departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Parallel Computer Architectures

Concurrency and Parallelism — 2016-17

Master in Computer Science

(Mestrado Integrado em Eng. Informática)

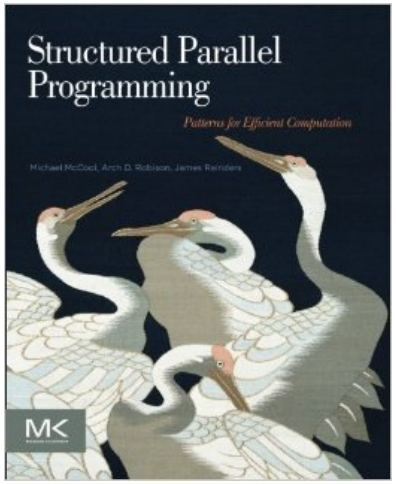
Joao Lourenço <joao.lourenco@fct.unl.pt>

Source: Parallel Computing, CIS 410/510, Department of Computer and Information Science

Outline

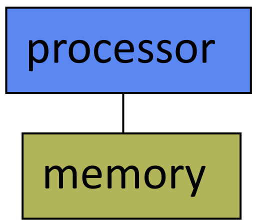
- Parallel architecture types
  - Instruction-level parallelism, Vector processing, SIMD
  - Shared memory
    - Memory organization: UMA, NUMA
    - Coherency: CC-UMA, CC-NUMA
  - Interconnection networks, Distributed memory
  - Clusters, Clusters of SMPs

  - Bibliography:
    - **Chapter 2** of book
      McCool M., Arch M., Reinders J.;
      Structured Parallel Programming: Patterns for Efficient Computation;
      Morgan Kaufmann (2012);
      ISBN: 978-0-12-415993-8

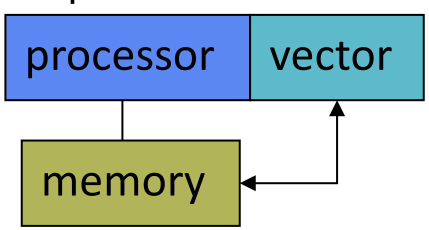


Parallel Architecture Types

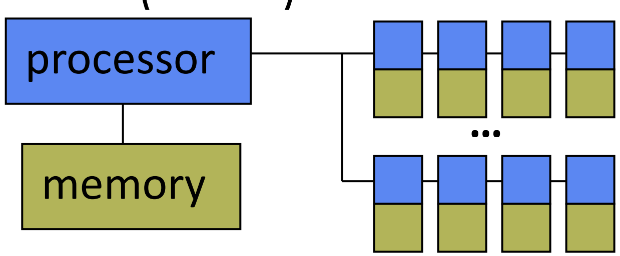
- Uniprocessor
  - Scalar processor



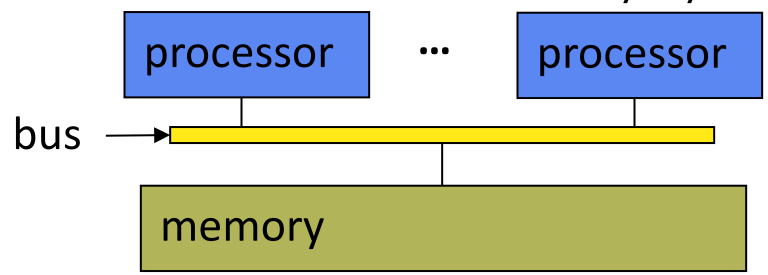
  - Vector processor



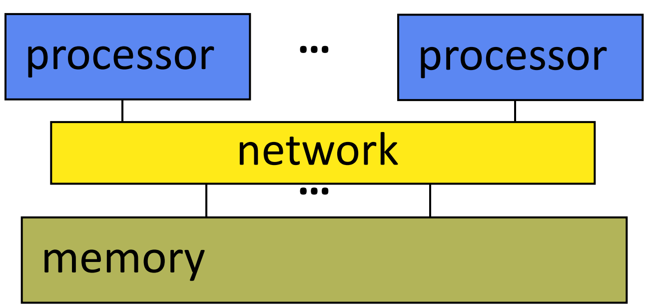
  - Single Instruction Multiple Data (SIMD)



- Shared Memory Multiprocessor (SMP)
  - Shared memory address space
  - Bus-based memory system

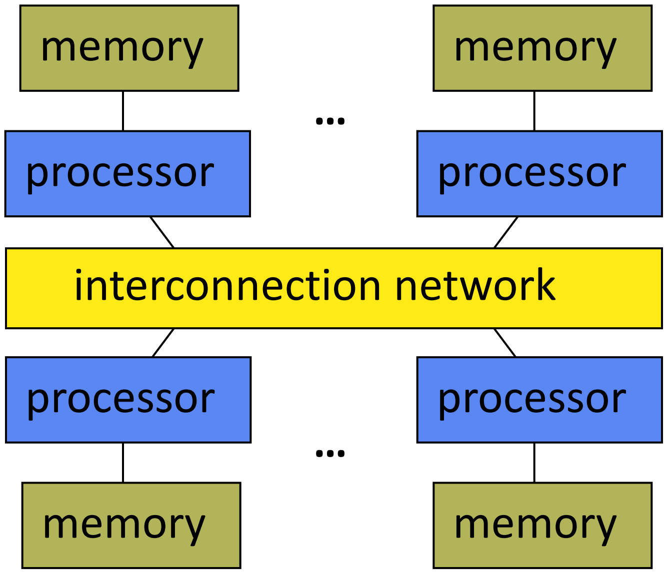


  - Interconnection network



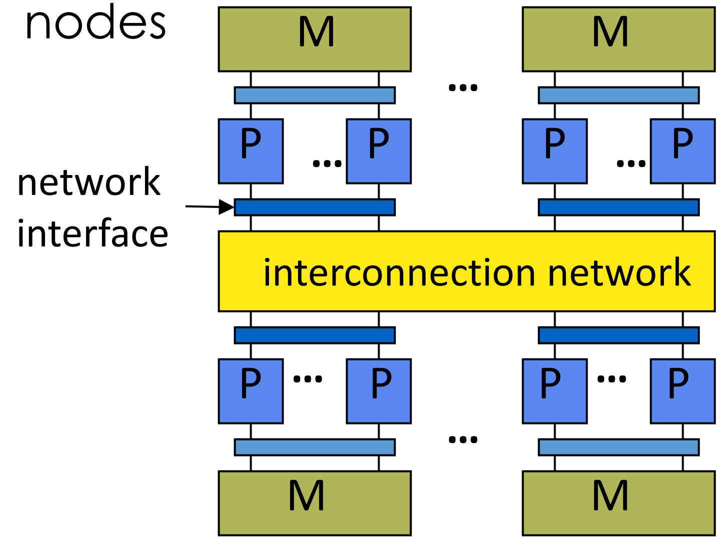
Parallel Architecture Types (2)

- Distributed Memory Multiprocessor
  - Message passing between nodes



  - Massively Parallel Processor (MPP)
    - Many, many processors

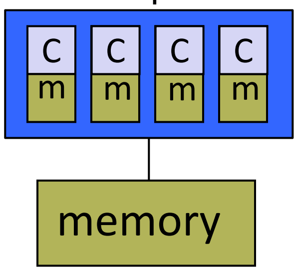
- Cluster of SMPs
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes



  - Can also be regarded as MPP if processor number is large

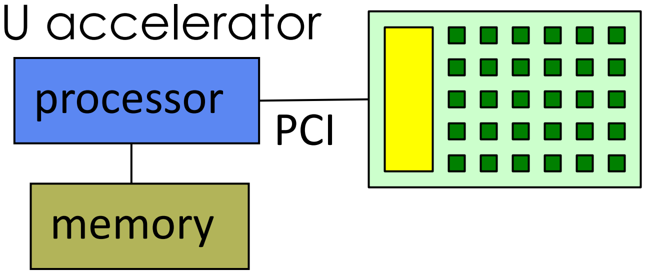
Parallel Architecture Types (3)

- Multicore
  - Multicore processor

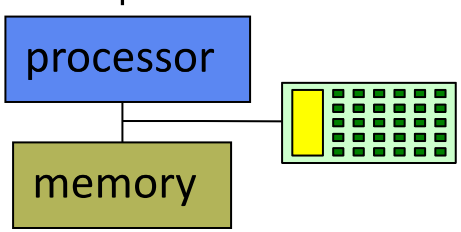


cores can be hardware multithreaded (hyperthread)

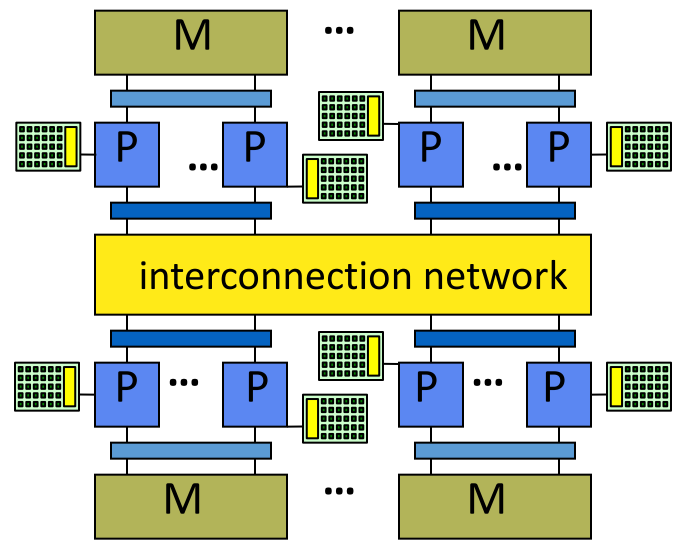
  - GPU accelerator



  - "Fused" processor accelerator



- Multicore SMP+GPU Cluster
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes
  - GPU accelerators attached



How do you get parallelism in the hardware?

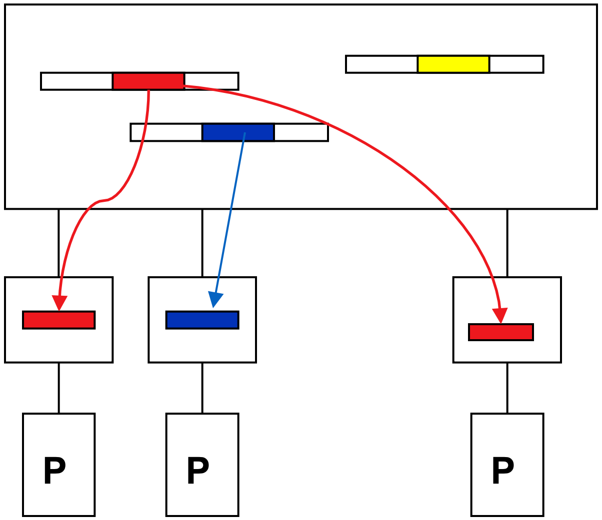
- Instruction-Level Parallelism (ILP)
- Data parallelism
  - Increase amount of data to be operated on at same time
- Processor parallelism
  - Increase number of processors
- Memory system parallelism
  - Increase number of memory units
  - Increase bandwidth to memory
- Communication parallelism
  - Increase amount of interconnection between elements
  - Increase communication bandwidth

Instruction-Level Parallelism

- Opportunities for splitting up instruction processing
- Pipelining within instruction
- Overlapped execution
- Multiple functional units
- Out of order execution
- Superscalar processing
- Superpipelining
- Very Long Instruction Word (VLIW)
- Hardware multithreading (hyperthreading)

Caching in Shared Memory Systems

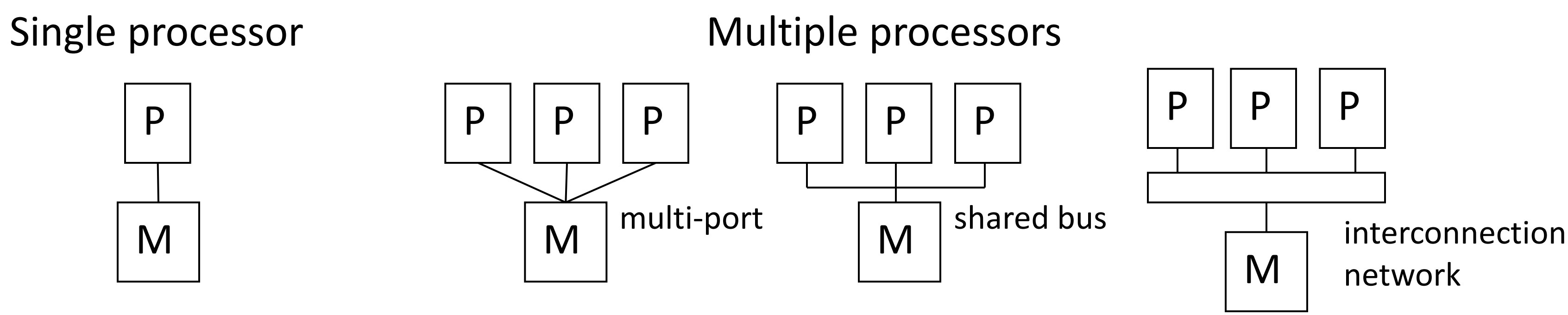
- Reduce average latency
  - automatic replication closer to processor
- Reduce average bandwidth
- Data is logically transferred from producer to consumer by memory
  - store reg $\rightarrow$ mem
  - load reg $\leftarrow$ mem



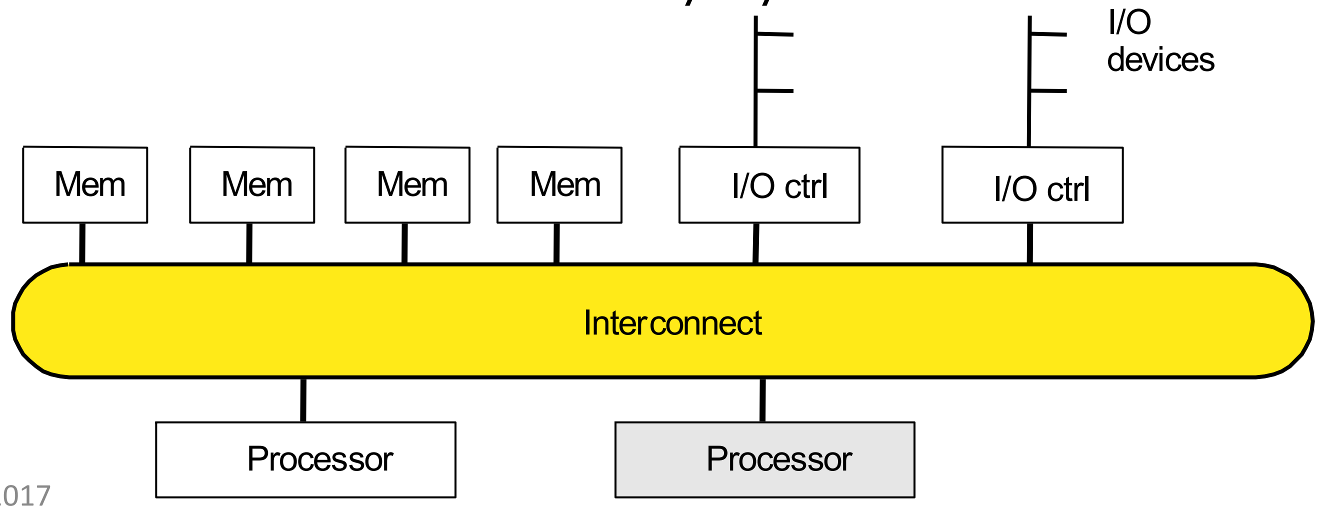
- Processors can share data efficiently
- What happens when store and load are executed on different processors?
- Cache coherence problems

Shared Memory Multiprocessors (SMP)

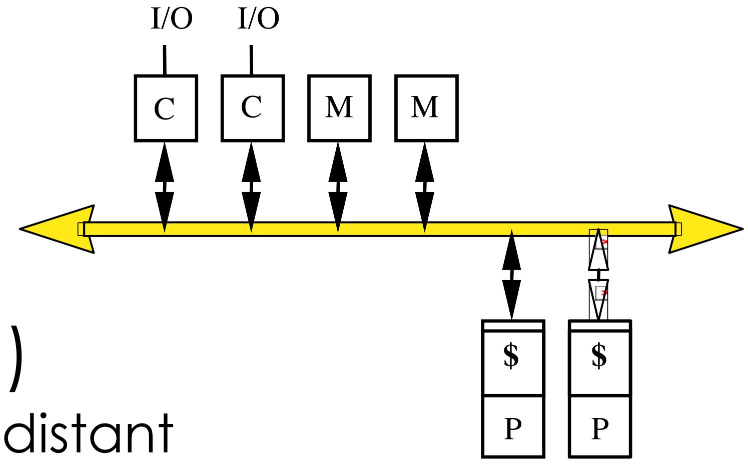
- Architecture types



- Differences lie in memory system interconnection

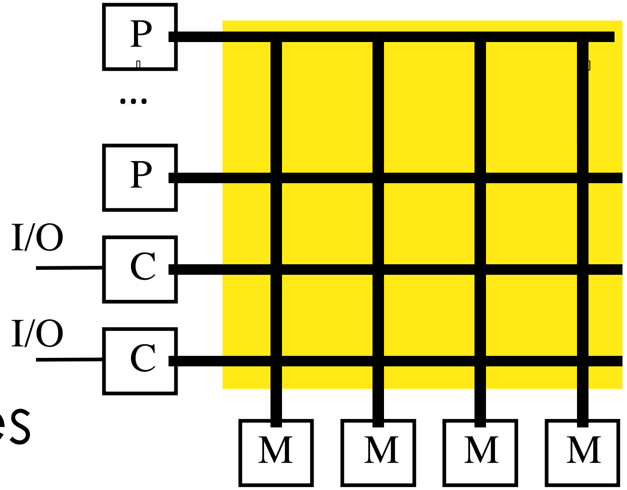


Bus-based SMP



- Memory bus handles all memory read/write traffic
- Processors share bus
- *Uniform Memory Access* (*UMA*)
  - Memory (not cache) uniformly equidistant
  - Take same amount of time (generally) to complete
- May have multiple memory modules
  - Interleaving of physical address space
- Caches introduce memory hierarchy
  - Lead to data consistency problems
  - Cache coherency hardware necessary (*CC-UMA*)

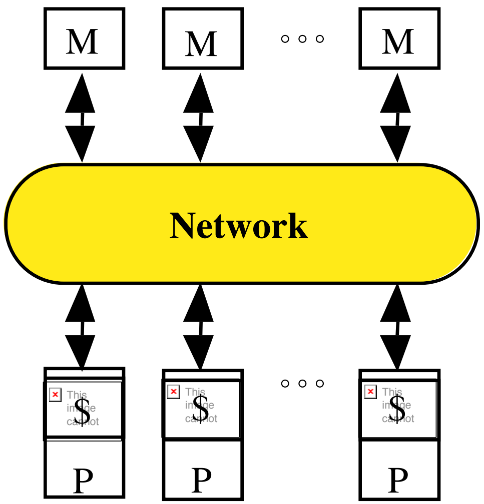
Crossbar SMP

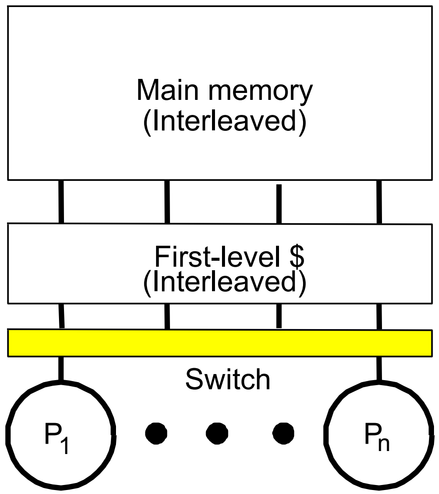


- Replicates memory bus for every processor and I/O controller
  - Every processor has direct path
- UMA SMP architecture
- Can still have cache coherency issues
- Multi-bank memory or interleaved memory
- Advantages
  - Bandwidth scales linearly (no shared links)
- Problems
  - High incremental cost (cannot afford for many processors)
  - Use switched multi-stage interconnection network

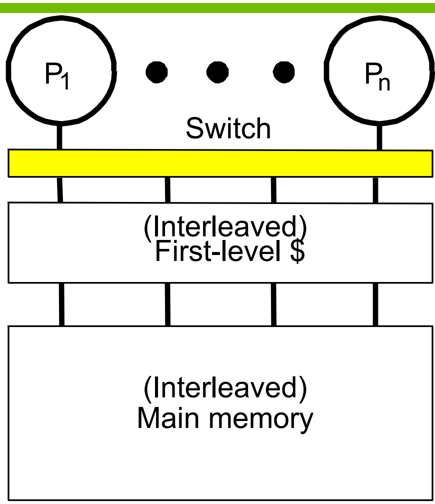
SMP and Shared Cache

- Interconnection network connects processors to memory
- Centralized memory (UMA)
- Network determines performance
  - Continuum from bus to crossbar
  - Scalable memory bandwidth
- Memory is physically separated from processors
- Could have cache coherence problems
- Shared cache reduces coherence problem and provides fine grained data sharing

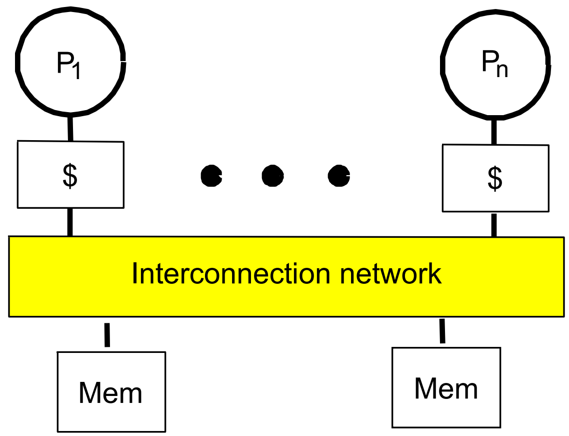




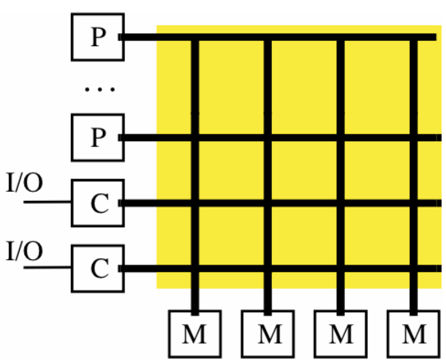
Natural Extensions of the Memory System



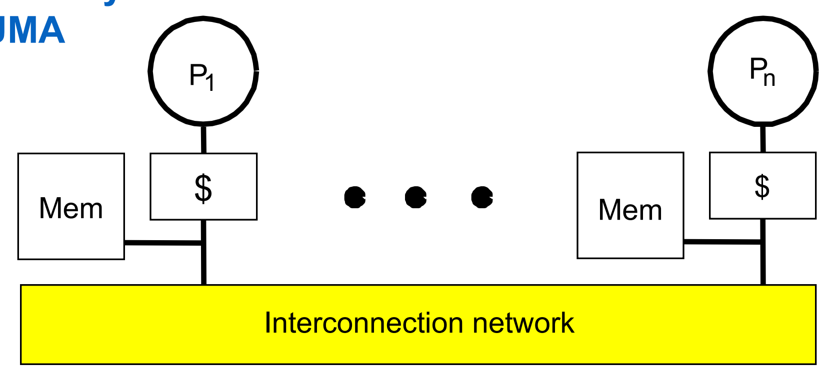
Shared Cache



Centralized Memory
Dance Hall, UMA

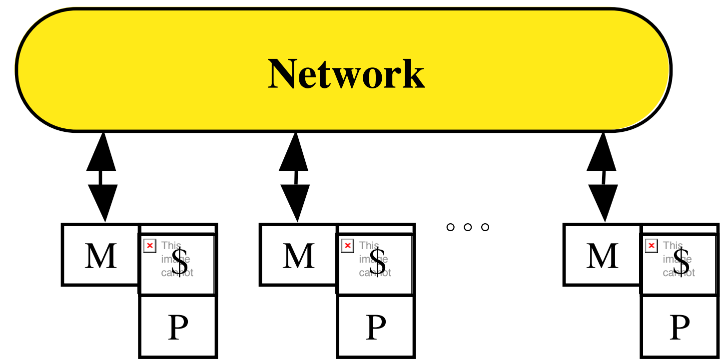


Crossbar, Interleaved



Distributed Memory (NUMA)

Non-Uniform Memory Access (NUMA) SMPs



- Distributed memory
- Memory is physically resident close to each processor
- Memory is still shared
- *Non-Uniform Memory Access* (NUMA)
  - Local memory and remote memory
  - Access to local memory is faster, remote memory slower
  - Access is non-uniform
  - Performance will depend on data locality
- Cache coherency is still an issue (more serious)
- Interconnection network architecture is more scalable

Cache Coherency and SMPs

- Caches play key role in SMP performance
  - Reduce average data access time
  - Reduce bandwidth demands placed on shared interconnect
- Private processor caches create a problem
  - Copies of a variable can be present in multiple caches
  - A write by one processor may not become visible to others
    - they'll keep accessing stale value in their caches

  $\Rightarrow$ *Cache coherence* problem
- What do we do about it?
  - Organize the memory hierarchy to make it go away
  - Detect and take actions to eliminate the problem

Definitions

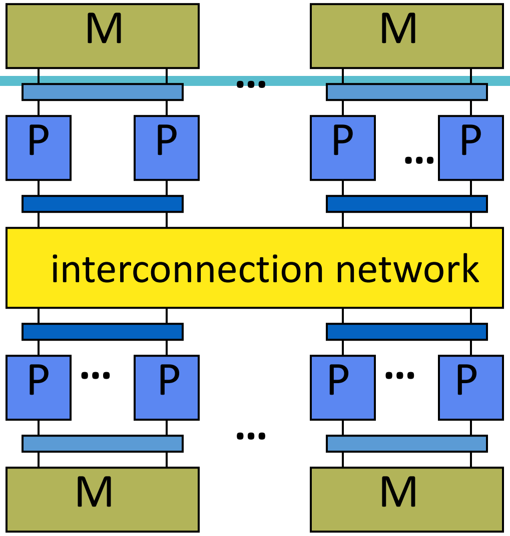
- Memory operation (load, store, read-modify-write, …)
- Memory issue is when an operation is presented to the memory system:
- Processor perspective
  - Write: subsequent reads return the value
  - Read: subsequent writes cannot affect the value
- *Coherent memory system*
  - There exists a serial order of memory operations on each location such that
    - operations issued by a process appear in order issued
    - value returned by each read is that written by previous write

  $\Rightarrow$ write propagation + write serialization

Motivation for Memory Consistency

- Coherence implies that writes to a location become visible to all processors in the same order
- But when does a write become visible?
- How do we establish orders between a write and a read by different processors?
  - Use event synchronization
- Implement hardware protocol for cache coherency
- Protocol will be based on model of memory consistency

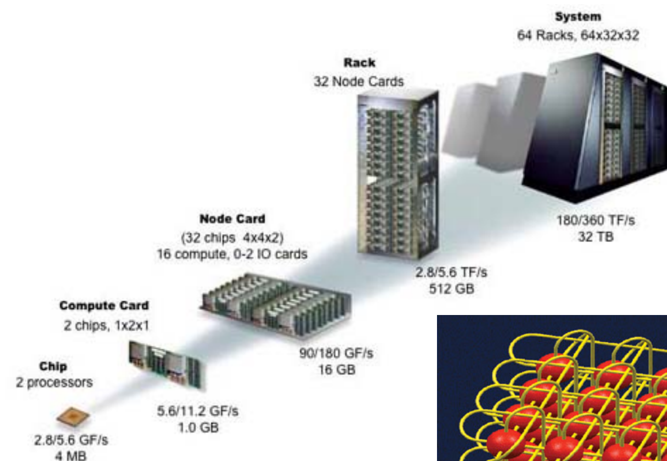
Clusters of SMPs

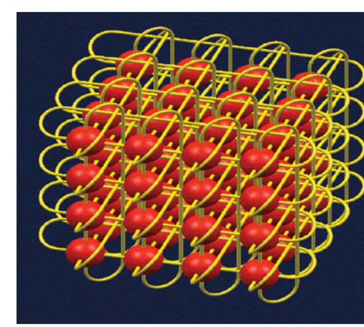


- Clustering
  - Integrated packaging of nodes
- Motivation
  - Ammortize node costs by sharing packaging and resources
  - Reduce network costs
  - Reduce communications bandwidth requirements
  - Reduce overall latency
  - More parallelism in a smaller space
  - Increase node performance
- Scalable parallel systems today are built as SMP clusters

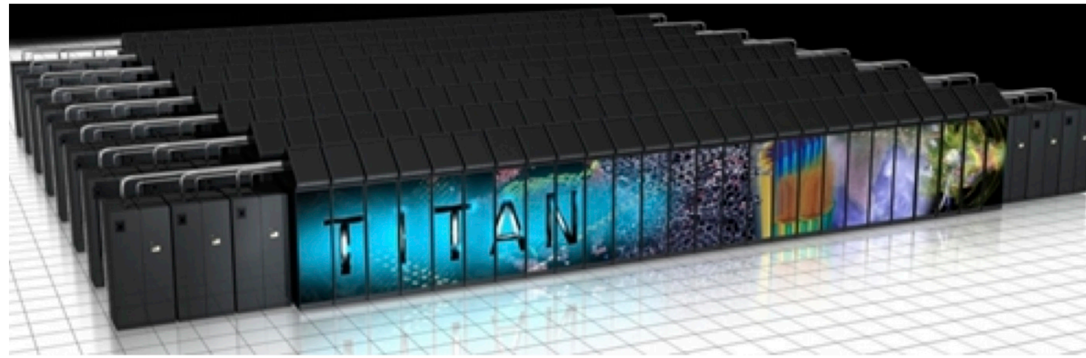
LLNL BG/L

- System hardware
  - IBM BG/L (BlueGene)
  - 65,536 dual-processor compute nodes
    - PowerPC processors
    - “double hummer” floating point
  - I/O node per 32 compute nodes
  - 32x32x64 3D torus network
  - Global reduction tree
  - Global barrier and interrupt networks
  - Scalable tree network for I/O
- Software
  - Compute node kernel (CNK)
  - Linux I/O node kernel (ION)
  - MPI
  - Different operating modes

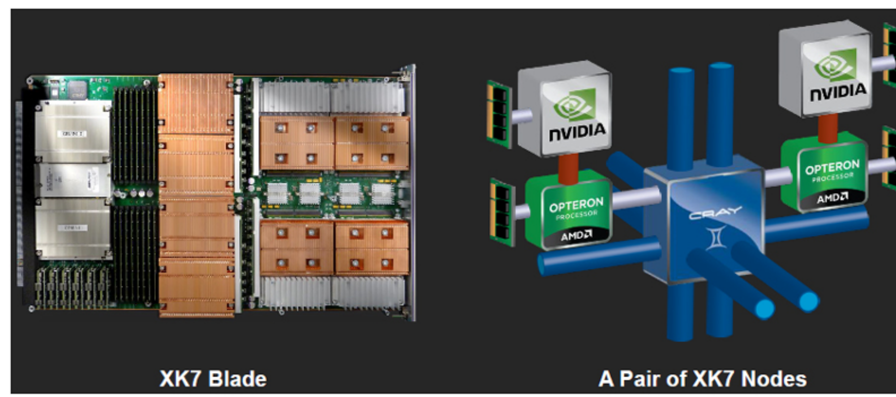


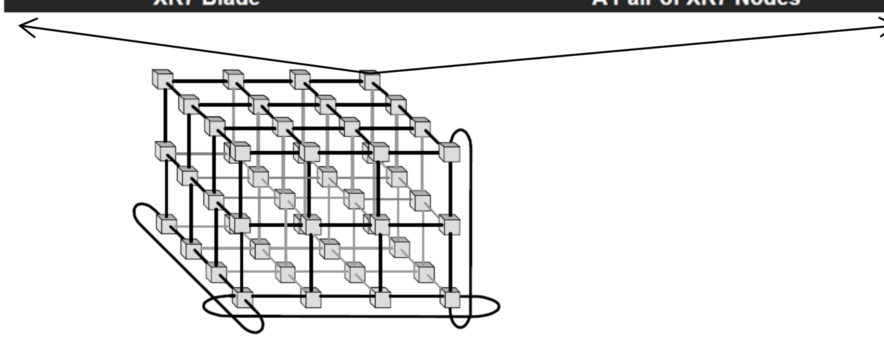


ORNL Titan (http://www.olcf.ornl.gov/titan)



- Cray XK7
  - 18,688 nodes
  - AMD Opteron
    - 16-core Interlagos
    - 299,008 Opteron cores
  - NVIDIA K20x
    - 18,688 GPUs
    - 50,233,344 GPU cores
- Gemini interconnect
  - 3D torus
- 20+ petaflops





The END