

Лекция № 1. Модель клиент-сервер

УЧЕБНЫЕ ВОПРОСЫ

1. Клиенты и серверы
2. Разделение приложений по уровням
3. Варианты архитектуры клиент-сервер

В базовой модели **клиент-сервер** все процессы в распределенных системах делятся на две возможно перекрывающиеся группы:

- Процессы, реализующие некоторую службу, например службу файловой системы или базы данных, называются **серверами** (servers).
- Процессы, запрашивающие службы у серверов путем отправки запроса и последующего ожидания ответа от сервера, называются **клиентами** (clients).

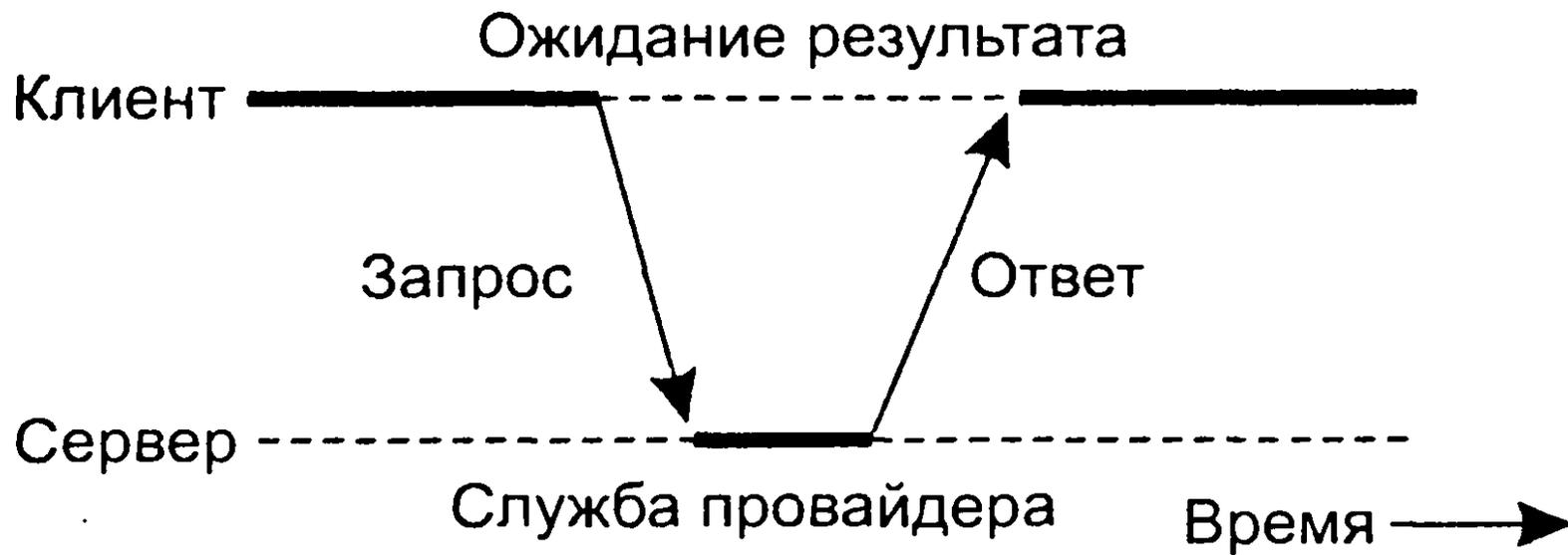


Рис. 1. Обобщенное взаимодействие между клиентом и сервером

Алгоритм взаимодействия посредством простого протокола, не требующего установления соединения:

1. Клиент, запрашивая службу, облакает свой запрос в форму сообщения с указанием в нем службы, которой он желает воспользоваться, и необходимых для этого исходных данных.
2. Затем сообщение посылается серверу.
3. Сервер, в свою очередь, постоянно ожидает входящего сообщения, получив его, обрабатывает, упаковывает результат обработки в ответное сообщение и отправляет его клиенту.

Алгоритм взаимодействия посредством надежного протокола с установкой соединения:

1. Клиент запрашивает службу, до отправки запроса серверу он должен установить с ним соединение.
2. Сервер использует для отправки ответного сообщения то же самое соединение,
3. Соединение разрывается.

Приложения типа клиент-сервер разделяются на три уровня:

1. уровень пользовательского интерфейса;
2. уровень обработки;
3. уровень данных.

Уровень пользовательского интерфейса обычно реализуется на **клиентах**. Этот уровень содержит программы, посредством которых пользователь может взаимодействовать с приложением.

- 1. Вариант 1.** Пользовательский интерфейс содержит только символьный (не графического) дисплей.
- 2. Вариант 2.** Имеется как минимум графический дисплей, на котором можно задействовать всплывающие или выпадающие меню и множество управляющих элементов, доступных для мыши или клавиатуры.
- 3. Вариант 3.** Пользовательские интерфейсы поддерживают совместную работу приложений через единственное графическое окно и в ходе действий пользователя обеспечивают через это окно обмен данными.

Вопрос №2

Уровень обработки реализует основную функциональность приложения.

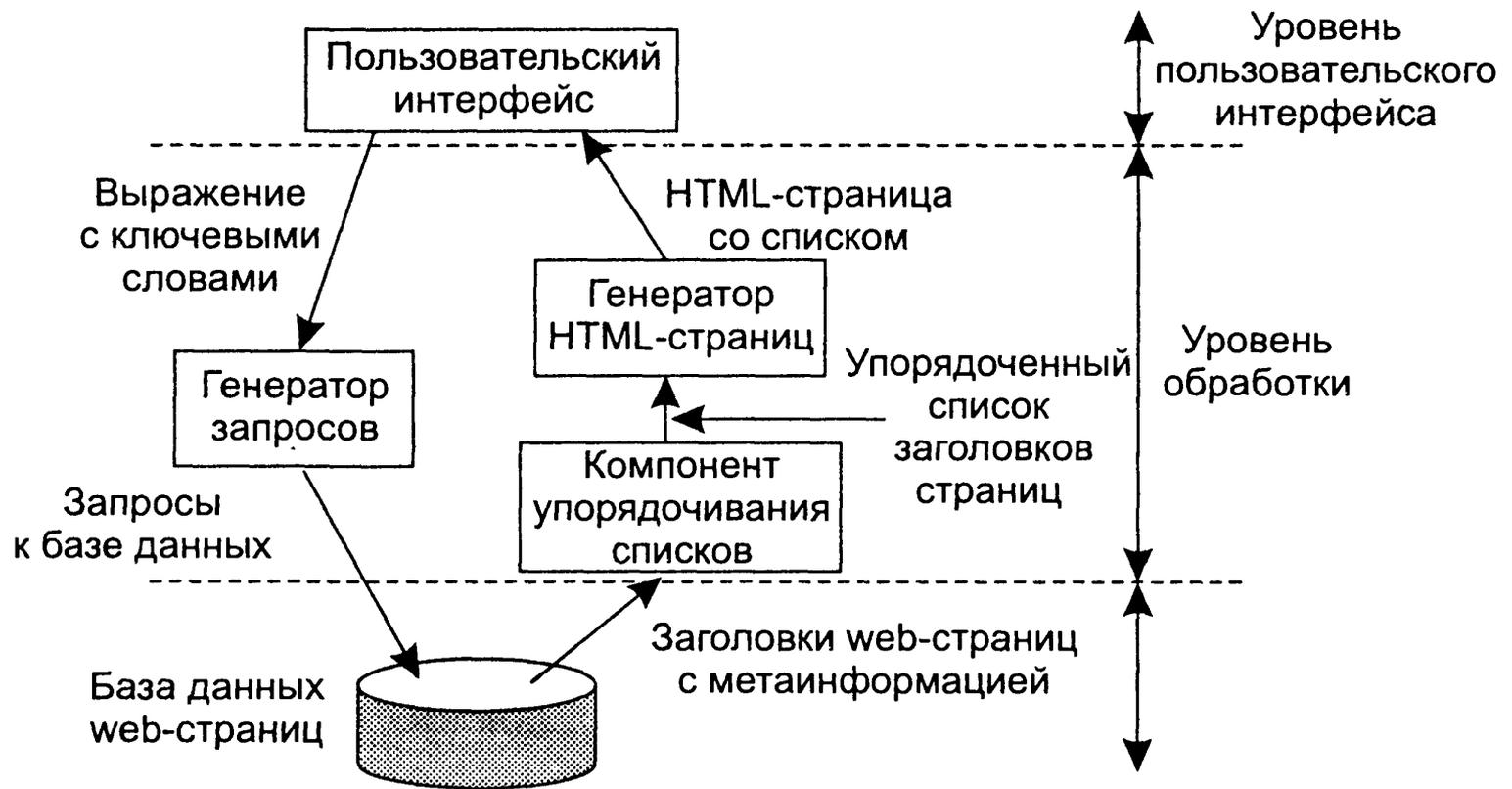


Рис. 2. Обобщенная организация трехуровневой поисковой машины для Интернета

Уровень данных в модели клиент-сервер содержит программы, которые предоставляют данные обрабатывающим их приложениям.

Специфическим свойством этого уровня является требование *сохранности (persistence)*.

Кроме простого хранения информации уровень данных обычно также отвечает за *поддержание целостности данных* для различных приложений.

Обычно **уровень данных** организуется в форме **реляционной базы данных**. Ключевым здесь является независимость данных.

Данные организуются независимо от приложений так, чтобы изменения в организации данных не влияли на приложения, а приложения не оказывали влияния на организацию данных.

В тех случаях, когда операции с данными значительно проще выразить в понятиях работы с объектами, уровень данных реализуют средствами *объектно-ориентированных баз данных*.

Подобные базы данных не только поддерживают организацию сложных данных в форме объектов, но и хранят реализации операций над этими объектами.

Простейшая организация предполагает наличие всего **двух** типов машин (**двухзвенная архитектура**).

1. *Клиентские машины*, на которых имеются программы, реализующие только пользовательский интерфейс или его часть.

2. *Серверы*, реализующие все остальное, то есть уровни обработки и данных.

Проблема подобной организации состоит в том, что на самом деле **система не является распределенной**: все происходит на сервере, а клиент представляет собой не что иное, как простой терминал.

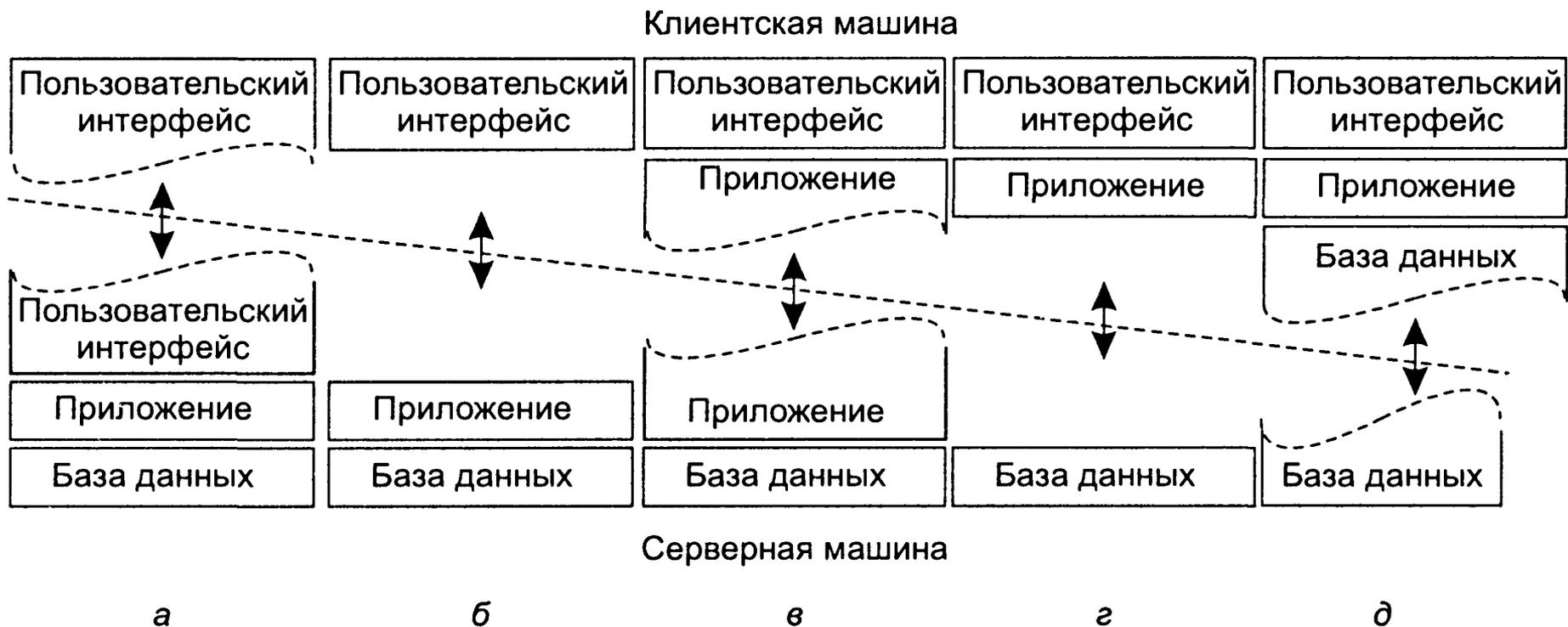


Рис. 3. Альтернативные формы организации архитектуры клиент-сервер

На клиентскую сторону помещают только терминальную часть пользовательского интерфейса, как показано на рис. 3, а, позволив приложению удаленно контролировать представление данных.

Альтернативой этому подходу будет передача клиенту всей работы с пользовательским интерфейсом (рис. 3, б).

Можно перенести во внешний интерфейс часть нашего приложения, как показано на рис. 3, в. Примером может быть вариант, когда приложение создает форму непосредственно перед ее заполнением. Внешний интерфейс затем проверяет правильность и полноту заполнения формы и при необходимости взаимодействует с пользователем.

В некоторых случаях серверу иногда может понадобиться работать в качестве клиента.

Такая ситуация приводит к *физически трехзвенной архитектуре (physically three-tiered architecture)*.

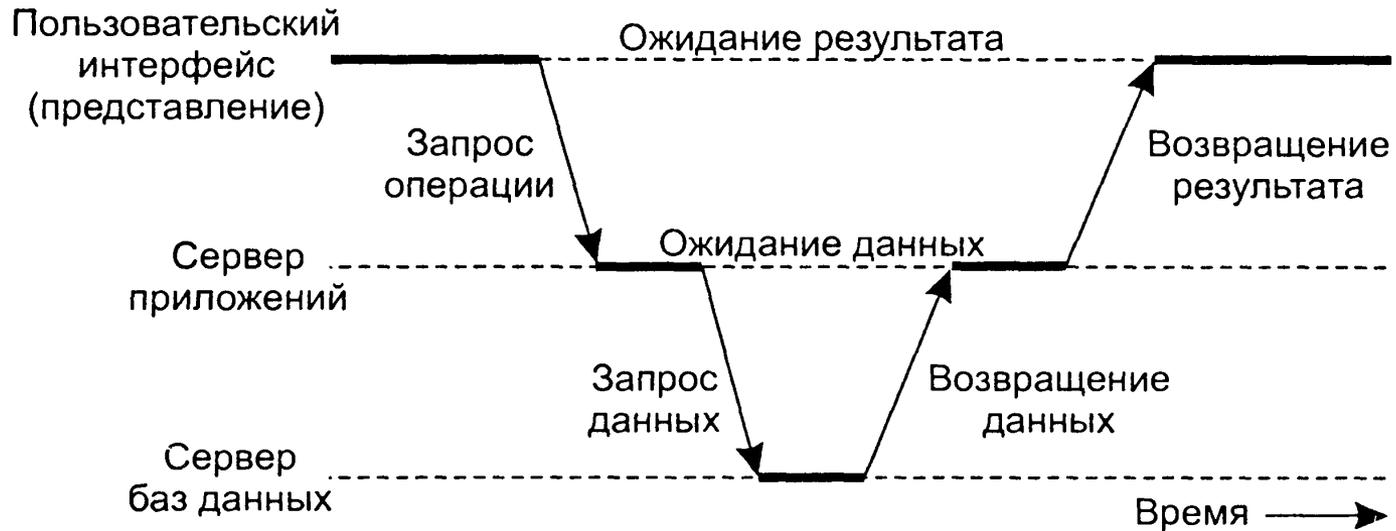


Рис. 4. Пример сервера, действующего как клиент

В трехзвенной архитектуре *программы*, составляющие часть уровня обработки, *выносятся* на отдельный сервер, но дополнительно могут *частично находиться* и на машинах клиентов и серверов.

Типичный пример трехзвенной архитектуры — обработка транзакций. В этом случае отдельный процесс — монитор транзакций — координирует все транзакции, возможно, на нескольких серверах данных.

В трехзвенной архитектуре *программы*, составляющие часть уровня обработки, *выносятся* на отдельный сервер, но дополнительно могут *частично находиться* и на машинах клиентов и серверов.

Типичный пример трехзвенной архитектуры — обработка транзакций. В этом случае отдельный процесс — монитор транзакций — координирует все транзакции, возможно, на нескольких серверах данных.

Такой тип распределения называют *вертикальным распределением (vertical distribution)*.

Характеристической особенностью вертикального распределения является то, что оно достигается размещением логически различных компонентов на разных машинах.

Это понятие связано с концепцией *вертикального разбиения (vertical fragmentation)*, используемой в распределенных реляционных базах данных, где под этим термином понимается разбиение по столбцам таблиц для их хранения на различных машинах

В современных архитектурах распределение на клиенты и серверы происходит способом, известным как *горизонтальное распределение (horizontal distribution)*.

При таком типе распределения клиент или сервер может содержать физически разделенные части логически однородного модуля, причем работа с каждой из частей может происходить независимо.

Это делается для выравнивания загрузки.

Вопрос №3

Встреча и
обработка
входящих
запросов

Реплицированные web-серверы,
каждый из которых содержит одни и
те же web-страницы

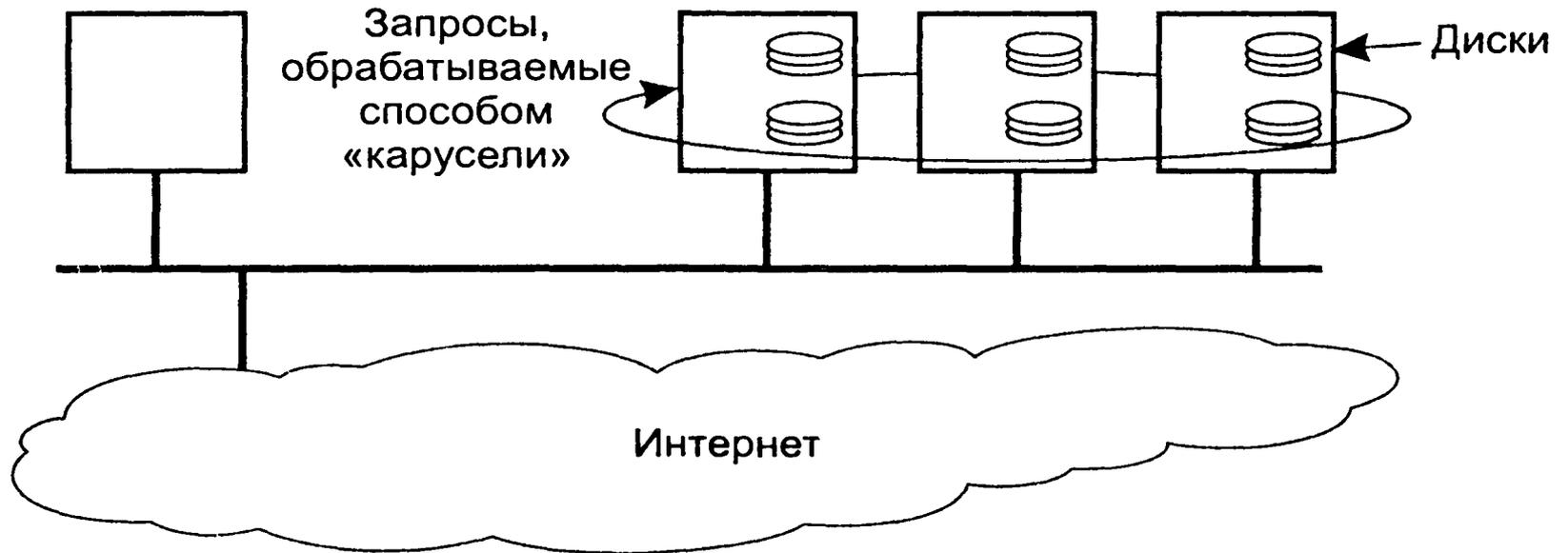


Рис. 5. Пример горизонтального распределения web-службы

Горизонтально могут быть распределены и **клиенты**.

Для несложного приложения, предназначенного для коллективной работы, можно не иметь сервера вообще. В этом случае говорят об *одноранговом распределении (peer-to-peer distribution)*.

Подобное происходит, например, если пользователь хочет связаться с другим пользователем. Оба они должны запустить одно и то же приложение, чтобы начать сеанс. Третий клиент может общаться с одним из них или обоими, для чего ему нужно запустить то же самое приложение.