START SLIDE Next: 1

File Name: /mnt/a/schragge/html/CSC8530/Intro.html

Distributed Systems

What is a distributed system?

A collection of autonomous computers
a) linked by a network
b) using software to produce an integrated computing facility

What size is a distributed system?

Local Area Network (10's of hosts)
Metropolitan Area Networks (100's of hosts)
Wide Area Networks (internet) (1000's or 1,000,000's of hosts)

Simple Distributed System

What applications are currently distributed?

Why are they distributed?

Examples

- Distributed Unix
- Wide area network applications
  - email - electronic mail
  - bbs - bulletin board systems
  - netnews - group discussions on single subject
  - gopher - text retrieval service
  - WWW - world wide web
  - multimedia/teleconferencing over networks
  - Multimedia information access and conferencing applications

BANKING Example

Key characteristics of distributed systems

1. Resource sharing

2. Openess
3. Concurrency
4. Scalability
5. Fault Tolerance
6. Transparency

Resource Sharing

Resource:
    hardware - disks and printers
    software - files, windows, and data objects
Hardware sharing for:
    convenience
    reduction of cost
Data sharing for:
    consistency - compilers and libraries
    exchange of information - database
    cooperative work - groupware

Resource Manager

Software module that manages a set of resources

Each resource requires its own management policies and methods

Client server model - server processes act as resource managers for a set of resources and a set of clients

Object based model- resources are objects that can move
    Object manager is movable.
    Request for a task on an object is sent to the current manager.
    Manager must be colocated with object.
Examples: ARGUS, Amoeba, Mach,
    migratory improvements - Arjuna, Clouds, Emerald

Openess

How it can be extended.

Open or closed with respect to
    hardware
    or software
Open
    - published specifications and interfaces
    - standardization of interfaces
UNIX was a relatively open operating system
    C language readily available
    System calls documented
    New hardware drivers were easy to add
    Applications were hardware independent
    IPC allowed extension of services and resources

Open distributed system

Published specifications

Provision for uniform interprocess communications and published interfaces for access

Conformance of all vendors to published standard must be tested and certified if users are to be protected from responsibility for resolving system integration problems

Concurrency

Multi-programming
Multi-processing

Parallel executions in distributed systems

1. Many users using the same resources, application interactions
2. Many servers responding to client requests

Scalability

How the system handles growth

Small system - two computers and a file server on a single network
Large system - current Internet

Scalability
- software should not change to support growth
- research area - for large, high-performance networks

Avoid centralization to support scalability
Choose your naming or numbering scheme carefully
Handle timing problems with caching and data replication

Fault Tolerance

Computers fail therefore we need:
    hardware redundancy
    software recovery

Increase in availability for services
Network is not normally redundant
Program recovery via the process group

Transparency

transparency of

- access
- location

- concurrency
- replication
- failure
- migration
- performance
- scaling

Key characteristics of distributed systems

1. Resource sharing
2. Openess
3. Concurrency
4. Scalability
5. Fault tolerance
6. Transparency

Key design goals

- High performance
- Reliability
- Scalability
- Consistency
- Security

Basic design issues

- naming
- communications
- software structure
- workload allocation
- consistency maintenance

Naming

- communication identifier
- name service
- contextual resolution of name
- name mapping
- pure names vs names with meaning

Communication

Reasons for communicating:
    transfer of data
    synchronization
Methods of communications
    message passing - send and receive primitives
    synchronous or asynchronous
    blocking or non-blocking
    mechanisms of message passing - channels, sockets, ports

client-server communication model
group multicast communication model

Client-server communications

Group Multicast

Software Structure

Workload Allocation

Workstation-server model
Processor pool model
Shared-memory multiprocessor

Workstation-server model

Processor pool model

Shared-memory multiprocessor

Consistency maintenance

Update consistency - changes must be atomic

Replication consistency - data must be consistent

Cache consistency (aka cache coherency)
- client hoarding of partial resources for local use
- usefulness depends on the principal of locality

Failure consistency - recovery via rollback

Clock consistency - use of timestamping and logical clocks

User interface consistency - action vs processing delay, delays < 0.1 second required

User Requirements

- Functionality
- Reconfigurability
- Quality of Service

Functionality

As a minimum:
    Distributed system should at least provide the function of a single computer
Improvement:
    - Sharing of resources - hardware
    - Ultilization of distributed resources for parallel processing and fault tolerance

Cooperative working environments

Migration paths from single computer to distributed system
1. adapt existing operating system
2. move to a new system designed for distributed systems
3. emulation of old system on new system (most practical)

Reconfigurability

Short-term changes
    Failed process or component
    Computational load shifted
    Caching and process migration (reduction of network overhead)
Medium to long-term evolution
    New machines
    New services
    Changed roles of machines
    Changed resources on existing machines

Quality of Service (QOS)

User needs guarantees

- Performance
- Reliability
- Availability
- Security