Offline Meeting, Feb 18 1998

Oject database (Objectivity/DB) working group status report

W. David Dagenhart,
M. Gao,
K. Karr,
R. Pordes,
Simona Rolli,
Krzysztof Sliwa

- The model: backword compatibility
- benchmarching cdfLoader
- benchmarching cdfReader
- data model considerations

The model

- C++ and FORTRAN modules will have to co-exist in the CDF Run II reconstruction code.

- In the solution we propose, reconstruction code does not interact directly with persistent objects, but only with transient ones (TRYBOS banks). clustering and managing of persistent objects is done in an intermediate level between transient and persistent layers through Loader and Reader Modules.

- Two possible scenarios:
  1) a Loader immediately before reconstruction: this reads and writes objy objects through input/output modules
  2) the Loader immediately after reconstruction: this reads YBOS files and writes OBJY objects through an output module. Only part of RAW data inside the DB.

  Persistent data will be permanently stored using the full flexibility of an object model, where one can have associations between objects allowing an analysis to access every piece of information in the event in a dynamical way, at the level of granularity required by the user.

benchmarching CdfLoad

cdfLoader module is running under FrameWork using TRYBos

Two data schemata:

1) Type information and granularity of one object per bank.

2) Objects contain one ooVArray of bytes with the granularity of all banks with a given name.

Schema 1

filling speed: 450KB/sec

Preliminary tests show that incresing the size of objects the speed increases.

In progress: try with different page sizes decreasing page size and increasing object size. Few large AOD's/ESD's vs many small AOD's/ESD's.

Space efficiency: Objectivity : 31%

YBOS overhead: 33%

Schema 2

filling speed: 1 MB/sec

Space efficiency: 19%

Benchmarching cdfReader

Schema 1

Objectivity read speed: objects into memory: 3.7 MB/sec

Objectivity read speed with copy to TRYBOS: 2.8 MB/sec

Schema 2

CdfReader will search through the event tag database and find those events that have the event number larger than 10000. Once the condition is meet, a DETS bank is located and an accumulation of all word2 is performed. Else a METS bank is located and an accumulation of all word1 is performed.

CdfRead is 3 times faster than in schema 1 over all.

Benchmarching cdfReader

on a federated database of 10000 events we perform a database traversal: we make a cut on EventNumber, which is an attribute of the EventTag object, residing in TagDB, and for events which pass the cut we access a single attribute of the OB_DETS object in the EventDB, and we print it.

```
eventNumber > 10000  events pass: 9997
10.051u 8.072s 2:31.83 11.9% 0+0k 10244+71io 11pf+0w

eventNumber > 50000  events pass: 7510
7.571u 6.581s 2:18.21 10.2% 0+0k 8623+81io 11pf+0w

eventNumber > 90000  events pass: 5241
5.461u 4.533s 1:43.77 9.6% 0+0k 6539+40io 12pf+0w

eventNumber > 120000  events pass: 4055
4.374u 3.715s 1:26.64 9.3% 0+0k 5395+30io 12pf+0w


eventNumber > 150000  events pass: 2276
2.632u 2.370s 0:58.69 8.5% 0+0k 3552+30io 12pf+0w

eventNumber > 200000  events pass: 707
1.037u 0.843s 0:19.26 9.7% 0+0k 1024+15io 9pf+0w

eventNumber > 240000  events pass: 553
0.895u 0.754s 0:16.54 9.9% 0+0k 849+39io 9pf+0w
```

Data Model

- Keep the tag object, keep the event object, the event data object is the Trybos record.
  Need TRYBOS to translate to banks.
  Never use OBJY directly.

- Only one generic bank: ooVArray(uint32) data; no structure.
  outside code to do type conversions, TRYBOS to access data element.
  Never use OBJY directly.

- Large is better. one large AOD ? Two large AOD's (one fixed size, one variable lenght ?)
  Page size vs object size: in progress

Conclusion

We are moving away from the concept of banks. This is good. We can come up with a entirely different data model in the DB, still use TRYBOS to translate into banks but allow eventually access to the db, once the data model is known.

Document in progress

- Gedanken epxeriments, due this coming Friday;
- Data Flow Spreadsheets, almost done.
- Specifications document: already existing.

LinuX porting

Port actually has been done, it needs testing, this is most likely what Dirk Duellmann will do.

A similar port to Sinux has been tested extensively in the past.

GNU compiler has been used.