6 F − 6

Extending Database Space of Inada/ODMG for Very Large Databases on 64 Bit Workstations

Botao Wang, Akifumi Makinouchi, Kunihiko Kaneko
Graduate School of Information Science and Electrical Engineering, Kyushu University

1 Introduction

Since the computer applications become more complex, and the users become more sophisticated based on 32 bit environment, the move to 64 bit environment is inevitable. The gains from 64 bit computing environment can be summed up in three aspects: performance, precision and capacity, which should be reconsidered in the applications which manipulate multimedia, scientific, data warehousing databases. They are usually huge in the size. For the 64 bit architecture, **LP64** is chosen as solution by Open System community considering the portability, interoperability with 32 bit environment.

The Sequoia 2000 Project explored the application of emerging database, network, storage, and visualization technologies of Earth science problems, resulting in a "database-centric" metaphor for scientific computation. About Sequoia 2000 benchmark, one problems is that the data size is huge. For example, the national benchmark data is 16 GBytes; another problem is how to represent the complicated operation, such as POSTGRES operator "||", " < | >" and " * &*" defined in its queries.

INADA is an application platform for building databases with a database language for object management. We integrating ODMG93 C++ binding interface into INADA and call it INADA/ODMG. For the persistent virtual memory used by INADA, UNIX **mmap()** function is used mapping virtual memory from file. The current 32 bit INADA shows incapabilities while dealing with above problems, such as 2 GBytes file size limit. Since the largest virtual memory available to the users is 2 GBytes, the file larger than that size cannot be mapped. This means that very large database such as Sequoia 2000 cannot be handled by the current 32 bit INADA. The key features of our design is to make use of the gains from the 64 bit architecture, to setup 64 bit INADA/ODMG based our current 32 bit INADA/ODMG, and realize the Sequoia 2000 benchmark on it trying to overcome above problems.

2 Extending INADA Address Space

- Extend current ODMG data type while porting.

  In **LP64**(known as 4/8/8) model, **long** and **pointer** is 64 bit type, but in current **ILP32**(known as 4/4/4) model, **long** and **pointer** is 32 bit type. In ODMG, only 32 bit **Long** and **Unsigned Long** are defined. It's incapable while dealing with 64 applications.

  We think new data type should be introduced in ODMG, **Int** and **Unsigned Int** which is 32 bit and **Pointer** which is 64 bit; the width of **Long** and **Unsigned Long** should change from 32 bit to 64 bit. Without extension, the application based on ODMG are limited by its 32 bit.

- Make use of the capability provided by 64 bit in our system

  Address space has become vast on the 64 bit processor. Besides addressing more memory directly, the file size can be much more huge than current 2 GBytes limit; with this change, the disk management can also be improved. GBytes of data need not to be stored in different files any more and all the data can be accessed via a single file if necessary, which is one critical point for the performance of DB and OS.

The INADA/ODMG is built on persistent virtual memory and distributed shared memory on Network of Workstation(NOW). The basic storage unit is heap. Logically, the heap offers storage for objects without size limitation. But in the 32 bit implementation, one heap is less than 2 GBytes, system has to manage multi heaps distributed on multi sites. In our new design based on 64 bit architecture, the size of one heap can be much larger than 2 GBytes, which allows to get a file mapped on the very large virtual memory. Simply, one heap is used as one database file, which is one critical point for mass data storage, such as multimedia data and geographic information data. Compared with 32 bit implementation, the capability is increased; the management become more simple, the performance can be improved for one file accessing, especially in the case with mass data storage.

Besides above changing, another redesign is OID structure. OID represents the identity of of an object. In INADA, OID is composed of two parts, Heap Identifier(h_id) and Object Reference Table Identifier(ort_id), the size of h_id is changed from 8 bit to 16 bit, and the size of ord_id is changed from 24 bit to 48 bit.

3 Benchmark Design

We use Sequoia 2000 benchmark to test the performance of the new system . Based on object design method, the data classes **RASTER**, **POINT**, **POLYGON**, **GRAPH** and additional classes **BTree** and **RTree** are defined also. All the data are stored using **d_Collection**, such as the extent of POLYGON, which is defined as:

```
d_Set<d_Ref<POLYGON>>
```

According to ODMG, OQL allows method invocations with or without parameters anywhere the result type of the method matches the expected type in the query. Besides attributes definition in data class, which is same as the attributes definition in relationship database, the methods supporting kinds of queries are defined too. With the method the benchmark queries can be defined easily in OQL without system extension, especially in the case the system has special operation. For example, the following is one benchmark query in POSTGRES:

```
retrieve into FOO-2(POLYGON.all)
where POLYGON.location||RECTANGLE
```

which find alls the polygons that intersect a specific rectangle and store them in the DBMS. Operator "||" means polygon intersect rectangle. The same query can be written in OQL as:

```
select p
from p in polygon
where p.IntersectBox(RECTANGLE)
```

The method **IntersectBox()** is defined in class **POLYGON**. The result type of the above OQL query is $d\_Set < d\_Ref < POLYGON >>$. The operator " < | >" (polygon in circle) and operator " * &*" (point on graph) can be defined in the same way.

4 Conclusion

In this paper, we discussed properties of 64 bit environment, redesigned our 32 bit INADA/ODMG for 64 bit environment. The extension provides very large space for database system. High performance and capability are expected based on our design. The design has same interface as that of ODMG C++ binding. The size of built in data type in ODMG are upgraded and extended.