Introduction to Distributed Systems

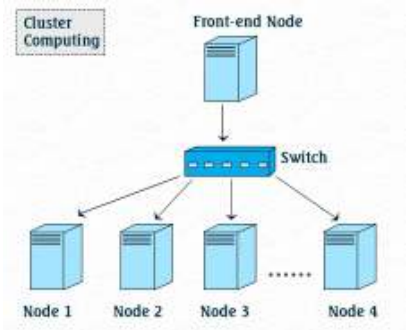
470410-1
Spring 2016
3/10/2016
Kyoung Shin Park
Multimedia Engineering
Dankook University

Distributed Systems

- World Wide Web
- Data centers and Cloud computing
- Cluster computing and Grid computing
- Wide area storage systems
- Banking systems and airline reservation systems

Cluster Computing

- It's a form of computing in which a group of PCs are linked together so that they can act like a single unit.
- Cheaper to build than a mainframe supercomputer
- It's scalable (can grow a cluster by adding more PCs)
- Homogeneous network; similar computers

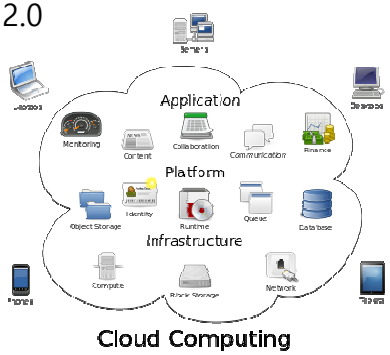


Grid Computing

- It's a form of distributed computing whereby resources of many computers in a network is used at the same time, to solve a single problem.
- To help scientists around the world to analyze and store massive amounts of data by sharing computing resources - **virtual organizations**; **job scheduling**
- Heterogeneous network; different computers running various kinds of OS

Cloud Computing

- It's a kind of Internet-based computing that provides shared processing resources and data to computers and other devices on demand.
- Cloud computing incorporates IaaS (infrastructure as a service), PaaS (platform as a service) and SaaS (software as a service) as well as Web 2.0



Cloud Computing

What is a Distributed System?

- **"a collection of independent computers that appears to its users as a single coherent system"** (Tanenbaum)
- "a collection of autonomous computers linked by a computer network with distributed system software" *(CDK01)*
- "a collection of processors interconnected by a communication network in which each processor has its own local memory and other peripherals and the communication between any two processors of the system takes place by message passing over the communication network" *(Sinha97)*
- "one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages" *(CDK01)*

What is a Distributed System?

- A collection of independent computers that appears to users as a single system - a virtual uniprocessor
  - Users do not know *(or care)* where *(on what machine)* files are located and where a job is executed
- Communicate with each other via messages
- Have no shared memory
- Have no shared clock
- Each computer has its own operating system

Why have Distributed Systems?

- Price/Performance
- Resource sharing
- Enhanced performance
  - Faster response time; higher throughput
- Improved reliability and availability
  - If one component goes down, the system does not
- Modular expandability
- Inherently distributed applications
  - Airline reservations; Bank ATMs
- Better flexibility

Problems with Distributed Systems

- All communication is done by **message passing** - meaning all coordination is decentralized
- There is a lack of global information
- There may be replication of data
- How can failures be detected and recoveries made?

Goals of a Distributed System

- Goals of a distributed system (Tanenbaum & van Steen)
  - Connecting users and resources
  - Transparency
  - Openness
  - Scalability

Goals of a Distributed System

- Challenges of a distributed system (Coulouris, Dollimore and Kindberg)
  - Heterogeneity
  - Openness
  - Security
  - Scalability
  - Failure handling
  - Concurrency
  - Transparency

Goals of a Distributed System

- Characteristics of a distributed system (Galli)
  - Shared resources
  - Openness
  - Concurrency
  - Scalability
  - Fault tolerance
  - Transparency

Concurrency and Shared Resources

- Clearly one of the goals of a distributed system is to **share resources** whether data, files, equipment, or machine cycles. This is similar to what is expected of a time-sharing system.
- However, in a distributed system, there are many processors.
- This means that any user can be active at any time and on any processor. This also means that any resource can have more than one **concurrent** request simultaneously.
- All the mutual exclusion and deadlock problems are once more to be considered, but with the caveat that more than one processor can be requesting any given shared resource.

Openness

- A characteristic that enables systems to be **extended to meet new** application requirements and user needs
- Achieved by specifying and documenting the key software interfaces of a system and making them available to software developers; i.e. the **interfaces** are published

Scalability

- Scalability refers to the ability of a distributed system to grow without users or applications knowing or being affected
- How can this be done
  - without a global clock or memory?
  - without a global state?
  - By avoiding any centralized components in the distributed system
    - software
    - Hardware
    - algorithms
  - By basing decisions
    - solely on locally-known information
  - By recognizing that
    - any component could go down at any time
    - and being able to continue anyway

Fault Tolerance

- Need **fault tolerance** if it is able to continue processing when one or more components of the system fail
- For distributed system to be fault tolerant, it must be able
  - to detect errors, faults, threats, or other failures
  - to tolerate the failures (i.e., not stumble or crash)
  - to mask the failures (i.e., hide them from the user)
  - to recover from the failures
- "Each component (of the distributed system) needs to be aware of the possible ways in which the components it depends on may fail or be designed to deal with each of those failures appropriately"
- Both redundancy and decentralization support fault tolerance

Heterogeneity

- A system is **heterogeneous** if it is composed of dissimilar hardware and software.
- Heterogeneity can be contrasted with portability:
  - A program that works in a heterogeneous environment must deal with various hardware and software components at the same time,
  - Whereas a portable program must run on different systems at different times.
- A related notion is **interoperability**. It denotes the ability of different components, possibly from different vendors, to interact.

Heterogeneity

- The goal of heterogeneity is to have
  - different operating systems
  - different computer hardware
  - different networks
  - different programming languages
- All working together to form a single distributed system
- **Communication protocols** can be used to mask the network differences
- **Middleware**, "an additional layer of software between the applications and the network OS" (Tanvan02) can be used to handle other differences

Transparency

- Network, Access, Location Transparency
- Name Transparency
- Concurrency, Parallelism Transparency
- Replication Transparency
- Migration (or Relocation), Persistence Transparency
- Failure Transparency
- Performance, Scaling Transparency
- Revision, Size Transparency

Network, Access, Location Transparency

- Network transparency = access + location transparency
- Access transparency
  - Enables local and remote information objects to be accessed using identical operations
  - This means that whether some processors in the distributed system are Windows machines, Unix machines, or Macs, whether they are Big or Little Endian machines
  - The user is able to **access objects** located on them **in exactly the same manner**
- Location transparency
  - Enables information objects to be accessed **without knowledge of their location**

Name Transparency

- Name transparency
  - The distributed system incorporates a **global naming scheme**
  - Objects (files, resources) are not tied to given nodes or sites by name
  - Name transparency assists migration, access, and location transparencies

Concurrency, Parallelism Transparency

- Concurrency transparency
  - Enables several processes to **operate concurrently using shared information objects without interference between them**
  - Permits efficient use of shared resources
  - Allows no interference between processes sharing resources
- Parallelism transparency
  - Permits parallel activities without users knowing how, where, and when these activities are carried out in the system

Replication Transparency

- Replication transparency
  - Enables **multiple instances of information objects** to be used to increase reliability and performance without knowledge of the replicas by users of application programs
  - This means that there may be multiple copies of files scattered over the entire distributed system

Migration, Persistence Transparency

- Migration (or Relocation) transparency
  - **Allows the movement of information objects** within a system without affecting the operation of users or application programs
  - This means that both resources and processes can migrate without users knowing and be accessed while being relocated
- Persistence transparency
  - Refers to the type of memory where files are located
  - Specifically, whether or not that memory is stable or volatile

Failure Transparency

- Failure transparency
  - Enables **the concealment of faults**, allowing users and application programs to complete their tasks despite the failure of hardware or software components
  - This means if a site goes down, it should be unapparent to other sites or users and work continues

Performance, Scaling Transparency

- Performance transparency
  - Allows the system to be **reconfigured** to improve performance as loads vary
- Scaling transparency
  - Allows the system and applications to **expand** in scale without change to the system structure or the application algorithms

Size, Revision Transparency

- Size transparency
  - Allows incremental growth of a system without the user's awareness
  - Clearly, this is a form of scaling transparency
- Revision transparency
  - Software revisions of the system are not visible to users
  - This is also a form of scaling transparency

What is a distributed operating system?

- "A collection of software components that simplifies the task of programming and supports the widest possible range of applications" (CDK01)
- "A distributed operating system is one that looks to its users like an ordinary centralized operating system but runs on multiple, independent central processing units (CPUs). The key concept here is transparency. In other words, the use of multiple processors should be invisible (transparent) to the user. Another way of expressing the same idea is to say that the user views the system as a 'virtual uniprocessor,' not as a collection of distinct machines." (Sinha97)

Issues in Distributed Operating System

- Global knowledge
- Naming
- Scalability
- Compatibility
- Process synchronization
- Resource management
- Security
- Structuring

Issues: Global Knowledge

- Unlike a uniprocessor or even a shared memory multiprocessor
- Unable to determine up-to-date global system state
  - No global memory
  - No common clock
  - Unpredictable message delays
- Need device-efficient distributed control
  - E.g. how to get a consensus between nodes in the system
- Need method for ordering events

Issues: Naming

- All objects (files, computers, services, etc) are named in order to differentiate one from another
- The name of an object should also be mapped onto its location
  - This can be done with a **naming service** that maps a logical name into a physical address
- Need a directory (or directories)
  - Replicated (how to maintain consistency between copies?)

    versus
  - Partitioned (i.e., should replicated pieces of it reside on different machines) (how to find the correct partition containing a name of interest?)

Issues: Scalability, Synchronization

- Scalability
  - Can system grow (both in size and in number of users) without performance degradation?
  - Want to avoid centralized components (i.e., centralized bottlenecks)
- Process synchronization
  - Enforce mutual exclusion to shared resources
  - Deal with potential for deadlock

Issues: Compatibility

- Compatibility refers to the notion of **interoperability** among the resources in a system
- Compatibility provides flexibility especially in a heterogeneous environment
- Compatibility may be possible at different levels
- **Binary** level:
  - All processing elements run same binary code
- **Execution** level:
  - Same source code can be compiled and run on all nodes (PC, Unix, Mac)
- **Protocol** level:
  - All processing elements support same protocols

Issues: Resource Management

- Data migration: bring data to the location
  - distributed file system
  - distributed shared memory
- Computation migration: computation moves to a remote location
  - e.g. RPC (Remote Procedure Call)
  - e.g. send a query for info computed remotely instead of requesting raw data
- Distributed scheduling
  - process migration

Issues: Security

- Authentication
  - verify user identification
- Authorization
  - determine user privileges

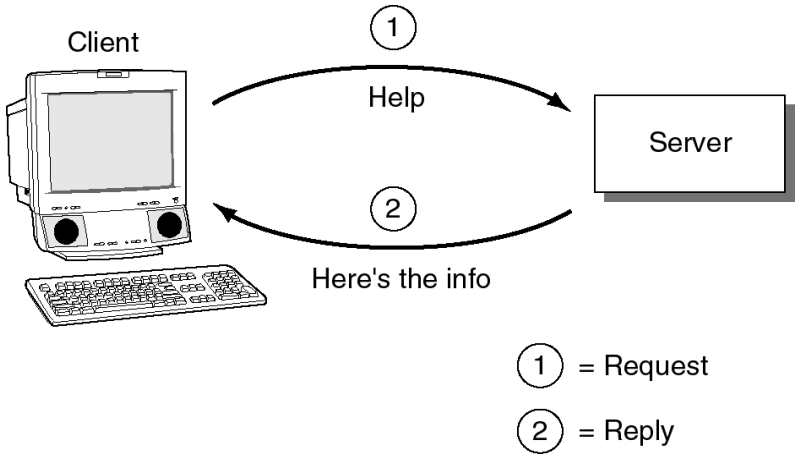
Issues: Structuring

- Monolithic kernel
  - Unix, OS/360
  - Each node doesn't need entire kernel
- Collective kernel
  - OS services are processes
  - Microkernel supports messages between such processes
- Object-oriented
  - OS services are a collection of objects

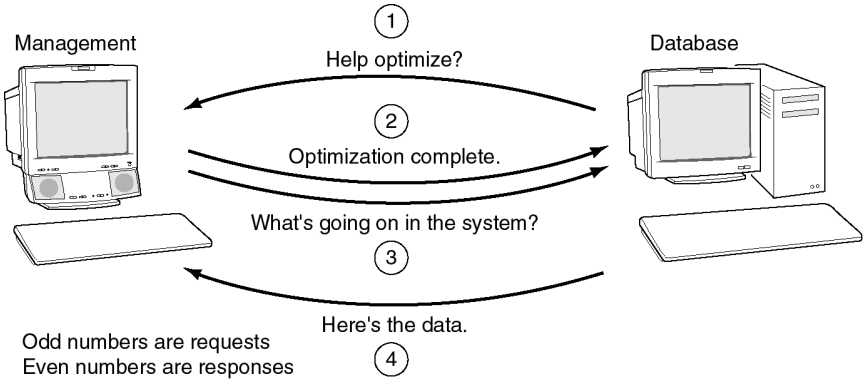
Client-Server vs. Peer-To-Peer

- Client-Server
  - Similar to collective kernel distributed O.S.
  - Servers respond to requests from clients
- Peer-to-Peer
  - An extension of client/server model
  - A many-to-many relationship between nodes

Client/Server Model



Peer-to-Peer Model

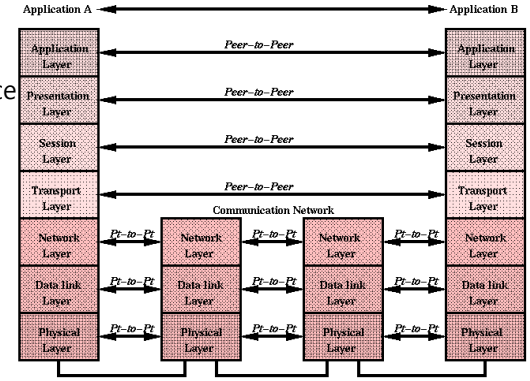


Network Protocol

- Protocol is a set of conventions for formatting data in an electronic communications system
- A method for
  - Establishing a connection between two sites
  - Sending a communication over the connection
  - Acknowledging receipt of message
  - Terminating the connection
- Examples: ISO/OSI, TCP/IP, UDP, SMTP

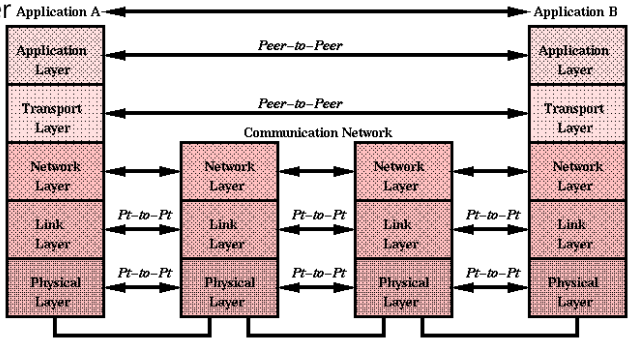
ISO/OSI Protocol

- Probably most popular network protocol model
- Implementation often takes efficiency-related shortcuts
- Includes 7 layers, grouped into 3 types
  - Application
  - Operating system
  - Communication service



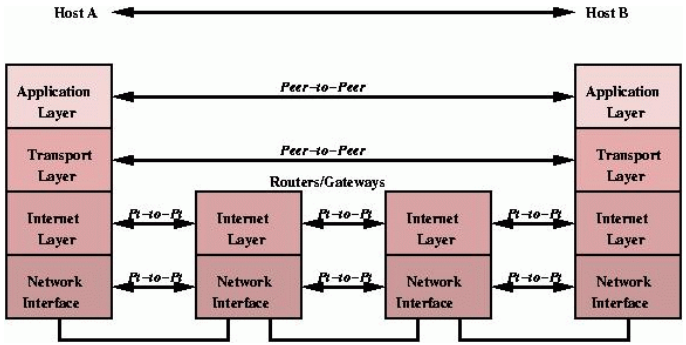
Internet Protocol Stack

- A 5 layer protocol
  - Application Layer
  - Transport Layer
  - Network Interface
  - Link Layer
  - Physical Layer



TCP/IP Network Architecture

- A 4-layer protocol on top of hardware (physical layer)
  - Application Layer
  - Transport Layer
  - Internet Layer
  - Network Interface



Reference

- http://www.cs.colostate.edu/~cs551/CourseNotes/Introduction/IntroTOC.html
- http://www.buzzle.com/articles/differences-and-similarities-between-grid-and-cluster-computing.html
- https://en.wikipedia.org/wiki/Cloud_computing