
Persistence Brokers in a Distributed Object Environment

Robert W. Carey

Lawrence Livermore National Laboratory

1. Abstract

Managing database resources for distributed, object-oriented applications can be done efficiently through the use of persistence brokers. CORBA enabled, distributed application objects communicate with the persistence brokers to obtain database services. Such an approach insulates application software from the details of the persistence mechanism and reduces requirements for concurrent database licenses. We are using Oracle8 in the control system of the National Ignition Facility at the Lawrence Livermore National Laboratory. Object extensions and the eventual inclusion of CORBA services in Oracle8 maps well to our multi-tier, distributed object architecture.

Keywords : Distributed control system, object-oriented development, CORBA, persistence brokers, object pooling

2. The National Ignition Facility

The \$1.2 billion National Ignition Facility (NIF) laser [5][7] is under construction at Lawrence Livermore National Laboratory in California. When completed in 2003, it will be housed in a building the size of a football stadium-704 feet long by 403 feet wide [Figure 1].

This laser is the latest in a series of experimental machines used to study inertial confinement fusion: nuclear fusion reactions produced in a plasma of deuterium and tritium that is compressed by a burst of laser energy [1]. The NIF laser, which will be the world's largest high power laser, will deliver 1.8 MegaJoule pulses of optical energy onto a BB-sized fusion fuel capsule in a pulse 25 nanoseconds long. In an experiment, the target will be heated to more than 100,000,000 degrees Celsius and compressed to a density more than 20 times that of lead.

The scientific data that NIF produces in support of the inertial confinement fusion program has three diverse objectives. As a key component of the U.S. Department of Energy's Stockpile Stewardship and Management Program, the NIF will enable the U.S. to maintain its nuclear device stockpile without resorting to underground testing [8]. It will also collect preliminary data about fusion as an environmentally attractive energy source [4]. The ability to re-create conditions existing inside the sun and stars will significantly impact the science of astrophysics and high energy density physics [2].

NIF's Integrated Computer Control System (ICCS) [10] must integrate more than 60,000 control points distributed among 192 laser beamlines. Each shot of the NIF, which generates about 400 Mb of data, will require aligning laser components along 600-foot paths and pointing each beam within 50 microns of its assigned spot on a centimeter-scale target. All beams must arrive at the target within 30 picoseconds. Thermal cooling times, laser alignment, and computerized inspection of thousands of optical components drive the design performance of three shots per day. Management of scalar, vector, and image data for actuators and diagnostic sensors is essential for the successful operation of the NIF machine.

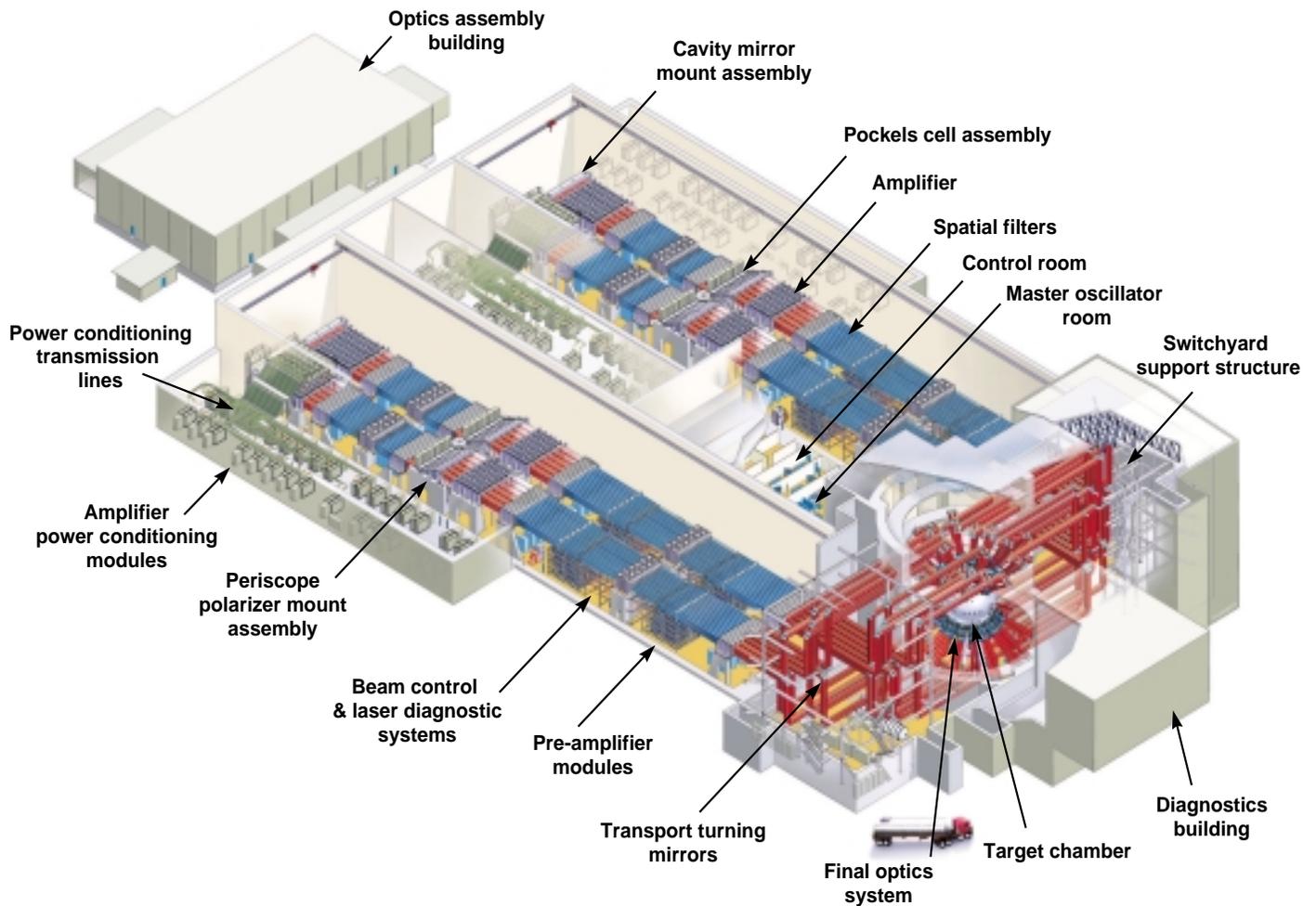


Figure 1 The National Ignition Facility

3. Control System Architecture

The ICCS is physically distributed among 450 processors, logically separated into laser subsystems, and layered to form a control hierarchy. Within the control hierarchy, front-end processors (FEP) are coordinated by a supervisory system [Figure 2]. Supervisory software, which is hosted on UNIX workstations, provides centralized operator controls and status, data archiving, and integration services. FEP units are constructed from VME/VXI-bus or PCI-bus crates of embedded controllers and interfaces that attach to laser hardware. FEP software provides the distributed services needed by the supervisory system to operate the laser hardware. Functions requiring real-time response are allocated to software within the FEP or embedded controller, which is directly connected to the laser hardware. The ICCS is an object-oriented design. The internationally standardized Common Object Request Broker Architecture (CORBA) [6] is an object to object communications mechanism. It is used for all supervisory and FEP communications.

Supervisory Controls

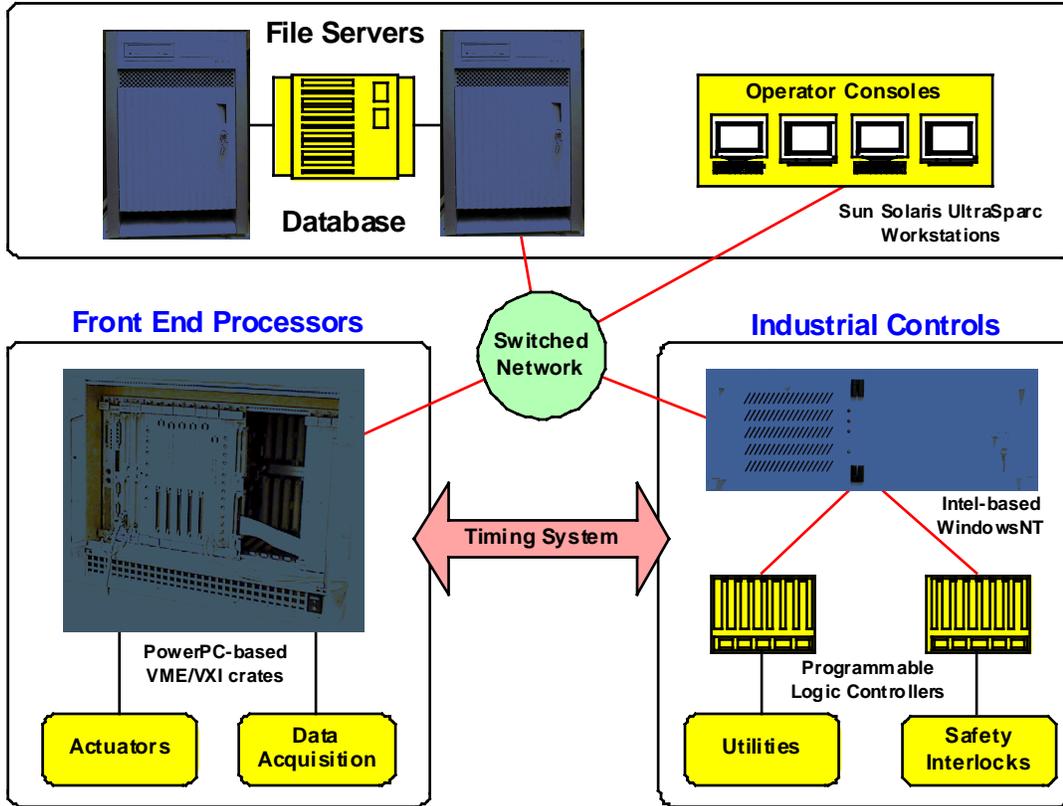


Figure 2 Integrated computer control system architecture

3.1 Supervisory Layer

Supervisory software is partitioned into eight cohesive subsystems, each of which controls a primary NIF subsystem such as laser alignment or power conditioning. The subsystems are integrated to coordinate operation of laser and target area equipment. The supervisory software is responsible for duties ranging from configuration and control sequencing to data processing and archival. It provides the human interface in the form of operator displays used to coordinate control functions for the NIF.

Two high-performance servers sharing redundant disk storage provide enhanced performance for database operations, with the added benefit of greater availability in the event one server fails. The database supports a complex data schema to manage both experimental data and data used during operations and maintenance.

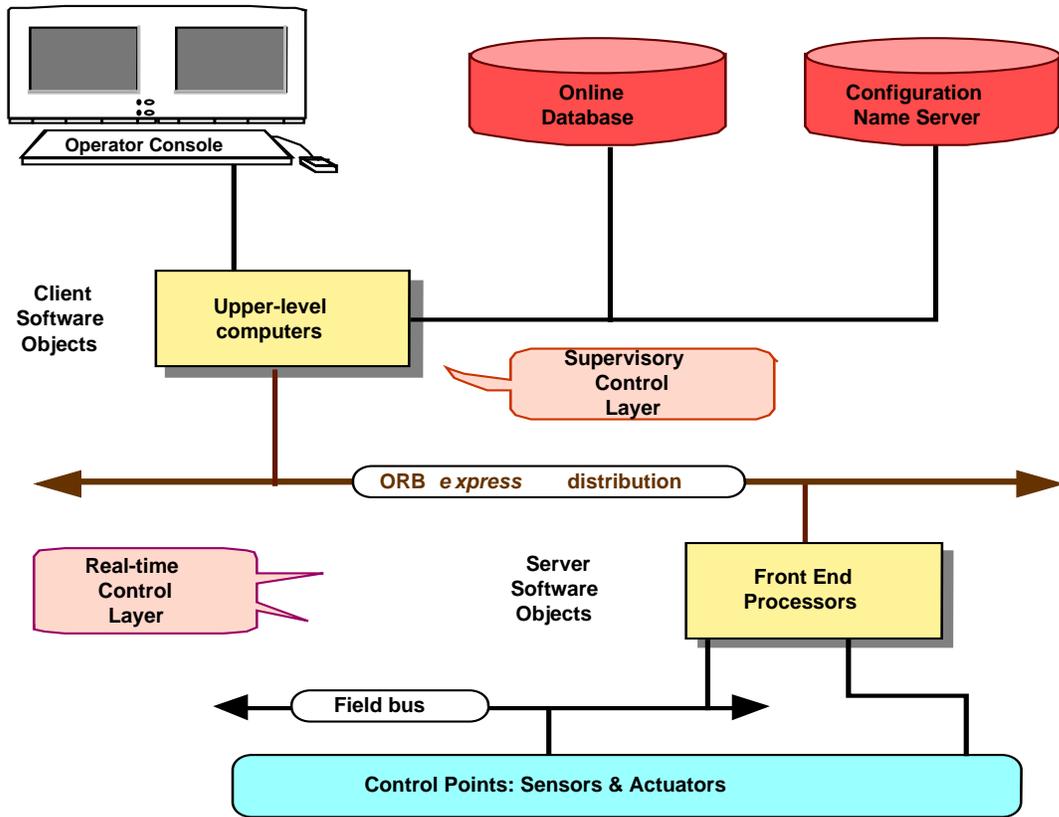


Figure 3 The Supervisor and Front-end layers communicate using CORBA

3.2 Front-end Layer

Front-end processors implement the distributed control functions of the ICCS by interfacing to the NIF control points. There are eighteen different types of FEP computers – some 450 computers in all – that differ by the devices that they control. A database provides the persistent repository of the data defining the FEP computers, the devices connected to them, and their configuration parameters.

Control points are sensors and actuators attached to interface boards plugged into a FEP back plane. In many cases, control points are handled by intelligent components that incorporate embedded controllers operated by small fixed programs. This firmware that runs in the embedded controller does much of the low-level work that would otherwise be allocated to an FEP. Example components implemented in this manner are stepping motor controllers, photodiode detectors, and power supply controls. Laser diagnostic sensors attach to FEP units by utilizing low-cost field bus micro-controllers.

The FEP software performs sequencing, instrumentation control, data acquisition and data reduction on control points that are collocated. The ICCS provides a standard way for FEP units to be integrated with the supervisory system. ICCS uses a common distribution mechanism coupled with software patterns for hardware configuration, command, and status monitoring functions.

4. Data Management Strategy

The ICCS has many complex data structures to represent the “real world” components that must be controlled, monitored, and stored. An object-oriented approach is used for the design and development of the control system. Figure 4 below depicts the development process for application software and database designs. Application requirements are modeled in the Unified Modeling Language (UML) [9] and subsequently built into code. Database requirements are modeled with entity relationship diagrams [13] and subsequently defined in data definition language for the database management system.

The ICCS database holds information for several different purposes that can be classified as different classes of data management [12] as follows:

- **large vector or image data** - the UNIX File System provides the services of storage and access to files for data analysis programs. The data analysis programs qualify as “no query” applications and operate on files that are a sequence of bytes of arbitrary length. Current database technology for large objects (LOB) allows for storing and/or cataloging arbitrarily large files in the database.
- **tabular data with queries** - a Relational DBMS provides the standard query language and client tool kits. The relational data model provides a wide variety of data query and aggregation mechanisms which are important for predefined and adhoc historical analysis of shot archive information.
- **complex data without queries** - an object-oriented DBMS is well suited for management of complex data. An Object DBMS allows us to describe control system objects and their persistent data identically as they appear in the run-time control system. Access to this information is done at object creation time when the control system starts up, and updates are done through custom interfaces with little or no query capabilities. The database objects are defined by the object-oriented software design process.

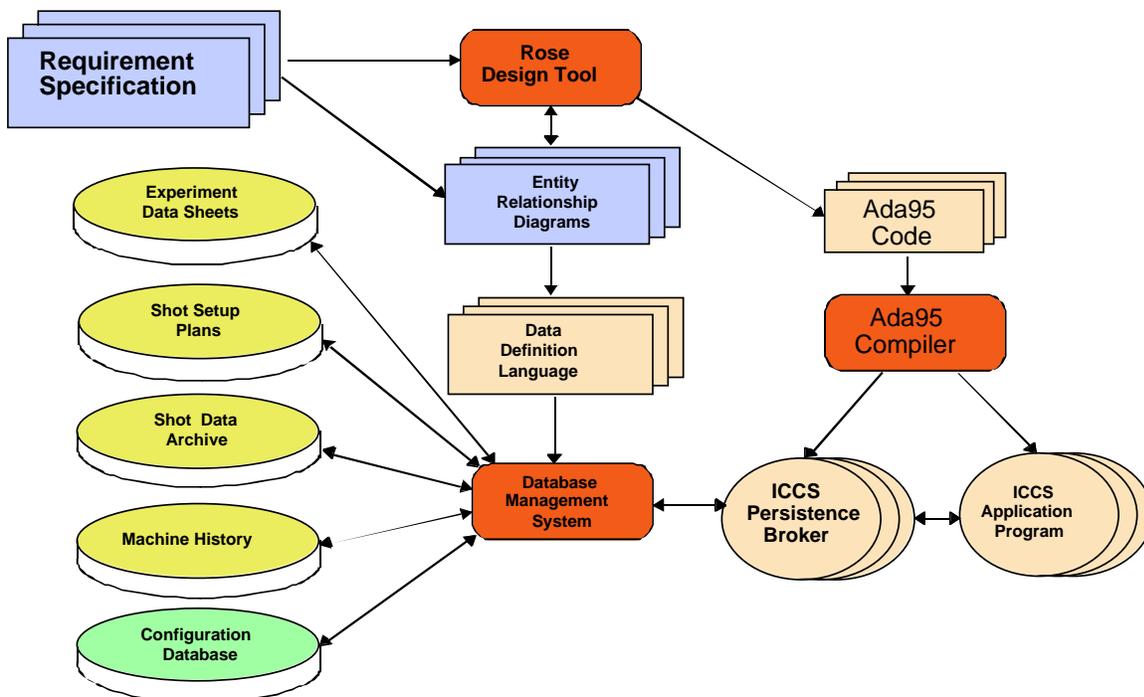


Figure 4. Database Designs are derived from the application requirements and the software design.

The data management requirements for the NIF ICCS encompass all of these classes of data. We have chosen the Oracle8 “object-relational” database management system (O-RDBMS) to address the NIF ICCS data management. The control system architecture is based upon object technology and will need object-oriented persistence to support its initialization and operation. In addition, the data archive components of the ICCS require a combination of relational, object-oriented, and large object features.

Oracle Corporation has a large market share for DBMS products and a large installed base at LLNL. We have found that Oracle8 contains many of the features that will help us solve the broad spectrum of the ICCS data management issues. Oracle8 supports user defined data types (objects), scalable on-line transaction processing (OLTP) performance, large object storage, and very large databases (> 100Gbytes).

5. Software Frameworks and the Database

We have developed a set of software frameworks that provide services and templates that are used to construct the ICCS application software. The frameworks are abstractions identified as common to most ICCS application software. We implement the abstraction once and deploy it as a framework service or template in our application software. Frameworks reduce the amount of coding necessary by providing pre-built components that can be extended to accommodate specific additional requirements. Software Engineers specialize the framework for each application to handle different kinds of control points, controllers, user interfaces, and functionality. The framework concept enables the cost-effective construction of the NIF software and provides the basis for long-term maintainability and upgrades.

Several of the framework components utilize the database as a tool for implementing different framework services. Application software is typically not aware of how framework services are provided or if the database is involved. A brief description of the framework components that utilize the database are as follows:

Configuration - The ICCS configuration schema contains the persistent data associated with the software objects that comprise the as-built control system. Examples include sensor calibration data, optical component characteristics, and device initialization data. Device initialization includes hardware location, CPU, bus address, channel, etc. This persistent data is generally extracted from the configuration schema when an object is created. The objects in the configuration schema map directly to software objects that implement the control system. This makes a common view of software objects and their associations in the object hierarchy for both persistent storage and run-time implementation.

Configuration provides a taxonomic system that is used as the key by which clients locate devices and other software objects in the CORBA domain. During normal operations, configuration provides to clients the CORBA references to all distributed objects. An important responsibility of configuration is the initialization of supervisory applications and front-end processors during start-up. During ICCS start-up, this framework collaborates with an object factory [15] located in the application program. Using the data and methods stored in the configuration schema, the object factory creates, initializes and determines the CORBA reference for each public software object in the application.

System Manager – provides services essential for the integrated management of the ICCS network of hundreds of computers. The database schema for the System Manager Framework identifies all processes and processors in the ICCS network and is used to keep track of the state and health of all supervisory application programs and FEPs.

Message Log – provides message archiving services to all subsystems within the ICCS. A central server collects incoming messages and associated attributes from processes on the network and writes them to appropriate persistent stores. Interested observers can view messages on operator consoles or produces historic audit trails of system operation.

Alert System – any application encountering a situation that requires immediate attention raises an alert, which then requires interaction with an operator to proceed. An alert manager program validates and records alert transactions in the database so that data can be analyzed after the fact.

Reservation – manages access to devices by giving one client exclusive rights to control or otherwise alter the state of the device. The framework employs a lock manager and a lock-and-key model that utilizes the database for tracking device locks and keys. Reserved devices that are “locked” can only be manipulated if and when a client present the “key”.

Experiment Data Sheets - The experiment data sheet schema typically identifies a target diagnostic and a desired set of inputs that can be measured by the diagnostic. Each active diagnostic on a shot has a corresponding experiment data sheet. The experimenter builds experiment data sheets that subsequently are to be allocated to specific shots. A single NIF shot may accommodate multiple experiment data sheets.

Shot Setup Plan - The Shot Setup schema contains desired shot characteristics and the configurable parameters for the devices that will be used in a shot. When a device is to be configured for a specific shot, the desired shot characteristics are analyzed to determine the settings necessary. These settings are communicated to the device in preparation for the shot. For example, The Laser Diagnostics Beam Balance process uses a model of the laser energy gain profile to calculate settings for devices to be used on a laser shot of a specific desired energy. The calculated settings from this model are stored in the Shot Setup schema and extracted when the actual devices are configured to those settings.

Shot Archive Data and Shot Archive Files - The Shot Archive data contains a snapshot of the control system objects at and closely around shot time. This includes the Shot Setup schema and any shot data acquired as a result of the shot. Each ICCS Supervisory System has some form of data archive process. The archive data process gathers subsystem results and stores them in the Shot Archive schema and Shot Archive Files. The shot data are used in off-line analysis to characterize the performance of NIF experiments. Image data and other large array data are stored in Shot Archive Files and accessed by various off-line data analysis programs. The Shot Archive Files will use a standard, common file format such as HDF.

Machine History - The Machine History schema is used to track history of component performance and assist with maintenance activities in the NIF. Periodic maintenance (PM) schedules will be required for many components and the Machine History schema will trigger reports that notify operations staff of required PM activities. Component failures and usage history are stored for failure mode analysis and reliability, availability, and maintainability (RAM) statistics. Examples of such information are abnormal conditions, operating service time or usage count, periodic readings of sensors, and alignment reference images.

5.1 Persistence Brokers for Database Services

In order to provide all of these database services to thousands of distributed objects on hundreds of computers, we have employed database server programs that act as persistence brokers[11][14]. They provide the run-time interface between the control system and the DBMS. These persistence brokers hide the DBMS specifics from the application software and implement a consistent, well known interface. One or more persistence brokers is built for each identified framework that utilizes the database to accomplish its function. The Standard Query Language (SQL) is used as the interface between the persistence brokers and the DBMS. Use of SQL serves to insulate the ICCS from changes in the database management system.

The persistence broker programs are constructed as state-less processes. Any state information that the persistence brokers require is obtained from the database. As the state information changes, the database is updated. This allows us to start/restart these programs as necessary without having to relearn the state of the control system. Because of the robust concurrency controls built into the Oracle8 universal server we can employ arbitrarily many of these state-less brokers running on the same database.

Addition of new classes and their associated persistence is easily accomplished when using persistence brokers. We employ template software that is used to build the persistent subclasses. Each persistence broker can support one or more persistent base classes. These base classes are extended by application programmers to implement specific persistent behavior. This is in keeping with our software framework concept. All objects that require persistence have an object identifier or taxon that uniquely identifies the object requesting database services.

Cursors can be employed by the persistent subclasses when the need to process many objects with a single command is necessary. The database administrator provides oversight of the use of cursors in persistent subclasses so individual clients are not allowed to appropriate database resources for extended periods in the runtime system.

Communication between the distributed application objects and the persistence brokers is accomplished using CORBA. Persistence brokers are constructed from SQL interface components, CORBA components, and the IDL (Interface Definition Language) service definitions as depicted in Figure 5. The IDL service definitions coupled with the persistent subclass definitions, give us complete control of what application objects can request from the DBMS and how that impacts the performance of the control system.

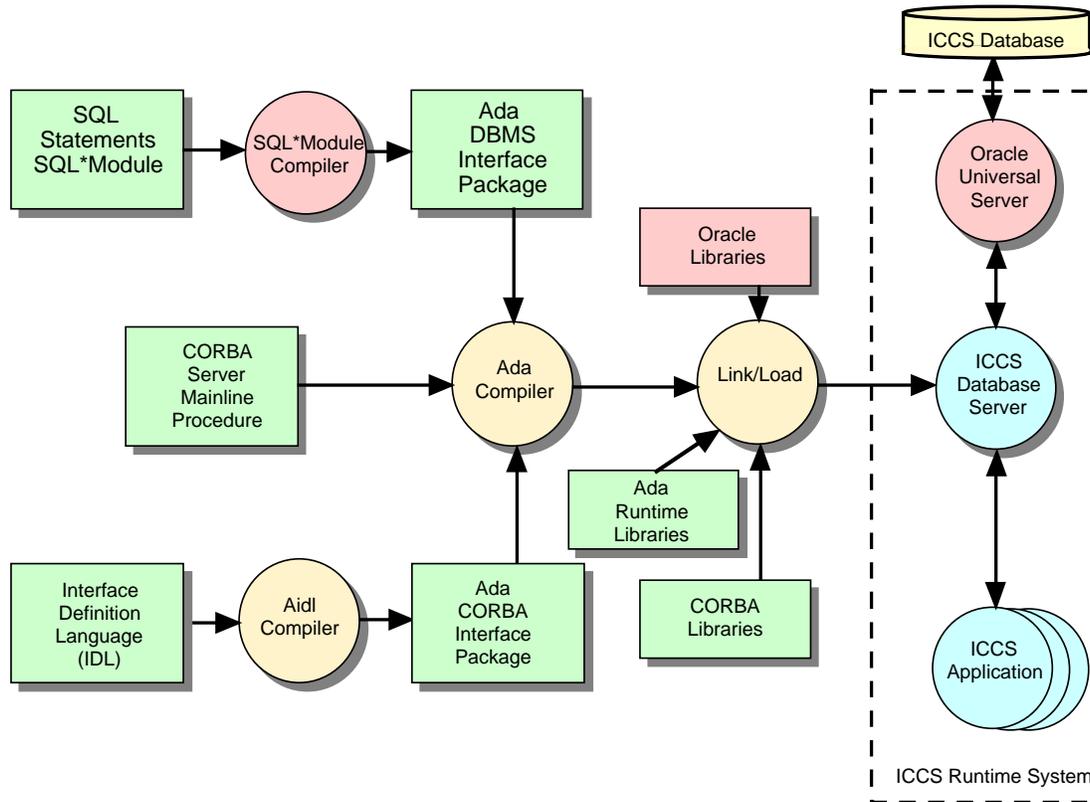


Figure 5 Construction of ICCS Persistence Brokers.

5.2 Object Pooling

The persistence brokers are implemented as multithreaded CORBA servers and multithreaded database servers. Each persistence broker can be configured with multiple concurrent threads (worker tasks) of CORBA event loop processing and multiple concurrent threads (database sessions) of database processing. Traffic determines the number of threads utilized within each persistence broker. An object pool [3] of database sessions is managed by each persistence broker. The database sessions are initialized (login and context area allocation) when the persistence broker program starts up. Thus, there are no delays to establish connections to the DBMS for clients of database requests. Database logins can be local and/or remote depending on the processor where the persistence broker is started. Oracle's Net8 handles this transparently.

A session object is allocated to a client database request on demand and released back to the session pool upon completion of the database service. If all sessions are busy, the next client must wait until some request completes. We are able to optimize use of our database user licenses using this technique. Thus we can provide database services to 10's of thousands of CORBA objects distributed over hundreds of computers with a small number (32) of Oracle user licenses. Since most of the database requests in our application are atomic transactions, client requests are handled quickly thereby only requiring a database session from the session pool for a brief time. As performance dictates, we can add database user licenses incrementally and distribute them to the database server programs that are experiencing excessive client delays for database services.

6. Future Developments

The persistence layer software we have developed for the ICCS encapsulates access to the DBMS. Database session management, persistent base classes, and error processing are all performed by the persistence layer. This allows us to incorporate upgrades, port to other persistence mechanisms, and make connections to other databases independent of the ICCS application software.

With the incorporation of a Java ORB and Java language support within the Oracle8 Universal Server, we will have several different options for migration of the design of our persistence brokers. Use of custom data cartridges, Java stored procedures, and JDBC all have potential in our application.. In addition, as the CORBA product suite matures, we will also examine commercial implementations of the CORBA Persistent Object Services Specification. These future developments will be scrutinized with an eye toward extensibility, performance, and isolating dependency on any of the commercial products and tools used to produce our ICCS application software. This includes Oracle.

7. Acknowledgements

I acknowledge the contributions of my colleagues without whose efforts this work would not be possible: R. Bettenhausen, T. Dahlgren, F. Deadrick, R. Demaret, C. Estes, K. Fong, M. Gorvad, F. Holloway, J. Jones, C. Karlsen, R. Kyker, L. Lagin, G. Larkin, P. McKay (Sandia National Laboratory, Albuquerque NM), M. Miller, V. Miller Kamm, C. Reynolds, R. Reed, R. A. Saroyan, W. Schaefer, J. Spann, E. Stout, W. Tapley, P. Van Arsdall, L. Van Atta, S. West, and J. Woodruff.

8. Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade

name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government of the University of California, and shall not be used for advertising or product endorsement purposes.

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

9. References

- [1] E. Michael Campbell, Neil C. Holmes, Steve B. Libby, Bruce A. Remington, and Edward Teller “The Evolution of High-Energy-Density Physics from Nuclear Testing to the Superlasers” *Laser and Partical Beams* 1997, vol. 15, no. 4, pp 607-626.
- [2] Keay Davidson “From Swords to Supernovae”, *Sky and Telescope*, November 1997, p. 36.
- [3] Mark Grand “Patterns in Java Volume 1 – A Catalog of Reusable Design Patterns Illustrated with UML,” John Wiley & Sons 1998, http://www.mindspring.com/~mgrand/pattern_synopses.htm
- [4] William J. Hogon, Roger O. Bangerter, and Charles P. Verdon “The Fire Next Time,” *The Sciences*, Vol. 36 No. 5, September/October 1996, p 20.
- [5] National Ignition Facility webpage <http://lasers.llnl.gov/lasers/nif.html>
- [6] Object Management Group “The Common Object Request Broker: Architecture and Specification” John Wiley and Sons 1995.
- [7] J. A. Paisner and J. R. Murray “The National Ignition Facility for Inertial Confinement Fusion,” 17th IEEE/NPSS Symposium on Fusion Engineering, San Diego CA, October 6-10, 1997.
- [8] Ted Perry and Bruce Remington “Nova Laser Experiments and Stockpile Stewardship,” *Science and Technology Review* September 1997, <http://www.llnl.gov/str/Reminton.html>
- [9] Martin Fowler and Kendall Scott “UML Distilled – Applying the Standard Object Modeling Language” Addison-Wesley, 1997.
- [10] Paul Van Arsdall, and Arnie Heller “Controlling the World’s Most Powerful Laser” *Science and Technology Review* November 1998, <http://www.llnl.gov/str/Vanarsdall.html>
- [11] Scott Ambler “Robust Persistence Layers,” *Software Development* February 1998. p73.
- [12] Michael Stonebraker “Object-Relational DBMSs The Next Great Wave,” Morgan Kaufmann Publishers 1996.
- [13] Richard Barker “CASE*Method Entity Relationship Modelling,” Addison-Wesley 1990.
- [14] Craig Larman “Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design,” Prentice Hall 1998, p455.
- [15] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides “Design Patterns – Elements of Reusable Object-Oriented Software,” Addison-Wesley 1995, p81.