Кондратьев Павел Сергеевич

УлГТУ ФИСТ, ИВТАСмд-11

РАЗВИТИЕ МЕХАНИЗМА УДАЛЕННОГО ВЫЗОВА ПРОЦЕДУР

Появление вычислительных систем с базами данных (БД) привело к смене прежних методов машинного моделирования процессов управления, в которых для каждого приложения определялись и поддерживались собственные наборы данных. В системах, использующих БД, все данные определяются и поддерживаются централизованно. Основной предпосылкой использования БД в автоматизированных системах является стремление объединить все обрабатываемые данные организации или предприятия в единое целое и обеспечить к ним контролируемый доступ.

Однако бурное развитие технологий сетевой связи и обмена данными, вызванное с развитием Интернета, а также появление мобильных беспроводных сетей снова привели к децентрализации обработки данных, но уже не на уровне предприятий, а на уровне ведомств, государств и международных организаций.

**Удалённый вызов процедур** (иногда вызов удалённых процедур; RPC от англ. remote procedure call) – класс технологий, позволяющих программам вызывать функции или процедуры в другом адресном пространстве (на удалённых узлах, либо в независимой сторонней системе на том же узле).

Обычно реализация RPC-технологии включает два компонента: сетевой протокол для обмена в режиме клиент-сервер и язык сериализации объектов (или структур для необъектных RPC).

Различные реализации имеют отличающуюся друг от друга архитектуру и разнятся в возможностях: одни реализуют архитектуру SOA, другие – ORBA или DCOM. На транспортном уровне RPC используют в основном протоколы TCP и UDP, однако, некоторые построены на основе HTTP (что нарушает архитектуру ISO/OSI, так как HTTP – изначально не транспортный протокол).

**Принцип**

Идея вызова удалённых процедур состоит в расширении механизма передачи управления и данных внутри программы, выполняющейся на одном узле, на передачу управления и данных через сеть. Средства удалённого вызова процедур предназначены для облегчения организации распределённых вычислений и создания распределенных клиент-серверных информационных систем. Наибольшая эффективность использования RPC достигается в тех приложениях, в которых существует интерактивная связь между удалёнными компонентами с небольшим временем ответов и относительно малым количеством передаваемых данных. Такие приложения называются RPC-ориентированными.

Характерными чертами вызова удалённых процедур являются:

* асимметричность, то есть одна из взаимодействующих сторон является инициатором;
* синхронность, то есть выполнение вызывающей процедуры приостанавливается с момента выдачи запроса и возобновляется только после возврата из вызываемой процедуры.

Реализация удалённых вызовов существенно сложнее реализации вызовов локальных процедур.

Так как вызывающая и вызываемая процедуры выполняются на разных узлах, то они имеют разные адресные пространства, и это создает проблемы при передаче параметров и результатов, особенно если машины находятся под управлением различных операционных систем или имеют различную архитектуру (например, используется прямой или обратный порядок байтов). Так как RPC не может рассчитывать на разделяемую память, то это означает, что параметры RPC не должны содержать указателей на ячейки нестековой памяти и что значения параметров должны копироваться с одного компьютера на другой. Для копирования параметров процедуры и результата выполнения через сеть выполняется их сериализация (процесс перевода какой-либо структуры данных в последовательность байтов.).

**Идея вызова удалѐнных процедур** (Remote Procedure Call – RPC) состоит в расширении механизма передачи управления и данных внутри программы, выполняющейся на одной машине, на передачу управления и данных через сеть.

Средства удаленного вызова процедур предназначены для:

* облегчения организации распределѐнных вычислений;
* создания распределенных клиент-серверных информационных систем.

Наибольшая эффективность использования RPC достигается в тех приложениях, в которых существует интерактивная связь между удалѐнными компонентами с небольшим временем ответов и относительно малым количеством передаваемых данных. Такие приложения называются RPC-ориентированными.

Проблемы и задачи, которые необходимо решить при реализации RPC:

1. вызывающая и вызываемая процедуры выполняются на разных машинах и имеют разные адресные пространства, что создает проблемы при передаче параметров и результатов, особенно если машины находятся под управлением различных операционных систем или имеют различную архитектуру
2. в отличие от локального вызова удаленный вызов процедур обязательно использует транспортный уровень сетевой архитектуры (например, TCP), однако это остается скрытым от разработчика.
3. в реализации RPC участвуют как минимум два процесса – по одному в каждой машине. В случае, если один из них аварийно завершится, могут возникнуть следующие ситуации:

* при аварии вызывающей процедуры удаленно вызванные процедуры станут «осиротевшими»;
* при аварийном завершении удаленных процедур вызывающие процедуры станут «обездоленными родителями», которые будут безрезультатно ожидать ответа от удаленных процедур

1. Неоднородность языков программирования и операционных сред => проблема совместимости, до сих пор не решенная ни с помощью введения одного общепринятого стандарта, ни с помощью реализации нескольких конкурирующих стандартов на всех архитектурах и во всех языках.

**Базовые операции RPC**

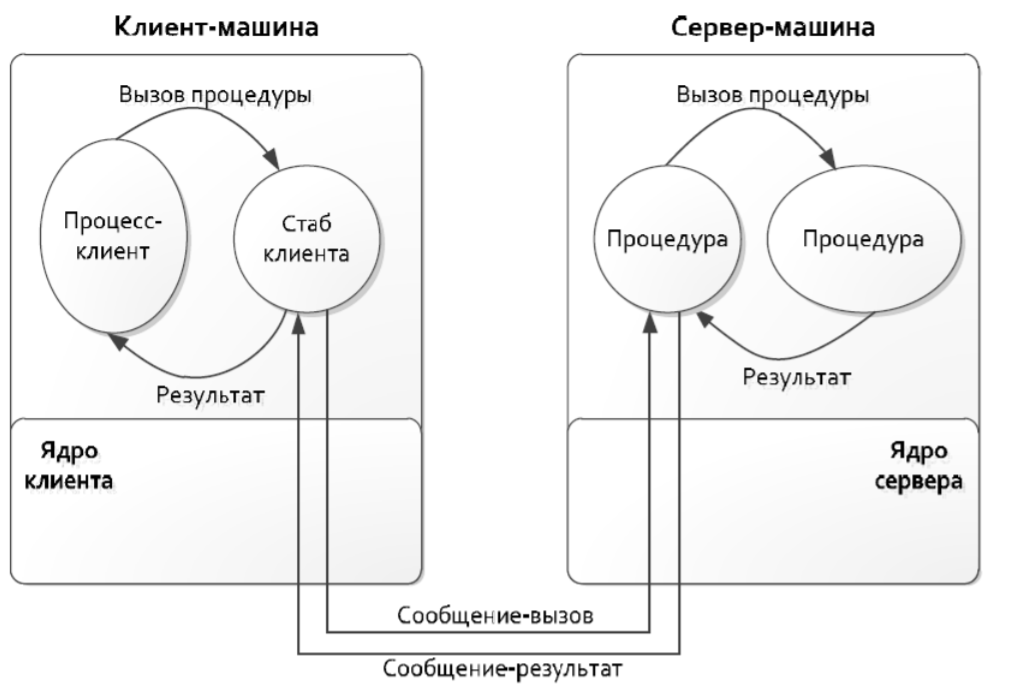
Чтобы понять работу RPC, рассмотрим вначале выполнение вызова локальной процедуры в обычном компьютере, работающем автономно. Чтобы осуществить вызов, вызывающая процедура помещает параметры в стек в обратном порядке. После того, как вызов выполнен, он помещает возвращаемое значение в регистр, перемещает адрес возврата и возвращает управление вызывающей процедуре, которая выбирает параметры из стека, возвращая его в исходное состояние.

**Идея, положенная в основу RPC**, состоит в том, чтобы сделать вызов удаленной процедуры выглядящим по возможности так же, как и вызов локальной процедуры.

Другими словами - вызывающей процедуре не требуется знать, что вызываемая процедура находится на другой машине, и наоборот.

**RPC достигает прозрачности следующим путем.** Когда вызываемая процедура действительно является удаленной, в библиотеку помещается вместо локальной процедуры другая версия процедуры, называемая клиентским стабом (англ. stub - заглушка). Подобно оригинальной процедуре, стаб вызывается с использованием вызывающей последовательности, так же происходит прерывание при обращении к ядру. Только в отличие от оригинальной процедуры он не помещает параметры в регистры и не запрашивает у ядра данные, вместо этого он формирует сообщение для отправки ядру удаленной машины.

**Взаимодействие программных компонентов при выполнении удаленного вызова процедуры**



**Этапы выполнения процедуры RPC**



После того, как клиентский стаб был вызван программой-клиентом, его первой задачей является заполнение буфера отправляемым сообщением. В некоторых системах клиентский стаб имеет единственный буфер фиксированной длины, заполняемый каждый раз с самого начала при поступлении каждого нового запроса. В других системах буфер сообщения представляет собой пул буферов для отдельных полей сообщения, причем некоторые из этих буферов уже заполнены.

Затем производится преобразование параметров в соответствующий формат и их вставка в буфер сообщения. К этому моменту сообщение готово к передаче, поэтому выполняется прерывание по вызову ядра.

Когда ядро получает управление, оно переключает контексты, сохраняет регистры процессора и карту памяти (дескрипторы страниц), устанавливает новую карту памяти, которая будет использоваться для работы в режиме ядра. Поскольку контексты ядра и пользователя различаются, ядро должно скопировать сообщение в свое собственное адресное пространство, запомнить адрес назначения, после чего передать его сетевому интерфейсу.

На этом завершается работа на клиентской стороне.

Включается таймер передачи, и ядро может либо выполнять циклический опрос наличия ответа, либо передать управление планировщику, который выберет какой-либо другой процесс на выполнение.

На стороне сервера поступающие биты помещаются принимающей аппаратурой либо во встроенный буфер, либо в оперативную память. Когда вся информация будет получена, генерируется прерывание.

Обработчик прерывания проверяет правильность данных пакета и определяет, какому стабу следует их передать. Если ни один из стабов не ожидает этот пакет, обработчик должен либо поместить его в буфер, либо вообще отказаться от него. Если имеется ожидающий стаб, то сообщение копируется ему. Наконец, выполняется переключение контекстов, в результате чего восстанавливаются регистры и карта памяти, принимая те значения, которые они имели в момент, когда стаб сделал вызов

Теперь начинает работу серверный стаб. Он распаковывает параметры и помещает их соответствующим образом в стек. По завершении работы, выполняется вызов сервера.

После выполнения процедуры сервер передает результаты клиенту. Для этого выполняются все описанные выше этапы, только в обратном порядке.

**Реализации RPC**

Различные реализации RPC имеют очень отличающуюся друг от друга архитектуру и разнятся в своих возможностях: одни реализуют архитектуру SOA, другие CORBA или DCOM.

Существует множество технологий, обеспечивающих RPC, например,

**JSON-RPC –** JavaScript Object Notation Remote Procedure Calls - протокол, использующий JSON (текстовый формат обмена данными, основанный на JavaScript) для кодирования сообщений; JSON-RPC определяет несколько типов данных и команд, поддерживает уведомления и множественные вызовы.

**Java RMI** – Java Remote Method Invocation — программный интерфейс вызова удаленных методов в языке Java, представляющий собою распределенную объектную модель, специфицирующую, способы вызовов удаленных методов, работающих на другой виртуальной машине Java.

**XML RPC** – XML-вызов удаленных процедур – протокол, использующий XML для кодирования своих сообщений и HTTP в качестве транспортного механизма. Является прародителем SOAP.

**SOAP** – протокол обмена структурированными сообщениями в распределенной вычислительной среде. Первоначально SOAP предназначался в основном для реализации RPC, но сейчас используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

**Организация связи с использованием удаленных объектов**

Объектно-ориентированная технология в настоящее время широко применяется при разработке приложений, в том числе и распределенных. Одним из наиболее важных свойств объекта является то, что он скрывает особенности реализации, предоставляя для взаимодействия строго описанный интерфейс. Это позволяет заменять или изменять объекты, оставляя интерфейс неизменным.

Развитие клиент-серверной архитектуры в начале 1990-х годов привело к формированию объектно-ориентированной концепции распределенных систем, ориентированной на инкапсуляцию механизма распределенных взаимодействий и уменьшение сложности разработки распределенных приложений посредством методов объектно-ориентированной разработки и удаленных вызовов методов объектов.

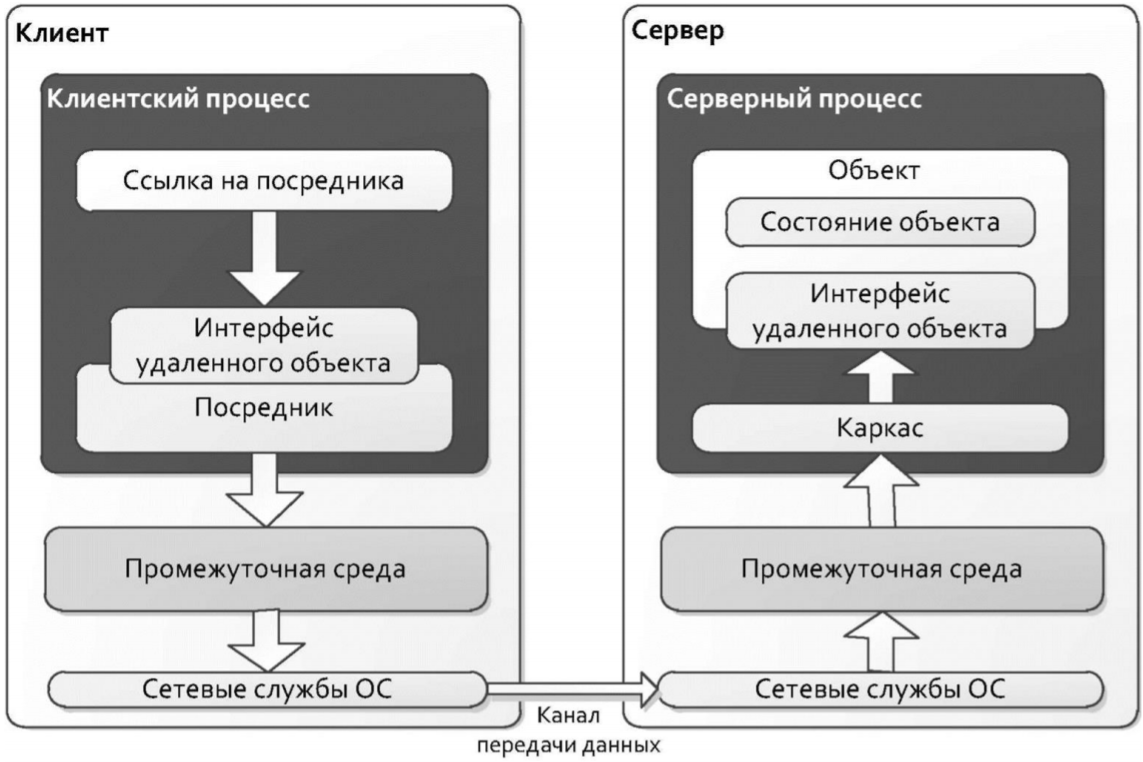
**Удаленный объект** представляет собой некоторые данные, совокупность которых определяет его состояние. Это состояние можно изменять путем вызова его методов. Если возможен прямой доступ к данным удаленного объекта, это происходит посредством неявного удаленного вызова, необходимого для передачи значения поля данных объекта между процессами. Методы и поля объекта, которые могут использоваться через удаленные вызовы, доступны через некоторый внешний интерфейс класса объекта.

Для передачи параметров по сети используется сериализация объектов и данных

**Сериализация** – это перевод состояния объекта в последовательность битов (чаще всего, бинарный или XML-файл), после чего его копия может быть передана в другой процесс.

Обратный процесс - **десериализация** - это восстановление состояния объекта из принятой последовательности битов.

**Использование удаленных объектов**



В момент, когда клиент начинает использовать удаленный объект, на стороне клиента создается клиентская заглушка, называемая посредником (англ. proxy). Посредник реализует тот же интерфейс, что и удаленный объект.

Вызывающий процесс использует методы посредника, который сериализует их параметры для передачи по сети, и передает их по сети серверу. Промежуточная среда на стороне сервера десериализует параметры и передает их заглушке на стороне сервера, которую называют каркасом (skeleton) или, как и в удаленном вызове процедур, заглушкой. Каркас связывается с некоторым экземпляром удаленного объекта. Это может быть, как вновь созданный, так и существующий экземпляр объекта, в зависимости от применяемой модели использования удаленных объектов.

При использовании удаленных объектов возникает ряд проблемных вопросов, связанных с временем их жизни:

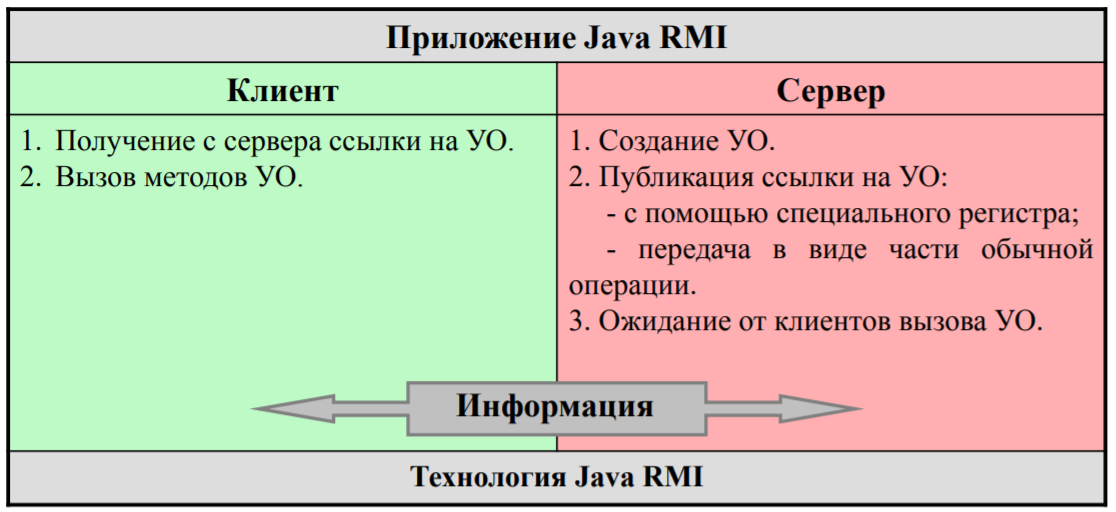
1. в какой момент времени создается экземпляр удаленного объекта;
2. в течение какого промежутка времени он существует.

Для описания жизненного цикла в системах с удаленными объектами используются два дополнительных понятия:

* активация объекта – процесс перевода созданного объекта в состояние обслуживания удаленного вызова, т.е. связывания с каркасом и посредником.
* деактивация объекта – процесс перевода объекта в неиспользуемое состояние.

**Java RMI**

Технология Java RMI (Remote Method Invocation - вызов удаленных методов) позволяет обеспечить прозрачный доступ к методам удаленных объектов (УО), обеспечивая доставку параметров вызываемого метода, сообщение объекту о необходимости выполнения метода и передачу возвращаемого значения клиенту обратно.



**Достоинства:**

* возможность разрабатывать систему целиком основываясь на объектно-ориентированной концепции, не погружаясь в разработку собственных протоколов взаимодействия между распределенными компонентами систем;
* кроссплатформенность, предоставляемая виртуальной машиной Java.

**Недостатки:**

* строгая ограниченность данной технологии платформой Java;
* необходимость обработки соединений между распределенными компонентами приложения => ограничение масштабируемости.