Високопроизводителни системи – от концепция до реализация

Д-р Христо Илиев

05.09.2018 г.

За мен

- Бакалавър по Физика, СУ, 2003 г.
- Доктор по Физика на атомите и молекулите, 2009 г.
- Научен сътрудник, RWTH Aachen, 02.2012 - 01.2018 г.
  HPC Team @ IT Center der RWTH Aachen University
- Chief data scientist, Consent IO BV, от 02.2018 г.

За мен

- Заедно с гл. ас. д-р Стоян Писов …
- … изграждане и поддръжка на клъстера Physon (ФзФ, СУ)
- … изграждане на клъстера Nestum (ЛВПИ, София Тех Парк)

План

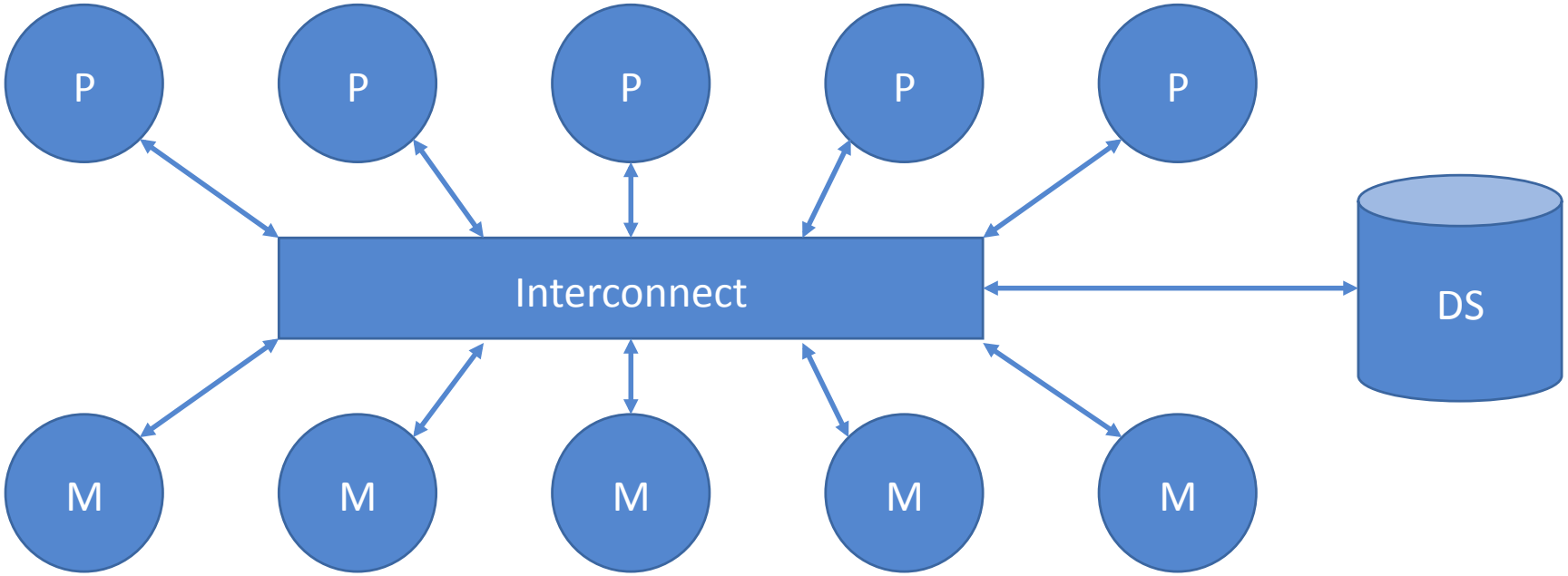
- Архитектура на типична система за ВПИ
- Клъстери от широкодостъпни части
- Управление на ресурси
- Особености при изпълнение на някои приложения

Високопроизводителни изчисления (HPC)

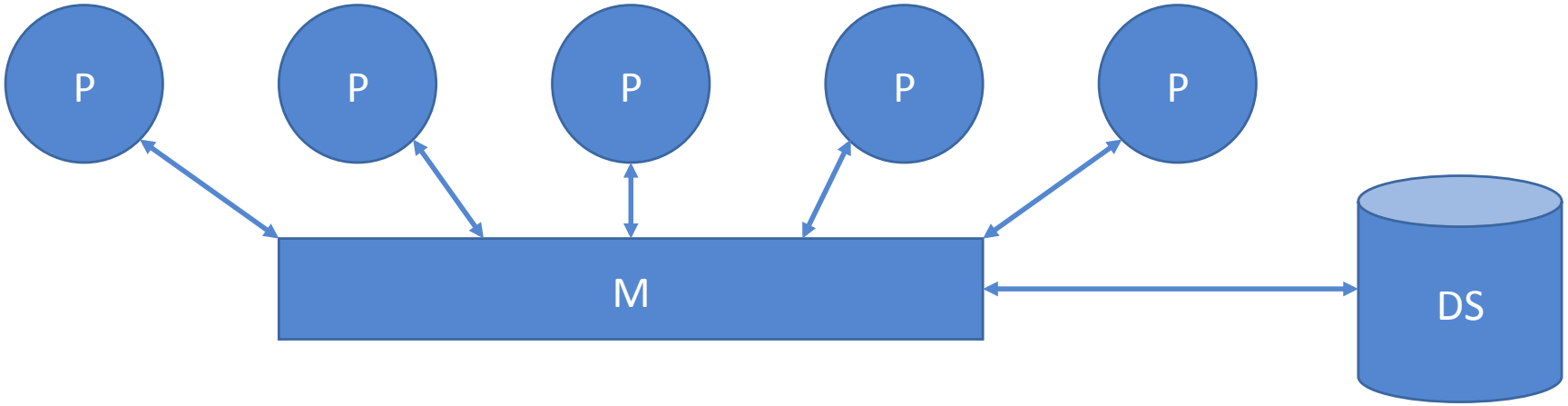
- Значителна изчислителна мощност
- Значителна по обем оперативна памет
- Значително по обем дисково пространство
- Значителна площ
- Значителна консумация на ел. мощност
- Значителна отделяна топлина
- Висока цена
- Следствие => Споделен достъп

Архитектура на система за ВПИ

- Силно интегрирани разпределени ресурси

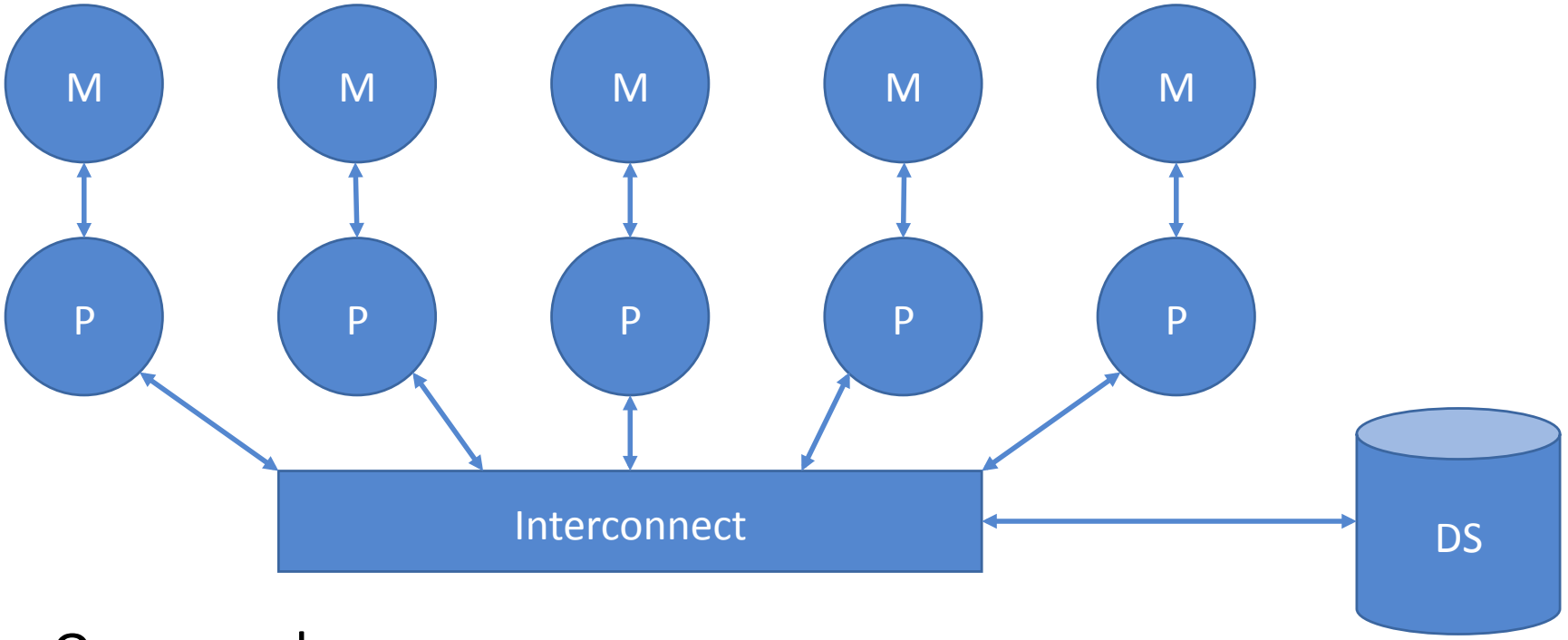


Споделена памет



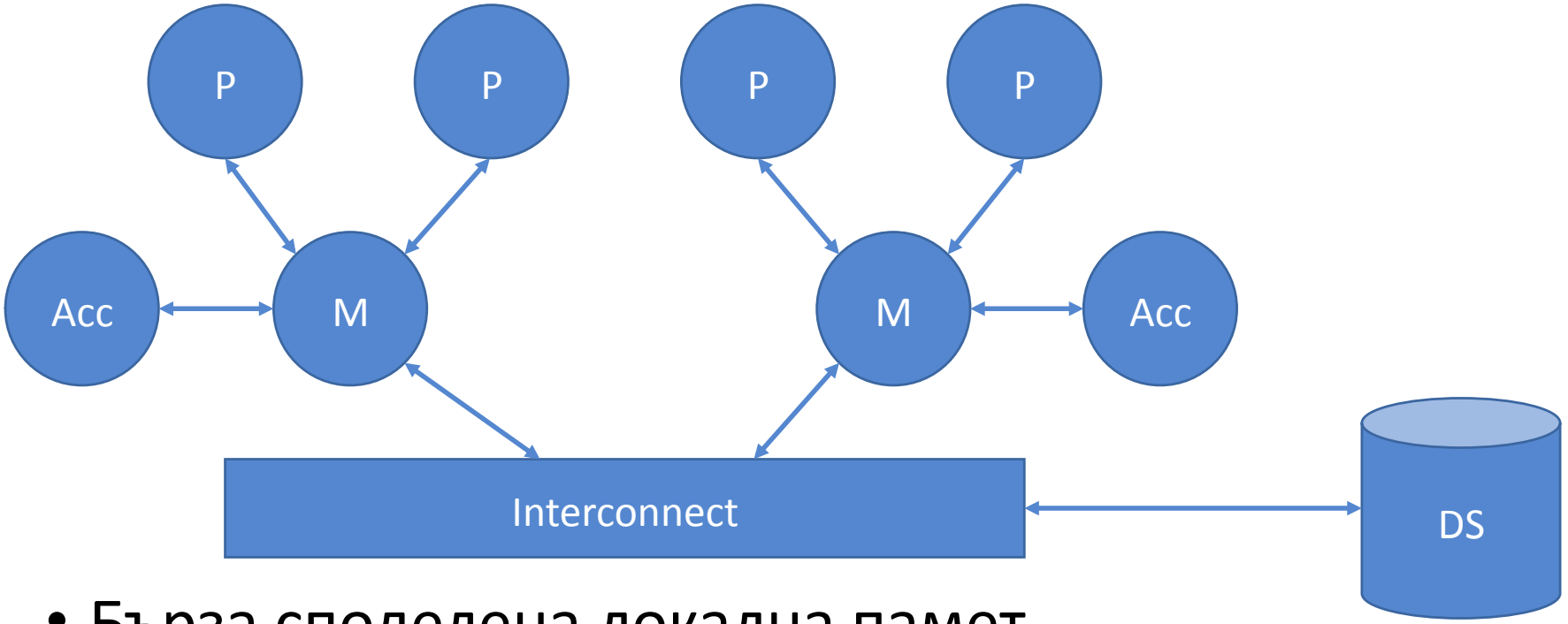
- Общо физическо адресно пространство
  - Една ОС
  - Многонишкови програми (OpenMP, CUDA)
- Ограничена мащабируемост
  - Кохерентност на кешовете
  - Сложността и цената растат експоненциално

Разпределена памет



- Отделни физически адресни пространства
  - Множество копия на ОС
  - Явна (MPI) или неявна (PGAS) обмяна на съобщения
- Теоретично неограничена мащабируемост

Хибриден модел

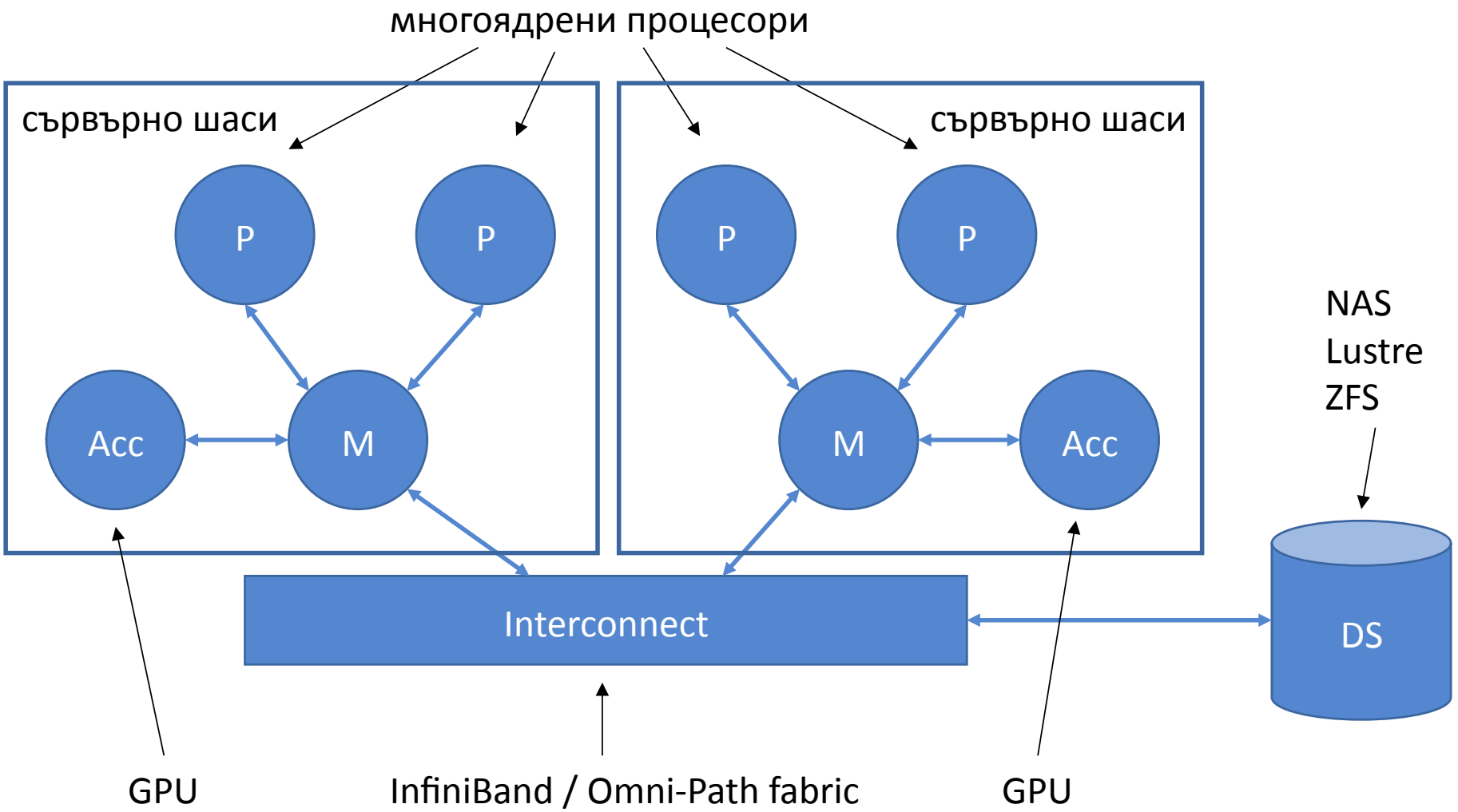


- Бърза споделена локална памет
- Обмяна на съобщения за отдалечен достъп
- Ускорители (GPGPU, many-core процесори)

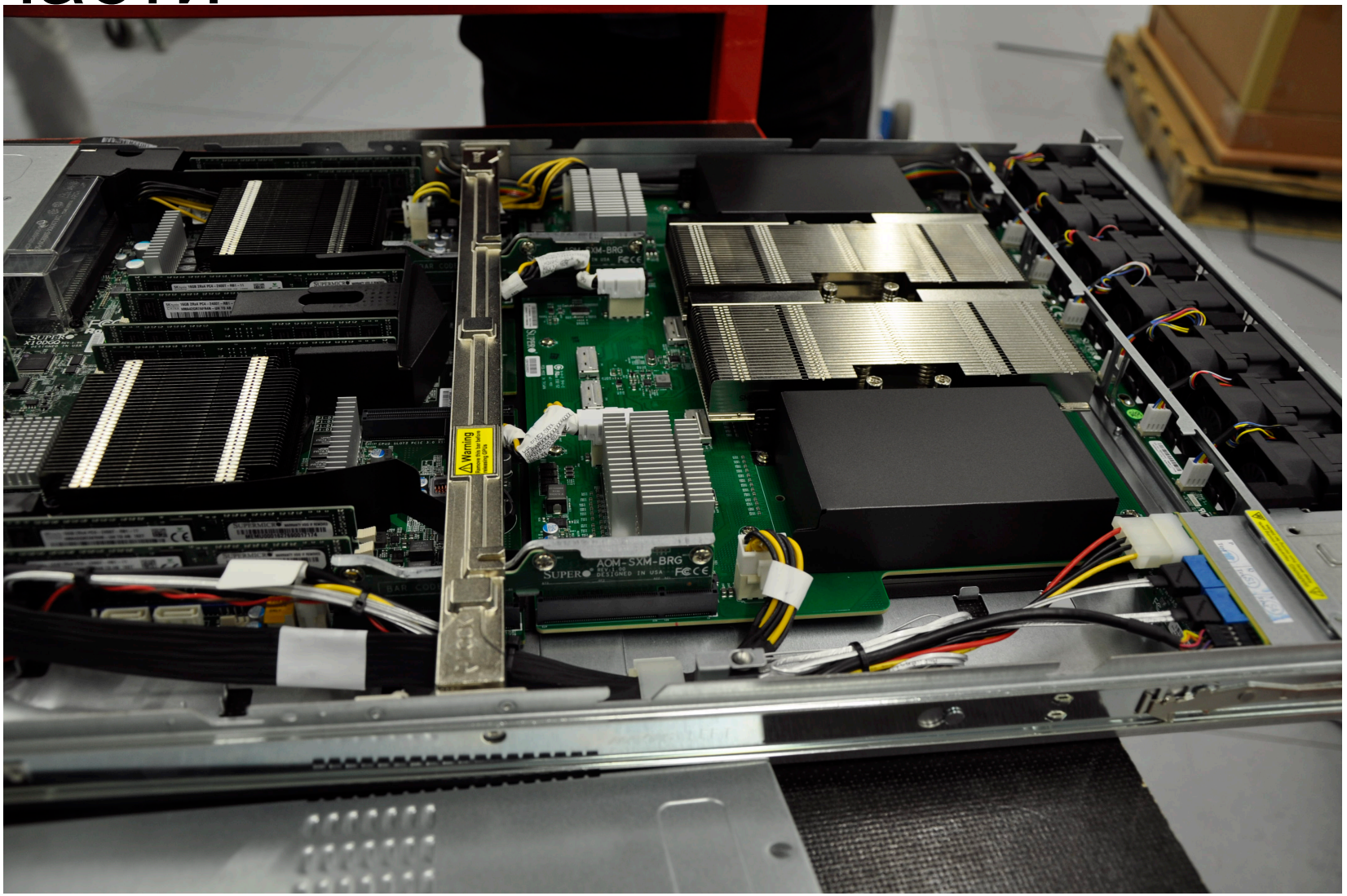
Допълнителни особености

- Многопотребителска ОС
- Паралелна файлова система
- Изпълнение на програми в пакетен режим
- Многостепенни хранилища на данни

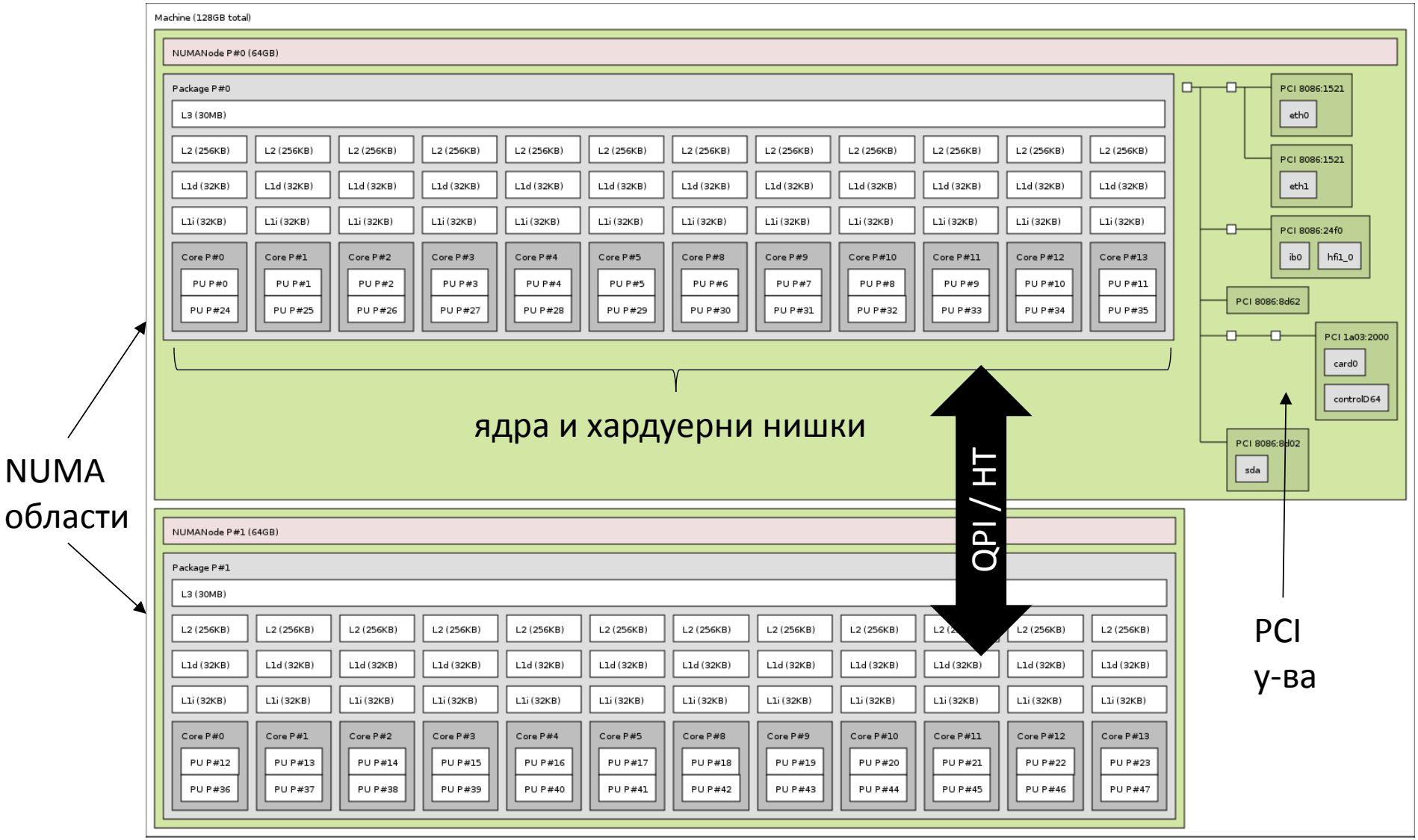
Клъстери от широкодостъпни части



Клъстери от широкодостъпни части



Типичен възел



NUMA области

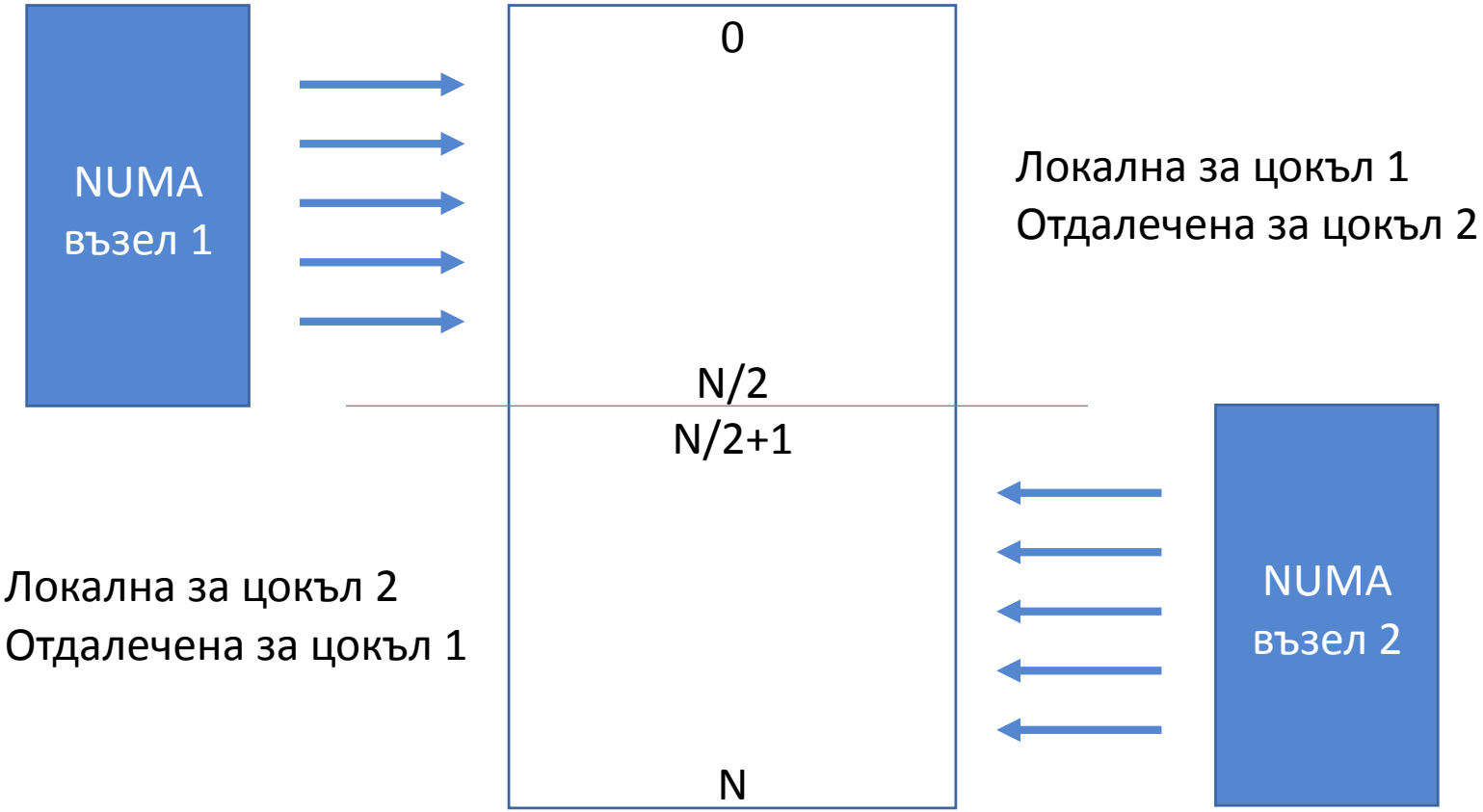
PCI у-ва

NUMA

- Йерархична структура на паметта
- Локална памет
  - Всеки процесор има собствен RAM контролер
  - Ниска латентност, висока пропускателна способност
- Отдалечена памет
  - Памет на чужд (далечен) контролер
  - Висока латентност, ниска пропускателна способност
- Кеш-кохерентна връзка между процесорите
  - QPI (Intel) / HT (AMD)

NUMA

- Физическо адресно пространство



Мрежова свързаност

- Ниска латентност
- Висока пропускателна способност

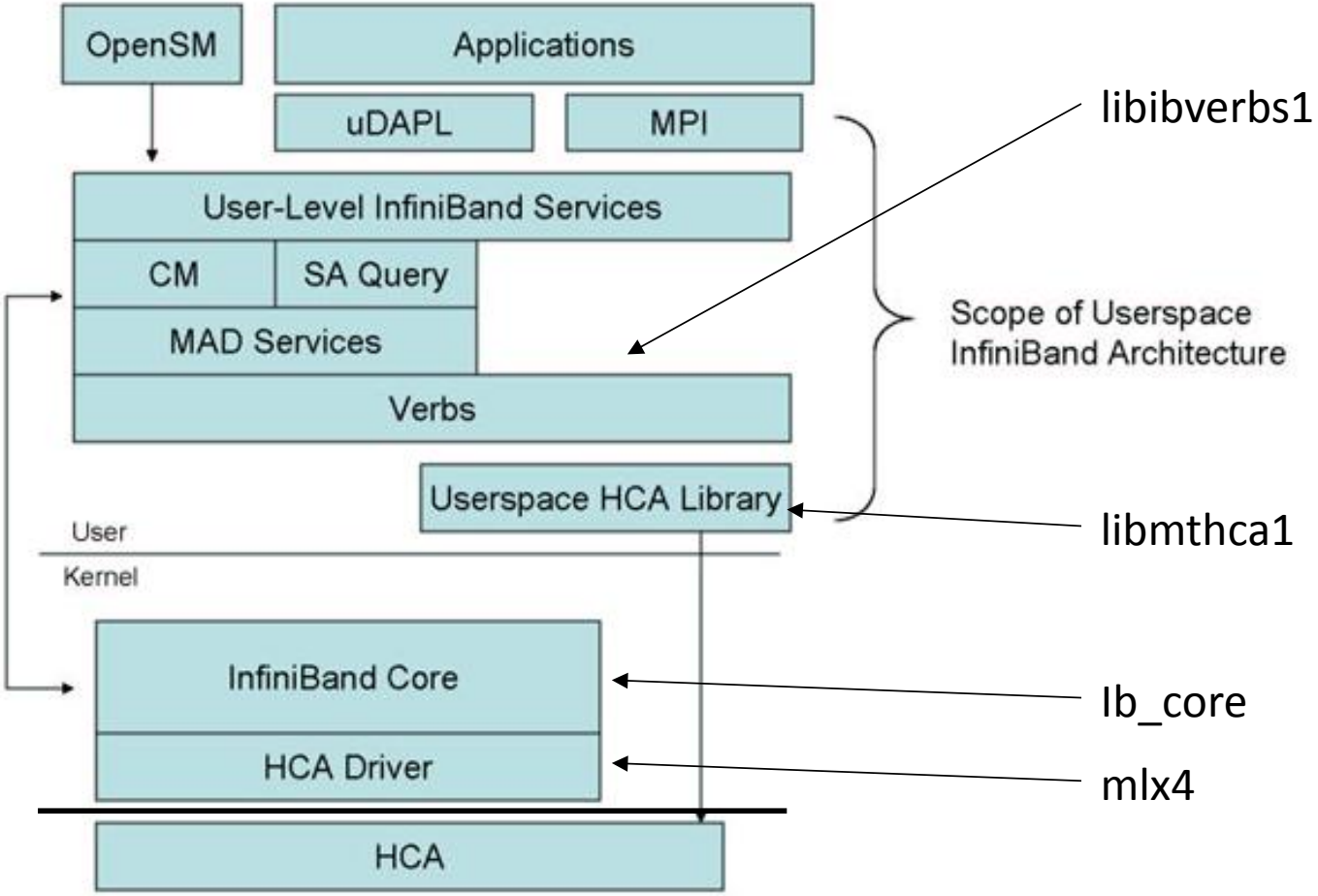
Ethernet	GigE	1 Gbps	29 us
InfiniBand (4x)	SDR	8 Gbps	5 us
	DDR	16 Gbps	2,5 us
	QDR	32 Gbps	1,3 us
	FDR	55 Gbps	0,7 us
	EDR	100 Gbps	0,5 us
Omni-Path	-	100 Gbps	0,5 us

- Типичен избор – InfiniBand или Omni-Path

InfiniBand / Omni-Path

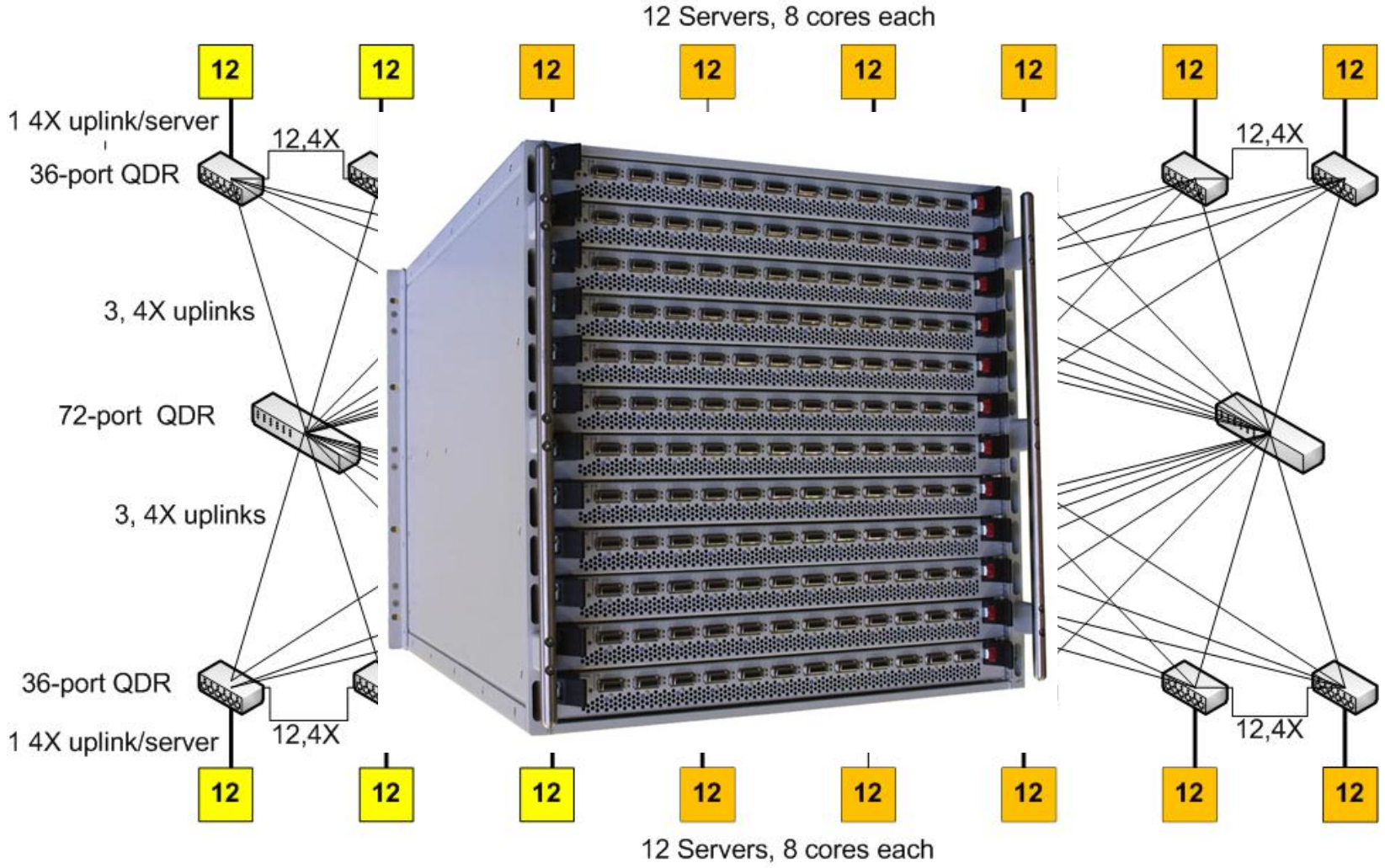
- Драйвери на две нива
  - Обща част в ядрото
  - Специфични драйвери в потребителски режим
- Директен достъп до хардуера от потребителски режим
  - Ниска латентност на операциите
  - Опростено програмиране
- OFED verbs API (InfiniBand, Omni-Path)
- PSM (Omni-Path)

InfiniBand verbs



Источник: Intel Corp

InfiniBand топология



Источник: ARC, University of Leeds, UK

OpenSM

- InfiniBand има нужда от външно управление
  - Subnet Manager (SM)
  - Статична и динамична маршрутизация
  - Отсъства в повечето маршрутизатори
- OpenSM
  - Част от OFED
  - Ubuntu: `opensm`, `libopensm5`
  - Поддръжка на редица топологии
- Едно или две копия на мрежа
  - на различни възли

MPI библиотеки

- Open MPI
  - InfiniBand през libibverbs
  - Omni-Path през libpsm2
  - Ръчна компилация
- Intel MPI
  - InfiniBand през uDAPL
  - Omni-Path през PSM2
  - „Комплектът включва батерии“

Файлова система

- Нужда от споделена мрежова ФС
  - Опростена употреба
  - Повечето софтуер не работи иначе
- Универсални мрежови ФС
  - NFS (Unix-like)
  - CIFS (Windows)
- Паралелни мрежови ФС
  - Lustre
  - BeeGFS
  - GPFS (IBM)

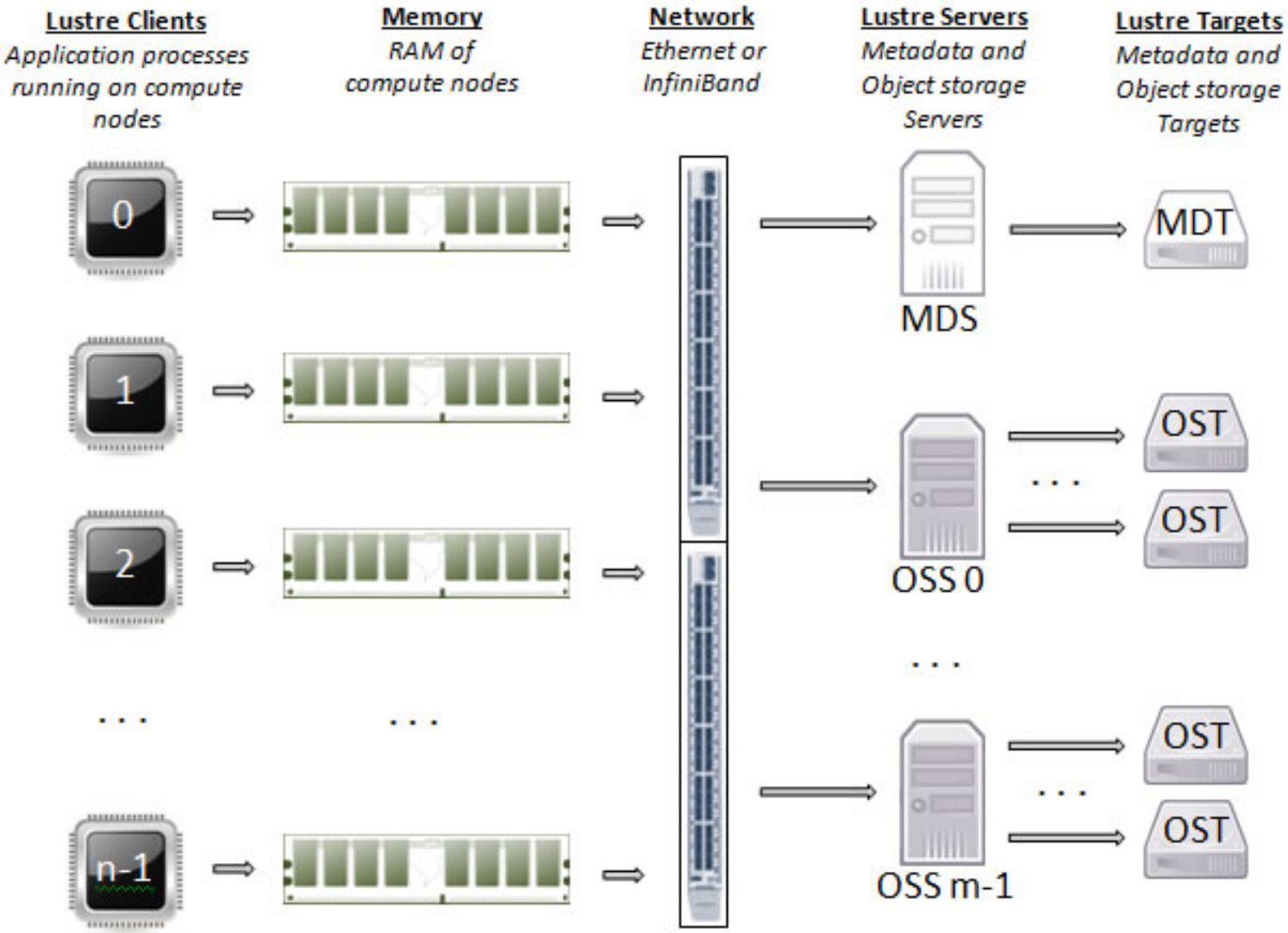
NFS

- Една от най-старите мрежови ФС
- Базирана на Sun RPC върху TCP/IP
  - InfiniBand през IP-o-IB
- Налична в почти всяка ОС
- Лесна за конфигуриране
- Универсална
- Бавна
- Подходяща за $HOME

Lustre

- Мащабируема паралелна ФС
- Отворен код (GPLv2)
- Комерсиална поддръжка
- Разпределена архитектура
  - MDS
  - OSS
- InfiniBand транспорт
- Подходяща за малък брой големи файлове

Lustre


Lustre Clients
Application processes running on compute nodes
Memory
RAM of compute nodes
Network
Ethernet or InfiniBand
Lustre Servers
Metadata and Object storage Servers
Lustre Targets
Metadata and Object storage Targets
0
1
2
n-1
MDS
MDT
OSS 0
OST
OST
OSS m-1
OST
OST

BeeGFS

- Разработка на Fraunhofer ITWM
- Достъпен изходен код (BeeGFS EULA)
- Комерсиална поддръжка от ThinkParQ
- Сходна на Lustre архитектура
- Сходна на Lustre функционалност
- BeeOND
  - Създаване на файлови системи при нужда

Потребителска идентификация

- Хомогенен изглед от страна на потребителя
- Уеднаквени потребителски и групови идентификатори
- Копиране на файлови бази данни с потребители (и евентуално пароли)
- Мрежови идентификационни услуги
  - NIS
  - LDAP
  - ActiveDirectory

LDAP

- Йерархична обектна база данни
- Универсална
  - Основно за директорни услуги
  - Клиенти за повечето ОС
- Интеграция в Unix-подобни ОС
  - модул за NSS, напр. libnss-ldap
    - потребители и групи
    - принадлежност на потребител в група
  - модул за PAM, напр. libpam-ldap
    - автентикация

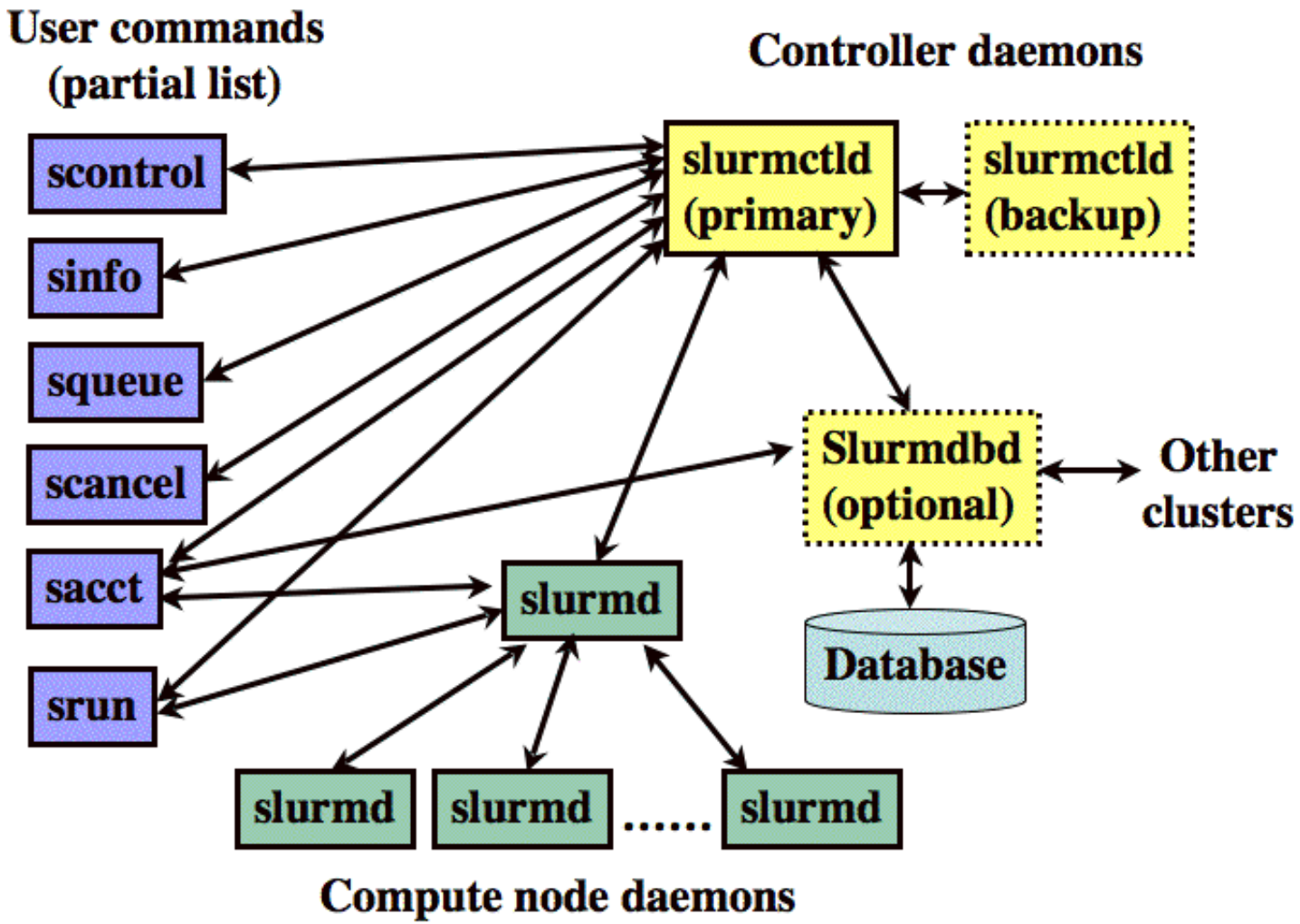
Управление на ресурсите

- Многопотребителски режим
  - Споделено използване без да си пречим
- Пакетно изпълнение (задачи)
- Проектни бюджети
- Уплътняване на използването
- Честно разпределение
- СУРР (DRM)
  - SLURM
  - Torque / OpenPBS
  - (Sun) GridEngine

SLURM

- СУРР на Nestum
- Simple Linux Utility for Resource Management
- Отворен код (GPLv2)
- Изключително модулен
  - От прости до сложни конфигурации
- Високодостъпен режим

SLURM



Източник: SLURM

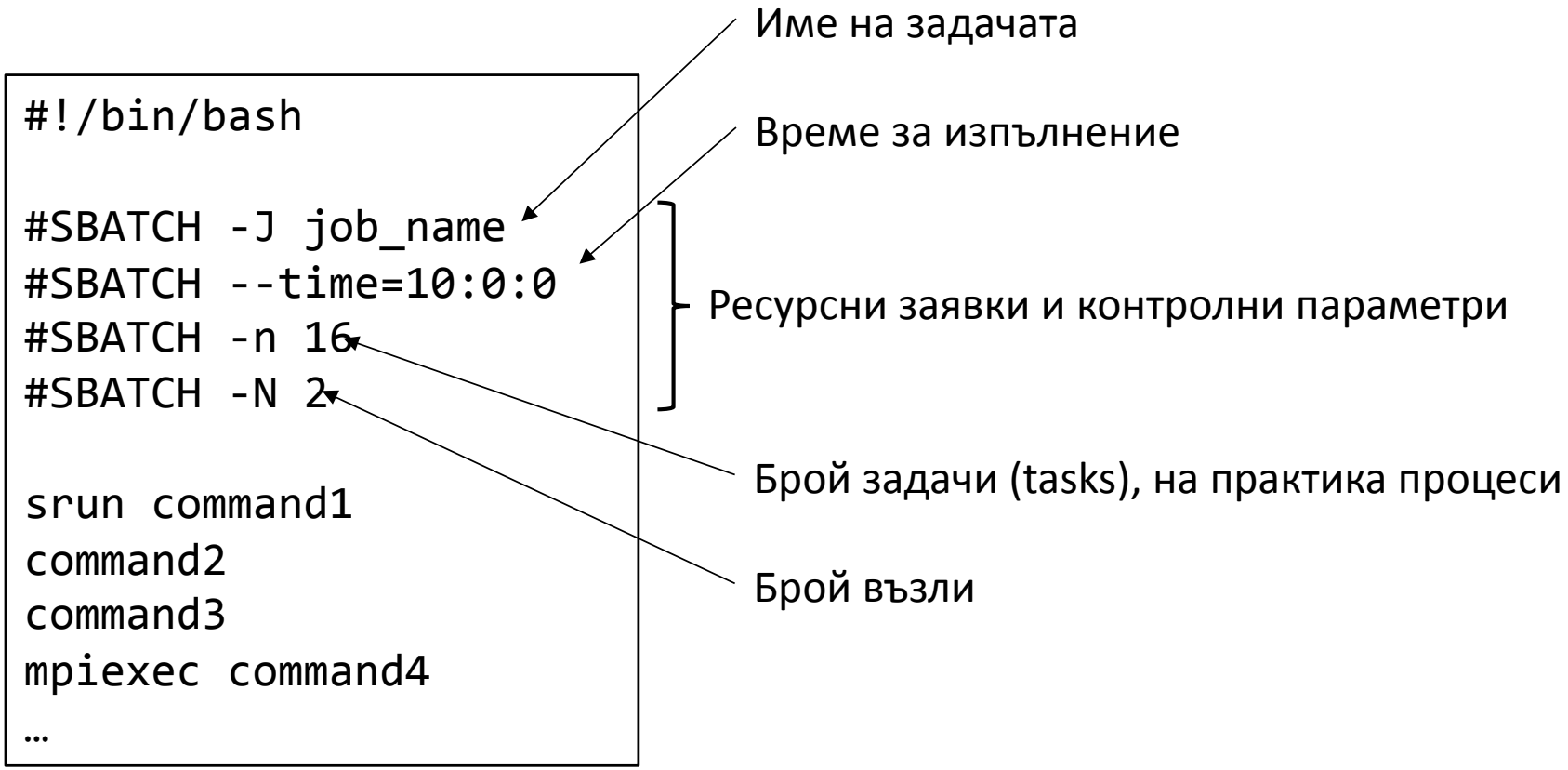
SLURM

- `srun`
  - Заделя необходимите ресурси (`salloc`)
  - Стартира процеси върху заделените ресурси
    - Режими на стартиране
    - Степени на интеграция
  - Пренасочва на В/И канали
- Пакетен режим (задачи) – `sbatch`
  - Заявява необходимите ресурси
  - Асинхронно изпълнение
  - Зависимости между задачите – вериги

SLURM задачи

- Набор ресурсни изисквания
  - Време за изпълнение
  - Брой процесорни ядра / възли
  - Оперативна памет
  - Лицензи
- Контролни параметри
  - Разпределение на процесите
  - “Заковаване“ на процесите
  - Отложено стартиране

SLURM задачи

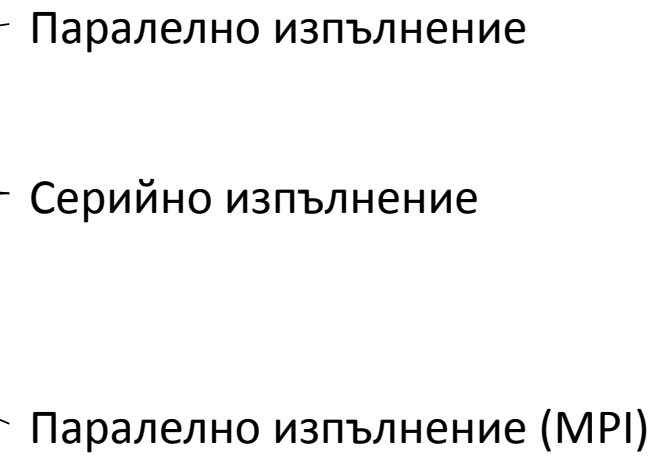


SLURM задачи

```
#!/bin/bash

#SBATCH -J job_name
#SBATCH --time=10:0:0
#SBATCH -n 16
#SBATCH -N 2

srun command1
command2
command3
mpiexec command4
…
```



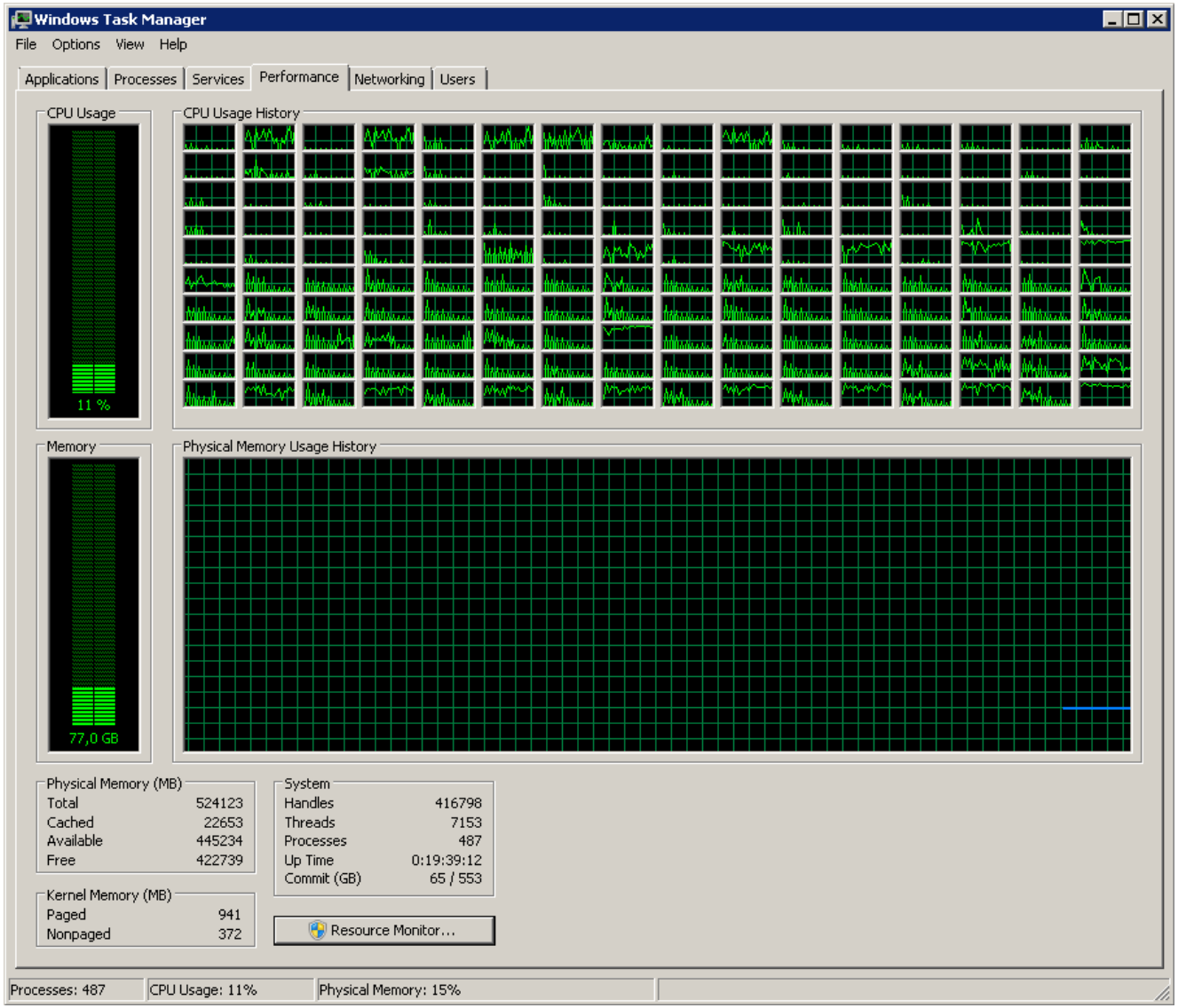
Избор на ОС

- Всяка многозадачна и многопотребителска ОС
- Поддръжка на високоскоростни мрежи
- Поддръжка на паралелни файлови системи
- Автентикация, права, групи, и т.н.
- Отдалечен достъп
- Поддържан софтуер

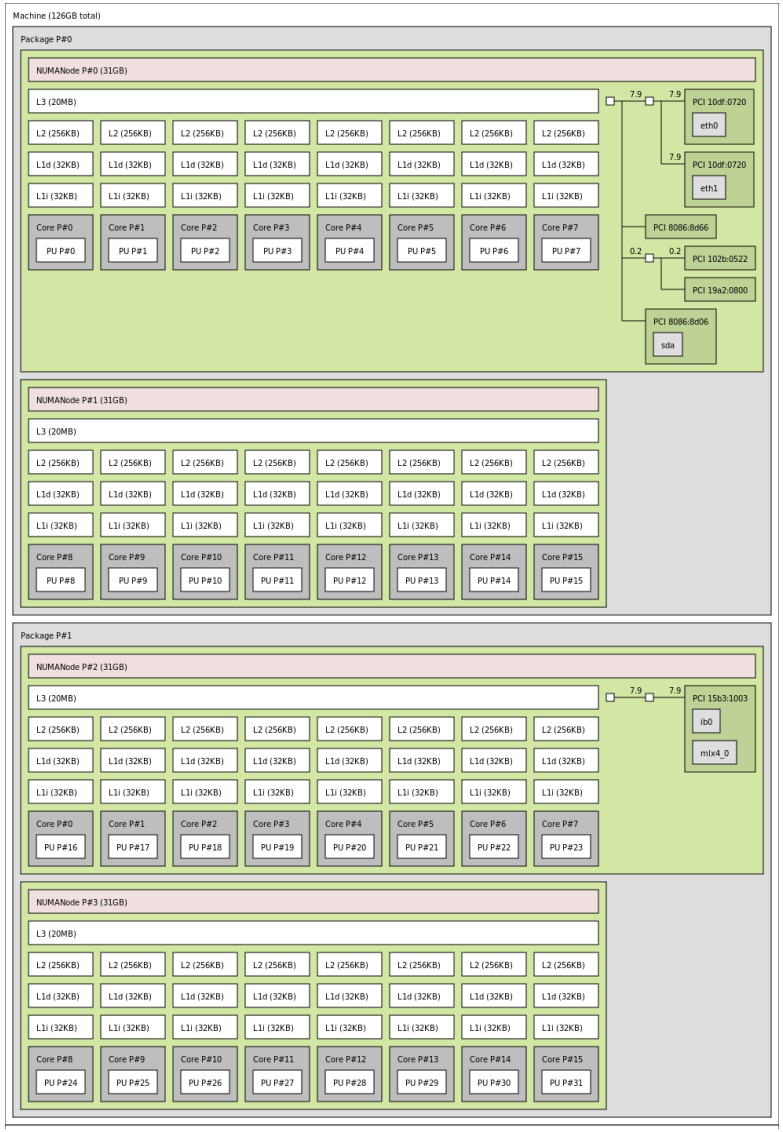
Избор на ОС

- Класически избор – Linux
- Проблем – OS jitter
  - Обработка на прекъсвания
  - Периодични системни процеси
  - Непредсказуеми В/И събития
- Специализирани изчислителни ОС ядра
  - Cray – CNL (Compute Node Linux)
  - IBM BlueGene – CNK (Compute Node Kernel)

Избор на ОС – не само Un*x



Особености при изпълнение

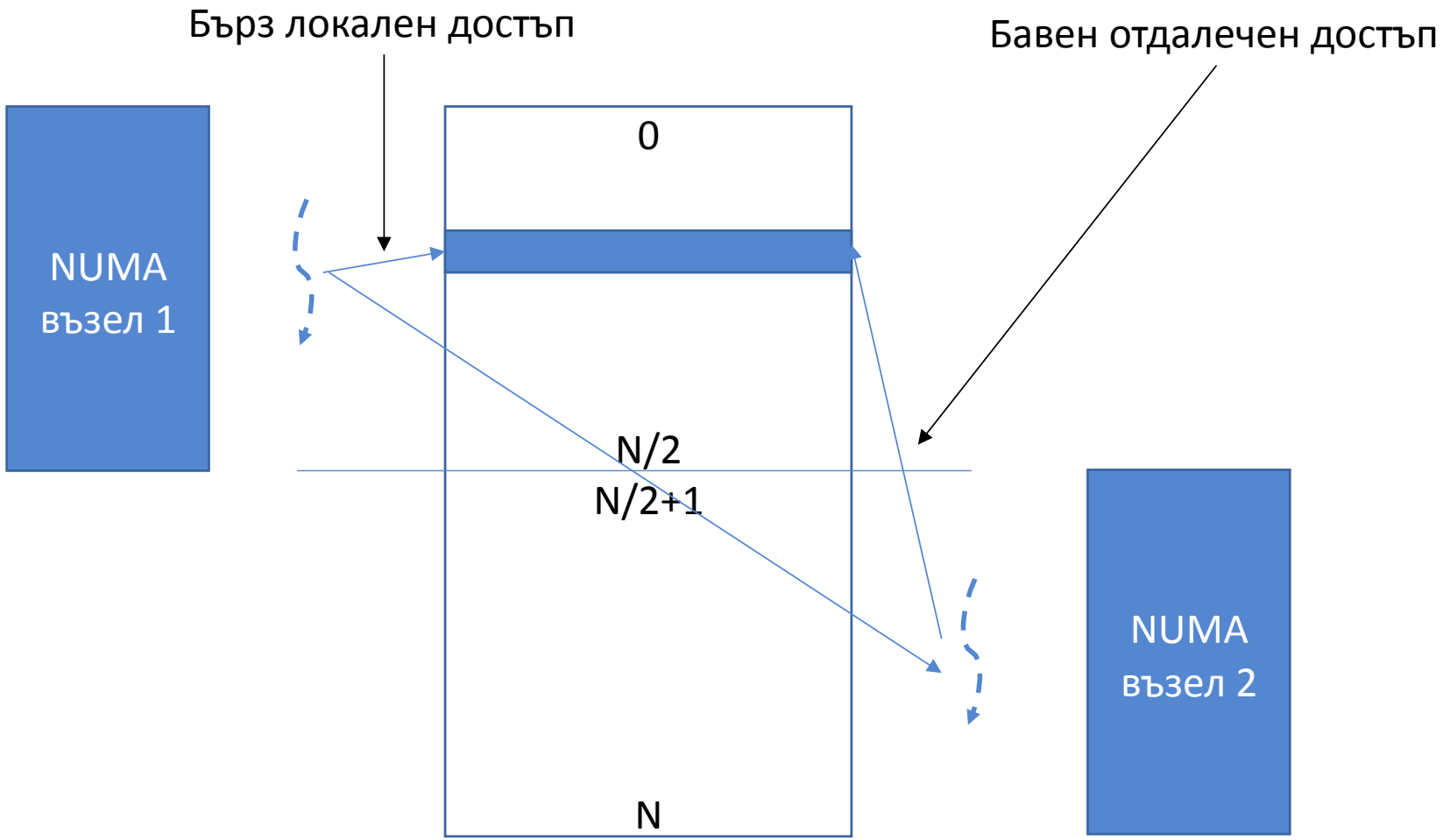


- Възел на Nestum
  - 2 бр. Intel E5-2698 v3
  - 16 ядра на процесор
  - 4 NUMA области!!
  - InfiniBand HCA на втори цокъл

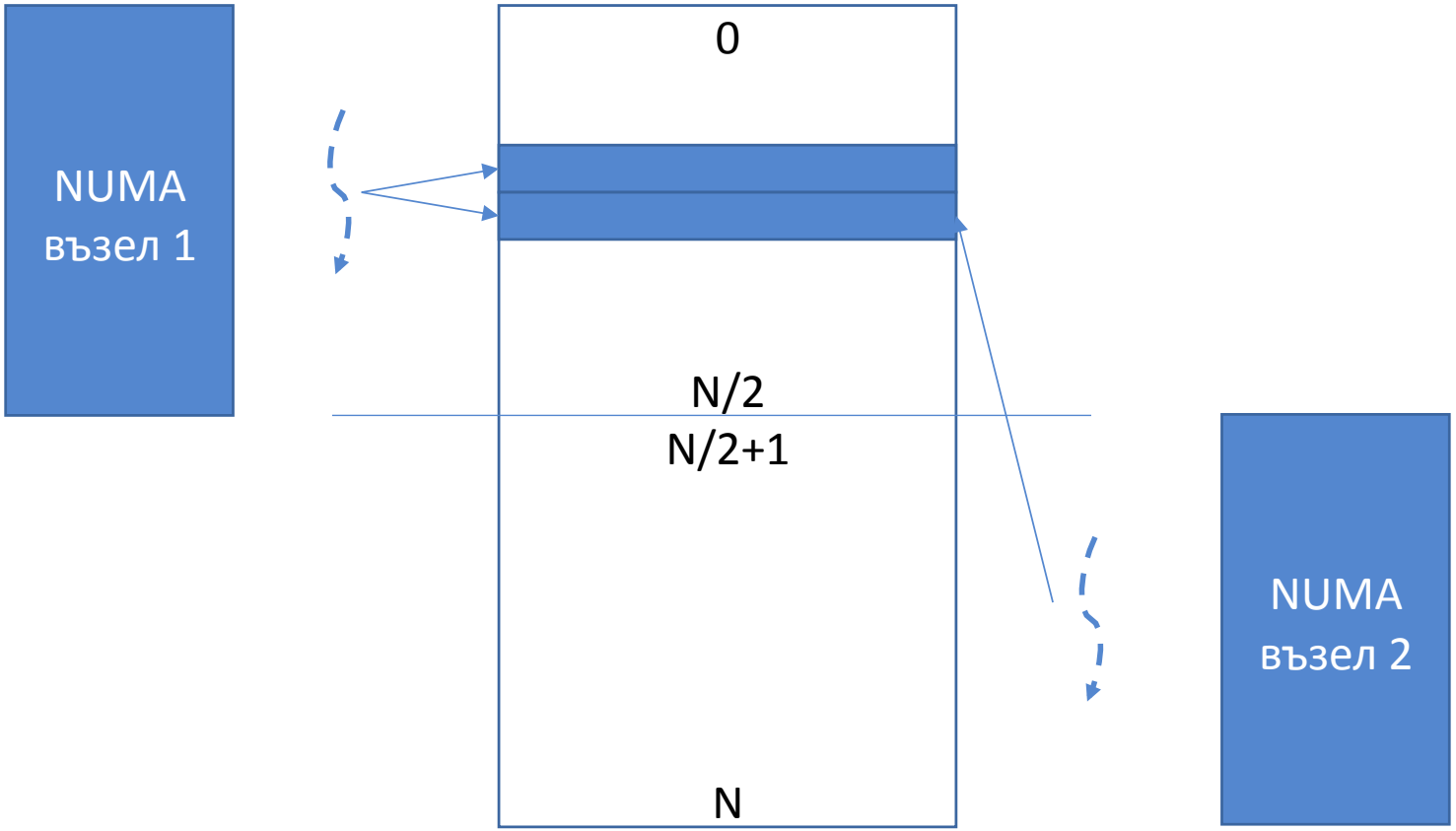
Особености на NUMA

- Политика “first touch”
  - Физическата памет се заема с предимство от областта, където се изпълнява кода
  - Локална памет – поне в началото
- Проблем при миграция на кода върху друг цокъл
  - Локална памет → отдалечена памет
  - Чест резултат от баланса на натоварването в ОС
- Проблем при достъп от код върху друг цокъл
  - Чест резултат от серийна инициализация на данни в паметта

Особености на NUMA



Особености на NUMA



Пределна производителност

- Каква е разликата в паралелното изпълнение на следните сегменти? $y = Ax$

```
for (r = 0; r < M; r++) {
  yy = 0.0;
  for (c = 0; c < N; c++)
    yy += A[r][c] * x[c];
  y[r] = yy;
}
```

A – плътна

```
for (r = 0; r < M; r++) {
  yy = 0.0;
  for (c = A.rptr[r]; c < A.rptr[r+1]; c++)
    yy += A.data[c] * x[A.cidx[c]];
  y[r] = yy;
}
```

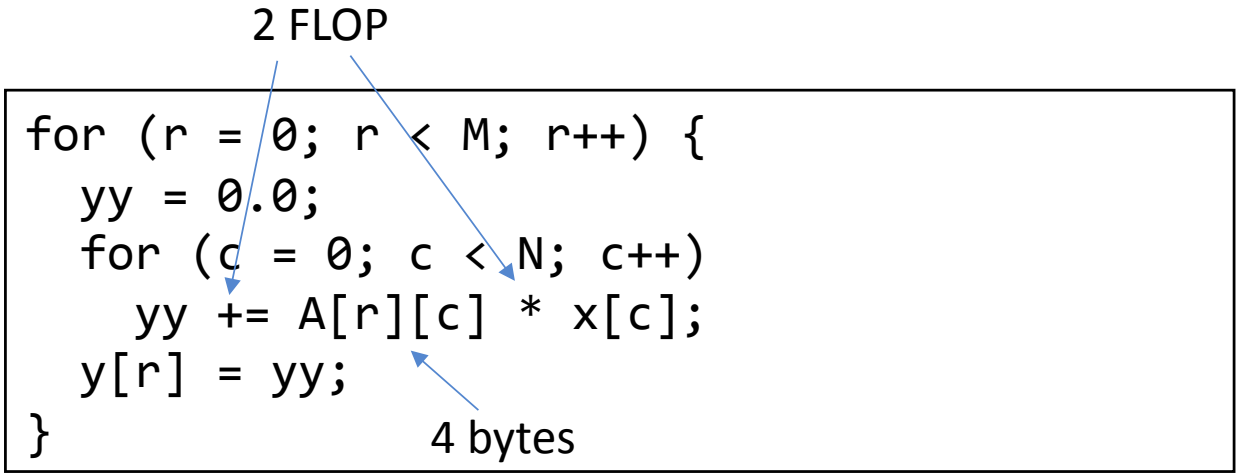
A – разредена
CSR формат

Пределна производителност

```
for (r = 0; r < M; r++) {
  yy = 0.0;
  for (c = 0; c < N; c++)
    yy += A[r][c] * x[c];
  y[r] = yy;
}
```

- M = N = 100000, тип float
- A – 40 GB, извън кеша; x и y – 800 KB, в L3 кеша
- Производителност на ядро – 18,4 GFLOP/s
  - 2,3 GHz x 2 FLOP/cycle x 4-wide SIMD
- До 68 GB/s към паметта

Пределна производителност



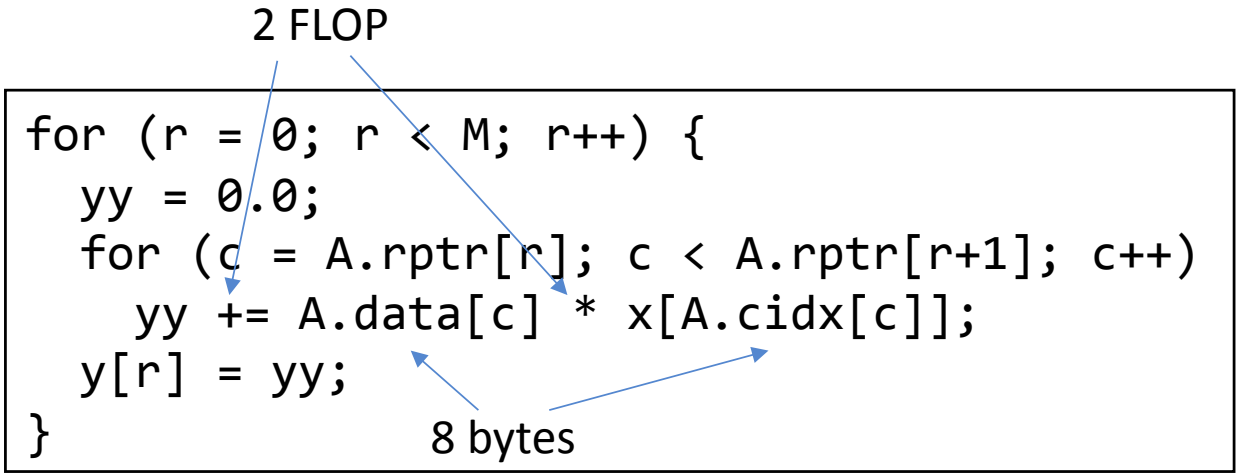
- Аритметична интензивност – 0,5 FLOP/byte
- 68 GB/s * 0,5 FLOP/byte → 34 GFLOP/s
  - >100% от производителността на 1 ядро
  - 92% от производителността на 2 ядра

Пределна производителност

```
for (r = 0; r < M; r++) {
  yy = 0.0;
  for (c = A.rptr[r]; c < A.rptr[r+1]; c++)
    yy += A.data[c] * x[A.cidx[c]];
  y[r] = yy;
}
```

- M = N = 100000, тип float
- A – 1% (100 млн) ненулеви елементи
  - A.data – 400 MB, извън кеша
  - A.cidx – 400 MB, извън кеша
  - A.rptr – 400 KB, в L3 кеша
- x и y – 800 KB, в L3 кеша

Пределна производителност



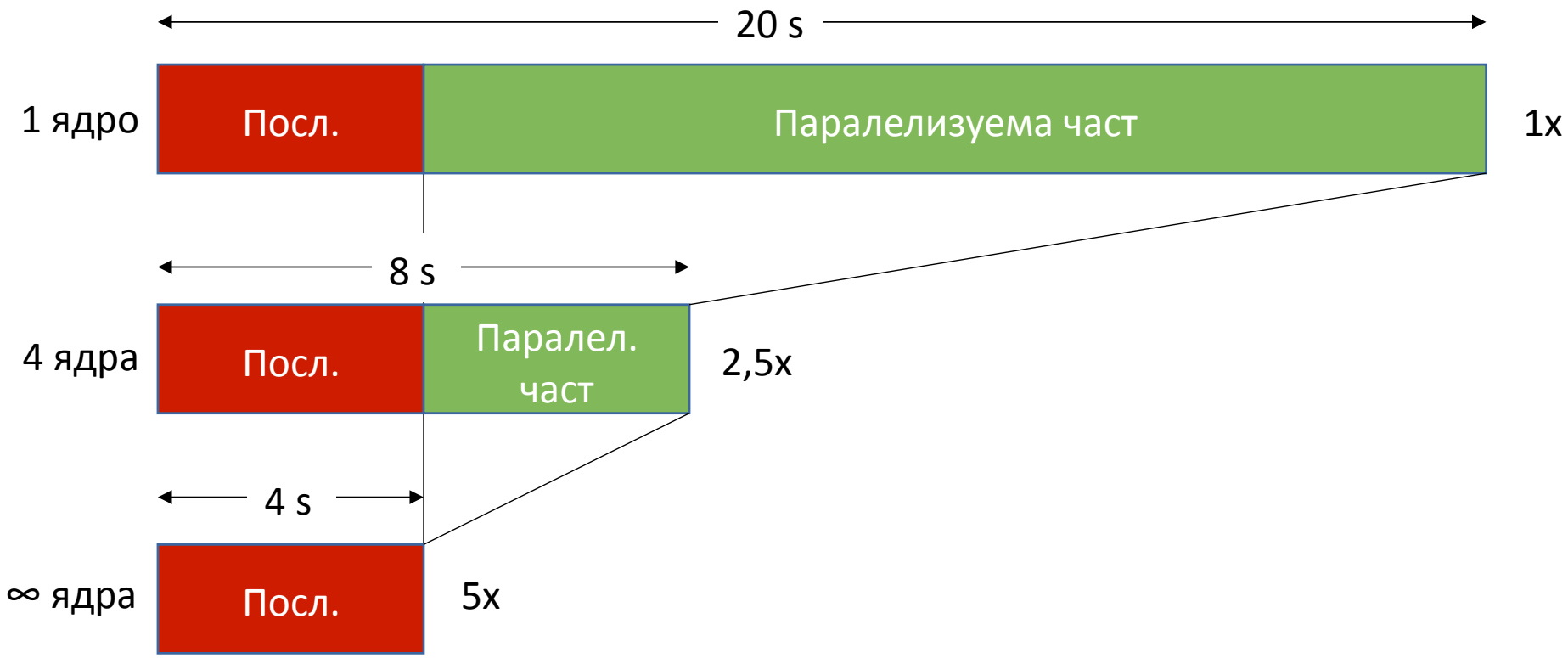
- Аритметична интензивност – 0,25 FLOP/byte
- 68 GB/s * 0,25 FLOP/byte → 17 GFLOP/s
  - 92% от производителността на 1 ядро
- Как можем да удвоим производителността?
  - Използваме ядро от друг цокъл (внимание, NUMA!)

Пределна производителност

- Два много различни класа приложения
- Ограничени откъм процесорна мощност
  - 1,8x ускорение на две ядра от един цокъл
  - 2x ускорение на две ядра от различни цокли
- Ограничени откъм пропускателна способност на паметта
  - Без промяна на две ядра от един цокъл
  - 2x ускорение на два ядра от различни цокли
  - Възможна оптимизация на енергопотреблението

Пределна производителност

- Закон на Амдал



Реална производителност

- Сравнителни тестове (benchmarks)
- HPL
  - High Performance LINPACK
  - TOP500
  - Висока аритметична интензивност
- HPCG
  - High Performance Conjugate Gradients
  - Допълнителен тест в TOP500
  - Ниска аритметична интензивност

Познавайте си приложенията

- Преглед на кода
  - Само за къси програми
- Измерване с хардуерни монитори и броячи
  - Налични в повечето x86 процесори

- LIKWID
  - Сравнителни тестове
  - Измерващи инструменти
  - Единичен възел

Познавайте си приложенията

- Score-P
  - Измервателна инфраструктура
  - Статистика на извикване на функции
  - Трасиране на MPI съобщения
    - Визуализация с комерсиален инструмент – Vampir
- Scalasca
  - Анализ от високо ниво
  - Открива източници на закъснения в MPI програми

Общи препоръки

- Оптимизирани библиотеки
  - Intel MKL вместо LAPACK, BLAS, ScaLAPACK
  - Intel MKL вместо FFTW
  - Intel OpenMP Runtime вместо libgomp
- Подобрени алгоритми
  - ELPA вместо ScaLAPACK за паралелно пресмятане на собствени стойности
- Оптимизиращи компилатори
  - Intel C/C++/Fortran вместо GCC

Благодарности

Авторът изразява благодарност за финансовата подкрепа, получена от „СДРУЖЕНИЕ ЗА НАУЧНОИЗСЛЕДОВАТЕЛСКА И РАЗВОЙНА ДЕЙНОСТ“ (СНИРД), в рамките на вътрешния проект с Лабораторията по високопроизводителни изчисления на тема на тема „Паралелни пресмятания в сложни системи и процесите в тях“, 20.06.2018-31.10.2018 с ръководител проф. дфзн Ана Пройкова