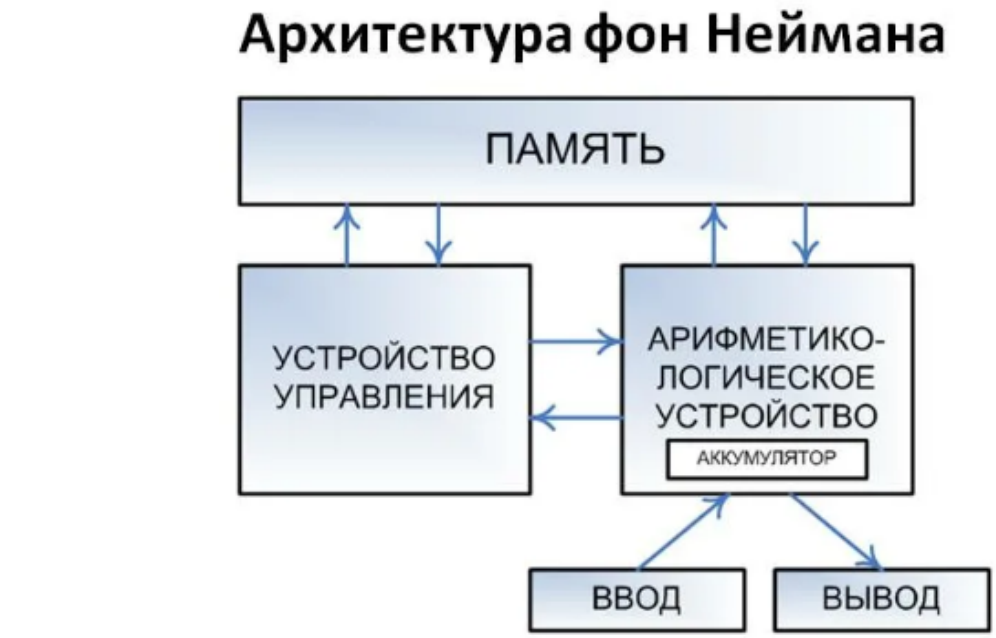
3. Облачная ИТ архитектура и микросервисы

ВданномразделепоказанаэволюцияИТархитектурыотпрограммногоавтомата, выполненного в виде монолитного исполняемого модуля операционной системы (ОС), до современного асинхронного приложения на базе микросервисной архитектуры.

3.1 В основе построения всех современных компьютеров лежит архитектура фон Неймана, суть которой состоит в последовательном выполнении команд, которые хранятся в оперативной памяти и оперируют с данными хранящимися там же.

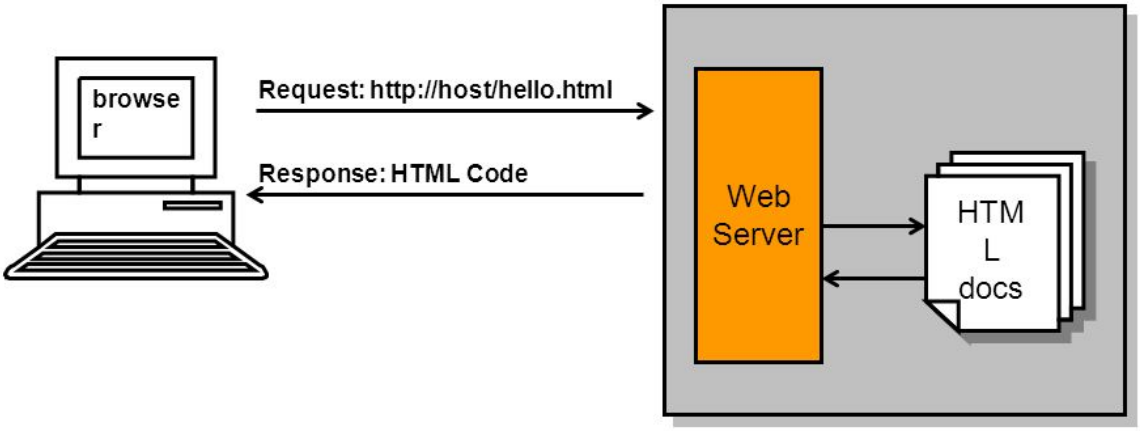


Принципы фон Неймана

- Принцип однородности памяти
- Принцип адресности
- Принцип программного управления
- Принцип двоичного кодирования

3.2 Появление сети Интернет привело к созданию *архитектуры всемирной паутины* — World Wide Web, представляющей из себя гипертекстовые документы, размещенные на веб-серверах и связанные друг с другом с помощью гиперссылок. Конечно, вся эта конструкция, состоит из огромного числа программных модулей и компьютеров, функционирующих в архитектуре фон Неймана, но давайте на мгновение забудем об этом и рассмотрим ее верхнеуровнево. Тогда основными компонентами этой архитектуры будут: веб-страница, веб-сервер, клиентский веб-браузер, сеть Интернет и HTTP протокол. ***Основной акцент сделаем не на структуре сети, а на процессах протекающие в ней.***

WEB архитектура



Основной цикл взаимодействия выглядит следующим образом:

Пользователь через web-браузер запрашивает документ, указывая его адрес в сети (URL), служба доменов (DNS) определяет web-сервер на котором расположен документ и пересылает запрос ему. Модуль маршрутизации web-сервера разбирает URL и выбирает контроллер (controller), ответственный за обработку данного типа запросов и передает запрос ему. Контроллер подготавливает необходимые данные и пересылает их модулю представления (view), который форматирует их в виде HTML документа. Полученная web-страница пересылается в web-браузер пользователя.

Приэтомсерверпараллельноможетобслуживатьмножестворазличных запросов, которые он получает асинхронно от различных пользователей. Было бы неприемлемо, если бы посетители сайта выстраивались в очередь один за другим, как в магазине на кассе. Обслуживание очередного запроса выполняется отдельным процессом операционной системы (fork) — этот объект довольно массивный, он долго стартует при запуске. По этой причине web-сервер после старта инициализирует пул (pool) таких процессов и, далее, выделяет “горячий” процесс из пула для обслуживания очередного входящего запроса (request).

Такимобразом,webархитектурапредставляетизсебякоммуникационнуюсетьв которой **обмен документом между web-сервером и web-браузером происходит в**

синхронном режиме, но сами **запросы от web-браузеров на web-сервер поступают асинхронно**, обслуживание запросов ***происходит в параллельном режиме*** в различных процессах ОС. Сами документы могут быть довольно большого размера, а периоды времени между запросами очередного документа от пользователем тоже достаточно значительные. Ведь человеку нужно время, чтобы ознакомиться со страницей и сделать следующий клик.

3.3 SOAP (от англ. Simple Object Access Protocol) — простой протокол доступа к объектам) — это протокол обмена структурированными сообщениями в распределенной вычислительной среде. ***Данная архитектура представляет собой протокол взаимодействия с web-сервисом и предназначен для обмена сообщениями между приложениями в формате XML (расширяемый язык разметки данных). Концептуально — это такая же коммуникационная сеть что и гипертекст, но она предназначен для обмена сообщениями не между web-серверами и пользователями web-страниц, а между различными приложениями корпорации, где для каждого приложения разрабатывается web-сервис и публикуется на сервисной шине.*** Протокол SOAP упакован в HTML, для его развертывания используется та же самая WEB инфраструктура.

К вопросу о фракталах — сравните архитектуру компьютера (приведена ниже) и архитектуру корпоративный сервисной шины (ESB).

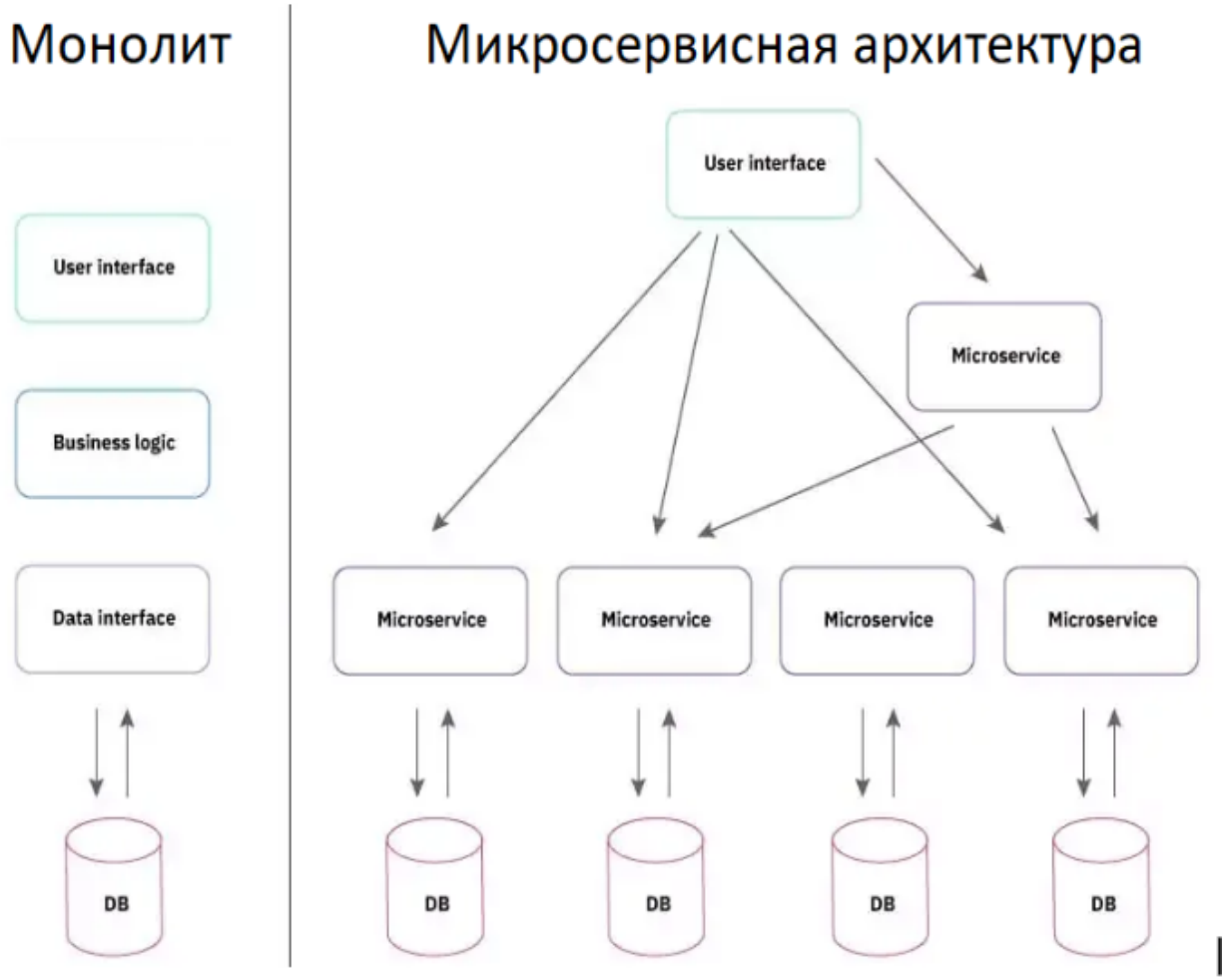
3.4 ***RESTful интерфейс и микросервисная архитектура*** это следующий шаг в эволюции коммуникационных ИТ архитектур.

Главное различие микросервисов и шины в том, что ***ESB была создана в контексте интеграции отдельных приложений***, чтобы получилось единое корпоративное распределенное приложение. ***А микросервисная архитектура предполагает проектирование приложений с разбиением их на микросервисы. Микросервисы — это маленькие автономные сервисы, работающие вместе и спроектированные вокруг бизнес-домена.***

Например, процедуру входа и регистрации на сайте можно разместить в один микросервис, процедуру оплаты в другой и так далее.

Это позволяет эффективно решать проблемы:

- независимое развертывания - не нужно заменять на сервере все приложение полностью, достаточно заменить необходимый микросервис;
- масштабирование нагрузки - при росте нагрузки можно выделить ресурсы точно для необходимого микросервиса;
- эффективная декомпозиция приложения - микросервисы слабо связаны и их разработка и отладка может вестись независимыми группами разработки и с использованием различных инструментов программирования;
- и другие.



REST (от англ. Representational State Transfer — «передача репрезентативного состояния» или «передача „самоописываемого“ состояния») — архитектурный стиль взаимодействия компонентов распределенного приложения в сети. ***Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было легко масштабировать.***

Главным из этих правил является “***Отсутствие состояния”*** - протокол взаимодействия между клиентом и сервером требует соблюдения следующего условия: в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится (Stateless protocol или «протокол без сохранения состояния»). Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса. Состояние сессии при этом сохраняется на стороне клиента. ***Именно это правило является основой для обеспечения слабой связности микросервисов.***

Отметим, что если SOAP - это программный интерфейс, разработчики технологии все еще пытаются имитировать обмен сообщениями по протоколу HTTP, как удаленный вызов одного программного модулями другим (RMI - Remote Method Invocation), то REST - это уже ясно выраженная концепция обмена информационными сообщениями между микросервисами.

REST-интерфейсыещеназываютAPI-интерфейсами.API(ApplicationProgramming Interface) — программный интерфейс приложения.

В качестве примера, можно привести API-интерфейс Центробанка, который возвращает валютные курсы. В качестве входного запроса мы перечисляем нужные нам валюты, а в качестве ответа получаем их курсы. Нас не интересует как устроены системы ЦБ. Мы строим свое приложение, например, чат-бот, которое в нужный нам момент должно получить курсы, чтобы запросы наших пользователей были удовлетворены. Сервисы ЦБ также не волнует состояние и устройство наших систем.

3.5 Тема контейнеризации на первый взгляд несколько в стороне от вышеизложенного, но это только кажется. Если в предыдущих пунктах мы рассматривали ИТ архитектуру с точки зрения парадигмы ее проектирования, то контейнеризация - это взгляд с точки зрения ее использования, фактически это инструменты для превращения операционной системы в облако.

Контейнеризацияисистемыуправленияконтейнерамиосновныеэлементы облачной архитектуры. Контейнер позволяет упаковать приложение со всеми зависимостями и быстро развернуть контейнер для использования - либо на продуктовом сервере, либо на компьютерах команды разработки. ***Как правило в контейнере устанавливаются один или несколько микросервисов некоторой доменной области***, что позволяет эффективно решать задачи бесперебойного обслуживания и масштабирование нагрузки - в нужный момент запускается необходимое количество контейнеров и выделяются нужные ресурсы.

Docker — программное обеспечение ***для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации***, контейнеризатор приложений. ***Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть развернут на***

любой Linux-системе с поддержкой контейнеризации, а также предоставляет набор команд для управления этими контейнерами.

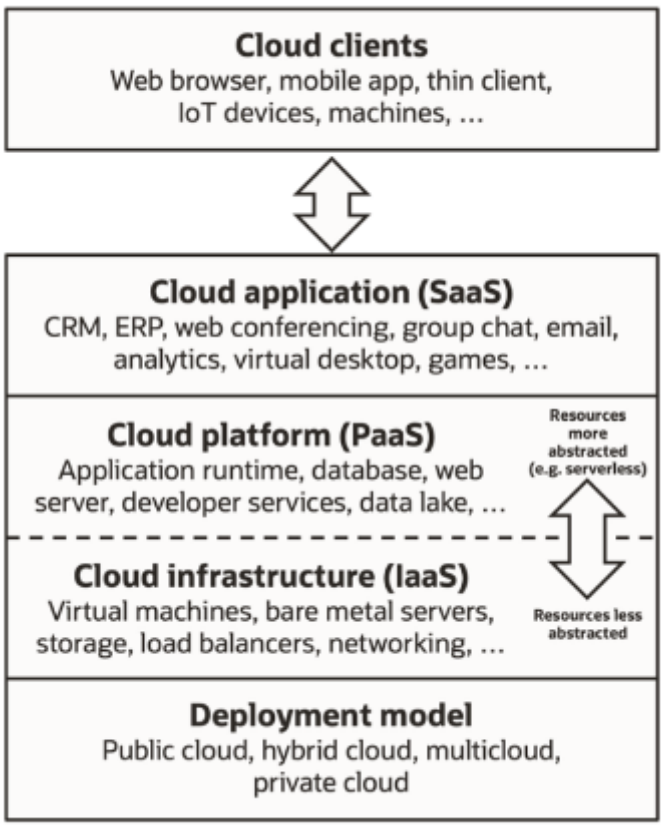
Kubernetes (от др.-греч. κυβερνήτης — «кормчий», «рулевой», часто также используется нумероним k8s) — открытое программное обеспечение для оркестровки контейнеризированных приложений — автоматизации их развертывания, масштабирования и координации в условиях кластера.

DevOps (акроним от англ. development & operations) — методология автоматизации технологических процессов сборки, настройки и развертывания программного обеспечения.

DevOps ***методология предполагает активное взаимодействие специалистов по разработке со специалистами по информационно-технологическому обслуживанию и взаимную интеграцию их технологических процессов друг в друга для обеспечения высокого качества программного продукта.*** DevOps предназначен для эффективной организации создания и обновления программных продуктов и услуг. Методология основана на идее тесной взаимозависимости создания продукта и эксплуатации программного обеспечения, которая прививается команде как культура создания продукта.

Cloud computing - облачные вычисления - ***это доступность ресурсов компьютерной системы по требованию, особенно хранилища данных (облачное хранилище) и вычислительной мощности, без прямого активного управления со стороны пользователя. Большие облака часто имеют функции, распределенные по нескольким местоположениям, каждое местоположение является центром обработки данных.*** Облачные вычисления основаны на совместном использовании ресурсов для достижения согласованности и обычно используют модель "оплаты по мере использования", которая может помочь в снижении капитальных затрат, но также может привести к непредвиденным операционным расходам для неосведомленных пользователей.

Облачная архитектура



Версия #6
GRN создал 18 July 2022 11:29:08
GRN обновил 28 July 2022 18:27:40