Содержание:

ВВЕДЕНИЕ

В настоящее время стремительными темпами распространяются сетевые компьютерные технологии. Предпосылками к этому служат процессы дальнейшего развития программных и аппаратных средств вычислительной техники. Так как любая информационная система предполагает одновременную работу с ней пользователей различных категорий, то разумней всего было построить такую систему по принципу «клиент-сервер».

Сеть клиент-сервер открывает множество возможностей: гибкое управление функциями и режимом работы устройств, обработка различных данных и сигналов в едином центре, автоматизация функций, удаленное управление, автономная работа по единому замыслу и многое другое.

В силу своей "глобальности" (нужно обеспечить доступ к системе из территориально разнесенных между собой точек), а также в силу ряда других причин такие системы часто имеют очень сложную архитектуру, предполагающую их функционирование в виде набора компонентов, каждый из которых выполняется на отдельном узле. Поскольку число таких систем постоянно возрастает, требования, предъявляемые к ним, достаточно серьезны.

По мере роста популярности систем "клиент-сервер" набирала силу и технология объектно-ориентированного программирования, которая предлагала перейти к системной архитектуре с тремя слоями: слой представления отводится пользовательскому интерфейсу, слой предметной области предназначен для описания основных функций приложения, необходимых для достижения поставленной перед ним цели, а третий слой представляет источник данных.

В настоящее время можно считать, что бум технологий, связанных с клиент-сервернойархитектурой, все еще продолжается - большинство работающих в настоящее время информационных систем выполнено в этой технологии. Однако актуальными являются направления, связанные с развитием этой идеи - так называемые трехслойные и многослойные, а также децентрализованные приложения.

Опыт последних лет разработки программного обеспечения (ПО) показывает, что архитектура информационной системы должна выбираться с учетом нужд бизнеса, а не личных пристрастий разработчиков.

Не секрет, что правильная и четкая организация информационных бизнес-решений является слагающим фактором успеха любой компании. Особенно важным этот фактор является для предприятий среднего и малого бизнеса,которым необходима система, которая способна предоставить весь объем бизнес-логикидля решения задач компании. В то же время, такие системы для компаний со средним и малым масштабом сетей часто попадают под критерий —цена - качество‖, то есть должны обладать максимальной производительностью и надежностью при доступной цене.

Актуальность курсовой работы заключается в том, что понимание вариантов архитектуры клиент-сервер помогает значитьльно упростить написание програмного обеспечения под определеную архектетуру сервера.

Цель курсовой работы - разработка регламента выполнения процесса «Варианты архитектуры клиент-сервер».

В связи с поставленной целью решались седующие задачи курсовой работы:

Описать предметную область. Поставить задачи;

Моделирование сортировки данных в массиве;

Сравнение алгаритмов сортировки;

Оценка эфективности алгоритмов.

Обектом исследования является изучение процесса алгоритма сортировки данных, сравнение алгоритмов .

Предметом исследования является Сортировка данных в массиве. Оценка эффективности метода.

Структура курсовой работы состоит из введения, двух глав, заключения и списка использованных источников.

Глава 1. Аналитическая часть

1. Описание предметной области. Архитектура сети клиент-сервер.

Как правило, компьютеры и программы, входящие в состав информационной системы, не являются равноправными. Некоторые из них владеют ресурсами (файловая система, процессор, принтер, база данных и т.д.), другие имеют возможность обращаться к этим ресурсам. Компьютер (или программу), управляющий ресурсом, называют сервером этого ресурса (файл-сервер, сервер базы данных, вычислительный сервер...). Клиент и сервер какого-либо ресурса могут находиться как в рамках одной вычислительной системы, так и на различных компьютерах, связанных сетью.

Самое примечательное свойство архитектуры клиент-сервер состоит в возможности удалить клиента от сервера на любое расстояние без существенного снижения скоростных характеристик системы (даже в случае сложных запросов) и без всяких изменений в программном обеспечении. Удаленный клиент подключается к серверу с помощью телефонного или иного канала. Это свойство очень ценно для организации распределенной обработки данных. Кроме того, оно позволяет заменять систему управления базами данных (СУБД), операционную систему и сервер, не изменяя программного обеспечения клиентской части системы.

Основные понятия архитектуры сети клиент-сервер

Уже само понятие «архитектура клиент-сервер» трактуется разработчиками по-разному. Все сходятся лишь в одном: для организации вычислительного процесса при распределенной обработке данных желательно использование архитектуры клиент-сервер. Так, некоторые определяют архитектуру клиент-сервер как модель взаимодействия компьютеров и процессов в сети. Для других утверждение, что некоторая информационная система имеет архитектуру клиент-сервер, означает, что прикладная составляющая этой системы имеет распределенный характер и состоит из двух взаимосвязанных компонент, одна из которых (клиент) формирует и посылает запросы высокого уровня другой компоненте (серверу), задача которой состоит в обслуживании этих запросов.

Клиент - это любой компьютер или программа, подключающиеся к службам другого компьютера или программы.

Сервер - это обычно компьютер, предоставляющий общие ресурсы пользователям сети.

В том случае, когда информационная система объединяет достаточно большое количество различных информационных ресурсов и серверов приложений, встает вопрос об оптимальном управлении всеми ее компонентами. В этом случае используют специализированные средства - менеджеры обработки транзакций (часто их называют просто «менеджеры транзакций»). При этом понятие транзакции расширяется по сравнению с используемым в теории баз данных. В данном случае это не атомарное действие над базой данных, а любое действие в системе - выдача сообщения, запись в индексный файл, печать отчета и т.д.

Транзакция - объединение нескольких действий в одно действие, которое выполняется или не выполняется как единое целое.

Топология - способ соединения компьютеров в сети.

1.1 Классификация архитектуры клиент-сервер

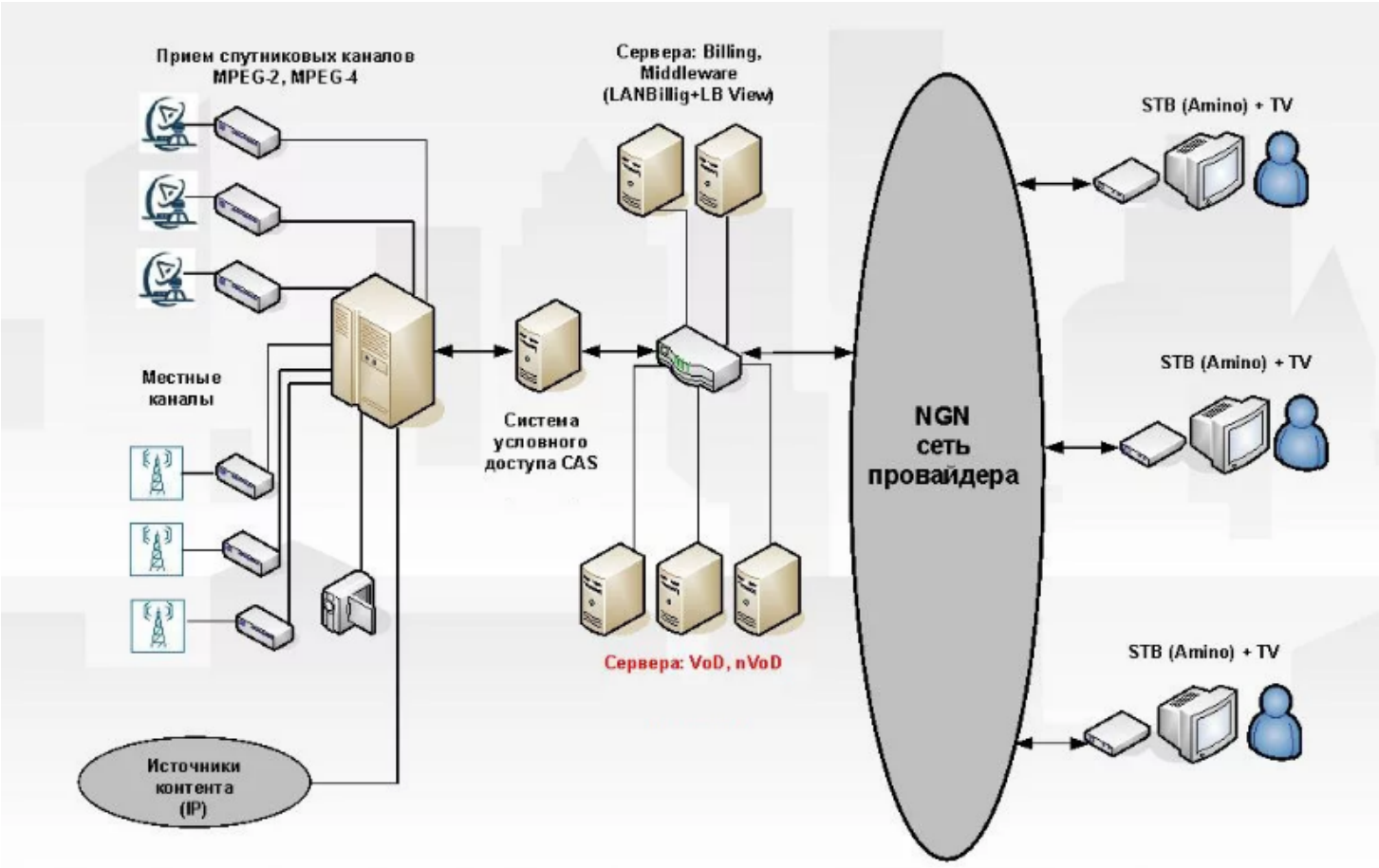
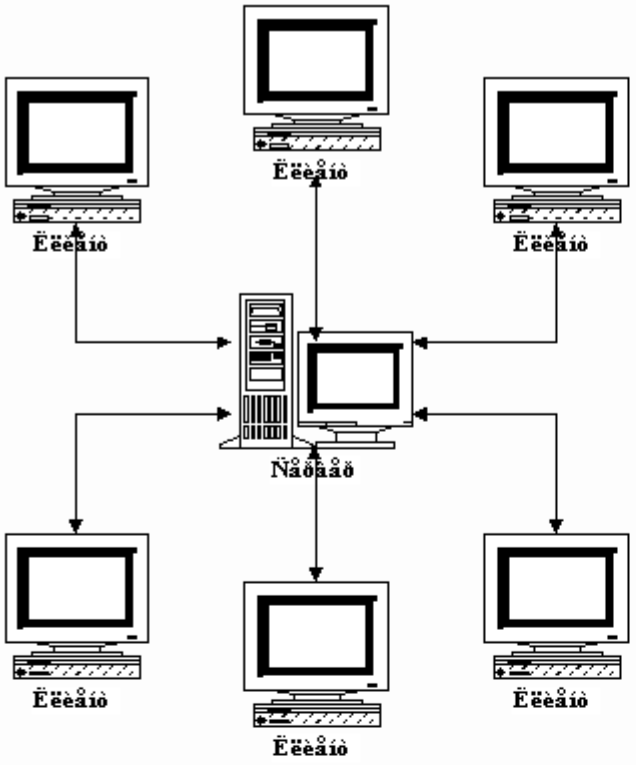
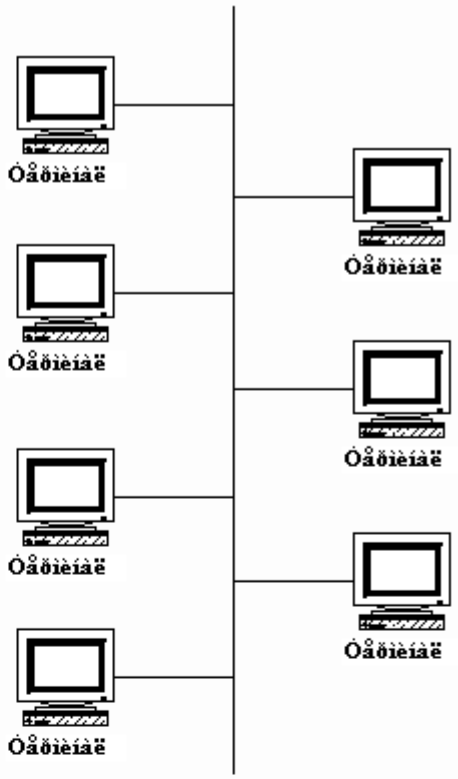


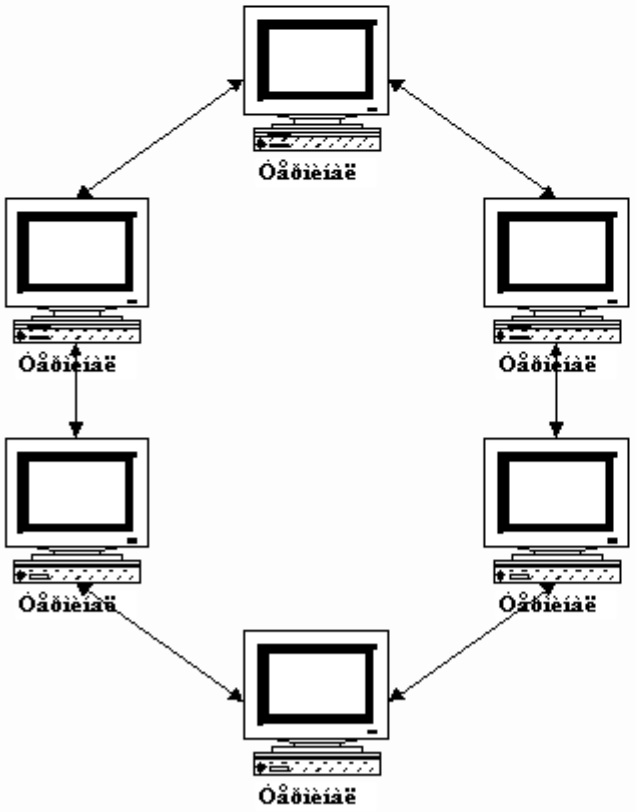
Рис. 1. Наглядная схема работы архитектуры сети клиент-сервер.


Ëëèåíò
Ëëèåíò
Ëëèåíò
Ñåðâåð
Ëëèåíò
Ëëèåíò
Ëëèåíò

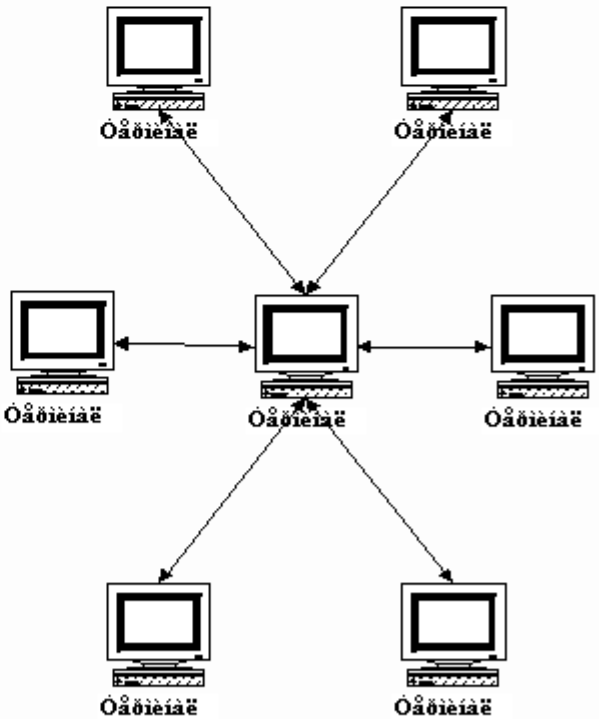
à)


Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë

â)


Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë

c)


Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë
Óåðìèíàë

d)

Рис. 2. Общая схема классификации архитектуры сети клиент-сервер.

Шинная топология - это тип сети клиент-сервер, который представляет собой центральную линию, к которой подключены сервер и клиент. (Рис.2. a.)

Топология «звезда» - это такой тип сети клиент-сервер, при котором файловый сервер находится в центре сети.(Рис.2. d.)

Кольцевая топология - это такой тип сети клиент-сервер, при котором все рабочие станции и сервер соединены друг с другом по кольцу, по которому посылается информация, снабженная адресом получателя. .(Рис.2. c.)

1.2 Характеристика архитектуры сети клиент-сервер.

Способы соединения компьютеров в сети клиент-сервер

Топология «Звезда»

Здесь файловый сервер находится в центре.

Рис. 2. топология «звезда»

Достоинства:

1. Повреждение кабеля является проблемой для одного конкретного компьютера и в целом не сказывается на работе сети.

2. Просто выполняется подключение, так как клиент должен соединяться только с сервером.

3. Механизмы защиты против несанкционированного доступа оптимальны.

4. Высокая скорость передачи данных от клиента к серверу, так как оба компьютера непосредственно соединены друг с другом.

Недостатки:

1. Если географически сервер находиться не в центре сети, то подключение к нему отдельных удаленных клиентов может быть затруднительным и дорогим.

2. В то время как передача данных от клиента к серверу (и обратно) происходит быстро, скорость передачи данных между отдельными клиентами мала.

3. Мощность всей сети зависит от возможностей сервера. Если он недостаточно оснащен или плохо сконфигурирован, то будет являться тормозом для всей системы.

4. Невозможна коммуникация между отдельными клиентами без помощи сервера.

Кольцевая топология

В этом случае все клиенты и сервер соединены друг с другом по кольцу, по которому посылается информация, снабженная адресом получателя. Клиенты получают соответствующие данные, анализируя адрес посланного сообщения.

Достоинства:

1. Так как различная информация постоянно циркулирует по кругу соединенных друг с другом компьютеров, то достигается наивысшая эффективность информационного потока.

2. Нет ограничений на длину всей сети, то есть имеет значение только расстояние между отдельными компьютерами.

Недостатки:

1. Время передачи данных увеличивается пропорционально числу соединенных в кольцо компьютеров.

2. Каждый клиент причастен к передаче данных. Выход из строя одного клиента может парализовать всю сеть, если не используются специальные переходные соединения.

3. При подключении новых клиентов сеть должна быть кратковременно выключена.

Шинная топология

Такая сеть похожа на центральную линию, к которой подключены сервер и клиент. Шинная топология получила широкое распространение, что прежде всего можно объяснить небольшими потребностями в кабеле и быстрой передачей данных. Так как в такой сети информация передается волнообразно, то концы линии должны иметь специальные заглушки, чтобы не происходило отражения сигналов.

Достоинства:

1. Небольшие затраты на кабели.

2. Клиенты в любое момент времени могут быть установлены или отключены без прерывания работы всей сети.

3. Клиенты могут коммутироваться друг с другом без помощи сервера.

Недостатки:

1. При обрыве кабеля выходит из строя весь участок сети от места разрыва.

2. Возможность несасанкционированного подключения к сети, поскольку для увеличения числа клиентов нет необходимости в прерывании работы сети.

Основной принцип технологии клиент-сервер заключается в разделении функций приложения на три группы:

-ввод и отображение данных (взаимодействие с пользователем);

-прикладные функции, характерные для данной предметной области;

-функции управления ресурсами (файловой системой, базой данных и т.д.).

Поэтому, в любом приложении выделяются следующие компоненты:

- компонент представления данных;

- прикладной компонент;

-компонент управления ресурсом (компонент доступа к информационным ресурсам или менеджер ресурсов).

Связь между компонентами осуществляется по определенным правилам, которые называют «протокол взаимодействия».

Модели взаимодействия клиента и сервера

Обычно выделяют три модели взаимодействия клиента и сервера:

RDA (Remote Data Access) - модель доступа к удаленным данным, в которой компонента представления (пользовательский интерфейс) и прикладная компонента (логика работы программы) совмещены в клиентской части, а

компонента доступа к информационным ресурсам (данным) размещена в серверной части.

DBS (DataBase Server) - модель сервера базы данных, в которой компонента представления размещена в клиентской части, а прикладная компонента и доступ к информационным ресурсам - в серверной.

AS (Application Server) - модель сервера приложений, в которой компонента представления находиться в клиентской части, прикладная компонента - в «сервере приложения», а компонента доступа к информационным ресурсам - в «сервере базы данных».

В RDA-модели коды компонента представления и прикладного компонента совмещены и выполняются на компьютере-клиенте. Последний поддерживает как функции ввода и отображения данных, так и чисто прикладные функции. Доступ к информационным ресурсам обеспечивается, как правило, операторами специального языка (языка SQL, например, если речь идет о базах данных) или вызовами функций специальной библиотеки. Запросы к информационным ресурсам направляются по сети удаленному компьютеру (например, серверу базы данных). Последний обрабатывает и выполняет запросы и возвращает клиенту блоки данных.

DBS-модель строится в предположении, что процесс, выполняемый на компьютере-клиенте, ограничивается функциями представления, в то время как собственно прикладные функции реализованы в хранимых процедурах. Они хранятся непосредственно в базе данных и выполняются на компьютере-сервере базы данных.

В AS-модели процесс, выполняющийся на компьютере-клиенте, отвечает за ввод и отображение данных. Прикладные функции выполняются программами в рамках процессов, функционирующих на компьютере-сервере. Доступ к информационным ресурсам, необходимым для решения прикладных задач, обеспечивается ровно тем же способом, что и в RDA-модели.

1.2. Двух- и трехуровневые системы архитектуры клиент-сервер

Системы с архитектурой клиент-сервер могут быть двух- или трехуровневыми.

Система является двухуровневой, если она построена с использованием набора прикладных клиентских программ, имеющих общий доступ к ресурсам системы и работающих с сервером базы данных. Прикладная программа может при этом размещаться как в клиентской, так и в серверной частях в виде хранимых процедур.

Система является трехуровневой, если она содержит три следующие самостоятельные компоненты:

· интерфейс пользователя, в функции которого входят только отображение (вывод) результатов и взаимодействие с пользователем;

· сервер приложения, в котором сосредоточены все бизнес-функции, правила и хранимые процедуры;

· сервер базы данных, он же менеджер ресурсов.

RDA- и DBS-модели опираются на двухуровневую систему разделения функций. В RDA-модели прикладные функции приданы программе-клиенту, в DBS-модели ответственность за их выполнение берет на себя ядро СУБД. В первом случае прикладной компонент сливается с компонентом представления, во-втором - интегрируется в компонент доступа к информационным ресурсам. Напротив, в AS-модели реализована классическая трехуровневая система разделения функций, где прикладной компонент выделен как важнейший элемент приложения, для его определения используются универсальные механизмы многозадачной операционной системы, и стандартизованы интерфейсы с двумя другими компонентами (компонентом представления и компонентом доступа к информационным ресурсам.)

Основным звеном принятой в AS-модели трехуровневой системы является сервер приложения. В его рамках реализовано несколько прикладных функций, каждая из которых оформлена как сервис и предоставляет некоторые услуги всем программам, которые желают ими воспользоваться. Внешне сервер приложения выглядит как завершенный набор сервисов. С точки зрения внутреннего устройства сервер приложения - это программа, в рамках которой реализованы сервисы.

Любая другая программа, которая пользуется ими, рассматривается как клиент приложения (Application Client - AC). Детали реализации прикладных функций в сервере приложений полностью скрыты от клиента приложения. AC обращается с

запросом к конкретному сервису, но не к AS, то есть серверы приложений обезличены и служат лишь своего рода «рамкой» для оформления сервисов. Запросы, поступающий от АС, выстраиваются в очередь к AS-процессу, который извлекает и передает их для обработки службе в соответствии с приоритетами запросов.

Клиент приложения трактуется более широко, чем компонент представления. Он может поддерживать интерфейс с конечным пользователем (тогда он является компонентом представления), может обеспечивать поступление данных от некоторых устройств, может, наконец, сам по себе быть сервером приложения, предоставляя услуги другим программам. То есть пара «клиент приложения - сервер приложения» динамична: сервер приложения, предоставляя услуги другим программам, сам может обращаться к услугам другого сервера приложения, и в этом момент он рассматривается как клиент приложения.

Серверов приложений может быть несколько, и каждый из них предоставляет определенный набор услуг.

Менеджеры транзакций.

В том случае, когда информационная система объединяет достаточно большое количество различных информационных ресурсов и серверов приложений, встает вопрос об оптимальном управлении всеми ее компонентами. В этом случае используют специализированные средства - менеджеры обработки транзакций (часто их называют просто «менеджеры транзакций»). При этом понятие транзакции расширяется по сравнению с используемым в теории баз данных. В данном случае это любое действие в системе - выдача сообщения, запись в индексный файл, печать отчета и т.д.

В теоретической части курсовой работы была рассмотрена тема «Основные понятия архитектуры клиент-сервер».

В настоящее время такой тип сети, как клиент-сервер, широко применим на многих предприятиях, школах, ВУЗах и т.д. Это значительно облегчает труд работников, т.к. вся информация хранится на едином файловом сервере, а люди могут работать с этой информацией на отдельных рабочих станциях.

Сеть клиент-сервер необходима современному миру. И руководители предприятий стараются совершенствовать работу этой сети, т.к. данная сеть позволяет обрабатывать информацию в едином центре, автоматизировать функции,

производится автономная работа по единому замыслу и многое другое.

2. Практическая часть.

2.1. Архитектура "файл-сервер"

Файл-серверные приложения - приложения, схожие по своей структуре с локальными приложениями и использующие сетевой ресурс для хранения программы и данных.

Функции сервера: хранения данных и кода программы.

Функции клиента: обработка данных происходит исключительно на стороне клиента.

Конечно, основным достоинством данной архитектуры является простота организации. Проектировщики и разработчики информационной системы находятся в привычных и комфортных условиях IBM PC в среде MS-DOS, Windows или какого-либо облегченного варианта Windows Server. Имеются удобные и развитые средства разработки графического пользовательского интерфейса, простые в использовании средства разработки систем баз данных и/или СУБД.

Достоинства такой архитектуры:

· многопользовательский режим работы с данными;

· удобство централизованного управления доступом;

· низкая стоимость разработки;

· высокая скорость разработки;

· невысокая стоимость обновления и изменения ПО.

Недостатки:

· проблемы многопользовательской работы с данными: последовательный доступ, отсутствие гарантии целостности;

· низкая производительность (зависит от производительности сети, сервера, клиента);

· плохая возможность подключения новых клиентов;

· ненадежность системы.

Простое, работающее с небольшими объемами информации и рассчитанное на применение в однопользовательском режиме, файл-серверное приложение можно спроектировать, разработать и отладить очень быстро. Очень часто для небольшой компании для ведения, например, кадрового учета достаточно иметь изолированную систему, работающую на отдельно стоящем PC.

2.3. Архитектура "клиент-сервер"

Клиент-сервер (Client-server) - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением.

Первоначально системы такого уровня базировались на классической двухуровневой клиент-серверной архитектуре (Two-tier architecture). Под клиент-серверным приложением в этом случае понимается информационная система, основанная на использовании серверов баз данных.

Преимуществами данной архитектуры являются:

· возможность, в большинстве случаев, распределить функции вычислительной системы между несколькими независимыми компьютерами в сети;

· все данные хранятся на сервере, который, как правило, защищен гораздо лучше большинства клиентов, а также на сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа;

· поддержка многопользовательской работы;

· гарантия целостности данных.

Недостатки:

· неработоспособность сервера может сделать неработоспособной всю вычислительную сеть;

· администрирование данной системы требует квалифицированного профессионала;

· высокая стоимость оборудования;

· бизнес логика приложений осталась в клиентском ПО.

При проектировании информационной системы, основанной на архитектуре "клиент-сервер", большее внимание следует обращать на грамотность общих решений. Технические средства пилотной версии могут быть минимальными (например, в качестве аппаратной основы сервера баз данных может использоваться одна из рабочих станций). После создания пилотной версии нужно провести дополнительную исследовательскую работу, чтобы выяснить узкие места системы. Только после этого необходимо принимать решение о выборе аппаратуры сервера, которая будет использоваться на практике.

Увеличение масштабов информационной системы не порождает принципиальных проблем. Обычным решением является замена аппаратуры сервера (и, может быть, аппаратуры рабочих станций, если требуется переход к локальному кэшированию баз данных). В любом случае практически не затрагивается прикладная часть информационной системы.

Также данный вид архитектуры называют архитектурой с "толстым" клиентом.

2.1. Многоуровневый "клиент-сервер".

Многоуровневая архитектура клиент-сервер (Multitier architecture) - разновидность архитектуры клиент-сервер, в которой функция обработки данных вынесена на один или несколько отдельных серверов. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов.

Среди многоуровневой архитектуры клиент-сервер наиболее распространена трехуровневая архитектура (трехзвенная архитектура, three-tier), предполагающая

наличие следующих компонентов приложения: клиентское приложение (обычно говорят "тонкий клиент" или терминал), подключенное к серверу приложений, который в свою очередь подключен к серверу базы данных.

Плюсами данной архитектуры являются:

· клиентское ПО не нуждается в администрировании;

· масштабируемость;

· конфигурируемость - изолированность уровней друг от друга позволяет быстро и простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней;

· высокая безопасность;

· высокая надежность;

· низкие требования к скорости канала (сети) между терминалами и сервером приложений;

· низкие требования к производительности и техническим характеристикам терминалов, как следствие снижение их стоимости.

Минусы:

· растет сложность серверной части и, как следствие, затраты на администрирование и обслуживание;

· более высокая сложность создания приложений;

· сложнее в разворачивании и администрировании;

· высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования;

· высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

Критерии	«Файловый сервер»	«Клиент – сервер»		
	FS - модель	RDA-модель	DBS-модель	AS-модель
1	2	3	4	5
Сложность разработки приложений	Низкая	Низкая	Высокая	Высокая
Сложность администрирования	Низкая	Высокая	Высокая	Высокая
Степень защиты данных	Высокая	Низкая	Высокая	Высокая
Требования к характеристикам сервера	Высокие	Низкие	Высокие	Высокие
Трафик, создаваемый в сети	Низкий	Очень высокий	Низкий	Низкий
Сложность обновления ПО	Низкая	Высокая	Низкая	Низкая
Требования к характеристикам сети	Низкие	Очень высокие	Низкие	Низкие
Распределение загрузки	Нет	Есть	Есть	Есть
Требования к характеристикам рабочих станций	-	Очень высокие	Низкие	Низкие
Использование графического интерфейса	-	+	+	+
Использование символьного интерфейса	+	+	+	+

Таблица 1 - Результаты анализа моделей технологий «Файловый сервер» и «Клиент – сервер»

Так как обработка осуществляется в любом месте сети, распределенные вычисления в архитектуре «Клиент–сервер» гарантируют эффективное

масштабирование. Чтобы добиться баланса между клиентом и сервером, компонент приложения должен выполняться на сервере только в том случае, когда централизованная обработка более эффективна. Если логика программы, взаимодействующей с централизованными данными, сосредоточена на той же машине, что и данные, их необязательно передавать по сети, поэтому требования к сетевой среде могут быть снижены.

Таким образом, если предстоит работа с небольшими информационными системами, не требующими графического интерфейса с пользователем, можно выбрать FS - модель. Проблему графического интерфейса легко решает RDA-модель. DBS-модель является хорошим вариантом для СУБД. AS-модель является лучшим вариантом для создания больших информационных систем, а также в случае использования низкоскоростных каналов связи.

ЗАКЛЮЧЕНИЕ

В процессе курсовго исследования была достигнута цель работы **–** «Варианты архитектуры клиент-сервер».

А так же решены следующие задачи:

1) Описана предметная область. Архитектура сети клиент-сервер.

2) Изучены двух и трехуровневые системы архитектуры клиент-сервер.

3)Рассмотрена проектная часть.

4) Изучен Многоуровневый "клиент-сервер

В архитектуру клиент-сервер, достоинствами которой, является то, что вся вычислительная' нагрузка переносится на сервер базы данных, осуществляется высокая защита данных, поддерживается большое количество пользователей и сложных приложений.

Две другие важные особенности, на которые стоит обратить внимание, - способность сервера обеспечивать целостность ссылочных данных и обоюдный контроль завершения транзакции. Ссылочная целостность данных – это механизм, обеспечивающий каждому внешнему ключу соответствующий первичный ключ. Обоюдный контроль завершения транзакций - это гарантия того, что данные не

будут повреждены даже при аппаратном сбое.

Несомненным преимуществом является приближенность данных к процессам вычисления. Практически, все расчеты выполняются на сервере, что увеличивает быстродействие в десятки и сотни раз.

Существуют три основных программных компонента архитектуры клиент-сервер :

-Программное обеспечение конечного пользователя.

-Промежуточное обеспечение.

-Программное обеспечение сервера

Таким образом, любая компьютерная сеть по сути является сетью клиент-сервер. Пользователь, подключивший свой компьютер к Интернет, будет иметь дело с сетью клиент-сервер, и даже если компьютер не имеет выхода в сеть, его программное обеспечение, да и сам он, скорее всего, организованы по схеме клиент-сервер.

Выполнение данной курсовой работы дало мне навык и понимание работы различной архитектуры клиент-сервер .

Список использованных источников

1. Грофф Дж., Вайнберг П. Энциклопедия SQL. 3-е изд СПб.: Питер, 2003
2. Конноли Т., Бэгг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика., 2013.
3. Дейт К.Дж. Введение в системы баз данных /К.Дж. Дейт - Москва: ДМК, 2000.
4. Конноли Т. Базы данных. Проектирование, реализация и сопровождение /Т. Конноли, К. Бегг. - Москва: Вильяме, 2003.
5. Информатика. Учебное пособие / под ред. Б.Е. Одинцова, А.Н. Романова – 2-ое издание, перераб.и доп. – М.: Вузовский учебник, ИНФРА-М, 2012.
6. Стандарты IDEF. Электронный ресурс. Режим доступа www.igef.ru.
7. Маклаков С. В. BPwin и ERwin: CASE-средства для разработки.
8. Информатика: Базовый курс: учебное пособие / под ред. С.В. Симоновича. – СПб.: Питер, 2011.
9. Агальцов В.П., Титов В.М. Информатика для экономистов: учебник. – М.: Форум : ИНФРА-М, 2011.

10. ЭБС ООО «Издательский Дом ИНФРА-М» (доступ через интернет-репозиторий образовательных ресурсов ВЗФЭИ). – URL: http://repository.vzfei.ru. Доступ по логину и паролю.
11. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Алгоритмы: построение и анализ.
12. Компьютерная обучающая программа по дисциплине «Информатика» / А.Н. Романов, В.С. Торопцов, Д.Б. Григорович, Л.А. Галкина, А.Ю. Артемьев, Н.И. Лобова, К.Е. Михайлов, Г.А. Жуков, О.Е. Кричевская, С.В. Ясеновский, Л.А. Вдовенко, Б.Е. Одинцов, Г.А. Титоренко, Г.Д. Савичев, В.И. Гусев, С.Е. Смирнов, В.И. Суворова, Г.В. Федорова, Г.Б. Коняшина. – М.: ВЗФЭИ, 2000. Дата обновления 24.11.2010. – URL: http://repository. vzfei.ru. Доступ по логину и паролю.
13. Видео лекция курсы Гарварда и Еля SC50 за 2015г.