Доклад

(c1) Уважаемые присутствующие, вашему вниманию представлен доклад на тему: «Разработка горизонтально масштабируемой системы легковесных виртуальных машин».

(c2) Мир вокруг нас постоянно развивается, появляются всё новые технологии и чтобы оставаться «на плаву» даже самые маленькие компании (и не только из сферы IT) начинают использовать облачные технологии.

Облачные вычисления - это технология распределённой обработки данных в которой компьютерные ресурсы и мощности предоставляются пользователю как интернет-сервис. Т.е. здесь инфраструктура предоставляется как услуга. Соответственно потребителю (компании) предоставляются средства обработки данных, хранения, сетей и других базовых вычислительных ресурсов, на которых потребитель (компания) может развертывать и выполнять произвольное программное обеспечение, включая операционные системы, приложения, web-сервисы и т.д.

Преимущества облачных технологий:

1) Передача облачному провайдеру хранения постоянно увеличивающихся данных.
2) Более быстрое и простое разворачивание проекта и сайта и т.д.
3) Быстрая скорость реакции на изменение нагрузки на систему, т. е. в любой момент времени можно арендовать ещё вычислительные ресурсы при увеличении нагрузки на систему и наоборот, отказаться от некоторого объёма ресурсов при снижении нагрузки на систему.
4) Малая вероятность отказа системы из-за технической неисправности оборудования.
5) Практически полное отсутствие операционных затрат — затрат на текущее содержание оборудования.
6) Практически полное отсутствие капитальных затрат — единовременных изначальных затрат на покупку оборудования.

(c3) Во многих IT компаниях для обработки однотипной информации в больших объёмах используют **микросервисы**. Архитектурный стиль микросервисов — это подход, при котором единое приложение строится как набор небольших сервисов, каждый из которых работает в собственном процессе и общается с остальными, используя легковесные

механизмы, как правило HTTP. Эти сервисы построены вокруг бизнес-потребностей и сами по себе могут быть написаны на разных языках программирования и использовать разные технологии хранения данных.

Сервис представляет собой программный продукт, выполняющий узконаправленные действия, такие как: конвертирование аудио файла из одного формата в другой, выполнение математических расчётов и т.д. Сервис запускается в реальной операционной системе, как правило это операционные системы семейства Windows или Unix - подобные, которые развёрнуты на виртуальных машинах по правилу: один сервис одна виртуальная машина. Для запуска, контролирования виртуальных машин, а также масштабирования всей системы используют гипервизор.

Гипервизор или монитор виртуальных машин — это, как правило, комплекс программ, обеспечивающий параллельное, одновременное выполнение нескольких операционных систем на одном и том же хост – компьютере, где каждая операционная система запущенна в своей виртуальной среде — виртуальной машине.

Как уже упоминалось ранее, сервисы на виртуальных машинах запускаются под управлением операционных систем семейства Windows или Unix-подобных. Windows или Unix-подобные операционные системы на виртуальных машинах занимают достаточно большое количество ресурсов реального хост – компьютера, часто превосходящие затраты на выполнение самого сервиса, т.к. необходимо поддерживать работу самой операционной системы, в которой запущен сервис, а именно, поддерживать графический интерфейс (для Windows), поддерживать работу неиспользуемых драйверов (видеокарты, драйвер принтера, звуковой карты и т. д.). Поэтому для развёртывания даже небольшой системы потребуются серьёзные затраты на аппаратное обеспечение.

В связи с этим в далёком 1990 году появились первые **Unikernels** системы.

Unikernels (или Экзоядро) специализированный образ виртуальной машины с единым адресным пространством, построенный с помощью библиотек операционной системы и выполняющий единственный сервис.

При сборке образа виртуальной машины разработчик выбирает минимальный набор библиотек, который будет поддерживать работоспособность операционной системы, а также необходимые модули и библиотеки для запуска приложения. Все используемые библиотеки, модули и само приложение компилируются в один «фиксированный» загрузочный образ, который будет выполнять только один сервис, без возможности его изменения не

пересобирая образ. Построенные таким способом загрузочные образы будут работать непосредственно на гипервизоре или на оборудовании без промежуточных ОС, таких как Linux или Windows.

Плюсы такого подхода:

1) максимально эффективное использование ресурсов;
2) мгновенный запуск;
3) маленький размер загрузочного образа;
4) не требует установки для работы;
5) минимальное кол-во использования аппаратных ресурсов;
6) минимальные энергозатраты для поддержания одного образа.

Минусы:

1) при падения приложения падает вся ОС;
2) необходимо пересобирать образ ОС при изменении сервиса.

(c4) В рамках данной работы были поставлены цели, представленные на слайде **№4**. Т. е. необходимо разработать:

1) клиентский компонент системы для формирования запросов с данными для обработки.
2) сервер для распределения нагрузки между виртуальными машинами;
3) контроллер для управления виртуальнми машинами на хост-компьютере;
4) консоль администратора для просмотра информации о количестве и загруженности виртуальных машин;
5) дополнения сервиса виртуальной машины.

Т. е. в моей системе используются образы виртуальных машин с экзоядром. В настоящее время разработана уже пара десятков операционных систем, работающих по принципу экзоядра, но была выбрана самая новая разработка в этой области.

(c5) В 2016 году на конференции CppCon был представлен рабочий alpha прототип проекта одномодульной микро операционной системы IncludeOS. IncludeOS предоставляет минимально необходимое, самодостаточное окружение, которое взаимодействует непосредственно с гипервизором и предоставляет загрузчик, ядро системы, минимальный набор библиотек, модулей и драйверов, достаточный для выполнения кода на языке C++, написанный с использованием стандартной библиотеки классов. Окружение компонуется с

предназначенным для выполнения приложением и оформляется в виде загрузочного образа виртуальной машины, образуя готовый облачный сервис. Из систем виртуализации, в которых могут работать подобные окружения, поддерживаются KVM/Linux, VirtualBox и QEMU. Схематичное представление построения загрузочного образа виртуальной машины с готовым микросервисом представлено на слайде **№5**.

IncludeOS, в основном, разрабатывается для выполнение web сервисов. Суммарный размер библиотек и компонентов скомпилированной операционной системы составляет около 1 Мб.

(с6) Формирование данных и запросы на их обработку формируются в клиентском компоненте системы, после чего данные передаются на сервер. Сервер представляет из себя посредника между клиентским компонентом и виртуальными машинами, также сервер собирает информацию о загруженности виртуальных машин для их горизонтального масштабирования. Информация о виртуальных машинах хранится в базе данных.

За работу виртуальных машин на хост-системе отвечает контроллер, его задача — производить запуск, остановку виртуальных машин, а также сбор информации о их загруженности и передачи данной информации на сервер. Для корректной работы системы необходимо, чтобы постоянно была запущенна хотя бы одна виртуальная машина, в связи с этим контроллер при старте запускает первую виртуальную машину.

Для просмотра количества запущенных виртуальных машин и загруженности каждой виртуальной машины разработана консоль администратора.

(с7) Бизнес логика системы строится на обработке данных, полученных от клиента. Пока в системе могут передаваться и обрабатываться только текстовые данные. Данные через транспортный уровень передаются на сервисы, запущенные на виртуальных машинах. После обработки данные также через транспортный уровень передаются обратно клиенту.

Клиентских компонентов в системе может быть огромное количество, в разы превосходящее количество запущенных виртуальных машин, но эта проблема решается на уровне IncludeOS, а именно, очередью входных соединений.

Также в бизнес логику входит консоль администратора для мониторинга состояния виртуальных машин и отображения графиков их состояний.

(c8) Транспортный уровень реализует основную логику системы, а именно, горизонтально масштабирует легковесные виртуальные машины.

В транспортный уровень входят:

1) сервер;
2) база данных;
3) контроллер;
4) виртуальные машины, с запущенными на них сервисами для обработки данных.

Сервер выполняет следующие задачи:

1. Принимает решения о запуске виртуальных машин, а именно, если при получении очередного запроса на обработку не будет ни одной свободной виртуальной машины и в пуле ip адресов базы данные будут свободные ip, то на наиболее свободную в данный момент виртуальную машину отправится запрос на запуск новой виртуальной машины с первым свободным ip адресом из базы.
2. Принимает решения об остановке виртуальных машин на хост — компьютере, а именно, если виртуальная машина несколько раз подряд отправляет своё состояние и эти состояния показывают, что система совсем не загружена, то виртуальная машина считается простаивающей и ей отправляется запрос на выключение.
3. Передаёт исходные данные от клиентского компонента системы для обработки сервисам и получает обработанные данные от виртуальных машин и передаёт эти данные клиентскому компоненту, а именно, если нет простаивающих виртуальных машин и нет свободных ip адресов, то выбирается из базы наименее загруженная виртуальная машина и на неё пересылается запрос на обработку;
4. Собирает информации о состоянии загруженности виртуальных машин, а именно, контроллер через определённые промежутки времени собирает данные о загрузке виртуальных машин и отправляет полученные данные на сервер, это было сделано для того, чтобы снять дополнительную нагрузку с виртуальных машин.

Контроллер виртуальных машин запускается на Unix подобной операционной системе, так как сборку образа IncludeOS с необходимым сервисом можно проводить через командную оболочку, а именно, запуск всех скриптов по сборке и запуску виртуальных машин. Контроллер взаимодействует с виртуальными машинами следующим образом:

1. При инициализации контроллера на хост машине автоматически запускается первая виртуальная машина
2. Через определённые промежутки времени виртуальная машина передает контроллеру информацию о загруженности процессора и используемой оперативной памяти
3. Контроллер через перенаправление вывода консоли виртуальной машины считывает данные
4. При получении новых данных контроллер обрабатывает их и совершает одно из трёх действий:
    1. Запускает виртуальную машину с новым ip
    2. Останавливает виртуальную машину
    3. Отправляет информацию об используемых ресурсах виртуальной машины

Сервис каждой виртуальной машины дополнен модулем, который выполняет следующие действия:

1. При старте считывает из параметров запуска ip адрес, на котором будет расположен сервис
2. В определённый промежуток времени выводит в консоль контроллера информацию о себе, а именно:
    2.1. Использование вычислительной мощности виртуального процессора, занимаемое виртуальной машиной
    2.2. Занимаемый объём оперативной памяти.

(с9) База данных содержит три таблицы:

1) system_params – настройки системы

   Поля:

   1. key – ключ;
   2. value – значение.

   Обязательные значения:

   1. free_count — максимальное количество простаивающих состояний виртуальной машины подряд;
   2. min_vm_count – минимальное количество виртуальных машин, запущенное на контроллере.
   3. last_connect_ms – время последнего обновления состояния виртуальной машины,

после которого её можно считать мёртвой;

4. ip_reserve_ms – максимальное время резервирования ip без проставления внешнего ключа id_ip в vmachine.

2) vmachine - в данной таблице находится информация о запущенных виртуальных машинах.

   Поля:

   1. id_vmachine — первичный ключ таблицы;
   2. id_ip — внешний ключ таблицы ip_pool;
   3. cpu – загрузка процессора в %;
   4. ram - загрузка памяти в байтах;
   5. free_count – значение, показывающее сколько раз подряд виртуальная машина отправляла состояние, указывающее, что она простаивает;
   6. last_connect – время обновления последнего состояния.

3) ip_pool — в данной таблице находятся ip адреса, которые можно выдавать виртуальным машинам, для запуска сервисов.

   Поля:

   1. id_ip — первичный ключ таблицы;
   2. ip – адрес на котором будет доступен сервис;
   3. port — порт на котором будет доступен сервис;
   4. reserve – флаг резервирования записи, выставляется в true, если была команда о запуске новой виртуальной машины с этим адресом и выставляется в false, есл была команда остановки машины;
   5. last_reserve — время резервирования записи, если с момента резервирования прошло больше времени чем ip_reserve_ms, а внешний ключ id_ip в vmachine не проставился ни к одной записи, то флаг reserve снимается.

(c10) coming soon...

(c11) coming soon...

(c12) Спасибо за внимание!