1) What is HPC?
2) Importance of HPC.
3) Need of HPC — time ↓ / No of op./sec ↑ / Fast computing.
4) Challenges — Cost. / Scalability. / Data Management / programming. / S/W & Tools. / Power consumption & cooling
5) Applications — Scientific Research / Weather forecasting / Healthcare / Energy / Env. studies / Engg & Design.

① HPC is the practice of combining computing power to deliver far greater performance than a typical desktop/workstation in order to solve complex problems in science, engineering & business.

② - Discoveries / Innovations / quality of life
- Foundation for scientific / Industrial advancements.
- IOT / AI / 3D imaging & evolving applications
- complex modelling problems. — AI / Nuclear physics / climate modeling
- Business uses / data warehouse transaction processing.

Absent

2 32 64
6 39 65
7 41 * 67
14 49 *
15 53 *
16 54
17 57
21
~~22~~
26

6, 10, 11, 14, 18, 19 20 13.
23, 27, 31, 38, 37 39.
56 62. 48
(Present)

P > Fetch → Decode → Execute.

> Pipelining & parallism.

> Performance:

1. Processor speed
2. memory
3. storage
4. I/O Devices
5. soft/w opt : S/W / Appl

Also, $P = 1/\text{Execution time}$.

> ↑Perf. Benefits:

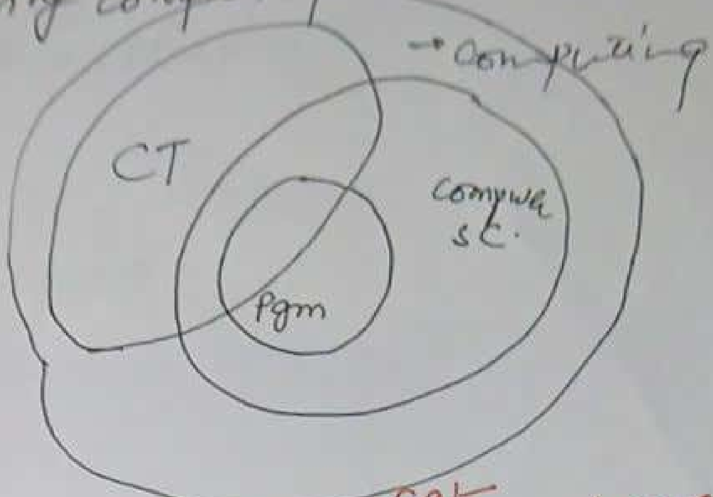
1. Increased productivity
2. Improved user experience
3. faster Data Processing
4. Enhanced gaming & MM performance
5. Better Eff.

foundation for computing-

computational Thinking is an interrelated set of skills & practices for solving complex problems.

> A problem-solving approach that utilize computer sc. concepts

> A way of thinking



Eg:-
1) planning a route
2) organising college schedule.
3) solving puzzle.
4) Building LEGO.

> computing: using computers & tech.

> A field of study & application.

> The use of computers & tech. to solve problems manage information & automate processes.

> CT.

Decomposition. → Breaking big problems into smaller, easier to manage problems.

↓

Pattern Recognition → Analyzing & Looking for a repeating sequence.

↓

Abstraction → removing unnecessary parts. one solution work for many problems.

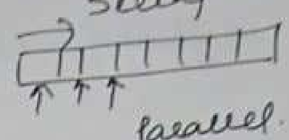
↓

Algorithmic design → steps-by-step instn how to do something

Parallel Programming

Q: What is result of 1 ... 2nd

> Seq vs Parallel.
> Simultaneous Execution
> Multiple Processes
> Parallel computing

eg 

> History:
> Started in 1950
> 60-70's — supercomputers.
> 1981 : Caltech concurrent computation
64 Intel 8086. Processors.
↓
Supercomp.

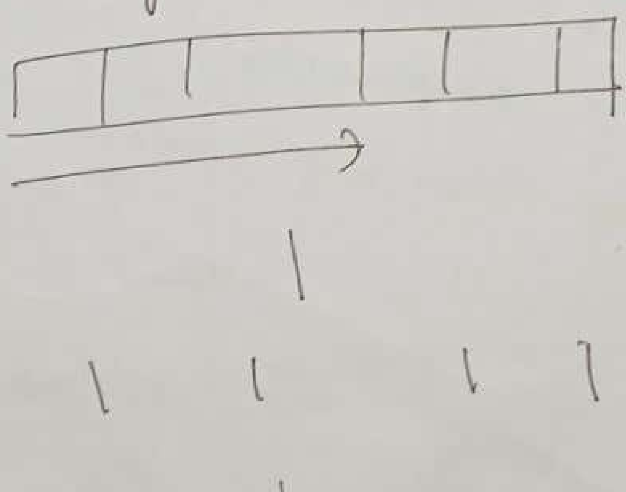
* clusters (Multiple Processors)
* Multicore Processors.

* Both H/W & S/W parallelism
* Extremely complex & hard to parallelize.
* Parallelism can improve a pgm as per Amdahl's law

$$\text{Speedup} = \frac{1}{1-p}$$

p = %age of code parallelizable

eg: Array summation.



Models:
- Shared memory. — OpenMP, Pthreads
- Distributed memory — MPI
- GPU Programming — CUDA, Open CL
- Hybrid model — MPI + openMP.

Applications:

* Scientific simulations (climate modelling)
* Big Data Analytics.
* AI/ML training (NN)
* Game Development
* financial modelling
* RT systems.

Challenges:

* Race conditions.
  Two threads accessing shared Data
  Two processors accessing shared memory.
* Deadlocks.
  processes waiting for each other indefinitely.
* Load Balancing
  Distributing tasks evenly b/w processors.
* Debugging complexity
  Harder than seq. code.

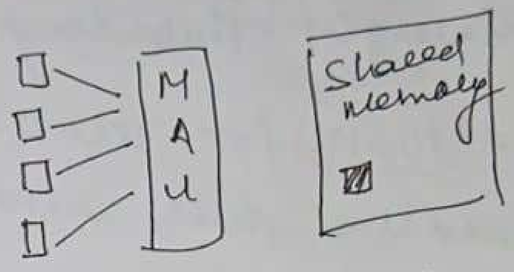
Platforms!

CPU based
- Open MPI
- MPI
- P threads

GPU ↔
- CUDA
- ~~open mp~~ open CL
- TBB
- ~~Open ACC~~
- ~~open CL~~

Open MP

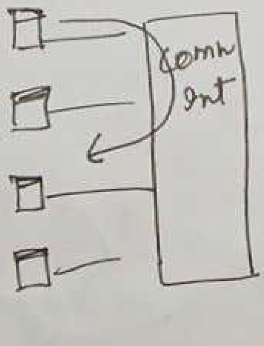
- Parallel Pgm & shared memory



- Portable scalable
- C, C++, Fortran
- used as $\boxed{\text{API}}$
- Desktop Application Pgm Interface
  Multiple CPU cores

MPI

- Distributed memory
- Message Passing Interface.
- individual memory assigned to each processor.



- clusters & supercomputers.

PThreads

> Shared memory.

> Single Processor/multiple threads.

> Lightweight Threading system

> used $\boxed{POSIX}$ for creation of threads & their control. Portable OS Interface

CUDA

> NVIDIA

> used on GPU's.

> Enable dramatic ↑ in perf by using GPU.

OpenCL.

> framework for using heterogeneous platforms

GPU+CPU+ FPGA etc.

> C-Based.

> Multicore platform.

> flexible Hardware.

TBB

> Template library written in C++

> Task Parallelism (facilitates)

Data P vs Task P.

> offers uI for ~~[illegible]~~ H parallelism.

~~Open ACC.~~

> ~~Heterogeneous comp~~

> ~~NVIDIA & other giants~~

Cloud computing:

> on demand delivery of IT resources over internet.
> pricing based on usage.
> Buying / owning / maintaining] × computing facilities / Data centers.

Rent ✓
↳ AWS
Microsoft Azure
Google Cloud Platform (GCP)

- * Data centers.
- * Virtualization. (VMware)
- * Resource pooling (CPU, RAM, BW)
- * API & Portals (API - Programme Interface)
- * Billing & metering.

Cloud service Models!

IaaS - Infrastructure	Virtual servers	OS/Apps	AWS
PaaS - Platform	Deployment ENV.	Apps/Data.	Google App Engine, AWS Beanstalk
SaaS - software.	Ready to use s/w.	Just Data & config	Gmail, Salesforce, Microsoft 365

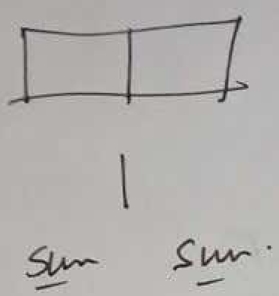
(Empty Apt → furnished Apartment + Hotel Room)

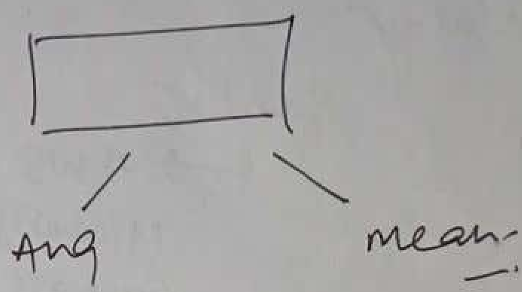
Deployment models	Advantages	challenges
Public Mode/cloud	scalability	Security Risk.
Private cloud	Global Reach	compliance.
Hybrid cloud.	Reduced IT Burden	Vendor Lock-In.
Multi cloud.	Innovation speed	Downtime
	Disaster Recovery.	

* Future Trends

Serverless computing: AWS Lambda → Run code without servers
Edge computing: Processing close to data source.
AI Integration: cloud + ML services. eg image rec & NLP.

* Traditional vs. Parallel computing
* Data vs. Task Parallelism:





Grid computing:

- Distributed computing model.
- Processor / Storage / Resources] many computers geographically dispersed.
- connected & coordinated to work on common task.
- cloud = Renting
  Grid = Pooling for virtual supercomputer.
- Multiple computers (Nodes)
  +
  Networking
  +
  Middleware - eg Globus Toolkit
  +
  Task splitting
  +
  Result Aggregation.

Characteristics

- Heterogenous.
- Loose coupling.
- Geographical Distribution
- Resource sharing
- Coordination

Types of Grid computing:

- computational G: (CPU)
- Data G : (Data)
- Collaborative G : (Virtual Labs)
- utility G : (metered service)

Advantages!

- Massive Processing Power
- Cost Efficiency
- Scalability
- Resource flexibility

Challenges

- Security & Trust
- Resource Management
- Network Dependency
- Software complexity

* Networked Supercomputer!

*

Cluster Computing:

california — * No1 - El Captain 1.742 exa flops (10^{18} quintillion)
* IIT Kgp: PARAM
* IBM: Watson

> method of using interconnected computers (called Nodes) to work together as a unified system for solving computational problems.

eg: Team in a office — CC
Teams in offices across world — GC.

> Better performance
Availability
Scalability

> Feature: Multiple Node.
High speed interconnection.
Single system image
Parallel Processing

> Types
- High Performance : IBM: Blue Gene *Speed, Ilel computing
- High Availability : Banking tx services * reliability & uptime
- Load Balancing : millions of req. balance

Advantages:
> Scalability.
> Cost Effective
> Fault Tolerant
> Flexible.

Disadvantages:
> complex (N/w, cluster s/w)
> Maintainance overhead
> comm. overhead

eg: Google search engine
NASA.
Stock Exchanges. (↑ speed transaction processing)

Cluster C	Grid C
> Tightly connected computers	> Loosely coupled.
> same physical location	> Multiple locations/countries.
> Homogeneous	> Heterogeneous.
> ↑ speed local network/LAN (Infiband, Gigabit Ethernet)	> Internet/WAN.
> ↑ performance, Low latency	> Resource utilization/ flexibility
eg > IBM Blue Gene	> BOINC (Berkley open infra for network computing)

Quantum Computing:

> CS + physics + Math

Quantum mechanics principles are used to process information.

Behaviour of particles at microscopic level.

> Key Difference:

Classical computers use bits 0 & 1

Q computers qubits 0, 1, or both.

> Why QC?? can solve certain problems much faster. eg:- Tossing coin

Factorization

optimization

> Core Quantum Principles:

* Superposition

Qubits can exist in multiple states simultaneously.

* Entanglement

Two qubits can be linked → changing one may effect other.

Enable powerful correlations for computation.

* Quantum Interference

QC can amplify correct paths & cancel wrong ones.

* Amazon → Bracket QC service → Q. Solution Labs. → Emulators.

* Decoherence

Loss of quantum state in a qubit

> Components:

Q H/W

Q S/W

> Where??-

ML

optimization

simulation.

Limitations & challenges:
- Decoherence
- ↑ error rates
- ↓↓ temperatures / cryogenic cooling / ↑ cost
- stable environments
- not yet faster for everyday tasks

eg: * Reading a book analogy / maze solving.

* classical C : General task
  Q C : Specific task. (* Embedded system)

Applications:

→ Cryptography
→ Drug Discovery
→ Financial modelling
→ AI (QNN)
→ Logistics & optimization.

Multicore CPU!

- It is a single physical chip.
- contains two or more independent processing units called cores.
- Each core* can execute instructions independently
  - * can handle its own set of tasks.
  - * share certain resources (memory, cache & bus connections)

*mini CPU's inside one CPU chip.

- 2005: Multicore Era
  - Dual ⟶ 96+ cores in servers.

- OS schedule programms → processes ⟶ Each process may have multiple threads → Threads cores. → H/W & S/W support → ||alel processing

How it works? → processes

- components!
  - Cores
  - Cache → L1 L2 L3
  - Memory controller
  - Interconnect → B/W cores.

- Architectures
  - Homogeneous
  - Heterogeneous

> Advantages:

* Parallelism
* Performance scaling
* Multitasking (* Multiple programs)
* Better Energy Efficiency.

> Limitations:

> Not all s/ws
> X2 cores X2 speed X
> Heat consumption.
> complex programming

Examples:

Laptop: Intel core i5, i7, AMD Ryzen

Server: AMD - EPYC (-96 cores)
Intel Xeon (60+ cores)

Phones: Apple A series (6 cores)
Qualcom Snapdragon (8 cores)

Note! Multicore! More physical cores.
HyperThreading: Each core can handle multiple threads.

eg: restaurant Kitchen

1 core = 1 chef 1 dish at a time
4 core = 4 chef 4 dish simultaneously
HT = Each chef can handle two task by switching quickly.

Multithreading

* an independent pgm

> Technique where a single (process) is divided in multiple small units called threads. & these threads can run concurrently.

a sub-process.

Multicore CPU! Hardware Feature
Multithreading! Software Technique

GPU!

- Graphic Processing unit
- It is a processor designed to handle parallel computation tasks.
- was made for graphics in games & multimedia.
- now used for general purpose computing
- AI, Data Science, HPC.

	GPU ↓	CPU ↓
Design Goal :	Parallel Task Repititive computation	General purpose sequential.
Cores :	Hundreds-to thousands simple	2-4-64 Powerful.
Parallelism :	Very high	Low → Moderate
Best for :	Logic heavy Branching tasks.	Data heavy Repititive tasks

Architecture: (Inside GPU)

- CUDA cores
- VRAM – video.
- Memory controller.
- SIMD Execution
- shader units (pixels! shading etc)
- compute units – Group of cores for GP.

Types: Integrated (with CPU)
Dedicated.
External.

Note:
A task is broken into thousands of small, identical opⁿˢ.
↓
Each GPU core executes one operation in parallel with others.

eg: 4K image: Each pixel's colour calculation handled by independent core.

Analogy: CPU: Skilled masterchef complex recipes one at a time.
GPU: hundreds of kitchen assistants each chopping stirring plating

GPGPU:

> it use CUDA (NVIDIA)
Open CL etc.

Applications
> - AI ML
- Scientific calc.
- video transcoding
- cryptocurrency mining (Blockchain)

Advantages
> - Massive Parallelism
- High throughput
- offloads work from CPU

Limitations
> Not suitable for seq. logic
> consumes more power
> needs special programming models.
> VRAM Requirements.

Examples:
* Apple M series Integ. GPU
* NVIDIA Geforce RTX series (Gaming)
* Tesla (Data Centres)
* Intel: Arc discrete.
* AMD: Radeon RX (Games)
Instinct MI (DC)