Distributed Systems: Architectural Issues

Dr. Yong Guan

Department of Electrical and Computer Engineering
& Information Assurance Center
Iowa State University
Outline for Today’s Talk

- Distributed Systems: Architectural Issues
Readings for Today’s Lecture

- Chapter 2 of “Distributed Systems: Principles and Paradigms”
Definition of a Distributed System

- From previous lecture

- A more precise definition:

  A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility.
# A Taxonomy of Distributed Systems

<table>
<thead>
<tr>
<th>Item</th>
<th>Distributed OS</th>
<th></th>
<th>Network OS</th>
<th>Middleware-based OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multiproc.</td>
<td>Multicomp.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree of transparency</td>
<td>Very High</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Same OS on all nodes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Number of copies of OS</td>
<td>1</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Basis for communication</td>
<td>Shared memory</td>
<td>Messages</td>
<td>Files</td>
<td>Model specific</td>
</tr>
<tr>
<td>Resource management</td>
<td>Global, central</td>
<td>Global, distributed</td>
<td>Per node</td>
<td>Per node</td>
</tr>
<tr>
<td>Scalability</td>
<td>No</td>
<td>Moderately</td>
<td>Yes</td>
<td>Varies</td>
</tr>
<tr>
<td>Openness</td>
<td>Closed</td>
<td>Closed</td>
<td>Open</td>
<td>Open</td>
</tr>
</tbody>
</table>

A comparison between multiprocessor operating systems, multicomputer operating systems, network operating systems, and middleware based distributed systems.
Architectural Styles

- Logical organization of software components in distributed systems

- The notion of architectural style:
  - Formulated in terms of components
    - How the components are connected each other
    - How and what data exchanged between them
    - How the components are jointly configured into a system

- Component and Connector
Architectural Styles (2)

- Important styles of architecture for distributed systems
  - Layered architectures
    - Network model: OSI & TCP/IP
  - Object-based architectures
    - Client/Server
  - Data-centered architectures
    - Web-based DS
  - Event-based architectures
    - Publish/Subscribe systems
  - Shared Data Spaces
Architectural Styles (3)

(a) Layered architectural style

(b) Object based architectural style
Architectural Styles (4)

Event-based architectural style

Shared data-space architectural style

Event delivery

Event bus

Publish

Component

Component

Data delivery

Publish

Component

Component

Shared (persistent) data space
System Architectures

- Centralized vs Decentralized organizations

- Client – Server Model
A Vanilla Network OS
(Remote Access System [Goscinsky ‘83])

Issues:
• Performance! (local and remote)
• Where is the state?
• Serialization of operations.
• Blocking operations
C/S Model: Application Layering

Issue: How to draw a distinction between client and server

Considering database apps, people have advocated a three-level distinction (following the layered architectural styles):

- The user-interface level
- The processing level
- The data level

Example: A simplified organization of an Internet search engine
C/S Model: Multi-tiered Architecture

The distinction into three levels suggests various possibilities for physically distributing a C/S application across several machines.

Alternative Client-Server Organizations (a-e)
C/S Model: Multi-tiered Architecture

Three-tiered Architecture:

An example of a server acting as client.
Decentralized Architecture

- **Vertical Distribution**
  - Distributed processing is equivalent to organizing a C/S application as a multi-tiered architecture

- **Horizontal Distribution**
  - A client or server may be physically split up into logically equivalent parts, but each part runs on its own share of complete data set, thus balance the load.
  - An example of horizontal distribution of a Web service

- Peer-to-peer systems
- Overlay Network
  - Structured
  - Unstructured
Structured Peer-to-Peer Architectures

- Distributed Hash Table (DHT)

- Example 1: Chord System
  - The mapping of data items onto nodes in Chord.
Structured Peer-to-Peer Architecture (2)

Example 2: CAN System
(a) The mapping of data items onto nodes in CAN. (b) Splitting a region when a node joins.
Unstructured Peer-to-Peer Architecture

- Randomized Algorithms
- Resemble Random Graph
- Framework for Overlay Construction (Jelasity 2004 and 2005)
  - Active Thread
  - Passive Thread
Framework for Overlay Construction

Actions by active thread (periodically repeated):

- select a peer \( P \) from the current partial view;
- if PUSH_MODE {
  - mybuffer = [[MyAddress, 0]];
  - permute partial view;
  - move \( H \) oldest entries to the end;
  - append first \( c/2 \) entries to mybuffer;
  - send mybuffer to \( P \);
} else {
  - send trigger to \( P \);
}
- if PULL_MODE {
  - receive \( P \)’s buffer;
}
- construct a new partial view from the current one and \( P \)’s buffer;
- increment the age of every entry in the new partial view;

The steps taken by the active thread.
Framework for Overlay Construction (2)

Actions by passive thread:

receive buffer from any process Q;
if PULL_MODE {
    mybuffer = [(MyAddress, 0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
}
construct a new partial view from the current one and P’s buffer;
increment the age of every entry in the new partial view;

(b)

The steps take by the passive thread
Overlay Networks: Topology Management

One key observation: By carefully exchanging and selecting entries from partial views, it is possible to construct and maintain certain topologies of overlay networks.

A two-layered approach for constructing and maintaining specific overlay topologies using techniques from unstructured peer-to-peer systems.
Overlay Networks: Topology Management

- If the lowest layer periodically executes the protocol in Fig. 2-9, the topology will evolve into a torus.
  - Examples of a torus include the surfaces of doughnuts and inner tubes.
Nodes such as those maintaining an index or acting as a broker are **Superpeers**.
Hybrid Architectures

- Edge-Server Systems
  - Viewing the Internet as consisting of a collection of edge servers
General structure of a distributed system as middleware.
In an open middleware-based distributed system, the protocols used by each middleware layer should be the same, as well as the interfaces they offer to applications.
**Middleware (3)**

- Interceptors: A software construct that breaks the usual flow of control and allows other application specific code to be executed.

- Using interceptors to handle remote-object invocations.
Middleware (4)

Interceptors: A software construct that breaks the usual flow of control and allows other application specific code to be executed.

Using interceptors to adapt the middleware

Three basic approaches to adaptive software:
- Separation of concerns
- Computational reflection
- Component-based design
Questions?

Thanks and See you next time
Self-Management in Distributed Systems

- Organizing DSs as high-level feedback-control systems allowing automatic adaptations to changes
  - Autonomic computing and Self-star systems
- Feedback Control Model

Diagram:

- Uncontrollable parameters (disturbance / noise)
- Initial configuration
- Corrections
- Core of distributed system
- Observed output
- Reference input
- Metric estimation
- Analysis
- Measured output
- Adjustment measures
- Adjustment triggers
Example 1: Systems Monitoring with Astrolabe

Data collection and information aggregation in Astrolabe

<table>
<thead>
<tr>
<th>Machine A</th>
<th>Machine B</th>
<th>Machine C</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP-addr</td>
<td>load</td>
<td>mem</td>
</tr>
<tr>
<td>192.168.1.2</td>
<td>0.03</td>
<td>0.80</td>
</tr>
<tr>
<td>192.168.1.3</td>
<td>0.05</td>
<td>0.50</td>
</tr>
<tr>
<td>192.168.1.4</td>
<td>0.10</td>
<td>0.35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>avg_load</th>
<th>avg_mem</th>
<th>avg_procs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.06</td>
<td>0.55</td>
<td>47</td>
</tr>
</tbody>
</table>
Example 2: Differentiating Replication Strategies in Globule

- The edge-server model assumed by Globule

- Dependency between prediction accuracy and trace length