ПРОБЛЕМЫ ДИНАМИЧЕСКОГО СОЗДАНИЯ И МОНИТОРИНГА ВИРТУАЛЬНЫХ МАШИН

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Побыванец Е. Н.

Волорова Н. А. – канд. техн. наук, доцент

Аппаратные мощности современных вычислительных систем постоянно растут, что неудивительно. Как следствие, программы, которые раньше требовали значительных ресурсов для своего выполнения, теперь потребляют только часть ресурсов вычислительной системы (и не редки случаи, когда это лишь малая часть). Многие приложения теперь можно (а в большинстве случаев и запускаются) одновременно на одном сервере, при этом иногда из виду упускаются проблемы безопасности или распределения нагрузки между приложениями. Одним из возможных решений данной проблемы может стать виртуализация.

Виртуализация – это логическое объединение некоторых ресурсов системы в отдельное изолированное программное окружение (Application Environment). При этом стоит отметить, что изолирование может происходить над различными ресурсами системы, в частности:

- Виртуализация оборудования (Виртуальная машина)
- Виртуализация представлений (Терминальные серверы \ службы)
- Виртуализация программного обеспечения (Портативные приложения, с эмуляцией необходимых сервисов)
- Виртуализации памяти (Изолированная область памяти, применяется практически в каждой ОС)
- Виртуализация хранилище данных (Представление нескольких носителей данных как единый носитель)
- Виртуализация базы данных (Абстрагирование от конкретной СУБД для представления чистых данных)
- Виртуализация сети (в частности, VPN)

В контексте решения поставленной проблемы, наибольший интерес представляют первые 3 подхода. Виртуализация представлений является достаточно знакомым подходом – это терминальные сервера или терминальные службы. Суть этого подхода весьма проста - терминальный сервер представляет своим клиентам ресурсы для выполнения определенного приложения, и все вычисления происходят на стороне сервера, в то время как клиенту показывается лишь результат или представление. Такая модель имеет ряд преимуществ: клиенты могут использовать практически любое аппаратное обеспечение, не требуется большая пропускная способность канала, терминальные системы осуществляют проверку действия пользователя, что повышает безопасность, и из-за того, что все клиенты подключаются к одному терминальному серверу, облегчается обслуживание и поддержка сервера. Но есть и недостатки: так как множество пользователей работают с одним сервером, увеличиваются требования к аппаратному обеспечению сервера, проблемы сервера сказываются на всех пользователях, а также пользователи в большинстве случаев работают с одинаковым набором ПО.

Виртуализация программного обеспечения – относительно новый подход. Он подразумевает под собой то, что приложение запускается в некоторой изолированной среде, внешне похожей на обычную структуру папок (часто называемой «Песочницей» или «Sandbox»). Для каждой «песочницы» эмулируются необходимые ресурсы системы (сервисы, драйвера, записи реестра), что позволяет запускать даже конфликтующие приложения в своих изолированных средах. Чаще всего виртуальные приложения доставляются на компьютер из определенного контейнера или поставщика, что позволяет контролировать актуальность версий данных приложений. Рассматривая плюсы данного подхода, легко выделить следующие плюсы: безопасность работы с приложениями, независимость от конфликтных приложений, простота администрирования. Но при этом стоит отметить, что данный подход имеет некоторые сложности с пониманием и начальной настройкой, что затрудняет его интеграцию.

И самый интересный подход – виртуализация оборудования. Данный подход позволяет создавать полностью изолированные виртуальные машины, полностью эмулирующую работу нормального сервера \ компьютера. Реализуется этот подход с помощью установки на сервер программы-гипервизора – программы, обеспечивающей создании параллельно работающих операционных систем и предоставляющий набор инструментов для управления этими машинами и общими ресурсами.

В ходе работы я разработал прототип веб-приложения, позволяющего создавать такие виртуальные машины обращаясь уже к предварительно установленному на сервер гипервизору (Microsoft Hyper-V), используя веб-интерфейс и предустановленные образы с необходимым ПО и анализируя возникающие проблемы, а также производительность гостевых ОС.

Как показывает анализ, при разработке и использовании таких систем возникают следующие проблемы

- Доступность – при реализации возникают проблемы с доступом к гостевой ОС расположенной на другом сервере.
- Мониторинг состояний виртуальных машин – по умолчанию, MS Hyper-V предоставляет не

самый легкий способ получения текущего состояния виртуальной машины, а хотелось бы видеть эти данные в динамике, в связи с чем требуется написание отдельного модуля для сборки статистики.

Рассматривая анализ производительности, стоит выявить, что гостевая ОС показывает сравнимую производительность с ОС, схожей по аппаратному обеспечению с созданной гостевой ОС (Используя бенчмарки, разница составляет примерно 4-7%)

В целом виртуализация на уровне оборудования имеет много преимуществ:

- Виртуальные машины не занимают места в серверных стойках
- Достаточно просты в администрировании, если есть доступ к гипервизору
- Предоставляют широкие возможности по автоматизации процесса
- Имеют много возможностей по перераспределению нагрузки и миграции систем

При этом стоит отметить, что даже у этого подхода есть недостатки, присущие виртуализации, основанной на представлении – требуется более мощное аппаратное обеспечение для серверов-хостов, а также в случае отказа физического сервера – отказывают и все виртуальные машины.

Исходя из вышесказанного, можно сделать вывод, что виртуализация имеет много преимуществ, и в определенных ситуациях может стать решением многих проблем. При этом в зависимости от конечной цели можно использовать различные подходы, предоставляющий разный уровень нагрузки как на сервер, так и разный уровень затрат на обслуживание.

Список использованных источников:

1. Электронный ресурс, Hyper-V Started Guide, https://technet.microsoft.com/en-us/library/cc732470%28v=ws.10%29.aspx?f=255&MSPPError=-2147217396
2. Aidan Finn, Mastering Hyper-V Deployment, 2010

ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ ВЕБ-СЕРВИСОВ НА ОСНОВЕ МОДЕЛИ АКТОРОВ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Смоляков К. Ю.

Ганжа В. А. - канд. физ.-мат. наук, доцент

В настоящее время широкое распространение получила парадигма «облачных вычислений». Она подразумевает использование нескольких вычислительных узлов и хранилищ различными клиентскими устройствами: персональными компьютерами, смартфонами, встроенным оборудованием, датчиками и т.д. Интерфейсом и связующим звеном для всех этих устройств обычно выступают TCP/HTTP веб-сервисы. Для того чтобы упростить создание распределенных систем с низким временем отклика и высокой доступностью, используются фреймворки на основе модели акторов: Microsoft Orleans для платформы .NET, Akka для платформы JVM, и другие реализации.

Основной задачей данной работы является обзор и анализ высокоуровневых паттернов проектирования распределенных систем для дальнейшего использования при решении прикладных задач. Основная цель — максимально эффективное использование системных ресурсов и преимуществ программной модели акторов.

Модель акторов впервые была упомянута в 1973 году. Широкое распространение в решении задач параллельного программирования в распределенных системах получила сравнительно недавно. Модель акторов представляет собой систему изолированных друг от друга объектов, которые общаются между собой посредством асинхронной передачи сообщений. Внутреннее состояние каждого актора может быть изменено только самим актором при получении сообщения. Каждая операция по обработке полученного сообщения выполняются в контексте одного программного потока. Таким образом прикладному программисту не нужно беспокоиться о проблемах параллельного программирования.

В дальнейшем для решения прикладных задач модель акторов была расширена. Так, например, в компании Microsoft была разработана библиотека Orleans, которая использует собственную расширенную модель «виртуальных акторов», основными принципами которой являются:

1. **Вечное существование.** В Orleans нет возможности создавать и удалять экземпляры акторов. Акторы являются чисто логическими сущностями и всегда существуют (виртуально), к ним всегда можно обратиться по известному ключу.
2. **Автоматическая активация.** Среда выполнения Orleans автоматически создает и удаляет экземпляры акторов на основе запросов клиентов и имеющихся ресурсов. В любой момент времени у каждого актора может быть несколько активаций, либо не быть ни одной.
3. **Независимость от размещения.** Экземпляр актора может быть создан в разных узлах системы, в зависимости от времени и других факторов. Иногда актор может оставаться чисто виртуальное сущностью, то есть не иметь физического размещения.
4. **Автоматическое масштабирование.** В данный момент Orleans поддерживает два режима