INTRODUCTION TO PARALLEL COMPUTING

Deepika H.V

System Software Development Group

C-DAC, Bengaluru

deepikahv@cdac.in

Contents

- What is Parallel Computing
- Parallel Architectures
- Memory Architectures
- Evolution of Parallel Architectures
  - Top 5 Super Computers
- Parallel Programming
  - Program Design
  - Parallel Programming Paradigms

What is Parallel Computing?

Serial Computing

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

How long will it take if 15 Men 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

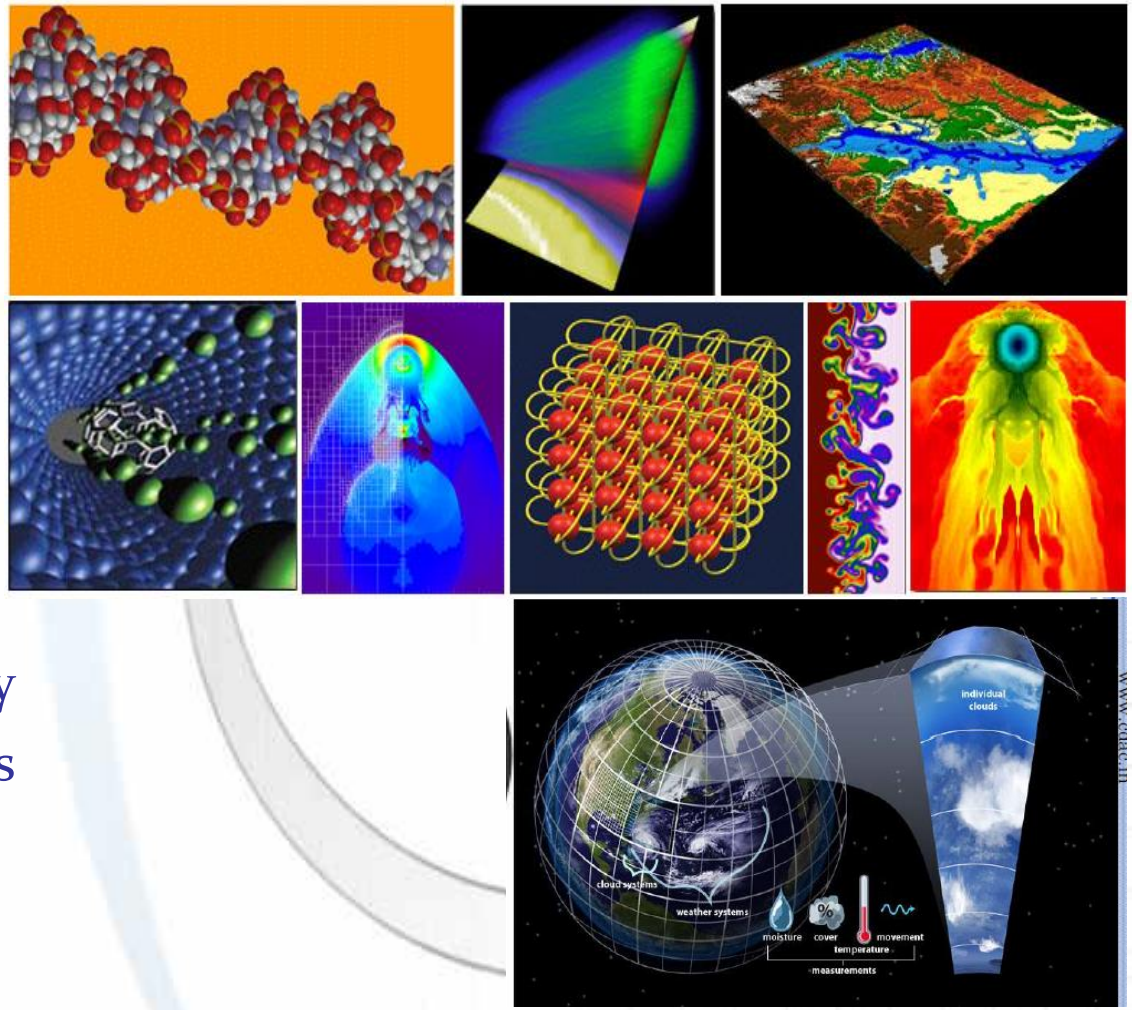
- Software methodology used to implement parallel processing.

Parallel Computing

- Term that encompasses all the technologies used in running multiple tasks simultaneously on multiple processors

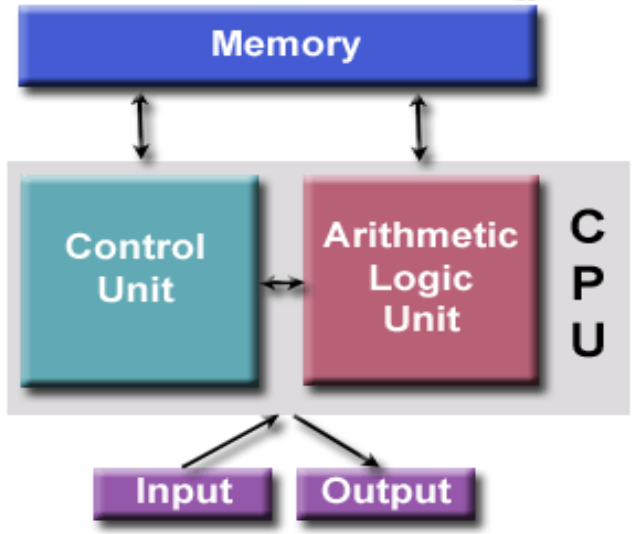
Use Cases of Parallel Computing

- Prediction of weather, climate, global changes
- Challenges in materials sciences
- Semiconductor design
- Structural biology
- Design of drugs
- Human genome
- Astronomy
- Challenges in transportation
- Vehicle dynamics
- Nuclear fusion
- Enhanced oil and gas recovery
- Computational ocean sciences
- Speech
- Vision
- Visualization and graphics



Parallel Architectures

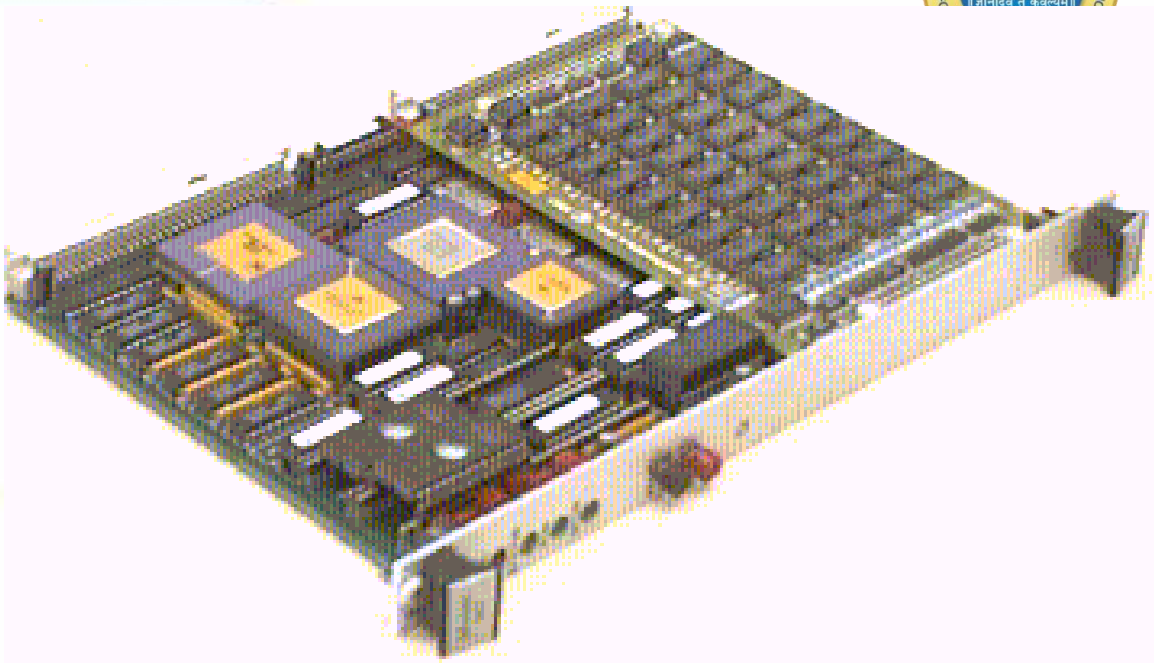
Simple Von Neumann Machine



Parallel Architectures

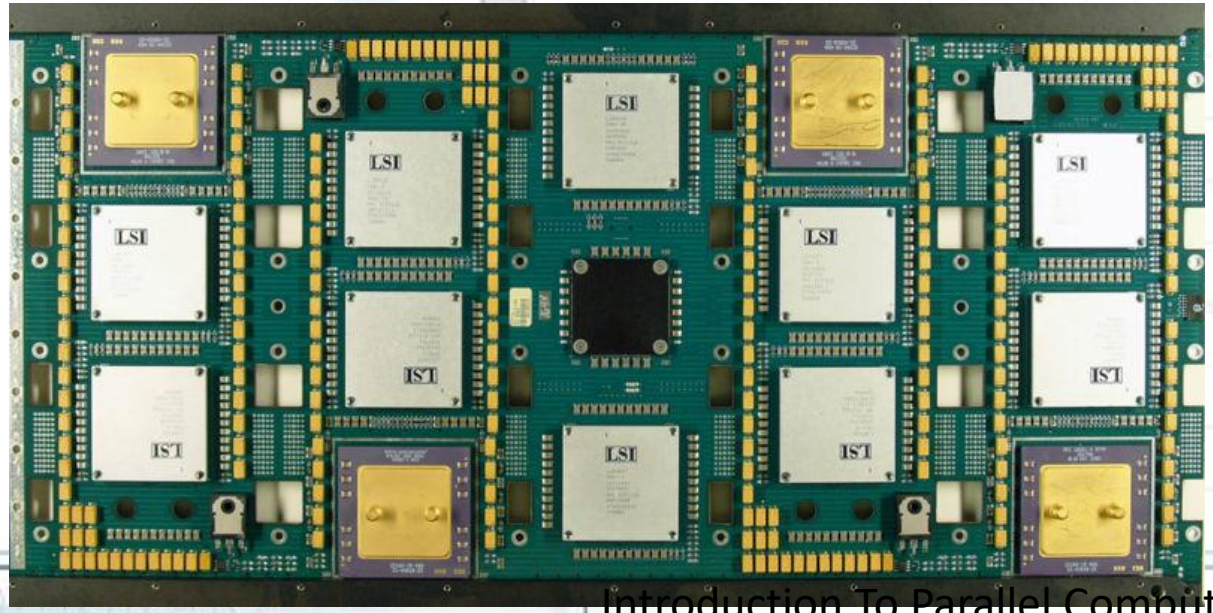
Vector processors

- able to run mathematical operations on multiple data elements simultaneously



Superscalar processors

- Instruction level parallelism with a processor



Multicore Machines

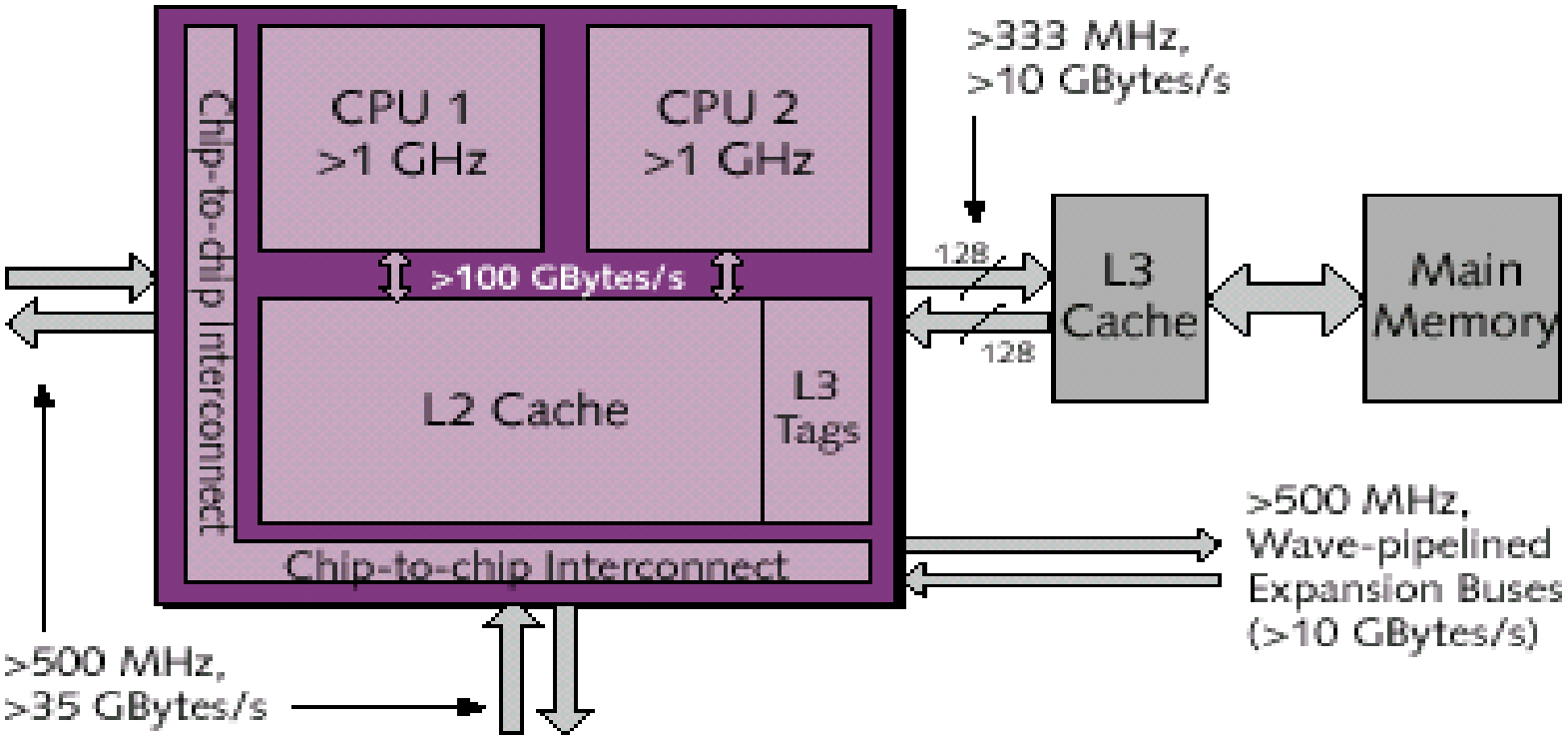
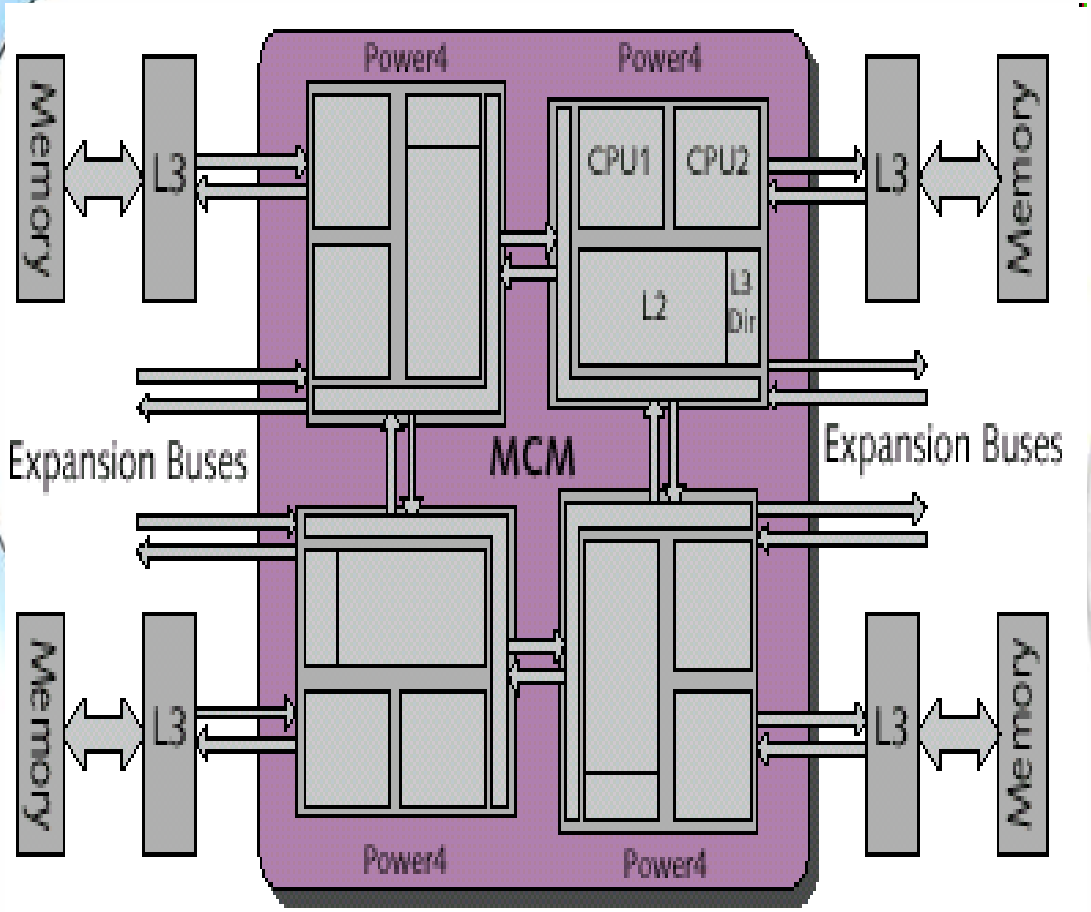
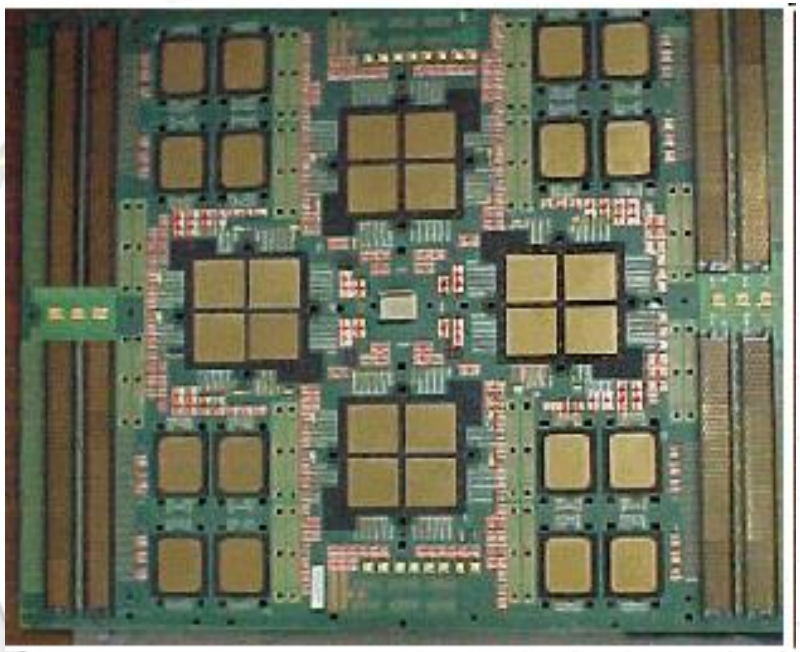


Fig: A 2-Processor Chip

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

Multi Chip Module (MCM)

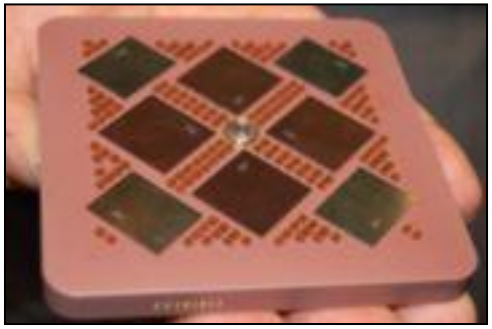


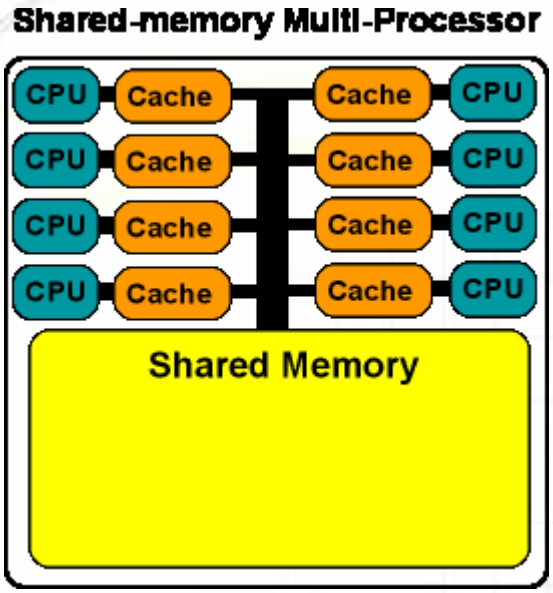


32-way System with 4 MCM and L3 cache

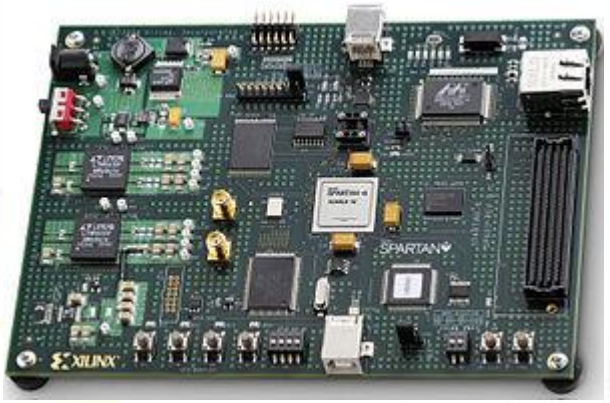
Symmetric Multiprocessor (SMP)

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





Accelerators



Field-Programmable Gate Arrays

- a computer chip that can rewire itself for a given task
- can be programmed with hardware description languages such as VHDL or Verilog

General Purpose GPU

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

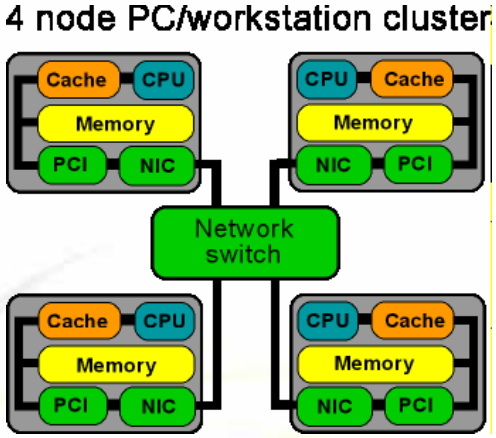
Supercomputers

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

Parallel Architectures

❑ Distributed Computing

- ➢ different parts of a program are run simultaneously on separate computers communicating over a network

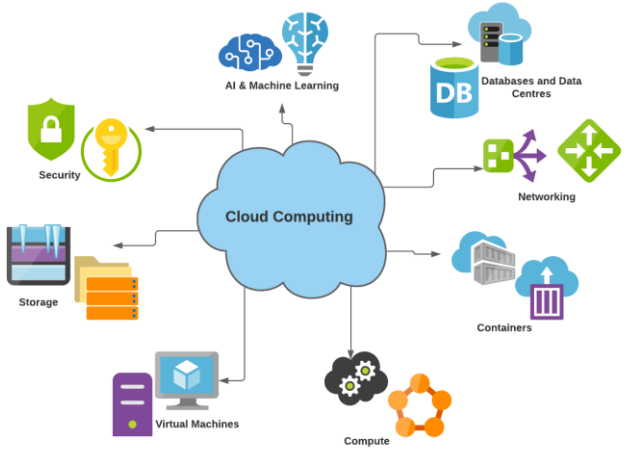


❑ Grid Computing

- ➢ Aggregation, sharing of distributed heterogeneous systems

❑ Cloud Computing

- ➢ on-demand available service – IaaS PaaS, SaaS
- ➢ pay-as-you-use over the Internet

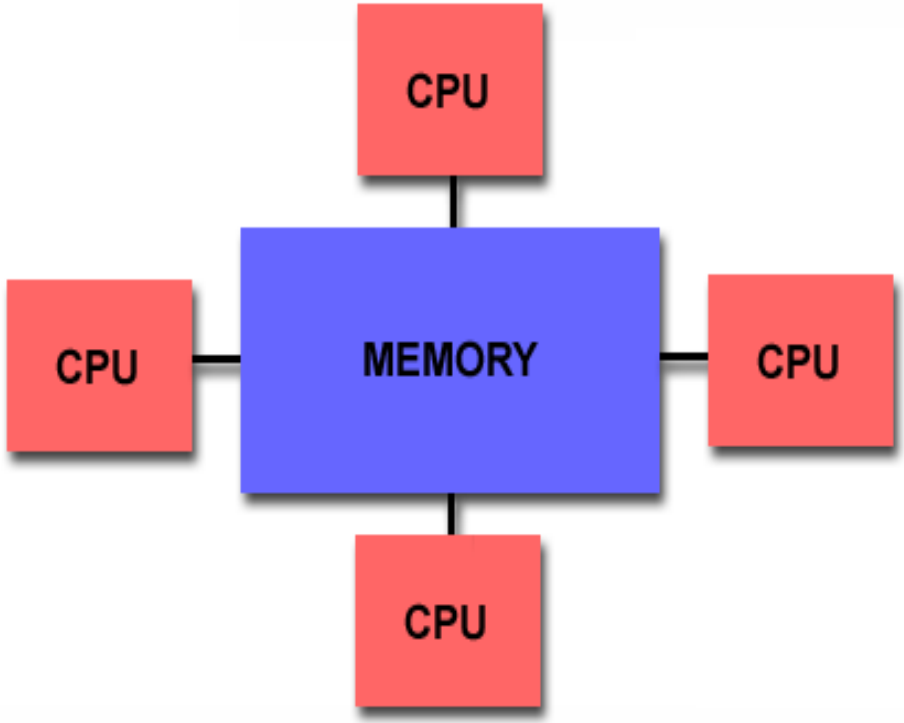


Memory Architecture

Shared Memory Architecture

❑Uniform Memory Access (UMA):

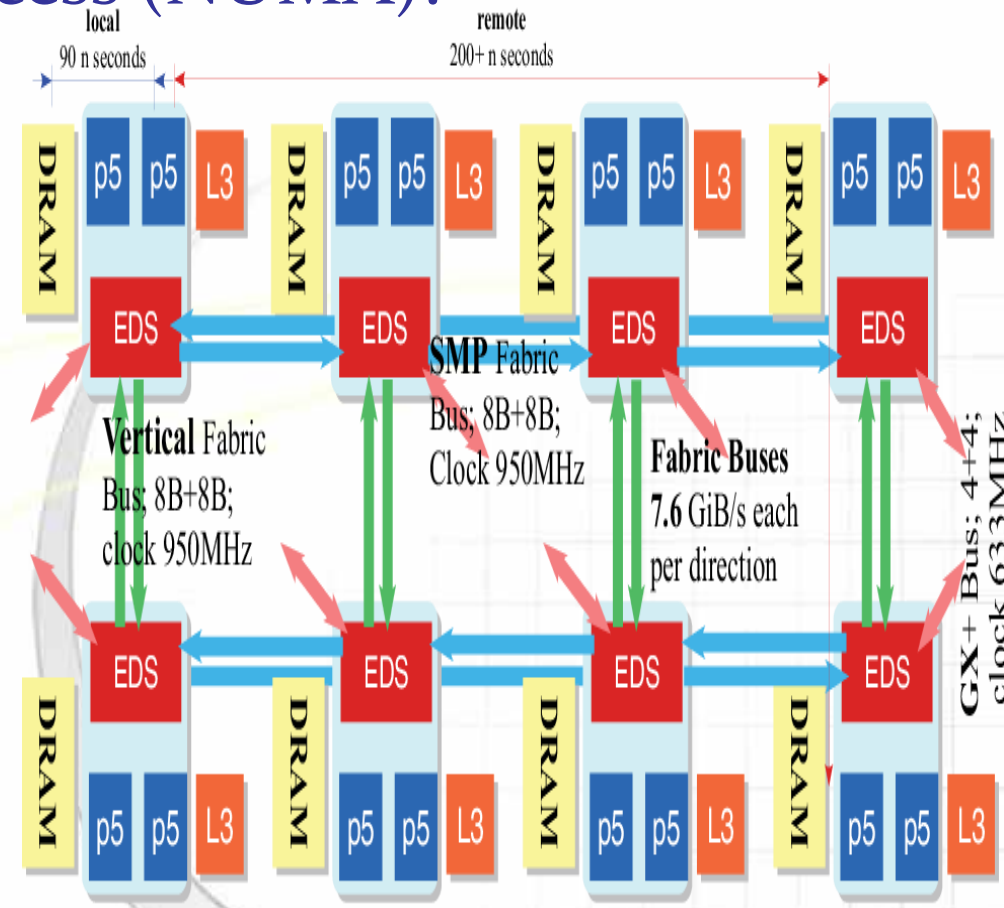
- ➢Example - SMP
- ➢Identical processors
- ➢Equal access and access times to memory
- ➢Sometimes called Cache Coherent UMA (CC-UMA). Cache coherency is accomplished at the hardware level.
- ➢Contention - as more CPUs are added, competition for access to the bus leads to a decline in performance.
- ➢Thus, scalability ~ 32 processors.



Shared Memory Architecture

❑Non-Uniform Memory Access (NUMA):

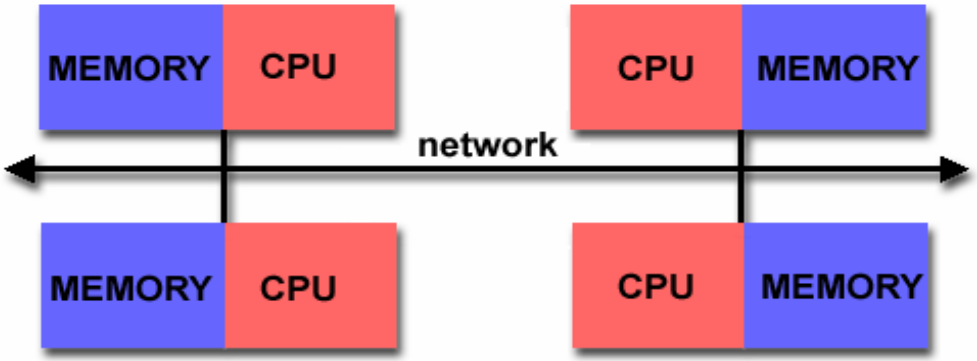
- Not all processors have equal access time to all memories
- Memory access across link is slower
- If cache coherency is maintained, then called **CC-NUMA**
- Designed to overcome scalability limits of SMPs.
- Can support up to **1024 processors**.



- 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

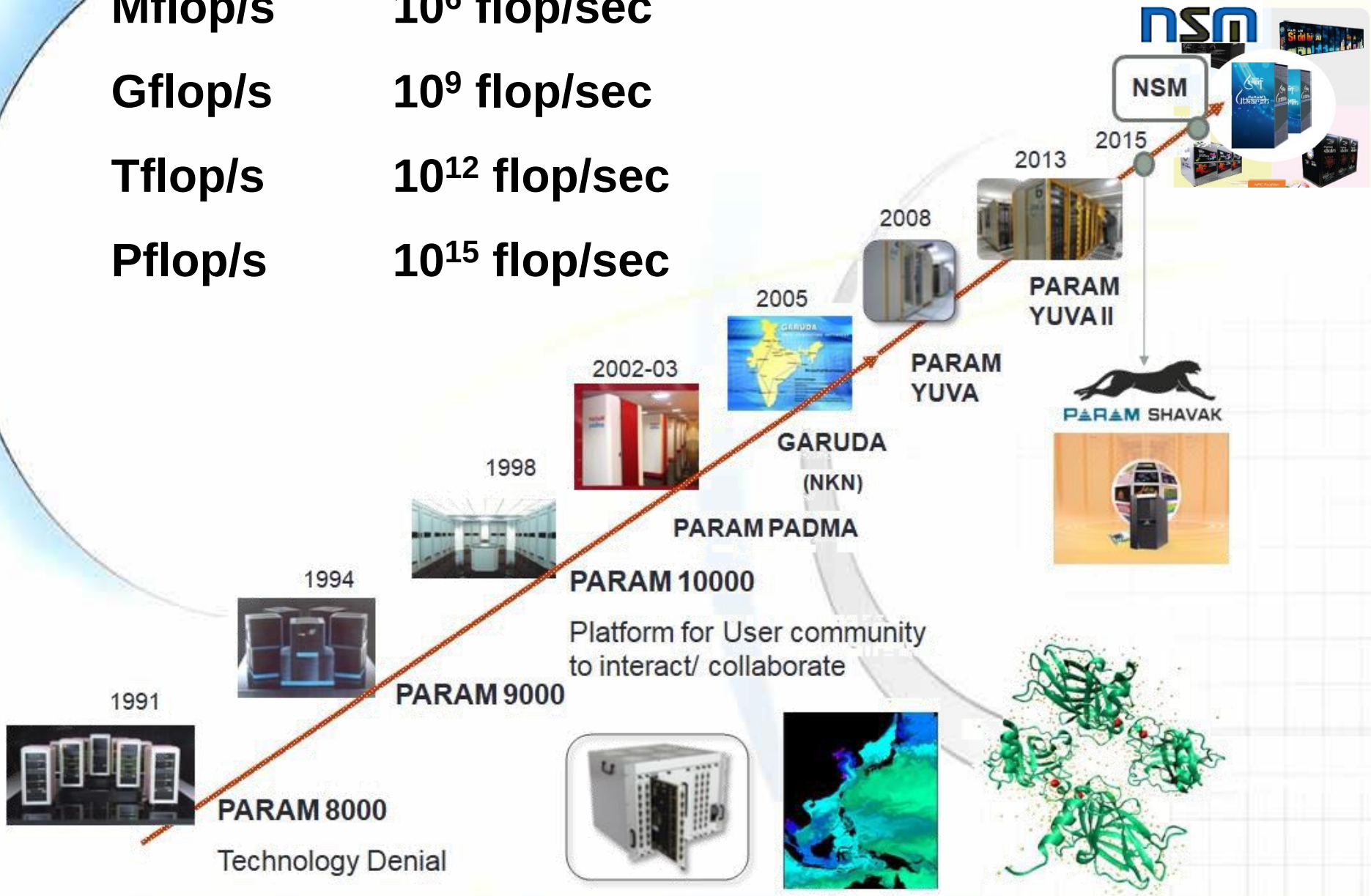
Supercomputer

- 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

Listing of TOP500 from 1993 is available on top500.org

Evolution 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



World Top5 Supercomputers

Rank	Site	System	Cores	Rmax (PFlop)	RPeak (PFlop)
1	Frontier , DOE/SC/Oak Ridge National Laboratory United States	HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE	8,730,112	1,102.00	1,685.65
2	A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu	A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu	7,630,848	442.01	537.21
3	LUMI, EuroHPC/CSC Finland	HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE	2,220,288	309.10	428.70
4.	Leonardo, EuroHPC/CINECA Italy	BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos	1,463,616	174.70	255.75
5.	Summit, DOE/SC/Oak Ridge National Laboratory United States	IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM	2,414,592	148.60	200.79

Indian Top5 Supercomputers

Rank	Site	System	Cores/ nodes	Rmax (TFlop)	RPeak (TFlop)
1 [120]	PARAM Siddhi-AI, C-DAC,Pune	NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband (OEM:ATOS under NSM initiative, Bidder)	41664/ 236	4619	5267.14
2 [143]	Pratyush, Indian Institute of Tropical Meteorology, Pune	Cray XC-40 class system with 3315 CPU-only (Intel Xeon Broadwell E5-2695 v4 CPU) nodes with Cray Linux environment as OS, and connected by Cray Aries interconnect	119232/ - -	3763.9	4006.19
3 [277]	Mihir, NCMRWF, Noida	Intel Xeon Broadwell E5-2695 v4 CPU with Cray Linux environment connected by Cray Aries interconnect OEM:Cray, Bidder:Cray	83592/ 1152	2570.4	2808.7
4.	PARAM Pravega, IISc, Bangalore	Intel Xeon Cascade Lake processors,NVIDIA Tesla V100 with NVLink, Mellanox HDR interconnect. OEM:Atos, Bidder:Atos	29952/ 624	1702	2565
5.	PARAM Shakti, Indian Institute of Technology ,Kharagpur	Intel Xeon Skylake, NVIDIA Tesla V100. OEM:Atos, Bidder:Atos	17280/ 432	935	1290.2

Parallel Programming

Flynn's Taxonomy

Data Streams

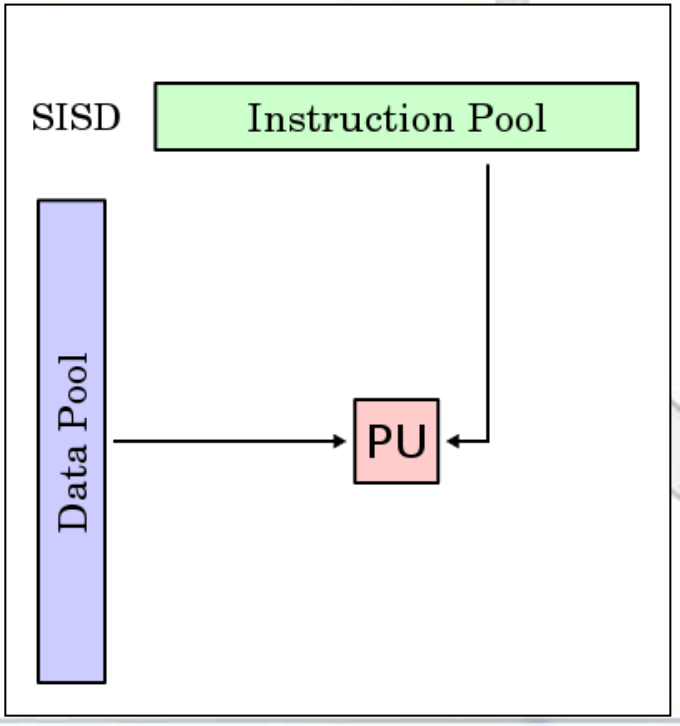
Instruction Streams

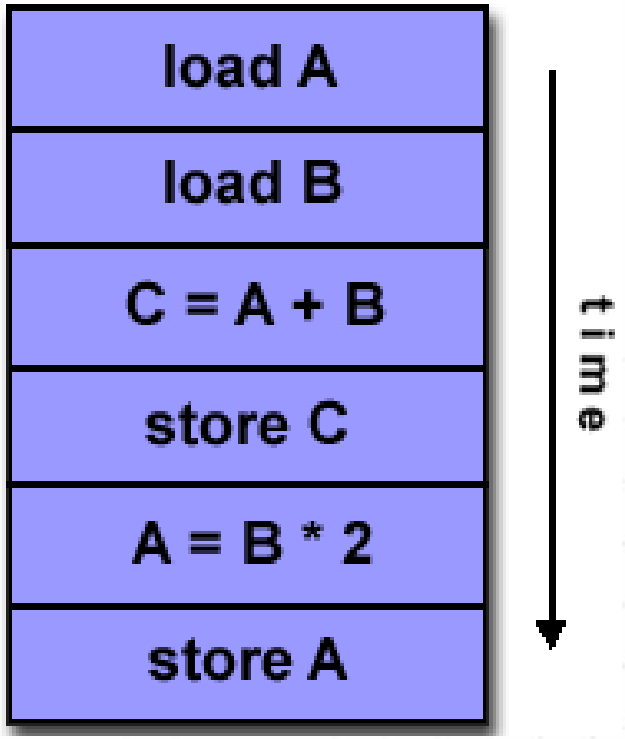
S I S D Single Instruction, Single Data	S I M D Single Instruction, Multiple Data
M I S D Multiple Instruction, Single Data	M I M D Multiple Instruction, Multiple Data

1. SISD

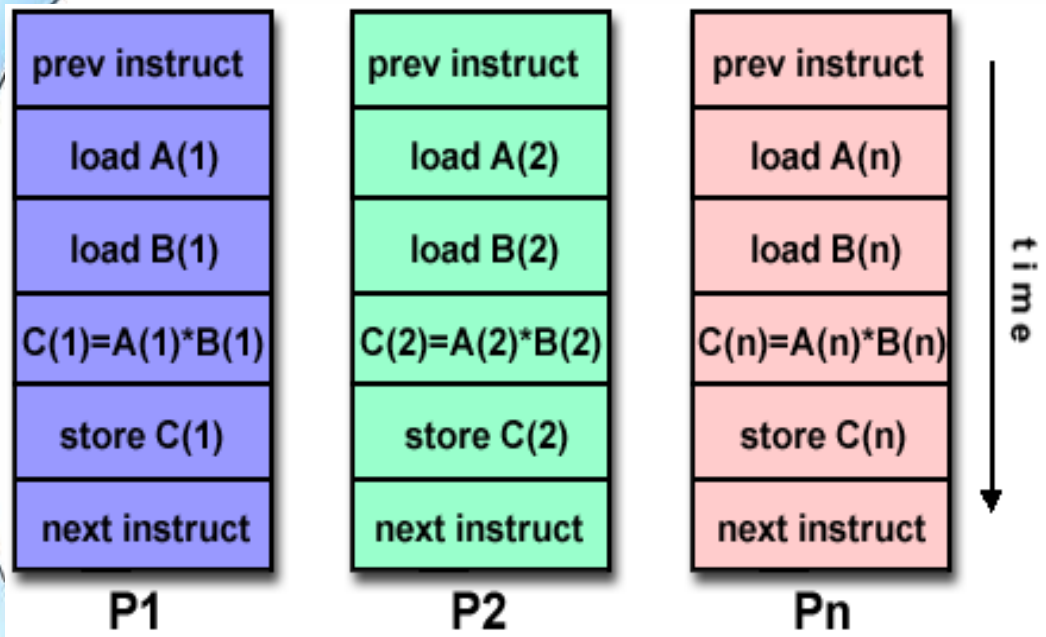

UNIVAC

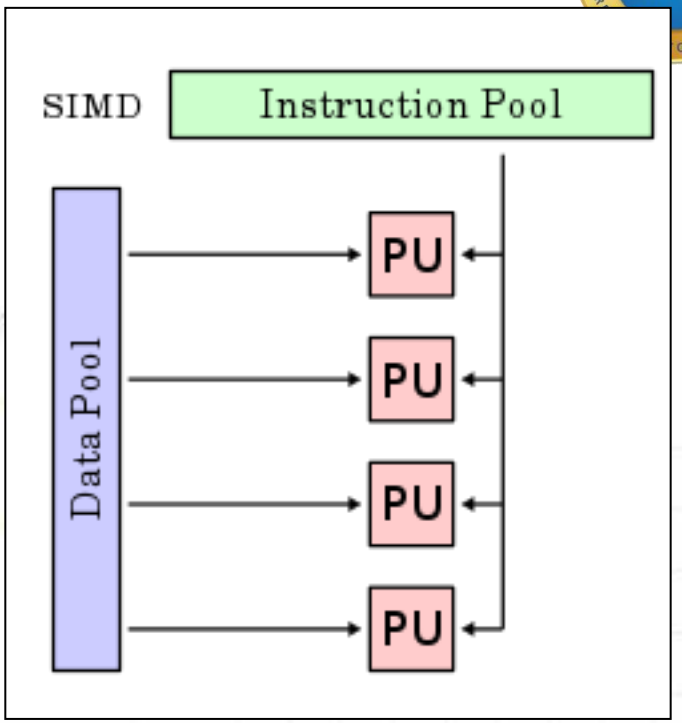

IBM 360


SISD
Instruction Pool
Data Pool
PU


load A
load B
C = A + B
store C
A = B * 2
store A
time

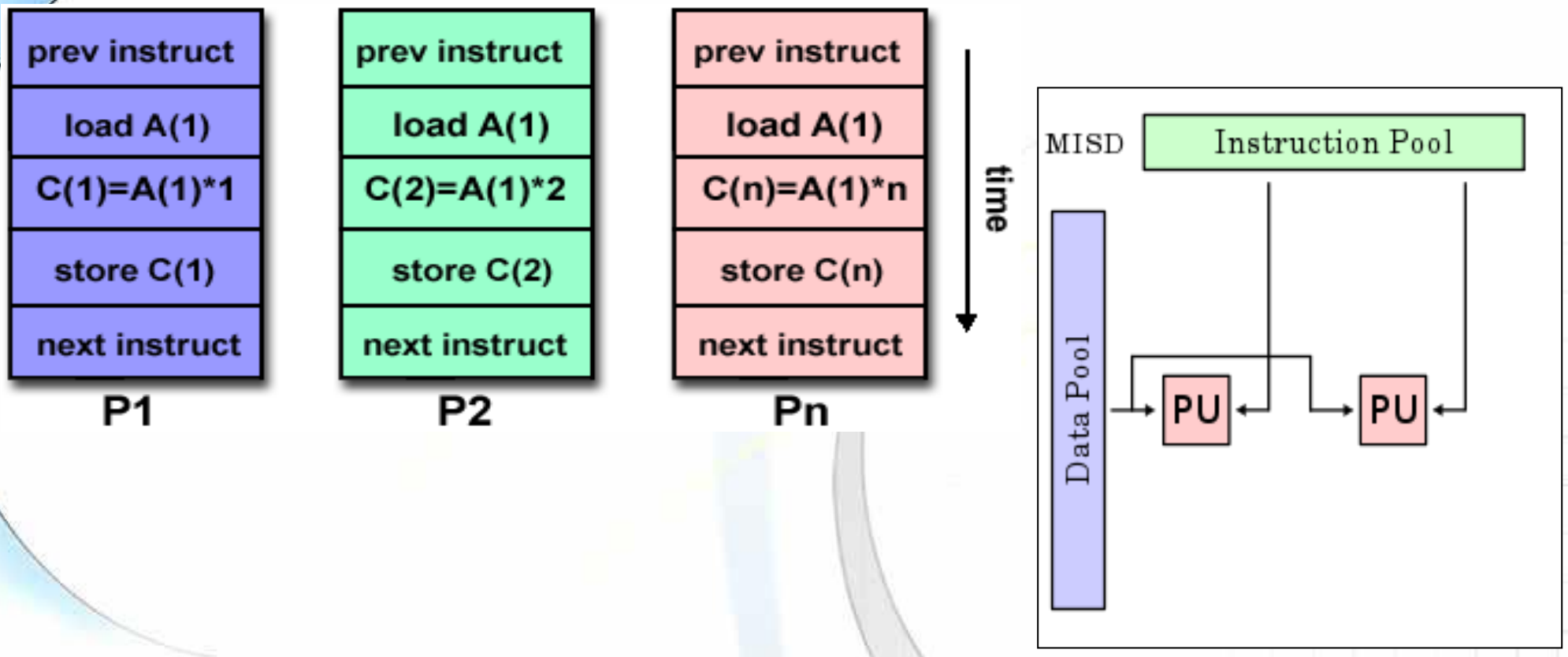
2. SIMD





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.

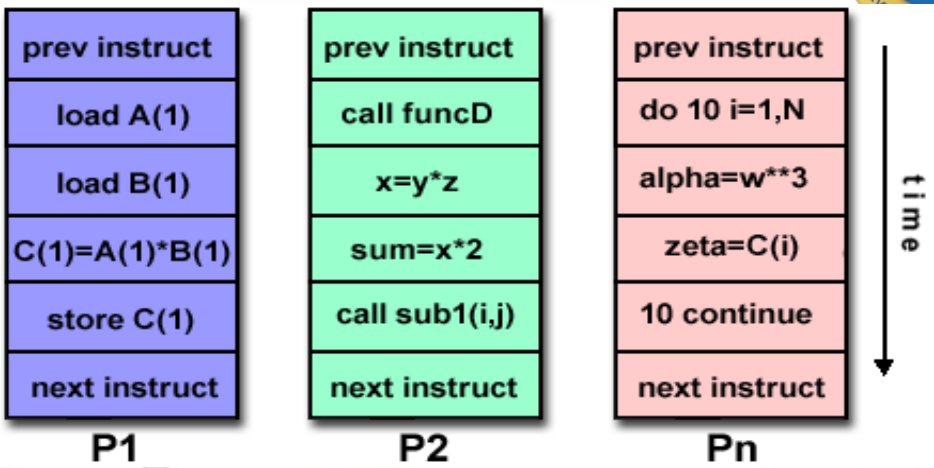
3. MISD

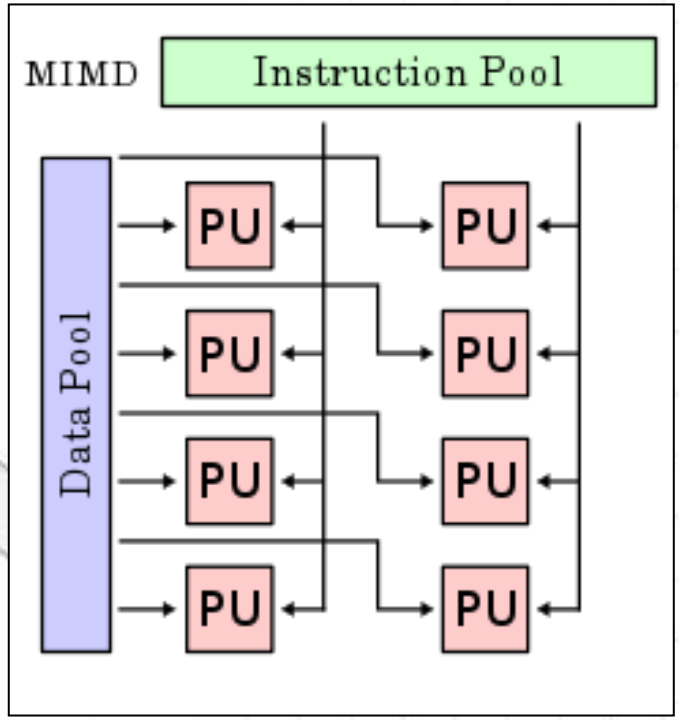


Some conceivable uses might be:

- multiple frequency filters operating on a single signal stream
- multiple cryptography algorithms attempting to crack a single coded message.

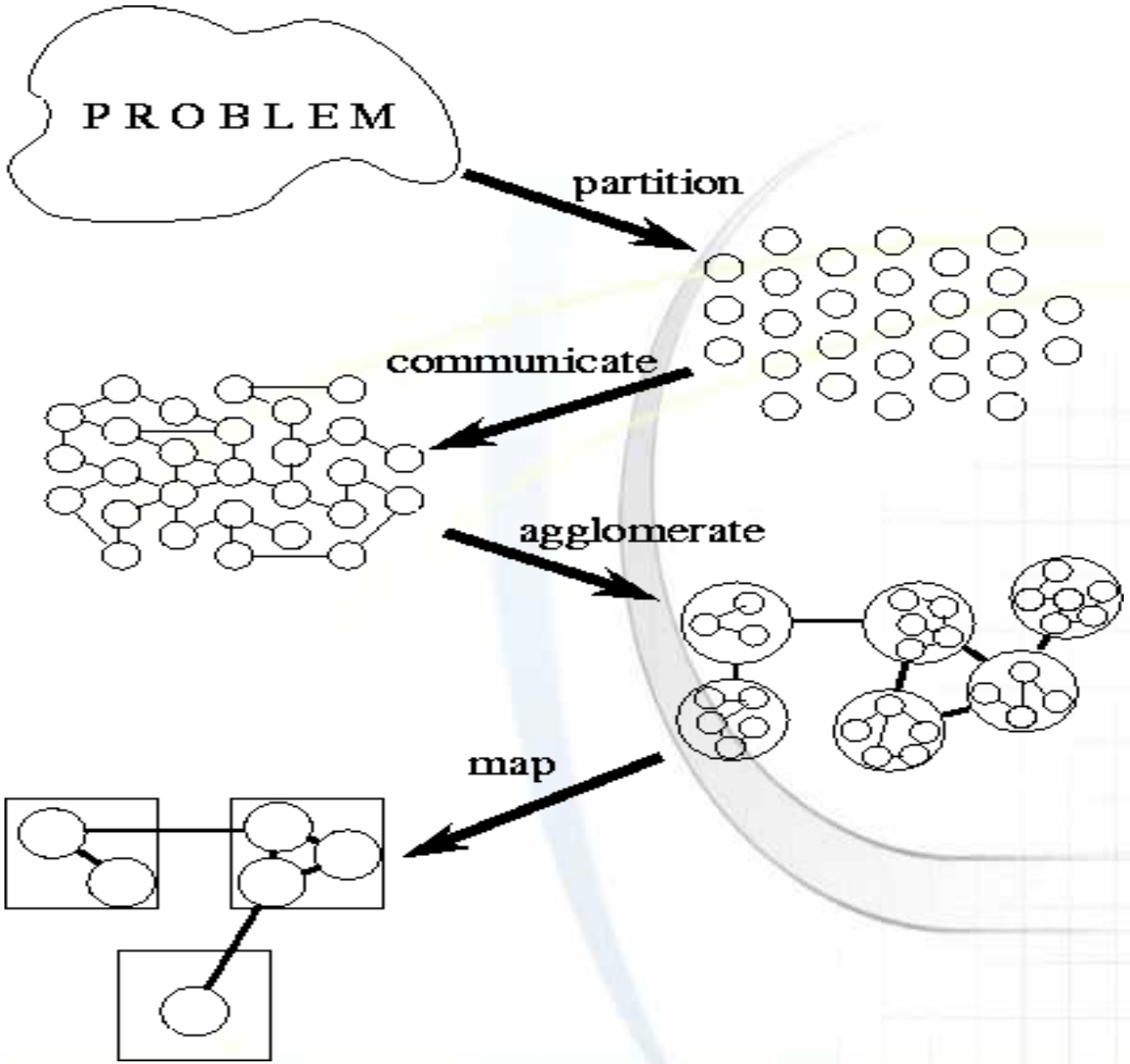
4. MIMD





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

Program Design



Partitioning:

- divide the computation to be performed and the data operated on by the computation into small tasks.

  The focus here should be on identifying tasks that can be executed in parallel.

Communication:

- determine what communication needs to be carried out among the tasks identified in the previous step.

Agglomeration or aggregation:

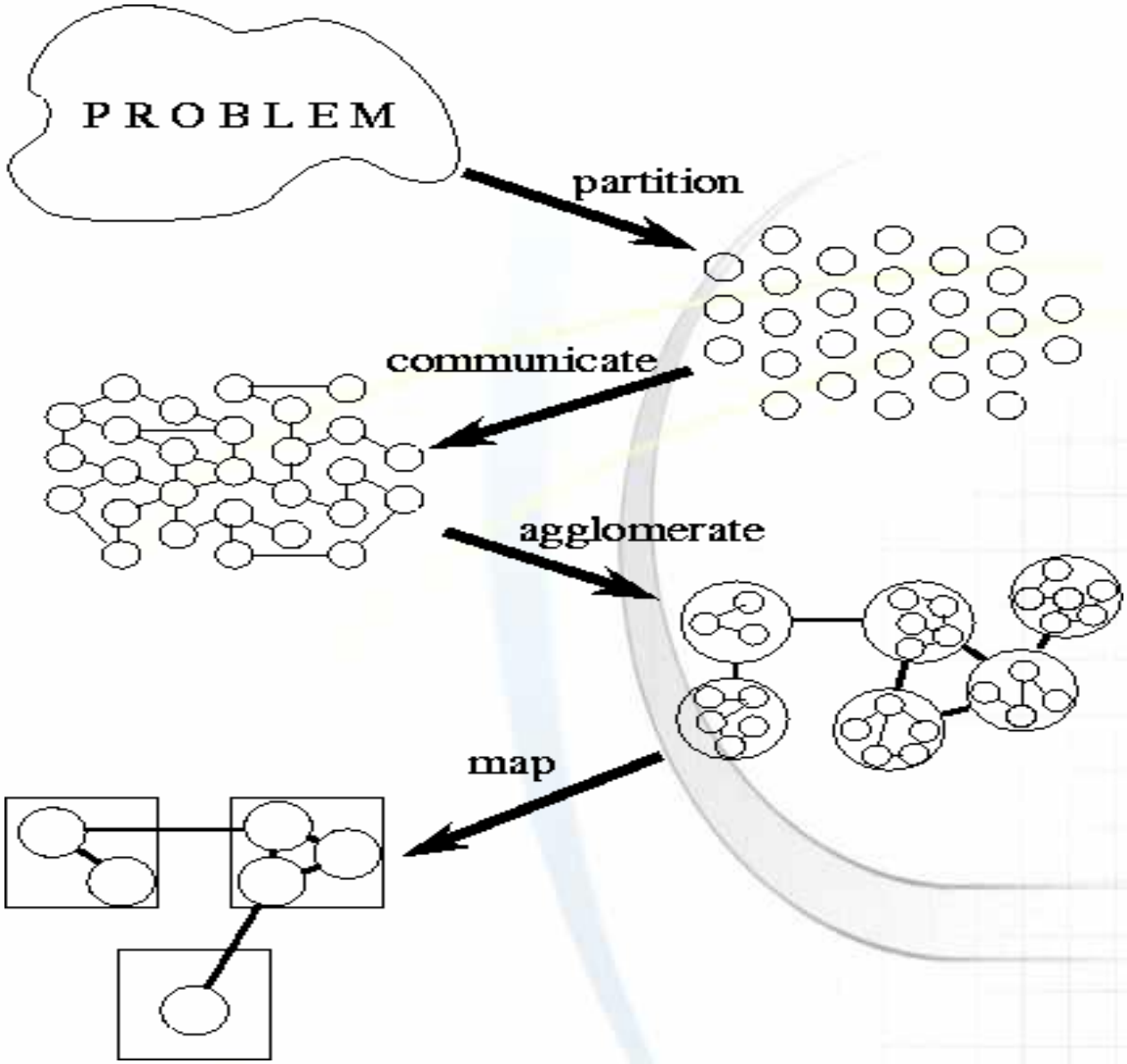
- combine tasks and communications identified in the first step into larger tasks.

  For example, if task A must be executed before task B can be executed, it may make sense to aggregate them into a single composite task.

Mapping:

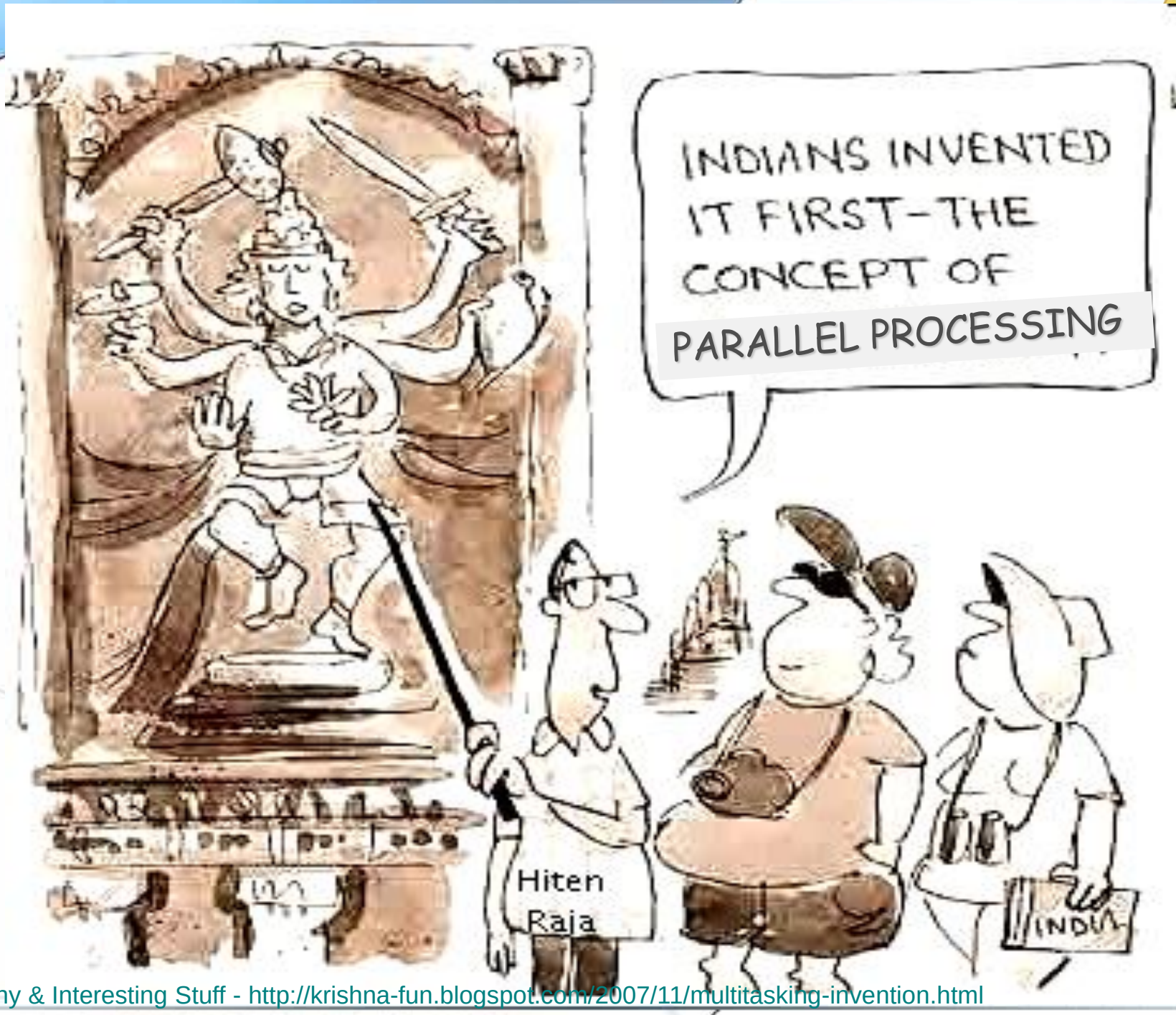
- assign the composite tasks identified in the previous step to processes/threads.

  This should be done so that communication is minimized, and each process/thread gets roughly the same amount of work.


PROBLEM
partition
communicate
agglomerate
map

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**
  - **HIP /SYCL / OneAPI**



Courtesy: Funny & Interesting Stuff - http://krishna-fun.blogspot.com/2007/11/multitasking-invention.html

Questions ???

Applying Advanced Computing for Human Advancement

Thank you!