ИНТЕГРИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Лекция №10

Архитектурные решения построения ИСУ

Распределенная система

В настоящее время практически все большие программные системы являются распределенными.

Распределенная система - система, в которой обработка информации сосредоточена не на одной вычислительной машине, а распределена между несколькими компьютерами. При проектировании распределенных систем, которое имеет много общего с проектированием ПО в общем, все же следует учитывать некоторые специфические особенности.

Существует шесть основных характеристик распределенных систем.

1. ***Совместное использование ресурсов***. Распределенные системы допускают совместное использование как аппаратных (жестких дисков, принтеров), так и программных (файлов, компиляторов) ресурсов.

2. ***Открытость***. Это возможность расширения системы путем добавления новых ресурсов.

Распределенная система

3. ***Параллельность.*** В распределенных системах несколько процессов могут одновременно выполнятся на разных компьютерах в сети. Эти процессы могут взаимодействовать во время их выполнения.

4. ***Масштабируемость***. Под масштабируемостью понимается возможность добавления новых свойств и методов.

5. ***Отказоустойчивость***. Наличие нескольких компьютеров позволяет дублирование информации и устойчивость к некоторым аппаратным и программным ошибкам. Распределенные системы в случае ошибки могут поддерживать частичную функциональность. Полный сбой в работе системы происходит только при сетевых ошибках.

6. ***Прозрачность***. Пользователям предоставляется полный доступ к ресурсам в системе, в то же время от них скрыта информация о распределении ресурсов по системе.

Распределенная система

Распределенные системы обладают и рядом недостатков.

1. ***Сложность***. Намного труднее понять и оценить свойства распределенных систем в целом, их сложнее проектировать, тестировать и обслуживать. Также производительность системы зависит от скорости работы сети, а не отдельных процессоров. Перераспределение ресурсов может существенно изменить скорость работы системы.
2. ***Безопасность***. Обычно доступ к системе можно получить с нескольких разных машин, сообщения в сети могут просматриваться и перехватываться. Поэтому в распределенной системе намного труднее поддерживать безопасность.
3. ***Управляемость***. Система может состоять из разнотипных компьютеров, на которых могут быть установлены различные версии операционных систем. Ошибки на одной машине могут распространиться непредсказуемым образом на другие машины.

Распределенная система

4. ***Непредсказуемость*.** Реакция распределенных систем на некоторые события непредсказуема и зависит от полной загрузки системы, ее организации и сетевой нагрузки. Так как эти параметры могут постоянно изменятся, поэтому время ответа на запрос может существенно отличаться от времени.

Из этих недостатков можно увидеть, что при проектировании распределенных систем возникает ряд проблем, которые надо учитывать разработчикам.

1. ***Идентификация ресурсов***. Ресурсы в распределенных системах располагаются на разных компьютерах, поэтому систему имен ресурсов следует продумать так, чтобы пользователи могли без труда открывать необходимые им ресурсы и ссылаться на них. Примером может служить система URL(унифицированный указатель ресурсов), которая определяет имена Web-страниц.

Распределенная система

2. ***Коммуникация***. Универсальная работоспособность Internet и эффективная реализация протоколов TCP/IP в Internet для большинства распределенных систем служат примером наиболее эффективного способа организации взаимодействия между компьютерами. Однако в некоторых случаях, когда требуется особая производительность или надежность, возможно использование специализированных средств.

3. ***Качество системного сервиса***. Этот параметр отражает производительность, работоспособность и надежность. На качество сервиса влияет ряд факторов: распределение процессов, ресурсов, аппаратные средства и возможности адаптации системы.

4. ***Архитектура программного обеспечения***. Архитектура ПО описывает распределение системных функций по компонентам системы, а также распределение этих компонентов по процессорам. Если необходимо поддерживать высокое качество системного сервиса, выбор правильной архитектуры является решающим фактором.

Распределенная система

Задача разработчиков распределенных систем - спроектировать программное и аппаратное обеспечение так, чтобы предоставить все необходимые характеристики распределенной системы. А для этого требуется знать преимущества и недостатки различных архитектур распределенных систем. Выделяется три типа архитектур распределенных систем.

Архитектура клиент/сервер. В этой модели систему можно представить как набор сервисов, предоставляемых серверами клиентам. В таких системах серверы и клиенты значительно отличаются друг от друга.

Трехзвенная архитектура. В этой модели сервер предоставляет клиентам сервисы не напрямую, а посредством сервера бизнес-логики.

Архитектура распределенных объектов. В этом случае между серверами и клиентами нет различий и систему можно представить как набор взаимодействующих объектов, местоположение которых не имеет особого значения. Между поставщиком сервисов и их пользователями не существует различий.

Распределенная система

Про первые две модели было сказано уже не раз, остановимся подробнее на третьей.

Эта архитектура широко применяется в настоящее время и носит также название *архитектуры веб-сервисов.*

Веб-сервис - это приложение, доступное через Internet и предоставляющее некоторые услуги, форма которых не зависит от поставщика (так как используется универсальный формат данных - XML) и платформы функционирования. В данное время существует три различные технологии, поддерживающие концепцию распределенных объектных систем. Это технологии EJB, CORBA и DCOM.

Веб-сервисы

Для начала несколько слов о том, что такое XML вообще. XML - универсальный формат данных, который используется для предоставления Web-сервисов. В основе Web-сервисов лежат открытые стандарты и протоколы: SOAP, UDDI и WSDL.

SOAP (Simple Object Access Protocol), разработанный консорциумом W3C, определяет формат запросов к Web-сервисам. Сообщения между Web-сервисом и его пользователем пакуются в так называемые SOAP-конверты (SOAP envelopes, иногда их ещё называют XML-конвертами). Само сообщение может содержать либо запрос на осуществление какого-либо действия, либо ответ - результат выполнения этого действия.

WSDL (Web Service Description Language). Интерфейс Web-сервиса описывается в WSDL-документах (а WSDL - это подмножество XML). Перед развертыванием службы разработчик составляет ее описание на языке WSDL, указывает адрес Web-сервиса, поддерживаемые протоколы, перечень допустимых операций, форматы запросов и ответов.

Веб-сервисы

UDDI (Universal Description, Discovery and Integration) - протокол поиска Web-сервисов в Internet. Представляет собой бизнес-реестр, в котором провайдеры Web-сервисов регистрируют службы, а разработчики находят необходимые сервисы для включения в свои приложения.

EJB

Основная идея, лежавшая в разработке технологии Enterprise JavaBeans -- создать такую инфраструктуру для компонент, чтобы они могли бы легко ``вставляться'' (``plug in'') и удаляться из серверов, тем самым увеличивая или снижая функциональность сервера. Технология Enterprise JavaBeans похожа на технологию JavaBeans в том смысле, что она использует ту же самую идею (а именно, создание новой компоненты из уже существующих, готовых и настраиваемых компонент, аналогично RAD-системам), но во всем остальном Enterprise JavaBeans -- совершенно иная технология.

Веб-сервисы

Опубликованная в марте 1998 года EJB-спецификация версии 1.0 (Недавно была опубликована версия 1.1 спецификации) определяет следующие цели:

1. Облегчить разработчикам создание приложений, избавив их от необходимости реализовывать с нуля такие сервисы, как транзакции (transactions), нити (threads), загрузка (load balancing) и другие. Разработчики могут сконцентрироваться на описании логики своих приложений, оставляя заботы о хранении, передаче и безопасности данных на EJB-систему. При этом все равно имеется возможность самому контролировать и описывать порученные системе процессы.

2. Описать основные структуры EJB-системы, описав при этом интерфейсы взаимодействия (contracts) между ее компонентами.

Веб-сервисы

3. EJB преследует цель стать стандартом для разработки клиент/сервер приложений на Java. Таким же образом, как исходные JavaBeans (Delphi, или другие) компоненты от различных производителей можно было составлять вместе с помощью соответствующих RAD-систем, получая в результате работоспособные клиенты, таким же образом серверные компоненты EJB от различных производителей также могут быть использованы вместе. EJB-компоненты, будучи Java-классами, должны без сомнения работать на любом EJB-совместимом сервере даже без перекомпиляции, что практически нереально для других систем.

4. EJB совместима с Java API, может взаимодействовать с другими (не обязательно Java) приложениями, а также совместима с CORBA.

Веб-сервисы

Разработчику, однако, не нужно самому реализовывать EJB-объект. Этот класс создается специальным кодогенератором, поставляемым вместе в EJB-контейнером. Как уже было сказано, EJB-объект (созданный с помощью сервисов контейнера) и EJB-компонента (созданная разработчиком), реализуют один и тот же интерфейс. В результате, когда приложение-клиент хочет вызвать метод у EJB-компоненты, то сначала вызывается аналогичный (по имени) метод у EJB-объекта, что находится на стороне клиента, а тот, в свою очередь, связывается с удаленной EJB-компонентой и вызывает у нее этот метод (с теми же аргументами).

Существует два различных типа ``бинов''.

Session bean представляет собой EJB-компоненту, связанную с одним клиентом. ``Бины'' этого типа, как правило, имеют ограниченный срок жизни (хотя это и не обязательно), и редко участвуют в транзакциях. В частности, они обычно не восстанавливаются после сбоя сервера. В качестве примера session bean можно взять ``бин'', который живет в Web-сервере и динамически создает HTML-страницы клиенту, при этом следя за тем, какая именно страница загружена у клиента.

Веб-сервисы

Когда же пользователь покидает Web-узел, или по истечении некоторого времени, session bean уничтожается. Несмотря на то, что в процессе своей работы, session bean мог сохранять некоторую информацию в базе данных, его предназначение заключается все-таки не в отображении состояния или в работе с ``вечными объектами'', а просто в выполнении некоторых функций на стороне сервера от имени одного клиента.

Entity bean, наоборот, представляет собой компоненту, работающую с постоянной (persistent) информацией, хранящейся, например, в базе данных. Entity beans ассоциируются с элементами баз данных и могут быть доступны одновременно нескольким пользователям. Так как информация в базе данных является постоянной, то и entity beans живут постоянно, ``выживая'', тем самым, после сбоев сервера (когда сервер восстанавливается после сбоя, он может восстановить ``бин'' из базы данных). Например, entity bean может представлять собой строку какой-нибудь таблицы из базы данных, или даже результат операции SELECT. В объектно-ориентированных базах данных, entity bean может представлять собой отдельный объект, со всеми его атрибутами и связями.

Веб-сервисы

EJB имеет следующие положительные и отрицательные стороны:

Достоинства

- Быстрое и простое создание
- Java-оптимизация
- Кроссплатформенность
- Динамическая загрузка компонент-переходников
- Возможность передачи объектов по значению
- Встроенная безопасность

Недостатки

1. Поддержка только одного языка - Java
2. Трудность интегрирования с существующими приложениями
3. Плохая масштабируемость
4. Производительность
5. Отсутствие международной стандартизации

Веб-сервисы

Благодаря своей легко используемой Java-модели, EJB является самым простым и самым быстрым способом создания распределенных систем. EJB - хороший выбор для создания RAD-компонент и небольших приложений на языке Java. Конечно, EJB не такая мощная технология, как DCOM или CORBA. Тем самым, роль RMI в создании больших, масштабируемых промышленных систем, снижается.