УДК 004; 621.398; 681.5; 681.324:681.3.001.57
doi:10.15217/issn1684-8853.2018.2.60

СРАВНИТЕЛЬНОЕ ТЕСТИРОВАНИЕ КОНТЕЙНЕРНОЙ И ГИПЕРВИЗОРНОЙ ВИРТУАЛИЗАЦИИ

А. В. Гордеев[a], *доктор техн. наук, профессор, ff2avg@mail.ru*
Д. В. Горелик[a], *магистрант, de5509@mail.ru*
[a]*Санкт-Петербургский государственный университет аэрокосмического приборостроения, Б. Морская ул., 67, Санкт-Петербург, 190000, РФ*

Введение: *при построении системы для удаленной работы студентов со своими виртуальными машинами важно правильно выбрать главный принцип виртуализации, что позволит надежно и легко разделять множества виртуальных машин разных пользователей и при этом наиболее эффективно расходовать вычислительные ресурсы сервера.* ***Цель:*** *определить, насколько медленнее будут работать виртуальные машины в вычислительном кластере при использовании технологии контейнеров по сравнению с традиционной технологией использования гипервизора, насколько меньше при этом потребуется оперативной памяти и стоит ли использовать контейнеры ради упрощения в решении важнейшей задачи разделения множеств виртуальных машин разных пользователей.* ***Результаты:*** *получены зависимости потребления основных вычислительных ресурсов от количества запускаемых виртуальных машин. Анализ полученных результатов показал, что в случае построения кластера серверов с общим внешним устройством хранения данных для запуска большого количества виртуальных машин, работающих в пакете программ VirtualBox, выгоднее использовать контейнеры с добавлением программного обеспечения Docker.* ***Практическая значимость:*** *при создании вычислительного кластера контейнерная технология виртуализации с использованием Docker и VirtualBox позволяет не только существенно экономить оперативную память при запуске большого количества виртуальных машин, но и увеличивает скорость вычислений.*

Ключевые слова *— виртуальные машины, кластер, гипервизор, контейнер, VirtualBox, Docker.*

Цитирование: Гордеев А. В., Горелик Д. В. Сравнительное тестирование контейнерной и гипервизорной виртуализации // Информационно-управляющие системы. 2018. № 2. С. 60–66. doi:10.15217/issn1684-8853.2018.2.60
Citation: Gordeev A. V., Gorelik D. V. Comparative Testing of Container and Hypervisor Virtualizations. Informatsionno-upravliaiushchie sistemy [Information and Control Systems], 2018, no. 2, pp. 60–66 (In Russian). doi:10.15217/issn1684-8853.2018.2.60

Введение

При построении системы для удаленной работы студентов со своими виртуальными машинами [1] возник вопрос о выборе главного принципа виртуализации. Обычно программное обеспечение Oracle VirtualBox [2], которое было выбрано в качестве основы для построения названной системы, используют на хост-системе, и получается гипервизорная виртуализация, где в роли гипервизора выступает хост-система с Ubuntu Server 16.04. Достоинства и недостатки этого подхода в целом известны. Обоснование выбора программного обеспечения Oracle VirtualBox и проблема, которую приходится решать для изоляции подмножества виртуальных машин и сетей одного пользователя от виртуальных машин и сетей другого пользователя (при необходимости предоставить возможность работать со своими виртуальными машинами большому количеству пользователей), была описана и решена в работе [1]. Однако проблемы изоляции виртуальных машин и сетей одного пользователя от виртуальных машин и сетей другого пользователя относительно легко решаются при использовании контейнеров и платформы Docker. Docker — открытая платформа для запуска приложений в свободно изолированной среде (в контейнере) [3]. Поэтому возникла идея провести исследования, которые ответили бы на вопросы: насколько можно сэкономить оперативной памяти (ОП) в хост-системе в случае использования контейнерной виртуализации по сравнению с гипервизорной и насколько мы при этом потеряем в быстродействии. Проведенные эксперименты неожиданно показали, что при использовании контейнеров и программного обеспечения Docker при организации кластерных вычислений мы не только легко решаем проблему изоляции виртуальных машин и сетей одного удаленного пользователя от другого, но и получаем даже выигрыш в скорости вычислений и затратах ОП.

Напомним, что контейнерная виртуализация — это такой метод виртуализации, при котором ядро операционной системы поддерживает вместо одного сразу несколько изолированных экземпляров пространства пользователя [4]. Скорее всего, эта идея была заимствована от реентерабельных программ, которые позволяют иметь один экземпляр кода, но несколько процессов могут его исполнять параллельно, каждый — со своими наборами переменных, что снижает расходы ОП. Как известно, обращение к переменным реентерабельной программы осуществляет-

ся с помощью косвенной адресации, причем текущие переменные, как правило, располагаются на вершине стека, из которого под новый процесс выделяется новый блок ячеек памяти. Запрос (и освобождение) такого блока памяти осуществляется посредством обращения к системной подпрограмме, которая выполняет затребованные действия при отключенной системе прерываний для того, чтобы обеспечить целостность этой операции. Таким образом, контейнерная виртуализация является более эффективной в плане расходования памяти, но теряет в быстродействии из-за дополнительных расходов на вычисление адресов. Сам гипервизор работает следующим образом: в операционной системе хоста эмулируется аппаратное обеспечение, поверх которого запускаются гостевые операционные системы. Это дает нам понять, что связь между гостевой и хостовой операционными системами следует четкому правилу: все, что делает используемая аппаратная часть, должно быть доступно и гостевой операционной системе со стороны хостовой [5].

Контейнеры, в свою очередь, напротив, являются виртуализацией на уровне операционной системы. А это уже подразумевает, что есть гостевая операционная система, которая использует то же самое ядро, что и хостовая. Такой подход дает контейнерам большое преимущество: они могут быть меньше и компактнее гипервизорных гостевых сред, поскольку у них с хостом намного больше общего, чем у виртуальных машин. То есть контейнерная виртуализация не использует полноценные виртуальные машины, как это делается в случае гипервизоров, а создает виртуальное окружение с собственным пространством процессов и сетевым стеком. Экземпляры пространств пользователя (часто называемые контейнерами или зонами) с точки зрения пользователя полностью идентичны реальному серверу, но они в своей работе используют один экземпляр ядра операционной системы. Ядро обеспечивает полную изолированность контейнеров, поэтому программы из разных контейнеров не могут воздействовать друг на друга. Очевидно, что главным недостатком этой технологии является то, что виртуальная среда задается хост-системой, и она будет одна и та же для всех пространств, которые мы можем создать. Для преодоления данного недостатка было предложено модернизировать эту технологию до понятия докера (Docker). Докеры могут быть с разным виртуальным окружением, хоть и работать на одной и той же хост-системе [6].

Для возможности запускать на GNU/Linux-системе виртуальные машины с ОС Microsoft Windows можно создать соответствующие Linux-контейнеры и с помощью программного обеспечения Docker установить пакет программ VirtualBox. Контейнеры изолируют виртуальные машины VirtualBox, запускаемые разными пользователями. Важно, что внутри одного контейнера могут работать сразу несколько виртуальных машин одного пользователя, а каждому пользователю предоставляется свой контейнер.

Описание эксперимента

Для построения кластерной системы это очень важно, так как виртуальные машины одного пользователя смогут легко и быстро взаимодействовать друг с другом через виртуальные сети. При этом виртуальные машины и сети разных пользователей могут быть запущены на разных узлах кластера. Поскольку это кластер серверов, то в качестве общего внешнего устройства выступает сетевая система хранения данных NetGear 3200, которая предоставляет по i-SCSI-протоколу единый том X-RAID-6-уровня. На этом томе располагаются файлы виртуальных машин, благодаря чему они могут быть доступны для любого узла кластера. Пользователь работает со своими виртуальными машинами через тот сервер, который будет иметь наименьшую нагрузку в кластере.

Соответственно, для сравнительного анализа рассматриваемых методов виртуализации посредством тестирования в целях выбора наиболее эффективного метода нужно было разработать методику получения интересующих нас параметров — потребляемых вычислительных ресурсов.

Было предложено в качестве нагрузки на сервер, на котором будут работать виртуальные машины, взять процедуру инсталляции операционной системы. Эта процедура задействует сразу все виды ресурсов и сопровождается высокой процессорной нагрузкой, большим потреблением ОП, интенсивной работой с внешними устройствами и, что очень важно, активной работой с накопителем на магнитных дисках (как правило, это виртуальные накопители, в качестве которых используются файлы хост-системы). Часть процедуры инсталляции, которая обычно требует интерактивного взаимодействия с пользователем (системный администратор во время инсталляции отвечает на некоторые вопросы), была автоматизирована. Автоматизация ответов была выполнена созданием нового файла ответов, в котором заранее подготовлены все ответы на все вопросы. Это позволило проводить эксперименты без воздействия интерактивной составляющей, которая могла бы повлиять на результаты эксперимента. Для снятия показаний был написан специальный скрипт на языке Bash, который с момента его запуска записывает в отдельный файл время в секундах, нагрузку CPU (Central

Processing Unit) в процентах и расходование оперативной памяти в процентах.

Следующим шагом подготовки к проведению исследования был выбор алгоритма, по которому непосредственно проводились бы эксперименты. В ходе подготовки был разработан следующий метод: так как в нашем кластере [1] имеются два идентичных сервера на базе платформы Supermicro E210C-M3, то было принято решение на одном из них проводить эксперимент с обычными виртуальными машинами, а на втором проводить эксперименты с докерами (контейнерами). А так как мы строим кластер серверов, то оба этих вычислительных сервера имеют доступ к общему хранилищу. Далее необходимо было подготовить серверы, установив VirtualBox для сервера, на котором используется гипервизорная виртуализация, а для сервера с контейнерной — программное обеспечение Docker. Это открытая платформа для разработки, доставки и эксплуатации приложений [7]. В своем ядре это программное обеспечение позволяет запускать практически любое приложение, безопасно изолированное внутри контейнера, что дает возможность обезопасить работу хост-системы. То есть события, происходящие в контейнере, никак не влияют. Благодаря безопасной изоляции можно запускать на одном хосте много контейнеров одновременно. За счет легковесной природы контейнера, который запускается без дополнительной нагрузки гипервизора на аппаратную часть сервера, используется меньше ресурсов.

Также в связи с возможностью использовать разные методы хранения данных необходимо провести сравнительный анализ быстродействия сетевого хранилища и обычного (локального) накопителя на жестких магнитных дисках, установленного непосредственно в сервер хранилища данных. Нужно проанализировать, как эти средства и технологии хранения данных повлияют на быстродействие и производительность виртуальных машин и контейнеров, и выбрать наилучший из имеющихся вариантов.

С целью имитировать работу студентов на кластере был определен необходимый минимум виртуальных машин и контейнеров, который соответствовал бы одновременной работе, например, пяти пользователей (студентов). Так, на первом сервере требовалось создать 10 виртуальных машин — по две машины на студента. После создания, т. е. определения конфигурации виртуального компьютера, а это процесс достаточно быстрый и не требующий больших ресурсов, запускается скрипт документирования потребляемых вычислительных ресурсов. Далее запускаются собственно виртуальные машины, и уже непосредственно в них сразу начинается автоматизированная установка операционных систем. Вышеупомянутый скрипт протоколирует потребление вычислительных ресурсов этими виртуальными машинами.

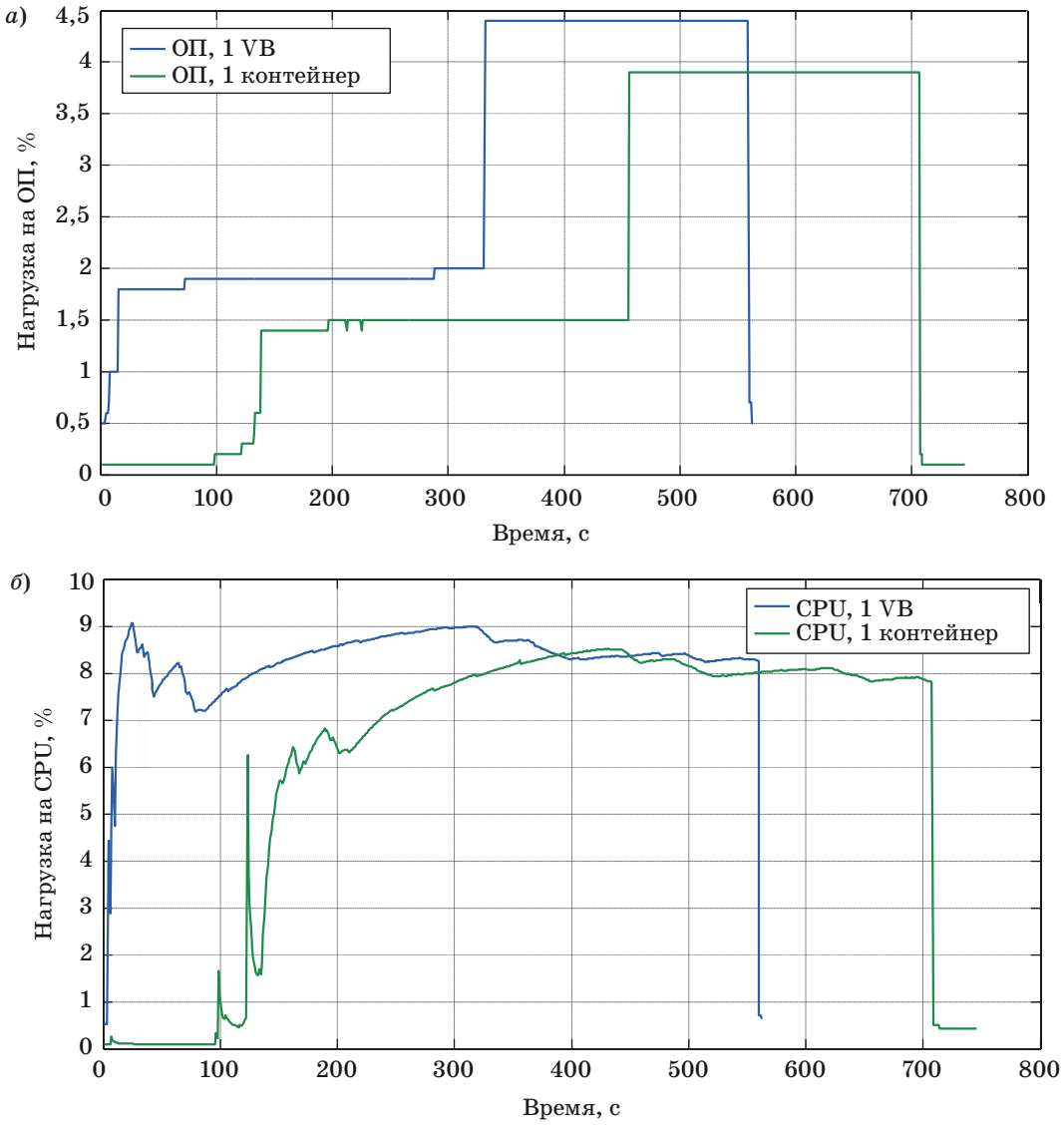
Для имитации нагрузки на второй сервер были созданы пять контейнеров со специальным набором программного обеспечения, необходимого для организации работы с графическими приложениями в контейнерах. Затем в каждом контейнере было создано по две виртуальные машины, и уже непосредственно после окончания процедуры их создания запускался скрипт документирования нагрузки, а следом запускались все виртуальные машины и шла автоматизированная установка операционных систем.

В обработку результатов входил ряд последовательных этапов, которые включают в себя приведение данных к табличному виду и постройку представленных ниже графиков.

Рассмотрим графики, на которых показан тестовый эксперимент с одной виртуальной машиной и одним контейнером с одной виртуальной машиной (рис. 1, *а* и *б*). Наглядно видно, что обычная виртуальная машина работает быстрее, чем аналогичная, но в контейнере, потребляя при этом больше вычислительных ресурсов. Как и ожидалось, контейнер показал меньшее потребление ресурсов, но более продолжительную работу. Если посмотреть на форму и поведение графиков, то они идентичны: основным пиком нагрузки на ОП является период непосредственной установки операционной системы, это вторая часть графика. По сравнению с первой частью графика, который отображает фазу распаковки файлов из образа, нагрузка возросла приблизительно в 2 раза. Это говорит о том, что собственно построение (сборка) системы из двоичных файлов требует существенно больше ресурсов, чем подготовка этих двоичных файлов.

По нагрузке на CPU виртуальная машина более требовательна, ей необходимо больше вычислительной мощности. Конечно, в масштабах сервера это не столь значительно, если мы говорим об одной виртуальной машине.

Посмотрим на графики тестирования виртуальных машин, расположенные на рис. 2, *а* и *б*. Здесь мы запускаем уже по 10 виртуальных машин на одном сервере. Обратим внимание на то, что тут представлены результаты двух экспериментов: кривые зеленого и синего цветов иллюстрируют поведение серверов с использованием внешнего сетевого хранилища, а красного и голубого — работу серверов с использованием локального накопителя. Различия в том, что были использованы разные технологии доступа к накопителям данных. На рис. 2, *а* представлена зависимость нагрузки на ОП от времени. Эксперимент номер один был проведен с использованием сетевой системы хранения данных NetGear 3200,

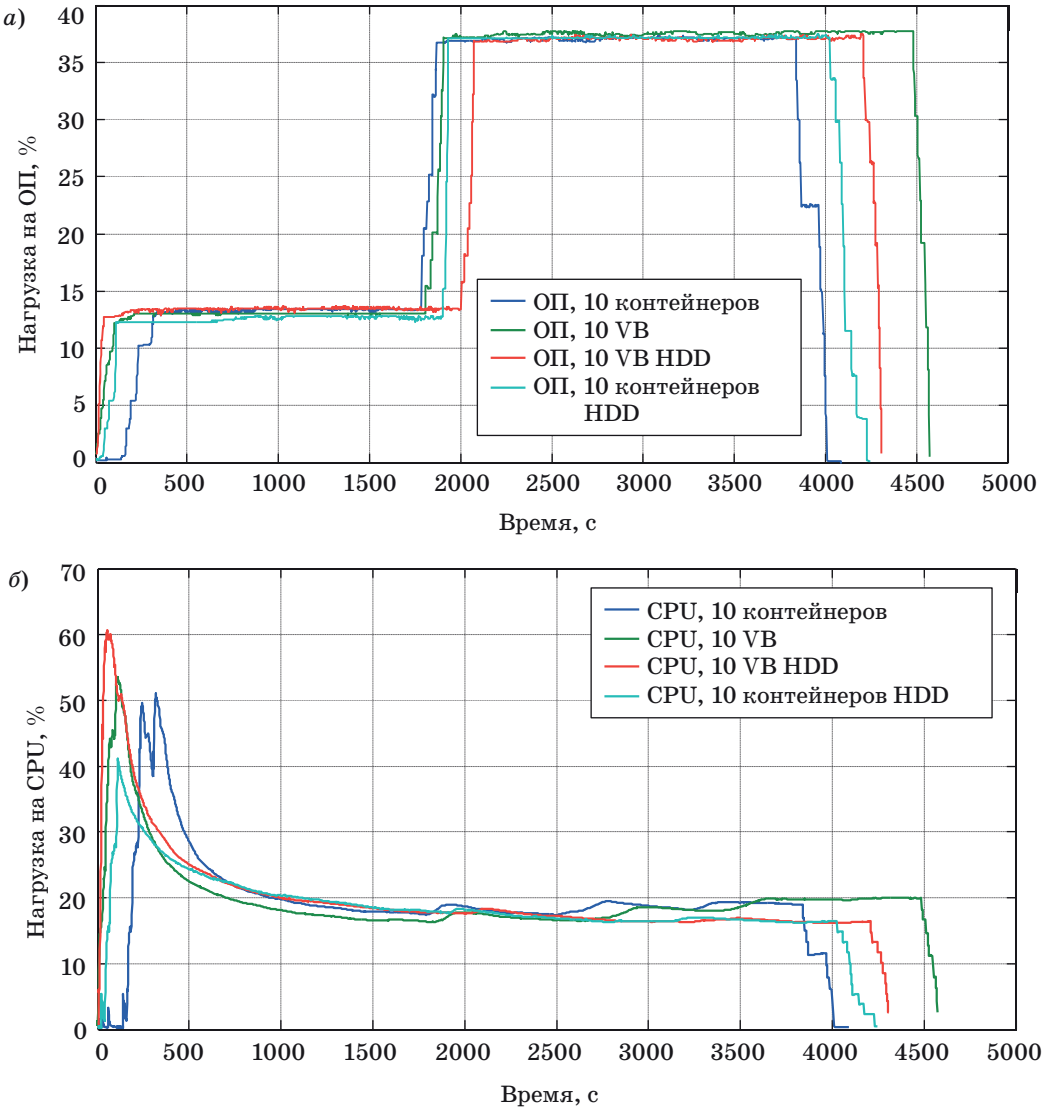


■ ***Рис. 1***. Нагрузка на оперативную память (*а*) и CPU (*б*) в эксперименте с одним контейнером и одной виртуальной машиной

■ ***Fig. 1***. RAM load (*а*) and CPU load (*б*) in experiment with one container and one virtual machine

которая предоставляет единый том X-RAID-6-уровня по i-SCSI-протоколу. Из графика видно, что контейнерное решение отрабатывает быстрее на 600–800 с. То есть при конфигурации с использованием протокола i-SCSI контейнерная виртуализация оказывается быстрее. Если посмотреть на график рис. 2, *б*, то можно обнаружить, что контейнерная виртуализация с применением Docker не только быстрее, но она еще и меньше нагружает процессор. То есть по графику рис. 2, *а* уровень потребления ОП одинаковый, а потребление ресурсов процессора меньше. Возникает вопрос: почему контейнеры работают быстрее, когда их становится больше?

Возможно, что этот эффект достигается за счет использования UnionFS (Union File System) [8]. Это вспомогательная файловая система, производящая каскадно-объединенное монтирование других файловых систем. Она позволяет файлам и каталогам изолированных файловых систем, известных как ветви, прозрачно перекрываться, формируя единую связанную файловую систему. Каталоги, которые имеют тот же путь в объединенных ветвях, будут совместно отображать содержимое в объединенном каталоге новой виртуальной файловой системы. Когда ветви монтируются, то указывается приоритет одной ветви над другой. Следовательно, когда обе ветви содержат файл с идентичным именем, одна ветвь будет иметь больший приоритет. Различные ветви могут одновременно находиться в режиме «только чтение» и «чтение-запись». Таким образом, за-



■ ***Рис. 2.*** Нагрузка на оперативную память (*а*) и CPU (*б*) в эксперименте с десятью контейнерами и десятью виртуальными машинами

■ ***Fig. 2.*** RAM load (*а*) and CPU load (*б*) in an experiment with 10 containers and 10 virtual machines

пись в объединенную виртуальную файловую систему будет направлена на определенную реальную файловую систему.

Union File System состоит из слоев (layers). Слои как бы накладываются друг на друга. Все используемые нами контейнеры используют общие защищенные от записи слои, в которых находятся неизменяемые файлы операционной системы контейнера, это приводит к ускорению доступа к данным. А для изменяемых файлов — файлов виртуальных машин, работающих в докерах, — каждый из контейнеров будет иметь собственный слой. Естественно, Docker использует это решение не только для операционной системы, но и для любых общих частей контейнеров, которые были созданы на основе общих «предков» их образов; скорее всего, что за счет этого он и выигрывает по скорости работы.

То есть получается, что виртуальные машины, использующие технологию гипервизорной виртуализации, создают много идентичных копий операционных систем, которые никак не связаны между собой и потребляют больше аппаратных ресурсов сервера, чем контейнеры, работающие на технологии контейнерной виртуализации с использованием UnionFS.

Рассмотрим последний эксперимент, который проводился с использованием установленных в сервер жестких дисков, т. е. без использования i-SCSI-протокола. Рассмотрев графики на рис. 2, можно заметить, что скорость работы контейнеров, в принципе, упала незначительно, при том

что общая картина потребления вычислительных ресурсов никак не изменилась. А если обратить внимание на работу виртуальных машин, то можно заметить, что в этом случае производительность возросла. То есть за счет исключения сетевых операций на обращение к хранилищу данных и работу с внутренним хранилищем сервера виртуальные машины получают ускорение.

Итак, в результате проведенного исследования был определен круг параметров, от которых зависит производительность двух технологий виртуализации — гипервизорной и контейнерной. Как показал первый (тестовый) эксперимент, если проводить сравнение одной виртуальной машины с одним контейнером, то виртуальная машина работает гораздо быстрее (см. рис. 1), потребляя больше вычислительных ресурсов сервера. Но при увеличении количества виртуальных машин до 10 экземпляров обстановка кардинально меняется в обратную сторону, и контейнеры начинают выигрывать не только по быстродействию, но и по потреблению аппаратных ресурсов, которое у них снижено, благодаря использованию UnionFS.

Существует зависимость и от технологии, применяемой для доступа к хранилищу данных. Мы рассматривали две технологии общего, внешнего серверного хранилища данных. Прежде всего, это по протоколу i-SCSI к единому для обоих серверов тому X-RAID-6-уровня. Кроме этого, провели тестирование при использовании локальных накопителей, организованных в RAID-массив. Исследование показало, что использование общего сетевого хранилища дает еще больший выигрыш контейнерам за счет того, что контейнеры используют UnionFS. А обычные виртуальные машины получают дополнительную задержку за счет использования сетевого протокола (в отличие от непосредственного доступа к локальным жестким дискам).

Рассмотрев зависимости потребления аппаратных ресурсов, можно сказать, что с увеличением количества единиц виртуальных машин и контейнеров потребность в ресурсах растет одинаково, т. е. зависимость расхода ресурсов относительно друг друга сохраняется. Время выполнения всех операций также растет, но с разной зависимостью.

Заключение

Исследование показало, что хотя потребление вычислительных ресурсов увеличивается практически одинаково с увеличением количества используемых контейнеров и виртуальных машин, контейнеры потребляют ресурсов меньше. Продолжительность выполнения вычислений растет приблизительно одинаково, и тут выигрывают контейнеры, так как отрабатывают быстрее обычных виртуальных машин. Для проектируемой кластерной системы этот критерий является очень важным. Так как используемые внутри контейнеров виртуальные машины безопасно изолированы, то это дает возможность студентам работать отдельно и никак не пересекаться в процессе выполнения лабораторных работ.

Принимая во внимание все полученные результаты, а также все очевидные плюсы контейнерного решения, можно сделать вывод о целесообразности проектировать систему на базе уже развернутой системы дистанционного обучения, но с применением технологии контейнерной виртуализации.

Литература

1. **Гордеев А. В.** Организация удаленной работы студентов со своими виртуальными машинами // Информационно-управляющие системы. 2017. № 2. С. 96–100. doi:10.15217/issn1684-8853.2017.2.96
2. **Горелик Д. В.** Использование контейнеров для построения системы работы с удаленными виртуальными машинами // Семидесятая Международная студенческая научная конференция ГУАП: сб. докл. Ч. 2. Технические науки. СПб.: ГУАП, 2017. С. 66–69.
3. Понимая Docker. https://habrahabr.ru/post/253877 (дата обращения: 15.11.2017).
4. **Моуэт Э.** Использование Docker. Разработка и внедрение программного обеспечения при помощи технологии контейнеров: руководство. — ДМК-Пресс, 2017. — 300 с.
5. **Маркелов А.** OpenStack. Знакомство с облачной операционной системой. 2-е изд. — М.: ДМК-Пресс, 2016. — 248 с.
6. **Matthias K., Kane S.** Docker: Up & Running. Shipping Reliable Containers in Production. — O'Reilly, 2015. — 198 p.
7. **Jessie Frazelle.** Docker Containers on the Desktop. https://blog.jessfraz.com/post/docker-containers-on-the-desktop/ (дата обращения: 07.04.2017).
8. **Попов С.** Образы и контейнеры Docker в картинках. https://habrahabr.ru/post/272145/ (дата обращения: 17.11.2017).

UDC 004; 621.398; 681.5; 681.324:681.3.001.57
doi:10.15217/issn1684-8853.2018.2.60

Comparative Testing of Container and Hypervisor Virtualizations

Gordeev A. V.[a], Dr. Sc., Tech., Professor, ff2avg@mail.ru
Gorelik D. V.[a], Master Student, de5509@mail.ru
[a]Saint-Petersburg State University of Aerospace Instrumentation, 67, B. Morskaia St., Saint-Petersburg, 190000, Russian Federation

Introduction: When building a system for students' remote work with their virtual machines, it is important to correctly choose the main principle of virtualization, which will reliably and easily split the sets of virtual machines of different users, and most efficiently use the computing resources of the server. **Purpose:** Determining how much slower virtual machines in a computing cluster will be when using container technology in comparison with the traditional technology of using hypervisors, how much less RAM is required, and whether it is worth using containers for the sake of simplicity in solving the most important task of splitting sets of virtual machines of different users. **Results:** We have obtained the dependencies of consumption of basic computing resources on the number of virtual machines being run. The analysis of the obtained graphs showed that in the case of building a cluster of servers with a common external storage device in order to run a large number of virtual machines, it is more advantageous to use containers with the addition of Docker software in which VirtualBox will work. **Practical relevance:** This technology allows you not only to significantly save RAM, but also to increase the calculation speed.

Keywords — Virtual Machines, Cluster, Hypervisor, Container, VirtualBox, Docker.

Citation: Gordeev A. V., Gorelik D. V. Comparative Testing of Container and Hypervisor Virtualizations. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2018, no. 2, pp. 60–66 (In Russian). doi:10.15217/issn1684-8853.2018.2.60

References

1. Gordeev A. V. Organization of Remote Work of Students with their Virtual Machines. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2017, no. 2, pp. 96–100 (In Russian). doi:10.15217/issn1684-8853.2017.2.96
2. Gorelik D. V. Using Containers to Build a System for Working with Remote Virtual Machines. *Semidesiataia Mezhdunarodnaia studencheskaia nauchnaia konferentsiia. Ch. 2. Tekhnicheskie nauki* [Proc. Seventieth Intern. Student Scientific Conf. SUAI. Part 2. Technical Science]. Saint-Petersburg, GUAP Publ., 2017, pp. 66–69 (In Russian).
3. *Ponimaia Docker* [Docker Overview]. Available at: https://habrahabr.ru/post/253877 (accessed 15 November 2017).
4. Adrian Mouat. *Using Docker. Developing and Deploying Software with Containers*. O'Reilly, 2015. 335 p.
5. Markelov A. *OpenStack. Znakomstvo s oblachnoi operatsionnoi sistemoi* [OpenStack. Introduction to the Cloud Operating System]. Moscow, DMK-Press Publ., 2016. 248 p. (In Russian).
6. Karl Matthias, Sean Kane. *Docker: Up & Running. Shipping Reliable Containers in Production*. O'Reilly, 2015. 198 p.
7. Jessie Frazelle. *Docker Containers on the Desktop.* Available at: https://blog.jessfraz.com/post/docker-containers-on-the-desktop/ (accessed 07 April 2017).
8. Popov S. *Obrazy i konteinery Docker v kartinkakh* [Images and Containers Docker in Pictures]. Available at: https://habrahabr.ru/post/272145/ (accessed 17 November 2017).