High Performance Computing

高性能計算論

Volume 1

Cyberscience Center, Tohoku Univ

Hiroyuki Takizawa
<takizawa@tohoku.ac.jp>

Instructor

- **Hiroyuki TAKIZAWA (滝沢寛之)**
  - Processor at Cyberscience Center
  - Research Interests
    - High-performance computing software and systems
    - Computer architectures
    - Parallel and distributed computing systems and apps

E-mail: takizawa@tohoku.ac.jp
http://www.sc.cc.tohoku.ac.jp/~tacky

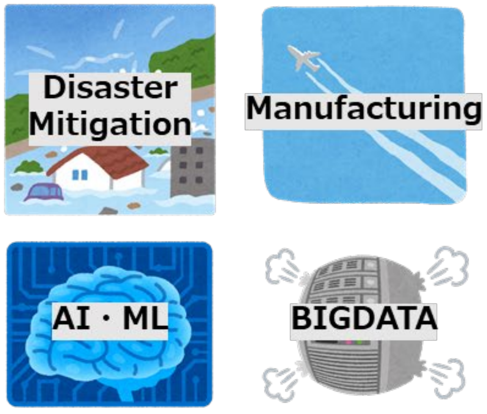
HPC Lab (Takizawa lab)

System Operation

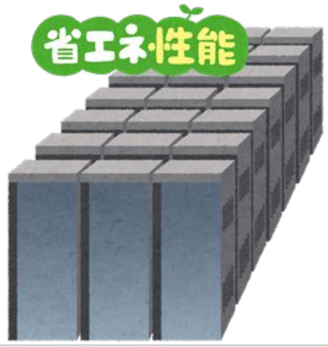
Performance-Aware Programming

Different kinds of processors available on a system
CPU
GPU
NEC
VE

Each has its own advantages and disadvantages
→ **Need to Assign Right Task to Right Processor**



Application Optimization



Power saving is one of the most important concerns.

Supercomputer as an Infrastructure working with sensors and IoT devices

Exploring Next-Gen Supercomputing Technologies

Class Schedule (tentative!)

1-Oct	Introduction to HPC (1)
1-Oct	Introduction to HPC (2)
8-Oct	Parallel Architectures
8-Oct	How to use Supercomputer AOBA
15-Oct	Parallel Algorithm Design (1)
15-Oct	Parallel Algorithm Design (2)
5-Nov	MPI Programming (1)
5-Nov	MPI Programming (2)
12-Nov	OpenMP Programming (1)
12-Nov	OpenMP Programming (2)
26-Nov	Hybrid Programming
26-Nov	Performance Modeling and Analysis

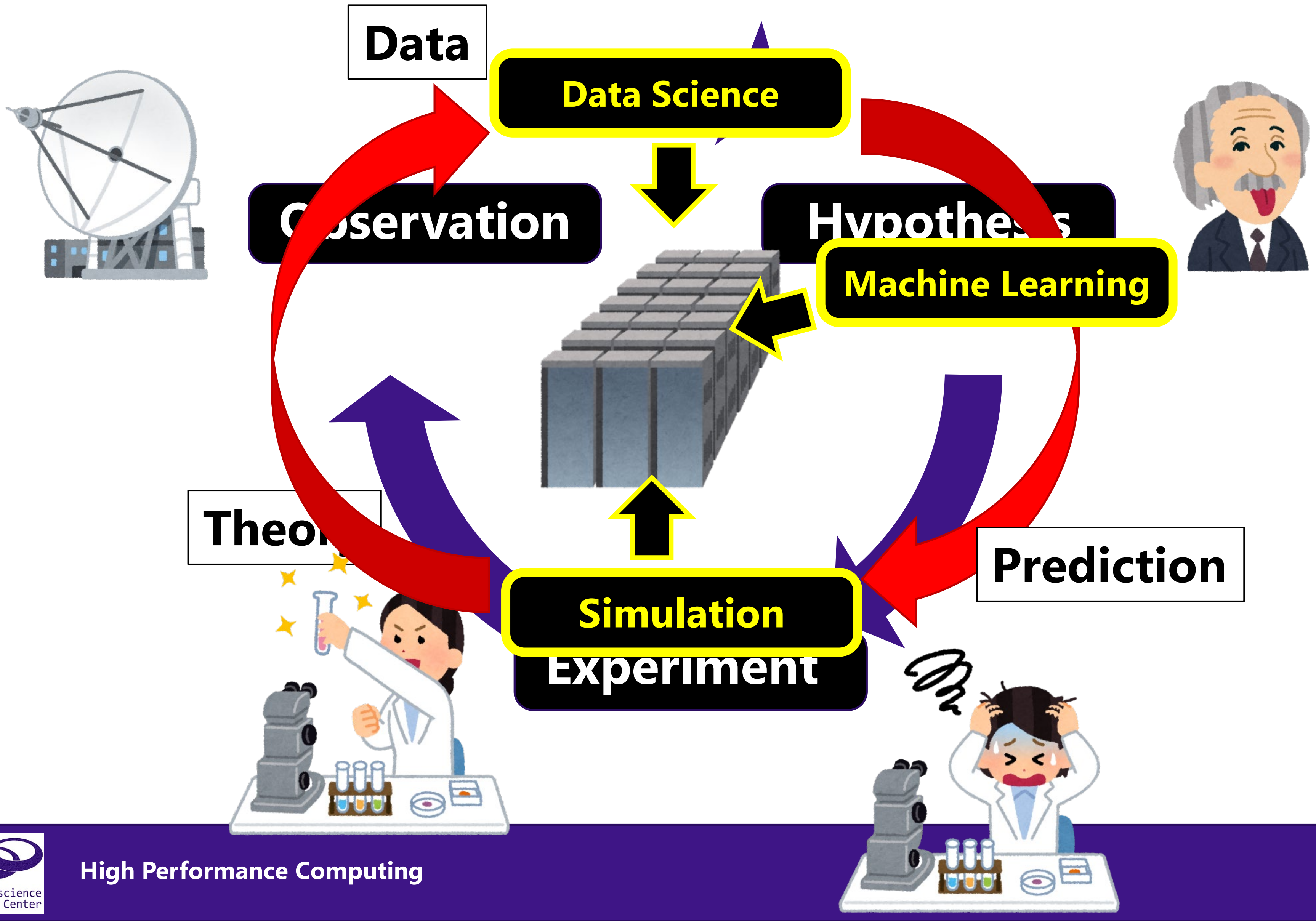
Introduction

- **High Performance Computing (HPC)**
  - An HPC system is often called a **Supercomputer**

  - What parallel computing is and why it's growing in importance.
  - Where parallelism exists in modern hardware.
  - Why the amount of parallelism in application software is important.
  - How to exploit parallelism.

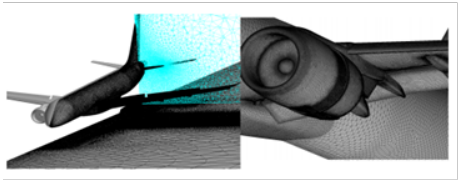
What is a supercomputer?

Modern Scientific Method

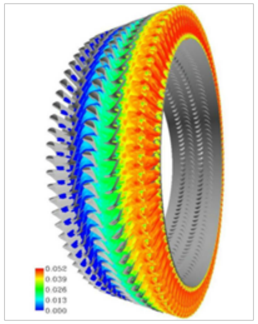


Target Applications

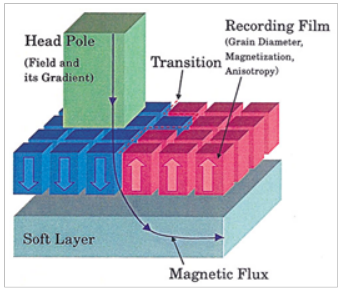
Next-Generation SuperSonic Transport



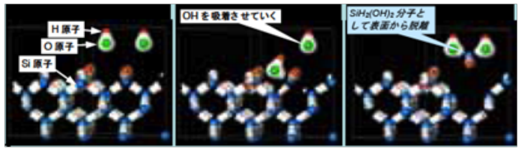
Turbine Design



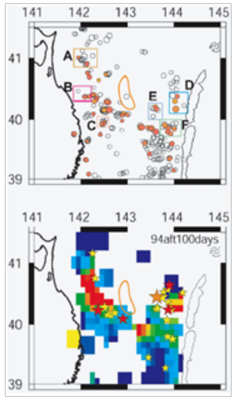
Perpendicular Magnetic Recording Medium Design



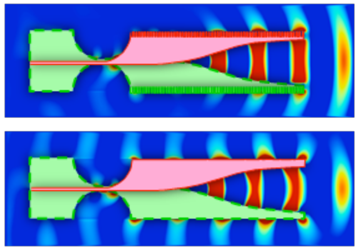
Nano Material Design

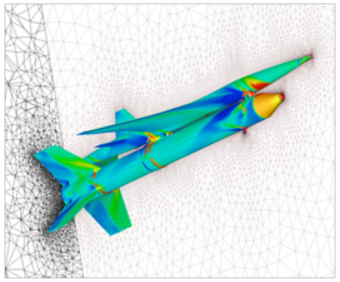


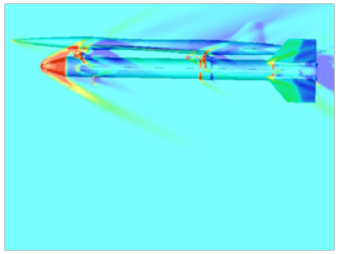
Earthquake Analysis



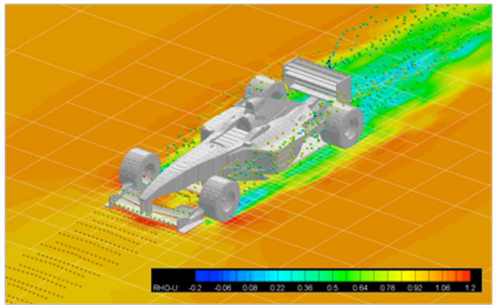
Antenna Analysis

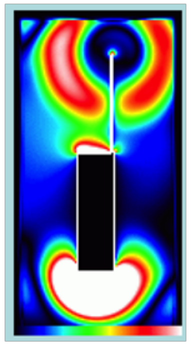




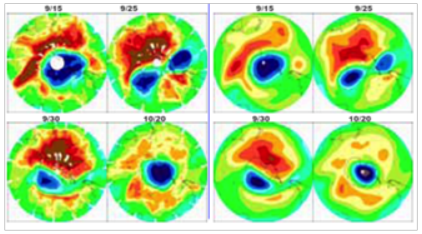


Next-Generation CFD Analysis



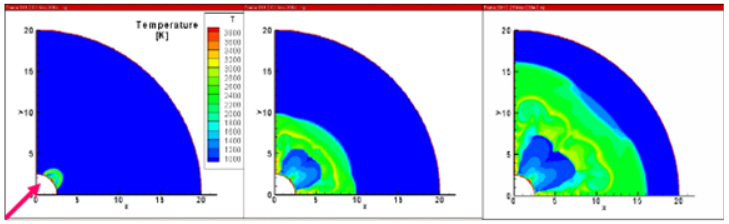


Ozone-hole Analysis

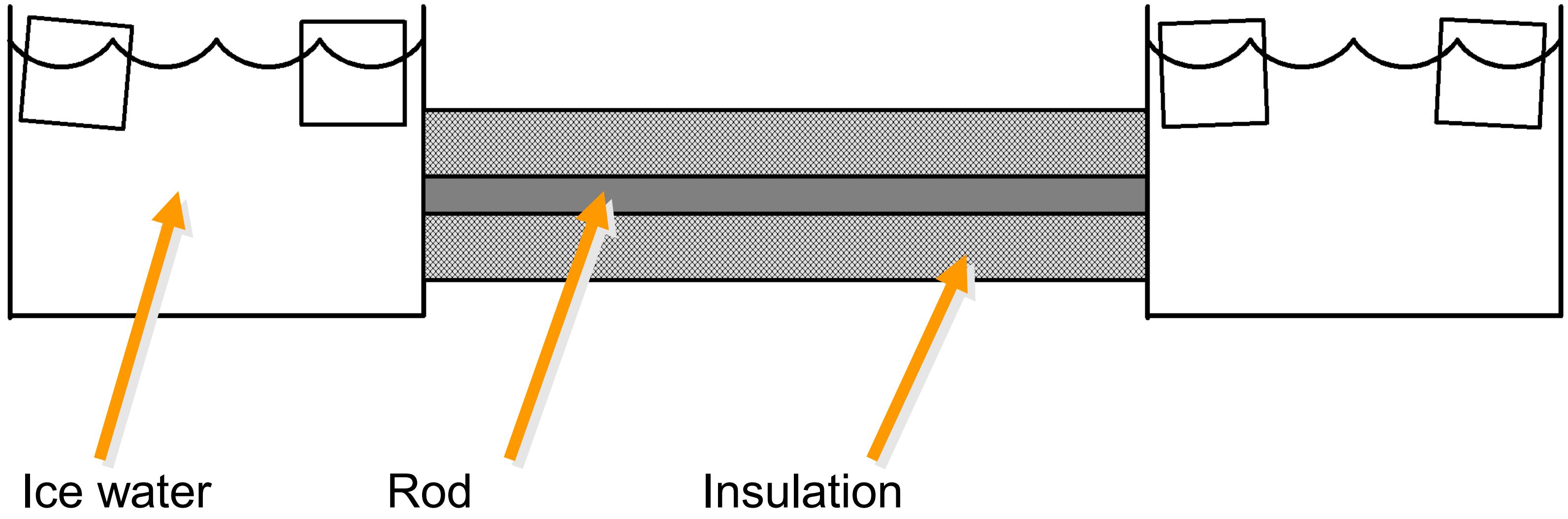


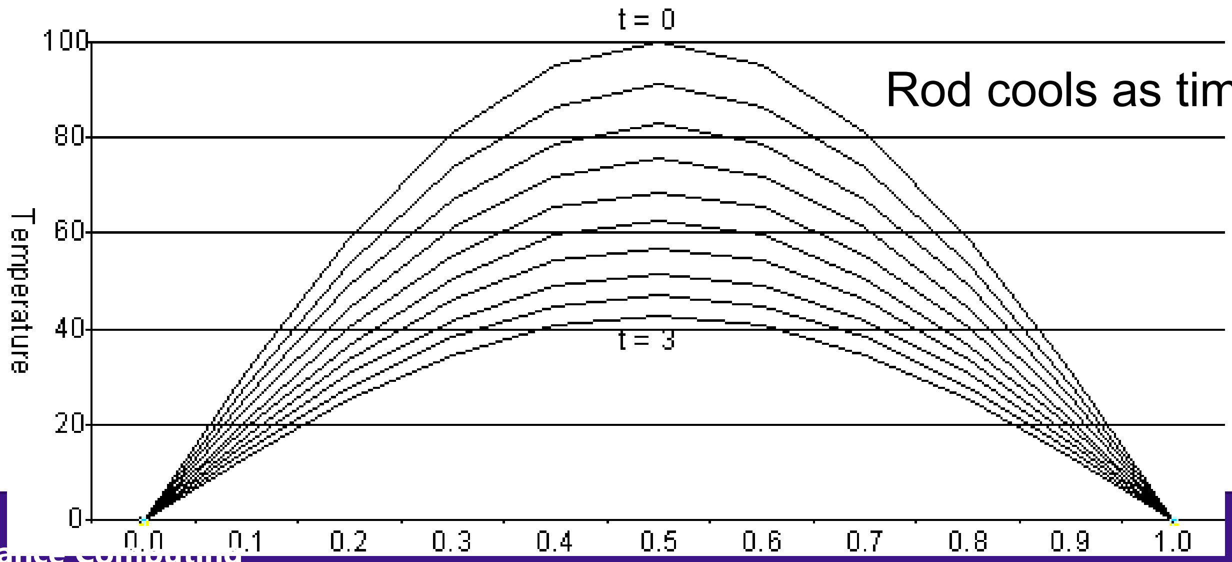
MRJ

Combustion Flow Simulation



Boundary Value Problem





Rod cools as time progresses

Difference Method (1/2)

- Heat conduction

Temparature at location x and time t

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2}$$

$$\frac{\partial u(x)}{\partial x} \approx \lim_{h\to 0} \frac{u(x+h) - u(x)}{h}$$

$$\frac{\partial^2 u(x)}{\partial x^2} \approx \lim_{h\to 0} \frac{u(x+h) - u(x)}{h^2} - \frac{u(x) - u(x-h)}{h^2}$$

$$= \lim_{h\to 0} \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}$$

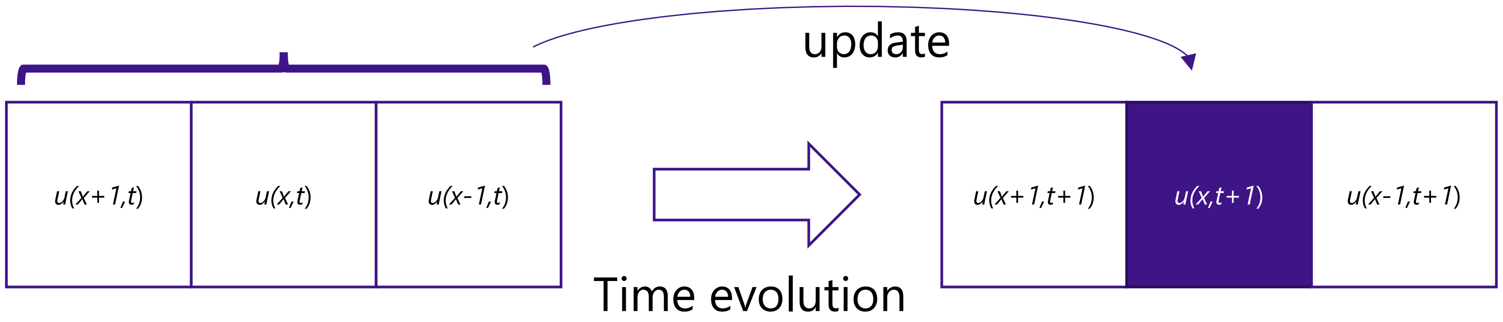
Difference Method (2/2)

■ Heat conduction

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2}$$

$$\frac{u(t+\Delta t)-u(t)}{\Delta t} = \frac{u(x+\Delta x)-2u(x)+u(x-\Delta x)}{\Delta x^2}$$

Temperature at time step t+Δt can be calculated by using the current temperatures of each location and its neighbors.



The simulation becomes more precise for smaller Δt and Δx , but needs more calculations (more locations and time steps) when the rod length and period are fixed.
→ More computing power is needed for numerical simulation.

Target Applications (Cont'd)

- AI and ML will be the predominant users ?

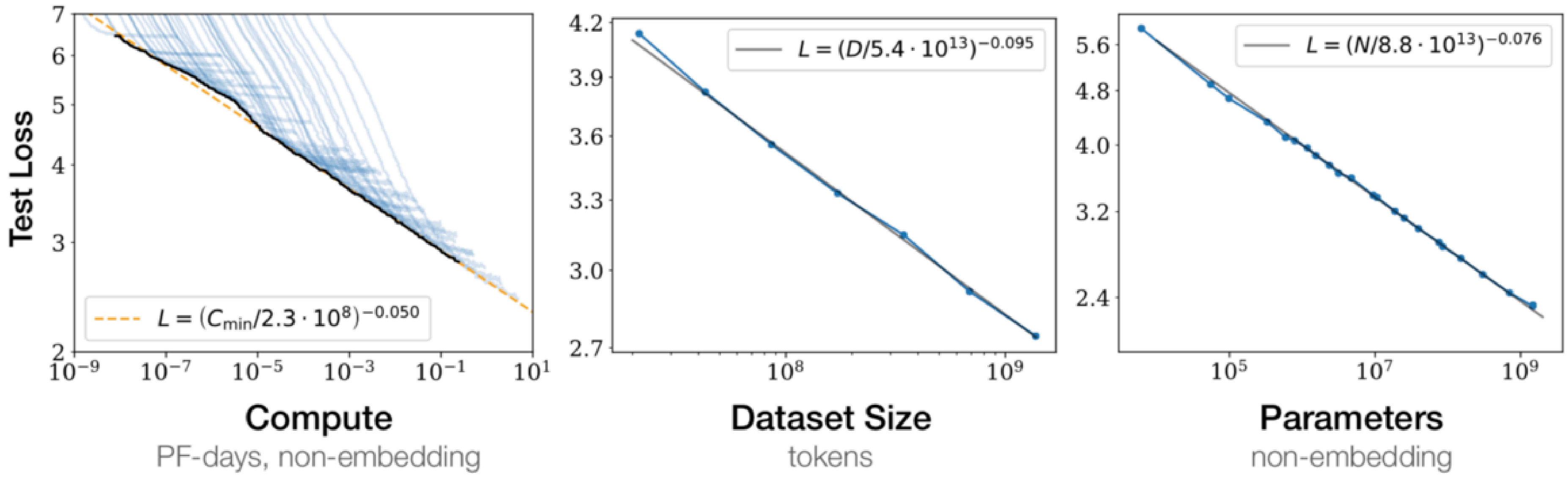
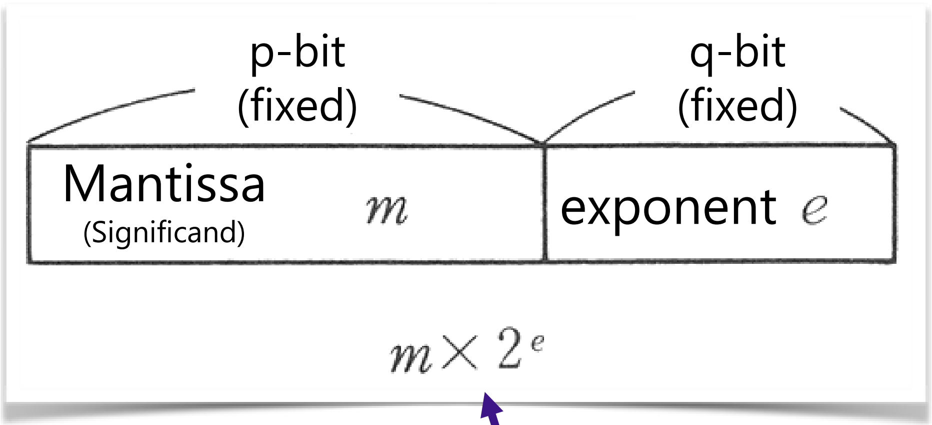


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Floating-Point Notation of Numbers

Floating-point notation

- A number is represented by a mantissa (significand) and an exponent, similar to scientific notation
- Representable range extended
- Complicated processing needed for arithmetic operations



Scientific notation of a number

FLOPS! Flops! flop/s!

- **Strong demand for high performance on floating-point operations.**
  - Numerical simulation of physical phenomena requires a huge number of floating-point operations.

  = a supercomputer must run it in practical time

➡ **A supercomputer is super in terms of FP operation performance**

- **Floating-point operation rate (FLOPS rate)**
  - The performance of a supercomputer is discussed based on **how many floating-point operations the supercomputer can execute per unit time**.

    The number of **fl**oating-point **op**erations per **s**ec
     = FLOPS, Flops, flop/s

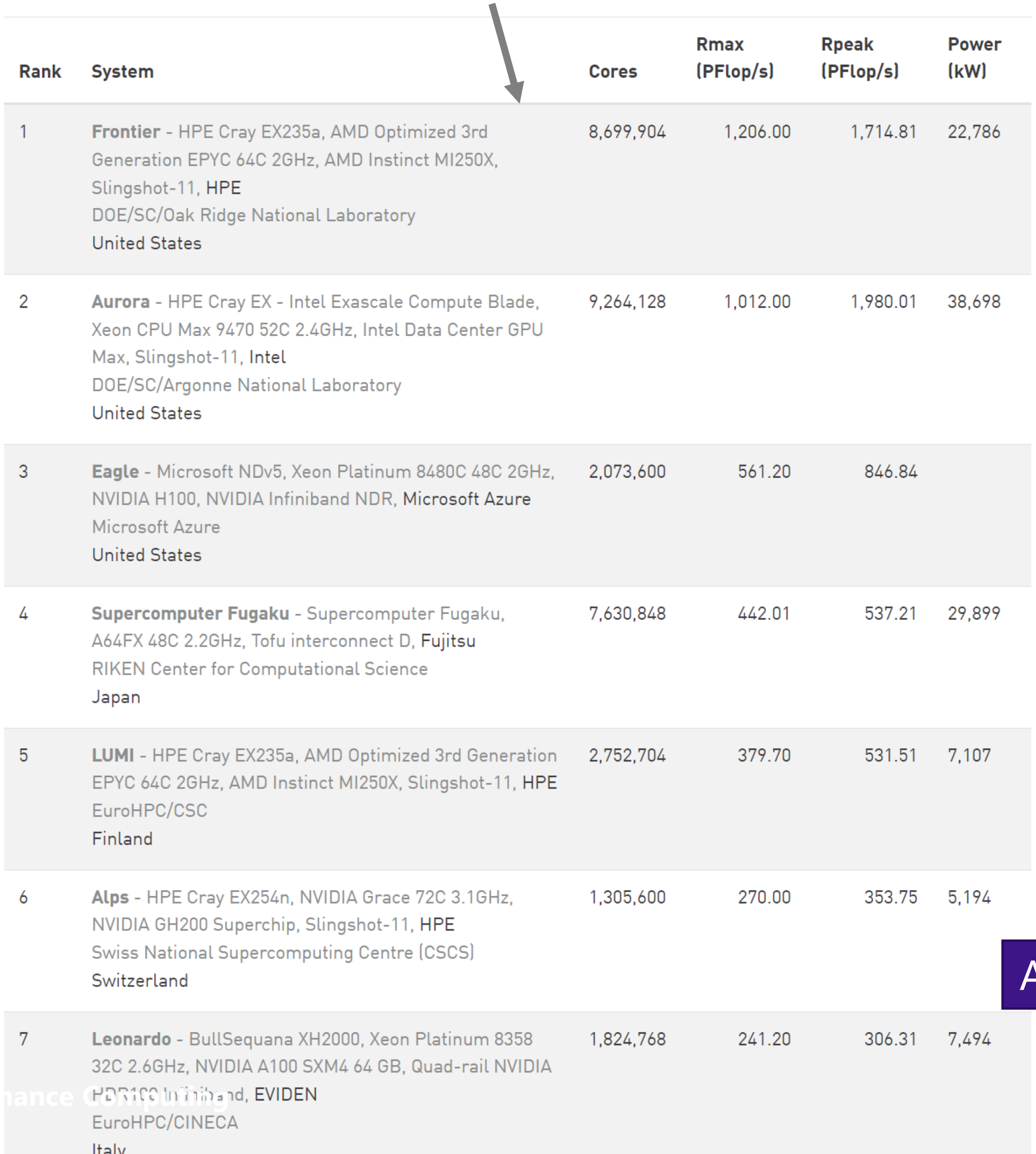
Prefixes

Chinese Characters (KANJI)	
■ 万 (man)	: 10^{4}
■ 億 (oku)	: 10^{8}
■ 兆 (chou)	: 10^{12}
■ 京 (kei)	: 10^{16}
■ 垓 (gai)	: 10^{20}
■ 秭/杼 (jo)	: 10^{24}
■ 穣 (jou)	: 10^{28}
…	
■ 無量大数	: 10^{68}

SI Prefixes (Intl. Syst. of Unit)	
■ K (kilo)	: 10^{3}
■ M (mega)	: 10^{6}
■ G (giga)	: 10^{9}
■ T (tera)	: 10^{12}
■ P (peta)	: 10^{15}
■ E (exa)	: 10^{18}
■ Z (zetta)	: 10^{21}
■ Y (yotta)	: 10^{24}

K Computer = 10 Pflop/s = 10^{16}

https://www.top500.org/



Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
6	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	1,305,600	270.00	353.75	5,194
7	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	241.20	306.31	7,494

As of June 2024

What is Parallel Computing?

What is Parallel Computing?

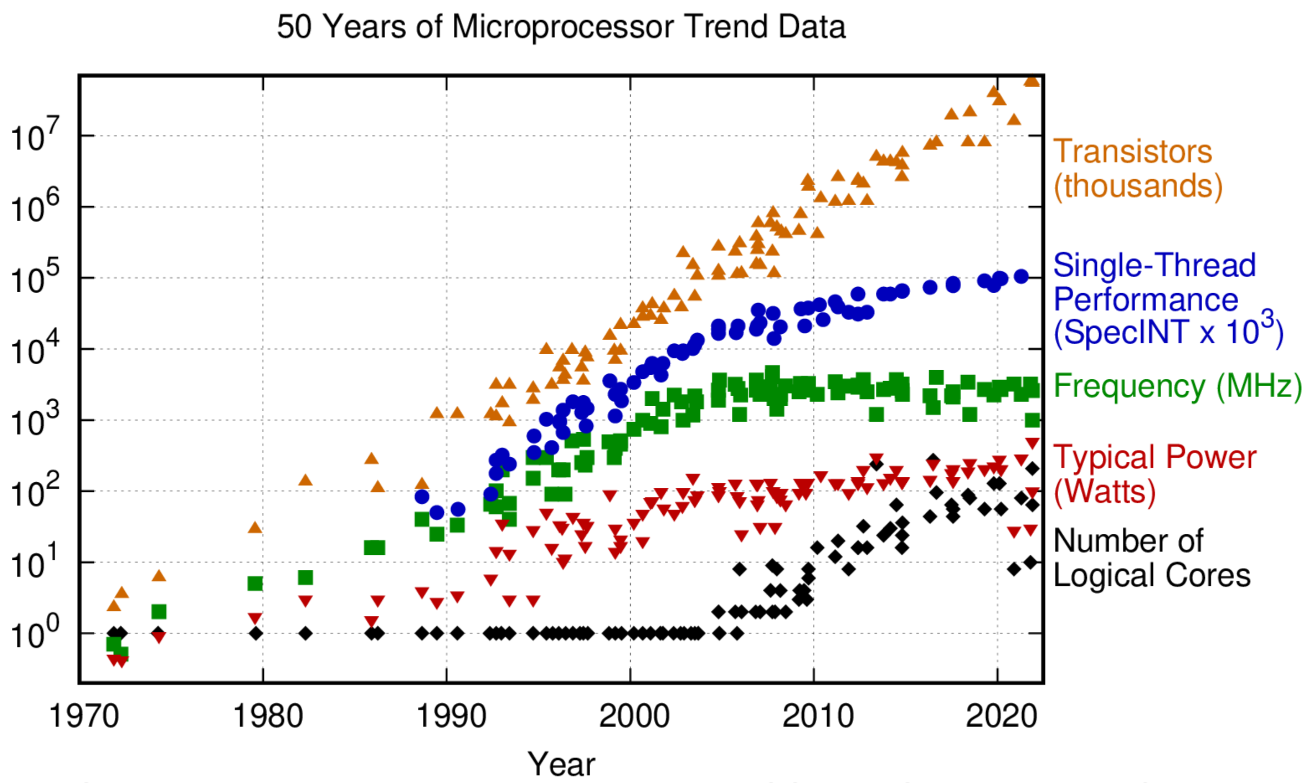
- **Parallel Computing**
  = execution of many operations at a time.

- **Parallel Computing is the practice of …**
  - Identifying and exposing parallelism in algorithms
  - Expressing the parallelism in the software implementation
  - Understanding the costs, benefits, and limitations of the chosen implementation

- **Parallel computing is about "performance"**
  - There could be various definitions of performance.
    - Ex) Speedup, Problem Size, Energy Efficiency

Why Parallel Computing?

- A computer no longer becomes faster but more parallel



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

Horowitz et al. and Rupp, https://github.com/karlrupp/microprocessor-trend-data

Clock Frequency and Power

- **Power consumption quickly increases with the clock frequency.**

$$P_d = \alpha \times f \times V_{dd}^2$$

P_d: dynamic power consumption
f: clock frequency
V_{dd}: power-supply voltage

<u>Note f is correlated with V_{dd}.</u>

→ P_d increases at the rate of $O(f^3)$
if f is proportional to V_{dd}.

Power Saving by Parallel Comp.

- **Let's calculate the power consumption of each system configuration.**
  - System 1: Processor A x 1

    $$f = 2GHz, V_{dd} = 1.5V$$
    $$P_d = \alpha \times 2000 \times 1.5 \times 1.5 = 4500\alpha$$

  - System 2: Processor B x 4

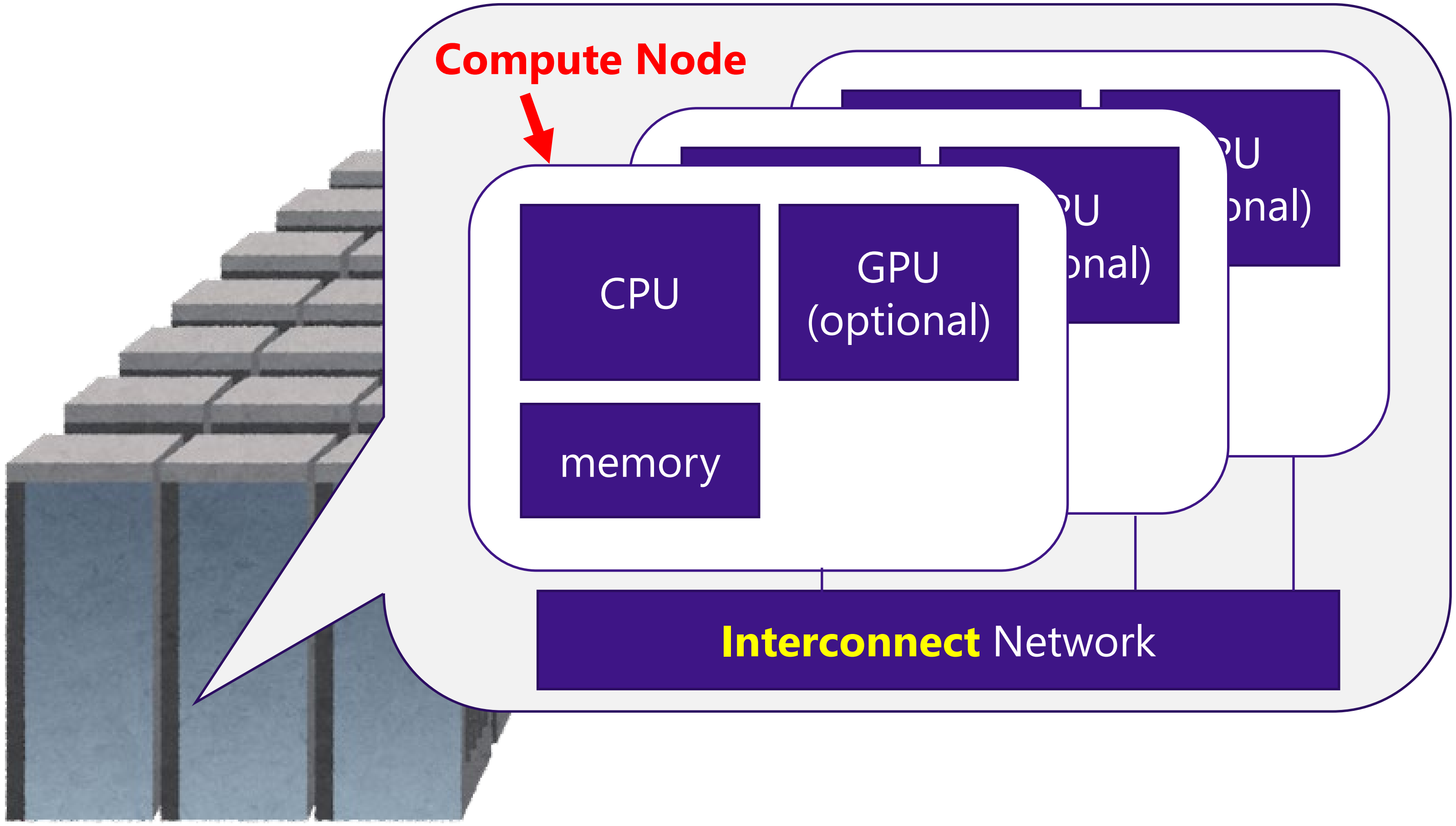
    $$f = 500MHz, V_{dd} = 0.8V$$
    $$P_d = \alpha \times 500 \times 0.8 \times 0.8 \times 4 = 1280\alpha$$

If 4 processors of System 2 work perfectly in parallel, System 2 is expected to achieve the same performance as System 1. The power consumption of System 2 is only about 28% of that of System 1.

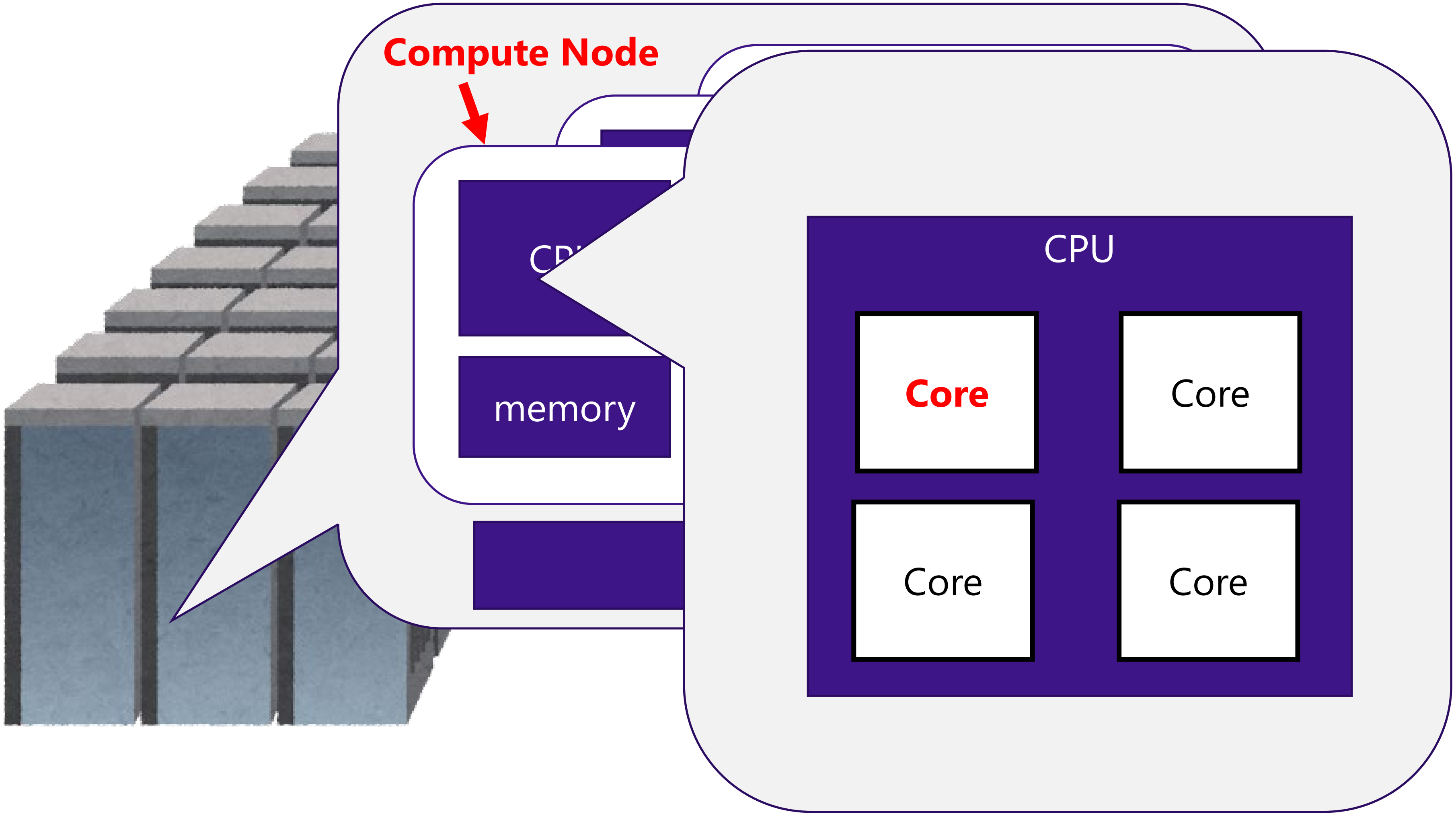
→ **Parallel computing is power-effective.**

Where Parallelism Exists

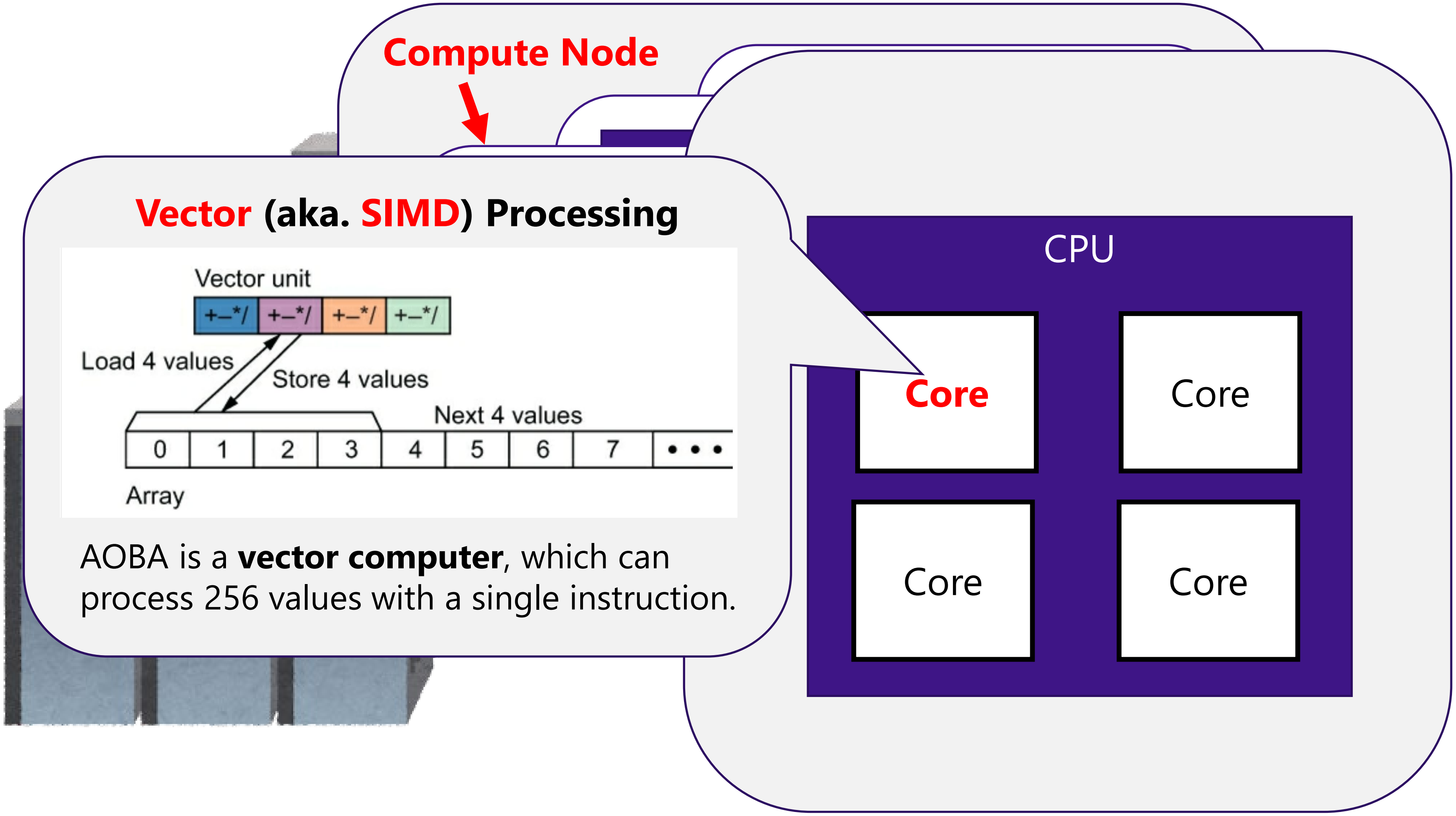
System Overview



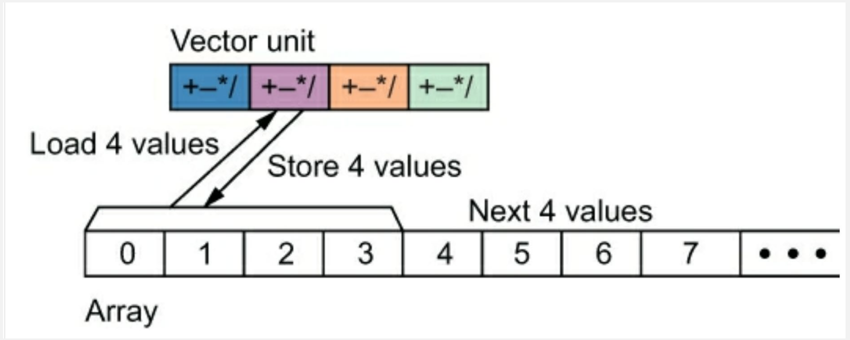
System Overview



System Overview



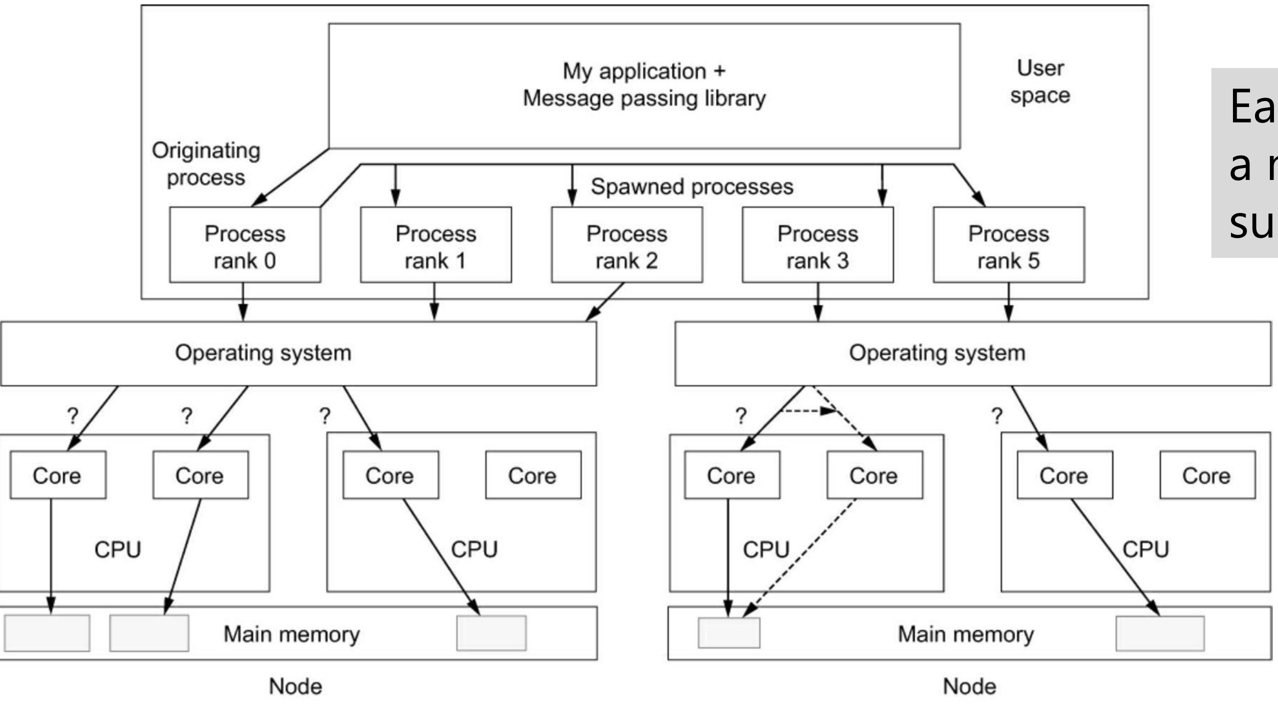
Vector (aka. SIMD) Processing



AOBA is a **vector computer**, which can process 256 values with a single instruction.

Software Overview

When a program is launched, several kinds of resources such as CPU time and memory are allocated for the execution. A unit of allocated resource is called a **process**. An **application** (user program) is executed by multiple processes.



Each process is assigned to a node, which runs the **OS**, such as **Linux**.

The OS on each node decides the core(s) to execute the process. A process can be executed by multiple cores. The execution sequence on each core is called a **thread**.

Benefit of Parallel Computing

- **Faster run time with more compute cores**
  - Reduction in an application's execution time, or **speedup**, is the primary goal of parallel computing.
- **Larger problem sizes with more compute nodes**
  - Exposing parallelism also permits to increase the problem size to be solved, because each node usually has its own memory.
- **Energy efficiency by doing more with less**
  - Using a larger number of slower but power-efficient cores allows the hardware to return to sleep mode sooner.

Energy = (No. CPUs) x (Power/CPU) x (Hour)

Importance of Application Parallelism

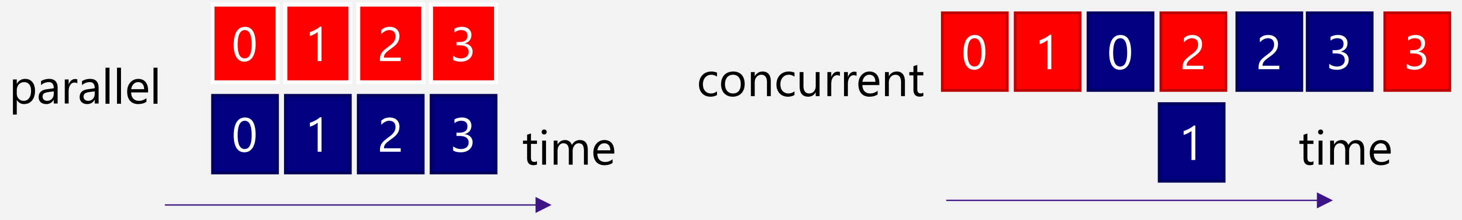
Seeking Concurrency

Data Dependence Graphs

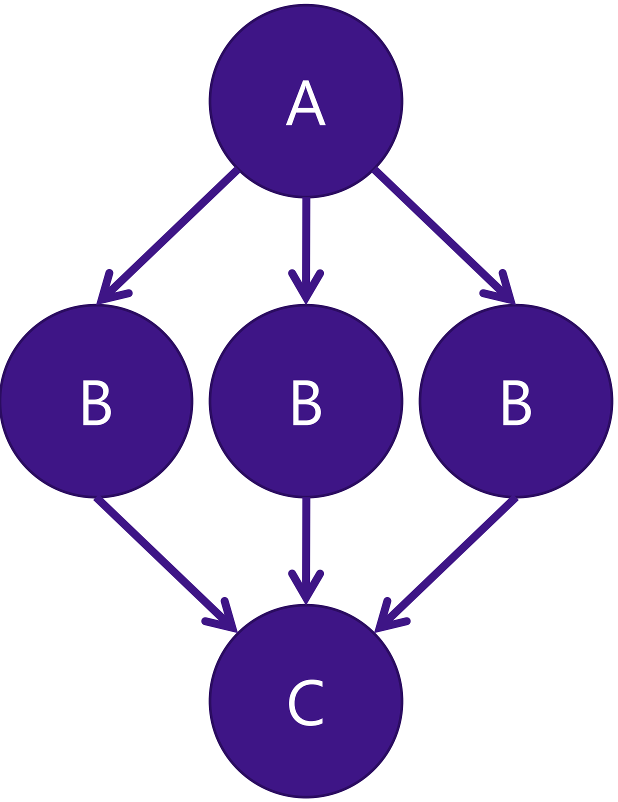
- Vertex
  - a task to be completed
- Edge from task *u* to task *v*
  - Task *u* must be completed before task *v* begins

If there is no edge between *u* and *v*, they are independent and may be performed concurrently.

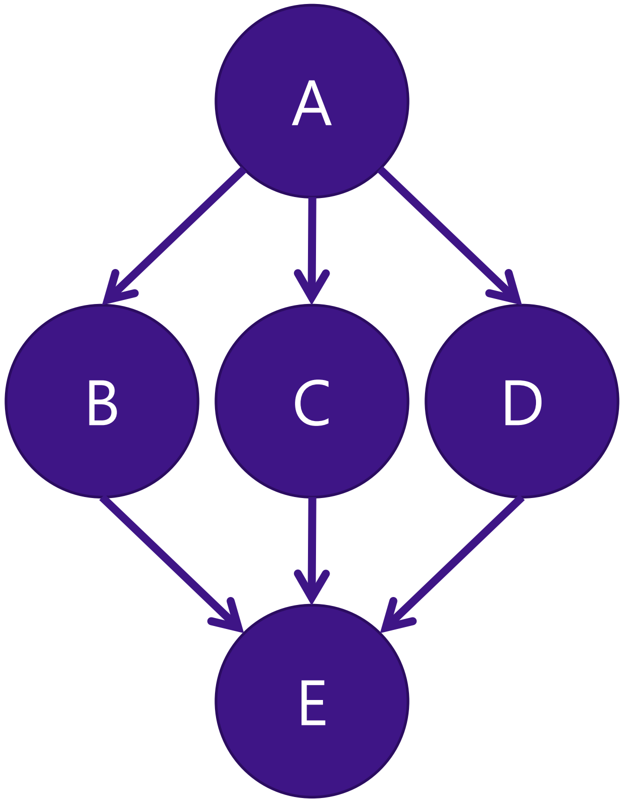
Parallel execution means multiple tasks are executed **physically at the same time**.
Concurrent execution means multiple tasks can be **in progress**, but not necessarily at the same time. **EX) Read and blue tasks of 4 steps are executed.**



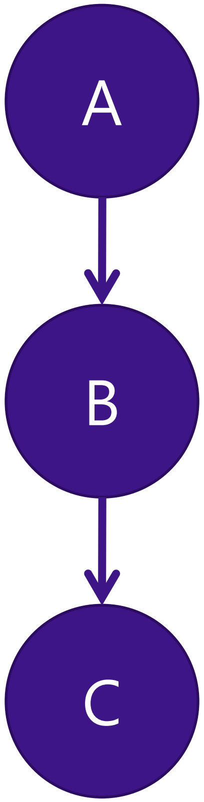
Parallelism in graphs



Data Parallelism

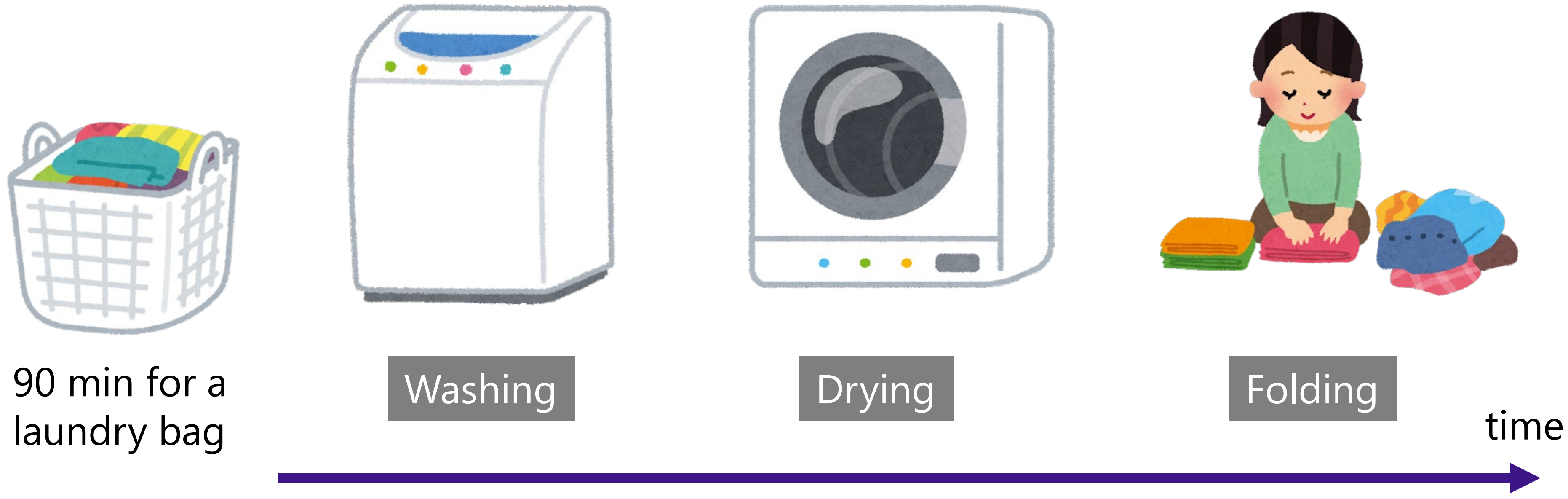


Functional Parallelism
(Task Parallelism)



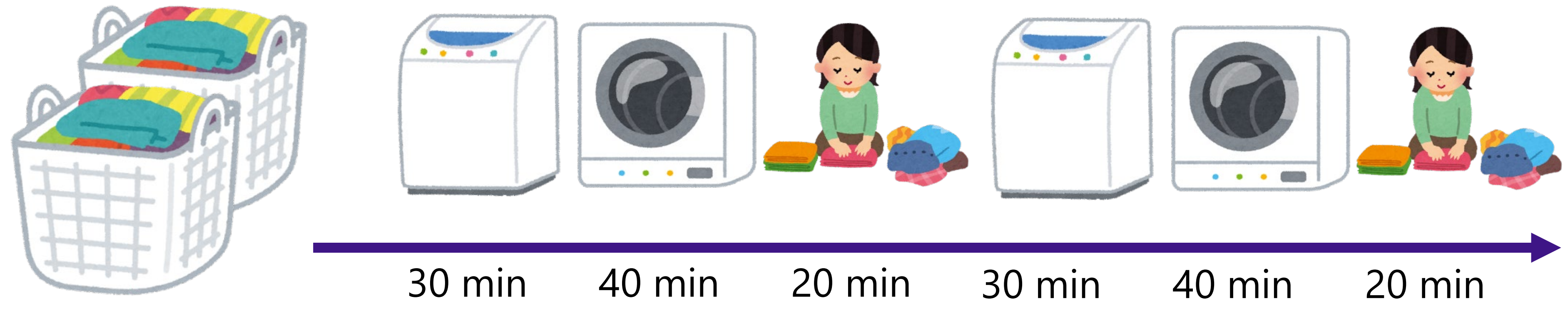
No Parallelism??

Laundry (serial version)

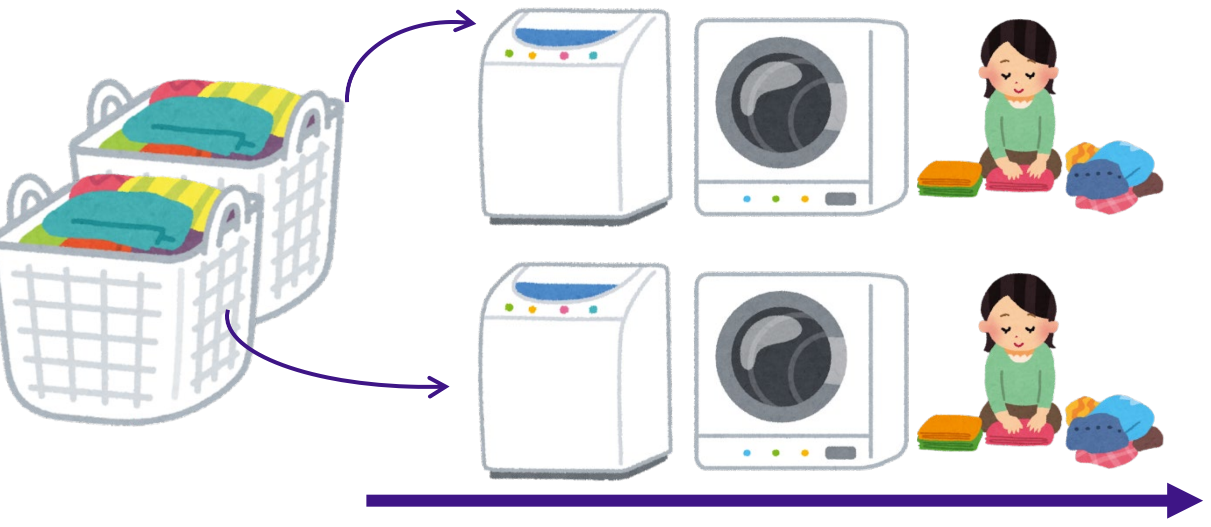


Any idea to finish the laundry work earlier?
In the past, the machine itself (=clock frequency) had become faster every year, but we cannot expect it now…

Laundry (data parallel)

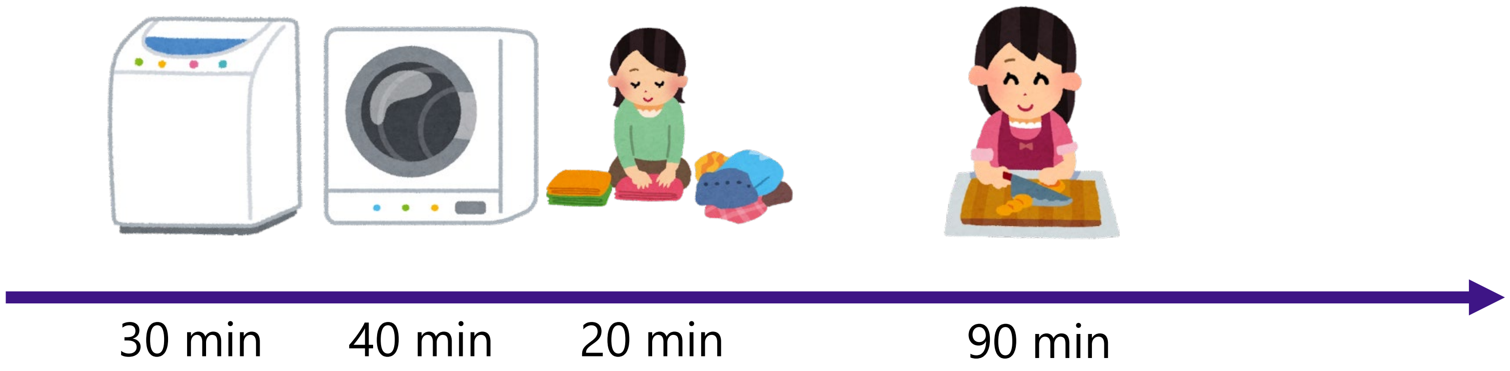


Serial execution takes 180 min for 2 bags.



Data-parallel execution takes 90 min for 2 bags.

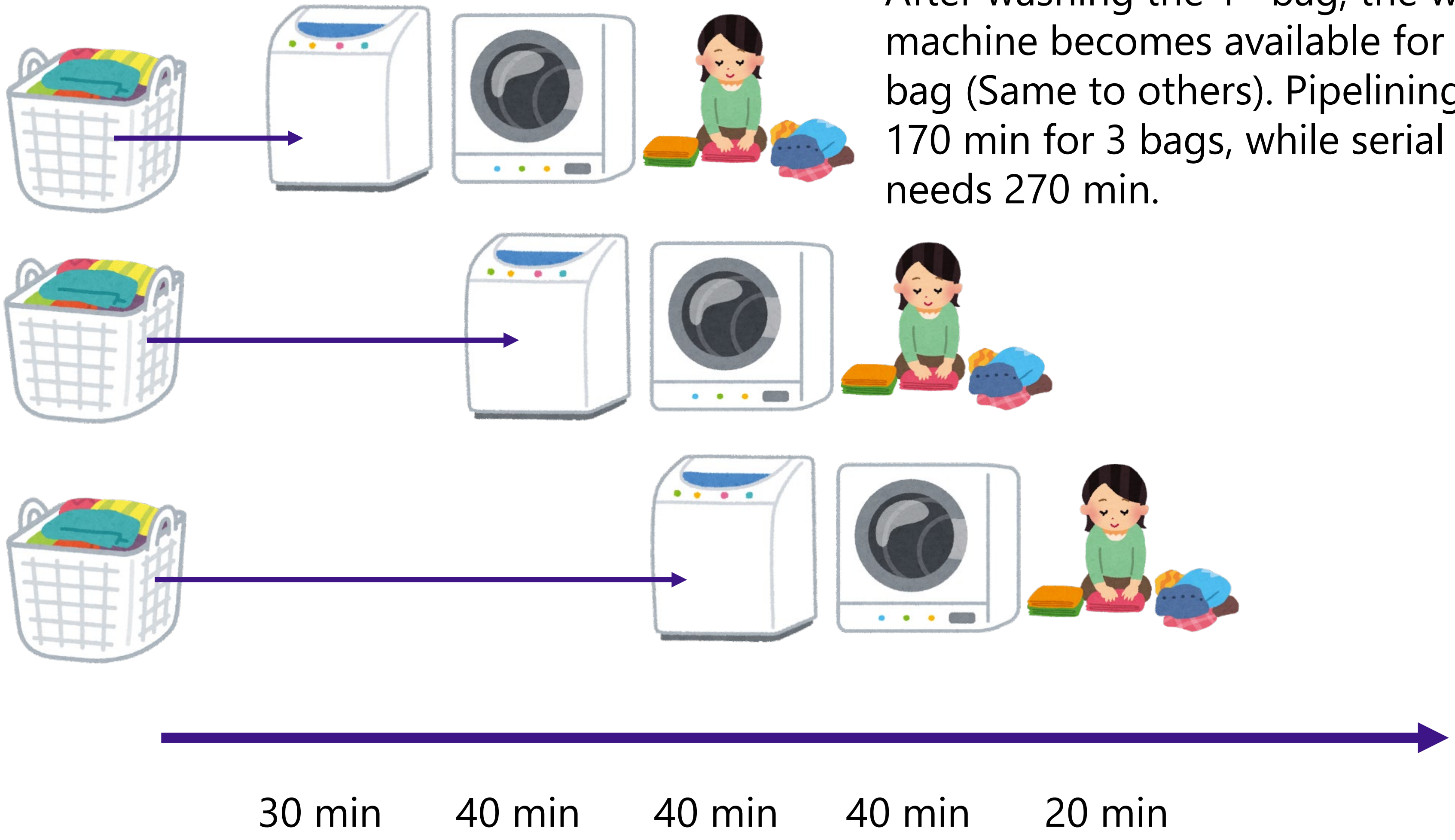
Laundry (task parallel)



Serial execution takes 180 min for laundry and cooking

While one person is doing laundry, another one can do cooking. Independent tasks are done in parallel. Task-parallel execution takes 90 min for doing laundry and cooking.

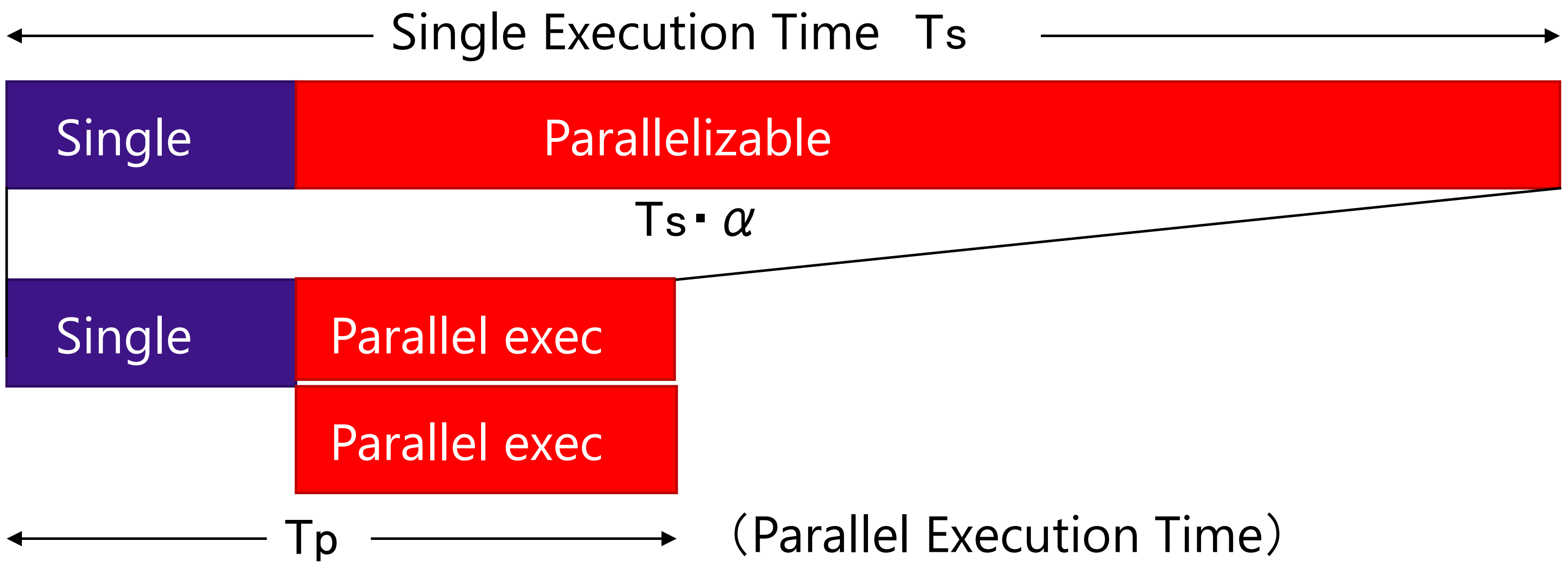
Laundry (Pipelining)



After washing the 1st bag, the washing machine becomes available for the 2nd bag (Same to others). Pipelining needs 170 min for 3 bags, while serial one needs 270 min.

Parallelization Ratio

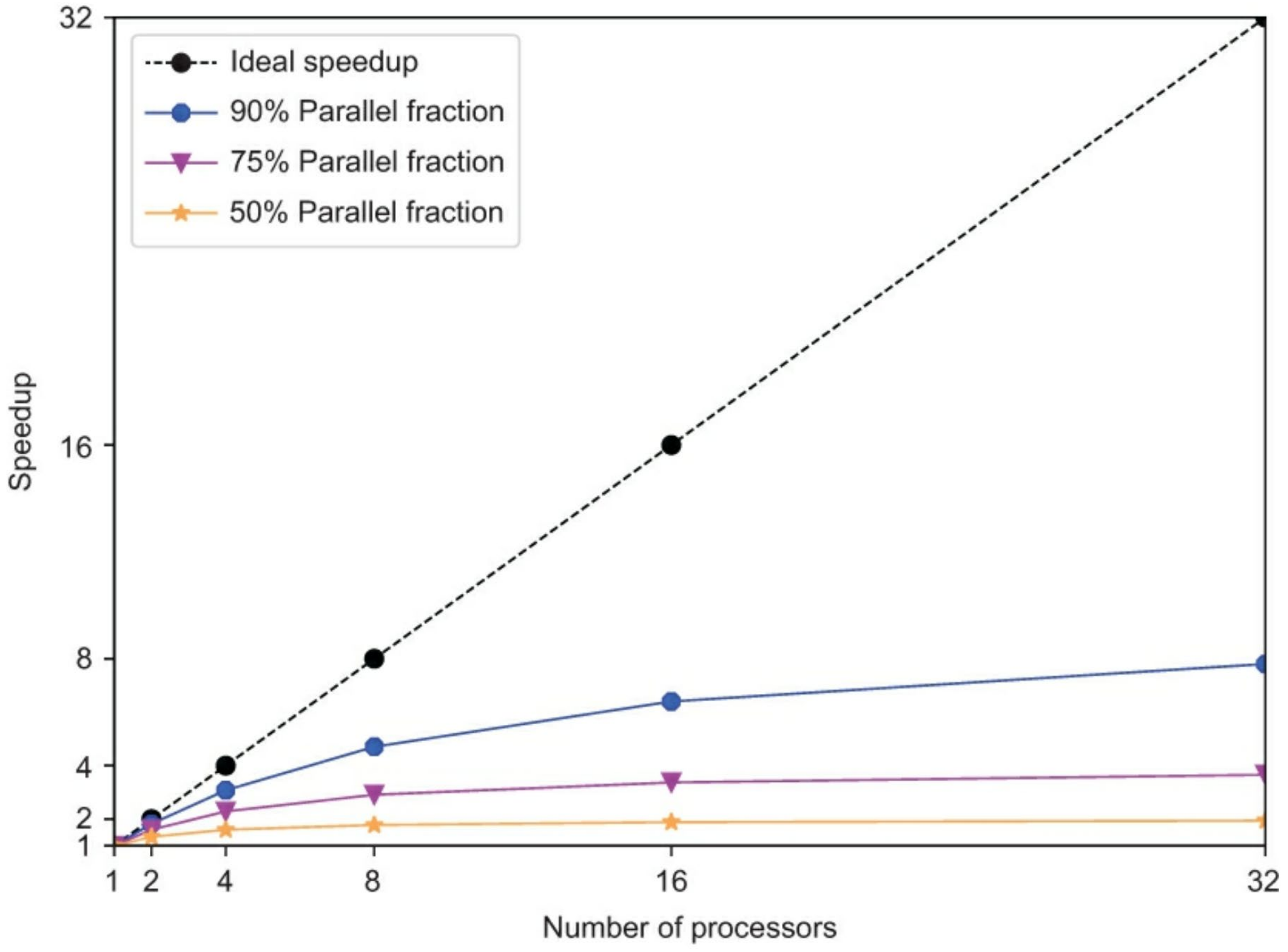
Parallelization Ratio α: The ratio of parallel exec.



$$\text{Speedup ratio} = \frac{Ts}{Tp} = \frac{1}{1-\alpha+\alpha/n} \quad \text{(Amdahl's law)}$$

Increasing n does not mean infinite speedup

Speedup



Cyberscience Center

Parallel Programming

What’s Parallel Programming

■ Parallel Programming

- Programming in a language that allows you to explicitly indicate parallel portions of a program.
  - Those portions can run concurrently on different processors
- Why required?
  - Automatic parallelization is difficult, even though there have been many studies on parallelizing compilers.
- **OpenMP** and **MPI**
  - Standards for expressing thread-level parallelism and process-level parallelism, respectively.

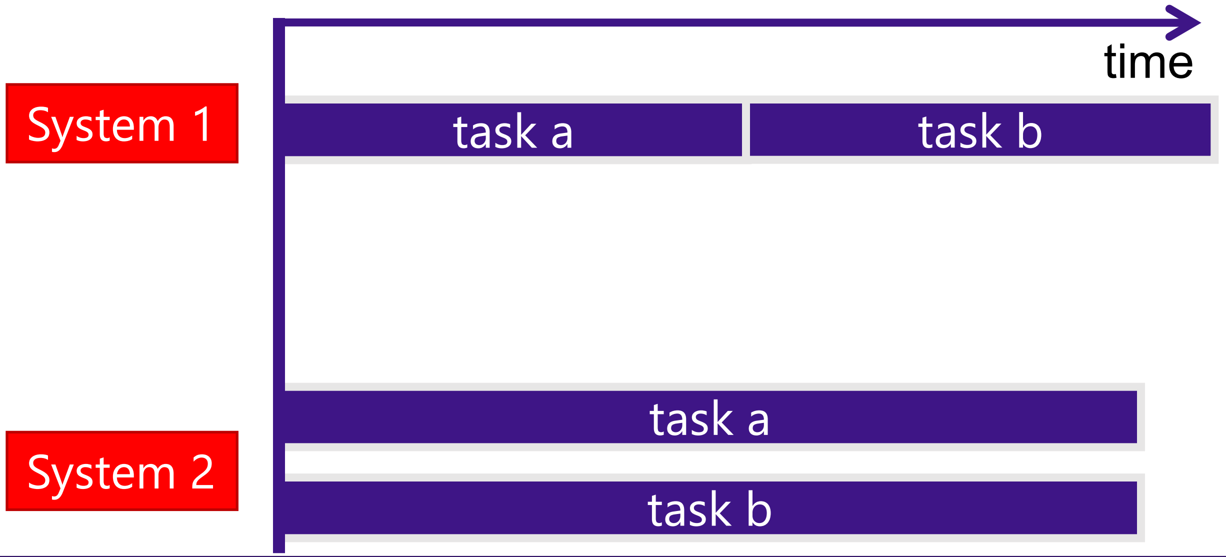
Accelerators

- **Performance characteristics of accelerators are different from those of general-purpose CPUs.**
  - Graphics Processing Unit (GPU)
    - NVIDIA Tesla series
      - Originally developed for 3D image rendering
  - Many-core processors
    - Intel Xeon Phi
      - A lot of simple x86 cores
  - Reconfigurable devices
    - Altair FPGA, Xilinx FPGA
      - Programmable circuits

We have to exploit both **parallelism** and **heterogeneity**

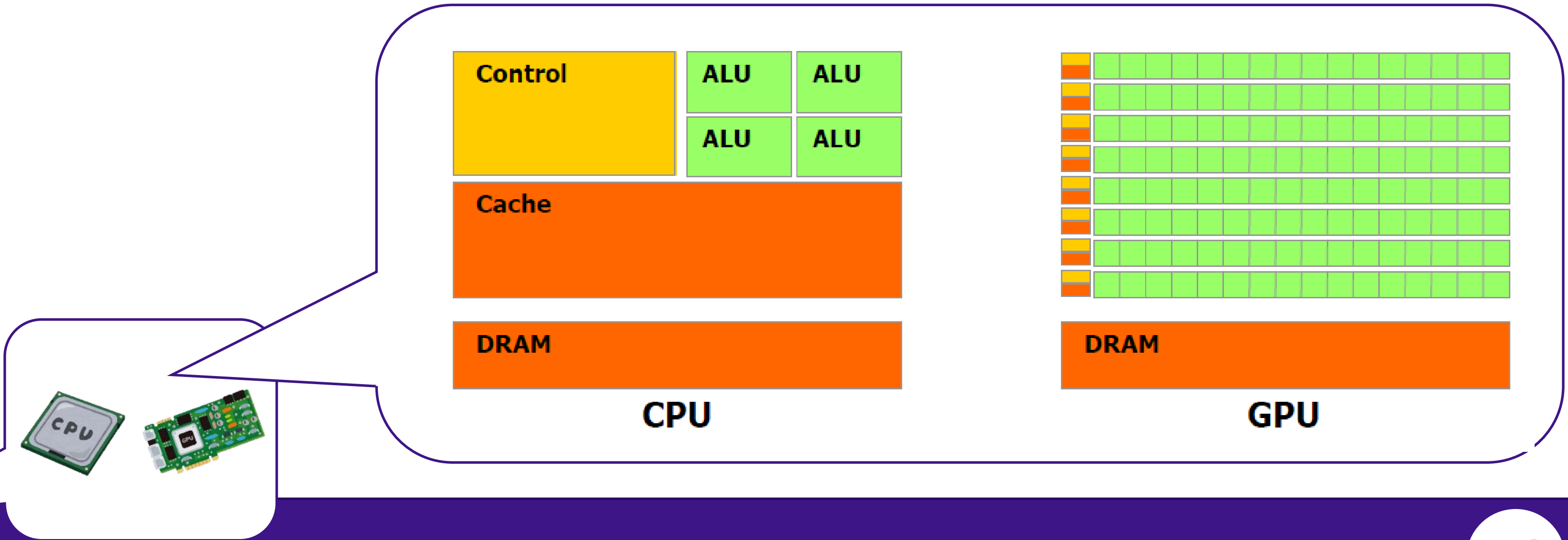
System performance metrics

- **Latency**
  - The execution time for a task (shorter is better).
    - System 1 is better in terms of latency.
- **Throughput**
  - The number of tasks per unit time (more is better).
    - System 2 is better in terms of throughput.



Heterogenous Computing

- **CPU and GPU are very different processors.**
  - Latency-oriented design (=speculative)
    - CPU has a large cache memory and control unit.
  - Throughput-oriented design (=parallel)
    - GPUs devote more hardware resources to ALUs.



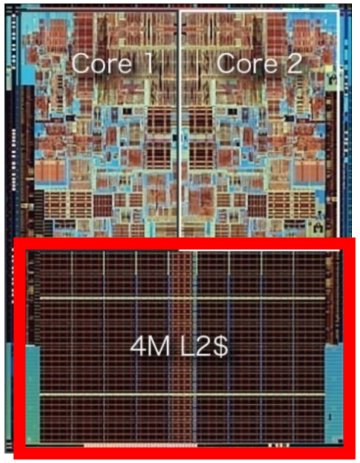
Latency-oriented features

- **Get ready for future events!**
  - Cache memory
    - hold data that are likely accessed in the future.
  - OoO execution and branch prediction
    - predict the branch target for speculative execution.

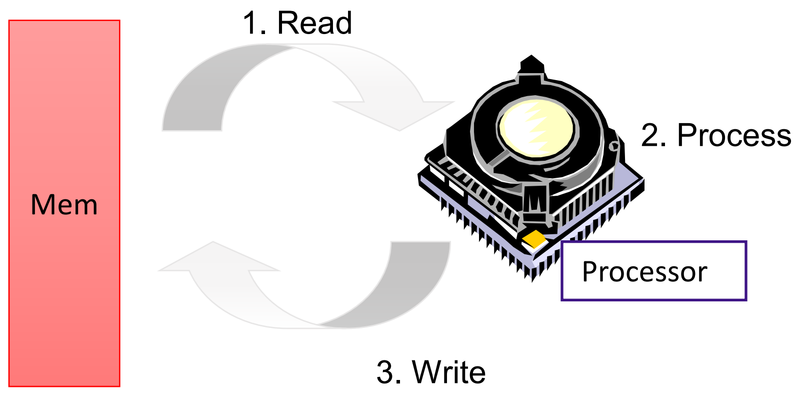
Speculation is one of key technologies in CPU.

- But it needs more hardware resources spent for other than ALU.

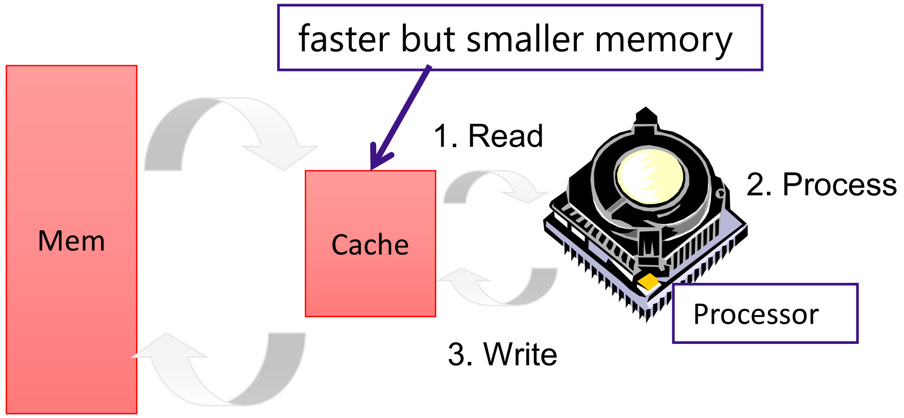
Cache Memory



- **Memory is much slower than CPU**
  - Prepare for the future memory accesses



The processing speed is limited by the memory speed.



The processing speed is limited by the cache speed if data are kept in the cache.

The data to be cached are determined based on the locality of reference.

The cache hit ratio is very important for modern processors to achieve high performance. To increase the cache hit ratio, cache memory occupies quite a large area of the chip.

Cyberscience Center

Branch Prediction

If we can predict the branch target, we can speculatively execute either A or B.

Conditional branch

```
if ( i == 0 )
{
  A;
}
B;
```

Loop

```
for ( i = 0; i < N; i++ )
{
  A;
}
B;
```

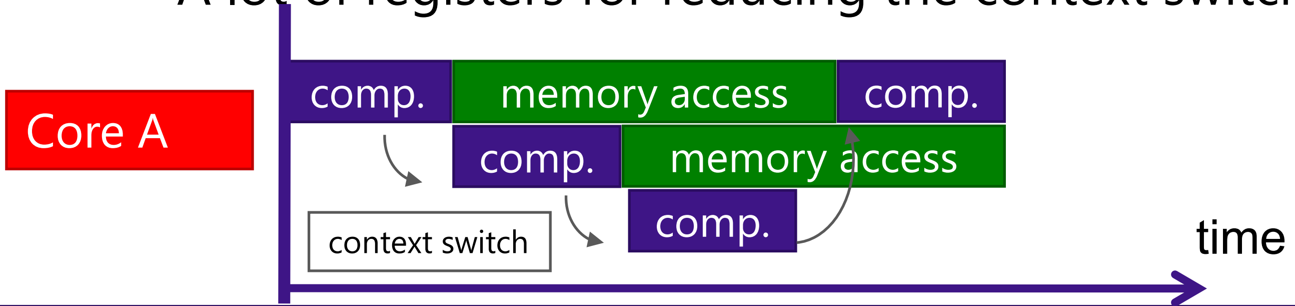
A is executed N times and then B is executed

There is usually a statistical bias in the branch targets.

Speculative execution by branch prediction is effective to shorten the latency by preventing pipeline stalls, and can be achieved based on the branch history. But it makes the datapath more complicated and significantly increases the hardware cost of each core.

Strategies for High Throughput

- **Many simple cores**
  - No speculation features
    - Simplicity to increase the number of cores on a chip
  - Many tasks (threads) are executed in parallel.
    - The execution of an individual thread may be slow.
- **Many in-flight threads**
  - When a thread is stalled, another thread is executed to keep every core working (context switch).
    - Memory access latency is hidden by other thread execution.
    - A lot of registers for reducing the context switch overhead.



Summary

- **Introduction to HPC (Supercomputing)**
  - What parallel computing is and why it’s growing in importance.
    - Scientific computing and emerging AI and ML
  - Where parallelism exists in modern hardware.
    - Modern HPC systems use parallelism at various levels
  - Why the amount of application parallelism is important.
    - An application needs to be parallelized well to exploit the system parallelism.
  - Software approaches to exploit parallelism.
    - OpenMP – thread-level parallelism
    - MPI – node-level (process-level) parallelism