УДК 004.3

Артамонова А.А.

бакалавр по направлению «Информационные системы и технологии»,

Национальный исследовательский ядерный университет «МИФИ»,

Российская Федерация, Москва

СРАВНЕНИЕ ГИПЕРВИЗОРНОЙ И КОНТЕЙНЕРНОЙ ТЕХНОЛОГИЙ ВИРТУАЛИЗАЦИИ

Аннотация: *в статье приводится описание двух технологий виртуализации – гипервизорной и контейнерной, анализируются преимущества и недостатки каждой из этих технологий. Рассматриваются варианты применения гипервизорной и контейнерной виртуализации с точки зрения повышения эффективности используемых ресурсов в различных сферах использования вычислительных платформ.*

Ключевые слова: *виртуализация, гипервизоры, контейнеры.*

COMPARISON BETWEEN HYPERVISORS AND CONTAINER-BASED TECHNOLOGIES OF VIRTUALIZATION

Annotation: *The article describes two virtualization technologies: hypervisors and container-based virtualization, analyzes the advantages and disadvantages of each of these technologies. Hypervisors and container-based virtualization application are considered from the point of view of increasing the efficiency in various spheres of the use of computing platforms.*

Key words: *virtualization, hypervisors, container-based virtualization*

Введение

В результате появления все более мощных вычислительных платформ растет пропускная способность компьютерных интерфейсов, и вследствие

этого возникает возможность перенести большинство серверов, функционирующих в организациях, в виртуальное пространство. Это достигается при помощи технологий виртуализации. Виртуализация – это процесс миграции физических систем в виртуальную среду. Другими словами, это создание виртуальной версии устройства или ресурса; фактически, чего угодно – от сервера до операционной системы. Предоставляя виртуальное представление вычислительных ресурсов, эти технологии позволяют превратить один физический сервер в узел для группы виртуальных серверов, которые используют одни и те же ресурсы. С помощью виртуализации можно мгновенно получить доступ к почти неограниченным вычислительным ресурсам, которые могут обеспечить широкие возможности. Использование этих технологий позволит сократить как количество физических серверов, так и количество системных администраторов благодаря меньшей потребности в обслуживании и управлении, из-за чего появится возможность уменьшить общие накладные расходы на IT-инфраструктуру компании.

Разработчики первых операционных систем с разделением времени столкнулись с противоречием между возможностью изолировать приложения и необходимостью совместного использования системных ресурсов, решив его при помощи создания иллюзии, что каждое приложение обладает своей собственной физической машиной, но одновременно обеспечив возможность использовать совместно с другими приложениями определенные объекты (например, файлы) [1]**Ошибка! Источник ссылки не найден.**. Другими словами, каждое приложение работает на «виртуальной» физической машине.

Возможности использования существующих методов виртуализации зависят от уровня, на котором они работают. Согласно эталонной модели виртуализации, она может работать на уровне системы команд (ISA) или на двоичном интерфейсе приложений (ABI). Поэтому методы виртуализации

можно разделить на виртуализацию на уровне операционной системы и на уровне процессора (или аппаратную виртуализацию).

Полная виртуализация и паравиртуализация

Основные используемых технологии в области аппаратной виртуализации – это полная виртуализация и паравиртуализация. В первом случае виртуализированный интерфейс, предоставляемый гостевой операционной системой, идентичен главному компьютеру, поэтому для работы на гостевой виртуальной машине не нужно модифицировать операционную систему. Паравиртуализация, как следует из названия, предоставляет интерфейс, который отличается от хост-машины; в этом случае для работы на виртуальных машинах необходимо специальным образом интегрировать код виртуализации в само ядро операционной системы. Следует отметить, что для виртуализации на аппаратном уровне требуется специальное программное обеспечение, называемое гипервизором (или VMM, от англ. Virtual Machine Manager – менеджер виртуальных машин). Существует два типа гипервизоров: первого и второго.

Гипервизор первого типа (Рисунок 1 (а)) – это программное обеспечение, которое запускается непосредственно на физическом оборудовании. После установки и настройки гипервизора можно начать создавать гостевые машины на главном сервере. Архитектурно гипервизор первого типа находится непосредственно на аппаратном уровне и поэтому может напрямую использовать аппаратные ресурсы. Гипервизор первого типа отвечает за выделение памяти виртуальным машинам и за предоставление интерфейса для администрирования и мониторинга. Из-за прямого доступа к главному серверу гипервизор первого типа не требует отдельных циклов ЦП или памяти для виртуальных машин, и, следовательно, обеспечивает большую производительность по сравнению с гипервизорами второго типа. Примерами гипервизоров первого типа являются VMWare,

ESX, XtratuM и Xen.

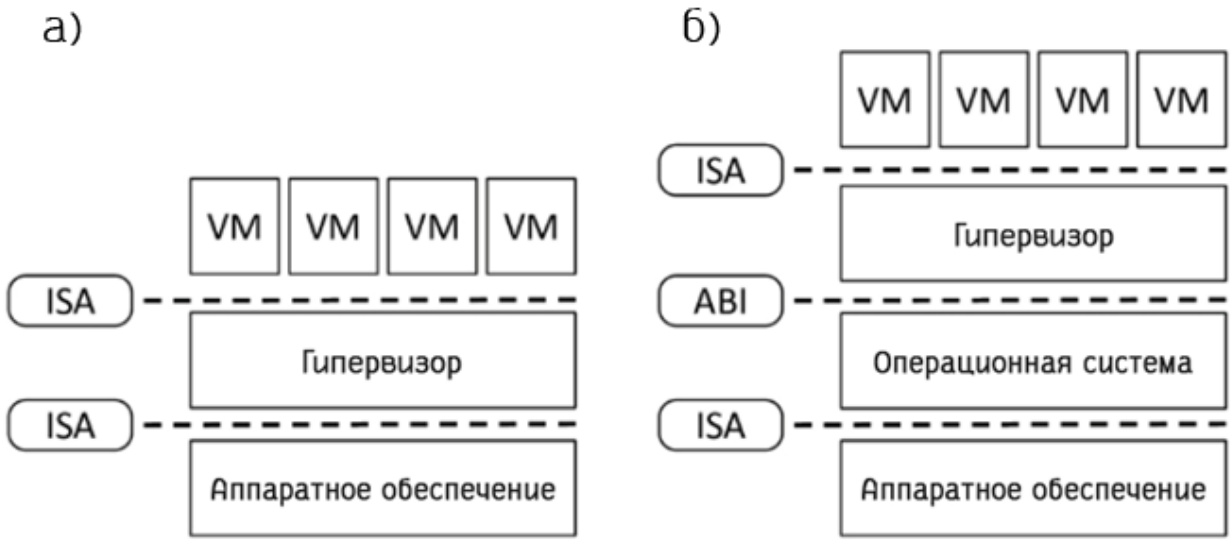

Рисунок 1. Типы гипервизоров

Гипервизоры второго типа (Рисунок 1 (б)) устанавливаются поверх базовой операционной системы, работающей на физическом сервере – например, Linux или Windows. В этом случае хостовая операционная система работает непосредственно на оборудовании, а сам гипервизор представляет собой программный уровень, на котором могут быть запущены различные гостевые операционные системы. Гипервизор предоставляет каждой из работающих на хост-системе виртуальных машин такие ресурсы, как виртуальный BIOS, виртуальные устройства и виртуальную память. Такие гипервизоры проще в использовании и управлении, но имеют более высокие расходы системных ресурсов по сравнению с гипервизорами первого типа. Они особенно полезны в ситуациях, когда пользователь не хочет выделять под виртуализацию весь сервер. Примерами гипервизоров второго типа являются Virtualbox, Oracle VM Server и KVM.

Достоинства и недостатки гипервизоров

Согласно [2]**Ошибка! Источник ссылки не найден.**, можно выделить

четыре основные особенности гипервизоров:

Прозрачность. Это означает, что программное обеспечение может выполняться в среде виртуальной машины напрямую. Другие функции гипервизора могут быть легко применены к любому программному обеспечению, запущенному на виртуальной машине без изменений.

Изоляция. Гипервизор позволяет размещать несколько экземпляров виртуальной машины на физическом сервере. Программное обеспечение может обмениваться аппаратными ресурсами через виртуальную машину, но также может быть защищено и изолировано. Каждая виртуальная машина может установить свою собственную версию программного обеспечения без необходимости учитывать совместимость с программным обеспечением, установленным на других виртуальных машинах. Каждая виртуальная машина также поддерживает разделенную среду выполнения, одновременно предотвращая сбой программного обеспечения, вызванный программным обеспечением, выполняемым на других виртуальных машинах.

Инкапсуляция. Вся система виртуальной машины помещена на виртуальный жесткий диск, который является обычным файлом главной машины. С помощью этого типа инкапсуляции установка, резервное копирование и восстановление виртуальных машин так же просто, как копирование и вставка файла в операционной системе, что может эффективно снизить сложность конфигурации и развертывания систем и повысить гибкость программного обеспечения.

Управляемость. Для операций виртуальной машины, таких как загрузка, выключение, спящий режим и даже добавления, изменения или удаления виртуального оборудования, есть программный интерфейс. Виртуальная машина может полностью управляться через программный интерфейс.

Несмотря на вышеперечисленные достоинства, гипервизорная виртуализация не лишена недостатков. Ссылаясь на [3]**Ошибка! Источник**

ссылки не найден., можно выделить следующие из них:

- превышение физических ресурсов при выделении памяти – потенциальная проблема в виртуализированной среде. Она может привести к снижению производительности, особенно во время пиковых нагрузок;
- портирование существующих приложений на виртуальную среду может привести к большим первичным затратам;
- выход из строя одного гипервизора может означать потерю всех виртуальных машин, установленных на нем;
- фактически, количество серверов не уменьшается – просто часть из них переходит в виртуальное пространство.

Контейнерная виртуализация

Виртуализация на уровне операционной системы – это метод серверной виртуализации, в котором ядро ОС допускает существование нескольких изолированных экземпляров пользовательского пространства, а не только одного. Такие экземпляры, которые иногда называются контейнерами или программными контейнерами [4], могут выглядеть и восприниматься как реальный сервер с точки зрения его владельцев и пользователей. В Unix-подобных операционных системах эта технология может рассматриваться как усовершенствованная реализация стандартного механизма chroot. Помимо механизмов изоляции, ядро часто предоставляет функции управления ресурсами, чтобы ограничить влияние действий одного контейнера на другие. Контейнерная виртуализация использует возможности ядра для создания изолированной среды для процессов. В отличие от виртуализации на основе гипервизоров, контейнеры не получают свое собственное виртуальное оборудование, а используют аппаратное обеспечение хостовой системы. Поэтому программное обеспечение, запущенное в контейнерах, напрямую общается с ядром хоста и должно

работать на той операционной системе (см. Рисунок 2) и архитектуре процессора, на которой работает хост. Отсутствие необходимости эмулировать аппаратное обеспечение и загружать полную операционную систему позволяет контейнерам запускаться за несколько миллисекунд и быть более эффективными, чем классические виртуальные машины. Образы контейнеров обычно меньше образов виртуальных машин, поскольку они не всегда должны содержать полную инструментальную цепочку для запуска операционной системы (драйверы устройств, ядро или систему инициализации). Это одна из причин, почему виртуализация на основе контейнеров набирает все большую популярность.

Примерами контейнерной виртуализации являются LXC, FreeBSD Jail, OpenVZ и Docker. Все инструменты контейнеризации основываются на двух механизмов ядра Linux: пространства имен (namespaces) и контрольные группы (cgroups).

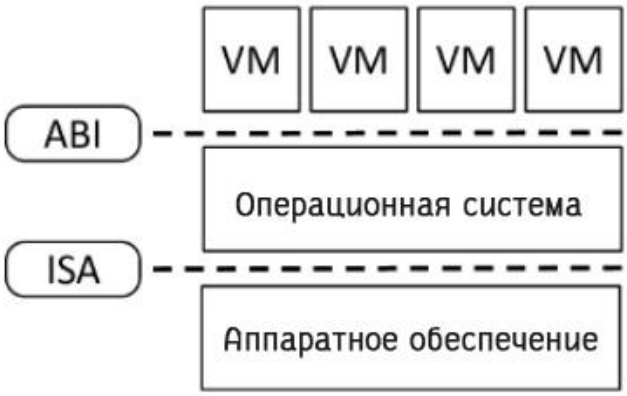

Рисунок 2. Контейнерная виртуализация

Механизм пространств имен (англ. namespaces) позволяет изолировать процессы друг от друга. Принцип состоит в том, чтобы создать специальную среду для процесса, которая позволит ему получить доступ к специальной копии этой информации, которая не должна быть той же самой, которую видят другие процессы. Но применение простого подхода, заключающегося в запрете процессу доступа к этой информации, может привести к сбою или неправильному поведению [5]. На самом деле, пространства имен ядра

являются основой разделения процессов и, следовательно, одной из ключевых концепций для реализации виртуализации на основе контейнеров. Они предоставляют множество функций, необходимых для сборки контейнеров и доступны начиная с ядра 2.6.26, что означает, что они существуют уже несколько лет и уже используются в производстве [6].

Контрольные группы (cgroups) – это механизм для отслеживания процессов и групп процессов [7]. Он не решает проблему, связанную с отделением процессов или созданием изоляции, а ограничивает и изолирует вычислительные ресурсы для групп процессов. Возможность выделять ресурсы для процессов и групп процессов и управлять ими позволяет планировать и контролировать использование контейнеров без неограниченной потери физических ресурсов с помощью простого контейнера. Точно так же можно гарантировать, что ресурсы не будут недоступны, потому что другие процессы заявляют о них сами.

Преимущества контейнерной виртуализации

Согласно [5], ниже перечислены функции, которые делают контейнер полезным и привлекательным для разработчиков приложений и администраторов инфраструктуры:

Быстрое развертывание приложений. Рабочий процесс контейнеров облегчает разработчикам и системным администраторам совместную работу, так как развертывание приложений в производственной среде может происходить очень быстро. Из-за стандартного расширения контейнеров только разработчикам приходится беспокоиться о приложениях, запущенных внутри него, а системным администраторам необходимо думать о развертывании контейнеров только на серверах. Такое хорошо разделенное управление контейнером приводит к более быстрой доставке приложений. Кроме того, из-за легковесности контейнеров их запуск происходит очень быстро. Все это сокращает время тестирования, разработки и развертывания.

Масштабируемость. Контейнеры могут работать практически на любой системе Linux, а также могут быть развернуты в облачных средах, настольных компьютерах, на базовых центрах обработки данных, на физических серверах и так далее. Перемещать контейнеры из своей рабочей среды в облако и обратно на физические серверы можно легко и быстро. Еще один интересный фактор, связанный с контейнерами, – масштабируемость. Таким образом, контейнеры идеально подходят для масштабирования приложений, спроектированных и созданных для общедоступных облачных платформ.

Производительность. При замене физических серверов виртуальными число оборудования уменьшается, но увеличение количества серверов не исчезает – просто один тип разрастания заменяется другим. На одном физическом сервере можно запустить в два раза больше контейнеров [8], чем виртуальных машин, поскольку для контейнеров не нужно создавать отдельную полноценную операционную систему. Контейнеры легко контролировать, поскольку все они выполняются на одной операционной системе. В режиме простоя контейнеры не используют ресурсы сервера (память и процессор), в отличие от виртуальной машины, которая захватывает эти ресурсы при запуске.

Легкость обслуживания. Управление жизненным циклом виртуальных машин не является простым делом, так как на каждом сервере есть как минимум две операционные системы, требующие исправления и обновления – гипервизор и гостевая ОС внутри виртуальной машины. При наличии сервера с двадцатью виртуальными машинами необходимо заботиться об обслуживании двадцать одной ОС (двадцать гостевых и гипервизор).

Микросервисы. Гипервизорная технология, в отличие от контейнерной, не подходит для микросервисов, которые могут использовать сотни тысяч процессов, поскольку для каждого процесса ОС требуется отдельная

виртуальная машина.

Переносимость. Поскольку контейнеры переносимы, приложения могут быть объединены в единый блок и затем развернуты в различных средах без внесения каких-либо изменений в сам контейнер.

Несмотря на все преимущества контейнеров, остаются области, где гипервизорную виртуализацию в принципе нельзя заменить контейнерной. Так как контейнер, по сути, представляет собой изолированную копию хостовой ОС со своим пользовательским окружением и базовыми ресурсами, отсутствует возможность глубокой настройки или самостоятельной установки ОС, а также использование ОС, отличной от хоста.

Использованные источники:

1. Soltesz S, Potzl H, Fiuczynski ME, Bavier A, Peterson L. Container-based operating system virtualization: ascalable, high-performance alternative to hypervisors. ACM SIGOPS Operat Syst Rev 2007;41(3):275–87.
2. Jinpeng Huai, Qin Li, Chunming Hu. CIVIC: A Hypervisor based Virtual Computing Environment. IEEE Computer Society, 2007.
3. J. Daniels. Server virtualization architecture and implementation. Crossroads, vol. 16, no. 1, 2009.
4. S. Hogg. Software Containers: Used More Frequently than Most Realize, 2014. [Электронный ресурс]- Режим доступа: http://www.networkworld.com/article/2226996/cisco-subnet/software-containers-used-more-frequently-than-most-realize.html
5. S. R. Alapati. Modern Linux Administration. How to Become a Cutting-Edge Linux Administrator. O'Reilly Media, October, 2016.
6. Docker Docs: Docker Security. August, 2015. [Электронный ресурс]- Режим доступа: https://docs.docker.com/engine/security/security/
7. P. Menage, P. Jackson, C. Lameter. Linux Kernel Documentation. April, 2017. [Электронный ресурс]- Режим доступа: https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

8. M. Xavier, M. Neves, F. Rossi, T. Ferreto, T. Lange, C. De Rose. Performance evaluation of container-based virtualization for high performance computing environments. In Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on, Febuary, 2013.