

Применение контейнерной виртуализации для реализации параллельных вычислений на суперкомпьютерных кластерах

© 2026 Е.А. Киселёв¹, А.В. Яровой^{1✉}

¹ НИЦ «Курчатовский институт», г. Москва, 123182, Россия

Ссылка для цитирования

Киселёв Е.А., Яровой А.В. Применение контейнерной виртуализации для реализации параллельных вычислений на суперкомпьютерных кластерах // Программные продукты и системы. 2026. Т. 39. № 1. С. 058–069. doi: 10.15827/0236-235X.153.058-069

Информация о статье

Группа специальностей ВАК: 2.3.5

Поступила в редакцию: 15.07.2025

После доработки: 12.08.2025

Принята к публикации: 20.08.2025

Аннотация. Предметом представленного в статье исследования является применение технологий контейнерной виртуализации в суперкомпьютерных системах. Авторами проведено сравнение наиболее распространенных в области высокопроизводительных вычислений: Docker, Singularity/Apptainer и Podman. Проанализированы преимущества и недостатки каждой технологии. Как результат анализа предложено два варианта использования технологий контейнерной виртуализации совместно с системой управления прохождением параллельных заданий (СУППЗ), применяемой в отечественных суперкомпьютерных центрах. Первый вариант основан на использовании контейнеров Singularity под управлением данной системы и без использования дополнительных средств оркестрации. Разработаны порядок создания и запуска заданий пользователей в виде контейнеров Singularity и программное средство автоматизации формирования и развертывания контейнеров Singularity. Второй вариант подразумевает использование контейнеров Podman совместно с системой оркестрации Kubernetes для автоматизации развертывания виртуальной высокопроизводительной вычислительной инфраструктуры под управлением СУППЗ. Предложен макет масштабируемой платформы для выполнения параллельных вычислений, внутри которого создается набор виртуальных кластеров под управлением СУППЗ с отдельными точками входа для различных групп пользователей, рассмотрены основные этапы построения макета. Предложен метод масштабирования решаемого поля виртуальной вычислительной системы, использующий функциональные возможности Kubernetes и СУППЗ. Разработан алгоритм работы программного средства автоматического масштабирования рабочих узлов в виртуальном вычислительном кластере. Результаты исследования на практике могут быть применены в образовательном процессе на дисциплинах, связанных с параллельными вычислениями, а также при проведении научно-исследовательских работ в области высокопроизводительных вычислений.

Ключевые слова: технологии контейнерной виртуализации, высокопроизводительные вычисления, суперкомпьютерные системы, Docker, Singularity, Kubernetes, СУППЗ

Благодарности. Работа выполнена в рамках государственного задания НИЦ «Курчатовский институт»

Введение. Суперкомпьютеры играют важную роль в решении сложных научных и инженерных задач. Благодаря высокой производительности такие системы позволяют выполнять вычисления, которые были бы невозможны на персональных компьютерах. Для обеспечения оптимальных показателей производительности и отказоустойчивости на состав ПО суперкомпьютера накладываются дополнительные ограничения. В частности, пользователям не всегда доступны последние версии системного ПО и библиотек, а настройка необходимого программного окружения может требовать устранения большого количества зависимостей и установки отсутствующего ПО исключительно в своей домашней директории. Некоторые программные средства и вовсе невозможно установить из-за отсутствия необходимых прав доступа. Вариантом решения данной проблемы является использование технологии контейнеризации.

В классическом варианте построения высокопроизводительных систем кластерного типа за управление вычислительными ресурсами и запуск заданий пользователей отвечает *система управления заданиями (СУЗ)*. Примерами являются Slurm [1] и *система управления прохождением параллельных заданий (СУППЗ)* [2], которые обеспечивают коллективный доступ пользователей к суперкомпьютеру, принимают входной поток различных заданий от разных пользователей, планируют их очереди, выделяют необходимые для выполнения задания вычислительные ресурсы и освобождают их после завершения [3]. Задание пользователя представляет собой исполняемый файл с описанием параметров его запуска на вычислительной системе (количество выделяемых вычислительных ядер для запуска, длительность выполнения, имя и аргументы исполняемой программы).

Существует ряд ограничений при работе с суперкомпьютерными системами:

- возможность изменения стека системного ПО только в исключительных случаях (формируется при создании суперкомпьютера);
- ограниченный доступ пользователей к системе [4];
- изоляция пользователей и их процессов друг от друга;
- требование к отсутствию пользовательских данных и настроек предыдущих запусков заданий на вычислительных узлах решающего поля (вычислительный узел – атомарный элемент выделенных для задания ресурсов; вся совокупность таких узлов СУЗ называется решающим полем).

Одним из способов преодоления указанных ограничений является использование контейнеризации, или виртуализации, – одной из ключевых технологий, активно применяемых разработчиками на всех этапах жизненного цикла ПО [5–7]. Контейнеризация на уровне операционных систем – метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пользовательского пространства на одном вычислительном узле. В сравнении с гипервизорной виртуализацией контейнеризация обладает меньшими накладными расходами, благодаря чему позволяет достичь больших производительности, скорости развертывания [8, 9], масштабируемости и плотности размещения пользовательских процессов на одном вычислительном узле [10]. При использовании контейнеров пользователи получают собственную среду с необходимым ПО требуемых версий.

Используемые для высокопроизводительных вычислений контейнеры должны обеспечивать не только изоляцию процессов пользователей друг от друга, но и поддерживать работу с внешними устройствами без вмешательства основной операционной системы и без использования прав суперпользователя, а также поддерживать современные технологии высокоскоростной сетевой коммуникации (InfiniBand, Omni-Path), работу с сетевыми файловыми системами, графическими ускорителями и сопроцессорами.

Применение различных технологий контейнерной виртуализации в составе ПО высокопроизводительных систем позволяет гибко настраивать программное окружение для заданий пользователей и оптимизировать использование вычислительных ресурсов. С этим связана актуальность задачи внедрения различных

технологий контейнеризации в состав ПО высокопроизводительных вычислительных систем [11–13].

Настоящая работа является развитием ранее проведенных исследований [14, 15] и посвящена описанию двух вариантов построения виртуальной вычислительной среды для организации параллельных вычислений, опробованных на инфраструктуре НИЦ «Курчатовский институт». Первый сценарий основан на использовании контейнеров без системы оркестрации, но под управлением СУППЗ, а второй – на использовании контейнеров с системой оркестрации Kubernetes [16] для автоматизации развертывания виртуальной масштабируемой вычислительной инфраструктуры под управлением СУППЗ.

Сравнение технологий контейнеризации

В 2024 году компания Containers Working Group провела опрос среди системных администраторов, инженеров, программистов и специалистов в области высокопроизводительных вычислительных систем. Результаты выявили, что наиболее популярными технологиями контейнеризации являются Docker, Singularity/Apptainer и Podman. В организациях, по мнению опрошенных, технологии контейнеризации используются преимущественно для проведения научных вычислений, в качестве среды разработки ПО для высокопроизводительных вычислений, а также для работы облачных сервисов и иных системных служб в среде Kubernetes.

Рассмотрим преимущества и недостатки наиболее популярных среди пользователей суперкомпьютерных центров технологий контейнерной виртуализации. При использовании технологии Docker задания представляются в виде одного контейнера или набора нескольких, связанных с помощью Docker Compose. Для создания и запуска контейнеров можно использовать командный интерфейс Docker или конфигурационный файл Dockerfile с набором команд для формирования команд, с информацией о базовом образе системы Linux и необходимых настройках.

К преимуществам технологии Docker следует отнести:

- поддержку со стороны сообщества разработчиков и пользователей;
- большое количество готовых образов для различных задач;

– совместимость с системой оркестрации контейнеров Kubernetes.

Среди недостатков отмечаются следующие:

– запуск контейнеров Docker с правами суперпользователя по умолчанию, ряд ограничений в непривилегированном режиме при работе с сетевыми интерфейсами и графическими ускорителями;

– обеспечение работы контейнера благодаря наличию активной службы на каждом вычислительном узле.

Опыт использования Docker для высокопроизводительных систем в Уральском отделении РАН опубликован в [13]. Он применялся для управления зависимостями расчетных приложений. Средой передачи информации выбрана Ethernet, на один узел кластера выдавалось эксклюзивно одно задание, а в качестве СУЗ использовалась система Slurm.

Изначально Singularity разрабатывалась для высокопроизводительных кластерных систем с целью обеспечения переносимости приложений между разными вычислительными платформами. По состоянию на конец 2023 года среди 178 высокопроизводительных вычислительных кластеров из рейтинга Top-500 в 48 заявлена поддержка контейнеризации, из них в 43 – с использованием Singularity [12]. Из-за разногласий в сообществе в 2021 году проект был разделен на две ветки – SingularityCE (продолжение оригинального проекта от коммерческого спонсора Sylabs) и Apptainer (копия проекта, разрабатываемая под эгидой Linux Foundation). Первая направлена на поддержание связи с остальными продуктами Sylabs, вторая остается ориентирована на открытое сообщество разработчиков. Для функционирования Singularity не требуется наличие сервисных служб на вычислительном узле, ее работа с контейнерами организована по тому же принципу, что и с Docker.

К основным преимуществам Singularity следует отнести:

– работу без прав суперпользователя;

– встроенную поддержку проброса в контейнер различных устройств с использованием драйверов операционной системы;

– возможность использования Docker-образов для создания контейнеров Singularity, а также преобразования готовых Docker-образов в образы Singularity.

Среди недостатков отмечаются следующие:

– ограниченное распространение только среди пользователей академической и научной среды;

– отсутствие свободно распространяемых систем управления контейнерами;

– отсутствие функционального аналога Docker Compose для описания и запуска многоконтейнерных приложений в одном конфигурационном файле.

Широко используемой среди разработчиков ПО альтернативой Docker также является Podman – разработка компании Red Hat. Соответствие стандартам Open Container Initiative обеспечивает совместимость, в том числе командного интерфейса, Podman с Docker. Как и в случае Singularity, для работы Podman не требуется наличие активных компонентов на узле, он по умолчанию запускает контейнеры от имени пользователя, а для управления ими использует службу systemd.

К преимуществам технологии Podman следует отнести:

– работу без прав суперпользователя;

– поддержку удаленного выполнения контейнеров с использованием SSH;

– возможность использования Docker-образов для создания контейнеров Podman;

– наличие встроенной системы управления многоконтейнерными приложениями, аналогичной Docker Compose;

– совместимость с системой оркестрации контейнеров Kubernetes.

Среди недостатков отмечаются следующие:

– меньшее сообщество Podman в сравнении с Docker;

– ограниченная поддержка использования графических ускорителей и сопроцессоров внутри контейнера;

– отсутствие встроенной поддержки распределенных файловых систем.

Результаты проведенного сравнения Docker, Singularity/Apptainer и Podman приведены в таблице 1, где «+» – полное выполнение требования, «*» – частичное, «–» – невыполнение требования.

В итоге было определено, что Singularity/Apptainer остается более подходящим вариантом для использования в вычислительных системах без средств оркестрации, но под управлением СУЗ (например, СУППЗ или Slurm); Podman целесообразно использовать в связке с системой оркестрации Kubernetes для построения масштабируемых вычислительных систем как альтернативу Docker.

Рассмотрим варианты применения технологий контейнерной виртуализации Singularity/Apptainer, Kubernetes и Podman совместно с СУППЗ.

Таблица 1

Сравнение систем контейнеризации

Table 1

Containerization systems comparison

Требование	Docker	Singularity/ Apptainer	Podman
Поддержка работы без прав суперпользователя	* Работа без прав суперпользователя имеет ряд ограничений	+	+
Поддержка работы с графическими ускорителями и сопроцессорами	* Поддержка через NVIDIA Toolkit требует права суперпользователя	+	* Ограниченная поддержка через NVIDIA Toolkit, AMD
Поддержка работы с высокоскоростной сетью InfiniBand, OmniPath	+	+	+
Поддержка сетевых файловых систем	* Требуется права суперпользователя для работы с частью файловых систем	+	+
Наличие свободно распространяемых систем управления контейнерами	+	-	+
Отсутствие активных компонентов на узлах для работы систем контейнеризации	-	+	+
Совместимость с Kubernetes	+	-	+

Развертывание набора контейнеров с прикладными программными пакетами под управлением СУППЗ

Образ контейнера Singularity (*Singularity Image Format*, sif-файл) – это файл, внутри которого упакован набор данных в формате squashfs, включающий метаданные и содержимое файла контейнера. Чтобы создать образ Singularity, необходимо использовать файл с описанием конфигурации контейнера (*Singularity Definition File*, def-файл).

Пример def-файла для контейнера Singularity с поддержкой OpenMPI приведен в листинге:

```
# указание агента, с помощью
# которого формируется базовый образ
# контейнера
Bootstrap: docker
From: ubuntu:latest

# определение переменных окружения
# контейнера и их значений
%environment
export OMPI_DIR=/opt/mpi

# перечисления команд для загрузки,
# установки и настройки ПО в контейнере
```

```
%post
apt-get update && \
apt-get install -y wget git bash
gcc gfortran g++ make
file bzip2
export OMPI_DIR=/opt/mpi
export OMPI_VERSION=4.0.1
ex-
port OMPI_URL="https://download.open-
mpi.org/release/open-mpi/v4.0/openmpi-
$OMPI_VERSION.tar.bz2"
mkdir -p /tmp/mpi
mkdir -p /opt
cd /tmp/mpi && \
wget -O openmpi-
$OMPI_VERSION.tar.bz2 $OMPI_URL && \
tar -xjf openmpi-
$OMPI_VERSION.tar.bz2
cd /tmp/mpi/openmpi-$OMPI_VERSION
&& \
./configure --prefix=$OMPI_DIR && \
make install
```

После формирования def-файла необходимо выполнить сборку контейнера с помощью команды вида

```
singularity build mpi-image.sif mpi-
image.def
```

Скомпилированный файл с контейнером копируется на кластерную вычислительную

систему и запускается на выполнение командой вида

```
mpirun -np 64 singularity exec mpi-image.sif /opt/mpitest
```

Для автоматизации формирования и развертывания контейнеров Singularity было разработано программное средство Autobuilder, состоящее из четырех модулей. Его архитектура представлена на рисунке 1.

В первом модуле производится генерация def-файла на основе данных, получаемых из конфигурационного файла в формате json. Извлечение данных производится с использованием утилиты jq. Следующий модуль вызывает вспомогательный командный сценарий Arptainer_build, который запускает выполнение команды сборки контейнера. При этом пользователь, собирающий контейнер, должен состоять в группе arptainer_users.

Далее третий модуль генерирует командный сценарий для постановки задания в очередь СУППЗ. После чего последний модуль выводит информацию по сгенерированному def-файлу и собранному образу и предлагает пользователю проверить выведенную информацию на предмет корректности. Если пользователь проверил информацию и указал, что она корректна, система выполняет команду установки задания в очередь. В противном случае программа завершается.

Программное средство подготовлено для сборки в deb-пакет, который отвечает за развертывание, настройку необходимых конфигурационных файлов, командных сценариев и состоит из трех файлов.

Autobuilder.conf – конфигурационный файл, хранящий набор переменных, необходимых для корректной подготовки контейнеров и их запуска на узлах (путь к сетевому каталогу пользователя, точка монтирования сетевого каталога внутри контейнера, каталог с базовыми образами для создания контейнера, каталог для файлов задания внутри контейнера, команда установки локальных пакетов внутри контейнера, директория, в которой будут находиться контейнеры, если они не размещены на общем сетевом хранилище).

Arptainer_build – bash-сценарий запуска команды сборки контейнера пользователя.

Autobuilder – система автоматического формирования и запуска контейнера через командный интерфейс СУППЗ.

При создании пользовательского образа можно использовать один из образов: базовый, специально подготовленный администратором суперкомпьютера; из контейнерных реестров (например, Docker Hub); собираемый автоматически программным средством Arptainer_build.

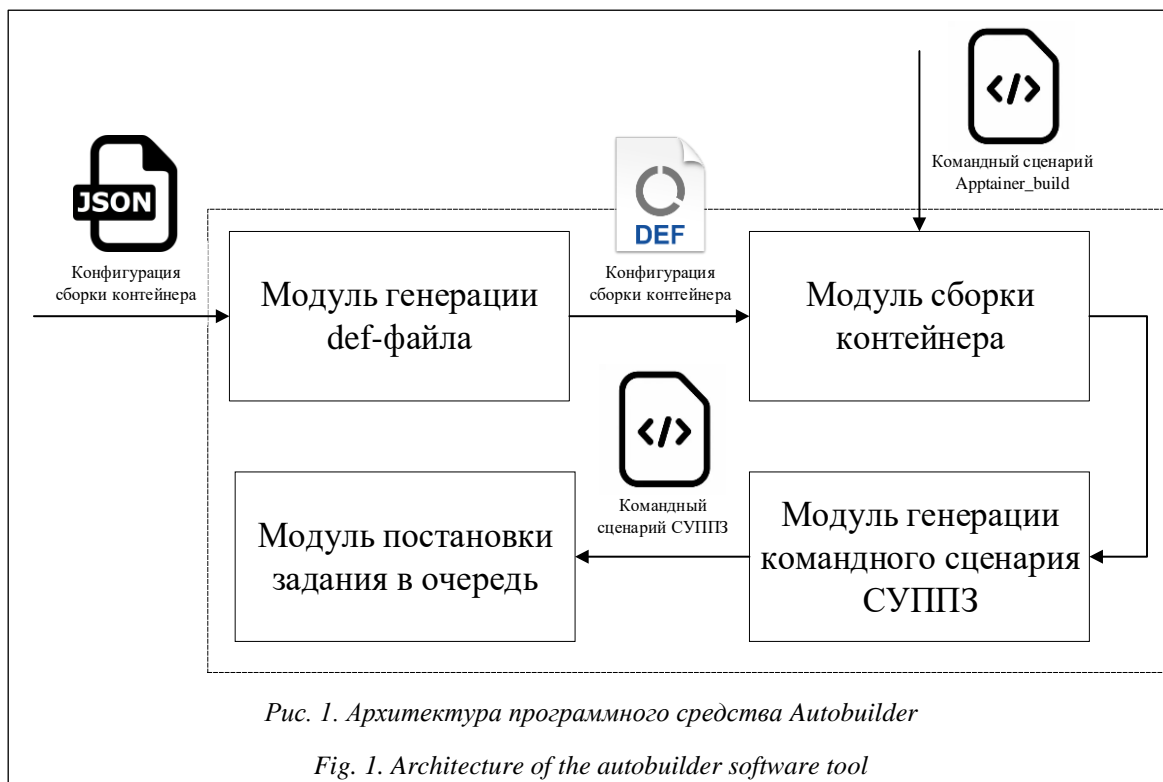


Рис. 1. Архитектура программного средства Autobuilder

Fig. 1. Architecture of the autobuilder software tool

Поскольку общим для всех программных средств контейнерной виртуализации является наличие накладных расходов на работу в сети Ethernet, рекомендуется настраивать InfiniBand или OmniPath. Однако, если такая настройка невозможна, следует использовать технологию Soft-RoCE (RXE), поскольку она обеспечивает меньшие накладные расходы и большую производительность в сравнении с Ethernet [17]. Поэтому разработанное ПО Apptainer_build собирает базовый образ с библиотеками для работы с InfiniBand, OmniPath и Soft-RoCE. В точку его входа добавляется командный сценарий, который определяет установленное на кластере оборудование (по загруженным модулям ядра) и в соответствии с ним фиксирует переменные окружения, указывающие, какой транспорт необходимо использовать библиотеке MPI.

Масштабируемая инфраструктура для высокопроизводительных вычислений на базе Kubernetes

В рамках настоящего исследования также рассмотрена возможность организации и авто-

матизированного развертывания высокопроизводительных вычислений под управлением СУППЗ, полностью основанные на контейнерной виртуализации. Основная сложность настоящего решения состоит в том, чтобы подобрать множество ПО, обоснованно определить взаимодействующие компоненты для него и сделать так, чтобы они функционировали в едином стеке.

Предложенный макет масштабируемой платформы для выполнения параллельных вычислений реализован на базе контейнеров Podman под управлением Kubernetes, внутри которого создается набор виртуальных кластеров под управлением СУППЗ с отдельными точками входа для различных групп пользователей. Компоненты разработанной масштабируемой инфраструктуры представлены на рисунке 2. Рассмотрим основные этапы ее построения.

На первом этапе выполняется установка и настройка среды Kubernetes. Для этого можно использовать различные средства и варианты его установки: локальная на персональном устройстве – Minikube, в облачной среде – Google Cloud Platform (GCP), Amazon Web Services (AWS), Amazon Elastic Container Service

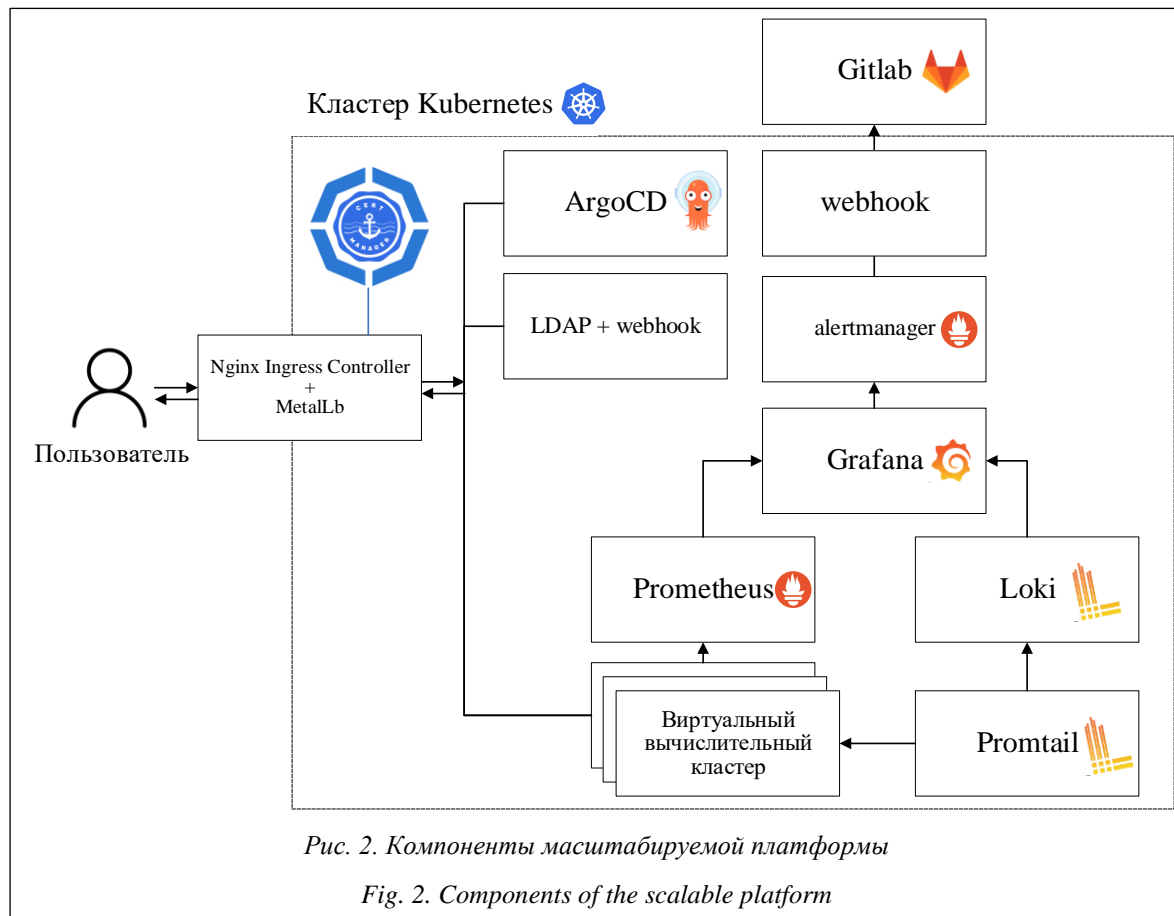


Рис. 2. Компоненты масштабируемой платформы

Fig. 2. Components of the scalable platform

for Kubernetes и аналогичные им сервисы, в корпоративной среде – kubespray, kubeadm, kops. Для развертывания кластера Kubernetes на ресурсах суперкомпьютерного центра был использован kubespray.

Работа kubespray строится на использовании специализированного инструмента Ansible, который автоматизирует настройку и предоставляет удаленное управление сетью серверов. Главное отличие Ansible от других подобных систем (например, Progress Chef или Puppet) заключается в использовании для работы SSH, в то время как остальные инструменты требуют установки специального PKI-окружения (системы, с помощью которой возможно подтвердить подлинность пользователей, устройств и сервисов). Также стоит отметить, что настройку удаленных узлов при применении Ansible возможно автоматизировать при помощи структурированных специальным образом текстовых файлов – сценариев (playbook).

Второй этап посвящен автоматизированному развертыванию компонентов масштабируемой платформы, представленных на рисунке 3. Для каждого подготавливается специальный контейнер, инкапсулирующий все необходимые ресурсы, настройки и шаблоны для развертывания приложения в виде единого логического объекта.

За развертывание контейнеризированных приложений в Kubernetes и автоматизацию процесса их обновления в построенном макете отвечает инструмент ArgoCD (<https://argo-cd.readthedocs.io/en/stable/>). Программное средство автоматически отслеживает изменения в Git-репозиториях приложений и выполняет их обновление на кластере Kubernetes в ручном или автоматическом режимах. Благодаря интеграции с системой контроля Git-версий ArgoCD также позволяет отменять ранее внесенные изменения.

Важными компонентами масштабируемой платформы являются системы журналирования и мониторинга. В качестве последней была использована система Prometheus с открытым исходным кодом (<https://prometheus.io/docs/introduction/overview/>), собирающая данные и сохраняющая их в базе временных рядов – специализированном программном решении, оптимизированном для хранения, индексирования и запроса данных, изменяющихся во времени. В отличие от большинства других систем мониторинга Prometheus имеет свой язык запросов для выборки данных из базы и их обработки. Благодаря совместимости с Kubernetes она позволяет автоматически находить и собирать наиболее часто встречающиеся в подобных средах метрики, а при необходимости – разрабатывать и использовать новых агентов.

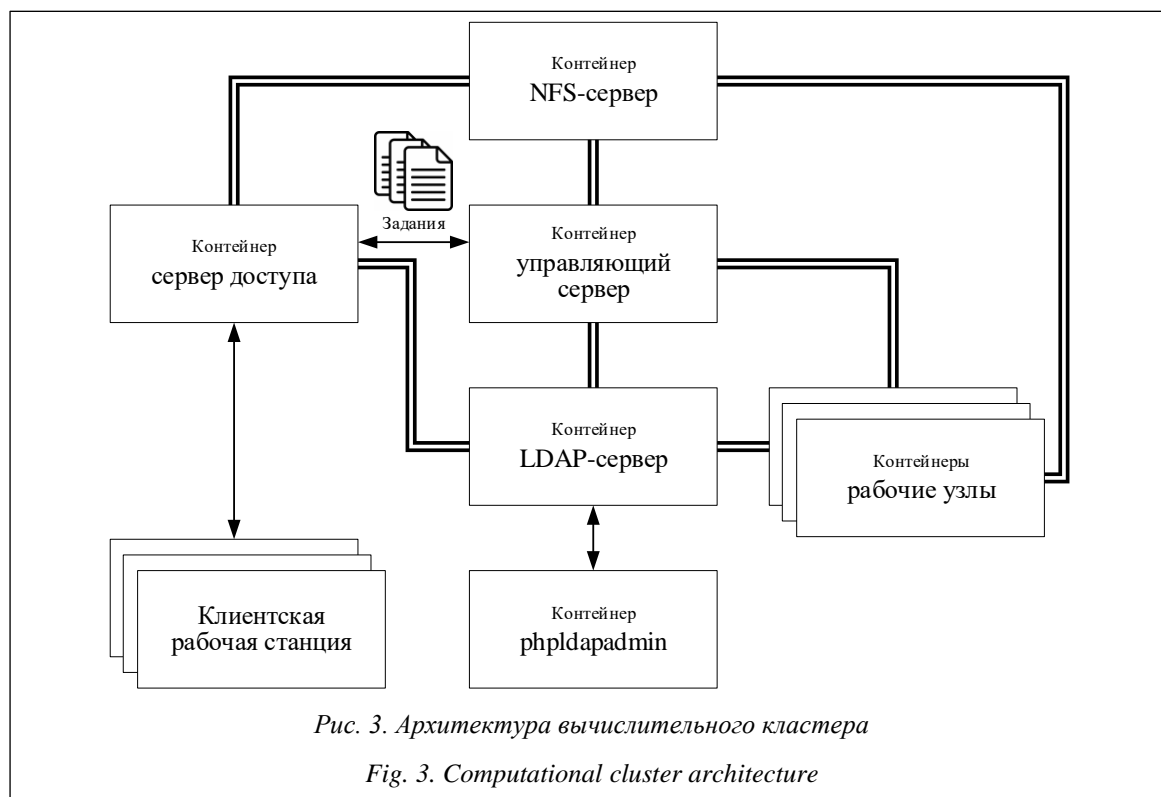


Рис. 3. Архитектура вычислительного кластера

Fig. 3. Computational cluster architecture

Централизованное хранение данных из системных файлов журналов реализовано программным средством Loki (<https://grafana.com/docs/loki/latest/>), где агентом сбора данных является Promtail, установленный внутри каждого контейнера среды Kubernetes. Система Loki интегрируется с Kubernetes и обладает масштабируемостью, надежностью и быстродействием – важными для систем такого вида качествами.

Для визуализации и анализа данных мониторинга и журналирования используется система с открытым исходным кодом Grafana (<https://grafana.com/docs/>). Она поддерживает различные методы представления данных (графики, диаграммы, гистограммы, карты, графы и др.) и отображает информацию через веб-интерфейс в виде динамически настраиваемых и размещаемых на панели инструментов Grafana элементов.

Для организации внешнего доступа к контейнерам Kubernetes необходимо установить и настроить средство NGINX Ingress Controller, которое отвечает за маршрутизацию запросов к сервисам внутри кластера Kubernetes. Оно позволяет выставлять два и более сервисов на одном IP-адресе, используя разные физические серверы. За сетевое взаимодействие между контейнерами Kubernetes отвечает компонент kube-proxy, за назначение внешних IP-адресов (из заранее определенного пула) сервисам Kubernetes – балансировщик сетевой нагрузки MetalLB (<https://metallb.io/>), необходимость применения которого обусловлена отсутствием аналогичного собственного решения.

На третьем этапе производится настройка вычислительного кластера, состоящего из набора связанных друг с другом контейнеров Podman для организации вычислений (сервер доступа, управляющий узел, узлы решающего поля) или выполнения сервисных функций по сопровождению вычислительного процесса (NFS-сервер, LDAP-сервер, сервер для администрирования LDAP).

Сервер доступа – контейнер, посредством которого пользователь получает доступ к виртуальной вычислительной инфраструктуре, осуществляет подготовку (сборку, отладку и компиляцию) программ и их запуск с помощью командного интерфейса СУППЗ.

На управляющем узле запущен сервер очередей СУППЗ, отвечающий за ведение очереди заданий, выделение и запуск заданий на выполнение, освобождение вычислительных ресурсов.

Рабочие узлы (решающее поле кластера) – набор контейнеров, на которых выполняются задания пользователей.

Файловый сервер – контейнер с настроенным NFS-сервером для централизованного хранения файлов пользователей и параллельного доступа к ним с узлов кластера в процессе вычислений.

Централизованное управление учетными записями пользователей вычислительной системы реализовано посредством сервера OpenLDAP. Администрирование учетных записей осуществляется через графический веб-интерфейс (phpldapadmin).

Для автоматизации развертывания контейнеров используется широко распространенный среди пользователей Kubernetes менеджер пакетов Helm, значительно упрощающий процесс управления и масштабирования контейнеризованными приложениями. Он позволяет автоматически разворачивать и при необходимости обновлять взаимосвязанные приложения с различным программным окружением, набором компонентов, внутренних и внешних связей. В терминологии Helm директория, в которой содержится все необходимое для запуска и работы приложения в среде Kubernetes, называется Helm-чартом. Для каждой конфигурации вычислительного кластера создается свой.

Приведем пример команды установки Helm-чарта mpi для запуска вычислительной системы, состоящей из сервера доступа (master), управляющего узла (suppz), двух вычислительных узлов (worker), NFS-сервера (nfs-server), LDAP-сервера (openldap) и веб-интерфейса управления LDAP (phpldapadmin):

```
helm install mpi . -f values.yaml -n production --create-namespace
```

С помощью команды `kubectl get pods -n production` можно проверить корректность запуска виртуального вычислительного кластера. Результат выполнения команды приведен в таблице 2, значение Running столбца STATUS подтверждает корректный запуск всех элементов вычислительного кластера.

Для использования виртуального вычислительного кластера необходимо создать учетную запись LDAP и группу для пользователей через веб-интерфейс phpldapadmin, для него создан ресурс Service, имеющий тип LoadBalancer (здесь работает сетевой балансировщик MetalLB), который предоставляет сетевой трафик из кластера Kubernetes. При помощи команды `kubectl get svc production` можно узнать, какие ресурсы присутствуют в простран-

стве имен виртуального вычислительного кластера и их IP-адреса. В таблице 3 представлен результат выполнения приведенной команды.

Доступ к серверу организован по SSH через внешний IP-адрес контейнера `mpi-ssh-gateway`, указанный в столбце `EXTERNAL-IP`.

В процессе вычислений часто возникает необходимость расширения или сокращения решающего поля вычислительной системы. В Kubernetes существует механизм автоматического масштабирования контейнеров. Для увеличения или уменьшения количества реплик нужного контейнера необходимо использовать команду `kubectl scale` с флагом `--replicas`. Например, команда вида `kubectl scale --replicas=6 production/mpi-worker` приведет к увеличению количества копий контейнера `mpi-worker` до 6, а изменение значения флага `--replicas=3` – к сокращению количества этих контейнеров до трех.

Добавление новых контейнеров с рабочими узлами виртуального кластера не приведет к масштабированию его решающего поля. Это связано с тем, что СУППЗ на узле управления не будет проинформирована о добавлении но-

вых контейнеров. С целью преодоления данного ограничения был разработан командный сценарий СУППЗ для автоматического масштабирования рабочих узлов в виртуальном вычислительном кластере. Представлен алгоритм программного средства (<http://www.swsys.ru/uploaded/image/2026-1/6.jpg>).

Предложенные в настоящей работе решения, связанные с оформлением собственной вычислительной инфраструктуры для работы каждого пользователя в рамках всей виртуальной вычислительной системы, могут быть применены в образовательном процессе при изучении распределенных вычислений, а также в научно-исследовательских подразделениях для расчета индивидуальных пользовательских задач.

Заключение

В рамках настоящей работы проведен сравнительный анализ существующих программных средств контейнеризации, предложены варианты их использования для организации высокопроизводительных вычислений. Опре-

Результат выполнения проверки корректности запуска кластера

Таблица 2

Results of the cluster launch correctness verification

Table 2

NAME	READY	STATUS	RESTARTS	AGE
mpi-master	1/1	Running	0	88s
mpi-suppz	1/1	Running	0	88s
mpi-worker-6b55d69978-796dk	1/1	Running	0	87s
mpi-worker-6b55d69978-f74fw	1/1	Running	0	87s
nfs-server-5bf4955987-b6jxd	1/1	Running	0	87s
openldap-7f6f498cdd-qhjwd	1/1	Running	0	87s
phpldapadmin-6bb7fff884-4m9pr	1/1	Running	0	87s

Информация о ресурсах вычислительного кластера

Таблица 3

Computational cluster resource information

Table 3

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
mpi	ClusterIP	None	<none>	1234/TCP
mpi-ssh-gateway	LoadBalancer	10.233.43.216	192.168.123.50	2022:31777/TCP
nfs-service	ClusterIP	10.233.38.242	<none>	2049/TCP,20048/TCP,111/TCP
openldap	ClusterIP	10.233.45.68	<none>	389/TCP
phpldapadmin	LoadBalancer	10.233.18.30	192.168.123.51	80:31792/TCP

делено, что Singularity/Apptainer остается более подходящим вариантом для использования в вычислительных системах без средств оркестрации, но под управлением СУППЗ или Slurm. Средство Podman целесообразно использовать в связке с системой оркестрации Kubernetes для построения масштабируемых вычислительных систем как альтернативу Docker.

Предложен вариант реализации автоматического масштабирования решающего поля виртуальной кластерной вычислительной систе-

мы с использованием командного интерфейса Kubernetes и СУППЗ.

Рассмотренные в работе варианты использования систем контейнеризации для построения кластерных вычислительных систем были успешно апробированы на вычислительных ресурсах НИЦ «Курчатовский институт».

Предметом дальнейшего исследования является оценка накладных расходов на контейнерную виртуализацию Singularity и Podman для различных высокопроизводительных платформ.

Список литературы

1. Sterling T., Anderson N., Brodowicz M. The essential SLURM: Resource management. In: High Performance Computing, 2025, pp. 119–152. doi: 10.1016/B978-0-12-823035-0.00007-9.
2. Баранов А.В. Построение системы управления заданиями пользователей суперкомпьютера на основе иерархической модели // Программные продукты и системы. 2025. Т. 38. № 2. С. 345–360. doi: 10.15827/0236-235X.150.345-360.
3. Баранов А.В., Ляховец Д.С. Влияние пакетирования на эффективность планирования параллельных заданий // Программные системы: теория и приложения. 2017. Т. 8. № 1. С. 193–208. URL: http://psta.psiras.ru/read/psta2017_1_193-208.pdf (дата обращения: 16.06.2025).
4. Баранов А.В., Корепанов П.М., Кузнецов Е.Е. Обеспечение информационной безопасности научного суперкомпьютерного центра // Программные продукты и системы. 2023. Т. 36. № 4. С. 615–631. doi: 10.15827/0236-235X.144.615-631.
5. Селезнев А.И., Селезнев И.Л. Контейнеризация в системах обработки данных // Молодой ученый. 2023. № 43. С. 7–12.
6. Ровягин М.М., Чугунков И.В., Савченко Н.А. Технология гибридных контейнеризированных вычислений для организации высокопроизводительной обработки данных в кластерных системах // Вопросы кибербезопасности. 2019. № 3. С. 39–44. doi: 10.21681/2311-3456-2019-3-39-44.
7. Соколов А.М., Ларионов А.А., Вишневский В.М., Мухтаров А.А. Архитектура распределенной системы для потоковых вычислений с контейнеризацией и приоритизацией задач // Информационные технологии и вычислительные системы. 2023. № 4. С. 5–18. doi: 10.14357/20718632230401.
8. Шейнман В., Стариков Д.Д., Тюменцев Д.В., Вавилов Г.Д. Повышение эффективности процессов разработки программного обеспечения: контейнерные технологии // Программные системы и вычислительные методы. 2024. № 4. С. 151–161. doi: 10.7256/2454-0714.2024.4.72015.
9. Топалов Н.К. Контейнеризация приложений: преимущества Docker и Kubernetes // Молодой ученый. 2024. № 51. С. 22–24.
10. Soltész S., Potzl H., Fiuczynski M.E., Bavier A., Peterson L. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. ACM SIGOPS Operating Systems Review, 2007, vol. 41, no. 3, pp. 275–287. doi: 10.1145/1272998.1273025.
11. Щапов В.А., Денисов А.В., Латышов С.Р. Применение контейнерной виртуализации Docker для запуска задач на суперкомпьютере // Суперкомпьютерные дни в России: тр. Междунар. конф. 2016. С. 505–511. URL: <https://www.russianscdays.org/files/pdf16/505.pdf> (дата обращения: 16.06.2025).
12. Polgár P., Menyhárt T., Baksay C.B., Kocsis G., Tajti T.G., Gál Z. Three level benchmarking of Singularity containers for scientific calculations. Annal. Math. et Inf., 2023, vol. 58, pp. 133–146. doi: 10.33039/ami.2023.08.014.
13. Баранов А.В., Долгов Б.В., Федотов А.В. Контейнеризация пользовательских заданий в суперкомпьютерной системе коллективного пользования. // Тр. НИИСИ РАН. 2019. Т. 9. № 6. С. 123–129. doi: 10.25682/niisi.2019.6.0016.
14. Баранов А.В., Шитик А.С. Способы и средства динамической реконфигурации сетей суперкомпьютера при представлении пользовательских заданий в виде контейнеров // Программные продукты, системы и алгоритмы. 2018. № 3. С. 62–70. doi: 10.15827/2311-6749.28.324.
15. Baranov A., Savin G., Shabanov B. et al. Methods of jobs containerization for supercomputer workload managers. Lobachevskii J. of Math., 2019, vol. 40, no. 5, pp. 525–534. doi: 10.1134/S1995080219050020.
16. Сайфан Дж. Осваиваем Kubernetes. Оркестрация контейнерных архитектур; [пер. с англ.]. СПб: Питер, 2019. 400 с.
17. Kaur G., Kumar M., Bala M. Comparing Ethernet & soft RoCE over 1 Gigabit Ethernet. IJCSIT, 2014, vol. 5, no. 1, pp. 323–327.

Application of container virtualization for implementing parallel computing on supercomputer clusters

© 2026 Evgeniy A. Kiselev¹, Aleksey V. Yarovoy^{1✉}

¹ National Research Centre “Kurchatov Institute”,
Moscow, 123182, Russian Federation

For citation

Kiselev, E.A., Yarovoy, A.V. (2026) ‘Application of container virtualization for implementing parallel computing on supercomputer clusters’, *Software & Systems*, 39(1), pp. 058–069 (in Russ.). doi: 10.15827/0236-235X.153.058-069

Article info

Received: 15.07.2025

After revision: 12.08.2025

Accepted: 20.08.2025

Abstract. The subject of the research presented in this article is the application of container virtualization technologies in supercomputing systems. The authors conducted a comparison of the most common technologies in the field of high-performance computing: Docker, Singularity/Apptainer, and Podman. The advantages and disadvantages of each technology were analyzed. Based on this analysis, two options for using container virtualization technologies in conjunction with the parallel job management system (SUPPZ), used in domestic supercomputing centers, are proposed. The first option is based on using Singularity containers managed by this SUPPZ without additional orchestration tools. A procedure for creating and launching user jobs as Singularity containers was developed, along with a software tool for automating the creation and deployment of Singularity containers. The second option involves using Podman containers together with the Kubernetes orchestration system to automate the deployment of a virtual high-performance computing infrastructure managed by the SUPPZ. A scalable platform prototype for parallel computing is proposed, within which a set of virtual clusters managed by the SUPPZ with separate entry points for different user groups is created; the main stages of building the prototype are considered. A method for scaling the problem-solving domain of a virtual computing system, utilizing the capabilities of both Kubernetes and the SUPPZ, is proposed. An algorithm for the operation of a software tool for automatic scaling of worker nodes in a virtual computing cluster was developed. The research results can be practically applied in the educational process for disciplines related to parallel computing, as well as in conducting research and development in the field of high-performance computing.

Keywords: container virtualization technologies, high-performance computing, supercomputing systems, Docker, Singularity, Kubernetes, SUPPZ (parallel job management system)

Acknowledgements. This work was carried out as part of the state assignment of the National Research Centre “Kurchatov Institute”

References

1. Sterling, T., Anderson, N., Brodowicz, M. (2025) ‘The essential SLURM: Resource management’, in *High Performance Computing*, pp. 119–152. doi: 10.1016/B978-0-12-823035-0.00007-9.
2. Baranov, A.V. (2025) ‘A supercomputer job management system based on a hierarchical resource management model’, *Software & Systems*, 38(2), pp. 345–360 (in Russ.). doi: 10.15827/0236-235X.150.345-360.
3. Baranov, A.V., Lyakhovets, D.S. (2017) ‘The influence of packaging on efficiency of parallel jobs scheduling’, *Program Systems: Theory and Applications*, 8(1), pp. 193–208, available at: http://psta.psiras.ru/read/psta2017_1_193-208.pdf (accessed June 16, 2025) (in Russ.).
4. Baranov, A.V., Korepanov, P.M., Kuznetsov, E.E. (2023) ‘Information security of a supercomputer center’, *Software & Systems*, 36(4), pp. 615–631 (in Russ.). doi: 10.15827/0236-235X.144.615-631.
5. Seleznev, A.I., Seleznev, I.L. (2023) ‘Containerization in data processing systems’, *Young Scientist*, (43), pp. 7–12 (in Russ.).
6. Rovnyagin, M.M., Chugunkov, I.V., Savchenko, N.A. (2019) ‘Hybrid containerized computing technology for high-performance data processing in cluster systems’, *Cybersecurity Issues*, (3), pp. 39–44 (in Russ.). doi: 10.21681/2311-3456-2019-3-39-44.
7. Sokolov, A.M., Larionov, A.A., Vishnevsky, V.M., Mukhtarov, A.A. (2023) ‘Architecture of a distributed system for streaming computing with containerization and prioritization of tasks’, *J. of Inform. Tech. and Computing Sys.*, (4), pp. 5–18 (in Russ.). doi: 10.14357/20718632230401.
8. Sheinman, V., Starikov, D.D., Tyumentsev, D.V., Vavilov, G.D. (2024) ‘Improving the efficiency of software development processes: Container technologies’, *Software Sys. and Computational Methods*, (4), pp. 151–161 (in Russ.). doi: 10.7256/2454-0714.2024.4.72015.
9. Topalov, N.K. (2024) ‘Application containerization: advantages of Docker and Kubernetes’, *Young Scientist*, (51), pp. 22–24 (in Russ.).
10. Soltesz, S., Potzl, H., Fiuczynski, M.E., Bavier, A., Peterson, L. (2007) ‘Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors’, *ACM SIGOPS Operating Systems Review*, 41(3), pp. 275–287. doi: 10.1145/1272998.1273025.
11. Shechapov, V.A., Denisov, A.V., Latypov, S.R. (2016) ‘Using Docker container virtualization to run tasks on supercomputer nodes’, *Proc. Int. Conf. Russian Supercomputing Days*, pp. 505–511, available at: <https://www.russianscdays.org/files/pdf16/505.pdf> (accessed June 16, 2025) (in Russ.).

12. Polgár, P., Menyhárt, T., Baksay, C.B., Kocsis, G., Tajti, T.G., Gál, Z. (2023) ‘Three level benchmarking of Singularity containers for scientific calculations’, *Annal. Math. et Inf.*, 58, pp. 133–146. doi: 10.33039/ami.2023.08.014.
13. Baranov, A.V., Dolgov, B.V., Fedotov, A.V. (2019) ‘Job containerization in a supercomputer job management system’, *Proc. of SRISA RAS*, 9(6), pp. 123–129 (in Russ.). doi: 10.25682/niisi.2019.6.0016.
14. Baranov, A.V., Shitik, A.S. (2018) ‘Methods and means of dynamic reconfiguration of supercomputer networks when presenting user tasks as containers’, *Software J.: Theory and Applications*, (3), pp. 62–70 (in Russ.). doi: 10.15827/2311-6749.28.324.
15. Baranov, A., Savin, G., Shabanov, B. (2019) et al. ‘Methods of jobs containerization for supercomputer workload managers’, *Lobachevskii J. of Math.*, 40(5), pp. 525–534. doi: 10.1134/S1995080219050020.
16. Sayfan, G. (2017) *Mastering Kubernetes*. Packt Publ., 642 p. (Russ. ed.: (2019) St. Petersburg, 400 p.).
17. Kaur, G., Kumar, M., Bala, M. (2014) ‘Comparing Ethernet & soft RoCE over 1 Gigabit Ethernet’, *IJCSIT*, 5(1), pp. 323–327.

Авторы

Киселёв Евгений Андреевич¹,
к.т.н., доцент, старший научный сотрудник,
kiselev@jscc.ru
Яровой Алексей Владимирович¹,
лаборант-исследователь, iarovoi-av@yandex.ru

¹ НИЦ «Курчатовский институт»,
г. Москва, 123182, Россия

Authors

Evgeniy A. Kiselev¹, Cand. of Sci. (Engineering),
Associate Professor, Senior Researcher,
kiselev@jscc.ru
Aleksey V. Yarovoy¹,
Research Assistant, iarovoi-av@yandex.ru

¹ National Research Centre “Kurchatov Institute”,
Moscow, 123182, Russian Federation