

Technology Infrastructure

Research and Advisory Services

Data Management

RESEARCH PAPER

Object Design Object Store

Abstract As database sizes grow to accommodate the proliferation of data within organisations, so a potential bottleneck is created in the access of data from the database. This is intensified as the number of users increases, due in no small part to the opening up of back-end systems to partners, suppliers, and customers. A number of techniques have been employed to alleviate this bottleneck, including: clustering; load-balancing; and mirroring. Object Design uses another approach with its object database, ObjectStore. Rather than access the database directly, it uses a middle-tier comprising local data caches, with which users interact. Only changes to the data are written back to the database.

► TARGET MARKETS

ObjectStore is ideally suited to large enterprise organisations that have a need to store very complex file types, want access to real-time information, and require a high performance rate. ObjectStore's patented cache-forward architecture provides a distributed persistent caching mechanism that enables middle-tier data access at in-memory speeds.

ObjectStore has been successfully adopted within the telecommunication market, with its aggressive performance and availability requirements, and 60 per cent of the company's 4,000 plus customers are in this area. However, the growth of e-business, with its requirement for rapid access to information, has broadened the appeal of ObjectStore to include e-business companies, financial institutions, and the embedded software market. At the same time, the requirements of the telcos have also changed to reflect the Internet economy, and they are becoming more mainstream in their requirements, resulting in a convergence between the needs of telcos and those of general e-business-focused organisations.

ObjectStore is sold directly by Object Design, which also offers professional services to help with the implementation, as many of the projects are very complex. The database is also sold under Original Equipment Manufacturer (OEM) agreements.

The Object Model

Over the last ten years there has been a rapid paradigm shift in development techniques. To better manage the complexity and rapid development cycles of today's environment, object-oriented design and development techniques were developed. The main features are powerful data modelling characteristics and the potential to build software components that are reusable, modular, and extensible.

Today's paradigms have also evolved from the once standard client-server based architectures to that of a distributed or multi-tier model. Distributing application logic across a middle-tier combats the performance and scalability limitations of a client/server architecture. Application servers that provide developers with a development and deployment environment to take advantage of an *n*-tier architecture are now commonplace.

ObjectStore's cache-forward architecture is designed for maximum application performance in an n -tier architecture through load-balancing, cache affinity, transaction services, and overall component co-ordination and management. ObjectStore provides efficient data caching and storage in a middle-tier scaling access to data along with application logic, removing any potential bottlenecks to central back-end data sources.

One of the major issues for companies, such as telcos working in extremely competitive fast-moving markets, is time-to-market. Large development projects traditionally took years rather than months to complete, something that is now unacceptable – an application written now could be superseded within months.

One of the main advantages of ObjectStore is that it promotes the reuse of objects, in that they do not require reassembling from their component tables each time they are used. This reduces processing overheads by increasing processing speeds. ObjectStore natively supports C++ and Java. C++ has almost become a de facto standard, and the use of Java, with its 'write once run anywhere' capability, means that, in theory anyway, applications written for ObjectStore should port easily between platforms.

Improvements in efficiency can also be made, due to a reduction in the need for paging, as only objects that are required at a particular time are loaded into memory.

Cache-Forward Architecture

At the time of its inception, ObjectStore was revolutionary in its approach, and incorporated one of the earliest examples of a middle-tier long before multi-tier paradigms became the norm. The biggest bottleneck for a database system can be at the database server-end, where multiple requests are received. Although technologies such as load-balancing and clustering can reduce the amount of traffic being received by a single server, performance can still be an issue.

Object Design adopts a different approach with ObjectStore. Instead of users accessing the database directly, it has a middle-tier, comprising local data caches. This allows distributed applications accessing data from ObjectStore to do so locally without the need to go back to a central data server. All of the required data is pushed out to applications that need it, effectively providing data access at in-memory speeds.

An immediate possible pitfall with this approach can be ensuring the integrity of data, especially if several users accessing different caches attempt to perform simultaneous updates on the same piece of data. ObjectStore deploys a lazy token mechanism for ensuring data integrity. When a request to update data is received it is sent to the database server, which will then issue a token to authorise the update. Once complete, the token is revoked, and each cache is updated with the new data.

A weakness with this approach is that users could view an out-of-date record, although this is the case with any type of database where some form of locking mechanism is deployed, and Butler Group does not believe that any company has yet managed to solve this particular problem. However, with ObjectStore, users will never view dirty data.

Because network traffic between the end-user and the server is very low, the impact on performance of scaling the solution is minimal. It is very simple to add additional caches to the implementation and high availability and failover are supported.

This lack of traffic also means that once cached, data can be accessed at real-time speeds. Enterprise databases are generally very large, which can impact on data retrieval times, especially as a search of many millions of records may be required to retrieve a single row. Although some database systems will retain in cache a requested record, this will be lost as soon as the session that requested the data ends.

ObjectStore's cache mechanism manages the available cache. Therefore, if there is not sufficient to store the entire database, only the objects that are requested will be loaded. Each cache will most likely only contain a subset of the whole database, further increasing performance rates. Cache-Forward technology allows distributed components to access data that is persistent, from objects that survive beyond the lifetime of the processes that created them at near in-memory speeds.

A master/slave configuration can be utilised for both the database server and the caches, whereby if the master server fails the slave will automatically take over the workload. At present, clustering facilities are not available, which Butler Group regards to be a weakness, although Object Design hopes to rectify this situation in the future.

Caches can be arranged in a number of ways. Multiple caches can be used for load-balancing purposes. They can also be configured to contain single or a number of applications, and can provide access to specific users.

Data stored within a Relational DataBase Management System (RDBMS) requires complex mapping code writing to translate the object model to traditional, relational, or sequential data models. This invariably leads to increased development times and time-to-market. This requirement is eliminated in ObjectStore as data is managed in a component-ready form and direct bindings are provided to Java and C++.

In fact a common deployment scenario for ObjectStore is as a transactionally consistent middle-tier cache sitting on an existing back-end RDBMS, providing data in a component-ready format to the distributed applications that use it, and relieving the back-end RDBMS from direct access.

Functionality

Business-critical systems and Web-enabled applications require 24x7 availability and high levels of reliability and performance. ObjectStore provides a number of safeguards to protect data within a distributed environment. These include concurrency control for transactional consistency, on-line back-up, roll-forward, automatic failover, and replication.

A potential danger point for any data management application is a system crash during a transaction. ObjectStore's recovery mechanism enables the database to be restored to a consistent state following a crash. XA-compliant, two-phase commit is used to ensure consistency across distributed servers and multiple databases. Following a system crash, the transaction log files are processed to restore databases to the last consistent state prior to the crash.

An added benefit of writing all changes to the transaction log and using the log for subsequent data access, is that fast response times during transaction commits can be ensured. If the database server is busy servicing requests from clients, the transaction log can wait before committing changes to the database, eliminating the risk of a bottleneck in the database server.

An object-oriented application's most common operation is normally object dereferencing, which in C++ is carried out using pointers, and in Java with references. In both cases the key to database performance is the speed with which the dereferencing can be performed. ObjectStore uses patented technology in the Virtual Memory Mapping Architecture.

ObjectStore takes advantage of a workstation's virtual address space by dynamically mapping referenced persistent data during application sessions, thus ensuring that access to persistent data is as fast as access to transient data. Very often, applications reference large numbers of small objects. As it normally takes one Central Processing Unit (CPU) instruction to retrieve a single object, this is an inefficient utilisation of resources. ObjectStore allows several objects stored on a single page to be requested in one CPU operation, placed in the cache, and mapped into virtual memory.

Wherever possible, objects are stored on the server in the same format that they are presented in virtual memory to improve efficiency. However, in order to allow heterogeneous access across different platforms, some measure of reformatting is normally required, for example, byte order switching.

Another way of improving efficiency, which is adopted by ObjectStore, is the ability to cluster objects on a page or cluster pages, so that when data is placed in cache, access to associated data is achieved speedily.

In an RDBMS, a level of Structured Query Language (SQL) knowledge is often required to produce ad hoc queries. In reality this probably means putting in a request to the IT department, which can entail a delay of several days waiting for a response – hardly ad hoc. Facilities are provided in ObjectStore for modelling relationships between objects, allowing large collections of objects to be managed. These collections form the basis of querying with users simply selecting the elements that satisfy a specified condition. Ad hoc associative and navigational retrievals are also supported.

A factor that can have a serious impact on the performance of a database is a lack of efficient indexing. However, the creation of indexes themselves can create physical space problems. Therefore it can be a fine balancing act between creating appropriate indexes to speed-up data retrieval and optimising the available space. ObjectStore eliminates this potential pitfall through the support for ad hoc indexing, which includes the ability to add indexes to attributes.

Development and Integration Tools

Any data management system can be judged by the ease with which applications can be developed, and ObjectStore includes a comprehensive set of rapid application development tools that support component-based development. These have been engineered to speed development, reduce time-to-market, increase flexibility, and minimise the maintenance burden of developing new applications using Object Design's products. They also allow the seamless integration of existing enterprise applications with ObjectStore.

These tools include Database Designer, a visual tool that allows developers to easily define an object model, comprising classes, data members, methods, and relationships. Inspector is a cross-platform tool that provides a monitoring facility. Objects and object models can be viewed in a tabular format, which can then be published to a SQL interface and embedded in Microsoft applications, with ObjectStore managing the mapping process.

► VENDOR PROFILE

Object Design, a division of eXcelon Corporation, is a leading provider of object data management solutions for eXtensible Markup Language (XML), Java, and C++ based applications. Headquartered in Burlington, Massachusetts, the company was founded in 1988, by Tom Atwood, considered by some to be "the father of object-oriented databases." The company completed its Initial Public Offering (IPO) in 1996.

The company has consistently reported double-digit growth in an area of the IT industry where the original ODBMS vendors have virtually disappeared. Revenue has grown from \$60.8 million in 1999 to \$70.3 million in 2000.

Object Design sells and supports its products via branch offices throughout the US, international subsidiaries in the UK, Germany, The Netherlands, Japan, and Australia, plus a global network of distributors. The company employs over 450 people worldwide.

Customers include: Alcatel; AT&T; BT; Deutsche Bank; Goldmann Sachs; Misys; amazon.com; Tour de France; Kodak; Cognos; OCE; Xerox; and Intel.

► CONTACT DETAILS

Object Design

Bldg. 1015, Arlington Business Park
Theale
Reading
Berkshire
RG7 4SA
UK

Tel: +44 (0)118 930 1200

Fax: +44 (0)118 930 1201

Important Notice:

The information available in this publication is given in good faith and is believed to be reliable. Butler Group expressly excludes any representation or warranty (express or implied) about the suitability of materials in this publication for your purposes and excludes to the fullest extent possible any liability in contract, tort or howsoever for implementation of, or reliance upon, the information contained in this publication. All rights reserved. This publication, or any part of it, may not be reproduced or adapted by any method whatsoever, without prior written Butler Direct Limited consent.