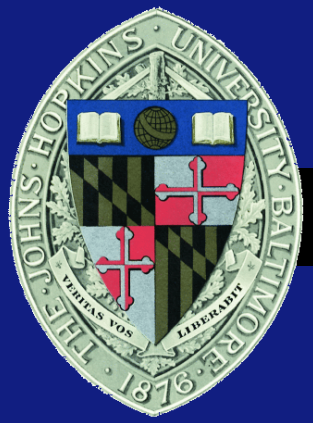
Lecture 4.2
MPI

EN 600.320/420

Instructor: Randal Burns

8 February 2017



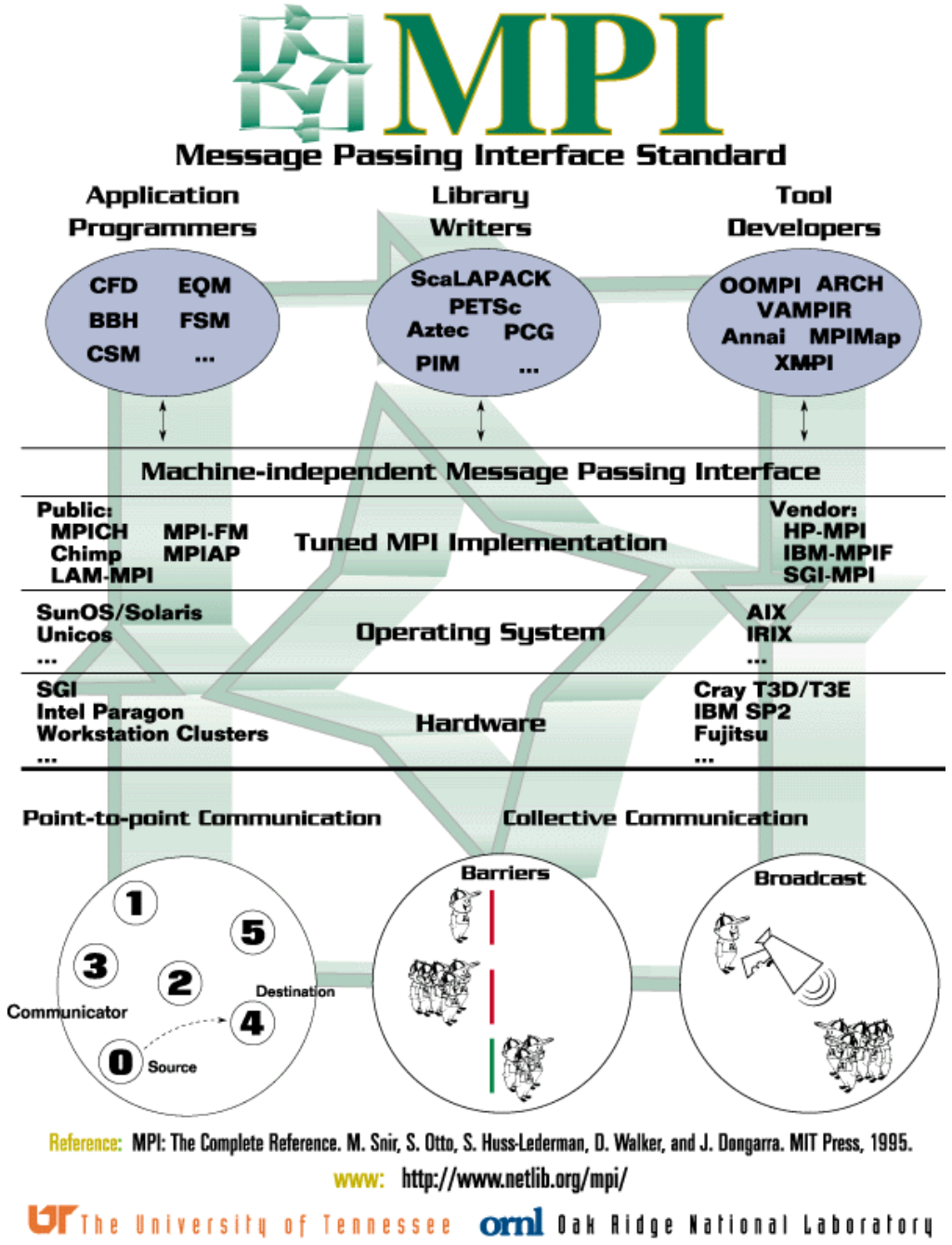
Department of Computer Science, *Johns Hopkins University*

MPI

- MPI = Message Passing Interface
  - Message passing parallelism
  - Cluster computing (no shared memory)
  - Process (not thread oriented)
- Parallelism model
  - SPMD: by definition
  - Also implement: master/worker, loop parallelism
- MPI environment
  - Application programming interface
  - Implemented in libraries
  - Multi-language support (C/C++ and Fortran)

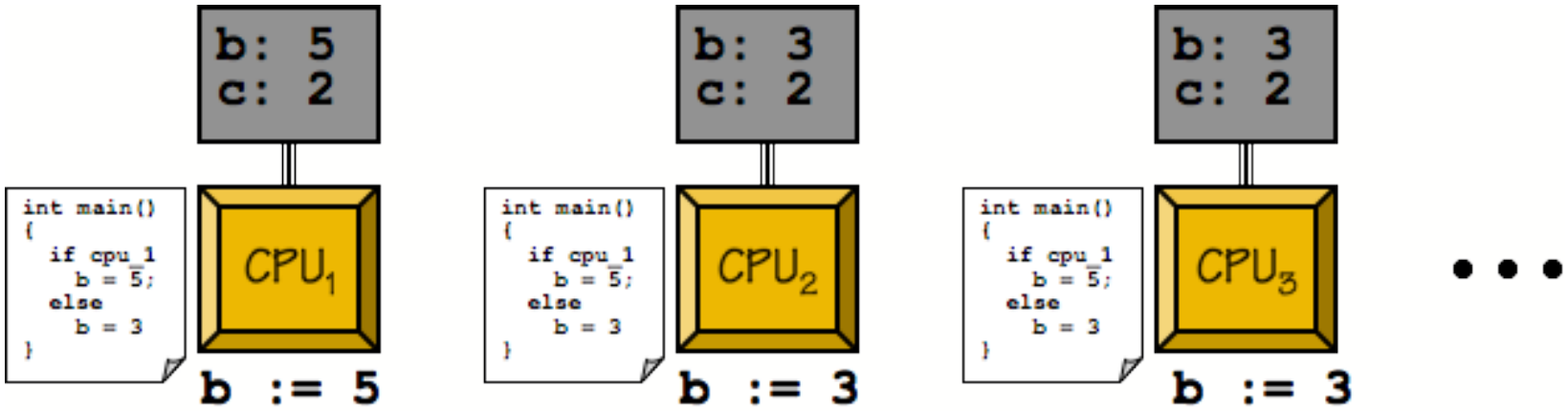
Vision

- Supercomputing Poster 1996



SPMD (Again)

- Single program multiple data
  - From wikipedia "Tasks are split up and run simultaneously on multiple processors with different input in order to obtain results faster. SPMD is the most common style of parallel programming."
  - Asynchronous execution of the same program (unlike SIMD)



https://www.sharcnet.ca/help/index.php/Getting_Started_with_MPI

A Simple MPI Program

- Configure the MPI environment
- Discover yourself
- Take some differentiated activity

See mpimsg.c

- Idioms
  - SPMD: all processes run the same program
  - MPI_Rank: tell yourself apart from other and customize the local processes behaviours
    - Find neighbors, select data region, etc.

Build and Launch Scripts

- Scripts wrap local compiler and link to MPI

- *mpirun* to launch MPI job on the local machine/cluster
  - Launch through scheduler on HPC clusters (do not run on the login node)

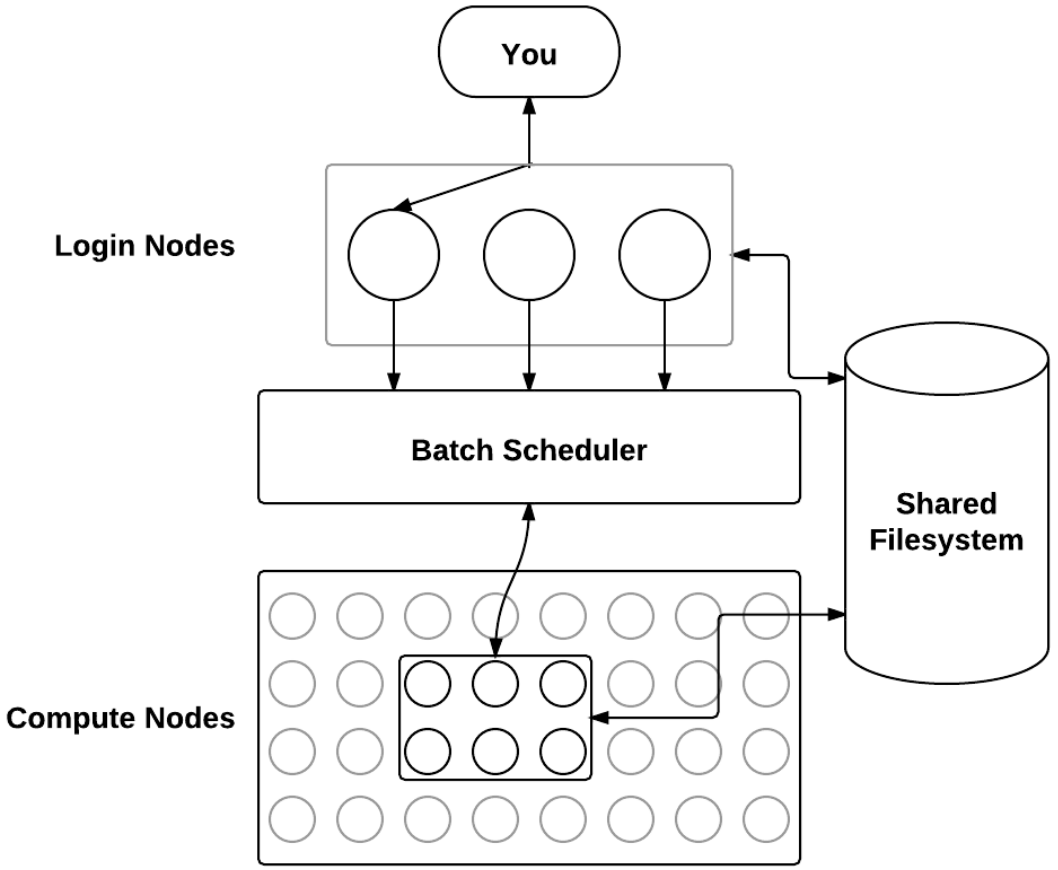
Language	Script Name	Underlying Compiler
C	mpicc	gcc
	mpigcc	gcc
	mpiicc	icc
	mpipgcc	pgcc
C++	mpiCC	g++
	mpig++	g++
	mpiicpc	icpc
	mpipgCC	pgCC
Fortran	mpif77	g77
	mpigfortran	gfortran
	mpiifort	ifort
	mpipgf77	pgf77
	mpipgf90	pgf90

HPC Schedulers

- Maui/Torque
- SLURM
- OGE

- Each with their own submission scripts
  - Not mpirun

https://www.osc.edu/supercomputing/getting-started/hpc-basics



Managing the runtime environment

- Initialize the environment
  - `MPI_Init ( &argc, &argv )`
- Acquire information for process
  - `MPI_Comm_size ( MPI_COMM_WORLD, &num_procs )`
  - `MPI_Comm_rank ( MPI_COMM_WORLD, &ID )`
  - To differentiate process behavior in SMPD
- And cleanup
  - `MPI_Finalize()`
- Some MPI instances leave orphan processes around
  - `MPI_Abort()`
  - Don't rely on this

MPI is just messaging

- And synchronization constructs, which are built on messaging
- And library calls for discovery and configuration

- Computation is done in C/C++/Fortran SPMD program

- I've heard MPI called the "assembly language" of supercomputing
  - Simple primitives
  - Build your own communication protocols, application topologies, parallel execution
  - The opposite end of the design space from MR, Spark