Applied Parallel Computing
parallel-computing.pro

Введение в параллельные вычисления

к.т.н. Алексей Ивахненко

- Что такое параллельные вычисления, насколько они распространены, востребованы и выгодны?
- Как устроены суперкомпьютеры?
- MPI, OpenMP, OpenACC

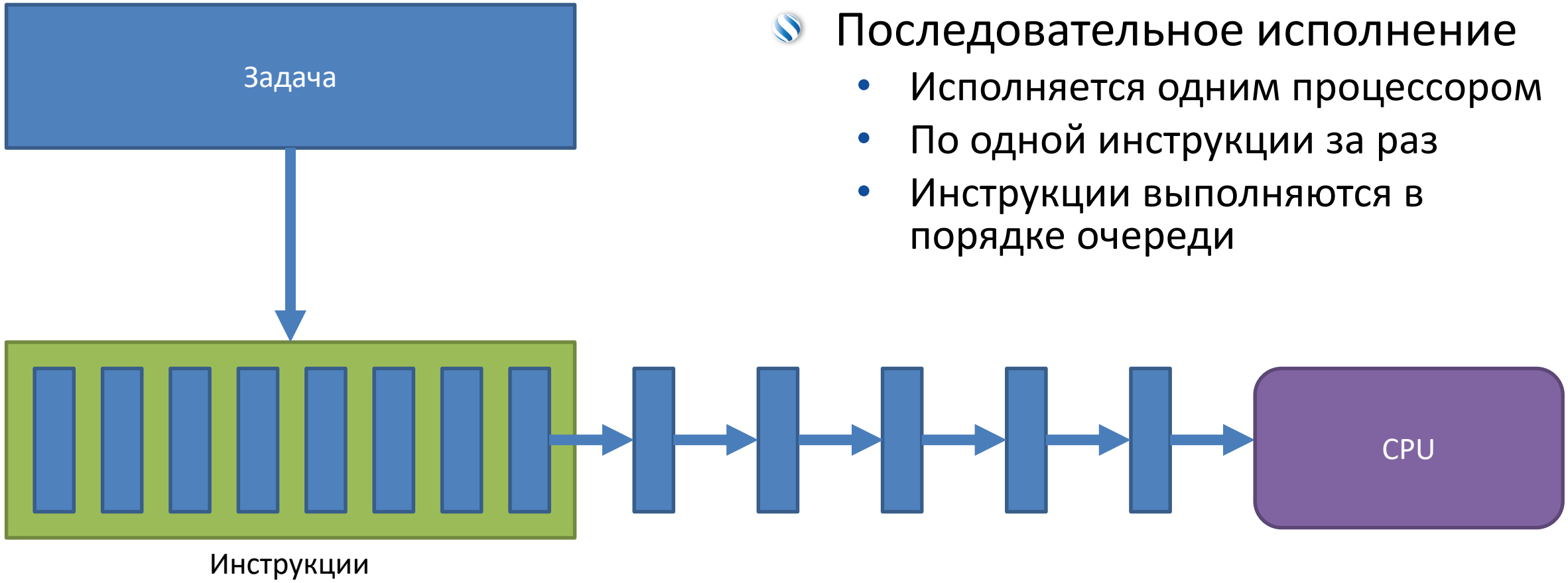
План

- Как работать с большими объемами данных и вычислений?
- Масштабируемость. Закон Амдаля.
- Производительность (FLOPS), пропускная способность памяти.
- Приложения, ограниченные вычислительными возможностями и пропускной способностью памяти.
- Частота процессора и частота памяти. Значение кэшей.
- Векторизация.
- Наборы векторных инструкций.
- Частота процессора и TDP.
- Примеры многоядерных систем:
  - Intel CPU
  - Cell Broadband Engine
  - Cavium ThunderX

План

- Roofline модель оценки производительности.
- Распределенная архитектура памяти.
- Важность размещения данных в памяти.
- Схема современного гибридного кластера с множеством сокетов, ядер и ускорителей.
- Классификация Флинна.
- Общая память и распределенная память.
- Ошибки, специфичные для параллельных вычислений: deadlocks, memory races.
- Потоки и процессы.
- Синхронное и Асинхронное исполнение.
- Критические секции, мьютексы и барьеры.
- Сигналы, сообщения, события для коммуникации между процессами.
- Наборы утилит: cpuinfo, lspci. Управление задачами на сервере с помощью SLURM.

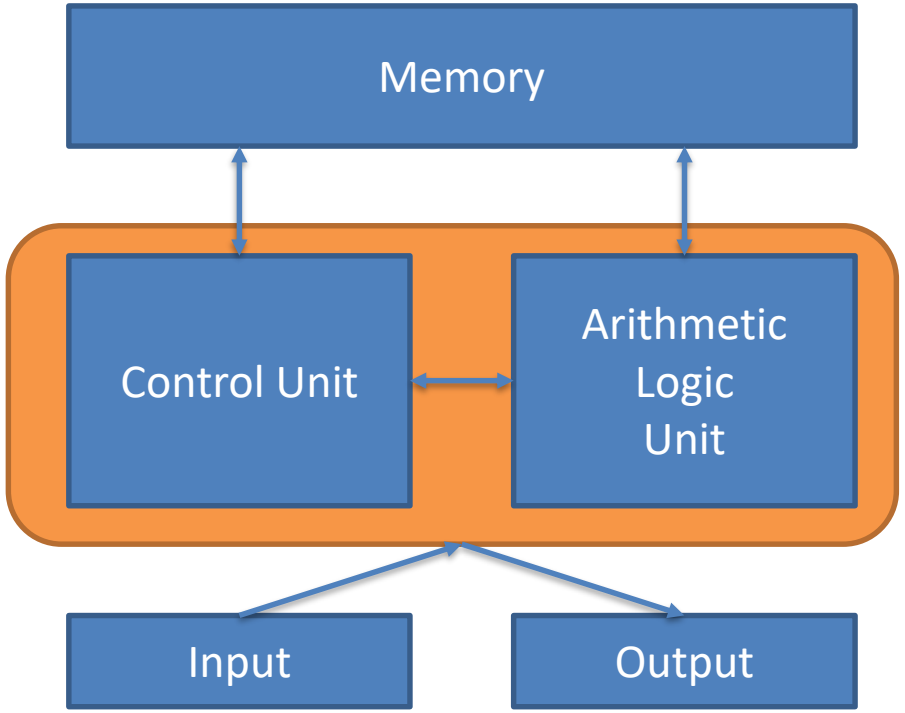
Последовательное исполнение



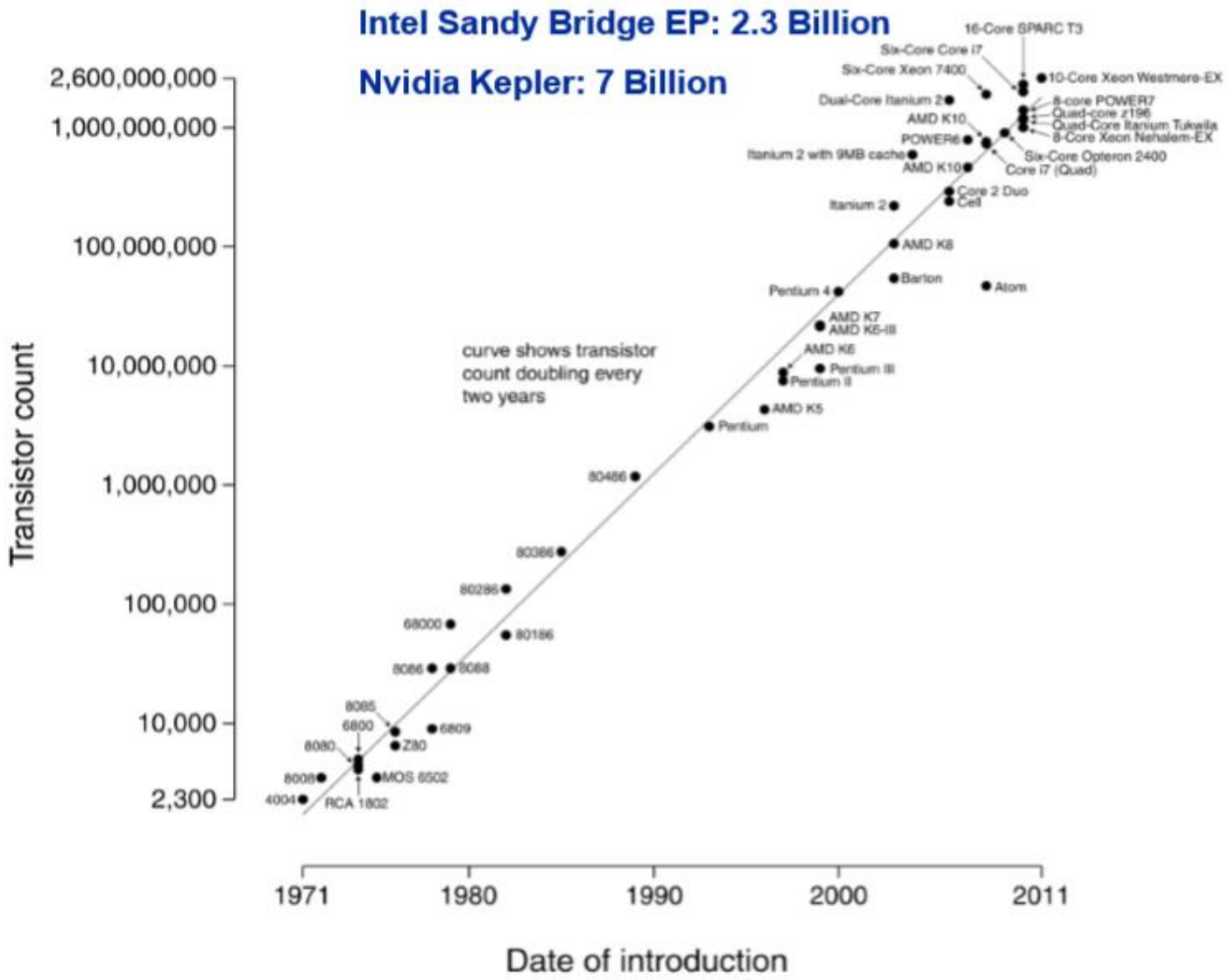
- Последовательное исполнение
  - Исполняется одним процессором
  - По одной инструкции за раз
  - Инструкции выполняются в порядке очереди

Архитектура фон Неймана

- Данные и инструкции хранятся в памяти
- Control unit загружает инструкции и данные из памяти, декодирует инструкции и последовательно выполняет операции
- АЛУ выполняет базовые арифметические операции
- I/O – интерфейсы для взаимодействия с оператором



Закон Мура


Intel Sandy Bridge EP: 2.3 Billion
Nvidia Kepler: 7 Billion
curve shows transistor count doubling every two years
Transistor count
2,600,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
1971
1980
1990
2000
2011
Date of introduction
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westmere-EX
Dual-Core Itanium 2
8-core POWER7
Quad-core z196
Quad-Core Itanium Tukwila
AMD K10
8-Core Xeon Nehalem-EX
POWER6
Itanium 2 with 9MB cache
Six-Core Opteron 2400
AMD K10
Core i7 (Quad)
Core 2 Duo
Cell
Itanium 2
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
80486
80386
80286
68000
80186
8086
8088
8085
6800
6809
8080
Z80
8008
MOS 6502
4004
RCA 1802

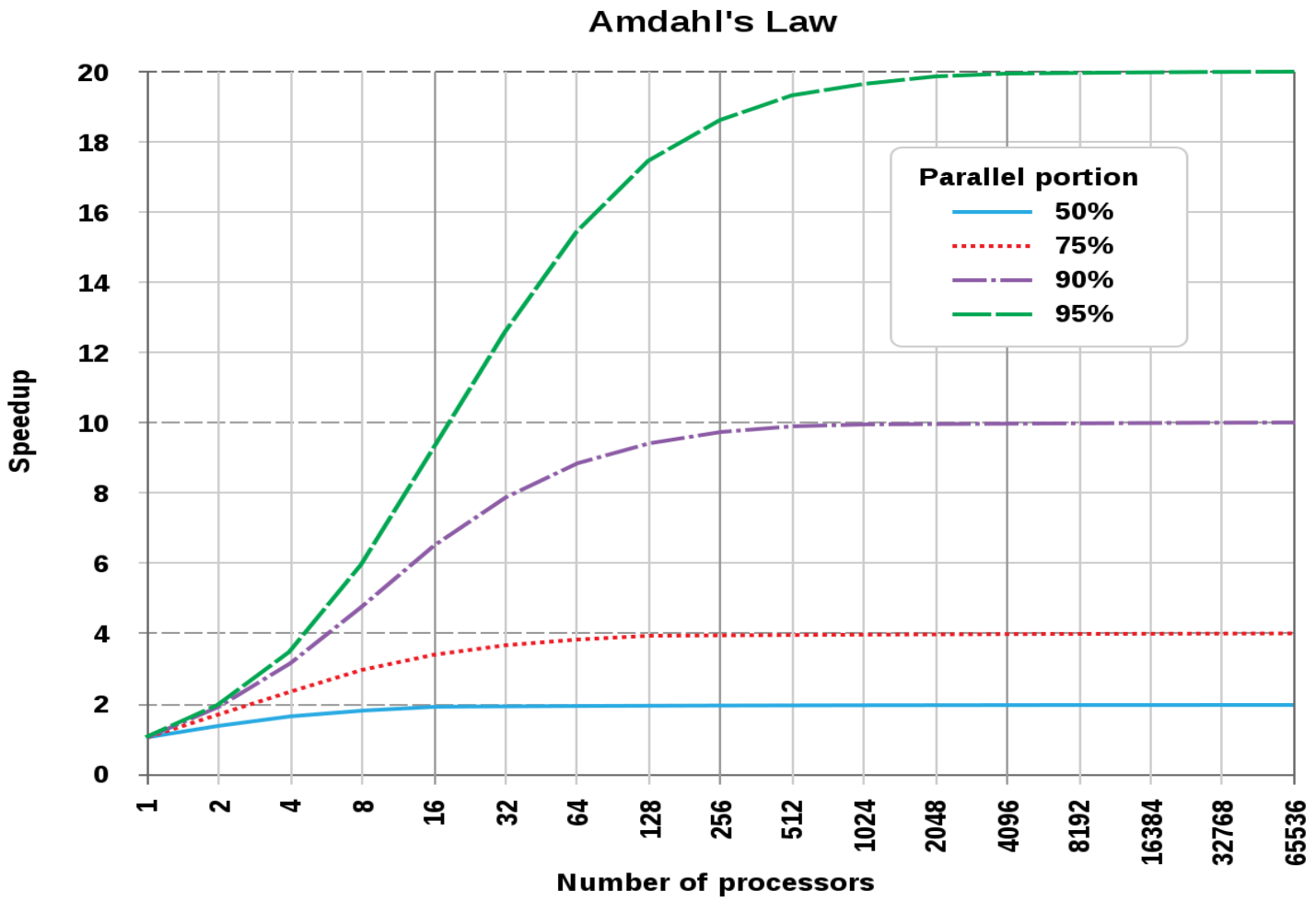
Закон Амдаля

$S(p) \leq 1/(f + (1 - f)/p)$

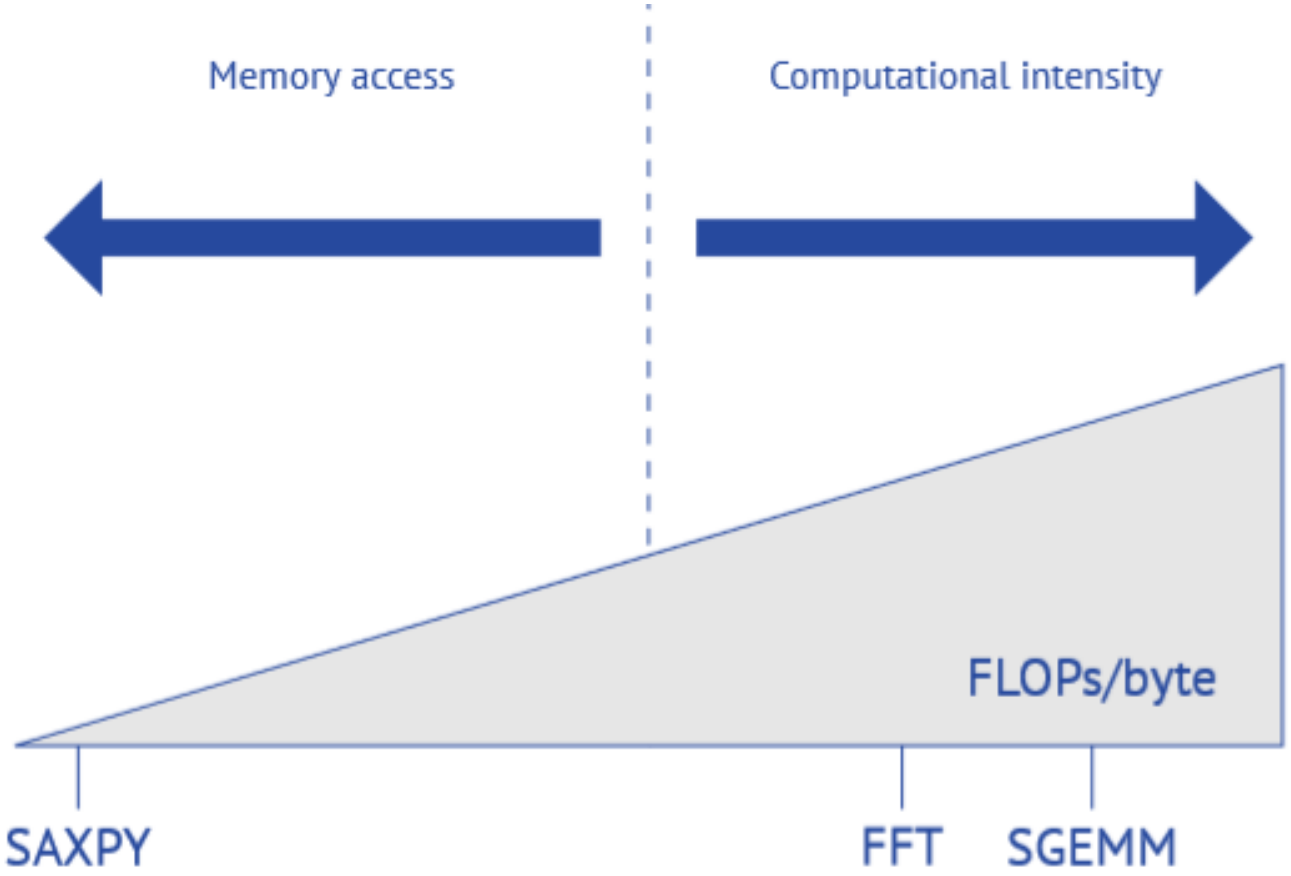
p – количество процессоров

f – доля программы, которая должна быть выполнена последовательно

S - ускорение



Ограничения



Optimal computational intensity for different GPUs (FLOPS : byte):

Tesla C1060	9.1 : 1
Tesla C2050	7.1 : 1
Tesla K20	16.9 : 1
Tesla K40	17.5 : 1
Tesla M40	23.7 : 1
Tesla P100	12.9 : 1
NVIDIA Titan X	22.8 : 1

Top Supercomputers

System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC	10,649,600	93,014.60	125,435.90	15,371
National Supercomputing Center in Wuxi				
China				
Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P , NUDT	3,120,000	33,862.70	54,902.40	17,808
National Super Computer Center in Guangzhou				
China				
Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc.	361,760	19,590.00	25,326.30	2,272
Swiss National Supercomputing Centre (CSCS)				
Switzerland				
Gyoukou - ZettaScaler-2.2 HPC system, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 700Mhz , ExaScaler	19,860,000	19,135.80	28,192.00	1,350
Japan Agency for Marine-Earth Science and Technology				
Japan				
Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x , Cray Inc.	560,640	17,590.00	27,112.50	8,209
DOE/SC/Oak Ridge National Laboratory				
United States				

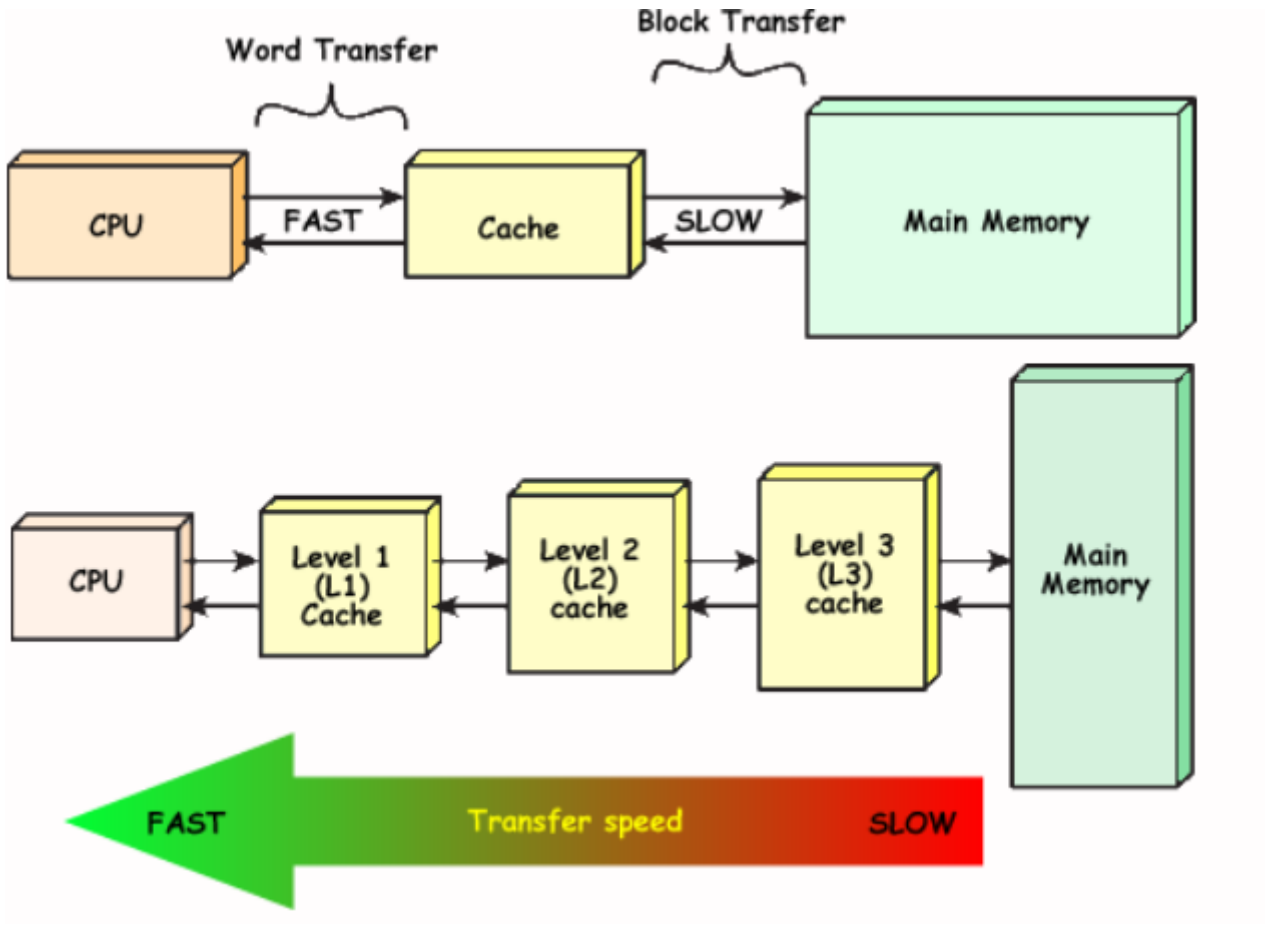
Методика тестирования

- Бенчмарк HPL (High-Performance Linpack): выбирается наибольший размер задачи, помещающийся в память.
- Предоставляет следующую информацию
  - Rpeak: теоретический максимум производительности (FLOPS) системы (FLOPS*Количество тактов CPU*количество CPU)
  - Rmax: Максимум, достигнутый для заданного размера задачи.
  - Nmax: размер матрицы
  - N1/2: Размер задачи при котором достигается 50% Rmax. Низкое значение N1/2 показывает способность системы работать со множеством различных размеров задачи (меньше-лучше).

Основные ограничители

- Вычислительные возможности
  - FLOPS
- Пропускная способность памяти
  - Чтение/запись
  - Обмен сообщениями и сигналами
  - Кэши
- Латентность

Кэш


Word Transfer
Block Transfer
CPU
FAST
Cache
SLOW
Main Memory
CPU
Level 1 (L1) Cache
Level 2 (L2) cache
Level 3 (L3) cache
Main Memory
FAST
Transfer speed
SLOW

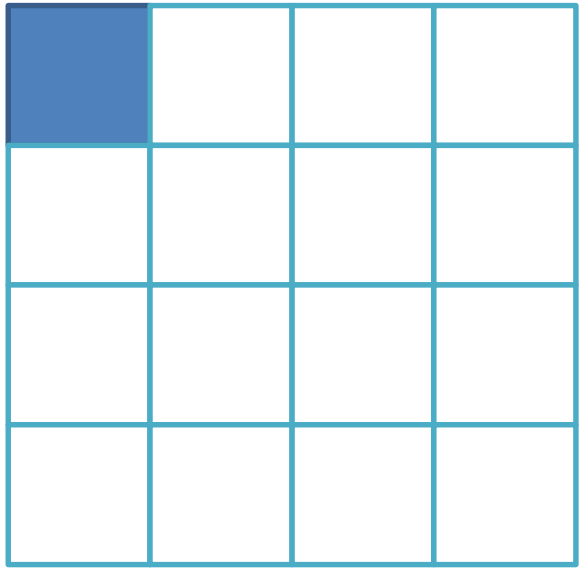
Последовательное исполнение

$$C_0 = A_0 + B_0$$

$$C_1 = A_1 + B_1$$

$$\cdots\cdots\cdots\cdots$$

$$C_{n-1} = A_{n-1} + B_{n-1}$$

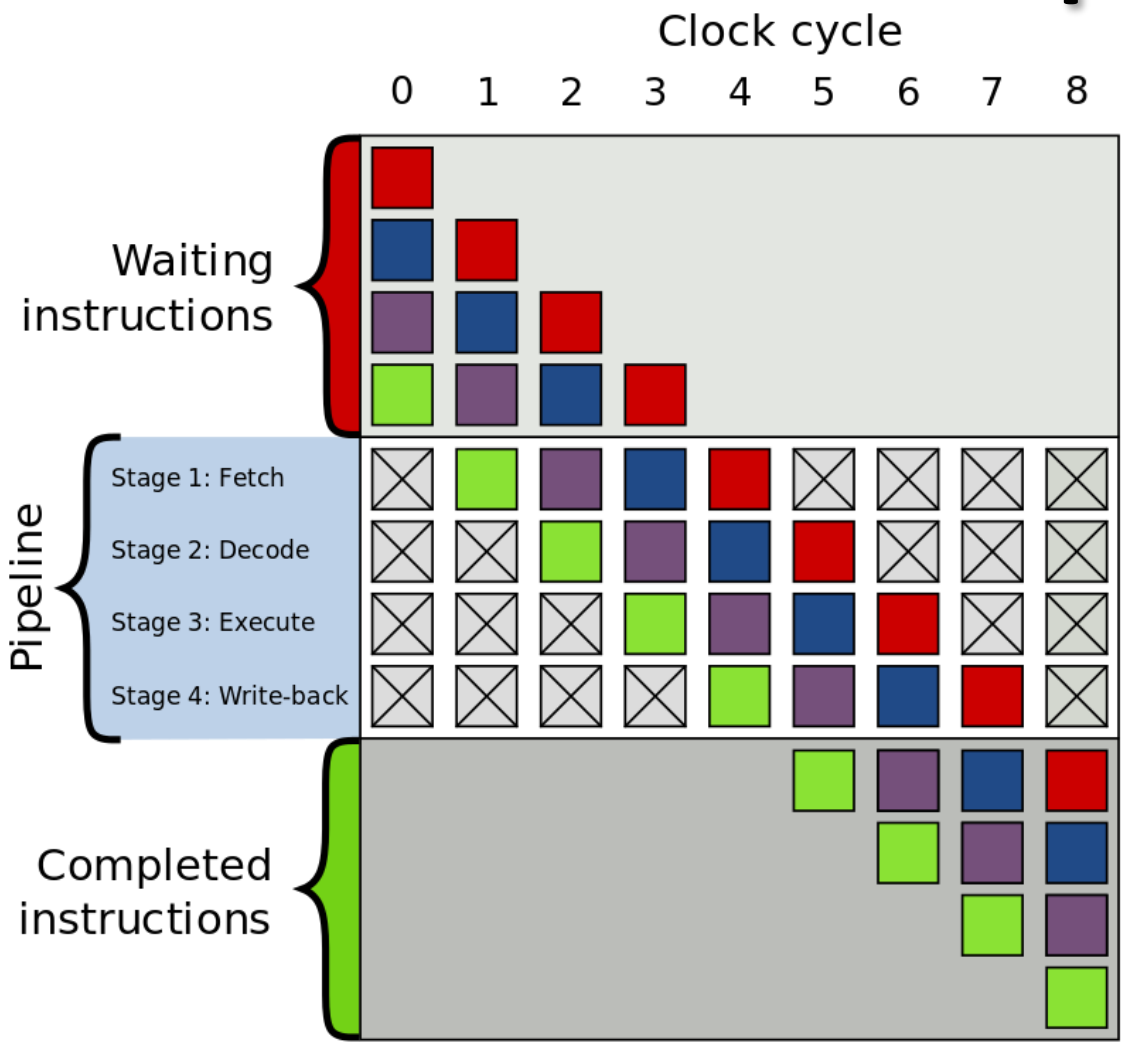


Последовательное исполнение

```
multiply_and_add(const float* a, const float* b, const float* c, float* d)
{
for(int i=0; i<8; i++)
 {
   d[i] = a[i] * b[i];
   d[i] = d[i] + c[i];
 }
}
```

Конвейер

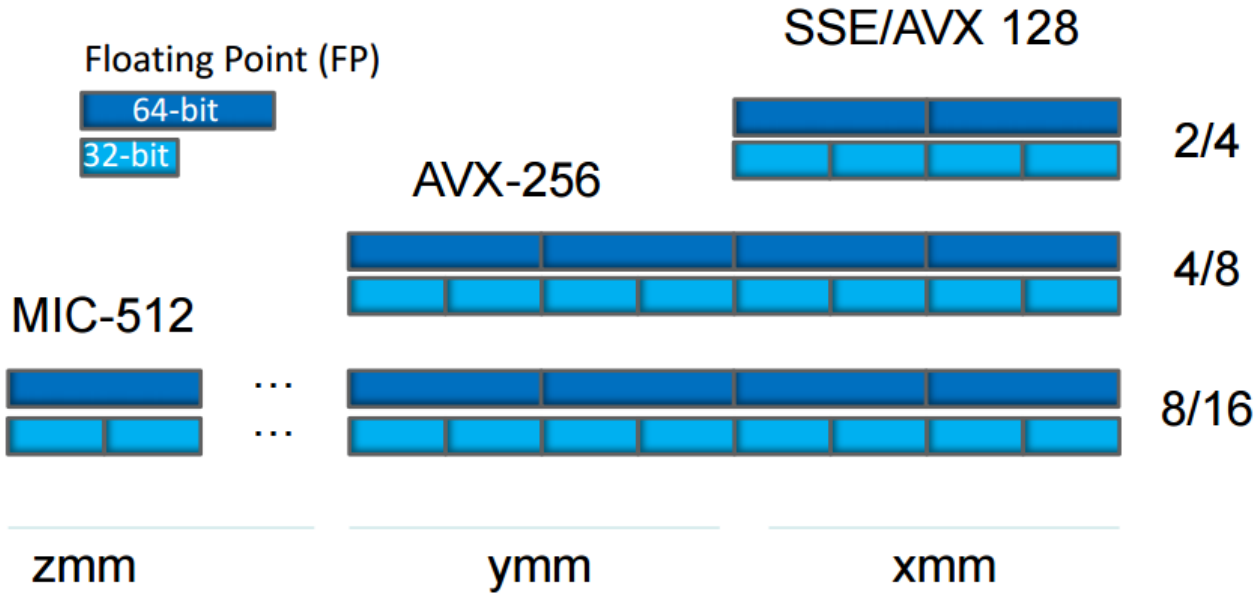
- Общий конвейер с четырьмя стадиями работы:
  - Получение
  - Раскодирование
  - Выполнение
  - Запись результата

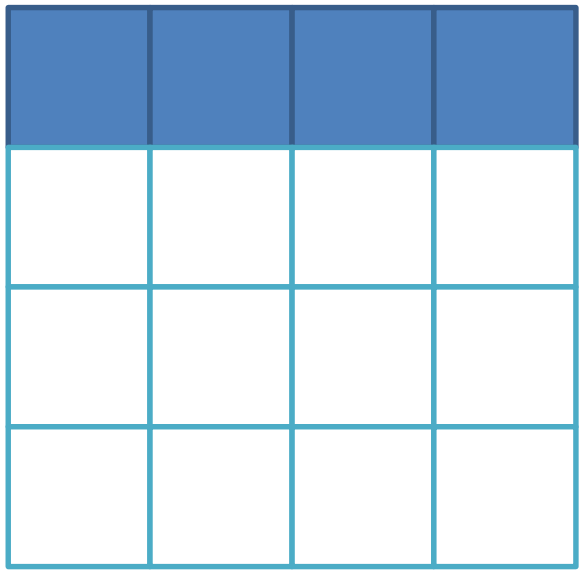


ILP

```
multiply_and_add(const float* a, const float* b, const float* c, float* d)
{
 #pragma unroll 4
 for(int i=0; i<8; i++)
 {
   d[i] = a[i] * b[i];
   d[i] = d[i] + c[i];
 }
}

__m256 multiply_and_add(__m256 a, __m256 b, __m256 c)
{
 return _mm256_fmadd_ps(a, b, c);
}
```

Векторные инструкции





Векторные инструкции

Векторизация

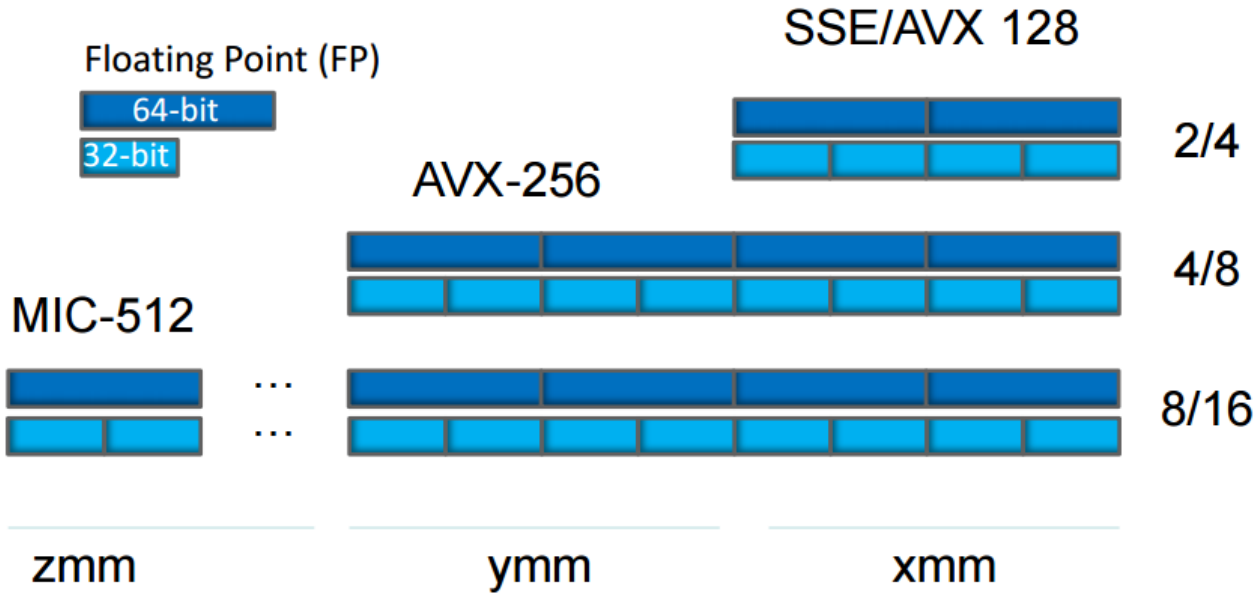
```
multiply_and_add(const float* a, const float* b, const float* c, float* d)
{
 for(int i=0; i<8; i++)
 {
   d[i] = a[i] * b[i];
   d[i] = d[i] + c[i];
 }
}


__m256 multiply_and_add(__m256 a, __m256 b, __m256 c)
{
 return _mm256_fmadd_ps(a, b, c);
}
```

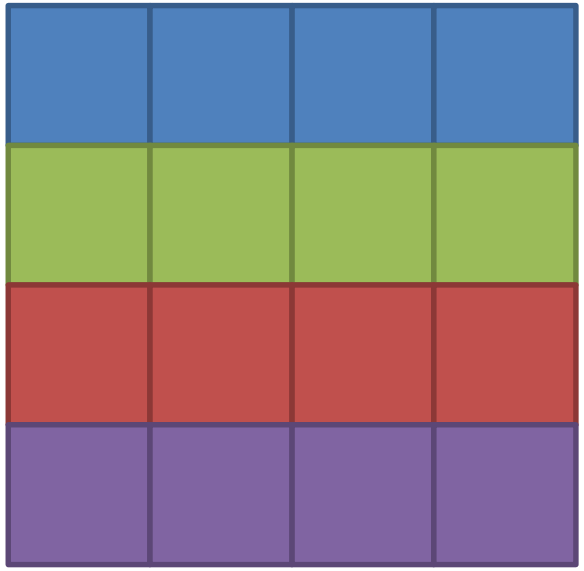
AVX/AVX2

- В общем виде инструкция AVX/AVX2 называется следующим образом:
  - _mm<bit_width>_<name>_<data_type>
- <data_type>:
  - ps - vectors contain floats (ps stands for packed single-precision)
  - pd - vectors contain doubles (pd stands for packed double-precision)
  - epi8/epi16/epi32/epi64 - vectors contain 8-bit/16-bit/32-bit/64-bit signed integers
  - epu8/epu16/epu32/epu64 - vectors contain 8-bit/16-bit/32-bit/64-bit unsigned integers
  - si128/si256 - unspecified 128-bit vector or 256-bit vector
  - m128/m128i/m128d/m256/m256i/m256d - identifies input vector types when they're different than the type of the returned vector

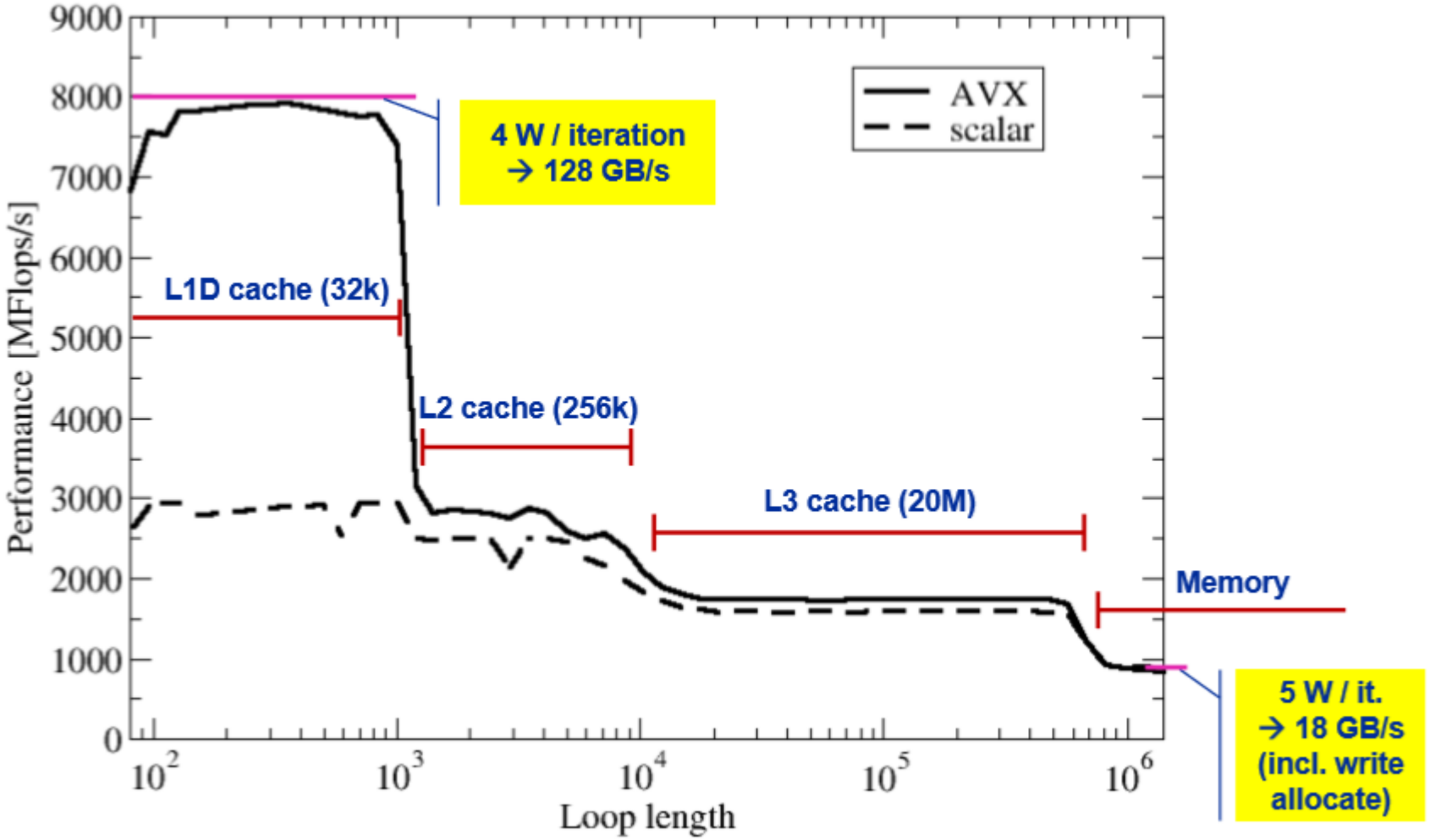
Data Type	Description
__m128	128-bit vector containing 4 floats
__m128d	128-bit vector containing 2 doubles
__m128i	128-bit vector containing integers
__m256	256-bit vector containing 8 floats
__m256d	256-bit vector containing 4 doubles
__m256i	256-bit vector containing integers

Векторные инструкции и параллельное исполнение

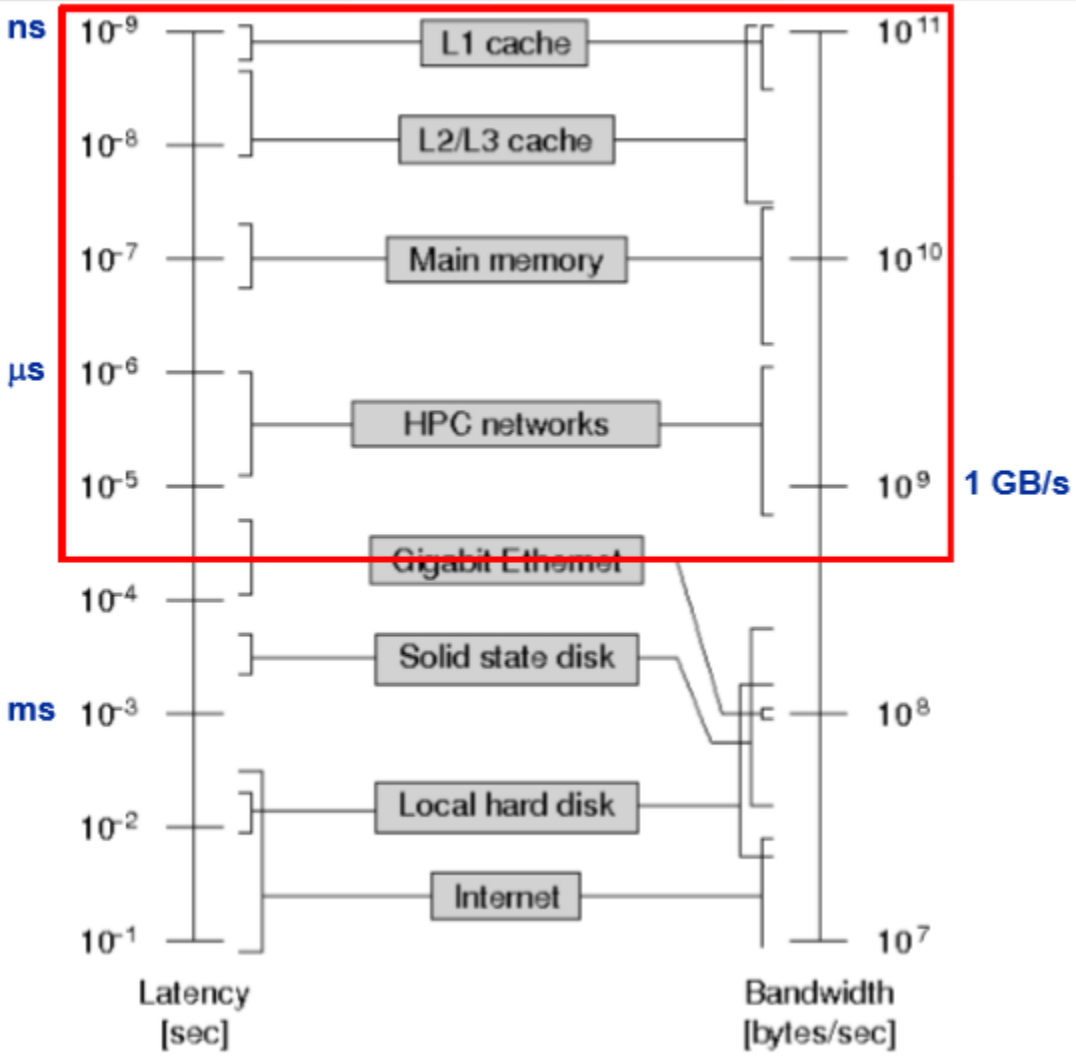




Влияние кэшей

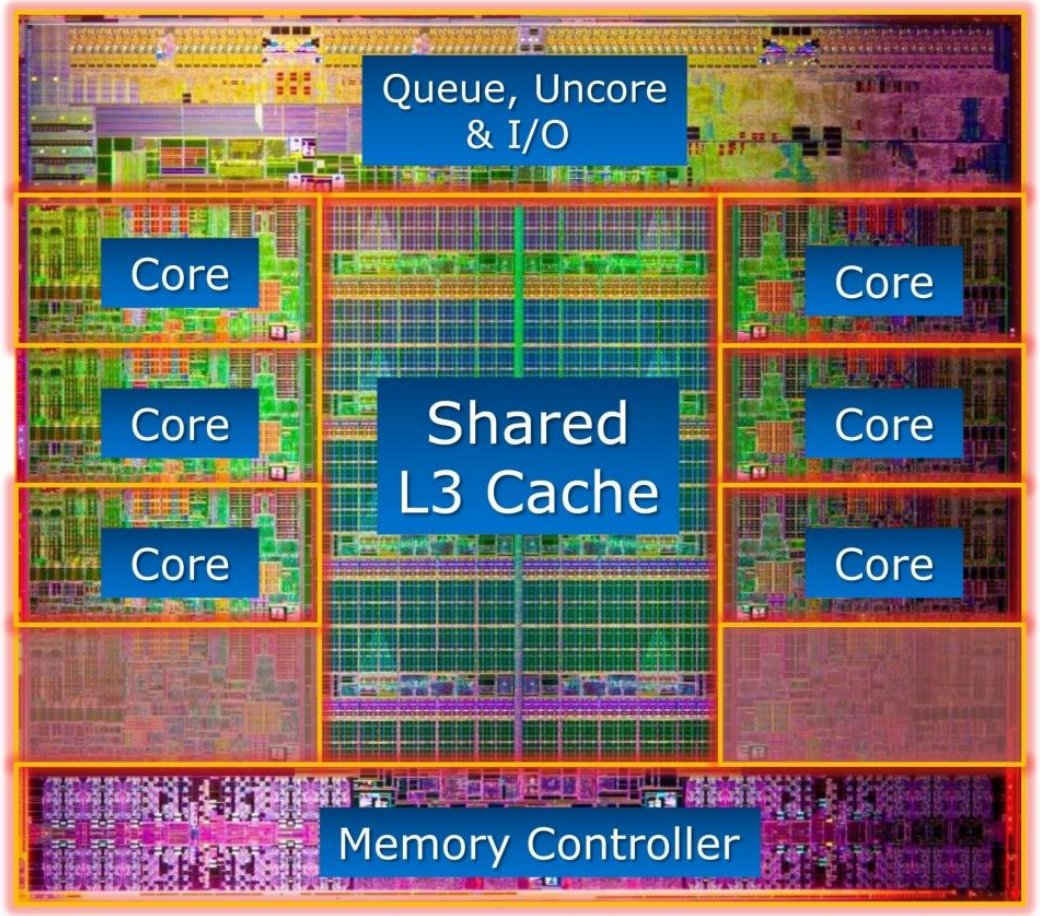


Латентность



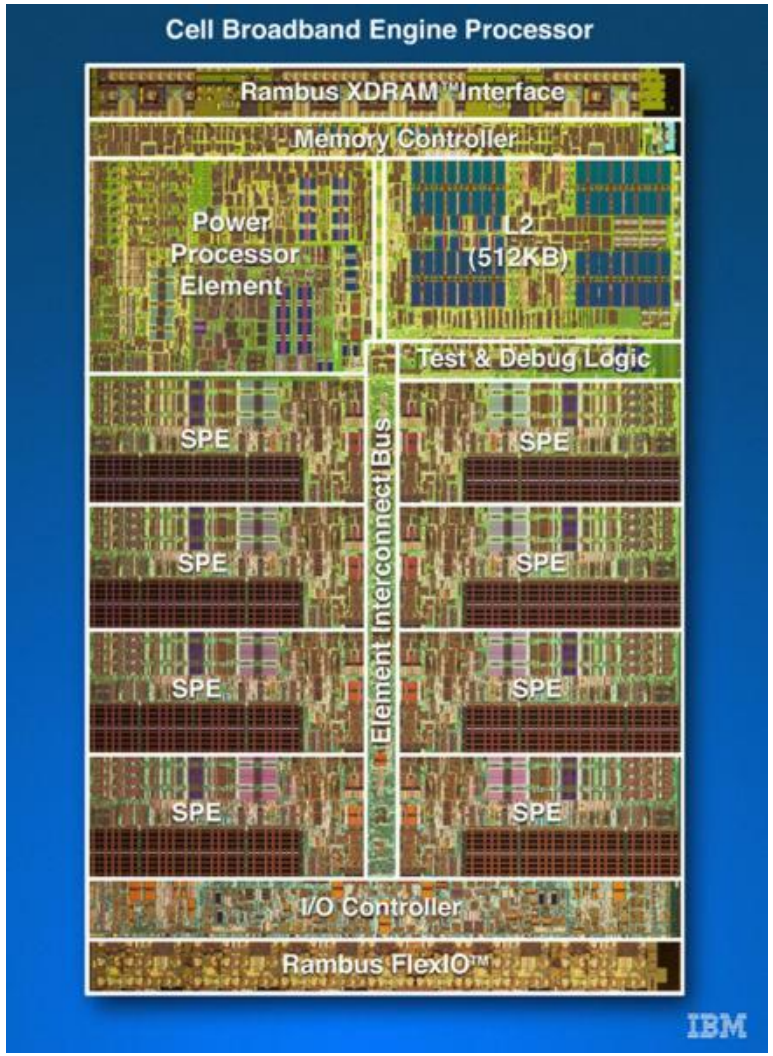
Core i7

- Несколько мощных ядер
  - 2,4,6,8 ядер, 2,66—3,6ГГц каждое
  - Каждое физическое ядро представляется в системе как 2 логических и может исполнять 2 потока конкурентно (Hyper-Threading)
- 3 уровня кэша, большой объем L3
  - На ядро: L1=32KB (data) + 32KB (Instructions), L2=256KB
  - Shared L3, до 20 MB
- Запросы к памяти управляются раздельно для каждого потока/процесса

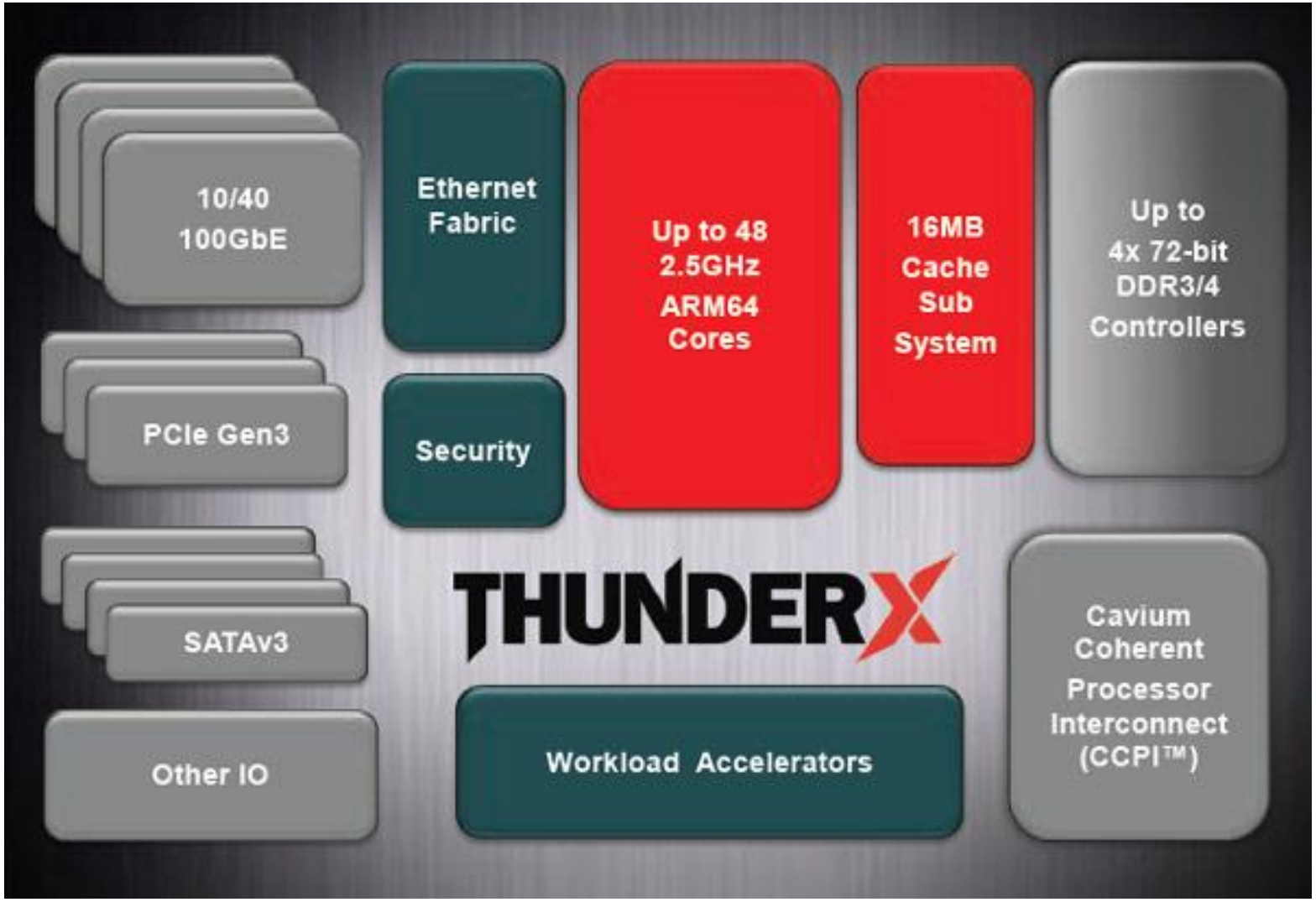


Cell Broadband

- **Cell (Cell Broadband Engine Architecture)**
  - микропроцессорная архитектура, совместно разработанная Sony, Toshiba и IBM.
  - Разработка архитектуры и первые прототипы были созданы в STI Design Center за четырёхлетний период с начала марта 2001 года.
  - Cell совмещает ядро общего назначения архитектуры POWER с сопроцессорами, которые значительно ускоряют обработку мультимедиа и векторных вычислений.



Cavium ThunderX



Nvidia Volta V100

- Volta SMM Specifications:
  - 84 SMs
  - 4096-bit HBM2 memory interface
  - 6144 KB L2 cache
  - NVLink support
  - 12nm FFN
  - 21.1 billion transistors
  - 7,5 TFLOPS of double precision
  - 15 TFLOPS of single precision
  - 30 TFLOPS of half-precision

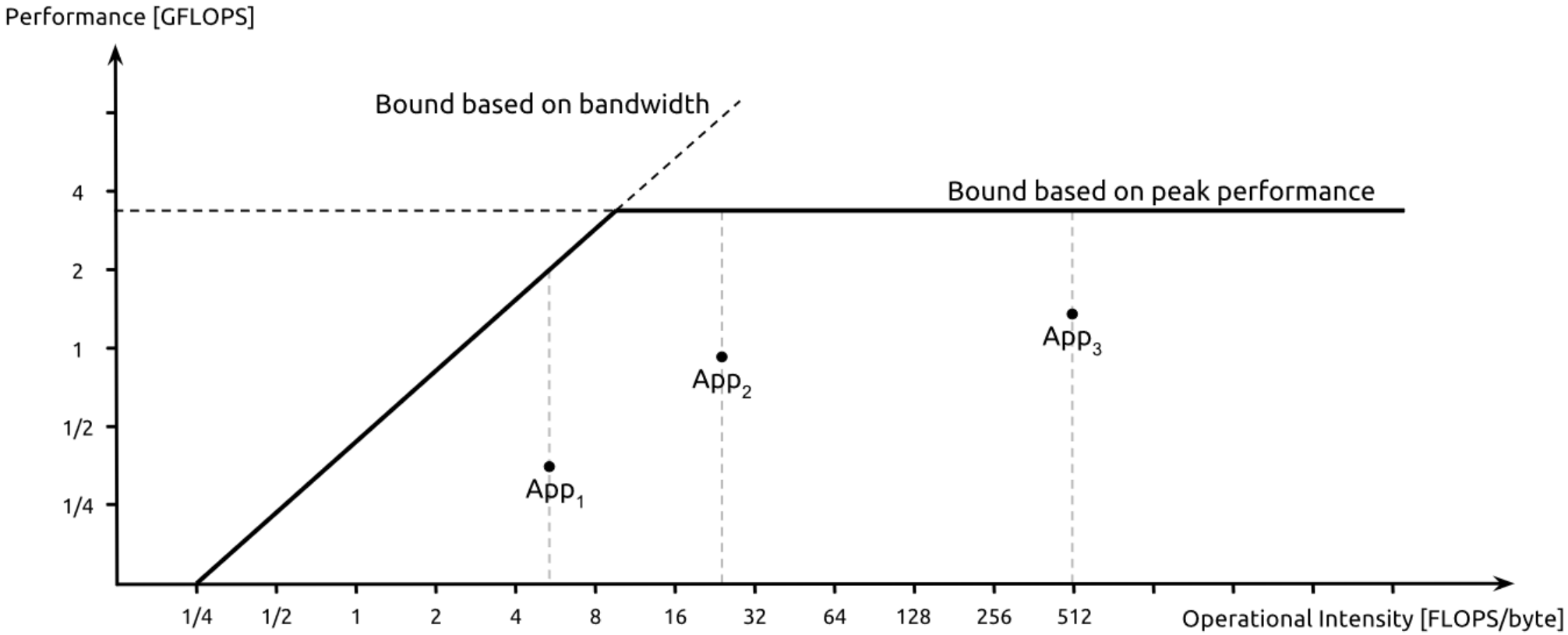
Nvidia Volta V100

- Volta SMM Specifications:
  - Up to 5376 CUDA cores
  - 14336 KB register file size/GPU
  - Up to 128 KB of shared memory

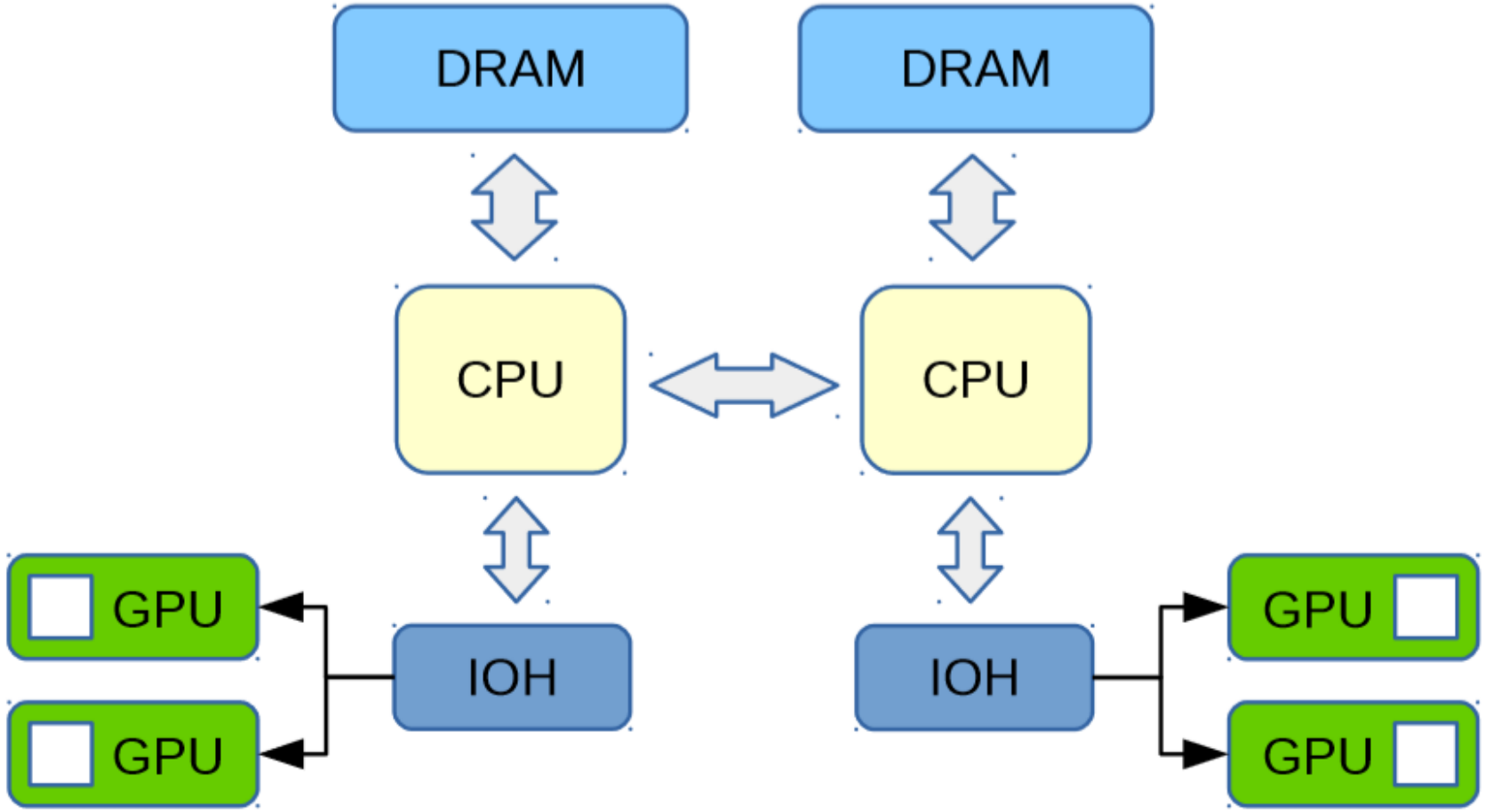
План

- Roofline модель оценки производительности.
- Распределенная архитектура памяти.
- Важность размещения данных в памяти.
- Схема современного гибридного кластера с множеством сокетов, ядер и ускорителей.
- Классификация Флинна.
- Общая память и распределенная память.
- Ошибки, специфичные для параллельных вычислений: deadlocks, memory races.
- Потоки и процессы.
- Синхронное и Асинхронное исполнение.
- Критические секции, мьютексы и барьеры.
- Сигналы, сообщения, события для коммуникации между процессами.
- Наборы утилит: cpuinfo, lspci. Управление задачами на сервере с помощью SLURM.

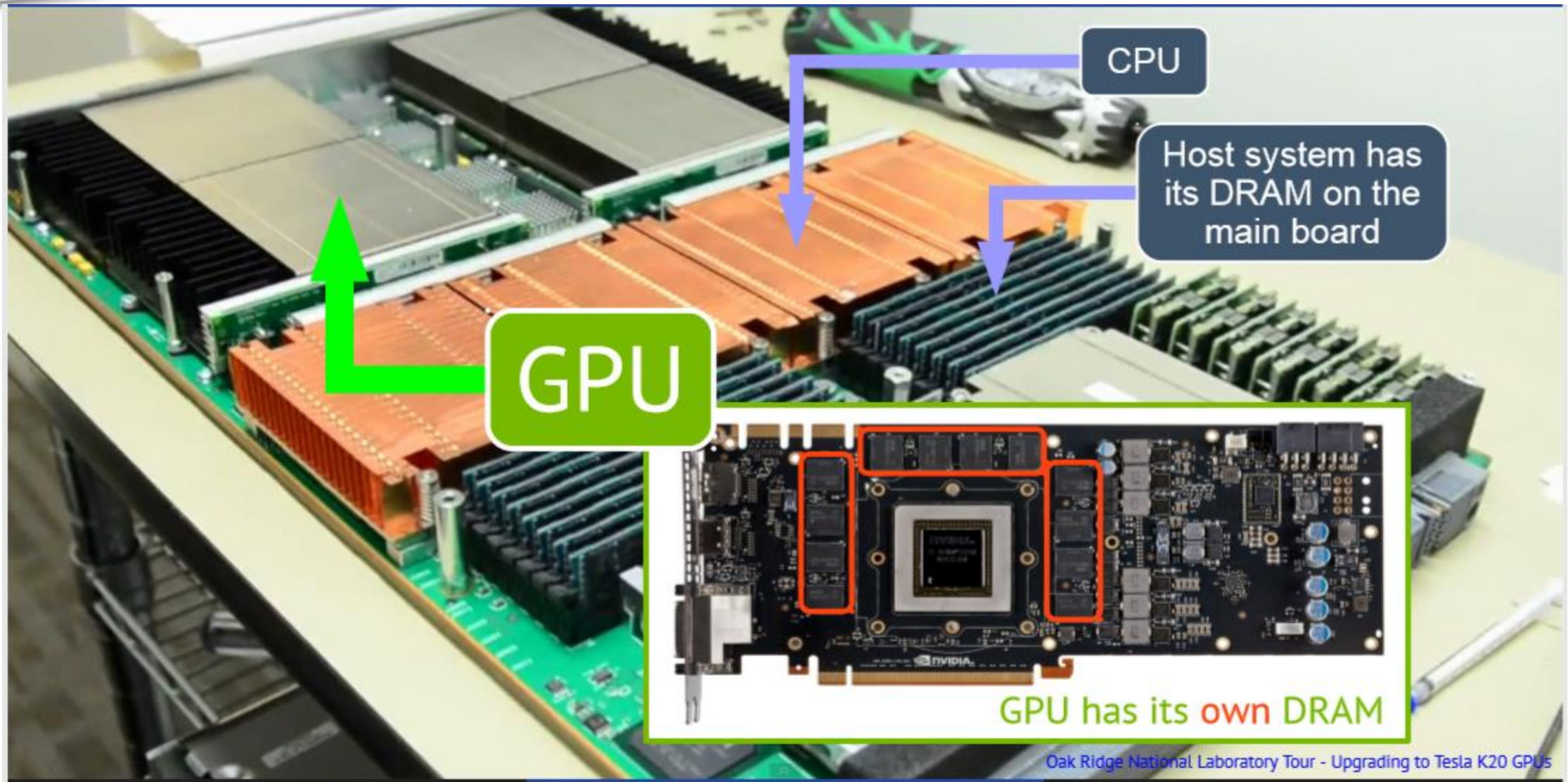
Roofline модель



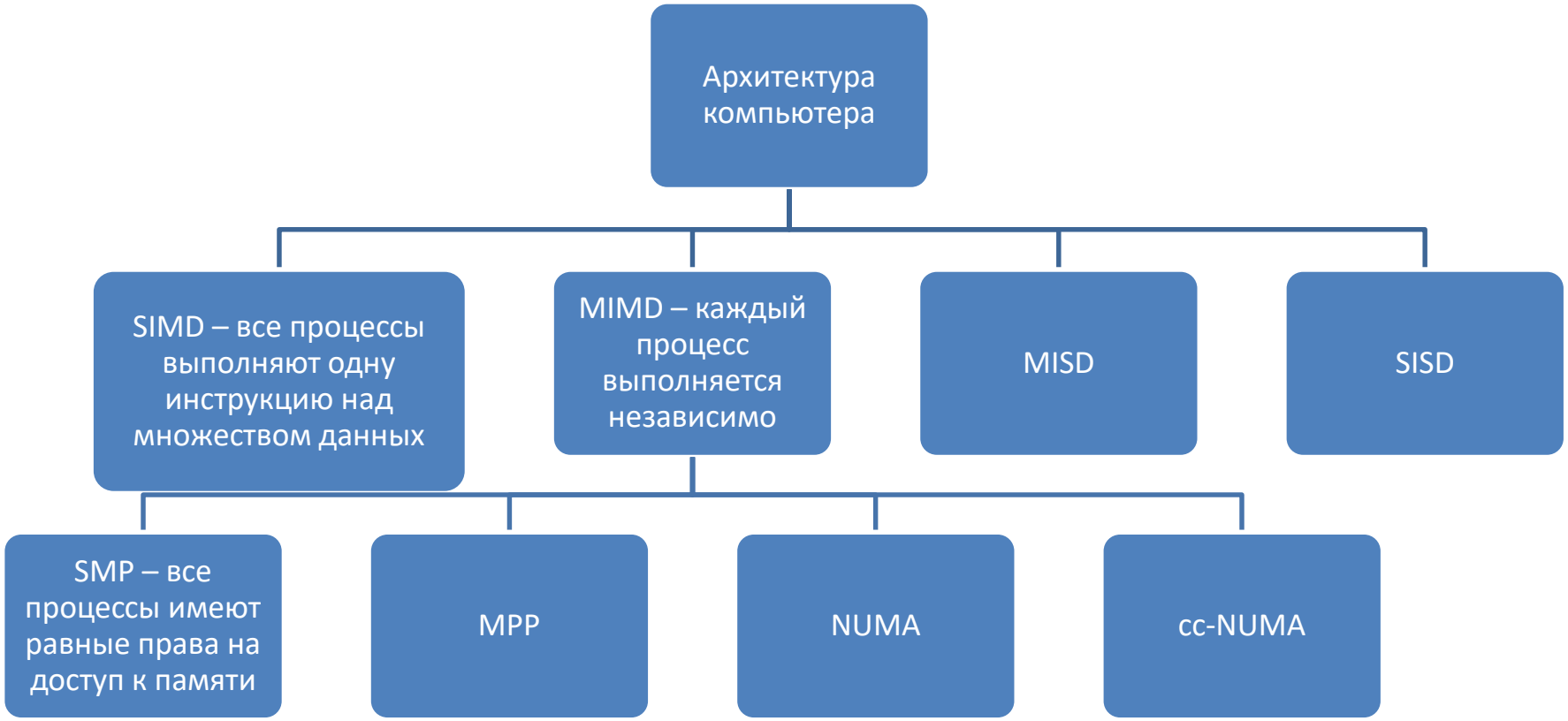
Современный кластер



Современный кластер

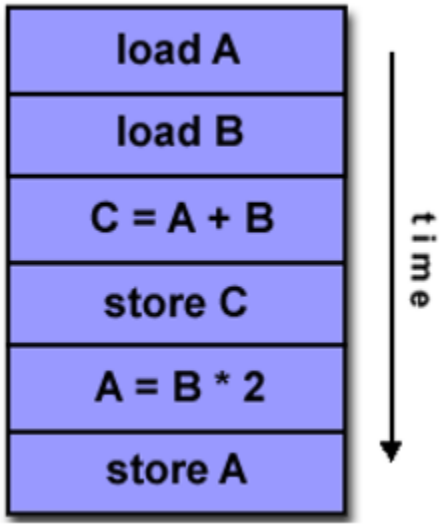


Классификация Флинна



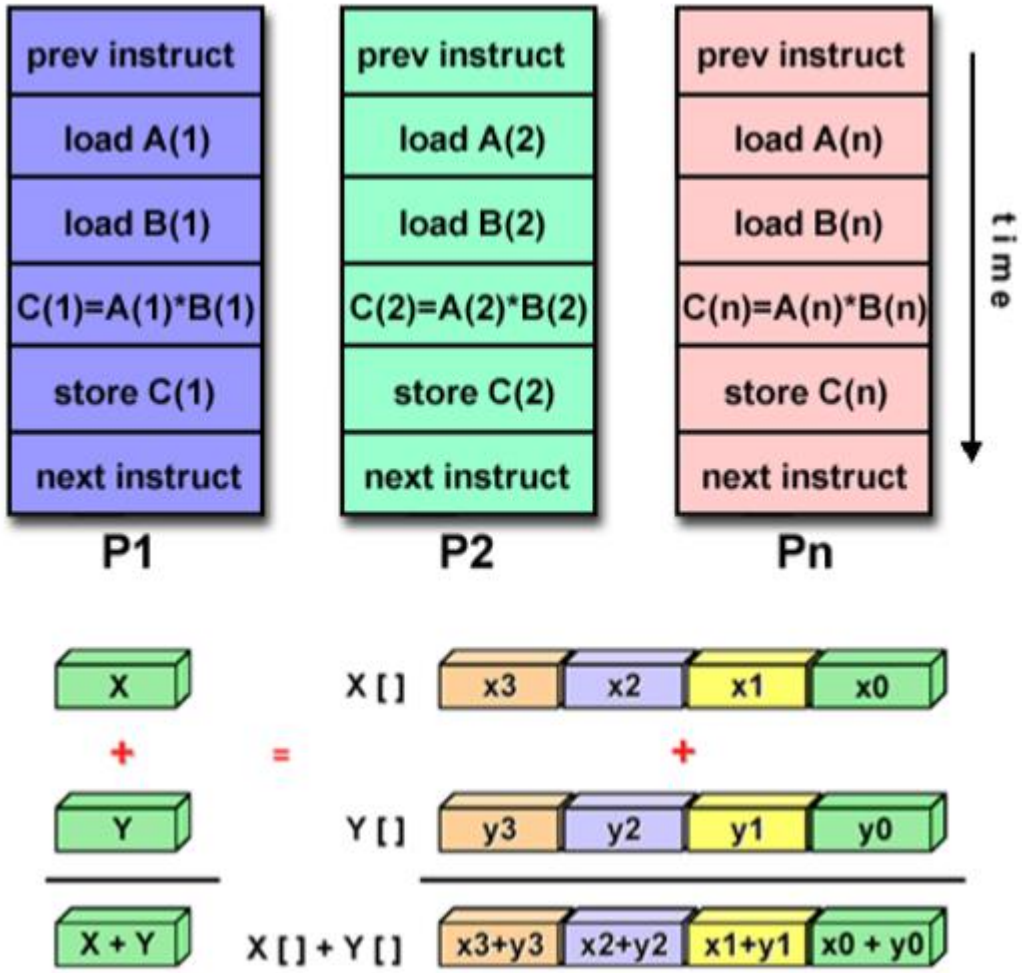
SISD

- Последовательный вычислитель
- Детерминированное исполнение



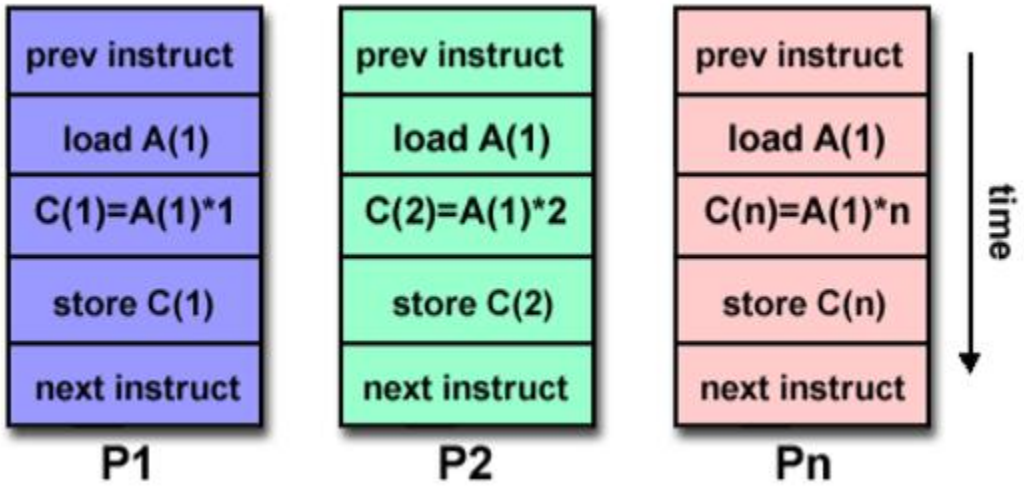
SIMD

- Все вычислители исполняют одну и ту же инструкцию синхронно по тактам
- Каждый вычислитель может оперировать отдельным данными
- 2 варианта:
  - Набор процессоров
  - Векторное исполнение
- Большинство современных компьютеров относятся к SIMD



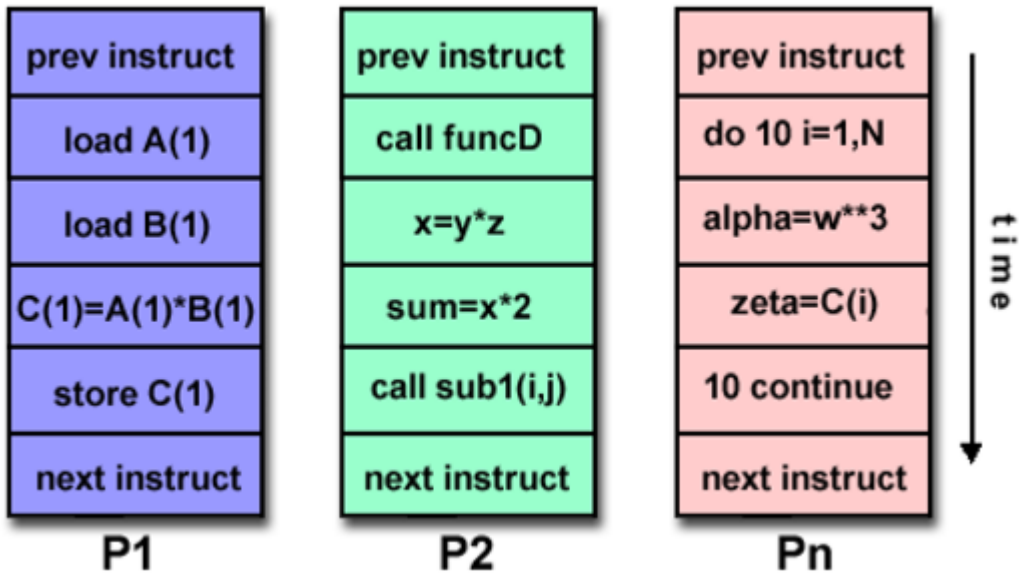
MISD

- Поток данных обрабатывается множеством процессоров
- Каждый юнит обрабатывает данные независимо с помощью собственного потока инструкций.
- Примеров не так много: Carnegie-Mellon C.mmp computer (1971).



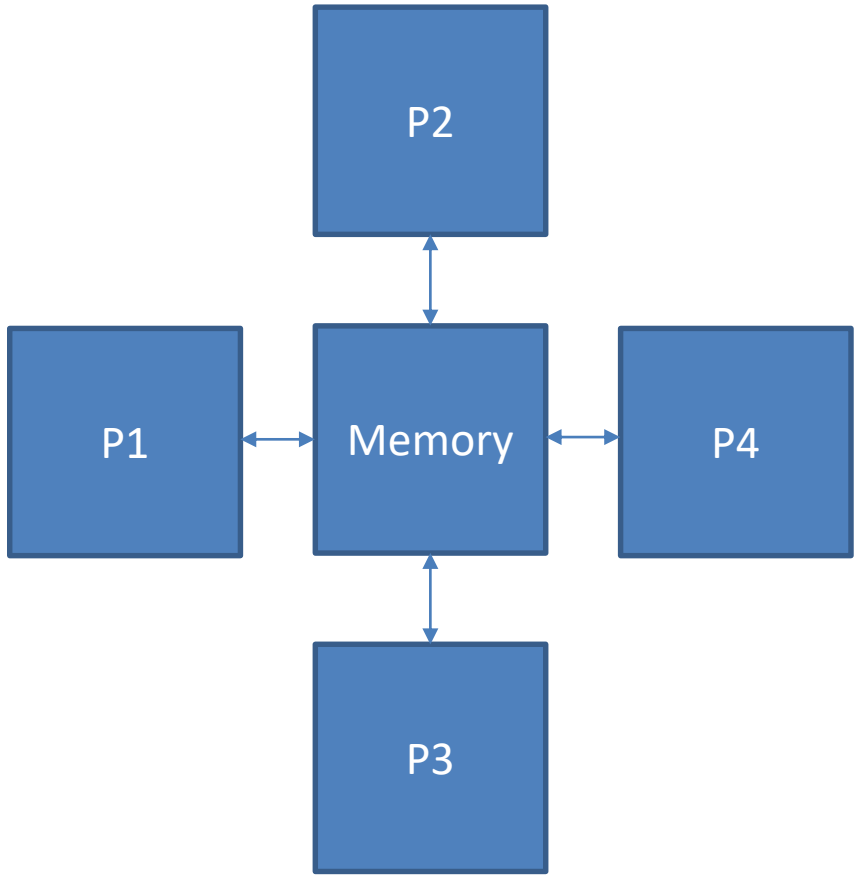
MIMD

- Самый распространенный тип параллельного компьютера
- Каждый процессор может исполнять свой собственный поток инструкций
- Исполнение может быть синхронным и асинхронным, детерминированным и недетерминированным
- Примеры:
  - Почти все современные суперкомпьютеры, многопроцессорные системы, многоядерные процессоры
- Многие MIMD архитектуры состоят из SIMD модулей



Общая память

- Все процессоры имеют доступ к памяти, как глобальному адресному пространству
- Процессоры работают независимо, но делят ресурсы памяти



Общая память

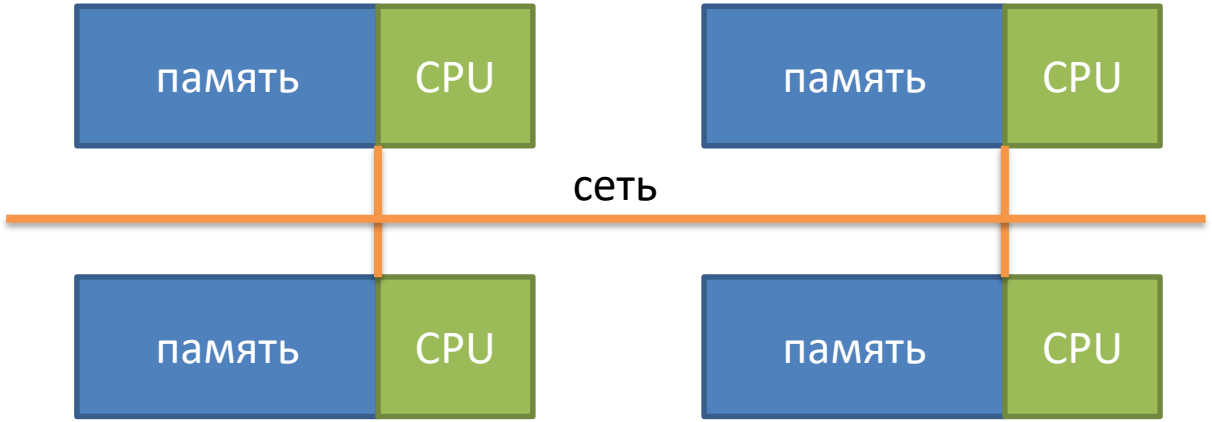
- Преимущества
  - Глобальное адресное пространство – упрощает программирование
  - Позволяет распараллеливать код по частям
  - Унифицирует и ускоряет обмен данными между CPU
- Недостатки
  - Память не масштабируется вместе с количеством процессоров
  - Накладные расходы на организацию общего доступа к памяти
  - Программист сам отвечает за синхронизацию доступов к памяти
  - Soaring expense of internal network.

Виды систем с общей памятью

- Uniform memory access (UMA)
  - Так же называется Symmetric Multi-Processors (SMP)
  - Процессоры равны между собой
  - Время доступа из всех Pn к памяти одинаково
  - Cache Coherent:
    - ✓ Если один процессор обновляет значения в памяти, они обновляются для всех процессоров.
- Non-Uniform memory access (NUMA)
  - Достигается с помощью соединения нескольких SMP
  - Один SMP имеет прямой доступ к памяти другого
  - Не все процессоры имеют одинаковое время доступа
  - Внутри одного SMP доступ к памяти быстрее
  - Медленный доступ к памяти по сети
  - Дополнительная работа для обеспечения когерентности (CC-NUMA)

Распределенная память

- Каждый процессор обладает своей отдельной памятью
  - Нет глобального адресного пространства
  - Для коммуникации между процессорами используется сеть

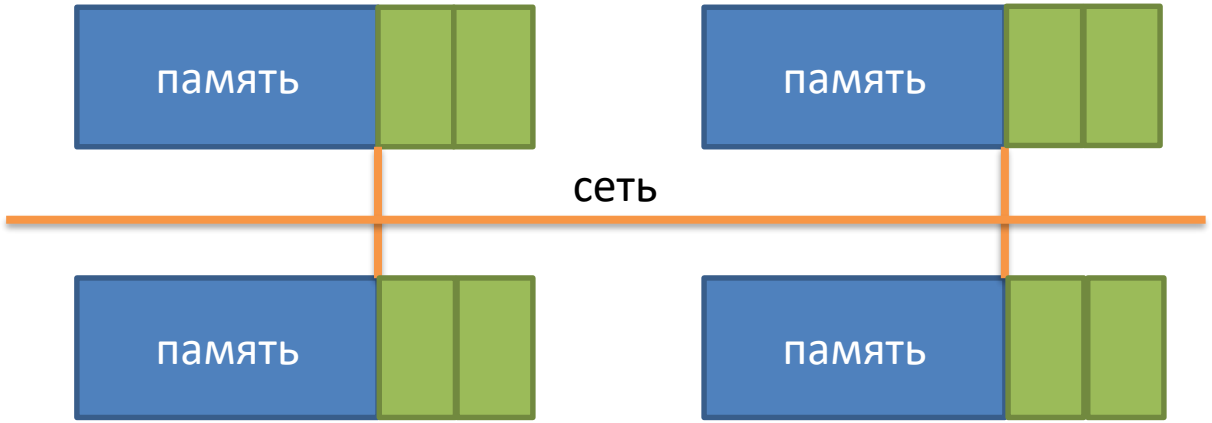


Распределенная память

- Преимущества
  - Размер памяти увеличивается с количеством процессоров
  - Быстрый доступ к памяти (кроме сети)
  - Cost effective (commodity components)
- Недостатки
  - Программист отвечает за коммуникацию
  - Преобразование структур данных, основанных на глобальной памяти в распределенную потребует дополнительных усилий
  - Неравномерное время доступа к памяти. Зависит от латентности, пропускной способности и загрузки сети
  - Сложность поэтапного распараллеливания

Гибридная модель

- Наиболее распространенная модель
- Компоненты общей памяти - CC-UMA SMP.
- Глобальные адресные пространства внутри каждого SMP



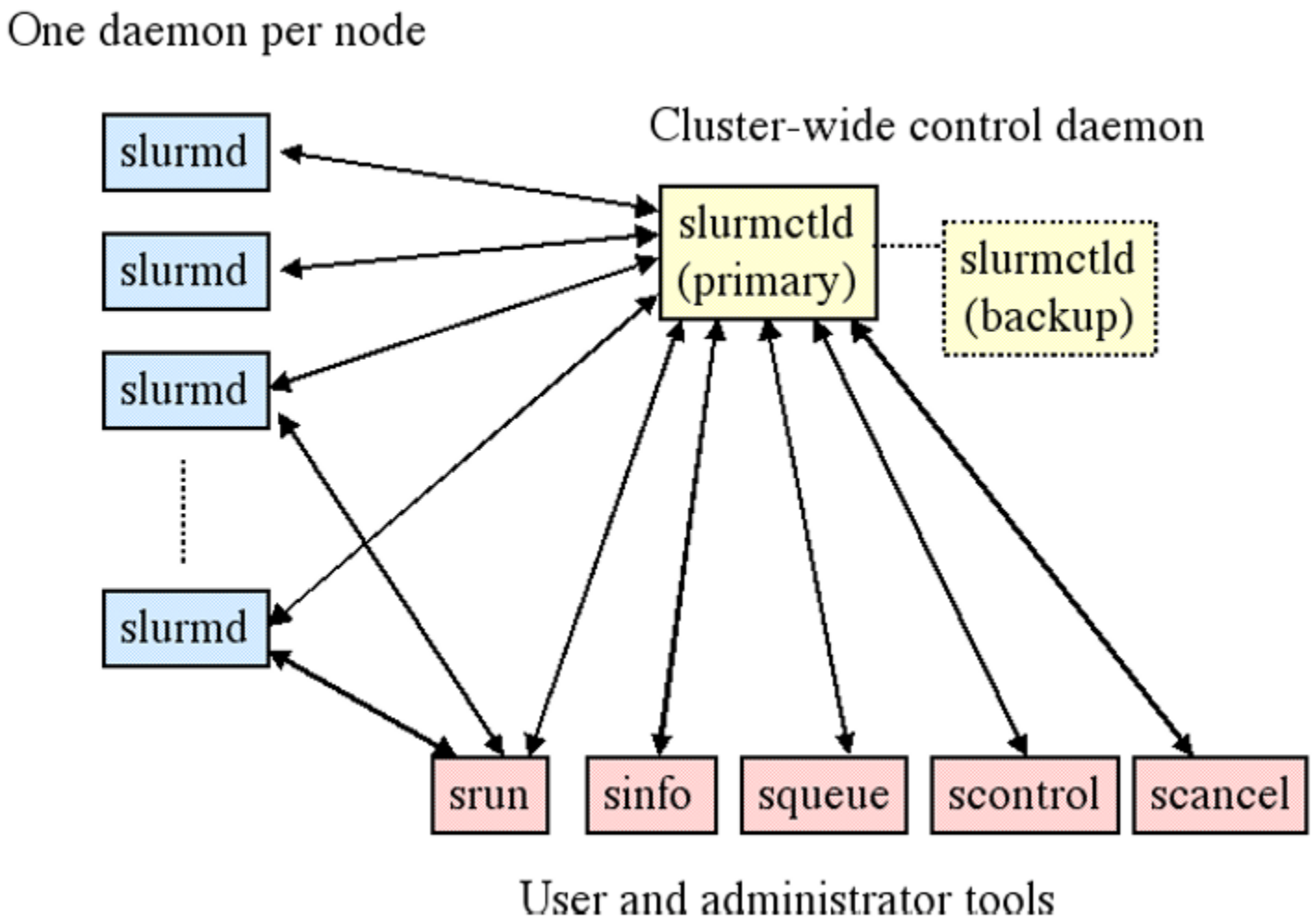
Модель общей памяти

- Параллельные задачи используют общее глобальное адресное пространство
- Операции чтения и записи могут происходить асинхронно
- Locks и semaphors для управления доступом к памяти
  - Следует избегать одновременного чтения и записи в одну и ту же ячейку памяти
  - Следует избегать нескольких одновременных операций записи в одну и ту же ячейку памяти
- Компилятор транслирует переменные в адреса глобальной памяти
- Пользователь определяет private и shared переменные
- Можно распараллеливать по частям

Синхронизация

- Барьер
  - Каждый поток выполняет свою работу пока не встретит барьер. После этого он останавливается (или «блокируется»)
  - Когда все потоки достигли барьера, они синхронизированы и продолжают работы с этой точки
- Lock / semaphore
  - Первый поток, занявший lock «отмечает» его. Этот поток может безопасно обращаться к защищенной памяти
  - Остальные потоки, пытающиеся отметить lock ожидают, когда lock будет освобожден
  - Может быть блокирующим или неблокирующим

SLURM


One daemon per node
Cluster-wide control daemon
slurmd
slurmd
slurmd
slurmd
slurmctld (primary)
slurmctld (backup)
srun
sinfo
squeue
scontrol
scancel
User and administrator tools

SLURM

- 2 daemons
  - slurmctld - controller, optional backup
  - slurmd - computer node daemon
- Команды
  - scontrol - administration tool, get/set configuration
  - sinfo - reports general system information
  - squeue - reports job and job step information
  - srun - submit/initiate job or job step
  - scancel - signal or cancel a job or job step