Performance of an object-oriented database at a HEP experiment

Akinori Kimura [a], Takashi Sasaki [b], Akihiro Shibata [b], Yoshinobu Takaiwa [b]
[a] *Niigata University, Niigata, Niigata 950-21, Japan*
[b] *KEK, Tsukuba, Ibaraki 305, Japan*

We have implemented a HEP data analysis software using an object-oriented database (OODB) system for AMY experiment at TRISTAN. The main purpose of this study is to explore the possibility of using OODB in HEP software; the key issue of object-oriented programming of the software is object-persistency which requires serious studies and performance tuning based on realistic prototypes.
In this study, Objectivity/DB is chosen as OODB. Performance and data sizes are compared between the original FORTRAN based analysis program and the OODB based one developed for this study. Also comparison is made for two different object database models.

Key words: Object Oriented Programming, OODBMS, Object Persistency,

1 introduction

Due to the growth of experiments of high energy physics in scale and complexity, software systems for them have become very huge and very complex, while further reliability and robustness are required. Most of recent projects have decided to use software engineering technique to cope with the reliability and maintenance problems. And the most popular one is the object-oriented methodology. Many of them already have tried to use the technique and confirmed that it's useful and help to solve the problems in HEP software development. In object-oriented programming (OOP), data and functions which handle them are combined tightly into one entity, an object. Class, definition of structure of objects, is recognized as an abstracted data type, and objects are recognized as instances of classes. To make an object persistent in long period, we need special machine independent I/O mechanism. This is very common requirement in OOP, and some of solutions are already known. One of the solution is to use an object-oriented database (OODB). OODB systems have advantage for long term projects, because it supports the evolution of classes, so-called schema evolution. OODB also has many useful features to access portion of data and thought to be useful for physics analysis in HEP. Some of projects already started to study usage of OODB in HEP. However, we still need further study of usage and performance of it to apply for the actual physics analysis, especially for small experiments.

2 System Design and Object Models

We implemented the analysis software using an OODB for the AMY experiment at TRISTAN. The detailed description of the experiment is found in elsewhere [1]. The experiment operated data acquisition from 1987 to 1994, and the data were stored in TRISTAN Bank System (TBS) [2] format. The typical event size is around 30KB and average track multiplicity is around 15 at AMY. Using this data samples, we made a comparison of performance between the existing FORTRAN based program and the newly developed object-oriented version.

For this study, we developed two programs: One is for reading the data in TBS format, and store them

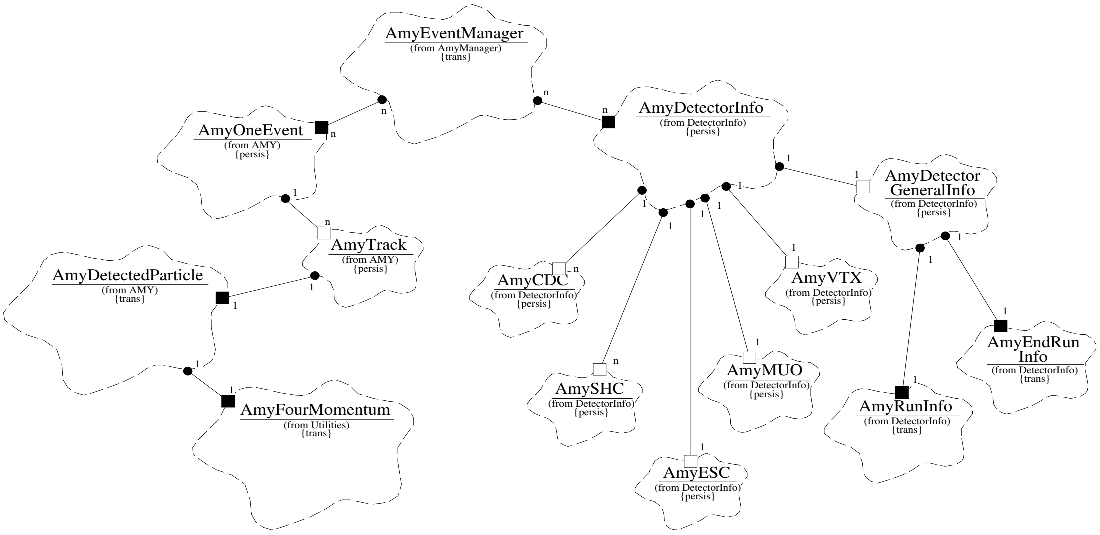
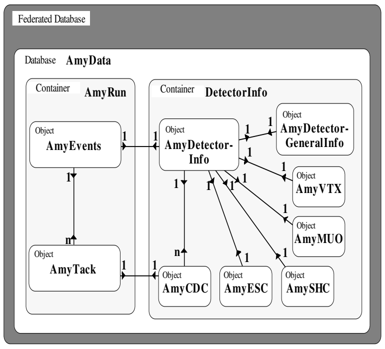


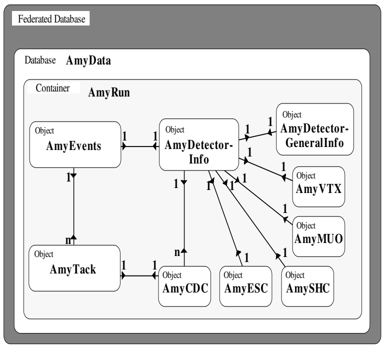
Fig. 1. Class Diagram

into OODB. The other is for accessing the OODB and performing physics analysis. Object-oriented analysis and design based on Booch method [3] for these software was done using Rational Rose/C++. Also we made use of the Objectivity/DB extension of Rose/C++ which is developed by the Japanese company, OGIS-RI. The class diagram for this system is shown in Fig. 1.

To compare performance with different database models, we tried following cases shown in Fig. 2.



model 1



model 2

Fig. 2. Object Database Model

In both of models, we stored the same number of objects on disks, but with different number of containers, which is provided by Objectivity/DB as a hierarchy of storage. In model 1, raw hits of the detectors noted as "Detector Info" in figures and reconstructed data noted as "AmyTrack" are stored in the different containers. While they are stored in the same container in model 2.

The object-oriented versions of the software was implemented using the C++ program language. Also "OO iterator" mechanism provided in Objectivity/DB was exploited for conditional search of events in the database. This mechanism is very powerful and quick to search objects which much the conditions given in boolean expressions[4]. With the simple user interface, the logic of event selection and track selection can be changed easily without recompiling the source codes. In the FORTRAN version, usual programs are written. To understand basic feature of Objectivity/DB and make fair comparison between the FORTRAN version and the object-oriented one, we made the software simple to remove unnecessary parts for this study. In any of case, total momentum of events are calculated only with tracks which pass the given qualification criteria.

3 Performance Comparison

For this study, we stored total 50,000 events both in the TBS file and the database. We striped the TBS data and stored only the same data items both in the OODB and the TBS data file to make fair comparison. We observed a small factor for the size of occupied storage compared between the FORTRAN version and the C++ versions. According to the Objectivity/DB manual, each object has 14 bytes overhead. And this makes the storage size of the OODB slightly larger than the TBS case. However, almost no difference are observed between the two object database models.

We made the comparison of the performance between the existing FORTRAN version and the newly developed C++ versions with different object database models to measure CPU consumption using them. As shown in Table 1, the object-oriented versions are almost factor 2 faster than the FORTRAN version. Here, we observed rather large performance difference between two object models, especially in the system time. Because we accessed all of the series objects in the database, the model 1 had

Table 1
Performance comparison

	sys(sec)	user (sec)	total (sec)
FORTRAN	185	8	193
C++ model 1	47	20	67
C++ model 2	51	29	80

an advantage. In Objectivity/DB, container is a basic unit of storage and caching mechanism works independently for containers. The case of random object access may give different results.

4 Summary

We implemented the data analysis software using the OODB at the AMY experiment, and measured the performance of it. The results were compared with the existing FORTRAN version of the software. The data size grows slightly because of the overhead of OODB, however the performance of the software using OODB was much better than the FORTRAN version.

Also we have measured the performance for two different object data base models. It turned out that we must be very careful to design an object database model to obtain better performance.

Even for experiments which stores small number of objects, OODB has an advantage in performance and efficiency of analysis.

5 Acknowledgment

We thank OGIS-RI to provide us their beta product of the objectivity extension for Rose/C++. We thank Professors Y. Watase and S. Kawabata for their supports on this research.

References

[1] AMY Collaboration, T.Kumita et al., Phys. Rev. D24 (1990) 1339

[2] TOPAZ Offline Note, TBS BANK SYSTEM User Manual, April 1986

[3] Object-Oriented Analysis And Design With Applications, G. Booch, ISBN 0-8053-5340-2

[4] Objectivity/DB C++ Developer's manual