

Виртуализация, контейнеры, ... - что дальше?

Для начала необходим небольшой экскурс в историю. Включаем машину времени (а кому-то вполне достаточно просто напрячь память) и поехали.

Давным-давно... Нет, неправда - все это происходило не более 60 лет назад. Поэтому так: совсем недавно...

Итак, первые компьютеры подразумевали однозадачный режим работы - в силу своего устройства. Далее был придуман многозадачный режим, когда несколько человек работали на одном компьютере (эпоха мэйнфреймов). Кстати, первые гипервизоры появились в 60-70-х годах и использовались в основном для систем с разделением времени, повсеместного применения они не имели. Затем появились первые персоналки и они были именно персональными - за ними как правило работал 1 человек. Понимая, что один компьютер может одновременно выполнять несколько программ (для этого просто необходимо их быстро переключать), были созданы первые резидентные программы (они же TSR-программы). Я помню каким чудом мне показалось когда компьютер **ОДНОВРЕМЕННО** выполняет более 1 программы. Но запускать текстовый редактор и калькулятор одновременно оказалось не интересно и мысль инженеров пошла дальше - они решили исправить концепцию «1 физический компьютер - 1 операционная система» и появились системы виртуализации. Тут подтянулись производители железа - обычные процессоры стали аппаратно поддерживать виртуализацию. Отныне на одном компьютере можно запускать все что заблагорассудится, практически не стесняясь - пара копий Windows с кучей приложений, Linux и еще что-нибудь интересное. Правда, у этого решения есть один недостаток - ограничения самого компьютера, а именно объем памяти (как оперативной, так и постоянной). Умные люди и здесь нашли решение: зачем запускать полноценные виртуальные машины с установленной ОС внутри, драйверами и т.д. когда для работы программы вполне достаточно контейнера, который просто выполняет свою задачу? Так появилась контейнеризация, всем знакомая под названиями Docker. Это позволило увеличить плотность размещения процессов на серверах. «Один контейнер - один процесс» - вот такая философия. Если процесс упал или не нужен - не страшно, запустим снова и уничтожим.

Что же далее, что еще изобретут инженеры и разные умные люди?

А вот тут я уже вступаю на территорию домыслов и предположений, но, надеюсь, это будет по меньшей мере интересно.

Сейчас контейнеры используются для разнообразных задач: запуск приложений и сервисов, тестирование, всякие-разные CI/CD (continuous integration and continuous delivery) и т.д. Все это работает под управлением систем оркестрации - надстроек, позволяющих довольно легко и просто (а главное - автоматически) управлять этими контейнерами. Например, нагрузка на веб-сервис компании увеличилась - система оркестрации самостоятельно включила несколько физических серверов, создала нужные сервисы, раздала настройки для работы, запустила и после того как необходимость в них пропала - отключила все ненужное, включая оборудование чтобы оно не потребляло электроэнергию. Это то что мы имеем на текущий момент.

А что если все это же применить ко всей компьютерной технике. Представьте - вы включили свой компьютер/ноутбук/смартфон/кофейный аппарат - любое оборудование. Запустилась

небольшая встроенная операционная система (ОС-загрузчик)и ... и все. ОС-загрузчик самостоятельно далее проверил и загрузил из сети необходимые образы того программного обеспечения которым вы пользуетесь и запустил их по мере необходимости. Т.е. принцип «любое приложение - это процесс» стал главным и основным. Что это дает? Ну что же, вот первое и самое несложное что приходит в голову:

- «Процессу все равно где он работает» - какая разница где и на каком оборудовании запускать процесс? Он может работать на вашем ноутбуке, где-нибудь в облаке или на смартфоне. Например, вы хотите запустить калькулятор. Но он уже запущен у вашего коллеги - тогда достаточно просто передать ему данные для расчета и получить ответ. Таким образом, человек даже и не поймет что ему всего лишь показался интерфейс, а не сам процесс.

- «Процесс запущен только тогда когда это необходимо». Это даст возможность существенно уменьшить объем памяти в устройствах. Если процесс не нужен - он автоматически через некоторое время убивается. Или замораживается, если не требуется непрерывная работа. Или перемещается на другое устройство или в облако.

- «Процесс отделен от данных» - это позволит быстро перезапускать процесс в случае его отключения по той или иной причине. Таким образом, например, можно сделать очень быстрое обновление процесса. Сначала в фоновом режиме загружается новая версия процесса, затем новая версия запускается и ей передаются данные от старого процесса, который при удачном переключении автоматически завершается. Обновление на лету позволит исключительно быстро устранять ошибки, которые в данный момент могут эксплуатироваться годами!

- «Данные хранятся и передаются в зашифрованном и сжатом виде» - таким образом перехватывать что-либо в канале не имеет смысла. А объем пересылаемых по сети данных не так уж и велик.

- «Всегда хранится несколько копий данных» - это позволит сделать практически 100% надежное хранилище с быстрым доступом. Количество и размещение реплик - автоматическое для увеличения как надежности, так и скорости доступа к данным.

Некоторые процессы можно использовать исключительно как средство отображения результатов. Пример - аналог текущего веб-браузера: одна закладка может обрабатываться прямо на вашем устройстве, а остальные - где-нибудь в облаке и на экран доставляться только результаты для отображения.

Сценарий работы системы на предприятии: утром пользователи включили/разбудили компьютеры (или используют принесенные с собой) - на них автоматически загрузились и запустились необходимые для работы процессы. Если какой-то процесс требует дополнительных ресурсов - он автоматически и абсолютно прозрачно для пользователя переместился в корпоративное облако (специально выделенное оборудование) или на соседние устройства - пока ваш коллега где-нибудь на совещании его компьютер не зря тратит электроэнергию. Не хватает собственных ресурсов? Оркестратор автоматически запросил в облаке необходимое количество ресурсов «на прокат» и после того как необходимость в них исчезла - вернул все ресурсы. Абонентская плата за отработанные ресурсы автоматически списалась.

Сценарий домашнего пользователя: с собой всегда телефон или ноутбук. Пришел на работу - процессы перескочили на вычислительные мощности работодателя, пошел с работы - опять переползли на устройство и/или в облако.

Сценарий хостинг-провайдера: купить кучу железа (часть для хранения данных, часть с большим объемом памяти), запустить систему оркестрации и дать доступ. Все остальное работает автоматически.

Таким образом, функционирование такой системы очень похоже на пузырьки в стакане: они существуют некоторое время и бесследно исчезают. Если вам не нравятся «пузырьки» - можно найти аналоги вроде понятия «кальпа», нынче это модно.

И вот нашел похожие размышления: <http://blog.vadmin.ru/2016/06/10.html>

А вот очень интересная система: <https://www.qubes-os.org>

[что дальше, виртуализация, контейнеры, будущее прекрасно, мысли вслух](#)

From:

<https://wiki.rtzra.ru/> - RTzRa's hive

Permanent link:

<https://wiki.rtzra.ru/sysadmin/whats-next>

Last update: **2018/03/29 22:56**

