Introduction to Parallel Architectures and High Performance Computing

From evolution to present

Compiled by Dr. DEJEY

Source: C-DAC, Bangalore

High Performance Computing

- Use of **supercomputers** and **parallel computing techniques to solve complex computational problems**.
- **Practice of aggregating computing power to deliver superior performance as opposed to other infrastructure.**
  - ✓ Work on the lowest-level technologies & circuit design
  - ✓ How to effectively employ in supercomputers**.**

Parallel Computing

Parallel computing can be defined as a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then **solved concurrently**.

Parallel computing usually involves two distinct areas of computing technologies:

➤ Computer architecture (hardware aspect)

➤ Parallel programming (software aspect)

Computer architecture focuses on supporting parallelism at an **architectural level**, while parallel programming focuses on **solving a problem concurrently by fully using the computational power of the computer architecture**.

Parallel program and Parallelism

There are two ways to classify the relationship between two pieces of computation: Some are related by a precedence restraint and therefore must be calculated sequentially; others have no such restraints and can be calculated concurrently. Any program containing tasks that are performed concurrently is a **parallel program.**

There are two fundamental types of parallelism in applications:

➤ **Task parallelism**

➤ **Data parallelism**

- **Task parallelism arises when there are many tasks or functions that can be operated independently and largely in parallel. Task parallelism focuses on distributing functions across multiple cores.**
- **Data parallelism arises when there are many data items that can be operated on at the same time. Data parallelism focuses on distributing the data across multiple cores.**

DATA PARTITIONS

There are two basic approaches to partitioning data:

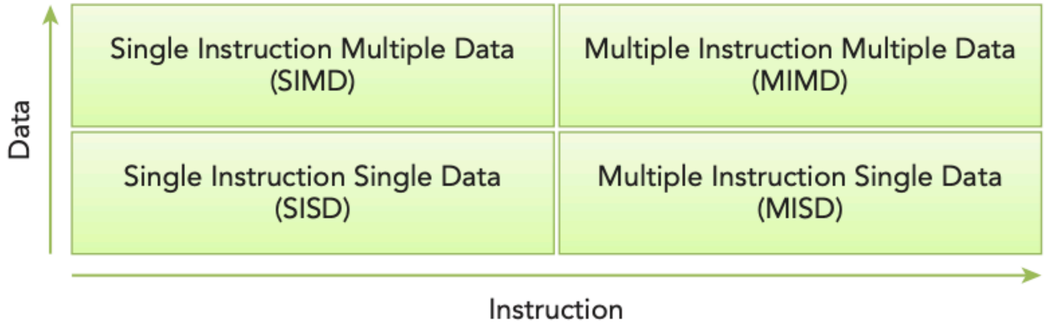
➤ **Block: Each thread takes one portion of the data, usually an equal portion of the data.**

➤ **Cyclic: Each thread takes more than one portion of the data.**

Flynns Taxonomy

- Flynn's Taxonomy

➤ Single Instruction Single Data (SISD)
➤ Single Instruction Multiple Data (SIMD)
➤ Multiple Instruction Single Data (MISD)
➤ Multiple Instruction Multiple Data (MIMD)



At the architectural level, many advances have been made to achieve the following objectives:
➤ Decrease latency
➤ Increase bandwidth
➤ Increase throughput

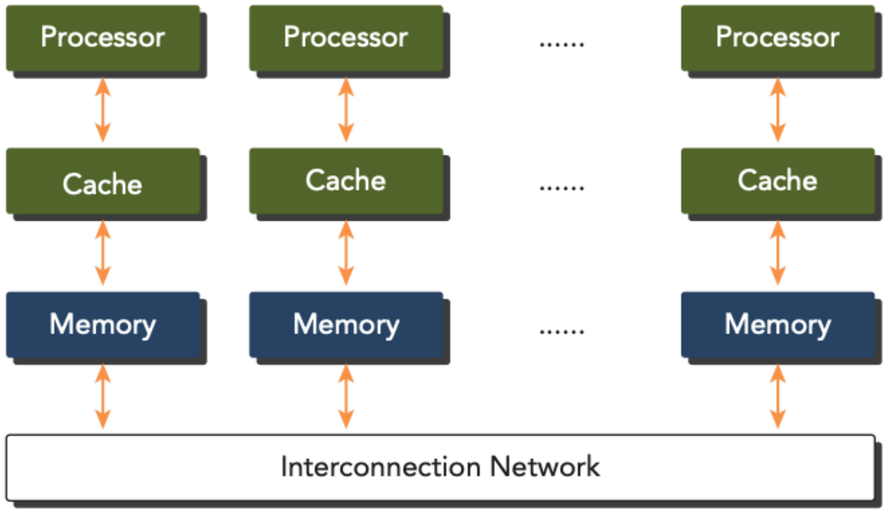
- **Latency** is the time it takes for an operation to start and complete, and is commonly expressed in microseconds.
- **Bandwidth** is the amount of data that can be processed per unit of time, commonly expressed as megabytes/sec or gigabytes/sec.
- **Throughput** is the amount of operations that can be processed per unit of time, commonly expressed as gflops (which stands for billion floating-point operations per second)

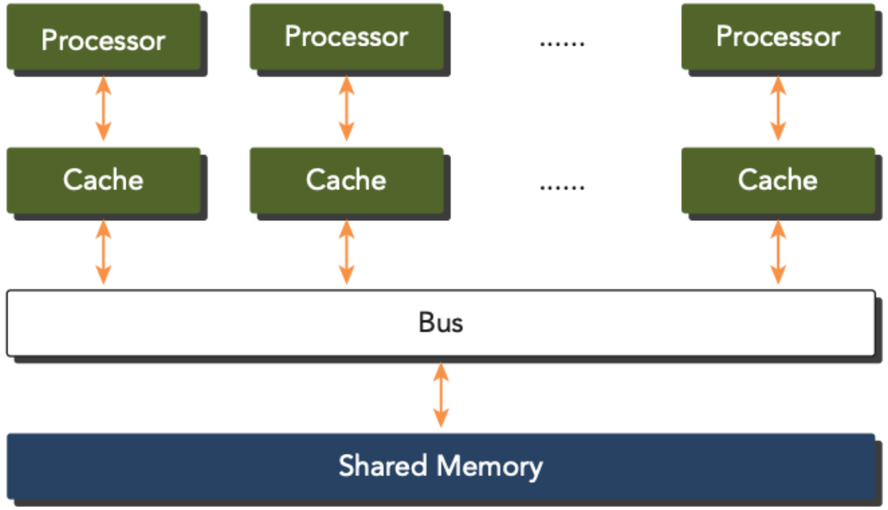
Computer Architectures

Computer architectures can also be subdivided by their memory organization, which is generally classifi ed into the following two types:

➤ Multi-node with distributed memory

➤ Multiprocessor with shared memory





GPUs represent a many-core architecture, and have virtually every type of parallelism described previously: multithreading, MIMD, SIMD, and instruction-level parallelism. **NVIDIA coined the phrase Single Instruction, Multiple Thread (SIMT) for this type of architecture.**

HETEROGENEOUS COMPUTING

GPU CORE VERSUS CPU CORE

Even though many-core and multicore are used to label GPU and CPU architectures, a GPU core is quite different than a CPU core.

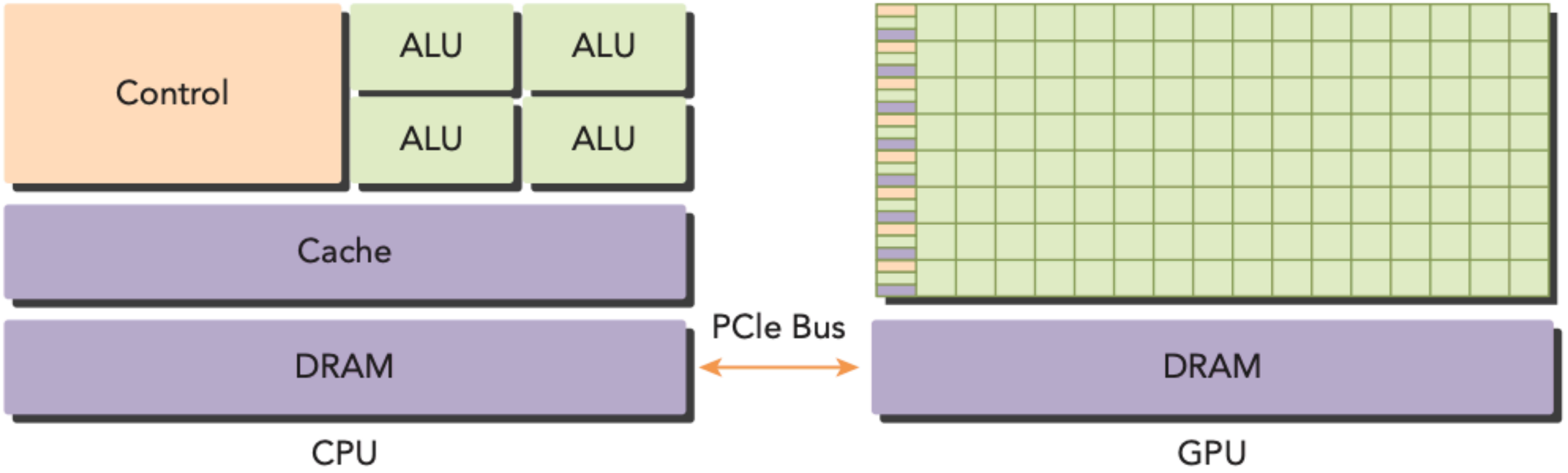
A CPU core, relatively heavy-weight, is designed for very complex control logic, seeking to optimize the execution of sequential programs.

A GPU core, relatively light-weight, is optimized for data-parallel tasks with simpler control logic, focusing on the throughput of parallel programs.

The most prevalent is the GPU, originally designed to **perform specialized graphics computations** in parallel. Over time, GPUs have become more powerful and more generalized, enabling them to be applied to general purpose parallel computing tasks with excellent performance and high power efficiency

Heterogeneous Architecture

A typical heterogeneous compute node nowadays consists of two multicore CPU sockets and two or more many-core GPUs. A GPU is currently not a standalone platform but **a co-processor to a CPU.** Therefore, GPUs must operate in conjunction with a CPU-based host through a PCI-Express bus



Heterogeneous Architecture

A heterogeneous application consists of two parts:

➤ Host code

➤ Device code

- Host code runs on CPUs and device code runs on GPUs. An application executing on a heterogeneous platform is typically initialized by the CPU. The CPU code is responsible for managing the environment, code, and data for the device before loading compute-intensive tasks on the device.
- With computational intensive applications, program sections often exhibit a rich amount of data parallelism. GPUs are used to accelerate the execution of this portion of data parallelism and hence called **hardware accelerator**.

There are two important features that describe GPU capability:

➤Number of CUDA cores

➤ Memory size

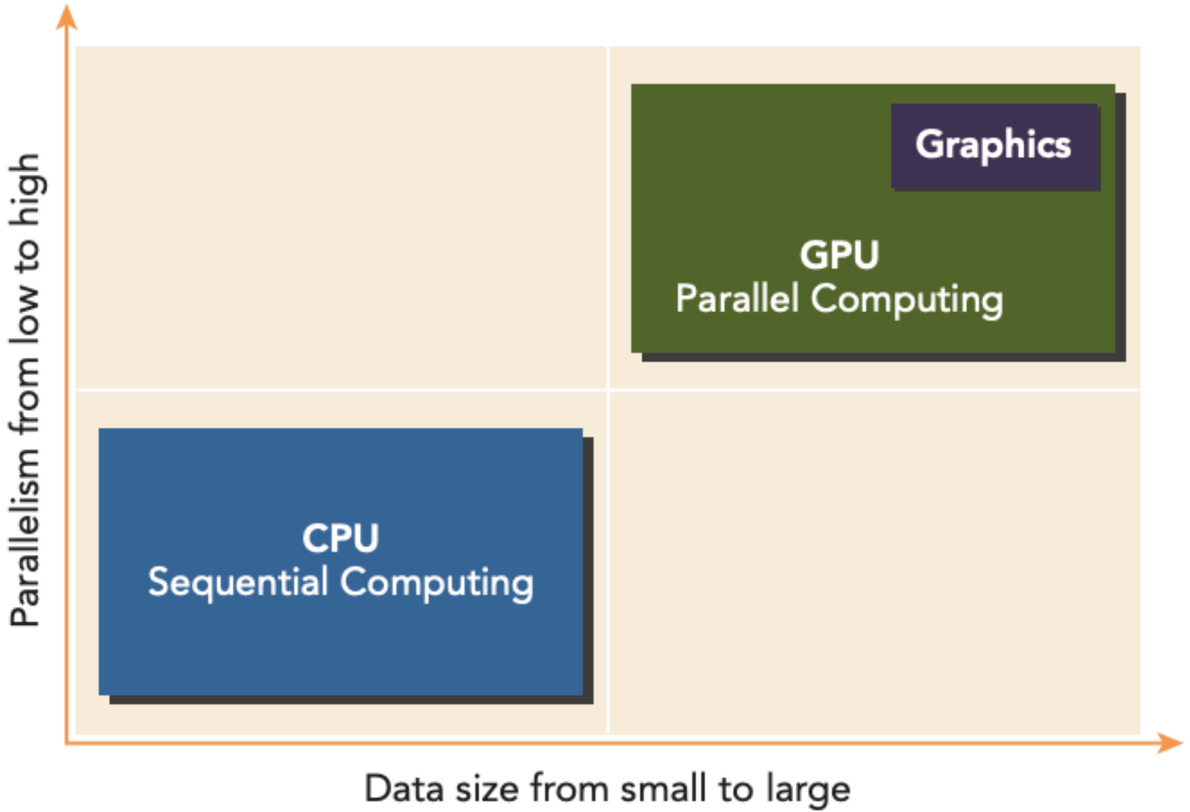
Accordingly, there are two different metrics for describing GPU performance:

➤➤ Peak computational performance

➤➤ Memory bandwidth

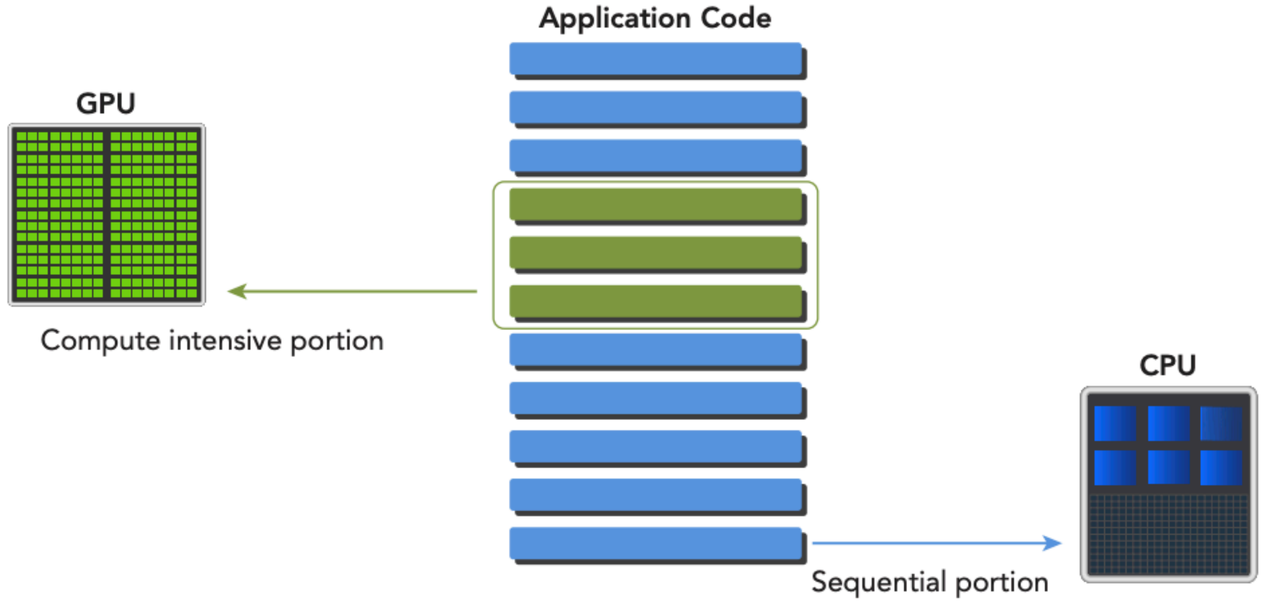
- Peak computational performance is a measure of computational capability, usually defined as how many single-precision or double-precision floating point calculations can be processed per second. Peak performance is usually expressed in **gflops** (billion floating-point operations per second) or **tflops** (trillion floating-point calculations per second).
- Memory bandwidth is a measure of the ratio at which data can be read from or stored to memory. Memory bandwidth is usually expressed in gigabytes per second, GB/s.

Paradigm of Heterogeneous Computing



Paradigm of Heterogeneous Computing

CPU + GPU heterogeneous parallel computing architectures evolved because the CPU and GPU have complementary attributes that enable applications to perform best using both types of processors. Therefore, for optimal performance **you may need to use both CPU and GPU for your application,** executing the sequential parts or task parallel parts on the CPU and intensive data parallel parts on the GPU.



Writing code this way ensures that the characteristics of the GPU and CPU complement each other, leading to full utilization of the computational power of the combined CPU + GPU system. To support joint CPU + GPU execution of an application, **NVIDIA designed a programming model called CUDA**.

CPU THREAD VERSUS GPU THREAD

Threads on a CPU are generally heavyweight entities. The operating system must swap threads on and off CPU execution channels to provide multithreading capability. Context switches are slow and expensive.

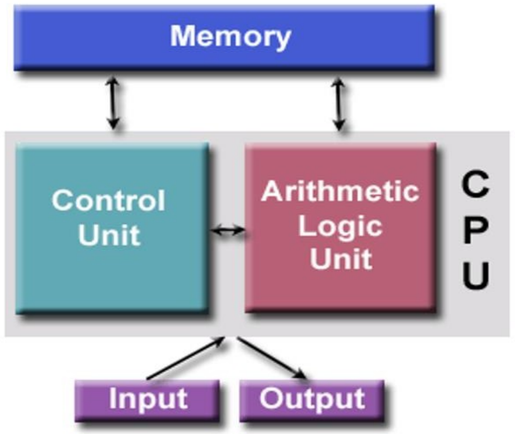
Threads on GPUs are extremely lightweight. In a typical system, thousands of threads are queued up for work. If the GPU must wait on one group of threads, it simply begins executing work on another.

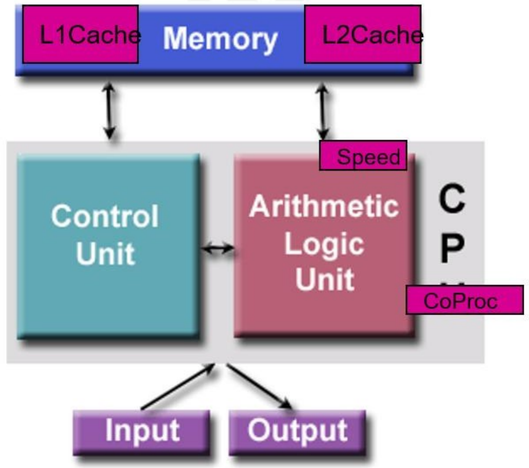
CPU cores are designed to minimize latency for one or two threads at a time, whereas GPU cores are designed to handle a large number of concurrent, lightweight threads in order to maximize throughput.

Today, a CPU with four quad core processors can run only 16 threads concurrently, or 32 if the CPUs support hyper-threading.

Modern NVIDIA GPUs can support up to 1,536 active threads concurrently per multiprocessor. On GPUs with 16 multiprocessors, this leads to more than 24,000 concurrently active threads.

Evolution of Architectures





Von Newman Architecture

Multicore Architecture

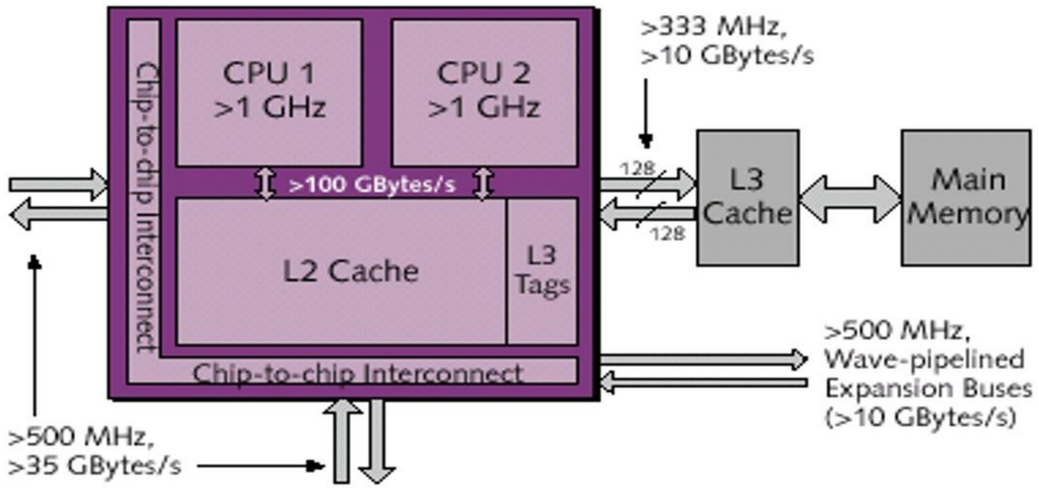
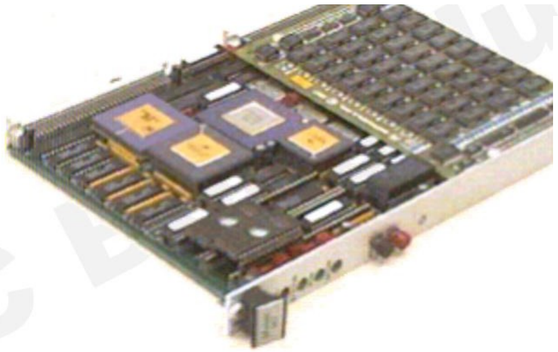


Fig: A 2-Processor Chip

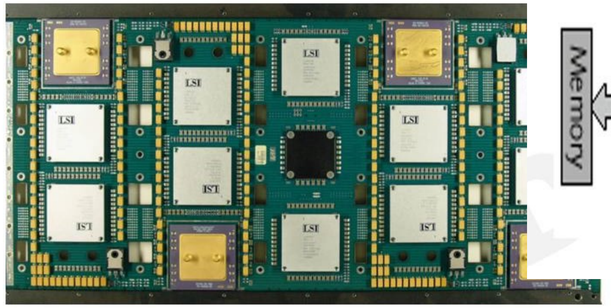
Fig: An Intel Core 2 Duo E6750 dual-core processor

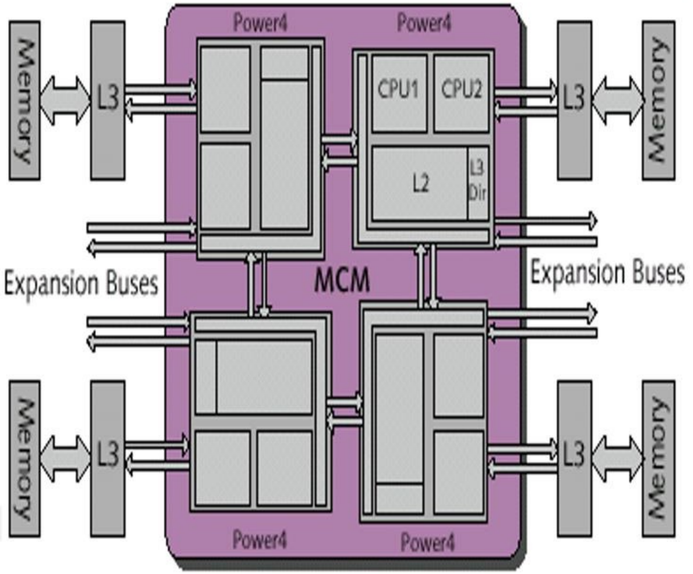
Evolution of Architectures

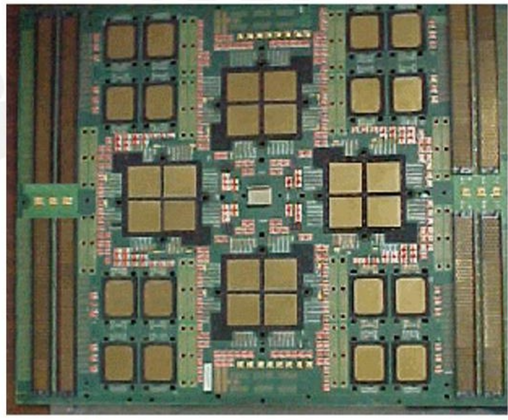
- **Vector processors**
  - ✓ able to run mathematical operations on multiple data elements simultaneously



- **Superscalar processors**
  - ✓ Instruction level parallelism with a processor
  - ✓ Intel 960, SunSparc, MIPS R, and AMD Am29000







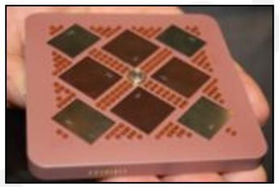
32-way System with 4 MCM and L3 cache

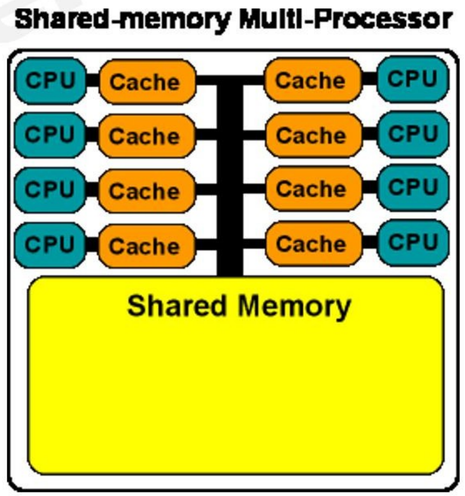
Multi Chip Module (MCM)

Parallel Architectures

Evolution of Architectures

- two or more identical processors are connected to a single shared main memory





Symmetric Multi Processors (SMP)

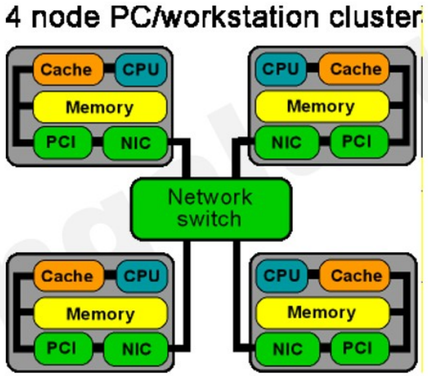
- Tightly coupled computers that work together closely as though they are a single computer

Super Computers

Evolution of Architectures

Other Parallel Systems

- **Distributed Computing**
  - ✓ parts of a program are run at the same time, on separate computers communicating over a network

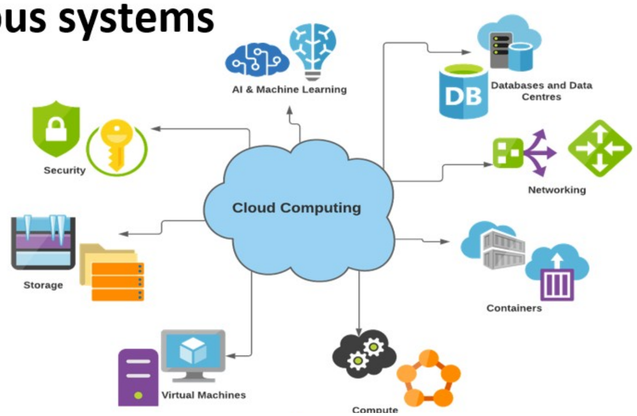


Disaggregated System on Chip

- Instead of being built on a single die and engineered from scratch, the company's chips could be composed of multiple smaller pieces (chiplets)
- "stitched" together as one.
- Universal Chiplet Interconnect (UCIe)

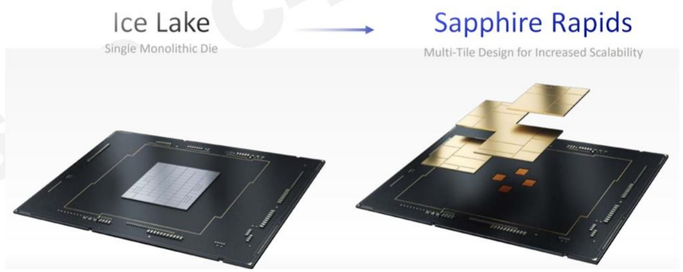
- **Grid Computing**
  - ✓ Aggregation, sharing of distributed heterogeneous systems

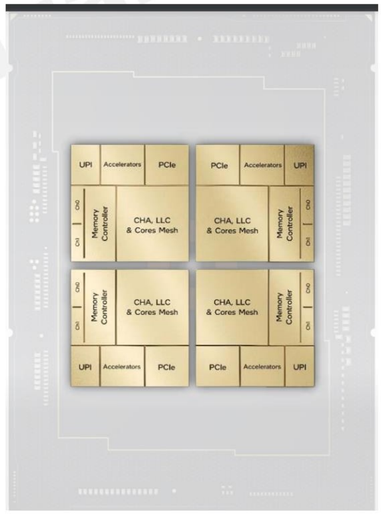
- **Cloud Computing**
  - ✓ on-demand available service – IaaS PaaS, SaaS
  - ✓ pay-as-you-use over the Internet



Intel (Sapphire Rapids)

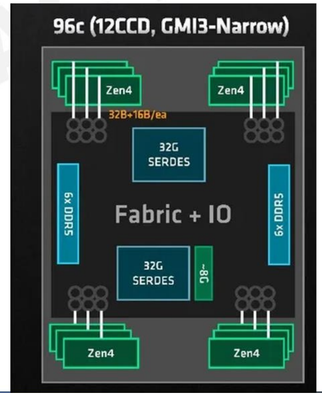
- 60 cores per package
- Multi-die architecture aims to minimize latency & maximize bandwidth
- Mesh topology b/w cores

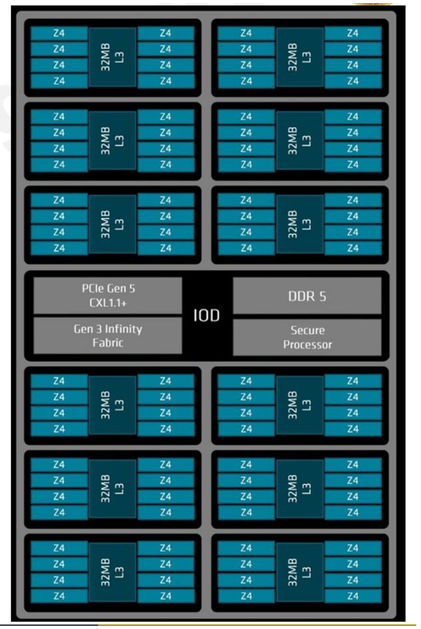




AMD (Genoa)

- Hybrid, multi-die architecture
- 96 cores (12* 8core dies)
- Bi-directional ring bus topology within 8 cores





Accelerators

FPGA

Field-Programmable Gate Arrays

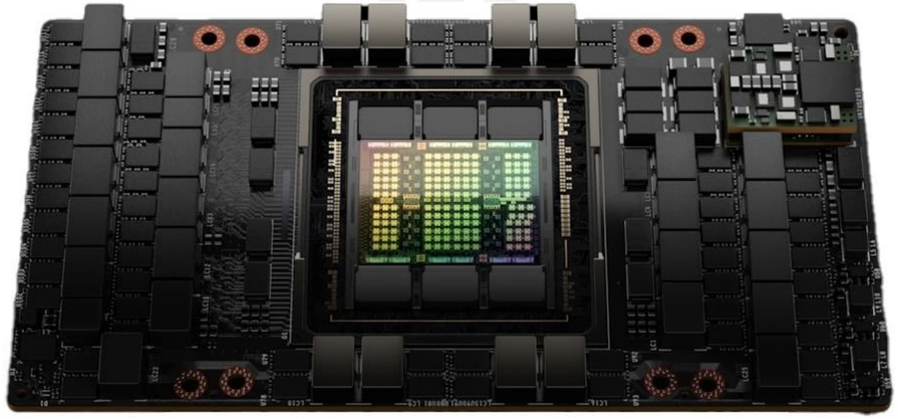
- ✓ computer chip that can rewire itself for given task
- ✓ Intel (Altera) and AMD (Xilinx)
- ✓ can be programmed with hardware description languages such as VHDL or Verilog

GPGPU

- ✓ General-purpose computing on graphics processing units (GPGPU)
- ✓ NVIDIA, Intel and AMD
- ✓ CUDA/OneAPI/ROCm programming environment

NVIDIA GPU

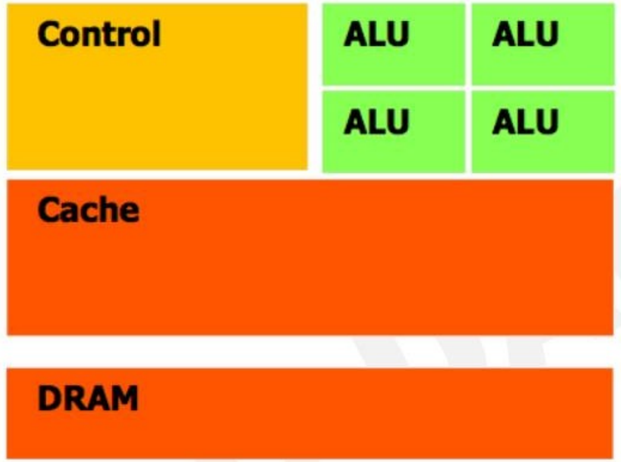
- **Hopper**
  - ✓ H100
  - ✓ 60TFLOPS for FP32
  - ✓ 1000 Tflops with FP16
  - ✓ 8 GPC – 144 SM/GPU
  - ✓ 14592 FP32/GPU



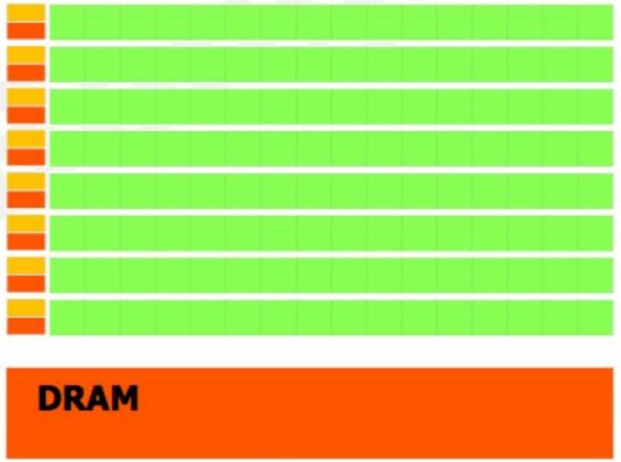
AMD GPU

- **MI200**
  - ✓ 45.3 TFLOPS Vector FP64
  - ✓ 90.5 TFLOPS Matrix FP64
  - ✓ 362 TFLOPS Matrix FP16
  - ✓ 256 CUs (active 224CUs)

CPU vs GPU



CPU

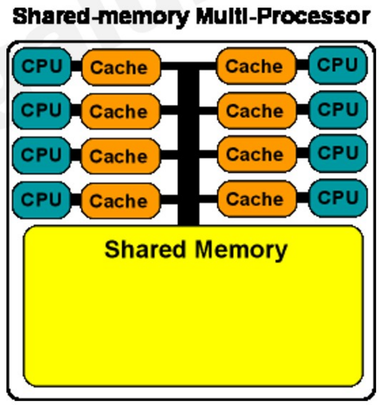


GPU

MEMORY ARCHITECTURES

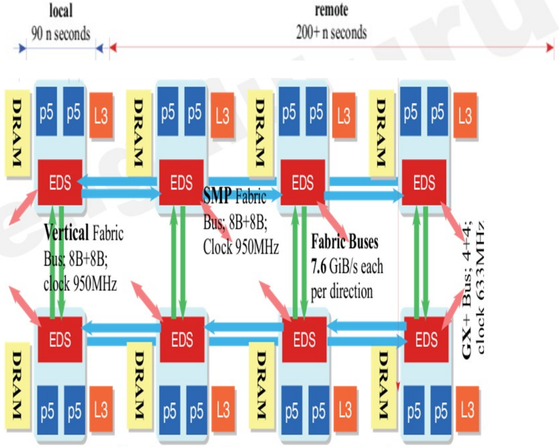
Shared Memory - UMA

- ✓ Uniform Memory Access (UMA)
- ✓ Example - SMP
- ✓ Identical processors
- ✓ Equal access & access times to memory
- ✓ Sometimes called Cache Coherent UMA (CC-UMA). Cache coherency is accomplished at the hardware level.
- ✓ Contention - as more CPUs are added, competition for access to the bus leads to a decline in performance.



Shared Memory - NUMA

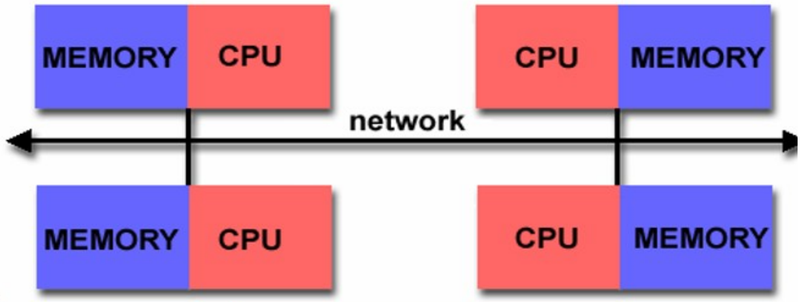
- ✓ Non-Uniform Memory Access
- ✓ Not all processors have equal access time to all memories
- ✓ Memory access across link is slower
- ✓ If cache coherency is maintained - CC-NUMA



- ✓ Designed to overcome scalability limits of SMPs.
- ✓ Can support up to 1024 processors.
- ✓ Processors directly attached to a memory module experience lower latency than those attached to "remote" memory modules.

Distributed Memory Architecture

- ✓ Processors have their own local memory. So operates independently.
- ✓ No concept of global address space across all processors.
- ✓ Changes in local memory have no effect on memory of other processors. Hence, cache coherency does not apply.
- ✓ Data access between processors is defined by programmer ; explicitly define how and when data is communicated.
- ✓ Synchronization between tasks is also programmer's responsibility.
- ✓ The network "fabric" used for data transfer varies widely, though it can be as simple as Ethernet.



Evolution of Super Computers

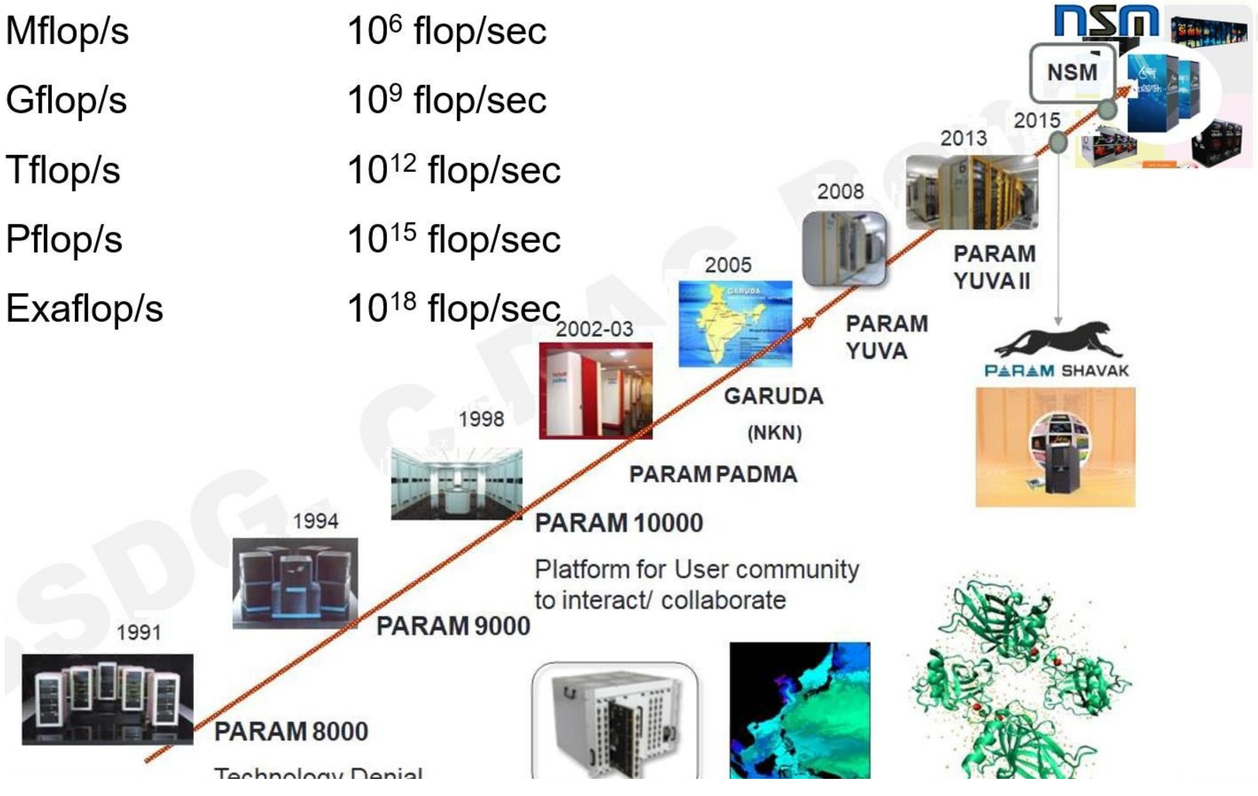
- ✓ 1961 -- IBM 7030, 1.2 MIPS

- ✓ 1964 -- CDC 6600 , 6 MFLOPS
- ✓ 1970s – Cray-1 , a vector processor , 160 MFLOPS

- ✓ 1985 – Cray-2, 4 processor liquid cooling , 1.9 GFLOPS
- ✓ 1996 – ASCI Red , Intel, Sandia National Labs, 1.3 TFLOPS

Indian/C-DAC line of Supercomputers

Mflop/s	10^6 flop/sec
Gflop/s	10^9 flop/sec
Tflop/s	10^{12} flop/sec
Pflop/s	10^{15} flop/sec
Exaflop/s	10^{18} flop/sec



Top Supercomputer Speeds over years

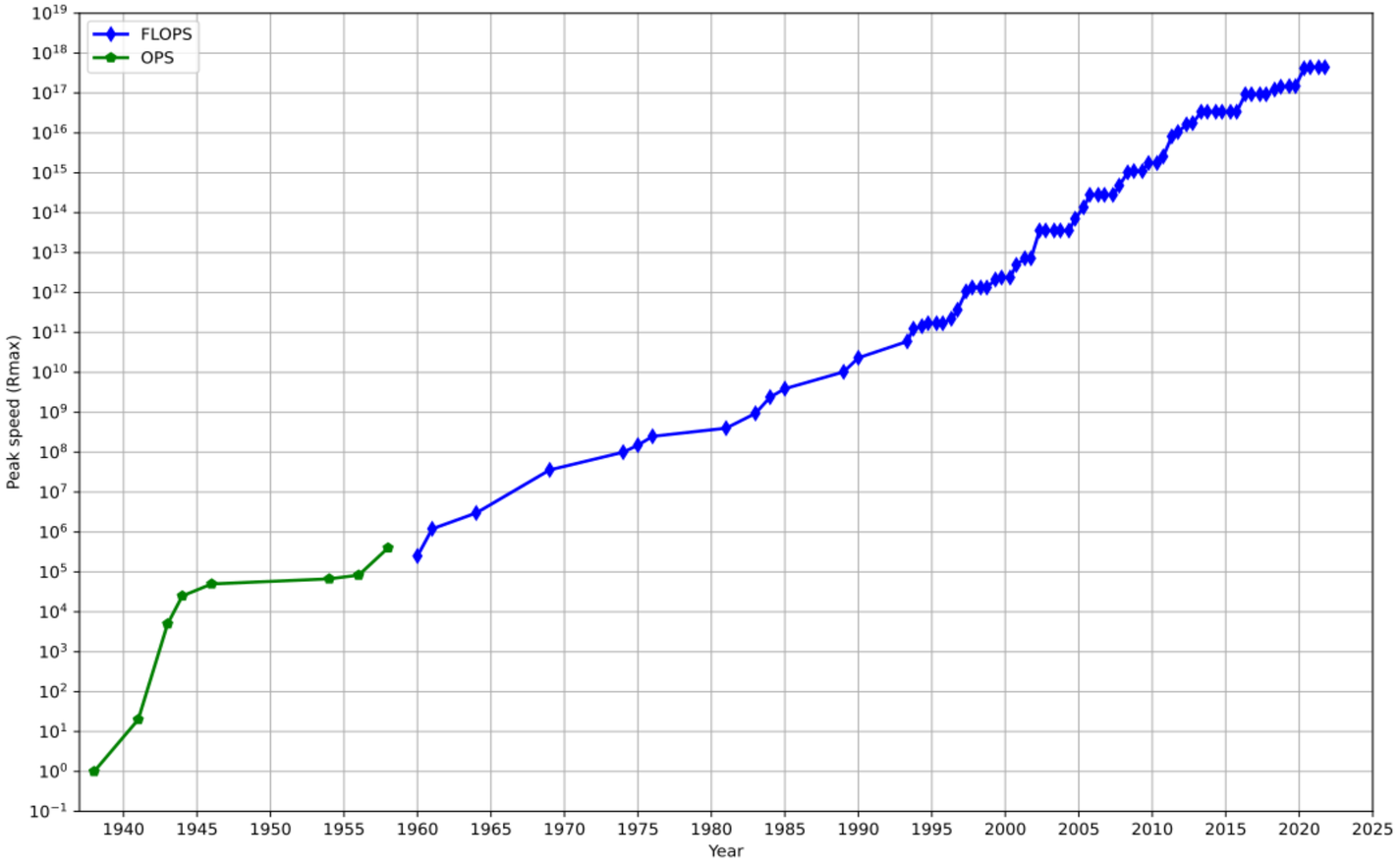


Image Source: Wikipedia

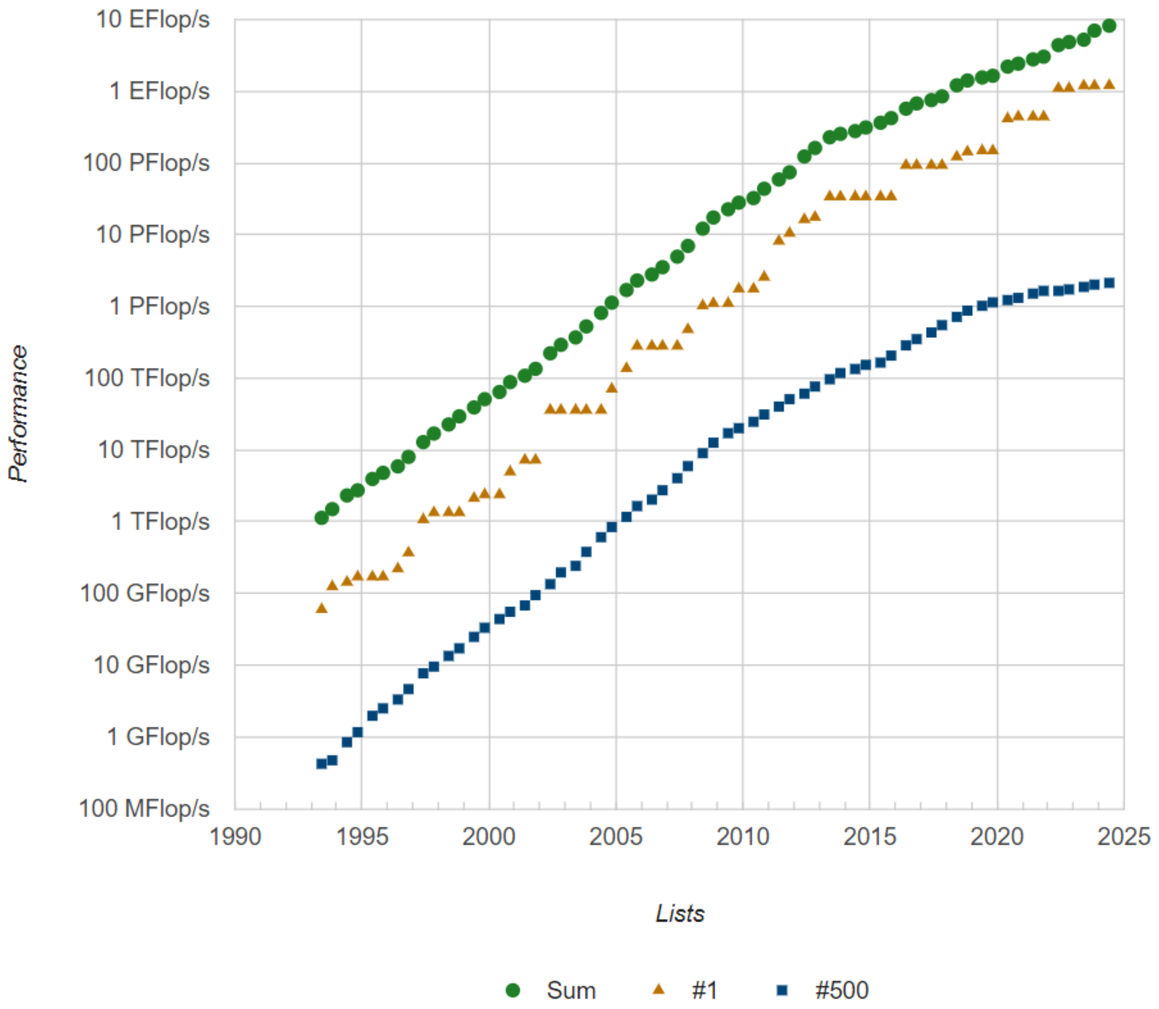
How is power of supercomputer measured

- Floating point operations per second is used as a measure of performance of supercomputers
  - Represented by FLOPS, flops or flop/s

Name	Represented as	Value
teraFLOPS	TFLOPS	10^{12}
petaFLOPS	PFLOPS	10^{15}
exaFLOPS	EFLOPS	10^{18}
zettaFLOPS	ZFLOPS	10^{21}
yottaFLOPS	YFLOPS	10^{24}

- Top500.org site lists world's top supercomputers using a program called High Performance Linpack (HPL)

Performance development over the years



- Top supercomputer in the world is Frontier, with HPL result of 1206 petaflops i.e. 1.2 Exaflops
- Has 8,699,904 combined CPU and GPU cores
- Installed at Oak Ridge National Laboratory, DoE, USA

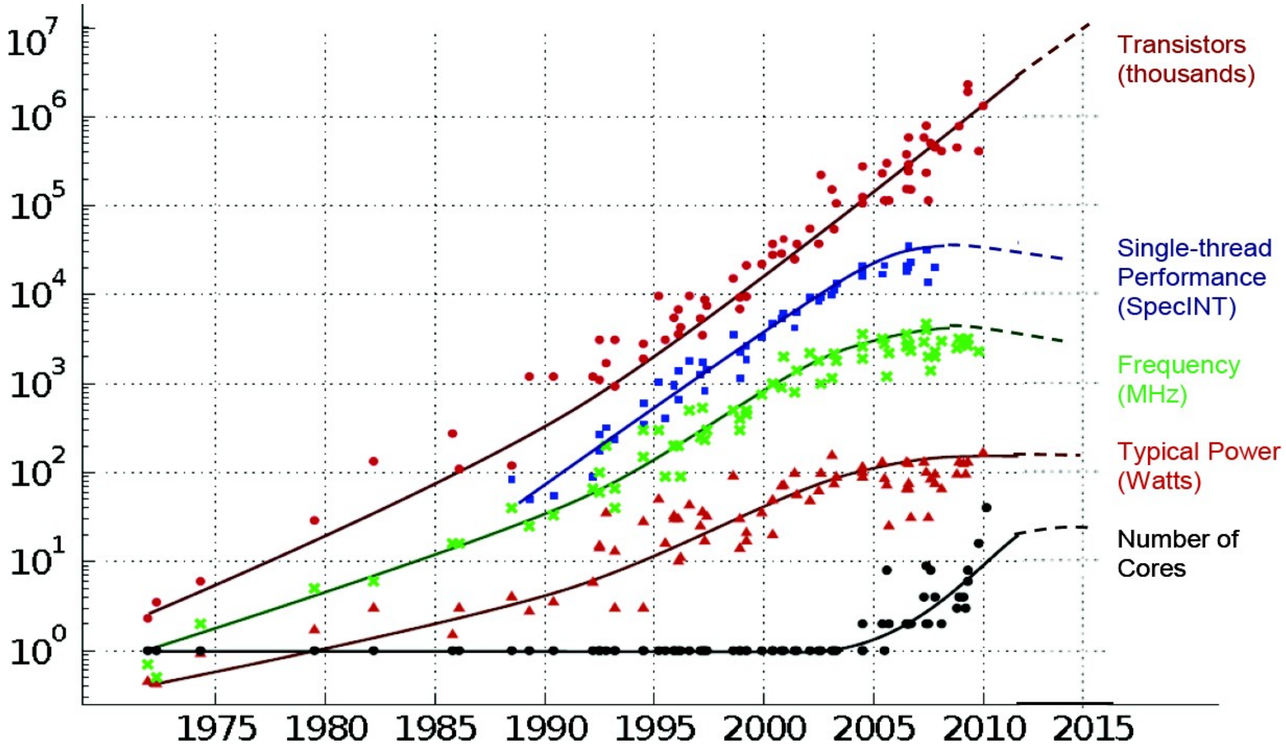
Source: www.top500.org

Parallel Computing Definition

- "A large collection of processing elements that can communicate and cooperate to solve large problems fast"

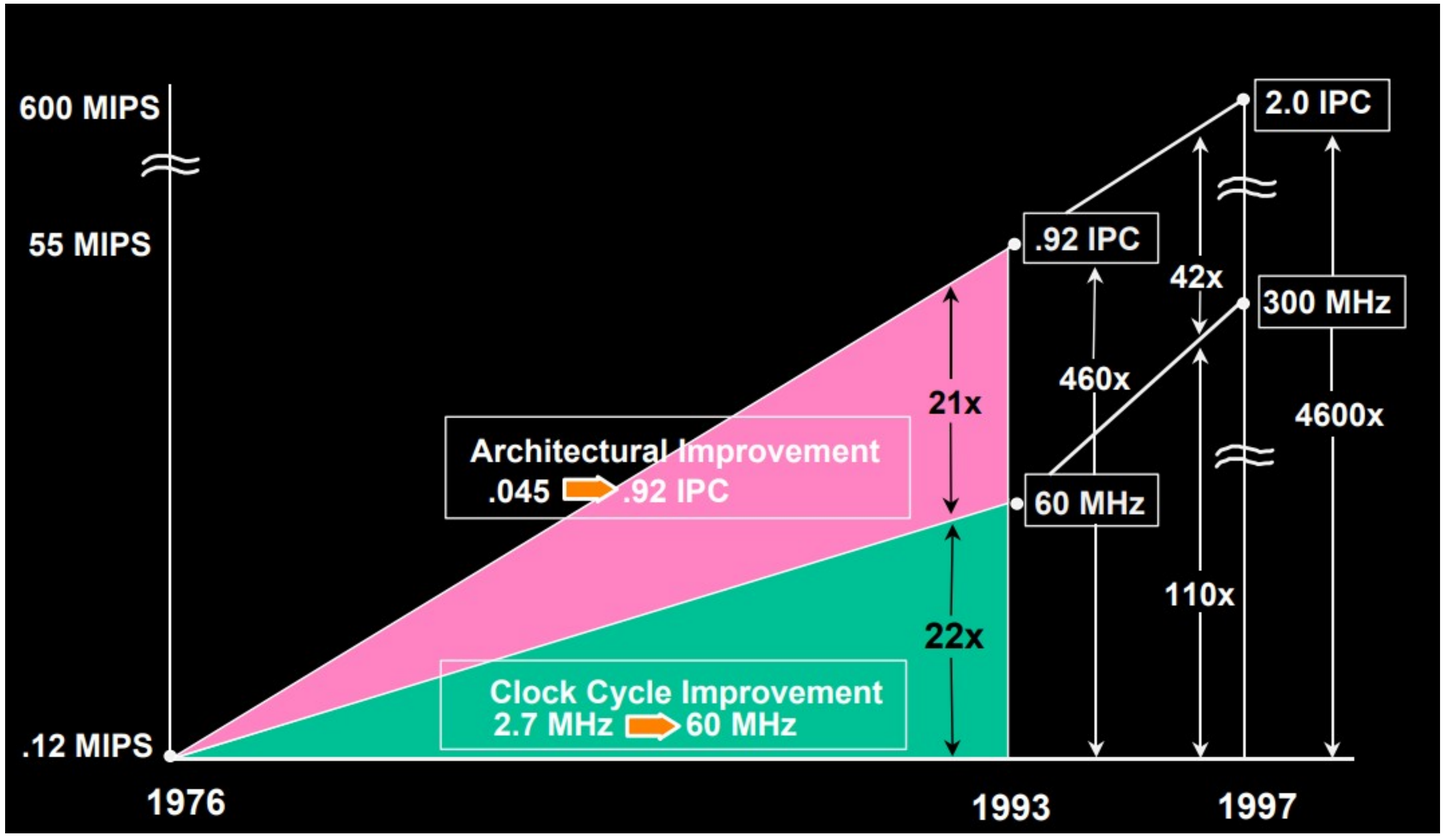
  - George S. Almasi and Allan Gottlieb

Microprocessor Trends Over the Years



Source: *35 Years of Microprocessor Trend Data* by M. Horowitz et al

Microprocessor Performance Growth



Source: Joel S. Birnbaum at Microprocessor Forum 1997

Methods of Improving Performance

- Instruction level Parallelism: Executing multiple instructions within
- Shared memory systems: Multiple CPUs sharing memory
- Distributed memory systems: A set of computers connected using
- Graphics Processing Units: Use of many light-weight cores as
- Field Programmable Gate Arrays: Embedding certain algorithms in
- Application Specific ICs: Hardware platforms for accelerating certain workloads

Parallel Computing Basics

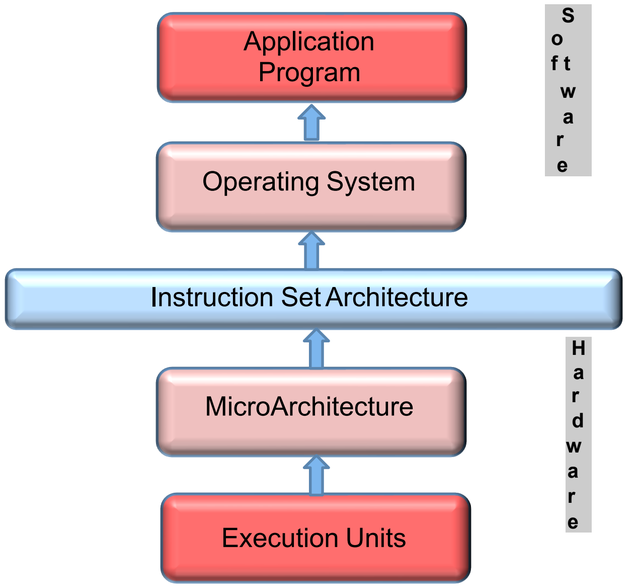
- Principle: Large problems can often be divided into smaller ones, which are then solved concurrently
- Parallel Processing: Solving a task by making simultaneous use of multiple processing elements
- Distributed Computing: Solving a task by simultaneous use of multiple processing elements, typically isolated and heterogeneous in nature

What is a High Performance Computing?

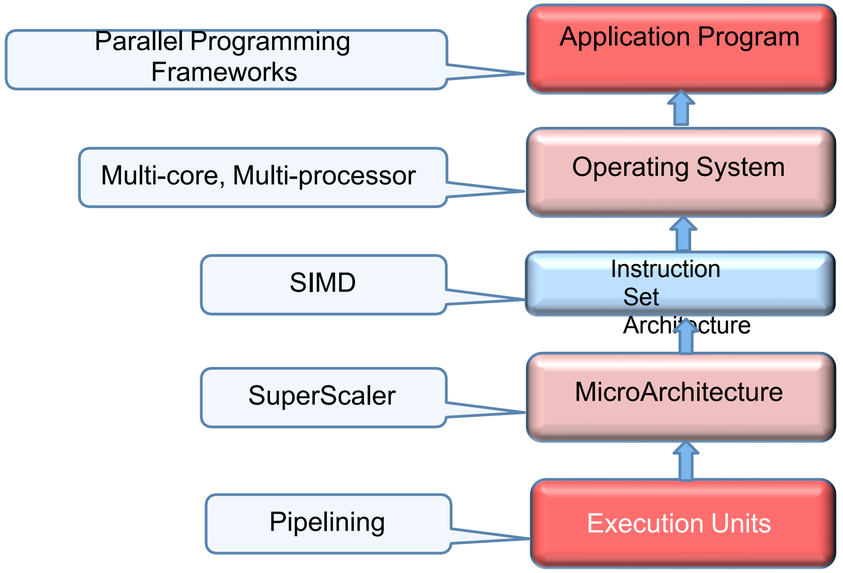
- Makes use of parallel processing for running advanced application programs
- Is a set of Computing technologies for very fast numeric simulation, modeling and data processing
- Is employed for specialised applications that require lot of mathematical calculations
- Uses computer power to execute a few applications extremely fast
- A supercomputer is a system that performs at or near the currently highest operational rate for computers.

Parallelism in Architecture

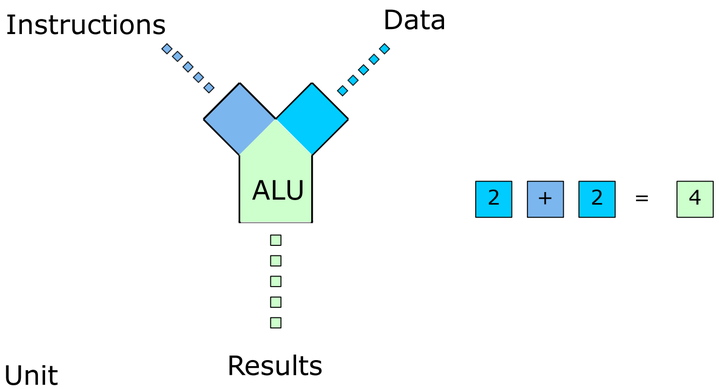
Computing Abstraction Layers



Parallelism associated with Layers



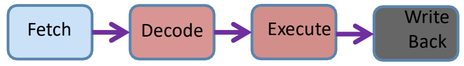
The heart of the processor - Simplified

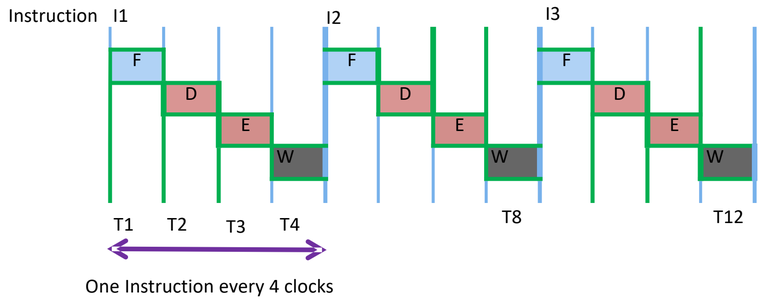


ALU : Arithmetic Logic Unit

Non-Pipelined instruction execution

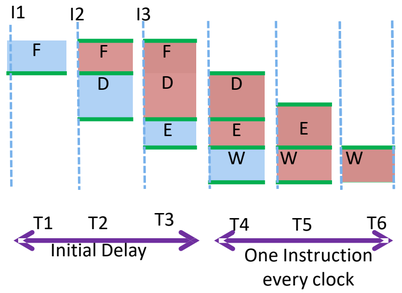
- An instruction typically takes 4 cycles

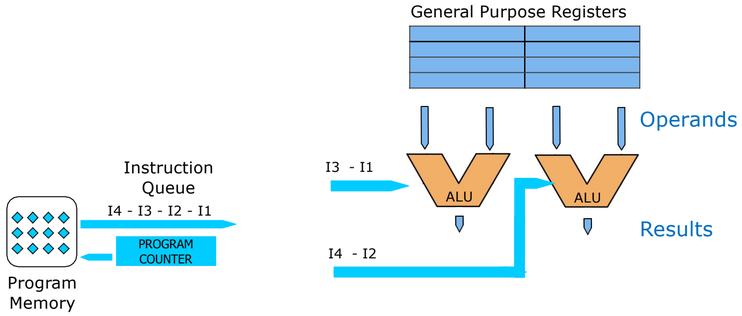




Pipelining

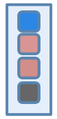
- By overlapping the activities in the stages, effectively one instruction is executed in every tick
- A new instruction enters every clock
- Instruction parallelism = No. ofpipeline stages
- Pipeline will not be able to function smoothly when it faces hazards
- For example, control hazard which results due to branch or any instruction that changes the Program Counter
  - Inserts bubbles in the pipeline





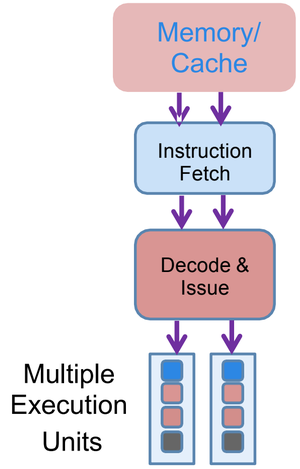
Superscalar

- CPU internally contains more than one execution unit
- Instructions are sequential, but issued multiple at a time



Pipelined
Execution
unit

Results in effective execution of one instruction per clock



Results in execution of up to 2 instruction per clock

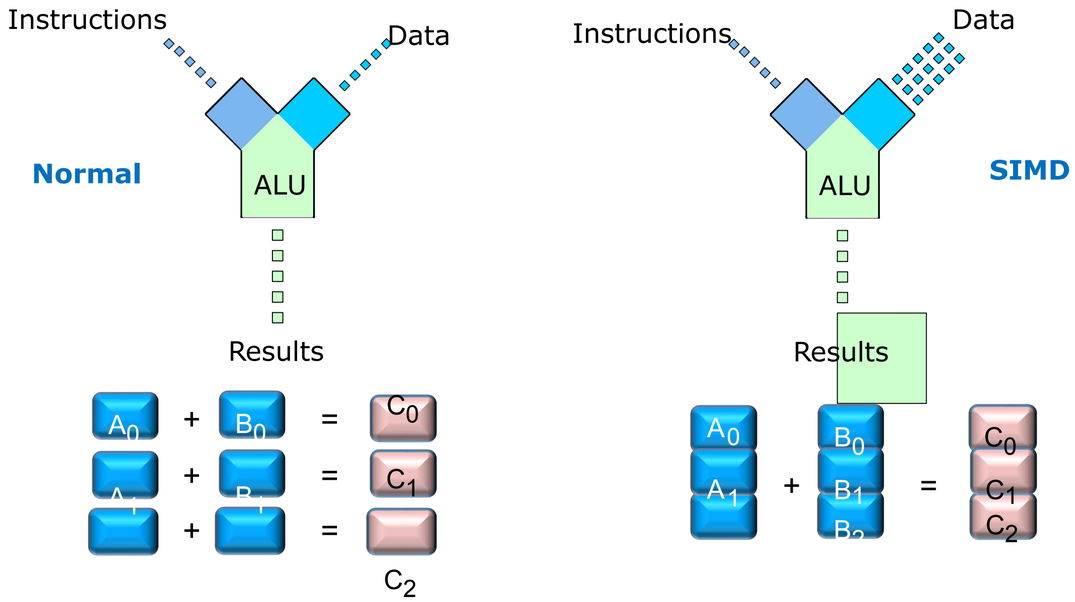
SIMD in Instruction Set Architecture

Advanced Vector eXtensions

- Useful for algorithms that can take advantage of SIMD
- AVX were introduced by Intel and AMD in x86
- Using AVX-512, applications can pack
  - 32 double precision or 64 single precision floating point operations

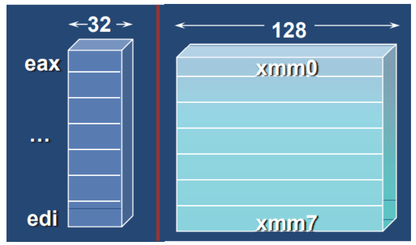
    or
  - Eight 64-bit and sixteen 32-bit integers operations
- Accelerates performance for workloads such as
  - Scientific simulations, artificial intelligence (AI)/deep learning, image and audio/video processing

SIMD Instructions



Parallelism inside the ALU – SIMD Instructions

- Single instruction takes multiple data operands packed in long word
- Increase processor throughput by performing multiple computations in a single instruction
- Streaming SIMD Extension (SSE) is architectural extensions introduced by Intel first in Pentium 3

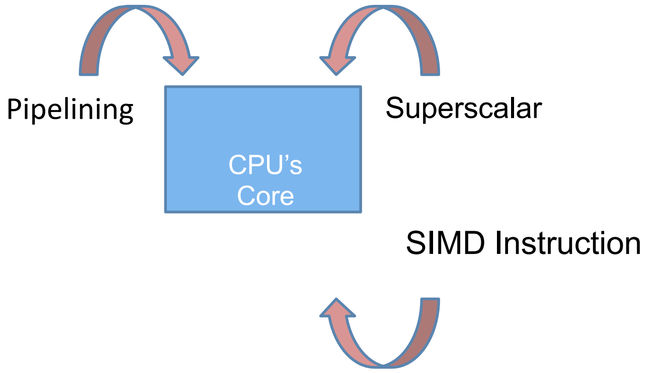


Integer Registers
- Fourteen 32-bit registers
- Holds Scalar data & addresses

SIMD Registers – SSE
- Eight 128-bit registers
- Hold data only:
  - 4 x single FP numbers
  - 2 x double FP numbers
  - 128-bit packed integers

Source: www.intel.com

Putting together the parallelism inside a core



Multicore CPU

- Single computing component with two or more independent processing units
- Each unit is called a core, which read and execute program instructions independently

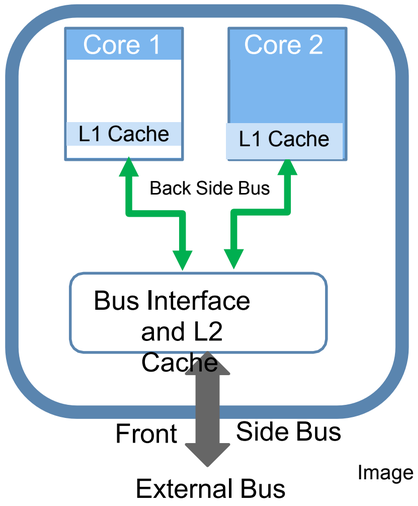
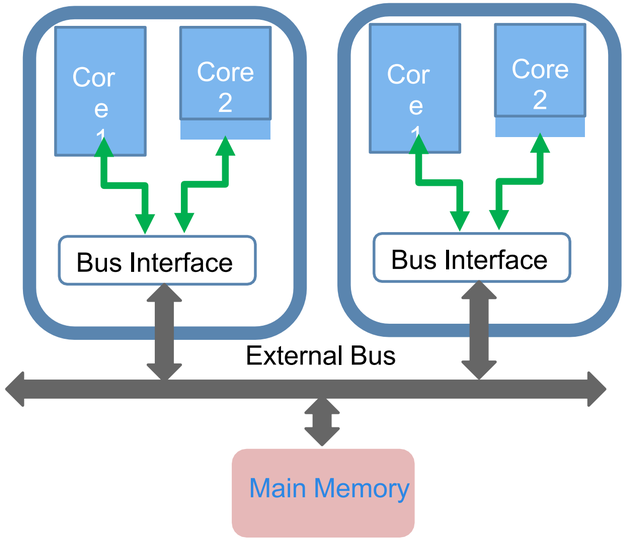


Image source: www.intel.com



Dual socket Motherboard in a node

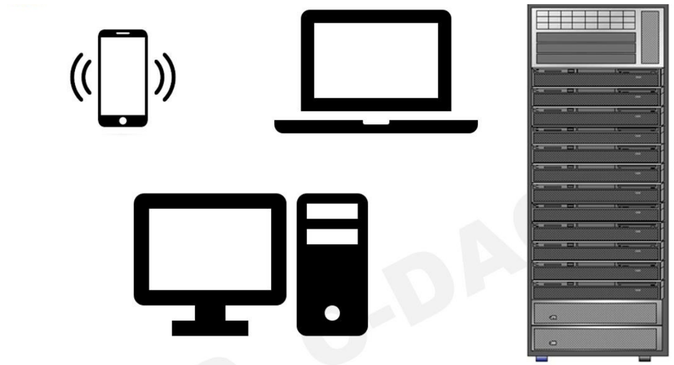
Serial Computing

If 1 person takes 15 hours to complete the job; then

How long will it take if 15 people work together

PARALLEL Processing or Parallelism!

- Saves total time taken
- Solve larger problems

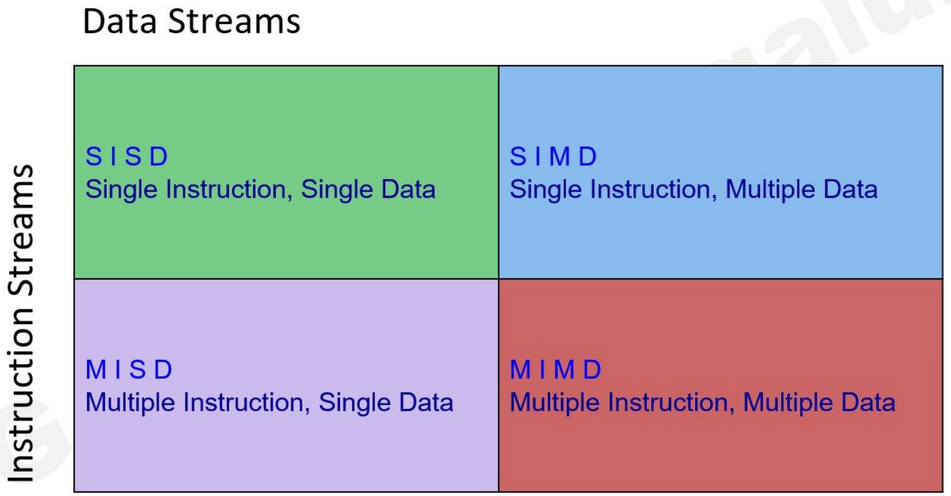


"Parallelism is a necessity of today's computing!!"

Terminologies

- **Parallel Processing**
  - ✓ **Processing multiple tasks simultaneously on multiple processors is called parallel processing.**
- **Parallel Programming**
  - ✓ **Software methodology used to implement parallel processing.**
- **Parallel Computing**
  - ✓ **Term that encompasses all the technologies used in running multiple tasks simultaneously on multiple processors**

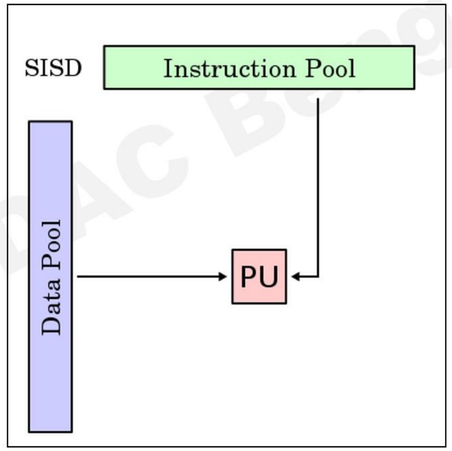
Flynn's Taxonomy

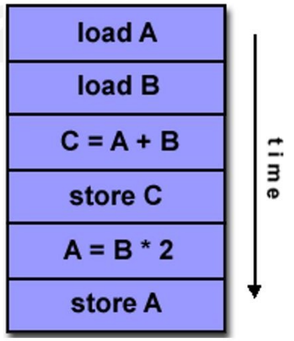


1. SISD

UNIVAC

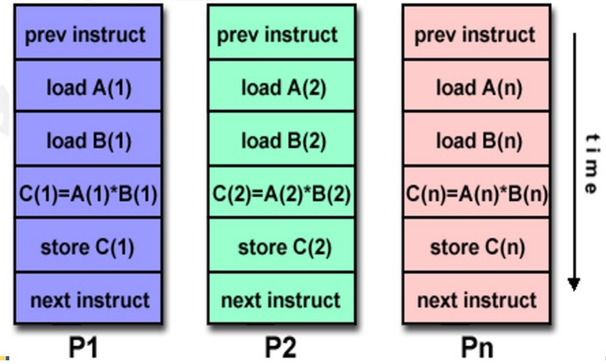
IBM 360

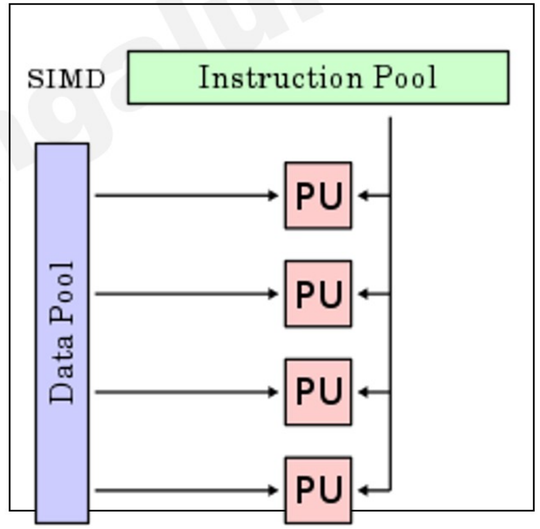




2. SIMD

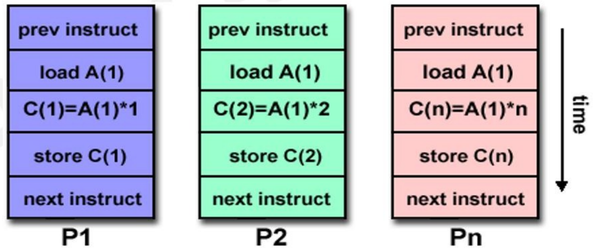
- ✓ Processor Arrays: Connection Machine CM-2, MasPar MP-1 & MP-2, ILLIAC IV
- ✓ Vector Pipelines: IBM 9000, Cray X-MP, Y-MP & C90, Fujitsu VP, NEC SX-2, Hitachi S820, ETA10
- ✓ Most modern computers, particularly those with graphics processor units (GPUs) employ SIMD instructions and execution units.

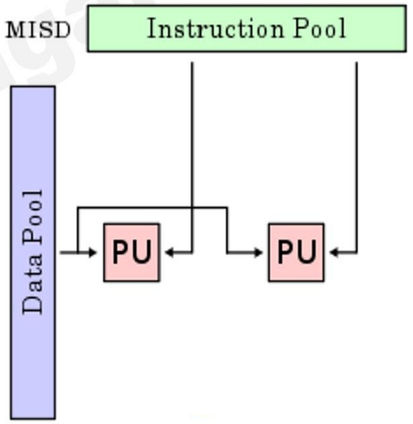




3. MISD

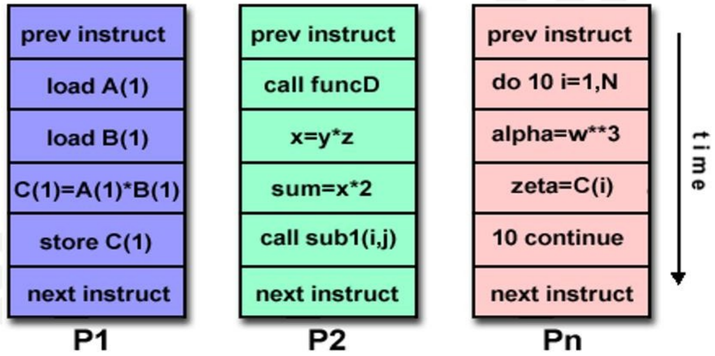
- Some conceivable uses might be:
  - ✓ multiple frequency filters operating on a single signal stream
  - ✓ multiple cryptography algorithms attempting to crack a single coded message.

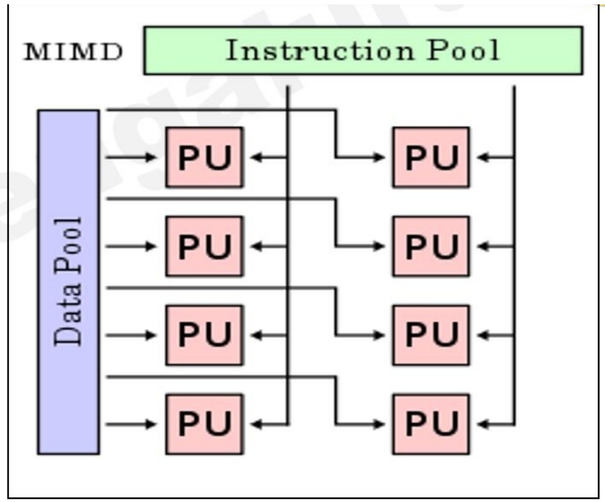


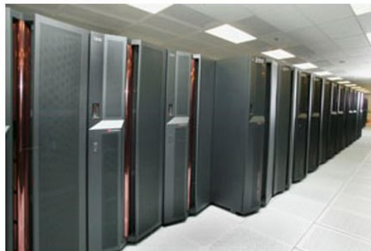


4. MIMD

- ✓ Most current supercomputers, networked parallel computer clusters and "grids", multi-processor SMP computers, multi-core PCs.







Paradigms

- On a single machine with shared memory
  - ✓ OpenMP 3 and lesser
  - ✓ Pthreads
- On machines connected via network and have no shared memory
  - ✓ MPI
  - ✓ PGAS
- Accelerators, offload tasks from CPU to it
  - ✓ CUDA
  - ✓ OpenACC /OpenMP 4
- Heterogenous
  - ✓ SYCL programming (implementations like ROCm /HIP / OneAPI)

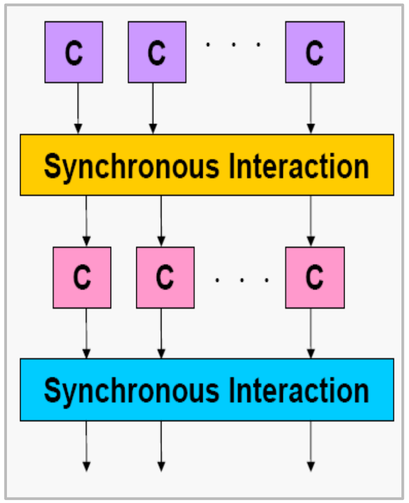
Ideal Situation

- Each Processor has a Unique work to do
- Communication among processes is largely unnecessary
- All processes do equal work

Parallelization Paradigms

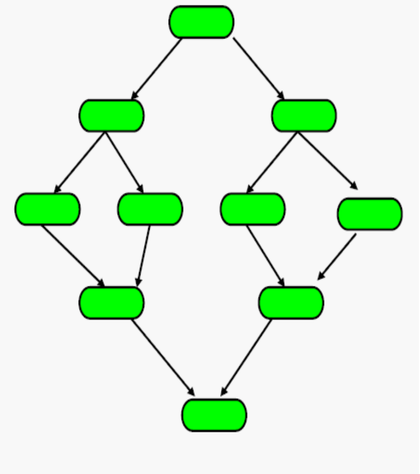
- Phase parallel
- Divide and conquer
- Pipeline
- Process farm
- Domain Decomposition

Phase Parallel Model



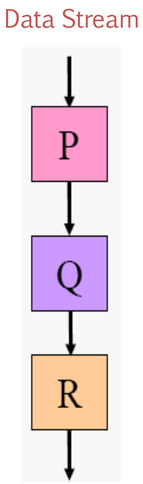
- The parallel program consists of a number of super steps, and each has two phases.
- In a computation phase, multiple processes each perform an independent computation.
- In interaction phase, the processes perform one or more synchronous interaction operations, such as a barrier or a blocking communication.

Divide and Conquer model



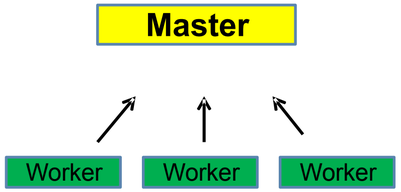
- A parent process divides its workload into several smaller pieces and assigns them to a number of child processes.
- The child processes then compute their workload in parallel and the results are merged by the parent.
- This paradigm is very natural for computations such as quick sort.

Pipeline Model



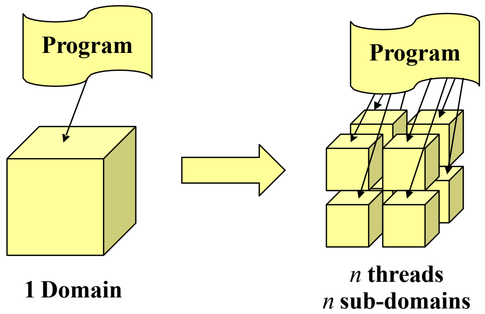
- In pipeline paradigm, a number of processes form a virtual pipeline.
- A continuous data stream is fed into the pipeline, and the processes execute at different pipeline stages simultaneously.

Process Farm Model



- Also known as the master-worker paradigm.
- A master process executes the essentially sequential part of the parallel program
- It spawns a number of worker processes to execute the parallel workload.
- When a worker finishes its workload, it informs the master which assigns a new workload to the slave.
- The coordination is done by the master.

Domain Decomposition



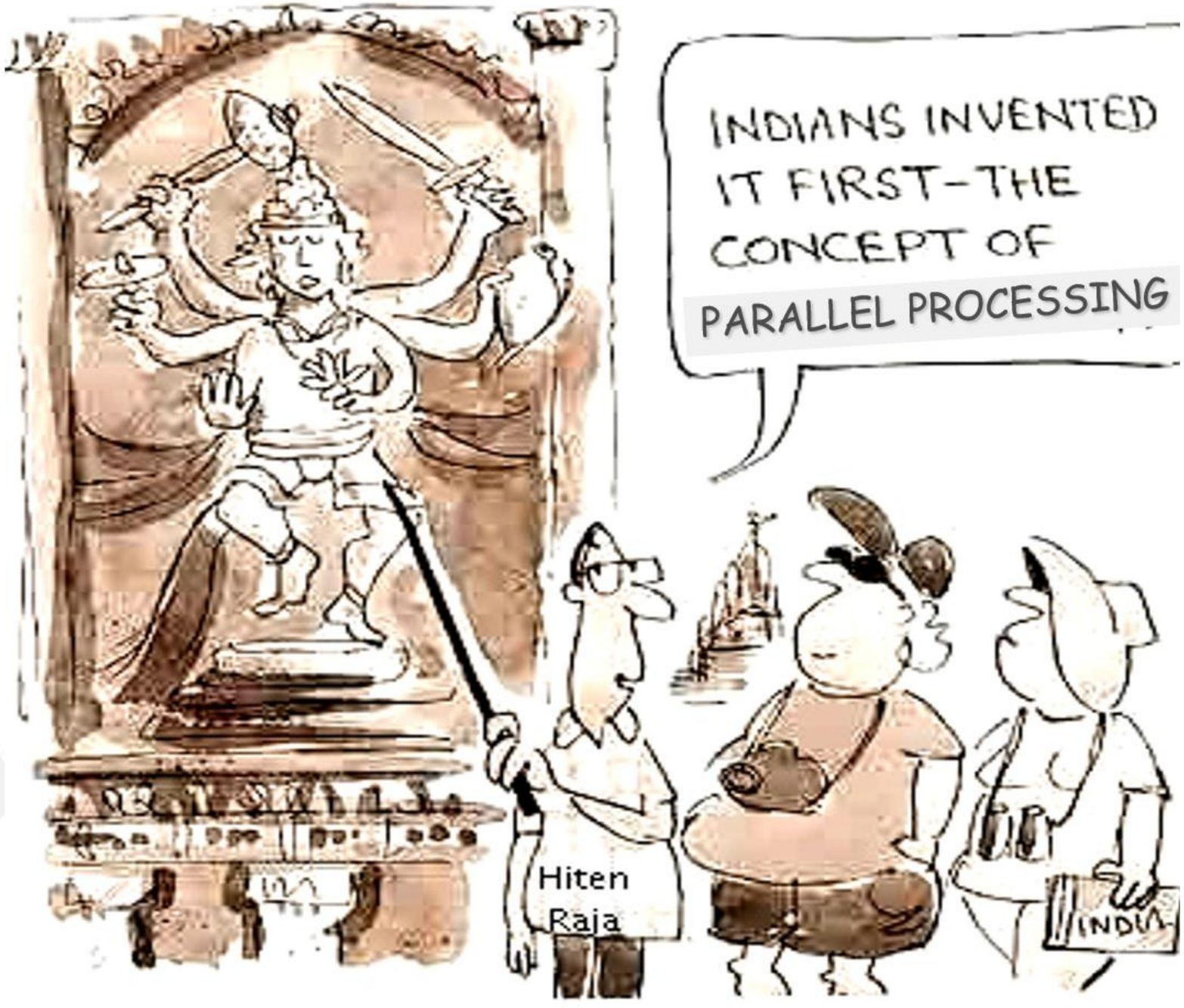
- This methods solve a boundary value problem by splitting it into smaller boundary value problems on subdomains
- Then iterating to coordinate the solution between adjacent subdomains.

Advantages of Parallel Computing

- Overcome limitations of single CPU systems
  - Sequential systems are slow
    - Calculations make take days, weeks, years
    - More CPUs can get job done faster
  - Sequential systems are small
    - Data set may not fit in memory
    - More CPUs can give access to more memory
- The advantages thus, are
    - Save time
    - Solve bigger problems

Summary

- Parallelism is the fundamental technique of HPC
- Scaling up of performance needs appropriate communication network and software stack
- Increasing the number of CPUs was the traditional method
- Complementing them with accelerators which uses light-weight, application specific cores helps in improving performance of certain workloads
- Appropriate algorithm complemented by a combination of frameworks that takes advantage of the underlying hardware helps in deriving optimum performance


INDIANS INVENTED IT FIRST-THE CONCEPT OF PARALLEL PROCESSING
Hiten
Raja
INDIA