академия больших данных

mail.ru group

Высокопроизводительные вычисления.

Лекция 1. Введение

Рыкованов Сергей

PhD, доцент Сколтеха

Матвеев Сергей

к.ф-м. наук, доцент ВМК МГУ

Логистика и правила

- Лекции / семинары / лабораторные занятия по пятницам 19:00-22:00
- Общение происходит в discord
- Если хотите задать вопрос – нажимайте raise hand в Zoom
- Можете писать в личный чат если хотите обсудить материалы или задать вопрос – будет возможность открыть breakout room
- Создайте приватный репозиторий на github MADE2021 - домашние задания по программированию будут приниматься через github

Рекомендуемая литература:

- Sterling, Anderson, Brodowicz. High Performance Computing. Modern Systems and Practices. Morgan Kaufmann Publishers, 2018
- В.П. Гергель. Высокопроизводительные вычисления для многоядерных многопроцессорных систем. Издательство Нижегородского госуниверситета, 2010
- В.Л. Баденко. Высокопроизводительные вычисления. Учебное пособие. Издательство Политехнического университета, Санкт-Петербург. 2010
- High Performance Computing for Dummies.

Видео ресурсы:

- Coursera: Введение в параллельное программирование с использованием OpenMP и MPI
- Udacity: Introduction to High Performance Computing

Предполагаемые навыки и установки

- Знание Linux (Unix) и возможность использования Linux или Unix системы
- Пользователи Windows: www.virtualbox.org или подобный софт
- Установленный компилятор gcc (желательно последней версии)
- Установленный openmpi
- Установленный openmp
- **Мы предоставим виртуальный кластер, к которому вы сможете подключиться по ssh**
- Знание C/C++ (уверенное понимание указателей и массивов)

- https://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/
- https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

Лекторы

Сергей Рыкованов
PhD, доцент Сколтеха

2000-2006 МГУ им. Ломоносова, физический факультет
2006-2009 PhD в университете Людвига Максимилиана в Мюнхене (вычислительная физика плазмы)
2009-2012 научный сотрудник в университете Людвига Максимилиана в Мюнхене
2012-2014 научный сотрудник в лаборатории им. Лоуренса в Беркли
2014-2018 руководитель лаборатории в институте Гельмгольца в Йене (Германия)
2018-… Сколтех

serge.rykovanov@gmail.com
github: ssstuvz

Сергей Матвеев
к. ф-м. наук, доцент МГУ

2010 - 2015 МГУ им. М. В. Ломоносова , ВМК
2014 - техник НИВЦ МГУ
2015-2018 аспирантура, МГУ им. М. В. Ломоносова
2015 - 2018 младший научный сотрудник Сколтех
2019 - по н.в. научный сотрудник ИВМ РАН (совместитель)
2018 – 2021 научный сотрудник Сколтех
с февраля 2021 доцент ВМК МГУ

matseralex@gmail.com
github: matseralex

Цели курса

Не бояться огромных суперкомпьютеров.

Уметь писать параллельные программы на современных многоядерных ЦПУ и ГПУ

Научиться использовать большие машины коллективного пользования

Получать удовольствие от учебы

Программа курса

1. Введение в HPC
2. Терминал, переменные окружения, линковка, Makefile, виртуализация
3. Навыки работы на суперкомпьютере
4. multithreading, OpenMP и др
5. Библиотека MPI для работы на распределенных серверах
6. Multiprocessing и MPI в Python
7. Python code profiling и Numba
8. CUDA, CUDA Libraries examples, основные примитивы
9. CUDA in Python
10. Тренды в HPC, дополнительные главы (parallel I/O, визуализация)

Баллы за д/з

- Каждое д/з содержит базовую часть и бонусные баллы.
- Итоговые баллы д/з нормируются на стоимость базовой части и получаются проценты успешности очередного д/з.
- Курс не полагается сложным, но информацию о лучших студентах курса мы перешлем кураторам MADE
- Из-за бонусов за конкретные д/з можно набрать >100% (!)
- По итоговым процентам д/з считается среднее -- это финальный балл

Какие домашки? после следующих лекций:

1. **Вводная лекция по истории HPC**	**5-6. Библиотека MPI и MPI4Py**
2. Терминал, переменные окружения Makefile	**7, 9. Cuda/PyCuda**
3. Pthreads, openmp	**8. Профилировка и биндинг C/C++/Python ****
4. Навыки работы на суперкомпьютере *	** полностью бонусная

Сегодня: введение, история, общие сведения

Four paradigms of modern science

1. Experiment

2. Theory

$$i\hbar\frac{\partial}{\partial t}\Psi = \hat{H}\Psi$$

Limitations:

- experiments/theory too complicated
- experiments too expensive (car/airplane construction)
- experiments too slow (evolution of galaxies)
- experiments too dangerous (explosives, chemicals, climate)
- experiments not possible at the moment

Four paradigms of modern science

1. Experiment

2. Theory

$$i\hbar\frac{\partial}{\partial t}\Psi=\hat{H}\Psi$$

3. Modeling

Mathematical models

Numerical methods

Computers and supercomputers

Four paradigms of modern science

1. Experiment

2. Theory

$$i\hbar\frac{\partial}{\partial t}\Psi=\hat{H}\Psi$$

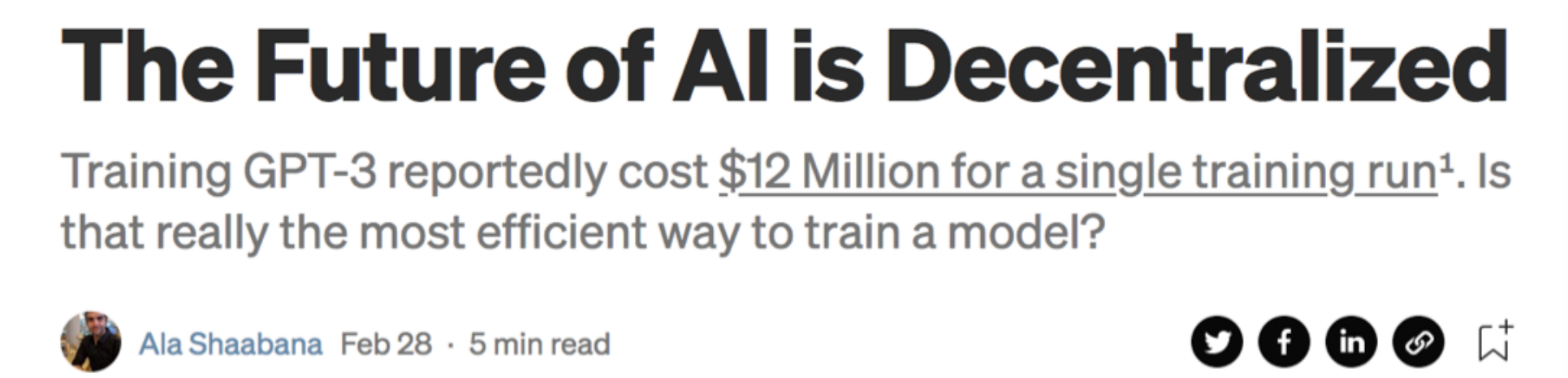
3. Modeling

4. Big Data and AI

High Performance Computing is a strategic necessity

- Main players like USA, China, European Union are investing **billions of dollars** into HPC
- HPC drives scientific discovery (plasma physics, computational chemistry, novel materials, artificial intelligence).
- HPC has entered almost all areas of human activity, for example:
  - new perfumes development
  - automobile tires development
  - entertainment industry, movies, cybersport
  - PayPal saved **~1 billion dollars** detecting online fraud with HPC
- HPC is one of the main drivers of the modern world innovation.

- **Airlines:**
  - System-wide logistics optimization systems on parallel systems.
  - Savings: approx. $100 million per airline per year.
- **Automotive design:**
  - Major automotive companies use large systems (500+ CPUs) for:
    - CAD-CAM, crash testing, structural integrity and aerodynamics.
    - One company has 500+ CPU parallel system.
  - Savings: approx. $1 billion per company per year.
- **Semiconductor industry:**
  - Semiconductor firms use large systems (500+ CPUs) for device electronics simulation and logic validation
  - Savings: approx. $1 billion per company per year.
- **Energy**
  - Computational modeling improved performance of current nuclear power plants, equivalent to building two new power plants.



Green AI

https://towardsdatascience.com/the-future-of-ai-is-decentralized-848d4931a29a
https://arxiv.org/pdf/1907.10597.pdf

Экспоненциальный рост данных и необходимость их обработки с помощью машинного обучения и искусственного интеллекта

- Люди с давних времен аккумулировали и каталогизировали информацию (в основном в текстовом формате): книги, справочники, энциклопедии, базы данных и т.п.
- В наше время произошел взрывной рост накопления аудио, видео и фото информации – такая информация существенно сложнее, ее структура плохо поддается анализу, в том числе из-за использования сжатия. Для того, чтобы ее обрабатывать необходимы качественно и количественно намного большие и специализированные вычислительные ресурсы, более сложные алгоритмы и подходы. Информацию уже окрестили «современной нефтью».
    - Тот, кто сможет эффективно обрабатывать большие объемы такой (плохо структурированной) информации, получит неоспоримое преимущество во многих областях науки и бизнеса.
        - В настоящее время между передовыми державами идет борьба за первенство в области высокопроизводительных вычислений для ИИ.

HPC share compared to the country GDP

24% GDP

38% HPC power

15% GDP

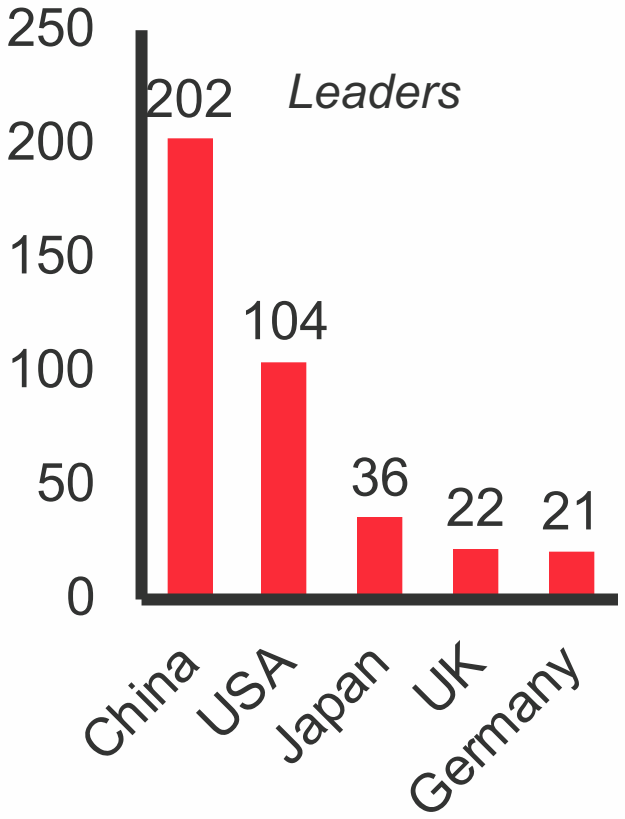
29% HPC power

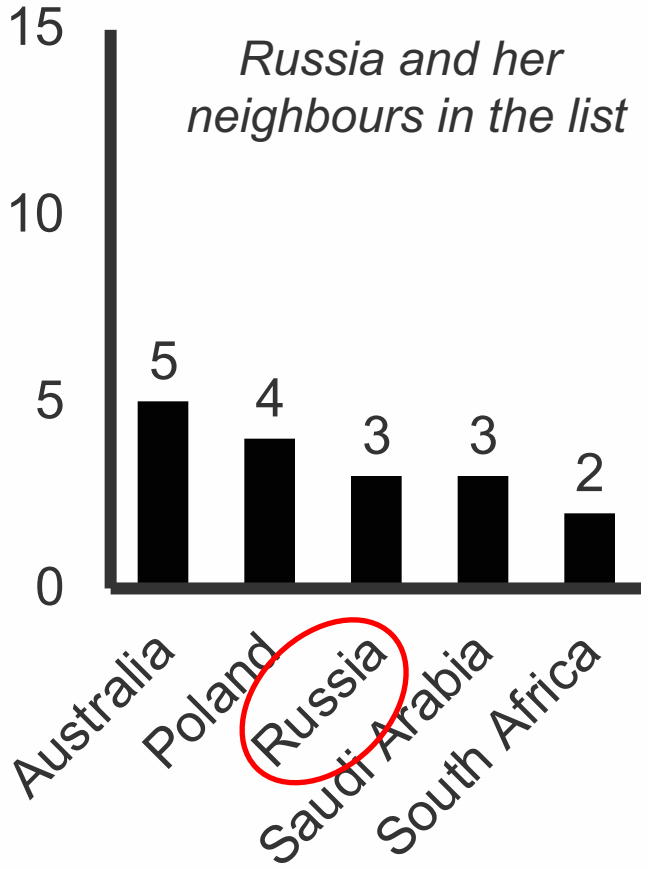
2% GDP

0.3% HPC power
(19th place, Poland – 18th)

Russia's position on the global HPC arena

Number of supercomputers in the world's top 500 list





Russia's position:

2009 year - 8

2010 year - **7**

2011 year - 9

2012 year - 9

2013 year - 13

2014 year - 9

2015 year - 9

2016 year - 10

2017 year -18

2018 year - **17**

* https://www.top500.org/ list of the most powerful supercomputers in the world

Units for measuring supercomputer performance

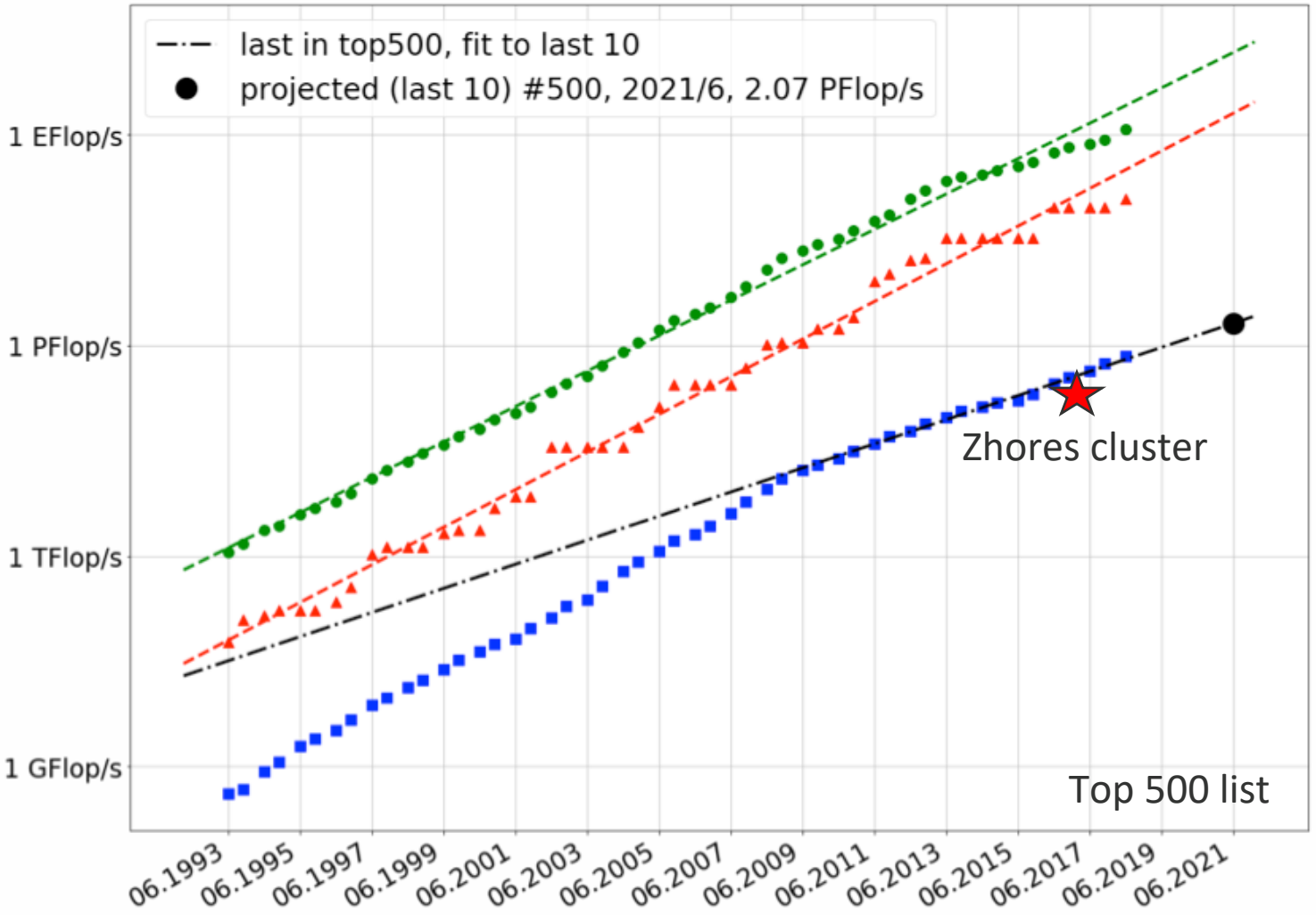
- High Performance Computing (HPC) units are:
  - Flop: floating point operation, usually double precision unless noted
  - Flop/s: floating point operations per second
  - Bytes: size of data (a double precision floating point number is 8 bytes)
- Typical sizes are millions, billions, trillions…

Kilo	Kflop/s = 10^3 flop/sec	Kbyte = $10^3 \sim 2^{10} = 1024$ bytes (KiB)
Mega	Mflop/s = 10^6 flop/sec	Mbyte = $10^6 \sim 2^{20}$ bytes (MiB)
Giga	Gflop/s = 10^9 flop/sec	Gbyte = $10^9 \sim 2^{30}$ bytes (GiB)
Tera	Tflop/s = 10^{12} flop/sec	Tbyte = $10^{12} \sim 2^{40}$ bytes (TiB)
Peta	Pflop/s = 10^{15} flop/sec	Pbyte = $10^{15} \sim 2^{50}$ bytes (PiB)
Exa	Eflop/s = 10^{18} flop/sec	Ebyte = $10^{18} \sim 2^{60}$ bytes (EiB)
Zetta	Zflop/s = 10^{21} flop/sec	Zbyte = $10^{21} \sim 2^{70}$ bytes (ZiB)
Yotta	Yflop/s = 10^{24} flop/sec	Ybyte = $10^{24} \sim 2^{80}$ bytes (YiB)

- Current fastest (public) machines are petaflop systems
  - Up-to-date list at www.top500.org

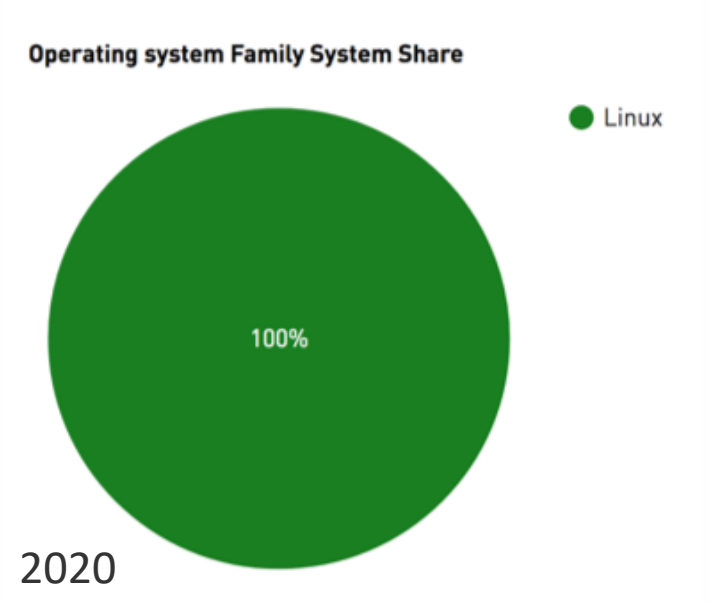
~ 10 Mflop/Watt

Top 500 – list of most powerful supercomputers

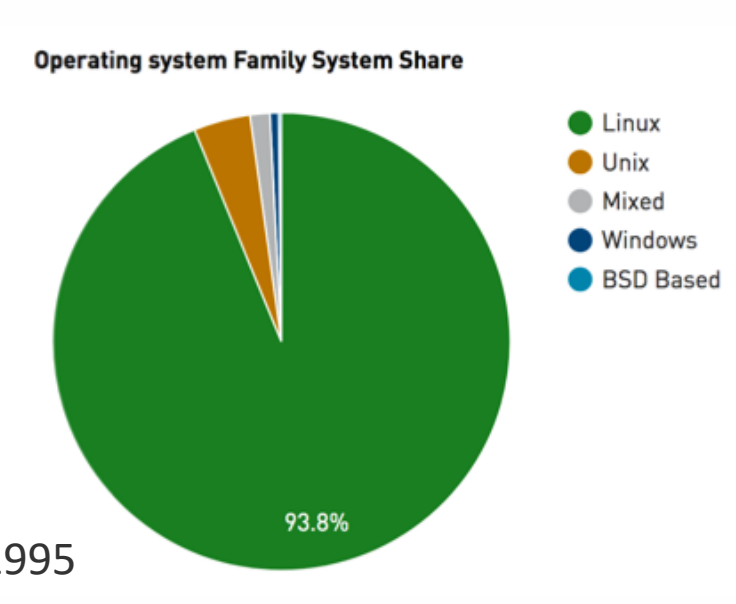


Currently only 3 Russian supercomputers are in the list

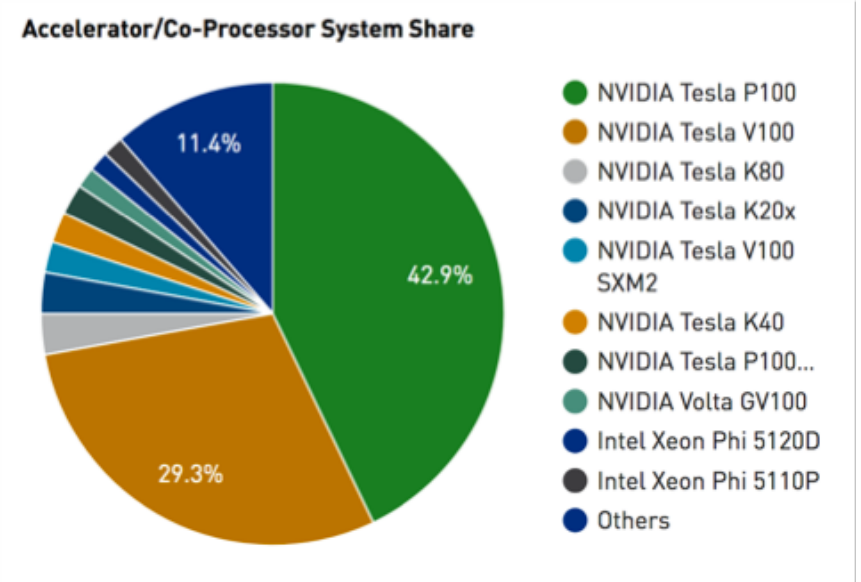
Top 500 – list of most powerful supercomputers

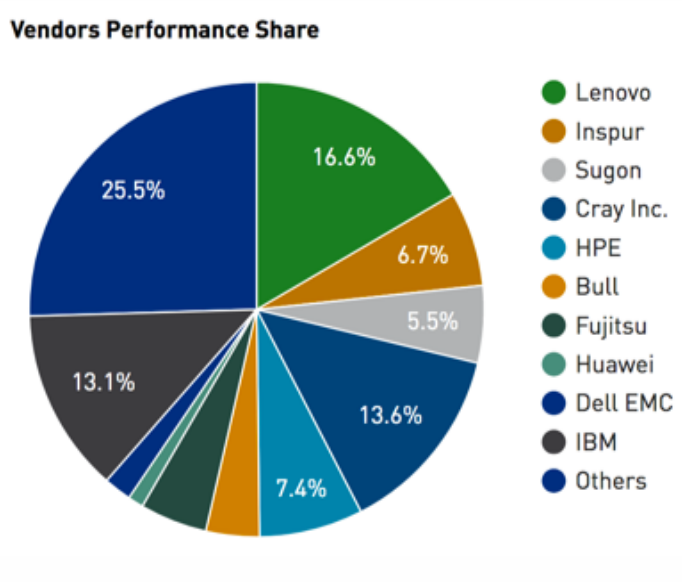


2020



1995





Summit (#2 machine) System Overview

System Performance

- Peak performance of 200 petaflops for modeling & simulation
- Peak of 3.3 ExaOps for data analytics and artificial intelligence

Each node has

- 2 IBM POWER9 processors
- 6 NVIDIA Tesla V100 GPUs
- 608 GB of fast memory
- 1.6 TB of NVMe memory

The system includes

- 4608 nodes
- Dual-rail Mellanox EDR InfiniBand network
- 250 PB IBM Spectrum Scale file system transferring data at 2.5 TB/s

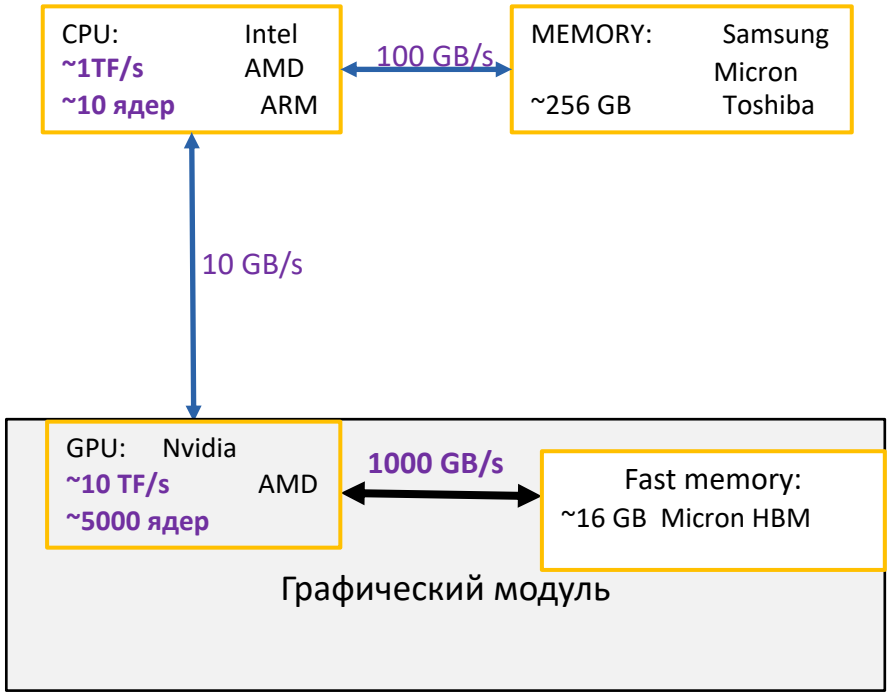
Russia in Top 500

3 entries found.

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
61	**Christofari** - NVIDIA DGX-2, Xeon Platinum 8168 24C 2.7GHz, Mellanox InfiniBand EDR, NVIDIA Tesla V100, Nvidia SberCloud Russia	99,600	6,669.0	8,789.8	
199	**Lomonosov 2** - T-Platform A-Class Cluster, Xeon E5-2697v3 14C 2.6GHz,Intel Xeon Gold 6126, Infiniband FDR, Nvidia K40m/P-100, T-Platforms Moscow State University - Research Computing Center Russia	64,384	2,478.0	4,946.8	
240	**MTS GROM** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100 40GB, Infiniband, Nvidia #CloudMTS Russia	19,840	2,258.0	3,011.8	

Functioning of a single compute node

- *Von Neumann Principle*
  - CPU loads data from memory, operates on it and puts the result back to memory: classical PC
  - Bottleneck is memory transfer CPU-memory which limits the computation speed



- *Specialized accelerator*
  - Applications need to be rewritten
  - Example: GPU computing (CUDA, OpenCL);
  - “Fast” memory inside the GPU;
  - Memory volume is bounded

    (V100: 16 ГБ)

  Memory hierarchy

Measuring the cluster performance

- Benchmarck for performance measurement (LINPACK): $Ax = b$

  where *A* is an *N x N matrix, N is large (10^6)*

  Performance: benchmark – R_{max}; theoretical or peak – R_{peak}

Системы на ключевых местах в списке ТОП-500

№	Cluster	Country	Rmax [Pflop/s]	Rpeak [Pflop/s]	Power [MW]
1	Summit - IBM	USA	122,3	187,6	8,8
2	Sunway TaihuLight	China	93	125,4	15,3
…	…	…			
500	CSCS Cray	Switzerland	0,7	0,84	0,3
…	…	…			
…	**Zhores - Dell**	**Russia, Skoltech**	**0,5**	**0,8**	**0,1**

Factors influencing the performance

- **Technology of semiconductors**
  - CMOS technology;
  - Decreasing the size of the semiconductor structures leads to better performance:

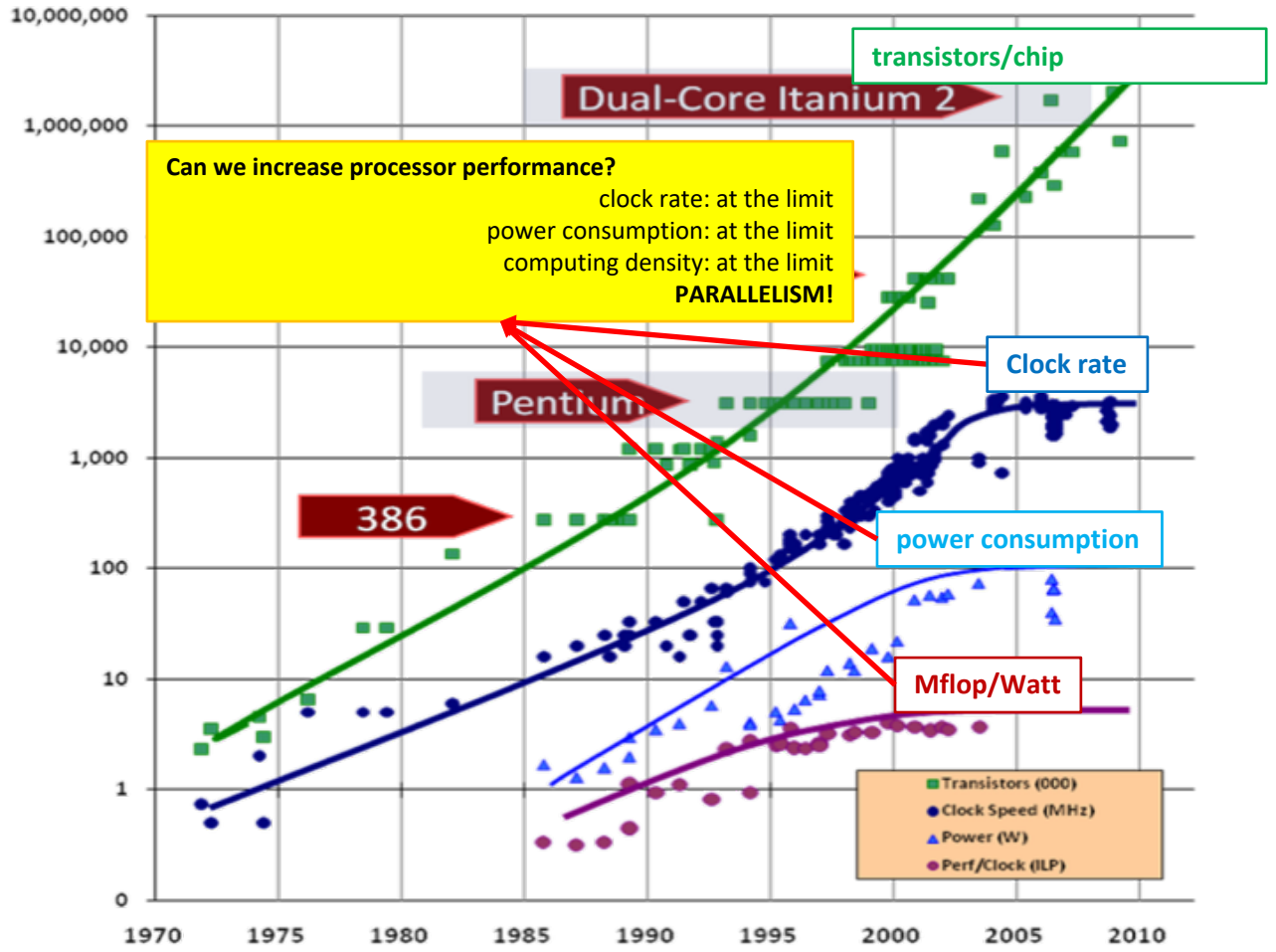
    size, energy consumption, clock rate, price

- **Compute node architecture**
  - Connection of CPUs and GPUs to memory, memory bandwidth, interconnect bandwidth

- **Infrastructure architecture**
  - Energy efficiency;
  - Cooling efficiency (influences the clock rate)
  - Computational density (less path for the signal to take)

Processor technology overview

Scale	Year Intro
130 nm	2001
90 nm	2004
65 nm	2006
45 nm	2008
32 nm	2010
22 nm	2012
14 nm	2014
10 nm	2017 (-19 Intel)
7 nm	2019 AMD (Roma)
5 nm	~*2020 (???)*



Why the ~~Fastest~~ *all* (since 2005) Computers are Parallel Computers

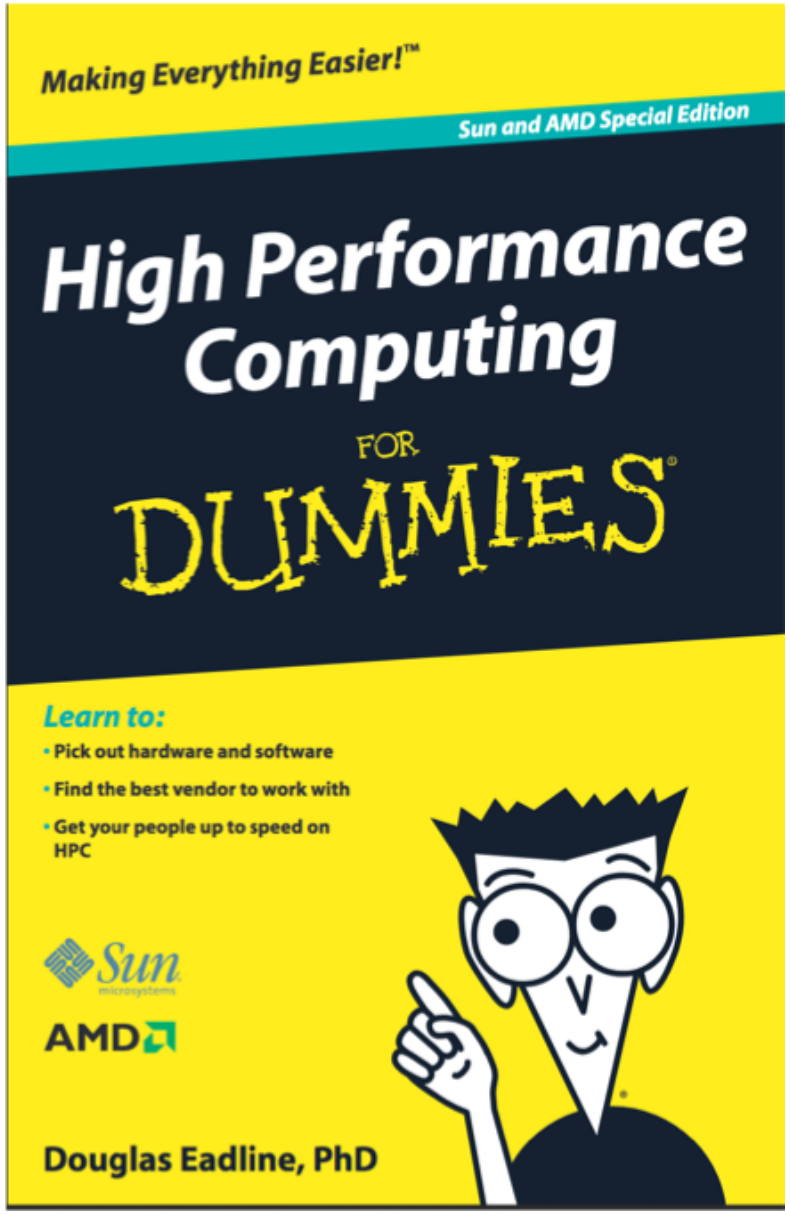
Including laptops and cell phones

Think parallel!

Moore’s Law revisited

- Number of cores per chip can double every 2 years
- Clock rate will not increase (probably it will decrease)
- Systems with millions of concurrent threads (GPUs)
- Inter-chip parallelism (MPI) or intra-chip parallelism (OpenMP)

What is a supercomputer and HPC?



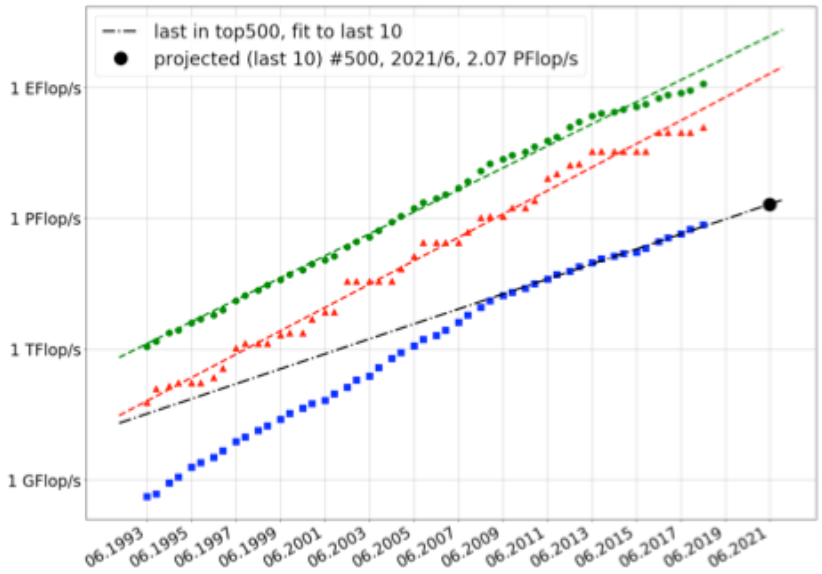
Highly recommended book
(google for pdf)

What is a supercomputer and HPC?

HPC is aggregation of computing power well beyond your typical laptop/desktop.

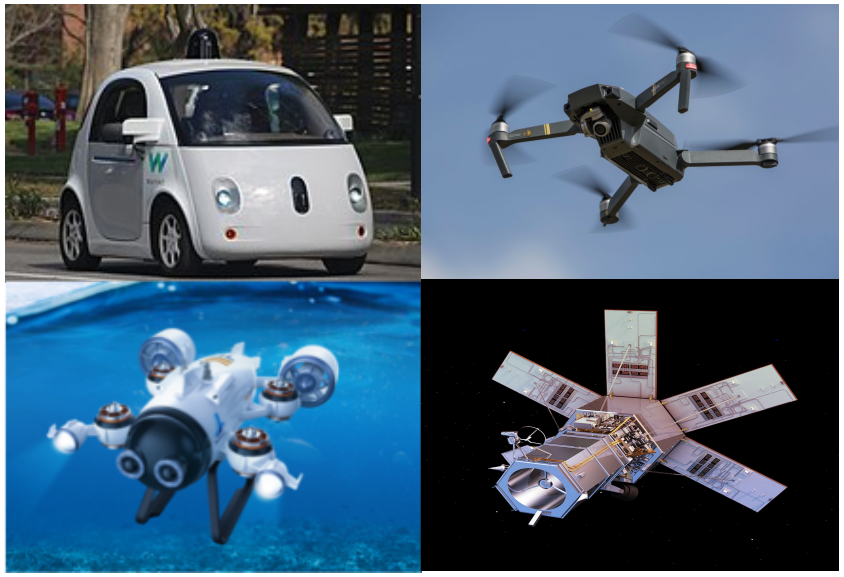
Definition is relativistic (relative in space-time)

Relative in time:



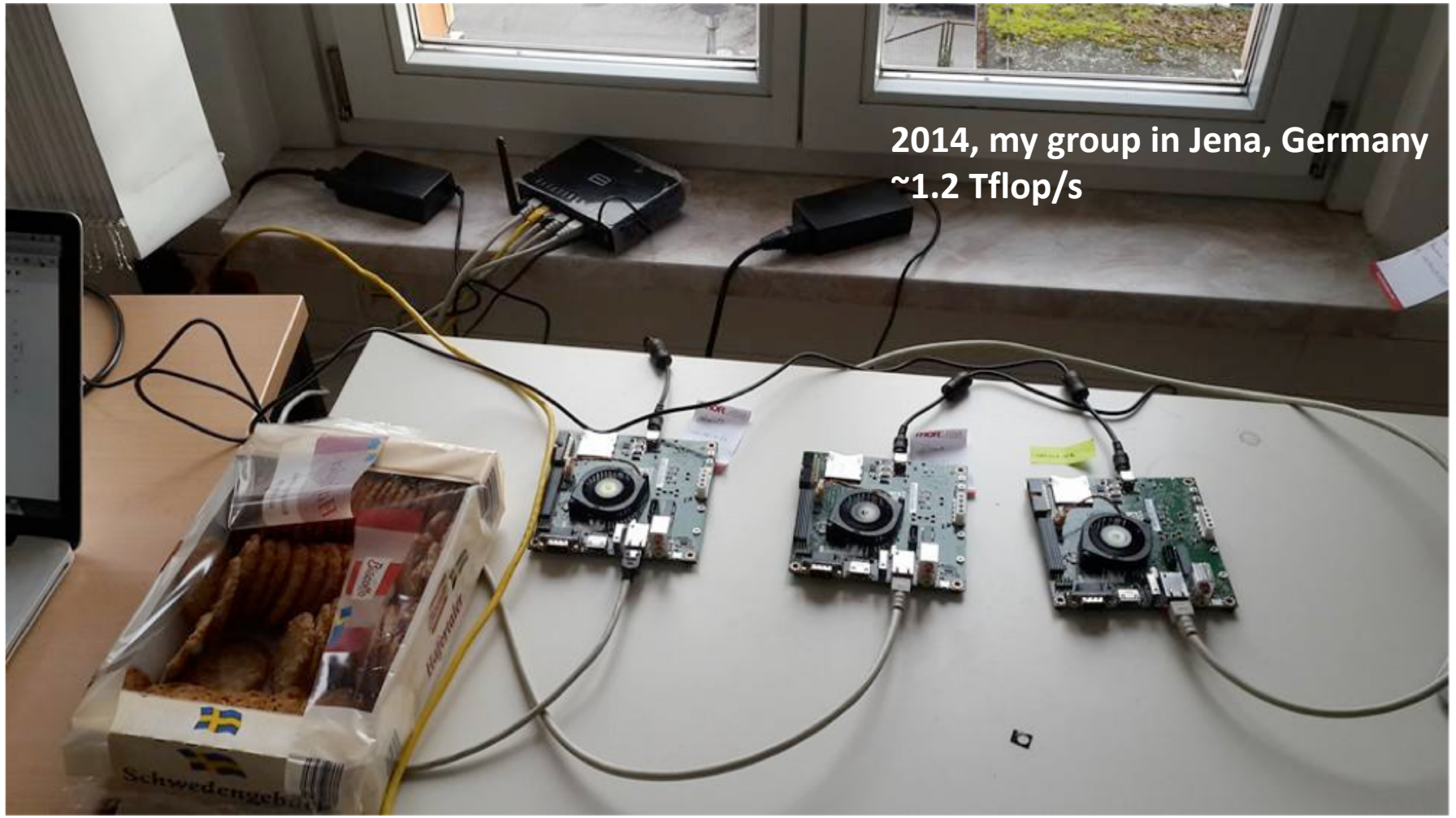
Your laptop was top 100 in the world in 1990s

Relative in space:



You can't put >1000kg computer on-board of autonomous vehicles

What is a supercomputer and HPC?



What is a supercomputer and HPC?

Visionary quotes about computers and HPC

Thomas Watson (chairman of IBM), 1943:

"I think there is a world market for maybe five computers."

Ken Olson (chairman of DEC), 1977:

"There is no reason for any individual to have a computer in his home."

Bill Gates, 1981:

"640K ought to be enough for anybody"

Popular mechanics 1949:

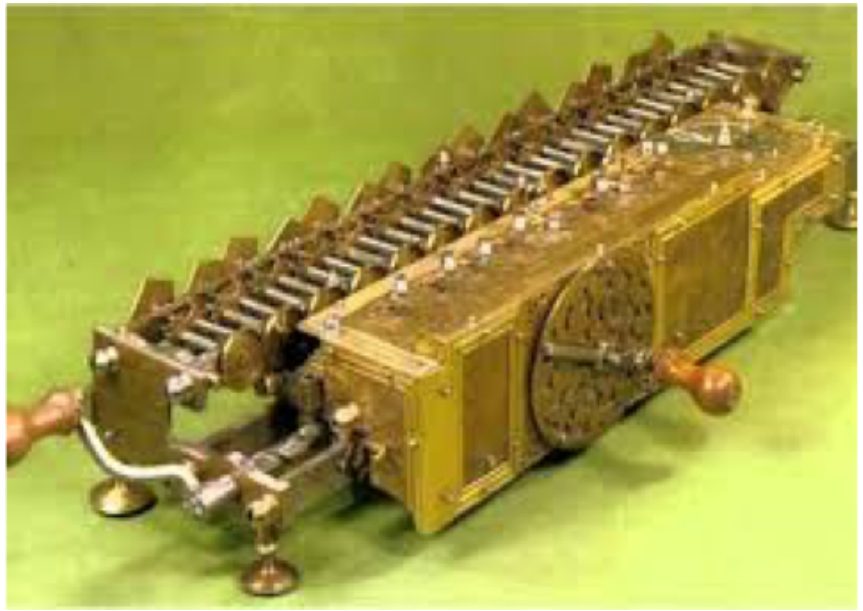
«Where a calculator on the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers in the future may have only 1,000 vacuum tubes and weigh only 1.5 tons.»

History of HPC

Abacus, 5 BC

17th century, Pascaline (1642)

Leibniz's Stepped reckoner (1671)



History of HPC

19th century

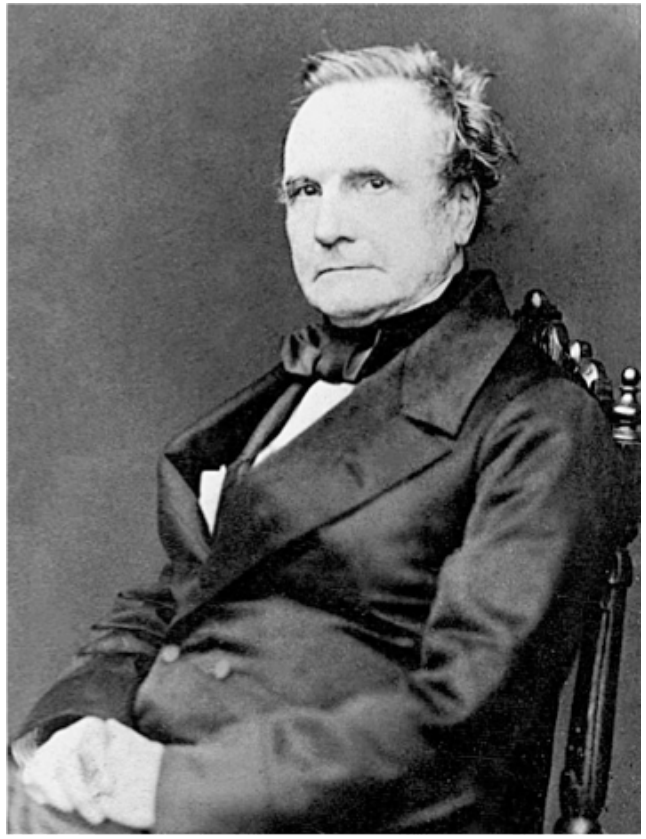
Arithmometer, Charles de Calmar (1820)

Punched card for weaving (1801)

Punched card + calculator, Herman Hollerith (1890)

History of HPC

19th century



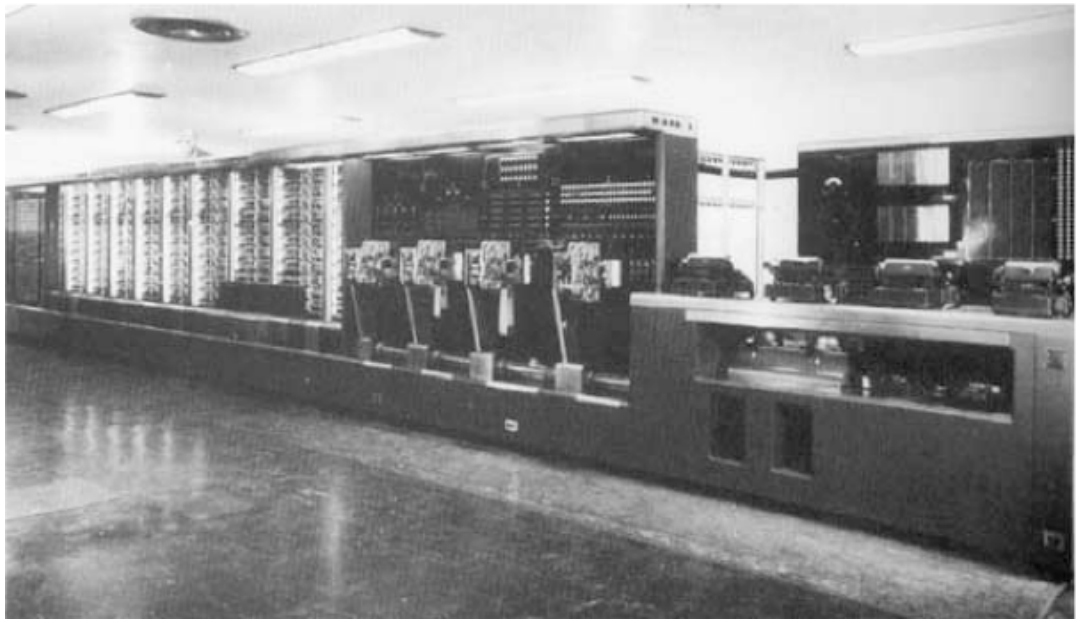
Charles Babbage, concept of fully automated calculation by mechanical means

Ada Lovelace, first programmer, first algorithm

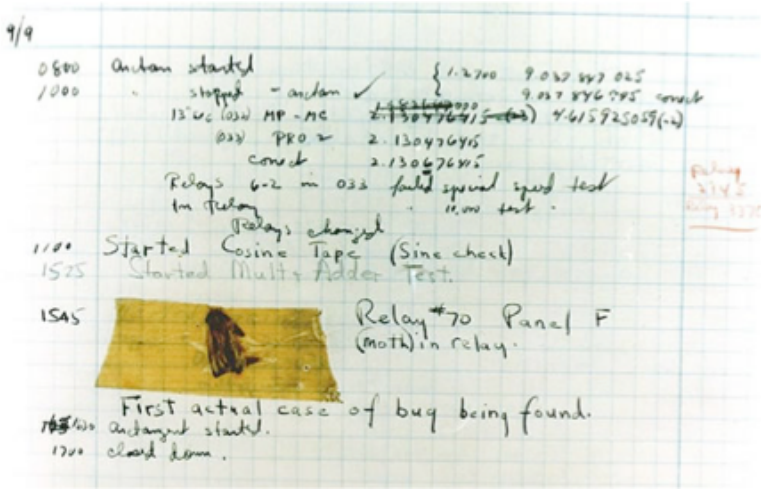
Punched card for weaving (1801)

History of HPC

Konrad Zuse 1938, Z1 – first programmable mechanical computer

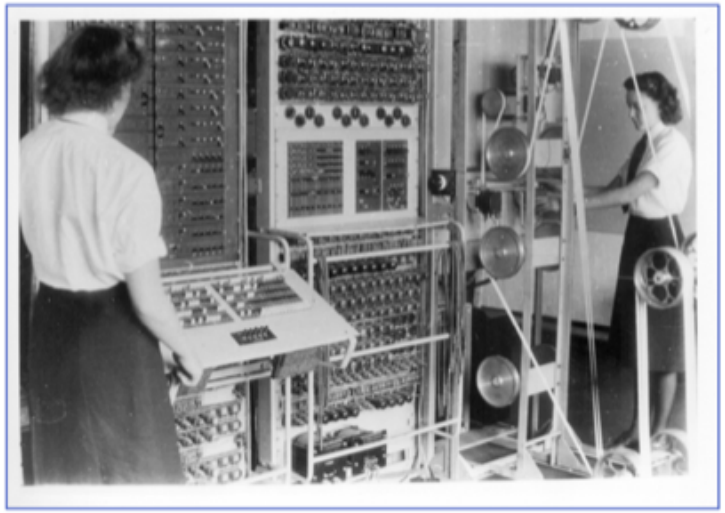


Harvard Mark I

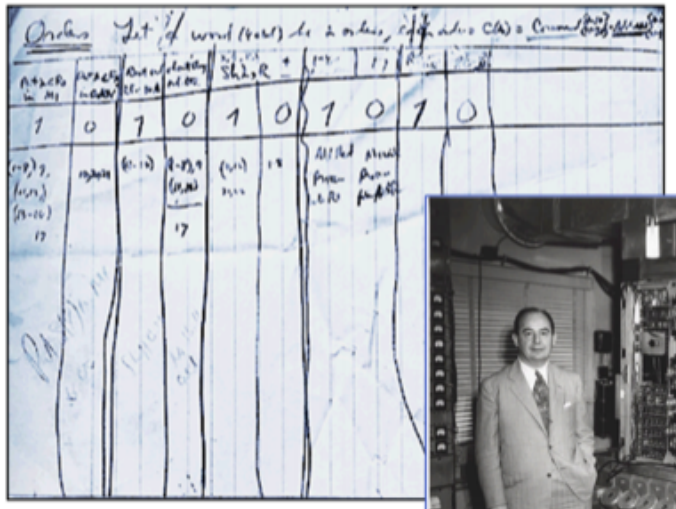


History of HPC

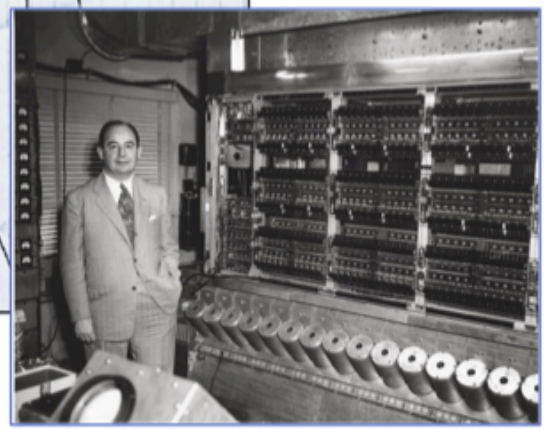
- **Electronic computing developed to meet military needs in WWII**
  - Colossus, Bletchley Park, 1943 … code breaking, dedicated function
  - ENIAC, U. Pennsylvania, 1945 … ballistics tables, plugboard programming
  - Von Neumann
    - Initiated 1946 at Institute for Advanced Studies (IAS)
    - Vacuum tubes, oscilloscopes, assembly language … many operational challenges
    - But momentous – flexible stored program, reliability architecture, hydrogen bomb
    - IAS, Princeton, 1951
    - MANIAC, LANL, 1952
    - ORACLE, ORNL, 1953



Women's Royal Naval Service operating Colossus during World War II



First line of code written for the von Neumann Digital Computing Project

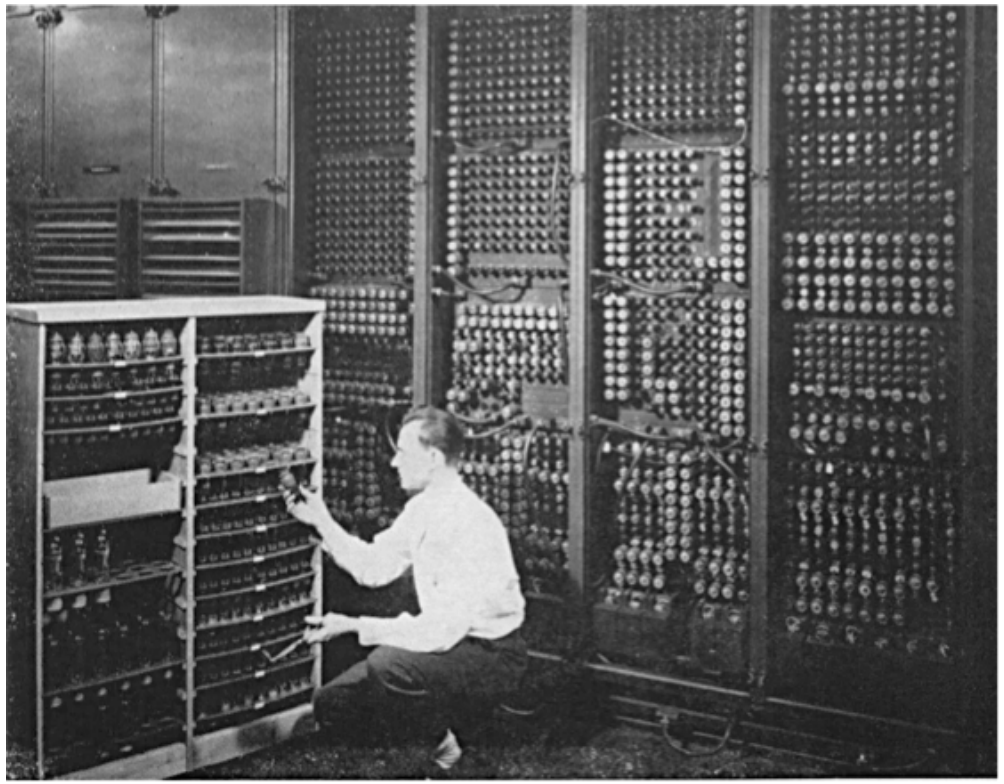


von Neumann with the IAS machine

History of HPC

Cray-1, 1976, specialized supercomputer
133 MFlops

Высокопроизводительные системы СССР



1948 МЭСМ, Лебедев

- 1948, МЭСМ, Лебедев
- с 1952 серия БЭСМ
- БЭСМ 6 выпуск до 1987 года,
  в работе до 1995 года
- серия «Эльбрус» в разработке до 90х
- 1968 – «гибель» отечественного компьютеростроение – решение о создании серии «РЯД» – клонирование IBM (ЕС)
- 50е-60е проекты сети ВЦ (**ЕГСВЦ) –> ОГАС** (аналог ARPANET)
- Инновационные проекты: Сетунь (тернарная логика)

https://www.ferra.ru/review/computers/ussr-pc-evolution.htm

https://www.the-village.ru/city/modern-architecture/171409-kak-v-sssr-stroili-doma-dlya-kompyuterov

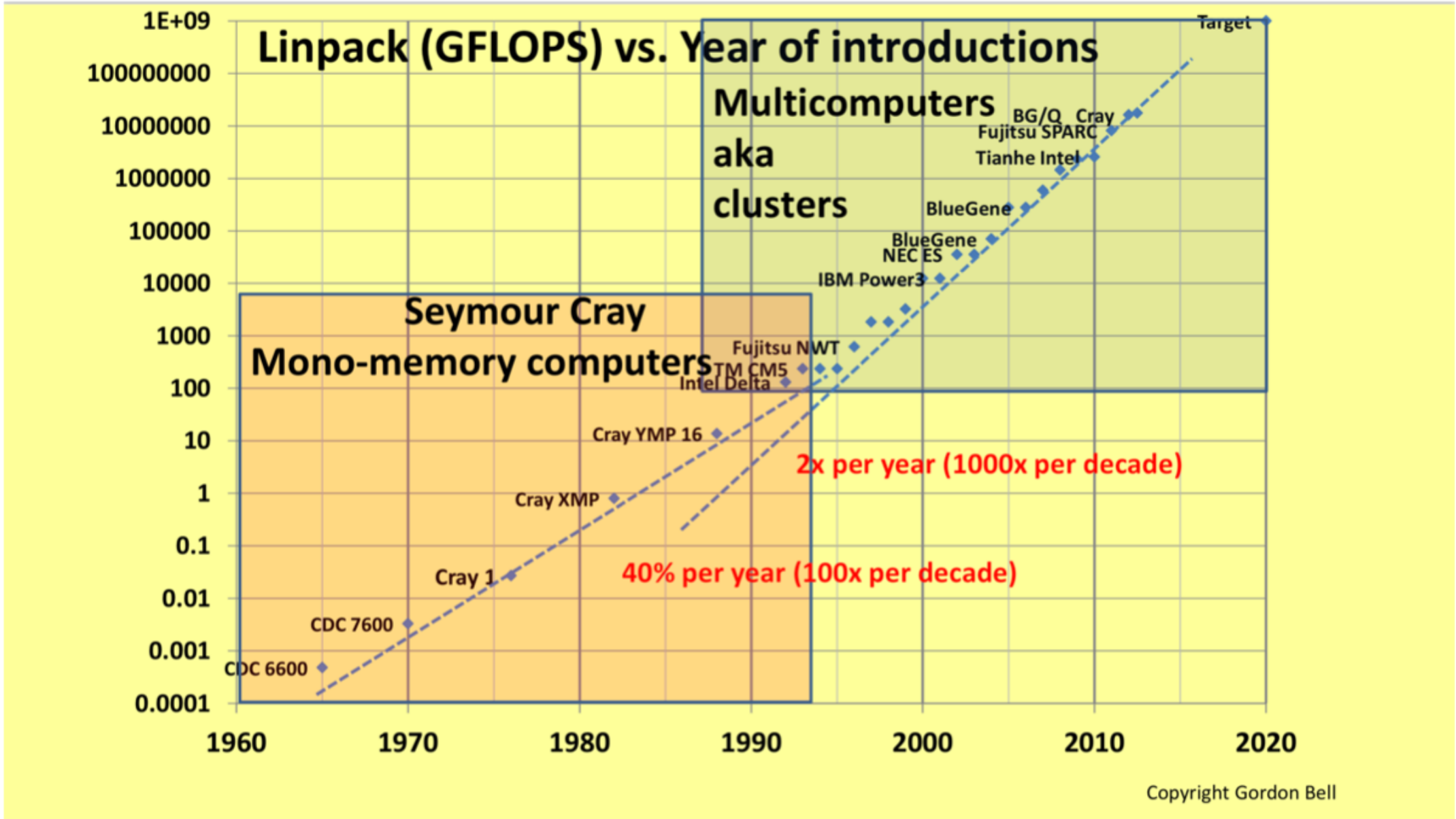
Советские дома для суперкомпьютеров

Вычислительный центр ГосПлана СССР (1970) – проспект ак. Сахарова 12

History of HPC

Scalable „Beowulf“ cluster made of similar compute nodes interconnected between each other.

1994, NASA (Thomas Sterling)


Linpack (GFLOPS) vs. Year of introductions
Multicomputers
aka
clusters
Seymour Cray
Mono-memory computers
2x per year (1000x per decade)
40% per year (100x per decade)
Target
BG/Q
Cray
Fujitsu SPARC
Tianhe Intel
BlueGene
BlueGene
NEC ES
IBM Power3
Fujitsu NWT
TM CM5
Intel Delta
Cray YMP 16
Cray XMP
Cray 1
CDC 7600
CDC 6600
1E+09
100000000
10000000
1000000
100000
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
1960
1970
1980
1990
2000
2010
2020
Copyright Gordon Bell

Flagship supercomputer “Zhores” for AI, Big data and HPC

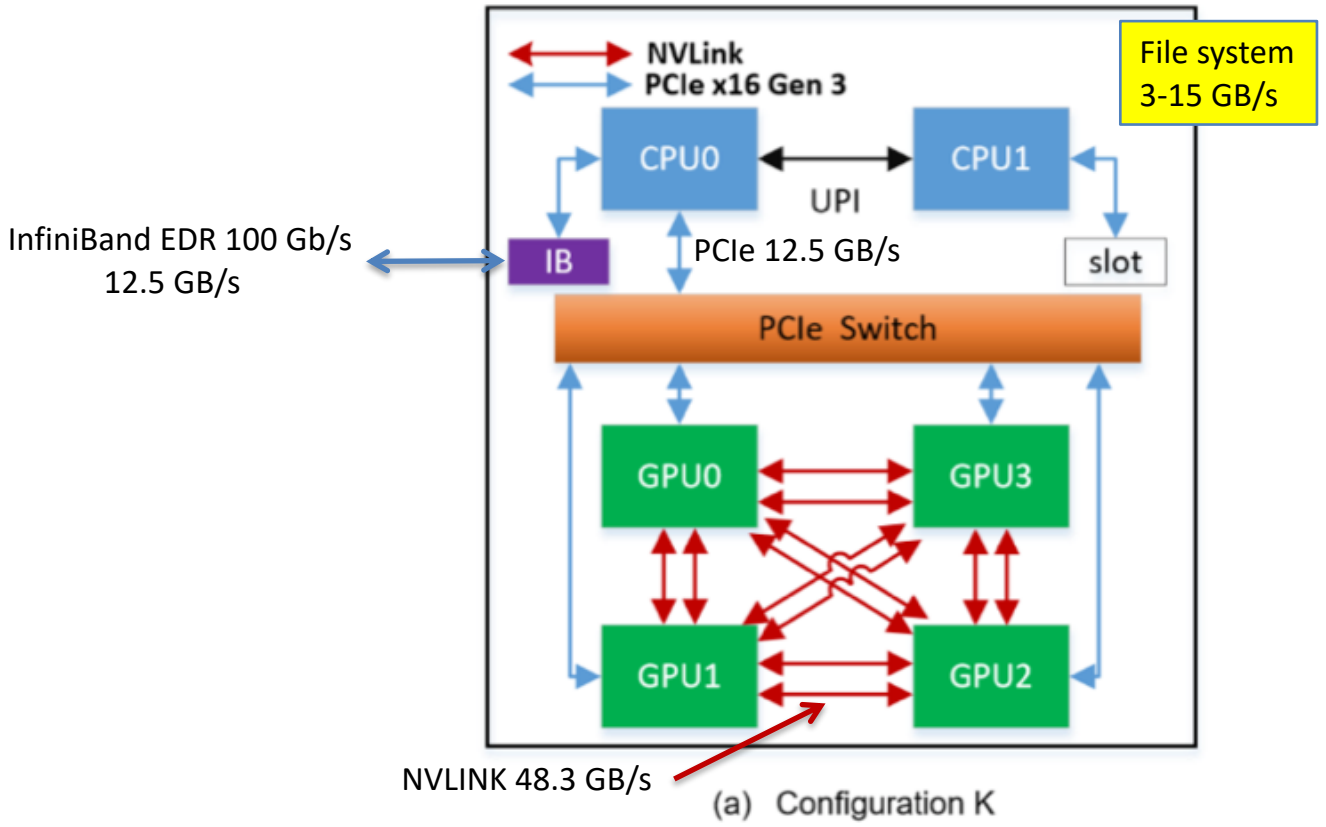
Hybrid energy-efficient architecture

- 74 compute nodes
- 26 nodes with powerful graphic cards Nvidia Tesla V100 (NVLink + RDMA)
- tensor cores for deep learning;
- 90 kWatt power consumption;
- 1PFlops peak performance;
- 0.5 Pbytes storage system
- #8 in Russia
- was installed by our own small team

«Zhores» is a unique for Russia supercomputer capable of solving a wide range of interdisciplinnary problems in machine learning, data science and mathematical modeling in such areas as: biomedicine, image classification, Digital Pharma, Photonics, predictive maintenance, new X- and gamma-ray sources

ZHORES
SUPERCOMPUTER
HIGH PERFORMANCE COMPUTING AND BIG DATA
SKOLTECH CDISE

Main power is in the GPU nodes



Platform	PowerEdge C4140 (configuration K and M)
CPU	2xCPU Intel 6140 (18c, 2.3 GHz)
Memory	384 GB DDR4 @ 2666MHz
GPU	NVidia Tesla V100 SXM2 (16 Gb @ 900 GB/s)

Parallel intuition

- Examples of parallelism in real world?

Parallel intuition
How to really understand concurrency.

Building a house

Say, we have 4 workers building a house

Construction will be faster than if it would be done by 1 worker

After each floor, workers have to synchronize

Connect their parts - communication

Can‘t build 7th floor before 3rd

Amdahl‘s law

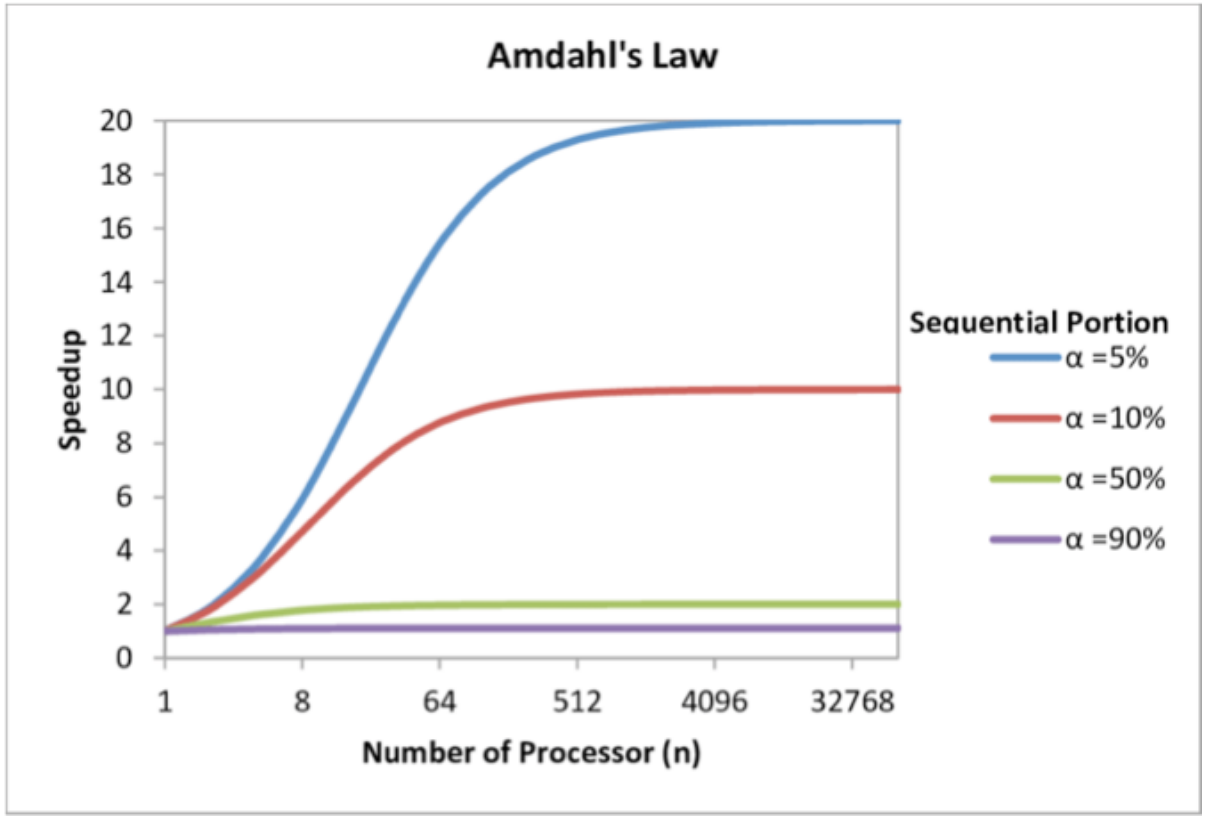
Speed-up of a parallel program/work:

$$Speedup = \frac{1}{(1-p) + p/N}$$

p – is a fraction of the work that can be done in parallel

N – number of parallel workers

Amdahl‘s law implication



$$Speedup = \frac{1}{(1-p) + p/N}$$

N	p=0.5	p=0.9	p=0.99
10	1.82	5.26	9.17
100	1.98	9.17	50.25
1000	1.99	9.91	90.99
10000	1.99	9.91	99.02
100000	1.99	9.99	99.9

For 90x speedup on 100 processors sequential part can only be 0.1%

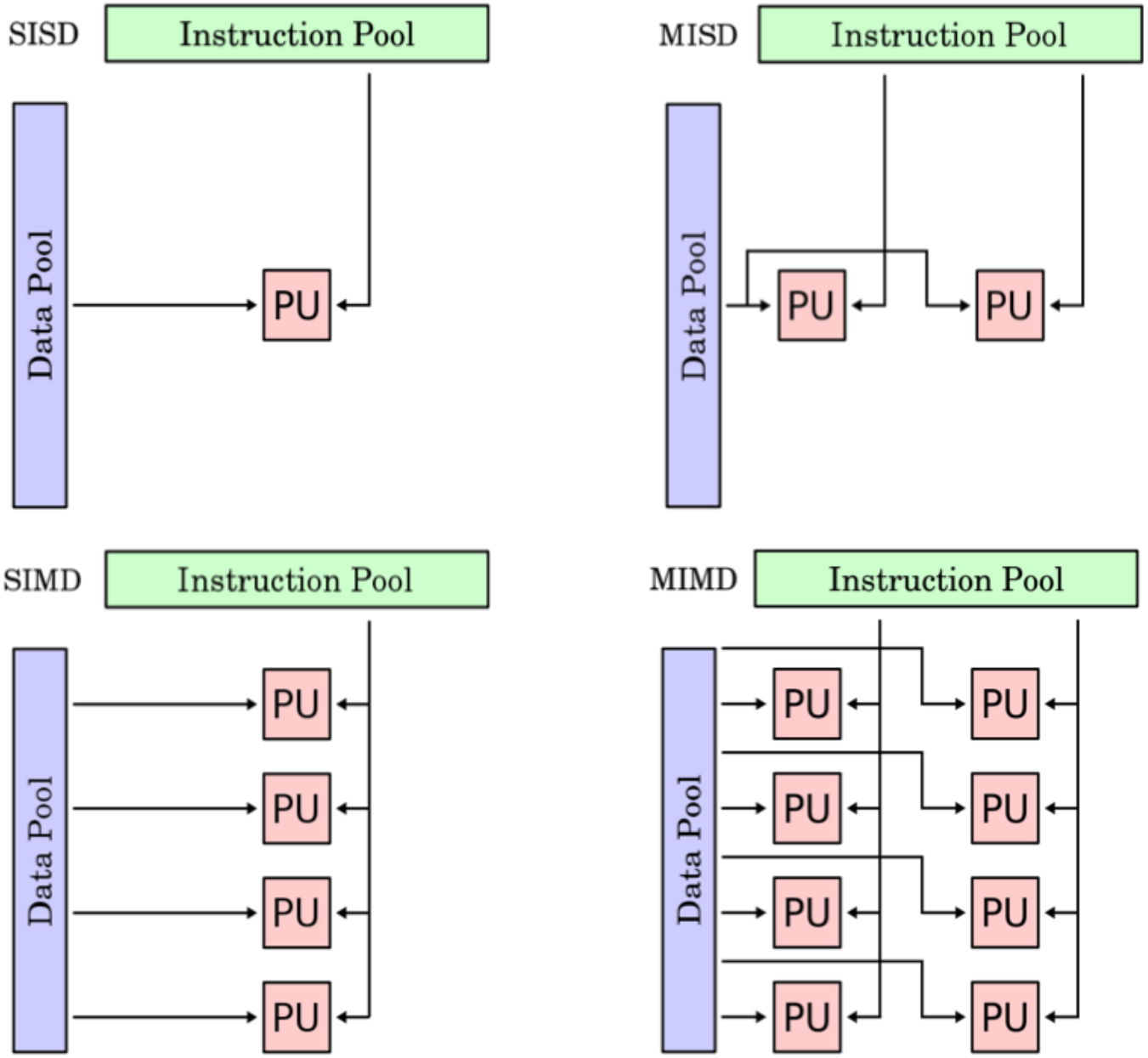
2 PITFALLS

- Expecting x N performance on N workers for a poorly parallelized code
- Expecting low energy usage on low resource utilization: servers draw up to 2/3 of power while running on 10% load

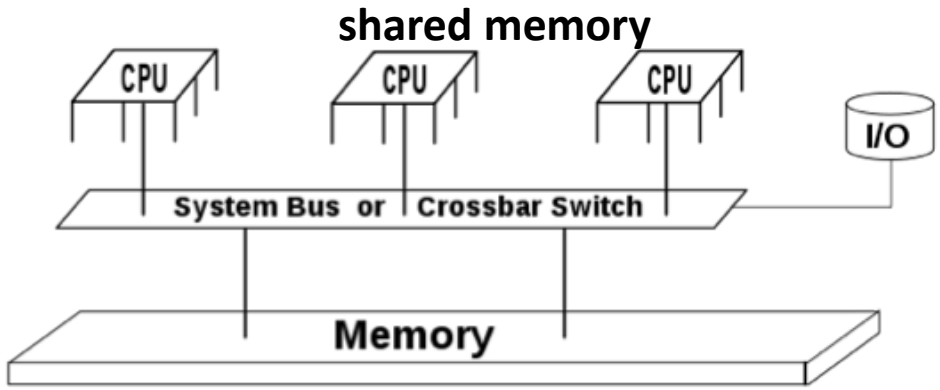
Why writing parallel programs is hard?

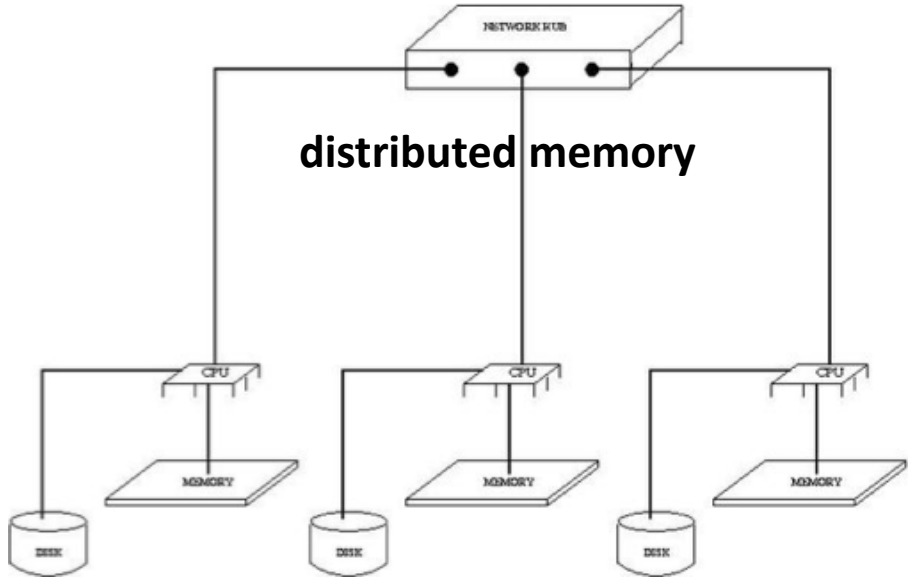
- Finding enough parallelism (Amdahl's law)
- Locality –moving data costs more than arithmetics
- Load balance
- Coordination and synchronization: data sharing safety
- Tuning / debugging / profiling

Flynn’s taxonomy



Shared memory and distributed memory





Ways to work parallel

1. Scientific software (Matlab, Mathematica, ANSYS, OpenFoam, Tensorflow, PyTorch, etc)
   Актуальная задача – Neural Architecture Search (i.e. for mobile devices https://www.mdpi.com/2073-431X/10/8/104)

2. Libraries (Math Kernel Library, CuDNN, CuBlas, Thrust, Boost)

3. Parallel instruments (OpenMP, MPI, Cuda, OpenCL)

Types of parallelism in computing

1. Task parallelism (multi-core CPUs) – different tasks or instructions can be done concurrently
2. Data parallelism (GPUs) – same instruction can be done on multiple data concurrently

Types of parallelism in computing

1. Task parallelism (multi-core CPUs) – different tasks or instructions can be done concurrently
2. Data parallelism (GPUs) – same instruction can be done on multiple data concurrently

Types of parallel tasks:

1. Embarrassingly parallel tasks (aka ”farming”) – no communication between workers

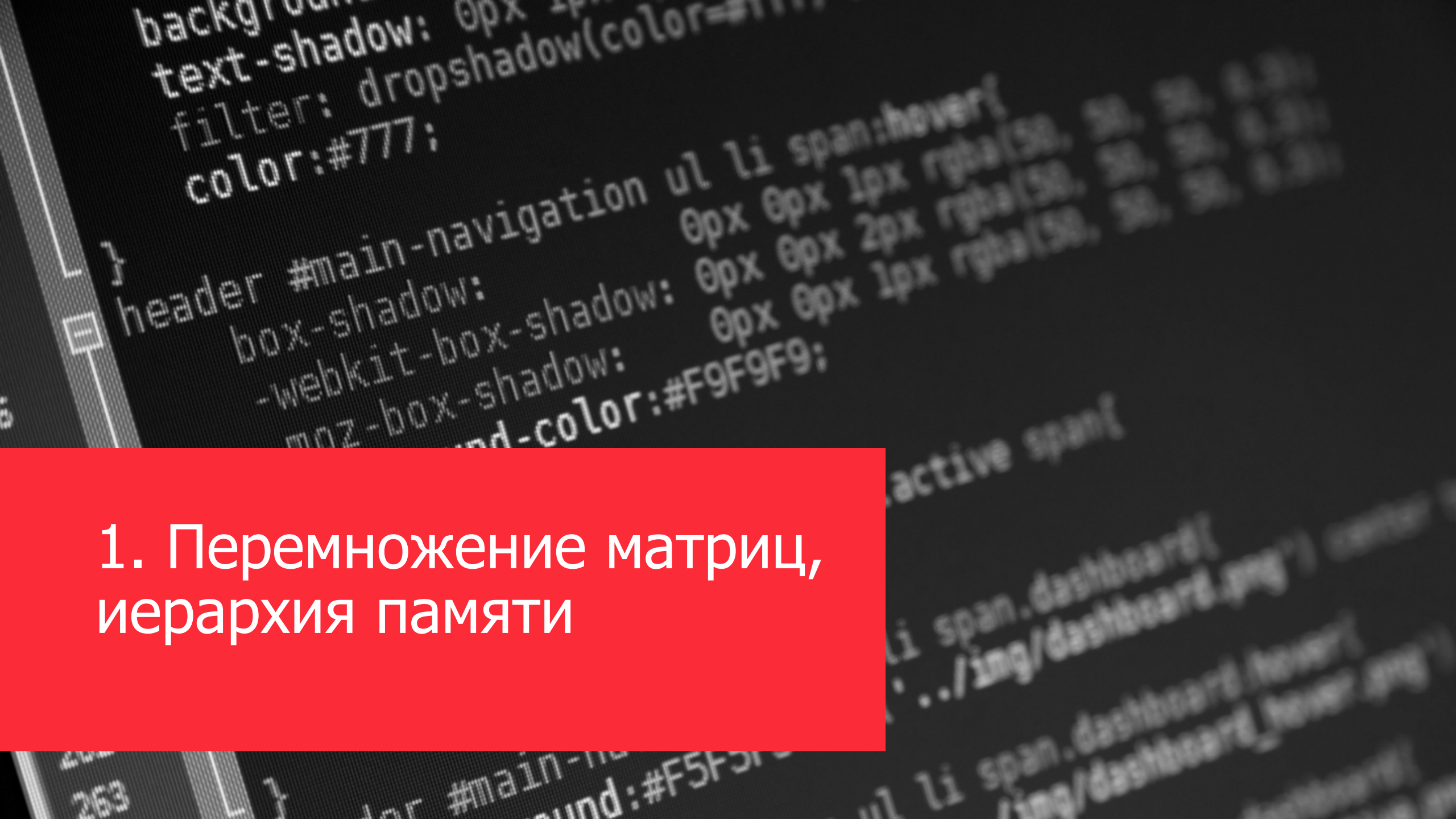
   bitcoin mining, solving integral with a parameter for 1e6 parameters, running simulation for different input parameters (different initial conditions)

2. Compute-intensive tasks

   main time is spent on computing, low level of communication, i.e. numerical photonics or plasma physics simulations

3. Data-intensive tasks

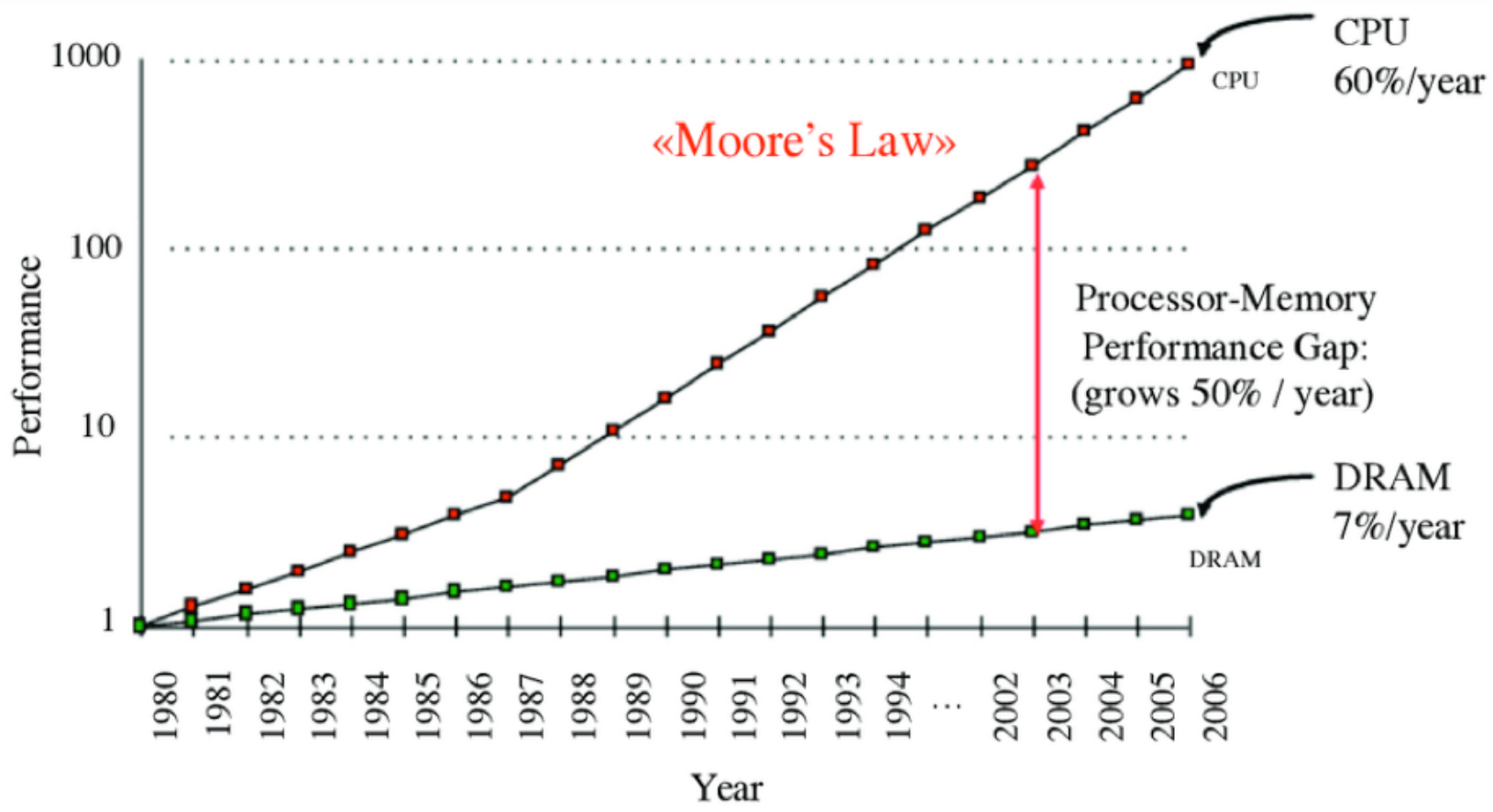
   main time is spent on data transfer (big data, operations with file system, databases)



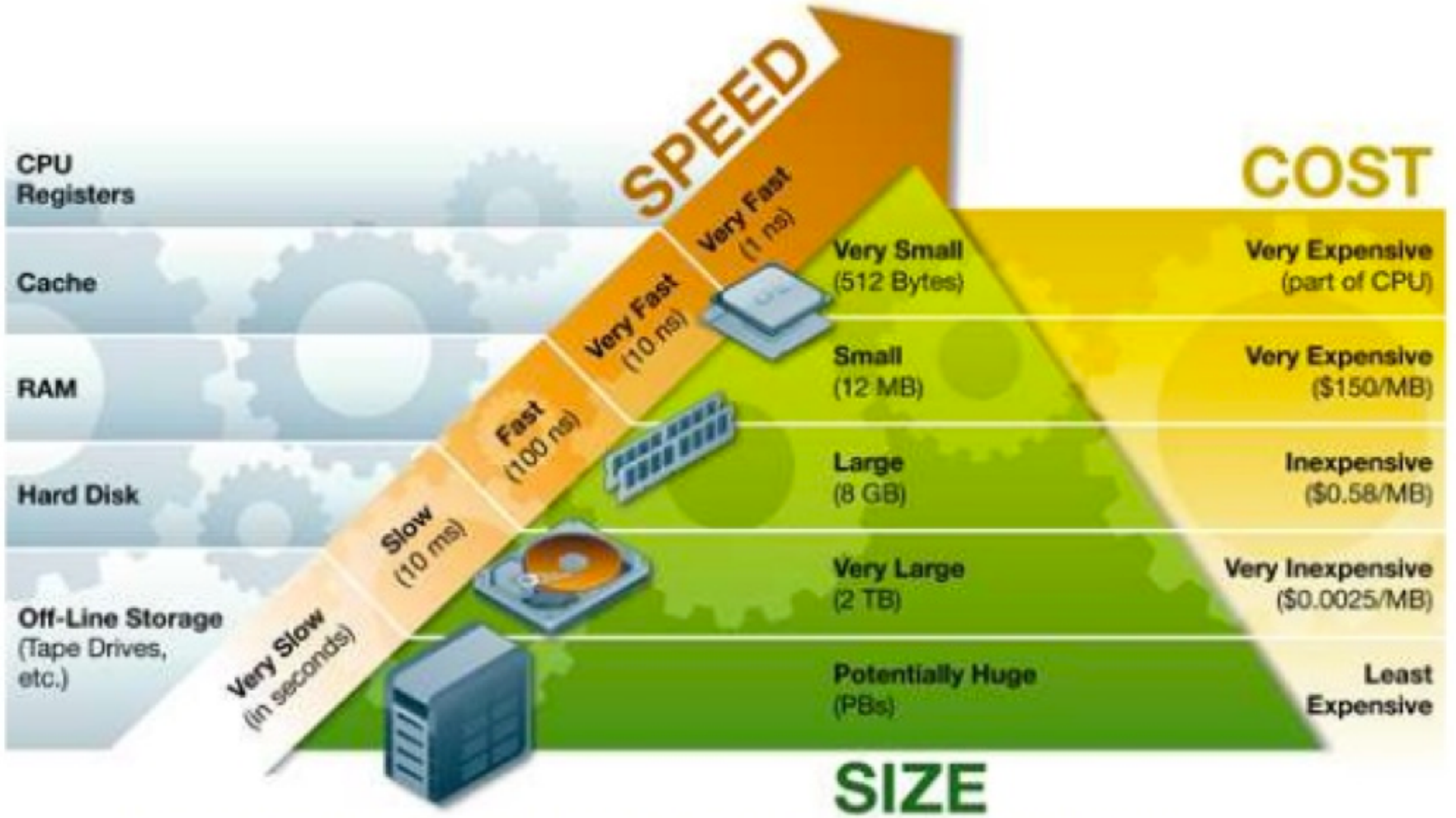
1. Перемножение матриц, иерархия памяти

Отставание производительности памяти

Memory performance gap

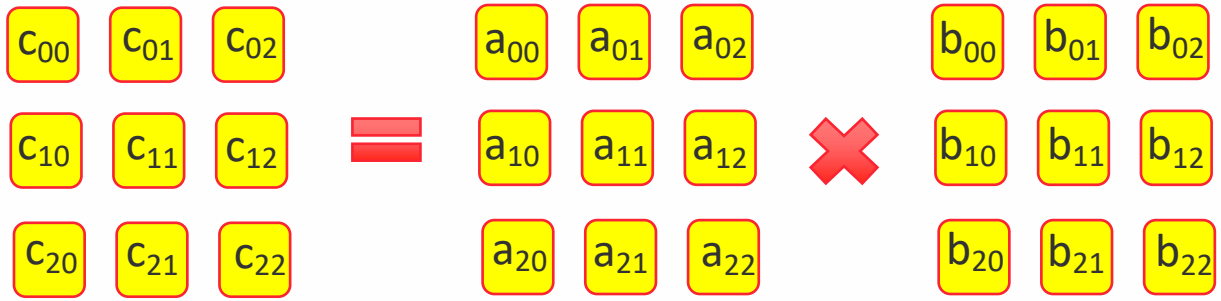


Иерархия памяти в компьютере



Source: http://www.ts.avnet.com/uk/products_and_solutions/storage/hierarchy.html

«Наивное» перемножение матриц



NxN matrices: A, B, C
N >> 1

C = A * B

$$c_{ij}=\sum_{n=0}^{n=N-1} a_{in}*b_{nj}$$

```
for i from 0 to N-1:
      for j from 0 to N-1:
            for n from 0 to N-1:

      C[i][j]+=A[i][n]*B[n][j]
```

https://bit.ly/339RAhp

Основные элементы кода: аллокация памяти и адресация

```
double *A, *B, *C;
A = (double *) malloc(N * N * sizeof(double));
B = (double *) malloc(N * N * sizeof(double));
C = (double *) malloc(N * N * sizeof(double));
free(A); free(B); free(C);
```

матрица NxN

```
void RandomMatrix(double * A, size_t N)
{
    srand(time(NULL));
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            A[i * N + j] = rand() / RAND_MAX;
        }
    }
}
```

адресация к элементу A_{ij}

Основные элементы кода: matmul и измерение времени

matmul:

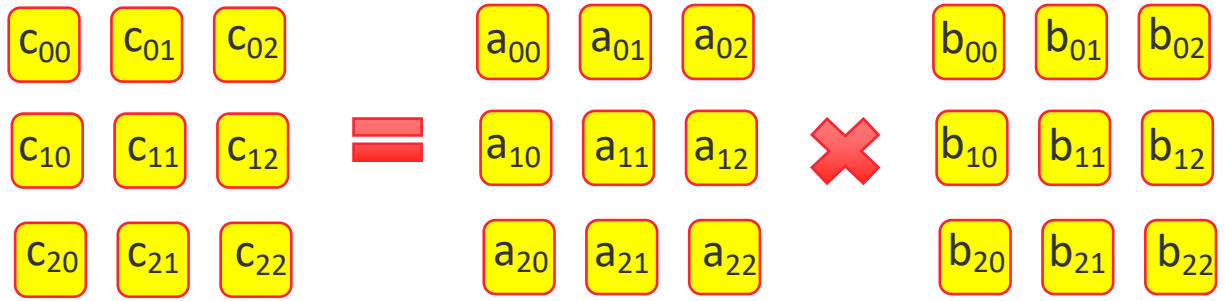
```
for (i = 0; i < N; i++)
        for(j = 0; j < N; j++)
            for(k = 0; k < N; k++)
                C[i * N + j] = C[i * N + j] + A[i * N + k] * B[k * N + j];
```

```
struct timeval start, end;
double r_time = 0.0;
gettimeofday(&start, NULL);
... работа
... еще работа
gettimeofday(&end, NULL);
r_time = end.tv_sec - start.tv_sec + ((double)(end.tv_usec - start.tv_usec)) / 1000000;
```

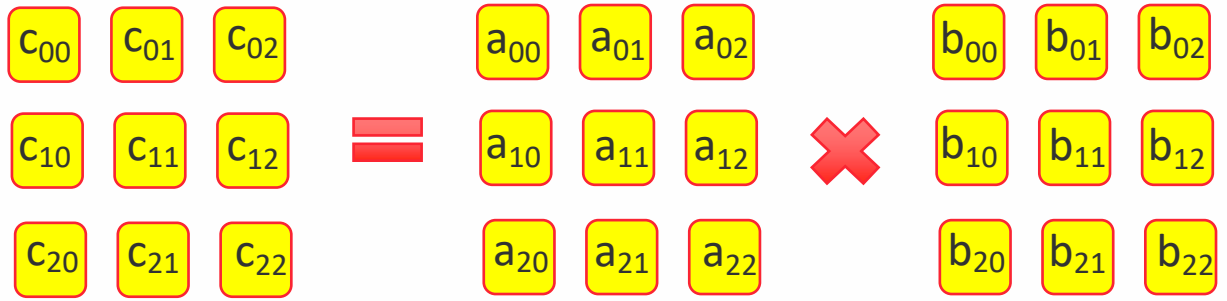
секунды

микросекунды

другие опции с разным разрешением: clock(); clock_gettime()

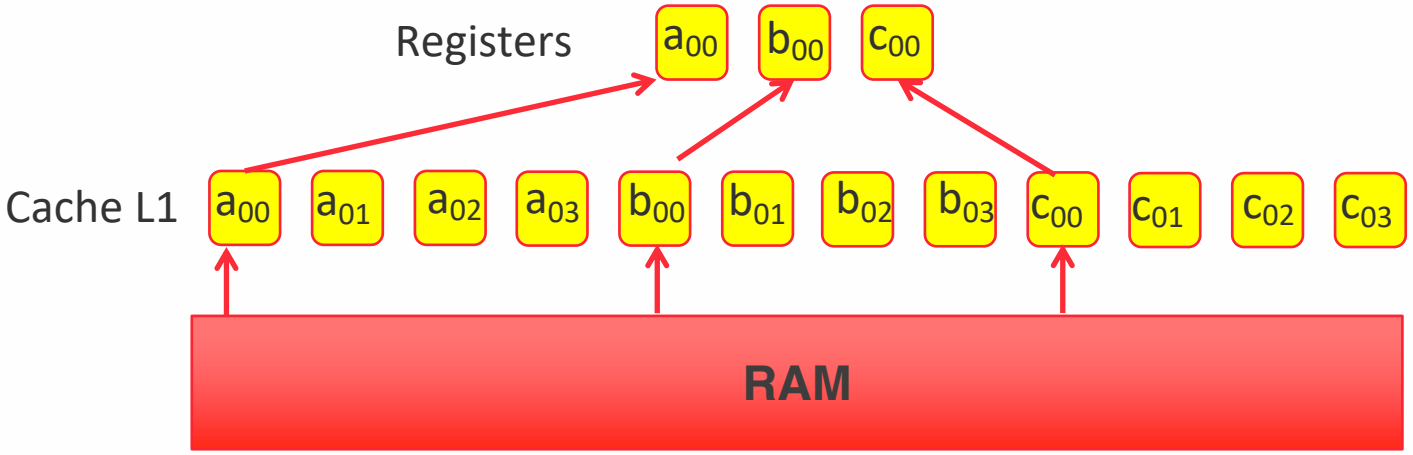


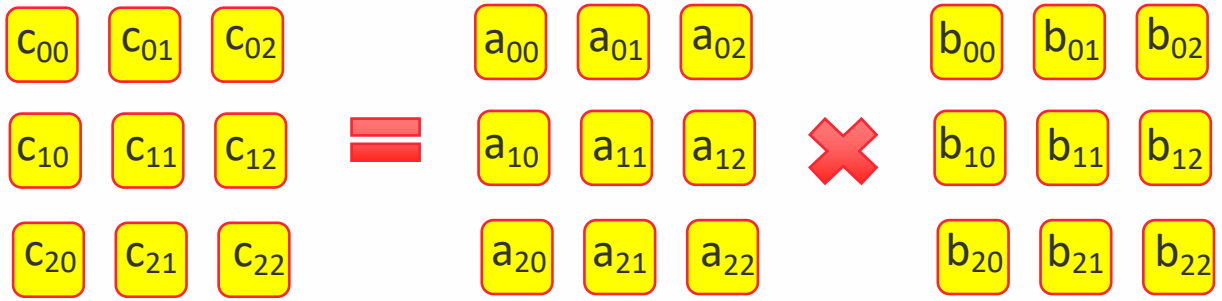

```
for i from 0 to N-1:
       for j from 0 to N-1:
              for n from 0 to N-1:

       C[i][j]+=A[i][n]*B[n][j]
```




```
for i from 0 to N-1:
       for j from 0 to N-1:
              for n from 0 to N-1:

       C[i][j]+=A[i][n]*B[n][j]
```

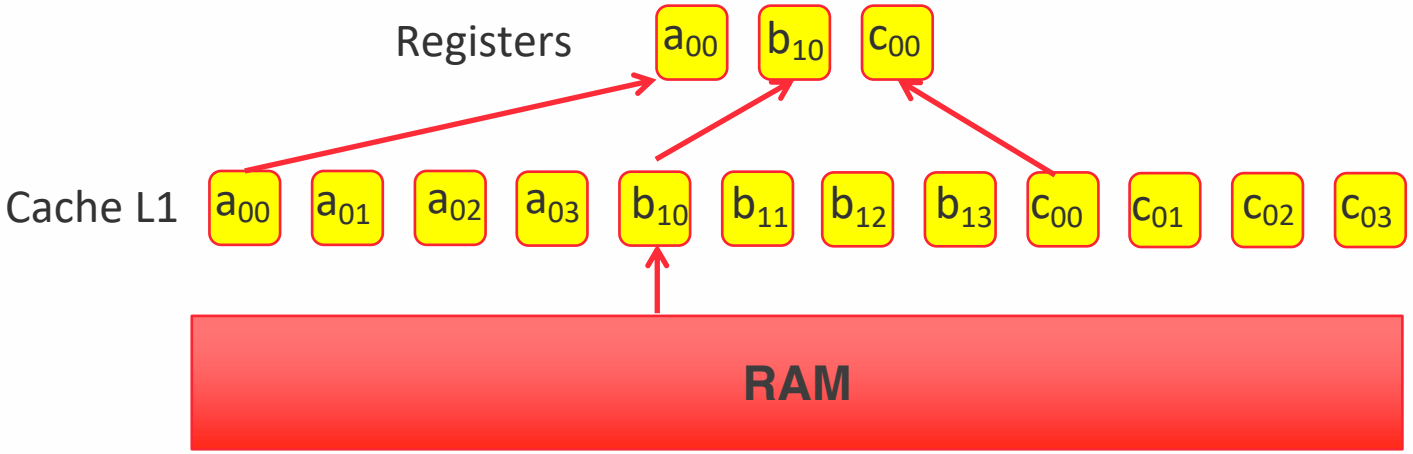
```
i=0
j=0
n=0
C[0][0]+=A[0][0]*B[0][0]
```

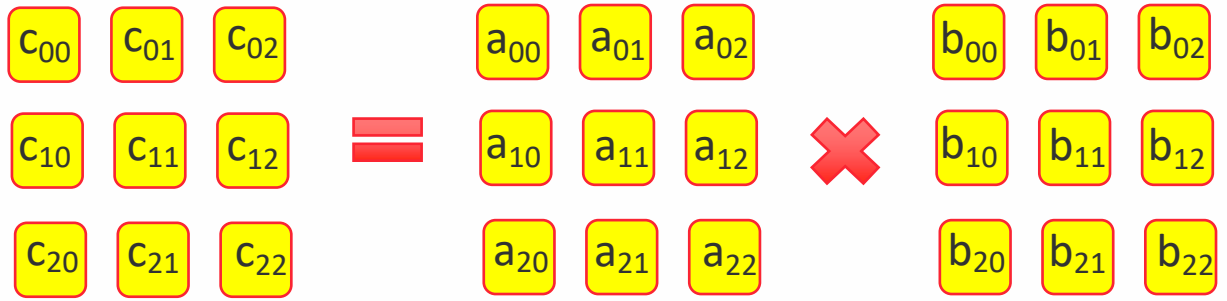





```
for i from 0 to N-1:
       for j from 0 to N-1:
              for n from 0 to N-1:

       C[i][j]+=A[i][n]*B[n][j]
```

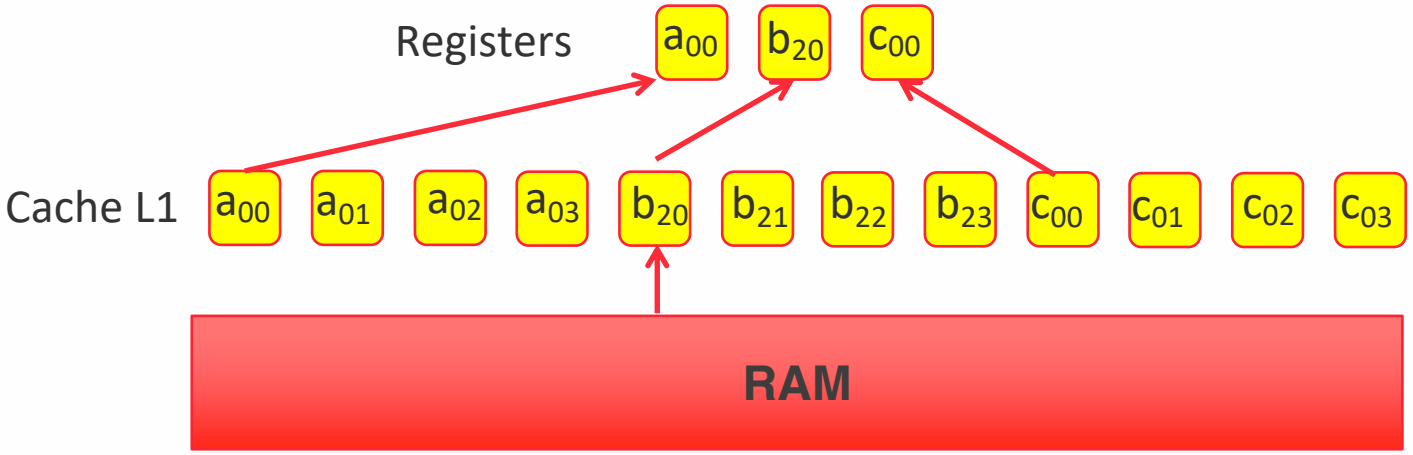
```
i=0
j=0
n=1
C[0][0]+=A[0][1]*B[1][0]
```

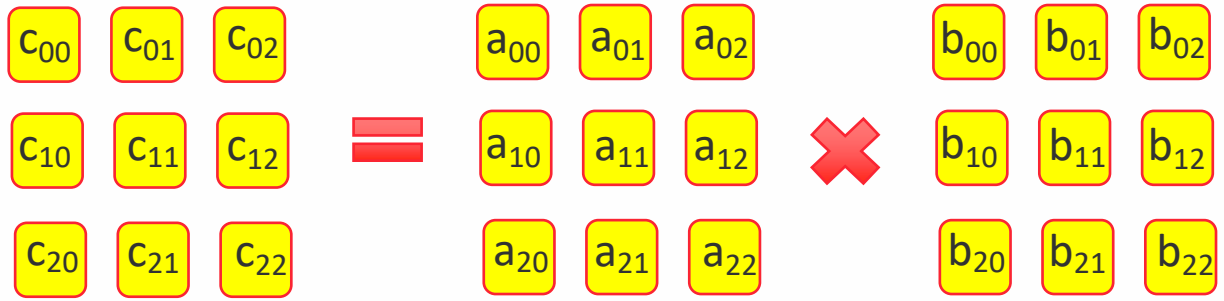




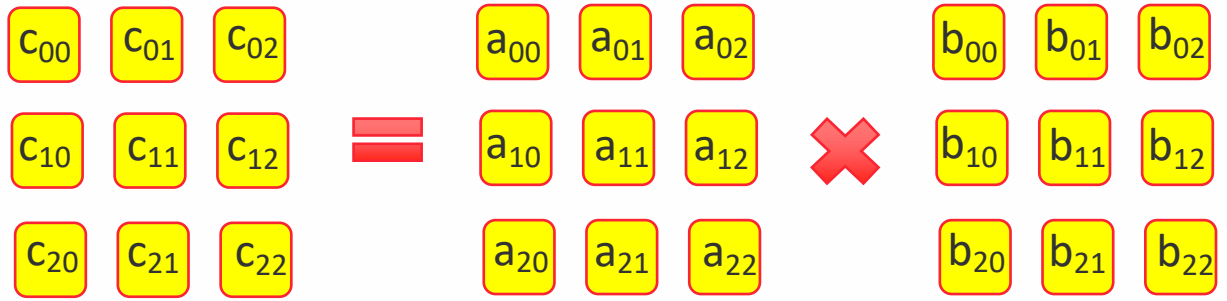

```
for i from 0 to N-1:
       for j from 0 to N-1:
              for n from 0 to N-1:

       C[i][j]+=A[i][n]*B[n][j]
```

```
i=0
j=0
n=2
C[0][0]+=A[0][2]*B[2][0]
```



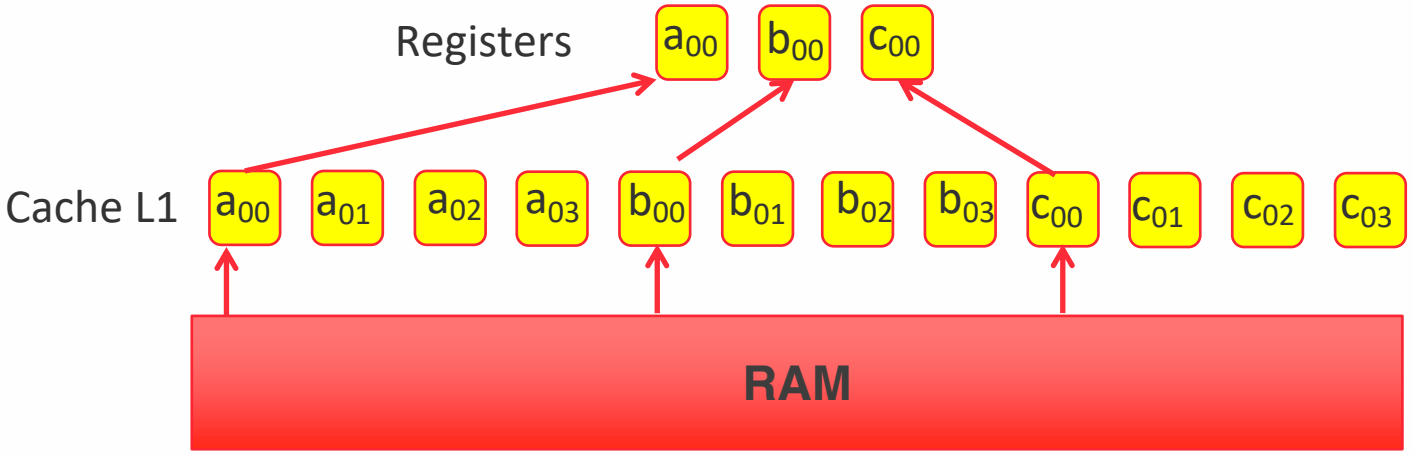


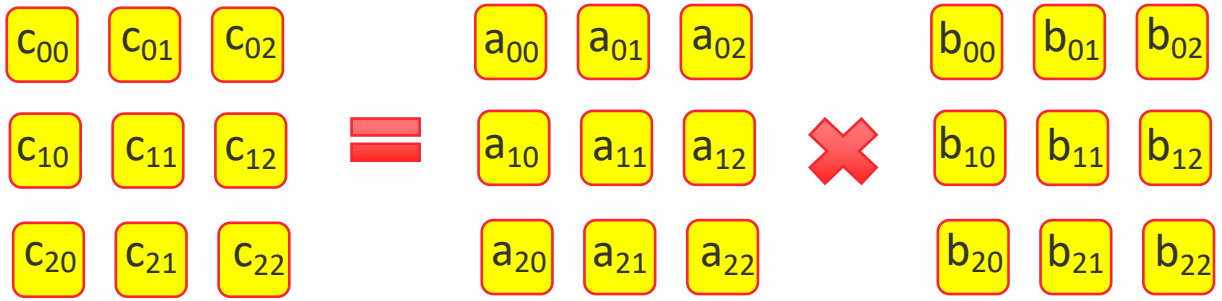

```
for n from 0 to N-1:
       for i from 0 to N-1:
              for j from 0 to N-1:

       C[i][j]+=A[i][n]*B[n][j]
```




```
for n from 0 to N-1:
       for i from 0 to N-1:
              for j from 0 to N-1:

       C[i][j]+=A[i][n]*B[n][j]
```

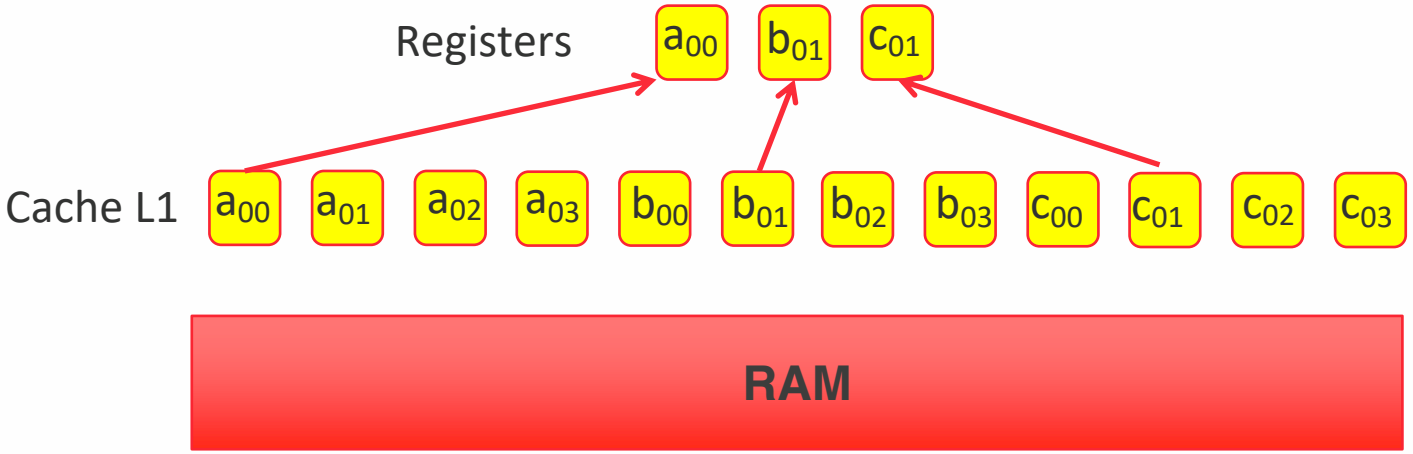
```
n=0
i=0
j=0
C[0][0]+=A[0][0]*B[0][0]
```






```
for n from 0 to N-1:
       for i from 0 to N-1:
              for j from 0 to N-1:

       C[i][j]+=A[i][n]*B[n][j]
```

```
n=0
i=0
j=1
C[0][1]+=A[0][0]*B[0][1]
```



Выводы к первой лекции

1. Высокопроизводительные вычисления активно развиваются и приносят ощутимый экономический эффект
2. Видна тенденция на соединение ИИ и высокопроизводительных вычислений
3. Виден запрос от индустрии на HPC и Green AI
4. Разнообразные вычислительные платформы требуют разнообразных приемов HPC

д/з 1

- 50% На основании данных рейтинга top500 спрогнозировать производительность компьютера №1 в 2025
- 50% оцените какой процент всего производимого на Земле электричества потребляют все компьютеры из top500 (на основании открытых данных top500.org)
- 10% Бонус: Улучшить производительность программы с реализацией алгоритма матричного умножения

Какие домашки? после следующих лекций:

1. Вводная лекция по истории HPC	**5-6. Библиотека MPI и MPI4Py**
2. Терминал, переменные окружения Makefile	**7, 9. Cuda/PyCuda**
3. Pthreads, openmp	**8. Профилировка и биндинг C/C++/Python ****
4. Навыки работы на суперкомпьютере *	** полностью бонусная