ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Лекция 9. Классические архитектуры ИС

Курс лекций

МТУСИ

Общие положения

Суммируя сказанное на предыдущей лекции, определим **архитектуру ИС,** как концепцию, определяющую модель, структуру, выполняемые функции и взаимосвязь компонентов ИС.

Единая классификация архитектур отсутствует. Различные авторы классифицируют ИС по-разному, а в архитектуре любой конкретной ИС часто можно найти элементы нескольких «чистых» архитектур.

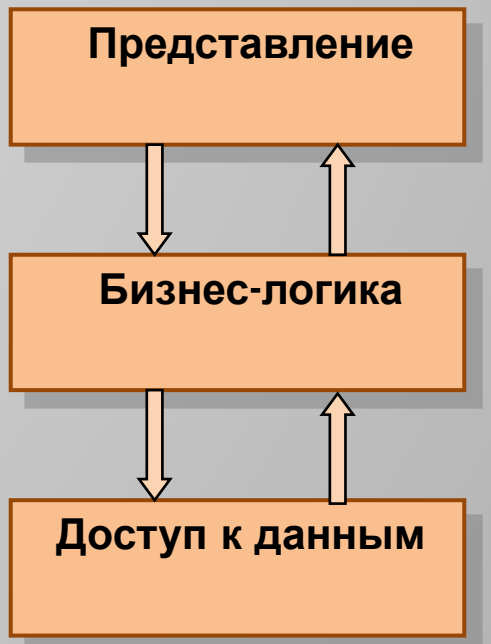
Если собрать все типы архитектур ИС из разных источников то можно выделить следующие:

1. Централизованная архитектура;
2. Архитектура «файл-сервер»;
3. Двухслойная архитектура «клиент-сервер»;
4. Архитектура «клиент-сервер» 2,5 слоя;
5. Трехслойная архитектура «клиент-сервер»;
6. Архитектура Веб-приложений (на основе технологии Intranet);
7. Архитектура распределенных систем;

Условно назовем архитектуры 1-5 классическими, поскольку они возникли раньше других. Они будут предметом данной лекции.

Классификация архитектур ИС, которые мы условно назвали классическими, основывается на разделении компонент ИС по выполняемым функциям на три уровня (слоя, звена – от англ. «tier») - уровни представления, бизнес логики и доступа к данным.

- **Уровень представления** – все, что связано с взаимодействием пользователя и рабочей станции (нажатие кнопок, движение мыши, вывод на монитор изображений и результатов поиска и т.д.).
- **Уровень бизнес логики (приложений)** – реакция приложений на действия пользователя или на внутренние события, правила обработки данных (формулы расчёта выплат по ссудам в финансовых ИС, автоматизированная отправка сообщений руководителю проекта по окончании выполнения заданий подчиненными в ИС управления проектами, отказ от отеля при отмене рейса авиакомпанией в ИС туристического бизнеса и т. д.).
- **Уровень доступа к данным** – все, что относится к данным (хранение, выборка, модификация, удаление).



Централизованная архитектура

Архитектура времен 70-х и 80-х годов, на базе мейнфреймов (например, БЭСМ-6, IBM-360/370 или их отечественных аналогов серии ЕС ЭВМ), либо на базе мини-ЭВМ (PDP-11 или СМ-4).

Характерная особенность – полная "неинтеллектуальность" терминалов. Их работой управляет хост-ЭВМ (от англ. host — хозяин, принимающий гостей, — любое устройство, предоставляющее сервисы формата «клиент-сервер» в режиме сервера; компьютер, сервер в сети, IP-адрес, сетевой интерфейс устройства, подключённого к IP-сети).

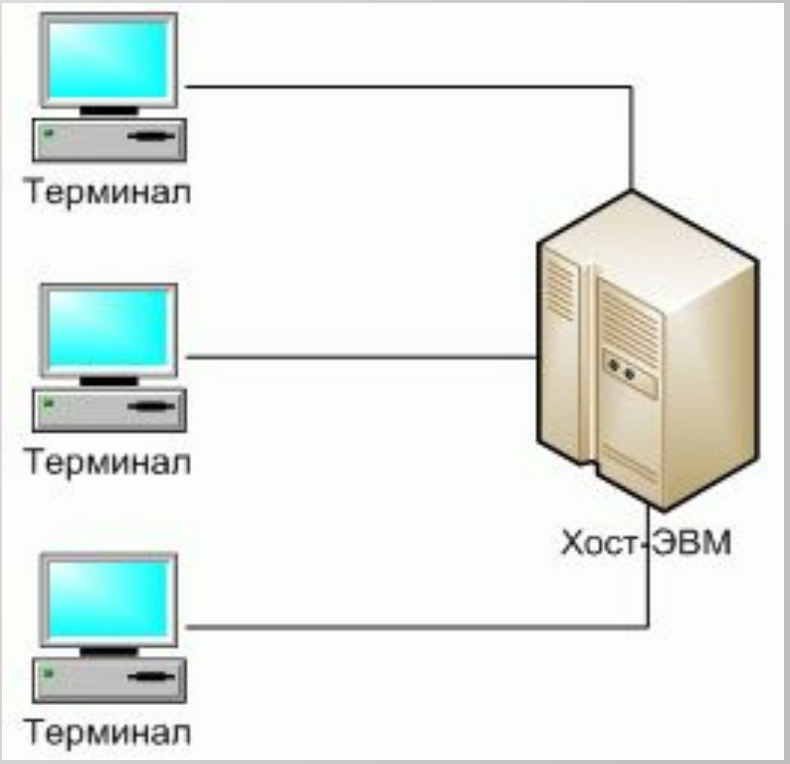
Современные супер-ЭВМ также можно отнести к ЭВМ с централизованной архитектурой.

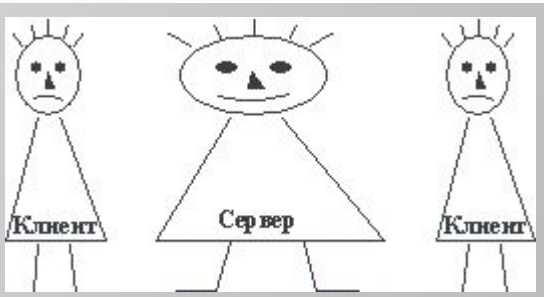
Достоинства:

- пользователи совместно используют дорогие ресурсы ЭВМ и дорогие периферийные устройства;
- централизация ресурсов и оборудования облегчает обслуживание и эксплуатацию вычислительной системы;
- отсутствует необходимость администрирования рабочих мест пользователей;

Недостатки:

- полная зависимость пользователя от администратора хост-ЭВМ;
- все используемое программное обеспечение является коллективным.

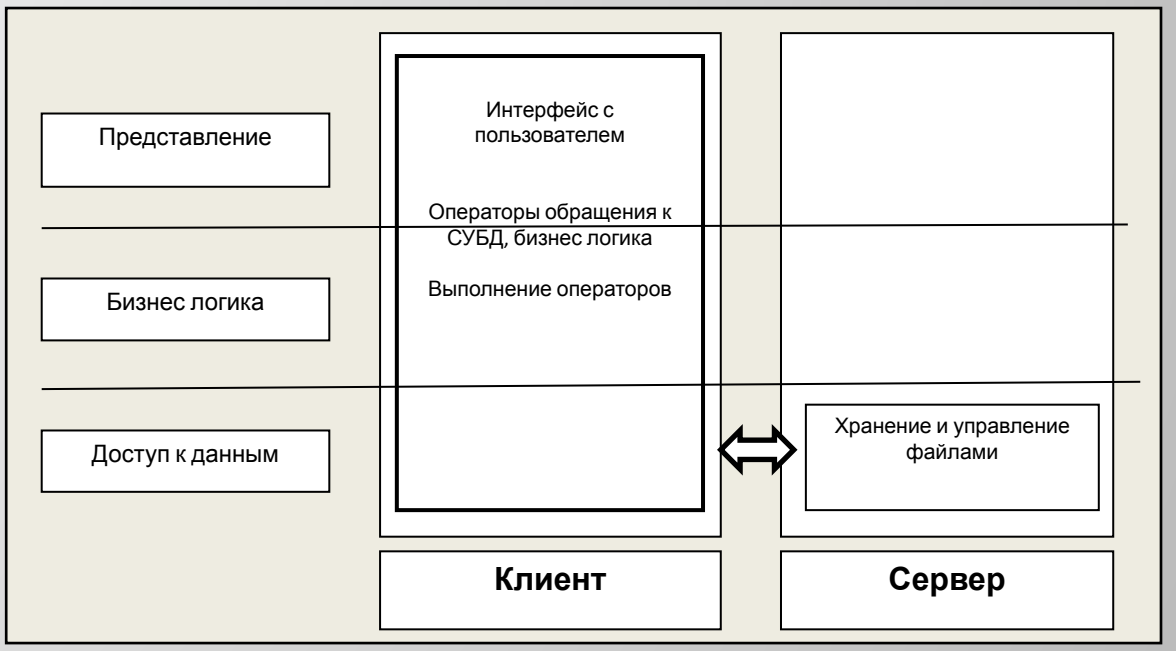




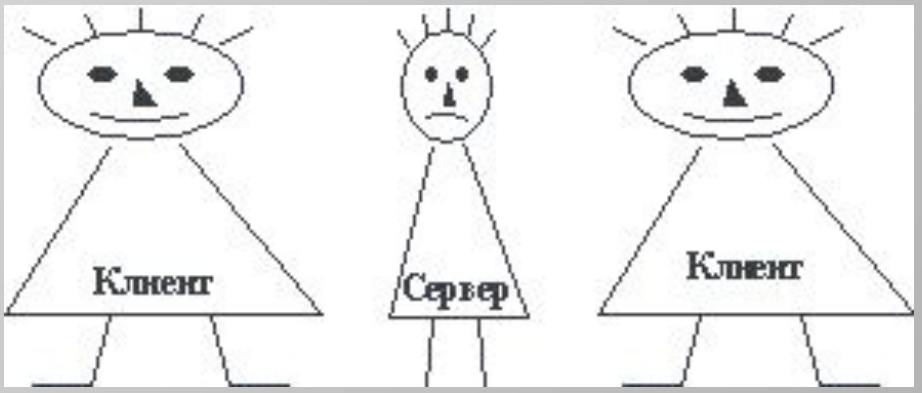
Использование централизованной архитектуры оправдано, если хост-ЭВМ очень дорогая, например, супер-ЭВМ или на сервере хранится очень важная информация.

Файл-серверная архитектура

Компоненты ИС на разных компьютерах, взаимодействуют только за счет наличия общего хранилища файлов, которое размещается на файл-сервере. На компьютерах дублируются не только прикладные программы, но и средства управления базами данных. Файл-сервер - просто расширение дисковой памяти сразу всех ПК.



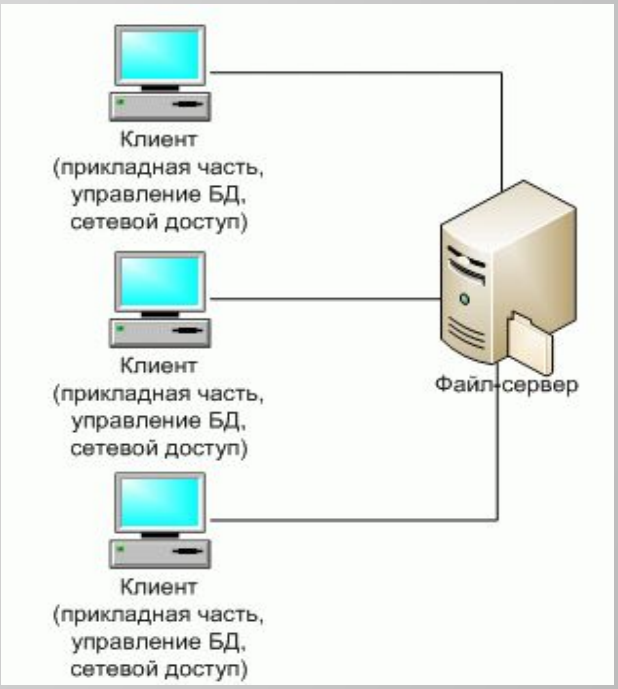
Организация ИС на основе использования выделенных файл-серверов все еще является распространенной в связи с наличием в сетях большого количества гетерогенных клиентов - персональных компьютеров достаточно высокой мощности от разных производителей и с различными ОС



Файл-серверная архитектура

Достоинства:

1. Простота организации, низкая стоимость и высокая скорость разработки.
2. Наличие развитых средств разработки интерфейса, систем БД и СУБД.
3. Многопользовательский режим работы с данными.
4. Удобство централизованного управления доступом.



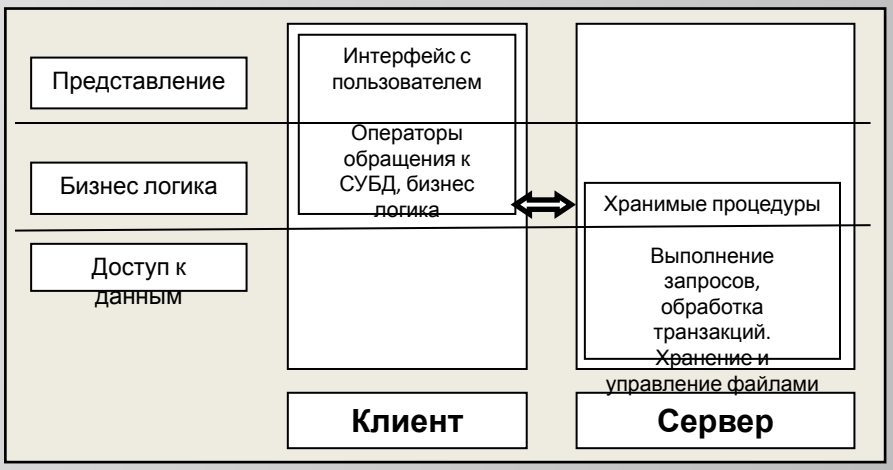
Недостатки:

1. Перегрузка трафика (для выборки полезных данных необходимо просмотреть на стороне клиента весь соответствующий файл целиком).
2. Децентрализованное решение проблем целостности и согласованности данных, одновременного доступа к ним, что снижает надежность приложения.
3. Слабые возможности расширения, необходимость переустановки ПО на клиентских местах.
4. Низкая производительность, зависящая от производительности сети, сервера, клиента.

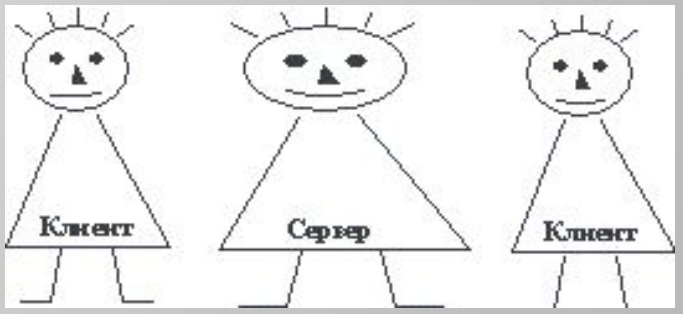
Клиент-серверная двухслойная архитектура

На выделенном сервере хранится не только сама БД, но и некоторая часть приложений, а на стороне клиента выполняются не полностью все приложения, а только их некоторая часть. При этом клиентская часть приложения взаимодействует с клиентской частью программного обеспечения управления БД.

Клиентские программы манипулируют данными на уровне слоя бизнес-логики.



На стороне клиента выполняется часть приложения, в которую обязательно входят компоненты, поддерживающие интерфейс с конечным пользователем, производящие отчеты, выполняющие другие специфичные для приложения функции.

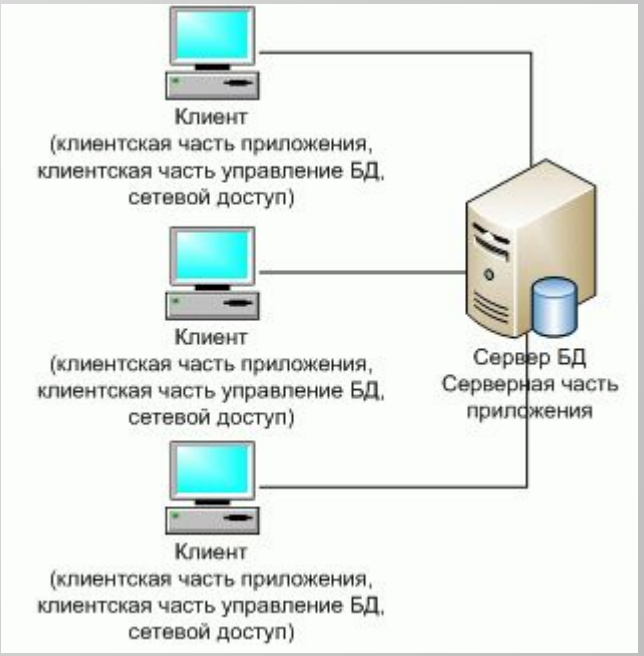


Клиентская часть приложения взаимодействует с клиентской частью программного обеспечения управления базами данных. Часто для эффективной работы конкретного клиента ИС требуется небольшая часть общей БД. Поэтому целесообразно создание локального кэша общей БД на стороне каждого клиента. При этом необходимо обеспечить согласованность кэшей и общей БД.

Клиент-серверная двухслойная архитектура

Достоинства:

1. Полная поддержка многопользовательской работы.
2. Гарантия целостности данных.
3. Возможность, в большинстве случаев, распределить функции вычислительной системы между несколькими независимыми компьютерами в сети;
4. Все данные хранятся на сервере, который защищен гораздо лучше клиентов; на сервере проще обеспечить контроль доступа к данным клиентов с соответствующими полномочиями.



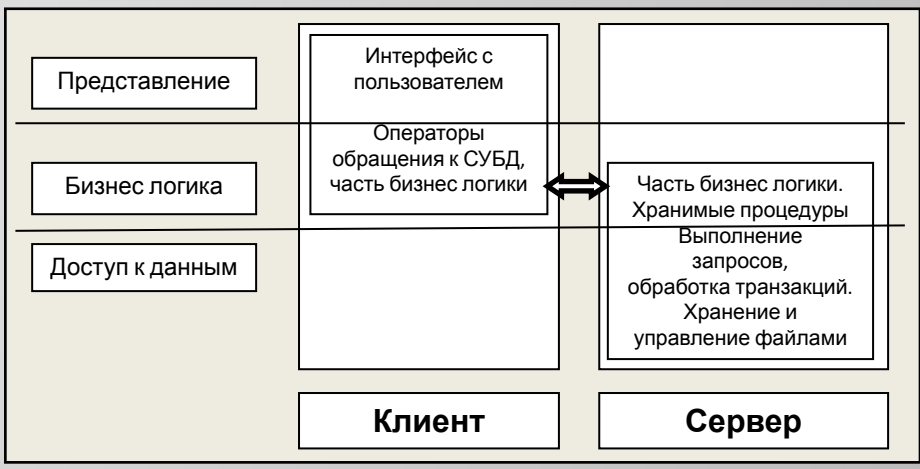
Недостатки:

1. При изменении бизнес-логики, надо обновлять пользовательское ПО на каждом клиенте.
2. Все еще высокие требования к пропускной способности коммуникационных каналов.
3. Слабая защита данных от взлома недобросовестными пользователями системы.
4. Сложность администрирования и настройки рабочих мест пользователей.
5. Необходимость использования мощных ПК на клиентских местах.
6. Высокая сложность разработки из-за того, что бизнес-логика и интерфейс находятся в одной программе.

Клиент-серверная архитектура, переходная к трехслойной(2.5 слоя)

Носит промежуточный характер, поскольку часть бизнес-функций переносится на сервер, но физически такие системы состоят по-прежнему из двух компонентов,

На клиентской рабочей станции все равно остается часть бизнес логики, поскольку не удается написать всю бизнес-логику приложения на не предназначенных для этого встроенных языках СУБД.



Достоинства:

1. Не требуются высокоскоростные каналы связи, так как по сети передаются уже готовые результаты работы с данными - почти полностью эта работа производится на стороне сервера.
2. Улучшается защита информации, поскольку пользователи имеют доступ к функциям системы, а не к ее данным.

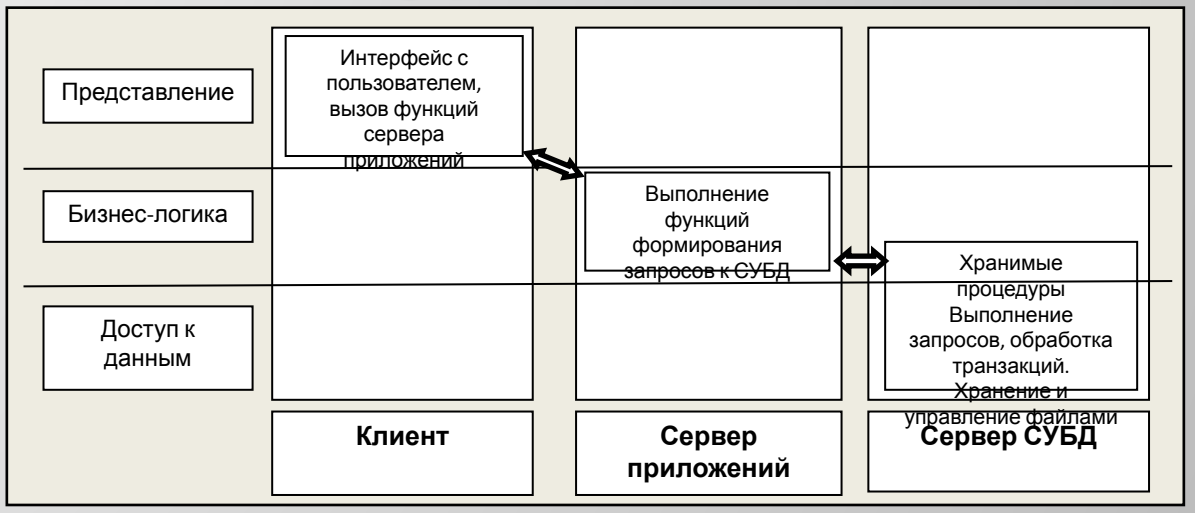
Недостатки:

1. Ограниченная масштабируемость.
2. Зависимость от программной платформы.
3. Невысокое быстродействие за счет использования встроенных в СУБД языков для написания ИС.

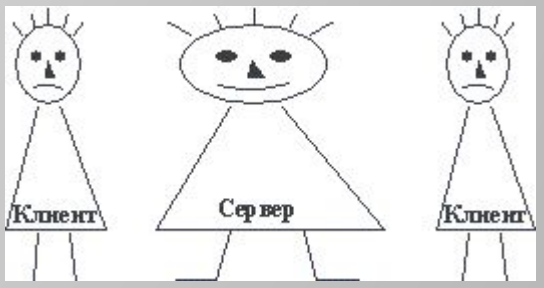
Клиент-серверная трехслойная архитектура

Особенности:

- кроме клиентской части системы и сервера базы данных, вводится промежуточный сервер приложений. На стороне клиента выполняются только интерфейсные действия, а логика сосредоточена в сервере приложений,



- каждый из слоев ИС реализуется на своих аппаратных средствах: слой представления – на рабочих станциях с «тонким» клиентом, слой бизнес логики – на сервере приложений и слой доступа к данным – на сервере БД.



- в качестве клиентских интерфейсных программ широко применяются стандартные интернет-браузеры.
- в случае, если для работы некоторых клиентский мест обеспечивается относительно небольшой частью БД, используется локальный кэш общей БД на стороне каждого клиента, где размещается информация, запрашиваемая с наибольшей вероятностью. Это частичная репликация данных.

Сходство с мэйнфреймами

Сходство:

в роли мэйнфрейма – сервера приложений и БД, в роли терминалов – клиентское место (персональный компьютер, обеспечивающий графический с интерфейс пользователя) – «тонкий» клиент.

Различие:

- иной технологический уровень при сохранении только внешних черт;
- широкое и эффективное применение стандартных интернет-браузеров (интерфейсного программного обеспечения для запроса Web-страниц преимущественно из Сети), в качестве клиентских интерфейсных программ;
- обязательное использование СУБД со всеми их преимуществами;
- использование специализированных программных языков при написании программ для серверной части ИС.

Достоинства и недостатки трехслойной архитектуры

Достоинства:

1. Упрощается модернизация ИС, поскольку «тонкого» (в том числе, удаленного) клиента легче переустановить.
2. Повышается также информационная безопасность и надежность ИС.
3. Наилучшая среди всех архитектур горизонтальная и вертикальная масштабируемость; горизонтальная - за счет того, что число клиентов может быть неограниченно увеличено; вертикальная – за счет того, что при добавлении новой функции меняется только ПО сервера приложений.
4. Между клиентом и сервером приложения передается минимально необходимый поток данных - аргументы вызываемых функций и возвращаемые от них значения.
5. Сервер приложения ИС может быть запущен в одном или нескольких экземплярах на одном или нескольких компьютерах, что позволяет эффективно и безопасно использовать вычислительные мощности организации.
6. Дешевый трафик между сервером приложений и БД; при запуске сервера приложений и БД на одной машине сетевой трафик сводится к нулю.
7. Снижение нагрузки на сервер БД по сравнению с 2.5-слойной схемой, а значит и повышение скорости работы системы в целом.

Недостатки:

1. Архитектура "клиент-сервер" требует более мощных и, следовательно, дорогих аппаратных средств, чем архитектура "файл-сервер".
2. Выше расходы на администрирование и обслуживание серверной части.