Distributed Systems

Overview

Distributed System: Definition

A distributed system is a piece of software that ensures that:

A collection of independent computers that appears to its users as a single coherent system
Distributed System (2)

Two aspects:
(1) independent computers and
(2) single system => Middleware

Goals of Distributed Systems

- Connecting resources and users
- Distribution transparency
- Openness
- Scalability
- etc
Distribution Transparency

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

**Note:** Distribution transparency may be set as a goal, but achieving it is a different story.

Degree of Transparency

**Observation:** Aiming at full distribution transparency may be too much:

- Users may be located in different continents; distribution is apparent and not something you want to hide
- Completely hiding failures of networks and nodes is (theoretically and practically) impossible
  - You cannot distinguish a slow computer from a failing one
  - You can never be sure that a server actually performed an operation before a crash
Degree of Transparency (2)

- Full transparency will cost performance, exposing distribution of the system
  - Keeping Web caches exactly up-to-date with the master copy
  - Immediately flushing write operations to disk for fault tolerance

Openness of Distributed Systems

- Open distributed system: Be able to interact with services from other open systems, irrespective of the underlying environment:
  - Systems should conform to well-defined interfaces
  - Systems should support portability of applications
  - Systems should easily interoperate
**Openness of Distributed Systems**

- **Achieving openness**: At least make the distributed system independent from heterogeneity of the underlying environment:
  - Hardware
  - Platforms
  - Languages

---

**Policies versus Mechanisms**

- **Implementing openness**: Requires support for different policies specified by applications and users:
  - What level of consistency do we require for client-cached data?
  - Which operations do we allow downloaded code to perform?
  - Which QoS requirements do we adjust in the face of varying bandwidth?
  - What level of secrecy do we require for communication?
### Policies versus Mechanisms

- **Implementing openness:** Ideally, a distributed system provides only **mechanisms**:
  - Allow (dynamic) setting of caching policies, preferably per cachable item
  - Support different levels of trust for mobile code
  - Provide adjustable QoS parameters per data stream
  - Offer different encryption algorithms

### Scale in Distributed Systems

**Observation:** Many developers of modern distributed system easily use the adjective "scalable" without making clear *why* their system actually scales.

**Scalability:** At least three components:
- Number of users and/or processes (*size scalability*)
- Maximum distance between nodes (*geographical scalability*)
- Number of administrative domains (*administrative scalability*)
Scale in Distributed Systems (2)

Most systems account only, to a certain extent, for size scalability. The (non)solution: powerful servers.

Today, the challenge lies in geographical and administrative scalability.

Techniques for Scaling

**Distribution:** Partition data and computations across multiple machines:
- Move computations to clients (Java applets)
- Decentralized naming services (DNS)
- Decentralized information systems (WWW)

**Replication:** Make copies of data available at different machines:
- Replicated file servers (mainly for fault tolerance)
- Replicated databases
- Mirrored Web sites
- Large-scale distributed shared memory systems
Techniques for Scaling (2)

Caching: Allow client processes to access local copies:
- Web caches (browser/Web proxy)
- File caching (at server and client)

Scaling – The Problem

Observation: Applying scaling techniques is easy, except for one thing:
- Having multiple copies (cached or replicated), leads to inconsistencies: modifying one copy makes that copy different from the rest.
- Always keeping copies consistent and in a general way requires global synchronization on each modification.
- Global synchronization precludes large-scale solutions.
Scaling – The Problem

1. **Observation:** If we can tolerate inconsistencies, we may reduce the need for global synchronization.

2. **Observation:** Tolerating inconsistencies is application dependent.

Distributed Systems: Hardware Concepts

- Multiprocessors
- Multicomputers
- Networks of Computers
Multiprocessors and Multicomputers

Distinguishing features:
• Private versus shared memory
• Bus versus switched interconnection

Multiprocessors (1)
### Multiprocessors (2)

- **a)** A crossbar switch
- **b)** An omega switching network

### Homogeneous Multicomputer Systems

- **a)** Grid
- **b)** Hypercube
Networks of Computers

High degree of node heterogeneity:
- High-performance parallel systems (multiprocessors as well as multicomputers)
- High-end PCs and workstations (servers)
- Simple network computers (offer users only network access)
- Mobile computers (palmtops, laptops)
- Multimedia workstations

Observation: Ideally, a distributed system hides these differences

Networks of Computers (2)

High degree of network heterogeneity:
- Local-area gigabit networks
- Wireless connections
- Long-haul, high-latency POTS connections
- Wide-area switched megabit connections

Observation: Ideally, a distributed system hides these differences
Distributed Systems: Software Concepts

- Distributed operating system
- Network operating system
- Middleware

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Main Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS</td>
<td>Tightly-coupled operating system for multi-processors and homogeneous</td>
<td>Hide and manage hardware resources</td>
</tr>
<tr>
<td></td>
<td>multicomputers</td>
<td></td>
</tr>
<tr>
<td>NOS</td>
<td>Loosely-coupled operating system for heterogeneous multicomputers (LAN and</td>
<td>Offer local services to remote clients</td>
</tr>
<tr>
<td></td>
<td>WAN)</td>
<td></td>
</tr>
<tr>
<td>Middleware</td>
<td>Additional layer atop of NOS implementing general-purpose services</td>
<td>Provide distribution transparency</td>
</tr>
</tbody>
</table>

- DOS (Distributed Operating Systems)
- NOS (Network Operating Systems)
- Middleware