Lecture 1

Distributed Systems

Lecture Outline

- Concept of Parallel Computing

- Introduction to Distributed System
  - Definition and History
  - DS Model
  - DS Strengths and Weaknesses
  - DS Challenges

- Performance Measurements

Parallel Computing

- Little sequential processing involved in human mind
- Parallelism in Nature
  - Traffic and crowd movements
  - Weather, ocean currents, tectonic plates
  - Industrial processes, assembly lines, chemical plants, building projects
- Universe, essentially a parallel computer

On Sequential Computing

- Essence of Sequential Processing
  - Architectural design for execute-stored-programs - Von Neumann model
  - Earlier models with fixed programs
  - Movie reel, musical box, clockwork
  - Simplicity in Programming, linear time-scale driven

- How fast can a sequential computer go?
  - Faster clock
  - Speed of light and energy disposal

- How to by-pass the universal barriers?
  - Will it open a pathway to extreme programming for strategic apps where speed matters, answering big questions, discovering more?

Application Areas

- Big Data
- Mission critical applications: Weather, decision support, urgent action
- Scientific and Engineering: Cars, buildings, oil exploration, medical image recognition
- Simulation and Modeling: Strategic responses, policy design, disaster modeling, climate projections
- Entertainment
  - OC’ing

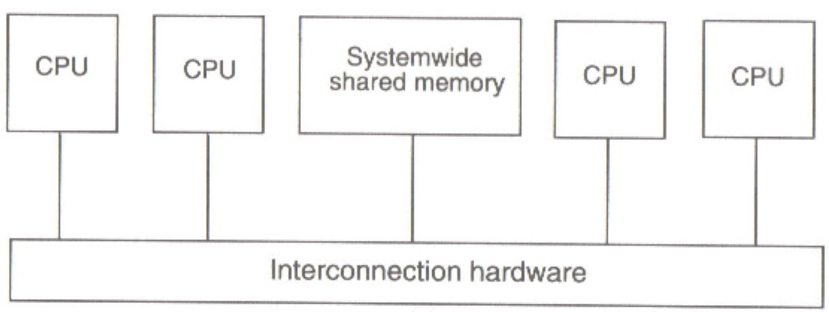
What is Parallel Computing?

- It is steps further beyond OC' ing
  - Tweaking versus building a purpose specific system
- Requires learning a *new way* of programming to by-pass universal limits
- Understand the computer architecture and re-designing it to support this *new way*
- *New way* is controlling computers where time and concurrency are added to linear time-scale programming
- The computers collectively and cooperatively function in a model termed as the distributed systems
- Requires formal understanding of the distributed systems

DS Introduction

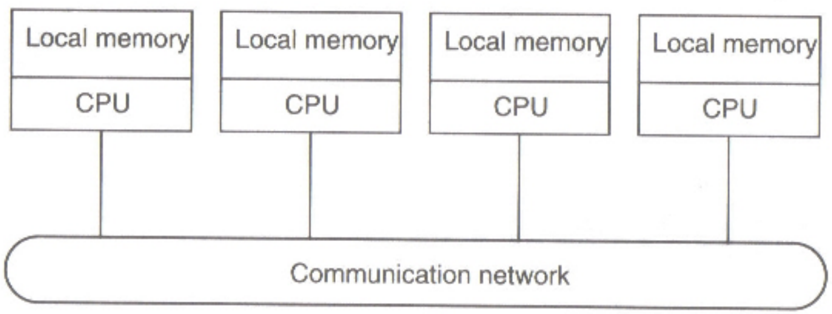
- Computer architectures consisting of interconnected multiple processors are basically of two types –

- Tightly coupled systems
  - Single system wide primary memory (address space) shared by all the processors



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)

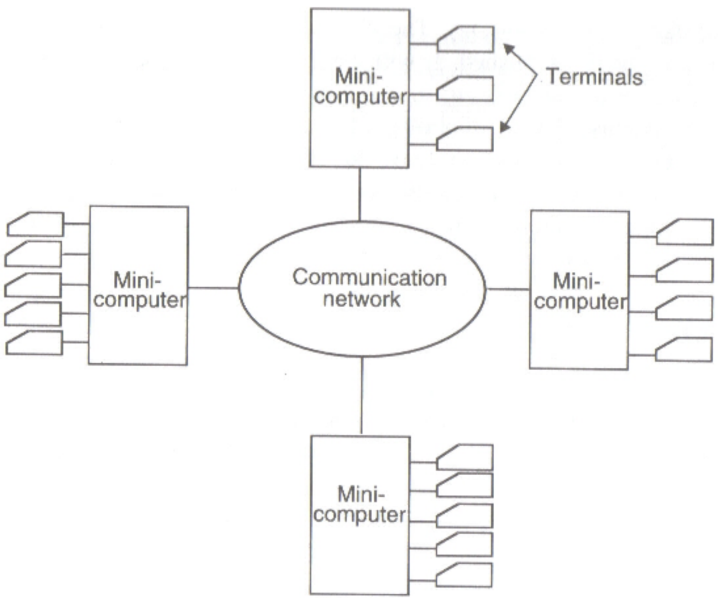
     Transfer rate: 64 kbps up to 1Tbps

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

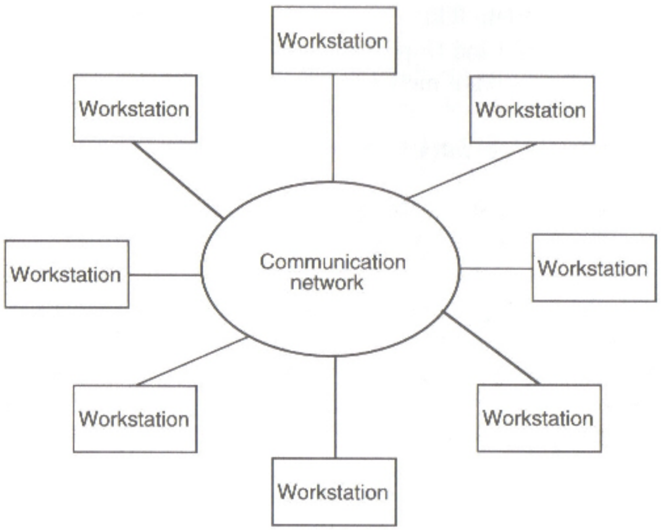
Distributed computing system models

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

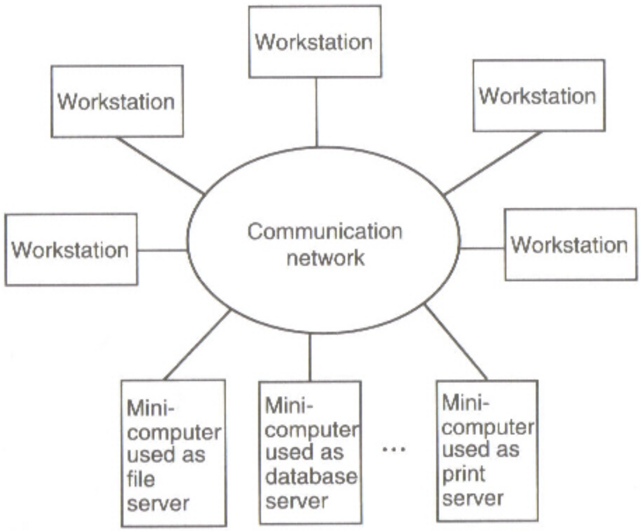
2. Workstation model:
   - several workstations interconnected by communication network
   - basic idea is to share processor power
   - user logs onto home workstation and submits jobs for execution, system might transfer one or more processed to other workstations
   - issues must be resolved –
     - 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

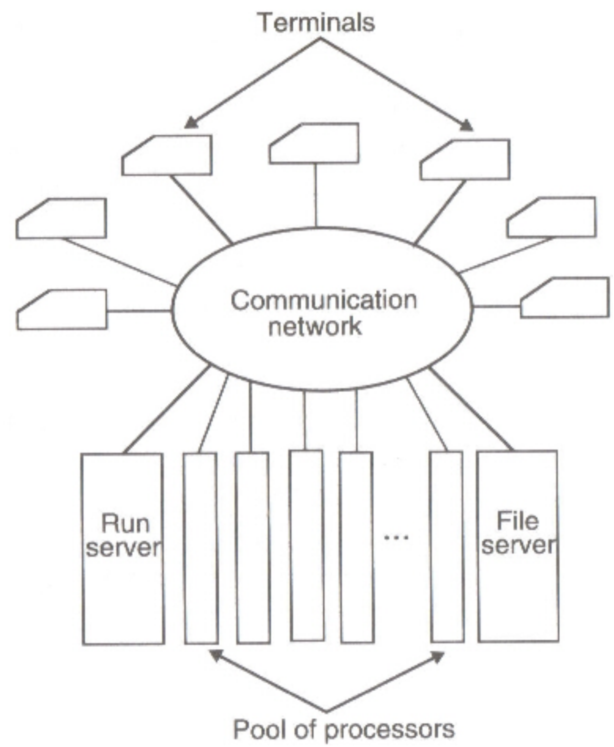
- Base observation –
  sometimes user need NO computing power, but once in a while he needs very large amount of computing power for a short period of time

- 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!!

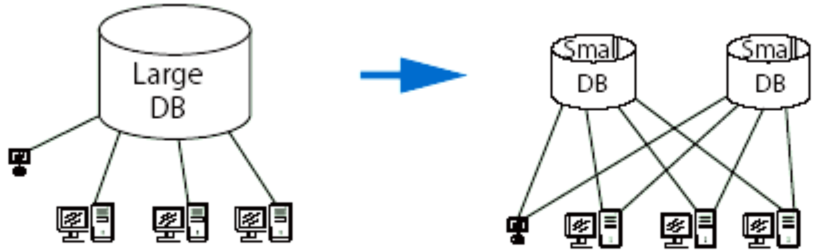
Distribution Model

- 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 the most 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

Heterogeneity

A distributed system is composed of a heterogeneous collection of computers

Heterogeneity arises in the following areas-

- networks: Even if the same Internet protocol is used to communicate, the performance parameters may widely vary within the inter-network.
- computer hardware: Internal representation of data is different for different processors.
- operating systems: The interface for exchanging messages is different from one operating system to another.
- programming languages: Characters and data structures are represented differently by different programming languages.
- implementations by different developers: Unless common standards are observed, different implementations cannot communicate.

Openness

- Openness means disclosing information: the usage of services provided by remote computers in particular.
- Open systems are easier to extend and reuse.
- By making services open, servers can be used by various clients.
- The clients which use services provided by other servers can extend the services and again provide services to other clients.
- The openness of distributed systems let us add new services and increase availability of services to different clients.

Transparency

- Transparency means hiding something.
- Transparency is an important issue to realize the single system image which makes systems as easy to use as a single processor system. E.g. WWW we can access whatever information by clicking links without knowing whereabouts of the host.

Classification of Transparency

- Access transparency: Data and resources can be used in a consistent way.
- Location transparency: A user cannot tell where resources are located
- Migration transparency: Resources can move at will without changing their names.

Classification of Transparency

- Replication transparency: A user cannot tell how many copies exist.
- Concurrency transparency: Multiple users can share resources automatically.
- Failure transparency: A user does not notice resource failure.
- Performance transparency: Systems are reconfigured to improve performance as loads vary
- Scaling transparency: Systems can expand in size without changing the system structure and the application programs.

Performance

- Fine-grained parallelism: Small programs are executed in parallel.
  - Large number of messages.
  - Communication overhead decreases the performance gain with parallel processing.

- Coarse-grained parallelism:
  - Long compute-bound programs executed in parallel.
  - Communication overhead is less in this case.

Scalability

- Scalability is the issue whether a distributed system works and the performance increases when more computers are added to the system.

- The followings are potential bottle-necks in very large distributed systems
  - Centralized components: A single mail server for all users.
  - Centralized tables: A single on-line telephone book
  - Centralized algorithms: Routing based on complete information

- Use decentralized algorithms for scalability:
  - No machine has complete information about the system state.
  - Machines make decisions based only on local information.
  - Failure of one machine does not completely invalidate the algorithm.

Reliability

- We have high probability to have faulty components in a distributed system because the system includes large number of components.

- On the other hand, it is theoretically possible to build a distributed system such that if a machine goes down, the other machine takes over the job.

- Reliability has several aspects.
  – **Availability: The fraction of time that the system is available. It can be expressed by the following equation-**

$$R = \frac{usable\_time}{total\_time}$$

  – **Fault tolerance: Distributed systems can hide failures from the users.**

Performance

- Maximum aggregate performance of the system can be measured in terms of Maximum aggregate floating-point operations.

$$P = N*C*F*R$$

- Where P performance in flops, N number of nodes, C number of CPUs, F floating point ops per clock period - FLOP, R clock rate.
- The similar measures with MOP/MIP.

Scalability

- It is computed as

$$S = T(1) / T(N)$$

- Where T(1) is the wall clock time for a program to run on a single processor.
- T(N) is the runtime over N processors.
- A scalability figure close to N means the program scales well.
- Scalability metric helps estimate the optimal number of processors for an application.

Utilization

- It is calculated as,

$$U = S(N)/N$$

- Values close to unity or 100% are ideally sought.

Summation

- 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
- Performance Metrics?