Object Level Physics Data Replication in the Grid

Koen Holtman

California Institute of Technology,
Mail Code 256-48, 1200 E. California Blvd., Pasadena, CA 91125, U.S.A.

Abstract. To support distributed physics analysis on a scale as foreseen by the LHC experiments, 'Grid' systems are needed that manage and streamline data distribution, replication, and synchronization. We report on the development of a tool that allows large physics datasets to be managed and replicated at the granularity level of single objects. Efficient and convenient support for data extraction and replication at the level of individual objects and events will enable for types of interactive data analysis that would be too inconvenient or costly to perform with tools that work on a file level only.

Our tool development effort is intended as both a demonstrator project for various types of existing Grid technology, and as a research effort to develop Grid technology further. The basic use case supported by our tool is one in which a physicist repeatedly selects some physics objects located at a central repository, and replicates them to a local site. The selection can be done using 'tag' or 'ntuple' analysis at the local site. The tool replicates the selected objects, and merges all replicated objects into a single single coherent 'virtual' dataset. This allows all objects to be used together seamlessly, even if they were replicated at different times or from different locations.

The version of the tool that is reported on in this paper replicates ORCA based physics data created by CMS in its ongoing high level trigger design studies. The basic capabilities and limitations of the tool are discussed, together with some performance results. Some tool internals are also presented. Finally we will report on experiences so far and on future plans.

INTRODUCTION

We report on the development of a prototype tool, in the Caltech/CMS group, for object level physics data replication in the grid. The creation of this tool is part of a longer development effort [1, 2] in which we aim to develop software tools and middle-ware that allows for large scientific datasets to be managed and replicated at the granularity level of single objects, with sizes of 100 bytes to 1 MB, rather than at the level of multi-MB files. The development of the tool was largely an integration effort, in which existing software was combined together with code that incorporated the results of previous R&D. We discuss the parts that were integrated below.

- To manipulate and store physics objects, the Objectivity/DB object database is used; this corresponds to the current CMS software development strategy. However the tool architecture does not depend strongly on the use of Objectivity: any middleware that provides portable object streaming to and from files could have been used.
- The data to be replicated in the prototype are ORCA [3] based physics objects, created by CMS in its ongoing high level trigger design studies. For our largest prototype test to date, we used the *SimEventBody* objects for 140,000 ORCA events. Each of these objects is close to a MB in size, the complete 'source' dataset used was 105 GB, divided over 103 Objectivity/DB database files.
- We use Globus middleware [5] to implement wide-area authentication without passwords sent in the clear, and also Globus GSI FTP [7] as a mechanism for fast, tunable data transport. This way, the WAN network efficiency of our tool is equal to that what can be achieved with a tuned file based replication tool. We intend to follow future improvements in the Globus FTP space. We re-use parts of GDMP [6] as a middleware layer on top of Globus to implement authenticated, message based client-server communication.
- We use the results of existing research [2] as a basis for the object indexing mechanisms used by the tool. Our tool implements large parts of the design of [2] – the biggest omission is that that the top level catalog structures are not synchronised across sites yet. Also, the 'chunk' level of granularity was replaced with a more fluid granularity level of event ID ranges.

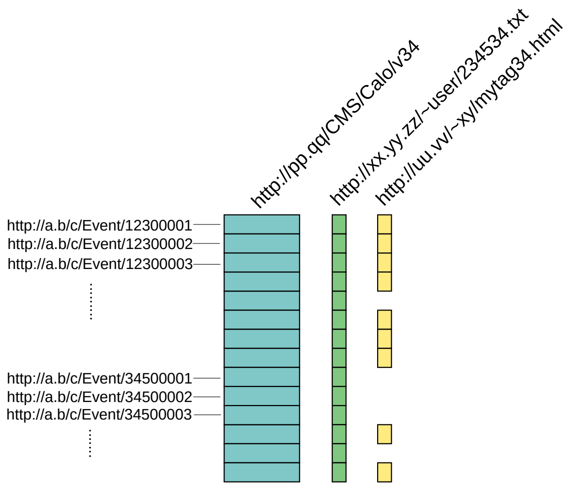

FIGURE 1. Logical datamodel of the tool

DATA MODEL

The data model defined by the tool corresponds to a sparse SQL table (see Fig. 1), where every table element is a physics object – a piece of physics data that can potentially be replicated. The rows correspond to events and the columns to the different types of physics objects (for example calorimeter raw data, tag object of user X, tag object of user Y) which may be present for the event.

Every row, every distinct event, is identified by an event ID, which is an ASCII prefix string combined with a 64-bit integer. For the topmost event (row) in Fig. 1, the prefix string is *http://a.b/c/Event/* and the integer is 12300001. The intention is then whenever a party (an experiment, a Monte Carlo production group inside an experiment) creates events, this party must choose a new unique prefix, and can then proceed to independently assign 64-bit event numbers to its events. The prefix can, for example, be an URL pointing to a page inside a website maintained by the party. Every column, every type of physics object, is identified by an ASCII string. Again, by using strings that are URLs pointing into private webspaces, multiple parties can create and fill multiple columns without naming conflicts.

All physics objects in the data model are read-only. This is an important restriction in terms of how the tool can be used. If data is to be changed, this can only be done by versioning: adding and filling a new column, and then instructing physics jobs to request objects from the new column in stead of the old one. Changing the instructions for the physics jobs will, in practice, involve updating the metadata structure used by the jobs to translate high level descriptions of a column ('the latest certified version of the production tag') to a specific column ID.

To access the local objects which have been replicated by the tool, a physics job must initialise an iterator object which is provided by the tool. The iterator is initialised with a column ID and an event list. The event list is a compact representation of a set of event (row) IDs. After initialisation the job calls the 'next' method of the iterator to retrieve the subsequent objects that were specified by the column ID and event list. If objects from multiple columns are needed for each event, multiple iterators can be initialised: their 'next' methods are guaranteed to visit the events in the same order.

TOOL INTERNALS

We explain the tool internals by following the most basic replication use case of the tool as shown in Fig. 2. In this most basic case, a central site has a one 'object with the full observations' for each event. Also, for each event, a summary object (tag object) is created. Both the user workstation and the central site have their own independent Objectivity/DB federation for storing objects. The only link between these federations is that their schema (internal class definitions) are identical with respect to the classes of any objects to be replicated. The summary objects are copied to the user workstation, and they are used to isolate an 'interesting' subset of the events, this set is saved as an event list. Then, the object replication tool is used to replicate the 'full observation' objects for these events to the user workstation. When the replication operation is finished, the full objects can be analysed by iterating over them.

The internal actions of the object replication tool, performing step 4 in Fig. 2, are as follows. The tool is started on the user workstation, with an event list and a column ID as its command line arguments. The tool connects to an object replication server (also implemented in this project) that runs on the central site. The connection uses TCP/IP, and Globus (GSI) user authentication is used to control access to the server. The tool will then check if any of the requested objects are already on the user work-

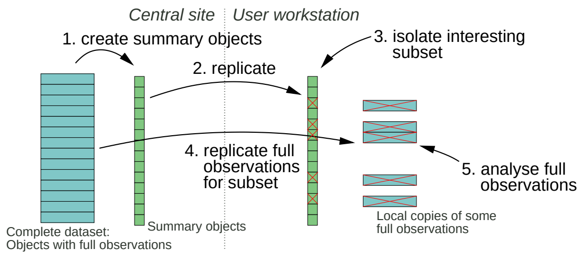

FIGURE 2. Most basic replication use case

station, if they were already replicated in a previous invocation of the tool. After having isolated the set of objects which is not yet present locally, the tool will request these objects from the server. The server locates the requested objects in its local database files, and extracts (copies) them into new Objectivity/DB database files, which are created on the server in a temporary area. The maximum size for each of these files is a server-side configuration parameter. When a file has been filled, it it sent to the user workstation using GSI FTP [7], a tunable FTP implementation which makes use of Globus authentication for security. The tool supports use of multiple FTP transfers in parallel, this is important to get good performance on a wide-area link. On arrival of a database file, the tool attaches it to the local Objectivity/DB federation catalog. The tool also adds a reference to the file to its own catalog, which is used to find replicated objects when an iterator is initialised. A specialised negotiation protocol is used between the tool and the server to make sure that the new database files have Objectivity database ID numbers which are unique at the user workstation side. When the replication operation is complete, the server deletes all database files it created in its temporary area.

When the tool is used a second time to to request the replication of a new, maybe partially overlapping, set of objects, it will transport only those objects which are not yet available locally, and will take care of seamlessly integrating all replicated objects on the user workstation. Thus the tool minimises the overhead associated with changing (refining) the cut predicate that selects the objects to be analysed on the user workstation. This allows the physicist using the tool to work in a fluid, iterative way, much more fluid than with most existing tools, where the cost of extending the object set to be analysed on the local workstation is often that the entire set has to be extracted and moved again.

PERFORMANCE RESULTS

Some performance tests were done to validate the functioning of the tool. Replication of *SimEventBody* objects from a server at Caltech to a user workstation at the Supercomputing 2000 floor in Dallas could be done at 1.2 MB/s, using 4 parallel FTPs (each with a 200 KB send buffer size), which maximised the share of the provided internet capacity we could use. Replication from a server on a 600 Mhz Dell workstation to a client on the same workstation, using the local loop-back interface, could be done at 5.8 MB/s, in that case the bottleneck was the speed of the disks in the workstation.

CONCLUSIONS

Coding of the prototype tool and server took some 2 person months after the design was completed. The tool was implemented on both Linux and Solaris, the related object replication server on Linux, Solaris, and partially (the data-intensive parts) on Windows 2000. The implementation on Windows 2000 took serious effort, compared to the time spent resolving the differences between Linux and Solaris, and our design was constrained significantly because only small parts of the Globus toolkit (the FTP clients) were available on Windows 2000. The integration effort was relatively painless, though some unexpected, nondeterministically occurring middleware bugs were encountered. A resolution to these bugs is still pending at the time of writing, which is about 1 month after these bugs were found. Temporary workarounds could be found, but this experience does suggest that, when a production system is being built that relies on large pieces of middleware, at least a few months of buffer time should be put into the planning to allow for the resolution of unexpected middleware bugs.

Future development plans in this effort are to integrate the current tool more closely with the CMS ORCA physics analysis system. Also, a system for peer-to-peer object granularity replication between many sites will be developed, which will make use of the Globus Replica Catalog to implement a global view of the contents of all sites.

REFERENCES

1. Holtman K.,van der Stok P., Willers I. "Towards Mass Storage Systems with Object Granularity." In *Proc. of the Eighth NASA Goddard Conference on Mass Storage Systems and Technologies*, Maryland, USA, March 27-30, 2000, p. 135-149.
2. Holtman K., Stockinger H. "Building a Large Location Table to Find Replicas of Physics Objects." In *Proc. of CHEP 2000*, Padova, Italy, February 2000.
3. http://cmsdoc.cern.ch/orca/
4. http://www.objy.com/
5. http://www.globus.org/
6. Samar A., Stockinger H. "Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication in High-Energy Physics." To appear in *Proc. of IASTED International Conference on Applied Informatics (AI 2001)*, Innsbruck, Austria, February 2001.
7. http://www.globus.org/datagrid/deliverables/gsiftp-tools.html