SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 431 Distributed Systems

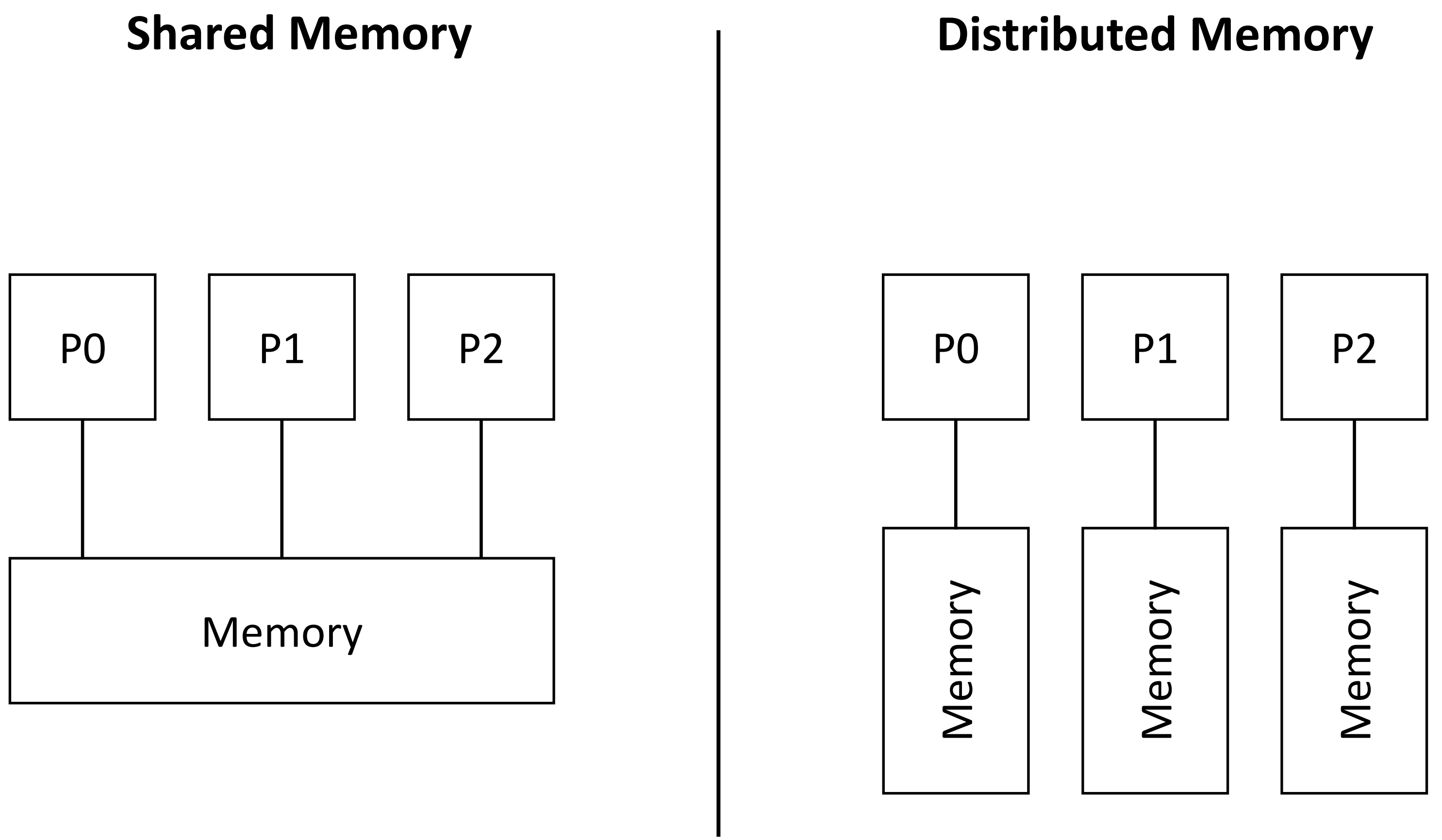
Fall 2019

Parallel Computing

https://www.cs.sfu.ca/~keval/teaching/cmpt431/fall19/

Instructor: Keval Vora

Memory Architectures



Shared Memory Architectures

- Processors access memory as a global address space
- Memory updates by one processor are visible to others

- Easier to program with global address space
- Typically fast memory access (supported by hardware)

- Difficult to scale
- Adding CPUs (geometrically) increases traffic
- Need to synchronization memory accesses

Shared Memory: Race Condition

```
withdraw(int account_id, int amount) {
   balance = get_balance(account_id);
   if(balance >= amount) {
      set_balance(account_id,
                  balance - amount);
      eject(amount);
  }
}
```

Distributed Memory Architectures

- Nodes (processors + memories) connected via communication network
- Access to another processor's data via communication protocols (e.g., send-receive calls)

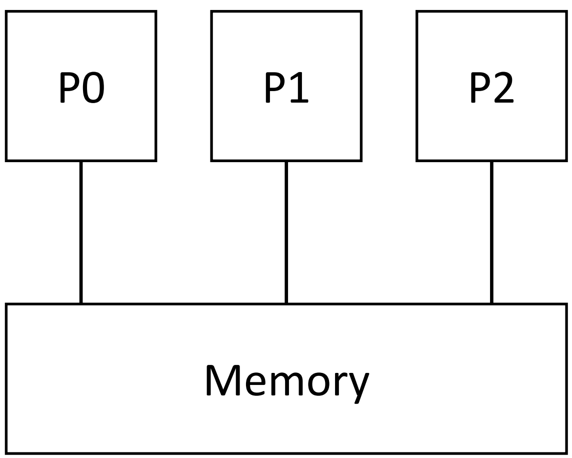
- Scalable (both processor and memory)
- Local access is fast (no coordination required)
- Cost effective

- Difficult to program
- Communication/Synchronization is difficult to manage

Shared Memory: UMA & NUMA

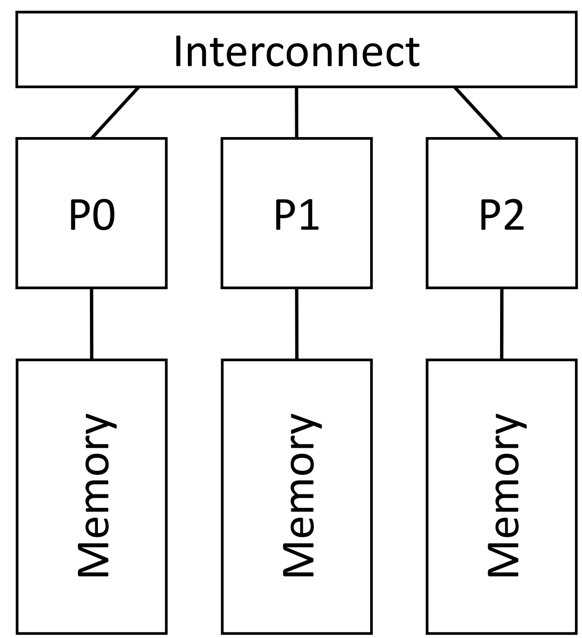
Uniform Memory Access (UMA)

- Typically Symmetric Multiprocessors (SMP)
- Equal memory access time



Non-Uniform Memory Access (NUMA)

- Typically multiple SMPs that can access each other's memories
- Memory access times are not equal



Communication Models

- Shared Memory
  - Tasks share a common address space they access asynchronously
  - Mutexes/Locks used to control access to shared memory
  - Compiler translates variables into global memory addresses

- Message Passing
  - No shared address space
  - Tasks use their own local memories
  - Data transfer often requires coordination: receive matching send

Parallel Programming Models

- Data parallel
  - Parallel operations over a collection of data items
  - Tasks collectively work on the collection and each task works on a different partition (subset) of the collection
  - E.g., increment all elements in an array by 1 (map operation)
- Task Parallel
  - Tasks defined based on the operations to be performed
  - Each task performs an operation which is different from that performed by other task
  - Operations should be *safe* to be concurrently executable with other operations
  - May or may not operate on the same collection of data items
  - E.g., pipelining

Parallel Programming

- Decompose problem into sub-problems
- Map sub-problems to concurrent tasks
- Ensure dependencies are correctly satisfied
  - E.g., Op1 in thread 1 should happen before op2 in thread 2
  - Coordination via synchronization/communication

- Overall execution is a mix of parallel and sequential executions

Measuring Performance

- How fast does a job compute?
  - Elapsed time (Latency)
  - compute + communicate + synchronize
- How many jobs complete in a given time?
  - Throughput
- How well does the system scale?
  - Increasing *processors* (compute nodes)
  - Increasing problem size (constant work per processor)

Weak v/s strong scaling: https://en.wikipedia.org/wiki/Scalability#Weak_versus_strong_scaling

Performance Metrics

- Speedup, $S_\mathrm{p} = \dfrac{\text{Execution time using 1 processor system } (\mathrm{T_1})}{\text{Execution time using p processor system } (\mathrm{Tp})}$

- Efficiency $= \dfrac{\mathrm{S_p}}{\mathrm{p}}$

- Cost, $C_p = p \times Tp$ Optimal if $C_p = T_1$

Amdahl’s Law

- f = fraction of problem that is sequential
  - (1 – f) = fraction of problem that is parallel

- Fastest parallel time, $T_p = T_1 \times (f + \frac{1-f}{p})$

- Speedup with p processors, $S_p = \frac{1}{f + \frac{(1-f)}{p}}$

Amdahl's Law

- Gives an upper bound on speedup
- Only fraction (1 - f) is shared by p processors
  - Increasing p cannot speed up fraction f
- Speedup with p processors, $S_{\mathrm{p}} = \frac{1}{\mathrm{f} + \frac{(1-\mathrm{f})}{\mathrm{p}}}$
- Upper bound on speedup at $S_{\infty} = \frac{1}{\mathrm{f}}$
- Example: f = 2%, S_{∞} = 1 / 0.02 = 50

Amdahl's Law

$$S_p = \frac{1}{f + \frac{(1-f)}{p}}$$