Introduction to Distributed Systems

20 October 2009
Lecture 1

Slide Credits: Insup Lee (UPenn)

Distributed Systems

- Why distributed systems?
  - availability of powerful yet cheap microprocessors (PCs, workstations, PDAs, embedded systems, etc.)
  - continuing advances in communication technology
- What is a distributed system?
  - A distributed system is a collection of independent computers that appear to the users of the system as a single coherent system.

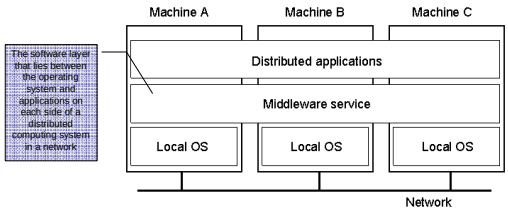
Examples

- The world wide web – information, resource sharing
- Clusters, Network of workstations
- Distributed manufacturing system (e.g., automated assembly line)
- Network of branch office computers - Information system to handle automatic processing of orders
- Network of embedded systems
- New Cell processor (PlayStation 3)

Advantages and disadvantages

- Advantages
  - Economics
  - Speed
  - Inherent distribution
  - Reliability
  - Incremental growth
- Disadvantages
  - Software
  - Network
  - More components to fail
  - Security

Organization of a Distributed System



A distributed system organized as middleware.
Note that the middleware layer extends over multiple machines.

Goals of Distributed Systems

- Transparency
- Openness
- Reliability
- Performance
- Scalability

Transparency

- How to achieve the single-system image, i.e., how to make a collection of computers appear as a single computer.
- Hiding all the distribution from the users as well as the application programs can be achieved at two levels:
  1) hide the distribution from users
  2) at a lower level, make the system look transparent to programs.

  1) and 2) requires uniform interfaces such as access to files, communication.

Transparency in a Distributed System

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	The users cannot tell how many copies exist
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

Different forms of transparency in a distributed system.

Openness

- Make it easier to build and change
- Monolithic Kernel: systems calls are trapped and executed by the kernel. All system calls are served by the kernel, e.g., UNIX.
- Microkernel: provides minimal services.
  - IPC
  - some memory management
  - some low-level process management and scheduling
  - low-level i/o (E.g., Mach can support multiple file systems, multiple system interfaces.)
- Standard interface, separation of policy from mechanism

Reliability

- Distributed system should be more reliable than single system.
  - Availability: fraction of time the system is usable. Redundancy improves it.
  - Need to maintain consistency
  - Need to be secure
  - Fault tolerance: need to mask failures, recover from errors.
- Example: 3 machines with .95 probability of being up
  - (1-.05)**3 vs 1-.05**3 probability of being up

Performance

- Without gain on this, why bother with distributed systems.
- Performance loss due to communication delays:
  - fine-grain parallelism: high degree of interaction
  - coarse-grain parallelism
- Performance loss due to making the system fault tolerant.

Scalability

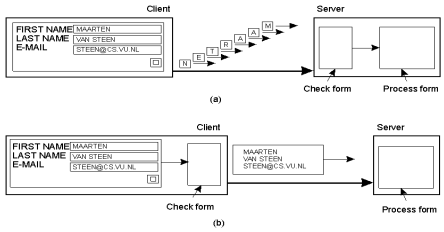
- Systems grow with time or become obsolete.
- Techniques that require resources linearly in terms of the size of the system are not scalable. (e.g., broadcast based query won't work for large distributed systems.)
- Examples of bottlenecks (i.e., scalability limitations)
  - Centralized components: a single mail server
  - Centralized tables/data: a single URL address book
  - Centralized algorithms: routing based on complete information

Scalability Problems

Characteristics of decentralized algorithms:

- No machine has complete information about the system state.
- Machines make decisions based only on local information.
- Failure of one machine does not ruin the algorithm.
- There is no implicit assumption that a global clock exists.

Scaling Techniques (1)

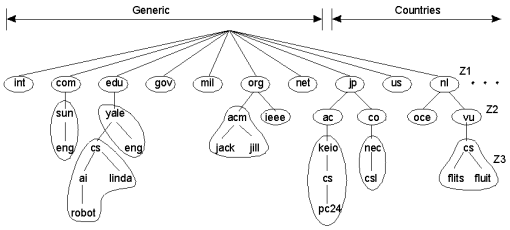


The difference between letting:

a) a server or

b) a client check forms as they are being filled

Scaling Techniques (2)



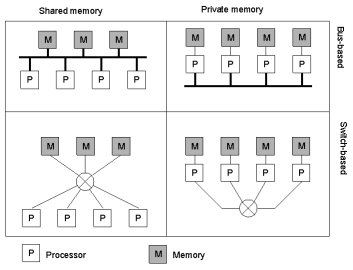
An example of dividing the DNS name space into zones.

Pitfalls when Developing Distributed Systems

False assumptions made by first time developer:

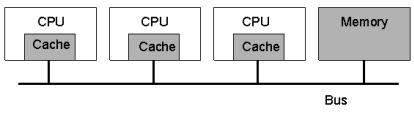
- The network is reliable.
- The network is secure.
- The network is homogeneous.
- The topology does not change.
- Latency is zero.
- Bandwidth is infinite.
- Transport cost is zero.
- There is one administrator.

Hardware Concepts



Different basic organizations and memories in distributed computer systems: multiprocessors vs. multicomputers

Multiprocessors (1)

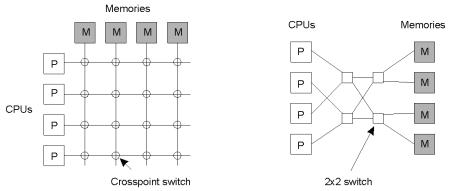


A bus-based multiprocessor

- Cache memory, hit rate, coherence, write-through cache, snoopy cache

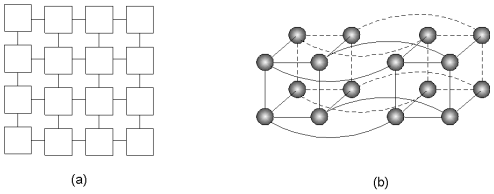
Multiprocessors (2)

a) A crossbar switch
b) An omega switching network



Homogeneous Multicomputer Systems

a) Grid
b) Hypercube



Tightly coupled vs. loosely coupled