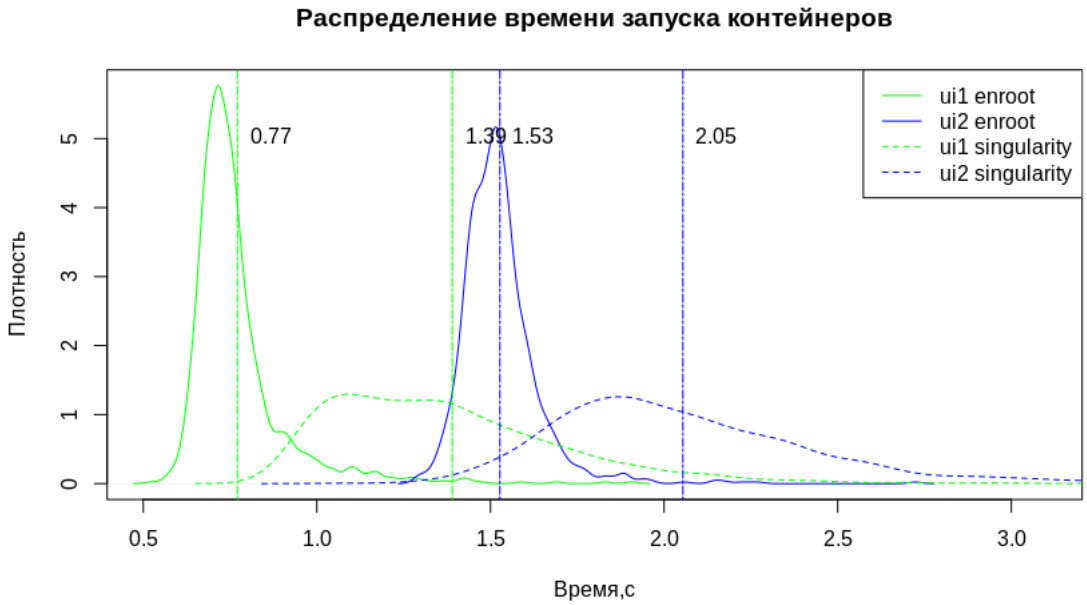

Рис. 5. Распределение времени запуска контейнеров для разных систем контейнеризации на Linux кластере НИЦ «Курчатовский институт» - ИФВЭ.

6. Заключение

Использование технологии контейнеризации для организации вычислений совместно с практиками из DevOps методов, позволяют заметно упростить написание, подготовку исполняемого окружения, запуск, последующую поддержку и модернизацию сложного программного обеспечения. На вычислительном Linux кластере НИЦ «Курчатовский институт» - ИФВЭ адаптированы две независимые системы запуска контейнеров для вычислений, которые совместно с уже применяемой системой организации распределённых вычислений на базе системы очередей позволяют легко запускать вычислительные контейнеры на ресурсах всего кластера. Связующим звеном подготовки контейнеров является сервис ИВЦ ИФВЭ, позволяющий организовывать цепочки непрерывной разработки, а также может использоваться для хранения и распространения уже готовых образов контейнеров среди других научных центров.

Применение контейнеров совместно с обычными задачами пользователей кластера и задачами Грид БАК на одних и тех же вычислительных узлах повышают общую эффектив-

ность использования компьютерных ресурсов Linux кластера НИЦ «Курчатовский институт» - ИФВЭ для вычислительных задач.

Список литературы

[1] Open source container-based virtualization for Linux, [OpenVZ], https://openvz.org/, доступ 29.11.2022.

[2] Linux container runtime that consists of tools, templates, and library and language bindings, [LCX], https://linuxcontainers.org/, доступ 29.11.2022.

[3] A platform designed to help developers build, share, and run modern applications, [Docker], https://www.docker.com/, доступ 29.11.2022.

[4] Ознакомительное описание DevOps методологии, [DevOps], https://ru.wikipedia.org/wiki/DevOps, доступ 29.11.2022.

[5] An open-source system for automating deployment, scaling, and management of containerized applications, [Kubernetes], https://kubernetes.io/, доступ 29.11.2022.

[6] An open standard for describing how to run command line tools and connect them to create workflows, [CWL], https://www.commonwl.org/, доступ 29.11.2022.

[7] Unified Middleware Distribution is a software for deployment services in Grid, [UMD], https://wiki.egi.eu/wiki/Middleware, доступ 29.11.2022.

[8] Ознакомительное описание алгоритмов хэширования, [sha256], https://ru.wikipedia.org/wiki/SHA-2, доступ 29.11.2022.

[9] The world's largest library and community for container images, [Docker Hub], https://hub.docker.com/, доступ 29.11.2022.

[10] Базовый проект для применения технологии контейнеризации на вычислительном кластере НИЦ «Курчатовский институт» - ИФВЭ, https://glab.ihep.su/annette/containerization-technology-practice, доступ 29.11.2022.

[11] A human-friendly data serialization language for all programming languages, [YAML], https://yaml.org/, доступ 29.11.2022.

[12] The container system for secure high performance computing, [Apptainer/Singularity], https://apptainer.org/, доступ 29.11.2022.

[13] A simple, yet powerful tool to turn traditional container/OS images into unprivileged sandboxes, [enroot], https://github.com/NVIDIA/enroot, доступ 29.11.2022.

[14] An open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters, [Slurm], доступ 29.11.2022.

Рукопись поступила 6 декабря 2022 г.

А.А. Котляр, В.В. Котляр
Применение технологии контейнеризации для вычислений на центральном Linux кластере НИЦ «Курчатовский институт» - ИФВЭ.

Препринт отпечатан с оригинала-макета, подготовленного авторами.

Подписано к печати 09.12.2022 Формат 60 × 84/16. Цифровая печать.
Печ.л. 1. Уч.– изд.л. 1,3. Тираж 60. Заказ 12. Индекс 3649.

НИЦ «Курчатовский институт» – ИФВЭ
142281, Московская область, г. Протвино, пл. Науки, 1

www.ihep.ru; библиотека http://web.ihep.su/library/pubs/all-w.htm

Индекс 3649

П Р Е П Р И Н Т 2022-11,
НИЦ «Курчатовский институт» – ИФВЭ, 2022