Distributed Computing and Java™

Sang Shin
Java™ Technology Evangelist
sang.shin@sun.com
Topics

- Overview of Distributed computing
- Java APIs for distributed computing
  - JDBC
  - RMI
  - Java IDL (Corba)
  - Java Servlets and JSP
  - J2EE™
  - Jini™ Network Technology and JavaSpaces (We will learn from next week)
Objectives

- Understand the *value* of distributed computing (network computing)
- Get some exposure to the *evolution* of distributed computing
- Understand the *7 fallacies* of distributed computing
- Get some exposure to distributed computing technologies of *Java*
Overview of Distributed Computing (Network Computing)
Virtual Fish Tank

“Most people grow up with a 'centralized mindset,' assuming that organized patterns can be created only by centralized control. The VFT exhibit helps people move beyond the centralized mindset, giving them a chance to create and participate in an artificial ecosystem -- and learn how patterns can, in fact, arise from decentralized interactions.”

- Michael Resnick, MIT Media Lab
Waves of Network Computing

- Computers
- Embedded Computers
- TVs
- Games
- Cars
- Phones
- Telecommunications
- IP v4
- IP v6
- SMTP
- HTTP
- RPC/XDR
- Jini
- JXTA
- UDDI
- LDAP
- Jini
- Telemetry

- Things - $10^{14}$
- Thermostats
- Packages
- Clothes
- Switches
- TVs
- Cars
- Games
- Phones
- Transfers
- Transactions
- Content
- Control
- Protocols
- Organization
- Clients
- Functions
- Transfers
- N-tier
- Client/Server
- Web Applications
- Web Services
- Telemetry
- Polyarchical
- Fractal
# Platform Evolution

<table>
<thead>
<tr>
<th>Catch Phrase</th>
<th>The Network Is the Computer</th>
<th>Objects</th>
<th>Legacy to the Web</th>
<th>The Computer Is the Network</th>
<th>Network of Embedded Things</th>
<th>Network of Things</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>100s</td>
<td>1,000s</td>
<td>1,000,000s</td>
<td>10,000,000s</td>
<td>100,000,000s</td>
<td>100,000,000s</td>
</tr>
<tr>
<td>Leaf Protocol(s)</td>
<td>X</td>
<td>X</td>
<td>+HTTP (+JVM)</td>
<td>+XML Portal</td>
<td>+RMI</td>
<td>Unknown</td>
</tr>
<tr>
<td>Directory(s)</td>
<td>NS, NS+</td>
<td>+CDS</td>
<td>+LDAP(*)</td>
<td>+UDDI +ebXML reg/rep</td>
<td>+Jini Lookup service</td>
<td>+?</td>
</tr>
<tr>
<td>Session</td>
<td>RPC, XDR</td>
<td>+CORBA</td>
<td>+CORBA, RMI</td>
<td>+SOAP, XML</td>
<td>+RMI/Jini</td>
<td>+?</td>
</tr>
<tr>
<td>Schematic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Design Patterns

Client-Server  3-Tier  Web Application  Web Services  Hybrid P2P  Fractal
Three Laws of Computing

- **Moore's Law**
  - Computing power doubles every 18 months
- **Gilder's Law**
  - Network bandwidth capacity doubles every 12 months
- **Metcalf's Law** *(Net Effect)*
  - Value of network increases exponentially as number of participants increases
Current Enterprise Reality

- Public Internet
- Corporate Internet
- Telco Server
- Web Server
- Video Server
- Solaris™ Workstation
- Thin Client
- NT Workstation
- Pager
- Cell Phone
- PDA
- Thin Client
- Solaris Workstation
- Workgroup Server
What is a distributed system?

- A collection of processes/processors that do not share memory or a clock
Why Distributed Systems?

- Resource sharing
- Higher performance
- Flexible
- Reliable
- Scalable
- Cheaper
- Manage’able (it depends)
Characteristics of Distributed Systems

- Complex programming
- Heterogeneous
  - Hardware, OS, networking behavior, object model, programming language
- Inherent problems
  - Network latency
  - Concurrency issues
  - Partial Failure
7 Fallacies of Distributed Computing

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
Terminology

- Naming and Transparency
  - Location transparency
  - Location Independence
- Stateful vs. Stateless
- Failure Detection and Recovery
  - Replication
  - Log based Recovery
- Parameter Marshaling/Unmarshaling
Distributed Coordination

- Mutual Exclusion
- Atomicity
- Concurrency Control
- Deadlock Handling
- Election Algorithms
- Transaction
  - 2 phase commit
Distributed Systems

• Evolution

RPC (procedural) | ORB (object based) | RMI (object oriented)

80s | 90s
Early Distributed Systems

- RPC
  - DCE, ONC
- Distributed File Systems
  - example - NFS, SMB
- Databases
  - example - Oracle
- Transaction Processing Monitors
  - example - CICS
At the root of it all ...

- Sockets
  - Socket, ServerSocket, SocketImpl classes

- Uses
  - Data compression
  - multicast
  - custom protocols, etc.
JDBC
What is JDBC?

- Java APIs for manipulating relational databases
- Uses standardized SQL
JDBC Architecture

Java application

JDBC API

JDBC Driver Manager

SPI

Bridge
ODBC

MS SQL

Sybase driver

Sybase

DB2 driver

DB2

Oracle driver

Oracle
A 2-tier Example

Front end

JDBC

Database

Front end

Database

Front end
JDBC Status

• Core API in JDK 1.1 onwards
• JDBC 3.0 Core and Extension API currently available
• JDBC drivers available for wide-range relational databases
JDBC Resources

• Java Packages
  ◆ java.sql
  ◆ javax.sql

• Websites
  ◆ java.sun.com/products/jdbc

• Books
  ◆ Database Programming with JDBC and Java by George E. Reese, George H. Reese
RMI
(Remote Method Invocation)
What is RMI?

- New Distributed system paradigm
  - Natural extension of the OO paradigm
  - Seamless integration of objects (objects everywhere)
- Allows method invocation across JVM boundaries
- Can pass or receive objects to/from remote object
  - Code moves with the object (code movement)
- Distributed garbage collection
- Activation
A 3-tier Example

![Diagram of a 3-tier system with Front end, Business Logic, and Database tiers connected by RMI and JDBC]
RPC Architecture (General)

The proxies deal with parameter marshaling/unmarshaling
RMI Architecture

The Stubs/Skeletons deal with Serializing/De-serializing
Class Loading

- RMI loads classes if not available locally
  - Remote object, stub, skeleton, parameter and return value classes
  - Loaded classes subject to installed security manager
Object Serialization

- Used for passing objects by value
  - Write/read objects and graphs of objects
  - Lightweight Persistence
  - Objects are converted into byte streams
  - Customization on a per-class basis is possible
RMI Development Process

- Four step development
  - Define the interfaces to your remote objects
  - Implement the remote objects
  - Run `rmic` on remote implementation classes
  - Make code network-accessible
RMI Deployment

- `rmiregistry`
  - Provides a bootstrap service to locate the first server object
- Start the server
- Start the client
RMI Status

• Core since JDK 1.1
• New Features in JDK 1.2
  ▪ Activation
  ▪ Custom socket types
  ▪ API enhancements
  ▪ Performance improvements
Customer Feedback

• Many customers like RMI:
  ♦ Easy to program
  ♦ No need to learn separate IDL
  ♦ Supports full Java programming language semantics (value objects)
  ♦ Ideal for communication between Java applications
Customer Feedback (Cont’d)

• Many customers like CORBA:
  ◆ Standardizing on CORBA IDL and IIOP
  ◆ Multi-vendor
  ◆ Cross-language
  ◆ Allows communication between Java applications and C++

• Common input: Why not RMI over IIOP?
RMI over IIOP

• RMI over IIOP supports a large subset of JDK 1.1 RMI API over CORBA IIOP
• Joint work between: IBM, Netscape, Oracle, Sun, Visigenic
• With input from many other partners
RMI over IIOP
RMI over IIOP (Simplified)

The ORBs deal with parameter marshalling/unmarshalling.
RMI over IIOP Features

- True subset of RMI over JRMP
- Uses CORBA objects-by-value extension
- Allows genuine interoperability with C++ and other languages
- RMI over IIOP combines:
  - RMI style ease of use plus cross-language interoperability
- **No** Distributed Garbage Collection
What is Java IDL

• Java IDL
  ♦ Uses OMG's Java Language Mapping
  ♦ Layered on Internet Inter ORB Protocol (IIOP)
  ♦ Generates Portable Stubs/Skeletons
Java IDL Development

1. Create your IDL Definitions
2. IDL to Java Compiler
   - stubs
   - Skeletons
   - (.java)
3. Write Client & Server Code
4. Java Compile
   - Client Stubs (.class)
   - Server Skeletons (.class)
   - Object Implementations (.class)
5. Object Adaptor
   - Register
   - Instantiate

Name Service
Java IDL runtime
## What Is XML?

- eXtensible Markup metaLanguage
- Universal Data description mechanism

<table>
<thead>
<tr>
<th>Language</th>
<th>Description</th>
<th>Pros</th>
<th>cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGML</td>
<td>Language for describing documents</td>
<td>Complete and extensible</td>
<td>Complex</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
<td>Simple</td>
<td>Focus on rendering; Not Extensible</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup metaLanguage</td>
<td>Simple, extensible</td>
<td>Need other technologies in conjunction</td>
</tr>
</tbody>
</table>
XML and Java Technology

- Complementary technologies
  - XML is portable reusable data
  - Java Technology is portable behavior
- Both are Web-friendly, open, vendor neutral, Unicode support
What is Needed?

- XML “Basic Plumbing” classes:
  - Ability to parse XML:
    - Parser(s)
    - SAX
    - DOM
    - Namespaces
    - Support in parser(s)
XML Schema

- XML = Data without syntax?
- Schemas specifies *syntax* of XML document
  - DTD
  - XML schema
J2EE
What Is the Java™ 2 Platform, Enterprise Edition (J2EE)?

- Open and standard based platform for
- developing, deploying and managing
- n-tier, Web-enabled, server-centric enterprise applications
The Java™ 2 Platform

Java Technology Enabled Devices
Java Technology Enabled Desktop
Workgroup Server
High-End Server
What Do You Get from J2EE?

- API and Technology specifications
- Development and Deployment Platform
- Reference implementation
- Compatibility Test Suite (CTS)
- J2EE BluePrints
# J2EE 1.3 APIs and Technologies

<table>
<thead>
<tr>
<th>Technology</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java 2 SDK, Standard Edition</td>
<td>1.3</td>
</tr>
<tr>
<td>RMI/ IIOP</td>
<td>1.0</td>
</tr>
<tr>
<td>JDBC™</td>
<td>3.0</td>
</tr>
<tr>
<td>Java Messaging Service</td>
<td>1.0.2b</td>
</tr>
<tr>
<td>JNDI</td>
<td>1.2.1</td>
</tr>
<tr>
<td>Servlet</td>
<td>2.3</td>
</tr>
<tr>
<td>JavaServer Pages™</td>
<td>1.2</td>
</tr>
<tr>
<td>JavaMail</td>
<td>1.2</td>
</tr>
<tr>
<td>JavaBeans™ Activation Framework</td>
<td>1.0.1</td>
</tr>
<tr>
<td>Enterprise JavaBeans</td>
<td>2.0</td>
</tr>
<tr>
<td>Java Transaction API</td>
<td>1.0.1</td>
</tr>
<tr>
<td>Java Transaction Service</td>
<td>1.1</td>
</tr>
<tr>
<td>Connector Architecture</td>
<td>1.0</td>
</tr>
<tr>
<td>ECPerf™</td>
<td>1.0</td>
</tr>
</tbody>
</table>
What is EJB Technology?

- Cornerstone of J2EE
- A server-side component technology
- Easy development and deployment of Java technology-based application that are:
  - Transactional, distributed, multi-tier, portable, scalable, secure, ...
Why EJB Technology?

- Leverages the benefits of component-model on the server side
- Separates business logic from system code
- Provides framework for portable components
  - Over different J2EE-compliant servers
  - Over different operational environments
- Enables deployment-time configuration
  - Deployment descriptor
EJB Architecture

- Client
- EJB Home
- EJB Object
- Enterprise JavaBeans™ Component
- Container
- EJB Server
- Database or Component

(Home Interface)
(Remote Interface)
What is a Servlet?

- Java™ objects which extend the functionality of a HTTP server
- Dynamic contents generation
- Better alternative to CGI, NSAPI, ISAPI, etc.
  - Efficient
  - Platform and server independent
  - Session management
  - Java-based
Servlets vs. CGI

- Request CGI1
- Request CGI2
- Request CGI1
- Request Servlet1
- Request Servlet2
- Request Servlet1

CGI Based Webserver

- Child for CGI1
- Child for CGI2
- Child for CGI1

Servlet Based Webserver

- Servlet1
- Servlet2

JVM
public class ExampleServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("Good Day USA!<BR>");
        Date rightNow = new Date();
        out.println("The time is: "+rightNow);
    }
}
What is JSP Technology?

• Enables separation of business logic from presentation
  ◆ Presentation is in the form of HTML or XML/XSLT
  ◆ Business logic is implemented as Java Beans or custom tags
  ◆ Better maintainability, reusability
• Extensible via custom tags
• Builds on Servlet technology
Big Picture

Client

HTML/XML
HTTP(S)

EJBs
IIOP
JDBC
JDBC
JSPs
## Servlets and JSP - Comparison

<table>
<thead>
<tr>
<th>Servlets</th>
<th>JSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>- HTML code in Java</td>
<td>- Java-like code in HTML</td>
</tr>
<tr>
<td>- Any form of Data</td>
<td>- Structured Text</td>
</tr>
<tr>
<td>- Not easy to author</td>
<td>- Very easy to author</td>
</tr>
<tr>
<td>- Underlying semantics</td>
<td>- Code is compiled into a servlet</td>
</tr>
</tbody>
</table>
JTS/JTA (Transaction Processing)

- Distributed transaction processing
- JTS: Mapping to CORBA Object Transaction Service (OTS)
  - Access to standard object transaction coordination
  - Who: IBM, Inprise, Bull, WebLogic
- JTA: API for transactions
  - Java interface to transactional systems
  - Who: Transaction/DB server vendors
• JNDI (Naming and Directory)

• JNDI: Java Naming and Directory
  ◆ Unified field theory for naming
  ◆ LDAP, NIS, NIS+, DNS, File, etc.
  ◆ Industry support
  ◆ Benefits:
    ▪ Single interface to enterprise network objects
    ▪ Cross platform
JMS (Messaging and Pub/Sub)

- JMS: Java Message Service API
  - Asynchronous communications
  - Publish and Subscribe
  - Reliable Queues
  - Guaranteed delivery
  - Benefits:
    - Single interface to MOM
    - Open and cross platform
  - Who: TIBCO, IBM, Modulus, Active
Java 2 Platform, Enterprise Edition

- API and Technology specifications
- Development and Deployment Platform
- Reference implementation
- Compatibility tests with branding
- J2EE BluePrints
Platform Value to Developers

- Unified platform
- Choice of platform implementation
  - Performance
  - Scalability
  - Reliability
  - Manageability
  - Tools
- Increased productivity
- Portability of code
- Ready-to-use security framework
J2EE is End-to-End Solution

**Client Tier**

**Middle Tier**

**Web Server**

**JSP, Servlets**

**HTML/XML**

**J2EE Application Server**

**Enterprise JavaBeans™**

**Enterprise Information Systems (EIS):**

- Relational Database,
- Legacy Applications,
- ERP Systems

**Other Services:**

- JNDI, JMS,
- JavaMail™

**Enterprise Information Tier**

**Firewall**
# Containers and Components

<table>
<thead>
<tr>
<th>Containers &amp; Handle</th>
<th>Components &amp; Handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrency</td>
<td>Presentation</td>
</tr>
<tr>
<td>Security</td>
<td>Business Logic</td>
</tr>
<tr>
<td>Availability</td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td></td>
</tr>
<tr>
<td>Persistence</td>
<td></td>
</tr>
<tr>
<td>Transaction</td>
<td></td>
</tr>
<tr>
<td>Lifecycle management</td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td></td>
</tr>
</tbody>
</table>
Lifecycle illustration

Creation
- Created by Component Developer

Assembly
- Assembled and Augmented by Application Assembler
- J2EE Modules

Deployment
- Processed by Deployer
- J2EE APP

Deploy
- J2EE Container
- Enterprise Components
Java 2 Platform, Enterprise Edition

• API and Technology specifications
• Development and Deployment Platform
• Reference implementation
• Compatibility Test Suite (CTS)
• J2EE BluePrints
What and Why a Reference Implementation?

- Validates specification
- Fully-compliant
- Fully-functional
- Not commercial quality
  - Scalability
  - Performance
- Use it for prototyping J2EE based applications
Compatibility Test Suite (CTS)

• Ultimate Java™ technology mission:
  ◆ Write Once, Run Anywhere™
  ◆ My Java-based application runs on any compatible Java virtual machines
  ◆ My J2EE based technology-based application will run on any J2EE based Compatible platforms
J2EE BluePrints

- Best practice guidelines, design patterns and design principles
  - MVC pattern
- Covers all tiers
  - Client tier
  - Web tier
  - Business logic (EJB) tier
  - Database access tier
- Sample code
  - Java Pet Store
Summary

• Java is unifying the fragmented distributed systems arena
• The ability to be able to move code makes Java unique
• Java is able to inter-operate with legacy applications