12. Облачные вычисления

Облачные вычисления привлекают много внимания в последнее время. В средствах массовой информации, в интернете, даже на телевидении можно встретить восторженные материалы, описывающие все прекрасные возможности, которые может предоставить данная технология.

Не смотря на то, что метафора «облако» уже давно используется специалистами в области сетевых технологий для изображения на сетевых диаграммах сложной вычислительной инфраструктуры (или же Интернета как такового), скрывающей свою внутреннюю организацию за определенным интерфейсом, термин «Облачные вычисления» появился на свет совсем недавно. Согласно результатам анализа поисковой системы Google, термин «Облачные вычисления» («Cloud Computing») начал набирать вес в конце 2007 – начале 2008 года, постепенно вытесняя популярное в то время словосочетание «Грид-вычисления» («Grid Computing»). Судя по заголовкам новостей того времени, одной из первых компаний, давших миру данный термин, стала компания IBM, развернувшая в начале 2008 года проект «Blue Cloud» и спонсировавшая Европейский проект «Joint Research Initiative for Cloud Computing».

На сегодняшний день уже можно говорить о том, что облачные вычисления прочно вошли в повседневную жизнь каждого пользователя Интернета (хотя многие об этом и не подозревают). По некоторым экспертным оценкам, технология облачных вычислений может в три-пять раз сократить стоимость бизнес-приложений и более чем в пять раз стоимость приложений для конечных потребителей, но, несмотря на общее радужное чувство по отношению к облачным технологиям, до сих пор нет единого мнения о том, что такое «Облачные Вычисления» и каким образом они соотносятся с парадигмой «Грид-вычислений». Для того, чтобы разобраться в этом, взглянем сначала на несколько существующих определений облачных вычислений, выясним основные характеристики облаков и рассмотрим, какие общие аспекты можно выявить в архитектуре облачных решений. Рассмотрим достоинства и недостатки, а также попробуем классифицировать платформы облачных вычислений. В конце главы мы попытаемся сделать сравнение двух концепций: грид-вычислений и облачных вычислений.

12.1 Определение облачных вычислений

С одной стороны, у термина «Облачные вычисления» нет устоявшегося стандартного определения. С другой стороны множество различных корпораций, ученых и аналитиков дают собственные определения этому термину. Определение облачных вычислений вызвало дебаты и в научном сообществе. В отличие от определений, которые можно найти в коммерческих изданиях, научные определения ориентируются не только на то, что будет предоставлено пользователю, но и на архитектурные особенности предлагаемой технологии. Например, в лаборатории Беркли дают следующее определение облачных вычислений:

«Облачные вычисления – это не только приложения, поставляемые в качестве услуг через Интернет, но и аппаратные средства и программные системы в центрах обработки данных, которые обеспечивают предоставление этих услуг. Услуги сами по себе уже давно называют «предоставление программного обеспечения как услуги» (Software-as-a-Service или SaaS). Облаком называется аппаратное и программное обеспечение центра обработки данных. Общественное облако предоставляет ресурсы облака широкому кругу пользователей по принципу «*оплата по мере использования*» (*pay-as-you-go* – принцип предоставления услуг, при котором пользователь оплачивает только те ресурсы, которые были по факту затрачены на решение поставленной задачи). Частное облако – это внутренние центры обработки данных, в коммерческой или иной организации, которые не доступны широкому кругу пользователей. Таким образом, облачные вычисления являются суммой SaaS и «*коммунальных вычислений*» (*Utility Computing* – модель вычислительных систем, в которой предоставление данных и процессорных мощностей организовано по принципам коммунальных услуг). Люди могут быть пользователями или провайдерами SaaS, либо пользователями или поставщиками коммунальных вычислений».

Данное сложное и пространное определение выделяет другую сторону облачных вычислений: с точки зрения провайдера, важнейшей составляющей облака является центр обработки данных (ЦОД). ЦОД содержит вычислительные ресурсы и хранилища информации, которые вместе с программным обеспечением предоставляются пользователю по принципу «оплата по мере использования».

Ян Фостер определяет облачные вычисления как «парадигму крупномасштабных распределенных вычислений, основанную на эффекте масштаба, в рамках которой пул абстрактных, виртуализованных, динамически-

масштабируемых вычислительных ресурсов, ресурсов хранения, платформ и сервисов предоставляется по запросу внешним пользователям через Интернет».

Данное определение добавляет два важнейших аспекта в определение облачных вычислений: *виртуализацию* и *масштабируемость*. Облачные вычисления абстрагируются от базовой аппаратной и программной инфраструктуры посредством виртуализации. Виртуализованные ресурсы предоставляются посредством определенных абстрактных интерфейсов (программных интерфейсов API или сервисов). Такая архитектура обеспечивает масштабируемость и гибкость физического уровня облака без последствий для интерфейса конечного пользователя.

Наконец в работе [70], после обработки более 20 определений облачных вычислений, было дано следующее определение облачных вычислений (возможно, чуть более целостное, чем остальные определения приведенные в данной главе):

«*Облако* – это большой пул легко используемых и легкодоступных виртуализованных ресурсов (таких как аппаратные комплексы, сервисы и др.). Эти ресурсы могут быть динамически перераспределены (масштабированы) для подстройки под динамически изменяющуюся нагрузку, обеспечивая оптимальное использование ресурсов. Этот пул ресурсов обычно предоставляется по принципу «оплата по мере использования». При этом владелец облака гарантирует качество обслуживания на основе определенных соглашений с пользователем».

Все определения иллюстрируют одну простую мысль: феномен облачных вычислений объединяет несколько различных концепций информационных технологий и представляет собой новую парадигму предоставления информационных ресурсов (аппаратных и программных комплексов). Со стороны владельца вычислительных ресурсов облачные вычисления ориентированы на предоставление информационных ресурсов внешним пользователям. Со стороны пользователя, облачные вычисления – это получение информационных ресурсов в виде услуги у внешнего поставщика, оплата за которую производится в зависимости от объема потребленных ресурсов согласно установленному тарифу. Ключевыми характеристиками облачных вычислений являются масштабируемость и виртуализация.

Масштабируемость представляет собой возможность динамической настройки информационных ресурсов к изменяющейся нагрузке, например к увеличению или уменьшению количества пользователей, изменению необходимой емкости хранилищ данных или вычислительной мощности. *Виртуализация*, которая также рассматривается как важнейшая технология всех облачных

систем, в основном используется для обеспечения абстракции и инкапсуляции. Абстракция позволяет унифицировать «сырые» вычислительные, коммуникационные ресурсы и хранилища информации в виде пула ресурсов и выстроить унифицированный слой ресурсов, который содержит те же ресурсы, но в абстрагированном виде. Они представляются пользователям и верхним слоям облачных систем как виртуализованные серверы, кластеры серверов, файловые системы и СУБД. Инкапсуляция приложений повышает безопасность, управляемость и изолированность. Еще одной важной особенностью облачных платформ является интеграция аппаратных ресурсов и системного ПО с приложениями, которые предоставляются конечному пользователю в виде сервисов.

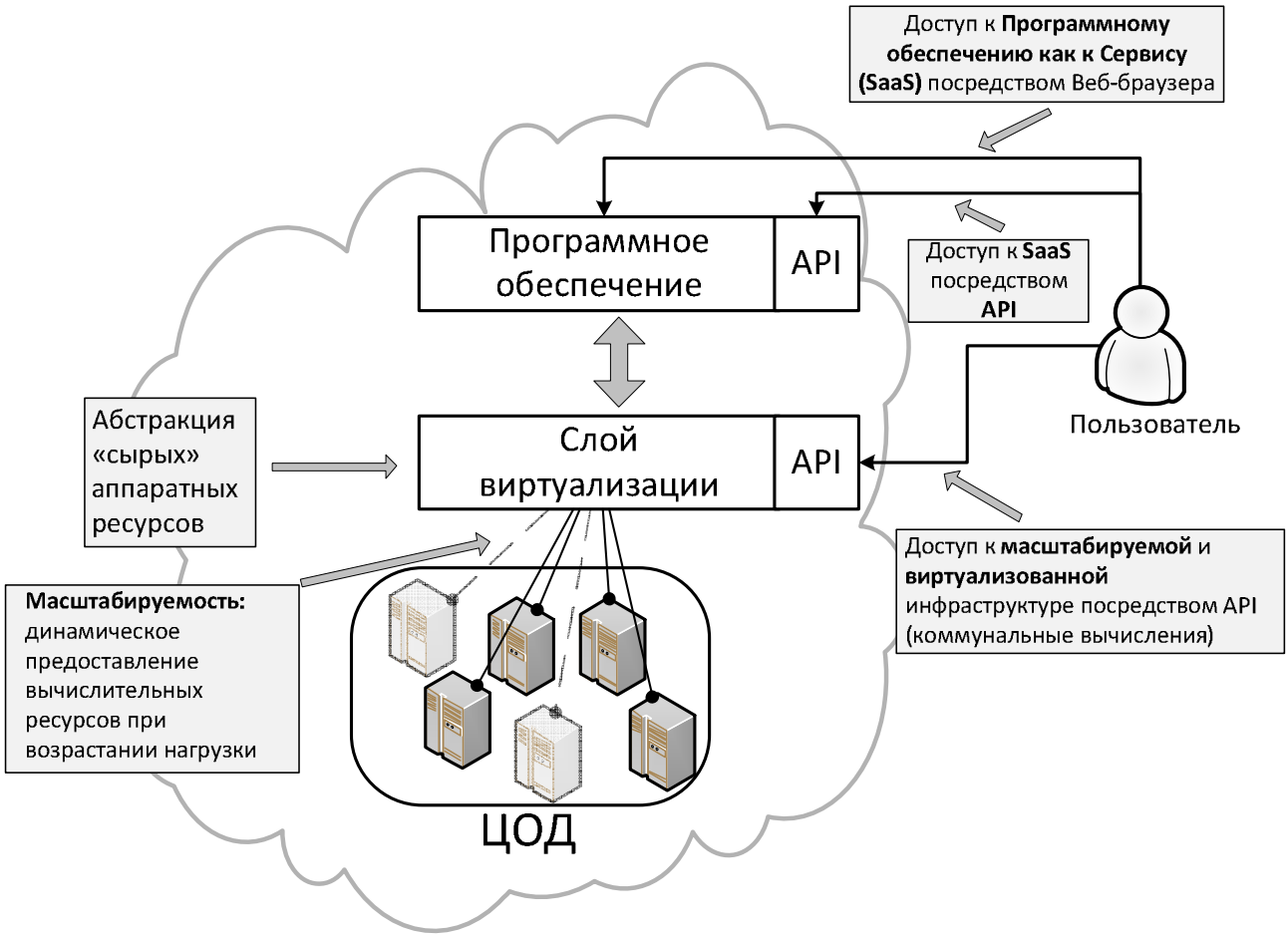

Рис. 60. Основные особенности облачных вычислений

В соответствии со всем вышесказанным, можно выделить следующие основные черты облачных вычислений:

– облачные вычисления представляют собой новую парадигму предоставления вычислительных ресурсов;
– базовые инфраструктурные ресурсы (аппаратные ресурсы, системы хранения данных, системное ПО) и приложения предоставляются в виде серви-

сов. Данные сервисы могут предоставляться независимым поставщиком для внешних пользователей по принципу «оплата по мере использования»;
- основными особенностями облачных вычислений являются виртуализация и динамическая масштабируемость;
- облачные сервисы могут предоставляться конечному пользователю через веб-браузер или посредством определенного программного интерфейса.

12.2 Многослойная архитектура облачных приложений

Все возможные методы классификации облаков можно свести к трехслойной архитектуре облачных систем, состоящей из следующих уровней:
- Инфраструктура как сервис (Infrastructure as a Service: IaaS);
- Платформа как сервис (Platform as a Service: PaaS);
- Программное обеспечение как сервис (Software as a Service: SaaS).

Рассмотрим более подробно, что собой представляет каждый из указанных уровней, и каким образом они взаимодействуют друг с другом.

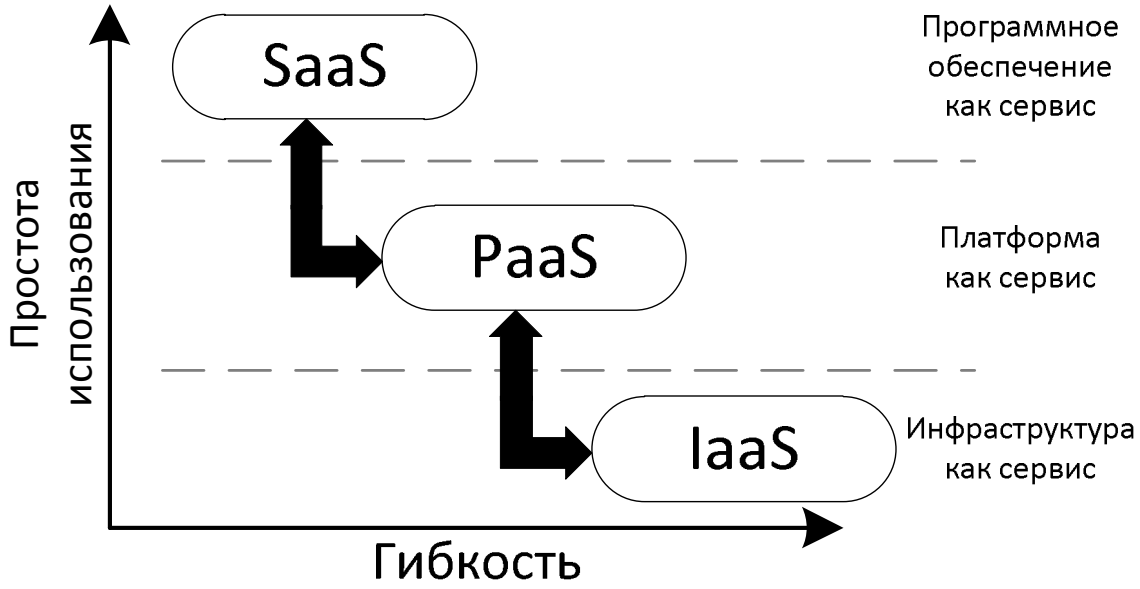

Рис. 61. Три слоя облачных вычислений

12.2.1 Инфраструктура как сервис (IaaS)

IaaS предлагает информационные ресурсы, такие как вычислительные циклы или ресурсы хранения информации, в виде сервиса. Ярким примером такого подхода является облако компании Amazon – Amazon Web Services, состоящее из Elastic Compute Cloud (EC2), предоставляющего информационные ресурсы в виде сервисов и Simple Storage Service (S3) для хранения информации. Другим примером такого подхода может являться сервис Joynet обеспечиваю-

щий хостинг высоко-масштабируемых веб-сайтов и веб-приложений. Вместо предоставления доступа к «сырым» вычислительным устройствам и системам хранения, поставщики IaaS обычно предоставляют виртуализованную инфраструктуру в виде сервиса. Обычно «сырые» ресурсы (процессорные циклы, сетевое оборудование, системы хранения) располагают на базовом уровне, над которым посредством виртуализации надстраивают слои сервисов, которые и предоставляются конечным пользователям в виде IaaS.

Надо сказать, что еще задолго до появления облачных вычислений инфраструктура была доступна как сервис. Такой подход назывался «коммунальные вычисления», и это словосочетание часто применяется некоторыми авторами при описании инфраструктурного уровня облачных систем. Например, в марте 2006 года компания Sun запустила систему Sun Grid Compute Utility. Эта система предоставляла пользователям вычислительные ресурсы по цене 1$ за 1 процессоро-час, то есть работала по принципу «оплата по мере использования». По мере развития этой системы, Sun открыла каталог приложений Application Catalog, который позволял разработчикам с легкостью запускать их приложения online, а двумя годами позже Sun анонсировала Open Cloud Platform, которая должна была стать конкурентом Amazon Web Services. К сожалению, проект Sun Cloud был закрыт сразу же после поглощения Sun компанией Oracle в 2010 году.

В сравнении с ранними попытками организации коммунальных вычислений, подход IaaS предоставляет разработчикам понятный интерфейс, к которому легко получить доступ и использовать в собственных приложениях. Данный интерфейс должен легко интегрироваться с инфраструктурой потенциальных пользователей и разработчиков решений SaaS. Давно замечено, что ресурсы поставщиков коммунальных вычислений могут быть эффективно использованы только в том случае, если они используются большим числом потребителей, а этого можно добиться путем организации хорошего программного интерфейса к своим ресурсам.

12.2.2 Платформа как сервис (PaaS)

Платформа – это слой абстракции между программными приложениями (SaaS) и виртуализованной инфраструктурой (IaaS). Основной целевой аудиторией PaaS являются разработчики приложений. Разработчики могут писать собственные приложения на основе спецификаций определенной платформы, не заботясь о том, каким образом организовать взаимодействие с нижележащей инфраструктурой (IaaS). Разработчики загружают свой программный код на

платформу, которая обеспечивает автоматическое масштабирование приложения в зависимости от нагрузки. Ярким примером реализации подхода PaaS является платформа Google App Enginge, обеспечивающая исполнение пользовательских приложений на инфраструктуре Google. Слой PaaS основывается на стандартизованном интерфейсе, предоставляемом слоем IaaS, который виртуализует базовые вычислительные ресурсы и предоставляет стандартный интерфейс для разработки приложений, функционирующих на слое SaaS.

12.2.3 Программное обеспечение как сервис (SaaS)

SaaS – это программное обеспечение, которое предоставляется по принципу «оплата по мере использования» и управляется удаленно одним или несколькими поставщиками. SaaS – это наиболее заметный слой облачных вычислений, так как именно он представляет реальную ценность для конечного пользователя и обеспечивает решение его задач.

С точки зрения пользователя, основным достоинством SaaS является ценовое преимущество перед «классическим» ПО. Оплата SaaS осуществляется по модели «оплата по мере использования», что означает отсутствие необходимости инвестиций в собственную аппаратную и программную инфраструктуру. Ярким примером SaaS является комплекс Google Apps, включающий в себя такие системы как Google Mail и Google Docs.

Типичный пользователь SaaS не может контролировать базовую инфраструктуру, будь это программная платформа, на которой SaaS основано (PaaS) или же непосредственно аппаратная инфраструктура (IaaS). Однако поставщик SaaS обязан проработать взаимодействие данных слоев, потому что они необходимы для работы системы. Например, SaaS-приложение может быть разработано на базе существующей платформы и исполняться на инфраструктуре, предоставленной сторонней компанией. Работа с платформами и/или инфраструктурой как с сервисом является очень привлекательной с точки зрения поставщиков SaaS, так как это может уменьшить в разы отчисления на используемые лицензии или затраты на инфраструктуру. Это также позволяет им сосредоточиться на тех областях, в которых они по-настоящему компетентны. Как можно заметить, это очень похоже на те преимущества, которые мотивируют конечных пользователей ПО переходить к использованию решений SaaS. По данным рыночных аналитиков, высокое давление рынка приводит компании к необходимости сокращения расходов на ИТ, что приводит к высокому спросу и росту решений SaaS, и, тем самым, росту облачных вычислений в целом.

12.3 Компоненты облачных приложений

На сегодняшний день не существует единой компонентной архитектуры облачных приложений. Это вызвано высокой закрытостью различных аспектов реализации наиболее распространенных облачных систем. Но, не смотря на это, можно выделить основные наиболее важные компоненты, присущие практически всем существующим облачным платформам.

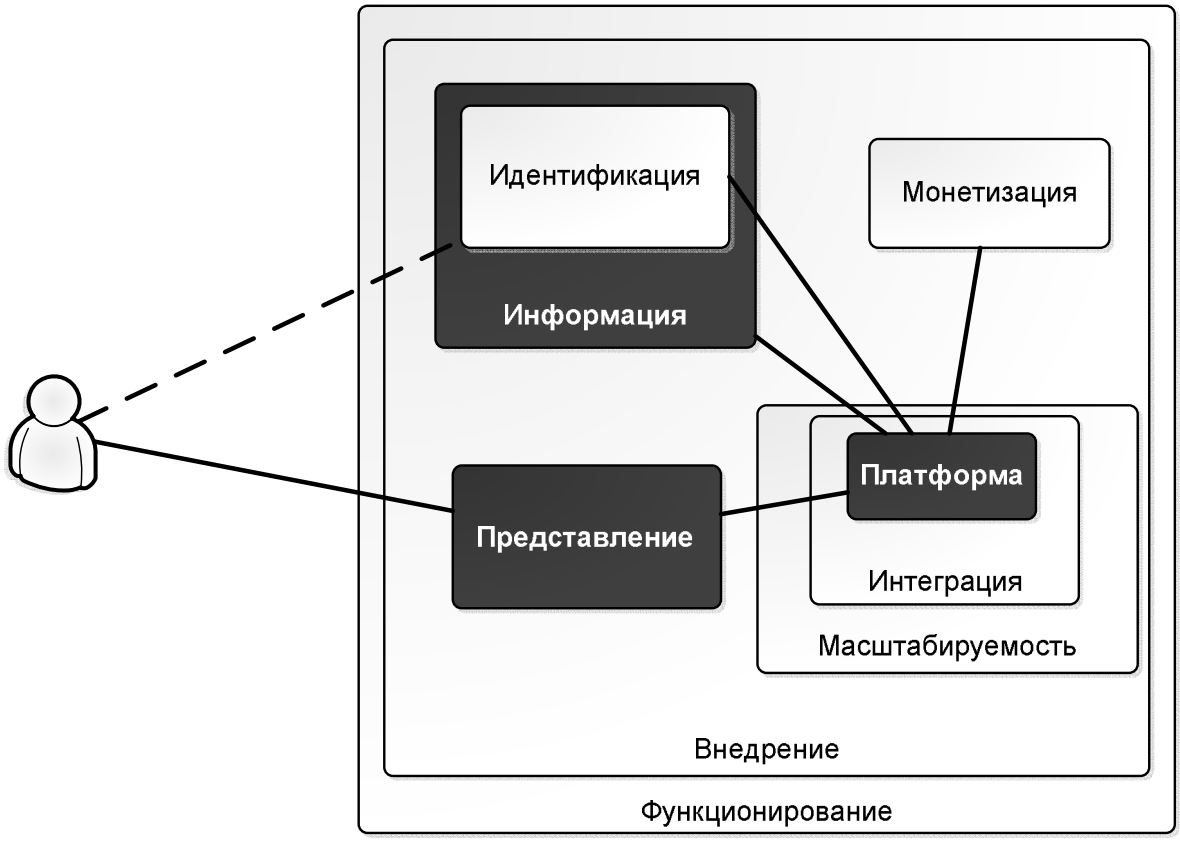

Рис. 62. Компонентная модель облачного решения

Облако можно разбить на основные компоненты, отражающие следующие ключевые особенности облачных решений.

1 **Платформа**: среда и набор утилит, обеспечивающих разработку, интеграцию и предоставление облачных сервисов. Платформа является центральным компонентом модели облака. Платформа с одной стороны, предоставляет набор базовых сервисов, доступных разработчику облачного приложения, а с другой накладывает определенные ограничения на методы разработки и предоставления приложения. При выборе платформы можно основываться как на уже готовых решениях (Google App Engine или Microsoft Azure), так и самостоятельно разработать масштабируемую платформу на базе готовой облачной инфраструктуры. При выборе базовой платформы необходимо исходить из критериев стоимости законченного

решения, производительности и необходимой масштабируемости. Также необходимо помнить, что любая выбранная платформа потребует использования определенных языков программирования и программных фреймворков для реализации приложения.

2 **Представление:** интерфейс, через который пользователь производит взаимодействие с облаком. Этот компонент обеспечивает получение входных данных и предоставление информации конечному пользователю. Наиболее типичным методом реализации представления является веб-приложение, обеспечивающее взаимодействие с пользователем посредством веб-браузера. В последнее время все чаще используются все возможности предоставления пользователю удобного интерфейса работы независимо от устройства, с которого он выходит в Интернет. Это влечет за собой разработку отдельных пользовательских интерфейсов для мобильных устройств (смартфонов, планшетов) которые могут представлять собой как отдельные интернет-страницы, так и полноценные мобильные приложения, взаимодействующие с облаком посредством API.

3 **Информация:** источники данных, обеспечивающие распределенное хранение структурированных или неструктурированных, статических или динамически-изменяющихся данных. Пользовательская информация в облачных системах может достигать огромных объемов. Например, в системе Gmail на начало 2010 года было более 170 миллионов активных пользователей. Даже если предположить, что в среднем используется не более 5% от доступных пользователю 7 Гб, получается что Google необходимо обеспечивать хранение более чем 60 000 Тб данных почтовой переписки. И этот объем экспоненциально увеличивается с каждым месяцем. На таких объемах данных классические SQL базы данных уже не дают удовлетворительных результатов по скорости обработки. Более того, облачным платформам часто приходится обрабатывать связанные структуры данных (графы, деревья) что становится очень тяжело при использовании SQL-подхода и реляционных баз данных. В связи с этим, в последние несколько лет стали активно развиваться альтернативные «NoSQL» системы управления базами данных (колоночные СУБД, такие как Hadoop; документно-ориентированные СУБД, такие как CouchDB и MongoDB) и альтернативные подходы к обработке сверхбольших объемов информации. Наиболее известной реализацией такого подхода является фреймворк MapReduce, представленный компанией Google. Он используется для параллельных

вычислений над очень большими (несколько петабайт) наборами данных в компьютерных кластерах.

4 **Идентификация:** информация об основных потребителях облачных ресурсов, используется для оптимизации и подстройки облака под их задачи. Большинству приложений требуется уметь отличать пользователей друг от друга для предоставления релевантной информации (например, почтовому клиенту необходимо обеспечить аутентификацию и авторизацию пользователей, чтобы представить каждому из них возможность чтения их личной почтовой корреспонденции). Такая информация имеет первостепенную значимость для облачной платформы, так как на нее завязаны большие объемы пользовательских данных, и при этом необходимо обеспечить ее максимальную доступность в рамках системы, т.к. процедура авторизации должна проходить максимально быстро. Также, необходимо обеспечить прозрачную авторизацию пользователя во всех сервисах одной облачной платформы, чтобы не требовалось каждый раз вводить имя пользователя и пароль заново. В связи с распределенной природой облачных сервисов необходимо обеспечить высочайший уровень безопасности при работе с пользовательской информацией.

5 **Интеграция:** инфраструктура, упрощающая обмен информацией и исполнение задач в распределенной вычислительной среде. Высокая степень декомпозиции сервисов позволяет достичь максимальной эффективности и гибкости выполнения облачных приложений, так как появляется возможность загрузки сразу нескольких вычислительных машин при исполнении одной пользовательской задачи. В связи с этим появляется необходимость организации обмена информацией и исполнения задач в распределенных вычислительных средах. В рамках этого компонента необходимо обеспечить максимальную производительность и безопасность процесса обмена данными между сервисами. Далее, необходимо обеспечить совместимость форматов данных и разработать механизмы синхронного и асинхронного взаимодействия с унаследованным ПО. На более высоком уровне необходимо обеспечить слабосвязанность программных компонентов и убедиться в отсутствии *узких мест* в программной архитектуре системы.

6 **Масштабируемость:** гибкость методов предоставления ресурсов, обеспечивающая поддержку выделения дополнительных информационных ресурсов при возрастании нагрузки на приложение. При этом необходимо учитывать не только возможность кратковременного увеличения нагрузки на приложение (например, в результате наплыва посетителей после появ-

ления рекламной статьи на одном из популярных Интернет-ресурсов) но и планировать долгосрочное увеличение производительности системы в результате постоянного прироста аудитории. В обоих случаях, необходимо обеспечить декомпозицию облачного приложения на отдельные модульные компоненты, которые могут быть распределены на несколько вычислительных устройств.

7 **Монетизация:** учет и биллинг ресурсов, затраченных на исполнение пользовательских задач. Это ключевой компонент множества коммерческих приложений. Для организации качественного биллинга облачных платформ необходимо организовать сбор и предоставление полноценной информации о всевозможных ресурсах, затрачиваемых на решение пользовательских задач. Также, необходимо обеспечить пользователю возможность удобной и быстрой оплаты затраченных ресурсов.

8 **Внедрение:** процесс разработки нового облачного приложения, который включает в себя разработку, тестирования и внедрение в эксплуатацию. На этапе разработки облачного приложения требуется совсем небольшой объем вычислительных ресурсов, который значительно увеличивается при переходе к этапу нагрузочного тестирования и внедрения в эксплуатацию. При этом очевидно, что применение готовой облачной инфраструктуры позволяет значительно сократить издержки на разработку и внедрение высокомасштабируемого приложения, так как оплата использованных информационных ресурсов производится на основе модели коммунальных вычислений и не требует значительных инвестиций в собственную инфраструктуру. Это позволяет минимизировать начальные затраты и сконцентрировать финансирование на всестороннем тестировании приложения. Но, не смотря на все перечисленные достоинства, необходимо оценить все возможные недостатки модели облачных вычислений, как то сложности организации репликации данных между сервисами, сложность отката на предыдущие версии при появлении неожиданных ошибок в процессе внедрения, необходимость аккуратного и всестороннего тестирования разработанных сервисов на совместимость данных и слаженность работы приложения.

9 **Функционирование:** мониторинг и поддержка приложений, находящихся в стадии эксплуатации. Приложение, которое запущено в эксплуатацию, необходимо администрировать, что может оказаться чрезвычайно сложной задачей, если учесть большое число отдельных сервисов, составляющих облачное приложение. В связи с этим необходимо обеспечить интеграцию

процессов администрирования и управления сервисами в виде единого «центра управления сервисами». Параллельно, в него можно включить мониторинг нагрузки приложения, панель управления пользовательскими задачами и т.п.

Все вышеперечисленные компоненты облачного приложения должны быть запланированы с самого начала разработки для обеспечения высокого уровня масштабируемости и автоматизации.

12.4 Достоинства и недостатки облачных вычислений

Как ранее было описано, облачные вычисления ориентированы на предоставление информационных ресурсов на трех уровнях: уровне инфраструктур (IaaS), уровне платформ (PaaS) и уровне программного обеспечения (SaaS). Предоставляя интерфейсы ко всем трем различным уровням, облака взаимодействуют с тремя различными типами потребителей.

1. *Конечные пользователи*, которые используют SaaS-решения через веб-браузер, или же какие-либо базовые ресурсы инфраструктурного слоя которые предоставляются посредством слоя SaaS (например, облачные ресурсы хранения посредством Dropbox.com).
2. *Корпоративные потребители*, которые могут использовать все три слоя: IaaS – для того, чтобы расширить собственную программно-аппаратную инфраструктуру или получить дополнительные вычислительные ресурсы по требованию; PaaS – для того, чтобы иметь возможность запуска собственных приложений в облаке; SaaS – для получения возможностей тех приложений, которые уже доступны в облаке.
3. *Разработчики и независимые поставщики программного обеспечения*, разрабатывающие приложения, которые предоставляются в виде облачных SaaS-решений. Обычно, эта категория пользователей напрямую взаимодействует со слоем PaaS, и уже через него, опосредованно, со слоем IaaS.

С точки зрения пользователя, основное достоинство облачных вычислений – это модель оплаты ресурсов по мере их использования. Отсутствует необходимость предварительных инвестиций в инфраструктуру: нет необходимости инвестиций в лицензионное ПО, отсутствует необходимость инвестиции в аппаратную инфраструктуру и связанные с этим затраты на обслуживание и персонал. Таким образом, капитальные затраты превращаются в текущие расходы. Покупатели облачных сервисов используют только тот объем информационных ресурсов, который им на самом деле необходим, и оплачивают только тот объ-

ем информационных ресурсов, которыми они реально воспользовались. В то же время, они пользуются такими достоинствами облачных вычислений как масштабируемость и гибкость. Облачные вычисления позволяет легко и быстро предоставить необходимые вычислительные ресурсы по требованию.

Тем не менее, существуют некоторые недостатки модели облачных вычислений, которые вытекают из одной очевидной особенности – облака обслуживают сразу множество различных клиентов. Пользователь облачной платформы не знает, задачи каких пользователей будут исполняться на том или ином сервере, входящем в облачную инфраструктуру. Типичное облако является внешним по отношению к внутренней инфраструктуре любой компании, то есть находится вне зоны ответственности администраторов и служб безопасности компании-потребителя. В то время как для отдельного потребителя это может быть несущественным, крупные компании уделяют этому вопросу очень большое значение

Пользователю приходится полагаться на обещания поставщика облачных решений в вопросах надежности, производительности и качества обслуживания инфраструктуры облака. Использование облаков также сопряжено с высокими рисками безопасности и защиты конфиденциальной информации. Это связано с двумя факторами: 1) необходимость загрузки и получения данных из облака; 2) хранение данных производится на базе удаленных хранилищ, в связи с чем владельцу информации приходится полагаться на гарантии поставщика облачных решений, что несанкционированного доступа к данным не произойдет. Более того, использование облаков требует определенных инвестиций в интеграцию собственной инфраструктуры и приложений в облако. В настоящее время нет стандартов для интерфейсов IaaS, PaaS и SaaS. В связи с этим выбор поставщика облачных решений становится довольно рискованным занятием (вспомним про то, как, на первый взгляд крупнейшая и старейшая облачная платформа Sun Cloud была закрыта и забыта в течение 2-х месяцев после поглощения компанией Oracle).

В связи с этим, необходимо всегда учитывать риски, связанные с использованием облаков. В каждом отдельном случае необходимо внимательно взвесить все потенциальные выгоды и угрозы при переходе на облачную платформу. Также, необходимо решить какие данные и какая обработка может производиться на базе «облачного аутсорсинга», а какие данные лучше никогда не выводить за рамки локальной сети организации.

12.5 Классификация облаков

В общем случае, облака можно классифицировать в соответствии с тем, кто владеет центрами обработки данных, формирующими облако. Далее будет представлена классификация единых облачных систем в соответствии с тем, кто владеет облачной инфраструктурой и классификация распределенных облачных сред в соответствии с тем, какие типы облаков объединяются.

12.5.1 Общественные и частные облака

Как было сказано ранее, облачные вычисления, с точки зрения поставщика ресурсов, можно определить как предоставление информационных ресурсов для внешних клиентов. При этом с точки зрения конечного пользователя, облако обеспечивает получение информационных ресурсов от внешнего поставщика, в виде сервиса доступного через Интерет за определенную плату. Кроме того, мы определили, что ключевыми характеристиками облачных вычислений являются масштабируемость и виртуализация.

Не смотря на это, виртуализация «сырых» аппаратных ресурсов и предоставление их в виде сервисов, не всегда связана с их предоставлением внешним пользователям. Организации часто используют виртуализацию и сервис-ориентированные вычисления для повышения коэффициента использования собственных информационных инфраструктур. Если «классические» методы использования серверного оборудования могут обеспечить 5-15% загрузки, то применение виртуализации может легко вывести этот показатель на уровень от 18% до 38% (у таких провайдеров как Google). Это приводит к уменьшению затрат на поддержку оборудования, затрат на помещения и охлаждение и в целом снижает стоимость владения вычислительными серверами.

Для того чтобы различать облака, которые изначально разрабатывались для использования внешними пользователями и те облака, которые разрабатываются для поддержки внутренней инфраструктуры предприятия часто используют термины «Общественное облако» и «Частное облако».

Общественным облаком называют центры обработки данных, предоставляющие свои ресурсы третьим лицам через Интернет (например, Google или Amazon). Общественное облако не ограничивает базу пользователей, каждый может подключиться к нему и получить ресурсы за определенную плату. Таким образом, общественные облака могут предоставлять свои ресурсы как частным лицам, так и сторонним организациям.

С другой стороны, любая организация может с опаской относиться к идее предоставления собственных внутренних данных через интернет. Тогда на по-

мощь приходит концепция *частного облака*, которое может быть развернуто в рамках внутренней сети любой организации. Частное облако полностью контролируется той организацией, на базе которой оно развернуто, включая управление доступными приложениями, инфраструктурой, физическим расположением вычислительных узлов и заканчивая управлением пользователями облака. Основным достоинством такого подхода, по сравнению с классическим методом построения центров обработки данных является увеличение коэффициента использования вычислительных ресурсов за счет применения технологии виртуализации.

Не смотря на то, что разница между частными и общественными облаками, по большому счету, состоит только в масштабе, многие исследователи и разработчики считают, что частные облака не могут быть отнесены к «истинной» облачной концепции, так как они не предоставляют информационные ресурсы внешним пользователям. Также, частные облака не обладают «потенциально бесконечной масштабируемостью и гибкостью настоящих облачных платформ».

12.5.2 Гибридные облака и федерации облачных платформ

Отдельные облака могут быть объединены в многооблачные вычислительные среды. В зависимости от того, какие базовые платформы используются в таких облачных объединениях, их можно разделить на следующие типы:

– гибридные облака;
– федерации облаков.

Гибридные облака объединяют общественные и частные облака, позволяя организациям запускать часть приложений внутри частного облака, а другую часть данных передавать на обработку в общественное облако. Такой подход позволяет распределить данные организации: с одной стороны, воспользоваться возможностями общественного облака для быстрой обработки больших объемов данных, а с другой стороны сохранить частную информацию в рамках корпоративной сети. Но такое решение может повлечь значительное усложнение логики распределения приложений и данных в рамках организации. Появляется необходимость мониторинга внутренней и внешней вычислительной инфраструктуры, усложнение политик безопасности и т.п. В связи с этим, такое решение не подходит для задач, связанных со сложным обменом данных.

Федерации облаков (federated clouds) представляют собой объединения нескольких общественных облачных платформ (хотя частные облака также могут входить в такие федерации). Не смотря на то, что с точки зрения пользователя

общественное облако предоставляет практически бесконечную масштабируемость и может справиться практически с любой нагрузкой, поставщики общественных облаков могут столкнуться с проблемой недостатка того или иного информационного ресурса (канала данных, вычислительной мощности и др.) в связи с большим количеством пользователей. Следовательно, в таком случае поставщикам облачных услуг приходится объединять свои инфраструктуры чтобы удовлетворить спрос потребителей на необходимые информационные ресурсы.

Федерации облаков представляют собой коллекцию отдельных облачных платформ, которые могут обмениваться между собой данными и вычислительными ресурсами посредством определенных интерфейсов. В соответствии с принципами федерации, каждое облако остается независимым, но может взаимодействовать с другими облаками посредством стандартного интерфейса. В настоящее время, федерация облаков – это скорее теоретическая концепция, в связи с тем что не существует общепринятого стандарта меж-облачного взаимодействия. Хотя сегодня предпринимаются первые шаги в этом направлении: организация Open Grid Forum, в настоящий момент, разрабатывает такой стандарт под названием Open Cloud Computing Interface. Целью данной работы является разработка стандартизованного API, который позволил бы обеспечить коммуникацию между различными поставщиками облачных услуг, а также обеспечил бы появление новых моделей предоставления ресурсов:

- интеграторов, которые предоставляли бы услуги по распределению работ по различным облакам;
- агрегаторов, которые предлагали бы единый интерфейс ко множеству различных облачных платформ.

Многие разработчики считают, что появление открытых стандартов меж-облачной коммуникации может очень серьезно повлиять на дальнейшее развитие всей технологии облачных вычислений.

12.6 Наиболее распространенные облачные платформы

Рассмотрим возможности и организацию наиболее распространенных на сегодняшний день платформ облачных вычислений: Amazon Web Services, Google App Engine и Microsoft Windows Azure (таблица 2).

Таблица 2. Сравнение платформ облачных вычислений

Платформы / Характеристики	Amazon Web Services	Google App Engine	Microsoft Windows Azure
Тип	IaaS	PaaS	PaaS
Разрабатываемые сервисы	Вычислительные сервисы, сервисы хранения	Web-приложения	Как Web-приложения, так и не Web-приложения
Виртуализация	Уровня ОС, с запущенным гипервизором Xen	Контейнер приложений	Уровня ОС
Интерфейс доступа пользователя	Утилиты консоли Amazon EC2	Web-консоль администрирования	Портал Microsoft Windows Azure
Web APIs	Да	Да	Да
Среда разработки	Отсутствует	Python, Java	Microsoft .NET

12.6.1 Amazon Web Services

Решение, предлагаемое компанией Amazon, стало стандартом «de facto» в области облачной инфраструктуры. Не смотря на то, что решение, предлагаемое компанией Amazon, не является уникальным, все остальные решения рассматриваются либо как дополнения к тому, что предлагает Amazon Web Services (AWS), либо как конкуренты (хотя может быть и не явные) данной платформы. Основным отличием AWS от всех остальных платформ является то, что каждый разработчик может использовать собственную среду исполнения приложений. При этом можно выбрать один из множества образов виртуальных машин предлагаемых компанией Amazon, так и использовать собственное решение.

На рисунке представлены наиболее существенные элементы решения, предлагаемого в рамках системы AWS. Как можно заметить, Amazon предлагает очень широкий набор возможностей организации вычислений, хранения информации и предоставления информации. При этом множество сервисов по управлению и биллингу, предлагаемые компанией, просто не поместились на диаграмму.

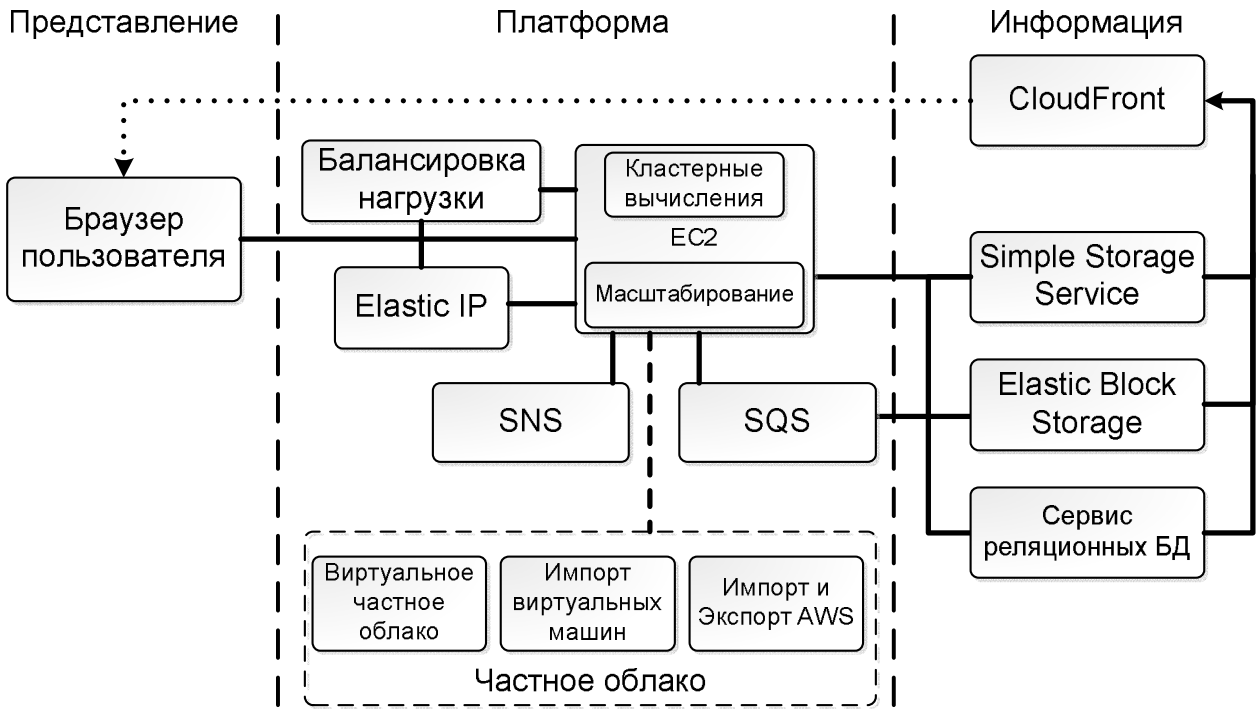

Рис. 63. Архитектура Amazon Web Services

Ядром Amazon Web Services служит система Amazon Elastic Compute Cloud (Amazon EC2) совместно с сервисами хранения. EC2 предоставляет пользователю выбор виртуальных машин, которые можно запустить в распределенной вычислительной среде. Виртуальная машина (называется Amazon Machine Image – AMI) может быть основана практически на любой операционной системе (различные версии Windows и дистрибутивы Linux) и обеспечивать работу с любой программной инфраструктурой (MySQL, Oracle и др.).

Наряду с виртуальными машинами, Amazon обеспечивает несколько механизмов хранения данных:

– *Simple Storage Service (S3)* представляет собой сервис хранения данных, предоставляющий доступ на основе REST и SOAP. Система S3 распределена по всему миру: сервера расположены в Европе, Азии и Соединенных Штатах Америки. При этом обеспечивается возможность работы с данными размером от одного байте до 5TB.
– *Система Elastic Block Storage (EBS)* представляет собой высокопроизводительный виртуальный жесткий диск размером от 1Гб до 1Тб. Он может быть подключен к любой виртуальной машине работающей в рамках EC2 или же может быть помещен на длительное хранение в систему S3.
– *Сервисы реляционных баз данных* – это веб-сервисы, обеспечивающие установку, управление и масштабирование реляционных баз данных в об-

лаке. В настоящий момент предоставляется поддержка баз данных на основе MySQL (в дальнейшем планируется внедрение и баз данных Oracle).

– *Amazon CloudFront* является веб-сервисом для предоставления контента. Обеспечивается статическая и потоковая передача данных. В связи с распределенной инфраструктурой системы, CloudFront может обеспечить минимальную латентность предоставления информации, выбирая наиболее географически-близкий к пользователю сервер.

Одной из отличительных особенностей платформы, предоставляемой Amazon, является предоставление сервисов интеграции приложений, обеспечивающих прозрачную адресацию и доступность:

– *Elastic IP* обеспечивает привязку статического IP адреса к аккаунту пользователя. При этом, Elastic IP может обеспечить обход ошибок и сбоев в работе отдельных виртуальных машин, обеспечивая автоматическое переназначение IP адреса другой виртуальной машине.
– *Simple Queue Service (SQS)* обеспечивает разработчиков практически бесконечным количеством очередей. Каждое авторизованное приложение может регистрироваться в очереди, отправлять, получать или удалять сообщения. При этом не принятые сообщения могут оставаться в системе вплоть до 4-х дней.
– *Simple Notification Service (SNS)* – это сервис, обеспечивающий оповещение об изменении состояния по подписке. Пользователи системы, облачные приложения и устройства могут отправлять и получать оповещения из облака.
– *Сервисы реляционных баз данных* – это веб-сервисы, обеспечивающие установку, управление и масштабирование реляционных баз данных в облаке. В настоящий момент предоставляется поддержка баз данных на основе MySQL (в дальнейшем планируется внедрение и баз данных Oracle).

Также, в настоящий момент Amazon выходит на корпоративный рынок, предоставляя услуги создания виртуальных частных облаков, расширяющих возможности корпоративных вычислительных инфраструктур. По сути, предлагается организация виртуальных частных сетей (VPN), обеспечивающих безопасную и прозрачную связь между внутренней корпоративной сетью и EC2. Обеспечивается импорт корпоративных виртуальных машин, а также возможность пересылки виртуальных машин на физических носителях, минуя интернет (повышается надежность и безопасность передачи данных).

Решение, предлагаемое Amazon Web Services, обеспечивает максимально возможную гибкость при разработке и внедрении облачных решений. Наиболее активными пользователями данной платформы являются крупные организации, или проекты, которым необходима максимальная масштабируемость и гибкость разработки. При этом такое решение может оказаться чрезмерно сложным и нагруженным для отдельных разработчиков или приложений, которым не требуется настолько мощные механизмы масштабирования.

12.6.2 Google App Engine

Платформа Google App Engine является одним из наиболее известных облачных сервисов, ориентированных на предоставление платформы (PaaS) для облачных приложений. Наряду с готовой средой исполнения облачных приложений, предоставляются расширенные средства для администрирования и разработки приложений. На рисунке представлена принципиальная схема архитектуры системы Google App Engine.

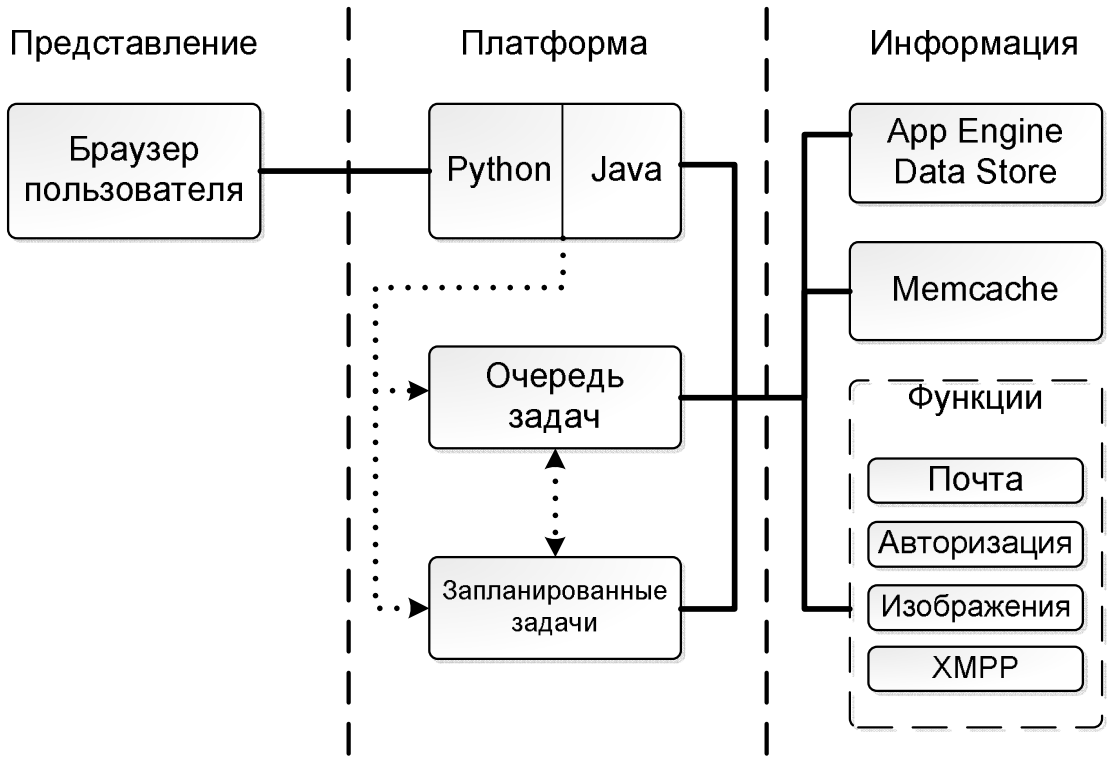

Рис. 64. Архитектура системы Google App Engine

На рисунке представлена упрощенная схема архитектуры системы Google App Engine. Пользователь получает доступ к облачному приложению посредством веб-браузера. До недавнего времени Google App Engine обеспечивало возможность разработки только на базе языка Python, но недавно была представлена поддержка разработки на базе любого языка, который может исполняться внутри виртуальной машины Java. Таким образом, в настоящий момент,

Google App Engine может запускать приложения написанные на Java, Jython, Scala и т.п. Разработчикам Google App Engine предоставляется полноценный комплект средств разработки (SDK), включающий в себя полноценную симуляцию работы Google App Engine на рабочей машине.

Исполнение в облаке накладывает определенные ограничения на набор библиотек, доступных разработчику. Например, в настоящее время не поддерживаются модули Python, написанные с использованием языка C. Аналогично Java-приложения могут использовать только ограниченный подкласс библиотек JRE SE, а также не могут работать с потоками. В связи с этим практически невозможно просто взять и скопировать готовое Python или Java приложение в Google App Engine и надеяться, что оно запустится в облаке. Основные трудности, с которыми придется столкнуться при переносе приложения в облако Google App Engine – это работа с хранилищем App Engine Datastore и ограничения доступа к локальным ресурсам в связи с работой в «песочнице» виртуальной машины.

Хранилище *App Engine Datastore* – это высоко распределенная система хранения данных, основывающаяся на проприетарной базе данных BigTable, разработанной компанией Google. Методы работы с хранилищем очень сильно отличаются от методов работы с реляционными базами данных. Основным отличием является то, что в App Engine Datastore отсутствуют схемы данных. Все данные хранятся в виде блоков «ключ»-«значение»-«метка времени». Соответственно, вся ответственность на соответствие сохраненных данных бизнес-логике приложения ложится на разработчика. Естественно, в SDK предоставляются специальные библиотеки, которые позволяют обеспечить согласованность данных. Также, Google разработали собственный язык запросов (язык GQL), который похож на выражения SELECT в языке SQL, но с большим количеством ограничений (например, отсутствует оператор JOIN).

Вторым важным ограничением, накладываемым на разработчиков приложений в рамках Google App Engine, является необходимость работы в рамках «песочницы» (виртуальной машины с ограниченным доступом к локальным ресурсам). Например, значительно ограничен набор протоколов и портов, по которым приложение может связываться со внешним миром (практически оставлена возможность связи только посредством HTTP и HTTPS). Также, приложению запрещены операции работы с локальной файловой системой (разрешено только чтение файлов, которые были загружены вместе с кодом приложения). С другой стороны, Google App Engine предоставляет большой набор библиотечных функций для выполнения стандартных операций:

– работы с почтовыми сообщениями;
– авторизации и аутентификации пользователей;
– обработки изображений;
– загрузки и обработки веб-страниц;
– планирования задач;
– обработки данных посредством MapReduce;
– хранения больших (до 2 GB) объемов информации;
– обмена сообщениями на основе протокола XMPP (Jabber).

Таким образом, не смотря на то, что Google App Engine не позволит вам развернуть облачное приложение, просто скопировав ваш старый код на Java или Python, он может обеспечить возможность предоставления высокопроизводительного сервиса без существенных затрат на инфраструктуру. Но процесс разработки облачного приложения может потребовать больших затрат, связанных с использованием новых моделей работы с данными и ограничениями виртуальной машины.

12.6.3 Microsoft Windows Azure

Платформа Windows Azure это PaaS от корпорации Microsoft, позволяющая запускать приложения, разработанные на базе платформы Microsoft .NET на серверах компании Microsoft [3]. Обеспечивается автоматическое управление вычислительными ресурсами, балансировка нагрузки и репликация данных.

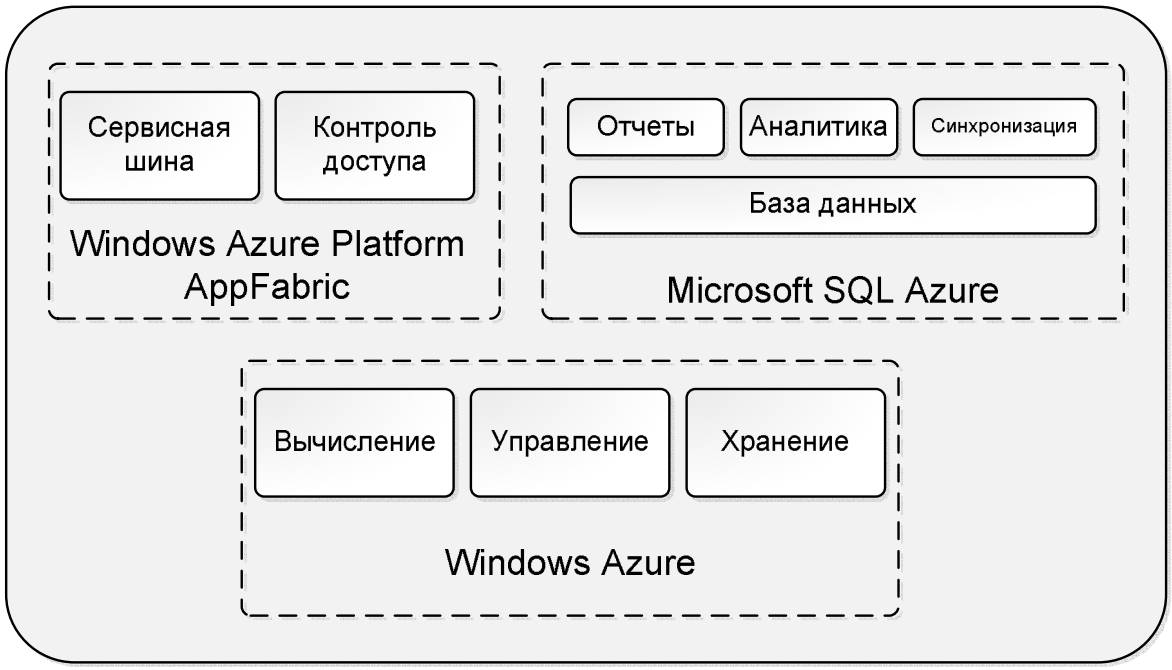

Рис. 65. Платформа Windows Azure

Доступ разработчика к платформе осуществляется посредством инструментария, интегрированного в последние версии Visual Studio (Windows Azure SDK, Windows Azure Tools for MVS). При этом поддерживается возможность локального тестирования приложений до их публикации на сервисе Azure.

Windows Azure создана на основе технологий виртуализации, схожих с технологией Windows Server Hyper-V и управляется с помощью специального инфраструктурного слоя, называемого Windows Azure Fabric Controller. Задача данного слоя – организация массива виртуальных машин на основе Windows Server в виде единого виртуального сервера, обеспечивающей автоматическое управление ресурсами, балансировкой нагрузки и др.

Платформа Windows Azure состоит из трех основных компонент: вычислительных сущностей, сущностей хранения и фабрик.

Вычислительные сущности – это контейнеры для приложений с поддержкой современных технологий разработки, включая .NET, Java, PHP, Python, Ruby on Rails и нативный код. Существует 2 основных роли, которые могут исполнять вычислительные сущности. *Веб-роль (Web Role)* – это веб-приложение, доступное пользователю через интернет, доступ к которой можно получить посредством веб-браузера. *Прикладная роль (Worker Role)* – это приложение, которое выполняет некоторую вычислительную нагрузку в фоновом режиме (что-то на подобие службы в Microsoft Windows).

Сущности хранения – это набор сервисов, обеспечивающих хранение данных. Каждый сервис обеспечивает свой тип хранения данных.

- *Бинарные объекты* – обеспечивается хранение больших наборов неструктурированных байтов (файлов). Обеспечивается возможность именования файлов и работы с метаданными. Максимальный объем хранимого файла – не более 1 Тб.
- *Таблицы* – используются для хранения структурированных данных. Таблицы представляют собой набор однородных строк (называемых сущностями) структура которых определяется набором колонок (называемых свойствами). Один объект может иметь до 256 свойств. Таблицы распределены таким образом, чтобы максимально поддерживать балансировку нагрузок.
- *Очереди* – механизм обеспечивающий асинхронное взаимодействие приложений посредством хранения и передачи сообщений. Допускается использование неограниченного числа очередей, а очереди могут содержать неограниченное число сообщений.
- *Диски* – аналогично AWS, Windows Azure позволяет управлять дисками в рамках своих виртуальных машин. Azure позволяет работать с томами

NTFS, обеспечивая их доступность для приложений. Это может позволить упростить процесс миграции обычных приложений в облако, т.к. появляется возможность сохранения состояния в файловой системе.

Также платформа Windows Azure обеспечивает ряд сервисов, доступных разработчикам как интернет-, так и классических десктопных приложений. Наиболее важным является компонент *SQL Azure*, обеспечивающий предоставление реляционной базы данных Microsoft SQL Server как сервиса. Так как данная технология основана на классическом подходе SQL Server, она не обеспечивает такой массивной масштабируемости как сущности хранения (максимальный объем хранимой информации составляет 10 Гб на одну базу данных). Но, не смотря на это, нельзя недооценивать все те возможности, которые могут быть предоставлены классической архитектурой, включая транзакционную целостность и возможность анализа данных. Также, предоставляются службы построения отчетов.

Очевидно, что решение Microsoft будет пользоваться популярностью на рынке облачных сервисов. Прежде всего, это связано с распространенностью решений Microsoft на рынке корпоративных систем, а технология Windows Azure предоставляет множество сервисов, призванных облегчить перенос таких приложений в облачную платформу. При этом естественно, что вся инфраструктура Azure заточена на программные продукты и технологии, предоставляемые компанией Microsoft. Соответственно, если ваше приложение разрабатывается на основе альтернативной идеологии, весьма возможно стоит поискать альтернативные облачные платформы.

12.7 Сравнение Грид и Облачных вычислений

Описание концепции грид-вычислений в предыдущей главе и концепции облачных вычислений, которая была представлена в этой, показывают, что между ними много общего. Это повлекло за собой множество дискуссий, как в коммерческой, так и в научной среде. Основной вопрос этих дискуссий заключался в следующем: чем облачные и грид вычисления отличаются друг от друга? Действительно ли термин «облачные вычисления» – это просто новая маркетинговая уловка, под соусом которой публике предоставляются услуги грид-сред?

Некоторые исследователи считают, что основным отличием облачных вычислений от грид является виртуализация: «Облачные вычисления, в отличие от грид, применяют виртуализацию для максимизации вычислительной мощно-

сти. Виртуализация, посредством отделения логического уровня от физического, решает множество проблем, с которыми сталкиваются грид-решения».

В то время как грид-системы обеспечивают высокую загрузку вычислительных ресурсов посредством распределения одной сложной задачи на несколько вычислительных узлов, облачные вычисления идут по пути исполнения нескольких задач на одном сервере в виде виртуальных машин. Кроме того, есть особенности в основных вариантах использования грид и облачных вычислений. Тогда как грид, в основном, используется для решения задач за определенный (ограниченный) промежуток времени, облачные вычисления в основном ориентированы на предоставление «долгоживущих» сервисов.

Мнения сходятся в одном – облачные вычисления выросли из концепции грид. Фостер определяет взаимодействие грид и облачных вычислений следующим образом:

«Мы считаем, что облачные вычисления не просто пересекаются с концепцией грид. На самом деле облака выросли из грид-вычислений и основываются на концепции инфраструктуры грид. Эволюция подхода заключается в том, что вместо предоставления «сырых» вычислительных ресурсов и ресурсов хранения обеспечивается предоставление более абстрактных ресурсов в виде сервисов».

Таким образом, можно считать что грид и облачные вычисления дополняют друг друга. Интерфейсы и протоколы грид могут обеспечить взаимодействие между облачными ресурсами или же обеспечить объединение облачных платформ. Также, более высокий уровень абстракции, предоставляемый облачными платформами, может помочь пользователям грид-систем в организации прозрачного и удобного предоставления ресурсов грид-платформ и привлечь новые группы пользователей к использованию таких ресурсов.

С точки зрения пользователя, разница между облачными вычислениями и грид вычислениями будет состоять в следующем:

– *Облачные платформы фокусируются на подходе «всё как сервис»*. Грид вычисления фокусируются на промежуточном программном обеспечении, которое предоставляется в виде открытых исходных кодов или же в виде готовых пакетов. При этом коммунальные вычисления являются всего лишь одной из форм предоставления грид. По сравнению с этим, облачные вычисления фокусируются исключительно на платном предоставлении информационных ресурсов конечному пользователю. При этом промежуточное программное обеспечение, которое позволило бы обеспечить разработку собственного облака, пока не очень распространено.

– *Грид и облачные вычисления фокусируются на различные типы вычислений*. Изначально, грид вычисления были ориентированы на решение научных задач посредством суперкомпьютерных систем. В настоящее время грид применяется для научно-исследовательских задач, решение которых требует объединение нескольких суперкомпьютерных платформ. С другой стороны, облачные вычисления ориентированы не на решение отдельных задач, а на перманентное предоставление определенных сервисов конечным пользователям. Они обеспечивают динамическое распределение физических ресурсов для удовлетворения переменной загрузки таких сервисов.
– *Различное взаимоотношение с поставщиками ресурсов*. Грид вычисления основываются на понятии виртуальных организаций, включающих в себя несколько различных отдельных организаций с четкими правилами взаимодействия между ними и четкими политиками предоставления программно-аппаратных ресурсов. Концепция облачных вычислений обеспечивает возможность любой компании использовать облачные сервисы для решения собственных задач, оплачивая только те ресурсы, которые необходимы.
– *Различные области применения*. Грид-платформы предоставляют базу для развертывания вычислительной инфраструктуры. Облачные вычисления предоставляют интегрированный подход на всех уровнях предоставления информационных ресурсов: IaaS, PaaS, SaaS.
– *Расширение количества пользовательских интерфейсов*. Грид вычисления ориентированы на представление различных вычислительных ресурсов в гетерогенных вычислительных средах для решения конкретных задач. Таким образом, интерфейсы грид ориентированы на взаимодействие вычислительных инфраструктур на физическом уровне посредством API, которым может воспользоваться только профессиональный программист. Облачные вычисления разрабатываются таким образом, чтобы предоставлять интерфейсы конечным пользователям через веб-доступ или посредством API. На каждом слое (IaaS, PaaS, SaaS) предоставляется свой собственный интерфейс. Повышение уровня абстракции позволяет обеспечить применение облачных вычислений как на уровне отдельных пользователей, так и на уровне корпоративных клиентов.

  В общем и целом, грид вычисления обеспечивают объединение гетерогенных вычислительных ресурсов в единую вычислительную среду. Это то, с чего начинаются и на чем основываются облачные вычисления. Облачные

вычисления обеспечивают более высокий уровень абстракции, предоставляя вычислительные ресурсы конечным пользователям (будь то частные клиенты или организации) в виде сервисов.