

Distributed Computing and Introduction to High Performance Computing

Imad Kissami¹

¹Mohammed VI Polytechnic University, Benguerir, Morocco

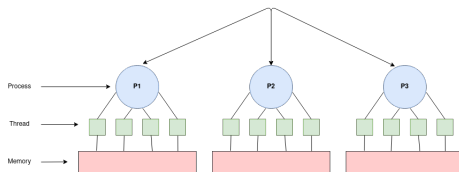


Outline of this lecture

- Distributed Memory Architectures
- The MPI Library
- My first example using MPI

Distributed Memory Architecture

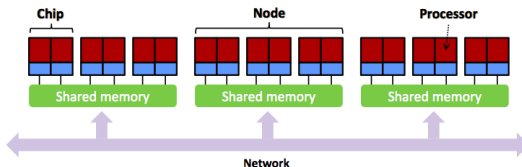
Distributed Memory Multiprocessors



- Each processor has a local memory
 - Physically separated memory address space
- Processors must communicate to access non-local data
 - Message communication (message passing)
 - ➡ Message passing architecture
 - Processor interconnection network
- Parallel applications must be partitioned across
 - Processors: execution units
 - Memory: data partitioning
- Scalable architecture
 - Small incremental cost to add hardware (cost of node)

Distributed Memory Architecture

The Message-Passing Model



- Nodes are complete computer systems
 - Including I/O
- Nodes communicate via interconnection network
 - Standard networks
 - Specialized networks
- Network interfaces
 - Communication integration
- Easier to build

Distributed Memory Architecture

Performance Metrics: Latency and Bandwidth

■ Bandwidth

- Need high bandwidth in communication
- Match limits in network, memory, and processor
- Network interface speed vs. network bisection bandwidth

■ Latency

- Performance affected since processor may have to wait
- Harder to overlap communication and computation
- Overhead to communicate is a problem in many machines

■ Latency hiding

- Increases programming system burden
- Examples: communication/computation overlaps, prefetch

Distributed Memory Architecture

Advantages of Distributed Memory Architectures

- The hardware can be simpler (especially versus NUMA) and is more scalable
- Communication is explicit and simpler to understand
- Explicit communication focuses attention on costly aspect of parallel computation
- Synchronization is naturally associated with sending messages, reducing the possibility for errors introduced by incorrect synchronization
- Easier to use sender-initiated communication, which may have some advantages in performance

Distributed Memory Architecture

MPI (Message Passing Interface)

- A standard message passing specification for the vendors to implement
- Context: distributed memory parallel computers
 - Each processor has its own memory and cannot access the memory of other processors
 - Any data to be shared must be explicitly transmitted from one to another
- Most message passing programs use the single program multiple data (SPMD) model
 - Each processor executes the same set of instructions
 - Parallelization is achieved by letting each processor operation a different piece of data MIMD (Multiple Instructions Multiple Data)

The MPI library

Rank and size

- One can access to the number of processes managed by a given communicator using the `MPI_COMM_SIZE()` function
- You can get the rank of a process, within a communicator, by calling `MPI_COMM_RANK()` function

```
1 from mpi4py import MPI
2
3 #Communicator, Rank and size
4 COMM = MPI.COMM_WORLD
5 SIZE = COMM.Get_size()
6 RANK = COMM.Get_rank()
7
8 print("I am the proccess {RANK} among {SIZE}".format(RANK = RANK, SIZE = SIZE))
```

```
$ mpirun -n 4 python ranksize.py
```

```
I am the proccess 0 among 4
I am the proccess 1 among 4
I am the proccess 2 among 4
I am the proccess 3 among 4
```