OBJECT DATABASE STANDARDS, PERSISTENCE SPECIFICATIONS, AND PHYSICS DATA*

DAVID M. MALON, EDWARD N. MAY

Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA

ROBERT L. GROSSMAN

University of Illinois at Chicago Chicago, IL, USA

CHRISTOPHER T. DAY, DAVID R. QUARRIE

Lawrence Berkeley National Laboratory Berkeley, CA, USA

Designers of data systems for next-generation physics experiments face a bewildering array of potential paths. On the one hand, object database technology is quite promising, and standardization efforts are underway in this arena. On another, lightweight object managers may offer greater potential for delivering the high performance needed from petabyte-scale data stores, and may allow more efficient tailoring to specific parallel and distributed environments, as well as to multilevel storage. Adding to the confusion is the evolution of persistence services specifications such as those promulgated by the Object Management Group (OMG). In this paper, we describe what we have learned in efforts to implement a physics data store using several of these technologies, including a lightweight object persistence manager aiming at plug-and-play with object databases, and a trial implementation of the OMG Persistent Data Services Direct Access protocol.

1 Introduction

1.1 Background

Object databases provide a formidable technology, and increasing standardization of interfaces, as exemplified by the evolution of the Object Database Management Group's ODMG-93 specification [1], bodes well for their future role in physics data storage. Lightweight object persistence managers, on the other hand, offer less functionality, but have already been shown (for example, in the Petabyte Access and Storage Solutions (PASS) project and in Fermilab's Computing for Analysis Project (CAP)) to be adaptable to specific high-performance environments, to operate effectively with multilevel mass storage, and to provide a consistent interface both at the workstation and at the parallel supercomputer level.

In a trial implementation, we have aimed to provide access to a physics data store by means of a lightweight object persistence manager, in a way that is upward compatible with ODMG-93-compliant databases. The idea is not merely that the persistence manager can one day be replaced by a true database, but further, that the two facilities can coexist—perhaps with the lightweight object persistence man-

*Work supported by the U.S. Department of Energy, Division of High Energy Physics, Contract W-31-109-ENG-38.

ager used where performance is critical, and the database used where transactional integrity is paramount—and that a user need not necessarily know which data are stored by means of which technology.

One litmus test is the ability to support a reasonable subset of the Object Management Group's Object Query Service specification with an interface that can be supported consistently by both a lightweight object persistence manager and a true object database. We describe our efforts in this direction, and their connection with efforts to implement the OMG's persistence services specification, which offers a different (and in some ways, philosophically conflicting) view of how objects, apart from databases, manage persistence.

2 Approaches to Persistence

2.1 Object Database Management Group Object Database Standard

The Object Database Management Group (ODMG) is an industry consortium of database vendors and others who have come together to agree on aspects of a common specification for object databases. These efforts have resulted in an evolving standard (currently ODMG-93 [1]) whose components include: an object model; an Object Definition Language (ODL); an Object Query Language (OQL); a C++ binding for ODL and OQL, and a C++ Object Manipulation Language; a Smalltalk binding for ODL and OQL, and a Smalltalk Object Manipulation Language.

2.2 Object Management Group Persistent Object Services Specification

The Object Management Group (OMG) is best known for its work on the Common Object Request Broker Architecture (CORBA). OMG has also produced specifications for a number of common object services likely to be needed in CORBA environments; among these is the Persistent Object Services Specification [2] (POS). While CORBA-compliant applications are not required to conform to this specification, the goal is "to provide common interfaces to the mechanisms used for retaining and managing the persistent state of objects." A key notion underlying POS is that the service is used to manage objects' *persistent state*; it does not manage *persistent objects*. The POS architecture provides interfaces for describing the location of persistent data (PID), for exporting persistence mechanisms to object clients (PO), for associating protocols and appropriate data service interfaces (PDS) with particular combinations of client objects and PIDs (POM), and more. See the POS specification [2] for details.

2.3 Lightweight Object Persistence Managers

Lightweight object persistence managers offer persistence mechanisms for objects, with less than full database functionality. While such software has been used successfully in a wide range of applications, there is no universal agreement on what minimal functionality a lightweight persistence manager must provide, nor on what database functionality is necessarily omitted. Typical designs strive to add persis-

tence to implementation language objects in a natural way, and may be designed for speed, for portability, or to exploit high-performance architectures.

2.4 Relationship between ODMG-93 and OMG POS

The Object Database Management Group and the Object Management Group have striven to define their specifications with an awareness of each others' work. While there is substantial common ground, there are a number of differences—for example, ODMG-93 databases store objects, for which language object mappings may be provided; OMG POS stores CORBA objects' states. The ODMG defines a richer object model, including templates and some specific relationships.

There are also several areas in which the specifications interact: in POS, for example, ODMG databases are one of many possible Datastores; moreover, POS specifically prescribes an ODMG-93 protocol (which does not seem, however, to be unambiguously defined). There is evidence of cooperation on other issues as well, including, significantly, definition of an Object Database Adapter more suitable to database applications than the CORBA default Basic Object Adapter.

3 Persistence Models: Trial Implementations

3.1 Argonne Lightweight Object Persistence Manager

Earlier PASS work successfully demonstrated use of a lightweight persistence manager based on the University of Illinois at Chicago's PTool [4] to build a multi-gigabyte physics data store, to deliver parallel query capabilities [5], and to provide transparent access to multilevel mass storage [6]. Our primary goal in defining the Argonne object persistence manager was to leverage this work in a way that would provide physicists with access to these capabilities without requiring them to write nonstandard software. If this effort is successful, data in the lightweight datastore could migrate to a true ODMG-compliant database without requiring users to re-specify data schema or rewrite user query code. A second goal of the effort was to explore the minimal interface needed by lightweight persistence software in order for ODMG databases to be buildable on top of them. A clearer understanding of this interface would make it easier to adapt even commercial database software to take advantage of high-performance architectures by means of special-purpose lightweight persistence managers as backend storage providers.

The user interface is a subset of the ODMG-93 C++ binding. The implementation is evolutionary: it does not support every ODMG-defined class (classes are added as needed), but every supported class is intended to behave as specified in ODMG-93. The interface obeys ODMG-93 semantics even where the corresponding functionality is unavailable; for example, the Transaction interface allows nested transactions and provides a potential scoping mechanism for pointer lifetimes, even though transactions cannot really be rolled back.

3.2 Physics Implementations

As part of the effort to understand these alternative approaches to persistence, three implementations were undertaken:

- a trial implementation of CORBA-level physics objects that used the POS-defined PDS_DA protocol for persistence. Because no such service is commercially available, we implemented PDS_DA on top of two lightweight object persistence managers—an Argonne version of UIC PTool, and the Argonne ODMG-aware lightweight persistence manager described above.

- a trial implementation of CORBA-level physics objects that used CORBA interfaces to ODMG-defined persistence mechanisms (e.g., class Database), rather than OMG POS. This work required writing CORBA wrappers for the ODMG interfaces, which were implemented using the Argonne persistence manager. We also used the opportunity to test OMG Object Query Service [3] collection interfaces as CORBA wrappers for ODMG collection classes.

- an implementation of a data store generated by ISAJET simulations, using the Argonne ODMG-aware persistence manager directly from C++, with no CORBA components. This was the easiest of the implementations in that it needed no CORBA layers, but the most complex in terms of physics data.

4 Some Observations

A detailed description of our alternative persistence model implementations is beyond the length constraints of this paper, but a few comments can be made about ODMG-93, OMG POS, and lightweight object managers.

4.1 ODMG

The evolving ODMG-93 specification looks quite promising from the point of view of modeling physics data. The ODMG Object Definition Language was rich enough to describe our ISAJET-based data model, and while we did not have access to Object Query Language facilities, we were able to express physics queries in C++ using the ODMG bindings. (We did hand-code several physics queries in OQL to investigate expressivity, but could not test them. A capability of which we are unsure in OQL, but which would be very useful for physics applications, is invocation of external user-defined functions or methods from with OQL queries.) We have minor concerns about the C++ binding, and some concerns about issues of scalability and parallelism, but many of these are expected to be addressed in later revisions of the ODMG specification.

4.2 OMG Persistent Object Service

The overall architecture of OMG's Persistent Object Service is appealing, though there seem to be some minor problems and many ambiguities. While POS is suited to storing objects' persistent states, it is probably not appropriate as the primary

interface to a database. In some cases, implementing POS protocols is only the beginning of the process of providing object persistence: while a protocol may describe the interfaces that are available for managing persistence, a specific object's use of the protocol to manage its own persistence may require substantial design and implementation effort. The amount of machinery involved in implementing POS can be quite daunting; finally, the granularity of access to persistent state data may be critical in deciding the appropriateness of POS for particular purposes.

4.3 Lightweight Object Persistence Managers

While use of lightweight object persistence managers may be appropriate in a variety of settings, a persistence interface that is compatible with standards-compliant object databases holds particular appeal. A litmus test is that data model definitions and client code should not need to be changed if some or all of the data move to such a database. A good lightweight persistence manager may provide interfaces to functionality outside the scope of an ODMG-93 implemention (e.g., for physical storage management), but such functionality should be provided in a way that does not conflict with ODMG interfaces.

A lightweight persistence manager could, in principle, cooperate with both the OMG Persistent Object Service and ODMG databases. One can use such a persistence manager as a Datastore behind OMG POS protocols, and we have done this in our implementations. One of many possible measures of a good lightweight object persistence manager may be whether its architecture and functionality are sufficient to support building an ODMG-93 database management system above it, with the lightweight manager providing the underlying persistence layer.

Acknowledgments

The submitted manuscript has been authored by a contractor of the U.S. Government under contract No. W-31-109-Eng-38. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

References

1. R.G.G. Cattell et al, *The Object Database Standard: ODMG-93 Release 1.1* (Morgan Kaufmann, San Francisco, 1994).
2. Jon Siegel et al, *Persistent Object Service Specification*, OMG Document Numbers 94-1-1 and 94-10-7 (Object Management Group, 1994).
3. IBM et al, *Joint Submission: Object Query Service Specification*, OMG TC Document 95.1.1 (Object Management Group, 1995).
4. R.L. Grossman and X. Qin in *Proceedings of SIGMOD 94*, (ACM, 1994).
5. D.M. Malon et al in *Computing in High Energy Physics '94*, (Lawrence Berkeley Laboratory, 1994).
6. E.N. May et al in *Computing in High Energy Physics '94*, (Lawrence Berkeley Laboratory, 1994).