Подходы к проектированию процессов выполняемых компонентами РС

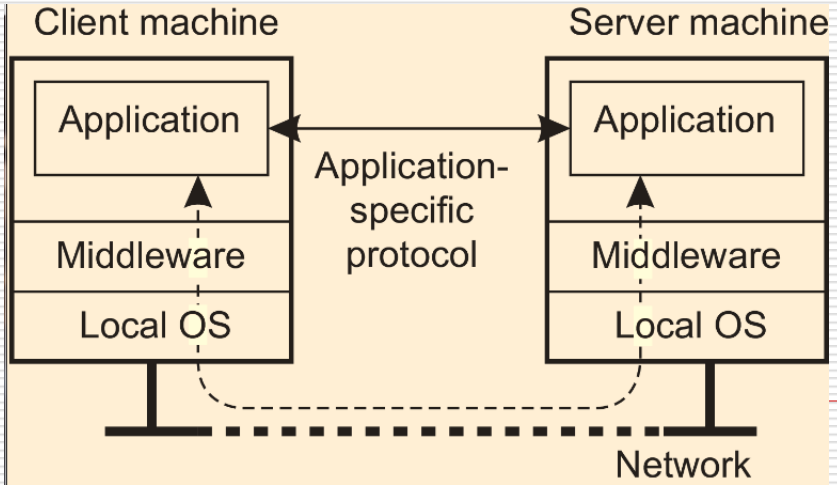
Клиенты PC

Функции клиента PC

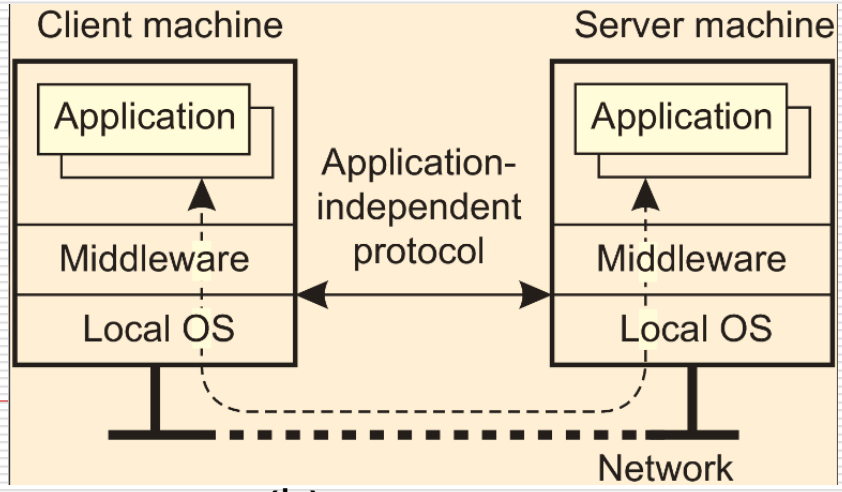
- Основное назначение:
  - Предоставление пользователю возможности взаимодействовать с сервером и получать нужный вид обслуживания.
  - В зависимости от природы (принципов, лежащих в основе) клиент-серверного взаимодействия, имеются различные способы реализации клиентского ПО.

Сетевые пользовательские интерфейсы

- При работе в сети пользователю требуется доступ к ресурсам сервера. Имеется два основных пути достижения этой цели:
  1. Обеспечить для работы с каждым видом ресурсов (видом сервиса) отдельную копию клиентского ПО. Например, файловый клиент, почтовый клиент и т.п. Вариант (a).
  2. Использовать прямой доступ к удаленному сервису (серверу), за счет обеспечения только, подходящего пользовательского интерфейса (терминальный доступ), часто средствами локальной ОС. Это подход получил название **тонкий-клиент**. Внимание к нему не ослабевает, в связи с расширением популярности Интернет, а также использованием мобильных устройств. Вариант (b).



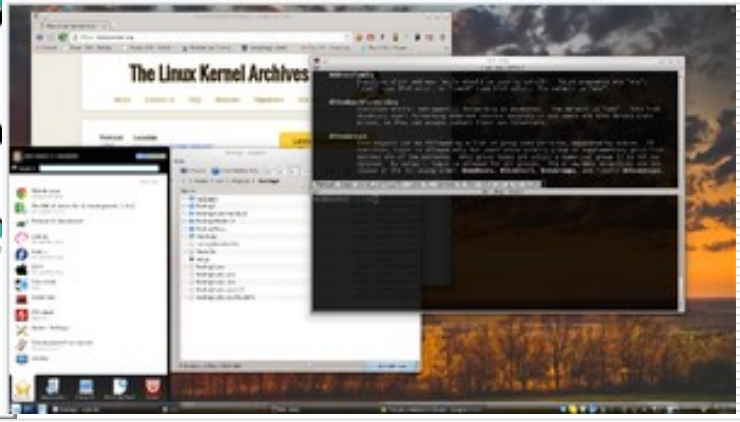
(a)



(b)

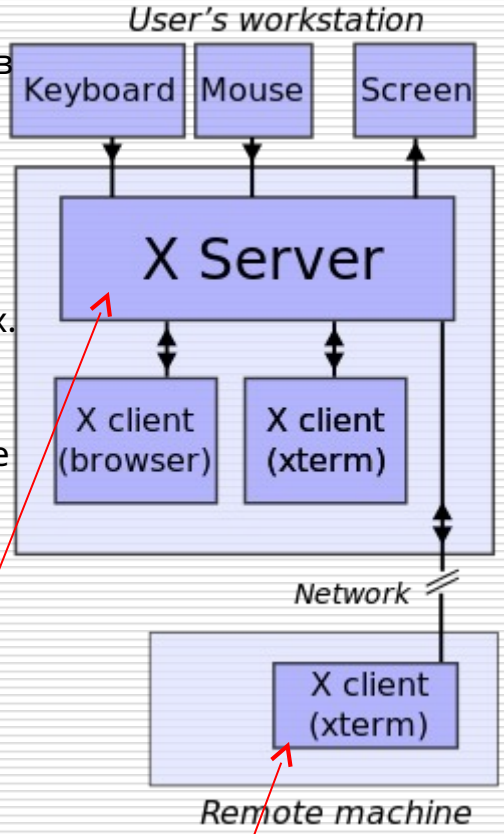
Пример: Система X-Window

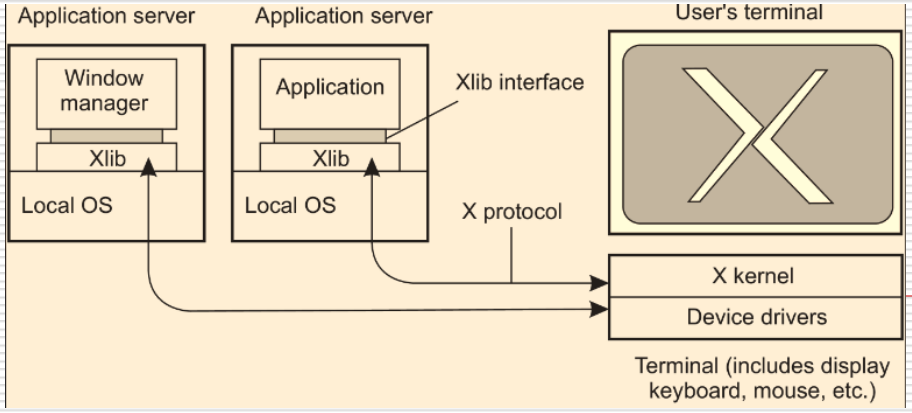
- Одним из старейших, но широко распространенных протоколов удаленного обеспечения пользовательского графического интерфейса является **X Window System** или просто **X**. В мире MS Windows аналогом является Terminal Service с использованием протокола RDP (Remote Desktop Protocol).
- Система была создана в конце 80-х в MIT. С тех пор она развивается без изменения основных принципов, заложенных при ее создании.
- Является основой реализации графического UI в Unix и Unix-подобных ОС.



Организация X Window

- Сердцем X является **X Kernel**. Оно содержит все драйвера устройств относящихся к терминалу (KVM – Keyboard, Video, Mouse). X Kernel обеспечивает низкоуровневый интерфейс управления экраном, а также перехватывает события связанные с работой клавиатуры и мыши. Это интерфейс поддерживается приложением, исполняющемся на сервере, библиотекой **Xlib**.
- **X Kernel** и **X приложение** могут располагаться на разных машинах.
- На сервере одновременно могут исполняться несколько приложений, взаимодействующих с X Kernel клиента. Управляет отображением приложение на X клиенте специальное приложение – Window manager.





- Особенностью X является то, что **X Kernel работает как сервер**, в то время как **X приложение играет роль клиента**.

Тонкий сетевой клиент

- Недостатки X Window:
  - X приложение в процессе своей работы посылает по сети команды, касающиеся управления дисплеем на X клиенте, которые последовательно выполняются X Kernel. Это синхронное поведение компонентов X в глобальных сетях, влечет за собой снижение производительности и увеличение задержек в работе X приложений.
  - В идеале, X приложение должно разделять логику работы приложения (бизнес логику) и функции управления пользовательским интерфейсом, однако, часто это требование не выполняется, что также, влечет за собой замедление работы X клиентов и увеличение задержек в их работе.
  - Имеется несколько решений этих проблем X Window:
    - **Технология NX**: повсеместно известна как “NX”, разработана компанией NoMachine (2003). ;
    - **VNC**: Virtual Network Computing (конец 90-х). система удалённого доступа к рабочему столу компьютера, использующая протокол RFB (англ. *Remote FrameBuffer*, *удалённый кадровый буфер*).;
    - **THINC:** Thin-Client Internet Computing (2007). Архитектура системы тонкого клиента высокопроизводительных вычислений для локального и глобального окружений.

Технология NX

- Основана на исходном протоколе X Window, со следующими улучшениями:
  - Использование сжатия данных при передаче;
  - Реализована передача только изменений, между смежными по времени образами.
  - Использование протокола SSH для обеспечения шифрования соединения.
- Начиная с 2013, с версии 4.0, NX технология стала закрытой.

VNC

- Является альтернативой X Window, в которой приложение полностью управляет отображением на удаленном дисплее, в плоть до пикселя.
- Управление осуществляется путём передачи нажатий клавиш на клавиатуре и движений мыши с одного компьютера на другой и ретрансляции содержимого экрана через компьютерную сеть.
- Система VNC платформонезависима: VNC-клиент, называемый VNC viewer, запущенный на одной операционной системе, может подключаться к VNC-серверу, работающему на любой другой ОС.
- Существуют реализации клиентской и серверной части практически для всех операционных систем, в том числе и для Java (включая мобильную платформу J2ME).
- К одному VNC-серверу одновременно могут подключаться множественные клиенты. Наиболее популярные способы использования VNC — удалённая техническая поддержка и доступ к рабочему компьютеру из дома.

THINC

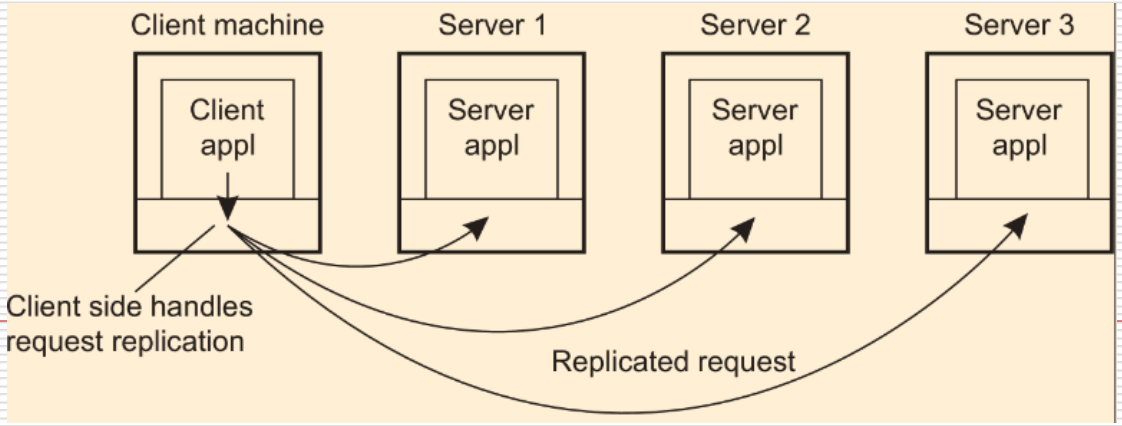
- Недостатком передачи сырых данных пикселей по сравнению с высоуровневыми протоколами такими как X Window является, то что невозможно использование семантики приложения.
- В 2005г. Баррато предложил другой подход, который получил название THINC. Было предложено использовать ограниченное число высокоуровневых команд для управления отображением на дисплее клиента на уровне драйверов устройств.
- Это более эффективно, чем прямое управление пикселями, но мене эффективно, чем обеспечивает X Window.

Клиентское ПО прозрачного доступа

- Во многих случаях требуется обеспечить прозрачное выполнение обработки данных на клиенте и передачу результатов на сторону сервера, например передача штрих кодов, оплаченных наличными сумм и т.п. В этих случаях пользовательский интерфейс составляет небольшую часть клиентского ПО.
- Прозрачный доступ обычно реализуется с помощью генерации клиентских заглушек на основе описаний интерфейса предоставляемого сервером.
- Заглушка предоставляет такой же интерфейс, что и сервер, но она скрывает возможную разницу в архитектуре конкретных серверов, а также разницу в способах коммуникации с серверами.
- В идеале клиент не должен знать где расположен сервер и даже тот факт, что он взаимодействует с удаленным сервером.
- Имеются различные способы обеспечения прозрачности местоположения, миграции и перемещения сервера. Удобная схема именования является ключевым элементом прозрачности.

Прозрачная транзакция с помощью решений размещаемых на клиенте

- Часто прозрачность репликации реализуется с помощью ПО размещаемом на стороне клиента.
- Например, представим систему в которой обеспечивается репликация данных между серверами.
- Такая репликация легко могла бы быть выполнена путем форвардирования запросов на очередную репликацию на серверы.
- Клиент собирает все ответы серверов на запросы репликации и предает их приложению.
- Обработку всех исключительных ситуаций, касающихся возникающих в работе сервера, выполняет промежуточное ПО клиента (прозрачность к сбоям).



Сервера PC

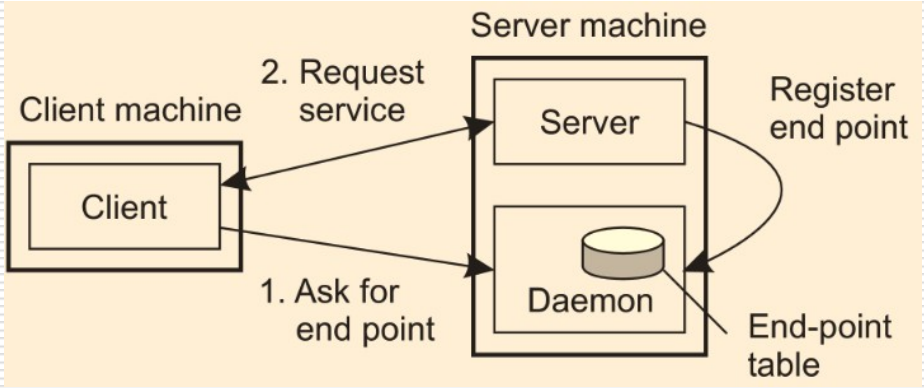
Общие проблемы создания серверов РС

- Сервер реализуется средствами исполнения процессов обеспечивающих реализацию сервиса, потребителями которого являются клиенты.
- Имеется несколько вариантов построения сервера:
  - Параллельная или интеративная обработка запросов.
  - Реализация сервера с отслеживание состояние или без.

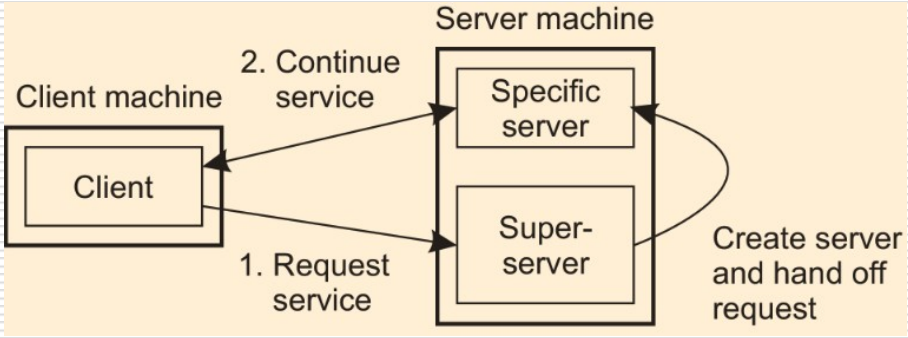
Параллельный сервер против итерационного

- В случае использования параллельной работы сервера, поступающий запрос перехватывается соответствующим процессом приема запросов.
- Для обработки запросов и формирования ответных сообщений используются отдельные потоки, запускаемые при получении очередного запроса.
- После обработки запроса и отправки ответа, поток обработчик переходит в ожидание или закрывается.
- Этот метод реализован в большинстве UNIX систем.
- При интерактивной работе сервера серверный процесс сам реализует обработку запроса, а также отправку ответа клиенту.

Конечная точка подключения к серверу



(a)



(b)

- Конечная точка подключения – это URL, при обращении к которому выполняется подключение к серверу.
- Сетевая адресация, номера портов служб, а также номера протоколов на уровне сетевого протокола назначаются IANA.
- Возможны следующие варианты реализации связи между поступившим запросом и процессом выполняющим обработку запросов.
  - В ОС Unix:
    - Модель одиночных серверов обработки – демонов.
    - Модель использования суперсервера inetd (xinetd).
  - В мире Windows:
    - Сервис зарегистрированный в системе;
    - Приложение запущенное как сервис;
    - Исполняемая программа, запущенная в системе.

Прерываемый сервер

- При создании сервера следует принимать во внимание когда и как работа сервера может быть прервана. Существует несколько способов сделать это:
  - Один из подходов, кстати единственный, надежно работающий в современном Интернете (а иногда и единственно возможный), — пользователь немедленно закрывает клиентское приложение (что автоматически вызывает разрыв соединения с сервером), тут же перезапускает его и продолжает работу. Сервер, естественно, разрывает старое соединение, полагая, что клиент прервал работу.
  - Более правильный способ - разрабатывать клиент и сервер так, чтобы они могли пересылать друг другу сигнал *конца связи {out-ofband),* который должен обрабатываться сервером раньше всех прочих передаваемых клиентом данных. Здесь также возможны разные варианты:
    - Потребовать от сервера просматривать отдельную управляющую конечную точку, на которую клиент будет отправлять сигнал конца связи и одновременно (с более низким приоритетом) — конечную точку, через которую передаются нормальные данные (Пример FTP сервис).
    - Другой вариант — пересылать сигнал конца связи через то же соединение, через которое клиент пересылал свой запрос. В TCP, например, можно посылать срочные данные. Когда срочные данные достигают сервера, он прерывает свою работу (в UNIX- системах — по сигналу), после чего может просмотреть эти данные и обработать их.

Сервер с фиксацией состояния

- Сервер с фиксацией состояния (stateful server) хранит и обрабатывает информацию о своих клиентах.
- Типичным примером такого сервера является файловый сервер, позволяющий клиенту создавать локальные копии файлов, скажем, для повышения производительности операций обновления.
- Подобный сервер поддерживает таблицу, содержащую записи пар (клиент, файл). Такая таблица позволяет серверу отслеживать, какой клиент с каким файлом работает и, таким образом, всегда определять самую «свежую» версию файла. Подобный подход повышает производительность операций чтения-записи, осуществляемых на клиенте.

Достоинства и недостатки сервера с фиксацией состояния

- Достоинства:
  - Рост производительности по сравнению с серверами без фиксации состояния, что является основной причиной разработки серверов с фиксацией состояния.
- Недостатки:
  - В случае сбоя сервера он вынужден восстанавливать свою таблицу с записями пар (клиент, файл), в противном случае не будет никакой гарантии в том, что работа происходит с последней обновленной версией файла. Как правило, серверы с фиксацией состояния нуждаются в восстановлении своего состояния в том виде, в котором оно было до сбоя.

Сервер без состояния

- Сервер без фиксации состояния (stateless sewer) не сохраняет информацию о состоянии своих клиентов и может менять свое собственное состояние, не информируя об этом своих клиентов
- Web-сервер, например, это сервер без фиксации состояния. Он просто отвечает на входящие HTTP-запросы, которые могут требовать загрузки файла как на сервер, так и (гораздо чаще) с сервера. После выполнения запроса web-сервер забывает о клиенте. Кроме того, набор файлов, которыми управляет web-сервер (возможно, в комбинации с файловым сервером), может быть изменен без уведомления клиентов об этом действии.
- Достоинства:
  - В случае архитектуры без фиксации состояния вообще нет необходимости принимать какие-то специальные меры по восстановлению серверов после сбоя. Они просто перезапускаются и работают, ожидая запросов клиента.
- Недостатки:
  - Отсутствие возможности отслеживания состояния сессии клиента.

Серверы объектов

- Сервер объектов (object server) — это сервер, ориентированный на поддержку распределенных объектов.
- Важная разница между стандартным сервером объектов и другими (более традиционными) серверами состоит в том, что сам по себе сервер объектов не предоставляет конкретной службы.
- Конкретные службы реализуются объектами, расположенными на сервере. Сервер предоставляет только средства обращения к локальным объектам на основе запросов от удаленных клиентов. Таким образом, можно относительно легко изменить набор служб, просто добавляя или удаляя объекты.

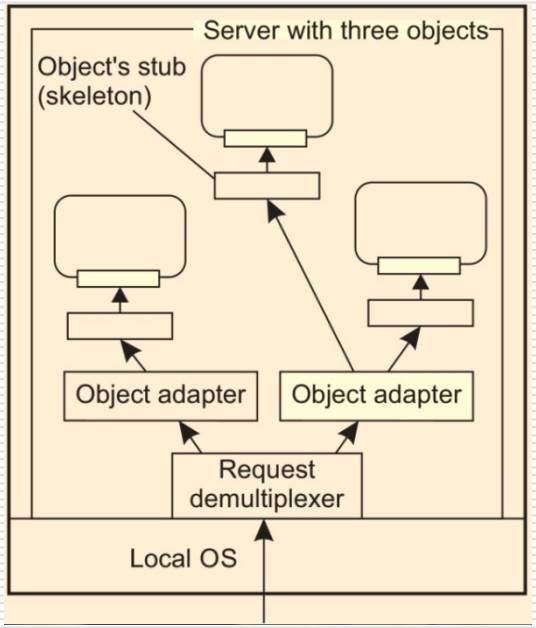
Обращение к объектам

- Объект состоит из двух частей:
  - данных, отражающих его состояние,
  - и кода, образующего реализацию его методов.
- Будут ли эти части храниться раздельно, а также смогут ли методы совместно использоваться несколькими объектами, зависит от сервера объектов.
- Кроме того, существует разница и в способе обращения сервера объектов к его объектам. Например, в многопоточном сервере объектов отдельный поток выполнения может быть назначен каждому объекту или каждому запросу на обращение к объекту.

Способы обращению к объектам. Политика активизации

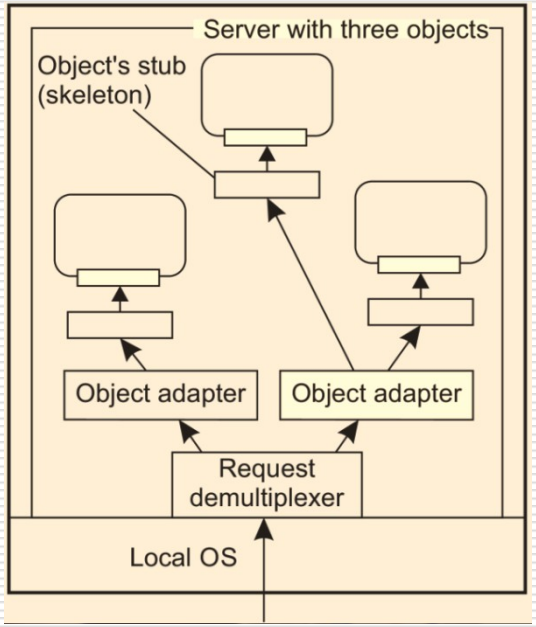
- Для любого объекта, к которому происходит обращение, сервер объектов должен знать:
  - какой код выполнять,
  - с какими данными работать,
  - запускать ли отдельный поток выполнения для поддержки обращения
  - и т. д.
- Возможны следующие подходы к реализации обращений к объектам:
  - Использовать единые правила при обращении ко всем объектам;
  - Использовать различные правила обработки объектов.
- Правила обращения к объекту обычно называют политикой активизации (activation policies), так как во многих случаях объект, перед тем как к нему можно будет обратиться, должен быть перемещен в адресное пространство сервера (то есть активизирован).

Адаптер объектов



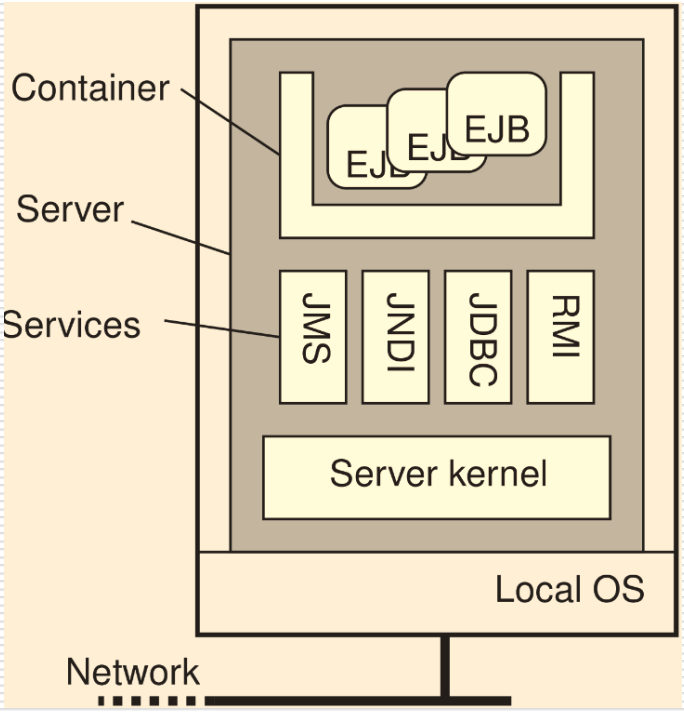
- Нужен механизм группирования объектов в соответствии с политикой активизации каждого из них. Этот механизм называют *адаптером объектов (object adapter), или упаковщиком объектов (object wrapper),* но чаще всего его существование скрыто в наборе средств построения сервера объектов.
- Адаптер объектов контролирует один или несколько объектов. Поскольку сервер должен одновременно поддерживать объекты с различной политикой активизации, на одном сервере может одновременно работать несколько адаптеров объектов.
- При получении сервером запроса с обращением к объекту этот запрос сначала передается соответствующему адаптеру объектов.
- Вместо прямой передачи запроса объекту адаптер передает запрос серверной заглушке этого объекта.
- Заглушка, называемая еще скелетоном и обычно генерируемая из определения интерфейса объекта, выполняет демаршалинг запроса (получает параметры запроса) и обращается к соответствующему методу.

Реализация адаптера



- Реализация адаптера не зависит от объектов, обращения к которым он обрабатывает. Соответственно, можно создать обобщенный адаптер и поместить его на промежуточный уровень программного обеспечения. После этого разработчикам серверов объектов можно сконцентрироваться исключительно на разработке объектов, просто указывая при необходимости, какой адаптер обращения к каким объектам должен контролировать.
- Хотя на рисунке показан специальный компонент, который занимается распределением поступивших запросов соответствующим адаптерам (демультиплексор), он не обязателен. Для этой цели мы вполне могли бы использовать адаптер объектов.

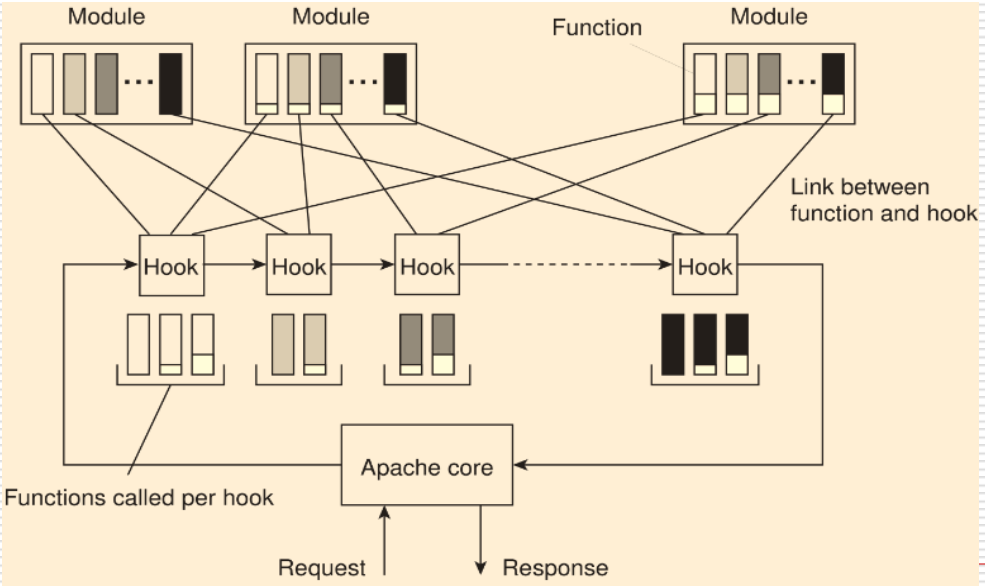
Пример: EJB



- Java Enterprise Beans по сути является объектом, который располагается на сервере и предоставляет для удаленных клиентов различные способы вызова этого объекта.
- Ключевым свойством сервера является, то что он поддерживает для ряда приложений различные виды функций.
- EJB размещается в контейнере, который предоставляет интерфейсы для нижележащих сервисов, которые реализуются сервером приложений.
- Такими сервисами являются:
  - RMI – удаленный вызов процедур;
  - JDBC – доступ к б/д;
  - JNDI – сервис именования;
  - JMS – сервис обмена сообщениями.
- Имеется возможность создания JEB 4-х видов:
  - Stateless session beans - Объект без отслеживания состояния.
  - Stateful session beans – Объект с отслеживанием состояния.
  - Entity beans – Объект сущность;
  - Message-driven beans – Объект управляемый сообщениями.

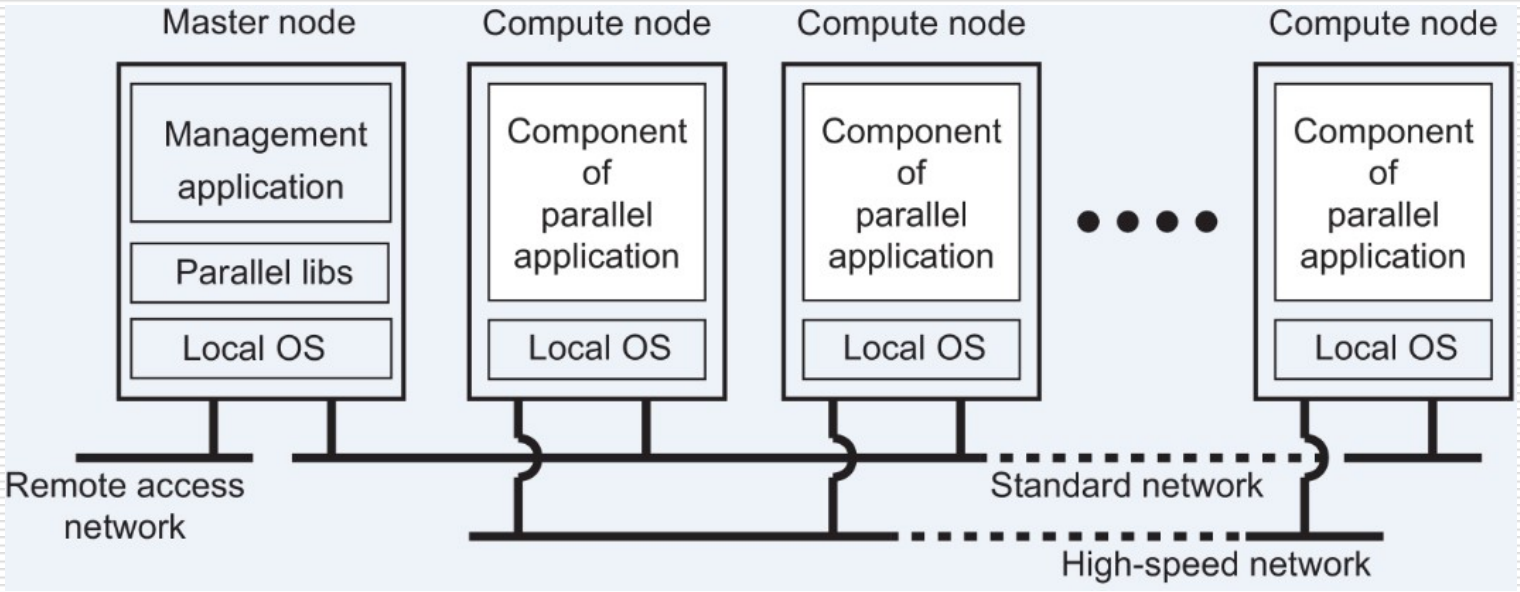
Пример: Веб-сервер Apache

- Сервер Apache – является примером того как могут быть сбалансированы между собой политики активизации и механизмы исполнения.
- Он может рассматриваться как сервер полностью приспособленный под обработку входящих запросов на предоставление Web документов.
- В основе организации работы сервера лежит понятие «ловушки» (hook).



- Сервер Apache исполняется в среде APR - Apache Portable Runtime, которая предоставляет платформо-независимый интерфейс для:
  - Управления файлами;
  - Сетевой работы;
  - Блокировки;
  - Использования потоков и т.п.
- Каждая «ловушка» представляет собой перехватчик группы однотипных действий выполняемых при обработке соотвествующих запросов.
- Функции связанные с той или иной заглушками реализуются с помощью различных модулей
- Имеется несколько десятков модулей Apache каждый из которых предназначен для выполнения определенных функций.

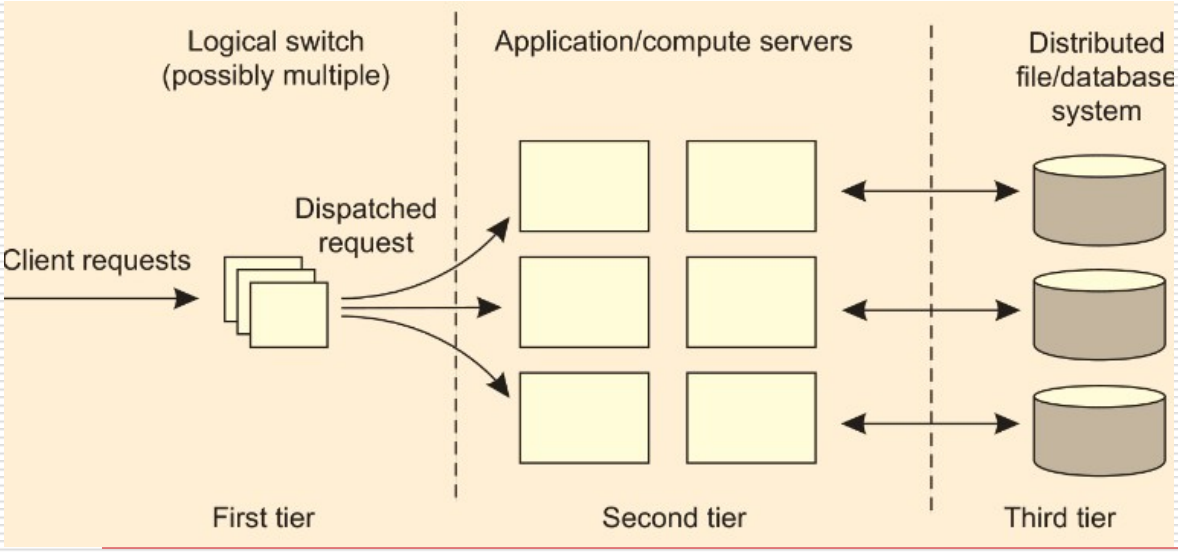
Кластера серверов



- В вычислительных кластерах один узел играет роль управляющего. Остальные узлы являются вычислительными.
- Управляющий кластер исполняет ПО промежуточного слоя, обеспечивающее управление кластером и выполнение вычислительными узлами параллельных фрагментов срвместно решаемой задачи.
- Вычислительные кластера могут быть как локальными так и глобальными.

Локальные кластера серверов. Общая организация.

- Кластер серверов представляет собой распределенную систему имеющую логическую в 3-х уровневую структуру:
  - 1-й уровень – логический коммутатор, распределяющий запросы клиентов между серверами кластера;
  - 2-й уровень – сервера приложений/вычислений, обрабатывающие запросы клиентов;
  - 3-й уровень – сервера обработки данных.



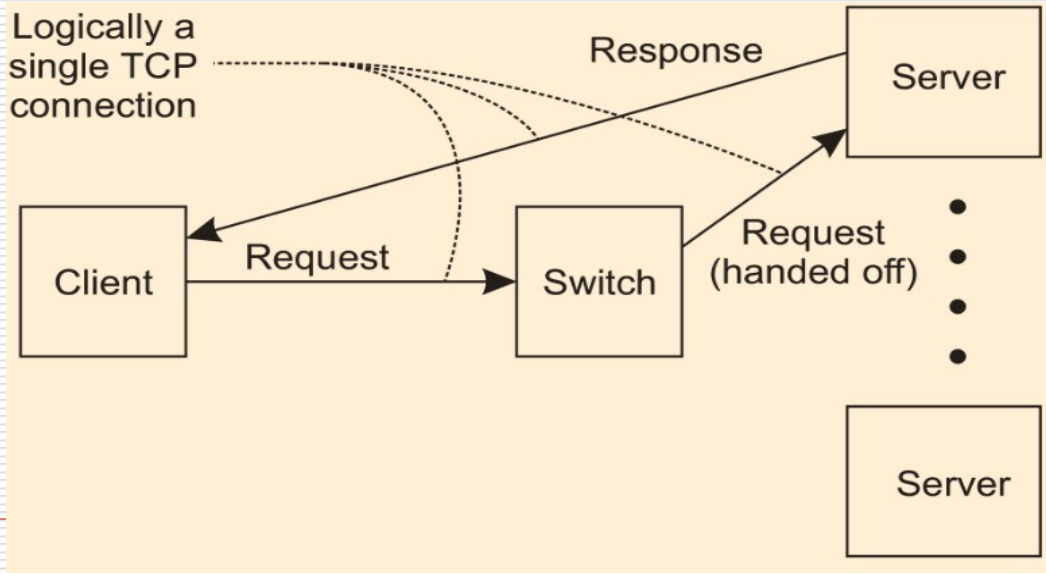
Когда кластер предлагает выполнение нескольких сервисов, они могут быть распределены на различных машинах кластера.

Как результат, некоторые машины могут находиться в режиме ожидания запросов, в то время как другие могут испытывать перегрузки.

Решением может быть – объединение машин в кластер высокой готовности и использование виртуальных машин.

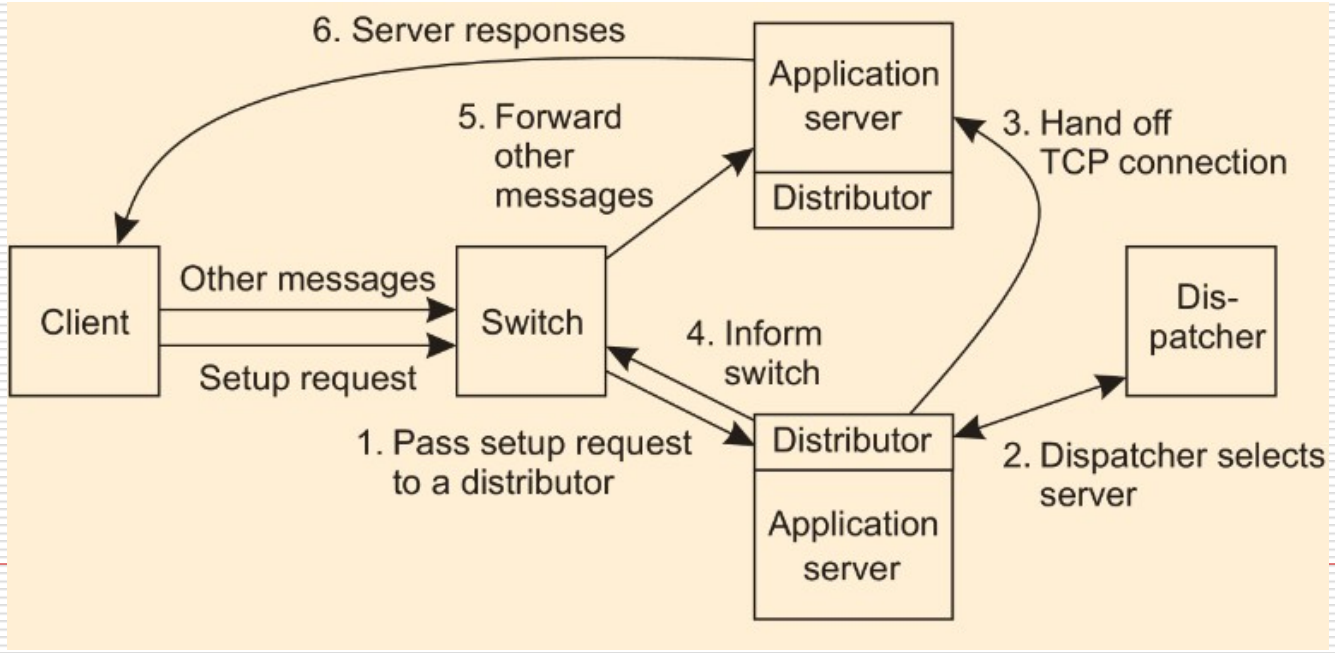
Диспетчирование запросов в локальных кластерах серверов

- Стандартным способом получения доступа к кластеру установление TCP соединения в рамках установления сессии на прикладном уровне. За установление такого соединения отвечает коммутатор входных TCP соединений между пользователями и серверами кластера (frontend).
- Имеется 2 способа организации работы коммутатора TCP соединений:
  - Трансляция сетевых адресов (NAT – Network Address Translations);
  - Переадресация вызова TCP (TCP handoff) см. рис. ниже.



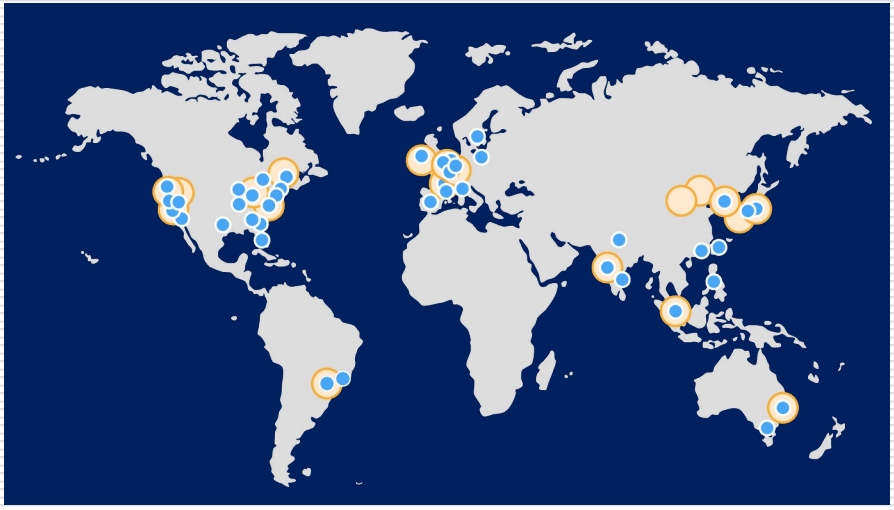
Эффективное рапределение на основе контента запроса

- В случаях, когда разные сервера кластера предоставляют услуги различных служб, то возникает необходимость выбора соответствующего сервера. Здесь тоже имеется 2 пути:
  - Выбор на основе номера порта транспортного протокола.
  - На основе инспектирования содержимого пакета транспортного уровня (content-aware т.е. на основе контента).



Глобальные кластеры серверов

- Облачные провайдеры Amazon и Google имеют собственные ЦОД в разных частях мира, в которых размещаются сервера доступа к облачным сервисам.
- В этих ЦОД размещаются виртуальные машины на базе которых поставщики облачных услуг предоставляют возможность пользователям создавать свои собственные глобальные распределенные системы, содержащие большое количество виртуальных машин объеденных сетью поверх Интернет.



(а) Карта размещения ЦОД AWS

✓ Важнейшей характеристикой такой системы является место расположения серверов служб, которые должны располагаться по близости с потенциальными пользователями системы.
✓ Если близость расположения серверов к местам расположения пользователей не является важной, то можно разместить все ВМ в одном ЦОД и получить преимущество в производительности за счет скорости исполнения межпроцессных коммуникаций по локальной сети ЦОД с низкими величинами задержек.
✓ Если близость пользователей к серверам кластера важна, то важность приобретает и способ диспетчирования запросов, так как они должны направляться ближайшему серверу кластера.

Политика перенаправления

- Необходимость выбора сервера ближайшего к источнику запроса порождает проблему политики перенаправления.
- Если предположить, что по аналогии с локальными кластерами серверов диспетчирование будет осуществляться с помощью коммутатора запросов, то диспетчер должен иметь способность оценивать величину задержки передачи между клиентом и различными серверами.
- Как только сервер будет выбран, диспетчер должен уведомить об этом клиента и перенаправить его запрос серверу. Для этого могут использоваться различные механизмы:
  - Использование в качестве диспетчера DNS сервер;

    Клиент делает запрос на разрешение имени сервера и ему возвращается адрес ближайшего к нему сервера кластера (конечно диспетчер должен знать IP адрес клиента).

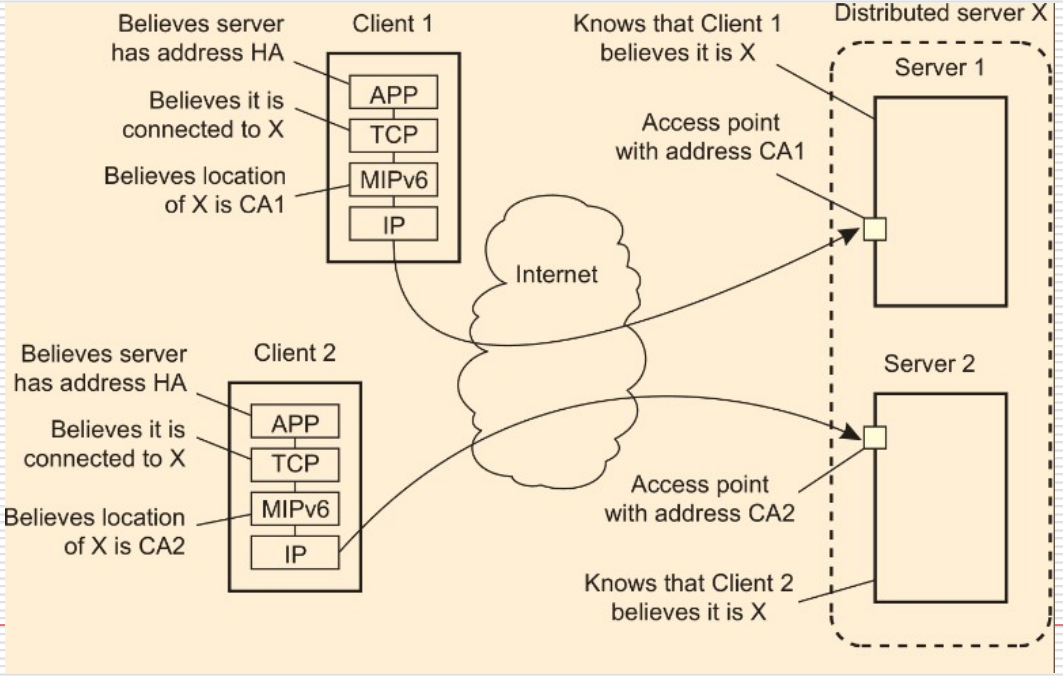
    Недостатки:

    1) клиент может работать через локальный DNS, адрес которого и будет представлен в запросе к диспетчеру на основе DNS сервера.

    2) огромные накладные затраты по времени при использовании локальных DNS, которые могут оказаться не локальными.

Диспетчирование запросов в глобальных кластерах серверов

- Надежное определение требуемого сервера может быть достигнуто с помощью механизма поддержки мобильности используемой в IPv6.
- Каждый мобильный узел сети IPv6 имеет домашний адрес (HA - home address)который определяет где действительно располагается узел. И уникальный адрес контакта (CA – contact address), который назначается ему и который может использоваться для подключения к узлу из внешней сети.



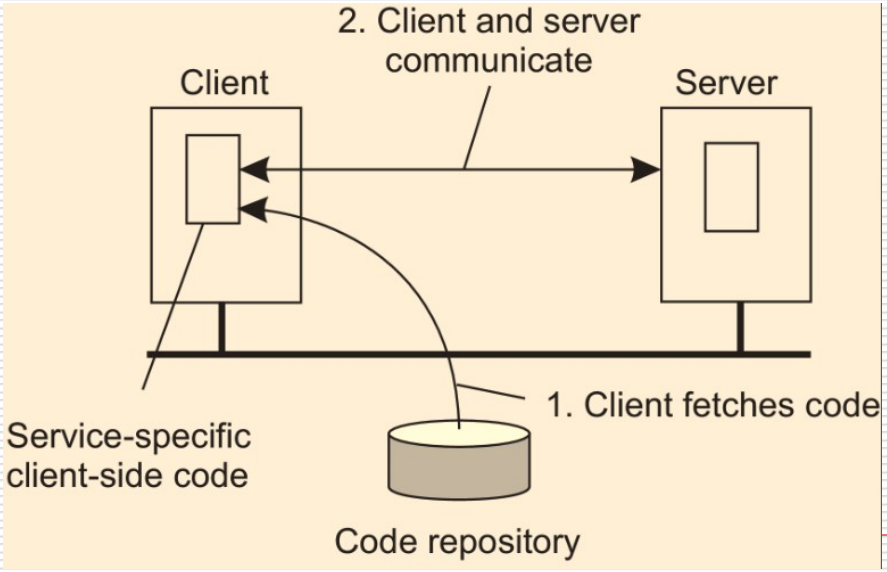
Миграция кода

Основания для переноса кода

- Традиционно перенос кода в распределенных системах происходит в форме переноса процессов (process migration)у в случае которых процесс целиком переносится с одной машины на другую.
- Поддержка переноса кода может также помочь повысить производительность на основе параллелизма, но без обычных сложностей, связанных с параллельным программированием. Типичным примером может быть поиск информации в Web.
- Относительно несложно реализовать поисковый запрос в виде небольшой мобильной программы, переносимой с сайта на сайт. Создав несколько копий этой программы и разослав их по разным сайтам, мы можем добиться линейного возрастания скорости поиска по сравнению с единственным экземпляром программы.
- Помимо повышения производительности существуют и другие причины поддержания переноса кода. Наиболее важная из них — это гибкость.

Пример: Файловый сервер

- Например, рассмотрим сервер, реализующий стандартизованный интерфейс к файловой системе.
- Чтобы предоставить удаленному клиенту доступ к файловой системе, сервер использует специальный протокол. В обычном варианте клиентская реализация интерфейса с файловой системой, основанная на этом протоколе, должна быть скомпонована с приложением клиента. Этот подход предполагает, что подобное программное обеспечение для клиента должно быть доступно уже тогда, когда создается клиентское приложение.



✓ Альтернативой является предоставление сервером клиенту реализации не ранее, чем это будет действительно необходимо, то есть в момент привязки клиента к серверу.

✓ В этот момент клиент динамически загружает реализацию, производит необходимые действия по инициализации, а затем обращается к серверу.

Модели переноса кода

- Перенос кода в широком смысле связан с переносом программ с машины на машину с целью исполнения этих программ в нужном месте.
- Для лучшего понимания различных моделей переноса кода используем шаблон, состоящий из:
  - *Сегмент кода — это часть, содержащая набор инструкций, которые выполняются* в ходе исполнения программы.
  - *Сегмент ресурсов содержит ссылки на* внешние ресурсы, необходимые процессу, такие как файлы, принтеры, устройства, другие процессы и т. п.
  - *Сегмент исполнения используется для хранения* текущего состояния процесса, включая закрытые данные, стек и счетчик программы.
- Различают 2 модели переноса кода, характеризующиеся степенью мобильности:
  - модель слабой мобильности (weak mobility). Согласно этой модели допускается перенос только сегмента кода, возможно вместе с некоторыми данными инициализации.
  - Модель сильной мобильности (strong mobility). В этом случае переносится также и сегмент исполнения.

Модель слабой мобильности

- Согласно этой модели допускается перенос только сегмента кода, возможно вместе с некоторыми данными инициализации. Характерной чертой слабой мобильности является то, что перенесенная программа всегда запускается из своего исходного состояния.
- Достоинство подобного подхода в его простоте.
- Слабая мобильность требуется только для того, чтобы машина, на которую переносится код, была в состоянии его исполнять. Этого вполне достаточно, чтобы сделать код переносимым.

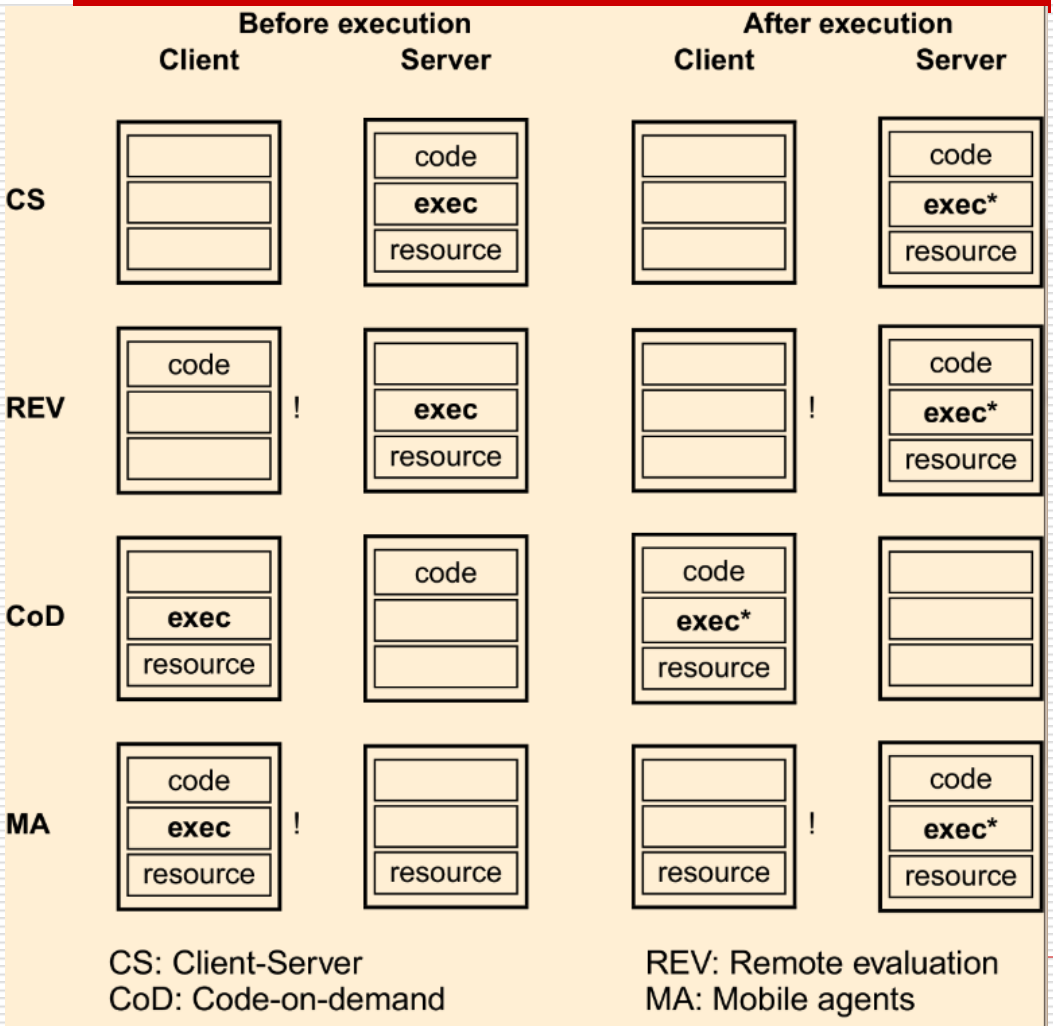
Модель сильной мобильности

- Характерная черта сильной мобильности — то, что работающий процесс может быть приостановлен, перенесен на другую машину и его выполнение продолжено с того места, на котором оно было приостановлено.
- Ясно, что сильная мобильность значительно мощнее слабой, но и значительно сложнее в реализации.

Инициатор переносимости кода

- Независимо от того, является мобильность слабой или сильной, следует провести разделение на:
  - системы с переносом, инициированным отправителем,
  - системы с переносом, инициированным получателем.
- При переносе, инициированном отправителем, перенос инициируется машиной, на которой переносимый код постоянно размещен или выполняется.
  - Обычно перенос, инициированный отправителем, происходит при загрузке программ на вычислительный сервер.
  - Другой пример — передача поисковых программ через Интернет на сервер баз данных в Web для выполнения запроса на этом сервере.
  - Безопасный перенос кода на сервер, как это происходит при переносе, инициированном отправителем, часто требует, чтобы клиент был сначала зарегистрирован и опознавался сервером.
- При переносе, инициированном получателем, инициатива в переносе кода принадлежит машине-получателю. Пример такого подхода — Java-апплеты.

Четыре варианта переноса кода



- Различные варианты переноса кода в зависимости от:
  - вида мобильности (сильная/слабая);
  - типа инициатора (отправитель/получатель);
- порождают следующие парадигмы (модели):
  - Клиент-сервер (CS) (нет мобильности кода);
  - Удаленные вычисления (REV) (слабая/отправителем);
  - Код по требованию (CoD) (слабая/получателем);
  - Мобильный агент (MA) (сильная/отправителем).

Удаленное клонирование процессов

- Помимо переноса работающего процесса, называемого также миграцией процесса, сильная мобильность может также осуществляться за счет удаленного клонирования.
- В отличие от миграции процесса клонирование создает точную копию исходного процесса, которая выполняется на удаленной машине. Клон процесса выполняется параллельно оригиналу. В UNIX-системах удаленное клонирование имеет место при ответвлениях дочернего процесса в том случае, когда этот дочерний процесс продолжает выполнение на удаленной машине.
- Преимущество клонирования — в схожести со стандартными процедурами, осуществляемыми в многочисленных приложениях.
- Единственная разница между ними состоит в том, что клонированный процесс выполняется на другой машине. С этой точки зрения миграция путем клонирования — самый простой способ повышения прозрачности распределения.

Миграция кода в гетерогенных системах

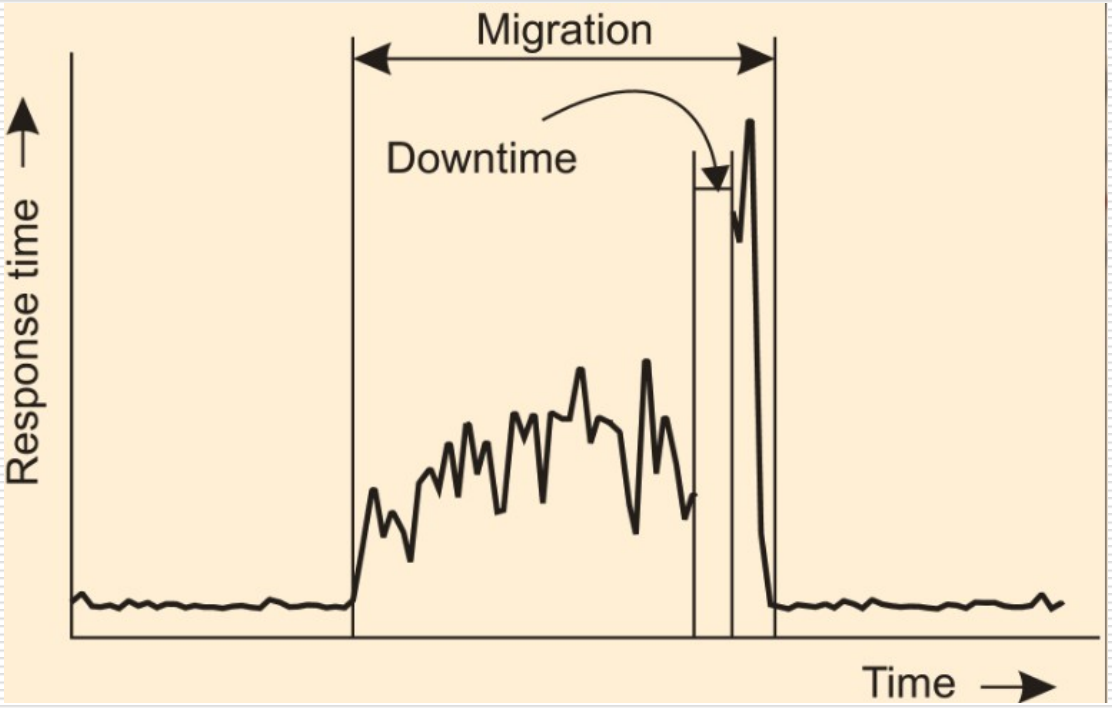
- Перенос в гетерогенных системах требует, чтобы поддерживались все эти платформы, то есть сегмент кода должен выполняться на всех этих платформах без перекомпиляции текста программы. Кроме того, мы должны быть уверены, что сегмент исполнения на каждой из этих платформ будет представлен правильно.
- Проблемы могут быть частично устранены в том случае, если мы ограничимся слабой мобильностью. В этом случае обычно не существует такой информации времени исполнения, которую надо было бы передавать от машины к машине. Это означает, что достаточно скомпилировать исходный текст программы, создав различные варианты сегмента кода — по одному на каждую потенциальную платформу. В конце 1970-х такой подход применялся для переноса Pascal программ.
- В случае сильной мобильности основной проблемой, которую надо будет решить, является перенос сегмента исполнения. Проблема заключается в том, что этот сегмент в значительной степени зависит от платформы, на которой выполняется задача.
- На самом деле перенести сегмент исполнения, не внося в него никаких изменений, можно только в том случае, если машина-приемник имеет ту же архитектуру и работает под управлением той же операционной системы.

Методы портирования процессов в гетерогенных системах.

- В разные годы были предложены следующие способы обеспечения мобильности кода в распределенных системах:
  - Генерация промежуточного кода для абстрактной виртуальной машины (1970-е для программ на Паскале);
  - Использование скриптовых языков и мобильных языков программирования, например Java;
  - Использование переноса кода вместе с окружением времени исполнения, включая операционную систему, с помощью технологии виртуализации (виртуальных машин).

Влияние миграции кода на время ответа ВМ

- При миграции виртуальной машины время неактивности этой ВМ составляет всего несколько минут.



Спасибо за внимание !