Операционные системы

Многопроцессорные системы [1]

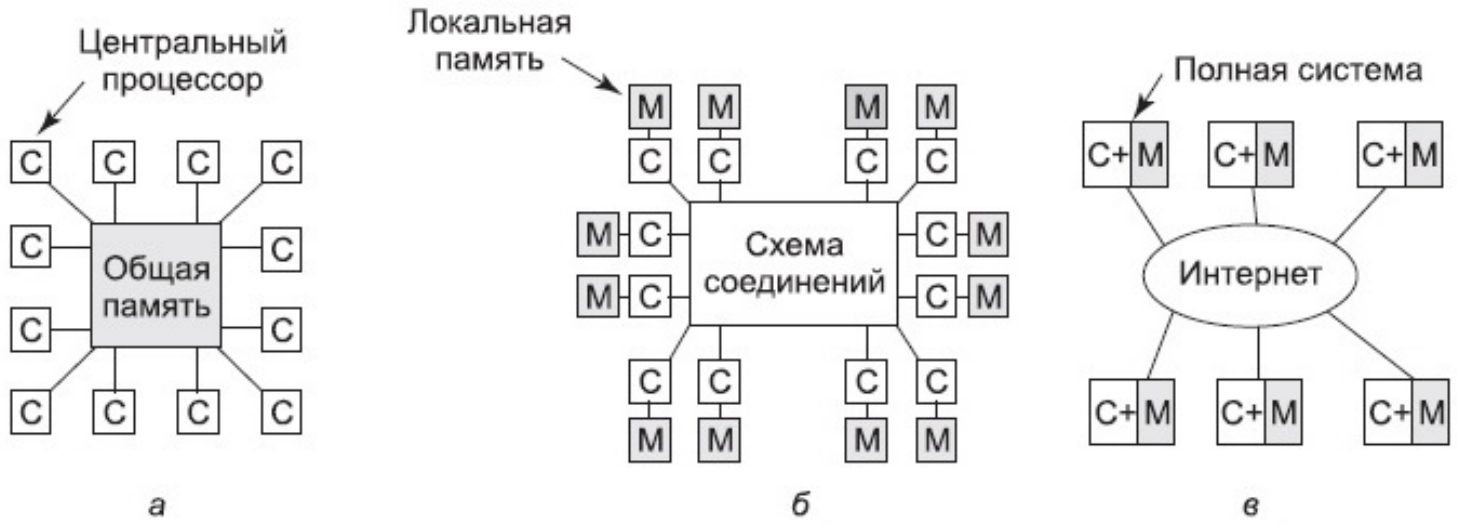
Басин А. Н., Соловьев А. В.

ПетрГУ

(Rev. 2018 06 07)

[1] По материалам «Таненбаум Э., Бос Х. Современные операционные системы. СПб.: Питер, 2015.»

Модели мультикомпьютера



Мультипроцессорная система: а — с общей памятью; б — с передачей сообщений; в — глобальная распределённая система

Мультипроцессоры

Мультипроцессоры

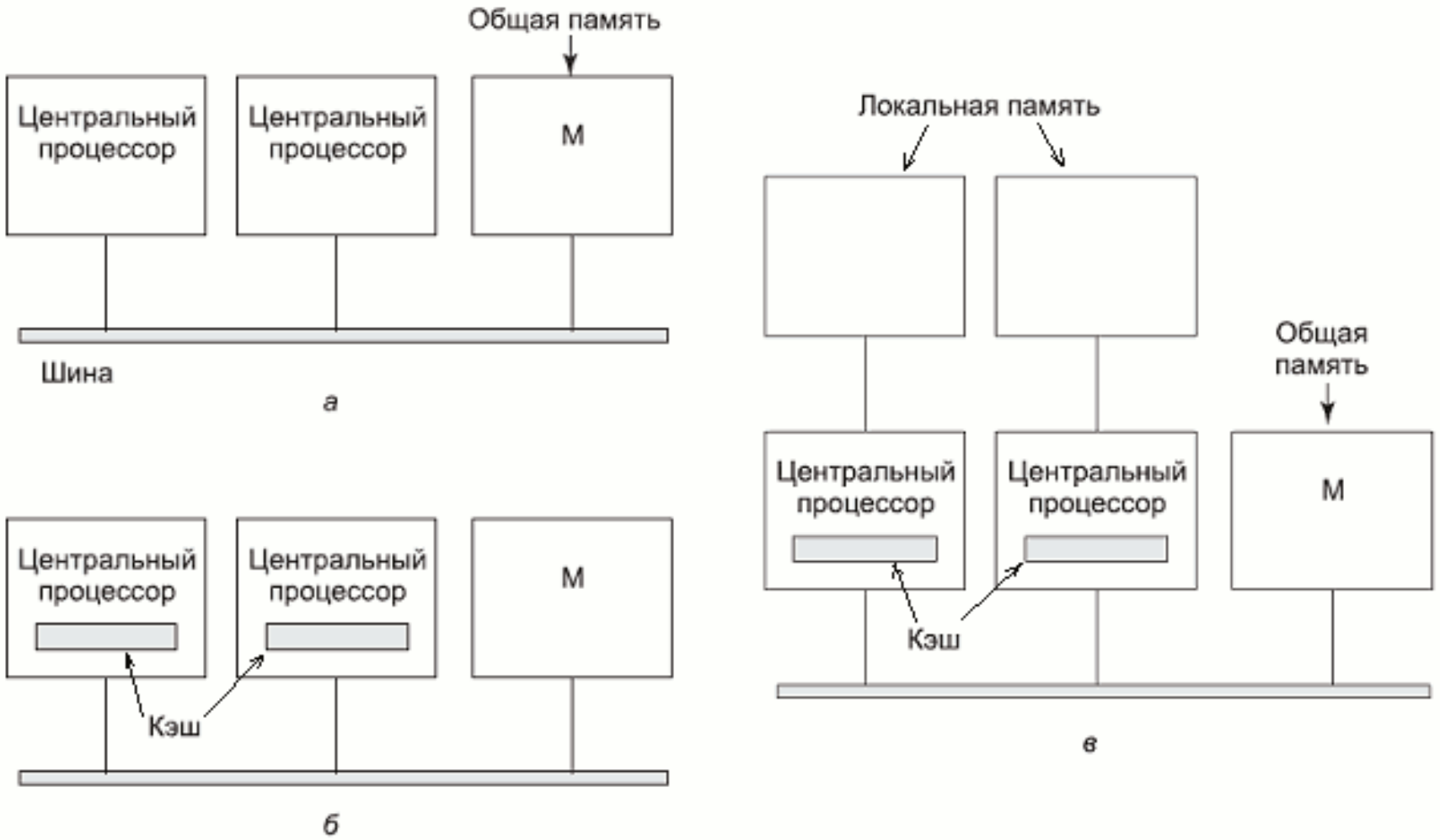
Мультипроцессор с общей памятью (shared-memory multiprocessor) — компьютерная система, в которой два и более ЦП имеют полный доступ к общей оперативной памяти.

UMA-мультипроцессоры (Uniform Memory Access — однородный доступ к памяти): каждое слово памяти может быть считано так же быстро, как и любое другое слово памяти.

NUMA-мультипроцессоры (Non-uniform Memory Access — неоднородный доступ к памяти)

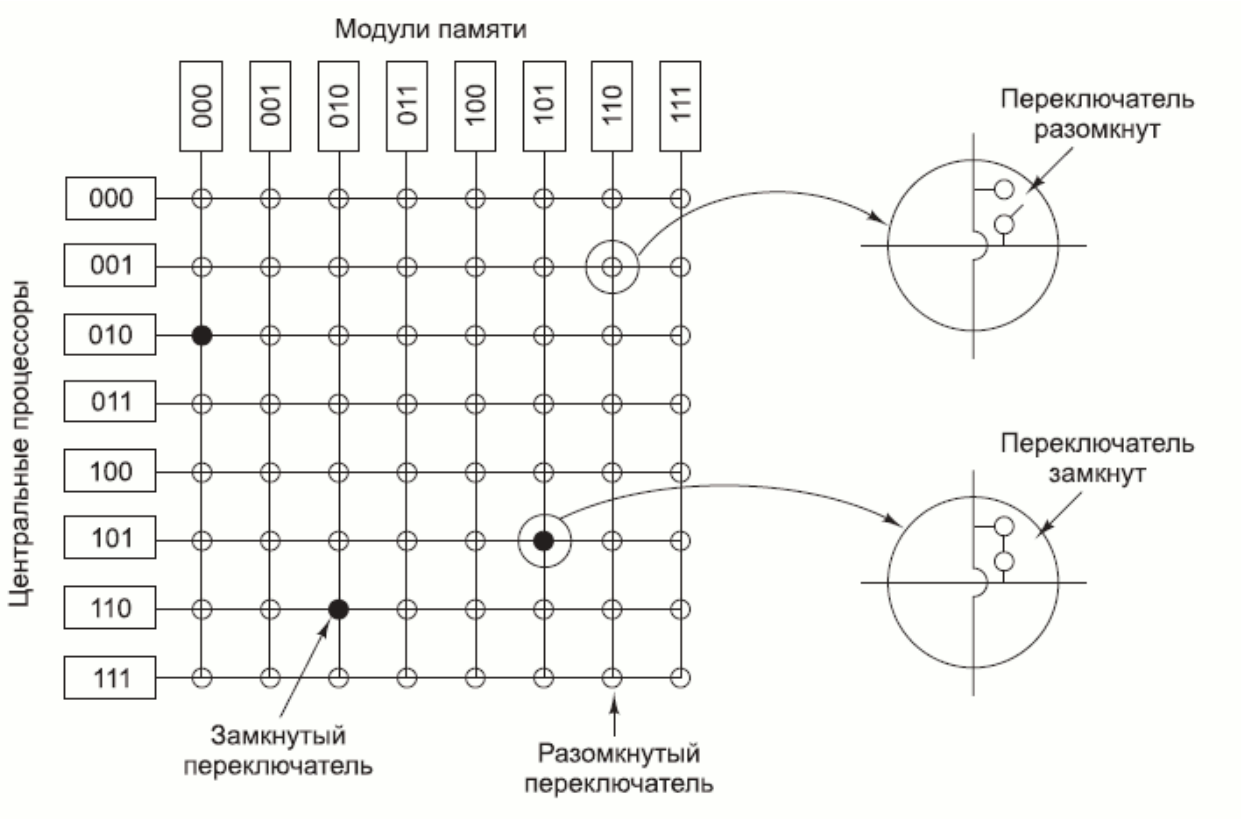
Мультипроцессоры с общей шиной

Мультипроцессоры с общей шиной: а — без кэш-памяти; б — с кэш-памятью; в — с кэш-памятью и собственной памятью процессора



Мультипроцессоры с координатным коммутатором

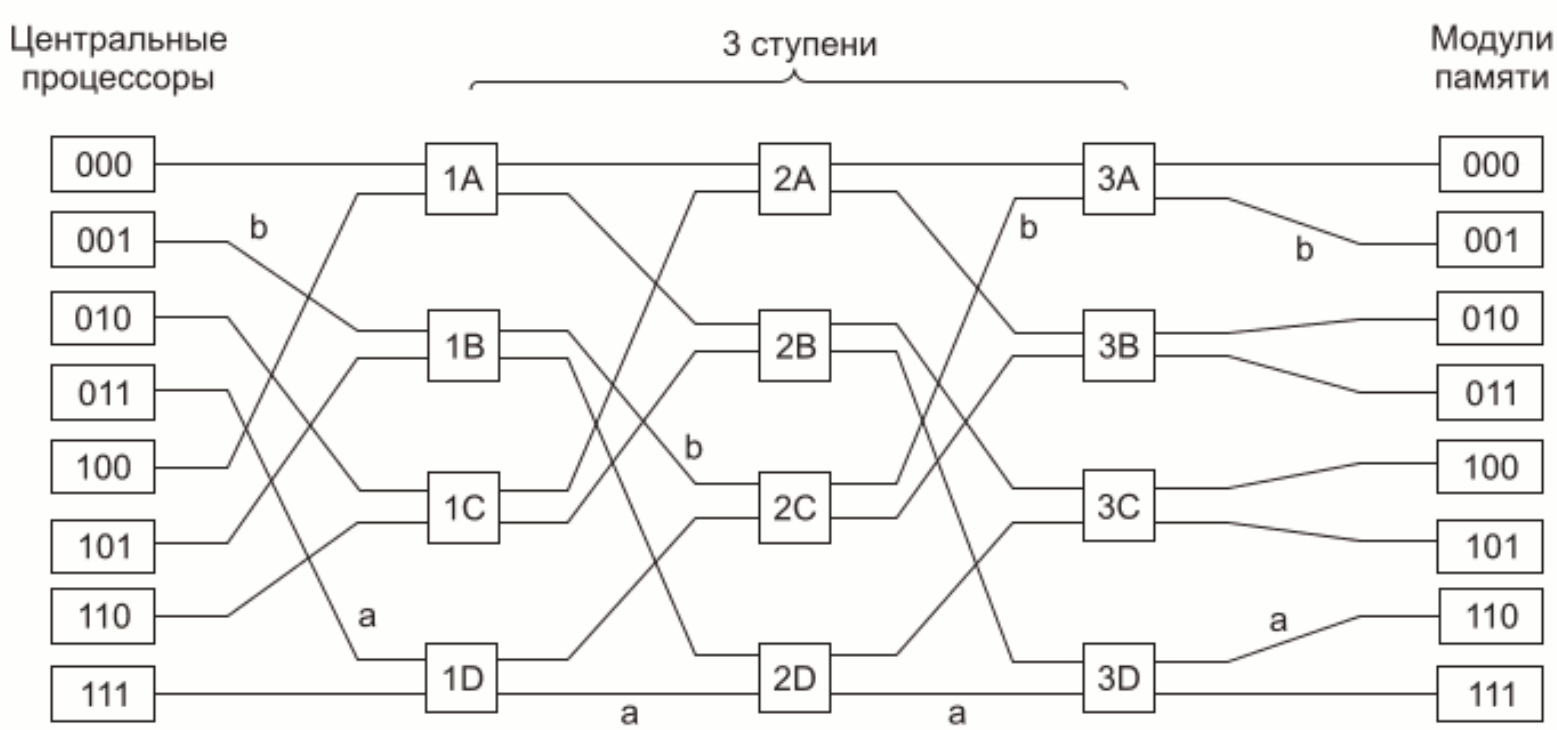
Координатный коммутатор: n^2 элементов коммутации



Неблокирующаяся сеть: ни одному ЦП не будет отказано в необходимом ему подключении по причине занятости какого-то элемента коммутации или линии (если предположить, что свободен сам требуемый модуль памяти).

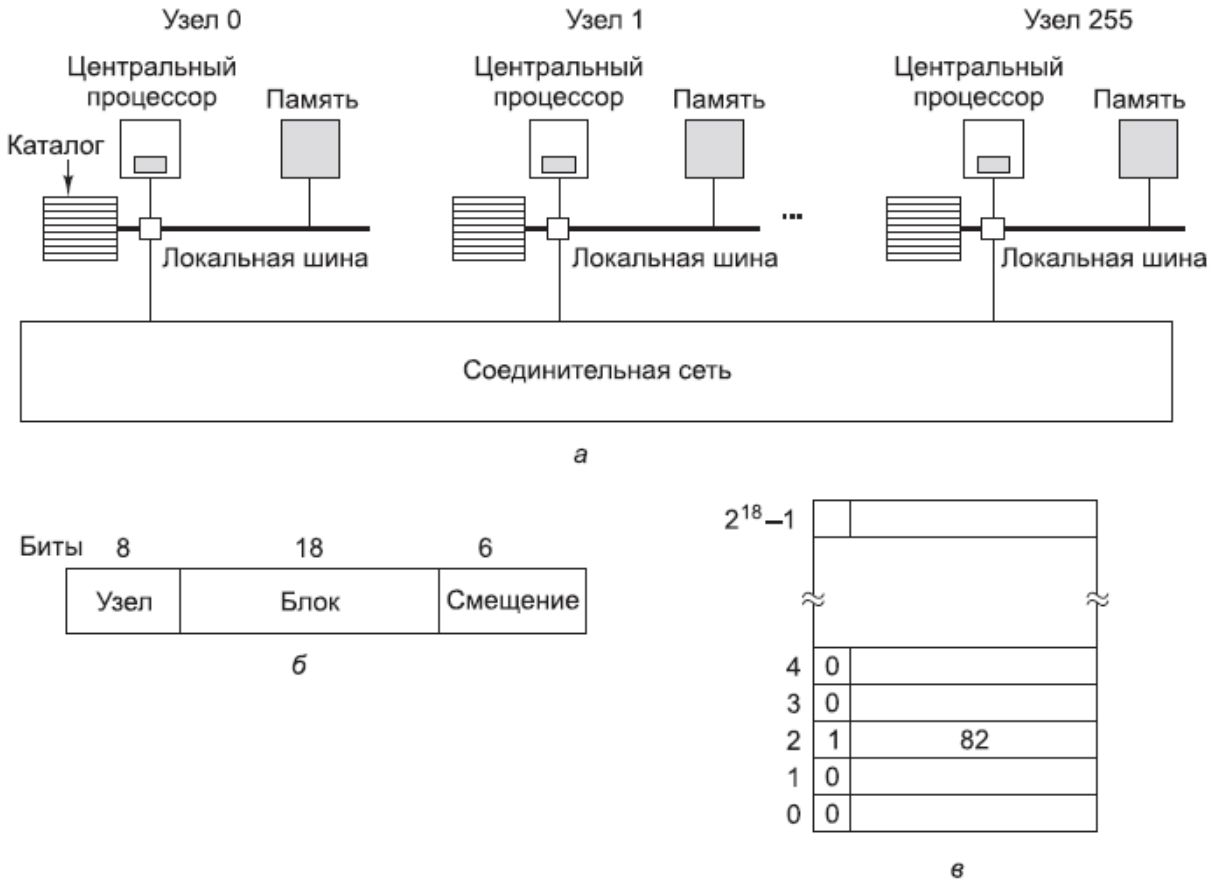
Многоступенчатые схемы коммутации

С помощью коммутатора 2×2 можно построить многоступенчатые коммутаторные сети. Например, *сеть омега*: $\frac{n}{2} \log_2 n$ коммутаторов



a – маршрут коммутации для ЦП 011 к модулю памяти 110.
b – маршрут коммутации для ЦП 001 к модулю памяти 001.
Блокирующаяся сеть – не все наборы запросов могут обрабатываться одновременно.

Мультипроцессоры NUMA



а — мультипроцессор на основе каталогов, содержащий 256 узлов; б — разделение 32-разрядного адреса памяти на поля; в — каталог в узле 36

Многоядерные микропроцессоры

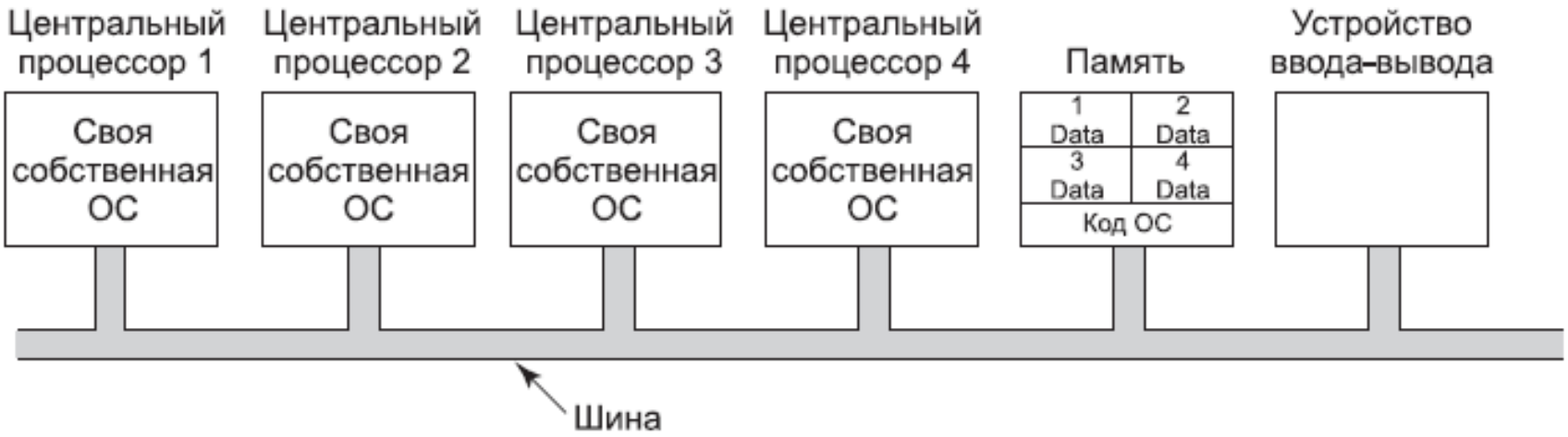
Есть возможность размещения на одной и той же микросхеме (технически на одном и том же кристалле) двух и более полноценных ЦП, обычно называемых *ядрами* (cores).

Барьер согласованности (coherency wall): оборудование, необходимое для сохранения согласованности кэшей многоядерных ЦП, становится очень сложным и весьма дорогостоящим; стоимость протокола согласованности в оборудовании будет столь высока, что все эти ядра не помогут существенно поднять производительность, потому что процессоры будут слишком заняты поддержанием кэшей в согласованном состоянии.
Гибридная модель: Например, ЦП с 1024 ядрами может быть поделен на 64 островка, каждый из которых имеет 16 ядер с согласованными кэшами при отказе от согласовенности кэшей между этими островками.

GPU, CUDA, ...

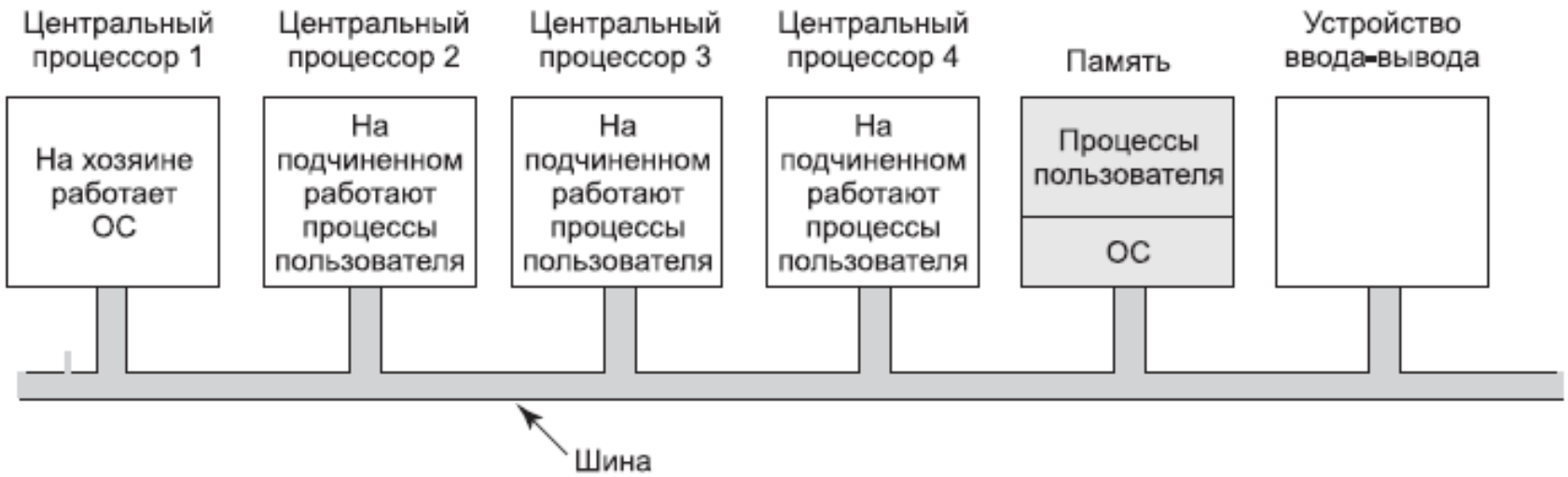
Типы мультипроцессорных операционных систем

Использование собственной ОС для каждого ЦП



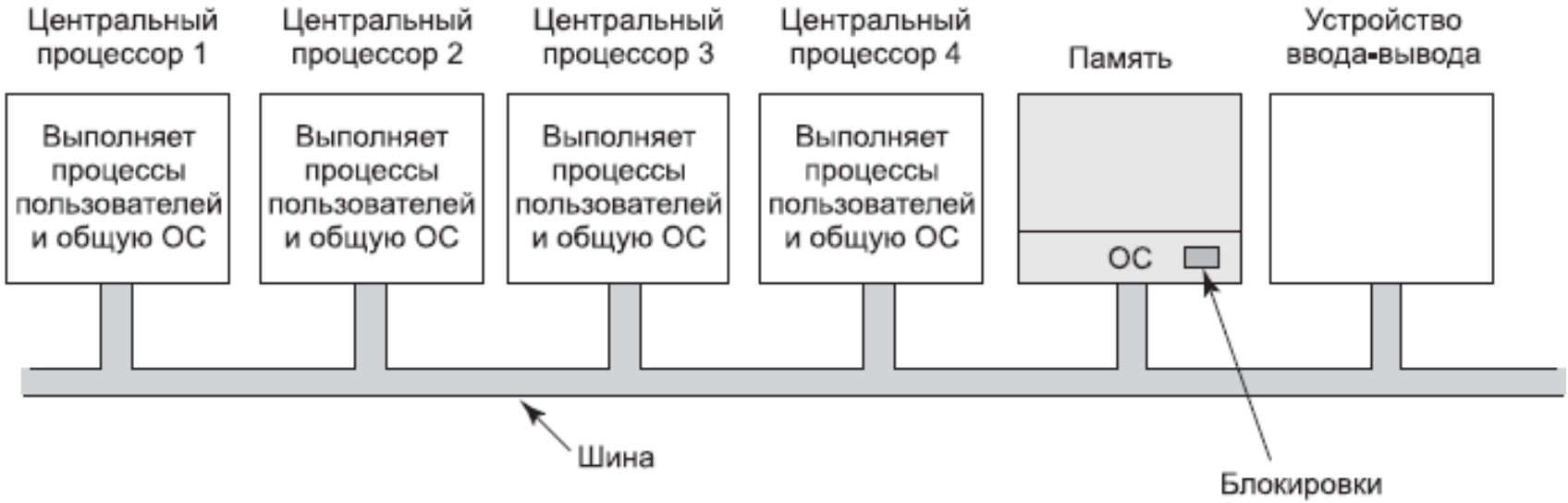
- Негибкое распределение ресурсов: у каждой ОС свой собственный набор процессов, таблиц страниц и т. п.
- Проблемы балансировки нагрузки.
- Проблемы согласования и синхронизации.

Схема «главный — подчиненный»



- Единые структуры данных (набор процессов, таблицы страниц).
- ЦП 1 – узкое место, при большом кол-ве процессов он может быть перегружен.

Симметричные мультипроцессоры (SMP)



- Балансировка процессов и памяти осуществляется в динамическом режиме.
- Отсутствует узкое место, связанное с главным ЦП.
- Требуется поддержка (совместимость) со стороны ОС: если два или более ЦП запускают код ОС в одно и то же время...

Большая блокировка ядра (big kernel lock);
отдельные мьютексы для независимых критических областей ядра; ...

Синхронизация мультипроцессоров

Запрет прерываний – не подходит!!!

Реализация спин-блокировки:
Необходима команда типа TSL. Кроме того, она должна корректно поддерживаться системной шиной (альтернатива – алгоритм Петерсона).

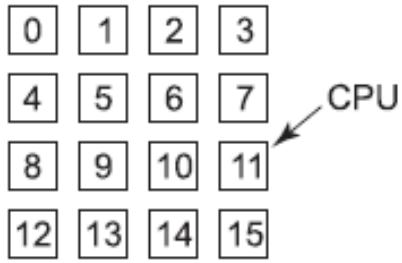
Что лучше, ожидание в цикле или переключение?

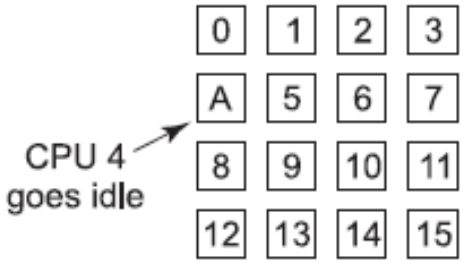
- На циклический опрос впустую тратится время ЦП.
- На переключение также тратится время ЦП (сохранить состояние текущего потока, получить блокировку списка готовых к работе процессов, выбрать поток, загрузить его состояние и запустить этот поток), при этом возникнут кэш-промахи и ошибки TLB.

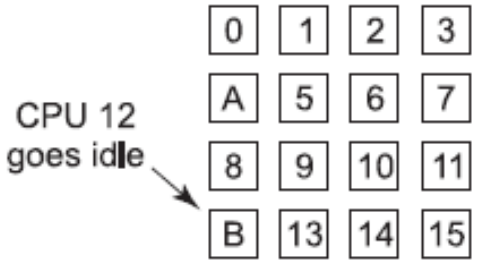
Необходима стратегия принятия решения – ждать или переключаться.
(У x86 есть инструкция MONITOR/MWAIT).

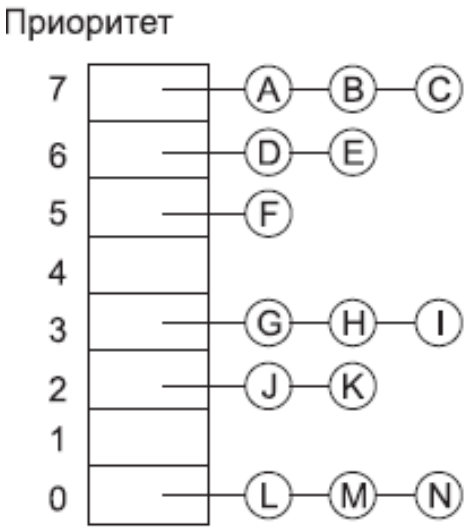
Планирование работы мультипроцессора (1)

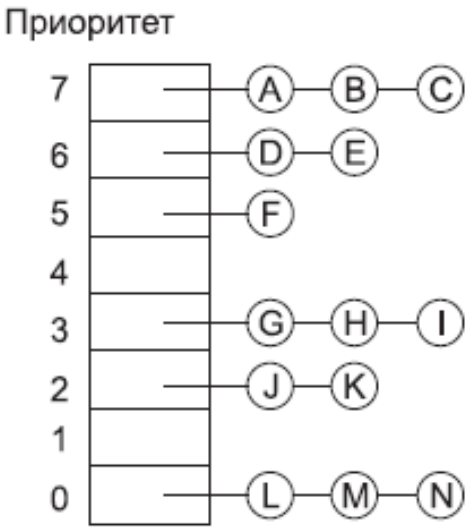
Использование единой структуры данных:

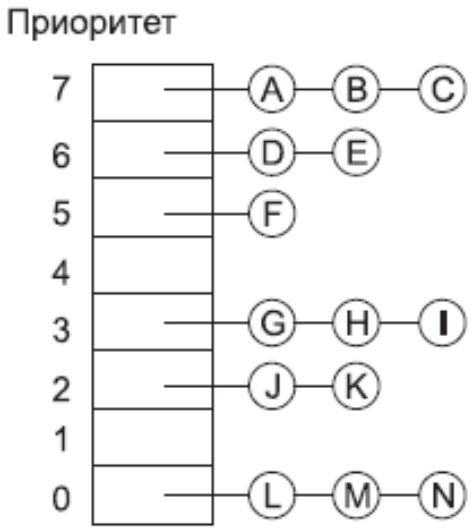












- Конкуренция за доступ к структуре данных, используемой при планировании.
- Издержки при переключении контекста, если поток блокируется на вв-выв.

Планирование работы мультипроцессора (2)

Разумное планирование (smart scheduling) – поток, получивший спин-блокировку, устанавливает флажок, видимый всему процессу, тогда планировщик не останавливает поток по истечении кванта времени, а даёт ему ещё немного времени на завершение выполнения кода в критической области и освобождение блокировки.

Родственное планирование (affinity scheduling) – стремиться выполнять поток на том же самом ЦП, на котором он запускался в последний раз, поскольку в его кэше все ещё могут находиться нужные потоку блоки, а в TLB также могут содержаться нужные страницы, что сокращает количество отказов TLB. Один из вариантов реализации этой техники – *двухуровневый алгоритм планирования* (two-level scheduling algorithm).

- Не препятствует равномерному распределению нагрузки между ЦП.
- По возможности используется родственность содержимого кэша запускаемому потоку.
- У каждого ЦП свой список готовых потоков – к минимуму сводится конкуренция за доступ к спискам.

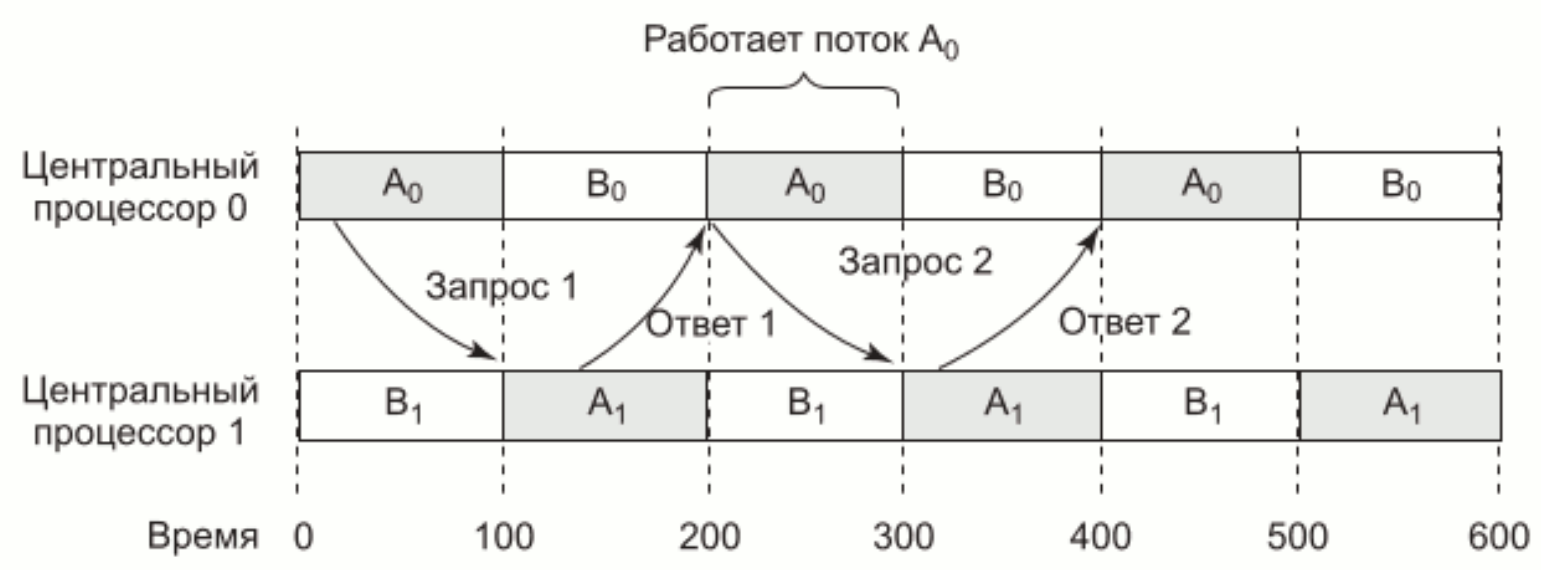
Планирование работы мультипроцессора (3)

Если потоки процесса часто обмениваются данными, то полезнее будет их выполнять одновременно. Планирование одновременной работы нескольких потоков на нескольких ЦП называется *совместным использованием пространства*:

Простейший вариант: процесс запрашивает под свои потоки некоторое количество ЦП и либо получает их все, либо вынужден ждать, пока они не освободятся. Альтернатива – активное управление степенью параллельности: приложение опрашивает центральный сервер, чтобы узнать, сколько ЦП оно может использовать, затем подгоняет количество потоков под количество доступных ЦП.

Планирование работы мультипроцессора (4)

Обмен данными между двумя запущенными не в фазе потоками, принадлежащими процессу A (проблема независимого планирования потоков):



Бригадное планирование (gang scheduling):

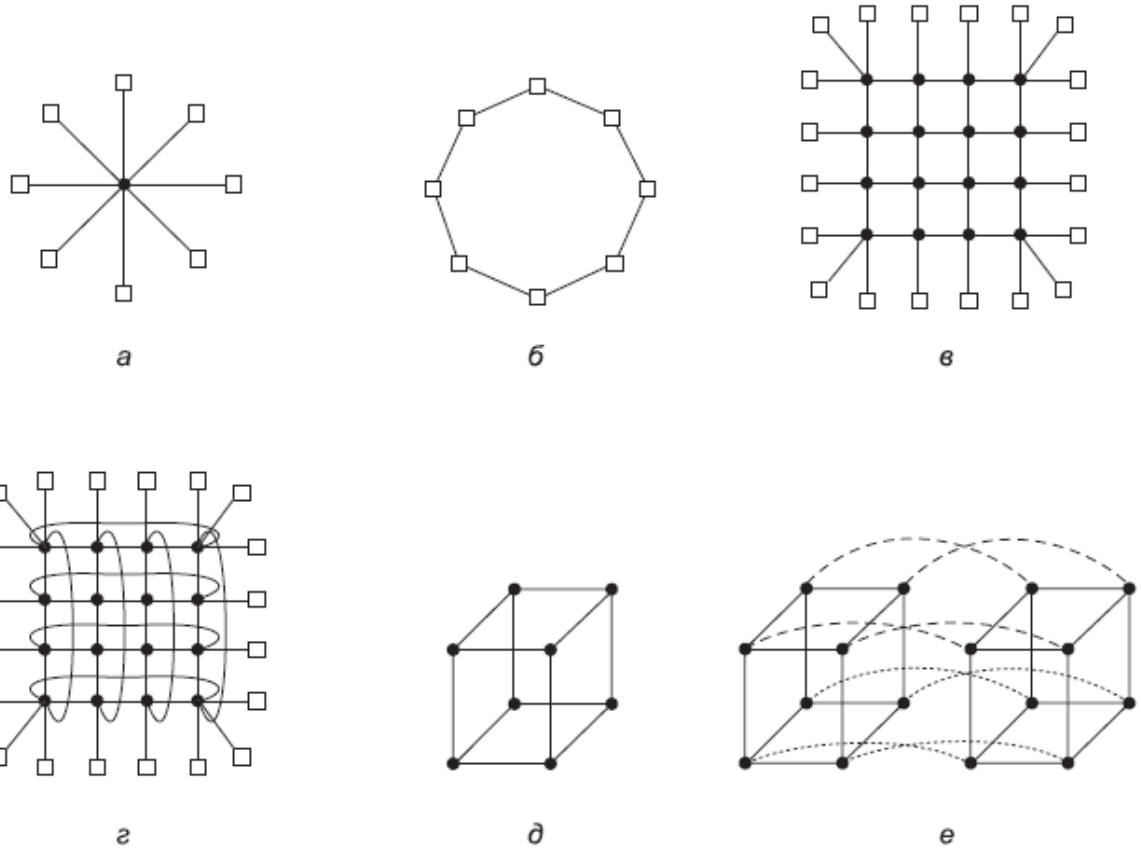
- Группа взаимосвязанных потоков планируется совместно.
- Все члены бригады запускаются одновременно на разных ЦП, работающих в режиме разделения времени.
- У всех членов бригады кванты времени начинаются и заканчиваются одновременно.

Мультикомпьютеры

Аппаратное обеспечение мультикомпьютеров (1)

Несколько *headless node* + коммуникации.

Топологии соединений:



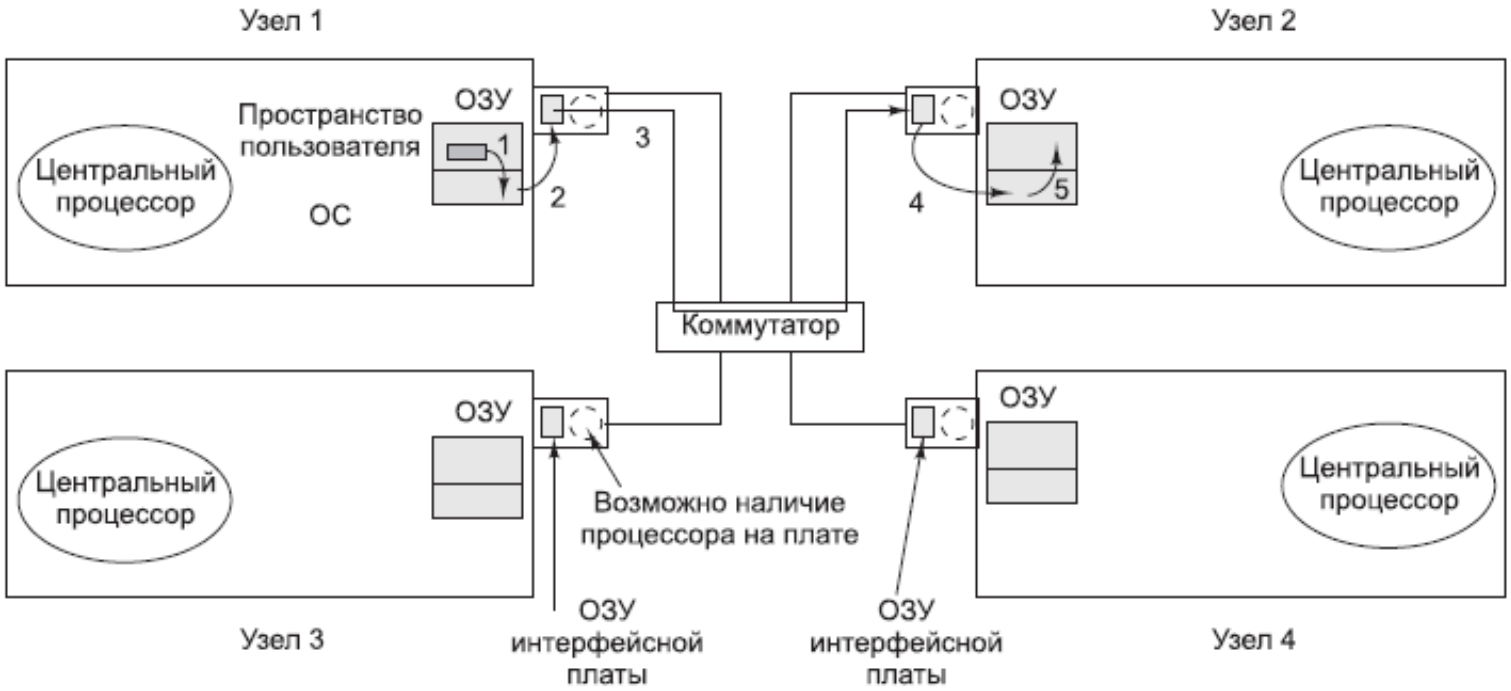
а — с одним коммутатором; б — кольцо; в — решетка; г — двойной тор; д — куб; е — четырехмерный куб

Аппаратное обеспечение мультикомпьютеров (2)

Схемы коммутации:

- *Коммутация пакетов с промежуточным хранением* (store-and-forward packet switching): пакет доставляется на первый коммутатор интерфейсной сетевой платой узла-источника, затем пакет копируется на линию, ведущую к следующему коммутатору на маршруте доставки, ... Когда пакет прибывает на коммутатор, подключенный к узлу-получателю, он копируется на карту сетевого интерфейса этого узла и в конечном счете попадает в его оперативную память.
- *Коммутация каналов* (circuit switching): предварительно устанавливается маршрут через все коммутаторы к коммутатору назначения. Как только маршрут будет установлен, биты без задержки с максимально возможной скоростью проследуют по всему маршруту от источника к получателю. Никакой буферизации на промежуточных коммутаторах не осуществляется. После того как пакет будет отправлен, маршрут может быть снова закрыт.

Проблема буферизации



Копирование данных через уровни проходит безопасно, но не всегда эффективно. Каждая копия влечет за собой издержки, связанные не только с самим копированием, но и с загрузкой кэша, TLB и т. д. Как следствие, задержка в таких сетевых соединениях получается довольно большой.

Низкоуровневые коммуникационные программы

Отображать память интерфейсной платы непосредственно на пользовательское пространство? (повышение производительности; проблема совместного использования несколькими процессами; проблема доступа ядра к памяти интерфейсной платы).

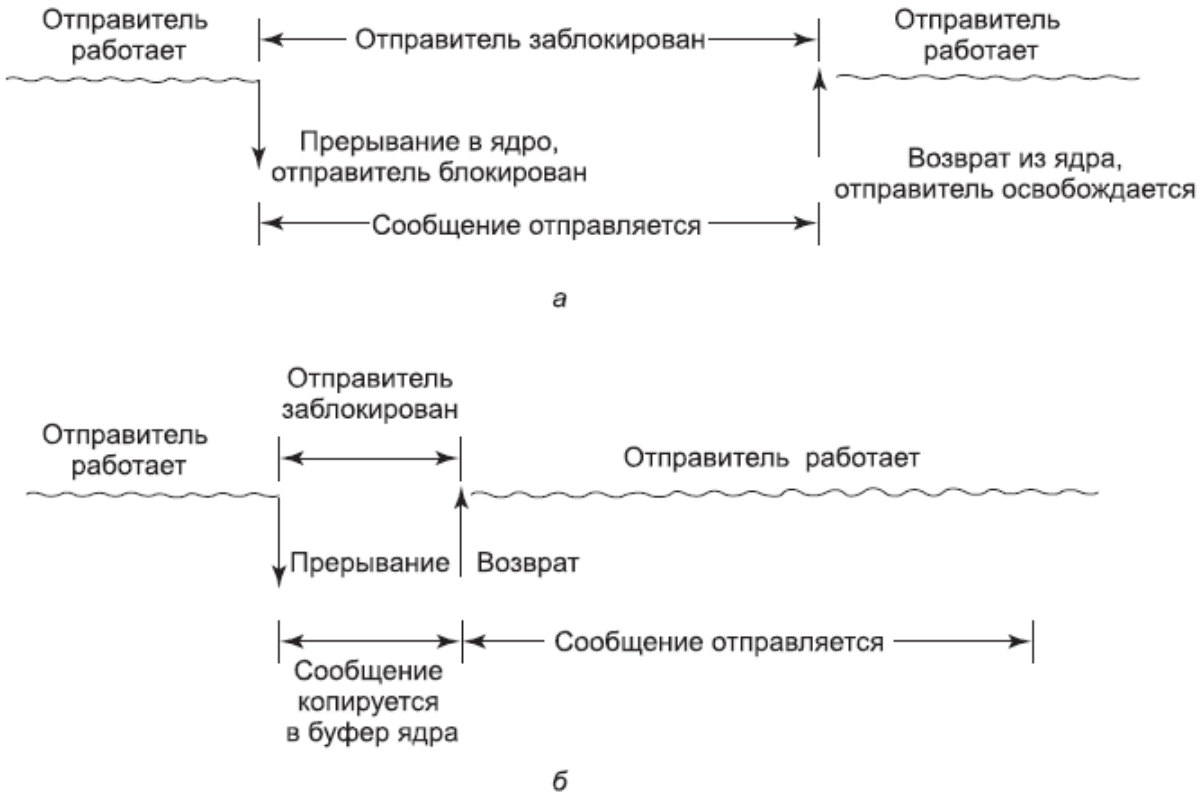
Две интерфейсных платы: одна, отображенная на пространство пользователя, – для обмена данными между приложениями и ещё одна, отображенная на пространство ядра, – для использования ОС.

Сетевые карты с несколькими очередями (т. е. несколько буферов).

Remote Direct Memory Access (RDMA) – удалённый непосредственный доступ к памяти. Пример: InfiniBand.

Коммуникационные программы уровня пользователя

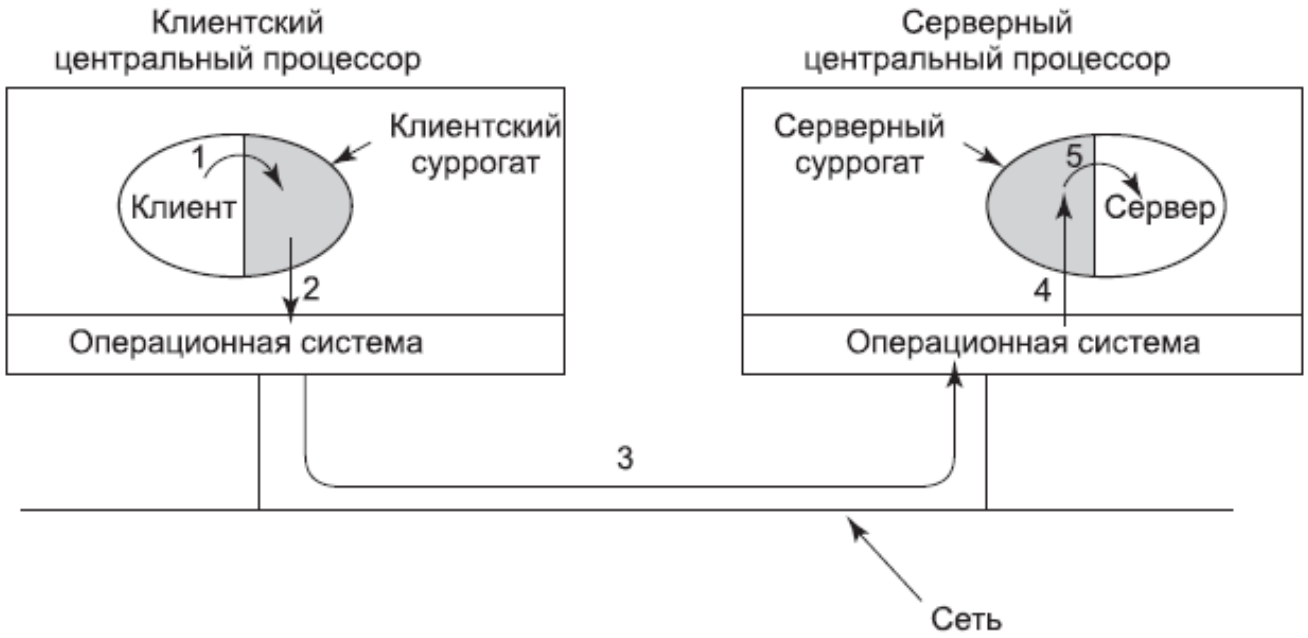
Основаны на вызовах send()/receive().
Сравнение блокирующих и неблокирующих вызовов:



Всплывающие потоки. Поступающее сообщение самопроизвольно порождает новый поток в адресном пространстве процесса-получателя сообщения.

Вызов удалённой процедуры

Remote Procedure Call (RPC)



Упаковка параметров – *маршализация* (marshaling) (шаг 2).

Как передавать указатели и сложные структуры переменного размера?
Использование глобальных переменных?

Распределённая совместно используемая память (1)

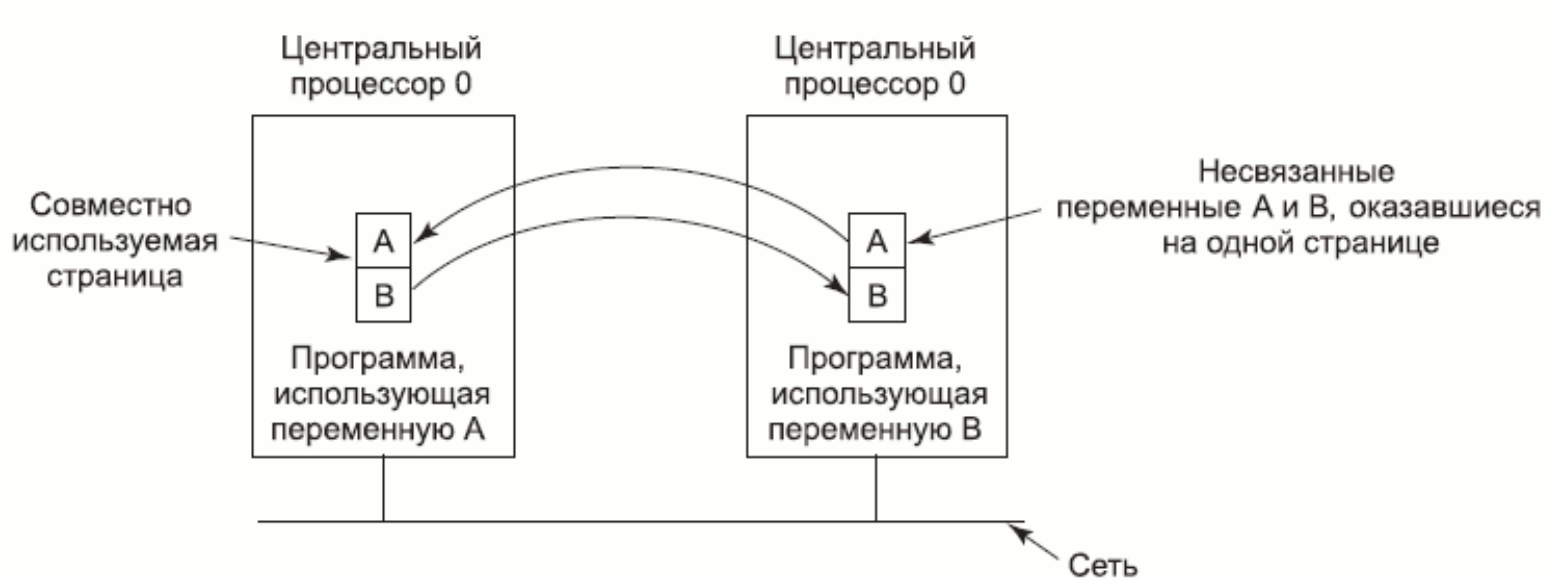
Distributed Shared Memory (DSM):

В системе с DSM адресное пространство поделено на страницы, которые распространены по всем узлам системы. Когда ЦП обращается к нелокальному адресу, происходит перехват управления и программное обеспечение DSM извлекает страницу, содержащую этот адрес, и перезапускает команду, на которой произошла ошибка отсутствия страницы.

Репликация: не удалять read-only страницы (хранить реплики).

Распределённая совместно используемая память (2)

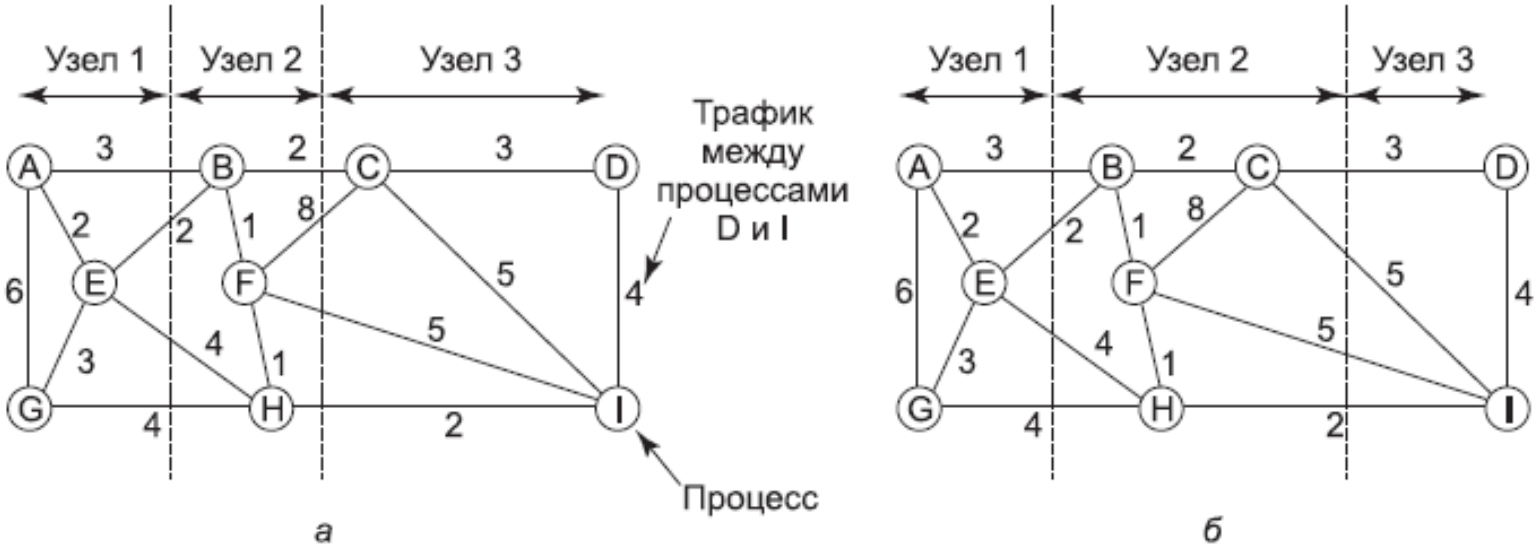
Слишком большой действительный размер страницы создает проблему – *неправильное совместное использование* (false sharing):



Решать на уровне компилятора?

Балансировка нагрузки (1)

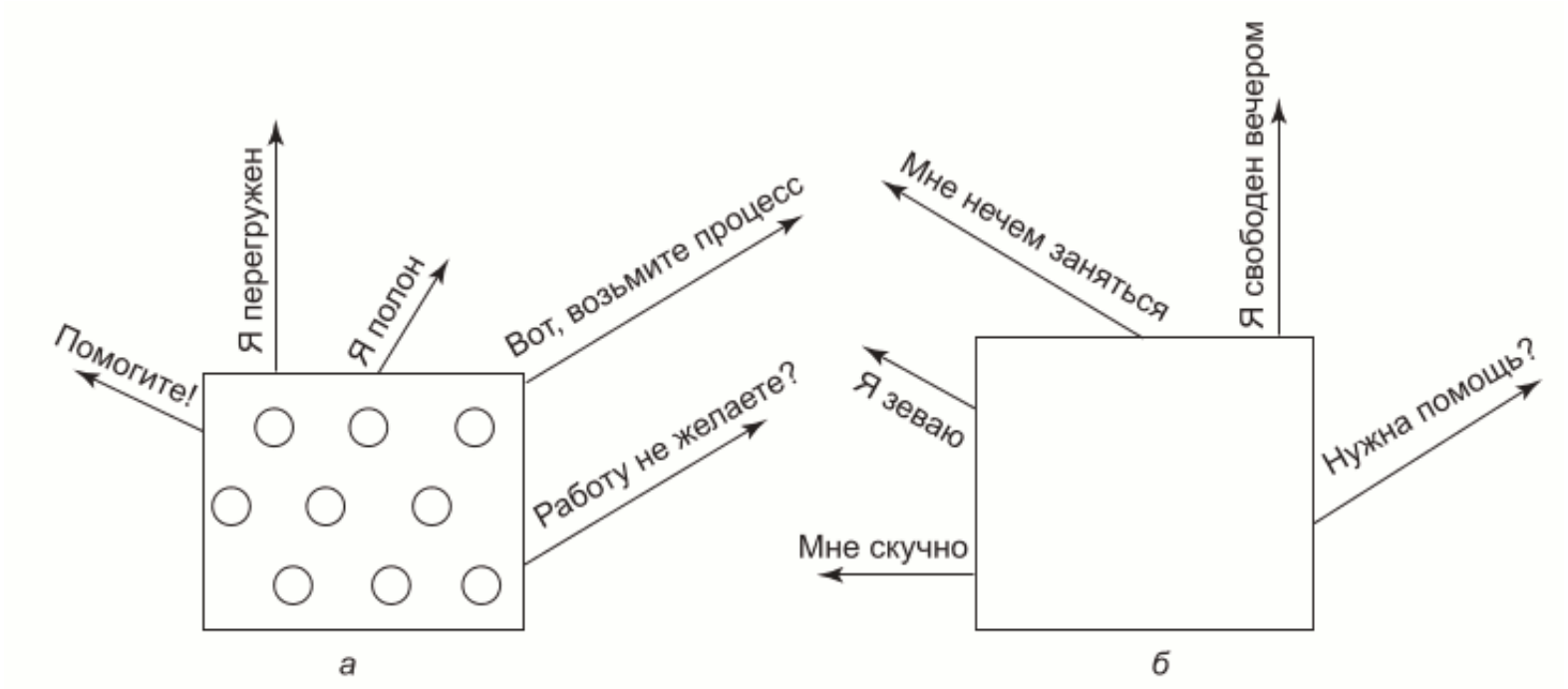
Для систем, состоящих из процессов с хорошо известными требованиями к ЦП и памяти и известной матрицей, дающей средний объём обмена данными между каждой парой процессов.
Если количество процессов превышает количество ЦП, то каждому ЦП должно быть назначено несколько процессов так, чтобы назначения сводили к минимуму сетевой трафик.
Детерминированный графовый алгоритм:



Два способа распределения девяти процессов по трём узлам

Балансировка нагрузки (2)

Распределённый эвристический алгоритм, инициируемый отправителем (а)
Распределённый эвристический алгоритм, инициируемый получателем (б)



а) В условиях большой загруженности все машины будут постоянно отправлять тестовые сообщения другим машинам, тщетно пытаясь найти машину, желающую получить дополнительную работу.
б) Не создает дополнительной нагрузки на систему в критические периоды.

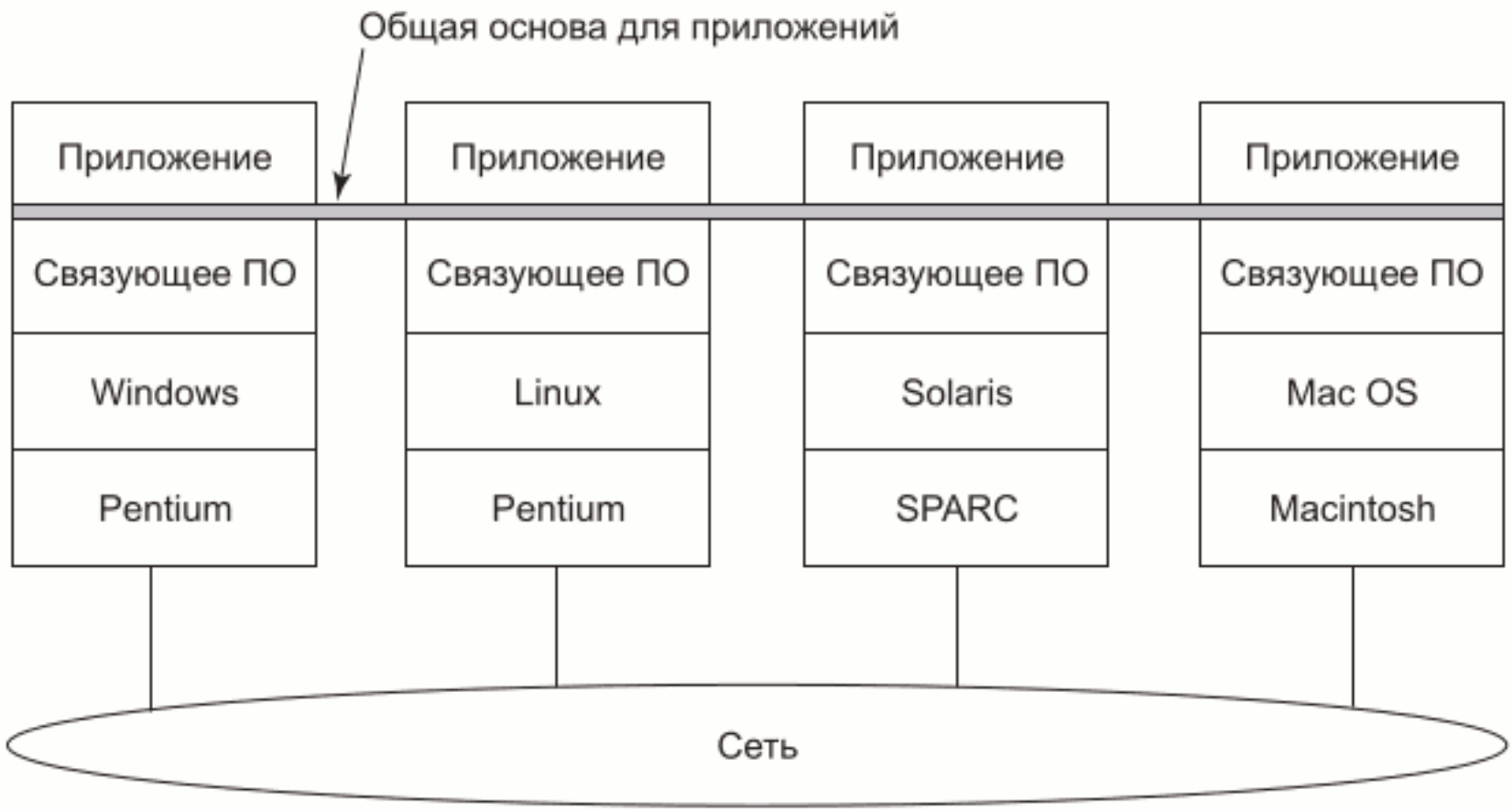
Распределённые системы

Сравнение типов многопроцессорных систем

Сравниваемые показатели	Мультипроцессор	Мультикомпьютер	Распределенная система
Конфигурация узла	Центральный процессор	Центральный процессор, оперативная память, сетевой интерфейс	Полноценный компьютер
Периферийные устройства узла	Все устройства общие	Общие, кроме, может быть, диска	Полный набор устройств для каждого узла
Расположение	В одном блоке	В одном помещении	Возможно, по всему миру
Связь между узлами	Общая память	Специальные линии	Традиционная сеть
Операционные системы	Одна, общая	Несколько, одинаковые	Все могут быть разными
Файловые системы	Одна, общая	Одна, общая	У каждого узла своя
Администрирование	Одна организация	Одна организация	Множество организаций

Связующее ПО

Связующее программное обеспечение (middleware):



Например, для распределённых вычислений – Message Passing Interface (MPI).