Grand Challenge Plan for Evolutionary Integration of Objectivity into Software Deployed at the RHIC Computing Facility

David Malon, Douglas Olson, Craig Tull

17 November 1997

Preface

While the HENP Grand Challenge project expects to develop an architecture for efficient data handling whose implementation does not require a specific commercial database package, it is the intention of the project to deliver software consistent with RHIC's long-term computing plans. For this reason the project has made an affirmative commitment to integration with Objectivity database software.

This note describes Grand Challenge plans for evolutionary use of Objectivity in the time frame of the RHIC mock data challenges (summer 1998). It also outlines how the proposed approach supports fallback positions that would use Objectivity to a reduced degree, or, if necessary, not at all.

Initial Objectivity Integration

There are several areas in which the presence or absence of Objectivity could make a substantial difference in the Grand Challenge architecture:

- how data are stored (instantiation of data model);
- how analysis programs access events (event iterator/analysis code interface);
- how storage management services make data available to analysis codes (storage manager/event iterator interface);
- how interesting events are identified and requested (Tag database).

The parenthetical clauses in the list above indicate the correspondence of each of these items to components of the Grand Challenge architecture. Our plans for use of Objectivity in each of these areas are described briefly below.

How Data Are Stored

The initial integration of Objectivity into STAF will mirror current DIO services, and will involve implementation of three persistent classes, corresponding to streams, datasets, and tables.

The user interface to Objectivity I/O will be identical to that currently supported in DIO (e.g., open, close, getEvent, putEvent). Each physics event will, initially, be stored as a separate dataset in the STAF (TDM) sense, with the usual STAF representation of a dataset as a collection of tables and other (nested) datasets. While this approach does not take significant advantage of many of the capabilities of object databases, it does provide a natural migration path for existing codes, and allows the fallback position of simply swapping in dioStreams for Objectivity I/O streams (oioStreams).

How analysis programs access events

In the Grand Challenge architecture, the result of a query execution request is an iterator over the collection of events satisfying the query. Invoking something akin to a next() method on the iterator returns the next qualifying event.

In STAF, a physics event will be represented in memory as an object of type tdmDataset (see above). If the STAF wrapper for this iterator is written, then, to return a tdmDataset*, user code will see no difference between events coming from an xdf file containing event datasets in XDR format, and events coming from Objectivity. Internally, the (Storage Access Enhanced) iterator should not be highly dependent upon Objectivity:

- Without Objectivity, the iterator would invoke getEvent() on a dioStream associated with the xdf file containing the event.
- With Objectivity, the iterator would invoke getEvent() identically, on a persistent oioStream object containing the persistent Event dataset.

How storage management services make data available to analysis codes

The Storage Manager makes data available to analysis programs one cell at a time. In the initial implementation, the unit of storage transfer will be a file—a database with Objectivity, an xdf file without. In either case, for each cell, the Storage Manager provides the Storage Access Enhanced Iterator with a list of Event Identifiers corresponding to qualifying events in the cell.

We expect that the primary dependence on Objectivity in this area will be in the representation of Event Identifiers. We are currently exploring approaches to Event Identifier representation that will, without unduly sacrificing performance, minimize implementation differences between identifiers corresponding to persistent objects stored by Objectivity and identifiers corresponding to datasets stored in xdf files.

How interesting events are identified and requested

Physicists will identify events of interest by issuing a selection query, which will be answered approximately by consulting an in-memory index, and exactly at execution time by consulting a Tag database containing, for each event, the most commonly queried attributes.

Our plan is to build the Tag database as a disk-resident Objectivity database of tag attributes for each event, with a corresponding Event Identifier (one example might be a d_Ref) pointing to the corresponding event. If we need to scale back our use of Objectivity in the time frame of the mock data challenges, we hope to build the Tag database using Objectivity in any case; only the Event Identifiers would be different. If Objectivity were completely unavailable, the Tag database would have the same logical structure, but would be stored, possibly in n-tuple format, in ordinary Unix files.

We foresee no difference in the in-memory index with or without Objectivity.

Implications for interface to HPSS

Because the Storage Manager will, in its initial incarnation, retrieve entire files from HPSS, there is no substantial Objectivity dependence in its implementation. The primary requirement placed by Objectivity on the Storage Manager is that Objectivity's catalog be notified (e.g., via oochangedb -catalogonly) of the location of any databases imported from tertiary storage.