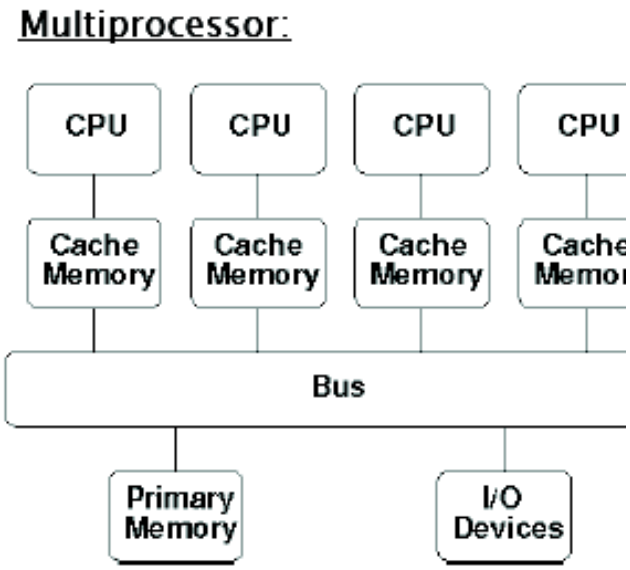
Parallel Programming

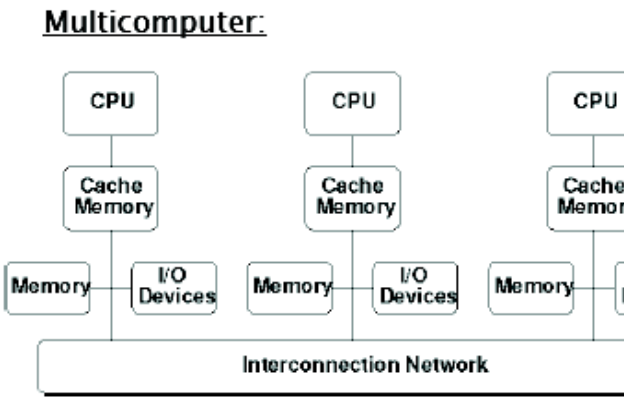
Shared memory:

- multiple cpus are attached to the BUS
- all processors share the same primary memory
- the same memory address on different CPU's refer to the same memory location
- CPU-to-memory connection becomes a bottleneck: shared memory computers cannot scale very well



Distributed memory:

- each processor has its own private memory
- computational tasks can only operate on local data
- infinite available memory through adding nodes
- requires more difficult programming



OpenMP versus MPI

OpenMP (Open Multi-Processing):

- easy to use; loop-level parallelism
- non-loop-level parallelism is more difficult
- limited to shared memory computers
- cannot handle very large problems

MPI(Message Passing Interface):

- require low-level programming; more difficult programming
- scalable cost/size
- can handle very large problems

Distributed memory:

- Each processor can access only the instructions/data stored in its own memory.
- The machine has an interconnection network that supports passing messages between processors.
- A user specifies a number of concurrent processes when program begins.
- Every process executes the same program, though the flow of execution may depend on the processors unique ID number (e.g. "if (my_id == 0) then $\cdots$").
- Each process performs computations on its local variables, then communicates with other processes (repeat), to eventually achieve the computed result.
- In this model, processors pass messages both to send/receive information, and to synchronize with one another.

Communicators and Groups:

- MPI uses objects called communicators and groups to define which collection of processes may communicate with each other.
- Groups are objects that represent groups of processes.
- Most MPI routines require you to specify a communicator as an argument.
- Communicators and groups will be covered in more detail later. For now, simply use MPI_COMM_WORLD whenever a communicator is required - it is the predefined communicator that includes all of your MPI processes.

Rank:

- Within a communicator, every process has its own unique, integer identifier assigned by the system when the process initializes. A rank is sometimes also called a "task ID". Ranks are contiguous and begin at zero
- Used by the programmer to specify the source and destination of messages. Often used conditionally by the application to control program execution (if rank=0 do this / if rank=1 do that).

Error Handling:

- Most MPI routines include a return/error code parameter.