Middleware

by

David G. Messerschmitt

Layering a computing infrastructure

Distributed Computing Applications and Infrastructure

The new infrastructure: middleware

- Layer of software between OS and application
  - Hides heterogeneity
  - Provides generic common services
  - Increases level of abstraction
- By its nature, not single platform, nor bundled with equipment
  - Microsoft attempts to be an exception

Middleware objectives

- Hide heterogeneity
- Location independence
- Common functionality needed by many applications
- Software portability and mobile code
- Help integrate legacy facilities
- Aid application interoperability
- Aid scalability

The new infrastructure

- Middleware is
  - where new capabilities are added by layering
  - where much of the experimentation and innovation happens in the infrastructure
  - where the successful approaches have a chance to become a spanning layer and/or integrated into a distributed OS
- Boundary to a distributed OS is fuzzy

Spanning layer

Network 1

Network 2

Spanning layers

Network protocol

Internet protocol

Application components

Middleware

Operating system

Network

Applications

Application components

Middleware
Some middleware categories

- Transaction processing
  – Simplify the coordination of complementary resource managers
- Message-oriented middleware
  – Support message and queuing capabilities where resource managers are not available simultaneously (like workflow)

Some middleware categories

- Distributed object management
  – Support applications that are distributed across heterogeneous platforms and organizations
- Mobile code
  – Allow application code to be moved and executed on heterogeneous platforms
  – Without prior software installation

Infrastructure software today

- With networks, new emphasis on:
  – Portability: applications run across multiple platforms (avoid lock-in)
  – Interoperability: pieces of application must work together (benefit from network effects)

Mobile code and Java

by
David G. Messerschmitt

Reminder: two key requirements

- With networks, new emphasis on:
  – Portability: applications run across multiple platforms (avoid lock-in)
  – Interoperability: pieces of application must work together (benefit from network effects)

Java emphasizes portability

Dynamic portability: mobile code

1. Send code (as a message) to a host
2. Execute the program represented by that code

Mobile code:
Code representing a software program that can be moved to heterogeneous platforms and executed there
Portability can aid interoperability

Mobile code originating from a common source can enhance interoperability

Java

- Portability
  - “Write once, run anywhere”
- Programming productivity
  - Garbage collection (no memory leaks)
  - Multi-threaded
- Scalability
  - Move execution cycles
- Interoperability
  - Software components come from common repository

Some mobile code advantages

- Executing program closer to user can enhance interactivity
- Mobile code originating from a common source can enhance interoperability and bypass network effects
- Shifting location of computation can enhance scalability

Idea of mobile code

1. Send as a message to a host
2. Execute the program

Mobile code:
Code representing a software program

Mobile agent:
Code and data representing an object or component

Idea of mobile agents

1. Agent launched
2. Agent executes in each host, modifying its state
3. Agent returns

Java virtual machine

- Java program
- Compilation
- Native machine instructions
- Bytecode: low level but machine independent
- Mobile code
VM as spanning layer

SUN/Java strategy

• License Java freely, even to rival Microsoft
  – Why?
• License terms give Sun a modicum of control over the “standard”
  – Why?
• How does Sun expect to make money?

Distributed object management

• Emphasis is on interoperability
  – Allows objects on one host to invoke methods of objects on another host
  – Platform, language independent
• CORBA vs DCOM
• Portability is not the emphasis

Interoperability

Interoperability also requires:
  • Common structure of data
  • Common interpretation of data
  • Agreement on protocols
Before and after

What is the acronym?

• Common Object Request Broker Architecture or
• Concerned Off-Road Bicyclist Association?

CORBA architecture

Protocol layer

Portability not promised

OMG process

• Identify need
• Request for proposals
• Process to
  – choose best
  – or ask proposal advocates to work together
Which is most effective?

- Industry de facto standard effort (CORBA)
- Single vendor integrated solution (DCOM)

Which is most effective?

CORBA vs DCOM

<table>
<thead>
<tr>
<th>CORBA</th>
<th>DCOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrate best ideas</td>
<td>Fast, no consensus required</td>
</tr>
<tr>
<td>Multi-vendor support</td>
<td>No vendor interoperability issues</td>
</tr>
<tr>
<td>Cross-platform and language</td>
<td></td>
</tr>
</tbody>
</table>

Two methods for application interaction

- CORBA and DCOM
- Exchange documents (XML)

What are their relative merits?

<table>
<thead>
<tr>
<th>COBA</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural OOP extension</td>
<td>Flexible data-sharing</td>
</tr>
<tr>
<td>No document interpretation</td>
<td>Good for document-like objects</td>
</tr>
<tr>
<td>Good for back-and-forth protocols</td>
<td>Natural cross-platform capability</td>
</tr>
</tbody>
</table>

Both have need for standardization of data or document interpretation

What CORBA provides

- Java
  - Language bindings
  - Transportable objects
- “Inter-galactic” software bus
  - Cross-platform and language
  - Interoperability (but not portability)
- High levels of abstraction
  - Remote method invocation on objects

What CORBA provides (con’t)

- Run-time flexibility
  - Everything self-describing
  - Interface discovery
  - Dynamic data structures and binding
- Useful services
  - Naming
  - Security
  - Many others
Importance of CORBA

- Inter-enterprise computing
  - Platform and language independence
  - Electronic commerce, network management, etc
- Reduction of network effects
  - Another spanning layer
  - Significance of platform reduced

Are Java and CORBA competitive or complementary?

- Both offer interoperability across different platforms
- Java offers portability and transportability
- CORBA offers heterogeneous language bindings
- CORBA offers many services, metadata, etc.
- Bottom line: they are complementary!
  - (but some Java proponents may not agree)

Who favors what?

Would these vendors be in favor of:
- Interoperability?
- Portability?

Microsoft
Intel
SUN
Novell
Iona

University of California at Berkeley
School of Information Management and Systems
Information Systems 206
Distributed Computing Applications and Infrastructure

Transaction processing

by
David G. Messerschmitt

The transaction

Durable, starting state
Collection of resource actions
Successful completion
Durable, consistent, ending state
Abort
Rollback

Transaction architecture

Application logic

Prepare, commit, abort
Join
Transaction manager

Resource managers
Commit or abort

Transaction manager

Phase 1: prepare()

Transaction manager

Phase 2: commit()

One or more “no’s”

All "yes”

Rollback

Atomic series of resource actions

Starting state

Inconsistent state

Final state

Locking to prevent conflicts

Starting state

Refused

Unlock

Final state

Abort

Starting state

Inconsistent state

Final state

Rollback

Starting state

Rollback

Transaction protocol

Application server

Resource manager

Transaction manager

request(tp_ID,...)

join(tp_ID)

commit(tp_ID)

commit_or_abort(tp_ID)

Lock

Commit or rollback