Building Mission-Critical Java Servers for Deployment on OS/390

Evolution & Revolution

P. Kanis
Managing Director
ADDC Infotech GmbH
Contents

• The SwiBi IS (Swisscom Billing Information System) project.
• Requirements & Constraints.
• Revolution, Java Servers.
• Architecture.
• Server Framework
  – Session Balancing.
  – Authentication and Access Control.
  – Monitoring & Automation.
  – Tracing and Logging.
  – Distributed garbage collection
• Getting it running; blood, sweat and tears
• Evolution, the future.
• Conclusion
Swisscom's billing information system for front and back office (GEFECO) was managed with a classic 3270 application running IMS transactions on the host.

SwiBi should replace most of this with an intelligent, modern enterprise server.

For initial deployment, the new must run in parallel with the old:
- Stuck with a legacy database for COBOL with fixed length fields.
- Have to use stored procedures written in COBOL.

Since successful deployment (Feb. 2000) Swisscom has decided to replace all the functionality of the old application with the new technology.

New servers for back-office functionality have also been added.
Requirements and Constraints

- Servers must run on **OS/390 USS** as well as **Unix** systems.
- Clients must run on standardised Windows NT stations.
- Servers must support up to 3500 concurrent intranet users with no noticeable performance degradation.
- Servers must be ~100% available between 6:30 to 22:00.
- Since deployment certain servers must also be available for batch processing of back-office reports.
- Users are already defined for the old software and should be authenticated through **RACF**.
- < 1 year from conception to deployment.
- The Architecture must be scaleable and applicable to other applications.
Java Servers, the Revolution

• Write once run anywhere?
  – March 1999 built a simple prototype on Win NT, tested on AIX then deployed it on OS/390.
• Chose to develop in Java
  – Iona promised us an OrbixWeb runtime for OS/390.
  – Fast development with good team development tools (VisualAge)
  – Many of the problems inherent in C/C++ disappear.
  – Much faster learning curve for new programmers compared to C++.
  – We found the Java promise of Java's "Write once run anywhere" to be 99.9% true.
  – If the application is well designed, this kind of application suffers no performance degradation compared to COBOL or C++.
  – Developing on OS/390 USS is not viable, Java can be developed on Win NT and deployed on the host.
Architecture on OS/390
Server Framework

- Required an infrastructure for our servers:
  - Hide the complexities of programming CORBA.
  - Free programmers to concentrate on the business logic
  - Supply a set of functionality that is applicable to any server built with this technology:
    - Session balancing.
    - Monitoring.
    - Tracing and logging.

- This resulted in the development of our Server Framework:
  - Fully configurable per server instance using java properties.
  - BaseServant interface inherited by all servers.
  - Customizable functionality using java reflection to load classes at runtime.
Server Framework Architecture

- **Application Server**
  - User App.
  - User App.
  - User App.
  - User App.
  - Repository Manager

**Framework Classes**
- Framework
- ORB
- Session Balancer
- Process Monitor
- Authentication Access Control
- Tracing & Logging
Session Balancing (1)

- Distributes client sessions between the available servers:
  - The balanced group class is a plug-in, the class is loaded at runtime. The default balancing algorithm is round-robin.
  - The balancing class also checks the status of the servers only returning valid object references to the clients.
- Registered servers use a “heart-beat” to check in with the Balancer.
  - If a Balancer crashes, the system automatically recovers when the new one is started.
- Servers are configured to register with the Balancer using java properties or command line switches.
- Supplies an interface for administration with a call-back object for the admin client.
Session Balancing (2)

OrbixWeb Administration Client

List Servers etc..

Server Status

OrbixWeb Session Balancer

Register check-in

shutdown

OrbixWeb SwiBi Servants

invocations

OrbixWeb Java 2 Client Applet

Connection info
Authentication and Access Control (1)

- SwiBi required Authentication & Access Control (AAC) using RACF.
- Other applications may require AAC using other security systems.
  - AAC Server has plug-in interface for the security access classes.
- The IBM JDK 1.1.8 security classes for RACF are not adequate for developing a serious authentication system:
  - They return erroneous error codes.
  - There is no way of verifying class/entity rights for a third party.
- We wrote our own JNI library for accessing RACF.
• Servers delegate their security to the AAC server
  – the servers log in to the AAC server and obtain a security context.
  – All other operations pass this context which is verified by the AAC server.
  – All calls to the AAC server are encrypted using 256 bit blowfish encryption.
  – The AAC server can be configured to only allow "server login" from specified servers with given userids.
  – The cipher class used is a plug-in class and can be easily changed.
• If an AAC server crashes (APAR in RACF) the delegation classes in the application can reconnect transparently to the new server.
• The clients for SwiBi also encrypt their userid and passwords when logging into the SwiBi servers using a different cipher.
Authentication and Access Control (3)

![Diagram of Authentication and Access Control interfaces and plug-ins]

- Authenticator Interface (IDL)
- Administration (IDL)
- Authentication & Access Control Server
  - Administration plug-in
  - Security Access plug-in
  - Server Manager plug-in
Monitoring & Automation (1)

- High availability requires Monitoring and Automation, no software is perfect by definition.
- The framework supplies a Monitoring server and Servers are configured to register with the monitor;
  - The monitor checks the servers at regular intervals.
  - If a server abends, the monitor will know about it and "tells" the application server to restart it.
  - The monitor sends messages to the syslog daemon.
- The servers also send start-up and shutdown messages to the syslog daemon.
- On OS/390 the messages sent to the syslog daemon end up in the System Log and can be automatically monitored and specific actions taken.
Monitoring & Automation (2)

- Originally, automation was achieved using the cron facility.
- The Application Server was developed to replace this primitive solution:
  - Start and stop servers at specific times/days.
  - Manually start, stop, suspend or kill a server.
  - Ensure that the servers are always available during their run time.
  - Facilitate set-up for running servers (especially on OS/390)
    - Set Job Name
    - Set UID/GID
    - Set configuration for balancing, monitoring, logging etc..
    - Servers obtain references for the AAC Server, Balancer, Monitor etc from the Application Server.
    - The Balancer etc. can be moved from one LPAR to another at run-time without reconfiguring the running servers.
Monitoring & Automation (3)

- User App.
- Process Monitor
- Application Server
- Start/Stop
- Monitor
- Restart Server
Debuggers are too invasive to be of any use during testing and leave no permanent record of events.

- Require a way of tracing distributed apps for testing.
- Require a permanent record of logs.
- Traces must be time stamped and contain information on the source of the trace such as Host, Process, Thread, Class, Method and line in the java file.

Require a script driven testing tool for repeatable testing and regression testing.

The same tracing/logging classes must also be used in production to maintain the validity of the testing phase.
Tracing & Logging (2)

• The Tracing and Logging Server and client classes along with the Repository Manager Server supply advanced logging functionality.

• The client class can be configured using Java properties or command line options:
  – Trace in the background
  – Trace to local log file and/or trace to the server
  – Echo traces to standard err with or without time stamps
  – Trace level for any thread can be set independently
  – The global trace level can be set at run time through the BaseServant interface.
Distributed Garbage Collection (1)

- Objects connected to the ORB **do not get deleted** as there is a reference to them in the **ORB**.
- There is no reference counting mechanism implemented in the OMG Java definition for CORBA.
- There is no direct access to the ORB object table in OrbixWeb.
- Must be automated, we don’t want to have to explicitly call garbage collection methods on the servant objects.
  - Should run in the finalizer of the client stubs.
- Have a standard interface **destroyable** which declares a **destroy** method which disconnects the objects from the ORB.
- All classes except the factory class inherit from this.
We built a post compiler for idl that adds a finalizer to the client classes if they contain the `destroy` method.

The Server Framework maintains a table of objects connected to the ORB.

The table is 2 dimensional, allowing a parent - child relationship between objects. If a parent is removed, all its children disconnected.

We added an “evictor” thread which disconnects objects after a configurable inactivity time (2 hrs).

The “evictor” thread is optional; it is started depending on a java property.
Getting it running; blood, sweat and tears (1)

- Deploying on OS/390 is not a trivial operation.
  - OS/390 is designed to run short running transactions. Persistent servers must have their goals defined in WLM or their priority is lowered to point where they are unusable.
  - Applications that use SAF must run under a userid with special privileges.
  - Java servers started in the foreground via an rlogin to USS have a nasty habit of going Zombie and can only be cancelled from the Operator's console!
  - Setting the job name for a server requires special privileges and the script must be started in the background.
  - The only viable way to set the uid/gid for a process is with a REXX script, after setting the job name.
  - Applications that start other applications require all the privileges required by their children.
Supporting products on OS/390 were very buggy:

- DB2 JDBC type 1 drivers hang, loop and crash under stress. They also block all the threads in the process during execution of the SQL statement and recovery of the result set.
- DB2 5.1 has a serious memory leak when processing complex dynamic SQL. Cannot maintain long lived database connections.
- The JIT compiler has a tendency to crash when compiling complex applications and for complex multi-threaded applications returns garbled stack traces.
- TCP/IP on v2r8 does not signal broken pipe for a socket when a process abends or is killed from USS; ioCallbacks and OrbixWeb bind no longer function correctly.
Getting it running (2)

• Solutions
  – Define a set of userids for the servers, each with a uid/gid for USS and with the necessary privileges.
  – In WLM define a 2 step goal for each server userid with a lower priority for the start-up period (2 min) and higher one for the run period.
  – Ensure that dB connections are not persistent. We close them after a number of transactions which terminates the DB2 thread and recovers the lost resources.
  – Run the servers without JIT, this has very little effect on performance and any loss can be recovered by "playing" with the WLM goal.
  – Run the servers on fixed ports and CORBA connection methods (IORs or COS Naming).
Getting it running (3)

• Results
  – We have a set of servers that has been running in production since Feb. 2000 with very high availability.
  – SwiBi less maintenance and operator intervention than GEFECO and is more stable.
  – The new technology performs far better than the old one.
  – The logging and monitoring services allow a high degree of automation using standard tools.
  – Installation and upgrade are much less complicated than for the old GEFECO system.
  – Hot fixes require simply changing a jar file or properties file (SQL statements are loaded at run time).
The Future, Evolution

• The version put into production in February was 5.0.1, we are now at 5.7.1. The revision number implies new functionality.
• New projects have been started or planned using the same architecture.
• By early next year, the whole of the billing information system will have been moved to Java/CORBA technology and the GEFECO will be taken off-line.
• Swisscom are moving the S/390 into the 21st century implementing distributed object technology on what many people refer to as the dinosaur.
• IBM have been fixing the bugs we found (lots of APARs) and the new JDBC drivers function much better.
Evolution (2)

- Testing with multiple LPARs and the Coupling Facility have been made and later this year Swisscom will be deploying this technology on Sysplexes.
- We are working on integrating the Balancing better with DNS/WLM and using VIPA for take-over in the case of failure of any of the major components of the system (TCP/IP, DB2 etc.).
- It is planned to make the framework Tivoli aware as Swisscom moves over to using this for system management.
Conclusion

- Contrary to popular belief the host is capable of supporting mission-critical distributed object technology.
- It's not easy and not cheap, but with the correct tuning and set-up, host systems supply all the necessary tools and services to maintain high availability Java/CORBA servers.
- Performance for applications that do a lot of database access is far superior to deploying on Unix boxes that access DB2 through DRDA (>10x).
- Time to market using Java is short, as is the learning curve for new programmers.
The SwiBi Team

Analysis, Design & Programming
Peter Kanis, Jürgen Fritz, Stefan Fink, Urs Schimming, Urs Badertscher, Markus Streit, Daniel Steiner, Daniel Wildi, Beat Bilang, Bruno Roth, Stefan Marty, Felix Boss, Niklaus Rötlisberger, Daniel Wildi

Management
Hans Brüschweiler, Heinz Bösch