

Описание слона: многоликость информационных технологий в форме службы

Ян Фостер Стивен Тьюке

* * *

Describing the Elephant: The Different Faces of IT as Service

Ian Foster, Argonne National Laboratory and University of Chicago, Steven Tuecke, Univa

From [Enterprise Distributed Computing](#)
ACM Queue vol. 3, no. 6 - July/August 2005

<http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=319>

*Такие термины, как **грид**, **компьютинг по требованию** и **ориентированная на службы архитектура** не имеют четкого однозначного толкования, но все они отражают суть некой стоящей за ними тенденции.*

В общеизвестной басне группу слепых попросили описать слона. Каждый из них ощупал одну, каждый свою, часть этого животного, и не удивительно, что предложил свое, отличное от других, описание.

С подобного рода путаницей мы сталкиваемся сегодня и в терминологии информационных технологий, когда происходит толкование таких терминов, как *ориентированная на службы архитектура* (**service-oriented architecture – SOA**), *грид* (**grid**), *компьютинг услуг* (**utility computing**), *компьютинг по требованию* (**on demand computing**), *адаптивное предприятие* (**adaptive enterprise**), *автоматизация центра данных* (**data center automation**) и *виртуализация* (**virtualization**). Слушая слепого, трудно понять, что в действительности лежит за его словами, "прилажены" ли и как различные части друг к другу, и как нам следует представить описанное животное. (Конечно, в примере со слепыми мы не имели в виду подразделения, занимающиеся маркетингом).

Цель данной статьи состоит в том, чтобы, в конечном счете, дать представление о слоне. Говоря более конкретно, мы описываем то, что считаем главной технологической тенденцией, которая проявляется во многих связанных работах – а именно, преобразование комплексов вертикально интегрированных программ, не поддерживающих информационное взаимодействие с другими комплексами, в горизонтально интегрированные, ориентированные на службы системы. Мы объясняем, как различные общераспространённые термины связаны с этой стоящей за ними тенденцией и описываем технологию, которая необходима для реализации такого преобразования.

НАДОБНОСТЬ ГОРИЗОНТАЛЬНОЙ ИНТЕГРАЦИИ

Предположим, что Ваш отдел информационных технологий (ИТ) получил заказ на разработку нового приложения, позволяющего получать оценки состояния портфелей клиентов. Разработка этого приложения начнётся с анализа предъявляемых к нему требований и далее проектирования, приобретения оборудования, развертывания (инсталляции) и тестирования. Несколько месяцев спустя новое приложение будет готово в форме прекрасно настроенного, вертикально-интегрированного стека специализированного прикладного и управляющего программного обеспечения, работающего на пуле выделенных серверов.

Со временем это приложение станет более востребованным, и значительные всплески запросов к нему приведут к перегрузке. При этом, несмотря даже на то, что предусмотрен значительный резерв пропускной способности серверов предприятия, оно всё же не сможет справиться с такими всплесками, поскольку стеки других приложений, расположенные на отведенных для них серверах, делают эти серверы недоступными. (Серверы, используемые каждым приложением, работают с совершенно разным программным обеспечением – возможно даже с разными операционными системами – и статически конфигурированы для выполнения одной задачи. И никто не хочет, чтобы посторонние вмешивались в работу этих серверов, так как любое изменение может нарушить работу приложений). Поэтому для того, чтобы повысить производительность обработки, отделу ИТ придётся устанавливать дополнительные серверы.

Такая организация ИТ-ресурсов в виде совокупности более или менее независимых, так называемых **silos-модулей**¹, каждый из которых отвечает только за выполнение определённой функции предприятия или приложения, банальна. Фактически, это является естественным следствием как децентрализации, так и собственных способов виртуализации ресурсов, управления рабочей нагрузкой и использования сегодняшних средств разработки приложений. К тому же эта изоляция становится всё более неприемлемой, поскольку императивами бизнеса являются как снижение капитальных и операционных расходов, так и повышение скорости реакции на бизнес-запросы. Нередко случается так, что отдельный silo-модуль простаивает 90 процентов времени из-за того, что необходимо обеспечивать дополнительную мощность при случайных пиковых нагрузках. Более того, почти каждый silo-модуль требует специального операционного обслуживания.

При поиске решения этих проблем разумно заглянуть в не столь уж далёкое прошлое. Не так давно, мэйнфрейм обеспечивал как общепринятый набор абстракций для разработки прикладных программ, так и развитые средства управления ресурсами, которые позволяли многим приложениям разделять доступные им ресурсы внутри этого мэйнфрейма, и при этом поддерживался соответствующий уровень качества

¹ Энциклопедия Wikipedia (<http://wikipedia.org>) определяет **information silo** как систему управления, неспособную к взаимной работе с другими, релевантными системами управления. Например, банковская система управления считается **silos**, если она не может обмениваться информацией с другими релевантными системами внутри данной организации, или с системами её заказчиков, поставщиков или бизнес-партнёров. – Прим. переводчика

обслуживания. Мэйнфреймы вполне эффективно обслуживали множество приложений, будучи обычно загружены почти на 100 процентов мощности. По сути дела, централизованные ИТ-архитектуры были *разделены вертикально* и *интегрированы горизонтально*, что обеспечивало возможность многократного использования функции внутри приложений и экономию в потреблении ресурсов. Кроме того, все приложения были доступны в пределах единообразной среды.

Рисунок 1 иллюстрирует, как движение к распределённым, недорогим и часто гетерогенным группам серверов, несмотря на целый ряд выгод, тем не менее, привело к дезинтеграции этой централизованной ИТ-архитектуры в изолированные silo-модули. Теперь возникает проблема: как реинтегрировать эти silo-модули, чтобы можно было достигнуть выгод вертикального разделения и горизонтальной интеграции в современной, более сложной распределённой ИТ-среде предприятия. В этом контексте вертикальное разделение означает, что мы так стандартизируем интерфейсы среди прикладных компонент, систем управления рабочей нагрузкой и физических ресурсов, чтобы различные компоненты можно было динамически объединять, если это диктуется требованиями приложений. Горизонтальная интеграция означает, что мы принимаем единые интерфейсы управления, которые дают возможность единообразно и автоматически распределять, использовать, мониторить и управлять большинством ресурсов, распределённых на прежних silo-модулях, улучшая их утилизацию и снижая операционные расходы.

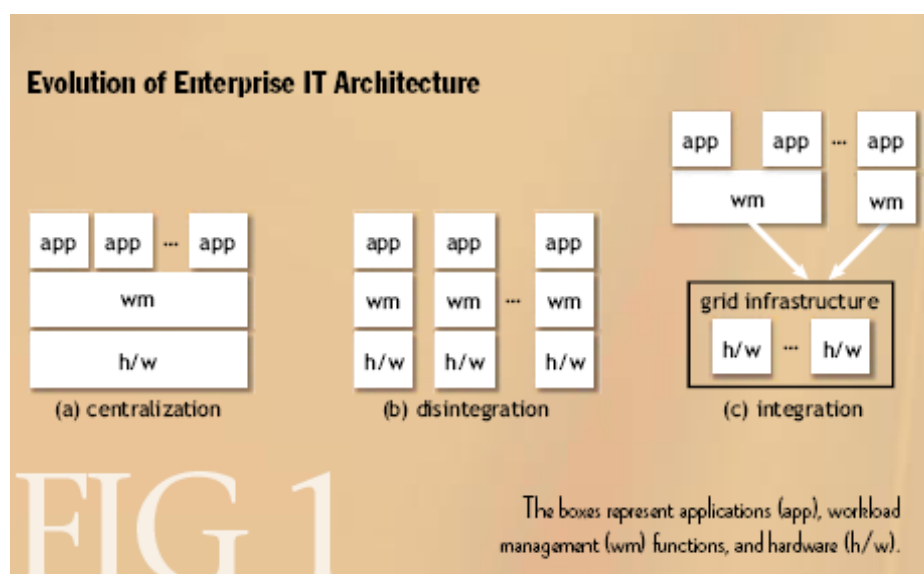


Рисунок 1

Чтобы понять практический смысл этих соображений, рассмотрим, как можно было бы реализовать в горизонтально интегрированной среде ранее описанное приложение оценки портфелей клиентов. Мы бы реализовали наше приложение в соответствии с интерфейсом управления рабочей нагрузкой, позволяющим описывать операции, которые должны быть выполнены приложением, и наши требования к производительности для этих операций. Менеджер рабочей нагрузки, который реализует этот интерфейс, использовал бы тогда обычные механизмы грид-инфраструктуры (развёрнутой на всём аппаратном оборудовании предприятия) для того, чтобы

обнаруживать доступные ресурсы и размещать компоненты приложения на этих ресурсах. По мере возрастания загрузки новые ресурсы могли бы автоматически (и временно) выделяться и использоваться приложением. Таким образом, "приложение" оказывается полностью отделённым от нижележащего аппаратного "оборудования".

СЛОН НАОЩУПЬ: ПОЯСНЕНИЯ СЛЕПЫХ

Теперь обратимся к различным терминам, упоминавшимся во введении, и рассмотрим, как каждый из них связан с целью горизонтальной интеграции.

Один из этих терминов – *grid*, кстати, введённый в употребление авторами, в последние годы привлек внимание многих специалистов по маркетингу. Согласно одному определению *grid* – ”это система, которая использует открытые, общецелевые протоколы для объединения распределённых ресурсов и предоставления самого наилучшего качества обслуживания” [1]. В основе такого толкования термина лежит метафора *grida* электроснабжения, который как технология характеризуется: (а) использованием при передаче электроэнергии стандартов, которые позволяют рассматривать потребителей и поставщиков как изолированные сущности; и (b) объединением разных поставщиков в некую управляемую систему коммунальных услуг [2]. По аналогии, *grid*-технологии делают возможным: (а) доступ по требованию к средствам компьютеринга; и (b) объединяют распределённые ресурсы для удовлетворения потребностей конечных пользователей. Таким образом, *grid* – это важный и ёмкий по значению термин, который употребляется, когда речь идёт о решениях, связанных с гибким использованием распределённых ресурсов для разных приложений, а также, как термин, подчёркивающий особое значение стандартов для интероперабельности.

Мы используем термин *grid-инфраструктура*, чтобы указать на исключительно важный аспект *grid*-пространства, а именно, на уровень горизонтальной интеграции. Термин *grid* также часто применяется к другим уровням программного стека: например, чтобы охарактеризовать приложения или диспетчер рабочей нагрузки, которые структурированы для того, чтобы обеспечить эффективное и гибкое использование распределённых и/или разделяемых ресурсов.

Родственный термин *компьютинг услуг* часто используется для того, чтобы обозначить как разделение между поставщиком услуг и потребителем, так и возможность договариваться о желаемом качестве обслуживания с поставщиком услуг, то есть два свойства, обычно ассоциируемые с услугами, подобными электроснабжению. Поставщиком может быть ИТ-подразделение организации или внешний поставщик услуг, таких, например, как хранение данных, компьютеринг или аренда приложения. (Например, фирма Sun предлагает на основе принципа немедленной оплаты вычислительные услуги при стоимости \$1 за час процессорного времени, в то время как поставщик *salesforce.com* предлагает в виде услуги приложение для управления отношениями с потребителями).

Термин *по требованию* широко используется для обозначения технологий и систем, которые позволяют пользователям или приложениям получать дополнительные ресурсы, потребность в которых возникла в связи с изменением рабочих требований. Поэтому мы можем говорить о предоставлении по требованию таких видов ресурсов, как компьютеры, системы хранения, пропускная способность канала связи и приложения. Здесь имеет место значительное перекрытие с терминологией компьютеринга услуг:

термины *обслуживание хранения (storage utility)* и *хранение по требованию (on-demand storage)* имеют одинаковый смысл.

Значения терминов *услуга, по требованию* и *грид* значительно перекрываются по смыслу. Все они придают ИТ сопутствующее значение службы, суть которой в том, что она обеспечивает доступ к физическим ресурсам или другим службам через сеть на основе использования некоторого стандартизированного протокола. Компьютинг услуг и компьютеринг по требованию можно рассматривать как специализированные формы использования грида (см. рисунок 2), хотя, конечно же, сегодня многие системы компьютеринга услуг и компьютеринга по требованию не используют грид-инфраструктуру. Фактически, обстоятельством, сдерживающим освоение компьютеринга услуг и компьютеринга по требованию, является отсутствие стандартов для горизонтальной интеграции. Без стандартных механизмов для обнаружения, проведения переговоров о получении доступа, конфигурирования и управления удалёнными ресурсами, использование этих систем обслуживания часто связано с проведением сложного ручного конфигурирования. Это отбивает желание производителей программного обеспечения и конечных пользователей разрабатывать программное обеспечение, зависящее от использования таких служб.

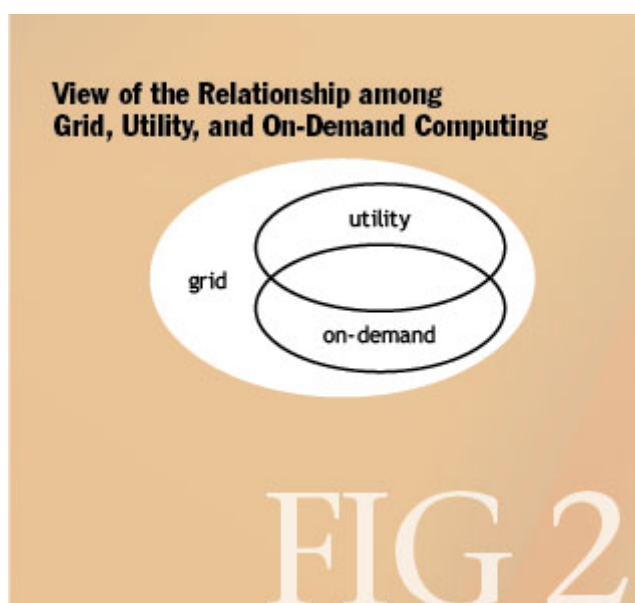


Рисунок 2

Непосредственное отношение к проводимому обсуждению имеет и термин *автоматизация центра данных*, который обычно используется для обозначения продуктов, позволяющих организовать скоординированное управление ресурсами внутри предприятия: например, для поддержки состояния большого числа компьютеров в соответствии с самыми последними программными исправлениями и обновлениями. Цель такого вида обслуживания состоит в том, чтобы автоматизировать те операции с приложениями, которые, как правило, не модифицированы для распределённого выполнения. Для достижения этой цели обычно используется специализированная нижележащая инфраструктура управления, но это же может быть успешно выполнено и средствами общей грид-инфраструктуры.

Термин *кластер* (**cluster**, или, иногда, **Beowulf cluster**, по названию первого проекта) обозначает многопроцессорную систему с неразделяемой оперативной памятью, сконструированную из товарных компонент вычислительной техники. Относительно низкая стоимость кластеров делает их чрезвычайно продуктивными узлами систем распределённого (грид/услуг/по требованию) компьютеринга, в частности потому, что высокий уровень интеграции уменьшает затраты, связанные с их размещением, энергообеспечением и администрированием, а технология виртуальной машины упрощает более гибкую настройку.

Для других терминов характерны как более широкие, так и более узкие определения. Так, например, фирма IBM использует термин *бизнес по требованию* (**on-demand business**), имея в виду "предприятие, бизнес-процессы которого насквозь интегрированы не только по всей его структуре, но также и с ключевыми партнёрами, поставщиками и заказчиками, и которое способно быстро реагировать на любой запрос заказчика, требование рынка или внешнюю угрозу". Термин *адаптивное предприятие* (**adaptive enterprise**), введённый в обращение фирмой Hewlett-Packard, имеет аналогичное дополнительное значение: "организация, которая так адаптируется к условиям рынка, что может отвечать и адекватно реагировать на их изменение, на окружающую рынок обстановку и/или состояние промышленности. Это позволяет более эффективно позиционироваться для выживания и получения прибыли".

Другой родственный термин – *аутсорсинг* (**outsourcing**), подразумевает участие сторонней независимой фирмы (**third party**), которая предлагает выполнять все или часть ИТ-операций предприятия. Аутсорсинг представляет собой скорее некую финансовую стратегию, нежели ИТ-архитектуру: многими соглашениями об аутсорсинге предусматривается, что *меняется* ИТ-персонал, однако приложения остаются размещёнными на тех же *silos*-модулях. Тем не менее, если фирмы, предлагающие аутсорсинг, имеют возможность модифицировать приложения, то для решения задачи повышения эффективности привлекательной может оказаться горизонтальная интеграция, позволяющая снизить расходы на оборудование. Кроме того, успешная реализация методов обслуживания может обеспечить более динамичный аутсорсинг ресурсов для решения вопросов, связанных с изменением рабочей нагрузки.

Ещё один термин – *программное обеспечение в форме службы* (**software as service**) – вошёл в обращение совсем недавно. Он отражает подход, при котором Web используется для предоставления многим потребителям доступа к функциям (в вышеупомянутом случае *salesforce.com* – это управление взаимосвязями с потребителями), которые специально спроектированы для такого режима использования. Таким образом, программное обеспечение в форме службы представляет собой подход для написания приложений и экспонирования интерфейсов пользователям (например, с помощью Web-браузеров). Этот подход может обеспечить хороший бизнес, поскольку производители программного обеспечения в форме службы могут получать значительные доходы за счёт массового использования подобных служб. Производитель программного обеспечения в форме службы знает все тонкости управления своим приложением и поэтому является главным кандидатом для освоения стратегий горизонтальной интеграции инфраструктуры.

Обсуждение общей картины глубоких концепций распределённого компьютеринга было бы незавершённым, если не упомянуть о терминах *ориентированная на службы*

архитектура и *Web-службы (Web services)*. Эти два термина означают, соответственно, набор архитектурных принципов и технологию реализации, которые играют важную роль при реализации ИТ в форме службы, и, как мы обсудим позже, проведения горизонтальной интеграции.

SOA означает подход к проектированию систем, который облегчает реализацию принципов **ИТ в форме службы** и горизонтальную интеграцию целевых решений, упомянутых ранее. Служба представляет собой самодостаточную реализацию некоторой(ых) функции(й) с чётко определённым интерфейсом, который устанавливает шаблоны для обмена сообщениями, используемые при взаимодействиях с функцией(ями). В таком случае, SOA рассматривается как некая совокупность служб. Поэтому, используя ориентированные на службы архитектуры, стремятся достигнуть строгого разделения интерфейса и реализации, необходимого для предоставления других желательных свойств, например таких, как интероперабельность, прозрачность расположения и слабая связанность между службой и клиентом. (Для обсуждения достоинств слабого связывания познакомьтесь с классической работой Кендалла и других [3]).

Web-службы – это совокупность технологий для реализации ориентированных на службы архитектур. Хотя, они и не являются единственным инструментом, который может использоваться для этой цели – например, в прошлом так использовались CORBA и DCOM – *Web-службы* обладают рядом технологических преимуществ по сравнению с ранее применявшимися подходами и широко освоены.

Web-службы определяются базовым комплектом технологических спецификаций, предоставляющих механизмы для описания набора сообщений (то есть интерфейсов), которые могут использоваться для взаимодействия со службой (WSDL) и кодирования сообщений между службами (SOAP). Другие спецификации определяют правила, касающиеся безопасного доступа к службам (WS-Security), адресации служб (WS-Addressing), управления состоянием (WS-Resource Framework), распространения уведомлений (WS-Notification) и т.д. Использование языка XML для описания интерфейсов служб и кодирования сообщений облегчает интеграцию приложений и распределённых систем из независимо определённых и слабо связанных служб.

Технологии создания *Web-служб* в настоящее время уже достаточно продуманы, и многие компании предлагают хорошие коммерческие стеки *Web-служб*, а Apache поддерживает надёжный свободно распространяемый стек. Тем не менее, важно понимать, что эти системы могут делать и чего не могут. Они действительно предоставляют инструментарий для разработки *Web-служб*, как правило, на языках Java или C#, и контейнеры для хостинга этих служб. Вместе с тем большинство из них не решает вопрос о том, как обеспечить универсальный набор абстракций и интерфейсов для эффективного использования ИТ-ресурсов – что является ключевым требованием к инструментарию для реализации горизонтальной интеграции. Как мы обсудим ниже, разработка таких абстракций и интерфейсов ведётся интенсивно, но пока ещё полностью не завершена

Термин *ориентированная на службы инфраструктура* характеризует использование SOA-подходов к проблеме управления ресурсами. Представление ресурсов в виде служб является важным шагом в направлении унификации операций обработки разнообразных компонент. Тем не менее, как мы покажем далее в разделе, посвящённом *грид-инфраструктуре*, главная цель горизонтальной интеграции подразумевает нечто

большее, чем только адаптация технологий Web-служб, использование которых без соответствующей координации приведёт к неструктурированной смеси разнородных, частично перекрывающихся интерфейсов Web-служб для ресурсов. Нам также необходима такая унификация абстракций и интерфейсов для сопряжения широкого набора компонент, которая, например, позволяла бы диспетчеру рабочей нагрузки унифицировано распределять, резервировать и управлять всеми видами ресурсов, предоставляемых различными производителями.

УПРАВЛЕНИЕ

ГОРИЗОНТАЛЬНАЯ ИНТЕГРАЦИЯ

Теперь рассмотрим более подробно детали выбора архитектуры и конструирования ориентированных на службы и горизонтально интегрированных систем, а также имеющиеся технологии их разработки.

Для иллюстрации проводимого обсуждения обратимся к рисунку 3, который дополняет рисунок 1 и показывает главные уровни в горизонтально интегрированной, ориентированной на службы ИТ-архитектуре предприятия. Короче, *приложения* используют *диспетчеры нагрузки* для координации их доступа к физическим ресурсам. При этом в отличие от архитектуры вертикально интегрированных silo-модулей, в данном случае приложение и его диспетчер нагрузки жестко не связаны с единственным физическим ресурсом (или группой ресурсов). Вместо этого, они динамически связываются, то есть *настраиваются*, с ресурсами через уровень общей *грид-инфраструктуры*. Отметим также, что для повышения гибкости в обслуживании своих пользователей, ресурсы сами могут реализовывать различные методы *виртуализации*.

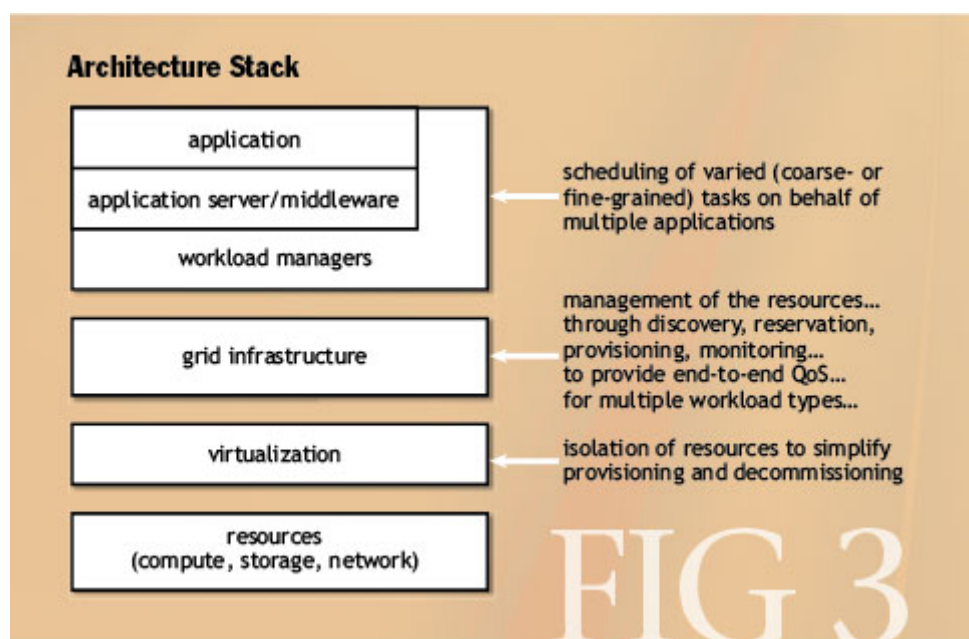


Рисунок 3

ПРИЛОЖЕНИЕ

Приложения предприятия охватывают широкий спектр решений: они могут быть крупномодульными или мелкоструктурными, пакетными или интерактивными, вычислительно- или информационно-ёмкими, обрабатывающими транзакции или нет. Они могут иметь различные формы реализации: фактически, разработка ориентированных на службу или иных приложений и инструментальных пакетов становится огромной индустрией. Главный интерес данной статьи состоит не в том, как разрабатываются приложения, а скорее в том, чтобы показать как требования, предъявляемые к выполнению приложений, отображаются на физических ресурсах. В эру мэйнфреймов решение этой проблемы осуществляла операционная система. В сегодняшних слабосвязанных кластерах и распределённых системах операционные системы уже не обеспечивают такой необходимой поддержки. Поэтому функции отображения, необходимые приложению, должны быть непосредственно встроены в него или – следуя парадигме модульности - должны обеспечиваться отдельными менеджерами рабочей нагрузки, которые мы обсудим в следующем разделе.

Термин *грид-приложение* часто используется тогда, когда речь идёт о приложениях, которые адаптированы к использованию в распределённой инфраструктуре. Такие приложения обычно распараллелены и написаны применительно к динамическому увеличению и сокращению физических ресурсов. В средах прикладного программирования, которые помогают программистам разрабатывать такие приложения, используются продукты фирм Data Synapse и United Devices; библиотека Message Passing Interface; системы рабочих потоков и свободно распространяемые системы, такие, как Condor и Nimrod.

УПРАВЛЕНИЕ РАБОЧЕЙ НАГРУЗКОЙ

Диспетчер рабочей нагрузки – это важная системная компонента комплекса информационных технологий предприятия. В настоящее время используется целый ряд таких систем, приспособленных к разным уровням гранулярности выполняемой задачи, характеристикам транзакций и требованиям производительности. Например, пакетные диспетчеры, типа разработанных фирмами Altair, Platform, Sun и United Devices, или свободно распространяемые системы, такие как Condor, используются при решении относительно крупномодульных задач, связанных с цифровым рендерингом, инженерным анализом и фармакологическими исследованиями. Другие приложения, например, связанные с финансово-экономической деятельностью, требуют поддержки рабочих нагрузок, состоящих из множества мелкоструктурированных задач, и могут использовать диспетчеры рабочих нагрузок фирм Data Synapse, Gigaspaces, Platform или других производителей. А некоторые приложения, связанные с финансово-экономической деятельностью, требуют поддержки для последовательно выполняемых задач, потоков данных и/или оркестровки: посредством, например, использования языка BPEL (**B**usiness **P**rocess **E**xecution **L**anguage). Система управления базами данных Oracle 10G продуктивно использует компьютерные кластеры для приложений баз данных.

Термин *грид* часто употребляется в связи с диспетчерами рабочей нагрузки, которые нацелены на использование разделяемых и/или распределённых ресурсов. Фактически большинство существующих грид-продуктов связаны с упрощением разработки параллельных или распределённых приложений и с управлением выполнения

декомпозиций этих приложений на распределённых ресурсах. Тем не менее, как мы покажем в следующем разделе, для истинного грида, способного поддерживать выполнение на распределённых ресурсах не одного, а множества приложений, требуется нечто большее.

ГРИД-ИНФРАСТРУКТУРА

Предприятие, которое хочет поддерживать выполнение некоего набора приложений на распределённых ресурсах, сталкивается с проблемой, состоящей в том, что разные приложения и связанные с ними диспетчеры нагрузок не интегрируются на уровне инфраструктуры. Вместо этого, каждый из них полагается на собственную реализацию функции управления инфраструктурой. Поэтому такое предприятие с разнообразием требований к приложениям в конце концов приходит к тому, что его ИТ-комплекс – это набор silo-модулей, каждый из которых состоит из отдельного приложения, диспетчера нагрузки и набора ресурсов, то есть к ситуации изображенной на рисунке 1.

Для решения этой проблемы необходимо установить некий общий горизонтальный уровень, на котором определяется и реализуется *согласованный набор абстракций и интерфейсов для доступа и управления разделяемыми ресурсами*. Мы рассматриваем этот уровень горизонтальной интеграции ресурсов как *грид-инфраструктуру*. Это то, что обеспечивает горизонтальную интеграцию в пространстве различных физических ресурсов, которая нам требуется для разъединения приложения и аппаратуры. Этот уровень грид-инфраструктуры составляет суть инструментального пакета Globus [5].

Грид-инфраструктура должна обеспечивать некую совокупность технологических возможностей, таких, как:

- 1 Моделирование ресурсов.** Эта возможность позволяет описывать доступные ресурсы, их характеристики и взаимоотношения, что совершенствует обнаружение, настройку и управление качеством обслуживания.
- 2 Мониторинг и уведомление.** Эти способности обеспечивают обзорность состояния ресурсов и, уведомляя приложения и службы управления инфраструктурой об изменении состояния, позволяют отслеживать и поддерживать качество обслуживания. Протоколирование важных событий и изменений состояний необходимо также и для поддержки функций учета использования ресурсов и аудита.
- 3 Аллокация.** Данная возможность гарантирует качество обслуживания на всём комплекте ресурсов в течение срока использования службы приложением. Это обеспечивается на основе проведения предварительных переговоров о требуемом уровне(ях) обслуживания и обеспечения доступности соответствующих ресурсов с помощью некоторого механизма резервирования, то есть, по существу, путём динамического создания соглашения об уровне обслуживания.
- 4 Настройка, управление сроком жизни и прекращение действия.** Эти возможности позволяют автоматически сконфигурировать выделенный ресурс применительно к использованию конкретным приложением, управлять ресурсом во время решения задачи и восстанавливать ресурс в исходное состояние для последующего использования.
- 5 Учёт использования ресурсов и аудит.** Эти возможности отслеживают использование разделяемых ресурсов и предоставляют механизмы для передачи пользовательским сообществам сведений о расценках и расходах за использование

ресурсов приложениями и пользователями.

Кроме того, грид-инфраструктура должна быть структурирована так, чтобы интерфейсы, посредством которых она обеспечивает эти возможности, описывались в терминах, эквивалентных для разных классов компонент абстракций. Например, клиент должен иметь возможность использовать одни и те же операции для авторизации или для проведения переговоров о качестве обслуживания, независимо от того, имеет ли он дело с доступом к системе хранения, сети или вычислительному ресурсу. Без такого единообразия для диспетчеров рабочих нагрузок и других систем управления становится затруднительным эффективно и автоматически формировать комплекты ресурсов для использования приложениями.

Эти рассуждения позволяют сделать вывод, что определение эффективной грид-инфраструктуры является весьма сложной задачей. Многие из названных стандартов и систем программного обеспечения, необходимых для реализации этой цели, впрочем, уже у нас есть.

ВИРТУАЛИЗАЦИЯ

Функции отображения ресурсов и управления, задаваемые грид-инфраструктурой, обеспечивают удобные и мощные абстракции для нижележащих физических ресурсов. Тем не менее, реализация этих абстракций в результативной, эффективной и единообразной форме может оказаться трудным делом, если ресурсы не обеспечивают соответствующих функций изоляции и управления. Действительно, недостаток таких функций в товарных ресурсах явился значительным препятствием к успешной реализации ИТ в форме службы.

В связи с этим важным шагом вперёд представляется более широкое применение технологий *виртуализации*. Такие технологии дают возможность реализовать уровень ресурсов таким образом, что одновременно обеспечивается как гибкое управление абстракцией физического ресурса (например, относительно производительности), так и поддержка множества виртуальных экземпляров на одном и том же физическом ресурсе с гарантией надежной изоляции экземпляров. Хотя такого рода технологии виртуализации уже длительное время применяются на мэйнфреймах и высокоуровневых серверных платформах, они только сейчас становятся широко доступными на товарном аппаратном оборудовании и в операционных системах, которые всё более осваиваются организациями.

Применение технологий виртуализации, например, таких, которые обеспечиваются открытым кодом Xen и патентованными продуктами VMware, Microsoft Virtual Server и Virtual Iron, позволяют по требованию создавать для грид-инфраструктуры такую виртуальную исполнительную среду, в которой образ операционной системы, распределение ресурсов и характеристики средств изоляции экземпляров соответствуют заданным спецификациям [6]. Виртуальные машинно-ориентированные системы такие, как J2EE и .NET могут исполнять подобную роль, хотя, поскольку они базируются на полностью новой абстракции ресурсов, они могут применяться только для приложений, написанных с использованием этой абстракции. Коммуникационные технологии виртуализации, например, VLANs, VPNs и средства виртуализации памяти (администраторы логических томов и т.д.) обеспечивают аналогичные функции для

коммуникаций и хранилищ данных соответственно.

Для того, чтобы быть действительно полезными внутри горизонтально интегрированной инфраструктуры, технологии виртуализации должны предоставлять интерфейсы для их изоляции, управляющие функции на соответствующих уровнях и использовать общие абстракции. Сегодня не все известные решения виртуализации удовлетворяют этим требованиям. Например, виртуализация памяти, бесспорно, является той технологической областью, в которой достигнут значительный прогресс в плане формирования стандартов виртуализации, особенно это касается нижних уровней стека. Тем не менее, многие производители памяти всё ещё применяют патентованные, вертикально интегрированные стеки управления памятью, и поэтому до сих пор появляются паллиативные решения, использующие как silo-модули различных производителей, так и интерфейсы управления памятью, отличающиеся от интерфейсов управления другими видами ресурсов.

Кроме того, функции управления распределением ресурсов должны проявляться на уровне грид-инфраструктуры так, чтобы диспетчеры рабочей нагрузки могли насквозь управлять качеством обслуживания на всей совокупности ресурсов. При этом возможности менеджера виртуализации не должны ограничиваться операциями локального перераспределения ресурсов, так как это делается многими средствами виртуализации в настоящее время.

РОЛЬ СТАНДАРТОВ И ОТКРЫТЫХ ИСТОЧНИКОВ

Эффективная грид-инфраструктура должна предоставить управление возможностями разнообразных ресурсов некоторым унифицированным способом, и, если мы хотим избежать блокировок производителей, то она должна это делать так, чтобы не оказаться связанной ни с какой частной, патентованной технологией.

Как это было в случае с Интернет и Web, и стандарты, и источники свободно распространяемого программного обеспечения играют важную роль в достижении этих целей. Стандарты обеспечивают возможность интероперабельности между продуктами различных производителей, в то время как программное обеспечение из открытых источников позволяет предприятиям приступить к использованию той или иной возможности *уже сейчас*, прежде, чем все стандарты станут доступны. В случае Интернет и Web стандарты были разработаны в течение нескольких лет внутри таких известных организаций, как IETF и W3C; одновременно со стандартами широкому освоению Интернет и Web способствовали свободно распространяемые системы программного обеспечения, такие, как BSD Unix и Apache Web сервер. Всё это стимулировало взрыв инноваций вокруг общих стандартов, что, в конечном счёте, привело к потрясающим результатам.

Подобную синергию взаимоотношений между стандартами и свободно распространяемым программным обеспечением мы наблюдаем и в случае грида. Стандартизация интерфейсов для моделирования ресурсов, мониторинга, уведомления, аллокации, настройки и функций учёта использования ресурсов проводится внутри таких организаций, как, например, **Global Grid Forum** (Open Grid Services Architecture), **Organization for Advancement of Structured Information Standards** (WS-Resource Framework, WS-Notification и WS Distributed Management) и **DMTF** (Common Information

Model). Одновременно с этими усилиями свободно распространяемое программное обеспечение реализует не только завершённые стандарты, но также и другие интерфейсы, необходимые для построения приемлемых систем. Доступность такого программного обеспечения позволяет предприятиям и независимым производителям программного обеспечения (independent software vendor, ISV) реализовывать грид-решения уже сейчас и ускоряет определение и освоение стандартов, уменьшая барьеры, препятствующие их принятию.

Необходимо отметить, что на фундаментальном уровне стандарты базовых Web-служб и программное обеспечение достаточно последовательно поддерживаются такими спецификациями, как WSDL, SOAP и WS-Security, которых придерживаются многочисленные ИТ-производители, а также свободно распространяемым программным обеспечением от Apache и других источников. Профили программ, определённые организацией WS-Interoperability, стимулируют интероперабельность среди инструментария и служб, предоставляемых различными производителями.

На более высоких уровнях высококачественное свободно распространяемое программное обеспечение доступно и играет важную роль в эволюции и принятии стандартов. Например, инструментальный пакет Globus Toolkit, в разработке которого принимали участие наши организации, реализует широкий набор функциональностей, в первую очередь, на уровне грид-инфраструктуры, касающихся безопасности, управления выполнением, управления данными, мониторинга и обнаружения, а также механизмов базовых Web-служб, рассмотренных выше. Он был первоначально разработан и применялся в научно-исследовательской и образовательной среде, а учреждение фирмы Univa Corporation, осуществляющей коммерческую поддержку и обслуживание этого пакета, ускоряет адаптацию этих функций предприятиями.

РЕЗЮМЕ

Мы показали, что термины *SOA*, *грид*, *компьютинг по требованию*, *компьютинг услуг*, *программное обеспечение в форме службы* и другие связанные с ними термины, все представляют различные перспективы одной и той же конечной цели, а именно, реструктуризации комплекса ИТ-предприятия в форме горизонтально интегрированной, ориентированной на службы архитектуры. Если такого рода постановку удастся успешно реализовать, то в комплексе ИТ-предприятия все приложения: *собственные (in-house)*, *сторонних организаций (third-party)* или *арендуемые у внешних источников (outsourced)* будут работать в единообразной исполнительской среде, которая обеспечивает, согласно заданным требованиям, предоставление как собственных, так и арендуемых ресурсов, и, конечно, высокоэффективные возможности безопасности, мониторинга, аудита и управления.

Однако эта чаша Грааля в виде открытого, базирующегося на стандартах, автоматически управляемого программного обеспечения и динамически настраиваемого аппаратного оборудования, пока ещё не у нас в руках. Тем не менее, это не означает, что предприятия не могут уже сегодня начать создавать горизонтально интегрированные, ориентированные на службы архитектуры. Хорошо зарекомендовавшие себя продукты Web-служб позволяют создавать ориентированные на службы приложения. Коммерческие или доступные из открытых источников развитые программные средства виртуализации и управления нагрузкой, а также свободно распространяемое программное обеспечение

гид-инфраструктуры обеспечивают то, что необходимо для создания горизонтально интегрированной инфраструктуры, позволяющей скрыть проблемы согласования указанных приложений. Интеграция требует проявления более значительных усилий потребителей (или их поставщиков услуг), чем хотелось бы, но такая ситуация должна измениться, поскольку независимые производители программного обеспечения уже начинают предлагать свои гид-ориентированные продукты. Тем временем, в результате накопления опыта, получаемого от распространения интероперабельных решений, и постоянного интереса к ним со стороны конечных пользователей ускоряется прогресс формирования будущих стандартов.

БЛАГОДАРНОСТИ

Мы признательны Грегу Астфальку (Greg Astfalk), Стюарту Фельдману (Stuart Feldman) и Вас Василидису (Vas Vasilidis) за замечания, сделанные к первоначальному варианту данной статьи. Работа Яна Фостера частично финансировалась по контракту W-31-109 Eng -38 Отделением математических, информационных и вычислительных наук в рамках подпрограммы Отдела передовых исследований по научному компьютерингу Министерства энергетики США и Национальным научным фондом.

ЛИТЕРАТУРА

1. Foster, I. 2002. What is the grid? A three point check-list; <http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf>
(Русский перевод: gridclub.ru)
2. Foster, I., Kesselman, C., Nick, J.M., and Tuecke, S. Grid services for distributed systems integration. *IEEE Computer* 35 (6):37-46. (Русский перевод ж. Открытые системы) .
3. Kendall, S.C., Waldo, J., Wollrath, A., and Wyant, G. 1994. A note on distributed computing. Sun Microsystems, Technical Report TR-94-29.
4. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. 2003. Web services architecture. W3C, working draft; <http://www.w3.org/TR/2003/WD-ws-arch-20030808/>
5. Foster, I. 2005. A Globus primer; <http://www.globus.org/primer/>
6. Rosenblum, M., and Garfinkel, T. 2005. Virtual machine monitors: Current technology and future trends. *IEEE Computer* (May): 39-47.