Introduction and Overview

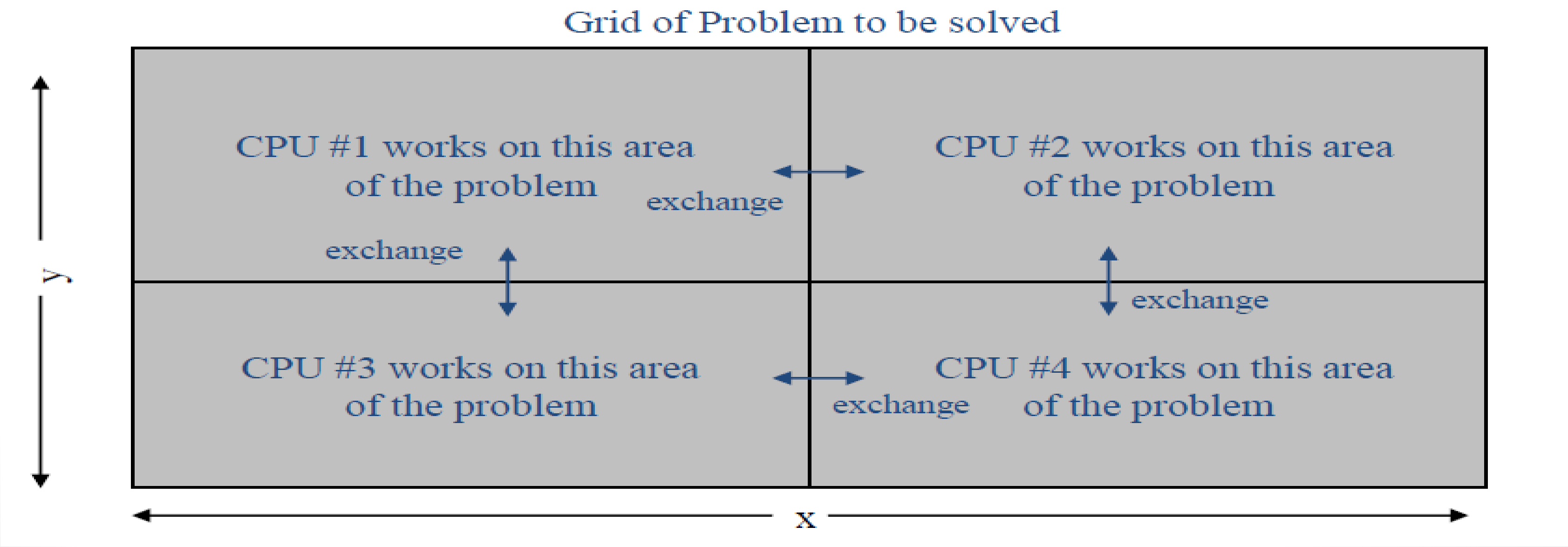
ECEG-6518 Parallel Computing

Introduction and Overview

ECEG-6518 Parallel Computing

What is Parallel Computing?

- Parallel computing: use of multiple processors or computers working together on a common task.



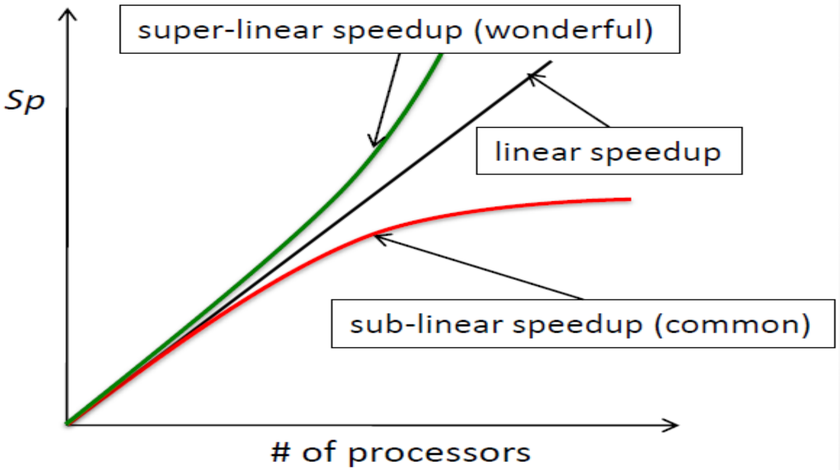
Why Do Parallel Computing?

- Limits of single CPU computing
  - performance
  - available memory
- Parallel computing allows one to:
  - solve problems that don’t fit on a single CPU
  - solve problems that can’t be solved in a reasonable time
- We can solve…
  - larger problems
  - the same problem faster
  - more cases
- All computers are parallel these days, even an iphone 4S has two cores…
  - e.g. a Qualcomm APQ 8064 (Snapdragon S4 Pro) has 4 cores @1.5GHz

Speedup & Parallel Efficiency

- Speedup: $$S_p = \frac{T_s}{T_p}$$
  - p = # of processors
  - Ts = execution time of the sequential algorithm
  - Tp = execution time of the parallel algorithm with p processors
  - $Sp = P$ (linear speedup: ideal)
- Parallel efficiency

$$E_p = \frac{S_p}{p} = \frac{T_s}{pT_p}$$



Limits of Parallel Computing

- Theoretical Upper Limits
  - Amdahl’s Law
  - Gustafson’s Law
- Practical Limits
  - Load balancing
  - Non-computational sections
- Other Considerations
  - time to re-write code

Amdahl's Law

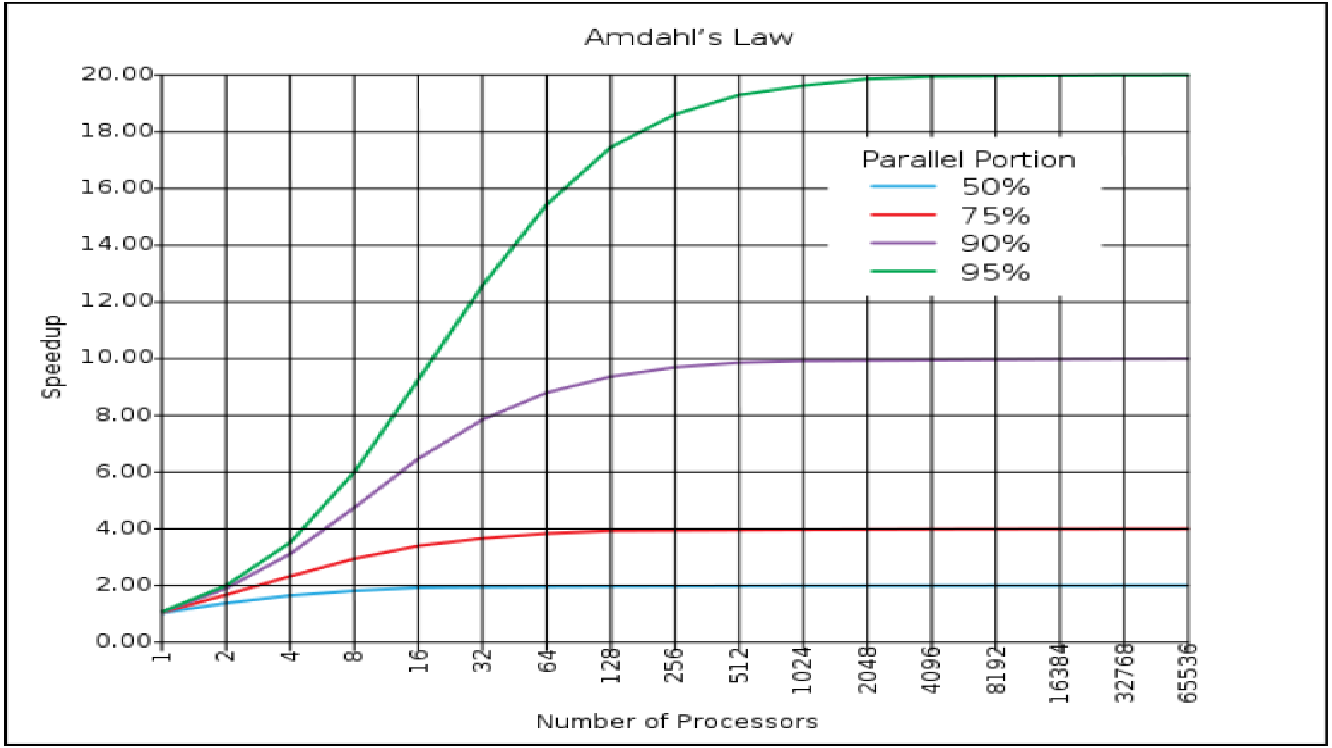
- All parallel programs contain:
  - parallel sections (we hope!)
  - serial sections (we despair!)
- Serial sections limit the parallel effectiveness
- Amdahl's Law states this formally
  - Effect of multiple processors on speed up

$$S_p = \frac{1}{f_s + \frac{f_p}{p}}$$

  - where
    - f_s = serial fraction of code
    - f_p = parallel fraction of code
    - P = number of processors

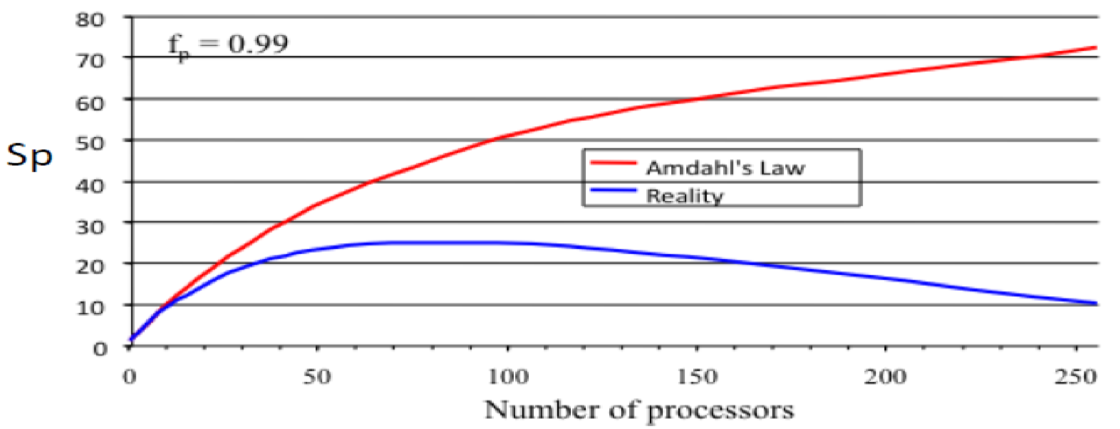
Example:
$f_s = 0.5, f_p = 0.5, P = 2$
$S_{p,\ max} = 1 / (0.5 + 0.25) = 1.333$

Amdahl’s Law


Amdahl’s Law
Speedup
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of Processors
Parallel Portion
50%
75%
90%
95%

Practical Limits: Amdahl's Law vs. Reality

- In reality, the situation is even worse than predicted by Amdahl's Law due to:
  - Load balancing (waiting)
  - Scheduling (shared processors or memory)
  - Cost of Communications
  - I/O



Gustafson’s Law

- Effect of multiple processors on run time of a problem with a *fixed amount of parallel work per processor.*

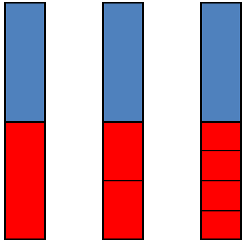
$$S_p = P - \alpha.(P - 1)$$

  - α is the fraction of non-parallelized code where the parallel work per processor is fixed (not the same as *fp* from Amdahl’s)
  - *P* is the number of processors

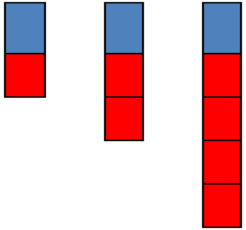
Comparison of Amdahl and Gustafson

Amdahl : fixed work

Gustafson : fixed work per processor



- $f_p = 0.5$



- α=0.5

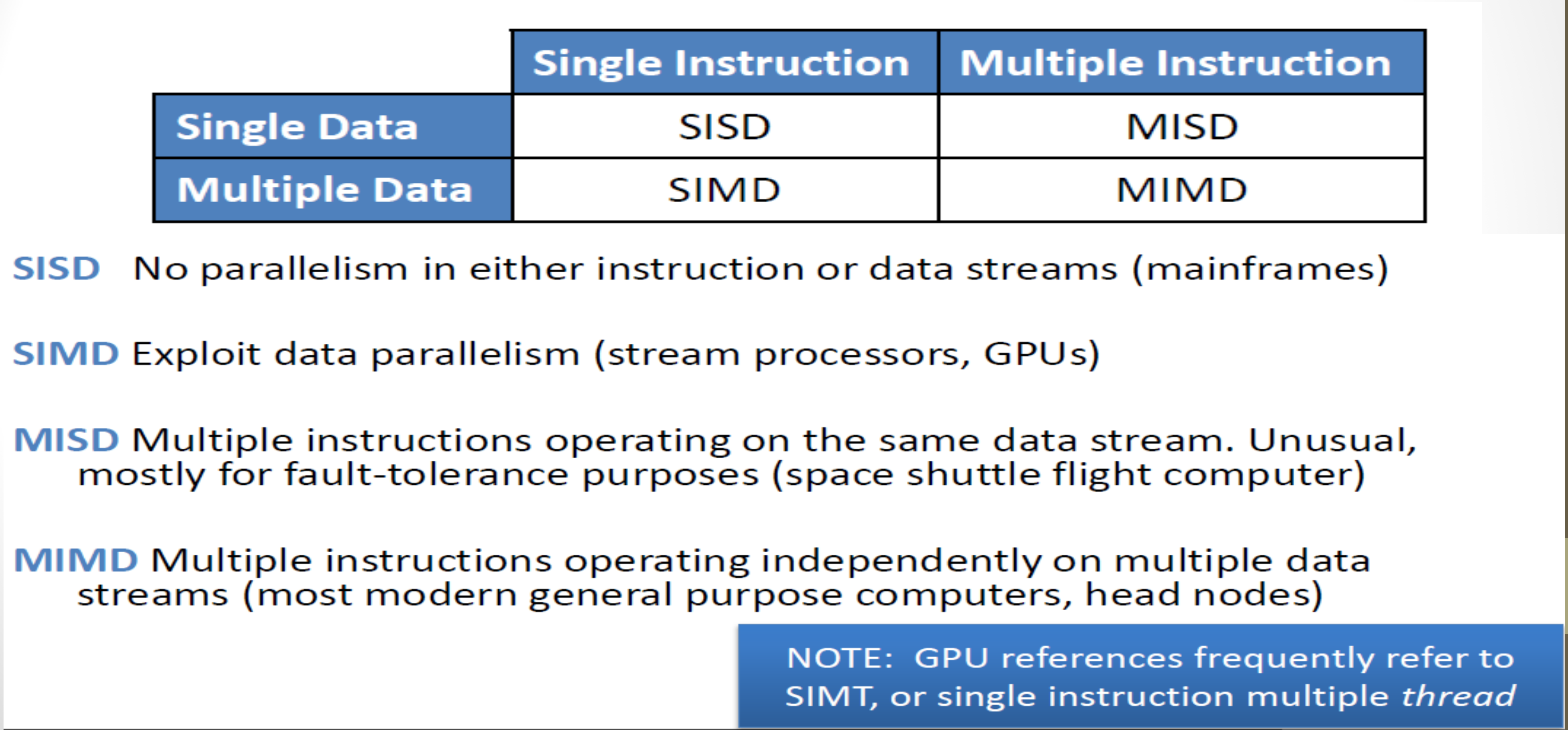
$$S_p = \frac{1}{f_s + \frac{f_p}{P}}$$

$$S_p = P - \alpha.(P-1)$$

Scaling: Strong vs. Weak

- We want to know how quickly we can complete analysis on a particular data set by increasing the PE(processing element) count
    - Amdahl’s Law
    - Known as “strong scaling”
- We want to know if we can analyze more data in approximately the same amount of time by increasing the PE count
    - Gustafson’s Law
    - Known as “weak scaling”

Hardware classification



	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

SISD No parallelism in either instruction or data streams (mainframes)

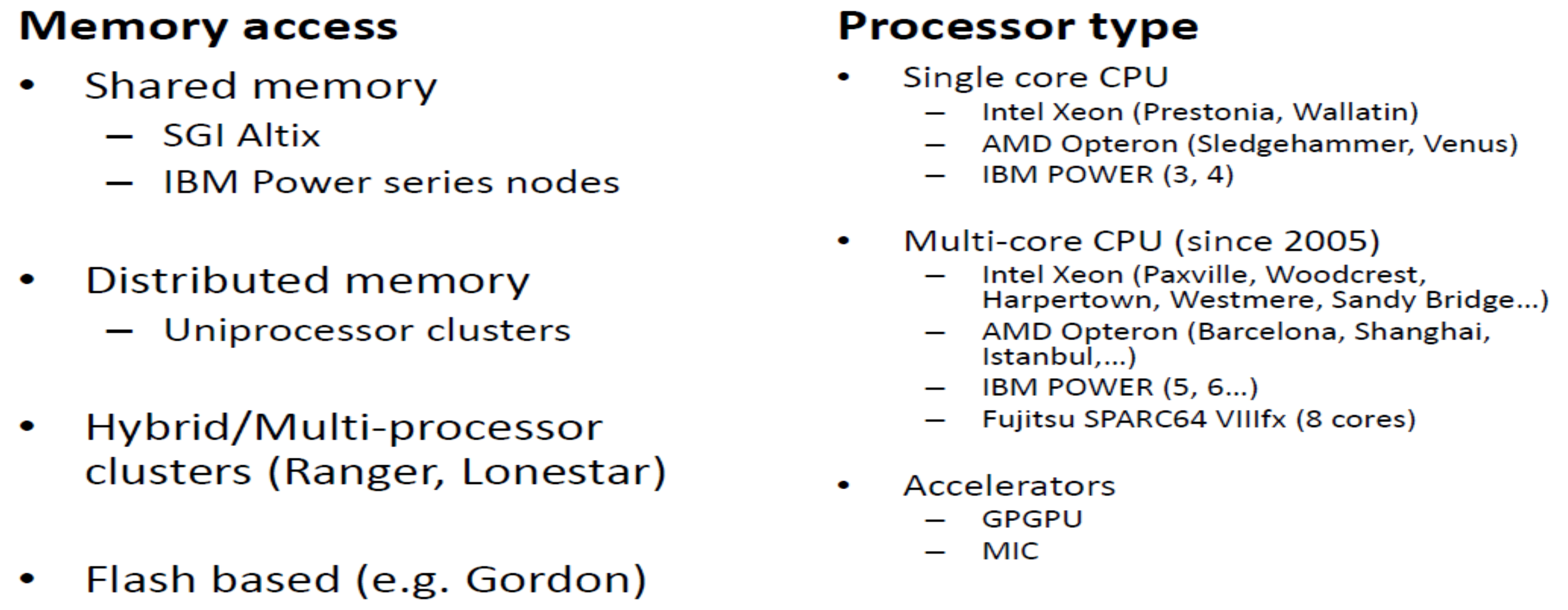
SIMD Exploit data parallelism (stream processors, GPUs)

MISD Multiple instructions operating on the same data stream. Unusual, mostly for fault-tolerance purposes (space shuttle flight computer)

MIMD Multiple instructions operating independently on multiple data streams (most modern general purpose computers, head nodes)

NOTE: GPU references frequently refer to SIMT, or single instruction multiple *thread*

Hardware in parallel computing



Memory access

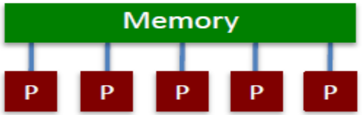
- Shared memory
  - SGI Altix
  - IBM Power series nodes
- Distributed memory
  - Uniprocessor clusters
- Hybrid/Multi-processor clusters (Ranger, Lonestar)
- Flash based (e.g. Gordon)

(Flexible Architecture for Shared Memory)

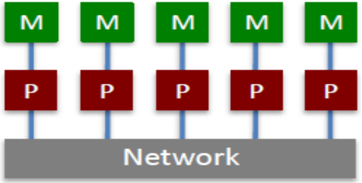
Processor type

- Single core CPU
  - Intel Xeon (Prestonia, Wallatin)
  - AMD Opteron (Sledgehammer, Venus)
  - IBM POWER (3, 4)
- Multi-core CPU (since 2005)
  - Intel Xeon (Paxville, Woodcrest, Harpertown, Westmere, Sandy Bridge...)
  - AMD Opteron (Barcelona, Shanghai, Istanbul,...)
  - IBM POWER (5, 6...)
  - Fujitsu SPARC64 VIIIfx (8 cores)
- Accelerators
  - GPGPU
  - MIC

Shared and distributed memory

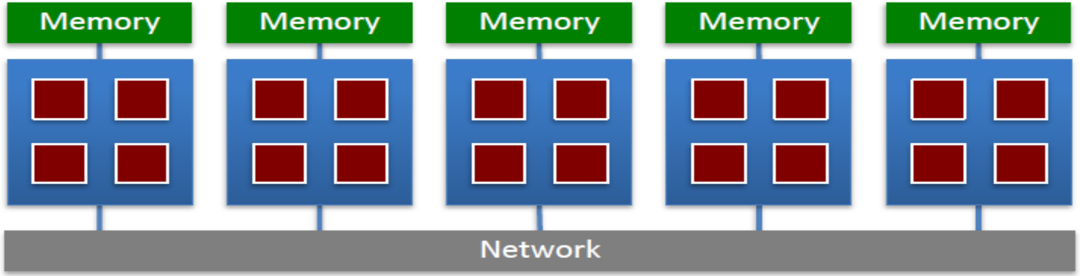


- All processors have access to a pool of shared memory
- Access times vary from CPU to CPU in NUMA systems
- Example: SGI Altix, IBM P5 nodes



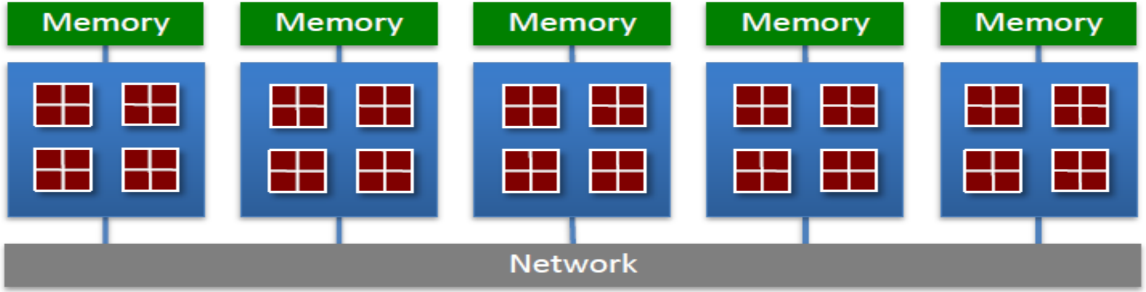
- Memory is local to each processor
- Data exchange by message passing over a network
- Example: Clusters with single-socket blades

Hybrid systems



- A limited number, N, of processors have access to a common pool of shared memory
- To use more than N processors requires data exchange over a network
- Example: Cluster with multi-socket blades

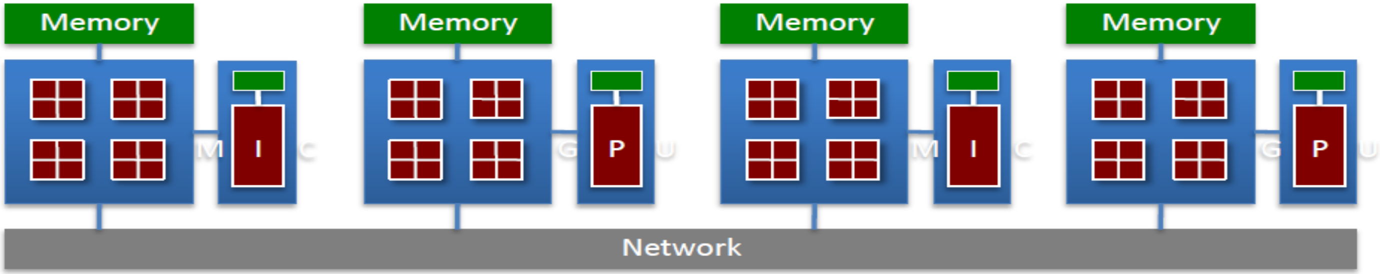
Multi-core systems



- Extension of hybrid model

- Communication details increasingly complex
  - Cache access
  - Main memory access
  - Quick Path / Hyper Transport socket connections
  - Node to node connection via network

Accelerated (GPGPU and MIC) Systems

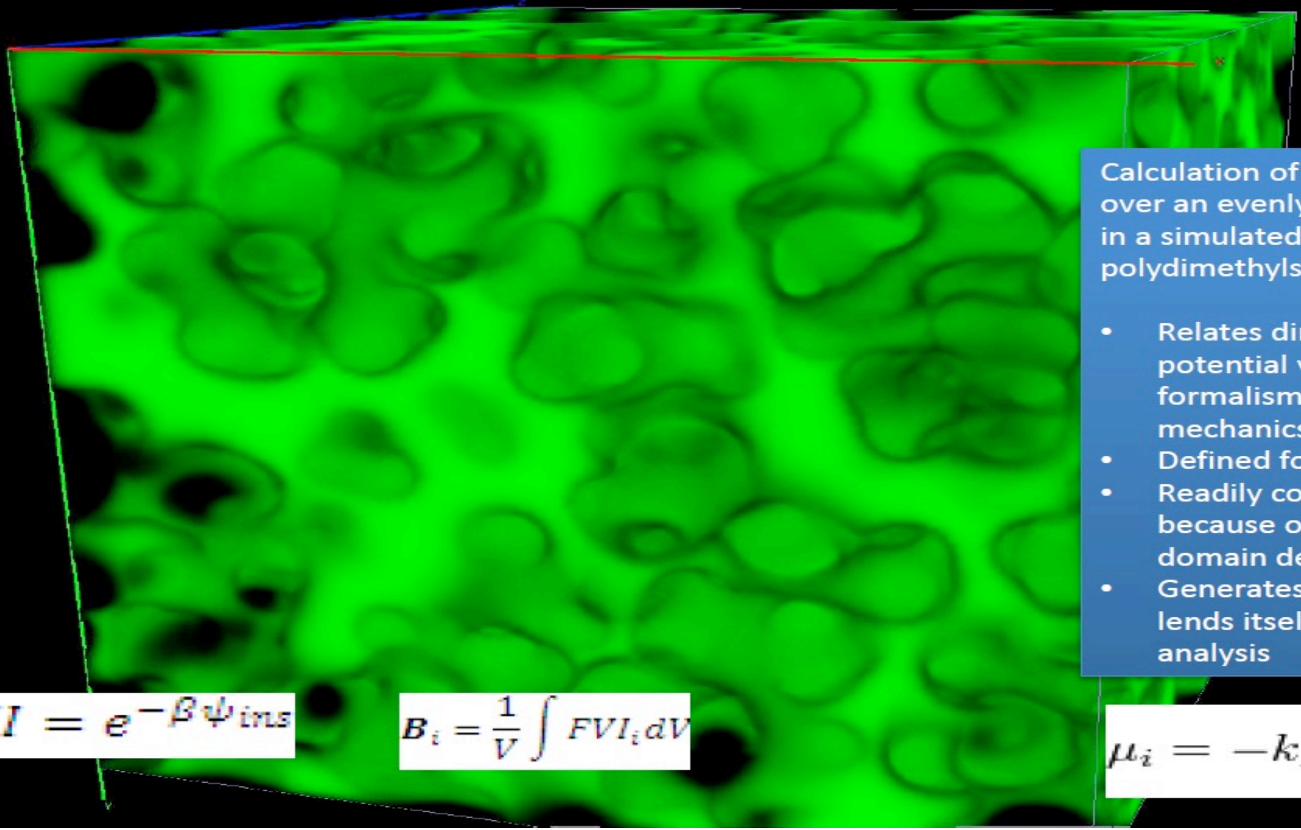


- Calculations made in both CPU and accelerator
- Provide abundance of low-cost flops
- Typically communicate over PCI-e bus
- Load balancing critical for performance

Rendering a frame: Canonical example of a GPU task

- Single instruction: “Given a model and set of scene parameters…”
- Multiple data: Evenly spaced pixel locations (x_i,y_i)
- Output: “What are my red/green/blue/alpha values at (xi, yi)?”
- The first uses of GPUs as accelerators were performed by posing physics problems as if they were rendering problems!

A GPGPU example:



Calculation of a free volume index over an evenly spaced set of points in a simulated sample of polydimethylsiloxane (PDMS)

- Relates directly to chemical potential via Widom insertion formalism of statistical mechanics
- Defined for all space
- Readily computable on GPU because of parallel nature of domain decomposition
- Generates voxel data which lends itself to spatial/shape analysis

$$FVI = e^{-\beta\psi_{ins}}$$

$$B_i = \frac{1}{V}\int FVI_i\,dV$$

$$\mu_i = -k_B T \ln\left(\frac{\mathbf{B}_i}{\rho_i \lambda^3}\right)$$

QUESTIONS?