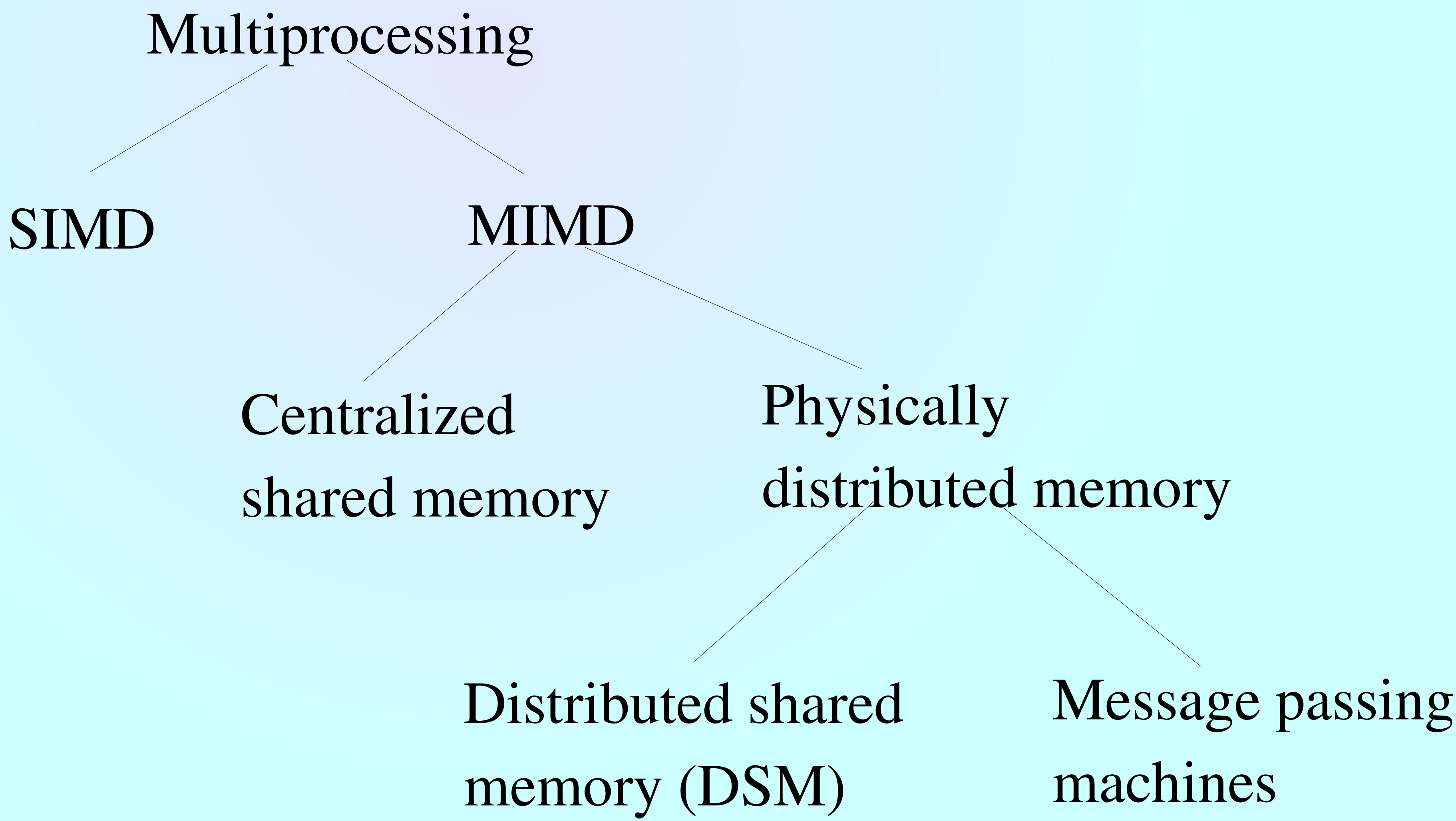
LECTURE - 24

Topic for Today's Lecture

- Multiprocessing
- Parallel applications
- Cache coherence
-
- ***Scribe for today?***

Multiprocessing: Classification



DSM vs. Message Passing

Shared Memory

Well understood mechanisms for programming

Program independent of communication pattern

Low overhead for communicating small items

Hardware controlled caching

Message Passing

Hardware simplicity

Communication is explicit – forces programmer to pay attention to what is expensive

Achieving the Desired Communication Model

Message Passing on top of Shared Memory

- Considerable easier
- Difficulty arises in dealing with arbitrary message lengths

Shared Memory on top of Message Passing

- Harder since every load/store has to be faked
- Every memory reference may involve OS
- One promising direction: use of VM to share objects at page level: shared VM

Challenges in Parallel Processing

- Limited parallelism available in programs
  - 90% parallelizable ==> max speed possible?
  - Exception: super-linear speedup
    - Increased memory/cache available
    - Usually not very great however
- Large latency of communication
  - 50-10000 clock cycles
  - 0.5% instructions access remote memory ==> what is the increase in CPI?

Addressing the Challenges

- Limited parallelism
  - Tackled mainly by redesigning the algorithm or software
- Avoiding large latency
  - Hardware mechanism: caching
  - Software mechanism: restructure to make more accesses local

Some Example Applications

- Two classes
  - Parallel programs or program kernels
  - Multi-programmed OS
- Spatial and temporal data access patterns are important
- Computation to communication ratio is important

Parallel Application Kernels

- The FFT kernel
  - Used in spectral methods
  - Data represented as array
  - Computation involves
    - 1D FFT on each row
    - Transpose
    - 1D FFT on each row again
  - Each processor gets a few rows of data
  - Main communication step is the transpose (all to all communication)

Parallel Application Kernels (continued)

- The LU kernel
  - LU factorization of a matrix
  - Blocking is used
  - Computation (dense matrix multiply) is performed by processor which owns the destination block
  - Communication happens at regular intervals

Parallel Applications

- Barnes application
  - N-body problem
  - Octree representation
  - Each processor is allocated a subtree
  - Tree expansion as required (communication in this process)

Parallel Applications (continued)

- Ocean application
  - Influence of eddy and boundary currents on ocean flows
  - Involves solving PDEs
  - Ocean divided into hierarchy of grids (finer grid for more accuracy)
  - Each processor gets a set of grids
  - Communication to exchange boundary conditions, at each step of the process

Computation to Communication Ratios

Application	Computation scaling	Communication scaling	Scaling of computation to communication
FFT	nlogn/p	n/p	Logn
LU	n/p	sqrt(n/p)	sqrt(n/p)
Barnes	nlogn/p	logn*sqrt(n/p)	sqrt(n/p)
Ocean	n/p	sqrt(n/p)	sqrt(n/p)

Multiprogrammed OS workload

- Workload used here is:
  - Two independent copies of the compilation of the Andrew benchmark
  - Three steps:
    - Compilation: compute intensive
    - Installing object files in a library: I/O intensive
    - Removing the object files: I/O intensive