МИРЭА
Российский
технологический
университет

Технологические основы Интернета Вещей

Лекция 5. Топология облачных и туманных вычислений

Жматов Дмитрий Владимирович
кандидат технических наук, доцент
доцент кафедры Математического обеспечения и
стандартизации информационных технологий

Понятие облака

Облако обеспечивает возможность иметь простые датчики, камеры, переключатели, и исполнительные устройства, взаимодействуют на общем языке друг с другом.
Облако – общий знаменатель валюты данных.
Понятие «облако» относится к инфраструктуре вычислительных служб, которые обычно требуются по запросу. Набор ресурсов (вычислений, сетей, хранилищ и связанных с ними программных сервисов) может динамически масштабироваться в сторону увеличения или уменьшения в зависимости от средней нагрузки и качества обслуживания. Облака, как правило, представляют собой крупные центры обработки данных, которые предоставляют клиентам услуги, ориентированные на внешнего потребителя, и модель оплаты за использование. Эти центры создают иллюзию единого облачного ресурса, в то время как на самом деле может быть использовано много географически распределенных ресурсов. Это дает пользователю чувство независимости от местоположения. Ресурсы являются эластичными (что означает масштабируемость), а сервисы – это эквивалент платы за использование, неизменный доход для провайдера. Сервисы, которые работают в облаке, отличаются от традиционного программного обеспечения своей конструкцией и реализацией. Облачные приложения могут разрабатываться и развертываться быстрее и в меньшей степени зависеть от изменчивости среды. Таким образом, развертывание облаков происходит очень быстро.

Облачная архитектура

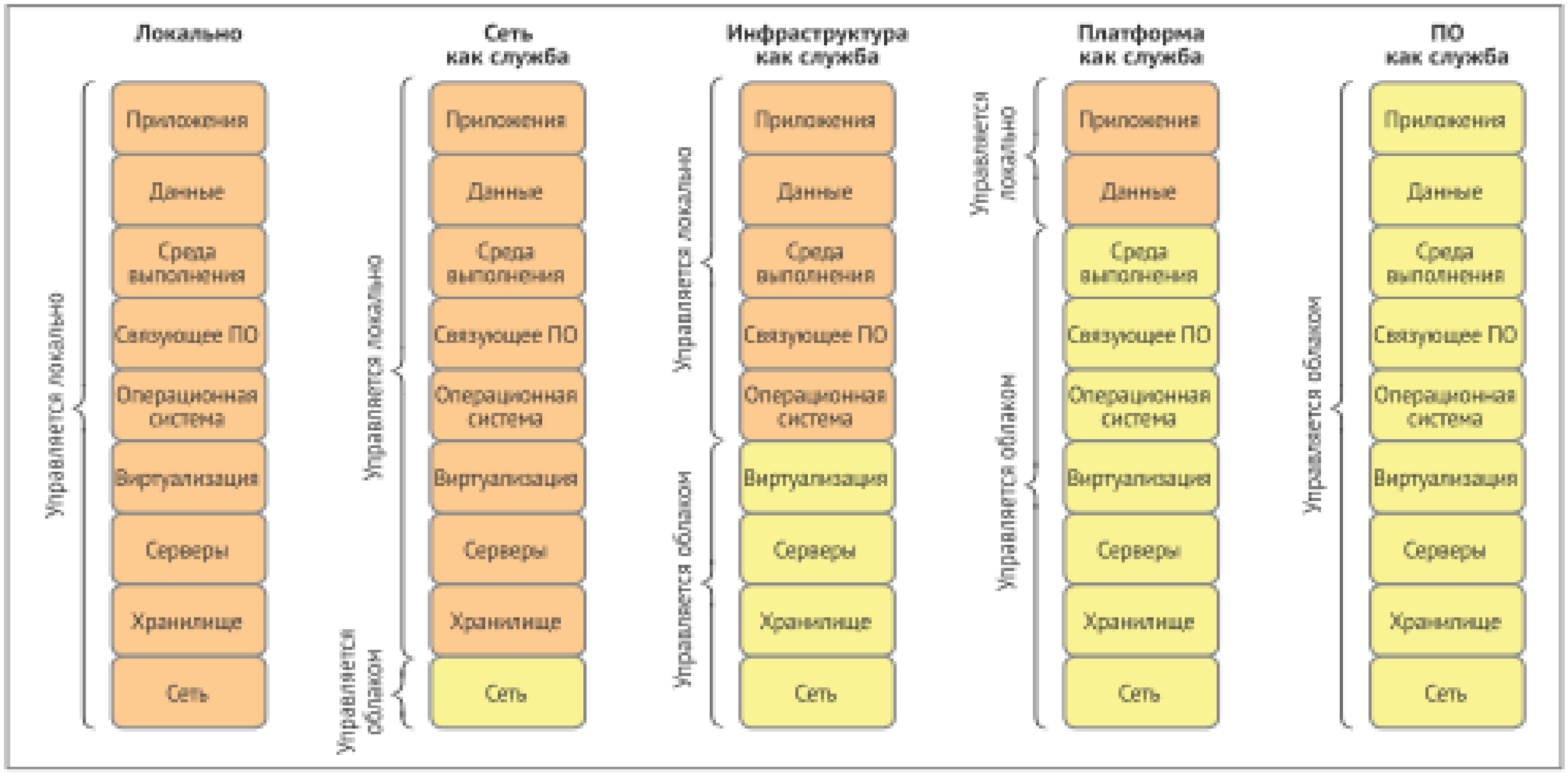
Рассмотрим облачную архитектуру и следующие моменты:

- формальное определение облачных топологий и общеупотребительных терминов;
- обзор архитектуры облака OpenStack;
- изучение фундаментальной проблемы чисто облачной архитектуры;
- обзор туманных вычислений;
- справка по архитектуре OpenFog;
- топологии и примеры использования туманных вычислений.

Модель облачных сервисов

Облачные провайдеры обычно поддерживают целый ряд продуктов «Все как сервис» (XaaS). То есть, услуга программного обеспечения с оплатой за использование. Сервис включает в себя службу сети (NaaS), программное обеспечение как услугу (SaaS), платформу как услугу (PaaS) и инфраструктуру как услугу (IaaS). Каждая модель представляет все больше и больше облачных сервисов от поставщиков. Эти сервисные предложения – добавленная стоимость облачных вычислений. Как минимум, эти услуги должны компенсировать капитальные затраты, с которыми сталкивается клиент для приобретения и обслуживания такого оборудования центра обработки данных, и учесть это как эксплуатационные расходы.

Модели облачной архитектуры



Собственный объект – это то, где управление всеми службами, инфраструктурой и хранилищем осуществляется владельцем

NaaS

Для NaaS характерны такие сервисы, как Сетевое взаимодействие, определенное ПО (SDN) и Программно-определенные периметры (SDP). Эти продукты являются управляемыми облаками и организованными механизмами для обеспечения оверлейных сетей и безопасности предприятий. Вместо того, чтобы создавать глобальную инфраструктуру и выделять капитал для поддержки корпоративных коммуникаций, при создании виртуальной сети может использоваться облачный подход. Это позволяет сети оптимально масштабировать ресурсы в сторону увеличения или уменьшения в зависимости от потребностей, а новые сетевые качества могут быть приобретены и развернуты быстро. Эта тема будет подробно рассмотрена в соответствующей главе SDN.

SaaS

SaaS является основой облачных вычислений. У провайдера обычно есть предлагаемые приложения или услуги, которые предлагаются конечным пользователям с помощью таких клиентов, как мобильные устройства, тонкие клиенты или фреймворки в других облаках. С точки зрения пользователя, виртуальный SaaS фактически работает на клиенте пользователя. Эта абстракция программного обеспечения позволила отрасли добиться значительного роста в облачном сервисе. Сервисы SaaS работают для таких устройств, как Google Apps, Salesforce и Microsoft Office 365.

PaaS

PaaS использует базовое оборудование и программные средства нижнего уровня, предоставляемые облаком. В таком случае конечный пользователь просто использует аппаратное обеспечение центра обработки данных, операционную систему, промежуточное ПО и различные базы данных поставщика для размещения своего частного приложения или сервисов. Промежуточное ПО может состоять из систем баз данных. При построении многих отраслей промышленности было использовано оборудование облачных поставщиков, например для Swedbank, Trek Bicycles и Toshiba. Примерами публичных поставщиков PaaS являются IBM Bluemix, Google App Engine и Microsoft Azure. Разница между PaaS и IaaS заключается в том, что вы получаете преимущества масштабируемости и OPEX с облачной инфраструктурой, но у вас также есть проверенное промежуточное ПО и операционные системы от провайдера. Это такие системы, как Docker, где программное обеспечение развертывается в контейнерах. Если ваше приложение развертывается в пределах ограничений предоставляемой поставщиком инфраструктуры, вы можете ожидать более быстрый выход на рынок, поскольку большинство компонентов, ОС и промежуточного программного обеспечения гарантированно будут доступны.

IaaS

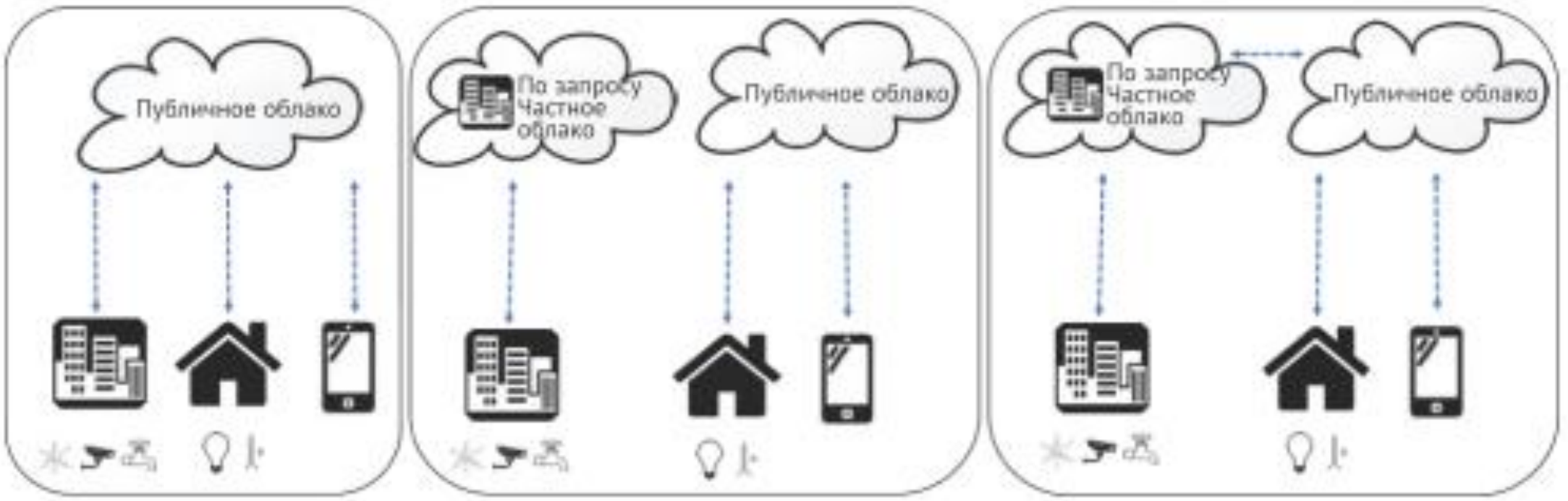
IaaS была изначальной концепцией облачных сервисов. В этой модели поставщик создает масштабируемые аппаратные службы в облаке и предоставляет модификацию программных фреймворков для создания клиентских виртуальных машин. Это обеспечивает максимальную гибкость при развертывании, но требует бóльших усилий со стороны клиента.

Публичное, частное и гибридное облако

В облачной среде существуют три различные модели топологии облаков, которые обычно используются: частное облако, облако общего пользования и гибридное облако. Независимо от модели, фреймворки облаков должны обеспечивать динамическую масштабируемость, быстроту разработки и развертывания, а также появление в локальном месте независимо от его близости.

Частные облака также подразумевают управляемые компоненты по запросу.

Современные корпоративные системы, как правило, используют гибридную архитектуру для обеспечения безопасности критически важных приложений и данных на местности и используют публичное облако для подключения, простоты и быстроты развертывания.



Слева: публичное облако. Посередине: частное против публичного. Справа: гибридное облако

Частное облако

В частном облаке инфраструктура предоставлена одной организации или корпорации. Нет концепции совместного использования ресурсов или объединения за пределами собственной инфраструктуры владельца. В помещениях совместное использование и распоряжение ресурсами являются общими. Частное облако существует по ряду причин, включая безопасность и проверенность качества. То есть, для гарантии, что информация обрабатывается исключительно системами, управляемыми клиентом. Однако, чтобы считаться облаком, должны существовать некоторые аспекты облачных сервисов, такие как виртуализация и балансировка нагрузки. Частное облако может быть локальным или может быть специализированным в оборудовании, предоставляемым третьей стороной исключительно для его использования.

Публичное облако

Публичное облако – противоположная ситуация. Здесь инфраструктура предоставляется по требованию для множества клиентов и приложений. Инфраструктура представляет собой набор ресурсов, которые любой человек может использовать в любое время в рамках своих соглашений об уровне обслуживания. Преимущество здесь в том, что явная шкала облачных центров обработки данных позволяет обеспечить беспрецедентную масштабируемость для многих клиентов, которые ограничены только тем, какую часть услуг они хотят приобрести.

Гибридное облако

Гибридная архитектурная модель представляет собой сочетание частных и общественных облаков. Такими комбинациями могут быть множественные публичные облака, используемые одновременно или комбинация общественной и частной облачной инфраструктуры. Организации предпочитают гибридную модель, если есть данные, которые нуждаются в уникальном подходе, а интероблачная архитектура OpenStack а интерфейс может использовать облако. Другим вариантом использования является поддержание соглашения с облачными областями для компенсации условий, когда масштабируемость лучше, чем у частной корпорации в целом. В этом случае публичное облако будет использоваться как балансировщик нагрузки до тех пор, пока набирание данных и их использование не вернутся в ограниченное пространство частного облака. Этот вариант использования называется облачным взрывом и относится к использованию облаков в качестве условных ресурсов.

Облачная архитектура Open Stack

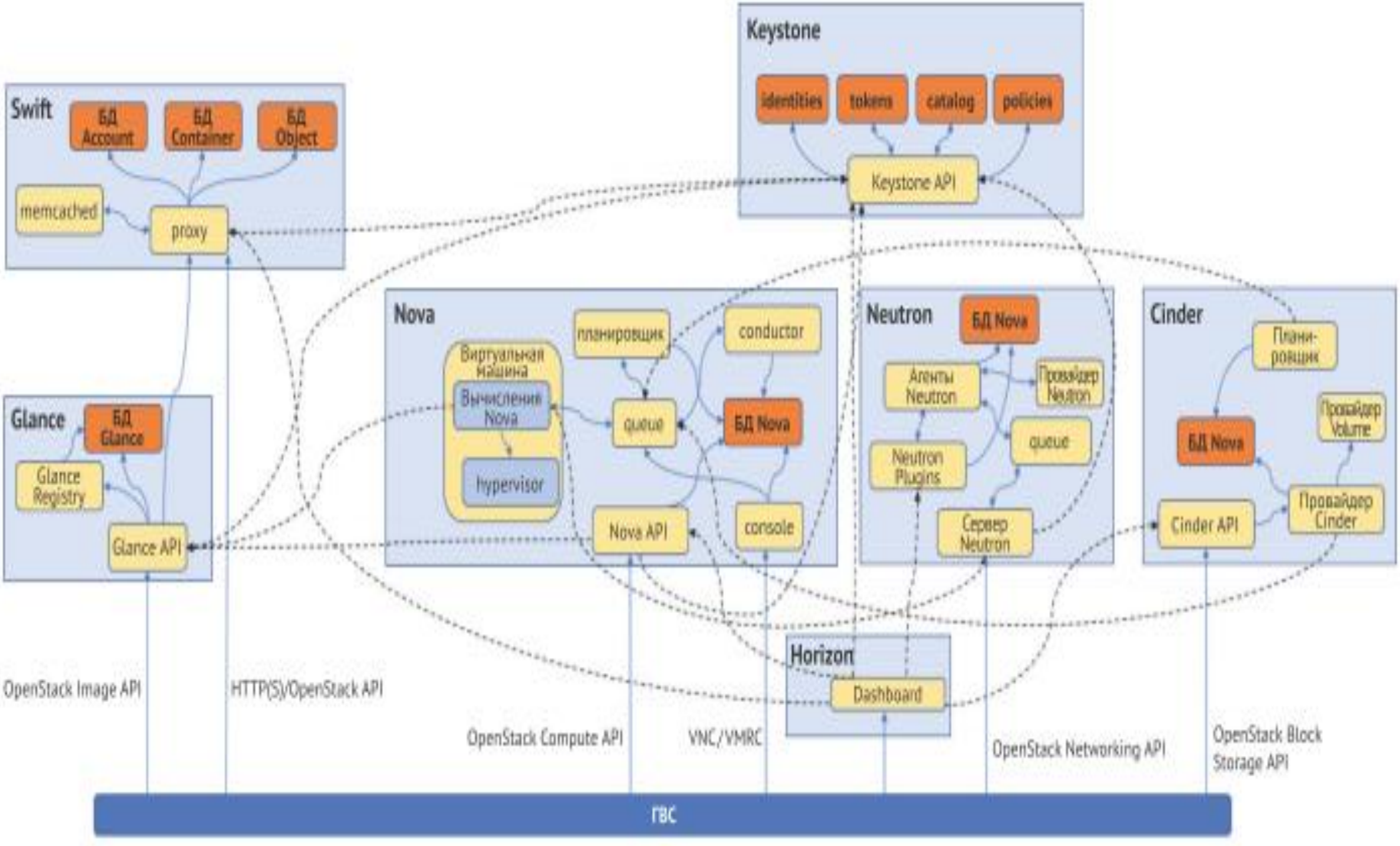
OpenStack – это сервер Apache 2.0 с открытым исходным кодом, используемый для создания облачных платформ. Это IaaS разрабатывается сообществом разработчиков с 2010 г. OpenStack Foundation управляет программным обеспечением и поддерживает более 500 компаний, включая Intel, IBM, Red Hat и Ericsson. Мы будем использовать OpenStack в качестве эталонной архитектуры для других поставщиков облачных вычислений, поскольку большая часть компонентов и терминология также используются в коммерческих облаках.

OpenStack начинался как совместный проект NASA и Rackspace в 2010 г. Архитектура имеет все основные компоненты других облачных систем, включая вычисление и балансировку нагрузки; компоненты хранения, включая резервное копирование и восстановление; сетевые компоненты, информационные панели, системы безопасности и идентификации, пакеты данных и аналитики, инструменты развертывания, мониторы, счетчики и приложения. Это те компоненты, которые будет использовать архитектор при выборе облачного сервиса.

С точки зрения архитектуры, OpenStack представляет собой смешанные слои компонентов. Каждый сервис имеет определенную функцию и уникальное имя (например, Nova). Система работает, в целом, предоставляя масштабируемые функциональные возможности облачного класса масштаба корпорации.

Все коммуникации в компонентах OpenStack выполняются через протокол расширенной очереди сообщений (AMQP), в частности, RabbitMQ или Qpid. Сообщения могут быть либо неблокирующими, либо блокирующими в зависимости от того, как было отправлено сообщение. Сообщение будет отправлено как объект JSON в RabbitMQ, и получатели получат свои сообщения в одном сервисе. Это метод связи (Remote Procedure Call – RPC) между основными подсистемами. Преимущество облачной среды заключается в том, что проблемы клиента и сервера полностью независимы друг от друга, и это позволяет серверам динамически масштабироваться в сторону увеличения или уменьшения. Сообщения не передаются, а направляются, что снижает трафик до минимума. Вы, возможно, помните, что AMQP – это стандартный протокол обмена сообщениями, используемый в пространстве IoT.

Высокоуровневая архитектурная диаграмма OpenStack



Keystone – управление идентификацией и обслуживанием

Keystone – это служба управления идентификаторами облака OpenStack. Менеджер идентификации устанавливает учетные данные пользователя и авторизацию для входа. Это, по сути, отправная точка или точка входа в облако.

Этот ресурс будет поддерживать центральный каталог пользователей и их прав доступа. Это самый высокий уровень безопасности, обеспечивающий независимость и безопасность пользовательских сред. Keystone может взаимодействовать с такими сервисами, как LDAP, на корпоративном уровне. Keystone также поддерживает базу данных токенов и предоставляет временные токены пользователям аналогично тому, как Amazon Web Services (AWS) устанавливает учетные данные. Реестр служб используется для запроса продуктов или услуг, доступных пользователю программно.

Glance – сервис изображений

Glance – это сердцевина управления виртуальными машинами для OpenStack. Большинство облачных сервисов обеспечит некоторую степень виртуализации и будет иметь аналоговый ресурс, подобный Glance. API службы изображений – это служба RESTful, что позволяет клиенту разрабатывать шаблоны VM, обнаруживать доступные виртуальные машины, клонировать изображения на другие серверы, регистрировать виртуальные машины и даже беспрепятственно перемещать работающие виртуальные машины на разные физические серверы без перерыва в работе. Glance вызывает Swift (хранилище объектов) для извлечения или хранения различных изображений. Glance поддерживает разные стили виртуальных образов:

- raw – неструктурированные изображения;
- vhd – VMWare, Xen, OracleVirtualBox;
- vmdk – общий формат диска;
- vdi – изображение эмулятора QEMU;
- iso – изображение на оптическом диске
- aki/ari/ami – изображение Amazon.

Вычисления Nova

Это основа службы управления вычислительными ресурсами OpenStack. Ее цель – определить и учесть вычислительные ресурсы на основе спроса. Она также несет ответственность за управление системным гипервизором и виртуальными машинами. Nova может работать с несколькими виртуальными машинами, например с VMware или Xen, или может управлять контейнерами.

Масштабирование по требованию является неотъемлемой частью любого облачного предложения. Nova основана на API веб-службы RESTful для упрощения управления. Чтобы получить список серверов, можно выполнить следующее в Nova через запрос API GET:{url_вашей_вычислительной_службы}/servers

Чтобы создать банк серверов (в пределах: минимум 10 и максимум 20), мы используем метод POST:

```
{
   "server": {
      "name": "IoT-Server-Array",
      "imageRef": "8a9a114e-71e1-aa7e-4181-92cc41c72721",
      "flavorRef": "1",
      "metadata": {
         "My Server Name": "IoT"
      },
      "return_reservation_id": "True",
      "min_count": "10",
      "max_count": "20"
   }
}
```

Nova в ответ выдаст reservation_id:

```
{
   "reservation_id": "84.urcyplh"
}
```

Таким образом, модель программирования достаточно проста для управления инфраструктурой. База данных Nova необходима для поддержания текущего состояния всех объектов в кластере. Например, несколько состояний различных серверов в кластере могут быть следующими:

- ACTIVE – сервер активно работает;
- BUILD – сервер в состоянии сборки и пока не закончен;
- DELETED – сервер был удален;
- MIGRATING – сервер переносится на другой хост.

Nova полагается на планировщик, чтобы определить, какую задачу выполнить и где ее выполнить. Планировщик может случайно ассоциировать сродство или использовать фильтры, чтобы выбрать набор хостов, которые наилучшим образом соответствуют некоторым наборам параметров. Конечным продуктом фильтра будет упорядоченный список хост-серверов для использования от лучшего к худшему (несовместимые хосты будут удалены из списка).

Ниже приведен фильтр по умолчанию, используемый для выбора родственного сервера:

scheduler_available_filters = nova.scheduler.filters.all_filters

Пользовательский фильтр может быть создан (например, Python или JSON-фильтр с именем IoTFilter.IoTFilter) и прикреплен к планировщику следующим образом:

scheduler_available_filters = IoTFilter.IoTFilter

Чтобы установить фильтр для поиска серверов, имеющих 16 VCPU программным путем через API, мы создаем файл JSON следующим образом:


```
{
   "server": {
      "name": "IoT_16",
      "imageRef": "8a9a114e-71e1-aa7e-4181-92cc41c72721",
      "flavorRef": "1"
   },
   "os:scheduler_hints": {
      "query": "[&gt;=,$vcpus_used,16]"
   }
}
```

Как вариант, OpenStack также позволяет управлять облаком через интерфейс командной строки:

```
$ openstack server create --image 8a9a114e-71e1-aa7e-4181-92cc41c72721 \
  --flavor 1 --hint query='["&gt;=","$vcpus_used",16]' IoT_16
```

OpenStack имеет богатый набор фильтров, позволяющих настраивать распределение серверов и сервисов. Это позволяет очень четко контролировать обеспечение и масштабирование сервера. Это классический и очень важный аспект облачного дизайна. Такие фильтры включают, но не ограничиваются такими характеристиками, как:

- размер оперативной памяти;
- емкость и тип диска;
- уровни IOPS;
- использование процессора;
- групповая родственность;
- родственность с CIDR.

Swift – хранение объектов

Swift предоставляет резервную систему хранения для центра обработки данных OpenStack. Swift позволяет масштабировать кластеры путем добавления новых серверов. Хранилище объектов будет содержать такие вещи, как учетные записи и контейнеры. Виртуальная машина пользователя может храниться или кэшироваться в Swift. Вычислительный узел Nova может вызывать непосредственно Swift и загружать изображение при первом запуске.

Neutron – сетевые сервисы

Neutron – это управление сетью OpenStack и служба VLAN. Вся сеть является настраиваемой и предоставляет такие услуги, как:

- доменные службы имен;
- DHCP – протокол динамической конфигурации хостов;
- функции шлюза;
- управление VLAN;
- соединения на втором уровне модели OSI;
- SDN;
- протоколы с покрытием и туннелированием;
- VPN;
- NAT (SNAT и DNAT);
- системы обнаружения вторжений;
- балансировка нагрузки;
- брандмауэры.

Cinder – блочное хранилище

Cinder обеспечивает OpenStack постоянными службами хранения блоков, необходимыми для облака. Он выступает в роли хранилища как службы для использования с базами данных, динамическими файловыми системами и в других случаях, где важна защита от утечек данных. Это важно и для потоковых сценариев IoT. Как и другие компоненты OpenStack, система хранения сама по себе динамична и масштабируется по мере необходимости. Архитектура построена на принципах высокой доступности и открытых стандартах.

Функциональность, предоставляемая Cinder, включает:

- создание, удаление и привязку устройств хранения к экземплярам Nova;
- совместимость с несколькими хранилищами (HP 3PAR, EMC, IBM, Ceph, CloudByte, Scality);
- поддержку нескольких интерфейсов (Fibre Channel, NFS, Shared SAS, IBM GPFS, iSCSI);
- резервное копирование и извлечение образов дисков;
- сохранение изображений в определенные моменты времени;
- альтернативное хранилище для изображений VM.

Horizon

Последний элемент, рассматриваемый здесь – Horizon. Horizon – это панель инструментов OpenStack. Это упрощенный вид в OpenStack для клиента. Он обеспечивает веб-представление различных компонентов, которые включают OpenStack (Nova, Cinder, Neutron и другие). Horizon представляет собой изображение пользовательского интерфейса облачной системы в качестве альтернативного средства поверх API. Horizon расширяем, поэтому третья сторона может добавлять свои виджеты или инструменты в панель инструментов. Можно добавить новый компонент биллинга, и затем для клиентов может быть создан соответствующий элемент панели Horizon.

Большинство систем IoT, которые используют облачные вычисления, будут иметь некоторую форму панели мониторинга с аналогичными функциями.

Heat – оркестрация (опция)

Heat может запускать несколько составных облачных приложений и управлять облачной инфраструктурой на основе шаблонов в экземпляре OpenStack. Heat интегрируется с телеметрией для автоматической настройки системы в соответствии с нагрузкой. Шаблоны в Heat пытаются соответствовать форматам AWS CloudFormation, а отношения между ресурсами могут быть указаны аналогичным образом (например, данный том подключен к данному серверу). Шаблон Heat может быть похож на следующее:

```
heat_template_version: 2015-04-30
description: example template
resources:
my_instance:
 type: OS::Nova::Server
 properties:
  key_name: { get_param: key_name }
  image: { get_param: image }
  flavor: { get_param: flavor }
  admin_pass: { get_param: admin_pass }
  user_data:
   str_replace:
    template: |
     #!/bin/bash
     echo hello_world
```

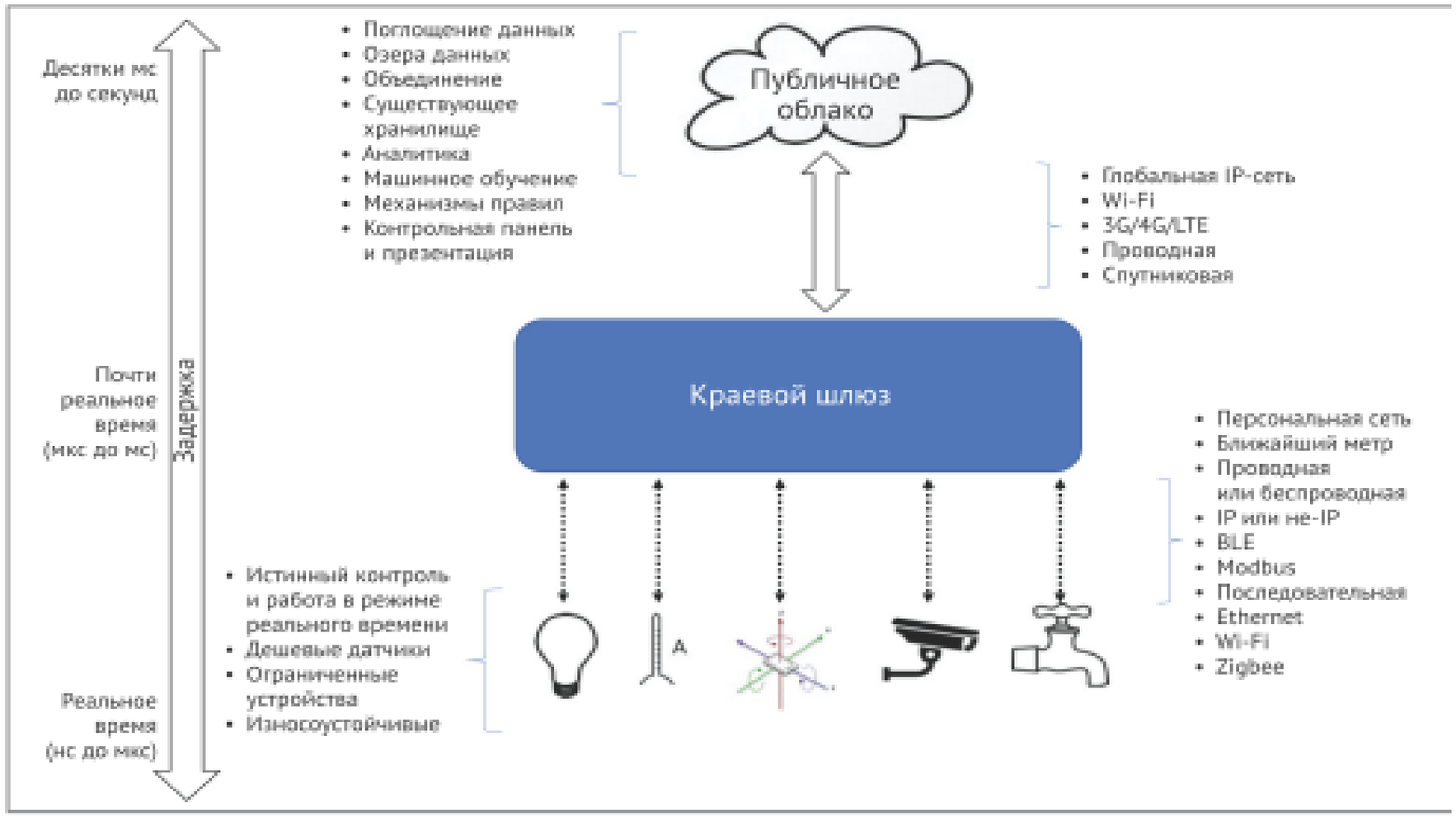
Ceilometer – телеметрия (опция)

OpenStack предоставляет дополнительный сервис под названием Ceilometer, который может использоваться для сбора данных телеметрии и учета ресурсов, используемых каждой службой. Измерение используется для сбора информации об использовании и преобразования его в счета клиента. Ceilometer также предоставляет инструменты оценки и выставления счетов. Значение выставленной стоимости конвертируется в эквивалентную валюту, а биллинг используется для начала процесса оплаты.

Ceilometer контролирует и измеряет различные события, такие как запуск службы, добавление тома и остановка экземпляра. Метрики собираются по использованию ЦПУ, количеству ядер, использованию памяти и перемещению данных. Все это собирается и хранится в базе данных MongoDB.

Ограничения облачных архитектур Для IoT

Поставщик облачных сервисов находится за пределами граничного устройства IoT и руководит глобальной сетью. Одной из особенностей архитектуры IoT является то, что устройства PAN и WPAN могут не соответствовать протоколу IP. Протоколы, такие как Bluetooth Low Energy (BLE) и Zigbee, не основаны на IP, тогда как все в глобальных сетях, включая облака, основано на IP. Таким образом, роль пограничного шлюза заключается в выполнении перевода из одного протокола в другой



Эффекты задержек в облаке. Реакция в реальном времени имеет решающее значение во многих приложениях IoT и вынуждает передвигать обработку ближе к оконечному устройству

Эффект задержки

Другим эффектом является время ожидания и время отклика для событий. По мере приближения к датчику вы входите в область, где действуют жесткие требования выполнения в реальном времени. Эти системы, как правило, представляют собой глубоко встроенные системы или микроконтроллеры с задержкой, предназначенные для реальных событий. Например, видеокамера чувствительна к частоте кадров (как правило, 30 или 60 кадров в секунду) и должна выполнять ряд последовательных задач в конвейере потока данных (избавление от мозаики, обозначение, баланс белого и гамма-регулирование, отображение гаммы, масштабирование и сжатие). Объем данных, проходящих через конвейер видеоизображения (видео 1080p с использованием 8 бит на канал со скоростью 60 кадров в секунду) составляет примерно 1,5 ГБ/с. Каждый кадр должен проходить через этот конвейер в режиме реального времени, поэтому большинство процессоров сигналов видеоизображения используют для этих преобразований контроллеры. Если мы переместимся вверх по стеку, шлюз будет иметь лучшее время отклика и обычно реагирует за миллисекунды с одной цифрой. Коэффициент стробирования во времени отклика – это латентность WPAN и нагрузка на шлюз. Как упоминалось ранее в главе о WPAN, большинство WPAN, таких как BLE, являются переменными и зависят от количества устройств BLE под шлюзом, интервалов сканирования, интервалов рекламы и т. д. Интервалы соединения BLE могут достигать 7,5 мс, но могут варьироваться в зависимости от того, как клиент настраивает интервалы рекламы, чтобы свести к минимуму потребление энергии. Сигналы Wi-Fi обычно имеют задержку 1,5 мс. Задержка такого уровня требует физического интерфейса с PAN. Нельзя ожидать, что при передаче необработанных пакетов BLE в облако, будет быстродействие почти в реальном времени.

Задержка пинга по регионам

Компонент обработки в облаке добавляет еще одну степень задержки в WAN. Маршрут между шлюзом и провайдером облачных вычислений может пролегать несколькими путями на основе расположения центров обработки данных и шлюза. Облачные провайдеры обычно предоставляют набор региональных центров обработки данных для нормализации трафика. Чтобы понять истинное влияние провайдера облачных вычислений на латентность, нужно отследить задержку пинга в течение недель или месяцев и по регионам.

Регион	Задержка
US East (Virginia)	80 мс
US East (Ohio)	87 мс
US West (California)	48 мс
US West (Oregon)	39 мс
Canada (Central)	75 мс
Europe (Ireland)	147 мс
Europe (London)	140 мс
Europe (Frankfurt)	152 мс
Asia Pacific (Mumbai)	307 мс
Asia Pacific (Seoul)	192 мс
Asia Pacific (Singapore)	306 мс
Asia Pacific (Sydney)	205 мс
Asia Pacific (Tokyo)	149 мс
South America (São Paulo)	334 мс
China (Beijing)	436 мс
GovCloud (US)	39 мс

Туманные вычисления

Туманные вычисления – это эволюционное расширение облачных вычислений на краю. В этом разделе описывается разница между вычислениями Fog и Edge и представлены различные топологии и архитектурные ссылки для Fog Computing.

Туманные вычисления развили успех Hadoop и MapReduce, и для того, чтобы лучше понять важность Fog Computing, стоит рассмотреть, как работает Hadoop. MapReduce – это метод сопоставления, а Hadoop – это платформа с от-крытым исходным кодом, основанная на алгоритме MapReduce.

MapReduce включает три шага: отображение, перетасовка и уменьшение. На фазе отображения вычислительные функции работают с локальными данным. Шаг перетасовки перераспределяет данные по мере необходимости. Это критический шаг, когда система пытается совместить все зависимые данные в одном узле. Последним шагом является фаза уменьшения, при которой обработка происходит параллельно на всех узлах.

Общий вывод здесь заключается в том, что MapReduce пытается приблизить обработку туда, где находятся данные, и не перемещать данные туда, где находятся процессоры. Эта схема эффективно устраняет перегрузку коммуникаций и естественное узкое место в системах с чрезвычайно большими структурированными или неструктурированными наборами данных. Эта парадигма также применима и к IoT. В пространстве IoT данные (возможно, очень большой объем данных) создаются в реальном времени в виде потока данных. Это данные большого объема в случае IoT. Это не статические данные, такие как база данных или кластерное хранилище Google, а бесконечный поток данных из всех уголков мира. Туманные конструкции – естественный способ решить эту новую проблему с большими данными.

Сравнение туманных, граничных и облачных вычислений

В случае IoT граничным устройством может быть сам датчик с небольшим микроконтроллером или встроенной системой, способной к WAN-связи. В других случаях граница будет шлюзом в архитектурах с особенно ограниченными оконечными точками, заставляющими шлюз зависать. Граничная обработка также обычно упоминается в контексте «машина-машина», где существует плотная корреляция между краем (клиентом) и сервером, расположенным в другом месте.

Граничные вычисления существуют, как указано, для устранения проблем с задержкой и ненужной загрузкой полосы пропускания, а также для добавления таких сервисов, как денатурация и безопасность, близким к источнику данных. У граничного устройства может быть связь с облачным сервисом ценой задержки и несущей; оно не принимает активного участия в облачной инфраструктуре.

Облачные вычисления немного отличаются от парадигмы граничных вычислений. В облачных вычислениях в первую очередь разделяется API инфраструктуры и стандарты связи с другими туманными узлами и/или покрывающей облачной службой. Туманные узлы являются расширениями облака, граничные устройства могут использовать или не использовать облако.

Другим ключевым принципом туманных вычислений является то, что туман может существовать в иерархических слоях. Туманные вычисления также могут балансировать нагрузку и управлять данными с востока на запад и с севера на юг, чтобы помочь в балансировке ресурсов.

Архитектура OpenFog RA

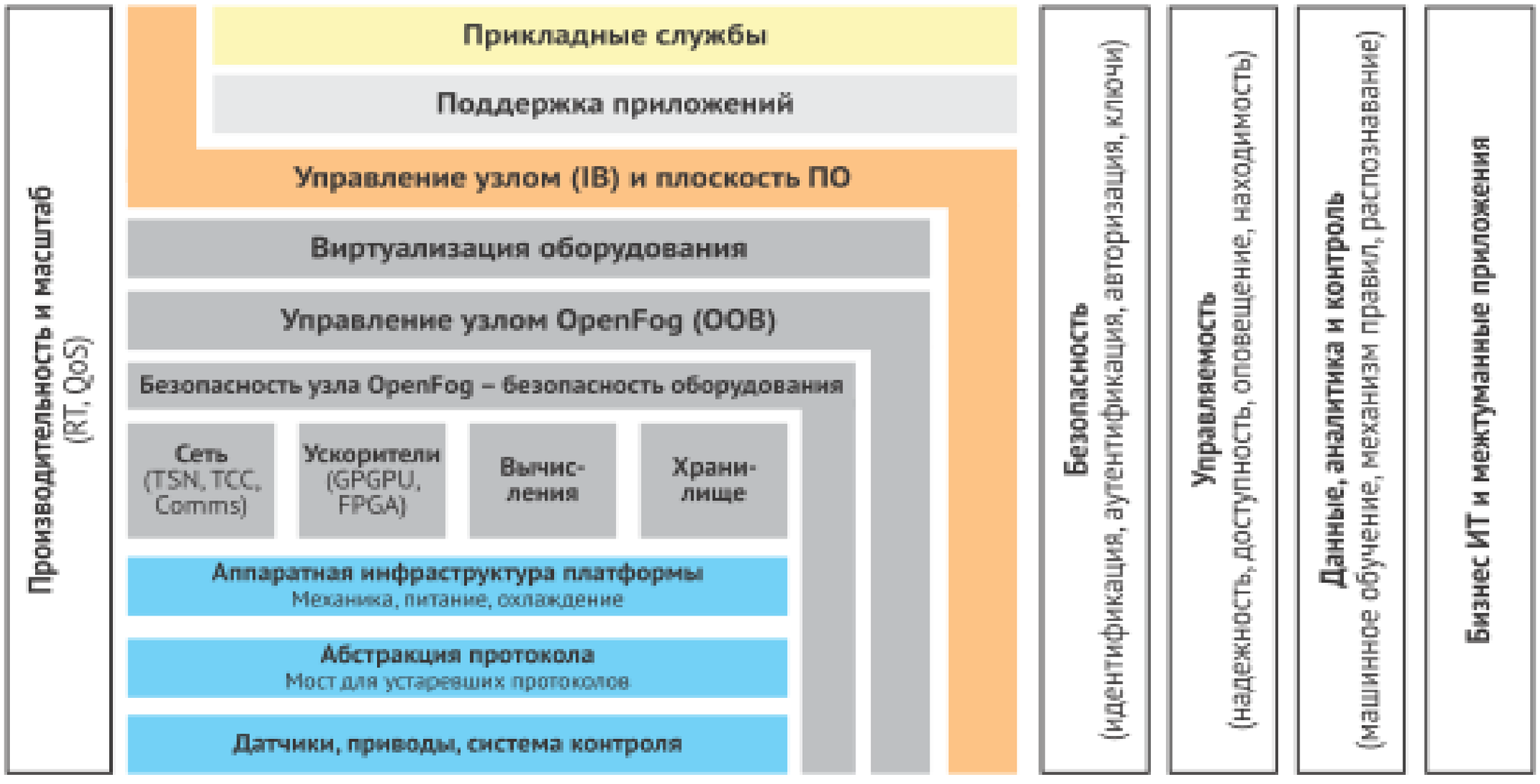
Архитектура туманного фреймворка, такого как облачный фреймворк, необходима для понимания межсетевого взаимодействия и контрактов по данным между различными уровнями. Консорциум OpenFog является некоммерческой отраслевой группой, занятой определением стандартов совместимости с туманными вычислениями. Хотя они не являются органом по стандартизации, но влияют на направление деятельности других организаций посредством взаимодействия и влияния в отрасли. Справочная архитектура OpenFog – это модель, помогающая архитекторам и бизнес-лидерам в создании оборудования, программного обеспечения и приобретении инфраструктуры для туманных вычислений. OpenFog понимает преимущества облачных решений и желание расширить этот уровень вычислений, хранения, сетей и масштабируемости до граничного уровня, не жертвуя задержками и пропускной способностью.

OpenFog Reference Architecture представляет собой многослойный подход от граничных датчиков и исполнительных механизмов внизу, к приложениям наверху. Архитектура имеет некоторое сходство с типичной облачной архитектурой, такой как OpenStack, но расширяет ее, поскольку она более похожа на PaaS, чем на IaaS. В этом случае OpenFog предоставляет полный стек и, как правило, аппаратно независим или, по крайней мере, абстрагирует платформу от остальной части системы.

Службы приложений

Роль сервисного уровня заключается в предоставлении «оконного стекла» и специальных услуг, необходимых для выполнения задачи. Это включает в себя предоставление соединителей для других служб, содержание пакетов для аналитики данных, предоставление пользовательского интерфейса при необходимости и предоставление основных услуг.

Архитектура OpenFog RA



Соединители на прикладном уровне соединяют службы с уровнем поддержки. Уровень абстракции протокола обеспечивает путь для общения соединителя непосредственно с датчиком. Каждая услуга должна рассматриваться как микросервис в контейнере. Консорциум OpenFog выступает за метод развертывания контейнеров в качестве основного метода развертывания программного обеспечения на границе. Это имеет смысл, когда мы рассматриваем граничные устройства как расширения облака.

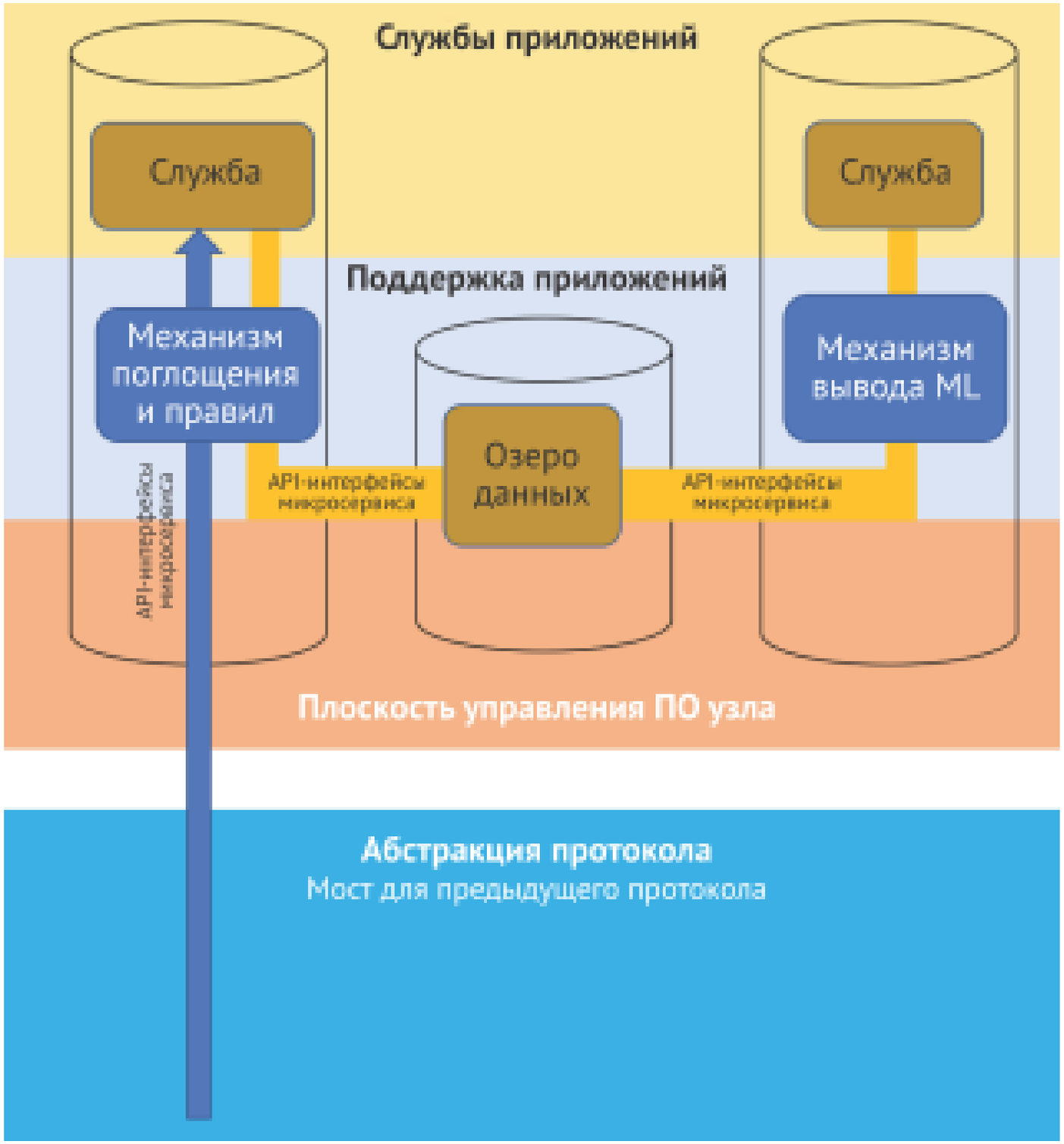
Пример развертывания контейнера может выглядеть следующим образом. Каждый цилиндр представляет собой отдельный контейнер, который можно развернуть и управлять им отдельно. Затем каждый сервис предоставляет API для взаимодействия контейнеров и слоев

Поддержка приложения

Это компонент инфраструктуры, который поможет собрать окончательное решение для клиентов. Этот уровень может иметь зависимости с точки зрения того, как он был развернут (например, как контейнер). Поддержка осуществляется во многих формах, в том числе:

- управление приложением (идентификация образа, проверка образа, установка образа, аутентификация);
- средства журналирования;
- Топология облачных и туманных вычислений
- движки выполнения (контейнеры, виртуальные машины);
- языки выполнения (Node.js, Java, Python);
- серверы приложений (Tomcat);
- шины сообщений (RabbitMQ);
- базы данных и архивы (SQL, NoSQL, Cassandra);
- аналитические фреймворки (Spark);
- службы безопасности;
- веб-серверы (Apache);
- инструменты аналитики (Spark, Drool).

Пример приложения OpenFog



Определение цифрового двойника

Цифровой двойник (англ. Digital Twin) – цифровая модель физического объекта или процесса, подкрепленная данными о его реальном состоянии.

Выгоды

- ✓ Полезные данные для оптимизации процессов
- ✓ Тестирование устойчивости процессов в виртуальной среде
- ✓ Прогнозирование производительности процесса

Объекты в IoT платформе

Объект контроля представляет из себя объект, над которым осуществляется контроль и управление посредством использования функционала платформы.

Подразумевается, что объект является неким виртуальным аналогом подключенного устройства, так как для каждого объекта отображаются текущие показания и параметры, считываемые с помощью датчиков подключенного устройства.

Объекты в IoT платформе

Любые функции объекта контроля можно разделить на два базовых класса:

- сбор информации (сенсорная сеть);
- управление устройствами (контрольная сеть).

В платформе объекты контроля имеют виртуальное представление в виде моделей.

Модели используются для нормализации данных, построения сценариев автоматизации, при обработке данных телеметрии об объекте и др. К каждому объекту обязательно должна быть подключена модель.

Модель объекта контроля

Модель объекта контроля - это формализованное представление логических и аппаратных функций устройств, управляющих объектом контроля.

Модель обладает древовидной структурой, состоящей из подсистем, аргументов, событий и действий.

Параметры модели объекта контроля

Аргументы - это параметры, которые передает объект контроля в систему от датчиков.

Аргументы могут обладать числовым, строковым, логическим типом данных, представлять собой объект или массив. Причем для числовых аргументов можно указать единицу измерения;

Параметры модели объекта контроля

События - это совершенные действия, которые либо были самостоятельно зафиксированы самим объектом контроля и в соответствии с заданной логикой поведения отправлены в систему, либо произошли во внешних по отношению к объекту контроля приложениях, тоже участвующих в бизнес-процессе;

Параметры модели объекта контроля

действия - это те операции, которые выполняет устройство, они разделяются на несколько видов:

- **команда к устройству** - это команда, на которую оно способно реагировать, и которая им выполняется;
- **автомат** - это сценарий автоматизации (конечный автомат), позволяющий выстраивать логику поведения устройства;
- **действие (severless)** - это действие, которое не заложено в список команд во внешние программные модули и которое пользователь может создать сам с целью его применения в своем конечном автомате при построении бизнес-логики;

Параметры модели объекта контроля

действия - это те операции, которые выполняет устройство, они разделяются на несколько видов:

- **команда к устройству** - это команда, на которую оно способно реагировать, и которая им выполняется;
- **автомат** - это сценарий автоматизации (конечный автомат), позволяющий выстраивать логику поведения устройства;
- **действие (severless)** - это действие, которое не заложено в список команд во внешние программные модули и которое пользователь может создать сам с целью его применения в своем конечном автомате при построении бизнес-логики;