FACULDADE DE
CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Parallel Programming Models and Architectures

lecture 01 (2025-03-10)

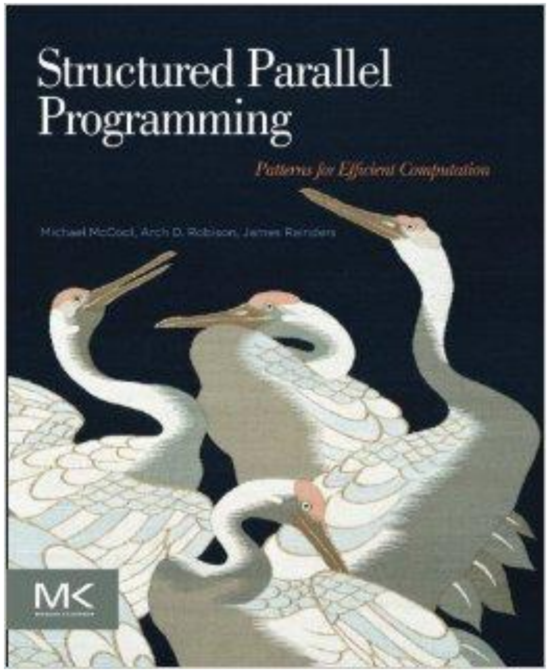
Master in Computer Science and Engineering

— Concurrency and Parallelism / 2024-25 —

João Lourenço <joao.lourenco@fct.unl.pt>

Outline

- Parallel Programming Models
- Parallel Architectures



  – Bibliography:
    - **Chapters 1 and 2** of book

      McCool M., Arch M., Reinders J.;
      Structured Parallel Programming:
      Patterns for Efficient Computation;
      Morgan Kaufmann (2012);
      ISBN: 978-0-12-415993-8

Why Concurrency & Parallelism?

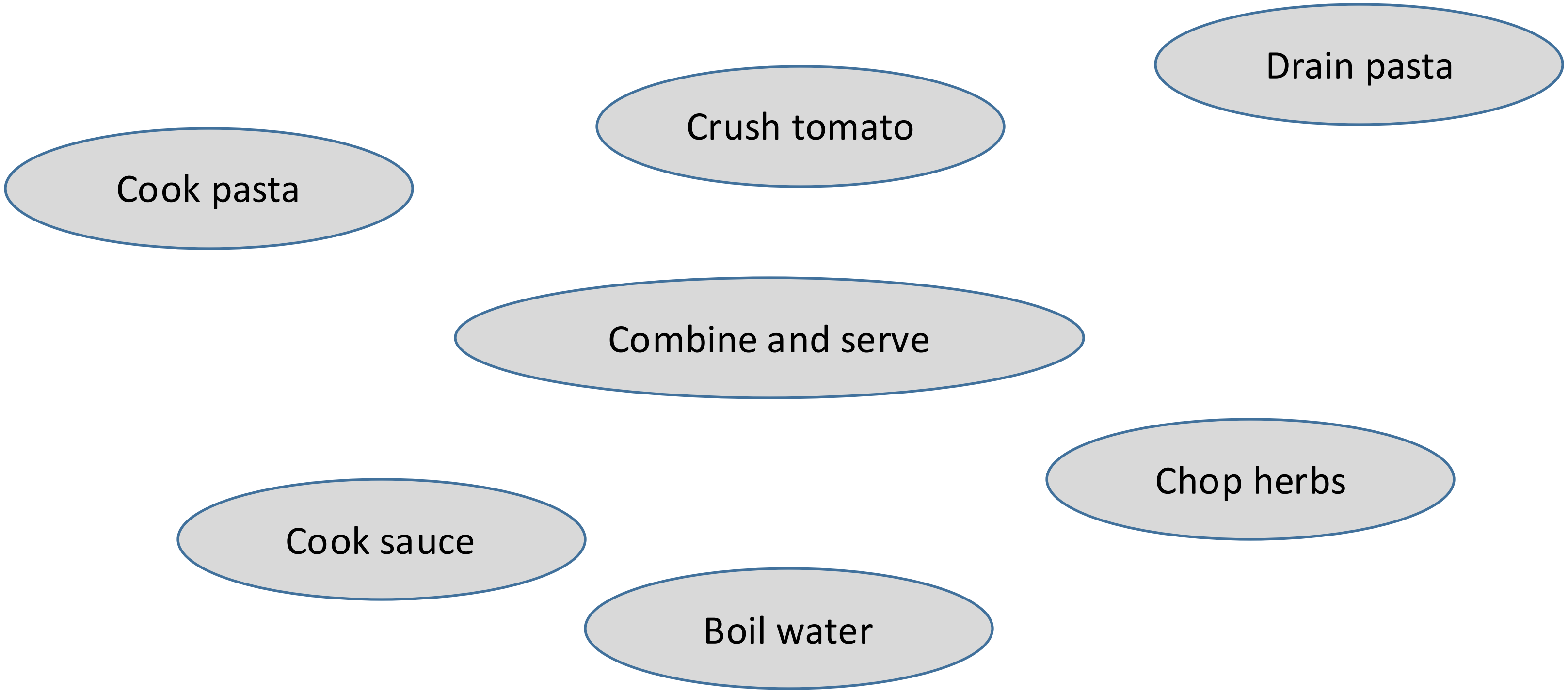
1. **Efficient Resource Utilization**
   - Modern CPUs have multiple cores; Concurrent/parallel programs make better use of CPU cores, GPUs, and distributed systems.
2. **Responsiveness in UI & Applications**
   - Concurrency is crucial for applications requiring real-time interactions (e.g., mobile apps, games, real-time analytics).
   - Prevents UI freezing when executing background operations (e.g., downloading files, fetching API data).
3. **Emerging Technologies Rely on It**
   - AI, ML, blockchain, and IoT demand parallel processing for real-time computations.
   - Edge computing and real-time analytics rely on concurrency for quick decision-making.
4. **Performance and Speedup**
   - Parallelism allows multiple computations to run simultaneously, reducing execution time for large tasks (e.g., data processing, simulations, AI training).
5. **Scalability in Distributed Systems**
   - Cloud computing, microservices, and large-scale data processing rely on concurrency and parallelism to scale efficiently.
   - Technologies like Kubernetes, Apache Spark, and serverless architectures leverage parallel execution.
6. **Better Software Design**
   - Understanding concurrency helps in designing correct thread-safe applications.
   - Avoids issues like race conditions, deadlocks, and inconsistent states.

Remember…

Even if you're not building high-performance applications today, understanding concurrency and parallelism makes you a better software engineer, helping you write efficient, scalable, and future-proof software.

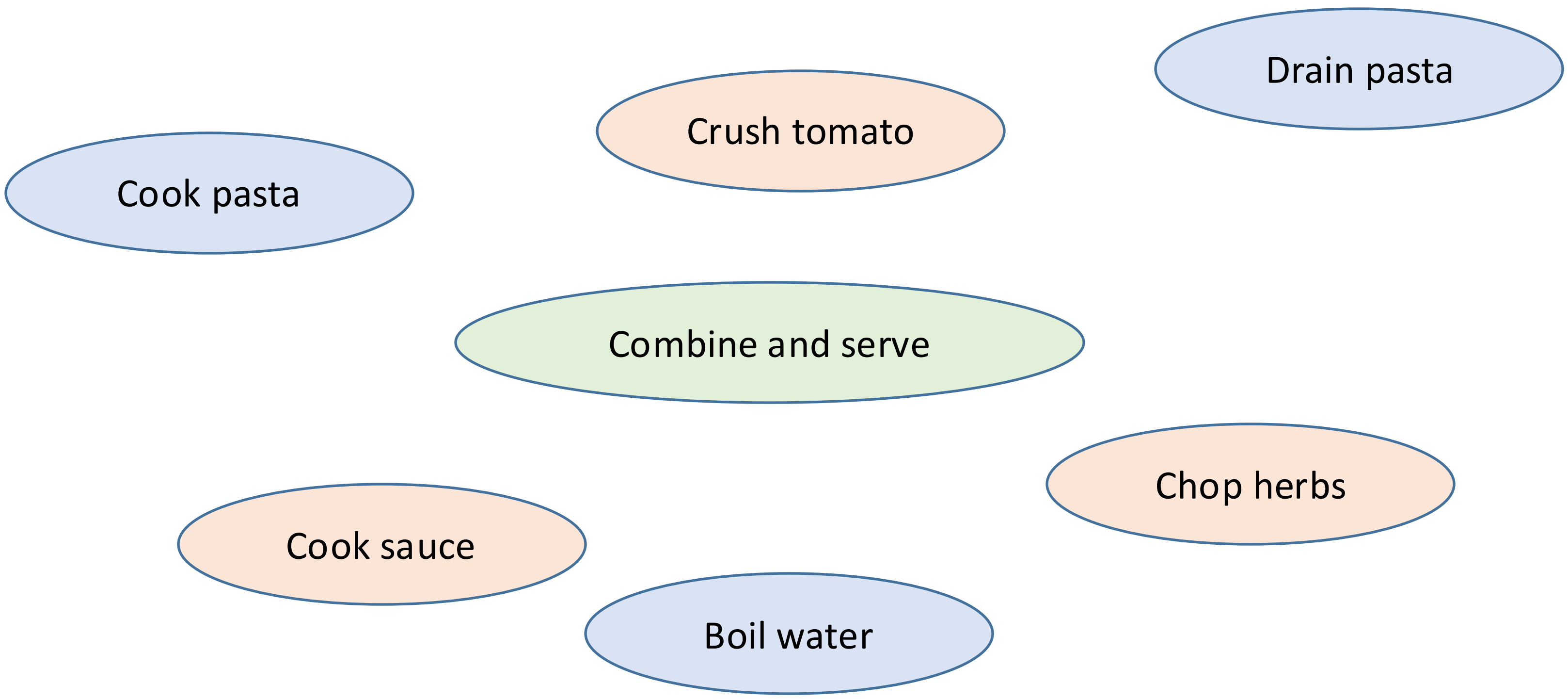
Concurrency & Parallel Computing

- Let's prepare a pasta



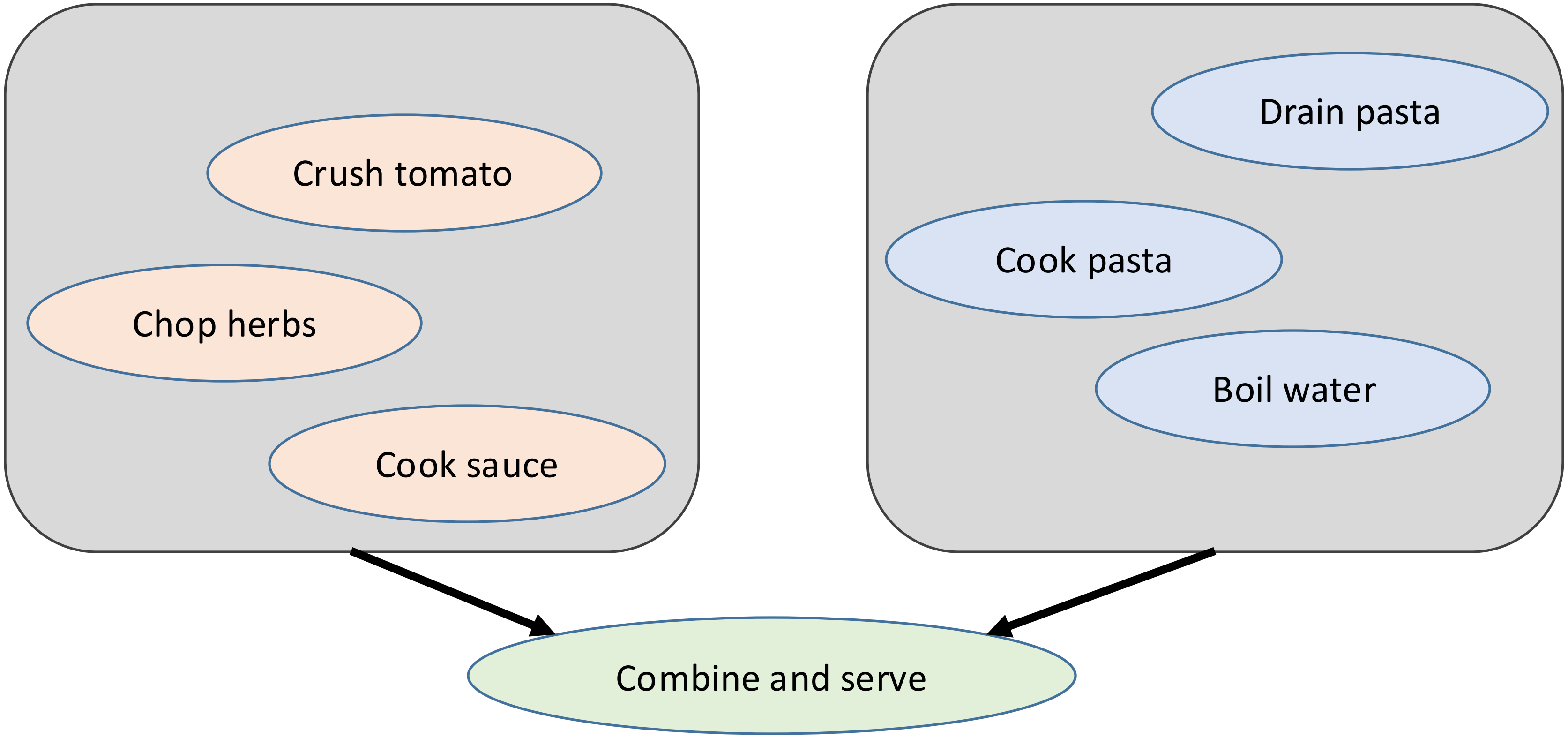
Concurrency & Parallel Computing

- Let’s prepare a pasta



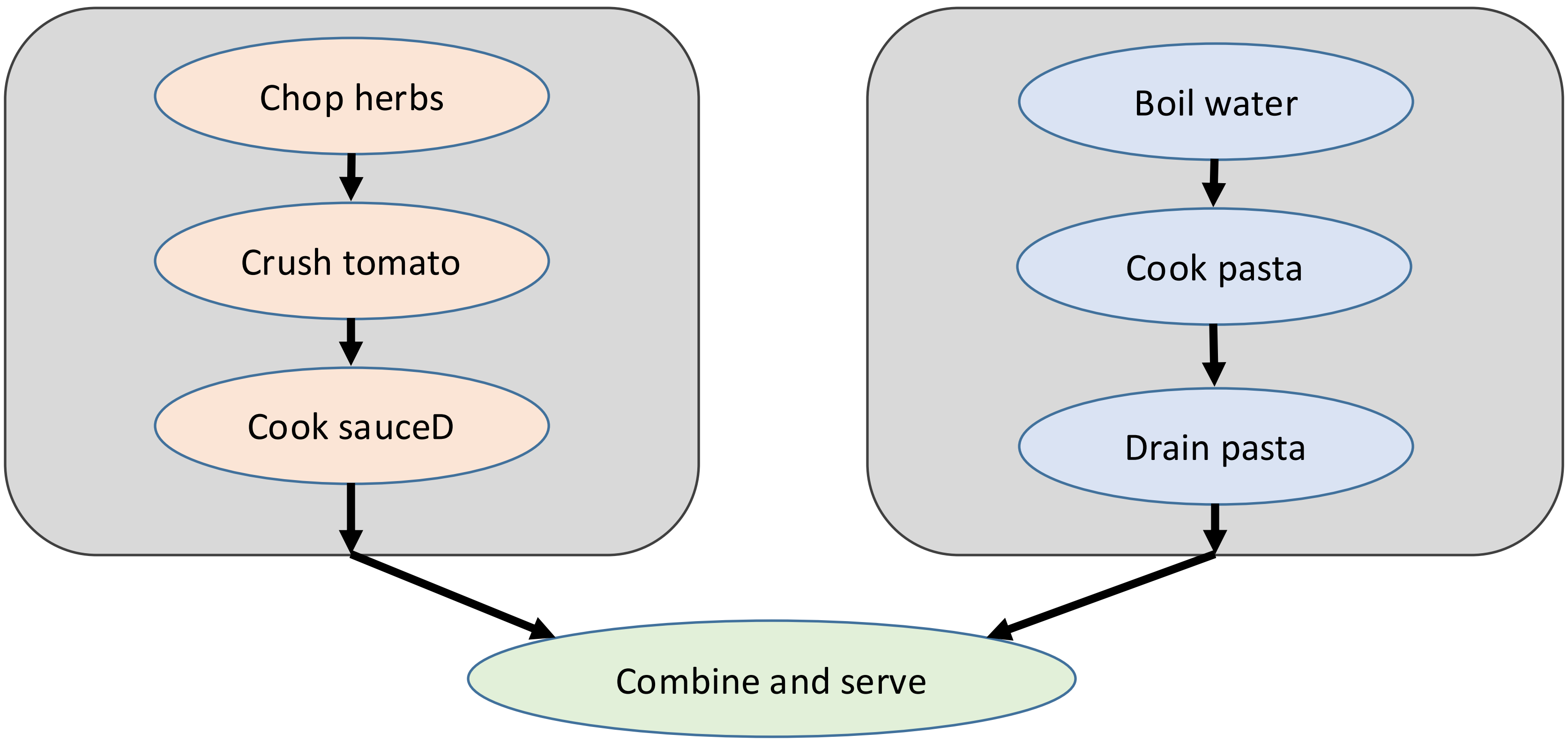
Concurrency & Parallel Computing

- Let's prepare a pasta



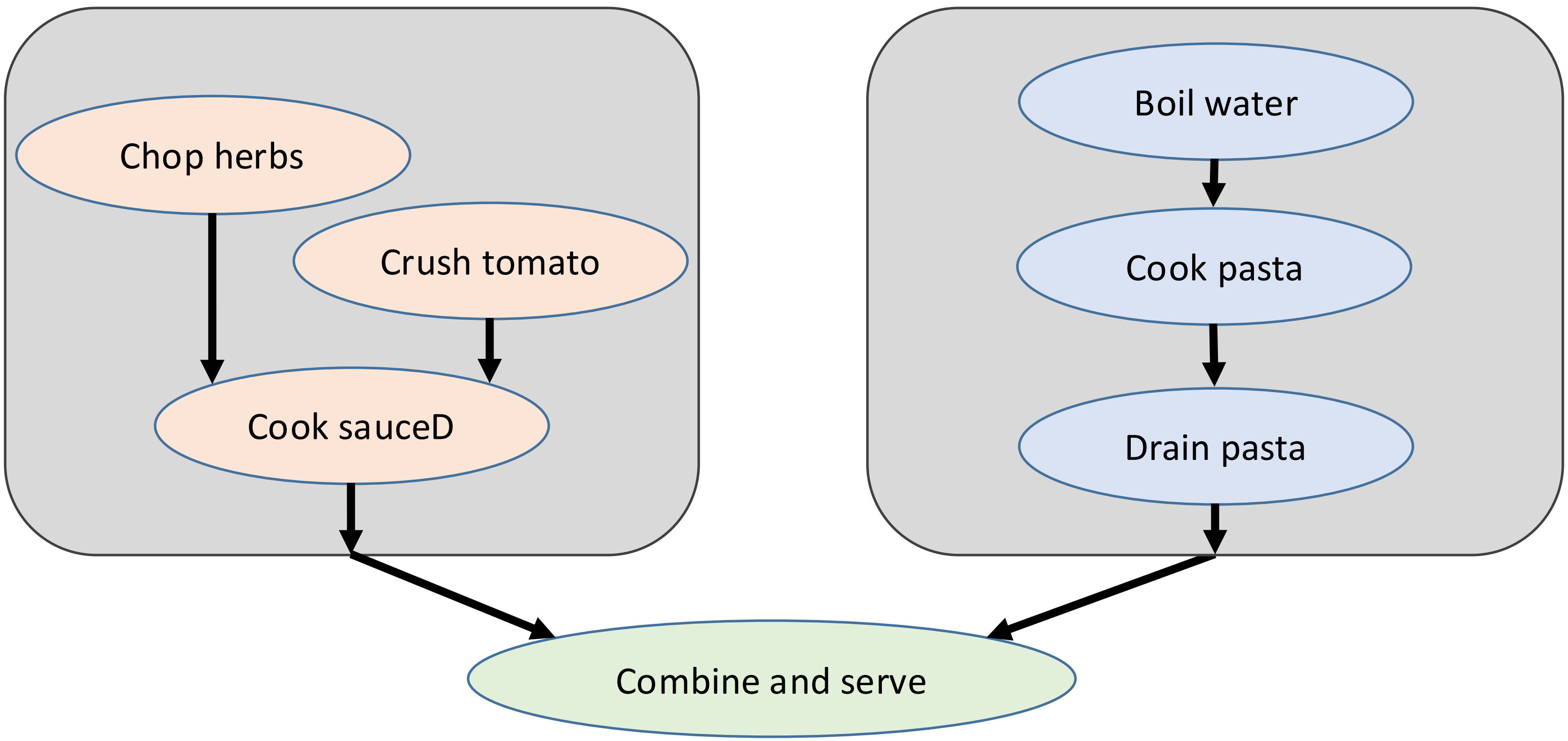
Concurrency & Parallel Computing

- Let’s prepare a pasta



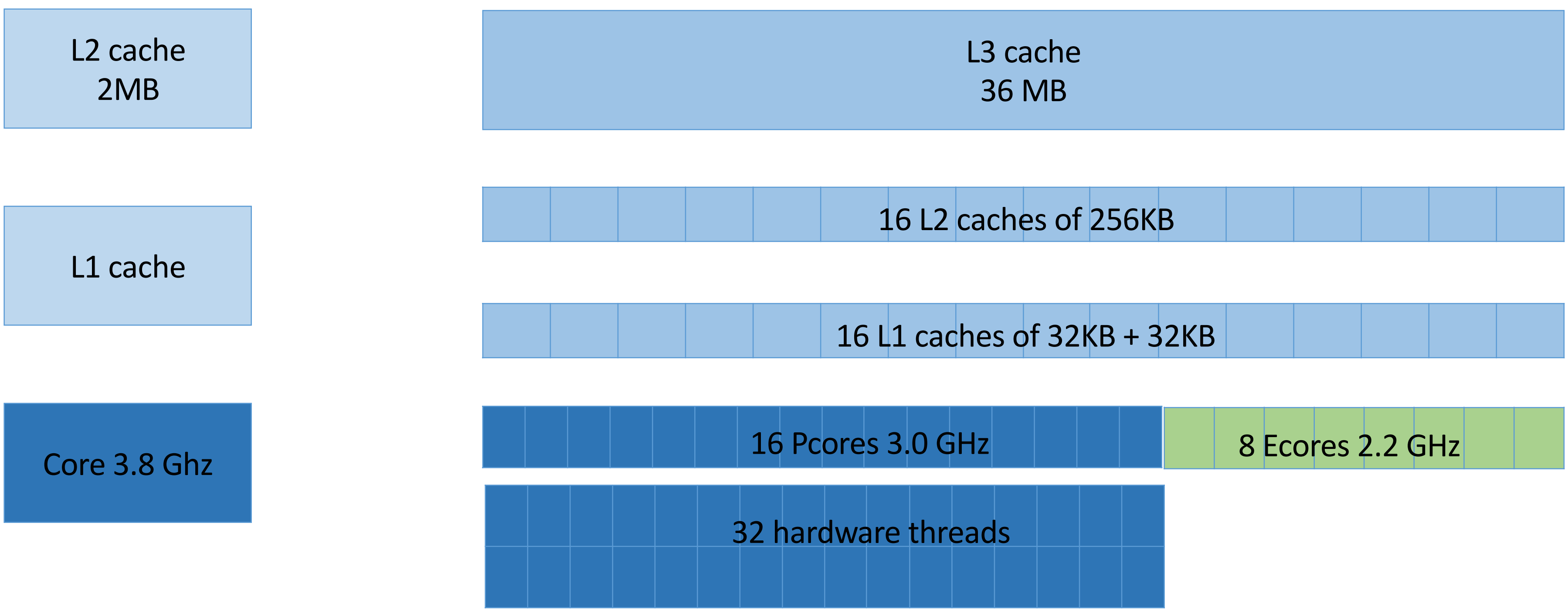
Concurrency & Parallel Computing

- Let's prepare a pasta



Reason 1: Modern Processor Architecture

Intel Xeon (2005) **Intel® Core™ i9-13900K (2023)**



Reason 2: Software Requirements

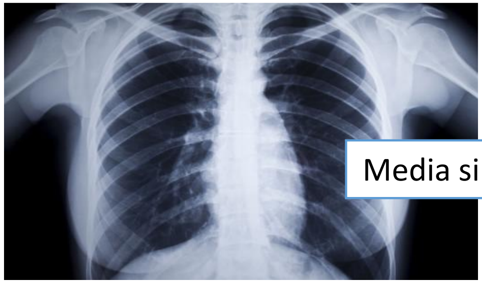
Complex image rendering

Space search projects

- Highly **complex problems**

and/or

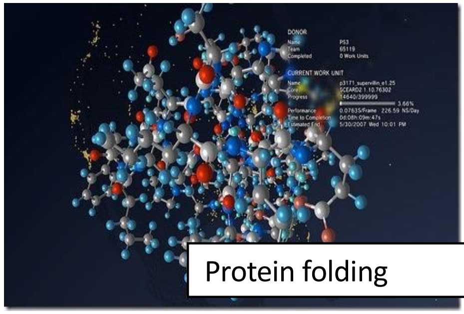
- **Lots of data** to process



Media similarity

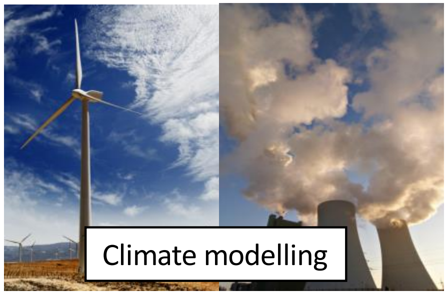
Weather forecast

Resource use optimization, such as fuel

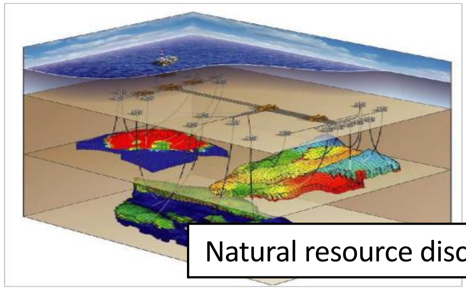


Protein folding

Drug research



Climate modelling



Natural resource discovery

Parallel Architectures

- Flynn's taxonomy

	Single Data	Multiple Data
Single Instruction	SISD	SIMD
Multiple Instruction	MISD	MIMD

Flynn's Taxonomy

- Single Instruction, Single Data (SISD) architecture
  - A hardware thread in a modern CPU
  - May implement several kinds of instruction level parallelism

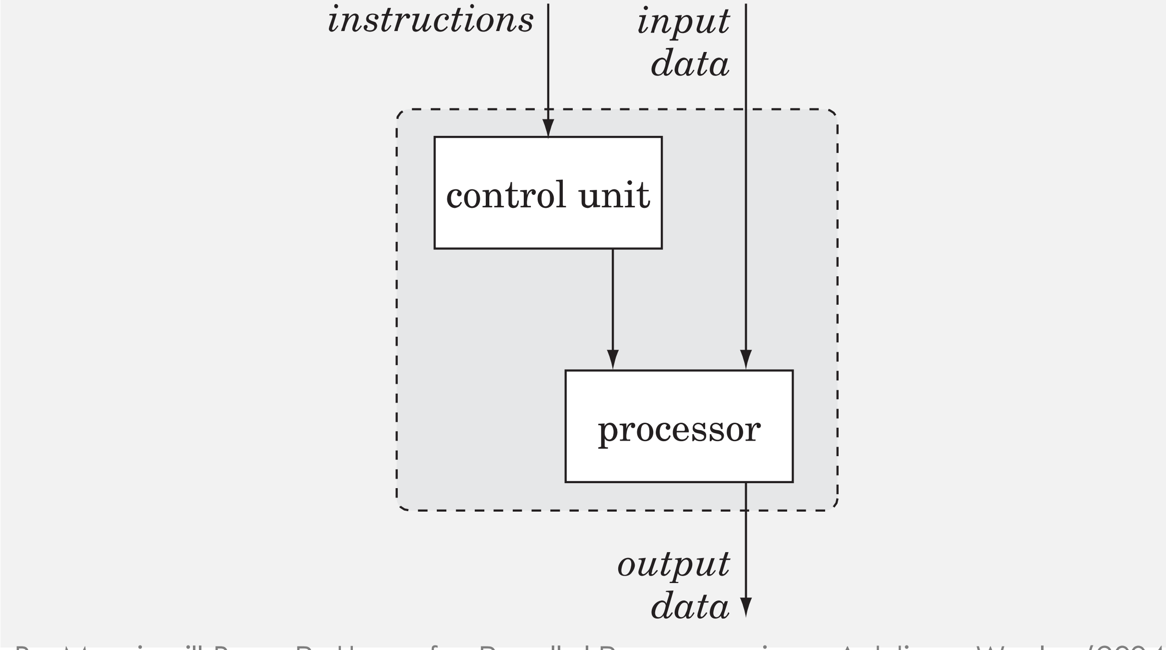


Image from: Mattson T., Sanders B., Massingill B.; Patterns for Parallel Programming; Addison-Wesley (2004).

Flynn's Taxonomy

- Single Instruction, Multiple Data (SIMD) architect
  - Available in the processor’s instruction set
  - Available in all GPUs and other specialized hardware

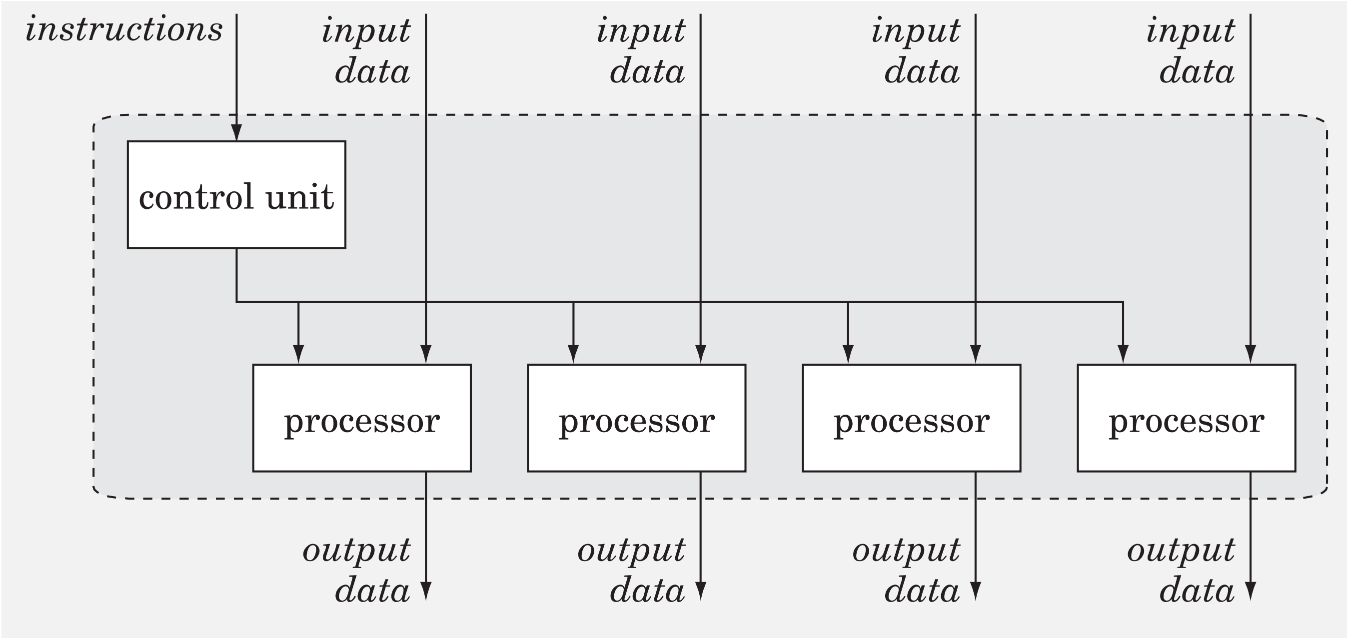
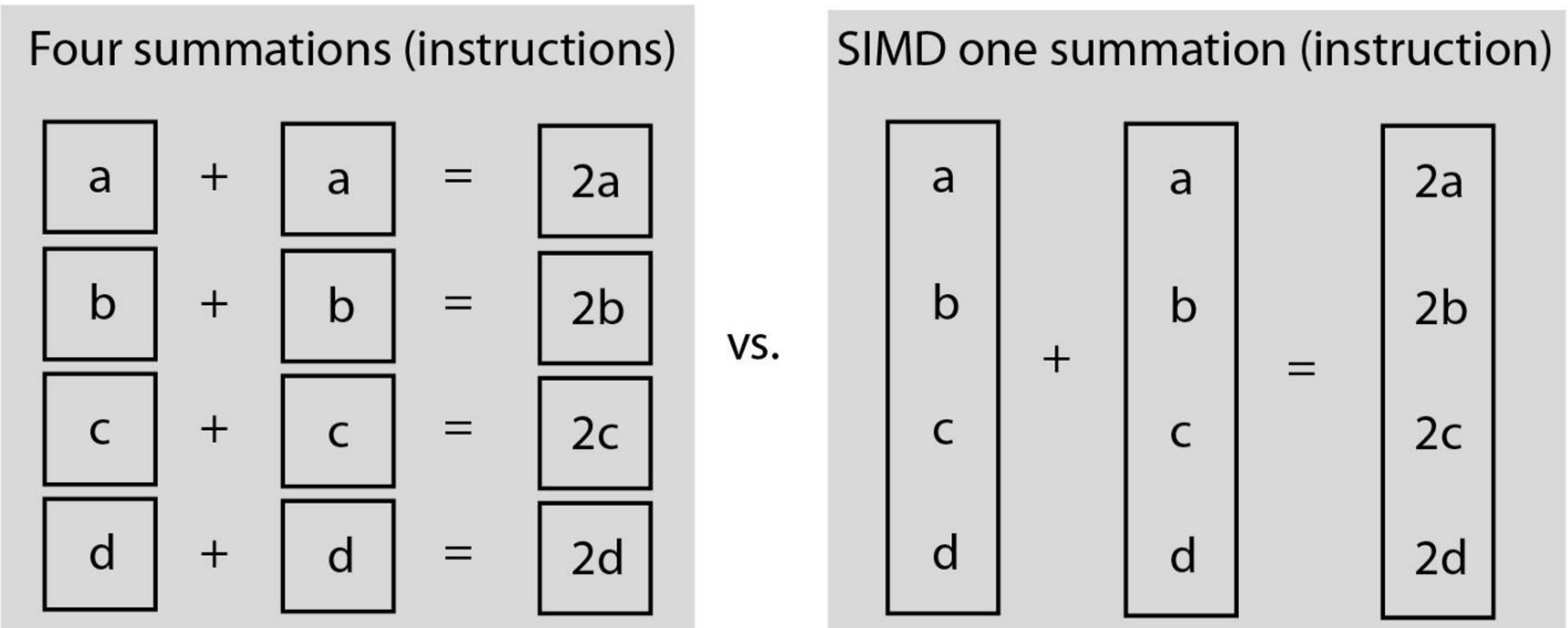
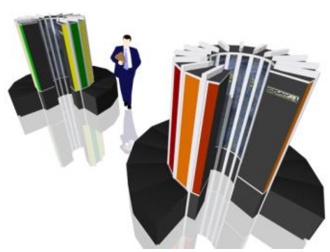


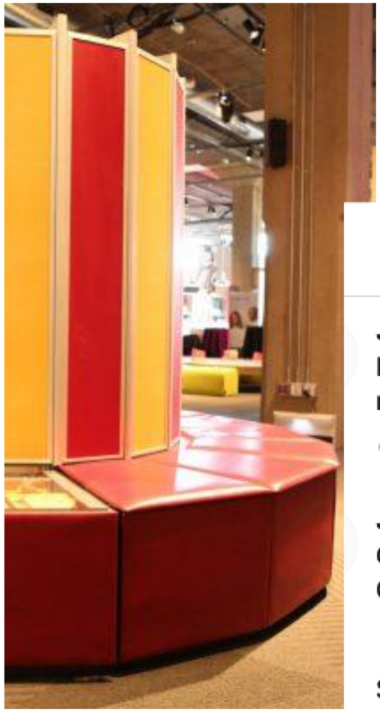
Image from: Mattson T., Sanders B., Massingill B.; Patterns for Parallel Programming; Addison-Wesley (2004).

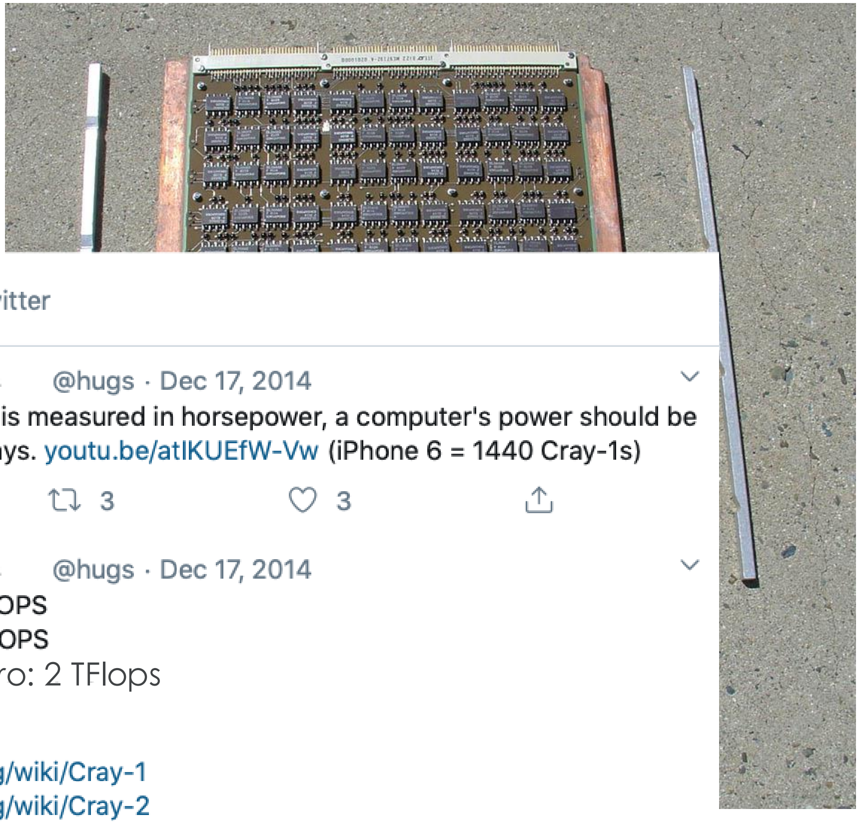
SISD vs. SIMD - example



CRAY-1 Vector Machine (1976)

Cray-1	
	
3D rendering of two Cray-1 with a figure as scale	
Design	
Manufacturer	Cray Research
Designer	Seymour Cray
Release date	1975
Units sold	Over 80
Price	US$7.9 million in 1977 (equivalent to $33.3 million in 2019)
Casing	
Dimensions	Height: 196 cm (77 in)[1] Dia. (base): 263 cm (104 in)[1] Dia. (columns): 145 cm (57 in)[1]
Weight	5.5 tons (Cray-1A)
Power	115 kW @ 208 V 400 Hz[1]
System	
Front-end	Data General Eclipse
Operating system	COS & UNICOS
CPU	64-bit processor @ 80 MHz[1]
Memory	8.39 Megabytes (up to 1 048 576 words)[1]
Storage	303 Megabytes (DD19 Unit)[1]
FLOPS	160 MFLOPS
Successor	Cray X-MP





Search Twitter

Jason Huggins @hugs · Dec 17, 2014
If a car's power is measured in horsepower, a computer's power should be measured in crays. youtu.be/atIKUEfW-Vw (iPhone 6 = 1440 Cray-1s)

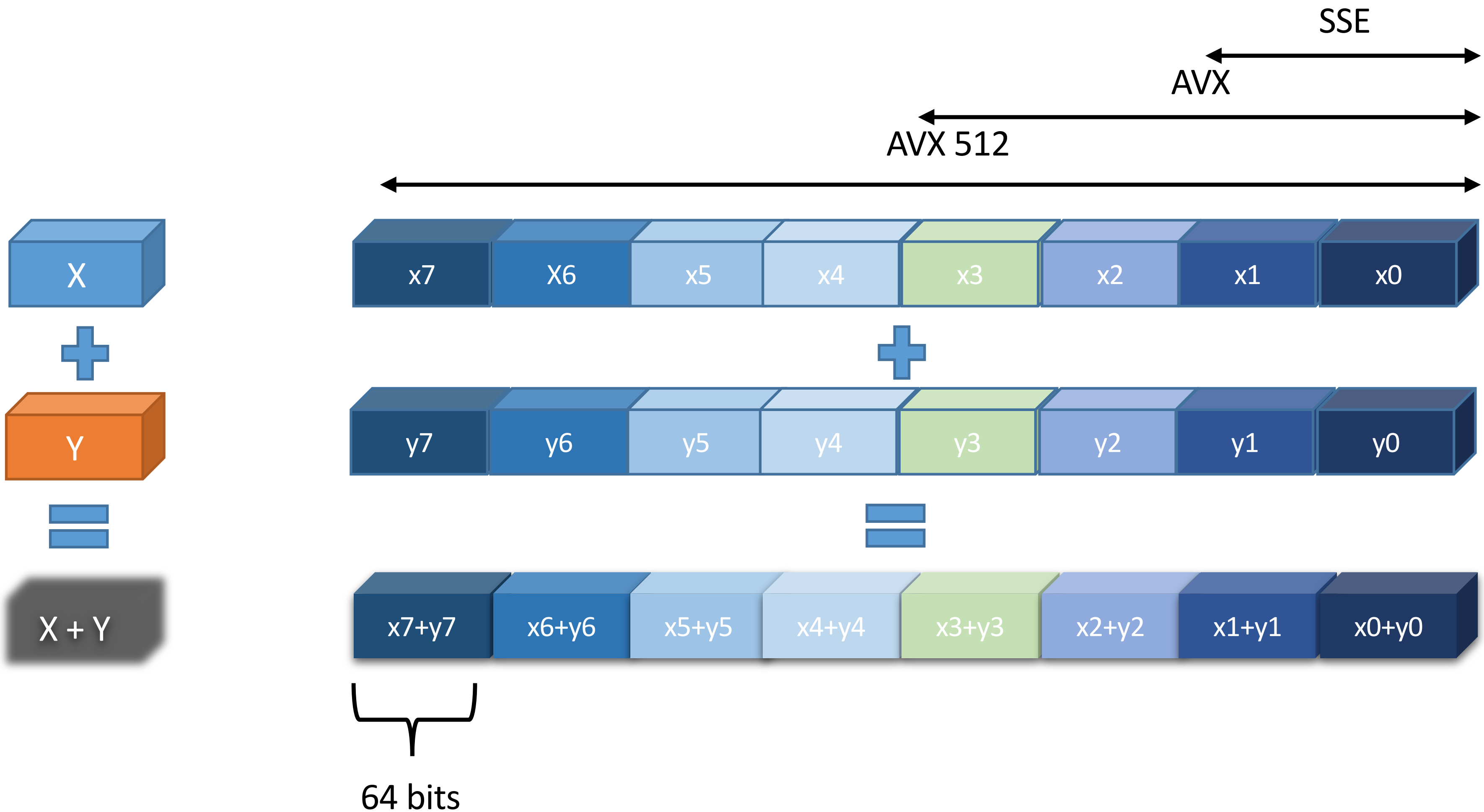
2 3 3

Jason Huggins @hugs · Dec 17, 2014
Cray-1: 80 MFLOPS
Cray-2: 1.9 GFLOPS
Iphone 14 Pro: 2 TFlops

Sources:
en.wikipedia.org/wiki/Cray-1
en.wikipedia.org/wiki/Cray-2
en.wikipedia.org/wiki/Apple_sys...

3 6 7

Vector Processing Today: Intel AVX



Vector Machines Today: GPUs

NVIDIA GeForce RTX 5090

GB202	21760	680	176	32 GB	GDDR7	512 bit
GRAPHICS PROCESSOR	CORES	TMUS	ROPS	MEMORY SIZE	MEMORY TYPE	BUS WIDTH

Theoretical Performance	
Pixel Rate:	423.6 GPixel/s
Texture Rate:	1,637 GTexel/s
FP16 (half):	104.8 TFLOPS (1:1)
FP32 (float):	104.8 TFLOPS
FP64 (double):	1.637 TFLOPS (1:64)

Flynn's Taxonomy

- Multiple Instruction, Single Data (MISD)

Flynn's Taxonomy

- Multiple Instruction, Multiple Data (MIMD) archit.
  - Multicore processor
  - Cluster of computers (MPI, Spark, …)

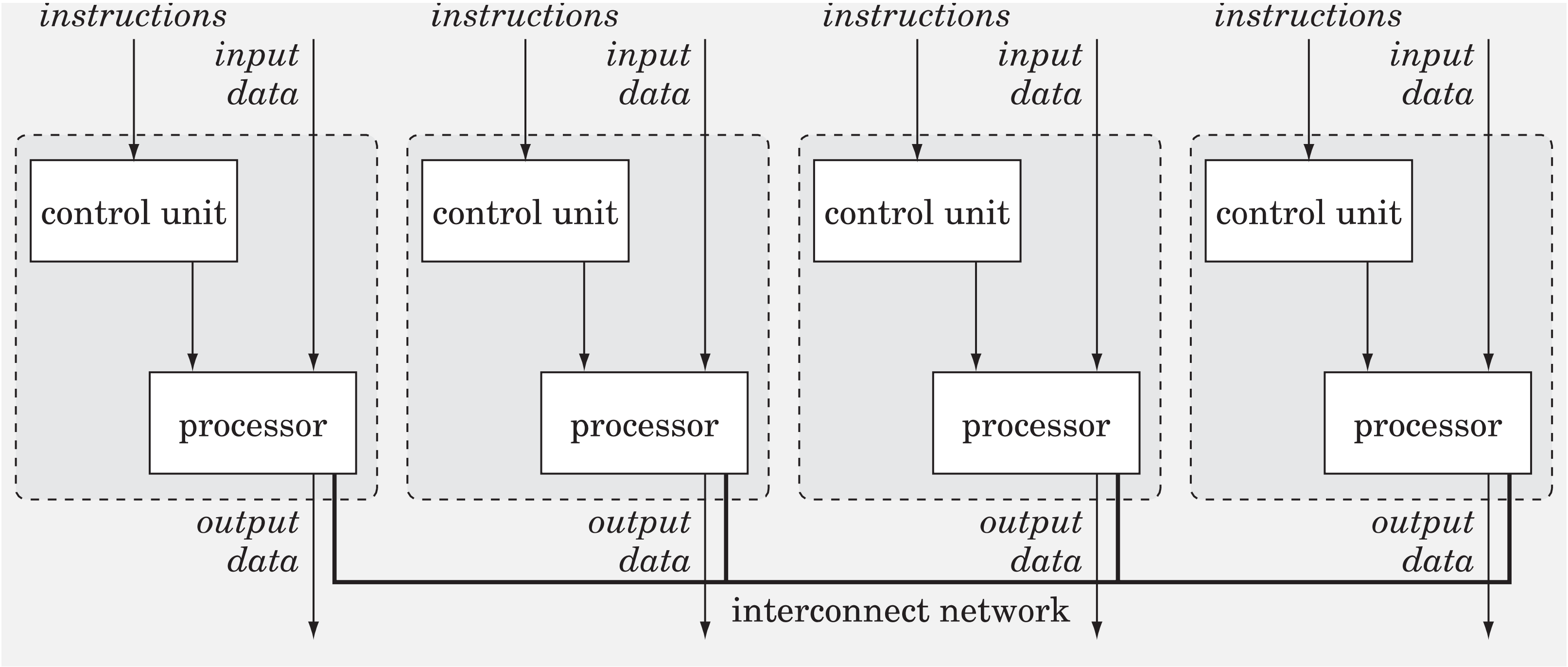


Image from: Mattson T., Sanders B., Massingill B.; Patterns for Parallel Programming; Addison-Wesley (2004).

Parallel Architectures — MIMD Shared Memory Architectures

- The Symmetric Multiprocessor (SMP) architecture
  - Typical personal computers, phones, TVs, …

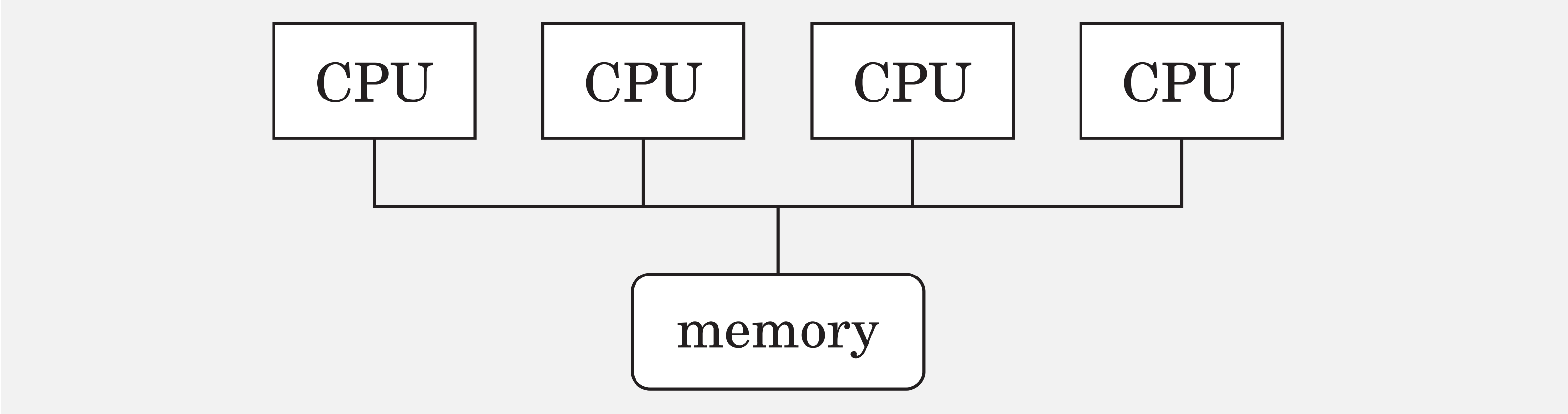
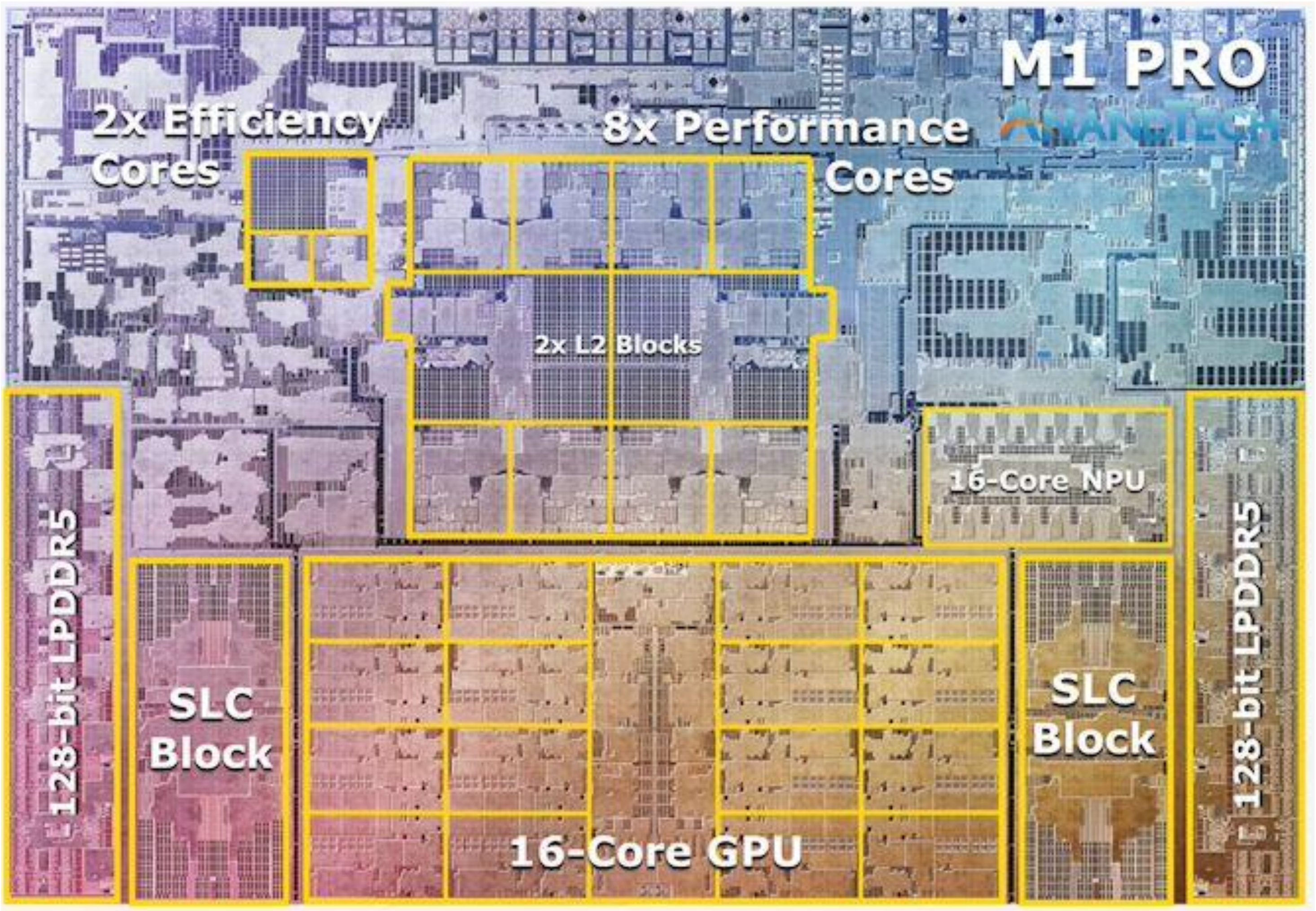


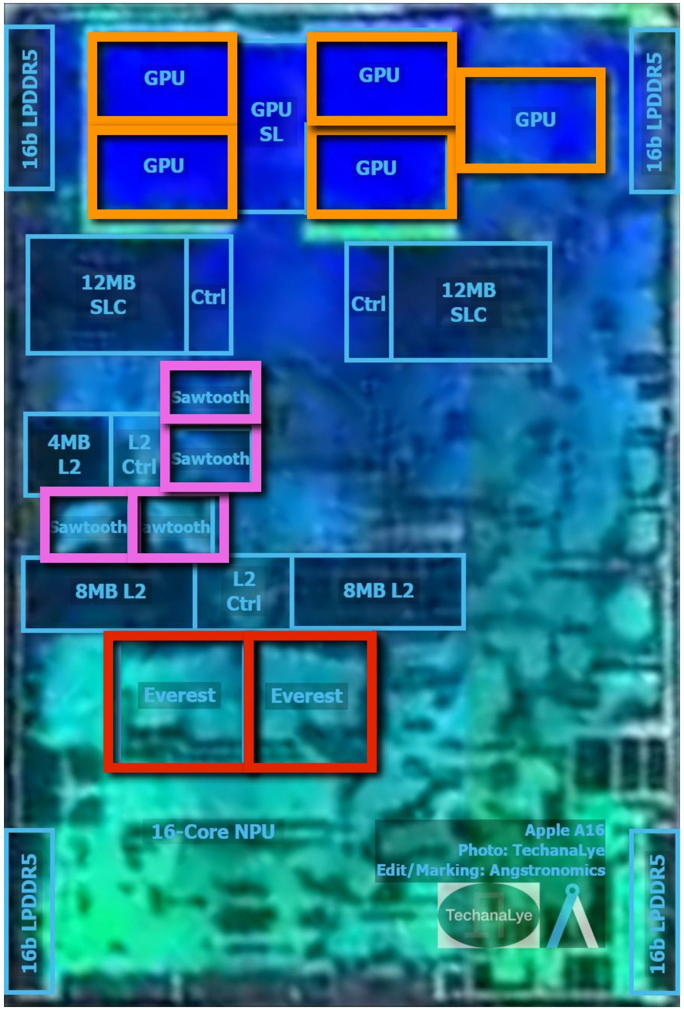
Image from: Mattson T., Sanders B., Massingill B.; Patterns for Parallel Programming; Addison-Wesley (2004).

Intel Alder Lake-S (2021)

Apple M1 Pro

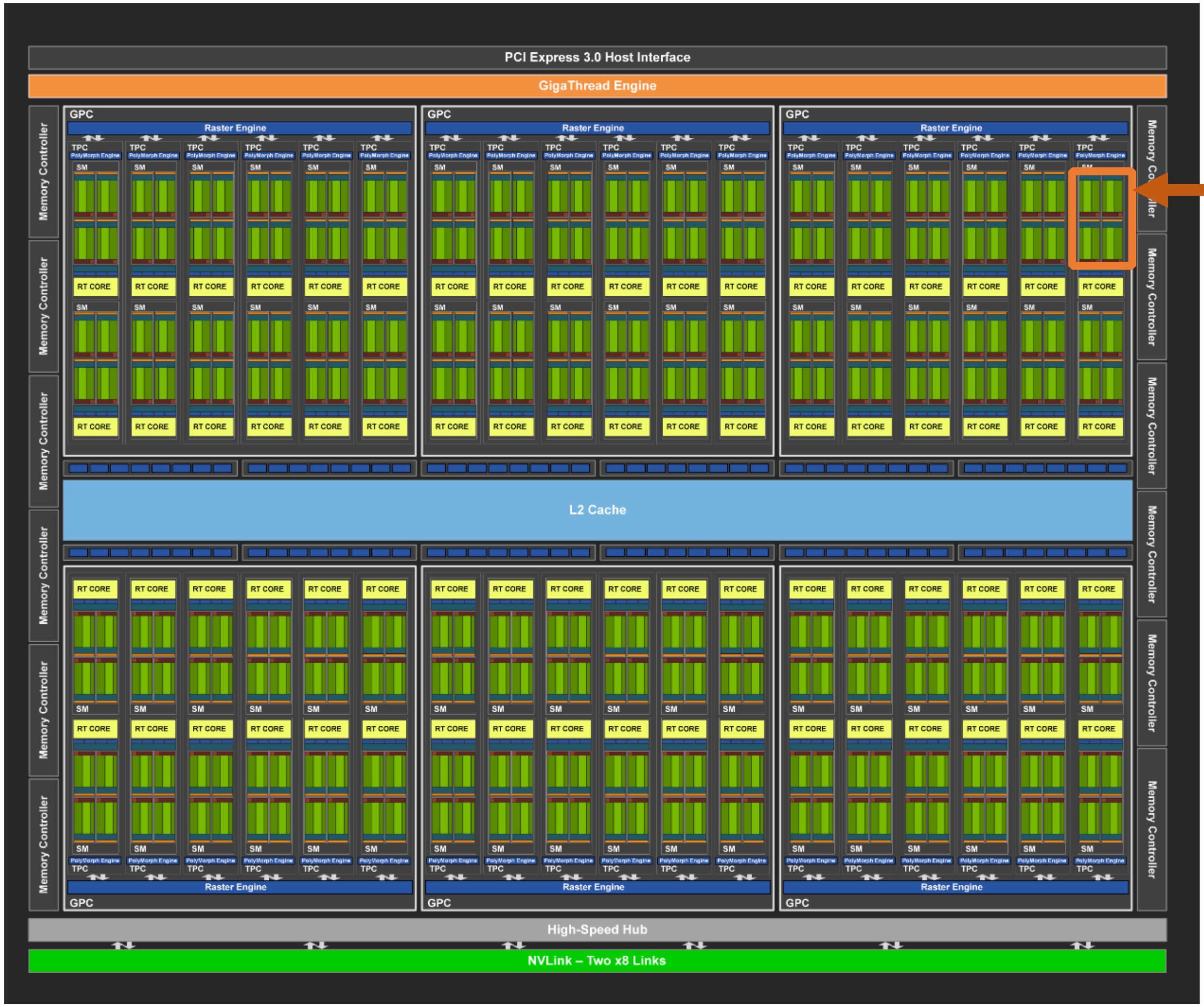

M1 PRO
2x Efficiency Cores
8x Performance Cores
2x L2 Blocks
16-Core NPU
128-bit LPDDR5
SLC Block
16-Core GPU
SLC Block
128-bit LPDDR5

Apple A16 (iPhone 14)



- 6 CPU Cores:
  - 2 performance (everest)
  - 4 efficiency (sawtooth)
- 5 GPU Cores

GPUS
NVidia Turing Architecture



At **Symmetric Multiprocessor (SM)** level, the execution is SIMD, but the overall execution, that includes multiple SMs, is MIMD

Parallel Architectures — MIMD Shared Memory Architectures

- Nonuniform memory access (NUMA) architect
  - Example: each node of DI's cluster

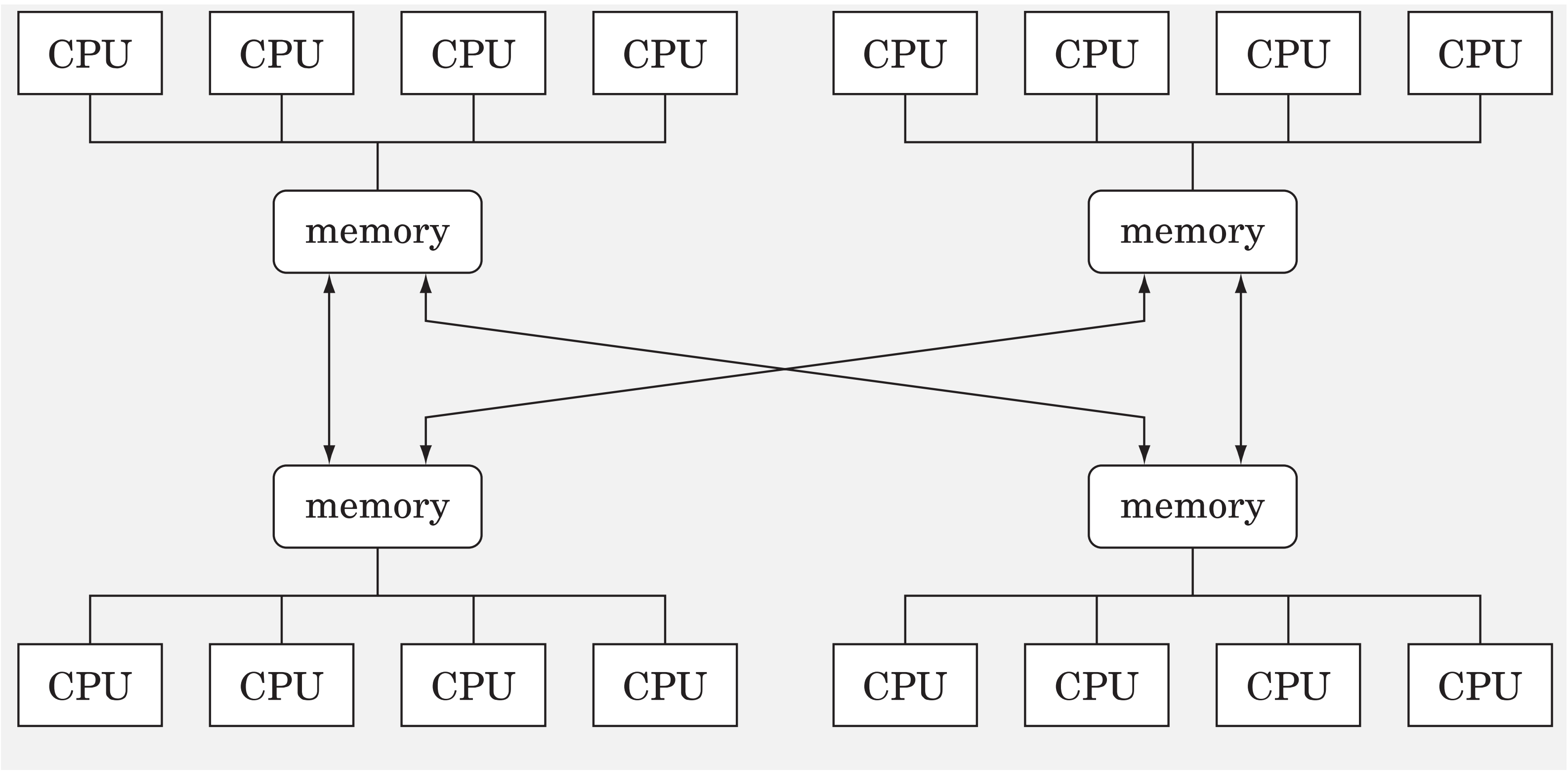


Image from: Mattson T., Sanders B., Massingill B.; Patterns for Parallel Programming; Addison-Wesley (2004).

Parallel Architectures — MIMD Distributed Memory Architectures

- Nonuniform memory access (NUMA) architect.
  - Example: all the node of DI's cluster

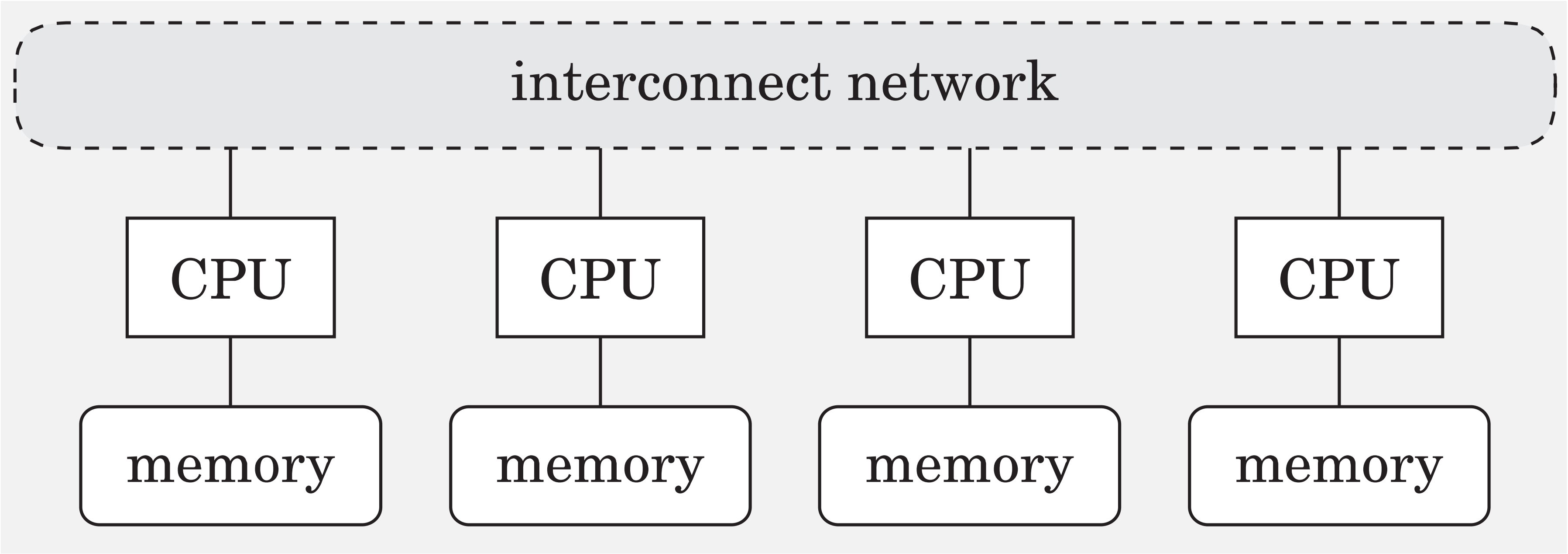


Image from: Mattson T., Sanders B., Massingill B.; Patterns for Parallel Programming; Addison-Wesley (2004).

Typical Distributed Memory MIMD Architecture of Today

- Multiple nodes connected via high-speed local networks
- Each node is a NUMA node with multiple multicore processors
- Each node may also have 1 or more GPUs
- Examples
  - DI cluster: http://cluster.di.fct.un.pt
  - Top 500: https://www.top500.org/lists/top500/2024/11/

Parallel Architectures – MIMD Distributed Memory Architectures

- Nonuniform memory access (NUMA) architect.
  - Example: all the node of DI's cluster

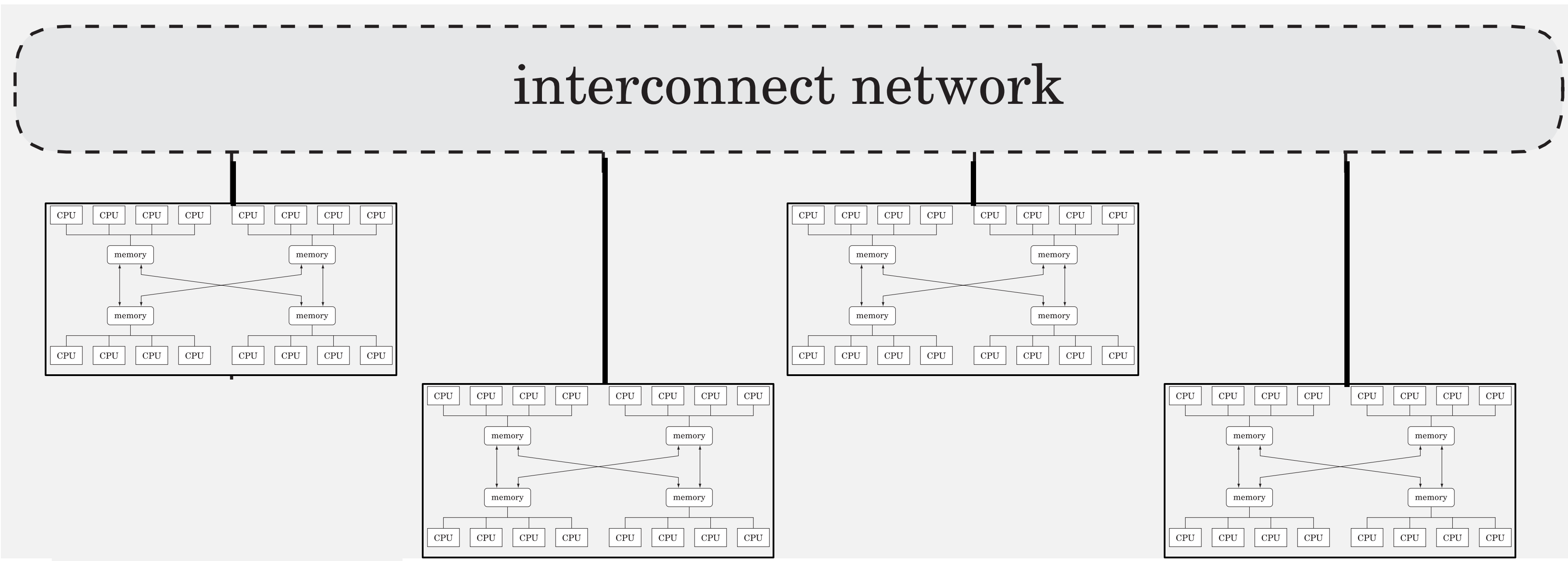


Image from: Mattson T., Sanders B., Massingill B.; Patterns for Parallel Programming; Addison-Wesley(2004).

Cluster @ DI

Cluster	Nodes	CPU	Total Cores/Threads	Memory	Network	Main Disk (/, /tmp)	Extra Disks (/mnt/localhddX)	Graphics
charmander	5	AMD EPYC 7281	16/32	128 GiB DDR4 2666 MHz	2 x 10 Gbps	1.8 TB HDD	-	-
squirtle	4	2 x Intel Xeon E5-2620 v2	12/24	64 GiB DDR3 1600 MHz	2 x 1 Gbps	100 GB SSD	-	-
psyduck	3	Intel Xeon X3450	4/8	8 GiB DDR3 1333 MHz	2 x 1 Gbps	230 GB HDD	[1,3] 230 GB HDD [2] -	-
shelder	1	4 x AMD Opteron 6272	32/64	64 GiB DDR3 1600 MHz	2 x 1 Gbps	120 GB SSD	2x 460 GB HDD	-
magikarp	1	8 x AMD Opteron 8220	16/16	27 GiB	2 x 1 Gbps	130 GB HDD	-	-
oddish	1	Intel Xeon E5-2603 v2	4/4	16 GiB DDR3 1333 MHz	2 x 1 Gbps	930 GB HDD	-	NVIDIA GeForce GTX 1050 Ti
bulbasaur	3	2 x Intel Xeon E5-2609 v4	16/16	32 GiB DDR4 2400 MHz	2 x 1 Gbps	110 GB SSD	-	NVIDIA Quadro M2000
gengar	5	2 x AMD Opteron 2376	8/8	16 GiB DDR2 667 MHz	2 x 1 Gbps	150 GB HDD	-	-
sudowoodo	1	Intel i7 10700	8/16	16 GiB DDR4 2933 MHz	1 Gbps	500 GB SSD	-	NVIDIA GeForce RTX 3070
lugia	5	2 x Intel Xeon Gold 6346	32/64	128 GiB DDR4 3200 MHz	2x10 Gbps	440 GB SSD	-	-
moltres	10	2 x AMD EPYC 7343	32/64	128 GiB DDR4 3200 MHz	2x10 Gbps	450 GB SSD	-	-

Example Top 500 (Nov 2024)

- Total: **11,039,616** cores
- **43,808** AMD 4th Gen EPYC 24C "Genoa" **24-core** 1.8 GHz CPUs (**1,051,392 cores**)
- **43,808** AMD Instinct MI300A GPUs (**9,988,224 cores**)

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	El Capitan - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	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
4	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	
5	HPC6 - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461
6	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
7	Alps - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cray OS, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	2,121,600	434.90	574.84	7,124
8	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
9	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
10	Tuolumne - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	1,161,216	208.10	288.88	3,387

Parallel Computing in current architectures

- Superscalar:
  - Exploits Instruction-Level Parallelism (ILP) within an instruction stream
  - Concurrently processes different instructions from the same instruction stream within a hardware thread
  - Parallelism is automatically discovered by the hardware during execution
  - *Transparent to the programmer, studied in the Computer Architecture course (**<= AC**)*
- SIMD (Single Instruction, Multiple Data):
  - Make use of SIMD hardware
  - Particularly efficient for data-parallel workloads that may be distributed among many ALUs
  - Vectorization can occur either through explicit SIMD instructions by the compiler or dynamically at runtime by the hardware
  - The absence of dependencies is typically declared by the programmer or inferred by compiler loop analysis before execution.
  - *Transparent to the programmer (compiler optimizations) or explicit (GPU programming **<= CAD**)*
- Make use of several hardware threads:
  - Involves the use of several hardware threads in one or more multi-processors across one or more computing nodes
  - Software determines the creation of software threads, often through parallel computing libraries, such as OpenMP
  - These software threads operate independently and communicate as needed
  - ***This topic will be the subject of the upcoming lectures in the course (<= CP)***

Conclusion

- To leverage the computing power of today's processors **sequential programming by itself is not enough anymore**
- Writing parallel programs can be **challenging**
  - Identify what to parallelize, partition the problem, synchronize/communicate
  - Many solutions are architecture dependent

The END