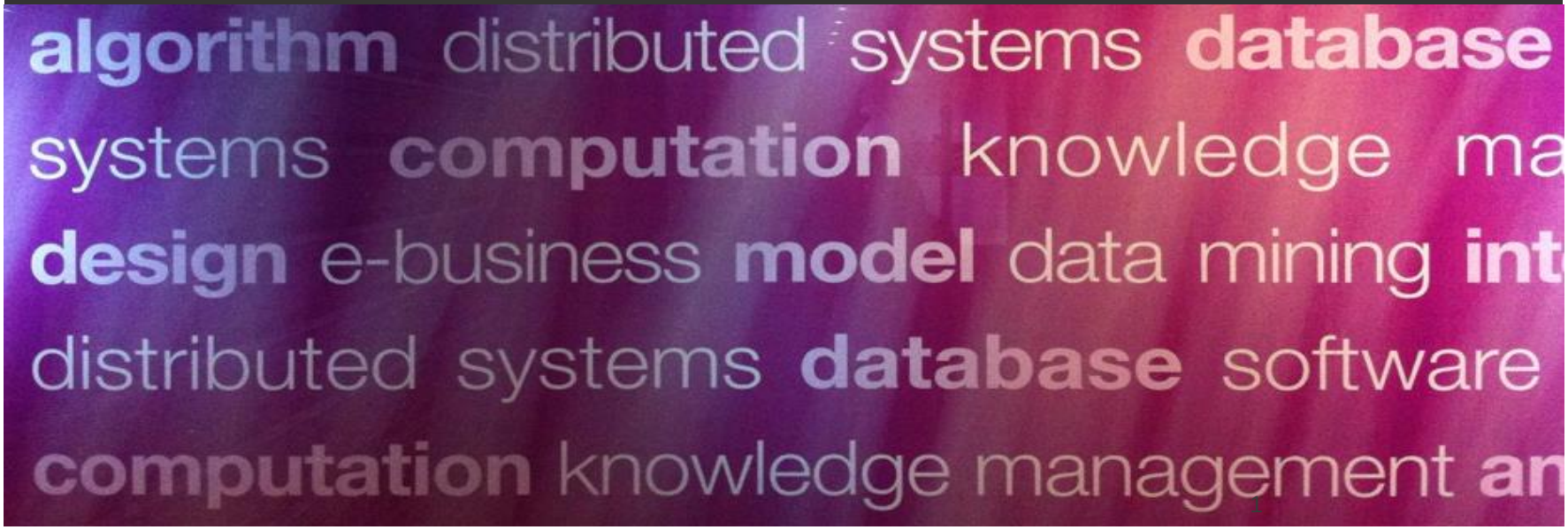
MONASH University

Information Technology

FIT3143 - LECTURE WEEK 4a

INTRODUCTION TO PARALLEL COMPUTING IN DISTRIBUTED MEMORY



Overview

1. Distributed memory parallelism via distributed computing
2. Parallel vs. distributed vs. asynchronous computing

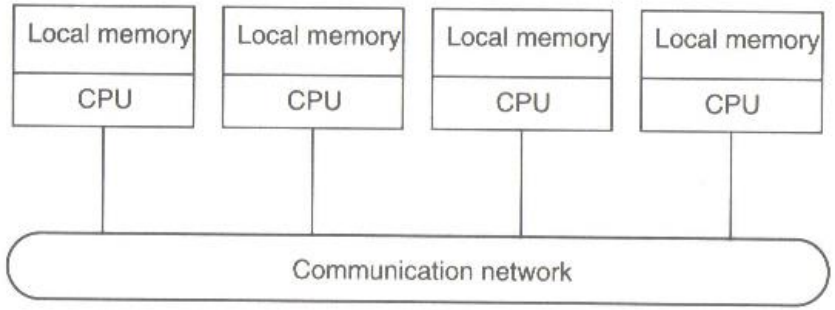
Associated learning outcomes

- Explain the fundamental principles of parallel computing architectures and algorithms (LO1)
- Analyze and evaluate the performance of parallel algorithms (LO4)

1. Distributed computing

DS Introduction

- Loosely coupled systems
  - Processors do not share memory
  - Each processor has its own local memory



Definition of Distributed Systems

- *"A distributed system is a collection of independent computers that appears to its users as a single coherent system."*

- The definition has several important aspects
  - Autonomous components
  - Users (whether people or program) think they are dealing with a single system

- A distributed system is a system in which components located at networked computers communicate and coordinate their actions only by passing messages.

Evolution of Distribution System

- Two advances as the reason for spread of distributed systems

1. Powerful micro-processor:
   - 8-bit, 16-bit, 32-bit, 64-bit
   - x86 family, 68k family, CRAY, SPARC, dual core, multi-core
   - Clock rate: up to 4Ghz

2. Computer network:

   Local Area Network (LAN), Wide Area Network (WAN), MAN, Wireless Network type: Ethernet, Token-bus, Token-ring, Fiber Distributed Data Interface (FDDI), Asynchronous Transfer Mode (ATM), Fast-Ethernet, Gigabit Ethernet, Fiber Channel, Wavelength-Division Multiplex (WDM)

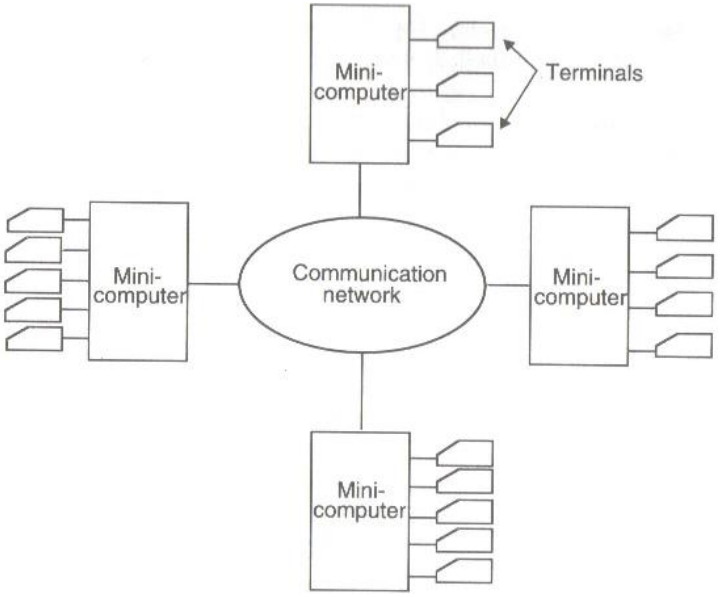
   Transfer rate: 64 kbps up to 1Tbps

Distributed computing system models

- Various models are used for building distributed computing systems.
- These models can be broadly classified into five categories-
  - Minicomputer model
  - Workstation model
  - Workstation-server model
  - Processor-pool model
  - Hybrid model

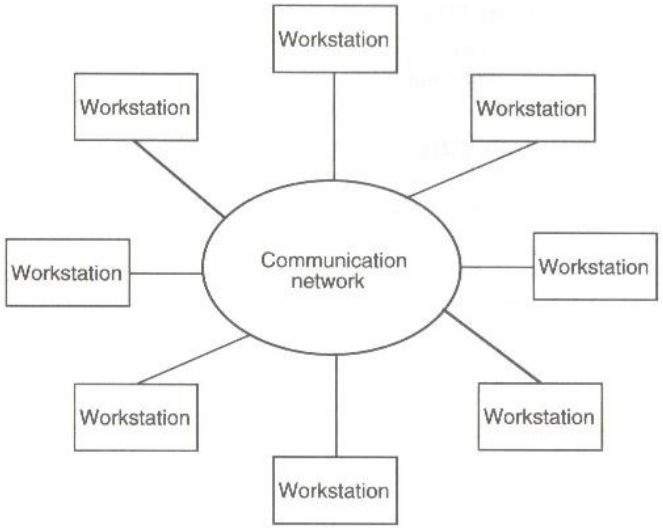
Distributed computing system models

1. Minicomputer model:
   - simple extension of centralized time-sharing systems
   - few minicomputers interconnected by communication network
   - each minicomputer has multiple users simultaneously logged on to it
   - this model may be used when when resource sharing with remote users is desired
   - Example: the early ARPAnet



Distributed computing system models

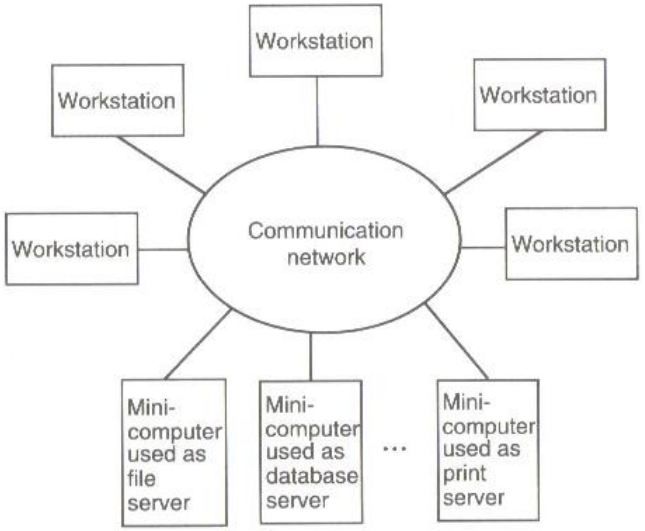
2. Workstation model:
   - several workstations interconnected by communication network
   - basic idea is to share processor power
   - user logs onto home workstation and submits jobs for execution, system might transfer one or more processed to other workstations
   - issues must be resolved –
     - how to find an idle workstation
     - how to transfer
     - what happens to a remote process
   - Examples- Sprite system, Xerox PARC



Distributed computing system models

3. Workstation-server model

- It consists of a few minicomputers and several workstations (diskless or diskful)
- Minicomputers are used for providing services
- For higher reliability and better scalability multiple servers may be used for a purpose.
- Compare this model with workstation model .
- Example- The V-System



Distributed computing system models

4. Processor-pool model

- Base observation –

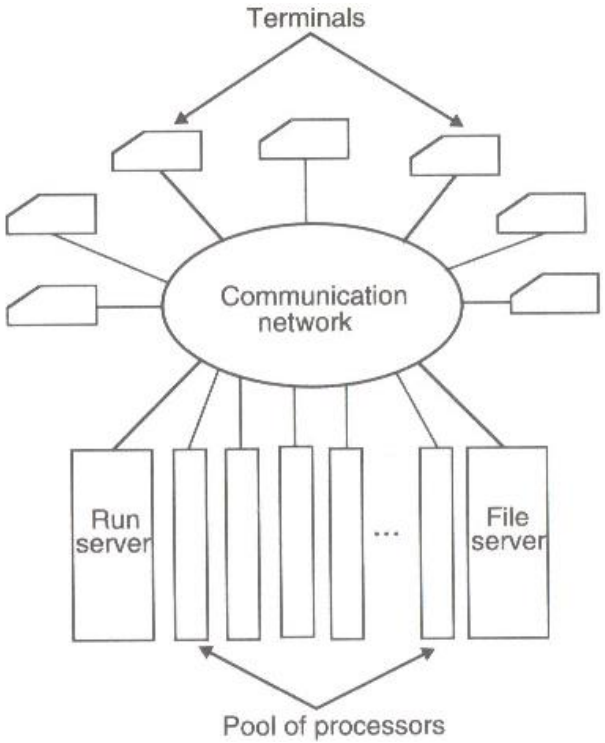
  sometimes user need NO computing power, but once in a while he needs very large amount of computing power for a short period of time

- Run server manages and allocates the processors to different users

- No concept of a home machine, i.e., a user does not log onto a particular machine but to the system as a whole.

- Offers better utilization of processing power compared to other models.

- Example: Amoeba, Plan9, Cambridge DCS.



Distributed computing system models

5. Hybrid Model

- To combine the advantages of both workstation-server model and processor-pool model a hybrid model may be used
- It is based on the workstation-server model but with addition of a pool of processors
- Expensive!!

Distribution Model

- There are several distribution models for accessing distributed resources and executing distributed applications as follows.

- File Model - Resources are modeled as files. Remote resources are accessible simply by accessing files.

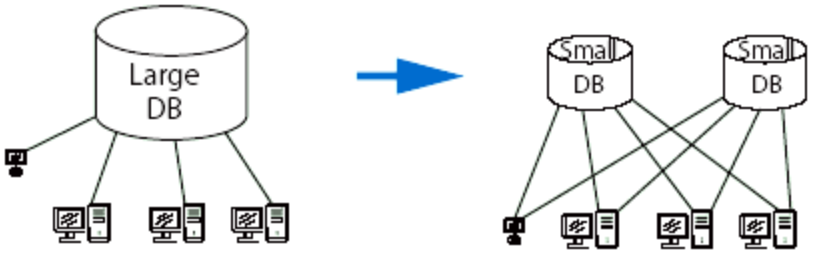
- Remote Procedure Call Model - Resource accesses are modeled as function calls. Remote resources can be accessed by calling functions.

- Distributed Object Model - Resources are modeled as objects which are a set of data and functions to be performed on the data. Remote resources are accessible simply by accessing an object.

Advantages of Distributed Systems

Economics: Microprocessors offer a better price / performance than mainframes.

Speed: A distributed system may have more total computing power than a mainframe. E.g., one large database may be split into many small databases. In that way, we may improve response time.



Inherent distribution: Some application like banking, inventory systems involve spatially separated machines.

Advantages of Distributed Systems

Reliability: If 5% of the machines are downed, the system as a whole can still survive with a 5% degradation of performance.

Incremental growth: Computing power can be added in small increments

Sharing: Allow many users access to a common database and peripherals.

Communication: Make human-to-human communication easier.

Effective Resource Utilization: Spread the workload over the available machines in a cost-effective way.

Disadvantages of Distributed Systems

- Software: It is harder to develop distributed software than centralized one.
- Networking: The network can saturate or cause other problems.
- Security: Easy access also applies to secret data.

Challenges in Distributed Systems

- Heterogeneity - Within a distributed system, we have variety in networks, computer hardware, operating systems, programming languages, etc.
- Openness - New services are added to distributed systems. To do that, specifications of components, or at least the interfaces to the components, must be published.
- Transparency - One distributed system looks like a single computer by concealing distribution.
- Performance - One of the objectives of distributed systems is achieving high performance out of cheap computers.

Challenges in Distributed Systems

- Scalability - A distributed system may include thousands of computers. Whether the system works is the question in that large scale.
- Failure Handling - One distributed system is composed of many components. That results in high probability of having failure in the system.
- Security - Because many stake-holders are involved in a distributed system, the interaction must be authenticated, the data must be concealed from unauthorized users, and so on.
- Concurrency - Many programs run simultaneously in a system, and they share resources. They should not interfere with each other

2. Parallel vs. distributed vs. asynchronous computing

Parallel vs. Distributed computing

S.NO	PARALLEL COMPUTING	DISTRIBUTED COMPUTING
1.	Many operations are performed simultaneously	System components are located at different locations
2.	Single computer is required	Uses multiple computers
3.	Multiple processors perform multiple operations	Multiple computers perform multiple operations
4.	It may have shared or distributed memory	It have only distributed memory
5.	Processors communicate with each other through bus	Computer communicate with each other through message passing.
6.	Improves the system performance	Improves system scalability, fault tolerance and resource sharing capabilities

Source: https://www.geeksforgeeks.org/difference-between-parallel-computing-and-distributed-computing/

Parallel vs. Asynchronous computing

- Both parallel and asynchronous programming models perform similar tasks and functions in modern programming languages.
- However, these models have conceptual differences.
- Asynchronous programming is used to avoid “blocking” within a software application. Example, database queries or network connections are best implemented using asynchronous programming.
- An asynchronous call spins off a thread (e.g. an I/O thread) to complete the target task.
- An asynchronous calls prevents the user interface from the “freeze” effect.

Parallel vs. Asynchronous computing

- As for parallel programming, the main task is segmented into smaller tasks, to be executed by a set of threads within the reach of a common variable pool.
- Parallel programming can also prevent user interface “freeze” effect when running computational expensive tasks on a CPU.
- Key difference: In an asynchronous call, control over threads is limited and is system dependent. In parallel programming, the user has more control over task distribution, based on the number of available logical processors.

Source: http://www.peter-urda.com/2010/10/asynchronous-versus-parallel-programming

Summary

- What is a Distributed System?
  - *"A distributed system is a collection of independent computers that appears to its users as a single coherent system."*
- Models of Distributed System?
  - Minicomputer model
  - Workstation model
  - Workstation-server model
  - Processor-pool model
  - Hybrid model
- What are the Strengths and Weaknesses of a Distributed System?
  - S: Reliability, Incremental growth, Resource sharing
  - W: Programming, Reliance on network, Security
- Important characteristics of of a Distributed System?
  - Heterogeneity, Openness, Transparency