Chapter 1
Introduction

Definition of a Distributed System (1)

A distributed system is:

A collection of independent computers that appears to its users as a single coherent system.

Definition of a Distributed System (2)

Figure 1-1. A distributed system organized as middleware. The middleware layer extends over multiple machines, and offers each application the same interface.

Transparency in a Distributed System

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how a resource is accessed</td>
</tr>
<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 is replicated</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>
</tbody>
</table>

Figure 1-2. Different forms of transparency in a distributed system (ISO, 1995).
Scalability Problems

<table>
<thead>
<tr>
<th>Concept</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized services</td>
<td>A single server for all users</td>
</tr>
<tr>
<td>Centralized data</td>
<td>A single on-line telephone book</td>
</tr>
<tr>
<td>Centralized algorithms</td>
<td>Doing routing based on complete information</td>
</tr>
</tbody>
</table>

Figure 1-3. Examples of scalability limitations.

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.

Figure 1-4. The difference between letting (a) a server or (b) a client check forms as they are being filled.

Scaling Techniques (1)

Figure 1-5. 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.

Cluster Computing Systems

Figure 1-6. An example of a cluster computing system.

Grid Computing Systems

Figure 1-7. A layered architecture for grid computing systems.

Transaction Processing Systems (1)

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN_TRANSACTION</td>
<td>Mark the start of a transaction</td>
</tr>
<tr>
<td>END_TRANSACTION</td>
<td>Terminate the transaction and try to commit</td>
</tr>
<tr>
<td>ABORT_TRANSACTION</td>
<td>Kill the transaction and restore the old values</td>
</tr>
<tr>
<td>READ</td>
<td>Read data from a file, a table, or otherwise</td>
</tr>
<tr>
<td>WRITE</td>
<td>Write data to a file, a table, or otherwise</td>
</tr>
</tbody>
</table>

Figure 1-8. Example primitives for transactions.
Transaction Processing Systems (2)

Characteristic properties of transactions:

- Atomic: To the outside world, the transaction happens indivisibly.
- Consistent: The transaction does not violate system invariants.
- Isolated: Concurrent transactions do not interfere with each other.
- Durable: Once a transaction commits, the changes are permanent.

Transaction Processing Systems (3)

![Nested transaction](image)

Subtransaction

Subtransaction

Airline database

Hotel database

Two different (independent) databases

Figure 1-9. A nested transaction.

Transaction Processing Systems (4)

![TP monitor in distributed systems](image)

Transaction

Client application

Requests

Reply

TP monitor

Request

Reply

Server

Request

Reply

Server

Server

Figure 1-10. The role of a TP monitor in distributed systems.

Enterprise Application Integration

![Middleware](image)

Client application

Communication middleware

Server-side application

Server-side application

Server-side application

Figure 1-11. Middleware as a communication facilitator in enterprise application integration.
Distributed Pervasive Systems

Requirements for pervasive systems

- Embrace contextual changes.
- Encourage ad hoc composition.
- Recognize sharing as the default.

Electronic Health Care Systems (1)

Figure 1-12. Monitoring a person in a pervasive electronic health care system, using (a) a local hub or (b) a continuous wireless connection.

Electronic Health Care Systems (2)

Questions to be addressed for health care systems:

- Where and how should monitored data be stored?
- How can we prevent loss of crucial data?
- What infrastructure is needed to generate and propagate alerts?
- How can physicians provide online feedback?
- How can extreme robustness of the monitoring system be realized?
- What are the security issues and how can the proper policies be enforced?

Sensor Networks (1)

Figure 1-13. Organizing a sensor network database, while storing and processing data (a) only at the operator’s site or...
Sensor Networks (2)

Figure 1-13. Organizing a sensor network database, while storing and processing data ... or (b) only at the sensors.

Sensor Networks (3)

Questions concerning sensor networks:

- How do we (dynamically) set up an efficient tree in a sensor network?
- How does aggregation of results take place? Can it be controlled?
- What happens when network links fail?