Objectivity Case History (SE)

Customer Information

Customer: Meta-4 Inc

Date: March 25, 1999
Status: (X) SOLD ( ) Prospect ( )Integrator
Industry: Configuration Management
Application Domain: Year 2000 Compliance Management
Status: First version already deployed. About to re-engineer product from scratch.
Platform: NT
Compiler: Java
Other Tools: Rational Rose, Stingray Object Blend GUI widget set (planning to replace it with Swing), Kawa, Codewright, JPROBE, ObjectSpace JGL, ODBC, Crystal Reports, MS-Access

Currently they generate a lot of their code with Rational. They wrote VB Scripts to generate fetch() and markModified() calls, and create add methods for collections, etc.

Buying Criteria

SQL Access/Reporting: The ability to do ad-hoc reporting on the objects that were stored in the object database was a critical feature for the application. Objectivity's language heterogeneity and SQL++ interface met this need.

Minimal Impact on Development: Time-to-market on limited resources was an important factor. As long as minimum performance and functionality were met, the less intrusive the ODB package on application development the better. This correlates to an ease-of-use and a consistency of behavior and API criteria.

Why Objectivity

Relationship Management: The ability to store object relationships as bi-directional as part of the class definition and schema and to have referential integrity on these relationships enforced by the database kernel was a key factor in the selection of Objectivity. Other object databases only supported uni-directional relationships and left the responsibility of referential integrity up to the application developer, resulting in more development and testing effort.

Distributed Database Architecture: The distributed architecture of Objectivity provided reassurance that the solutions constructed today could be adequately scaled in the future with little to no additional effort in initial development or when deploying a scaled up solution.

Transparent Clustering Control: The ability to control object clustering through the use of the ClusterStrategy class provided a sophisticated interface for tuning and persistent object location on disk while transparently isolating those decisions from the rest of the application, allowing for easy modifications at a later date.

Dynamic Schema Definitions: The unique characteristic of Objectivity of using Java reflection to define schema at run-time as opposed to a pre- or post-processor resulted in several benefits. It simplified the development effort, allows for the use of integrated development environments with much less impact, and allows for new classes to be loaded into a running application and begin to immediately make instances of those classes persistent.

Technical Details

What does the application that you are developing do?

App Name: ICM, short for Intelligent Configuration Manager
It helps to keep track of Y2K compliance for components in an enterprise

What aspect of industry does it address?

Configuration Management

Who are the end users?

Systems Administrators, and Network Managers. This software permits them to see an overview of the whole system they administer. Some of the major customers of Meta-4 include those who manage biomedical devices, in hospitals and medical research centers, for example.

Will they sell it or use it in-house?

sell it

Development environment:

Hardware/operating system:

Windows NT

Language/compiler:

Java/JDK 1.7

development tools/3rd party libraries:

Rational Rose, Stingray Object Blend GUI widget set (planning to replace it with Swing), Kawa, Codewright, JPROBE, ObjectSpace JGL, ODBC, Crystal Reports, MS-Access

Currently they generate a lot of their code with Rational. They wrote VB Scripts to generate fetch() and markModified() calls, and create add methods for collections, etc.

**Data model:*

* How many classes?

250 approx

Their class hierarchy is divided up into 2 subcategories - one that derives from "instance" one that derives from "variety". For each "instance" class there is a corresponding "var" class, so there are mirror-hierarchies. Since they are all generated from Rational, they don't have the anti-pattern code maintainability problem.

There is a 1 to many relationship from a particular instance to a bunch of variety objects.

Each leaf node in this class hierarchy is a concrete class, and is mapped to its own container, so in other words, all instances of a particular type share a container. This is not very scalable and we discussed ways of re-organizing their objects so it is.

They have composite objects which may make sense to cluster together, for example, A computer is related to software, department, user, peripherals.
So they frequently open these "computer components" together.

* Is it complex data (or using lots of inheritance)?

8 deep

* How much data? (objects, megabytes)?

DB Name	# containers	Size
variety	250	27mb
[redacted]		

* *What is your physical system architecture? # clients/etc.?*

First of all, the ICM Client is actually an RMI client which does not use Objectivity directly. It communicates with an RMI server called ICM_APP_SERVER, which is a multithreaded Objy application.

When an ICM_CLIENT starts, there is a login screen and an RMI ICMSERVERMANAGER returns a remote reference to an ICM_APP_SERVER implementation. The client holds onto the App server remote reference, and then returns it back to a pool of recycled appserver implementations.

There is only one process running objectivity, with multiple sessions, each session joined to one thread, and all clients go through this one process.

Currently, their concurrency needs run around a dozen, but they're having problems getting more than a couple of users to do stuff simultaneously.

Anyway, the ICM_APP_SERVER is what sends viewable data sets back to the client by serializing the data.

The user can do multithreaded things, such as find, or report, while editing objects in another window. Therefore, each app server implementation can have multiple threads/sessions.

* What are your concurrency requirements?

10 users hammering away simultaneously using up to 50 thread/sessions total

* What is an example of a unit of work of your application - i.e. transaction?

Transactions are short because they have a cross-transaction locking strategy handled by the app server. Basically they open a transaction, do the persistent operations as fast as they can, adn close it, serialize the data over the wire, and let the client play around with it until it's done. Then they serialize data back to the server, do some identity resolution, and write back if necessary.

Contact Information

Objectivity Rep: Steve Fox
Customer Contact: Howard Olah-Reiken
Phone:
Email: