High Performance Computing Systems (CMSC714)

Lecture 2: Terminology and Definitions

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

Announcements

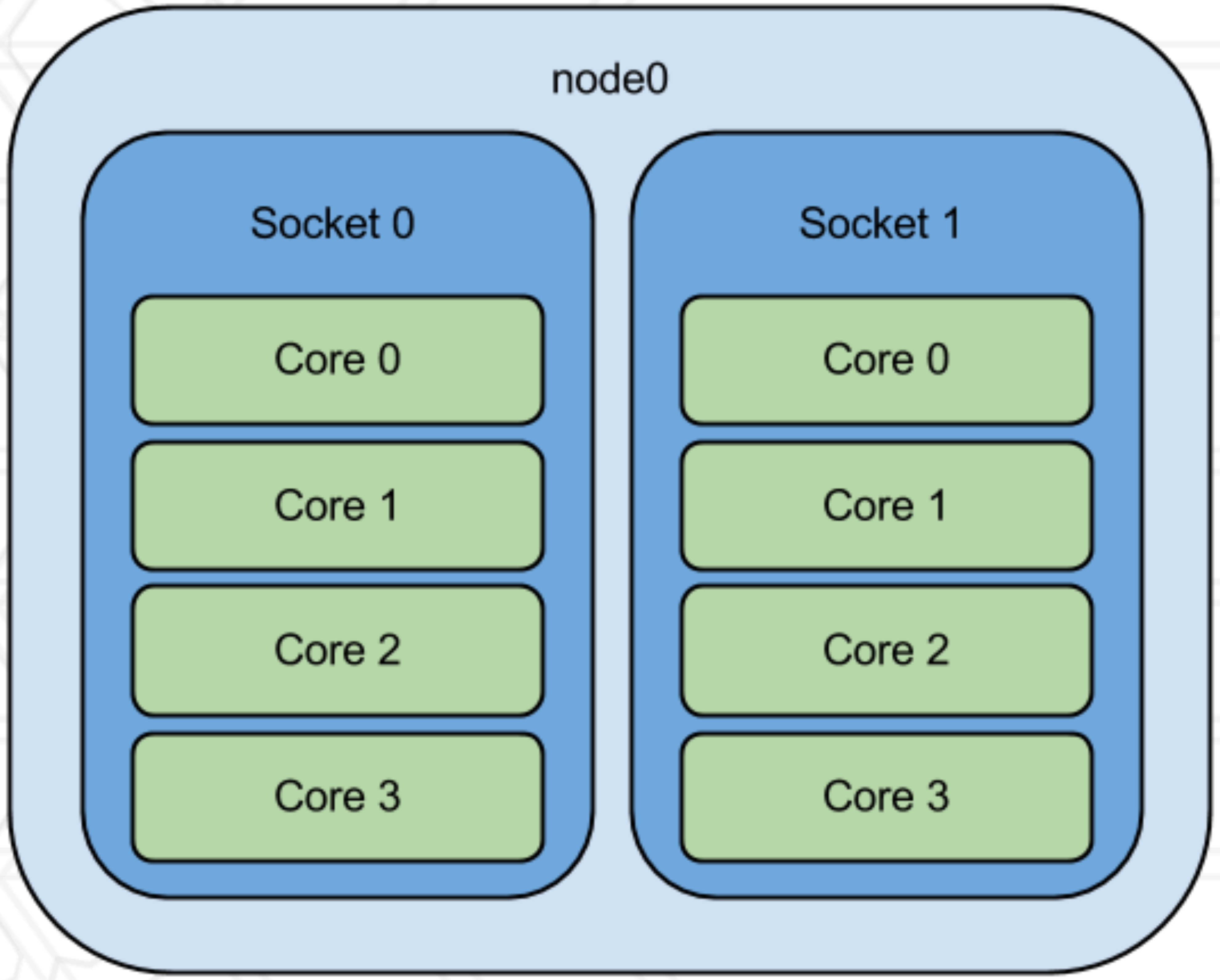
- ELMS/Canvas page for the course is up: myelms.umd.edu
  - https://umd.instructure.com/courses/1273118
- Slides from previous class are now posted online
- Assignments, project and midterm dates are also online

Summary of last lecture

- Need for high performance computing
- Parallel architecture: nodes, memory, network, storage
- Programming models: shared memory vs. distributed
- Performance and debugging tools
- Systems issues: job scheduling, routing, parallel I/O, *fault tolerance, power*
- Parallel algorithms and applications

Cores, sockets, nodes

- CPU: processor
  - Single or multi-core: core is a processing unit, multiple such units on a single chip make it a multi-core processor
- Socket: chip
- Node: packaging of sockets



https://www.glennklockwood.com/hpc-howtos/process-affinity.html

Serial vs. parallel code

- Thread: a thread or path of execution managed by the OS
- Process: heavy-weight, processes do not share resources such as memory, file descriptors etc.
- Serial or sequential code: can only run on a single thread or process
- Parallel code: can be run on one or more threads or processes

Scaling and scalable

- Scaling: running a parallel program on 1 to n processes
  - 1, 2, 3, … , n
  - 1, 2, 4, 8, …, n
- Scalable: A program is scalable if it's performance improves when using more resources

Weak versus strong scaling

- Strong scaling: *Fixed total* problem size as we run on more processes
- Weak scaling: Fixed problem size per process but *increasing total* problem size as we run on more processes

Speedup and efficiency

- Speedup: Ratio of execution time on one process to that on *n* processes

$$\text{Speedup} = \frac{t_1}{t_n}$$

- Efficiency: Speedup per process

$$\text{Efficiency} = \frac{t_1}{t_n \times n}$$

Amdahl's law

- Speedup is limited by the serial portion of the code
  - Often referred to as serial "bottleneck"
- Lets say only a fraction p of the code can be parallelized on n processes

$$\text{Speedup} = \frac{1}{(1-p) + p/n}$$

Supercomputers vs. commodity clusters

- Typically, supercomputer refers to customized hardware
  - IBM Blue Gene, Cray XT, Cray XC
- Cluster refers to a parallel machine put together using off-the-shelf hardware

Communication and synchronization

- Each physical node might compute independently for a while
- When data is needed from other (remote) nodes, messaging occurs
  - Referred to as communication or synchronization or MPI messages
- Intra-node vs. inter-node communication
- Bulk synchronous programs: All processes compute simultaneously, then synchronize together

Different models of parallel computation

- SIMD: Single Instruction Multiple Data
- MIMD: Multiple Instruction Multiple Data
- SPMD: Single Program Multiple Data
  - Typical in HPC

Writing parallel programs

- Decide the algorithm first
- Data: how to distribute data among threads/processes?
  - Data locality
- Computation: how to divide work among threads/processes?

Writing parallel programs: examples

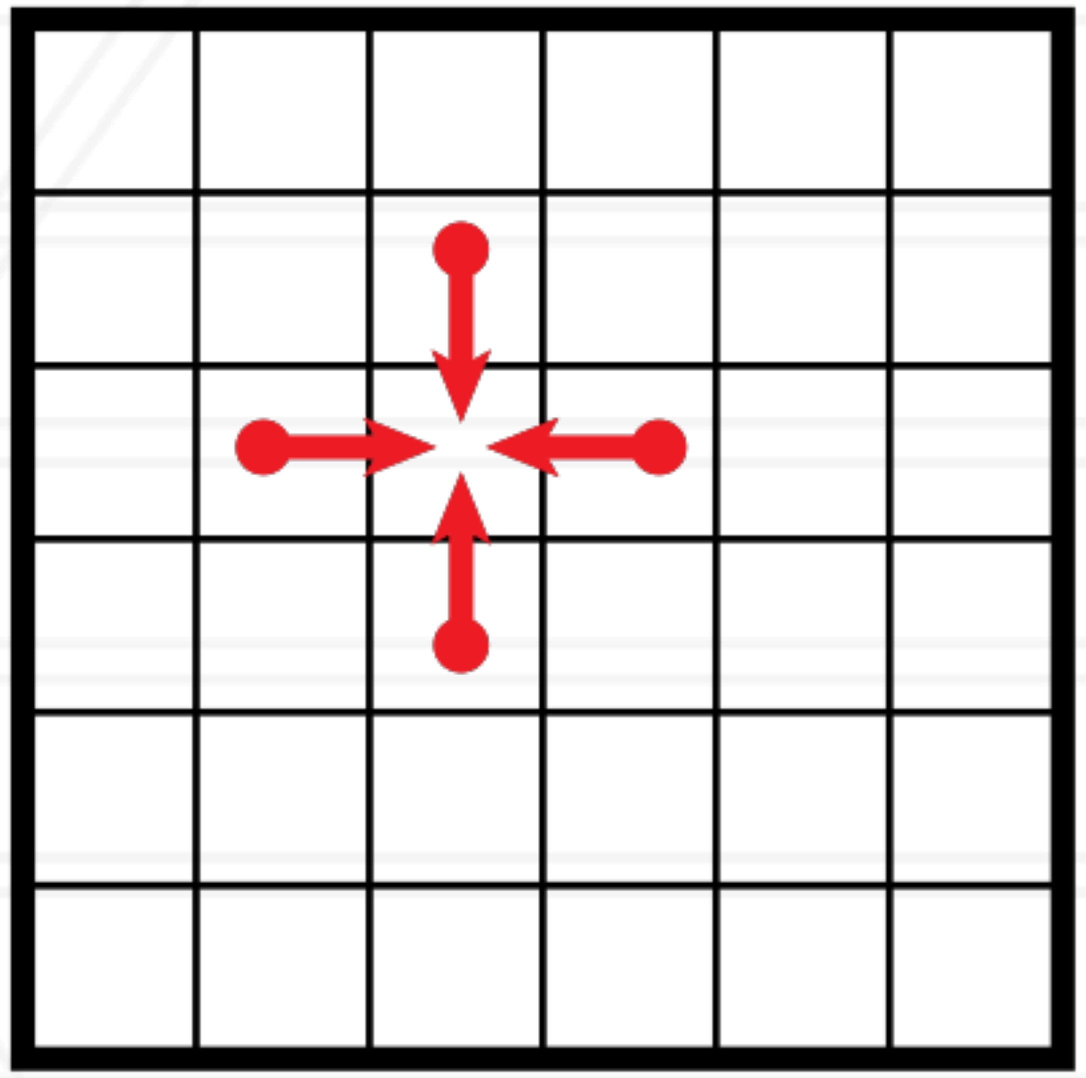
- Molecular Dynamics
- N-body Simulations

Load balance and grain size

- Load balance: try to balance the amount of work (computation) assigned to different threads/ processes
- Grain size: ratio of computation-to-communication
  - Coarse-grained vs. fine-grained

2D Jacobi iteration

- Stencil computation
- Commonly found kernel in computational codes



$$A[i,j] = \frac{A[i,j] + A[i-1,j] + A[i+1,j] + A[i,j-1] + A[i,j+1]}{5}$$

Questions?

UNIVERSITY OF MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu