

Операционные системы

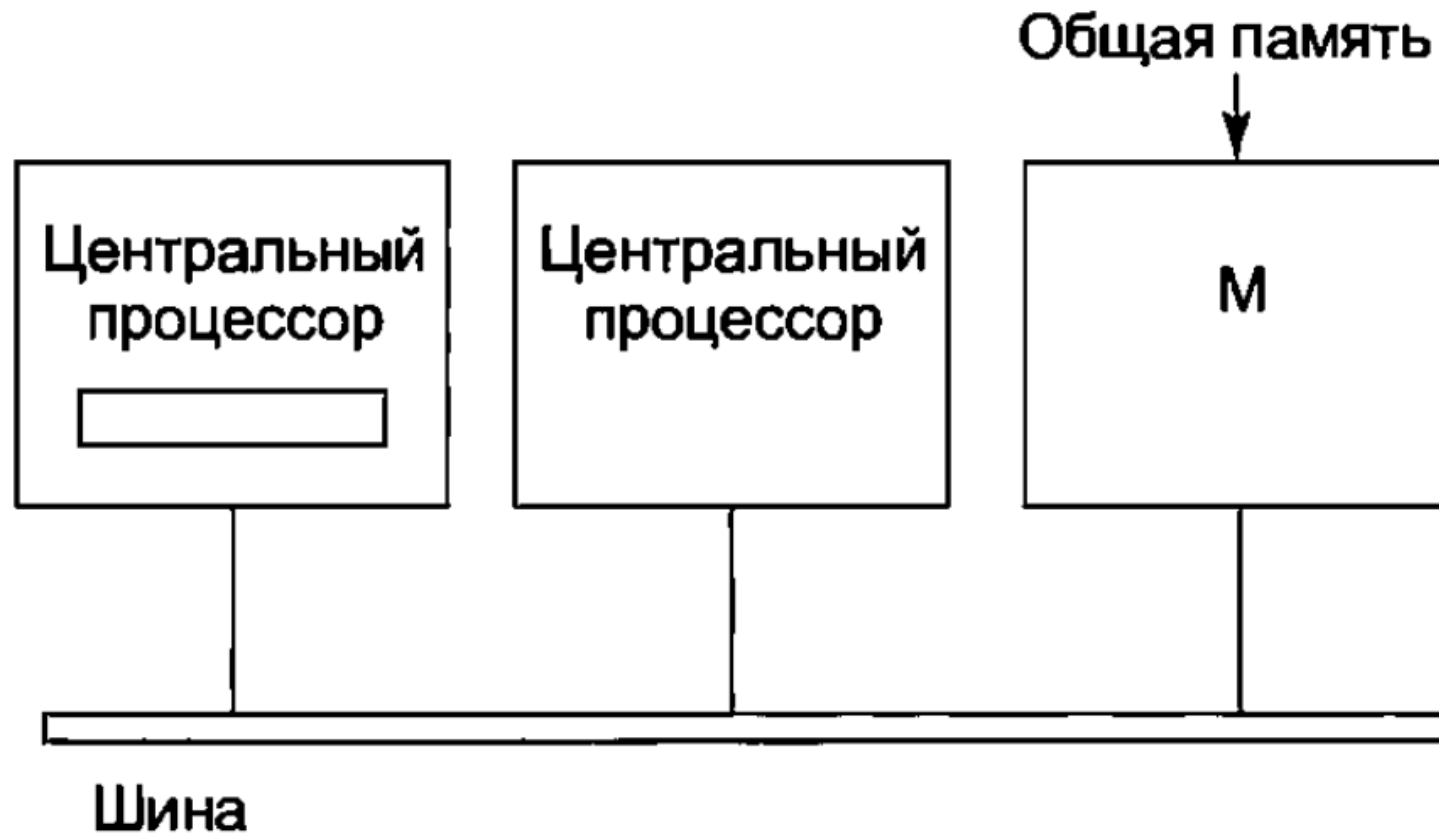


Процессы и потоки.
Взаимодействие между
потоками

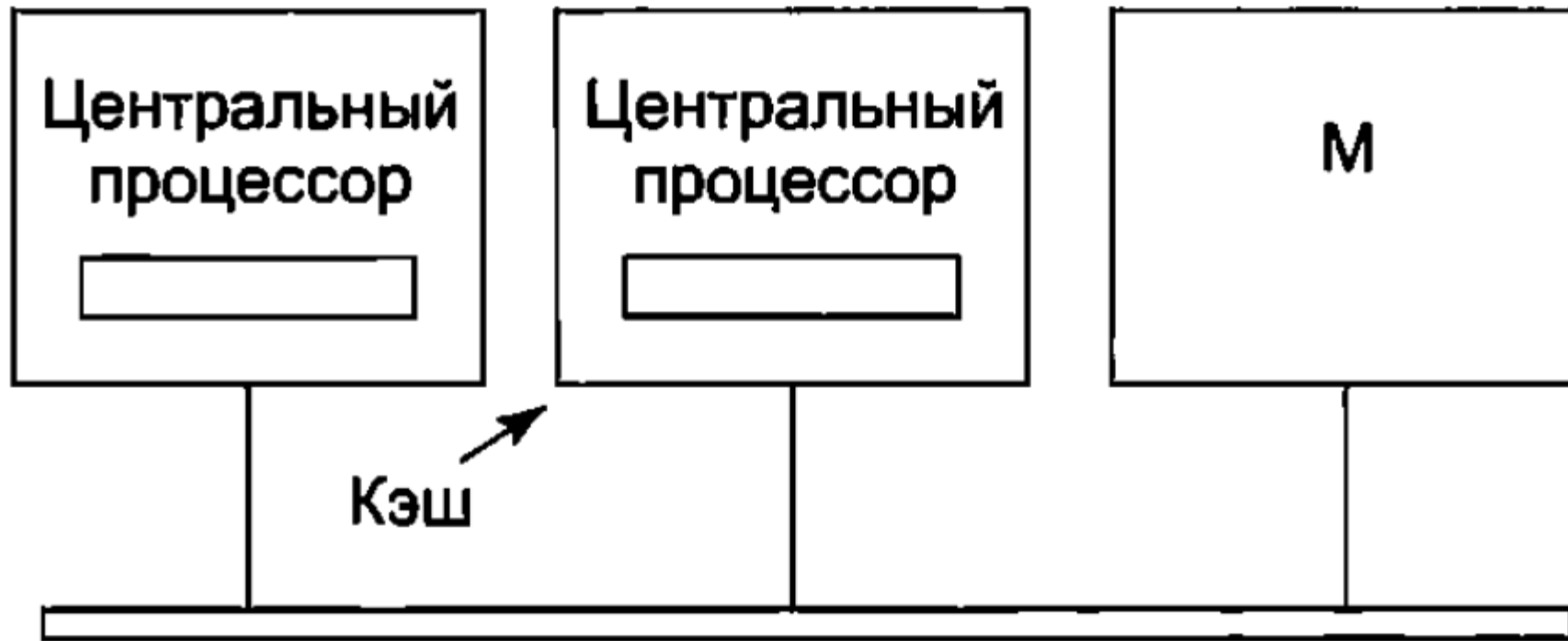
Аппаратные средства для распараллеливания задач

1. Мультипроцессоры
UMA, NUMA
2. Многоядерные системы
3. Вычислительный кластер
Grid (Open MPI), Load balancing

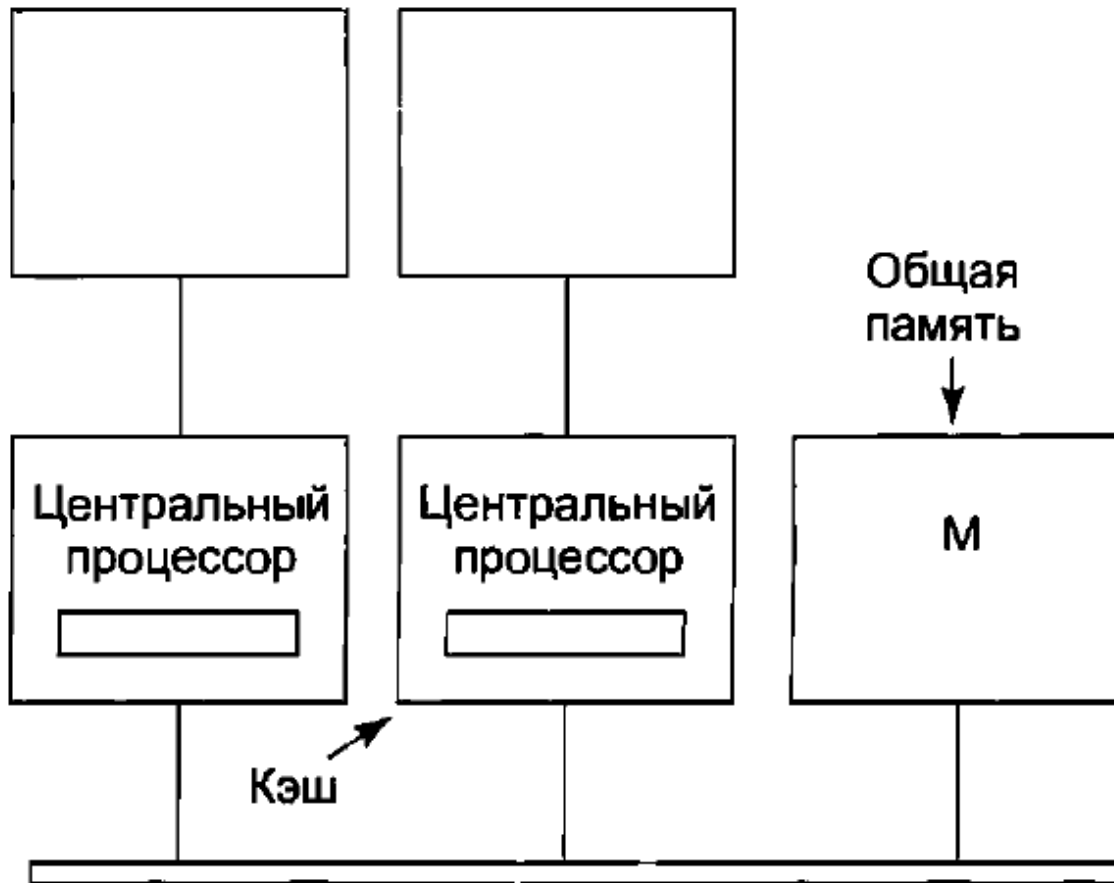
UMA-мультипроцессоры



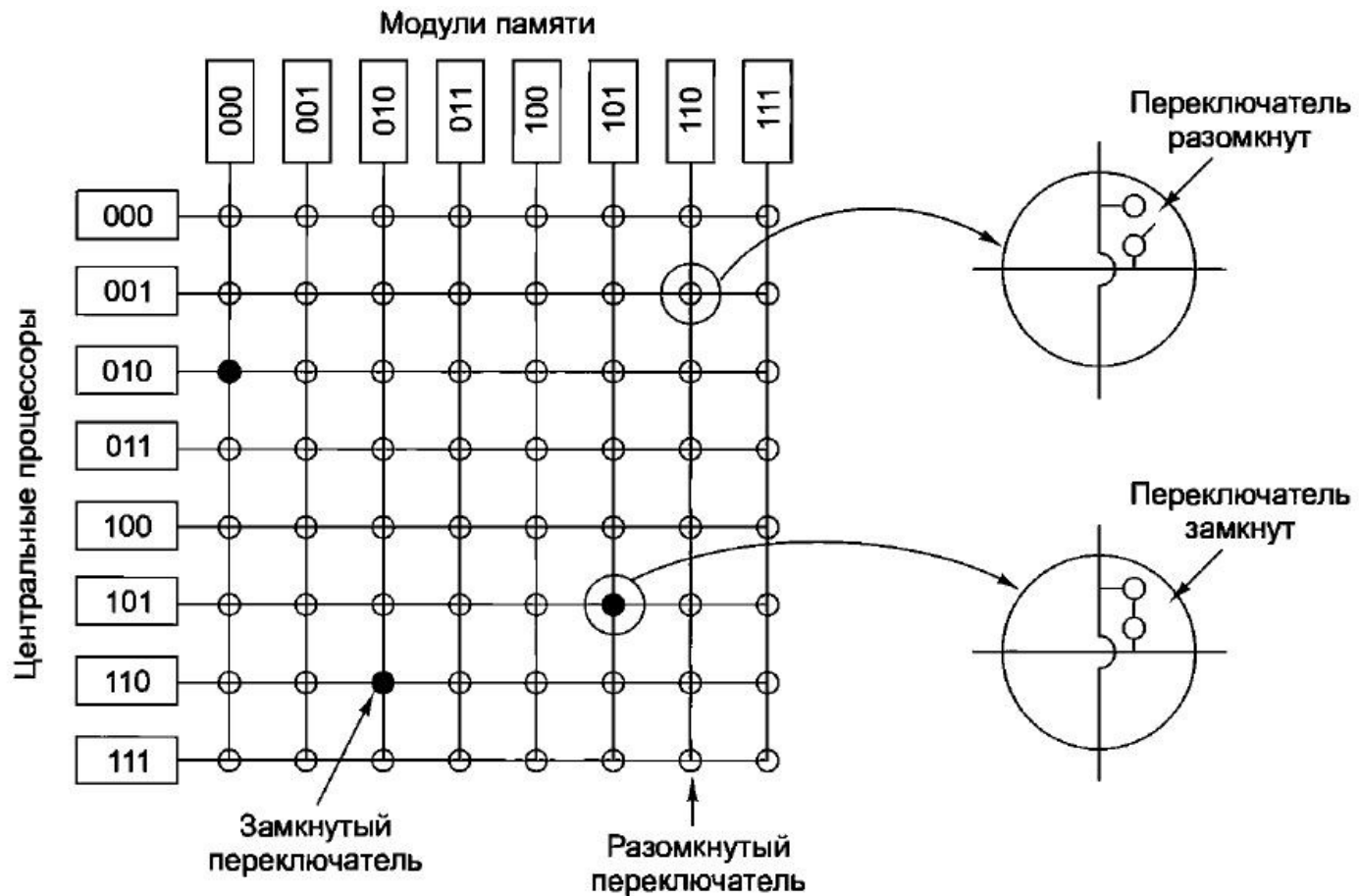
UMA-мультипроцессоры



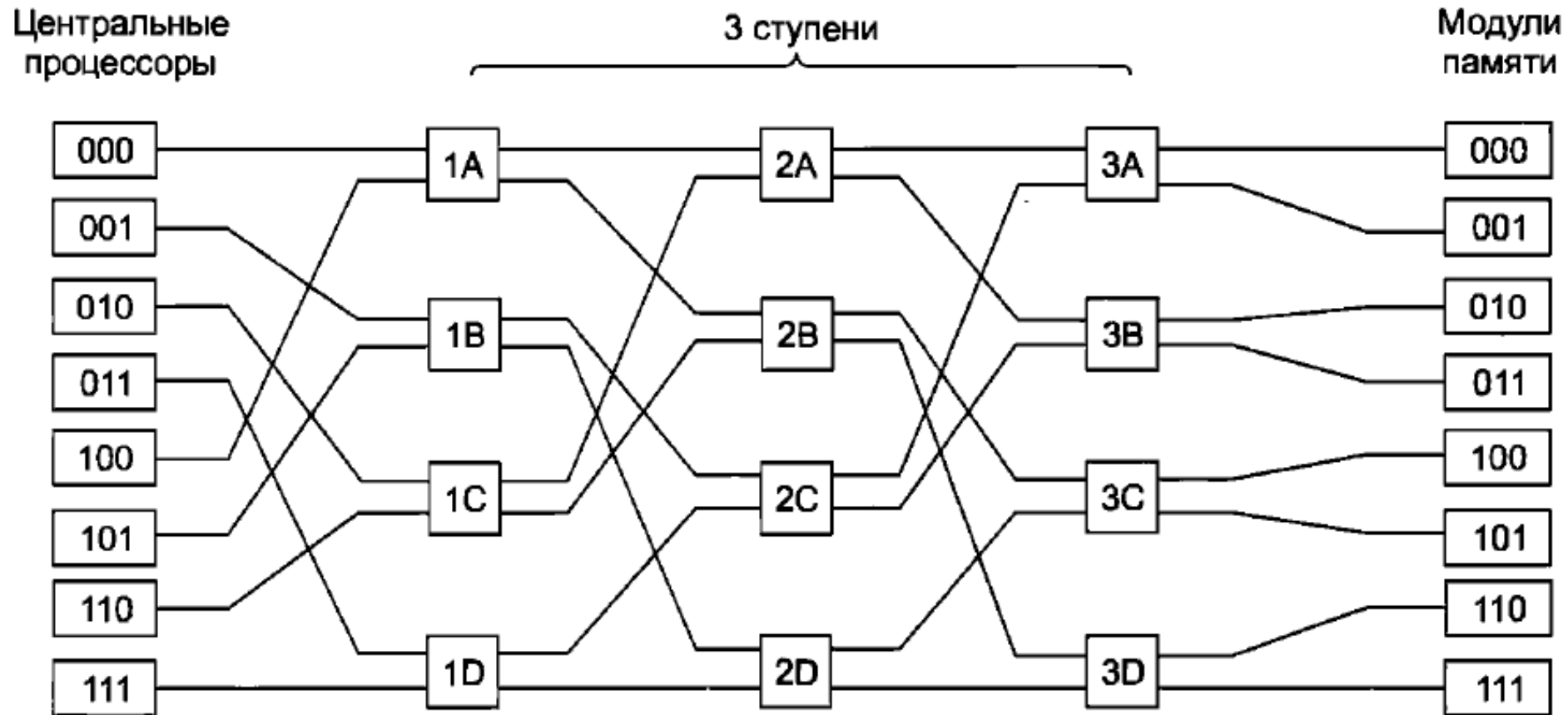
UMA-мультипроцессоры



Координатный коммутатор



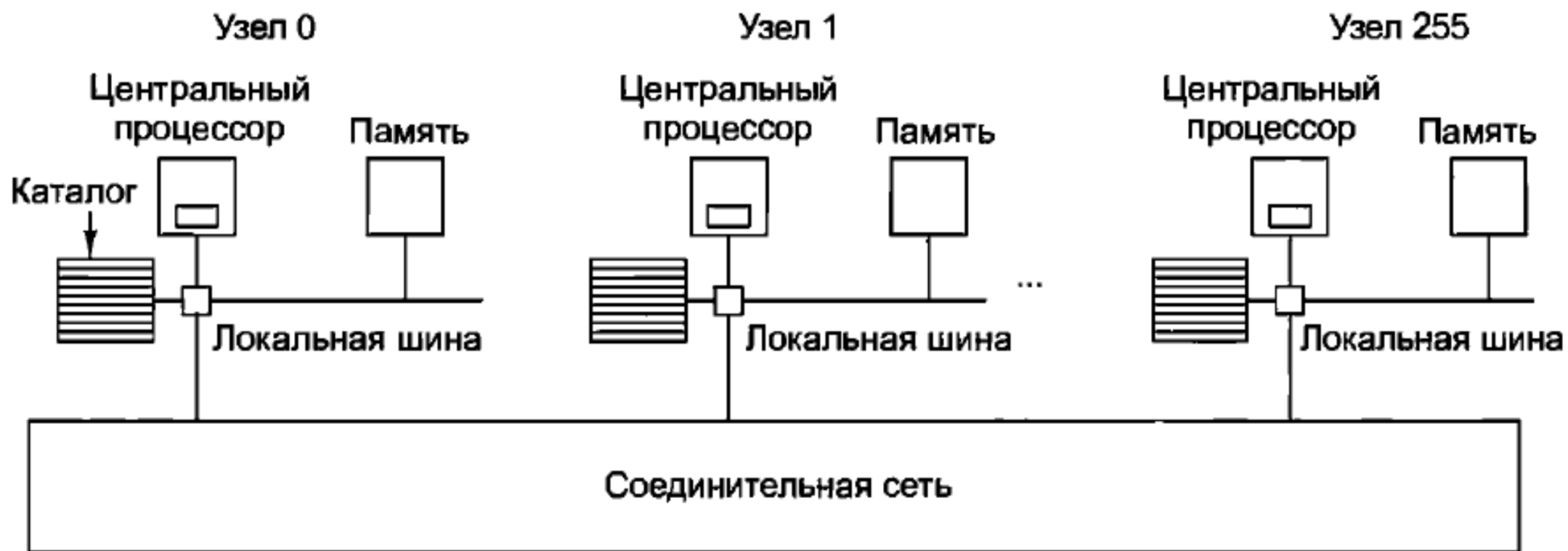
Многоступенчатые коммутаторные сети



NUMA

1. Единое адресное пространство
2. Доступ осуществляется LOAD|STORE
3. Доступ к удаленной памяти медленнее, чем к локальной

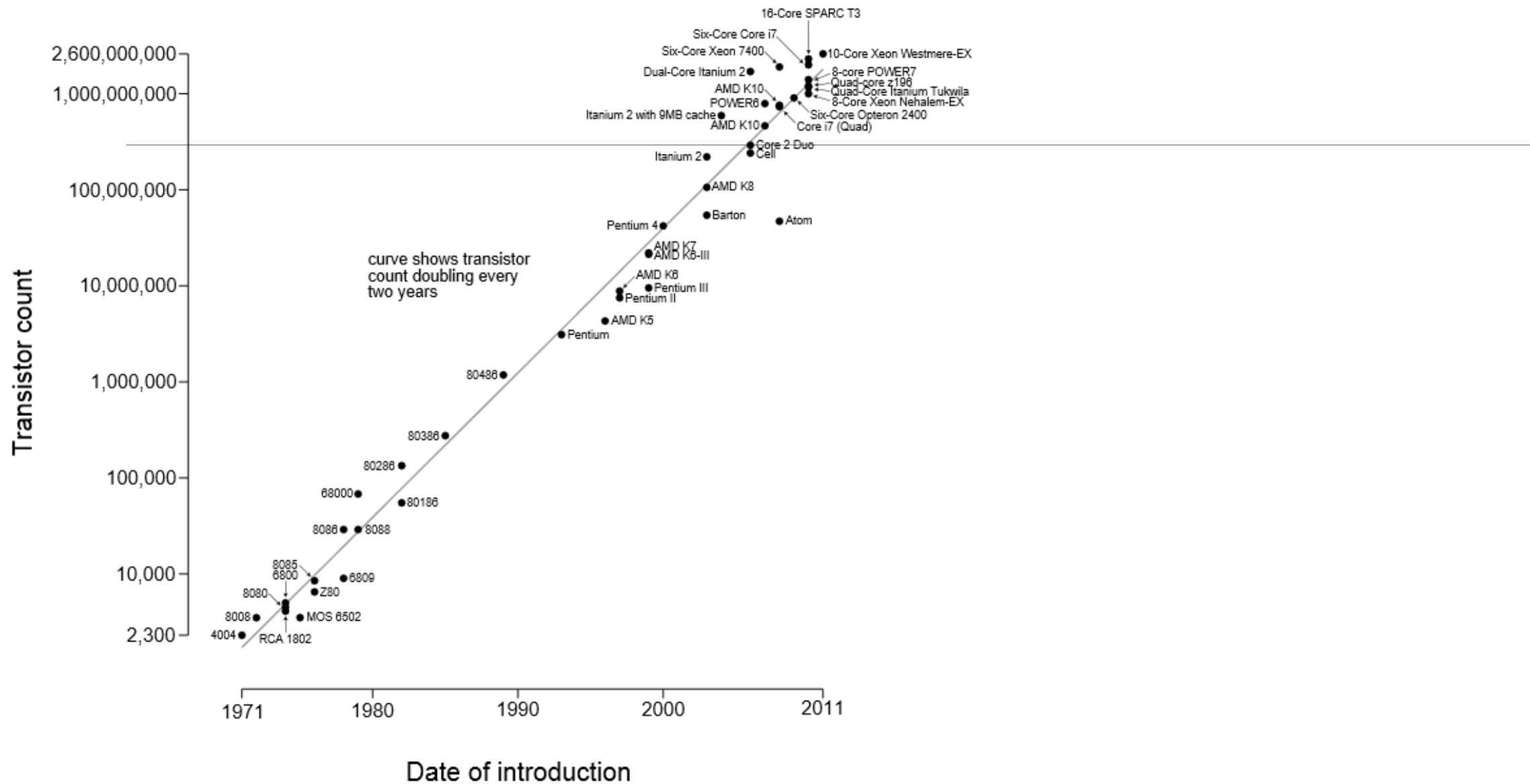
NUMA



Многоядерные системы

1. Различные вариации кэш-памяти. Вопросы согласованности кэшей
2. Низкая отказоустойчивость
3. Tesla K40 – 2880 ядер

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Кластер

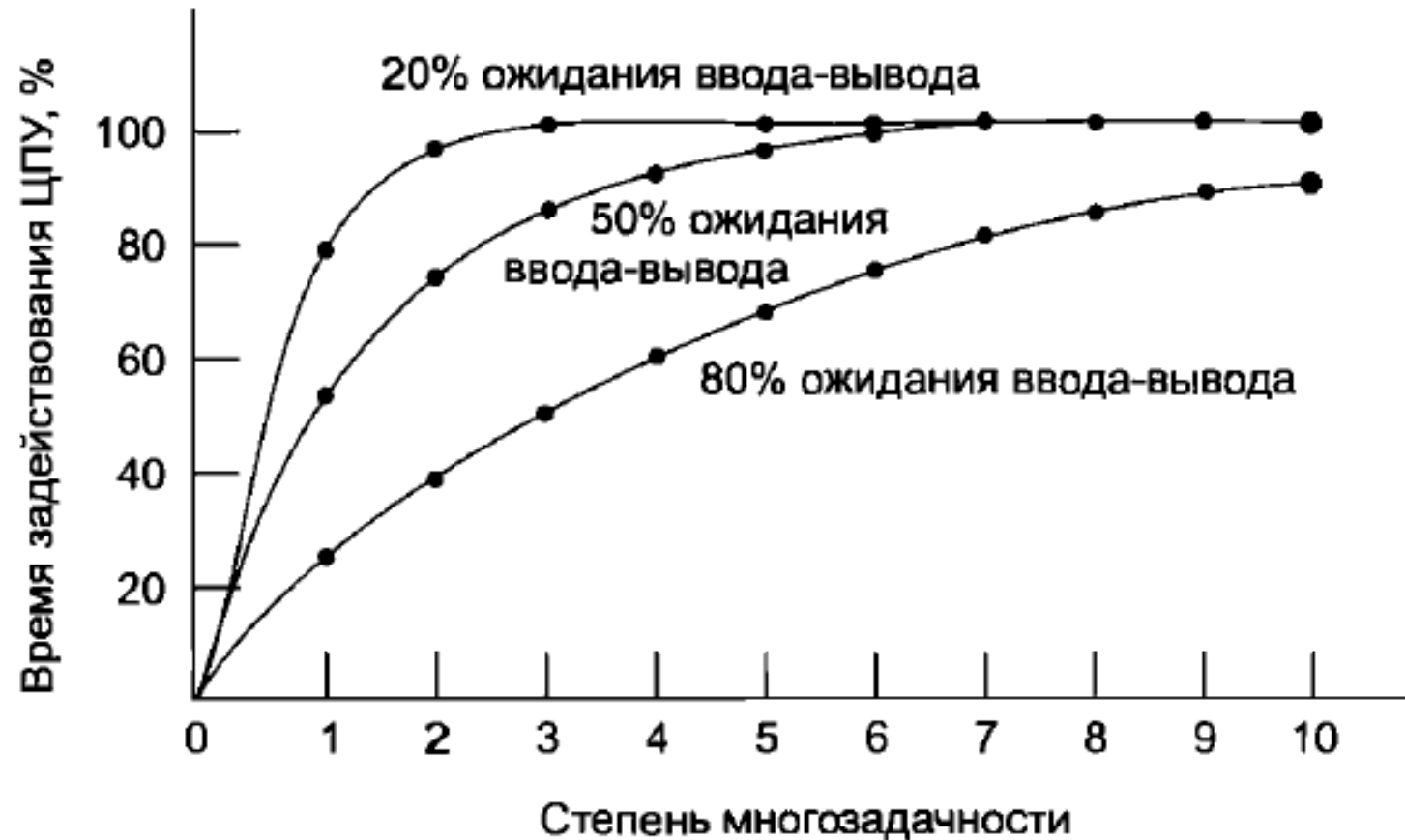
1. Высокая доступность
2. Load balancing
3. Вычислительный кластер
4. GRID

Моделирование режима многозадачности

$$\underline{ЦП = 1 - pn}$$

p – ожидание ввода-вывода

n – количество процессов



Метрики параллельных алгоритмов

1. T_p – время выполнения на p различных вычислительных ядрах
2. Ускорение
 $S_p = T_1/T_p$
 $S_p < p$
3. Эффективность/загруженность
 $X_p = S_p/p$
 $X_p < 1$

Верхние оценки ускорения

1. Закон Амдала

$$S_p = \frac{1}{\alpha + \frac{1 - \alpha}{p}}$$

2. Закон Густавсона — Барсиса

$$S_p = g + (1 - g)p = p + (1 - p)g$$

$$g = \frac{\tau(n)}{\tau(n) + \pi(n)/p}$$

Создание потока (Windows)

```
CreateThread(  
    NULL,    // default security attributes  
    0,       // default stack size  
    (LPTHREAD_START_ROUTINE) ActionTest, // routine  
    (LPVOID)arg,    // no thread function arguments  
    0,           // default creation flags  
    &ThreadId); // receive thread identifier
```


Примитивы синхронизации

1. Mutex
2. Semaphore
3. Barrier
4. Join thread
5. Conditional variable
6. ...

Conditional variable

1. При вхождении на вход мьютекс (другой примитив синхронизации)
2. При блокировке освобождает мьютекс
3. При приеме сигнала на разблокировку
 - a) Блокирует мьютекс
 - b) Продолжает исполнение с момента блокирования

CondVar Example

```
Get()
```

```
{
```

```
    lock(mutex)
```

```
    if(i == 0)
```

```
        LockCondVar(cond_var, mutex);
```

```
    i--;
```

```
    DoWorkGet(i);
```

```
    unlock(mutex)
```

```
}
```

```
Put()
```

```
{
```

```
    lock(mutex)
```

```
    i++;
```

```
    if(i > 0)
```

```
        UnlockCondVar(cond_var);
```

```
    DoWorkGet(i);
```

```
    unlock(mutex)
```

```
}
```

Wait functions (Windows)

1. WaitForSingleObject
2. WaitForMultipleObjects
MAXIMUM_WAIT_OBJECTS

Events (Windows)

1. CreateEvent
2. SetEvent
3. ResetEvent (ManualReset)
4. Global/Local