

# Распределенные системы

---

Введение

# Литература

---

- Таненбаум Э., Ван Стеен М. Распределенные системы. Принципы и парадигмы. – СПб.:Питер, 2003. – 887 с.
- Эммерих В. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft/COM, Java/RMI. – М.:Мир, 2002.- 510с.
- Дейтел Х., Дейтел П., Чофнес Д. Операционные системыю Распределенные системы, сети, безопасность: Третье издание. – М.:ООО «Бином-Пресс», 2006.-704с.



# Дополнительно

- Биберштейн Н., Боуз С. Компас в мире сервис-ориентированной архитектуры(SOA): ценность для бизнеса, планирование и план развития предприятия. – М.: КУДИЦ-ПРЕСС, 2007 – 256 с.
- Фаулер М. Архитектура корпоративных программных приложений. — М.: Издательский дом "Вильямс", 2006. — 544 с.
- Coulouris G., Dollimore J. Distributed Systems. Concepts and Design. – Addison-Wesley, 1994. – 664p.
- Цимбал А., Аншина А. Технологии создания распределенных систем. Для профессионалов. – СПб.: Питер, 2003 – 576 с.
- Ферара А., Мак-Дональд М. Программирование Web-сервисов для .NET. Библиотека программиста. – Киев.: BHV; СПб.: Питер, 2003 – 430 с.



# Отчетность

---

- Экзамен/Дифференцированный зачет
- Реферат (октябрь-ноябрь)



# Определение

---

- Распределенная система – несколько компьютеров, которые видны для конечного пользователя как единое целое



# Предпосылки появления

- Мощные микропроцессоры
  - 8-bit, 16-bit, 32-bit, 64-bit
  - x86 family, 68k family, Alpha chip
  - Частота от 4.77MHz до 4.0 GHz
- Компьютерные сети
  - Local Area Network (LAN), Wide Area Network (WAN), MAN, Wireless Network (Wi-Fi), Mobile Network (3G/UMTS)
  - Типы сетей: Ethernet, Token-bus, Token-ring, FDDI, ATM, Fast-Ethernet, Gigabit Ethernet, Fibre Channel
  - Скорость передачи: от 64 kbps до 1Gbps
- Устройства хранения (Hard Disk)
  - 5-10Mb(85), 100-250Mb(90), 1Gb(93), 4-6Gb(97), 10-20Gb(00), 80-120+Gb.



# Распределенные и централизованные системы

## ■ Преимущества

- Экономические: Микропроцессоры предлагают лучшее соотношение цена/качество чем мэйнфреймы
- Производительность: Распределенная система может иметь большую суммарную производительность чем мэйнфрейм
- Унаследованное распределение: Некоторые приложения в банковской сфере, сфере логистик изначально включают в себя географически распределенные компьютеры
- Надежность: Если 5% компьютеров будут выведены из строя, система в целом будет работоспособна с 5% потерями в производительности
- Пошаговый рост: Производительность может наращиваться небольшими порциями
- Разделение данных: Много пользователей могут получить доступ к общим данным
- Разделение устройств: Многие пользователи могут получить доступ к дорогому периферийному оборудованию
- Связь: Делают взаимодействие человек-человек очень легким (Email, ICQ)
- Гибкость: Разделяют задачи по всем доступным компьютерам наиболее эффективным способом с точки зрения затрат.



# Распределенные и централизованные системы

---

- Недостатки
  - Программное обеспечение: В настоящее время существует небольшое количество программного обеспечения для распределенных систем
  - Сеть: Наличие сети может добавить другие проблемы
  - Безопасность: Легкость доступа относится также и к секретным данным





# Прозрачность в распределенных системах

Прозрачность	Описание
Доступ	Скрывается разница в представлении данных
Местоположение	Скрывается местоположение ресурса
Перенос	Скрывается факт перемещения ресурса
Смена местоположения	Скрывается факт перемещения ресурса во время обработки
Репликация	Скрывается факт репликации ресурса
Параллельный доступ	Скрывается факт возможного параллельного использования
Отказ	Скрывается отказ и восстановление
Сохранность	Скрывается, хранится ресурса в памяти или на диске

# Ловушки при разработке распределенных систем

---

- Неверные предположения, которые делает начинающий
  - Передача по сети надежна
  - Передача по сети безопасна
  - Сеть гомогенна
  - Топология не меняется
  - Задержки при передаче равны 0
  - Полоса пропускания канала передачи бесконечна
  - Затраты на передачу нулевые
  - Существует один администратор



# Проблемы проектирования

- Надежность
  - Доступность
  - Отказоустойчивость
- Производительность
  - Низко уровневый параллелизм
  - Высокоуровневый параллелизм
- Масштабируемость
  - Потенциальные узкие места в очень больших распределенных системах
    - Централизованные компоненты
    - Централизованные данные
    - Централизованные алгоритмы ( алгоритм маршрутизации)
  - Используйте децентрализованные алгоритмы
    - Ни одна машина не владеет информацией полностью
    - Можно принимать решения только на основе локальной информации
    - Выход из строя одной машины не должен разрушать алгоритм
    - Нет предположения что общее глобальное время существует



# Концепция аппаратного обеспечения

---

- Слабо и сильно связанное аппаратное обеспечение
- SISD: Single Instruction stream, Single Data stream – традиционные компьютеры
- SIMD: Single Instruction stream, Multiple Data streams – параллельные супер-компьютеры.
- MISD: Multiple Instruction streams, Single Data stream – такого компьютера нет
- MIMD: Multiple Instruction streams, Multiple Data streams – все распределенные системы
- Мультипроцессоры на общей шине – несколько процессоров подключаются к памяти через общую шину.
- Коммутируемые мультипроцессоры – подключаются более 64 процессоров через устройство коммутации



# Концепция аппаратного обеспечения

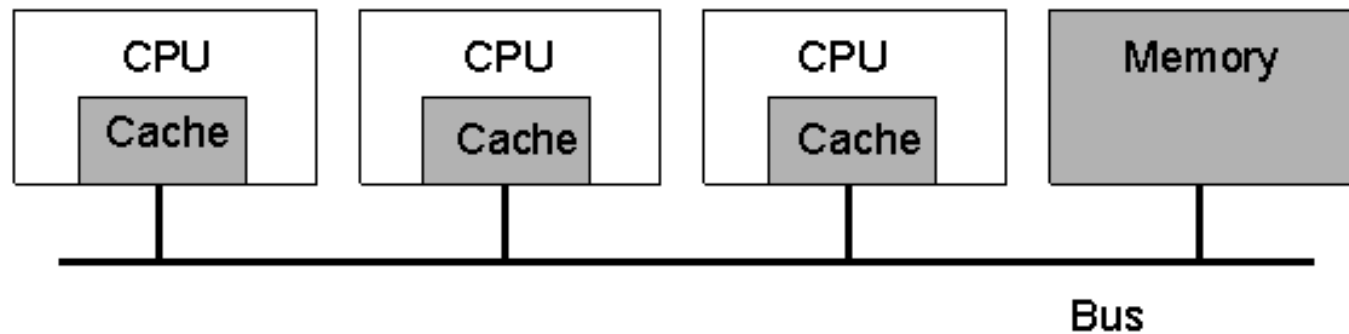
---

- Шинная архитектура – несколько компьютеров, соединенных сетью
- Коммутируемая архитектура – компьютеры соединены различной топологией - Grid, Hypercube.



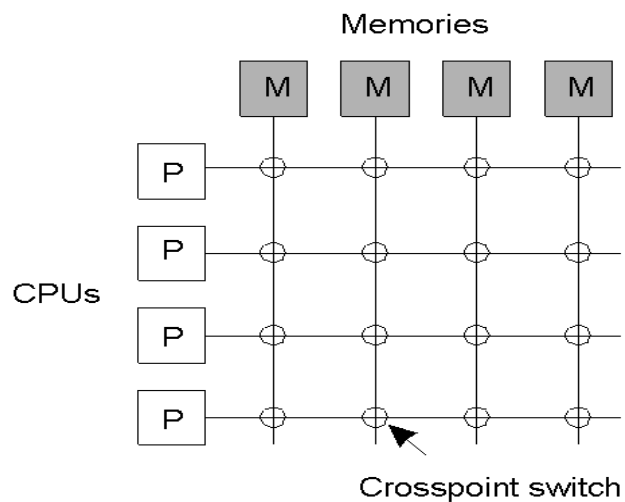
# Мультипроцессоры

- Шинная архитектура

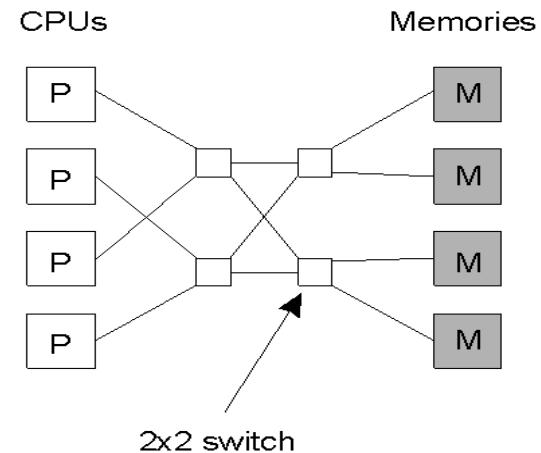


# Мультипроцессоры

- Узловой коммутатор
- Омега сеть



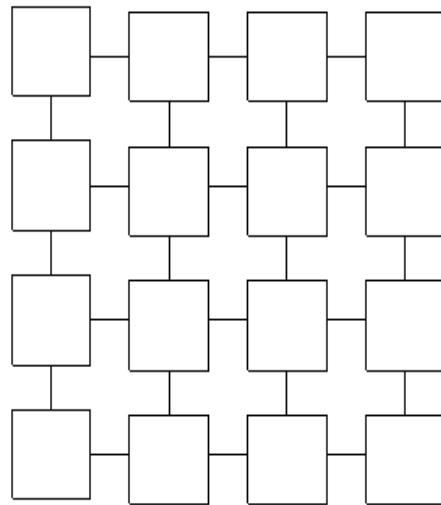
(a)



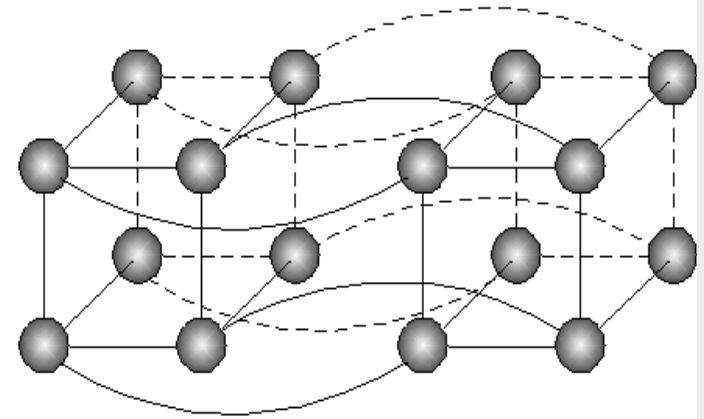
(b)

# Гомогенные мультимикомпьютерные системы

- a) Grid
- b) Hypercube



(a)



(b)

Большое количество процессоров  
приводит к регулярным сбоям



# Гетерогенные мультикомпьютерные системы


---

- Размер памяти, тип процессора, быстродействие не важны
- Связь через локальные сети или интернет

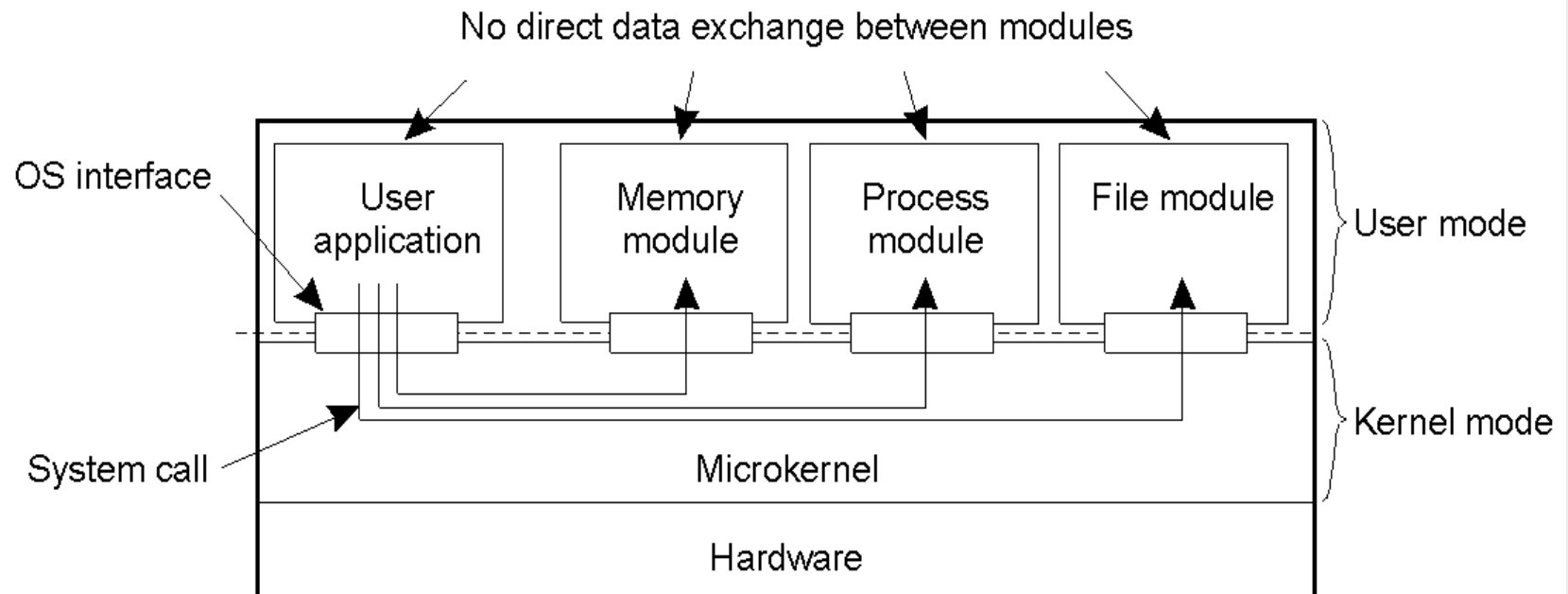


# Концепция программного обеспечения

Система	Описание	Основная цель
DOS	Сильносвязанная операционная система для мультипроцессоров и гомогенных мультикомпьютеров	Соккрытие и управление аппаратным обеспечением
NOS	Слабосвязанная операционная система для гетерогенных мультикомпьютеров (LAN, WAN)	Предлагает локальные сервисы для клиентов
Middleware	Дополнительный уровень поверх сетевых операционных систем, реализующий сервисы общего пользования	Предоставляют прозрачность распределения

- 
- DOS (Distributed Operating Systems)
  - NOS (Network Operating Systems)
  - Middleware

# Однопроцессорные операционные системы



- Разграничение доступа между модулями через микроядро
- Противоположность – монолитное ядро с режимами user и kernel mode

# Мультипроцессорные операционные системы (1)

- Используем мониторы для синхронизации доступа
- Более структурированный код по сравнению с семафорами

```
monitor Counter {  
    private:  
        int count = 0;  
    public:  
        int value() { return count;}  
        void incr () { count = count + 1;}  
        void decr() { count = count - 1;}  
}
```



# Мультипроцессорные операционные системы (2)

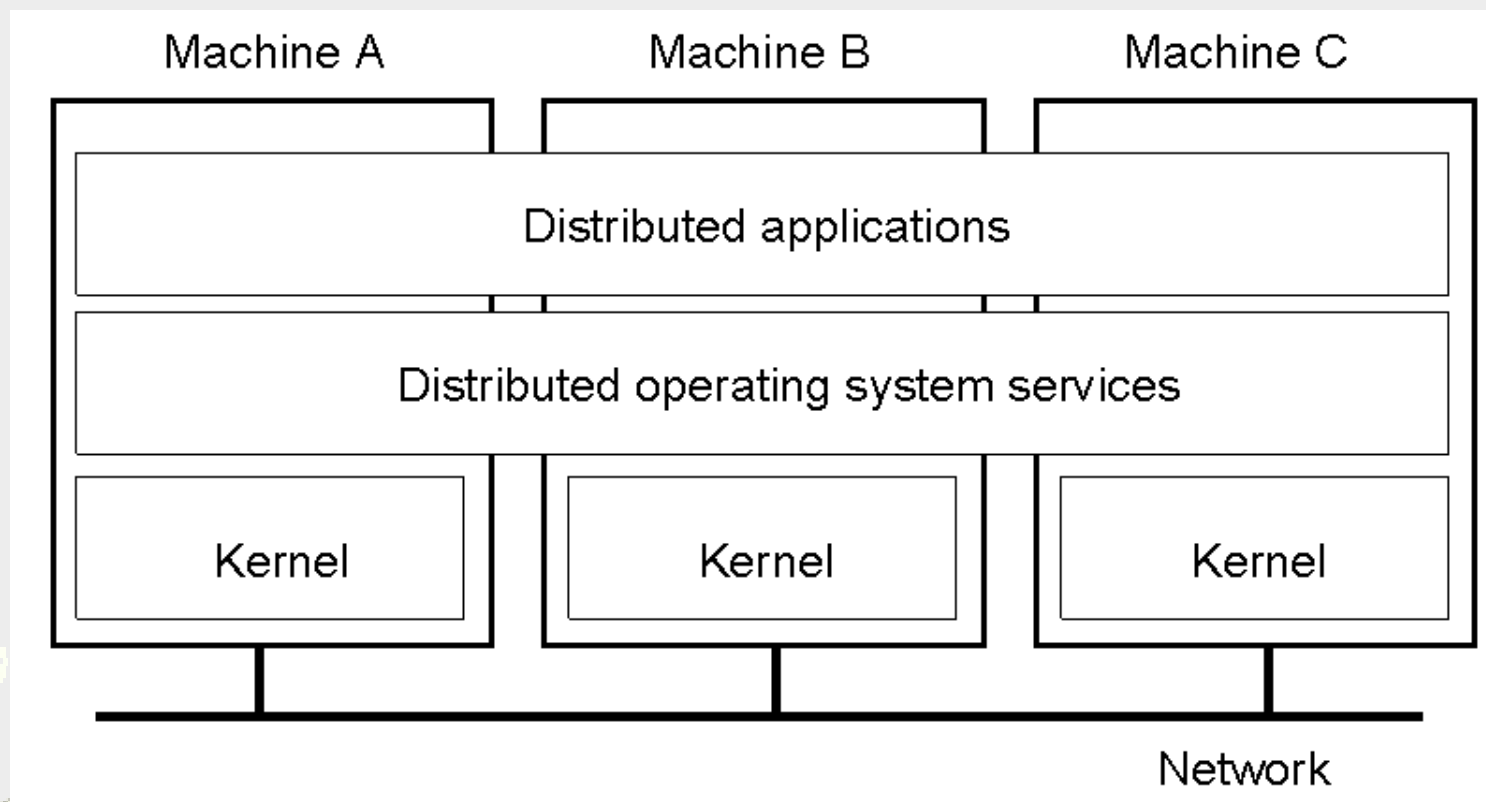
```
monitor Counter {  
private:  
    int count = 0;  
    int blocked_procs = 0;  
    condition unblocked;  
public:  
    int value () { return count;}  
    void incr () {  
        if (blocked_procs == 0)  
            count = count + 1;  
        else  
            signal (unblocked);  
    }  
}
```

```
void decr() {  
    if (count == 0) {  
        blocked_procs = blocked_procs + 1;  
        wait (unblocked);  
        blocked_procs = blocked_procs - 1;  
    }  
    else  
        count = count - 1;  
}
```

- монитор с условием
- wait(x) – предоставляет монитор другим процессам и ожидает события signal(x)

# Мультикомпьютерная операционная система

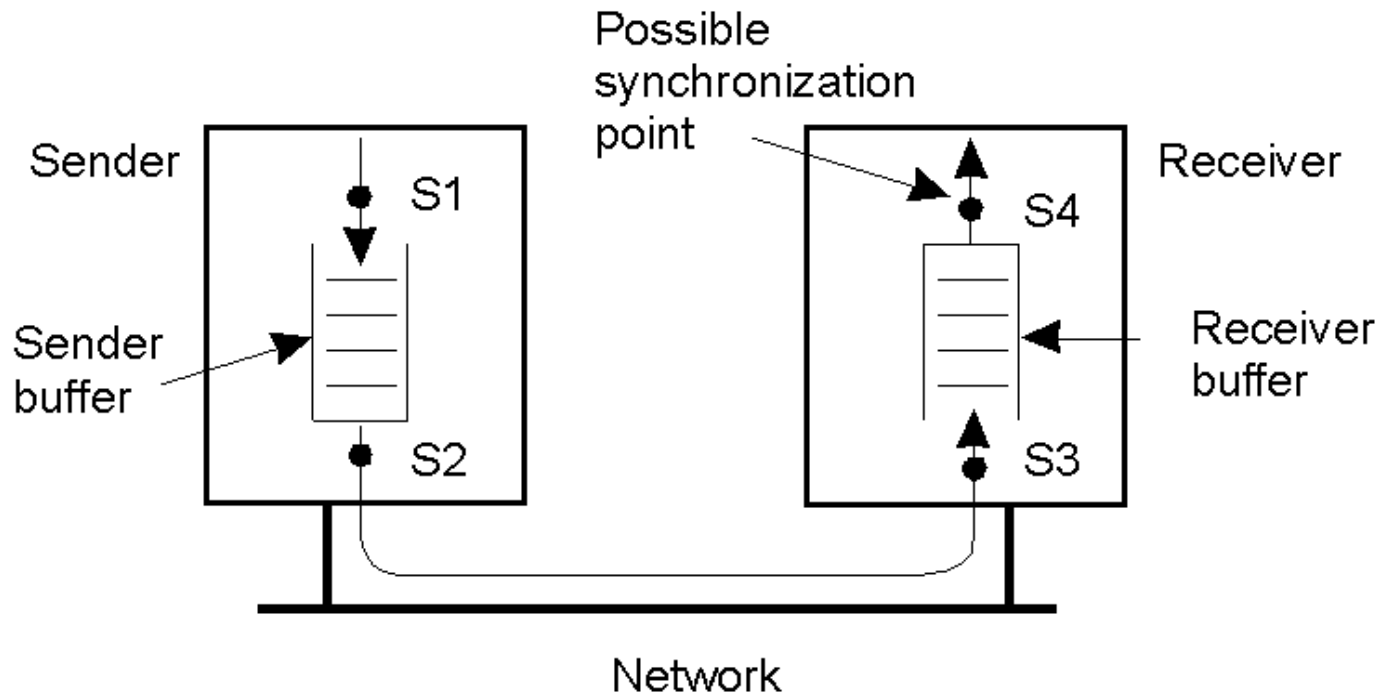
- Общая структура



Отсутствие общей разделяемой памяти

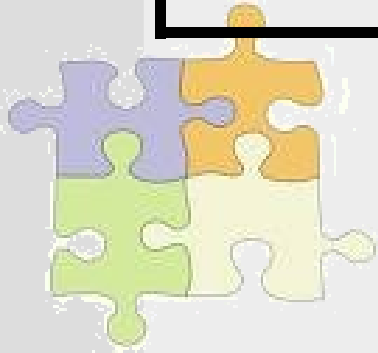
# Мультикомпьютерная операционная система

- Механизм передачи сообщений
- Блокировки и буферизация при обмене сообщениями



# Мультикомпьютерная операционная система

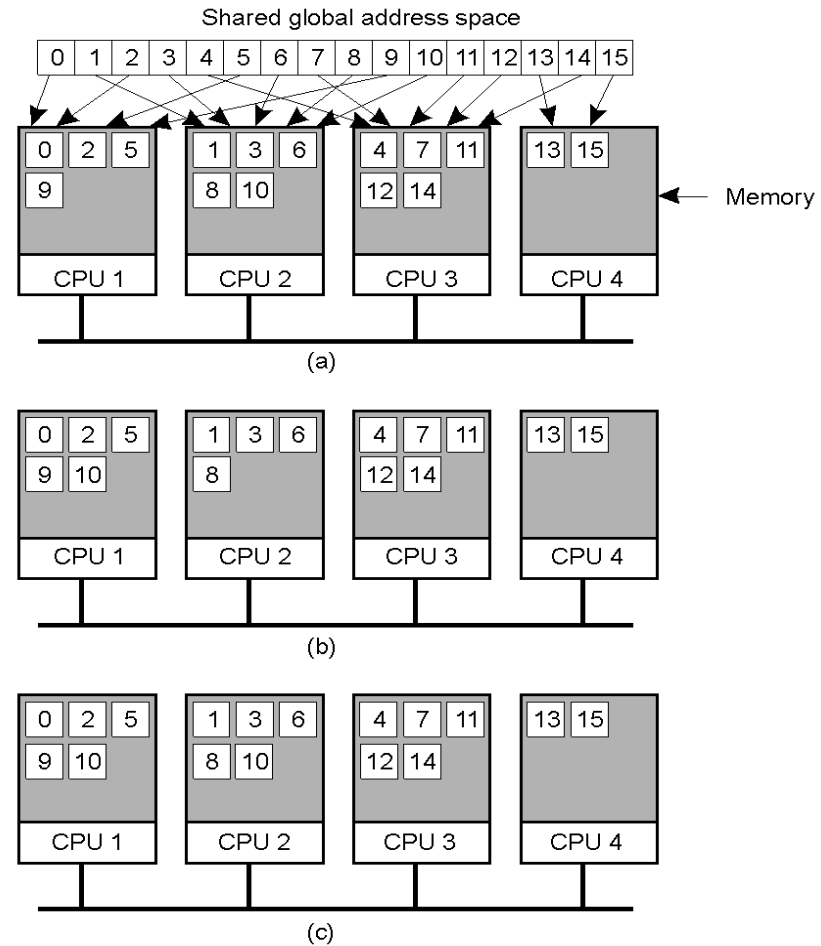
Точка синхронизации	Буферизация отправителя	Гарантия надежной связи
Блокировка отправителя до наличия свободного места в буфере	Да	Не требуется
Блокировка отправителя пока сообщение не будет отправлено	Нет	Не требуется
Блокировка отправителя до приема сообщения	Нет	Необходима
Блокировка отправителя до обработки сообщения	Нет	Необходима





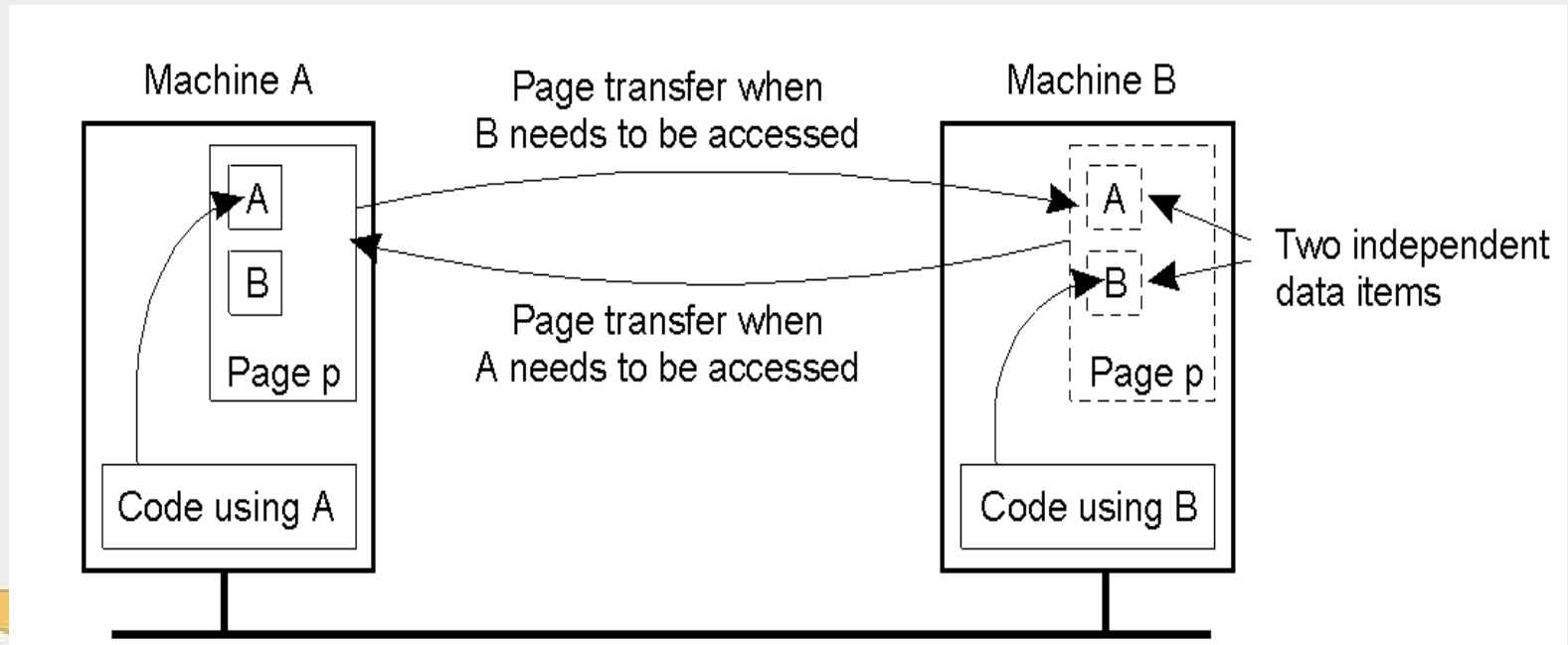
# Системы с разделяемой распределенной памятью

- a) Страницы памяти распределены между машинами
- b) Страница 10 перемещена при обращении
- c) Использование репликации для страниц закрытых на запись



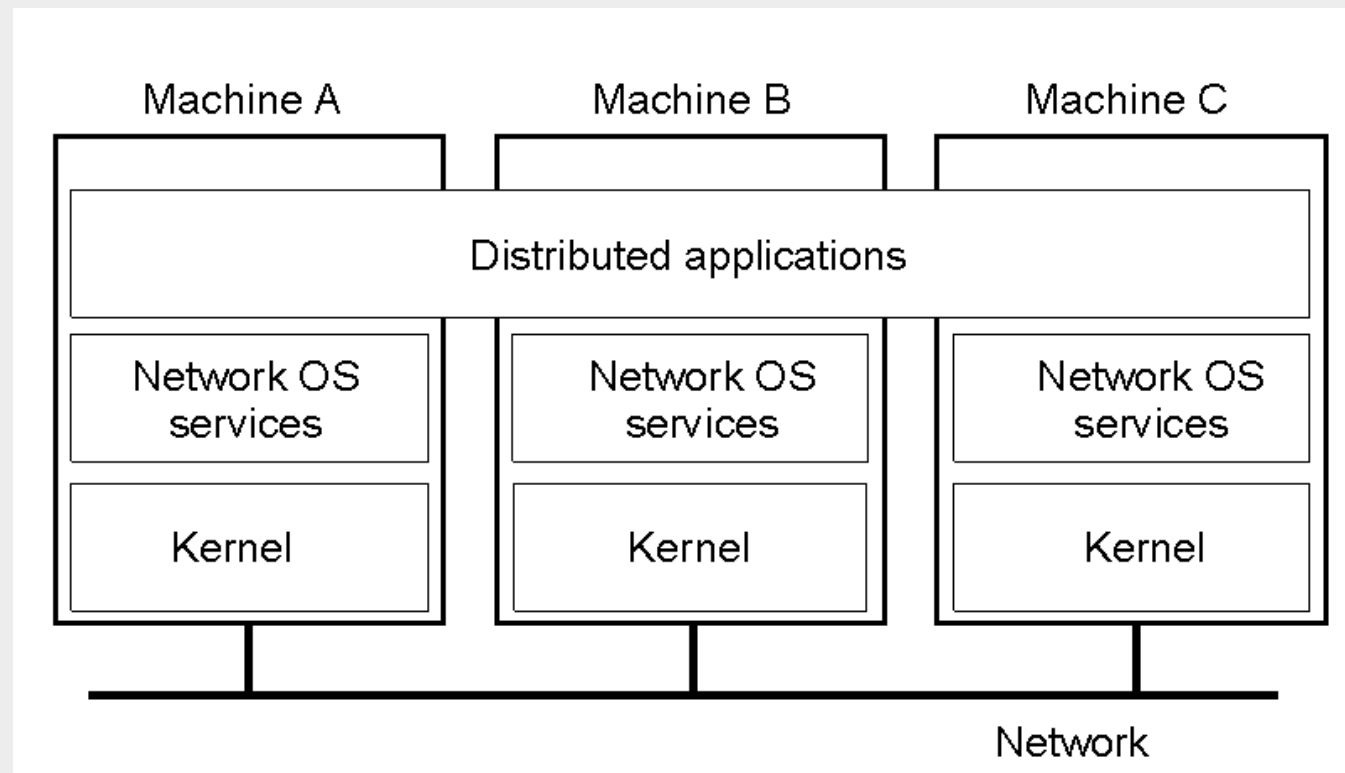
# Системы с разделяемой распределенной памятью

- Ошибочное разделение станицы между двумя процессами
- Уменьшение производительности



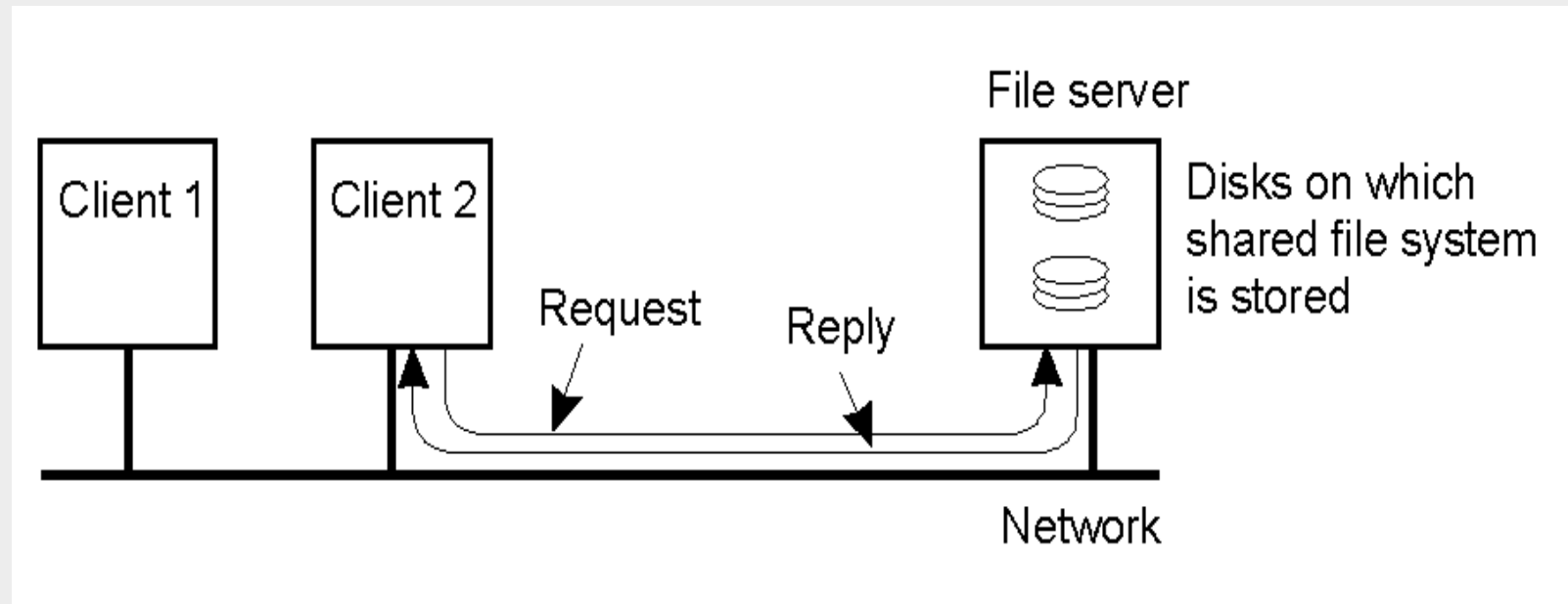
# Сетевые операционные системы

- Общая структура
- Сервисы: удаленное подключение, копирование файлов, глобальная общая файловая система



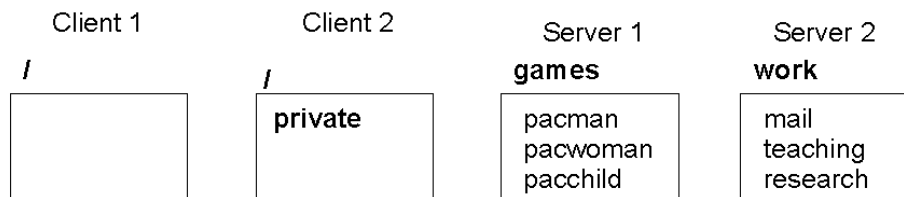
# Сетевые операционные системы

- Два клиента и сервер обмениваются файлами

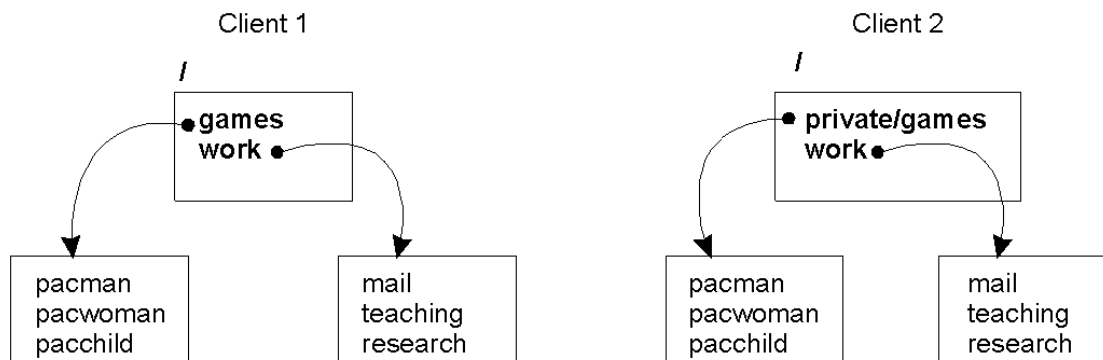


# Сетевые операционные системы

- Разные клиенты монтируют файловую систему по-разному
- Неполная прозрачность



(a)



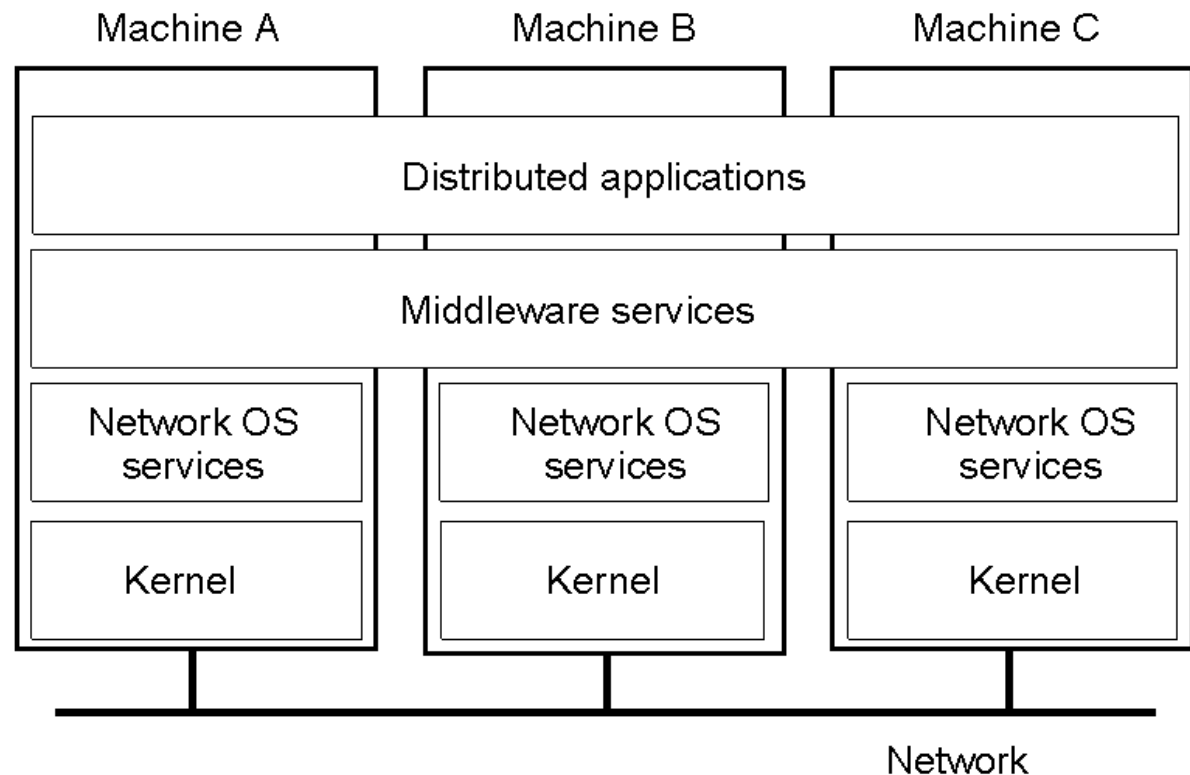
(b)

(c)

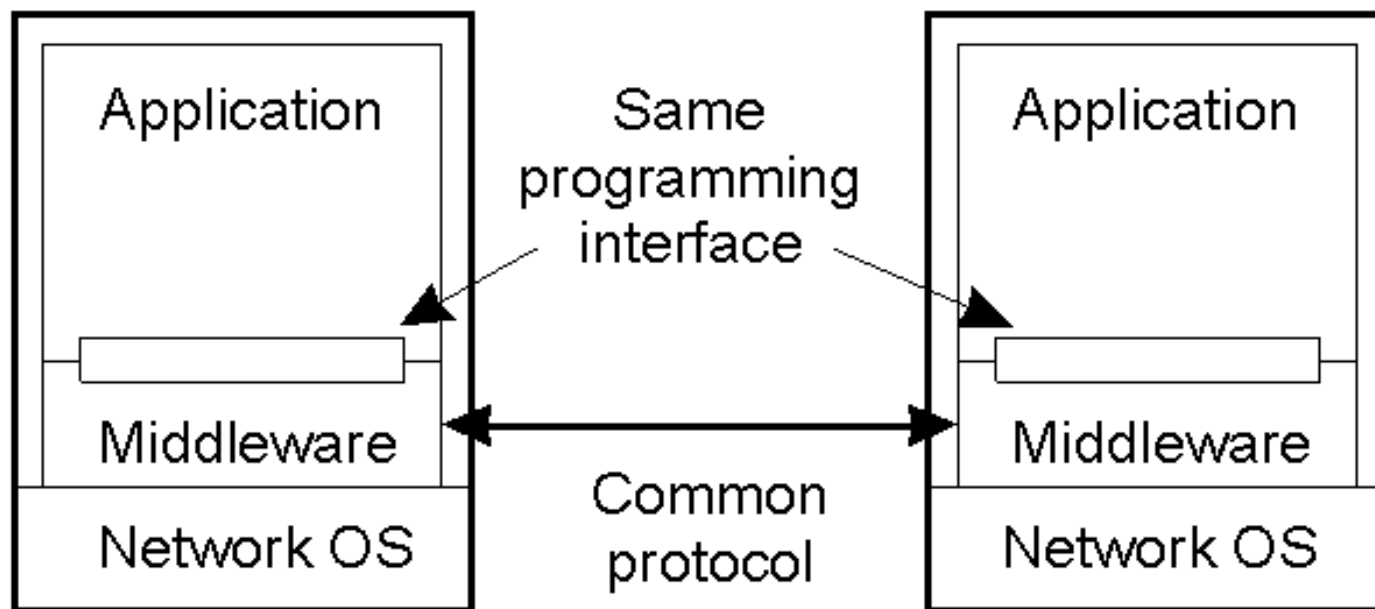


# Позиционирование промежуточного уровня

- Общая структура



# Промежуточный уровень и открытость



- Единый протокол(интерфейс) для взаимодействия с промежуточным уровнем



# Модели промежуточного уровня

---

- Распределенная файловая система
- Механизм вызова удаленных процедур (RPC)
- Модель распределенных объектов
- Модель распределенных документов (WWW)





# Сервисы промежуточного уровня

---

- Высокоуровневые средства связи
- Сервис именованя
- Средства хранения данных
- Распределенные транзакции
- Средства обеспечения защиты

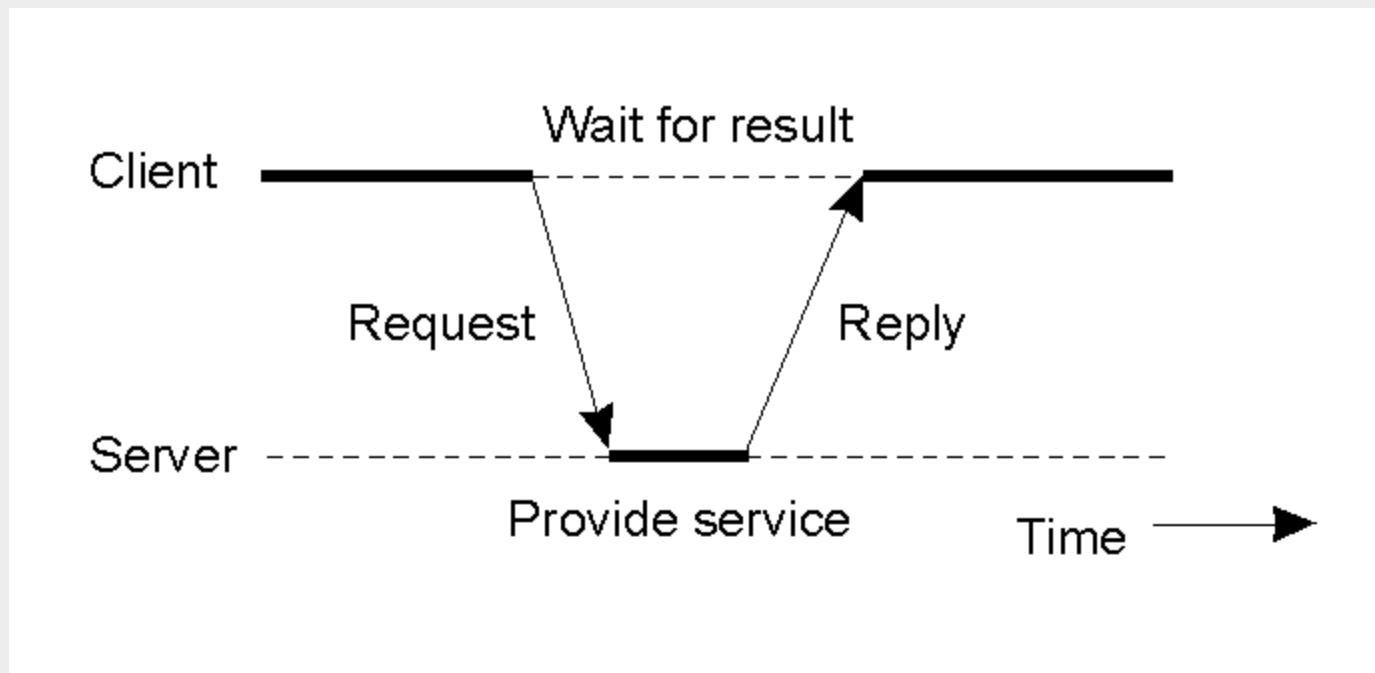


# Сравнение между системами

Характеристика	Распределенная ОС		Сетевая ОС	ОС с распредел. уровнем
	Мультипроц	Мультикомп.		
Уровень прозрачности	Очень высокий	Высокий	Низкий	Высокий
Идентичная ОС на всех узлах	Да	Да	Нет	Нет
Число копий ОС	1	N	N	N
Коммуникации на основе	Раздел.память	Сообщения	Файла	Зависит от модели
Управление ресурсами	Глобальное, центральное	Глобальное распредел	Отдельно на узле	Отдельно на узле
Масштабируемость	Отсутствует	Умеренная	Да	Различная
Открытость	Закрытая(Патент)	Закрытая	Открытая	Открытая

# Модель клиент-сервер

- Общая схема взаимодействия
- Режим работы «запрос-ответ»



# Пример взаимодействия клиент-сервер

```
/* Definitions needed by clients and servers. */
#define TRUE 1
#define MAX_PATH 255 /* maximum length of file name */
#define BUF_SIZE 1024 /* how much data to transfer at once */
#define FILE_SERVER 243 /* file server's network address */

/* Definitions of the allowed operations */
#define CREATE 1 /* create a new file */
#define READ 2 /* read data from a file and return it */
#define WRITE 3 /* write data to a file */
#define DELETE 4 /* delete an existing file */

/* Error codes. */
#define OK 0 /* operation performed correctly */
#define E_BAD_OPCODE -1 /* unknown operation requested */
#define E_BAD_PARAM -2 /* error in a parameter */
#define E_IO -3 /* disk error or other I/O error */

/* Definition of the message format. */
struct message {
    long source; /* sender's identity */
    long dest; /* receiver's identity */
    long opcode; /* requested operation */
    long count; /* number of bytes to transfer */
    long offset; /* position in file to start I/O */
    long result; /* result of the operation */
    char name[MAX_PATH]; /* name of file being operated on */
    char data[BUF_SIZE]; /* data to be read or written */
};
```



# Пример взаимодействия клиент-сервер

```
#include <header.h>
void main(void) {
    struct message m1, m2;           /* incoming and outgoing messages */
    int r;                           /* result code */

    while(TRUE) {                   /* server runs forever */
        receive(FILE_SERVER, &m1);  /* block waiting for a message */
        switch(m1.opcode) {         /* dispatch on type of request */
            case CREATE:             r = do_create(&m1, &m2); break;
            case READ:               r = do_read(&m1, &m2); break;
            case WRITE:              r = do_write(&m1, &m2); break;
            case DELETE:             r = do_delete(&m1, &m2); break;
            default:                 r = E_BAD_OPCODE;
        }
        m2.result = r;              /* return result to client */
        send(m1.source, &m2);       /* send reply */
    }
}
```

# Пример взаимодействия клиент-сервер

```
#include <header.h>
int copy(char *src, char *dst){
    struct message ml;
    long position;
    long client = 110;

    initialize( );
    position = 0;
    do {
        ml.opcode = READ;
        ml.offset = position;
        ml.count = BUF_SIZE;
        strcpy(&ml.name, src);
        send(FILESERVER, &ml);
        receive(client, &ml);

        /* Write the data just received to the destination file.
        ml.opcode = WRITE;
        ml.offset = position;
        ml.count = ml.result;
        strcpy(&ml.name, dst);
        send(FILE_SERVER, &ml);
        receive(client, &ml);
        position += ml.result;
    } while( ml.result > 0 );
    return(ml.result >= 0 ? OK : ml.result);
}
```

(a)

```
/* procedure to copy file using the server */
/* message buffer */
/* current file position */
/* client's address */

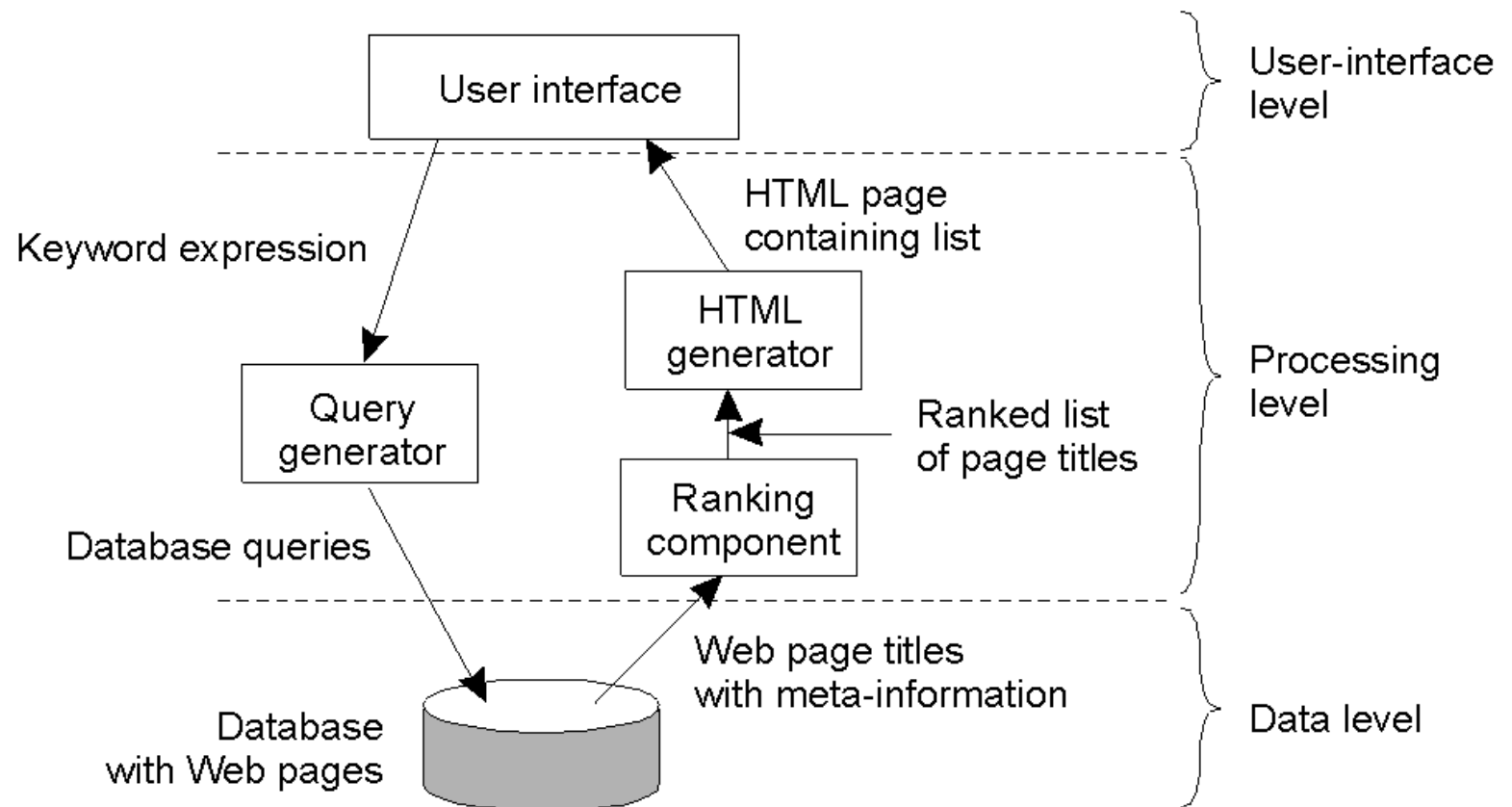
/* prepare for execution */

/* operation is a read */
/* current position in the file */
/* how many bytes to read*/
/* copy name of file to be read to message */
/* send the message to the file server */
/* block waiting for the reply */

/* operation is a write */
/* current position in the file */
/* how many bytes to write */
/* copy name of file to be written to buf */
/* send the message to the file server */
/* block waiting for the reply */
/* ml.result is number of bytes written */
/* iterate until done */
/* return OK or error code */
```

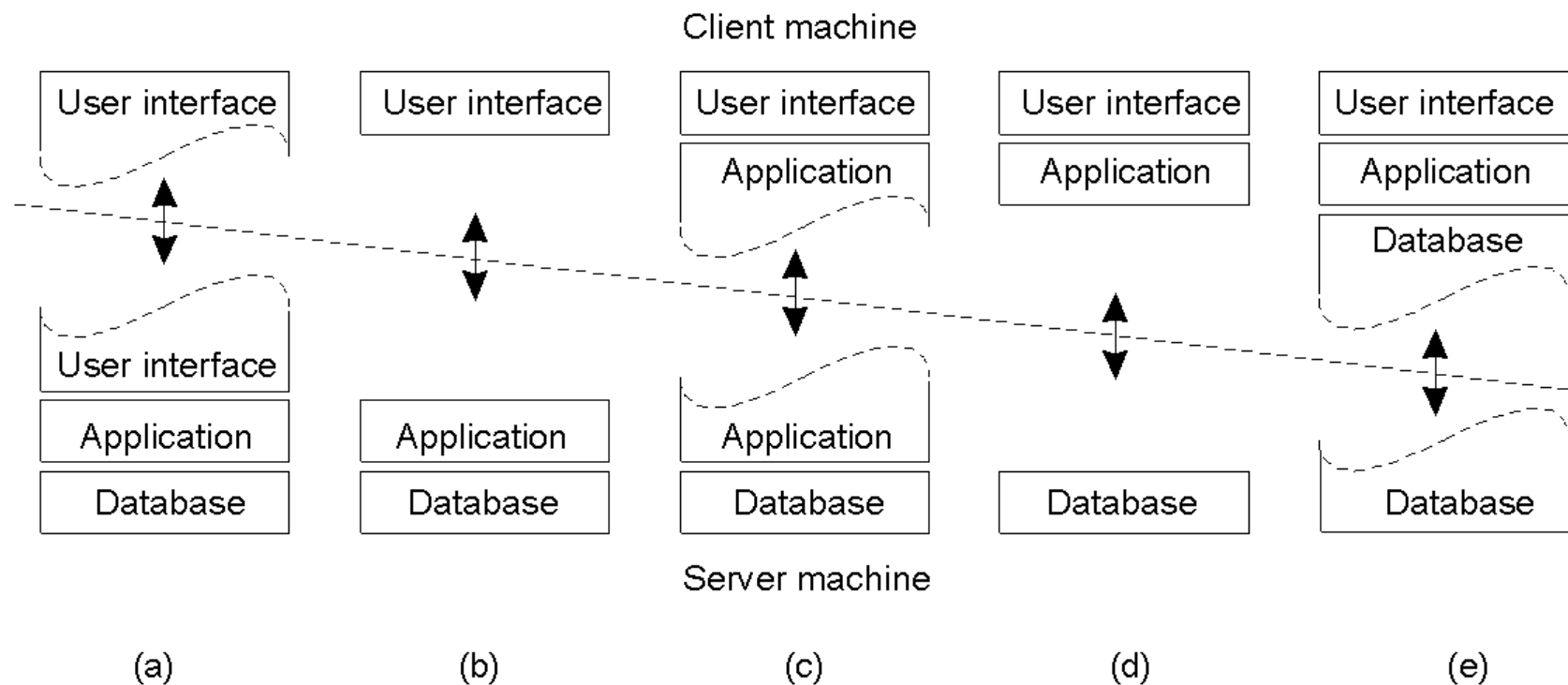
# Уровни обработки

## ■ Пример работы поисковой системы



# Многоуровневые архитектуры

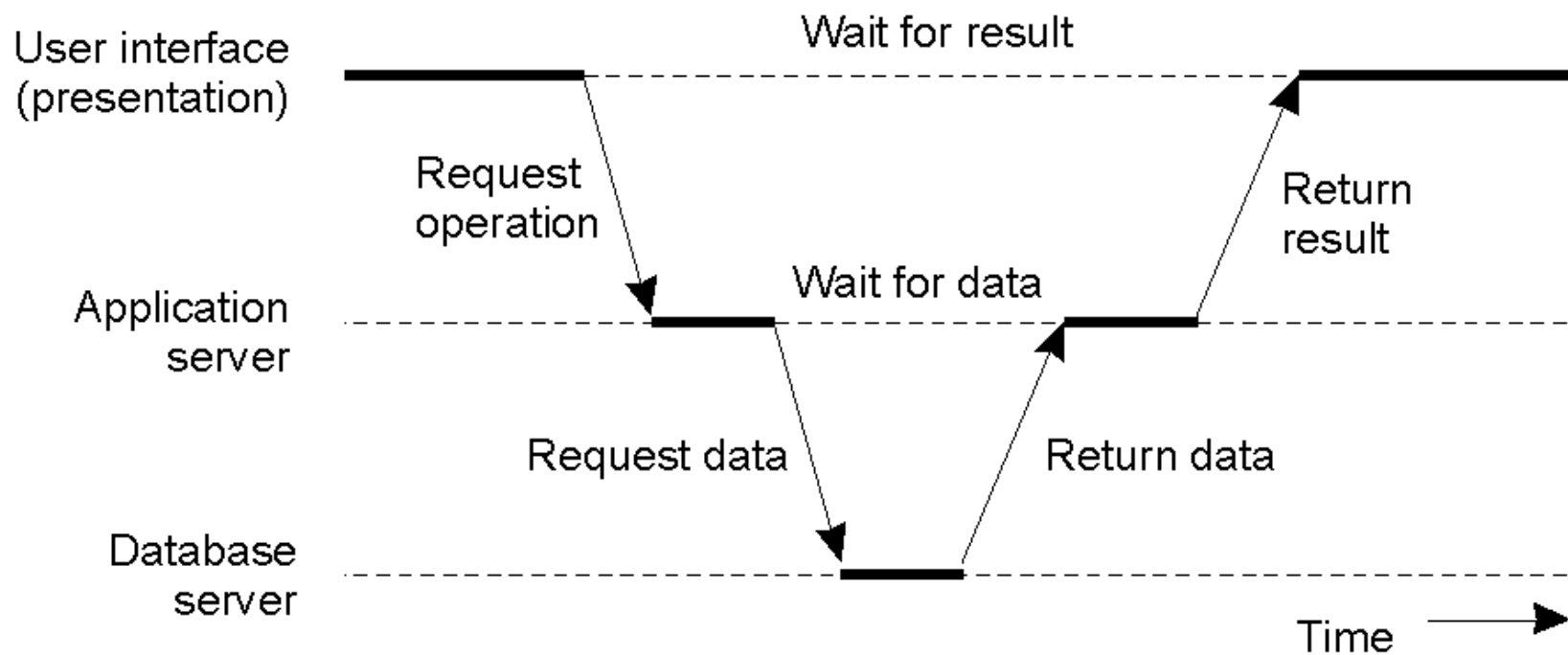
- Альтернативные варианты взаимодействия клиент-сервер





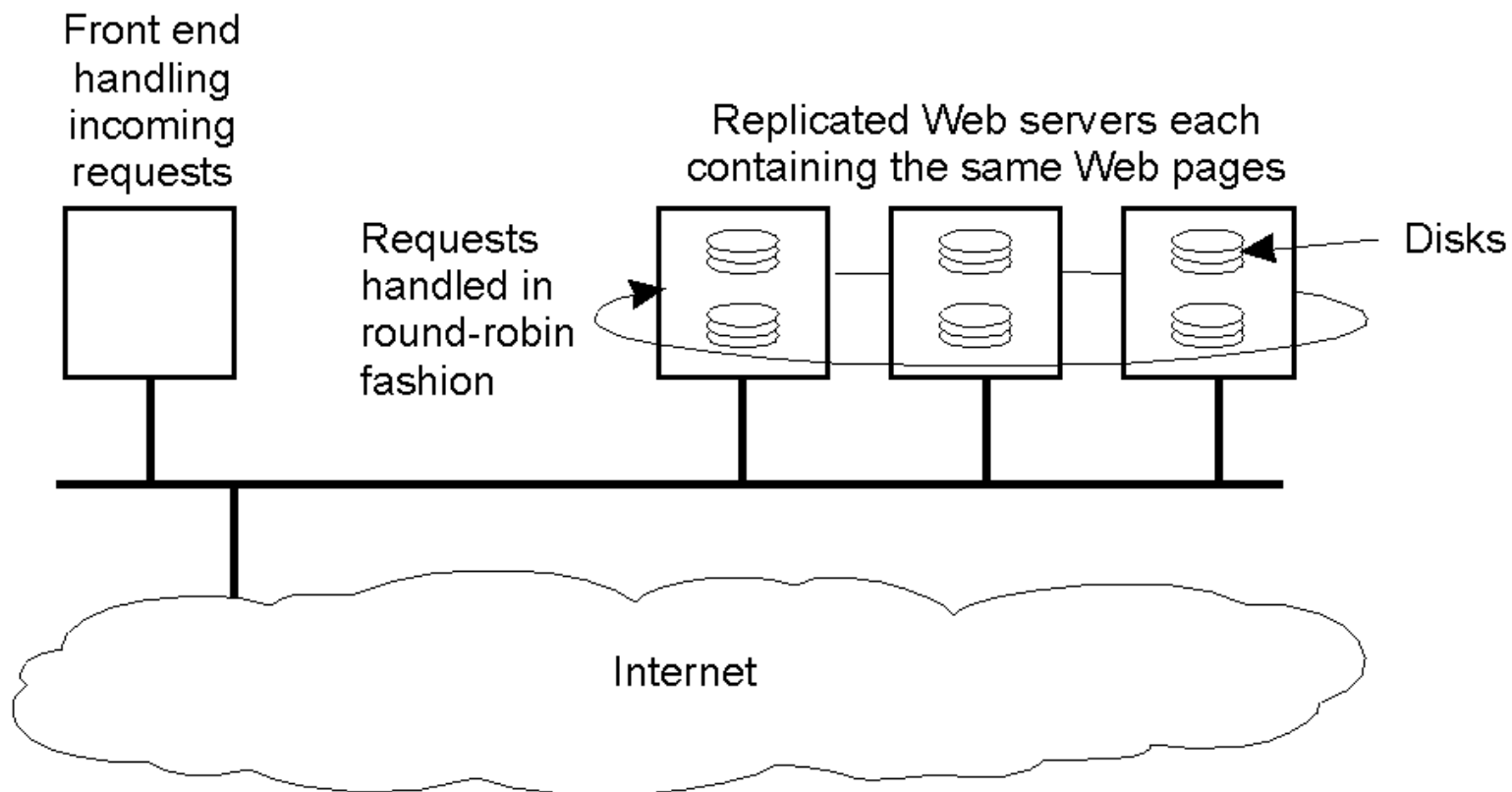
# Многоуровневые архитектуры

- Сервер действует как клиент



# Современные архитектуры

- Пример горизонтального распределения веб-сервиса



Вопросы?

---