Самарский государственный аэрокосмический университет
имени академика С.П. Королёва

Занятие 2

Модели и архитектура

Самара
2013

План занятия

- Модели распределенных систем
- Архитектура распределенных систем
- Слои и уровни
- Возможные архитектурные решения

Модели распределенных систем

- Модели архитектуры
  - Физическое размещение компонентов между узлами
  - Взаимодействие между компонентами
- Другие модели
  - Формальное описание различных параметров и свойств системы
  - Модели взаимодействия, обработки ошибок, безопасности, …

Модели архитектуры

- Модель архитектуры распределенной системы должна содержать решение двух проблем:
  - физическое размещение компонентов между узлами
  - взаимодействие между компонентами

Модели архитектуры

- Реальные функции отдельных компонентов не указываются
- Указываются:
  - Расположение (размещение по узлам)
  - Шаблоны распределения данных и задач по их обработке
  - Взаимодействие компонентов
    - Роли компонентов
    - Шаблоны взаимодействия

Архитектура

- Определяет разделение системы на наиболее крупные составные части
- Определяет конструктивные решения, которые после их принятия с трудом поддаются изменению
- Отображает общий взгляд разработчиков на результаты проектирования системы: идентификация главных компонентов системы, способов их взаимодействия, выбор основополагающих решений, не подлежащих изменению в будущем

Основные принципы архитектуры

- **Согласованность**
  Частичное знание системы позволяет предсказать остальное
- **Ортогональность**
  Функции независимы и специфицированы по отдельности
- **Соответствие**
  Включаются только функции, соответствующие существенным требованиям к системе, нет ненужных функций
- **Экономичность**
  Отсутствие дублирования

Основные принципы архитектуры

- **Прозрачность**
  Функции должны быть известны пользователю
- **Общность**
  Если функция должна быть введена, ее следует вводить в таком виде, чтобы она отвечала как можно большему числу назначений
- **Открытость**
  Можно использовать функцию иначе, чем это предполагалось при проектировании
- **Полнота**
  Введенные функции должны с учетом экономических и технологических ограничений как можно полнее соответствовать требованиям и пожеланиям пользователя

Преодоление сложности

- Использование паттернов проектирования
- Разделение системы на слои (расслоение)

Типовые решения – паттерны проектирования

Кристофер Александер (Christopher Alexander):

Каждое типовое решение описывает некую повторяющуюся проблему и ключ к ее разгадке, причем таким образом, что вы можете пользоваться этим ключом многократно, ни разу не придя к одному и тому же результату

Структура типовых решений

- Название решения
- Назначение (аннотация)
- Мотивация, применимость
- Принцип действия (структура, участники, отношения)
- Результаты
- Реализация

ПРАВИЛО ДОБАВЛЕНИЯ КОСВЕННОСТИ

Если есть проблема – введите посредника, т.е. вместо прямого взаимодействия используйте косвенное

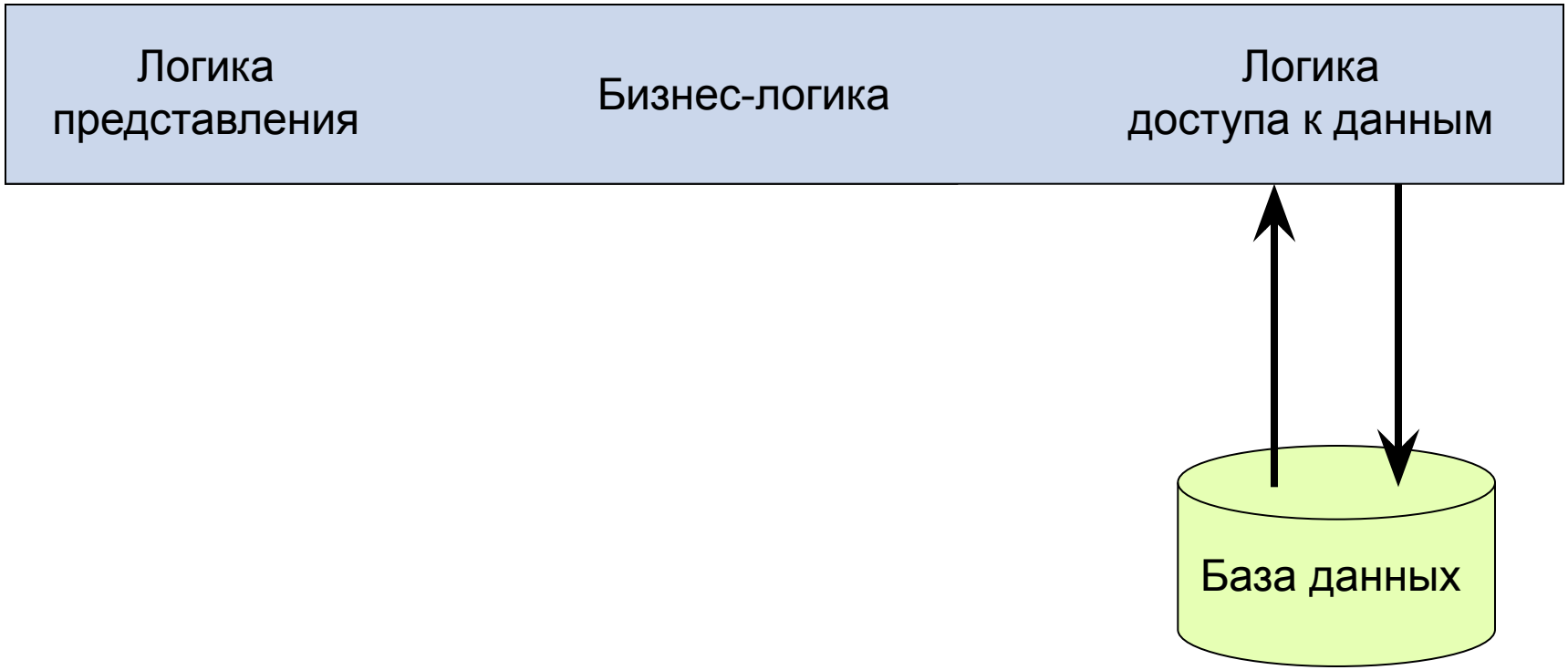
Слои

- Основная идея: независимость нижележащих уровней от вышележащих
- Основная задача: уменьшать сложность систем, разделяя их на слои и сервисы
  - Слой (уровень): группа сильно связанных и закрытых элементов, реализующих одну функциональность
  - Сервис: функциональность, обеспечиваемая для вышестоящего слоя

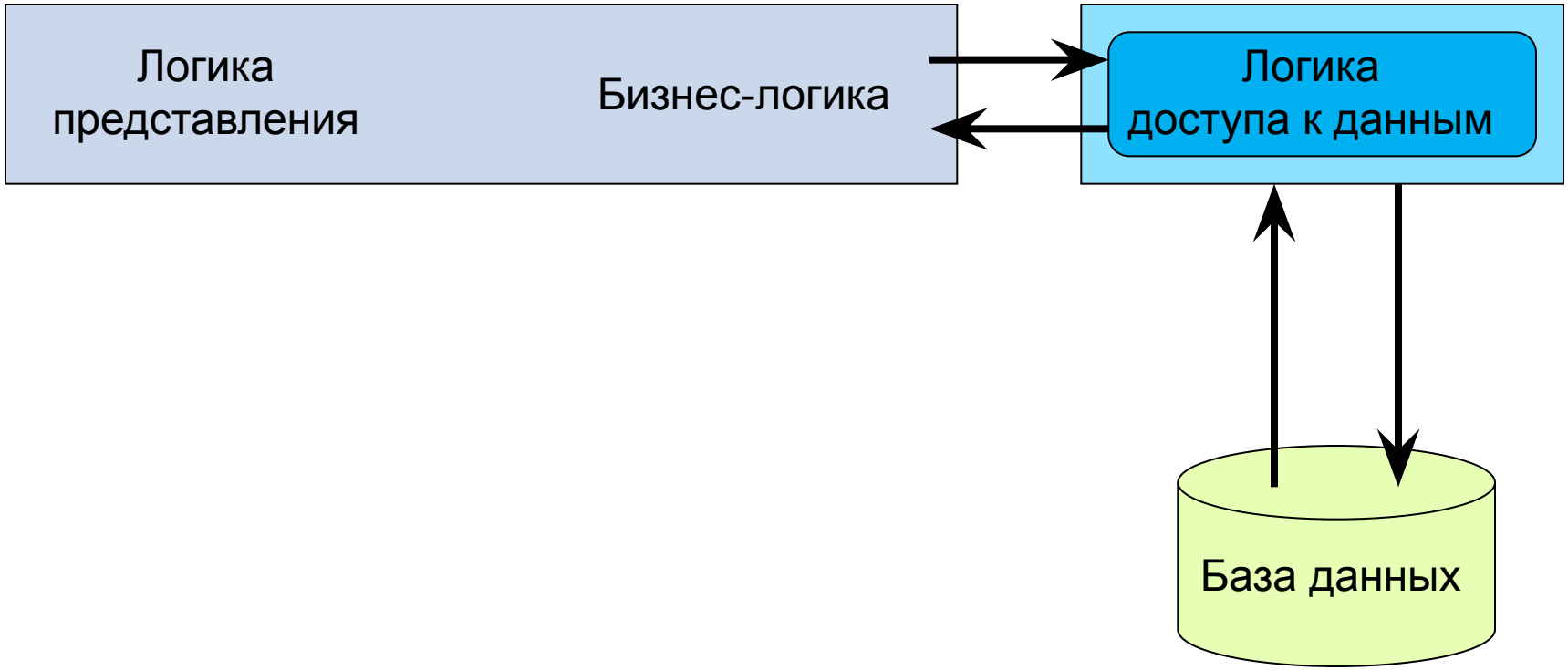
Примеры подхода

- Архитектура сетевых протоколов
  - Физический уровень
  - Уровень соединения
  - Сетевой уровень
  - Транспортный уровень
  - Сеансовый уровень
  - Уровень представления
  - Прикладной уровень
- Распределенные приложения
  - Аппаратура
  - Операционная система } **Платформа**
  - Промежуточное программное обеспечение
  - Приложения, сервисы

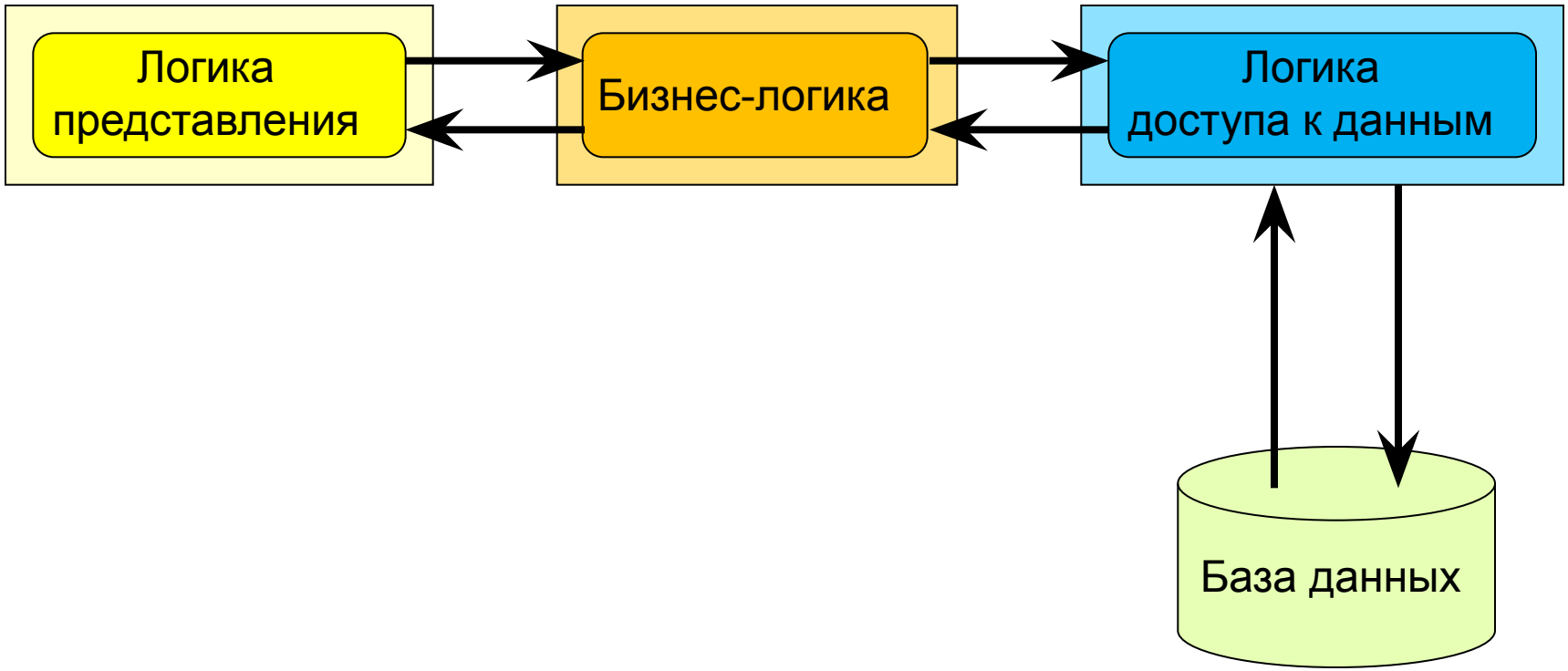
Монолитная система



Двухслойная система



Трехслойная система

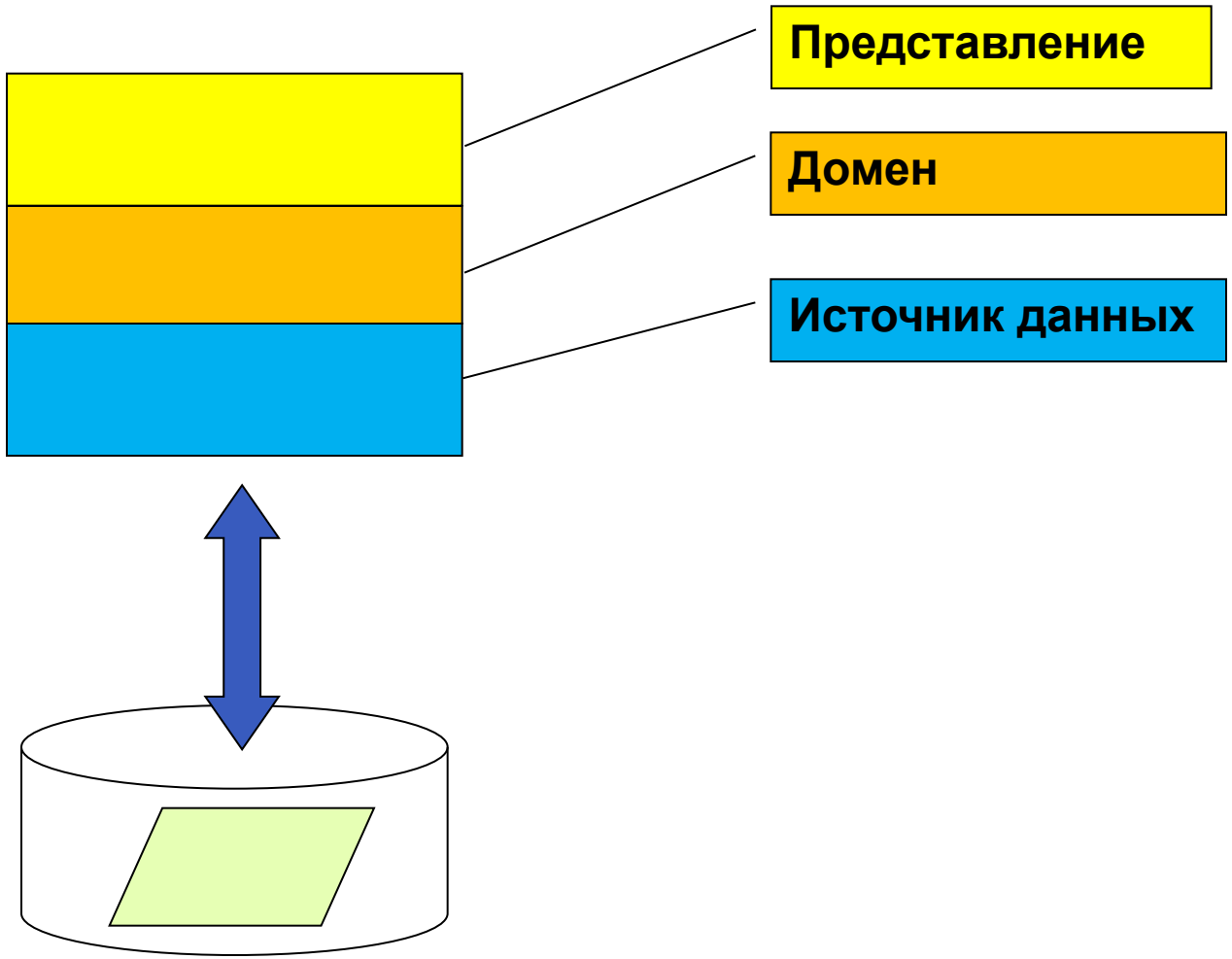


Три основных слоя

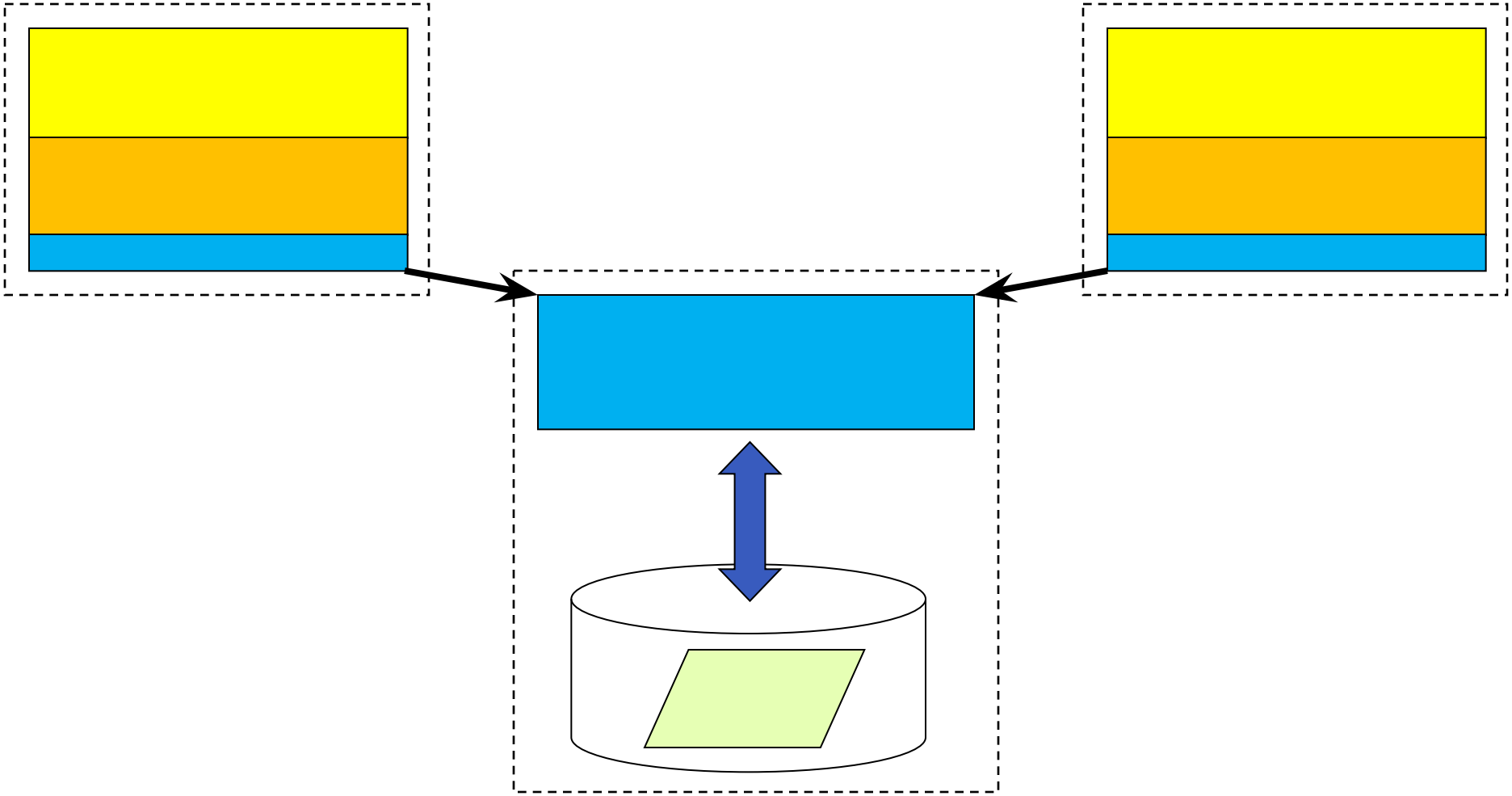
- Слои:
  - представление (presentation)
  - домен (domen) – предметная область, бизнес-логика
  - работа с данными (data source)

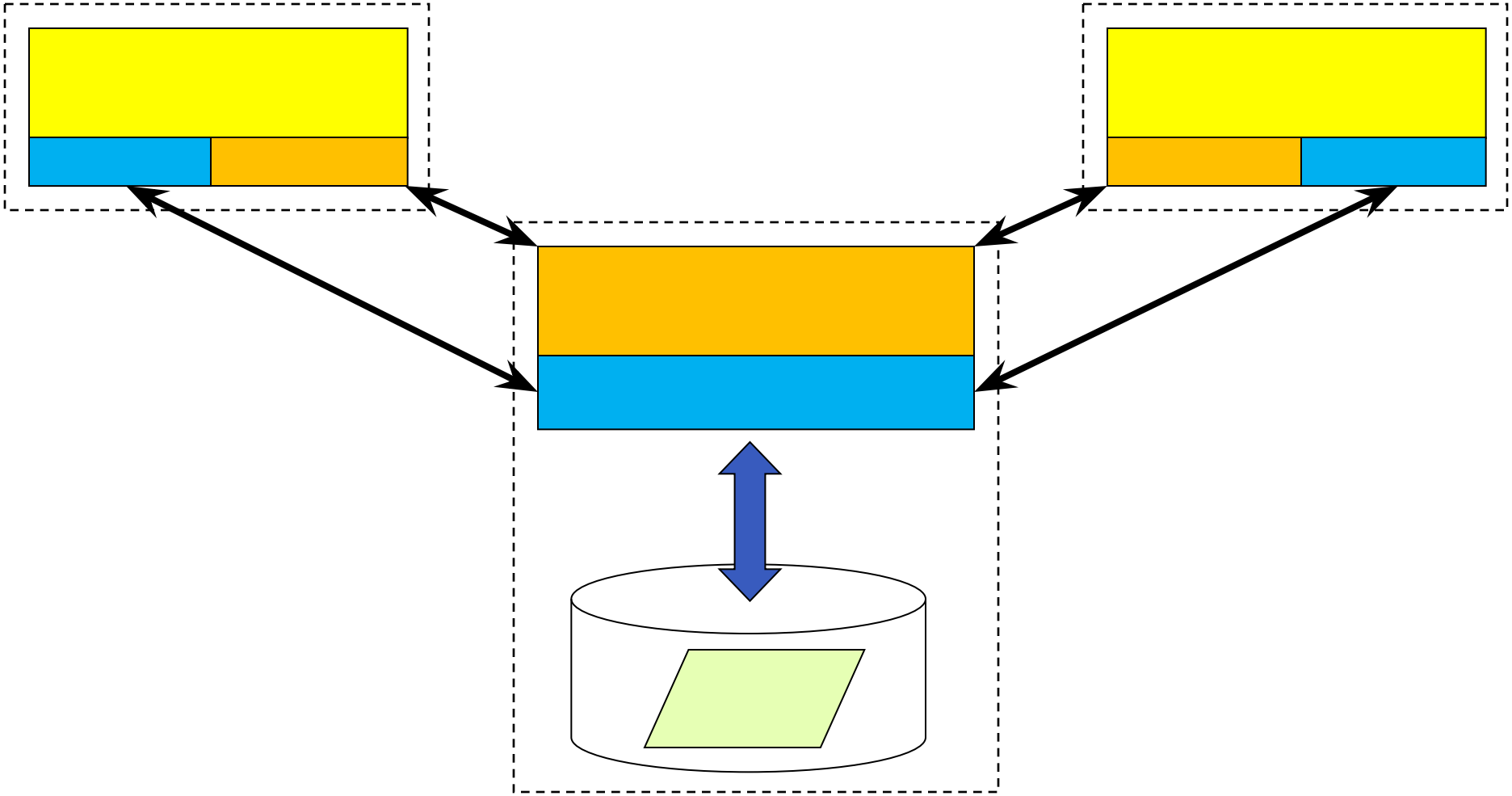
- Рекомендуется различать:
  - **Слой** (layer) – логическое разделение
  - **Уровень** или **Ярус** (tier) – физическое разделение

Одноуровневая система

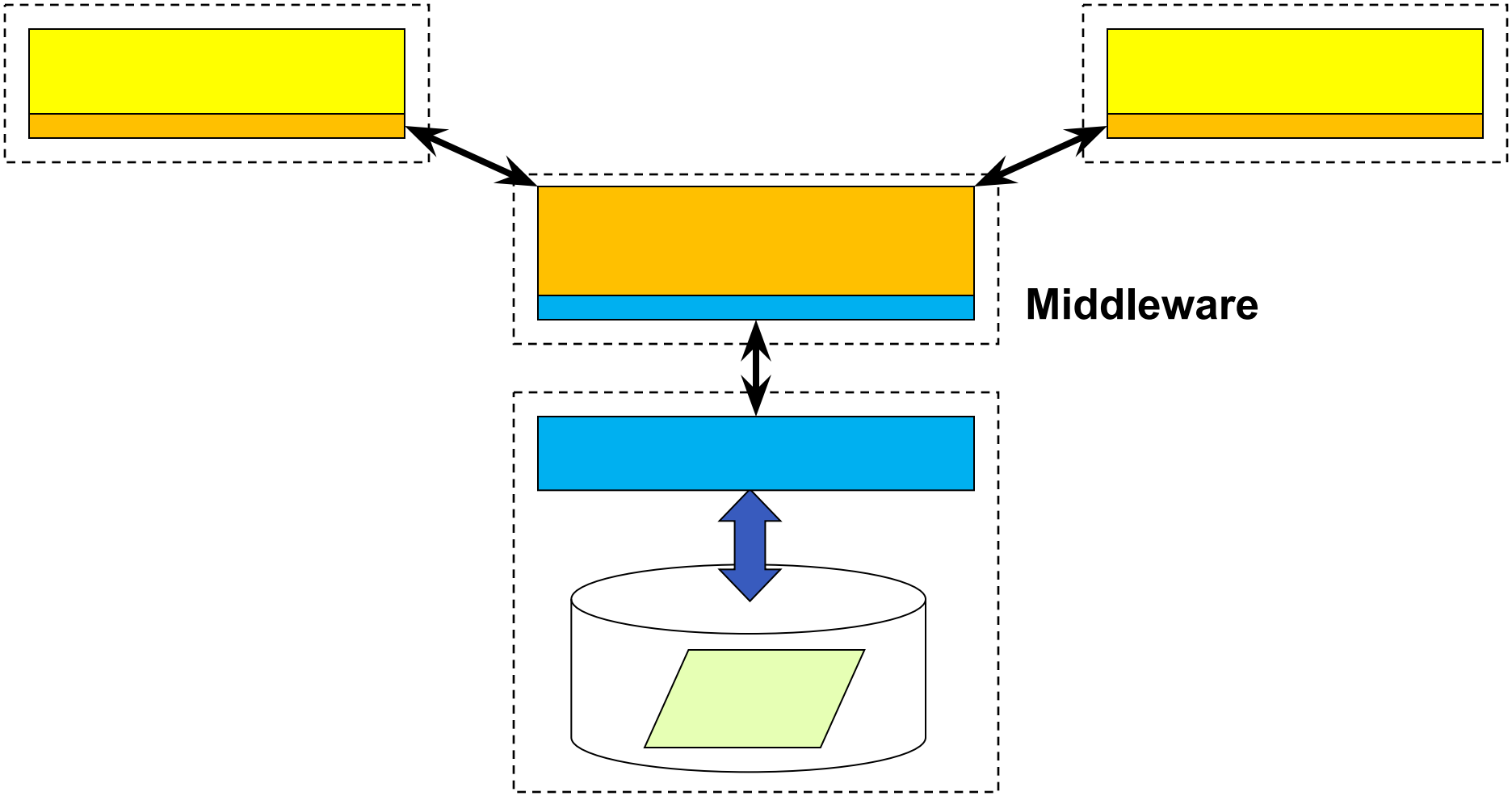


Система клиент/сервер (двухуровневая)





Трехуровневая система



Middleware

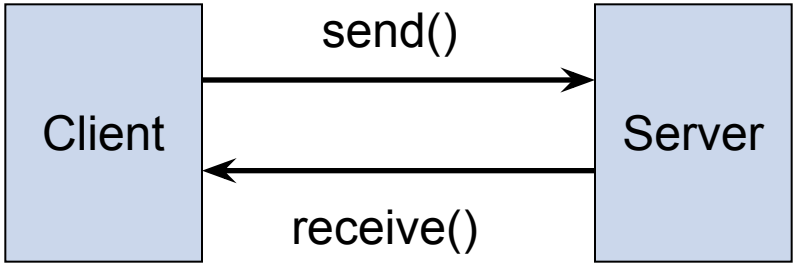
- “Слой программного обеспечения, чья цель состоит в том, чтобы скрывать неоднородность и обеспечивать удобную модель программирования для разработчиков”
  - Предоставляет прикладной интерфейс программирования
- Примеры
  - CORBA (OMG)
  - .Net (Microsoft)
  - Remote Procedure Call (Sun)
  - Java Remote Method Invocation (Sun)
  - Технология EJB (Enterprise JavaBeans, Sun)
- Может также предоставлять услуги (сервисы)

Сервисы Middleware

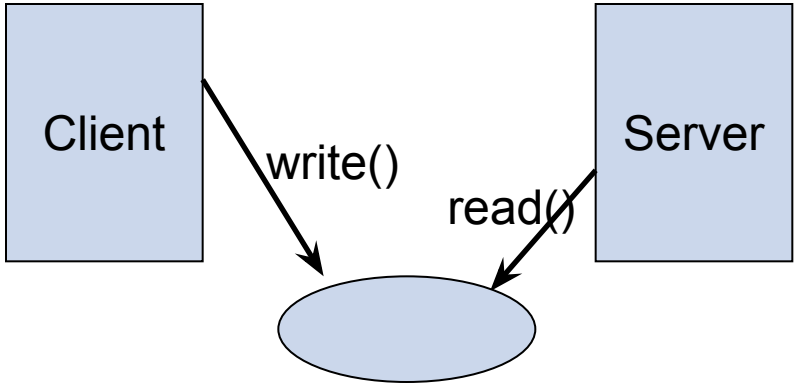
- Сервисы
  - Именования
  - Безопасности
  - Транзакций
  - Долговременного хранения
  - Уведомления о событиях
  - …

Основные парадигмы программирования

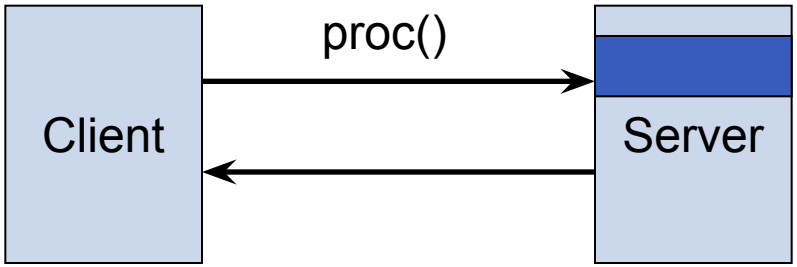
Message Passing



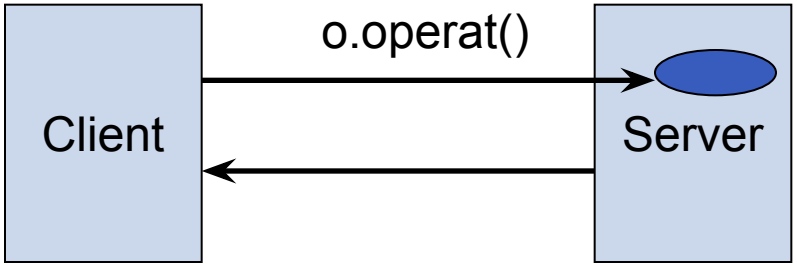
Virtual Shared Memory



Remote Procedure Call



Distributed Object System



Требования к дизайну

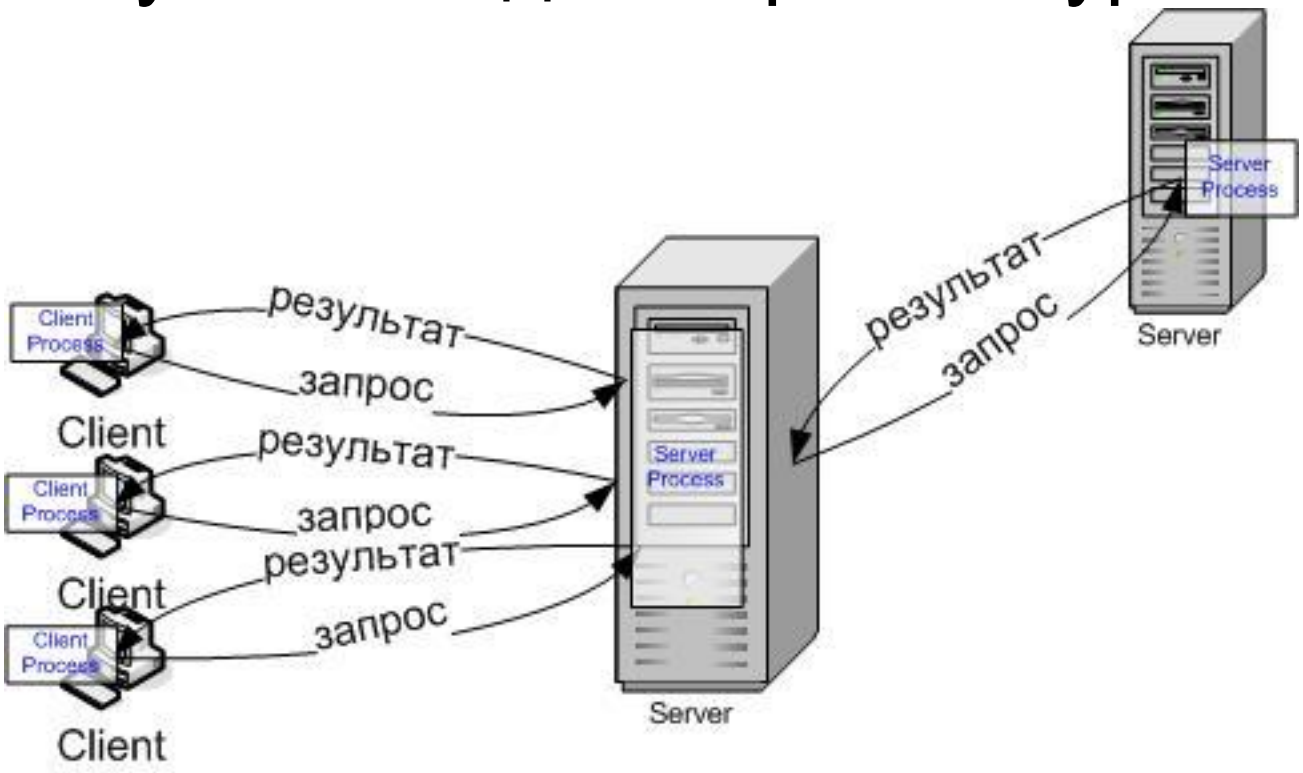
- Требования, накладываемые обеспечением требуемой производительности
  - Время отклика
  - Производительность
  - Балансировка нагрузки
- Использование кэширования и репликации
  - Очень многие проблемы производительности системы могут быть решены путем кэширования данных
- Требование надежности

Возможные архитектуры

- Клиент-сервер
- Модель предоставления услуг пулом серверов
- Модель прокси- и кэш- серверов
- Модель равных процессов

Модель клиент-сервер

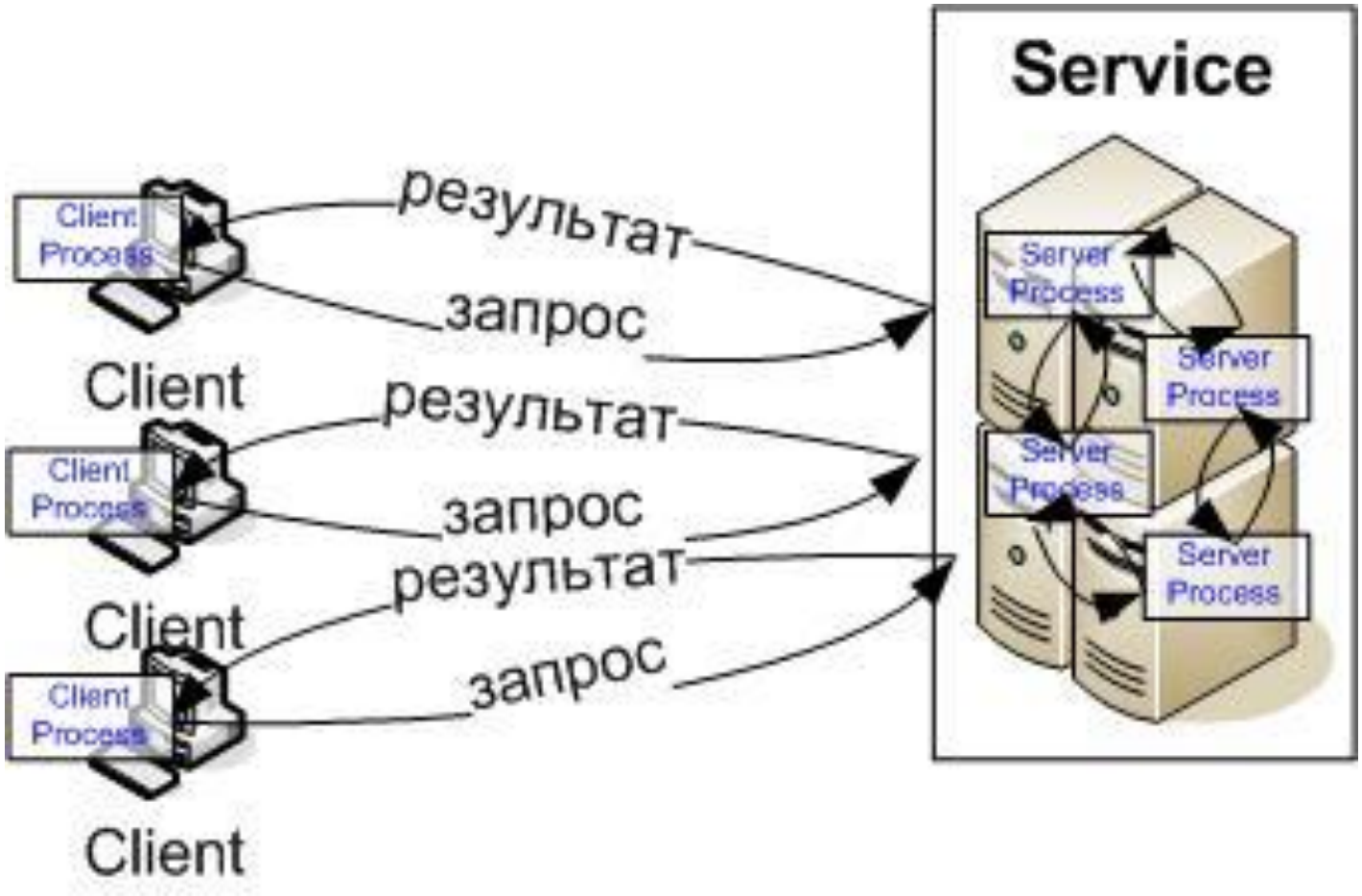
- Наиболее распространенная и часто используемая модель архитектуры



Модель клиент-сервер

- **Клиент**
  Процесс, желающий получить доступ к данным, использует ресурсы и/или выполняет действия на удаленном узле
- **Сервер**
  Процесс, управляющий данными и всеми другими разделяемыми ресурсами, обеспечивающий клиентам доступ к ресурсам и производящий вычисления
- **Взаимодействие**
  Пары запрос / результат
- **Пример**
  http-сервер: клиент (браузер) запрашивает страницу, сервер поставляет страницу

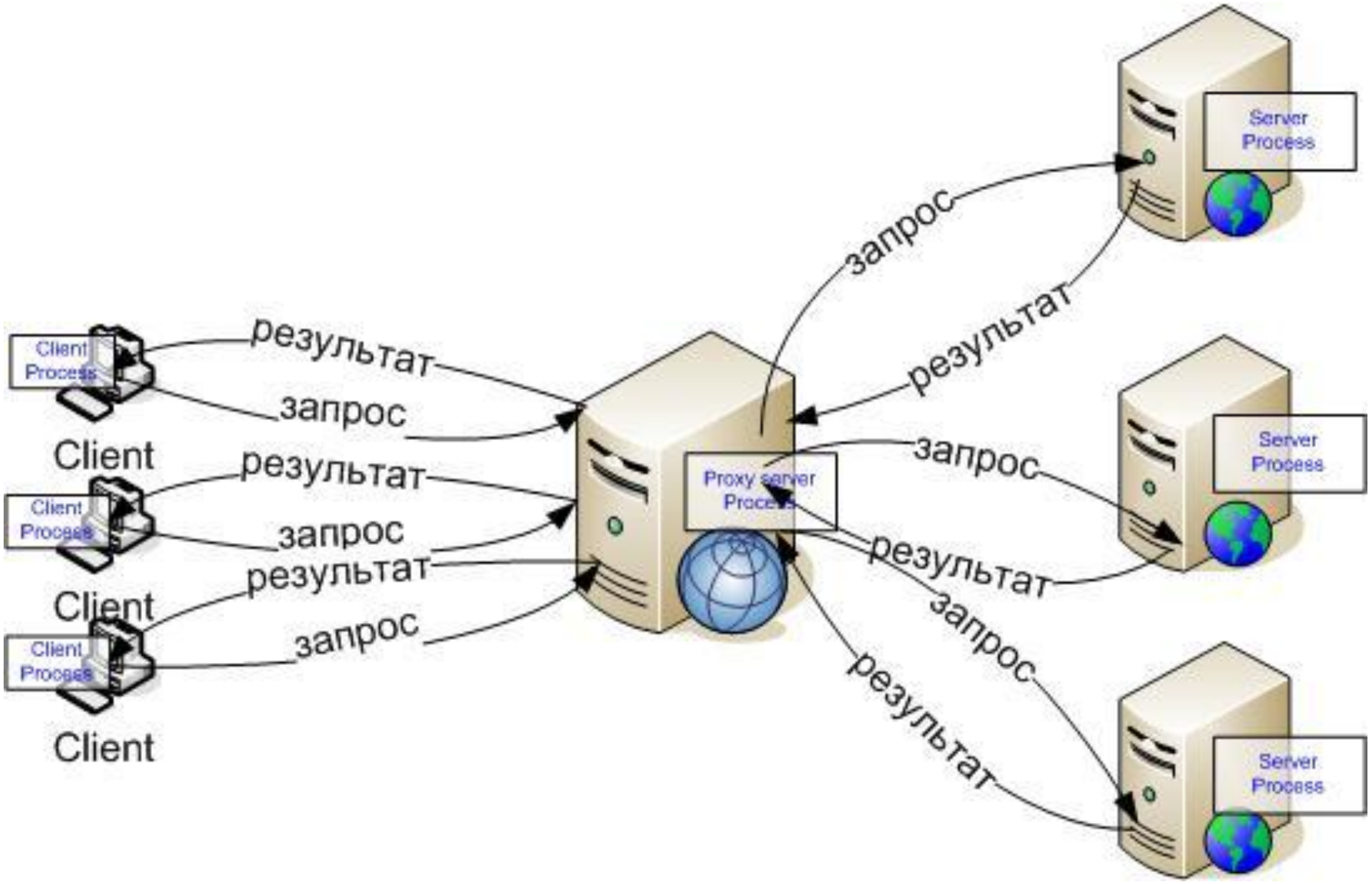
Модель «пул серверов»



Модель «пул серверов»

- Услуги могут обеспечиваться многими серверами
- Распределенные между серверами объекты
- Реплицированные объекты
  - Увеличение производительности, доступности и отказоустойчивости
- Но требуют координации копий / консистентности представления
- Например, высокодоступные серверы (порталы, диллинговые центры), информационные службы
- Серверы, обслуживающие распределенную БД

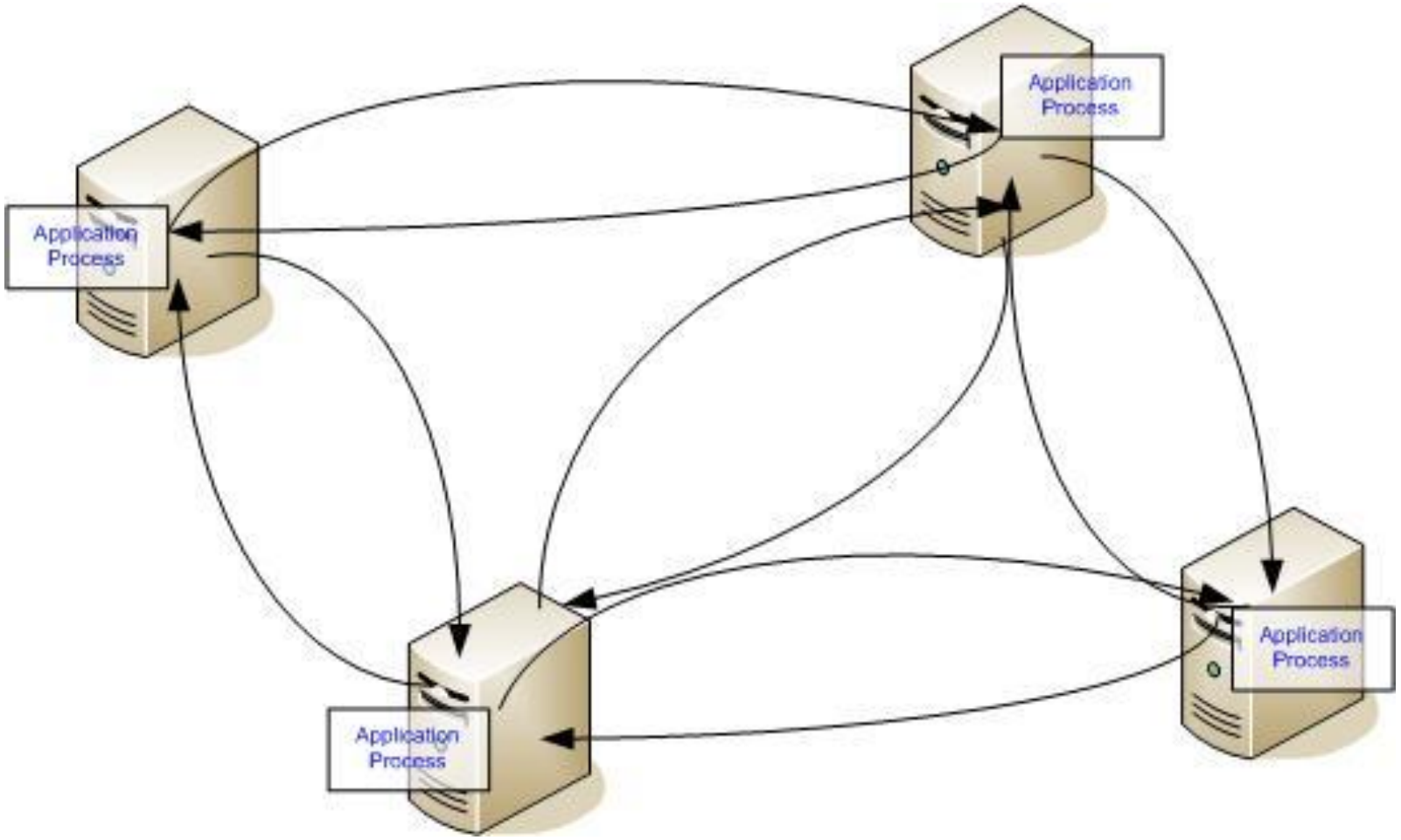
Модель с прокси-сервером



Модель с прокси-сервером

- **Кэш**: «близкая» копия наиболее часто используемых данных
  - **значительно** повышает производительность большинства приложений
  - Но требует усилий по поддержанию когерентности
- **Прокси-сервер**: разделяемый кэш ресурсов
  - Еще сложнее, чем простой кэш…
  - Обычно используется (и хорошо подходит) для доступа к веб-ресурсам

Равноправные процессы (P2P)



Равноправные процессы (P2P)

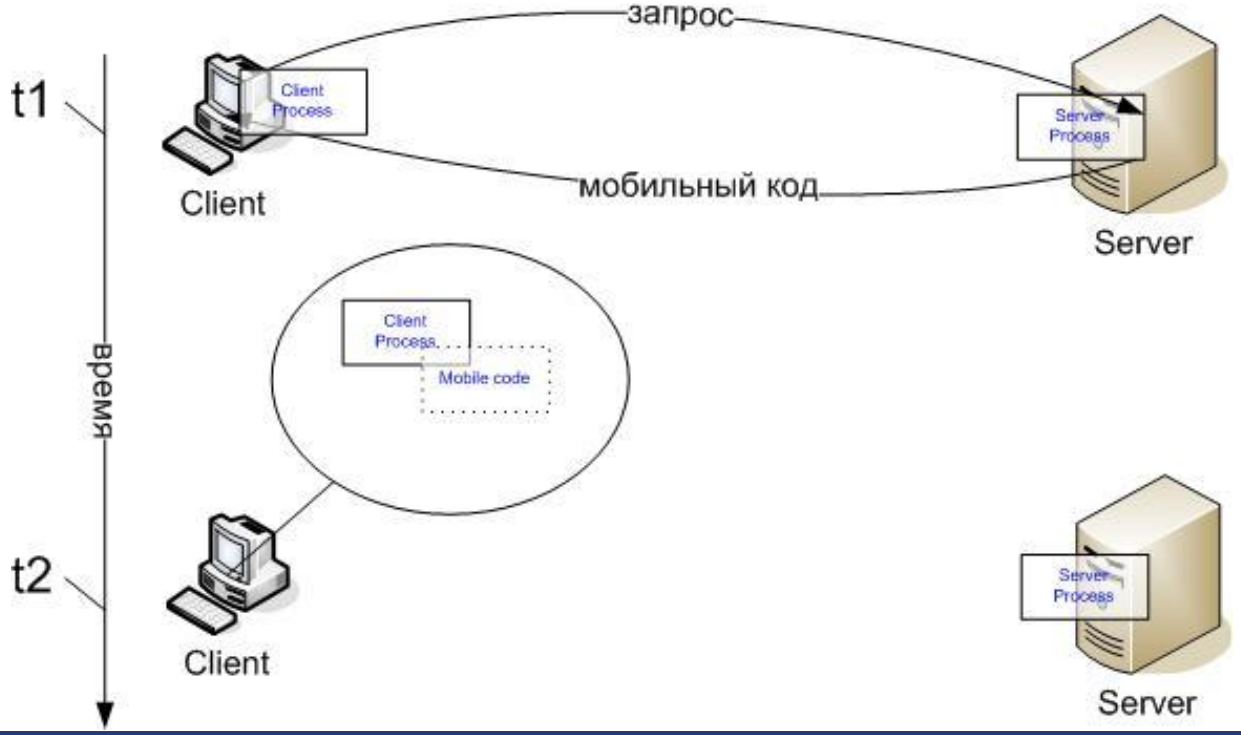
- Равные процессы: процессы, которые играют равные роли
  - Никакого абсолютного различия между клиентом / сервером
  - Роли клиента и сервера различаются от вызова к вызову (или со временем)
- Увеличивает устойчивость к сбоям и масштабируемость
- Трудности с координацией
- Примеры: BitTorrent, распределенное вычисление

Вариации модели клиент-сервер

- Вариации возможны по следующим параметрам:
  - Связь инициируется сервером
  - Использование мобильного кода или мобильных агентов
  - Легкие клиенты базирующиеся на потребности пользователей в дешевых компьютерах и простом управлении (тонкий клиент)

Мобильный код

- Мобильный код: код, посланный процессу клиента, чтобы выполнить его же задачу



Мобильные агенты

- Выполнение программы (код + данные), которая перемещается между узлами в сети
  - Выполняет автономную задачу обычно под управлением некоторого другого процесса
  - Имеет внутреннее знание и цели
- Преимущество: всюду локальный доступ
  - Снижает затраты на коммуникации
- Потенциальная угроза безопасности
  - Ограниченная применимость
- Примеры: сбор данных из многих источников, установка программ, программы типа червей

Тонкие клиенты

Тонкие клиенты

- Аппаратные
  (сетевые компьютеры, network computers)
  - Все файлы сохраняются на удаленном носителе
  - Минимум локального программного обеспечения
  - Любой локальный диск используется только под кэш
- Программные
  - Реализуют только интерфейс пользователя на локальном компьютере
  - Непосредственно программы работают на вычислительном сервере

Итоги

- Использование модели слоев для снижения сложности системы
  - Middleware обеспечивает дополнительное удобство и дополнительные сервисы
- Выбор модели архитектуры в зависимости от особенностей задачи
  - Клиент – сервер
  - Модель предоставления услуг пулом серверов
  - Модель прокси- и кэш-серверов
  - Модель равных процессов

Спасибо за внимание!

Дополнительные источники

- Таненбаум, Э. Распределенные системы. Принципы и парадигмы [Текст] / Э. Таненбаум, М. ван Стеем. – СПб. : Питер, 2003. – 877 с.
- Эндрюс, Г.Р. Основы многопоточного, параллельного и распределенного программирования [Текст] / Грегори Р. Эндрюс. – М. : Издательский дом «Вильямс», 2003. – 512 с.
- Фаулер, М. Архитектура корпоративных программных приложений [Текст] / Мартин Фаулер. – М. : Издательский дом «Вильямс», 2004. – 544 с.
- Обзор распределённых систем [Электронный ресурс]. – Режим доступа: http://masters.donntu.edu.ua/2008/fvti/prihodko/library/dist2.htm, дата доступа: 21.10.2011.
- Распределённые вычисления [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/Распределенные_вычисления, дата доступа: 21.10.2011.