



Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем вычислительных комплексов

Михеев Павел Алексеевич

**Исследование эффективности использования
физических ресурсов
легковесными контейнерами в облачных системах.**

Научный руководитель:
к.ф.-м.н.
В.А. Антоненко

Москва, 2018

Аннотация

Контейнер - изолированная группа процессов в операционной системе, имеющая доступ к ограниченному количеству ресурсов системы. Данная изоляция достигается за счет механизмов ядра ОС, таким образом, контейнеры используют это ядро, не тратя ресурсы на поддержку гостевого ядра.

Контейнеры обладают небольшим временем запуска, большой пропускной способностью, и показатель производительности вычислений в них ближе к показателю вычислений на физических ресурсах. Однако не существует систем, позволяющих исследовать производительность и деградацию производительности контейнеров при масштабировании и сравнить ее с производительностью и деградацией для виртуальных машин.

В данной работе предлагается использовать существующие методики измерения производительности виртуальных машин и применить их к легковесным контейнерам.

Целью работы является разработка системы измерения производительности и ее деградации у виртуальных машин и контейнеров, запущенных на физическом сервере.

Данная система позволит сравнить численно производительность контейнеров и виртуальных машин.

Введение

Облачные вычисления в современной информационной среде занимают все больше и больше пространства. Развитие открытого программного обеспечения по управлению облаками увеличивает количество тех, кто использует облачные системы как для использования внутри компании, так и для предоставления услуг клиентам. Одним из немногих сдерживающих факторов такого роста вовлеченности использования облачных вычислений является цена физического оборудования.

Традиционно облачные вычисления используют виртуальные машины как сущности, которые так или иначе предоставляются пользователю. Так, в случае облака, работающего по модели Infrastructure-as-a-Service (IaaS), пользователю предоставляется полный доступ к заказанной виртуальной машине, которую он настраивает под собственные требования. В облаках типа Platform-as-a-Service предоставляется доступ к интерфейсам некоторых приложений, запущенных в виртуальной машине. В облаках Software-as-a-Service пользователь запрашивает услугу, обработка которой происходит с помощью виртуальных машин, доступа к которым у пользователя нет.

Количество виртуальных машин, которое может быть запущено в облаке, напрямую зависит от количества физических ресурсов. Под количеством физических ресурсов в данной работе будет пониматься совокупное (по всем серверам) количество ядер процессоров, совокупное количество оперативной памяти и совокупный объем жестких дисков системы.

Пусть X - количество доступных физических ресурсов в облаке, а χ_i - количество ресурсов, требующихся для i -ой виртуальной машины. Тогда n - максимальное число виртуальных машин с такими требованиями, которое может быть запущено на данных ресурсах, если $\sum_{i=1}^n \chi_i = X$.

Однако может возникать ситуация, когда количество запрашиваемых

виртуальных машин превосходит максимальное число n , определенное выше. При этом у владельца облака нет возможности или необходимости докупить оборудование, так как, к примеру, такая ситуация возникает редко. Важный момент, что ресурсы виртуальной машины могут быть использованы ее процессами не на 100% процентов. Тогда, если предоставить доступ к этим же физическим ресурсам или их части той виртуальной машине, номер которой превосходит n , то обе эти виртуальные машины смогут осуществлять свои функции. В таком случае $\sum_{i=1}^n \chi_i > X$, но при этом все виртуальные машины размещены.

Введем коэффициент $\theta = \frac{\sum_i \chi_i}{X}$ - коэффициент перекрытия (overlap), являющийся отношением суммы ресурсов, требуемых виртуальным машинам системы, к физическим ресурсам системы.

Разумеется, при увеличении коэффициента перекрытия производительность процессов в виртуальных машинах падает, так как при использовании одних физических ресурсов (в первую очередь, ядер процессора) разными виртуальными машинами исполнение машин одновременно будет невозможно. Из-за этого будет наблюдаться деградация производительности виртуальных машин при увеличении коэффициента перекрытия.

Еще одной быстро занявшей рынок технологией стала легковесная виртуализация. В этой технологии виртуализации гостевая операционная система отсутствует, а все процессы запускаются в рамках ядра хостовой операционной системы. Изоляция подобных процессов друг от друга и ограничение доступных им ресурсов достигается за счет специальных механизмов ядра. Такие процессы и их потомки с наложенными на них ограничениями называются контейнерами. Так же как и процессы в виртуальной машине изолированы от процессов другой виртуальной машины, процессы в контейнере изолированы от процессов других контейнеров. Отличие заключается в том, что системные вызовы в вирту-

альной машине идут в операционную систему машины, тогда как системные вызовы контейнеров идут напрямую к ядру операционной системы, в которой данный контейнер запущен.

Отсутствие необходимости поддерживать гостевую операционную систему обладает как достоинствами, так и недостатками. Подробнее о них будет сказано ниже.

Контейнеры могут быть использованы в облачных системах так же, как и виртуальные машины. Они потребляют некоторое количество ресурсов, ресурсы могут разделять между несколькими контейнерами, а при увеличении коэффициента перекрытия будет наблюдаться деградация производительности.

В данной работе предлагается исследовать и сравнить производительность виртуальных машин и контейнеров, а так же деградацию производительности машин и контейнеров при увеличении коэффициента перекрытия. Для этого была разработана система, способная запускать виртуальные машины или контейнеры и запускать в них приложения, позволяющие измерять производительность, а так же собирать результаты работы этих приложений.

Основной гипотезой данной работы является следующее: контейнеры обладают более высокой производительностью по сравнению с виртуальными машинами, деградация производительности виртуальных машин при увеличении коэффициента перекрытия происходит быстрее, чем у контейнеров.

Глава 1.

Постановка задачи

Цель работы

Разработать и реализовать систему, позволяющую сравнить производительность виртуальных машин и контейнеров, а так же деградацию производительности виртуальных машин и контейнеров при увеличении коэффициента перекрытия.

С помощью разработанной системы провести эксперименты, позволяющие проверить следующую гипотезу: **контейнеры обладают более высокой производительностью по сравнению с виртуальными машинами, деградация производительности виртуальных машин при увеличении коэффициента перекрытия происходит быстрее, чем у контейнеров.**

План решения задачи

1. Сравнить различные технологии виртуализации.
2. Составить обзор существующих решений, с помощью которого выделить методики оценки производительности виртуальных машин.
3. Выбрать из предыдущего обзора методику оценки производительности

сти или разработать свою, которую возможно применить для оценки производительности контейнеров.

4. Разработать систему, реализующую методику оценки производительности из предыдущего пункта, позволяющий численно оценить производительность и ее деградацию при увеличении коэффициента перекрытия в случае виртуальных машин и контейнеров.
5. С помощью разработанной системы провести эксперименты, позволяющие проверить следующую гипотезу: **контейнеры обладают более высокой производительностью по сравнению с виртуальными машинами, деградация производительности виртуальных машин при увеличении коэффициента перекрытия происходит быстрее, чем у контейнеров.**

Глава 2.

Технологии виртуализации

В данном разделе под хостовой операционной системой будет пониматься система, в которой могут быть запущены гостевые операционные системы. Гостевые операционные системы - это те системы, которые видят лишь свое изолированное окружение, и которые не могут быть осведомлены о наличии других гостевых систем, кроме как через сеть. Под виртуальной сущностью будет пониматься тот процесс в хостовой операционной системе, который исполняет вычисления гостевой операционной системы.

Гипервизор - специализированное программное обеспечение, которое занимается управлением гостевыми операционными системами: запуском, остановкой, наблюдением и выделением ресурсов. Гипервизоры бывают двух типов:

1. Нативные гипервизоры, которые запускаются напрямую на оборудовании хоста, контролируют это оборудования и осуществляют наблюдение за гостевыми операционными системами.
2. Гипервизоры, которые запускаются поверх операционной системы хоста и осуществляют мониторинг гостевой операционной системы.

Виртуализация - такой подход к организации вычислений, при котором каждая виртуализированная сущность изолирована от других, при-

чем ей может быть доступна лишь часть общих ресурсов. В данном разделе будет рассматриваться виртуализация центрального процессора, то есть каким образом возможно исполнять гостевую операционную систему изолированно на центральном процессоре хостовой системы.

Существует четыре основных вида виртуализации: полная виртуализация, паравиртуализация, аппаратная виртуализация и легковесная виртуализация. Далее будут рассмотрены каждый из этих видов, а также их достоинства, недостатки и применимость в облачных системах.

2.1. Полная виртуализация

Данный вид виртуализации является исторически первым. Основная его особенность заключается в том, что гостевая операционная системы полностью отделяется от управления инфраструктурой хоста. Гостевая ОС не требует никаких изменений, и не осведомлена, что запущена в виртуальном окружении.

Как следует из названия, данный вид виртуализации позволяет запустить любую гостевую операционную систему в любой хостовой. Единственное требование - это наличие динамического транслятора из машинного языка архитектуры, с которой работает гостевая операционная система, в машинный язык хостовой архитектуры. В основе работы данного вида виртуализации лежит принцип динамической трансляции [1].

Преимуществом данного вида виртуализации является гипотетическая возможность запускать любую гостевую операционную систему.

При этом основное преимущество оборачивается и основным недостатком. При современном разнообразии вычислительной техники невозможно иметь трансляторы с любого машинного языка в любой. Но даже при наличии транслятора возможно, что скорость исполнения гостевого кода будет гораздо медленнее, чем в случае исполнения на настоящей ар-

хитектуры. Дополнительно к этим недостаткам добавляются сложность реализации динамической трансляции, связанной, к примеру, с неразличимостью команд и данных.

При этом существует ряд гипервизоров, поддерживающих полную виртуализацию. Примерами могут быть VMware ESXi [?], Microsoft Virtual Server [1]. Стоит отметить, что в силу закрытости данного программного обеспечения, их исследование в данной работе произведено не будет.

2.2. Паравиртуализация

Проблемой динамической трансляции является исполнение привилегированных инструкций гостевой операционной системы. При обработке данных инструкций работа гостевой операционной системы ухудшается.

Паравиртуализация пытается исправить данный недостаток. Это подход подразумевает модификацию гостевой операционной системы таким образом, чтобы исполнение привилегированных инструкций было оформлено как системный вызов в хостовую операционную систему. При этом при использовании паравиртуализации производительность гостевой операционной системы выше, чем при использовании полной виртуализации [3].

Основным же недостатком данного вида виртуализации является необходимость модификации ядра гостевой системы. Это уменьшает количество операционных систем, которые могут быть виртуализированы с помощью данного подхода.

Наиболее популярные гипервизоры, поддерживающие паравиртуализацию: Xen [4] и VMware [1]. Однако в данной работе они не будут рассматриваться. Это связано с тем, что технология Xen практически не поддерживается сообществом, и, как следствие, не используется в облачных системах. Продукты же VMware являются закрытыми, что так же

не позволяет их использовать.

2.3. Аппаратная виртуализация

Подход аппаратной реализации подразумевает совпадение машинного языка гостевой системы и хостовой. При этом архитектура процессора хостовой операционной системы должна поддерживать аппаратную виртуализацию, то есть исполнять код гостевой системы так, будто бы это код хостовой. Привилегированные вызовы автоматически отслеживаются гипервизором, и при обнаружении исполняются на процессоре напрямую [1].

Такой подход позволяет решить проблему необходимости модификации ядра для повышения производительности гостевой системы, а также использование полной виртуализации, хоть и требует специфической архитектуры процессора хостовой системы. Однако данная технология есть почти во всех современных процессорах, что позволяет использовать данный вид виртуализации в облачных системах.

Аппаратная виртуализация поддерживается множеством гипервизоров. Основные представители - это VirtualBox [5], KVM [6], VMware ESXi [2]. В данной работе будет рассматривать гипервизор KVM, который является модулем ядра Linux.

2.4. Общие достоинства и недостатки

Такой подход имеет ряд достоинств и недостатков. С одной стороны, поддержка гостевого ядра расширяет набор возможных операционных систем, которые можно запустить. С другой стороны, поддержка гостевого ядра требует ресурсов. Так, гостевая операционная система будет потреблять часть выделенной оперативной памяти, часть выделенного

дискового пространства и часть ресурсов процессора. При запуске виртуальной машины будет тратиться время на запуск ядра ОС. Оперативная память, выделяемая гипервизором под виртуальную машину, чаще выделяется непрерывным участком, что может приводить к внутренней фрагментации памяти.

Глава 3.

Обзор существующих решений

Привет, git. Почему ты не работаешь?

Литература

- [1] Hyungro Lee. Virtualization Basics: Understanding Techniques and Fundamentals. // School of Informatics and Computing, Indiana University, 2014.
- [2] Homepage of VMWare ESXi. <http://www.vmware.com/ru/products/esxi-and-esx.html> (дата обращения 01.09.2017)
- [3] Hasan Fayyad-Kazan, Luc Perneel, Martin Timmerman. Full and Para-Virtualization with Xen: A Performance Comparison. // Journal of Emerging Trends in Computing and Information Sciences, 2013.
- [4] Homepage of Xen. <https://www.xenproject.org> (дата обращения 01.09.2017)
- [5] Homepage of VirtualBox. <https://www.virtualbox.org> (дата обращения 01.09.2017)
- [6] Homepage of KVM. <https://www.linux-kvm.org> (дата обращения 01.09.2017)
- [7] Wes Felter, Alexandre Ferreira [и др.]. An Updated Performance Comparison of Virtual Machines and Linux Containers. // IBM Research Report, 2014.
- [8] George Kousiouris, Tommaso Cucinotta, Theodora Varvarigou. The effects of scheduling, workload type and consolidation scenarios on

virtual machine performance and their prediction through optimized artificial neural networks. // The Journal of Systems and Software, 2011.

- [9] Anton Beloglazov, Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. // Wiley Online Library, 2011.
- [10] Haikun Liu, Hai Jin, Cheng-Zhong Xu, Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. // Springer US, 2013.
- [11] Homepage of Qemu. <http://www.qemu.org> (дата обращения 01.09.2017)
- [12] Homepage of LXC. URL: <https://linuxcontainers.org/ru/> (дата обращения 01.09.2017)
- [13] Homepage of OpenVZ. URL: <https://openvz.org/> (дата обращения 31.10.2017)
- [14] Homepage of Rocket. URL: <https://coreos.com/rkt/docs/latest/> (дата обращения 31.10.2017)
- [15] Homepage of DrawBridge. URL: <http://research.microsoft.com/en-us/projects/drawbridge/> (дата обращения 31.10.2017)
- [16] Homepage of runC. URL: <https://runc.io/> (дата обращения 30.04.2017)
- [17] Homepage of Docker. URL: <https://docs.docker.com/> (дата обращения 01.09.2017)