M7

Parallel Computer Architecture and Programming Models

Module Topics

Parallel Computing Architectures & Parallel Programming Paradigms - Memory Hierarchies - Shared vs Distributed Memory - Cache Optimisation & Techniques

Introduction to GPUs, CUDA GPU Ecosystem, GPU-based ML libraries: cuDF, cuML, and cuPY.

Multiprocessing with OpenMP, Distributed Computing with MPI.

Introduction to Scalability in ML, Challenges in Scaling ML Algorithms, ML Libraries and its parallel Capabilities, NUMBA and Dask.

Role of Scalability in DL's Evolution, Distributed training with TensorFlow, Distributed training with PyTorch. Challenges and Opportunities.

Scalable ML

Intended Learning Outcomes (ILOs)

- **Introduction to Computers**
  - Understand the basic components and operations of computers, including hardware and software, and how they work together to process data.
  - Identify different types of computers and their applications in everyday life.
- Memory Hierarchy
  - Comprehend the structure and function of the memory hierarchy in computers, including the role and characteristics of cache, RAM, and disk storage.
  - Analyze how the memory hierarchy affects computer performance and program execution.

Scalable ML

Intended Learning Outcomes (ILOs)

- Parallel Architectures
  - Recognize the fundamental concepts of parallel computing and the importance of parallel architectures in improving computational speed and efficiency.
  - Differentiate between various parallel computing architectures and understand their appropriate applications.
- Introduction to GPUs & CUDA
  - Gain insights into the architecture of GPUs and how they differ from CPUs in terms of processing power and efficiency.
  - Understand the basics of CUDA programming and how it enables the use of GPUs for general-purpose computing.

Scalable ML

Intended Learning Outcomes (ILOs)

- Distributed Computing, GPU Ecosystem, GPU-based ML libraries: cuDF, cuML, and cuPY
  - Explore the GPU ecosystem and learn how to leverage GPU-accelerated libraries like cuDF, cuML, and cuPY for efficient data science and machine learning tasks.
  - Apply these libraries in practical ML workflows to achieve significant performance improvements.
- Multiprocessing with OpenMP
  - Acquire knowledge on how to utilize OpenMP for multiprocessing in order to optimize computational tasks across multiple CPU cores.
  - Develop skills to implement parallel programming techniques using OpenMP in real-world applications.

Scalable ML

Intended Learning Outcomes (ILOs)

- Distributed Computing with MPI
  - Understand the principles of distributed computing and the role of MPI (Message Passing Interface) in facilitating communication between processes in a distributed system.
  - Apply MPI to design and execute parallel algorithms on distributed computing environments.
- Introduction to Scalability in ML
  - Recognize the importance of scalability in machine learning and its impact on the performance and efficiency of ML algorithms.
  - Identify key strategies for scaling ML workloads across multiple processors and machines.

Scalable ML

Intended Learning Outcomes (ILOs)

- Challenges in Scaling ML Algorithms
  - Analyze common challenges faced when scaling machine learning algorithms, including data distribution, communication overhead, and algorithmic efficiency.
  - Explore solutions to overcome these challenges and achieve scalable ML deployments.
- ML Libraries and its parallel Capabilities, NUMBA and Dask
  - Explore the capabilities of ML libraries that support parallel processing, focusing on NUMBA and Dask, and how they can be used to accelerate ML workflows.
  - Apply these libraries to parallelize data processing and model training tasks effectively

Scalable ML

Intended Learning Outcomes (ILOs)

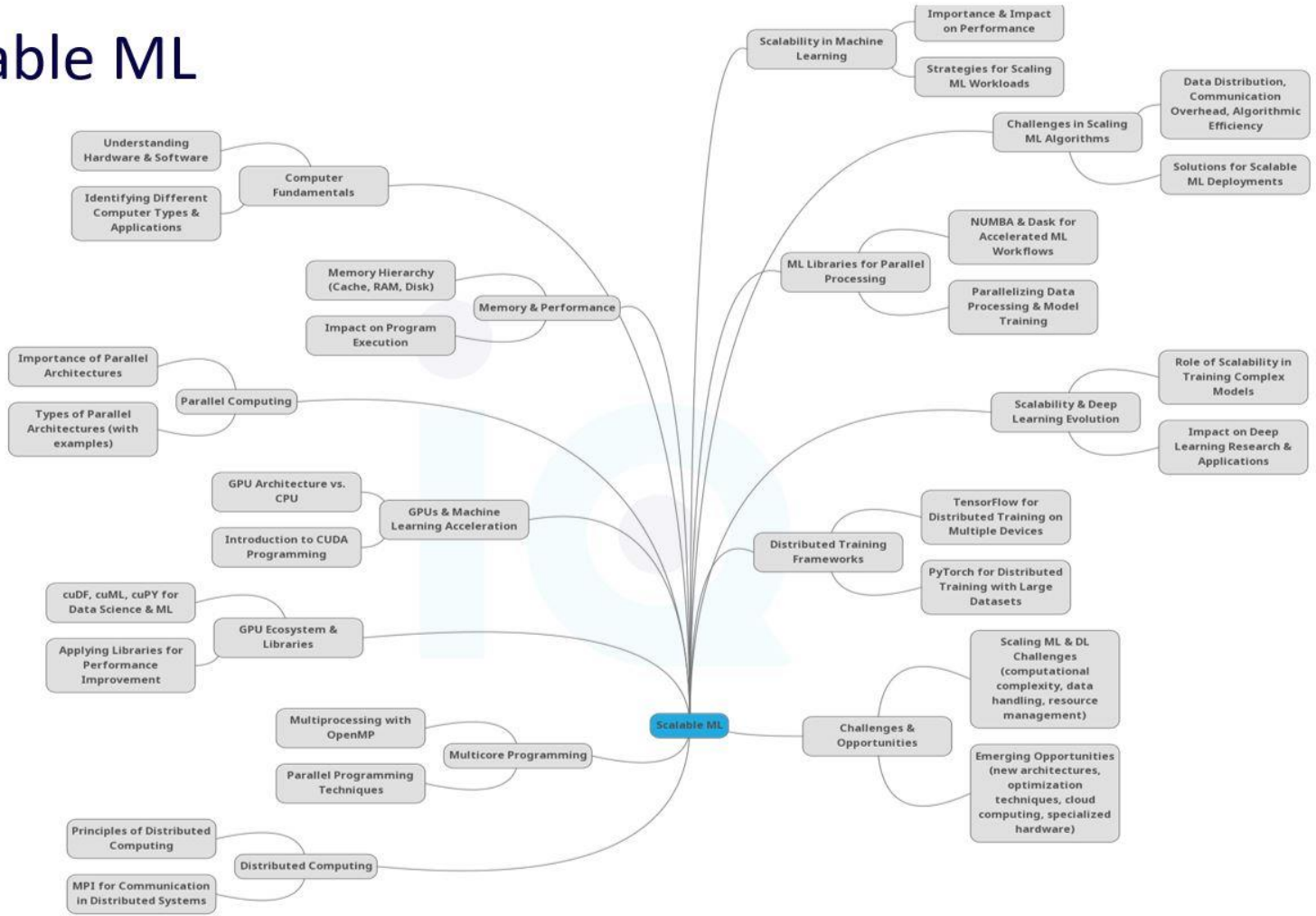
- Role of Scalability in DL's Evolution
  - Understand how scalability has driven the evolution of deep learning, enabling the training of complex models on large datasets.
  - Discuss the impact of scalable computing resources on the advancement of deep learning research and applications.
- Distributed Training with TensorFlow
  - Learn how to implement distributed training with TensorFlow, utilizing its built-in support for distributing computations across multiple devices.
  - Execute scalable training of deep learning models with TensorFlow on clusters of GPUs or across multiple machines.
  - Distributed Training with PyTorch - Gain skills in using PyTorch for distributed training, focusing on its distributed computing tools and libraries.

Scalable ML

Intended Learning Outcomes (ILOs)

- Challenges and Opportunities
  - Assess the challenges in scaling ML and DL algorithms, including computational complexity, data handling, and resource management.
  - Explore the emerging opportunities in scalable ML and DL, including new architectures, optimization techniques, and the role of cloud computing and specialized hardware.

Scalable ML


Scalable ML
Computer Fundamentals
Understanding Hardware & Software
Identifying Different Computer Types & Applications
Memory & Performance
Memory Hierarchy (Cache, RAM, Disk)
Impact on Program Execution
Parallel Computing
Importance of Parallel Architectures
Types of Parallel Architectures (with examples)
GPUs & Machine Learning Acceleration
GPU Architecture vs. CPU
Introduction to CUDA Programming
GPU Ecosystem & Libraries
cuDF, cuML, cuPY for Data Science & ML
Applying Libraries for Performance Improvement
Multicore Programming
Multiprocessing with OpenMP
Parallel Programming Techniques
Distributed Computing
Principles of Distributed Computing
MPI for Communication in Distributed Systems
Scalability in Machine Learning
Importance & Impact on Performance
Strategies for Scaling ML Workloads
Challenges in Scaling ML Algorithms
Data Distribution, Communication Overhead, Algorithmic Efficiency
Solutions for Scalable ML Deployments
ML Libraries for Parallel Processing
NUMBA & Dask for Accelerated ML Workflows
Parallelizing Data Processing & Model Training
Scalability & Deep Learning Evolution
Role of Scalability in Training Complex Models
Impact on Deep Learning Research & Applications
Distributed Training Frameworks
TensorFlow for Distributed Training on Multiple Devices
PyTorch for Distributed Training with Large Datasets
Challenges & Opportunities
Scaling ML & DL Challenges (computational complexity, data handling, resource management)
Emerging Opportunities (new architectures, optimization techniques, cloud computing, specialized hardware)

1

Introduction to Parallel Computer Architecture

Topics

Introduction to Computers

Memory Hierarchy

Parallel Architectures

Introduction to Computers

Introduction to Computers

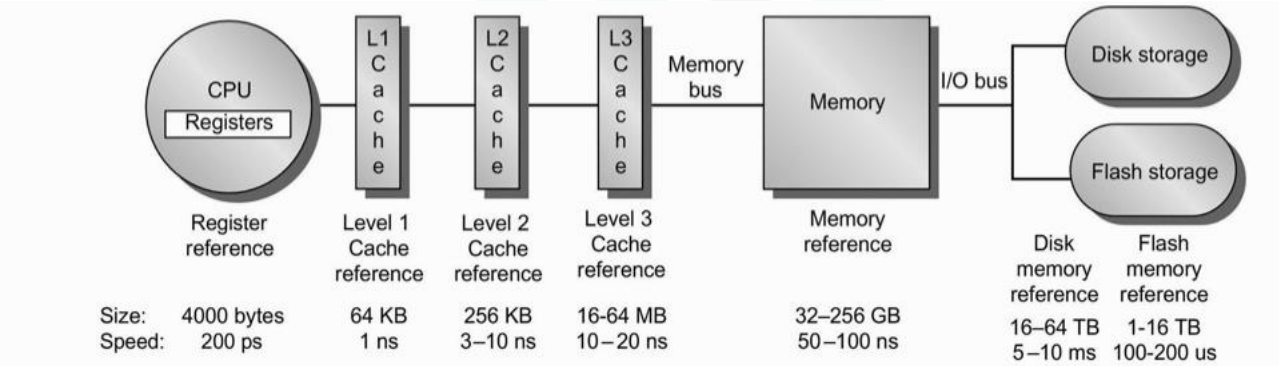
Core Components

- **CPU (Central Processing Unit):** The brain of the computer, responsible for executing instructions.
- **Memory:** Stores data temporarily for quick access by the CPU.
- **Storage:** Hard drives or solid-state drives that store data persistently.
- **Motherboard:** The main circuit board that connects all components.
- **Power Supply:** Converts electrical power from outlets into a form the computer can use.
- **Input & Output Devices:** Keyboards, mice, monitors,

Computer Memory

Memory Hierarchy

- Separates computer storage into a hierarchy based on response time.
- Used to provide an organized and efficient method of storing and accessing data.



Computer Memory

Memory Hierarchy

- **Registers:** Smallest, fastest, and closest to the CPU. Used for immediate operations.
- **Cache Memory:** Small-sized, faster memory that stores copies of frequently accessed data from main memory.
- **Main Memory (RAM):** Directly accessible by the CPU. Volatile memory used for currently running processes.
- **Secondary Storage**: Non-volatile memory including hard drives and SSDs. Used for long-term data storage.

Computer Memory

Cache Memory

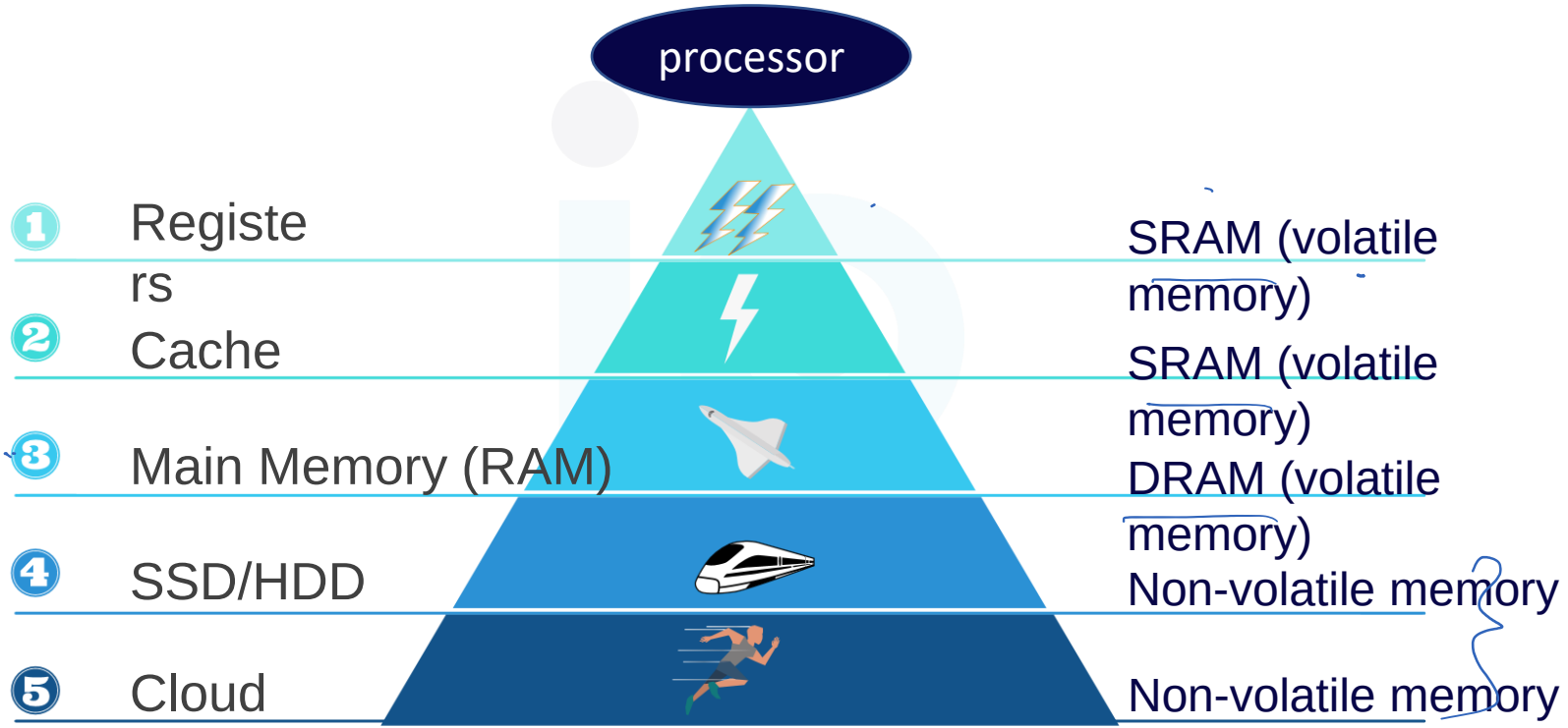
- **L1 Cache:** Located on the CPU chip, provides the fastest access to data but has a very limited size.
- **L2 Cache:** Slightly slower than L1, larger in size, can be on the CPU or separate chip.
- **L3 Cache:** Slower and larger than L2, shared among cores in multi-core processors.

Main Memory

- Serves as the working memory of the computer.
- Stores data and programs that are currently being used or executed by the CPU.
- Volatile memory, meaning it loses its content when the power is turned off.

Memory Hierarchy Design

Summary



“Data

Bits, Bytes and words

Bit

- The most fundamental unit of memory is a bit (binary digits)
- The smallest unit has two states:
  - Charged or cleared in RAM (random access memory)
  - Magnetized or not in magnetic storage devices
  - Different levels of light reflectance in optimal storage devices
- Two states can be represented by 0 or 1

Bits, Bytes and words

Bytes

- A unit of information formed from bits
- One byte equals to 8 bits
  - Eg. 01011010

Word

- A group of one to eight bytes
- Seldom used

Memory

Memory and data representation

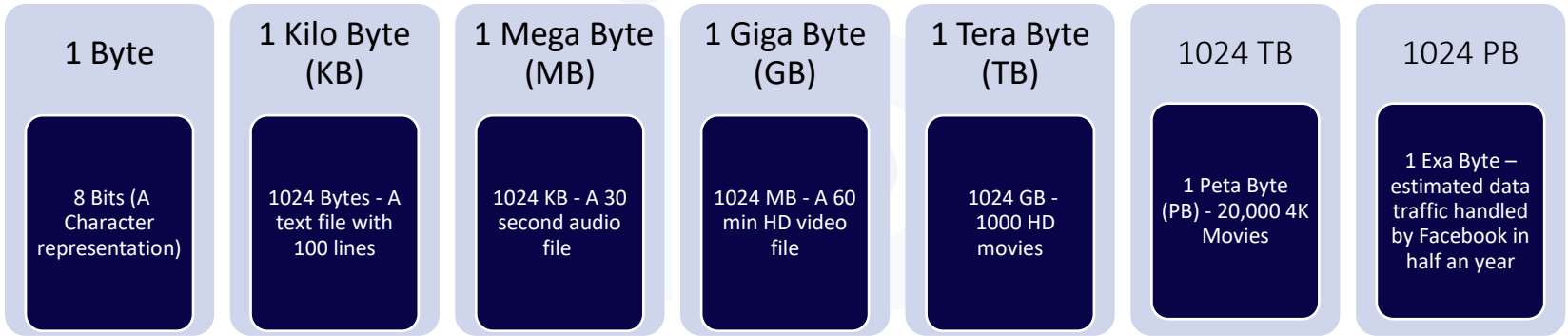
- Computers uses a fixed number of bits to represent or store data
- Each n-bit storage location can represent one of the 2^n distinct entries
  - Eg. a 2-bit memory location can hold one of these four binary patterns: 00, 01, 10, 11

Number systems

- binary (base 2), octal (base 8), decimal (base 10), hexadecimal (base 16)
  - Eg. $8 = 1000_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

Digital Data Representation

- The fundamental unit of memory is called a **bit** (binary digit)
- 8-bits corresponds to one **byte**, and a group of 1 to 8 bytes is **a word**



Data Facts:

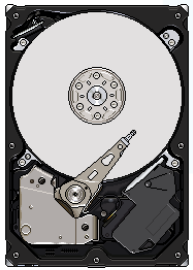
- Entire size of Wikipedia data without media (4 billion words and 57 million pages) is just **21.23 GB.** In comparison, a 4k blue ray movie like Avatar can take **40 - 100 GB** of storage space.

  Ref: https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia

Digital Data Representation

The data is represented in the binary form using the state of the smallest unit in a storage device

Charged / cleared (Voltage level)



Magnetized / Non-Magnetized

Different levels of light reflectance

Memory Access

How many bits a processor can access at a time?

- A 32-bit processor can access 2^{32} different memory addresses, i.e, 4,294,967,296 bytes or 4 GB of RAM or physical memory
- A 64-bit processor can access 2^{64} different memory addresses, i.e, 18,446,744,073,709,551,616 bytes, or 17,179,869,184 GB (16 exabytes) of memory

“How to measure the growth/speed of computers?

Performance of a computer

Clock speed

- Microprocessor clock speed measures the number of pulses per second generated by an oscillator in the processor. (Hz, pulses per sec)
- During each cycle, billions of transistors within the processor open and close.
  - E.g; 3.2 GHz CPU executes 3.2 billion cycles per second.
- Sometimes, multiple instructions are completed in a single clock cycle; in other cases, one instruction might be handled over multiple clock cycles
- Is this enough to check the performance of a computer?
- As a Data scientist/analyst, what do you check?

Performance of a computer

MIPS (million instructions per second)

- measures the number of instructions a processor can handle in a second
- each instruction may require several clock cycles to complete
- In general, MIPS measures integer performance of a computer. Ex. integers operations including data movement, (a=b), testing, if(a=b then e), etc. Applications including database queries, word processing, running multiple virtual operating systems.
- How about floating-point operations?

Performance of a computer

FLOPS (floating point operations per second)

- used to measure the performance of a processor that performs floating point operations.
- integer operations are excluded
- all ML/Scientific computations use floating point numbers

Class of Computers

Class of Computers

IoT/Embedded Computers

- 8 to 32-bit processors
- Everyday machines: microwaves, washing machines, most printers, network switches, automobiles, etc.
- IoT refers to embedded computers that are connected to internet.
  - ⇒ Smart applications: watches, home, speakers, cars, cities, etc
- In 2020 I 20-50 billion devices

Class of Computers

Personal Mobile Device (PMD)

- Laptop/desktop capacity (late 2010s)
- Mobile devices

Desktop Computers

- largest market but sales are declining
- more than enough for day-to-day Calculations a software developments.

Class of Computers

Servers

- More reliable & Computing services
- backbone of large-scale enterprise computing.
- runs 24/7, Scalable, designed for efficient throughput.

Clusters/Warehouse-scale computers (WSC)

- Collection of desktop computers & servers connected by local area network
- each node runs with its own OS & communicate using network protocol.
- efficient network ⇒ supercomputer.

Performance of a computer

More on performance …

I. Can CPUs do more FLOPs than clock speed or MIPS?
II. What is cache? L1, L2, L3
III. What is bandwidth and Latency?
IV. Is cache memory different from RAM?
V. What is the difference between hard disc & RAM, and what is the role/need RAM?
VI. What does dual/quadcore CPU mean?

History of a computers

History of Computers

Computer Performance

- Exponential growth of computing power in the last 75 yrs. has changed the scientific computing, and key reason for the success of AI & ML
- Consequence of
  - Moore's law + Dennard scaling
  - Improvements in computer architecture (what to do with all the transistors?)
  - Improvement in computational algorithms

Source: YouTube

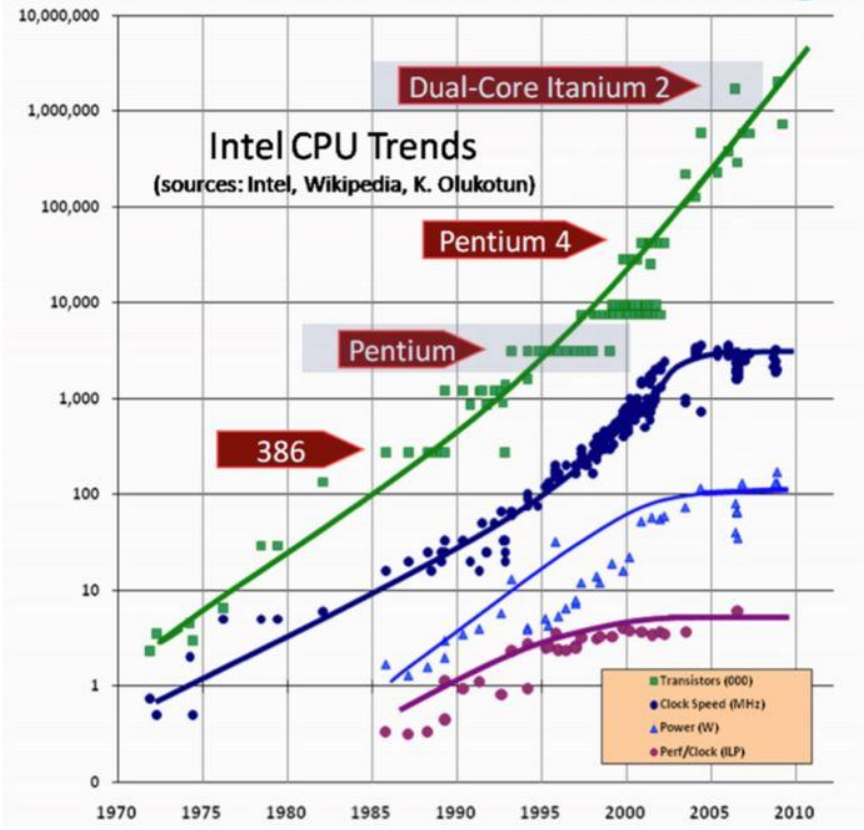
History of Computers

Moore's Law and Dennard Scaling

- **Moore’s law:** Number of transistors on a chip doubles every two years (Gordon Moore 1965, changed from 1 year to two years in 1975)
- **Dennard Scaling:** Power density (W/m^2) can be kept constant at scaling even with increasing clock rate (essentially by lowering supply voltage)
- Dennard scaling ended 2004
- Moore’s law has significantly slowed down in recent years

Source: YouTube

End of Dennard Scaling in Detail

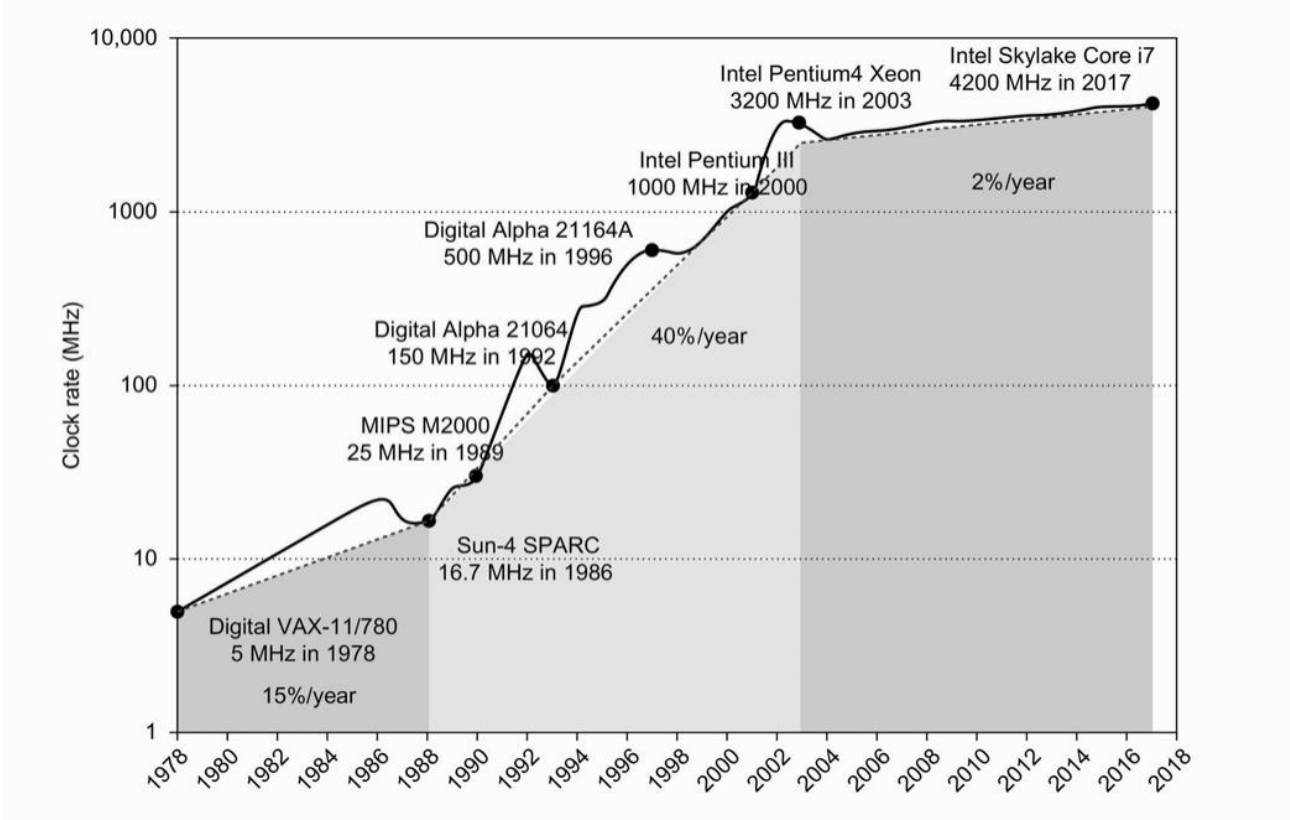


Dramatic events in 2003:

- End of Dennard scaling lead to stagnation of clock rate
- Improvement of instruction level parallelism (ILP) came to halt: This ended the automatic increase of instructions executed per clock
- Way out: **Multicore architecture**

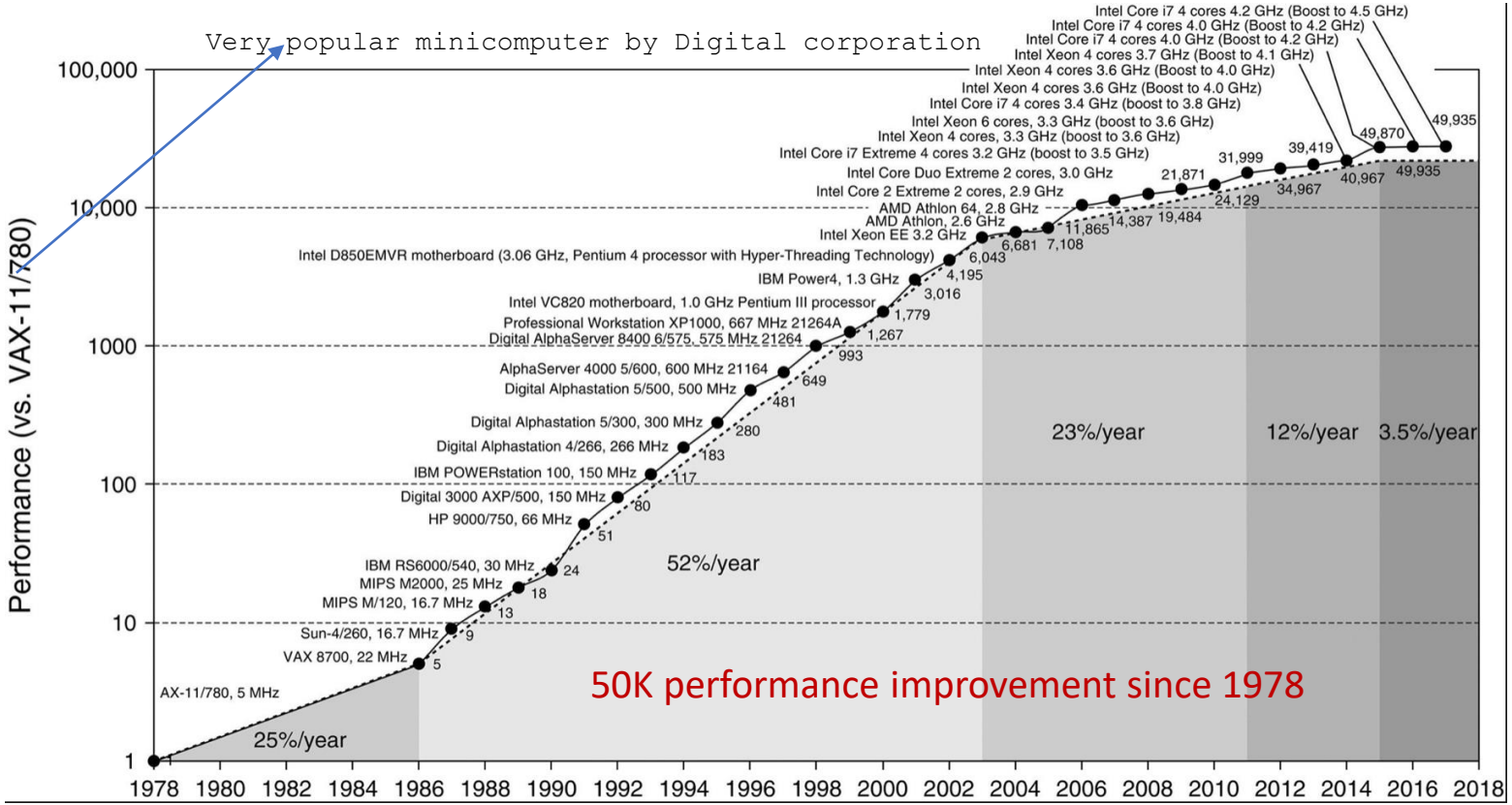
Image credit: Herb Sutter. *The Free Lunch Is Over.* Dr. Dobb's Journal. 2005.

History of Computers



Source: Hennesy/Patterson: Computer Architecture

History of Computers



Source: Hennesy/Patterson: Computer Architecture

History of Computers

Timeline of hardware development

- 1945: Mainframes: build from individual transistors
- 1970: Minicomputer: first discrete, then low integration
- End of 1970s: Microprocessors: Gradually into all machines
- Early 1980s: Reduced instruction set computers (RISC)
  - Instruction level parallelism: Goal - one instruction/clock
- Early 2000s: End of Dennard scalling lead to Multicore CPUs
  - thread level parallelism
- mid 2010s: Special purpose architectures
  - GPGPU computing
  - Massive use of data parallelism

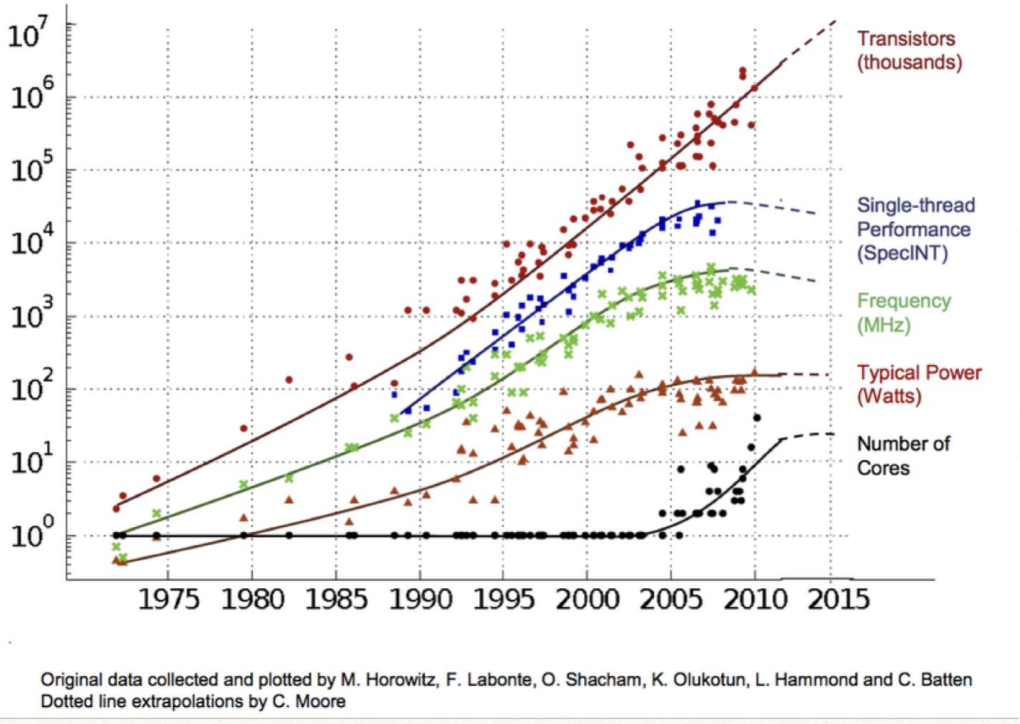
History of Computers

Timeline of hardware development

- till mid 1980s :
  - Technology driven
  - Doubles every 3.5 Yrs
- starting 1986:
  - Technology + RISC (reduced instruction set computer)
  - Doubles every 2 yrs
- 2003 - 2011 :
  - Doubles every 3-5 yrs
  - End of Dennard scaling
    - ➔ Forced multicore!
- 2011 - 2015:
  - Doubles every 8 yrs.

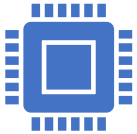
History of Computers

Timeline of hardware development



- end of Dennard scaling leads to stagnation of clock rate
- improvement of Instruction Level parallelism came to halt
- ended automatic increase of instruction executed per clock
- Way out: multicore

Birth of Parallel Computing



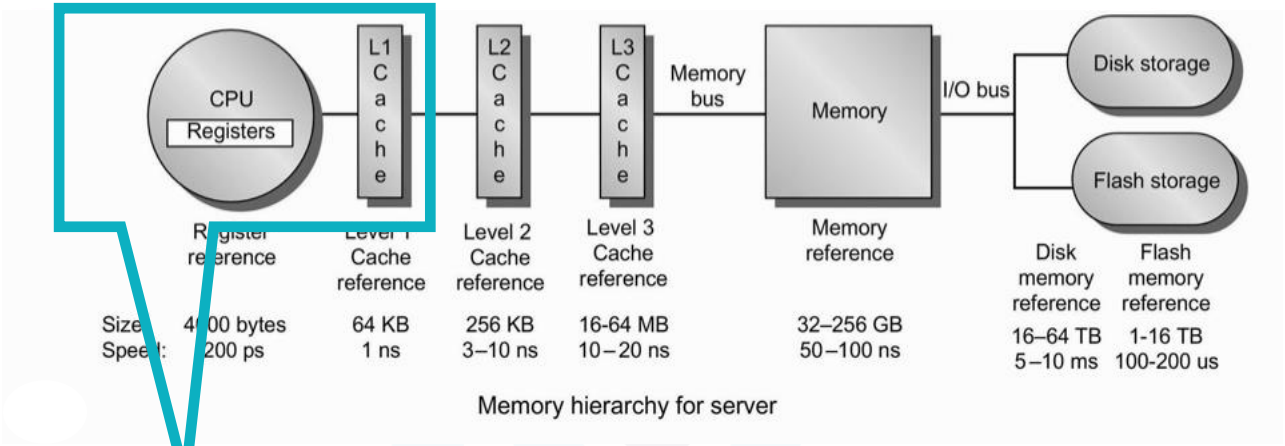
The end of exponential growth in single-processor performance makes the end of the single processor computing

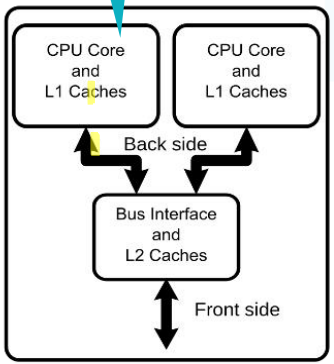
The era of sequential computing must give way to new era of parallel computing

Parallel computing is not easy as yesterday's sequential computing, but no alternative exists to improve computing performance, else game over for growth in computing

AI most all AI/ML package and tools use multicore, multiprocessor, GPU, TPUs, etc.

Class of Computers

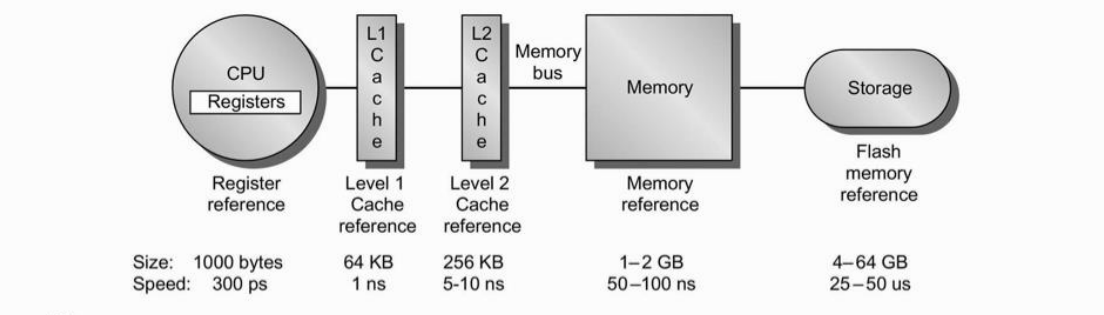




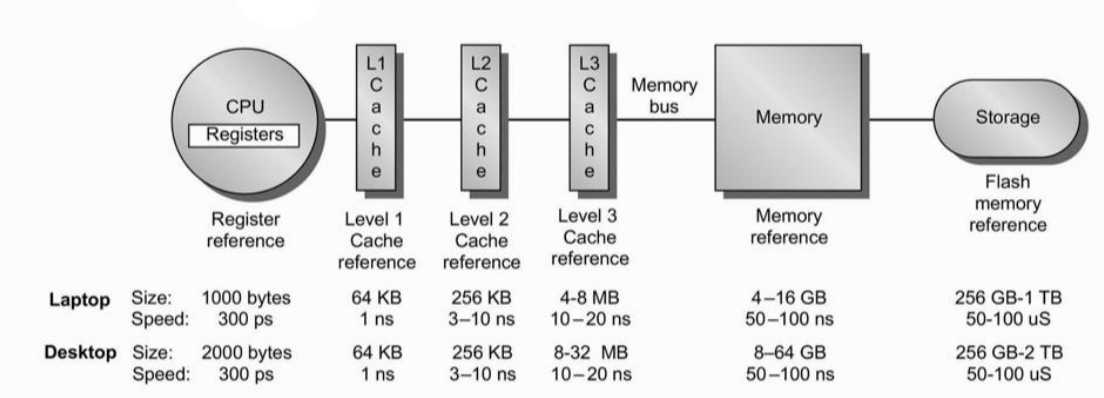
Birth of Parallel Computers

Memory Hierarchy - Revisit

Memory Hierarchy Design

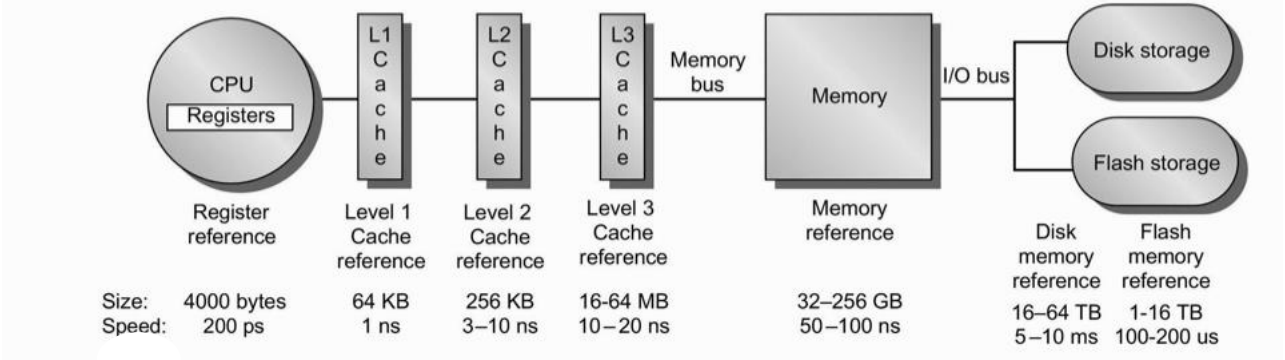


Mobile Devices



Desktops

Memory Hierarchy Design



Servers

Memory Hierarchy Design

Volatile memory

Volatile memory is a computer memory that requires power to retain the stored information

E.g; RAM

Dynamic RAM (DRAM, cost-effective)

- DRAM chip needs single capacitor and one transistor to store each bit of data.
- Space-efficient and less expensive.
- Capacitors lose its charge a bit (transistors draw a little current) and thus need to be refreshed often.

Memory Hierarchy Design

Static RAM (SRAM, expensive)

- SRAM is much faster than DRAM but expensive.
- Does not use capacitors but several transistors (typically six) and does not have leakage issue.
- Does not need to be refreshed but still needs constant power to keep the data intact.
- Mainly used as CPU cache or processor registers.

How about SDRAM?

Memory Hierarchy Design

Non-Volatile memory

Non-Volatile memory is a computer memory that retains the stored information even after power is removed.

E.g: Secondary Device : Hard disks

Mechanically addressed systems

Make use of a contact structure to read and write on a selected storage device.

E.g. optical disks, hard disks.

Memory Hierarchy Design

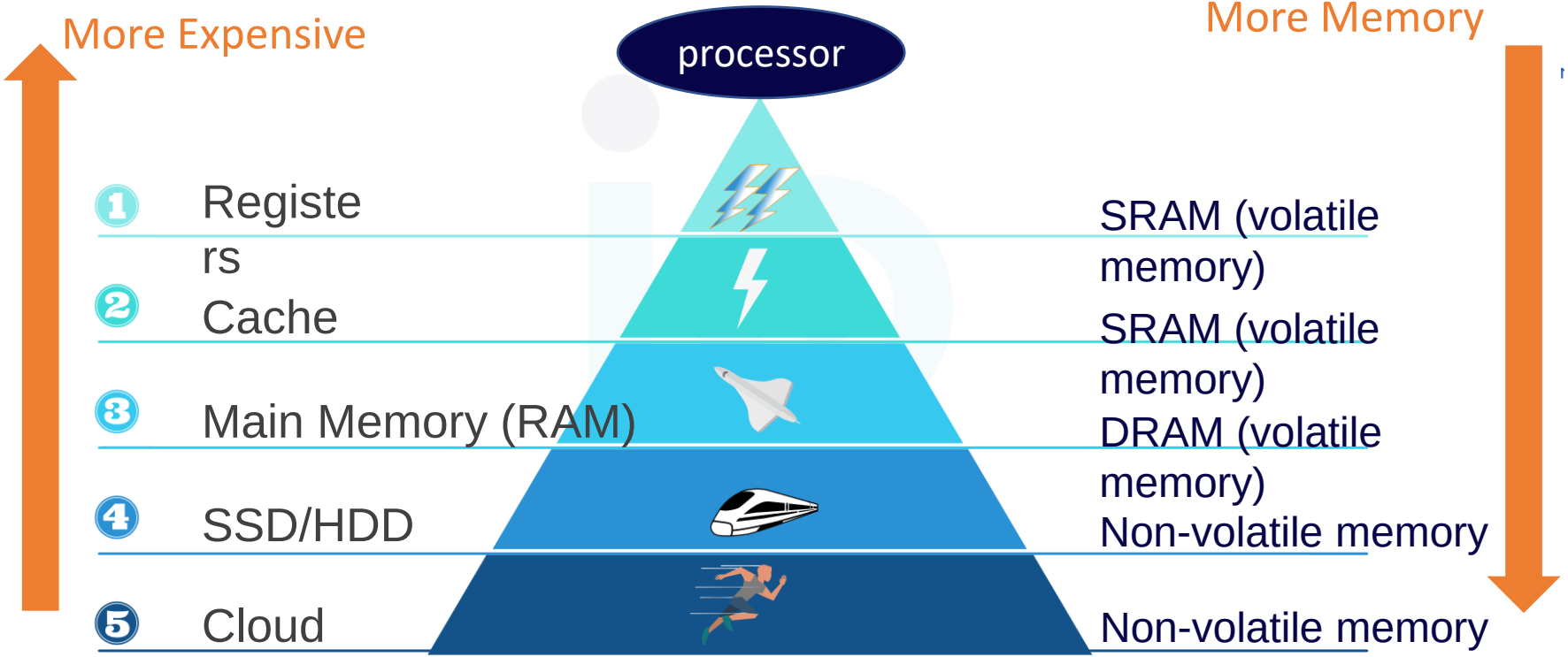
Electronically addressed systems

- Categorised according to their write mechanism
- It is faster than mechanically addressed systems but expensive
  - Eg. Mask ROM, Programmable ROM, Erasable PROM, Electrically EPROM

What do programmers want?

- Programmers want unlimited amount of fast primary memory
- An economical solution to that desire is a memory hierarchy

Memory Hierarchy Design



Memory Hierarchy Design

True/False

- RAM is the primary memory of a computer system
- Multicore system consists a single CPU (processor) with two or more independent processing units
- Multiprocessor system consists of two or more CPUs each sharing RAM and peripherals

Parallel computer Architectures

Parallel Architectures

Fastest supercomputer (as on Nov 2023)

- ?

Supercomputer at IISc

- ?

Classification of Parallel Computers

- Shared Memory Parallelism (SMP)
- Distributed Memory Process (DMP)
- Hybrid Supercomputers

Parallel Architectures

Fastest supercomputer (as on Nov 2024)

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

Supercomputers in India (Nov 2023)

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
90	**AIRAWAT - PSAI** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Infiniband HDR, Netweb Technologies Center for Development of Advanced Computing (C-DAC) India	81,344	8.50	13.17	
163	**PARAM Siddhi-AI** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, EVIDEN Center for Development of Advanced Computing (C-DAC) India	41,664	4.62	5.27	
201	**Pratyush** - Cray XC40, Xeon E5-2695v4 18C 2.1GHz, Aries interconnect , HPE Indian Institute of Tropical Meteorology India	119,232	3.76	4.01	1,353
354	**Mihir** - Cray XC40, Xeon E5-2695v4 18C 2.1GHz, Aries interconnect , HPE National Centre for Medium Range Weather Forecasting India	83,592	2.57	2.81	955

Param Pravega @ IISc

Total Cores : 28k cores
Flops : 3.3 Peta Flops (10^{15})
GPU's : 80 V100 GPU's

Supercomputers in India (Nov 2024)

Param Pravega @ IISc

Total Cores : 28k cores
Flops : 3.3 Peta Flops (10^{15})
GPU's : 80 V100 GPU's

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
136	**AIRAWAT - PSAI** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Infiniband HDR, Netweb Technologies Center for Development of Advanced Computing (C-DAC) India	81,344	8.50	13.17	
188	**Arka** - BullSequana XH2000, AMD EPYC 7643 48C 2.3GHz, Infiniband HDR, Red Hat Enterprise Linux, EVIDEN Indian Institute of Tropical Meteorology India	290,016	5.94	7.40	1,236
189	**Arunika** - BullSequana XH2000, AMD EPYC 7643 48C 2.3GHz, Infiniband HDR, Red Hat Enterprise Linux, EVIDEN National Centre for Medium Range Weather Forecasting India	203,040	5.94	7.40	1,236
268	**Pratyush** - Cray XC40, Xeon E5-2695v4 18C 2.1GHz, Aries interconnect , HPE Indian Institute of Tropical Meteorology India	119,232	3.76	4.01	1,353
400	**Arka AI/ML** - NVIDIA DGX H100, Xeon Platinum 8480C 56C 2GHz, NVIDIA H100 SXM5 80GB, Octo-rail NVIDIA HDR100 Infiniband, Red Hat Enterprise Linux, EVIDEN Indian Institute of Tropical Meteorology India	8,176	2.70	3.75	
431	**Mihir** - Cray XC40, Xeon E5-2695v4 18C 2.1GHz, Aries interconnect , HPE National Centre for Medium Range Weather Forecasting India	83,592	2.57	2.81	955

Energy for Supercomputers

Did you know ?

- Frontier consumes about 20 MW of power at peak performance, which is sufficient to power a mini city (4000 homes with 5 KW load per home)
- Bangalore's daily power demand is about 8128 MW

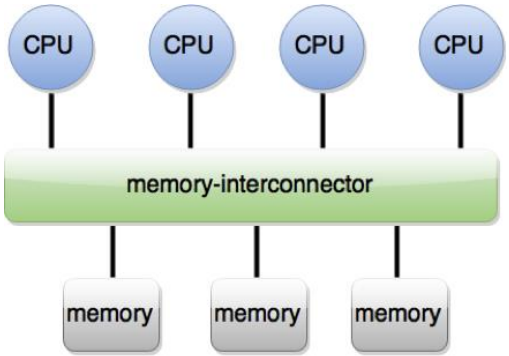
Seems too much of power consumption, right ?

- This power consumption accounts for a 62.68 Gigaflops of performance per watt of power consumed
- Latest Intel i9 series processors typically have a performance of around 800 to 950 Gigaflops (depending on variant) consuming around 100W of power
- This accounts for less than 10 Gigaflops of performance per watt of power consumed.
- The approximate cost of building the supercomputer is 600 Million USD

Parallel Architectures

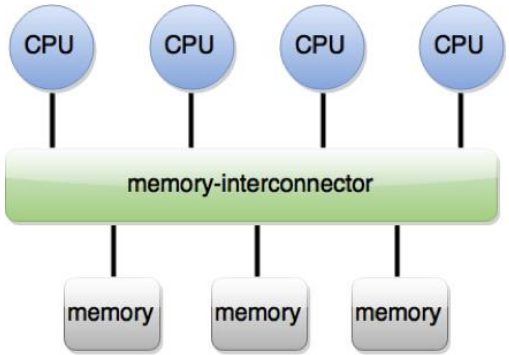
Shared Memory Parallelism (SMP)

- All CPUs connected to all memory banks with same speed
- Uniform memory access
- OpenMP implementation
  - shared memory directives
  - to define work decomposition NOT data decomposition
  - synchronisation is implicit (can also be used-defined)
  - https://www.openmp.org



Parallel Architectures

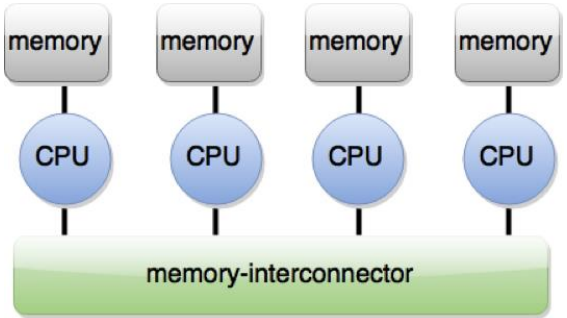
Shared Memory Parallelism (SMP)



Parallel Architectures

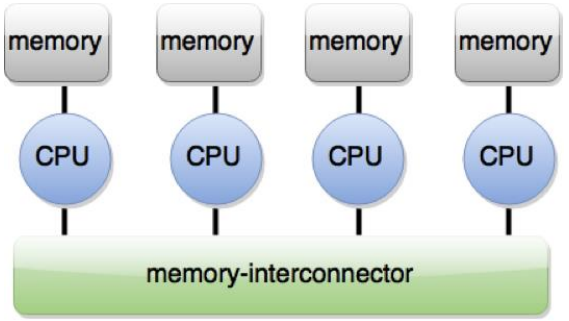
Distributed Memory Parallelism (DMP)

- CPUs connected by node-interconnect
- non-uniform memory access and access only to own memory
- access to other CPU's memory through MPI implementation
- user specifies how and when work and data need to be distributed
- user specifies how and when the communication needs to be done



Parallel Architectures

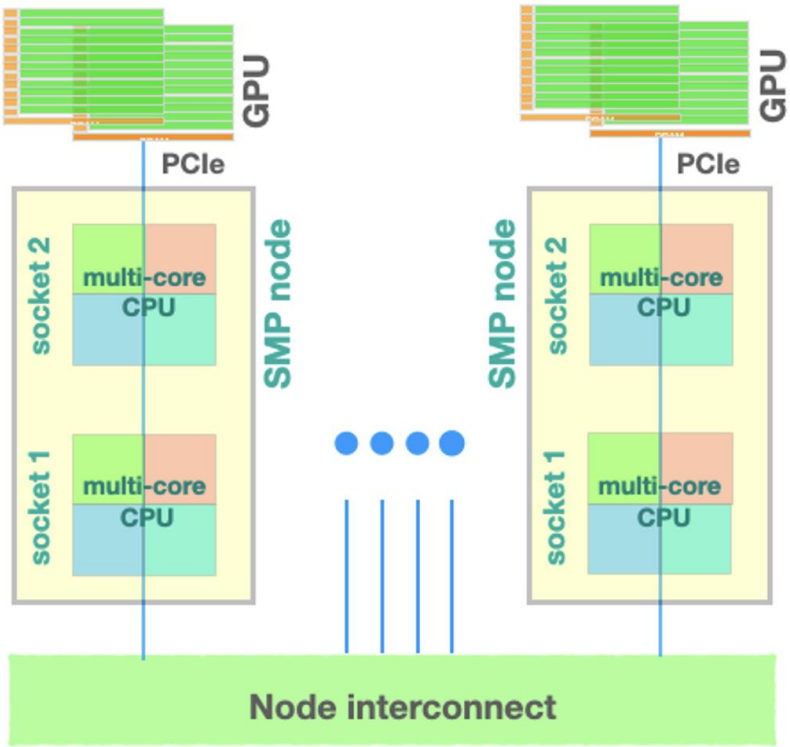
Distributed Memory Parallelism (DMP)



Parallel Architectures

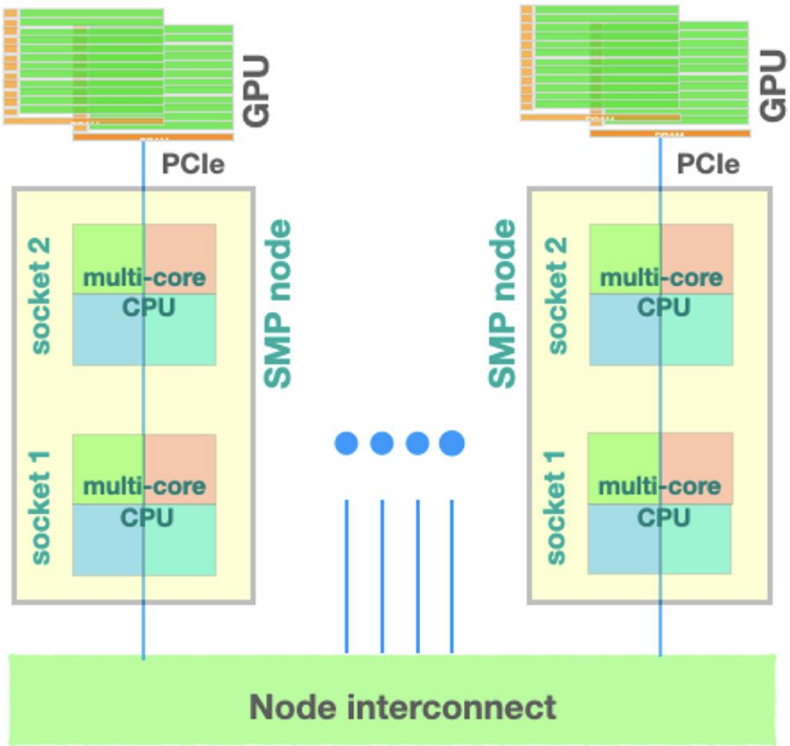
Hybrid Architectures

- SMP on each node
- DMP on node interconnect
- GPU accelerators or Intel Xeon-Phi Coprocessors
- MPI+X implementations



Parallel Architectures

Hybrid Architectures



Parallel Architectures

True/False

- Distributed Memory Machine shares the primary memory (RAM) among all CPUs
- A set of SMP systems (nodes) can be connected to build a Hybrid parallel cluster
- OpenMP parallel implementation is needed to execute programs DMP