Александр Масальских rusalmas@gmail.com

КОМПЬЮТЕРНЫЕ СЕТИ

Лекция №14

Сетевые операционные системы. Виртуализация, кластеры, облачные сервисы

Санкт-Петербург, 2012

CCENT CCNA

- CCENT 2 семестра 140 часов

- CCNA. Part 1. Network Fundamentals 85 h
- CCNA. Part 2. Routing Protocols and Concepts 85 h
- CCNA. Part 3. LAN switching and wireless 70 h
- CCNA. Part 4. Accessing the WAN 80 h

- 4 семестра по 45 дней.

Операционные системы

- Операционная система (англ. operating system, OS) — комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы и прикладными программами, а с другой стороны — предназначены для управления устройствами, управления вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надёжных вычислений. Это определение применимо к большинству современных операционных систем общего назначения.

Операционные системы

- В логической структуре типичной вычислительной системы операционная система занимает положение между устройствами с их микроархитектурой, машинным языком и, возможно, собственными (встроенными) микропрограммами — с одной стороны — и прикладными программами с другой.

- Разработчикам программного обеспечения операционная система позволяет абстрагироваться от деталей реализации и функционирования устройств, предоставляя минимально необходимый набор функций.

Операционные системы

- Основные функции:
  - Исполнение запросов программ (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.).
  - Загрузка программ в оперативную память и их выполнение.
  - Стандартизованный доступ к периферийным устройствам (устройства ввода-вывода).
  - Управление оперативной памятью (распределение между процессами, организация виртуальной памяти).
  - Управление доступом к данным на энергонезависимых носителях (таких как жёсткий диск, оптические диски и др.), организованным в той или иной файловой системе.
  - Обеспечение пользовательского интерфейса.
  - Сохранение информации об ошибках системы.

Операционные системы

- Дополнительные функции:
  - Параллельное или псевдопараллельное выполнение задач (многозадачность).
  - Эффективное распределение ресурсов вычислительной системы между процессами.
  - Разграничение доступа различных процессов к ресурсам.
  - Организация надёжных вычислений (невозможности одного вычислительного процесса намеренно или по ошибке повлиять на вычисления в другом процессе), основана на разграничении доступа к ресурсам.
  - Взаимодействие между процессами: обмен данными, взаимная синхронизация.
  - Защита самой системы, а также пользовательских данных и программ от действий пользователей (злонамеренных или по незнанию) или приложений.
  - Многопользовательский режим работы и разграничение прав доступа (см.: аутентификация, авторизация).

Классификация ОС

- Особенности алгоритмов управления ресурсами:
  - Поддержка многозадачности:
    - Однозадачные
    - Многозадачные
  - Поддержка многопользовательского режима:
    - Однопользовательские
    - Многопользовательские
  - Тип многозадачности
    - Вытесняющая
    - Не вытесняющая
  - Поддержка многонитевости
  - Поддержка многопроцессорной обработки

Классификация ОС

- Особенности аппаратных платформ
  - Одно- многопроцессорность
  - Кластеры
  - Мобильные ОС
- Особенности области использования
  - системы пакетной обработки (например, ОС ЕС)
  - системы разделения времени (UNIX, VMS)
  - системы реального времени (QNX, RT/11)
- Особенности методов построения

Сетевая операционная система

- Сетевая операционная система составляет основу любой вычислительной сети.
- Каждый компьютер в сети в значительной степени автономен, поэтому под сетевой операционной системой в широком смысле понимается совокупность операционных систем отдельных компьютеров, взаимодействующих с целью обмена сообщениями и разделения ресурсов по единым правилам - протоколам.
- В узком смысле сетевая ОС - это операционная система отдельного компьютера, обеспечивающая ему возможность работать в сети.

Сетевая операционная система

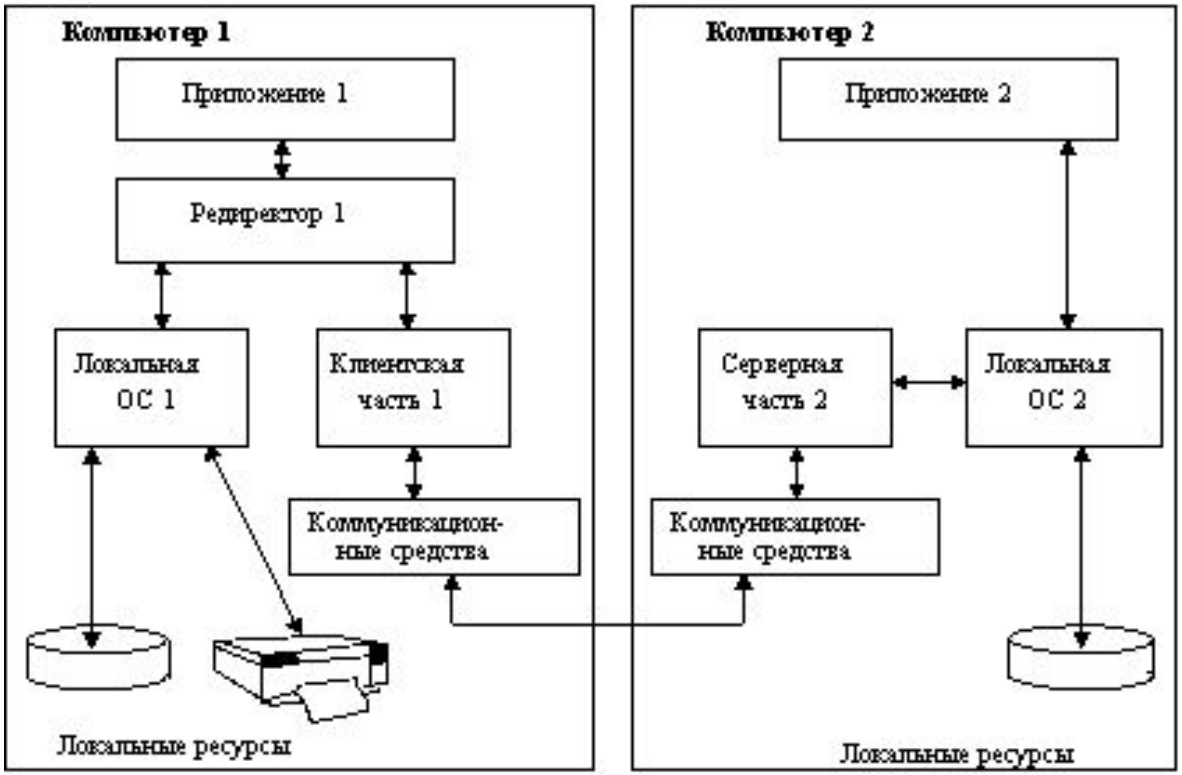
Сетевая операционная система

- Средства управления локальными ресурсами компьютера: функции распределения оперативной памяти между процессами, планирования и диспетчеризации процессов, управления процессорами в мультипроцессорных машинах, управления периферийными устройствами и другие функции управления ресурсами локальных ОС.
- Средства предоставления собственных ресурсов и услуг в общее пользование - серверная часть ОС (сервер). Эти средства обеспечивают, например, блокировку файлов и записей, что необходимо для их совместного использования; ведение справочников имен сетевых ресурсов; обработку запросов удаленного доступа к собственной файловой системе и базе данных; управление очередями запросов удаленных пользователей к своим периферийным устройствам.

Сетевая операционная система

- Средства запроса доступа к удаленным ресурсам и услугам и их использования - клиентская часть ОС (редиректор). Эта часть выполняет распознавание и перенаправление в сеть запросов к удаленным ресурсам от приложений и пользователей, при этом запрос поступает от приложения в локальной форме, а передается в сеть в другой форме, соответствующей требованиям сервера. Клиентская часть также осуществляет прием ответов от серверов и преобразование их в локальный формат, так что для приложения выполнение локальных и удаленных запросов неразличимо.
- Коммуникационные средства ОС, с помощью которых происходит обмен сообщениями в сети. Эта часть обеспечивает адресацию и буферизацию сообщений, выбор маршрута передачи сообщения по сети, надежность передачи и т.п., то есть является средством транспортировки сообщений.

Сетевая операционная система



Сетевая операционная система

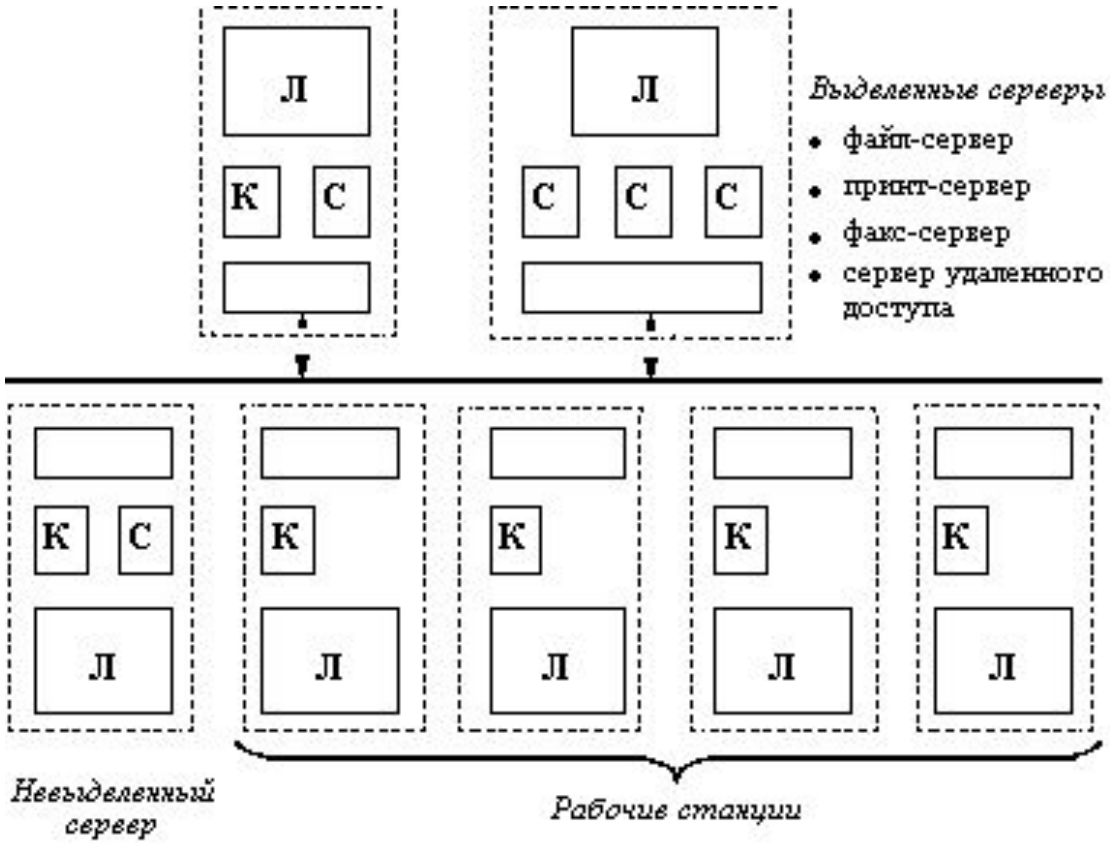
- Первые сетевые ОС представляли собой совокупность существующей локальной ОС и надстроенной над ней ***сетевой оболочки***. При этом в локальную ОС встраивался минимум сетевых функций, необходимых для работы сетевой оболочки, которая выполняла основные сетевые функции. Примером такого подхода является использование на каждой машине сети операционной системы MS DOS (у которой начиная с ее третьей версии появились такие встроенные функции, как блокировка файлов и записей, необходимые для совместного доступа к файлам). Принцип построения сетевых ОС в виде сетевой оболочки над локальной ОС используется и в современных ОС, таких, например, как LANtastic или Personal Ware.

Сетевая операционная система

- Однако более эффективным представляется путь разработки операционных систем, изначально предназначенных для работы в сети. Сетевые функции у ОС такого типа глубоко ***встроены*** в основные модули системы, что обеспечивает их логическую стройность, простоту эксплуатации и модификации, а также высокую производительность. Примером такой ОС является система Windows NT фирмы Microsoft, которая за счет встроенности сетевых средств обеспечивает более высокие показатели производительности и защищенности информации по сравнению с сетевой ОС LAN Manager той же фирмы (совместная разработка с IBM), являющейся надстройкой над локальной операционной системой OS/2.

Сетевая операционная система

- Одноранговые сетевые ОС и ОС с выделенными серверами



Сетевая операционная система

- *Сети отделов* - используются небольшой группой сотрудников, решающих общие задачи. Главной целью сети отдела является разделение локальных ресурсов, таких как приложения, данные, лазерные принтеры и модемы. Сети отделов обычно не разделяются на подсети.
- *Сети кампусов* - соединяют несколько сетей отделов внутри отдельного здания или внутри одной территории предприятия. Эти сети являются все еще локальными сетями, хотя и могут покрывать территорию в несколько квадратных километров. Сервисы такой сети включают взаимодействие между сетями отделов, доступ к базам данных предприятия, доступ к факс-серверам, высокоскоростным модемам и высокоскоростным принтерам.
- *Сети предприятия (корпоративные сети)* - объединяют все компьютеры всех территорий отдельного предприятия. Они могут покрывать город, регион или даже континент. В таких сетях пользователям предоставляется доступ к информации и приложениям, находящимся в других рабочих группах, других отделах, подразделениях и штаб-квартирах корпорации.

Вызов удаленных процедур

- Идея вызова удаленных процедур ***(Remote Procedure Call - RPC)*** состоит в расширении хорошо известного и понятного 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 распределенных вычислений. Наибольшая эффективность использования RPC достигается в тех приложениях, в которых существует интерактивная связь между удаленными компонентами с небольшим временем ответов и относительно малым количеством передаваемых данных. Такие приложения называются RPC-ориентированными.

Вызов удаленных процедур

- Характерными чертами вызова локальных процедур являются:
  - Асимметричность, то есть одна из взаимодействующих сторон является инициатором;
  - Синхронность, то есть выполнение вызывающей процедуры при останавливается с момента выдачи запроса и возобновляется только после возврата из вызываемой процедуры.
- Реализация удаленных вызовов существенно сложнее реализации вызовов локальных процедур.

Вызов удаленных процедур

- Поскольку вызывающая и вызываемая процедуры выполняются на разных машинах, то они имеют разные адресные пространства, и это создает проблемы при передаче параметров и результатов, особенно если машины не идентичны. 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.
- Следующим отличием RPC от локального вызова является то, что он обязательно использует нижележащую систему связи, однако это не должно быть явно видно ни в определении процедур, ни в самих процедурах.

Вызов удаленных процедур

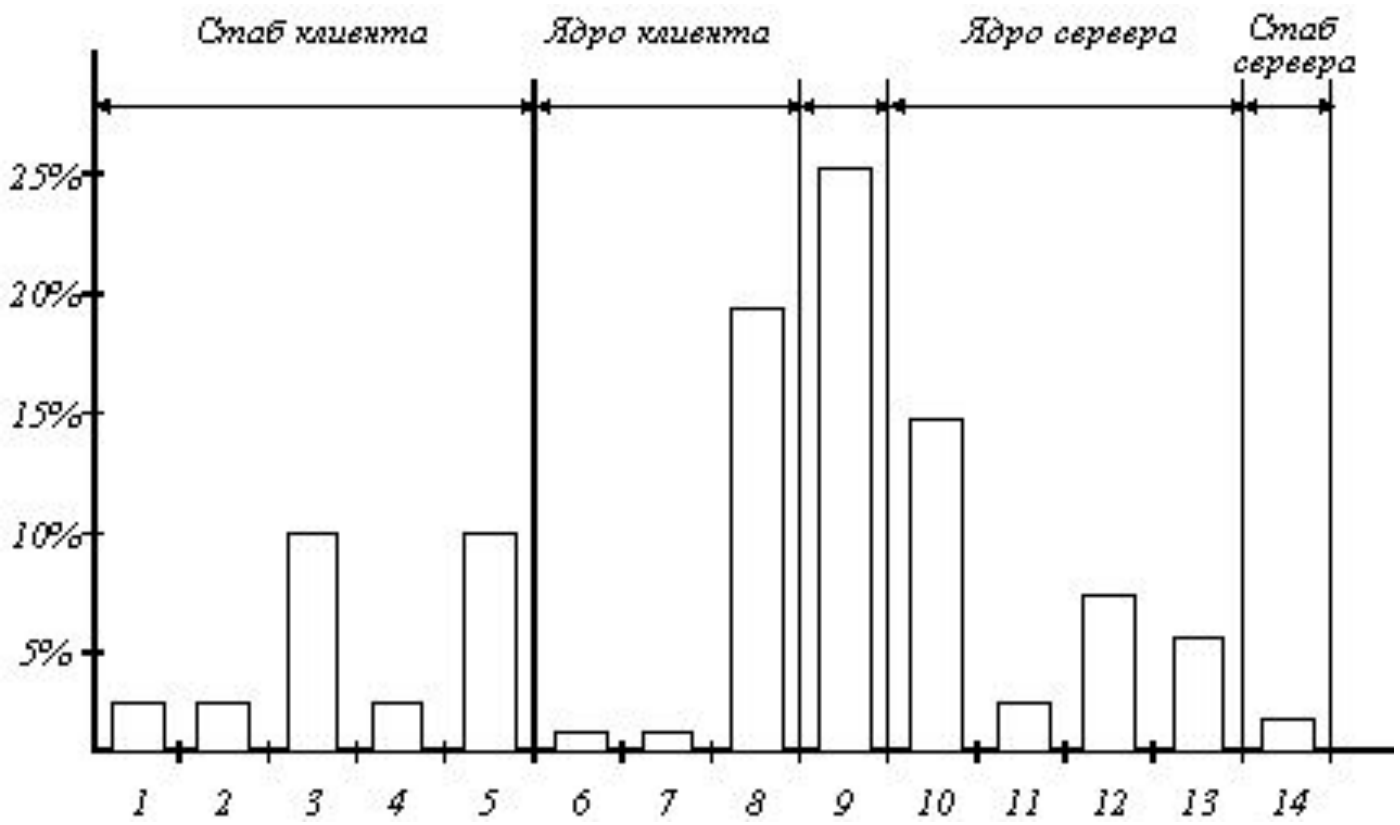
- Удаленность вносит дополнительные проблемы. Выполнение вызывающей программы и вызываемой локальной процедуры в одной машине реализуется в рамках единого процесса. Но в реализации RPC участвуют как минимум два процесса - по одному в каждой машине. В случае, если один из них аварийно завершится, могут возникнуть следующие ситуации: при аварии вызывающей процедуры удаленно вызванные процедуры станут "осиротевшими", а при аварийном завершении удаленных процедур станут "обездоленными родителями" вызывающие процедуры, которые будут безрезультатно ожидать ответа от удаленных процедур.
- Кроме того, существует ряд проблем, связанных с неоднородностью языков программирования и операционных сред: структуры данных и структуры вызова процедур, поддерживаемые в каком-либо одном языке программирования, не поддерживаются точно так же во всех других языках.
- Эти и некоторые другие проблемы решает широко распространенная технология RPC, лежащая в основе многих распределенных операционных систем.

Вызов удаленных процедур

- Идея, положенная в основу RPC, состоит в том, чтобы сделать вызов удаленной процедуры выглядящим по возможности также, как и вызов локальной процедуры. Другими словами - сделать RPC прозрачным: 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 (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.

Вызов удаленных процедур

Вызов удаленных процедур



Вызов удаленных процедур

- 1. Вызов стаба
- 2. Подготовить буфер
- 3. Упаковать параметры
- 4. Заполнить поле заголовка
- 5. Вычислить контрольную сумму в сообщении
- 6. Прерывание к ядру
- 7. Очередь пакета на выполнение
- 8. Передача сообщения контроллеру по шине QBUS
- 9. Время передачи по сети Ethernet
- 10. Получить пакет от контроллера
- 11. Процедура обработки прерывания
- 12. Вычисление контрольной суммы
- 13. Переключение контекста в пространство пользователя
- 14. Выполнение серверного стаба

Вызов удаленных процедур. Динамическое связывание

- Рассмотрим вопрос о том, как клиент задает месторасположение сервера. Одним из методов решения этой проблемы является непосредственное использование сетевого адреса сервера в клиентской программе. Недостаток такого подхода - его чрезвычайная негибкость: при перемещении сервера, или при увеличении числа серверов, или при изменении интерфейса во всех этих и многих других случаях необходимо перекомпилировать все программы, которые использовали жесткое задание адреса сервера. Для того, чтобы избежать всех этих проблем, в некоторых распределенных системах используется так называемое динамическое связывание.
- Начальным моментом для динамического связывания является формальное определение (спецификация) сервера. Спецификация содержит имя файл-сервера, номер версии и список процедур-услуг, предоставляемых данным сервером для клиентов. Для каждой процедуры дается описание ее параметров с указанием того, является ли данный параметр входным или выходным относительно сервера. Некоторые параметры могут быть одновременно входными и выходными - например, некоторый массив, который посылается клиентом на сервер, модифицируется там, а затем возвращается обратно клиенту (операция copy/ restore).

Вызов удаленных процедур. Динамическое связывание

- Формальная спецификация сервера используется в качестве исходных данных для программы-генератора стабов, которая создает как клиентские, так и серверные стабы. Затем они помещаются в соответствующие библиотеки. Когда пользовательская (клиентская) программа вызывает любую процедуру, определенную в спецификации сервера, соответствующая стаб-процедура связывается с двоичным кодом программы. Аналогично, когда компилируется сервер, с ним связываются серверные стабы.
- При запуске сервера самым первым его действием является передача своего серверного интерфейса специальной программе, называемой binder'ом. Этот процесс, известный как процесс регистрации сервера, включает передачу сервером своего имени, номера версии, уникального идентификатора и описателя местонахождения сервера. Описатель системно независим и может представлять собой IP, Ethernet, X.500 или еще какой-либо адрес. Кроме того, он может содержать и другую информацию, например, относящуюся к аутентификации.

Вызов удаленных процедур. Динамическое связывание

- Когда клиент вызывает одну из удаленных процедур первый раз, например, read, клиентский стаб видит, что он еще не подсоединен к серверу, и посылает сообщение binder-программе с просьбой об импорте интерфейса нужной версии нужного сервера. Если такой сервер существует, то binder передает описатель и уникальный идентификатор клиентскому стабу.
- Клиентский стаб при посылке сообщения с запросом использует в качестве адреса описатель. В сообщении содержатся параметры и уникальный идентификатор, который ядро сервера использует для того, чтобы направить поступившее сообщение в нужный сервер в случае, если их несколько на этой машине.
- Этот метод, заключающийся в импорте/экспорте интерфейсов, обладает высокой гибкостью. Например, может быть несколько серверов, поддерживающих один и тот же интерфейс, и клиенты распределяются по серверам случайным образом. Этот метод может также поддерживать аутентификацию клиента. Например, сервер может определить, что он может быть использован только клиентами из определенного списка.

Вызов удаленных процедур. Динамическое связывание

- У динамического связывания имеются недостатки, например, дополнительные накладные расходы (временные затраты) на экспорт и импорт интерфейсов. Величина этих затрат может быть значительна, так как многие клиентские процессы существуют короткое время, а при каждом старте процесса процедура импорта интерфейса должна быть снова выполнена.
- Кроме того, в больших распределенных системах может стать узким местом программа binder, а создание нескольких программ аналогичного назначения также увеличивает накладные расходы на создание и синхронизацию процессов.

Вызов удаленных процедур

- У динамического связывания имеются недостатки, например, дополнительные накладные расходы (временные затраты) на экспорт и импорт интерфейсов. Величина этих затрат может быть значительна, так как многие клиентские процессы существуют короткое время, а при каждом старте процесса процедура импорта интерфейса должна быть снова выполнена.
- Кроме того, в больших распределенных системах может стать узким местом программа binder, а создание нескольких программ аналогичного назначения также увеличивает накладные расходы на создание и синхронизацию процессов.

Семантика RPC в случае отказов

1. Клиент не может определить местонахождения сервера, например, в случае отказа нужного сервера, или из-за того, что программа клиента была скомпилирована давно и использовала старую версию интерфейса сервера. *В этом случае в ответ на запрос клиента поступает сообщение, содержащее код ошибки.*
2. Потерян запрос от клиента к серверу. *Самое простое решение - через определенное время повторить запрос.*

Семантика RPC в случае отказов

3. Потеряно ответное сообщение от сервера клиенту. Этот вариант сложнее предыдущего, так как некоторые процедуры не являются идемпотентными. Но вот процедура снятия некоторой суммы с банковского счета не является идемпотентной, и в случае потери ответа повторный запрос может существенно изменить состояние счета клиента. *Одним из возможных решений является приведение всех процедур к идемпотентному виду. Однако на практике это не всегда удается, поэтому может быть использован другой метод - последовательная нумерация всех запросов клиентским ядром. Ядро сервера запоминает номер самого последнего запроса от каждого из клиентов, и при получении каждого запроса выполняет анализ - является ли этот запрос первичным или повторным.*

Семантика RPC в случае отказов

4. Сервер потерпел аварию после получения запроса. Здесь также важно свойство идемпотентности, но к сожалению не может быть применен подход с нумерацией запросов. В данном случае имеет значение, когда произошел отказ - до или после выполнения операции. Но клиентское ядро не может распознать эти ситуации, для него известно только то, что время ответа истекло. Существует три подхода к этой проблеме:
   a) *Ждать до тех пор, пока сервер не перезагрузится и пытаться выполнить операцию снова. Этот подход гарантирует, что RPC был выполнен до конца по крайней мере один раз, а возможно и более.*
   b) *Сразу сообщить приложению об ошибке. Этот подход гарантирует, что RPC был выполнен не более одного раза.*
   c) *Третий подход не гарантирует ничего. Когда сервер отказывает, клиенту не оказывается никакой поддержки. RPC может быть или не выполнен вообще, или выполнен много раз. Во всяком случае этот способ очень легко реализовать.*

Стратегии борьбы с сиротами

- *Уничтожение*. До того, как клиентский стаб посылает RPC-сообщение, он делает отметку в журнале, оповещая о том, что он будет сейчас делать. Журнал хранится на диске или в другой памяти, устойчивой к сбоям. После аварии система перезагружается, журнал анализируется и сироты ликвидируются. К недостаткам такого подхода относятся, во-первых, повышенные затраты, связанные с записью о каждом RPC на диск, а, во-вторых, возможная неэффективность из-за появления сирот второго поколения, порожденных RPC-вызовами, выданными сиротами первого поколения.
- *Перевоплощение*. В этом случае все проблемы решаются без использования записи на диск. Метод состоит в делении времени на последовательно пронумерованные периоды. Когда клиент перезагружается, он передает широковещательное сообщение всем машинам о начале нового периода. После приема этого сообщения все удаленные вычисления ликвидируются. Конечно, если сеть сегментированная, то некоторые сироты могут и уцелеть.

Стратегии борьбы с сиротами

- *Мягкое перевоплощение* аналогично предыдущему случаю, за исключением того, что отыскиваются и уничтожаются не все удаленные вычисления, а только вычисления перезагружающегося клиента.
- *Истечение срока*. Каждому запросу отводится стандартный отрезок времени T, в течение которого он должен быть выполнен. Если запрос не выполняется за отведенное время, то выделяется дополнительный квант. Хотя это и требует дополнительной работы, но если после аварии клиента сервер ждет в течение интервала T до перезагрузки клиента, то все сироты обязательно уничтожаются.

Нити и RPC

- Обычно в распределенных системах используются как RPC, так и нити. Так как нити были введены как дешевая альтернатива стандартным процессам, то естественно, что исследователи обратили особое внимание в этом контексте на RPC: нельзя ли их также сделать облегченными. Было замечено, что в распределенных системах значительное количество RPC обрабатывается на той же машине, на которой они были вызваны (локально), например, вызовы к менеджеру окон. Поэтому была предложена новая схема, которая делает возможным для нити одного процесса вызвать нить другого процесса на этой же машине более эффективно, чем обычным способом.

Нити и RPC

- Идея заключается в следующем. Когда стартует серверная нить S, то она экспортирует свой интерфейс, сообщая о нем ядру. Интерфейс определяет, какие процедуры могут быть вызваны, каковы их параметры и т.п. Когда стартует клиентская нить C, то она импортирует интерфейс из ядра в том случае, если собирается вызвать S, и ей дается специальный идентификатор для выполнения определенного вызова. Ядро теперь знает, что C собирается позже вызвать S и создает специальные структуры данных для подготовки к вызову.

Нити и RPC

- Одна из этих структур данных является стеком аргументов, который разделяется нитями C и S и отображается в оба адресных пространства для чтения и записи. Для вызова сервера нить C помещает аргументы в разделяемый стек, используя обычную процедуру передачи параметров, а затем прерывает ядро, помещая данный ей идентификатор в регистр. По этому идентификатору ядро видит, что вызов является локальным. (Если бы он был удаленным, то ядро обработало бы его обычным способом для удаленных вызовов.) Затем ядро выполняет переключение из адресного пространства клиента в адресное пространство нити-сервера и запускает в рамках клиентской нити требуемую процедуру сервера. При таком способе вызова аргументы уже загружены в нужное место, так что копирование или перегруппировка аргументов не требуется. Главный результат - локальный вызов RPC - будет выполнен этим способом гораздо быстрее.

Нити и RPC

- Другой прием широко используется для ускорения удаленных RPC. Идея основана на следующем наблюдении: когда нить-сервер блокируется, ожидая нового запроса, ее контекст почти всегда не содержит важной информации. Следовательно, когда нить завершает обработку запроса, то ее просто удаляют. При поступлении на сервер нового сообщения ядро создает новую нить для обслуживания этого запроса. Кроме того ядро помещает сообщение в адресное пространство сервера и устанавливает новый стек нити для доступа к сообщению. Эту схему иногда называют неявным вызовом.
- Этот метод имеет несколько преимуществ по сравнению с обычным RPC. Во-первых, нити не должны блокироваться, ожидая новую работу, следовательно контекст не нужно сохранять, во-вторых, создание новой нити проще, чем активизация существующей приостановленной, так как не нужно восстанавливать контекст.
-

RPC

- RMI (англ. Remote Method Invocation) — программный интерфейс вызова удаленных методов в языке Java.
- COM (англ. Component Object Model — объектная модель компонентов;) — это технологический стандарт от компании Microsoft, предназначенный для создания программного обеспечения на основе взаимодействующих компонентов, каждый из которых может использоваться во многих программах одновременно. Выпущенная в 1996 году технология DCOM (англ. Distributed COM — распределённая COM) основана на технологии DCE/RPC (разновидности RPC). DCOM позволяет COM-компонентам взаимодействовать друг с другом по сети.
- **CORBA** (англ. *Common Object Request Broker Architecture* — общая архитектура брокера объектных запросов) — технологический стандарт написания распределённых приложений, продвигаемый консорциумом (рабочей группой) OMG и соответствующая ему информационная технология.

Распределенные файловые системы

- Ключевым компонентом любой распределенной системы является файловая система. Как и в централизованных системах, в распределенной системе функцией файловой системы является хранение программ и данных и предоставление доступа к ним по мере необходимости.
- Файловая система поддерживается одной или более машинами, называемыми файл-серверами. Файл-серверы перехватывают запросы на чтение или запись файлов, поступающие от других машин (не серверов). Эти другие машины называются клиентами. Каждый посланный запрос проверяется и выполняется, а ответ отсылается обратно.
- Файл-серверы обычно содержат иерархические файловые системы, каждая из которых имеет корневой каталог и каталоги более низких уровней. Рабочая станция может подсоединять и монтировать эти файловые системы к своим локальным файловым системам. При этом монтируемые файловые системы остаются на серверах.

Распределенные файловые системы

- **Файловый сервис** - это описание функций, которые файловая система предлагает своим пользователям. Это описание включает имеющиеся примитивы, их параметры и функции, которые они выполняют. С точки зрения пользователей файловый сервис определяет то, с чем пользователи могут работать, но ничего не говорит о том, как все это реализовано. В сущности, файловый сервис определяет интерфейс файловой системы с клиентами.
- **Файловый сервер** - это процесс, который выполняется на отдельной машине и помогает реализовывать файловый сервис. В системе может быть один файловый сервер или несколько, но в хорошо организованной распределенной системе пользователи не знают, как реализована файловая система.

Распределенные файловые системы

- Файловый сервис в распределенных файловых системах имеет две функционально различные части:
  - собственно файловый сервис
  - сервис каталогов.
- Первый имеет дело с операциями над отдельными файлами, такими, как чтение, запись или добавление, а второй - с созданием каталогов и управлением ими, добавлением и удалением файлов из каталогов и т. п.

Распределенные файловые системы. Семантика разделения файлов

- Есть четыре различных подхода к работе с разделяемыми файлами в распределенных системах:
  - **Семантика UNIX**. Каждая операция над файлом немедленно становится видимой для всех процессов.
  - **Сессионная семантика**. Изменения не видны до тех пор, пока файл не закрывается.
  - **Неизменяемые файлы**. Модификации невозможны, разделение файлов и репликация упрощаются.
  - **Транзакции**. Все изменения делаются по принципу "все или ничего".

Распределенные файловые системы. Вопросы разработки структуры файловой системы

- Распределение серверной и клиентской частей между машинами может быть разным. В некоторых системах (например, NFS) нет разницы между клиентом и сервером, на всех машинах работает одно и то же базовое программное обеспечение, так что любая машина, которая хочет предложить файловый сервис, свободно может это сделать. Для этого ей достаточно экспортировать имена выбранных каталогов, чтобы другие машины могли иметь к ним доступ.
- В других системах файловый сервер - это только пользовательская программа, так что система может быть сконфигурирована как клиент, как сервер или как клиент и сервер одновременно. Третьим, крайним случаем, является система, в которой клиенты и серверы - это принципиально различные машины, как в терминах аппаратуры, так и в терминах программного обеспечения. Серверы могут даже работать под управлением другой операционной системы.

Распределенные файловые системы. Вопросы разработки структуры файловой системы

- Вторым важным вопросом реализации файловой системы является структуризация сервиса файлов и каталогов. Один подход заключается в комбинировании этих двух сервисов на одном сервере. При другом подходе эти сервисы разделяются. В последнем случае при открытии файла требуется обращение к серверу каталогов, который отображает символьное имя в двоичное, а затем обращение к файловому серверу с двоичным именем для действительного чтения или записи файла.
- Аргументом в пользу разделения сервисов является тот факт, что они на самом деле слабо связаны, поэтому их раздельная реализация более гибкая. Например, можно реализовать сервер каталогов MS-DOS и сервер каталогов UNIX, которые будут использовать один и тот же файловый сервер для физического хранения файлов. Разделение этих функций также упрощает программное обеспечение. Недостатком является то, что использование двух серверов увеличивает интенсивность сетевого обмена.

Распределенные файловые системы. Вопросы разработки структуры файловой системы

- Последний структурный вопрос связан с хранением на серверах информации о состоянии клиентов. Существует две конкурирующие точки зрения.
- *Stateless-серверы:*
  - отказоустойчивы;
  - не нужны вызовы OPEN/CLOSE;
  - меньше памяти сервера расходуется на таблицы;
  - нет ограничений на число открытых файлов;
  - отказ клиента не создает проблем для сервера.
- *Statefull-серверы:*
  - более короткие сообщения при запросах;
  - лучше производительность;
  - возможно опережающее чтение;
  - легче достичь идемпотентности;
  - возможна блокировка файлов.

Распределенные файловые системы.

- Network File System (NFS) — протокол сетевого доступа к файловым системам, первоначально разработан Sun Microsystems в 1984 году. Основан на протоколе вызова удалённых процедур (ONC RPC, Open Network Computing Remote Procedure Call, RFC 1057, RFC 1831). Позволяет подключать (монтировать) удалённые файловые системы через сеть, описан в RFC 1094, RFC 1813, RFC 3530 и RFC 5661.
- NFS абстрагирована от типов файловых систем как сервера, так и клиента, существует множество реализаций NFS-серверов и клиентов для различных операционных систем и аппаратных архитектур. В настоящее время используется наиболее зрелая версия NFS v.4 (RFC 3010,RFC 3530), поддерживающая различные средства аутентификации (в частности, Kerberos и LIPKEY с использованием протокола RPCSEC GSS) и списков контроля доступа (как POSIX, так и Windows-типов).

Распределенные файловые системы.

- NFS предоставляет клиентам прозрачный доступ к файлам и файловой системе сервера. Любое приложение клиента, которое может работать с локальным файлом, с таким же успехом может работать и с NFS файлом, без каких либо модификаций самой программы.
- NFS клиенты получают доступ к файлам на NFS сервере путем отправки RPC-запросов на сервер. Это может быть реализовано с использованием обычных пользовательских процессов — а именно, NFS клиент может быть пользовательским процессом, который осуществляет конкретные RPC вызовы на сервер, который так же может быть пользовательским процессом.
- Важной частью последней версии стандарта NFS (v4.1) стала спецификация pNFS, нацеленная на обеспечение распараллеленной реализации общего доступа к файлам, увеличивающая скорость передачи данных пропорционально размерам и степени параллелизма системы.

Распределенные файловые системы.

- Первоначальная разработка NFS имела следующие цели:

1. NFS не должна ограничиваться операционной системой UNIX. Любая операционная система должна быть способной реализовать сервер и клиент NFS.
2. Протокол не должен зависеть от каких-либо определённых аппаратных средств.
3. Должны быть реализованы простые механизмы восстановления в случае отказов сервера или клиента.
4. Приложения должны иметь прозрачный доступ к удаленным файлам без использования специальных путевых имен или библиотек и без перекомпиляции.
5. Для UNIX-клиентов должна поддерживаться семантика UNIX.
6. Производительность NFS должна быть сравнима с производительностью локальных дисков.
7. Реализация не должна быть зависимой от транспортных средств.

Распределенные файловые системы.

- Версия 4
- NFSv4 (RFC 3010, декабрь 2000 г., RFC 3530, пересмотренная в апреле 2003), под влиянием AFS и CIFS, включила в себя улучшение производительности, высокую безопасность, и предстала полноценным протоколом. Версия 4 стала первой версией, разработанной совместно с Internet Engineering Task Force (IETF), после того, как Sun Microsystems передала развитие протоколов NFS. NFS версии v4.1 была одобрена IESG в январе 2010 года, и получила номерRFC 5661 (новая спецификация, объёмом 612 страниц, стала известна как самый длинный документ, одобренный IETF). Важным нововведением версии 4.1 является спецификация pNFS — Parallel NFS, механизма параллельного доступа NFS-клиента к данным множества распределенных NFS-серверов. Наличие такого механизма в стандарте сетевой файловой системы поможет строить распределённые «облачные» («cloud») хранилища и информационные системы.

Распределенные файловые системы

- Версия 4
- NFSv4 (RFC 3010, декабрь 2000 г., RFC 3530, пересмотренная в апреле 2003), под влиянием AFS и CIFS, включила в себя улучшение производительности, высокую безопасность, и предстала полноценным протоколом. Версия 4 стала первой версией, разработанной совместно с Internet Engineering Task Force (IETF), после того, как Sun Microsystems передала развитие протоколов NFS. NFS версии v4.1 была одобрена IESG в январе 2010 года, и получила номерRFC 5661 (новая спецификация, объёмом 612 страниц, стала известна как самый длинный документ, одобренный IETF). Важным нововведением версии 4.1 является спецификация pNFS — Parallel NFS, механизма параллельного доступа NFS-клиента к данным множества распределенных NFS-серверов. Наличие такого механизма в стандарте сетевой файловой системы поможет строить распределённые «облачные» («cloud») хранилища и информационные системы.

Сеть хранения данных

- Сеть хранения данных, СХД (англ. Storage Area Network, SAN) — представляет собой архитектурное решение для подключения внешних устройств хранения данных, таких как дисковые массивы, ленточные библиотеки, оптические приводы к серверам таким образом, чтобы операционная система распознала подключённые ресурсы как локальные.
- SAN характеризуются предоставлением так называемых сетевых блочных устройств (обычно посредством протоколов Fibre Channel, iSCSI или AoE), в то время как сетевые хранилища данных (англ. Network Attached Storage, NAS) нацелены на предоставление доступа к хранящимся на их файловой системе данным при помощи сетевой файловой системы (такой как NFS, SMB/CIFS, или Apple Filing Protocol).

iSCSI

- iSCSI (англ. Internet Small Computer System Interface) — протокол, который базируется на TCP/IP и разработан для установления взаимодействия и управления системами хранения данных, серверами и клиентами.
- iSCSI описывает:
- Транспортный протокол для SCSI, который работает поверх TCP.
- Механизм инкапсуляции SCSI команд в IP сети.
- Протокол для нового поколения систем хранения данных, которые будут использовать «родной» TCP/IP.
- Протокол iSCSI является стандартизованным по RFC 3720. Существует много коммерческих и некоммерческих реализаций этого протокола.

Виртуализация

- Виртуализация в вычислениях — процесс представления набора вычислительных ресурсов, или их логического объединения, который даёт какие-либо преимущества перед оригинальной конфигурацией. Это новый виртуальный взгляд на ресурсы составных частей, не ограниченных реализацией, физической конфигурацией или географическим положением. Обычно виртуализированные ресурсы включают в себя вычислительные мощности и хранилище данных. По-научному, виртуализация — это изоляция вычислительных процессов и ресурсов друг от друга.

Виртуализация

- Гипервизор (или Монитор виртуальных машин) — в компьютерах программа или аппаратная схема, обеспечивающая или позволяющая одновременное, параллельное выполнение нескольких или даже многих операционных системна одном и том же хост-компьютере. Гипервизор также обеспечивает изоляцию операционных систем друг от друга, защиту и безопасность, разделение ресурсов между различными запущенными ОС и управление ресурсами.
- Гипервизор также может (но не обязан) предоставлять работающим под его управлением на одном хост-компьютере ОС средства связи и взаимодействия между собой (например, через обмен файлами или сетевые соединения) так, как если бы эти ОС выполнялись на разных физических компьютерах.

Виртуализация

- Гипервизор сам по себе в некотором роде является минимальной операционной системой (микроядром или наноядром). Он предоставляет запущенным под его управлением операционным системам сервис виртуальной машины, виртуализируя или эмулируя реальное (физическое) аппаратное обеспечение конкретной машины, и управляет этими виртуальными машинами, выделением и освобождением ресурсов для них. Гипервизор позволяет независимое «включение», перезагрузку, «выключение» любой из виртуальных машин с той или иной ОС. При этом операционная система, работающая в виртуальной машине под управлением гипервизора, может, но не обязана «знать», что она выполняется в виртуальной машине, а не на реальном аппаратном обеспечении.

Виртуализация

- Автономный гипервизор (Тип 1)
- Имеет свои встроенные драйверы устройств, модели драйверов и планировщик и поэтому не зависит от базовой ОС. Так как автономный гипервизор работает непосредственно на оборудовании, то он более производителен.
- Пример: VMware ESX
- Требует Virtualization Technology (или SVM, Secure Virtual Machine)

Виртуализация

- На основе базовой ОС (Тип 2, V)
- Это компонент, работающий в одном кольце с ядром основной ОС (кольцо 0). Гостевой код может выполняться прямо на физическом процессоре, но доступ к устройствам ввода-вывода компьютера из гостевой ОС осуществляется через второй компонент, обычный процесс основной ОС — монитор уровня пользователя.
- Примеры: Microsoft Virtual PC, VMware Workstation, QEMU, Parallels, VirtualBox.

Виртуализация

- Гибридный (Тип 1+)
- Гибридный гипервизор состоит из двух частей: из тонкого гипервизора, контролирующего процессор и память, а также работающей под его управлением специальной сервисной ОС в кольце пониженного уровня. Через сервисную ОС гостевые ОС получают доступ к физическому оборудованию.
- Примеры: Microsoft Virtual Server, Sun Logical Domains, Xen, Citrix XenServer, Microsoft Hyper-V

Виртуализация

- VMWare вырос из исследовательского проекта DISCO в Стэнфордском университете
- XEN зародился в Кембриджском университете

Кластеры

- Кластер — группа компьютеров, объединённых высокоскоростными каналами связи и представляющая с точки зрения пользователя единый аппаратный ресурс.
- Один из первых архитекторов кластерной технологии Грегори Пфистер дал кластеру следующее определение: «Кластер — это разновидность параллельной или распределённойсистемы, которая:

1. состоит из нескольких связанных между собой компьютеров;
2. используется как единый, унифицированный компьютерный ресурс».

Кластеры

- Обычно различают следующие основные виды кластеров:

1. отказоустойчивые кластеры (High-availability clusters, HA, кластеры высокой доступности)
2. кластеры с балансировкой нагрузки (Load balancing clusters)
3. вычислительные кластеры (High perfomance computing clusters)
4. grid-системы