**Цели и задачи:**

* **Цель:**
* Изучить архитектуру “Клиент-Сервер”.
* Задачи:
* Узнать принцип работы архитектуры.
* Знать типы клиент-серверной архитектуры.
* Изучить характеристики архитектуры “клиент-сервер”.
* Понять практические применения архитектуры “клиент-сервер”.
* Узнать преимущества и недостатки архитектуры “клиент-сервер”.

Архитектура «Клиент-Сервер» (также используются термины «сеть Клиент-Сервер» или «модель Клиент-Сервер») предусматривает разделение процессов предоставление услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняют свои задачи независимо от других. В архитектуре «Клиент-Сервер» несколько компьютеров-клиентов (удалённые системы) посылают запросы и получают услуги от централизованной служебной машины – сервера, которая также может называться хост-системой. Клиентская машина предоставляет пользователю т.н. «дружественный интерфейс» (user-friendly interface), чтобы облегчить его взаимодействие с сервером.

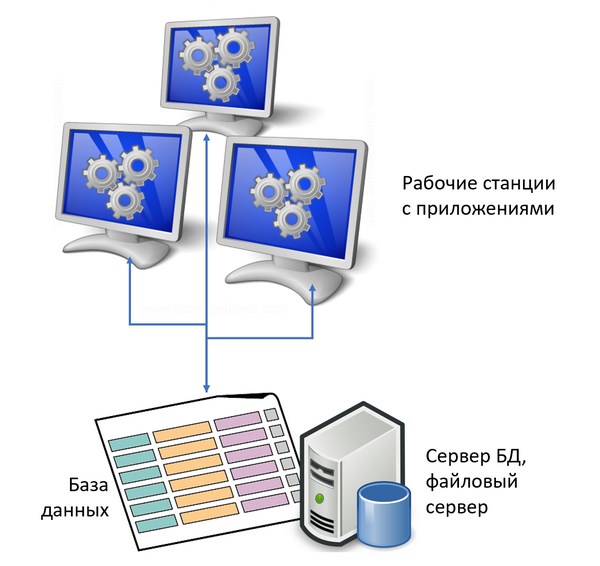
**Типы клиент-серверной архитектуры**

Архитектуру «клиент-сервер» принято разделять на три класса: одно-, двух- и трёхуровневую. Однако, нельзя сказать, что в вопросе о таком разделении в сообществе ИТ-специалистов существует полный консенсус. Многие называют одноуровневую архитектуру двухуровневой и наоборот, то же можно сказать о соотношении двух- и трёхуровневой архитектур.

Постараемся внести ясность в этот вопрос.

**Одноуровневая архитектура (1-Tier)**

Одноуровневая архитектура «клиент-сервер» (1-Tier) – такая, где все прикладные программы рассредоточены по рабочим станциям, которые обращаются к общему [серверу баз данных](https://itelon.ru/solution/server_mssql/) или к общему [файловому серверу](https://itelon.ru/solution/server_file/). Никаких прикладных программ сервер при этом не исполняет, только предоставляет данные.

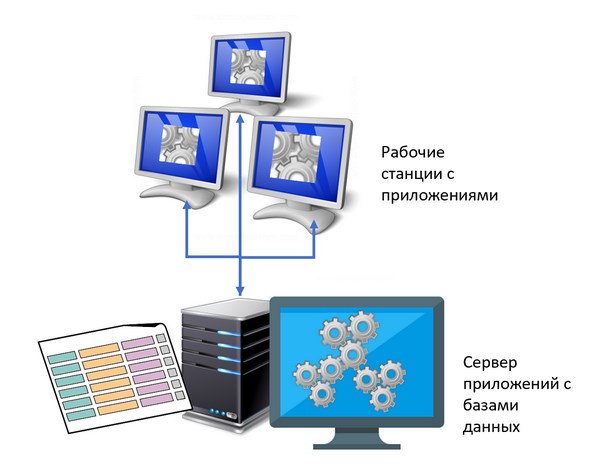


*Рис. 2. Одноуровневая архитектура «клиент-сервер» (1-Tier).*

В целом, такая архитектура очень надёжна, однако, ей сложно управлять, поскольку в каждой рабочей станции данные будут присутствовать в разных вариантах. Поэтому возникает проблема их синхронизации на отдельных машинах. В общем, как можно видеть из рисунка, в этой архитектуре просматривается ещё один уровень – базы данных, что даёт повод во многих случаях называть её двухуровневой.

**Двухуровневая архитектура (2-Tier)**

К двухуровневой архитектуре «клиент-сервер» следует относить такую, в которой прикладные программы сосредоточены на сервере приложений (Application Server), например, [сервере 1С](https://itelon.ru/solution/server_1c/) или [сервере CRM](https://itelon.ru/solution/server_crm/), а в рабочих станциях находятся программы-клиенты, которые предоставляют для пользователей интерфейс для работы с приложениями на общем сервере.



*Рис. 3. Двухуровневая архитектура «клиент-сервер» (2-Tier).*

Такая архитектура представляется наиболее логичной для архитектуры «клиент-сервер». В ней, однако, можно выделить два варианта. Когда общие данные хранятся на сервере, а логика их обработки и бизнес-данные хранятся на клиентской машине, то такая архитектура носит название “fat client thin server” (толстый клиент, тонкий сервер). Когда не только данные, но и логика их обработки и бизнес-данные хранятся на сервере, то это называется “thin client fat server” (тонкий клиент, толстый сервер). Такая архитектура послужила прообразом облачных вычислений (Cloud Computing).

Преимущества двухуровневой архитектуры:

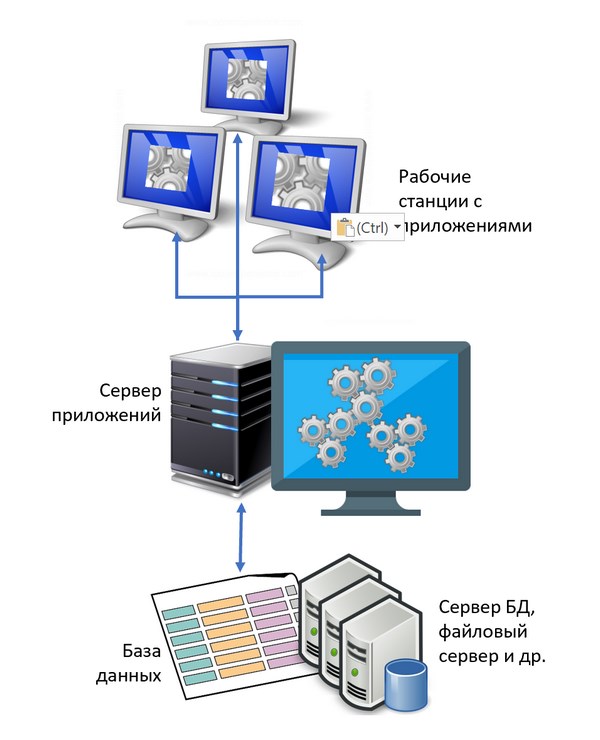
* Легко конфигурировать и модифицировать приложения;
* Пользователю обычно легко работать в такой среде;
* Хорошая производительность и масштабируемость.

Однако, у двухуровневой архитектуры есть и ограничения:

* Производительность может падать при увеличении числа пользователей;
* Потенциальные проблемы с безопасностью, поскольку все данные и программы находятся на центральном сервере;
* Все клиенты зависимы от базы данных одного производителя;

**Трёхуровневая архитектура (3-Tier)**

В трёхуровневой архитектуре сервер баз данных, файловый сервер и другие представляют собой отдельный уровень, результаты работы которого использует сервер приложений. Логика данных и бизнес-логика находятся в сервере приложений. Все обращения клиентов к базе данных происходят через промежуточное программное обеспечение (middleware), которое находится на сервере приложений. Вследствие этого, повышается гибкость работы и производительность.



*Рис. 4. Трёхуровневая архитектура «клиент-сервер» (3-Tier).*

Преимущества трёхуровневой архитектуры:

* Целостность данных;
* Более высокая безопасность, по сравнению с двухуровневой архитектурой;
* Защищённость базы данных от несанкционированного проникновения.

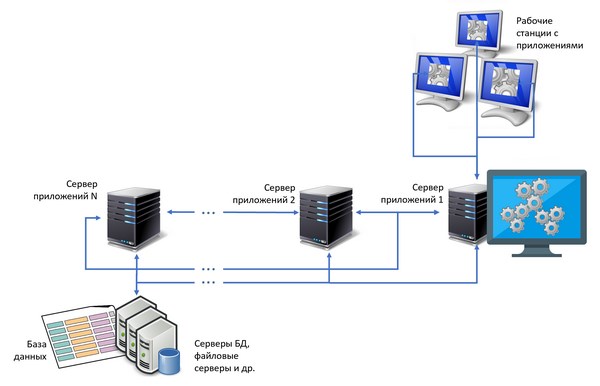
Ограничения:

* Более сложная структура коммуникаций между клиентов и сервером, поскольку в нём также находится middleware.

**Многоуровневая архитектура (N-Tier)**

В отдельный класс архитектуры «клиент-сервер» можно вынести многоуровневую архитектуру, в которой несколько серверов приложений используют результаты работы друг друга, а также данные от различных серверов баз данных, файловых серверов и других видов серверов.

По сути, предыдущий вариант, трёхуровневая архитектура – не более, чем частный случай многоуровневой архитектуры.



*Рис. 5. Многоуровневая архитектура «клиент-сервер» (N-Tier).*

Преимуществом многоуровневой архитектуры является гибкость предоставления услуг, которые могут являться комбинацией работы различных приложений серверов разных уровней и элементов этих приложений.

Очевидным недостатком является сложность, многокомпонентность такой архитектуры.

**Характеристики архитектуры «клиент-сервер»**

* **Асимметричность протоколов**. Между клиентами и сервером существуют отношения «один ко многим». Инициатором диалога с сервером обычно является клиент.
* **Инкапсуляция услуг**. После получения запроса на услугу от клиента, сервер решает, как должна быть выполнена данная услуга. Модификация («апгрейд») сервера может производиться без влияния на работу клиентов, поскольку это не влияет на опубликованный интерфейс взаимодействия между ними. Иными словами, максимум, что может при этом почувствовать пользователь – незначительная задержка отклика сервера в течение небольшого времени апгрейда.
* **Целостность**. Программы и общие данные для сервера управляются централизованно, что снижает стоимость обслуживания и защищает целостность данных. В то же время, данные клиентов остаются персонифицированными и независимыми.
* **Местная прозрачность**. Сервер – это программный процесс, который может исполняться на той же машине, что и клиент, либо на другой машине, подключенной по сети. Программное обеспечение «клиент-сервер» обычно скрывает местоположение сервера от клиентов, перенаправляя запрос на услуги через сеть.
* **Обмен на основе сообщений**. Клиенты и сервер являются нежёстко связанными («loosely-coupled») процессами, которые обмениваются сообщениями: запросами на услуги и ответами на них.
* **Модульный дизайн, способный к расширению**. Модульный дизайн программной платформы «клиент-сервер» придаёт ей устойчивость к отказам, то есть, отказ в каком-то модуле не вызывает отказа всего приложения. В такой системе, один или больше серверов могут отказать без остановки всей системы в целом, до тех пор, пока услуги отказавшего сервера могут быть предоставлены с резервного сервера. Другое преимущество модульности в том, что приложение «клиент-сервер» может автоматически реагировать на повышение или понижение нагрузки на систему, путём добавления или отключения услуг или серверов.
* **Независимость от платформы**. Идеальное приложение «клиент-сервер» не зависит от платформ оборудования или операционной системы. Клиенты и серверы могут развёртываться на различных аппаратных платформах и разных операционных системах.
* **Масштабируемость**. Системы «клиент-сервер» могут масштабироваться как горизонтально (по числу серверов и клиентов), так и вертикально (по производительности и спектру услуг).
* **Разделение функционала**. Система «клиент-сервер» — это соотношение между процессами, работающими на одной или на разных машинах. Сервер – это процесс предоставления услуг. Клиент – это потребитель услуг.
* **Общее использование ресурсов**. Один сервер может предоставлять услуги множеству клиентов одновременно, и регулировать их доступ к совместно используемым ресурсам.

**Практические применения архитектуры «клиент-сервер»**

Архитектуры «клиент-сервер» - один из основных принципов работы сети Интернет. Любой веб-сайт, или приложение в Интернет работает на сервере, а его пользователи являются клиентами. Социальные сети (Фейсбук, ВК и пр.), сайты электронной коммерции (Amazon, Озон и др.) , мобильные приложения (Instagram и т.д.), устройства Интернета вещей (умные колонки или смарт-часы) работают на основе клиент-серверной архитектуры.

Хорошим примером работы системы «клиент-сервер» является автомобильный навигатор. Приложение навигации на сервере собирает данные с многих смартфонов пользователей, на которых установлены клиенты приложения. Кроме того, приложение навигации использует ещё и данные с сервера базы данных – геоинформационной системы, который предоставляет данные, например, о текущих ремонтах дорог, о появлении новых дорог и пр. Данные со многих клиентов (местоположение, скорость) обрабатывается сервером навигации и выдаётся на смартфоны пользователей в виде информации о средней скорости движения по тому или иному участку маршрута.

Практически любая корпоративная сеть или ИТ-система предприятия, как правило, строится по архитектуре «клиент-сервер». В небольших сетях (3-5 компьютеров в компании) функции сервера может выполнять один из рабочих компьютеров. Если число машин в организации более 10, то лучше сделать выделенный сервер ([почтовый сервер](https://itelon.ru/solution/server_mail/), приложений, баз данных и пр.), который будет заниматься обслуживанием клиентов – компьютеров и телефонов сотрудников организации.

В домашних сетях архитектура «клиент-сервер» тоже используется довольно часто. Например, в домашнюю сеть могут быть объединены компьютеры членов семьи, один из которых выполняет функции сервера. В домашнюю сеть также могут быть включены такие устройства, как умные колонки, умные домашние устройства (пылесосы-роботы, фотоаппараты, DVD-плееры и пр.), а также «умные» счётчики (вода, электричество) и т.д. Тогда в системе управления сервера, будут видны все параметры, данные и медифайлы (музыка, видео, фото), а также «умные устройства».

**Преимущества и недостатки архитектуры «клиент-сервер»**

К преимуществам архитектуры «клиент-сервер» можно отнести:

* **Централизованность**, поскольку все данные и управление сосредоточены в центральном сервере;
* **Информационная безопасность**, поскольку ресурсы общего пользования администрируются централизованно;
* **Производительность**, использование выделенного сервера повышает скорость работы ресурсов общего пользования;
* **Масштабируемость**, количество клиентов и серверов можно увеличивать независимо друг от друга.

К недостаткам архитектуры «клиент-сервер» следует отнести:

* **Перегрузку трафика в сети**, что является главной проблемой в сетях «клиент-сервер». Когда большое число клиентов одновременно запрашивают одну услугу на сервере, то число запросов может создать перегрузку в сети;
* **Наличие единой точки отказа в небольших сетях с одним сервером.**Если он отказывает, все клиенты остаются без обслуживания;
* **Превышение пределов ресурсов сервера,**когда новые клиенты, запрашивающие услугу, остаются без обслуживания. В таких случаях, требуется срочное расширение ресурсов сервера;
* Иногда клиентские программы могут не работать на терминалах пользователей, если не установлены соответствующие драйверы. Например, пользователь посылает запрос на печать документа, а на сервере нет подходящего драйвера для печати данного формата документа на определённом принтере.

**Заключение**

В настоящее время можно встретить термин Serverless Architecture, т.н. «бессерверная архитектура». Однако, по сути, она представляет собой процесс получения функций сервера в виде облачной услуги. То есть, серверы в облаке тоже есть, но для конечного пользователя они не видны, и он получает их сервисы в виде абстрактной «функции как услуги» FaaS (Function as a Service).

Архитектура «клиент-сервер» является основой большинства корпоративных сетей и берёт свое начало от самых первых вычислительных машин, т.н. «мэйнфреймов». [Программное обеспечение](https://itelon.ru/catalog/oborudovanie/software/) для локальных компьютерных сетей, подавляющее большинство которых основано на архитектуре «клиент-сервер», начало создаваться около 50 лет назад.

Дальнейшее развитие информационных технологий также будет происходить в значительной степени с использованием архитектуры «клиент-сервер».