Chapter 8: Parallel Processors

Ngo Lam Trung, Pham Ngoc Hung, Hoang Van Hiep

[with materials from *Computer Organization and Design,* Patterson & Hennessy, *Computer Organization and Architecture*, William Stallings, and *M.J. Irwin's presentation*, PSU 2008]

Introduction

- Overall goal: increasing performance
  - For a large software
  - For a large number of small individual software
  - With energy efficiency
- Approaches in previous chapters?
  - Pipeline
  - Super scaler
  - Multi-threaded

    → increase performance of a single CPU core

Difficulty in parallelism

- The difficulty with parallelism is not the hardware!
- It is difficult to write software that uses multiple processors to complete one task faster, and the problem gets worse as the number of processors increases.
  - Conventional algorithm have been designed as sequential
  - Divide the job to multiple processors may lead to large communication overhead

Multiprocessing

- Pollack:
  - “Performance is roughly proportional to square root of increase in complexity”
  - double the logic in a processor core, then it delivers only 40% more performance
- The use of multiple cores has the potential to provide near-linear performance improvement with the increase in the number of cores

Multiprocessing

- Replacing large inefficient processors with many smaller, efficient processors

➔better performance per joule

➔Improved energy efficiency joins scalable performance

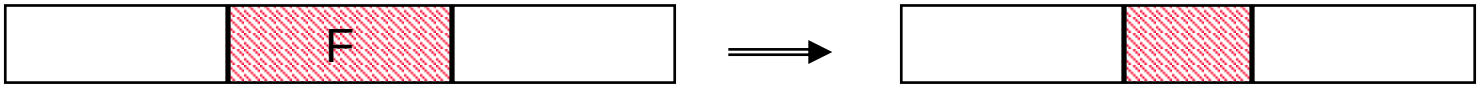
- ***Task-level/Process-level parallelism***: multiple independent tasks running on multiple processors
- ***Parallel processing program***: single program running on multiple processors simultaneously

Encountering Amdahl's Law

- Speedup due to enhancement E is

$$\text{Speedup w/ E} = \frac{\text{Exec time w/o E}}{\text{Exec time w/ E}}$$

- Suppose that enhancement E accelerates a fraction F (F <1) of the task by a factor S (S>1) and the remainder of the task is unaffected



ExTime w/ E = ExTime w/o E ×

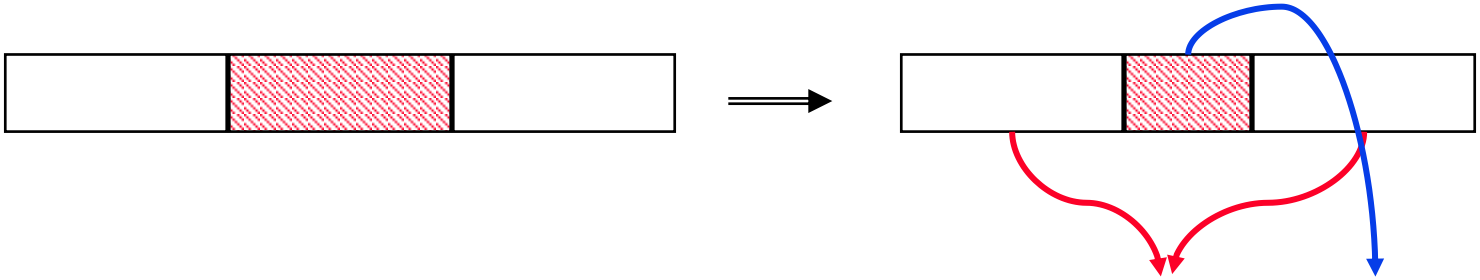
Speedup w/ E =

Encountering Amdahl's Law

- Speedup due to enhancement E is

$$\text{Speedup w/ E} = \frac{\text{Exec time w/o E}}{\text{Exec time w/ E}}$$

- Suppose that enhancement E accelerates a fraction F (F <1) of the task by a factor S (S>1) and the remainder of the task is unaffected



$$\text{ExTime w/ E} = \text{ExTime w/o E} \times ((1\text{-}F) + F/S)$$

$$\text{Speedup w/ E} = 1 / ((1\text{-}F) + F/S)$$

Example 1: Amdahl's Law

Speedup w/ E =

- Consider an enhancement which runs 20 times faster but which is only usable 25% of the time.

  Speedup w/ E =

- What if its usable only 15% of the time?

  Speedup w/ E =

- Amdahl's Law tells us that to achieve linear speedup with 100 processors, none of the original computation can be scalar!

- To get a speedup of 90 from 100 processors, the percentage of the original program that could be scalar would have to be 0.1% or less

  Speedup w/ E =

Example 1: Amdahl's Law

Speedup w/ E = 1 / ((1-F) + F/S)

- Consider an enhancement which runs 20 times faster but which is only usable 25% of the time.

  Speedup w/ E = 1/(.75 + .25/20) = 1.31

- What if its usable only 15% of the time?

  Speedup w/ E = 1/(.85 + .15/20) = 1.17

- To achieve linear speedup with 100 processors, none of the original computation can be scalar!

- To get a speedup of 90 from 100 processors, the percentage of the original program that could be scalar would have to be 0.1% or less

  Speedup w/ E = 1/(.001 + .999/100) = 90.99

Example 2: Amdahl's Law

Speedup w/ E = 1 / ((1-F) + F/S)

- Consider summing 10 scalar variables and two 10x10 matrices (matrix sum) on 10 processors

  Speedup w/ E =

- What if there are 100 processors ?

  Speedup w/ E =

- What if the matrices are 100x100 (or 10,010 adds in total) on 10 processors?

  Speedup w/ E =

- What if the matrices are 100x100 and there are 100 processors?

  Speedup w/ E =

Example 2: Amdahl's Law

Speedup w/ E = 1 / ((1-F) + F/S)

- Consider summing 10 scalar variables and two 10x10 matrices (matrix sum) on 10 processors

  Speedup w/ E = 1/(.091 + .909/10) = 1/0.1819 = 5.5

- What if there are 100 processors ?

  Speedup w/ E = 1/(.091 + .909/100) = 1/0.10009 = 10.0

- What if the matrices are 100x100 (or 10,010 adds in total) on 10 processors?

  Speedup w/ E = 1/(.001 + .999/10) = 1/0.1009 = 9.9

- What if the matrices are 100x100 and there are 100 processors?

  Speedup w/ E = 1/(.001 + .999/100) = 1/0.01099 = 91

Scaling

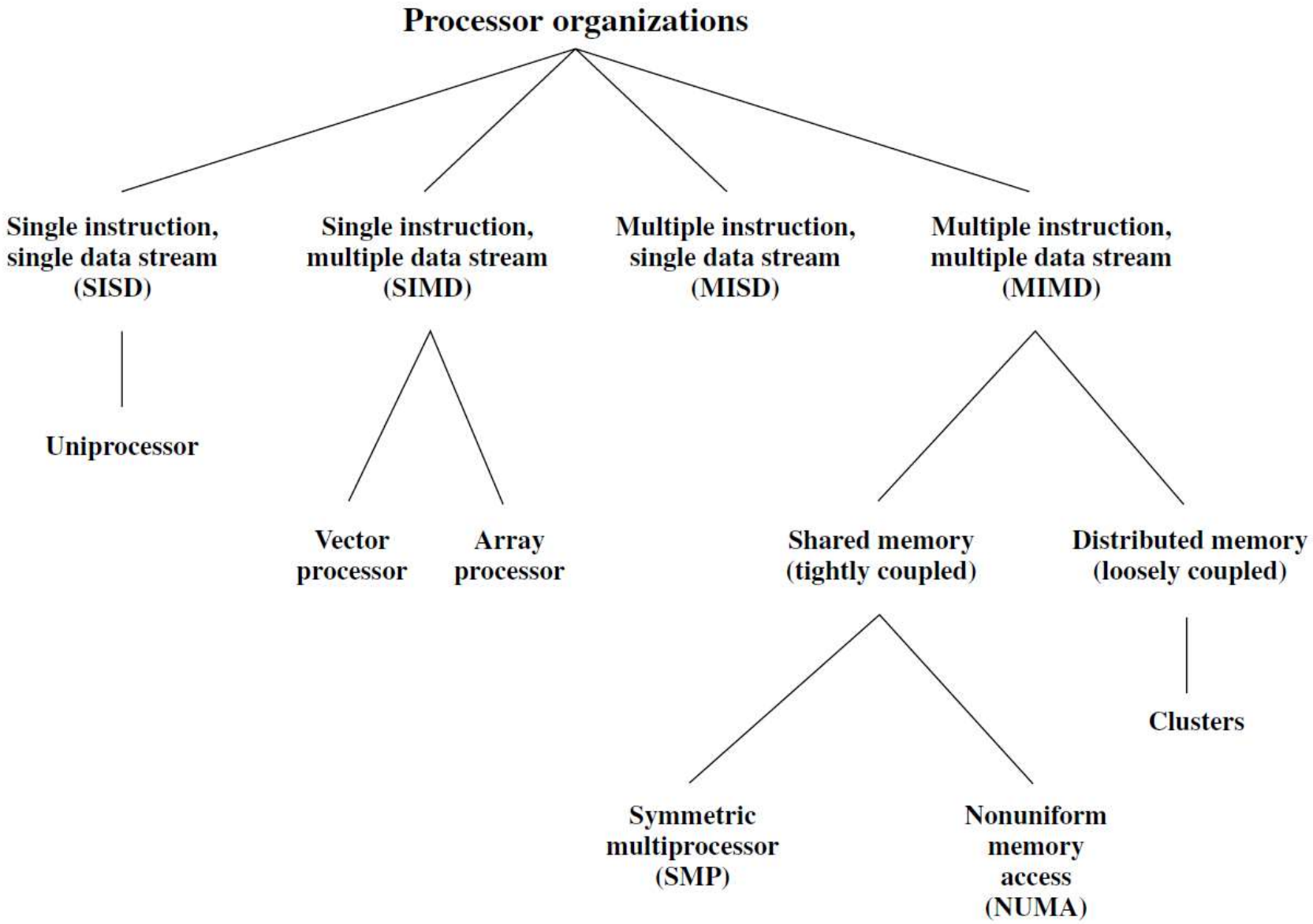
- To get good speedup on a multiprocessor while keeping the problem size fixed is harder than getting good speedup by increasing the size of the problem.
  - Strong scaling – when speedup can be achieved on a multiprocessor without increasing the size of the problem
  - Weak scaling – when speedup is achieved on a multiprocessor by increasing the size of the problem proportionally to the increase in the number of processors

- Load balancing is another important factor.

Multiprocessor Key Questions

- Q1 – How do they share data?
- Q2 – How do they coordinate?
- Q3 – How scalable is the architecture? How many processors can be supported?

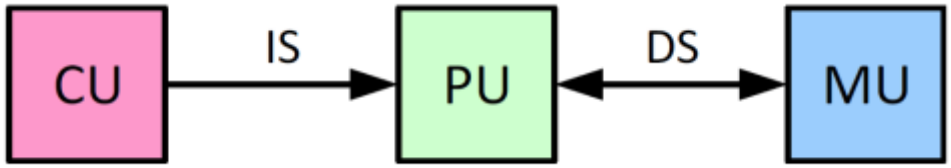
Types of Parallel Processor Systems



Types of Parallel Processor Systems

- Single instruction, single data (**SISD**) stream: A single processor executes a single instruction stream to operate on data stored in a single memory.
- Single instruction, multiple data (**SIMD**) stream: A single machine instruction controls the simultaneous execution of a number of processing elements
- Multiple instruction, single data (**MISD**) stream: not implemented
- Multiple instruction, multiple data (**MIMD**) stream: A set of processors simultaneously execute different instruction sequences on different data sets.

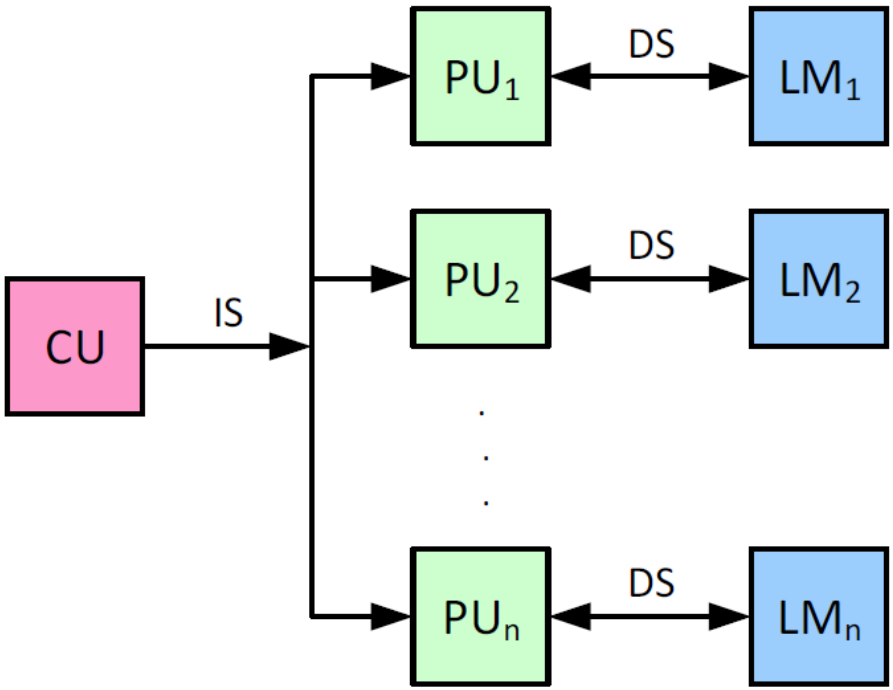
SISD



- CU: Control Unit
- PU: Processing Unit
- MU: Memory Unit
- Sequential execution
- Data stored in a single main memory
- ➔ Uniprocessor computer

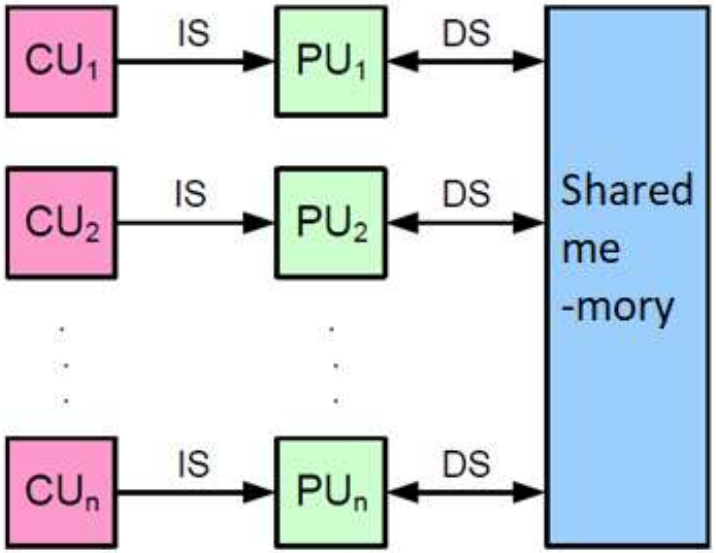
SIMD

- 1 instruction stream
- Multiple processing units
- Each PC processes data from a separate memory
- All PUs execute the same instruction stream from CU
- Example: GPU

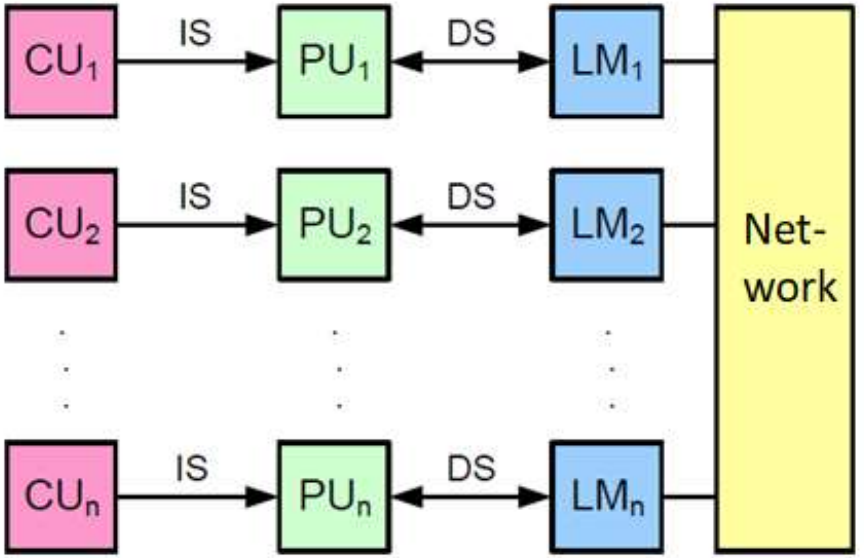


MIMD

- Multiple instruction, multiple data
- Require multiple CUs and PUs
- Shared or distributed memory



MIMD with shared memory

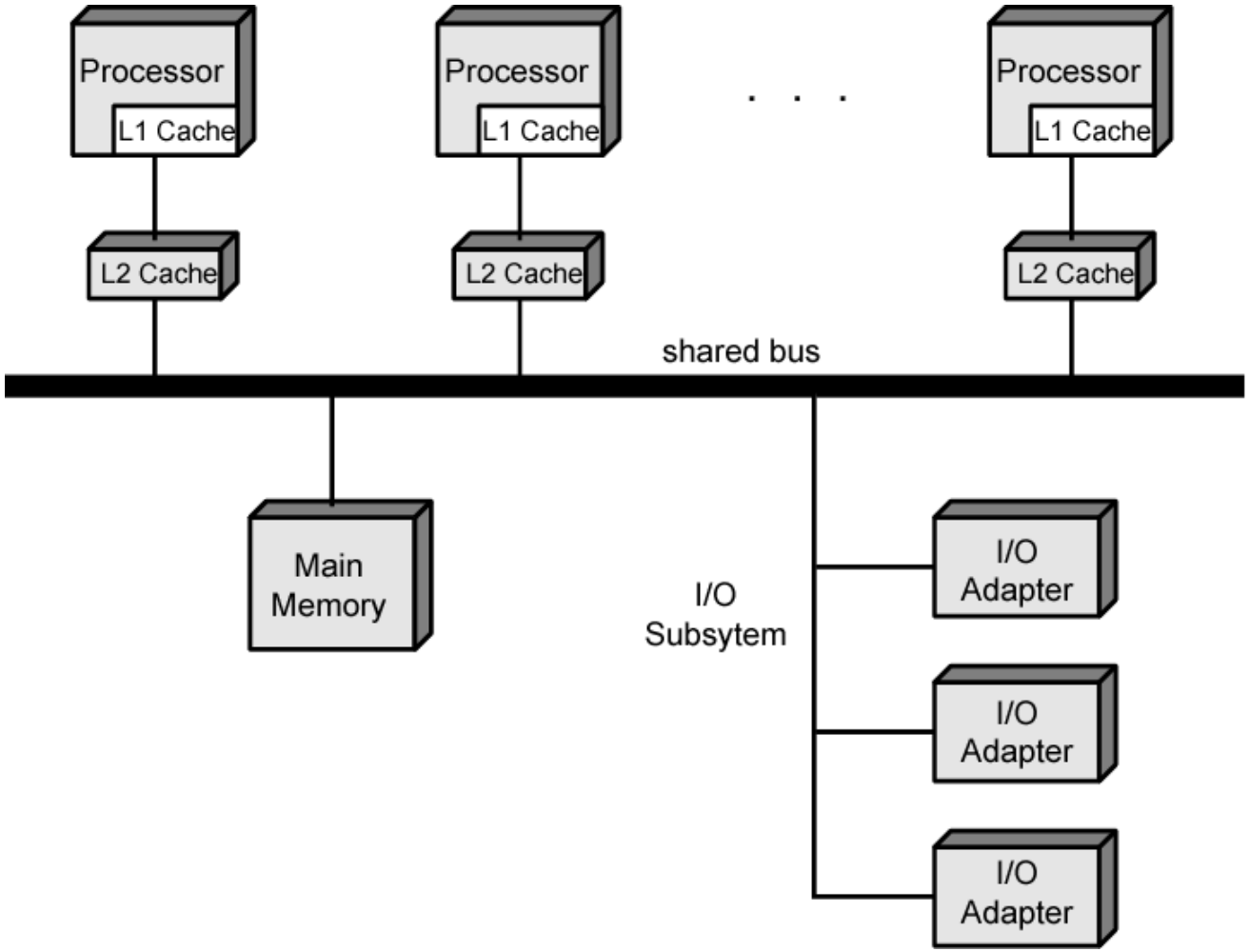


MIMD with distributed memory

Symmetric Multiprocessor (SMP)

- Two or more similar processors of comparable capability
- These processors share the same main memory and I/O facilities
- All processors share access to I/O devices
- All processors can perform the same functions (*symmetric*)
- The system is controlled by an integrated operating system
  - Provides interaction between processors and system resources

Symmetric Multiprocessor Organization

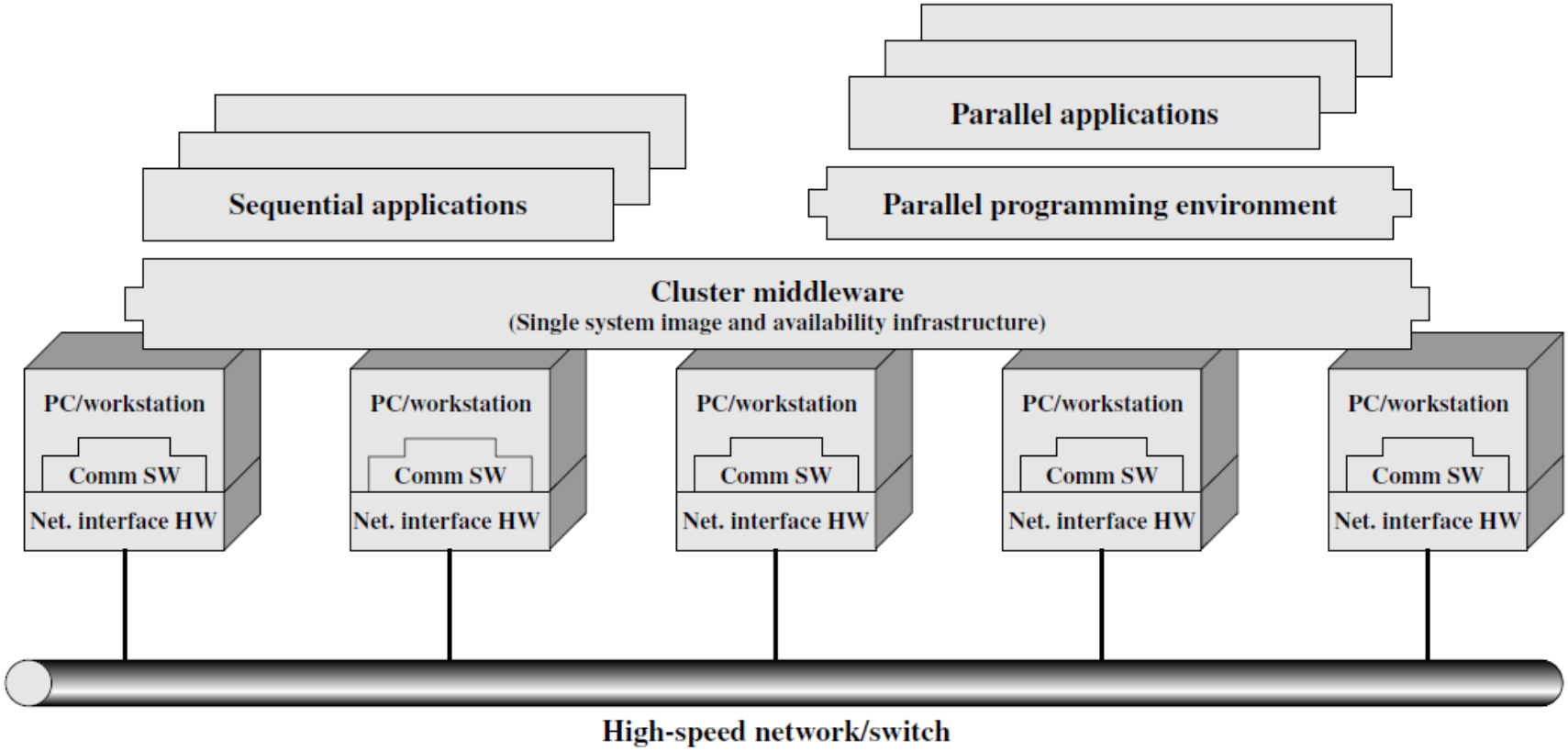


Cluster

- Group of interconnected, whole computers working together as a unified computing resource. Each computer in a cluster is typically referred to as a node.

- Absolute scalability
- Incremental scalability
- High availability
- Superior price/performance

Cluster Computer Architecture



SMP vs cluster

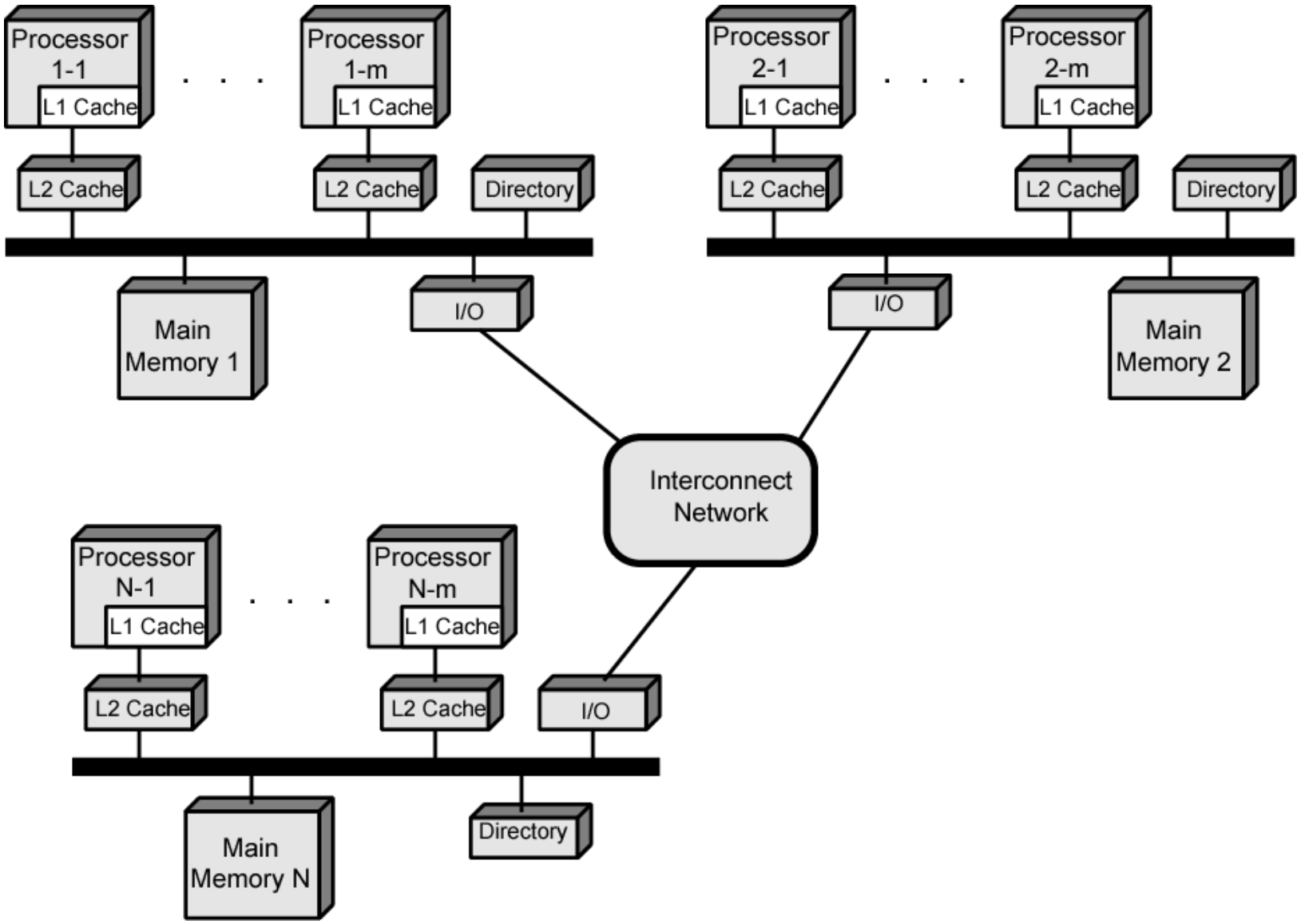
- Both are high performance computer architecture
- SMP
  - Easy to use and maintenance
  - Closer to uniprocessor system
  - Small size and low power consumption
- Cluster
  - High computing capability
  - Scalability
  - Hight dependability and availability

SMP vs cluster

- SMP
  - Limited capability.
- Cluster
  - Separated memory on each node
  - Complicated software

➔ combining advantages of SMP and cluster: NUMA

Cache-coherence NUMA



The end