Introduction to Parallel Computing

Short Course on HPC

15th February 2019

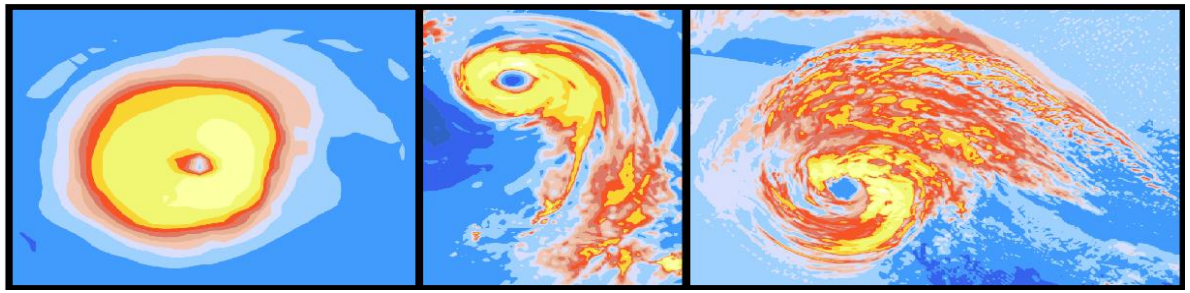
Aditya Krishna Swamy

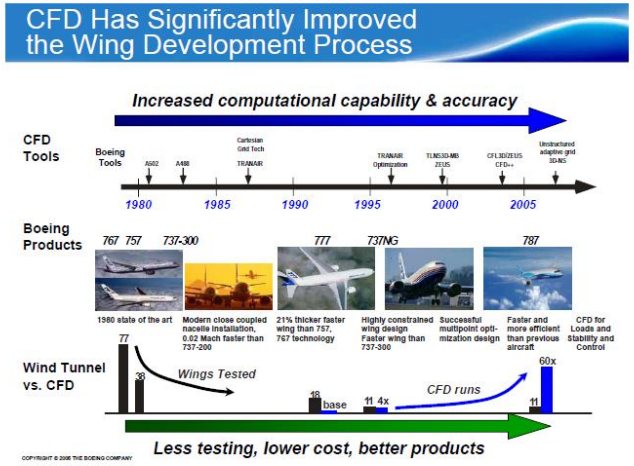
adityaks@iisc.ac.in

SERC, Indian Institute of Science

What is scientific computing?

- "Computational Science" - 3rd paradigm after Theoretical & Experimental Science
- “Scientific Computing” - encompass computations for science and engineering research and applications
- “computing” <- numerical computation aspects PLUS associated systems of networks, visualization, and data storage and retrieval
- Technological advance <-> limitations of analytic tractability, expensive experimentation





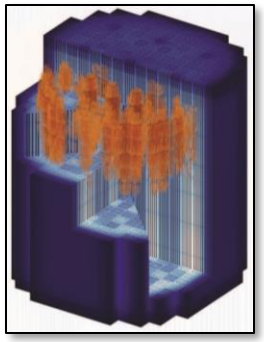
HPC for Scientific Computing

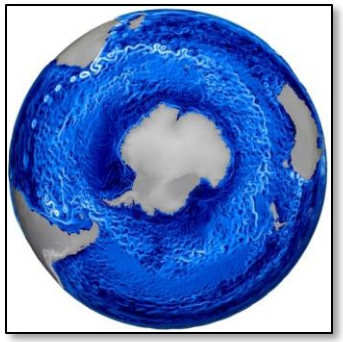
- **HPC is** Computing at the bleeding edge of performance
- **In scientific computing there is always demand for more speed, capacity**
- Computational scientists have an insatiable need for more powerful computers (faster, able to handle more data)
- Resolution (finer grid, more atoms)
- Dimensionality (1-D, 2-D, 3-D, etc.)
- Complexity (multi-physics, multiple time and spatial scales)
- Fidelity (equations, geometry, realistic conditions and dimensions)
- Time to solution (ex. ~1ns/day)
- Algorithmic complexity, e.g., O(N^7)

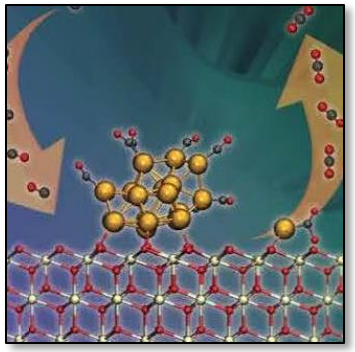
Exascale Applications Will Address National Challenges

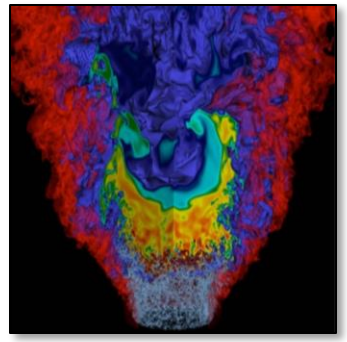
Summary of current DOE Science & Energy application development projects

Nuclear Energy (NE)	Climate (BER)	Chemical Science (BES, BER)	Wind Energy (EERE)	Combustion (BES)
Accelerate design and commercialization of next-generation small modular reactors	Accurate regional impact assessment of climate change	Biofuel catalysts design; stress-resistant crops	Increase efficiency and reduce cost of turbine wind plants sited in complex terrains	Design high-efficiency, low-emission combustion engines and gas turbines
✓Climate Action Plan ✓SMR licensing support ✓GAIN	✓Climate Action Plan	✓Climate Action Plan ✓MGI	✓Climate Action Plan	✓2020 greenhouse gas and 2030 carbon emission goals









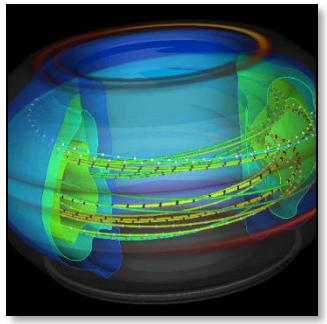
Exascale Applications Will Address National Challenges

Summary of current DOE Science & Energy application development projects

Magnetic Fusion Energy (FES)

Predict and guide stable ITER operational performance with an integrated whole device model

- ✓ITER
- ✓Fusion experiments: NSTX, DIII-D, Alcator C-Mod



Advanced Manufacturing (EERE)

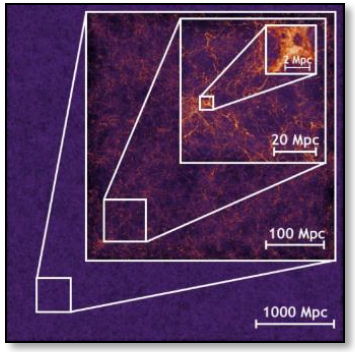
Additive manufacturing process design for qualifiable metal components

- ✓NNMIs
- ✓Clean Energy Manufacturing Initiative

Cosmology (HEP)

Cosmological probe of standard model of particle physics: Inflation, dark matter, dark energy

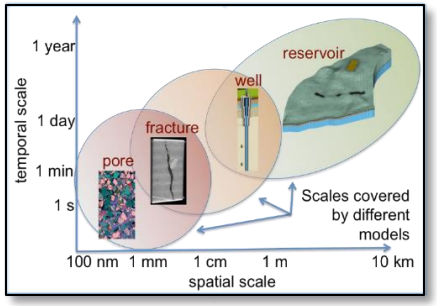
- ✓Particle Physics Project Prioritization Panel (P5)



Geoscience (BES, BER, EERE, FE, NE)

Safe and efficient use of subsurface for carbon capture and storage, petroleum extraction, geothermal energy, nuclear waste

- ✓EERE Forge
- ✓FE NRAP
- ✓Energy-Water Nexus
- ✓SubTER Crosscut



Precision Medicine for Cancer (NIH)

Accelerate and translate cancer research in RAS pathways, drug responses, treatment strategies

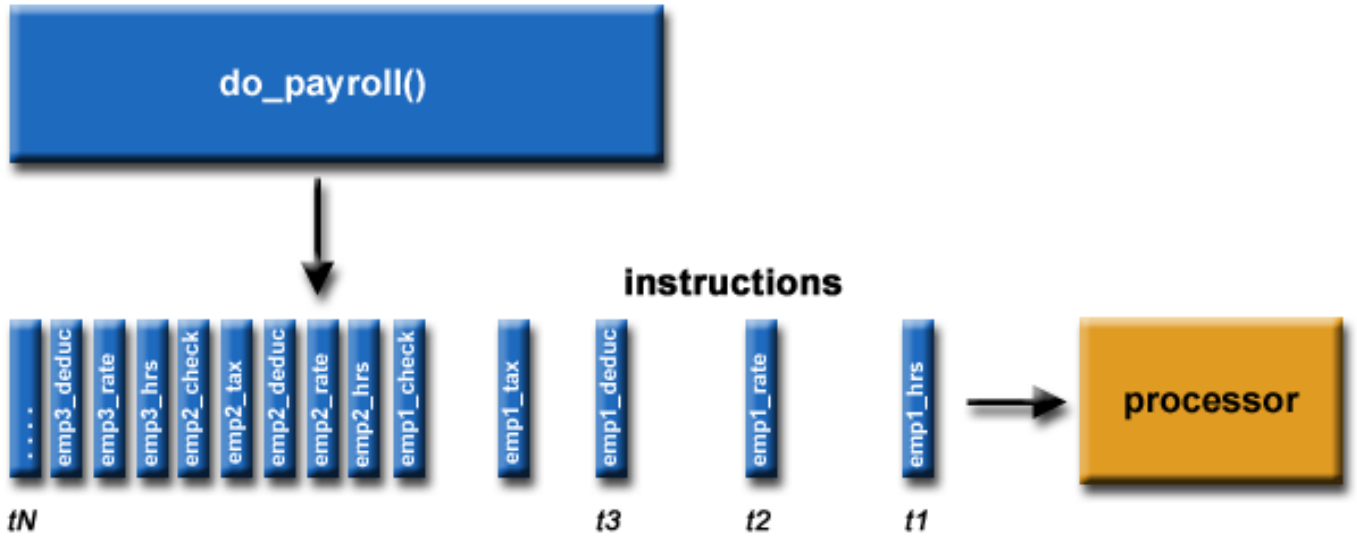
- ✓Precision Medicine in Oncology
- ✓Cancer Moonshot

Scientific computing requires broad expertise *in addition to the research domain*

- Knowledge of parallel computer architectures, mathematical models and numerical algorithms
- Proficiency in programming methodologies and languages
- Software architecture, debugging and performance measurement tools, visualization
- Software engineering for working in teams
- Methods for building "community codes"
- Methodologies and tools relevant for data-intensive applications
- Frameworks for scientific workflows

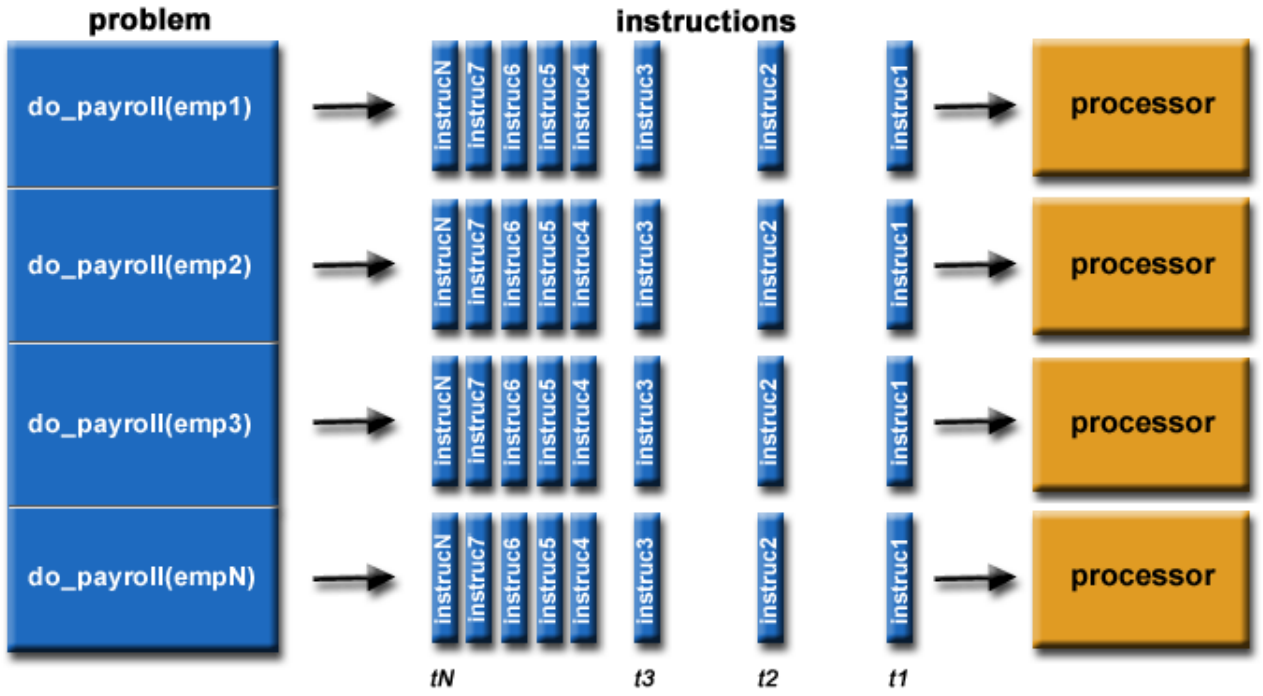
Serial Computing vs Parallel Computing

- Traditionally, software has been written for serial computation
- A problem is broken into a discrete series of instructions
- Instructions are executed sequentially one after another
- Executed on a single processor
- Only one instruction may execute at any moment in time

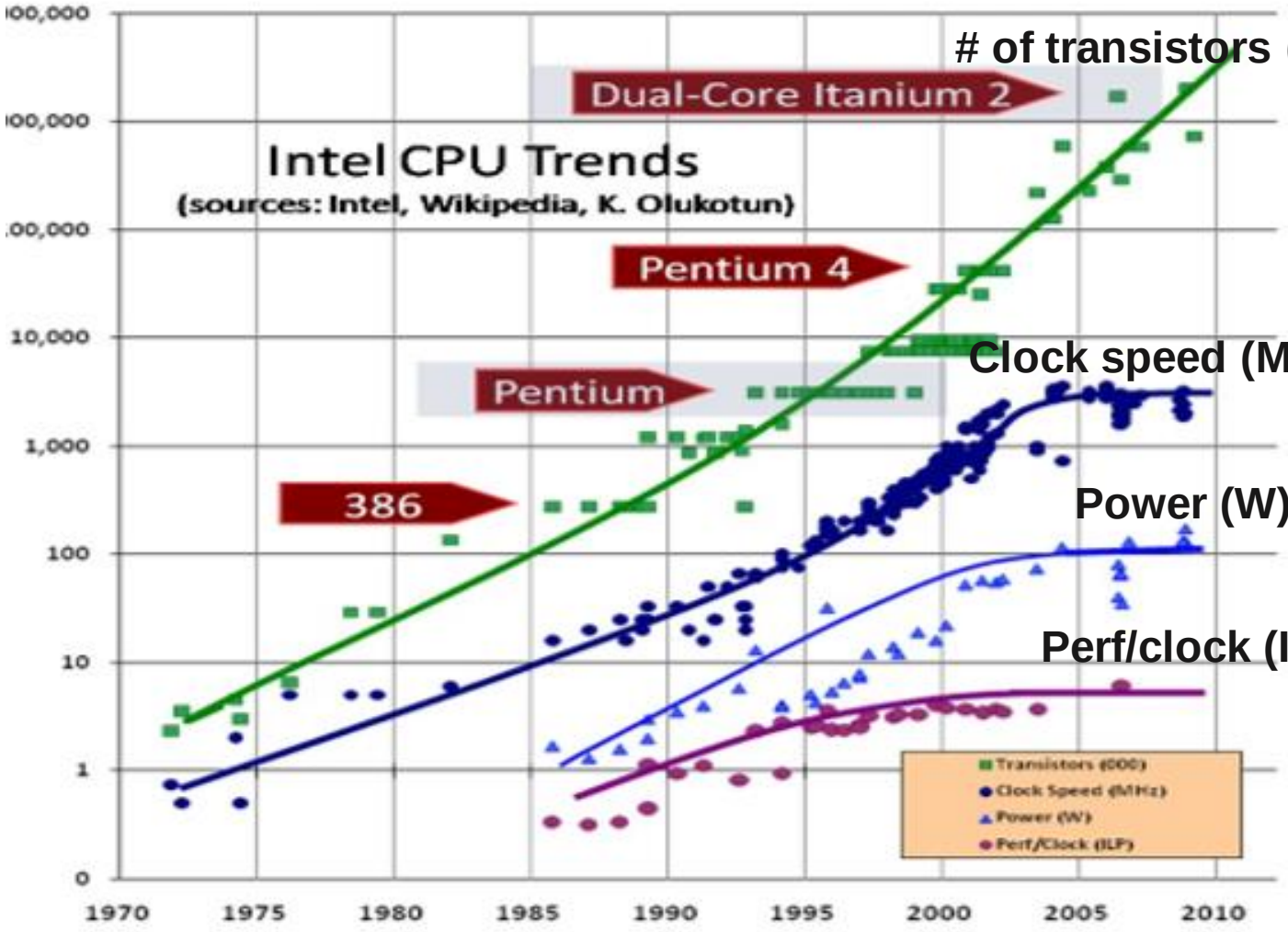


Serial Computing vs Parallel Computing

- Simultaneous use of multiple compute resources to solve a computational problem.
- Run on multiple CPUs
- Problem is decomposed into multiple parts that can be solved concurrently.
- Each part is decomposed into a set of instructions.
- Instructions are executed simultaneously on different CPUs



Why parallelism?

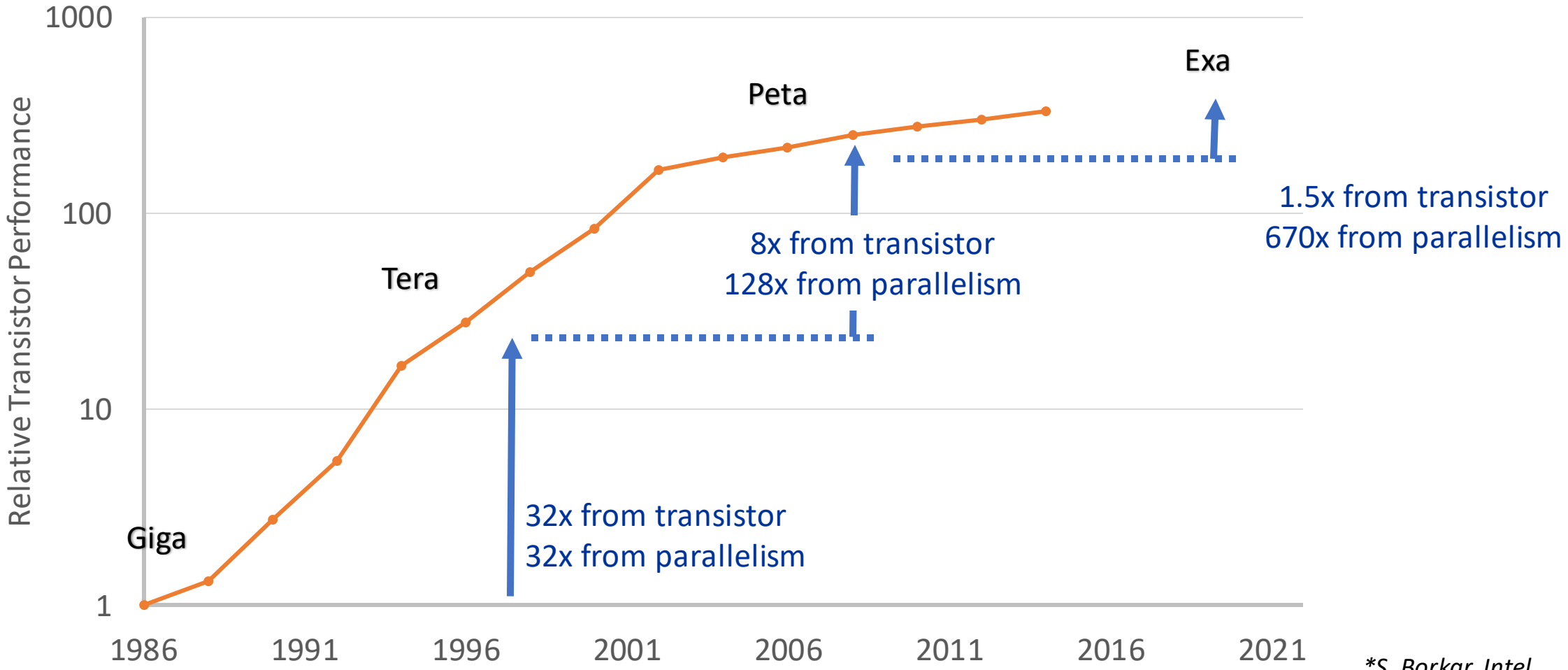


Moore's Law still holds
Number of transistors continues to increase

Dennard scaling does not:
Performance/Clock has flattened out because
at 9nm transistor gates became too thin to prevent current from leaking out into the substrate

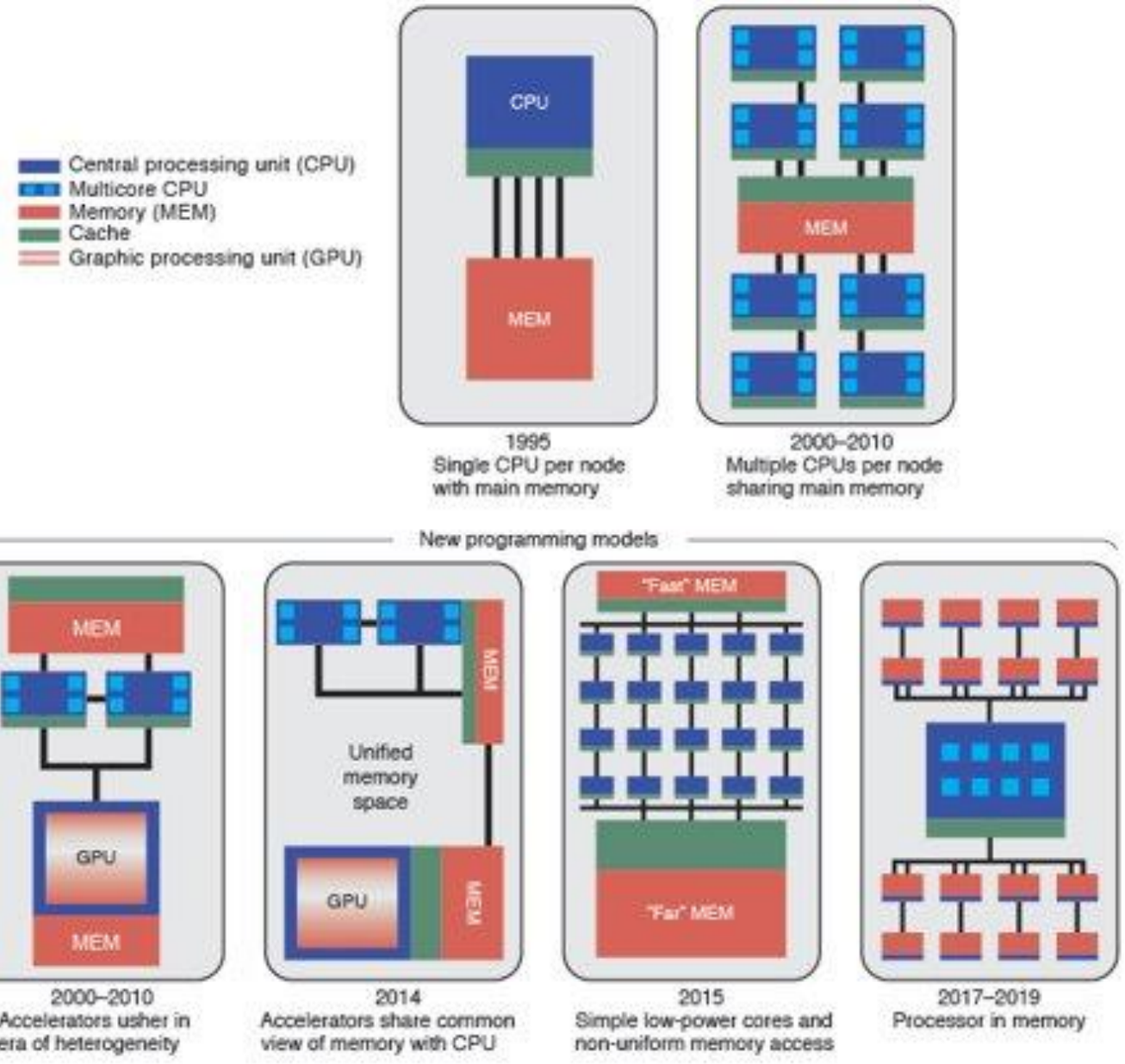
ILP: Instruction-Level Parallelism

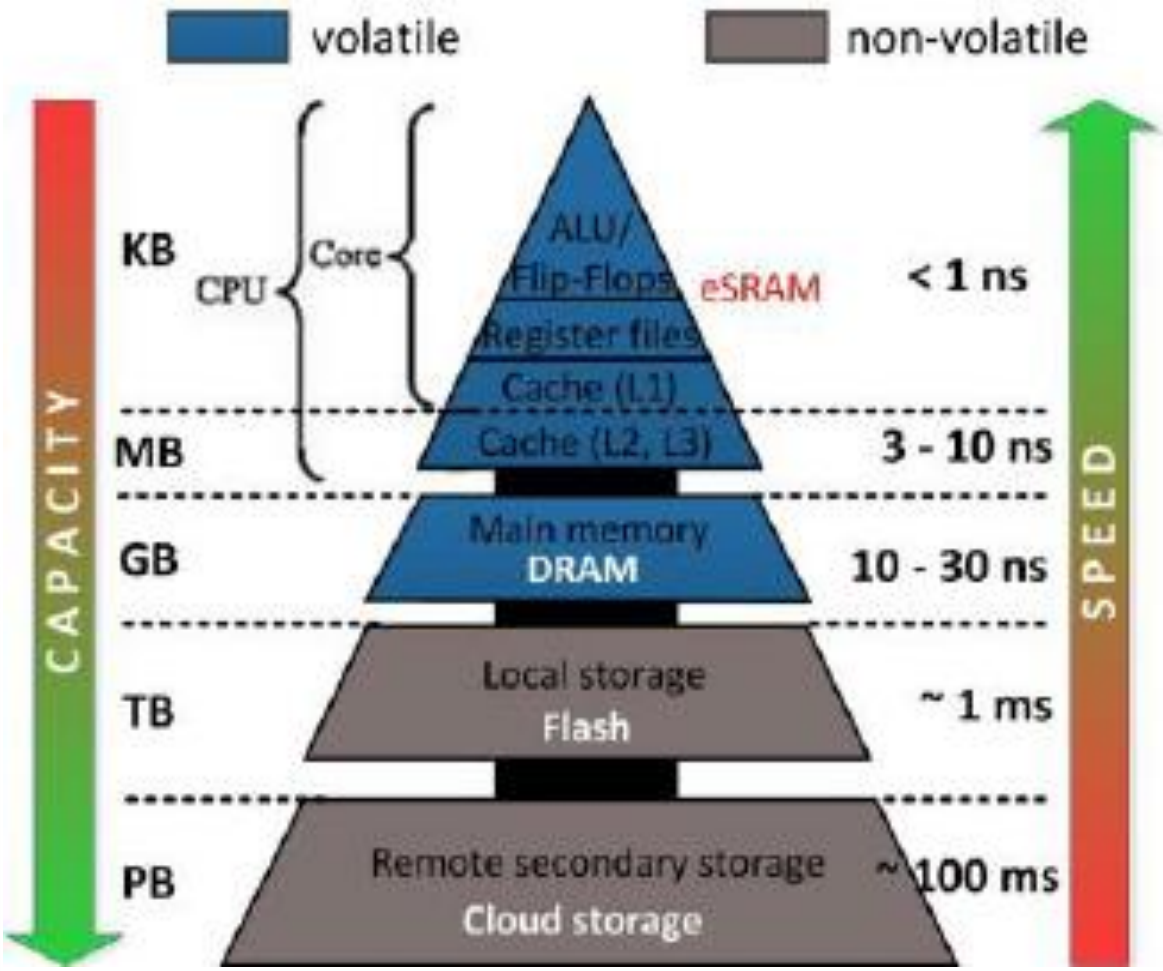
From Giga to Exa, via Tera & Peta*



*S. Borkar, Intel

Evolution of node architecture

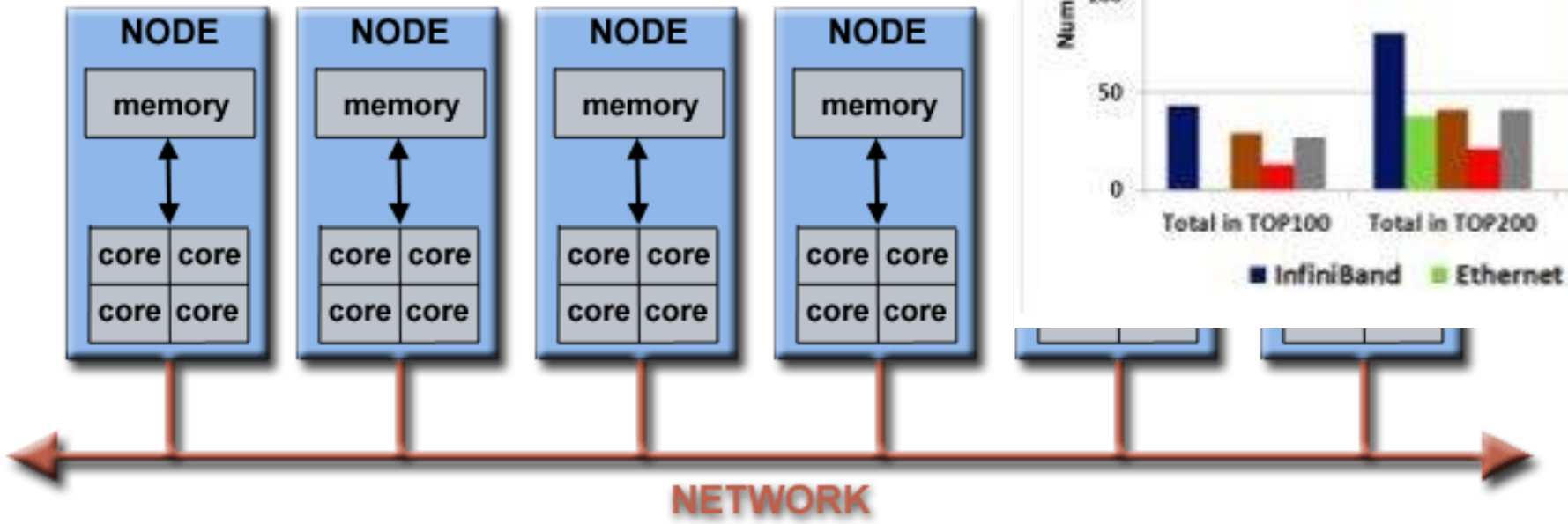


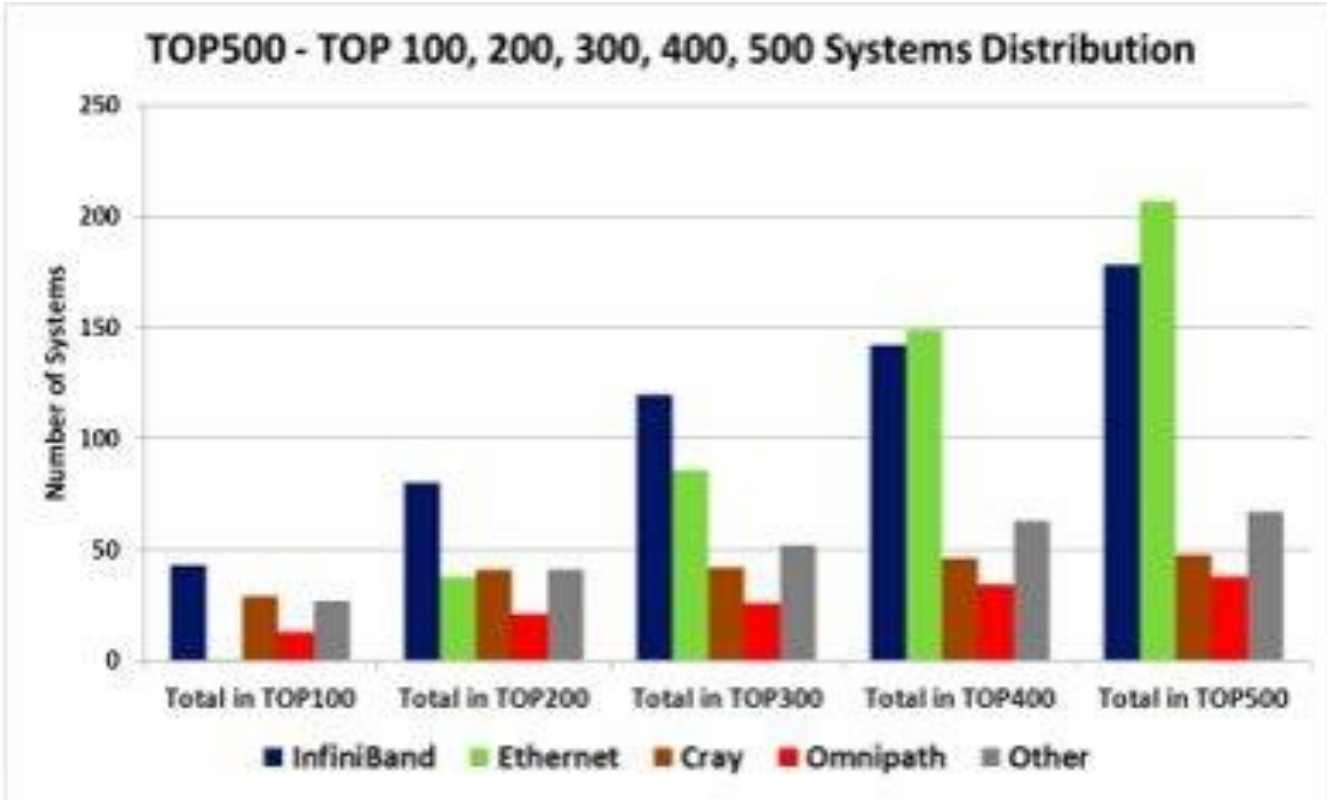


Parallel Computer Architecture

Compute Resources

- Single Computer with multiple processors.
- A number of Computers connected by a network.



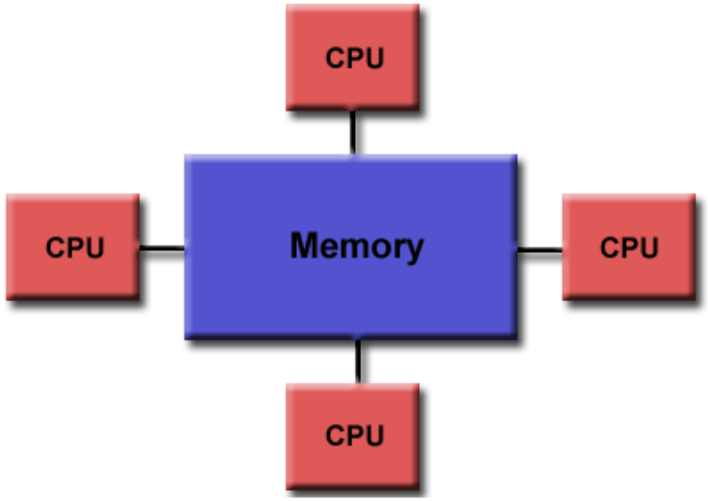


Example: Networks connect multiple standalone computers (nodes) to make larger parallel computer clusters.

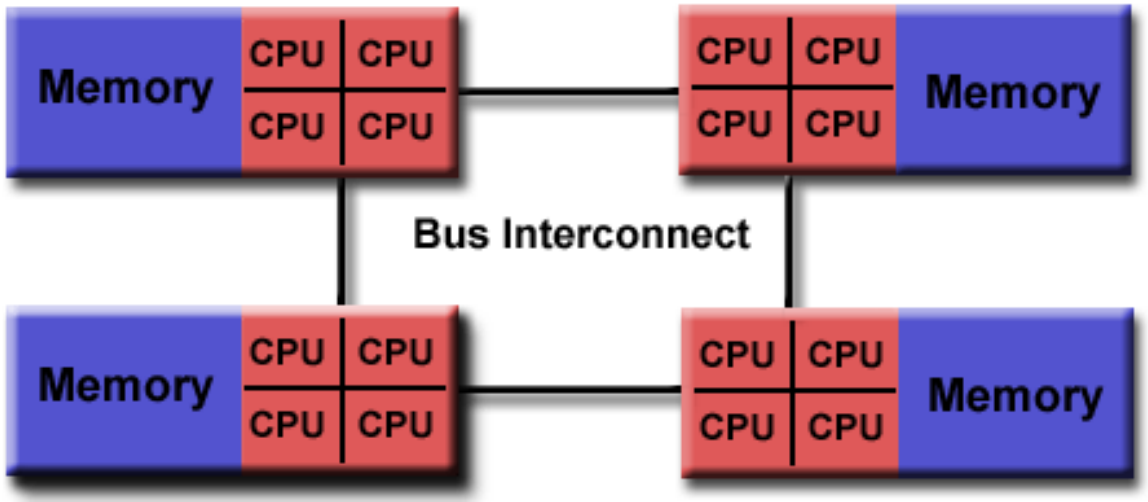
Parallel Computer Memory Architectures

Shared Memory – sharing the same address space

Symmetric Multiprocessor (SMP) machines



Shared Memory (UMA)



Shared Memory (NUMA)

Parallel Computer Memory Architectures

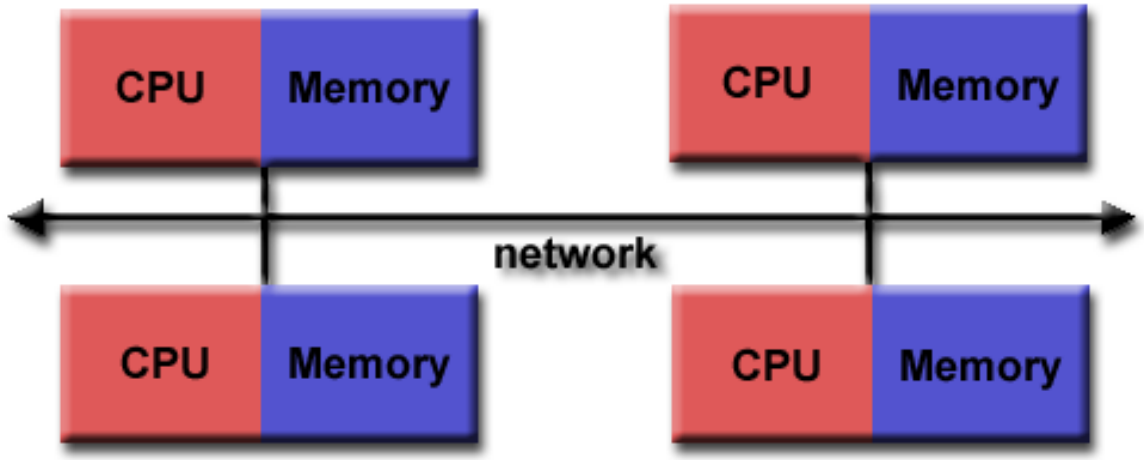
Shared Memory

- Advantages
  - Global address space provides a user-friendly programming perspective to memory
  - Data sharing between tasks is fast
  - NUMA - One SMP can directly access memory of another SMP
- Disadvantages
  - Lack of scalability between memory and CPUs.
  - Programmer responsibility for synchronization constructs that ensure "correct" access of global memory.
  - NUMA - inequal access time to all memories, Memory access across link is slower

Parallel Computer Memory Architectures

Distributed Memory

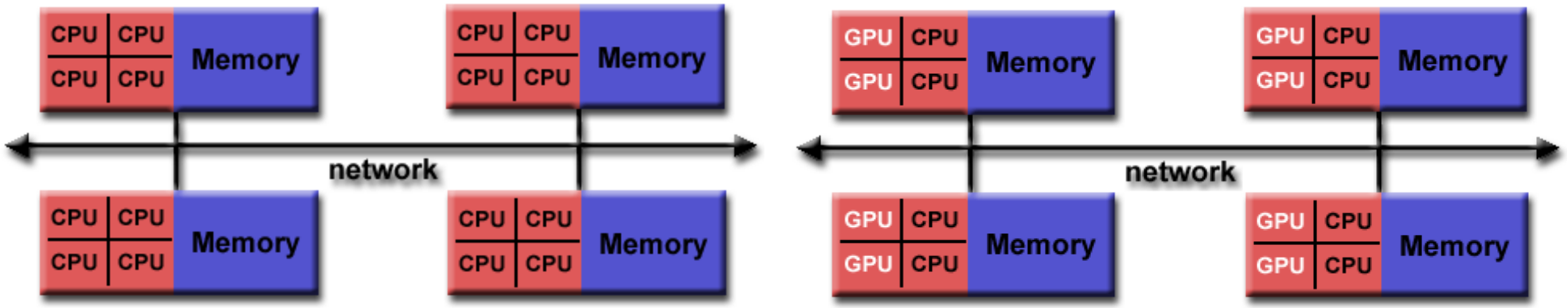
- A communication network to connect inter-processor memory.
- Processors have their own local memory. No concept of global address space across all processors.
- Changes it makes to its local memory have no effect on the memory of other processors.
- Task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.
- The network "fabric" used for data transfer varies widely



Parallel Computer Memory Architectures

Hybrid Distributed -Shared Memory

- The shared memory component can be a shared memory machine and/or graphics processing units (GPU).
- network communications are required to move data from one machine to another.
- Increased programmer complexity/effort



Limits and Costs of Parallel Programming

Parallel programs contain

- Serial Section
- Parallel Section

- Observed speedup of a code which has been parallelized, defined as:

$$\frac{\text{wall-clock time of serial execution}}{\text{wall-clock time of parallel execution}}$$

Designing Parallel Programs

- Can the problem be parallelized?
  - Calculation of the Fibonacci series (0,1,1,2,3,5,8,13,21,...) by use of the formula: F(n) = F(n-1) + F(n-2)
- Identify the program's **hotspots** *(real work)*
- Identify **bottlenecks** in the program
- Identify Data Dependencies and Task Dependencies (inhibitors to parallelism)
- Investigate other algorithms if possible & take advantage of optimized third party parallel software.
- third party parallel software and highly optimized math libraries available

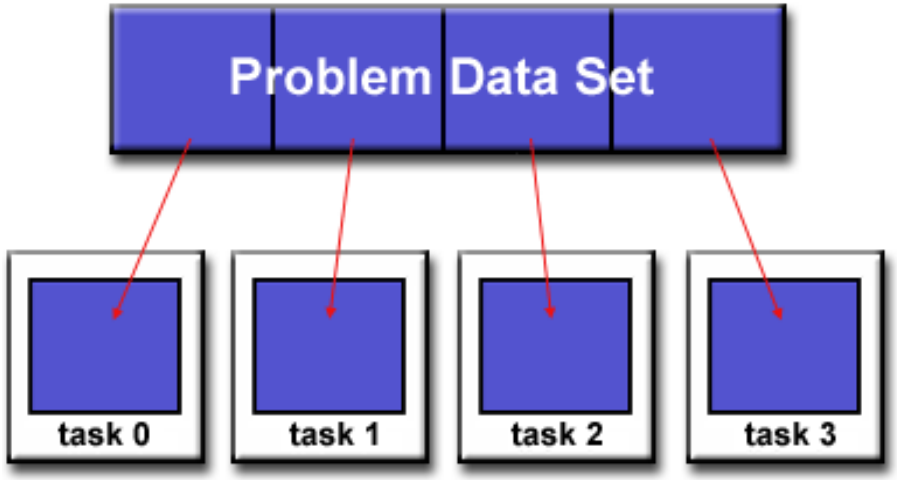
Designing Parallel Programs

Partitioning

- ***domain decomposition*** and ***functional decomposition***

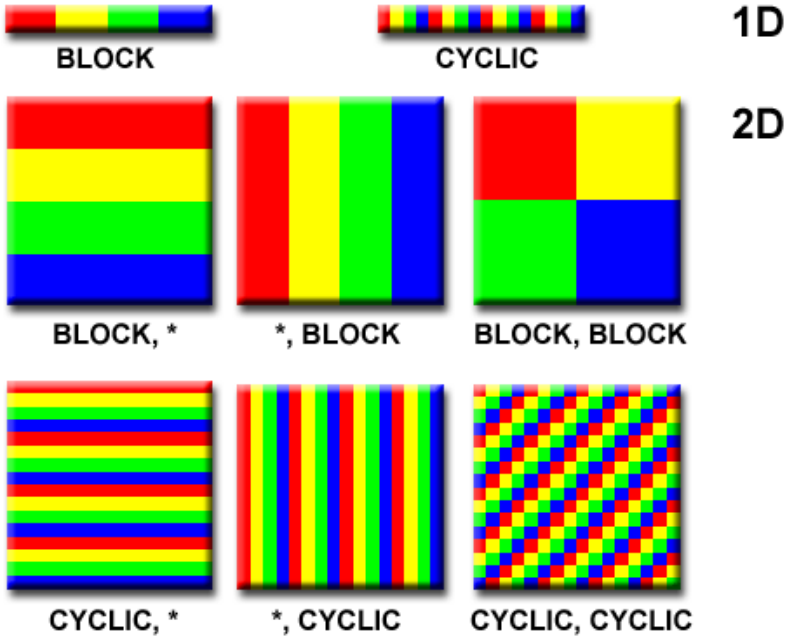
1. Data Decomposition

Data associated with the problem is decomposed



1. Domain Decomposition

Data associated with the problem is decomposed



Data Dependencies

- The order of statement execution affects the results of the program.
- Multiple use of the same location(s) in storage by different tasks.

```
DO   J =  MYSTART, MYEND
A(J) = A(J-1) * 2.0
END DO
```

Loop carried dependence

```
Task 1          Task 2
---------       ---------
X = 2           X = 4
....            ....
....            ....
Y = X**2        Y = X**3
```

Loop independent data dependency

Data Dependencies

```
DO   J =  MYSTART, MYEND
A(J) = A(J-1) * 2.0
END DO
```

If Task 2 has A(J) and task 1 has A(J-1)

- *Distributed memory architecture* - task 2 must obtain the value of A(J-1) from task 1 after task 1 finishes its computation
- *Shared memory architecture* - task 2 must read A(J-1) after task 1 updates it

Task 1	Task 2
X = 2	X = 4
....	....
Y = X**2	Y = X**3

(Race Condition) The value of Y is dependent on:

Distributed memory architecture - if or when the value of X is communicated between the tasks.

Shared memory architecture - which task last stores the value of X

Handling Dependencies

- Distributed memory architectures - communicate required data at synchronization points

- Shared memory architectures -synchronize read/write operations between tasks.
  - Data Dependencies:- Mutual Exclusion, Locks & Critical Sections

- Task Dependencies:- Explicit or Implicit Synchronization points called Barriers

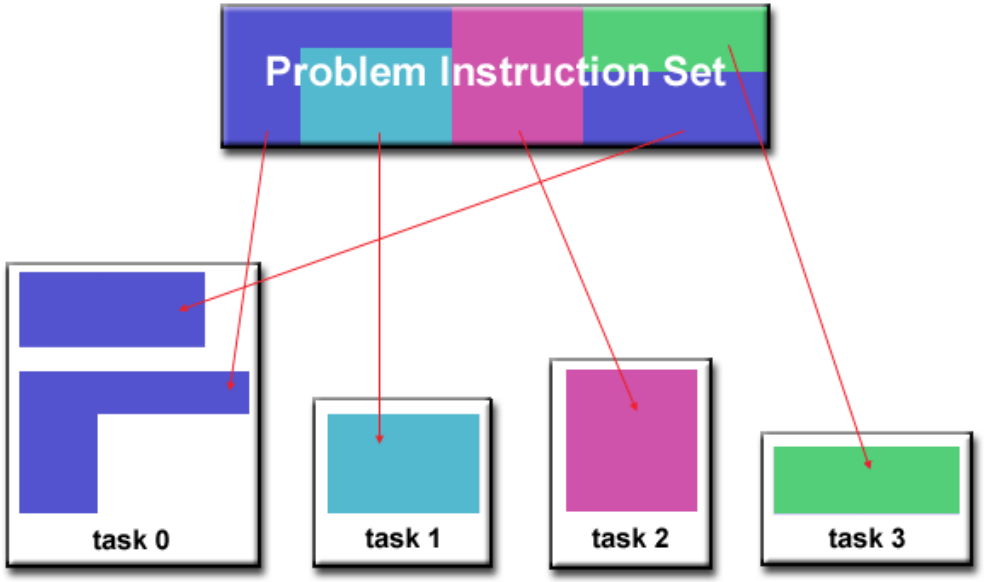
Acknowledgments

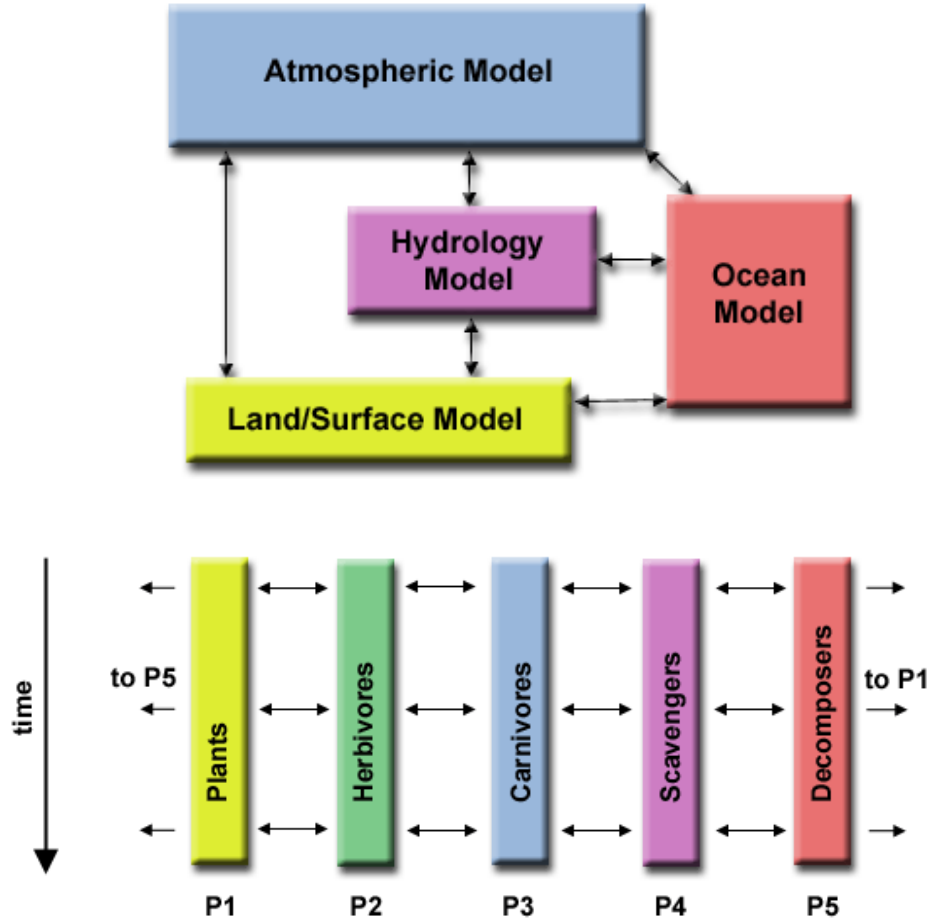
- Akhila, SERC
- S. Ethier, PPPL
- P. Messina, ECP, ANL
- LLNL HPC tutorials

THANK YOU

Designing Parallel Programs

- **Functional/ Task Decomposition**
  - The problem is decomposed according to the work that must be done. Each task then performs a portion of the overall work





Data Decomposition

For problems that operate on large amounts of data Data is divided up between CPUs : Each CPU has its own chunk of dataset to operate upon and then the results are collated.

Which data should we partition?
Input Data
Output Data
Intermediate Data

Ensure Load Balancing : Equal sized tasks not necessarily equal size data sets.

- Static Load Balancing
- Dynamic Load Balancing

Load Distribution

GOAL : Assigning the tasks/ processes to Processors while Minimizing Parallel Processing Overheads

- Maximize data locality
- Minimize volume of data-exchange
- Minimize frequency of interactions
- Minimize contention and hot spots
- Overlap computation with interactions
- Selective data and computation replication