Towards An Improved Evaluation Metric For Object Database Management Systems

Somnath Banerjee & Charles Gardner
Texas Instruments
email: sbanerjee@ti.com, phone: (214) 575.5953

Abstract

With the emergence of Object Technology, Object Oriented Database Systems are becoming increasingly important. However, there is very little information for users to evaluate these databases from the perspective of their own requirements. Based on the experiences of both traditional and Object databases, it is strongly felt that several features other than the mere performance are equally, if not more important in order to evaluate a Database system. These features include ***backup/recovery, online database compaction, multiuser support, clustering, security and automated monitoring.*** This paper first introduces the current benchmarks, discusses their limitations, then the so called ***7 by 24*** features are introduced for inclusion in a future OODBMS evaluation metric.

1. INTRODUCTION

We are witnessing a very interesting transition period. More and more computing environments are making the shift from monolithic mainframe architecture to a distributed Client/Server architecture [1]. Another key feature of this transition is the migration to Object Technology from traditional computing paradigms. Texas Instruments has been a pioneer in such migration. The [1]**MMST/WORKS** program was one of the first Object Oriented Computer Integrated Manufacturing system. TI is in the process of rolling mission critical applications based on this technology to production. By the end of 1996, key wafer manufacturing plants will make the transition to a complete Object Oriented System. By the end of 1997 almost all TI Fabs [22 in number WorldWide] will experience this migration.

A key component of this change is an Object Oriented Data Base. The architects of this system realized early that a complete Object Orientation is desirable from the Client GUI end to the persistent object storage end [2]. In the process of this development and also as a result of prolonged interaction with the developers/maintainers of the traditional systems the authors of this paper have discovered what a **PRODUCTION OODBMS** should look like. This paper emphasizes features which seem to be absolutely indispensable to support a ***24 by 7 by 365*** manufacturing site where a downtime of one hour can translate to a loss of business of millions of dollars.

Object databases are a fairly recent technology. They were first commercially available in the year 1988. This technology, even though maturing very rapidly, still has ways to go in order to achieve full industry strength. One of the key things which is lacking is an evaluation metric for OODBMS. While performance benchmarks are important, a complete metric should take both features and performance into account in order to evaluate a database. We feel that such a metric would not only help a group identify an OODBMS when they are trying to make the transition like ours, but such metrics would shape the direction of present OODBMS industry to help it become more mature, more robust and eventually more acceptable. In this paper the authors try to briefly introduce the current benchmarks, their shortcomings, and outline KEY items to consider in order to come up with a set of metrics which could be applied to evaluate an OODBMS. These items are a result of direct experience of OODBMS based development and deployment.

2. CURRENTLY AVAILABLE BENCHMARKS

There are 3 currently available benchmark and performance measurement metrics. They are

- Object Operation One
- Hypermodel benchmark
- Object Operation Seven

Object Operation One [3] was the first of its kind and it provided excellent insight into the performance of OODBMSs vs. relational DBs highlighting the effects of pointer navigation and lack of foreign key

references. Hypermodel benchmark provided a richer schema and wider operations. Finally, Object Operation Seven [4] introduced several key notions like complex objects, indexed vs nonindexed transactions, traversals with updates and so on. This benchmark has been both widely used and controversial among the OODBMS community .

However, all of these benchmarks are focussed on evaluating the database performance while running an application. They tell us about speed of traversal, update, creation and deletion of objects in various forms of complexity. This information, while being extremely valuable, is not sufficient to be used as a metric to judge an OODBMS which is intended to be used for a production system with high uptime.

3. SHORTCOMINGS OF THE AVAILABLE BENCHMARKS

These benchmarks keep several questions unanswered. We present the ones which seem to be of extreme importance in our context. Some of the these are easy to quantify where as others are extremely feature oriented in nature and hard to quantify. For those which seem obvious, an attempt has been made to quantify the attribute.

3.1 Multi User Workload

First of all, none of these benchmarks specify a multiuser environment and multi user workload. Effects of loading up an OODBMS with 100- 150 sessions [with 10-15 active concurrent sessions] will not only bring out the ability of the database to handle the workload but also throw light on the concurrency semantics of the database. Please note that the number of sessions used here are arbitrary.

3.2 Reorganization/Clustering

How efficiently can the objects be clustered so that all related objects are close together. This would ensure locality of reference and hence aid performance. This can be quantified by answering the question *"How long does it take to cluster a database of size X gigabytes with Y number of objects ?"*

3.3 Integrity Checking on the Database

Can the database perform integrity checks without impacting transaction throughput ? Can it check for corrupted objects and repair them? If it can do all these, how ***nonintrusive*** are such operations ?

3.4 Database Compaction

How efficiently can the database handle unused objects and reclaim the space for future use ? Can it discover unused or fragmented pages and compact them ? This operation should not demand the database to be taken offline or cause the currently logged sessions to experience a performance degradation. This can be quantified by answering the question *"How long does it take to compact a database of size X gigabytes ? What is the performance degradation on transaction throughput ?"*

3.5 Security

Can it plug in an external authentication system to the database? Instead of using the database provided userid/password authentication, the application site should have the flexibility to use an external authentication system like Kerberos to override the database authentication system. Object level authorization should be provided to make sure that certain classes or instance variables or instances can only be accessed by sessions with the proper privilege assigned to them. This might be purely a feature and could be difficult to quantify.

3.6 Automated monitoring of the Database

How easy is it to build agents to monitor database activities eg. size, freespace, number of commits etc. The agents should not only detect events, but should be able to take corrective actions in case of a problem. For example, when the database is running out of space the agent should send a red alert and also extend the repository by identifying another free filesystem or partition. This again will be a hard to quantify item as most database would have one or other form of monitoring.

3.7 Versioning Of Objects

Does the database provide versioning of Objects ? This is very key in maintaining history information and supporting activity based cost and other MIS applications.

3.8 Object Migration

How easy is it to perform object migration and schema evolution? One of the key promises of Object Oriented Paradigm is ease of maintenance. In the case of a production system this translates to modification of only selected methods and classes instead of recompiling the whole system. The database should also provide a very smooth transition of the schema and objects to facilitate this feature. The question is what impact does object migration/schema evolution have on avilability. This could be quantified by answering the question "*How long will it take to migrate a database of size X gigabytes with Y number of objects ? Does the database needs to be taken offline for such operation? "*

3.9 Multi database support

Can the database be designed to integrate with other databases ? Legacy systems still implement most of application components. Many architects are willing to migrate to an OODBMS solution but frozen because of the potential loss of a seamless access to relational tables. One method of quantifying this may be to define a relational table with standard access patterns. Measuring the access time of the contents of the relational table from the OODBMS space could be a metric of multi database support.

3.10 Very Large Data Bases

With the information age experiencing an exponential explosion, architects are dreaming of huge databases. How does the database scale up to sizes of the order of 40-50 gigabytes ? Are features like cursors implemented to support huge database queries ?

3.11 Optimization Of Queries and indexing

What are the features for query optimization and indexing of collections ? How difficult it is to maintain the indices? What is the cost of using indices ? The cost of building and maintaining indices could be easily built on top of an existing benchmark like OO7.

3.12 Online Line Backup / Fastest Recovery

The current benchmarks does not tell us about the capabilities of online backup, fault tolerance of the database (ie. keeping the application transparent to a disk or system crash). One of the toughest challenges provided to us is RECOVERY. In case of a crash, the database has to be brought online within an hour's time. This might be a very tall order when we are talking about a database greater than a few Giga bytes in size. This can be quantified by answering the question ***"How long does it take to recover a database of size X gigabytes?"***

4. OUR EXPERIENCE

4.1 System Description

During pre-deployment stress testing of a distributed reporting (**DR**) application we had the opportunity to closely examine the factors within the database which affected the performance of the application. The DR application has an OO model of a mainframe manufacturing system. As transactions happen on a mainframe, messages are sent to the OODB via communication link between the mainframe and the OODB. The database keeps information about initial, current and previous work in progress. The DR application is being deployed on either **SparcCenter 1000's** or **Sparc 20's.**

The database can range from 150 MB to 350 MB in size. The stress tests were conducted with a 350 MB database. The collection sizes maintained for reporting were between 12K and 15K. The primary collection had 5 sets of indices which are database provided and two others which are application provided. The SS1000's run approximately 40 users including feed programs and related servers. Transaction feeds from the mainframe are at 30/40 per minute on average, with highs at 70 to 80

transactions/min. This does not count reporting transactions which averaged 4 - 5 concurrent users at any one time. The system was tested at 30 concurrent logged in sessions. This application used the Gemstone Database from Gemstone Systems Inc.

Texas Instruments has been using the Gemstone database to implement and deploy various other applications other than the one used in this discussion. Many of these, including the DR application is running production in a 7 by 24 fashion in one or more wafer fabs. Our architects feel that this would have been an extremely difficult task using any other commercially available OODBMS with our architecture. In the next section we outline some of the features which surfaced to be very important for our operations.

4.2 Lessons Learned

4.2.1 Multiple User

During multiuser testing we needed to be able to measure the amount of process interaction with each other and with the shared page cache. This interaction was very important in tuning the database activities such as clustering and scavengeing to the required response time of the end user processes. Collection of these data points allowed us to make necessary changes in either code or policy to get the least amount of disruption of the reporting processes. Statistics provided by the database allowed monitoring of these data items.

4.2.2 Clustering

As a result of updating objects, clustering was being done to attempt to keep heavily used objects together. This activity have the potential to create two problems. If the clustering of objects was done by a separate process from the normal update process, concurrency problems would occur. One process or the other would get a failure to commit due to this action. If we could always be sure the cluster operation failed and not all the other update processes this might be acceptable. The other problem is created if the clustering is done by the same update process. The commit record for the commit in which the clustering was done tends to be very large and caused a disruption in the page allocation for all other processes.

4.2.3 Integrity Checking

Integrity checking could not be run on a live database. Object audits required that no one be logged in, which means that we would have to shutdown the database. Currently the only solution is to run object audits on a backup of the production database.

4.2.4 Compaction

Database maintenance activities such as de-fragmentation of the database and physical shrinking of the database files needs to happen with little or no impact to running processes. Typically, these activities create commit conflicts or are required to be run when no one is logged in. Neither mode is very acceptable. In the case of the activities which move objects from one page to another, the transaction throughput is affected by these types of activities. Perhaps conceptually, moving objects from one page to another is the same type of activity within the database but should not create concurrency conflicts between application processes and database processes.

Statistics supplied by the database allowed us to monitor the amount of garbage produced by each process and the amount of garbage collected by the epoch garbage collection process. We were able to tune the epoch with the use of these statistics. We were also able to detect some flaws in design of our feed application in terms of excessive garbage production using methods and processes supplied by Gemstone Systems. We had to code in the statistics collection if we wanted to measure transaction boundary data vs time based data but those calls were commented out when in production.

4.2.5 Security

Security within the database is implemented using the internal segment based authorization mechanism provided by the database. It allows protection at the object level and was relatively easy to implement. External authentication mechanisms to override the database provided mechanisms should be provided to implement corporate wide security to ensure protection of highly sensitive data.

4.2.6 Monitoring

The basic functionality to monitor the size of the database and related parameters were available. We added some scripts to automate the activity and monitoring of database size. The majority of the work was done using awk scripts and cron task which were required to log onto the the database and append information to a text file.

4.2.7 Object Migration

During releases of new object defintions, the migration of objects within the database was found to be relatively minor. We did not keep old objects around in any production database but were required to migrate some information which could not be reloaded or was too time consuming to reload. Once methods were written to handle each migration level they could be built into the load scripts.

4.3 General remark

The experience and examples provided in this section may be too specific to the nature of the application which we cited. Some of these may or may not be applicable to provide the future direction to the industry (which is the goal of this article). Also it should be remembered that Object Databases are still in their infancy and are undergoing a process of continous refinement. Overall, our experience has been quite positive with our early decision to select an Object database.

5. TOWARDS A FUTURE OODBMS METRIC

Some work has been accomplished to capture some of these features and perform a comparison [5]. However, the comparison seems to be subjective and lacks the preciseness presented in OO1 and OO7. There is no quantification of the features or capabilities. We propose that some kind of quantification be assigned to each of these capabilties so the ratings of the databases can be more objective in nature. A proper metric should very precisely define how to measure each of the features suggested in section 3 and experienced in section 4. This metric would then be truly reflective of a productionworthy database with 24 by 7 by 365 operation.

6. CONCLUSIONS

This paper emphasizes the features of a production Object database of the ***future***. Future work should attempt to carefully define and quantify these items in order to capture it in the form of a benchmark. If the academic community develops a model to quantify these features eventually the industry will come up with a benchmark and metric to evaluate OODBMS from the perspective of the aspects highlighted here.

REFERENCES

1. Thomas Atwood. Object Data Management: What's coming? *Object Magazine July-August 1994.*
2. OODBMSs Gaining MIS Ground. But RDBMSs Still Own the Road. *Software Magazine November1994.*
3. R. Cattel and J. Skeen. Object Operations Benchmark. *ACM Transactions on Database Systems, 17(1), March 1992*
4. Michael J. Carey, David J. Dewitt, and Jeffrey F. Naughton. The OO7 benchmark. *In Proceedings of the 1993 ACM SIGMOD Conference on the Management of Data, Washington D.C., May 1993.*
5. D. Barry. ODBMS Feature coverage in the current market. *Object Magazine July-August 1995*

---

[1]MMST was a joint project between TI, ARPA and the Wright Laboratory to develop a complete Object Oriented CIM system to run a wafer fab. 1989-1993.