1.1.6 Pre-requisites

- Undergrad course in operating systems
- Good programming skills in a high-level programming language (Student can pick the language).

1.2 Introduction

1.2.1 Why Distributed Systems

Most of the systems that we use on a day-to-day basis today are distributed in some form, e.g. World wide web, Google search engine, Amazon, peer-to-peer file sharing systems, SETI@Home, grid and cluster computing, and modern networked computers. It is useful to understand how such real-world systems work and the course covers basic principles for designing distributed systems.

1.2.2 Definition of a Distributed Systems

A distributed system is a set of CPUs that are interconnected and work together to achieve some common goal. It is a collection of independent computers that appears to its users as a single coherent system. There are two components: computational units (CPUs) and communication media to allow them to communicate as one.

This broad definition includes parallel machines and networked machines. In a parallel system (referred as tightly coupled distributed system), multiple CPUs constitute a single physical machine and communication media is high speed interconnected bus. While networked machines are more common (referred as loosely coupled distributed system). There are multiple independent machines connected. The communication media is an actual network (WIFI, ethernet cable or WAN). This course will mainly focused on the later kind of system.

1.2.3 Advantages

Advantages over centralized system:

- Communication and resource sharing possible: Write application to allow communication (e.g. emails, social media). Also, resource on machine can be shared with/be available to another machine. One example of resource sharing is a single application uses hardware resources (e.g. GPUs) on multiple machines. Other examples can be online disk space and network printer.
- Economics price-performance ratio: There are two ways to scale an application to larger number of users: (1) buy a bigger machine which has more CPU power, memory and disk space, (2) buy lots of cheaper machines and connect them together to form a distributed system. The later has better price-performance ratio.
- Reliability, scalability: If one application is running on different machines and some of them fail, the application can continue to run. While in centralized system approach, if some cores fail, the whole application stops functioning. As for scalability, user can adding machines to the distributed system to scale the capacity of application or web.

- Potential for incremental growth: a start application may not be popular at the very beginning and it is not necessary to buy a very expensive server at that time. A good approach is to start with a small server and add additional servers when the application gets popular (growing number of users).

1.2.4 Disadvantages

- Distribution-aware PLs, OSs and applications: The system, which to run PLs, OSs and applications, gets more complicated to enable all of the benefits.
- Network connectivity essential: If the network dies, some part of the application may become barrage. For example, when the fiber breaks, the server goes down, and the files are no longer accessible.
- Security and privacy: Because applications are distributed on multiple machines, and each machine is accessing resources on the other part of the application on another machine, this opens the security holes, and it allows hackers to access and authorize data on system to steal data.

1.2.5 Transparency in a Distributed System

Distributed system have several properties. The question is which of these properties should expose to the users and which should be hidden from the users. Transparency along the access in distributed systems refers to what is hidden from the user of the system. There are some forms of transparency:

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be shared by several competitive users
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

When design a system, there will be numerous of design choices. The important part of design choices is what aspects of the system to expose to the users and what to hide from the users. That is a balance to achieve.

1.2.6 Open Distributed Systems

In an open distributed system, each component exposes its interface to other components. The API that used for each component to take to other components is public. A simple example is Google map. Closed systems typically don't disclose that interfaces. The components do interact with one another but the interfaces are not public. More systems today are becoming more open and APIs are published. We can use that to build new applications.

1.2.7 Scalability

One of the advantages of distributed system is its scalability. There are three ways to scale the system: scale on the service dimension, the data dimension and the algorithm dimension.

The centralized system has several scalability limitations:

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

Principles for good decentralized algorithm design:

- No machine should have the complete state: State means any data in a system needed for decision making.
- Make decisions based on local information
- A single failure doesn't cause a total system failure.
- No global clock

Techniques to avoid scalability problems include asynchronous communication, caching and replication.

1.3 Distributed Systems Models

The very early distributed system is called minicomputer model. They communicate with dial-up network. In these systems, each user had a local machine for processing but remote data could be fetched. Example are FTP, databases and emails.

The next model is the workstation model in the pre-PC days. This model allows processing to also migrate. Workstations could be connected in a local network. An example of this was the Sprite system which we will discuss later.

Client-server model is a very popular model for distributed system. Users have local workstations and they connect to powerful workstations that serve as servers.

In the processor pool model, terminals are Xterms or diskless terminals, and there are pool of backend processors that handle processing.

Cluster computing systems are a set of servers siting in a machine room in data centers. They are used by distributed web servers, scientific applications, enterprise applications.

Grid computing is a generalization of cluster computing that there are more than one cluster. They are cluster of machines connected over a WAN.

WAN-based clusters and distributed data centers are modern applications such as Google and Amazon.

The more modern version of this is cloud computing.

1.3.1 Emerging Models

Distributed pervasive systems: It is inexpensive to put computing devices e.g. tiny sensor node with networking capabilities into anything we can think of. Today, even a car is a distributed system. Other examples are mobile computing and sensor networks.

1.4 Uniprocessor Operating Systems

An OS acts as a resource manager or an arbitrator (manages CPU, I/O devices, memory). And OS provides a virtual interface that is easier to use than hardware.

Structure of uniprocessor operating systems:

- Monolithic architecture: One large kernel that handles everything (e.g., MS-DOS, early UNIX).
- Layered design: The functionality is decomposed into N layers. Each layer can only use services provided by the layer below it (N-1) to implement new services for the layer above it (N+1).
- Microkernel architecture: The OS kernel is decomposed into micro kernels, which are very compact kernel and only provide some very basic minimum sets of features. User-level servers implement additional functionality. The primary advantage security. While the disadvantage is decreased performance due to additional communication.

1.5 Distributed Operating System

The distributed operating system:

- Manages all the resources in a distributed system, seamlessly and transparently to the user.
- Looks to the user like a centralized OS but operates on multiple independent CPUs
- Provides transparency (location, migration, concurrency, replication)
- Presents users with a virtual uniprocessor

1.5.1 Types of Distributed OSs

System	Description	Main Goal
DOS	Tightly-coupled operating system for multiprocessors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general purpose services	Provide distribution transparency