Applications Supported by an Orthogonally Persistent Java™ System

Mick Jordan
Michael L. Van De Vanter
Sun Microsystems Laboratories

in Collaboration with
The University of Glasgow, Scotland

OOPSLA '96, San Jose, California, USA, 6-10 October 1996

What is Orthogonally Persistent Java (PJava)?

- **An experimental Java platform (*not a product*)**
  - **provides *orthogonal* persistence**
- **A practical technology for persistent objects**
  - **extremely simple API**
  - **robust**
  - **scalable**
- **A research prototype**
  - **$PJava_0$ is in use**
- **An ongoing collaboration**
  - **Forest Project at SunLabs**
  - **Persistence & Distribution Group at Univ. of Glasgow, Scotland**
  - **$PJava_1$ under development**

Why Persistent Java?

- **Applications with complex, persistent state**
  - **manage very large webs of interconnected objects**
  - **require reliable storage between invocations**
  - **permit many users to modify data continuously**
  - **must be crash-recoverable**
- **Examples**
  - **Forest: software development environment for large Java systems**
  - **CAD/CAM systems**
  - **complex server-based applications**
- **Managing persistent objects is very hard**
  - **extra programming burden**
  - **loss of type safety and referential integrity**
  - **many simple solutions, but they don't scale**

Design Goals for Persistent Java

- **Extend Java's advantages to persistent objects**
  - **simplicity**
  - **safety**
  - **security**
- **Demonstrate Feasibility of Orthogonal Persistence**
  - **apply past research to an industrially supported language**
  - **expose a wider audience to its advantages**
- **Make *no language changes***
- **Offer a scalable solution**
  - **simple applications: nearly transparent persistence**
  - **complex applications: transactions**
  - **advanced applications: custom transaction support**

Principles of Orthogonal Persistence†

- **Data type orthogonality**
  - **all types have equal rights to persistence, longevity or brevity**
- **Persistence independence**
  - **all code should have the same form irrespective of the longevity of the data on which it acts**
- **Persistence identification**
  - **a simple and consistent mechanism for determining the longevity of values**
    - **persistence by reachability**

†From: "Orthogonally Persistent Object Systems," M. P. Atkinson and R. Morrison, *VLDB Journal* 4(3), pp. 319-401, 1995

Persistent Java Design Features

- **Orthogonal Persistence**
  - ***Data type orthogonality*: all types, including `Class` and `Thread`**
  - ***Persistence independence*: all Java code accepted without modification**
  - ***Persistence identification*: all objects that are reachable from registered roots**
- **API via class `PJavaStore`**
  - **denotes the *store* where persistent objects reside**
  - **supports global stabilization (checkpoint)**
  - **provides root registration, lookup, reflection on persistent classes**
- **Extensible Transaction Model**
  - **framework for defining specialized transaction kinds**
  - **standard kinds provided, e.g.**
    - **flat (ACID) transactions**
    - **nested transactions**

PJava$_0$ Prototype

- **Features**
  - **based on Sun's Java Development Kit 1.0.2**
  - **modified virtual machine**
    - **minimal changes**
  - **object cache architecture**
  - **global store stabilization**
  - **off-line garbage collection of a store**
- **Limitations**
  - **no persistent threads**
  - **extensible transaction model not implemented**
  - **some problems with native and external state**
- **Nevertheless, many useful applications are possible...**

Three PJava$_0$ Applications

1. Persistent Pi

- **convert a small program from Java to PJava**
- **show simplicity of API**

2. Oscar

- **prototype map display from geographical information systems area**
- **show advantages of PJava for large data**

3. Jeeves

- **extensible HTTP server from Sun**
- **extend with a persistent "servlet"**
- **show easy integration with existing systems**

PJava$_0$ Application 1: Persistent Pi

- **What - A simple Java program with interesting state**
  - **Want to retain partial results, even if crashed**
  - **Demo task: convert "Pi" to a persistent application**
- **Basic API and Programming model**
  - **`PJavaStore.getStore()` gives a handle on the store**
  - **`getPRoot(String rootName)` looks up a persistent root**
  - **`newPRoot(String rootName, Object obj)`**
  - **`stabilizeAll()` checkpoints current state of persistent roots**
  - **exceptions in class `PJSException` report store-related problems**
  - **create a null store specially or with stand-alone program `pjavans`**
  - **designate store at interpreter invocation: `pjava -store <name>`**
  - **interpreter performs crash recovery at startup if needed**

PJava$_0$ Application 2: Oscar

- **What - Map display for a Geographical Information System**
  - **transforms NTF format map data into objects**
  - **displays the map via AWT**
- **Suitability for Orthogonal Persistence - HIGH**
- **Ease of Programming - HIGH**
- **Performance**
  - **display from PJava store 60x faster than from file system**
  - **memory usage proportional to accessed data - no *big inhale***
- **Extra Functionality**
  - **easy to make road colors persistent**

PJava$_0$ Application 3: Jeeves

- **What - HTTP server written in Java**
  - **extensibility through *servlets***
- **Suitability for Orthogonal Persistence - HIGH**
  - **lots of configuration data currently stored in many files**
- **Ease of Programming - MEDIUM**
  - **simple forms example, few extra lines of code**
  - **no change to Jeeves server**
  - **concurrency demands transactional behaviour for real applications**
- **Performance - encouraging (~6% penalty for PJava)**
- **Extra Functionality**
  - **fast access to persistent objects (object caching for free)**

Persistent Java Status and Future Plans

- **PJava$_0$**
  - **prototype in use at Sun and Glasgow**
  - **soon to be made available to academic and research groups**
  - **Forest being ported: scalable software development environment for Java**
- **PJava$_1$**
  - **implementation of extensible transaction model**
  - **scale up to larger stores (multi-gigabyte)**
  - **port to more platforms**
  - **support type (schema) evolution**
  - **implement multi-user Forest**
- **More information**
  - **http://www.sunlabs.com/research/forest**
  - **http://www.dcs.gla.ac.uk/pjava**