Lecture 5.2
Parallel Memory Models

EN 600.320/420/620

Instructor: Randal Burns

12 February 2018

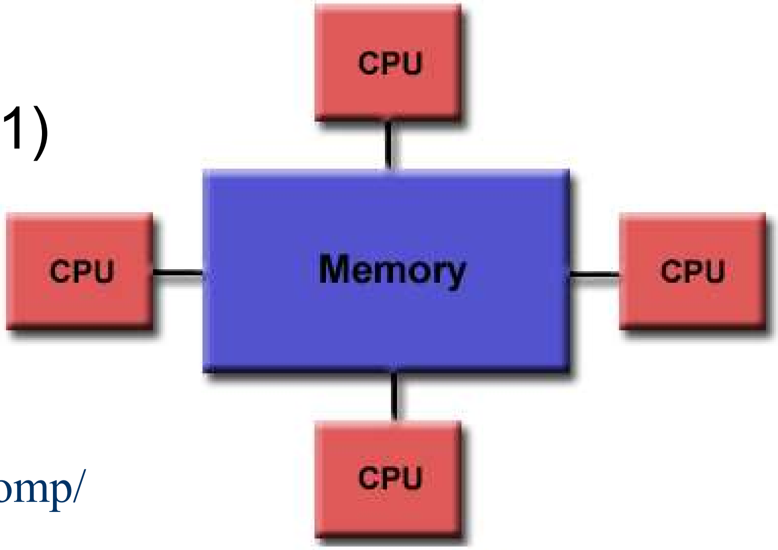
Department of Computer Science, *Johns Hopkins University*

Shared Memory Systems

- Large class defined by memory model
  - And thus, the programming model
- Shared-memory programming
  - Threads exchange information through reads and writes to memory
  - Synchronization constructs to control sharing
  - Easy to use abstraction
- Examples
  - OpenMP, Java, pthreads

Symmetric Multi-Processor (SMP)

- Shared memory MIMD system
  - All processors can address all memory
- Symmetric access to memory
  - Performance statement
- SMPs have scaling limits
- On symmetry
  - SMP not symmetric to caches
  - Multi-core (symmetric to L2, not L1)

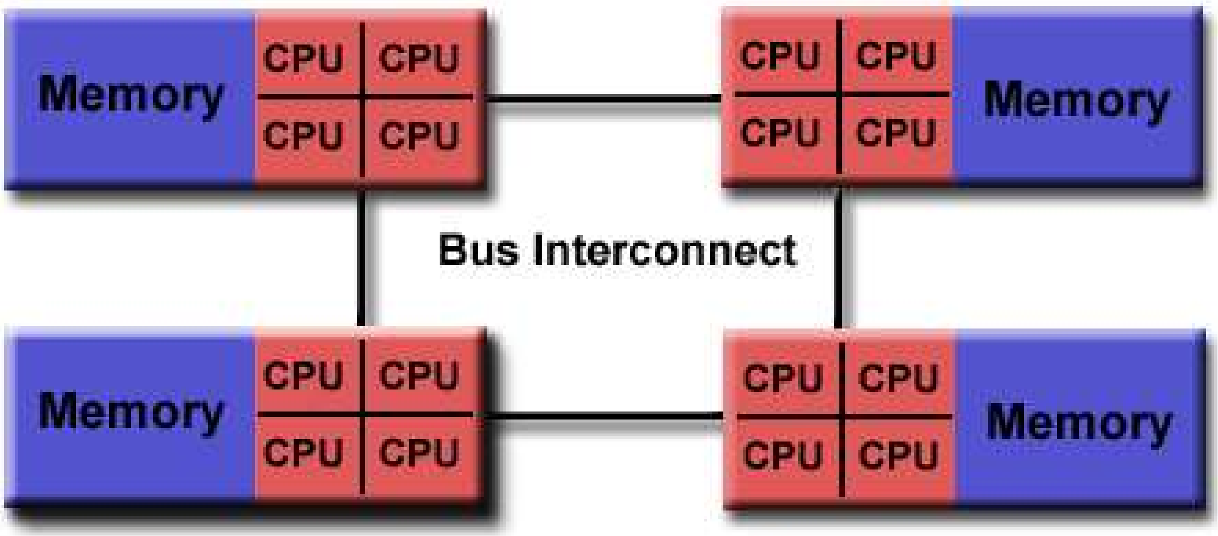


Shared Memory (UMA)

https://computing.llnl.gov/tutorials/parallel_comp/

NUMA: Non-Uniform Memory Access

- Shared memory MIMD systems
- Latency and bandwidth to physical memory differs
  - by address and location
- Same programming semantics as SMP



Shared Memory (NUMA)

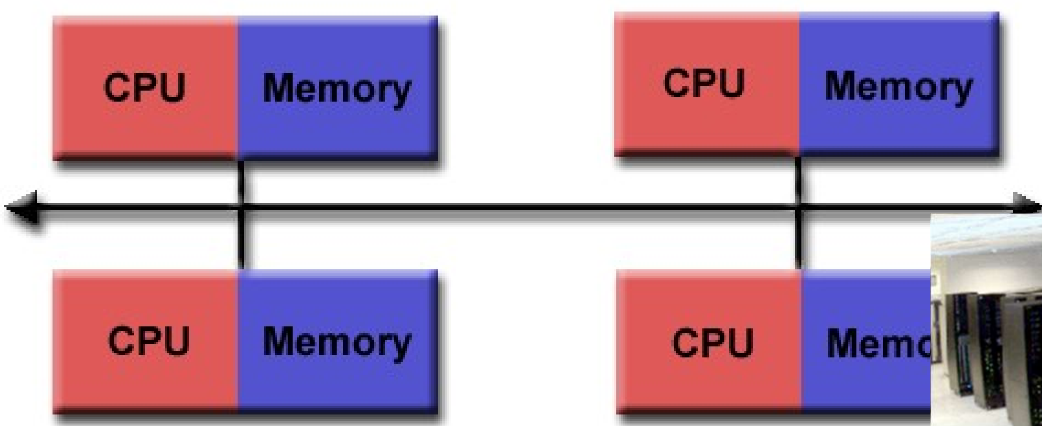
https://computing.llnl.gov/tutorials/parallel_comp/

RB's Take on NUMA

- Very difficult to program
  - The tools don't help programmer account for NU
  - Easy to write programs that work correctly
  - More difficult to write programs that run fast
- But, all multicore is NUMA
  - Even SMPs today have NUMA properties
  - Because of cache hierarchy
- New programming tools to help in Linux
  - hwloc: https://www.open-mpi.org/projects/hwloc/
  - libnuma and numactl: http://oss.sgi.com/projects/libnuma/

Message Passing

- Book calls these distributed memory machines
  - This term is deceptive to me
- Each processor/node has its own private memory
- Nodes synchronize actions and exchange data by sending messages to each other



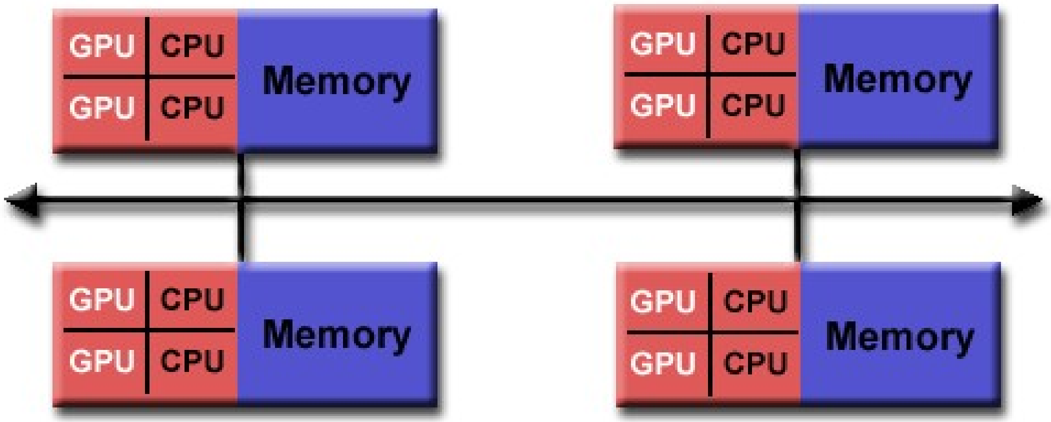
https://computing.llnl.gov/tutorials/parallel_comp/

Programming Message Passing

- MPI
  - The "assembly language" of supercomputing
  - Libraries that allow for collective operations, synchronization, etc.
  - Explicit handling of data distribution and inter-process communication
- Map/reduce and other cloud systems
  - New paradigm that emerged from Google
  - Divide computation into data parallel and data dependent portions
  - Better abstraction of HW. More restrictive.
  - MR, Hadoop!, Spark, GraphLab, etc.

Hybrid Architectures

- When a message passing machine has SMP parallelism at each of its nodes
  - Book is behind on this trend: every machine is a hybrid
- How to program
  - MPI: ignore the SMP aspects
  - MPI + (OpenMPI, pthreads, Java, CUDA, OpenCL)
    - Expensive, hard to maintain
  - Automated compilation



https://computing.llnl.gov/tutorials/parallel_comp/