CMS Event Store; Oracle showstoppers.

Version 2,
20 July 2001.

(This is a not a final version)

With respect to Oracle as a possible fallback persistency solution for CMS event data and meta data, CMS is proposing the following evaluation strategy. First, from now till the end of 2001, IT/DB investigates Oracle to determine if some `showstoppers' exist. Results should be presented at a workshop before the end of 2001. These showstoppers are defined by the experiments; if Oracle has a showstopper then by definition of the word showstopper Oracle is not suitable for use as a persistency fallback.

If by December 2001 it is found that no showstoppers apply, then a next phase of evaluation is started from Jan-Dec 2002 in which issues like the manpower needs to convert to Oracle by CMS, and to maintain an Oracle persistency layer are estimated. Obviously the manpower also has to be within CMS acceptable limits, for Oracle to be a feasible fallback.

Each showstopper definition below is a bit like a milestone definition, in that it defines clear criteria that can be tested. For example the test of a licensing showstopper will be a successfully completed negotiation with Oracle about that issue. It is clear that some showstoppers, especially connected to the scalability of the architecture, cannot be addressed directly by doing tests on existing hardware configurations alone: these will have to be addressed with a combination of tests on real hardware and extrapolations based on knowledge about the Oracle architecture.

The list should not be confused with a complete list of Oracle requirements. Some requirements that CMS has for its persistency solution, those for which we are sure Oracle will satisfy, are not included. This list of showstoppers was written as a result of what we learned from presentations and discussions at the recent IT/DB database workshop. CMS does not currently have a complete list of Oracle requirements yet, or a complete requirements document for any persistency solution for that matter. Such a document might be written in future, after the phase of showstopper tests.

Several of the showstoppers below would also apply to an evaluation of ROOT or any other alternative solution. However this list was not written with a hybrid solution in mind, and does probably not give a complete list of showstoppers for a ROOT evaluation.

Since this is a list of showstoppers, it is by its nature negative; it should not be inferred that CMS only views Oracle in a negative light, on the contrary we see that many technical possibilities could be opened up by working with such an industry/technology leader.

1 Implementation issues

1.1 The oracle software installation for data management needs to be uniform for tiers 0-4 for servers and 0-5 for clients. We do not want different software types to support different size installations. Servers on tier 3-4 sites should be able to co-exist with other applications running on the same machine or using the same file-systems.

1.2 The ODMG-like binding of Oracle which IT/DB proposes that we use needs to be available in a stable bug-free form for CMS to start coding against by December 2001. The method of defining persistent data types should not be overly cumbersome.

1.3 Data management needs to work across multiple client and server platforms. In particular data should be movable between servers on different platforms, and client side platform constraints need to be liberal enough to be acceptable to all CMS tier 0-4 sites.

1.4 Servers and clients must be able to run on commodity hardware. In particular it must be possible to run a server on commodity CPU hardware with commodity disk hardware. Commodity is defined as the type of client and server hardware that the CERN computer center is now buying to support the LHC experiments.

2 CMS Object data model support

2.1 The existing CMS persistent object model should be expressible in the Oracle ODMG-like binding. In particular we have templates, STL, and inheritance.

2.2 Object navigation needs to be supported between all objects in all servers on a single site. Navigation to objects in remote sites is nice but not a show-stopper if unavailable.

2.3 It should be possible to make shallow copies like we do now with objectivity. User-created shallow copies should be able to point to read-only objects that the user cannot modify.

2.4 It should be possible to make deep copies of objects like we do now with objectivity.

2.5 Users should be able to create new persistent data types (schema) for themselves, the objects in this new schema should be able to reference public shared data like event data.

3 WAN/GRID support

3.1 Oracle should be able to be used inside a grid system. In particular, Grid middleware should be able to steer Oracle servers o some extent, and import and export data on these servers. Integration with the Grid-maintained data security mechanism is needed.

3.2 Oracle should support a grid model where many sites, each with one or more Oracle servers inside, exist in the grid. Data replication and migration between these sites needs to be supported. Support for this for CMS involves particular abilities in handling OID-like-addresses when data is transported. Here an OID-like-address is defined as a reference[?] to an object which can be traversed to find the object if it is on a local server.

3.2.1 If an object with OID-like-address X is replicated (as part of a table space being replicated) to a new site, it needs to retain the same OID-like-address X.

3.2.2 If an object containing OID-like-addresses is replicated (as part of a table space being replicated) to another site, then the OID-like-addresses inside need to remain untouched no matter whether the objects pointed to exist or not on the old or new site.

It would be very nice if OID-like-addresses could also be preserved in replication actions with smaller granularity, but this is not a strong requirement.

3.3 On-demand staging (replication) of data from another site, when a client navigates to an object that is not local, is nice, but lack of support for this is not a showstopper.

4 Architecture

4.1 The architecture and implementation thereof must be scalable to support: 1) database sizes of at least 100PB, 2) around 30 tier 0-2 centres containing about 10,000 CPUs and many tiers 3 and 4 which are somewhat less tightly coupled, 3) an average sustained data flow from disk of at least 1 MB/s into each of the 10,000 client CPUs 4) a peak flow of at least 10 MB/s per client CPU 5) a peak of 500 connected but inactive clients per server and 200 active clients per server, 6) splitting of the 100PB over many files where the file size is small enough that a single file can be written to tape in 15 minutes. (Take 50 MB/s tape -> this leads to about 2e6 files, so at least 1e6 on disk in tier0, seeing that many files on disk will be smaller files with user data.)

4.2 The architecture needs to support the use of tape or other tertiary storage systems. On-demand staging of data, when client navigates to an object that is only on tape, is nice, but lack of support for this is not a showstopper.

4.3 An efficient storage format on tape is needed which has long-term survivability. Efficient is defined as: no bigger size overheads as under 5.1, and no bigger conversion overheads on staging and migration as under 5.2+5.3 (read + write overheads to disk). Long-term survivability is defined as the ability to read the data back up to 4 years later, with another platform, and then convert (write) it to a more up-to-date format.

4.4 The `fat server' architecture of Oracle should be compatible with the CMS data processing model of using CPU farms running CMS C++ code to do the majority of the data processing. As long as the Oracle server cannot run internally all GEANT4 and all CMS code, the server

[?] *It is unclear now if an oracle REF is an OID-like-address which has the features above. If the REF is not, then this is not a showstopper yet, as long as something else with these features can be implemented on top of Oracle, with a similar performance and API usability.*

role will be limited to being a data server that pushes data into clients which do run this code.

5 Performance

5.1 Data storage overheads: for typical CMS C++ objects with a transient representation size of 1 KB-1MB, the persistent storage overhead needs to be within acceptable limits, which is defined as not more than a 50% overhead. It is understood that Oracle might not implement native support for floats in non-beta before December 2001, but by December 2001 it is required that Oracle has a clear intention of supporting floats and doubles with low-enough storage overheads in production versions before January 2003.

5.2 Reading objects from disk: for typical CMS C++ objects with a transient representation size of 1 KB-1MB, the CPU overhead of reading objects from disk into memory by de-referencing the OID-like-address of each object should be less than 2 SI95 per MB/s effective data throughput, counting CPU power in both client and server. For 2001 CPUs (Pasta predictions) this means less than 6% of the client+server CPU capacity per MB/s. Disk I/O traffic generated by reading should be within 170% of the effective read I/O traffic. Read latencies for objects on site-local disks should be low, low enough that they can be hidden by running 1.5 clients per CPU (3 clients on a 2-cpu node).
(This 2 SI95 is based on the back-of-the-envelope calculation that in 2007 we want to support 25 MB/s per installed CPU taking only 20% of all CPU capacity, in terms of buying disk capacity the 25 MB/s/CPU is about what we expect to have installed (assume 1 installed disk for every 2 installed CPUs, with 50 MB/s per disk). The 170% is 100% + 50% storage overhead + 20% misc. traffic.)

5.3 Writing objects to disk: Writing will be much less frequent than reading, so the maximum acceptable overhead is 10 SI95 per MB/s throughput. Disk I/O traffic generated by writing should be within 300% of the effective write I/O traffic.

5.4 Transporting objects to another server (with same or other platform): local area data extract/ingest overheads should be not larger than those of 5.2+5.3. Use of an efficient FTP implementation to transport the data in terms of large files over the wide area should be possible.

5.5 Memory overheads: For much of the CMS bulk data processing we do not expect many benefits from huge in-memory caches in either clients or servers, in fact attempts to do caching for `streams' of data might be a source of inefficiency. The memory overhead in a client executable due to the use of Oracle should not be more than 40 MB. The memory needed to run a server with low I/O rates on a tier-4 (desktop or laptop) system should not be more 100 MB. For a high-I/O rate server on a tier 0-3 site, the memory overhead should not be more than 250 MB plus 10 MB per connecting client. These overheads are for deployment on current 2001 systems of the current Oracle version, as memory gets cheaper over time, higher overheads from later Oracle versions will start to become acceptable.

6 Security

6.1 Shared event data should be protected from modification by end users, while allowing end user read access. Users and groups should be able to store `private' data.

6.2 Users should be able to perform typical administrative operations like adding schema or exporting data, without this opening up access to all administrative operations for shared data.

7 Maintenance and support

7.1 It should be possible to operate servers on tier 3 and 4 sites without having to have a dedicated Oracle database administrator at each site.

7.2 If Oracle is taken into production for LHC experiment use from 2003 on, IT/DB must have sufficient staff to offer the necessary support to collaborators both inside and outside CERN. Providing this support must be compatible with a continuation in parallel of required Objectivity support.

7.3 Oracle support services should be available to the whole collaboration (maybe through a filter in IT/DB), support should also apply to bugs and problems encountered with Oracle on commodity hardware which is not `Oracle certified'. We can accept to some extent a limitation that we should try to reproduce bugs on 'certified hardware' first, but it should be recognised that we will likely encounter scalability limitations and bugs which we cannot possibly reproduce on certified hardware, because we do not have a large enough certified installation, and we will still need support in solving these problems.

8 Commercial conditions

8.1 Product licensing and support costs should be acceptable for CMS. We need a collaboration-wide license or set of licenses that is equally easy for us to manage. The license should support Oracle server use for CMS data handling purposes by all collaborators on all tier 0-4 sites (CERN down to desktops, laptops) and client use on all tier 0-5 sites (CERN down to desktops, laptops and portable devices). It should be possible for CMS to distribute the Oracle software to its collaborators from a central big repository (ANAPHE model). Scale of the license: 2000 collaborators, who all potentially have a server installed on both their laptop and their desktop. Enough active Oracle server installations to serve at least 10,000 MB/s sustained to clients. At least 500 active named users at any point in time. At least 10,000 client processes connected to servers at any point in time.

8.2 It should be possible to use Oracle in all CMS collaboration member states (including Russia, Pakistan, Iran, etc).

8.3 Oracle should have a clear long-term commitment to keep supporting the features we need (like the C++ ODMG-like binding) in the long term (at least 5 year timeframe), at the same price levels. We cannot expect any software company to make a 30-year commitment to support, but do require a relation that gives us a large amount of stability and forewarning about a need to switch or convert to new features.