Особенности и основные аспекты проектирования «облачных» архитектур

Управление экземплярами. Хранение данных. Облачные базы данных — это базы данных, которые запускаются на платформах облачных вычислений, таких как Amazon C2, GoGrid и Rackspace. Существуют две распространенные модели развертывания: пользователи могут приобрести непосредственно услугу доступа к базам данных, обслуживаемым поставщиком облачного сервиса, или же запустить базы данных в облаке независимо, используя образ виртуальной машины. Среди облачных баз данных присутствуют как SQL-ориентированные, так использующие модель данных NoSQL.

Модели развертывания

Существует два основных метода запуска базы данных в облаке:

- **Образ виртуальной машины** — облачные платформы позволяют приобретать виртуальные машины, где возможно запускать базы данных. Пользователи могут загружать свои образы с уже установленной базой или же воспользоваться готовыми, где установлен уже оптимизированный экземпляр. Например, Oracle предлагает готовый образ для виртуальной машины с Oracle Database 11g Enterprise Edition на Amazon EC2.
- **База данных как сервис** — некоторые облачные платформы предлагают сервис баз данных, при помощи которого можно обойтись без виртуальной машины. В данном случае, пользователю не нужно устанавливать и поддерживать базу данных самостоятельно. Вместо этого, поставщик сервиса берет на себя ответственность в установке и обслуживании базы данных. Например, Amazon Web Services предоставляет три базы данных, входящие в их облачный сервис: SimpleDB (NoSQL, где данные хранятся в парах ключ-значение), Amazon Relational Database Service (SQL-ориентированная база данных с MySQL интерфейсом) и DynamoDB.

Так же можно приобрести хостинг базы данных, в случае если база данных не предоставляется как сервис. Например, облачный провайдер Rackspace предлагает такую услугу для баз данных MySQL.

Архитектура и общие характеристики

Многие провайдеры к базам данных предоставляют веб-интерфейс, при помощи которого пользователи могут устанавливать и настраивать экземпляры баз данных. Например, веб-консоль Amazon Web Services позволяет запускать экземпляры баз данных, создавать снапшот (то же, что и резервное копирование) и следить за статистикой.

Предлагается компонент управления базами данных, который контролирует основную базу данных, используя специальное API сервиса. API открыто для пользователя и позволяет ему выполнять обслуживание и масштабирование своих экземпляров баз данных. Например, API для Amazon Relational Database Service позволяет создавать сам экземпляр базы данных, модифицировать его содержимое, а так же создавать снапшоты или восстанавливать данные из ранее созданных снапшотов.

Подобный сервис делает прозрачным для пользователя весь стек программного обеспечения, который используется для поддержания работоспособности базы. Обычно он включает в себя операционную систему, саму систему управления базами данных и стороннее программное обеспечение, используемое в работы. Поставщик услуг берет на себя ответственность за установку, исправление и управление данным программным обеспечением.

Данный сервис берет на себя масштабируемость и доступность базы данных. Причем особенности масштабируемости различаются у разных поставщиков — кто-то это делает автоматически, а другие позволяют пользователю производить расширение при помощи API. Также провайдеры обычно гарантируют высокую доступность сервиса (около 99,9 % или 99,99 %).

Модели данных

Также важно различать реляционные и не реляционные, NoSQL, базы данных:

- **SQL базы данных** — это такие базы, как NuoDB, Oracle Database, Microsoft SQL Server и MySQL. Любую из них можно запускать в облаке, причем только от поставщика зависит, будет ли это образ виртуальной машины или сервис. SQL базы данных трудно масштабировать, потому что изначально они не были рассчитаны на облачную среду.
- **NoSQL базы данных** — это такие базы, как Apache Cassandra, CouchDB и MongoDB. NoSQL базы данных были созданы, чтобы выдерживать большую нагрузку на чтение/запись данных, а так же легко расширяться и уменьшаться, к тому же они изначально создавались под облачные платформы. Однако, большинство современных программ были созданы с использованием SQL, поэтому работа с NoSQL базами данных часто требует полностью переписывать код приложения.

Поставщики облачных баз данных по модели развертывания и модели данных		
	Развертывание виртуальной машины	База данных как сервис
SQL	· Oracle Database · IBM DB2 · Ingres (database) · PostgreSQL · MySQL · ScaleDB · NuoDB · GaianDB	· Amazon Relational Database Service (MySQL) · Microsoft SQL Azure (MS SQL) · GenieDB · Heroku PostgreSQL as a Service (распределенная и выделенная база данных) · Clustrix Database as a Service · Xeround Cloud Database — MySQL front-end · EnterpriseDB Postgres Plus Cloud Database · GaianDB · ClearDB ACID-compliant MySQL
NoSQL	· CouchDB на Amazon EC2 · Hadoop на Amazon EC2 · Apache Cassandra на Amazon EC2 · Neo4J на Amazon EC2 или Microsoft Azure · MongoDB на Amazon EC2 или Microsoft Azure	· Amazon DynamoDB · Amazon SimpleDB · Cloudant Data Layer (CouchDB) · Database.com by SalesForce · Google App Engine Datastore · MongoDB Database as a Service · Cloudbase.io Cloud Database

Особенности проектирования облачных решений

Необходимо учитывать следующее (на примере платформы *Windows Azure*):

- Любое приложение, являющееся частью "облака" - удаленный сервис. На этапе планирования решения необходимо учитывать временные задержки, между отправкой запроса и получением результата, а также необходимость контроля статуса соединения и его возобновления.
- Динамичная инфраструктура облака обеспечивает горизонтальную масштабируемость, таким образом, количество единовременно работающих экземпляров приложения может меняться. В связи с этим состояние приложения необходимо хранить в долговременном хранилище (не на локальных дисках).

- Ряд сценариев, стоимость которых велика , в случае их реализации на основе локальной инфраструктуры (к примеру обработка больших объемов данных), могут быть реализованы в "облачном" решении в виде платформенных сервисов.
- Управление выделением вычислительных ресурсов для приложения, их сопровождение осуществляется платформой.
- Стоимость облачного решения. Данное понятие стало актуальным при появлении "облачных решений".

Стоимость "облачного" решения

В первую *очередь*, на *стоимость* решения будет влиять *трафик* - объем переданных, или полученных данных. Для уменьшения *стоимость* данной составляющей целесообразно использовать сжатие данных, переход к *SOA* - архитектуре и *размещение* кода, активно работающего с данными рядом с их хранилищем (т.е. вычисления не будут выходить за рамки "облака" и влиять на объем передаваемой информации).

Стоимость второй компоненты - *вычислительных ресурсов* - начинает измеряться с момента начала развертывания приложения и зависит от мощности предоставленных виртуальных машин.

.

Третьей компонентой является *хранилище данных*. *Стоимость* хранения и обработки данных определяется их объемом и количеством операций с ними.

Таким образом, для уменьшения *стоимость* дан-ной компоненты целесообразно подбирать хранилище оптимального размера и минимизировать количество операций с данными.

Мультитенантная архитектура

Способ снижения стоимости вычислительных ресурсов и хранилища данных является так называемая **мультитенантная архитектура приложения**.

Данная *архитектура* применяется, как правило, для приложений, обслуживающих группы изолированных друг от друга пользователей.

В отличие от общего случая, когда каждая *группа пользователей* использует свой экземпляр приложения и свою изолированную *БД*, состоит в том, что мультитенантное *приложение* обладает встроенными возможностями для обслуживания нескольких групп пользователей.

Таким образом, для каждой группы пользователей, в случае размещения приложения в облаке, выделяется только необходимое, не фиксированное количество ресурсов, т.е. "*стоимость*" начисляется по - факту использования "облачных" ресурсов.

Этапы перехода к мультитенантной архитектуре:

Выделенная архитектура. Каждая группа пользователей использует свою копию приложения (сервиса). Каждая копия ориентирована только на свою группу пользователей, таким образом, каждая копия содержит персональное расширение, в котором хранятся пользовательские настройки. Вся неэффективность данной модели станет наглядна при увеличении числа пользователей. Также становится невозможным повышения эффективности за счет роста масштабов, поскольку фактически пользователи пользуются разными экземплярами приложения.

Настраиваемая архитектура. В данном случае приложение, либо сервис настраивается для каждого конкретного пользователя через конфигурацию. Т.е. каждый пользователь работает со своей копией приложения, не смотря на то, что копии идентичны. Вычислительные мощности не разделяются между различными экземплярами приложения, что также не позволяет добиться повышения эффективности за счет увеличения масштабов. Разделения же самого приложения может быть как виртуальным, так и физическим.

Мультитенантная архитектура. Настройка для каждого пользователя выполняется исключительно через конфигурацию, при использовании инструментов для самостоятельной настройки. При этом отсутствует возможность горизонтального масштабирования, т.е. повышение производительности может быть достигнуто только через вертикальное масштабирование.

Масштабируемая архитектура. Данная модель поддерживает мультитенатный подход, а также возможность горизонтального масштабирования. При этом новые экземпляры приложений добавляются в общий пул. Общая нагрузка распределяется по всей инфраструктуре. Архитектура настраивается через конфигурацию.

Модели организации мультитенантного хранилища данных

Отдельные базы данных. Каждая группа пользователей имеет собственную базу данных, с собственной схемой данных. Может быть эффективной при небольшом числе пользователей в расчете на базу данных. Кроме того, данный вариант предпочтителен, в случае предъявляемых пользователями строгих требований к изоляции и безопасности данных.

Совместно используемые базы данных с разными схемами. Все пользователи работают с одной БД, но имеют различные наборы предопределенных полей. Наиболее полезен данный вариант в случае, если хранение данных разных пользователей в одной таблице допустимо, а также заранее известно какие поля потребуются. Может привести к ситуации разряженного наполнения таблиц.

Совместно используемые базы данных с совместной схемой. Для хранения расширений данных используются специальные подходы и техники, пользователи работают с одной и той же базой данных. При этом, можно предложить пользователям практически неограниченное количество полей, однако, возникают проблемы при организации индексации и поиска информации. Данный вариант применим, если хранение данных различных групп пользователей в одних и тех же таблицах удовлетворяет требованиям безопасности и изоляции данных.

Подход **SOA (Service-oriented architecture)**.

SOA, или *сервисно - ориентированная архитектура* - подход, суть которого заключается в разработке программного обеспечения, основанной на использовании сервисов со стандартизированными интерфейсами.

Главными целями, к достижению которых стремится *SOA*, являются:

- повышение масштабируемости создаваемых систем;
- упрощение процессов контроля и управления созданными системами;
- сокращение издержек, связанных с разработкой приложений;
- увеличение доли повторного использования кода;
- независимость систем от платформ, инструментов разработки и языков программирования.

В основе данного подхода лежат:

- многократное использование *функциональных элементов*;
- устранение дублирования функционала;
- стандартизация и унификация *типовых процессов*;
- переход компаний на *функциональную организацию*.

Одним из основных преимуществ *SOA* является надежность, особенно при реализации, основанной на применении веб - технологий.

В помощь проектировщикам решений, приведены некоторые соображения касательно того, в каком случае "облака" будут наиболее выгодными.

Эволюция развития сферы IT - услуг, в частности, концепция "облачных" вычислений позволяют предположить, что рано или поздно, но все компании будут либо формировать свои "облака", либо арендовать облачные платформы. При этом, очевидно, что построением собственных "облаков" займутся крупные компании, особенно те из них, кто уже вложился в формирование собственных центров обработки данных.

Согласно статистике наиболее востребованы следующие *SaaS* - приложения:

- почта и коммуникации;
- антивирусные системы;
- службы технической поддержки;
- проектный менеджмент;
- дистанционное обучение.

Когда же следует задуматься об использовании "облака"? Самый очевидный ответ - в тот момент, когда нехватка IT - ресурсов станет очевидной и острой. Но, строго говоря, если такая ситуация возникает, то это говорит о том, что задуматься над реорганизацией IT - инфраструктуры следовало много раньше.

Отметим, некоторые косвенные признаки, *по* наличию которых можно судить о назревающей необходимости в "облачном" обеспечении:

- Необходимость быстро разворачивать новые версии сред разработки и тестирования.
- Необходимость отсутствия ограничений на число экземпляров различных вариантов пользовательских сред.
- Необходимость оперативного учета количества и стоимости потребляемых IT - ресурсов.

Существует два способа использования "облачных" технологий:

1. *Размещение* приложения в "облаке"

Приложение и данные полностью находятся в рамках «облака», т.е. клиенту предоставляется полностью готовый к использованию сервис. При этом сам код и данные приложения развертываются либо как *PaaS*, либо как IaaS. В первом случае, код и данные помещаются в пакет, публикуемый в облаке, во втором - *контроль* и управление приложением осуществляется на уровне виртуальной машины.

2. Использование сервисов "облака"

Данные сервисы доступны посредством стандартных протоколов передачи данных в сети *Интернет* и, благодаря этому, не зависят от платформы или устройств. При этом, можно как использовать встроенные сервисы облачных платформ, так и размещать в облаке собственноручно созданные.

При этом, могут быть использованы следующие варианты использования сервисов, при которых часть кода остается в рамках локальной инфраструктуры:

- перенос данных в "облако", их часть, или все данные остаются в локальной инфраструктуре;
- перенос кода приложения в "облако", все данные, или их часть остается в локальной инфраструктуре;
- интеграция - код и данные, размещенные в "облаке" взаимодействуют с приложениями локальной инфраструктуры.

Стратегия развертывания облака

Условно представить стратегию перехода на "облачную" основу, как последовательность этапов:

1. Пробный этап. Переход на облако, это прежде всего смена стратегии, переосмысление роли информационных технологий в организации, в целом. Имеет смысл, предоставить своим сотрудникам возможность работы с публичными "облаками", следует перевести ряд задач на "облачную" основу. Данный этап поможет оценить степень готовности организации в целом к реорганизации IT - инфраструктуры, а также привить основные навыки работы с "облаками".

2. Формирование требований. После того, как в организации сформируются определенные правила и нормы, по началу скорее неформальные, использования облачных платформ, необходимо задуматься о формировании общеорганизационных требований к предоставляемым внешним поставщиком услугам. К примеру, необходимо определиться с тем, какие именно сервисы и платформы нужны для работы, сколько необходимо систем хранения данных. На данном этапе следует определиться с поставщиками "облачных" услуг, сформировать требования к архитектуре и функционалу используемых решений. Результатом данного этапа должен быть набор формализованных требований к "облакам" организации.

3. Построение частного облака. Данный этап актуален для крупных компаний. Началом данного этапа можно считать момент, когда большая часть сотрудников активно пользуется "облачными" сервисами в своей работе. При наличии экономической целесообразности, можно начать формирование своего корпоративного вычислительного облака, на базе имеющегося центра обработки данных. В конечном счете, эволюционным развитием подобного центра и является корпоративное "облако". При этом, формирование своего центра облачных вычисление не обязательно означает отказ от услуг внешних поставщиков подобных решений. Вполне может использоваться *гибридная схема* работы, когда для

Введение в технологии Azure

Платформа *Windows Azure* - это "облачная" платформа компании Microsoft, реализующая модель *PaaS*. Инструменты данной платформы предоставляют функционал для создания решения, включающее в себя "облачную" операционную систему и набор сервисов для разработчиков.

Платформа *Windows Azure* является частью "облака" компании Microsoft, которое состоит из следующих категорий сервисов:

- "Облачные" приложения (*cloud-based* applications) - представляют собой набор постоянно доступных, масштабируемых сервисов, размещенных в "облаке" Microsoft, которые потребители могут использовать напрямую. К примеру, к таковым относятся: *Bing*,*Windows Live Hotmail*, *Office Live* и т.д.

- Программные сервисы (software services) - представляют собой набор *SaaS* - сервисов, таких как *Exchange Online*, *SharePoint Online*,*Office Communications Online* и т.д.
- Платформенные сервисы (*platform services*) - используются, как платформа, представляющая собой публичное "облако", которую разработчики могут использовать для внедрения решений нового поколения. К данным сервисам, в частности относятся *SQL Azure*,*AppFabric* и *Windows Azure*.
- Инфраструктурные сервисы (infrastructure services) - набор компонент платформы *Windows Azure*, обеспечивающих поддержку "облачных" инфраструктурных ресурсов.

Платформа *Windows Azure* включает в себя:

- *Windows Azure* - операционная система в "облаке", предоставляет вычислительные ресурсы, средства хранения данных и инструменты управления сервисами.
- *SQL Azure* - реляционная база данных, предоставляет основные возможности *MS SQL Server* по хранению данных, предоставляется как сервис.
- *Windows Azure AppFabric* - программные модули, обеспечивающие коммуникации (*Service Bus*) и контроль доступа (*Access Control*). Используются для обеспечения взаимодействий между приложениями потребителя и приложениями облака.

По своей сути, *Windows Azure* представляет собой платформу для запуска *windows* - приложений и хранения данных в облаке.

Возможность использования различных экземпляров *Windows Azure* позволяет пользователям запускать разнообразные сервисы. При этом эластичность, *безопасность*, доступность и *масштабируемость* обеспечиваются самой платформой, предоставленной в виде сервиса.

Хранение данных в *Windows Azure* возможно с помощью ряда сервисов, в виде таблиц, бинарных данных и очередей.

Windows Azure работает на базе машин, расположенных в дата - центрах компании Microsoft, *доступ* к платформе обеспечивается посредством Интернета.

Платформа *Windows Azure* создана на основе технологий виртуализации. В отличие от обычного предоставления мощностей виртуальных машин для удаленных пользователей, платформа azure управляется с помощью специального инфраструктурного слоя *Windows Azure Fabric Controller*, или "фабрики".

Задача фабрики заключается в организации всего вычислительного массива виртуализированных экземпляров *Windows Azure* в виде логически единой единицы вычислений, а также *управление ресурсами*, нагрузкой и всем жизненным циклом приложений и сервисов в целом. Иными словами, фабрика связывает вычислительные мощности различных машин в единое целое.

Сервисы выполнения *Windows Azure*, естественно, основаны на *Windows* - технологиях. Создавать и запускать на *Windows* Azure можно только приложения основанные на *.Net Framework*, к примеру, это могут быть *ASP.Net* или WCF - приложения. Не стоит воспринимать *Windows Azure* только как платформу, для веб - приложений, платформа также поддерживает *фоновые процессы*, не зависящие от веб - приложений.

Также стоит отметить, что сервисы хранения данных, предоставляемые *Windows Azure* могут использоваться, как локальными приложениями пользователя, так и приложениями, размещенными непосредственно на *Azure* - платформе. Клиент может контролировать свое azure - *приложение*, редактируя вручную, или программно, его конфигурацион-ный *файл*, имеющийся у каждого приложения.

Доступ к возможностям конфигурации и мониторинга состояния приложения осуществляется через соответствующий портал, при предоставлении клиентского *Windows Live ID*.

Примеры применения Windows Azure

Подведем итог, для чего именно может быть использована *Windows Azure*:

- Размещение в "облаке" веб - приложения. При этом инфраструктура "облака" направляет запрос клиента к одному из экземпляров веб - роли виртуальной машины. Виртуальный веб - сервер данной роли, получив запрос активирует соответствующий код.
- *Параллельная обработка данных*. В данном случае, при обработке больших массивов данных, клиент, к примеру, с помощью WPF-приложения обращается к веб-сервису, запрос на обработку данных помещается в очередь *Azure Queue*. Обработка осуществляется в асинхронном режиме. Результат помещается в *Azure Table*.
- Объединение локальных вычислительных мощностей и ресурсов облака.

При использовании *Windows Azure* пользователи получают возможность запускать различные сервисы, при этом платформа обеспечивает *масштабируемость*, *безопасность* и доступность.

Windows Azure:

- добавляет возможности веб - служб существующим пакетным приложениям;
- позволяет создавать, изменять и распространять приложения через веб при наличии минимальной IT - инфраструктуры;
- предоставляет сервисы хранения большого количества данных, пакетной обработки и высокопроизводительных вычислений;
- обеспечивает возможности оперативного тестирования и распространения веб - служб при минимальных затратах;
- уменьшает издержки, связанные с содержанием IT - инфраструктуры;
- упрощает процесс управления IT - инфраструктурой.

Windows Azure:

- предоставляет "облачную" среду для разработки, хостинга и управления сервисами;
- представляет собой контейнер, содержащий код и логику, в рамках которого приложение может быть развернуто;
- предоставляет среду, схожую с существующей Windows Server средой;
- позволяет разворачивать *.Net* проекты напрямую, без внесения существенных и трудоемких изменений;
- позволяет хранить данные пользователей, поддерживает тройную репликацию.