CMS Conditions Data Access using FroNTier

Barry Blumenfeld[1]**,David Dykstra**[2]**, Lee Lueking**[2]**,Eric Wicklund**[2]

[1]Johns Hopkins University, Baltimore, Maryland 21218

[2]Fermilab, Batavia, Illinois 60510

E-mail: `lueking@fnal.gov`

Abstract. The CMS experiment at the LHC has established an infrastructure using the FroNTier framework to deliver conditions (i.e. calibration, alignment, etc.) data to processing clients worldwide. FroNTier is a simple web service approach providing client HTTP access to a central database service. The system for CMS has been developed to work with POOL which provides object relational mapping between the C++ clients and various database technologies. Because of the read only nature of the data, Squid proxy caching servers are maintained near clients and these caches provide high performance data access. Several features have been developed to make the system meet the needs of CMS including careful attention to cache coherency with the central database, and low latency loading required for the operation of the online High Level Trigger. The ease of deployment, stability of operation, and high performance make the FroNTier approach well suited to the GRID environment being used for CMS offline, as well as for the online environment used by the CMS High Level Trigger (HLT). The use of standard software, such as Squid and various monitoring tools, make the system reliable, highly configurable and easily maintained. We describe the architecture, software, deployment, performance, monitoring and overall operational experience for the system.

1. Introduction

The CMS Experiment is using a multi-tier web approach to deliver conditions data to a worldwide community of distributed processing and analysis clients. CMS conditions data includes calibration, alignment, and configuration information used for offline detector event data processing. Conditions data is keyed by time, or run number, and defined to be immutable, i.e. new entries require new tags or versions. A given object may be used by thousands of jobs and caching such information close to the processing activity provides significant performance gains. Readily deployable, highly reliable and easily maintainable web proxy/caching servers are a logical solution for this caching and fit seamlessly into the web approach.

2. Implementation and Advantages

The CMS software stack used by the client to access the conditions data is illustrated in Fig. 1. CMS uses POOL-ORA (Object Relational Access) [1] as an object to relational mapping tool to relate C++ objects to the relational database schema. A FroNTier plugin for CORAL [2] has been created to provide read-only access to the POOL DB objects via FroNTier. The system functions with the following steps:

(i) Pool and CORAL generate SQL queries from the CMS client framework C++ objects.

(ii) The FroNTier client converts the SQL into an HTTP GET and sends it over the network to the FroNTier server.

(iii) The FroNTier server unpacks the SQL request, sends it to the DB server, and retrieves the needed data.

(iv) The data is optionally compressed, then packed into an HTTP formatted stream and sent back to the client.

(v) Squid proxy/caching server(s) between the FroNTier server and client caches requested objects, significantly improving performance and greatly reducing the load on the central database.

The system uses standard tools to make it reliable, readily deployable, highly configurable and easily maintained. The FroNTier server employs Tomcat [3] as the servlet container, Squid [4] is used for the proxy/caching servers; both are well proven and extremely reliable. Squid provides convenient monitoring through SNMP [5], and MRTG [6] are employed to chart interesting metrics including access rates, network throughput and cache hit statistics. Many tools are available to analyze the logs produced by squid which are formatted similar to those produced by Apache servers, and we currently are using AWStats [7] to do this for the central servers. These products are well documented in existing literature and on the web.

The only deployments needed at Tier-1, Tier-2, and Tier-3 centers are Squids, and installation is quick, and administration is straightforward. Squids are highly configurable and customizable to match specific site environments, security and other needs. No DBA's are needed beyond the central database at CERN as the caches are loaded on demand and self managing. If a cache is lost or corrupted it is simply repopulated automatically with little or no intervention required.

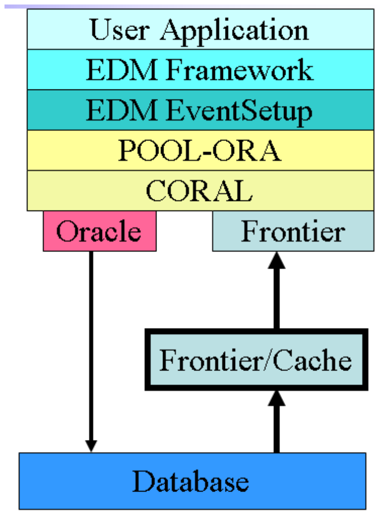

Figure 1. The CMS software stack.

3. FroNTier Deployment

The FroNTier system has been, or is being, deployed for all processing that requires access to the CMS conditions data. Fig. 2 provides an overview of this deployment at the Experiment site, for the Tier-0 prompt reconstruction farm, and beyond to Tier-1 and Tier-N sites. The High Level Trigger (HLT) filter farm and Tier-0 prompt reconstruction farm represent special challenges that will be discussed later. One key component of the offline deployment is the central service referred to as the "Launchpad" shown in Fig. 3. Here, two machines are operated in parallel, each with a Tomcat and Squid operating in tandem. The two boxes provide load

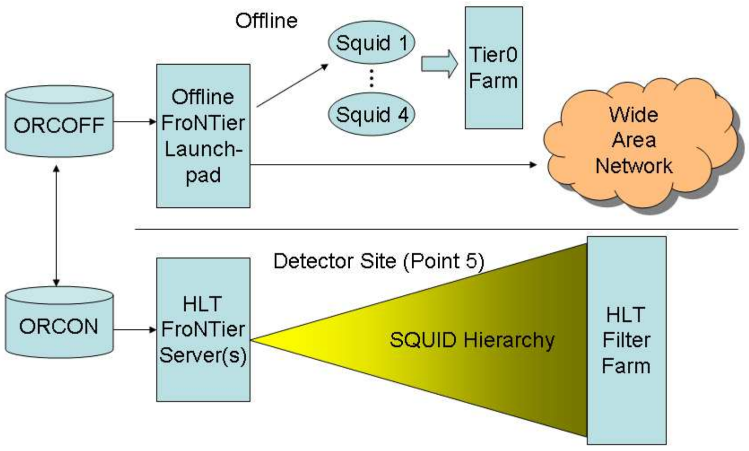

Figure 2. Overview of FroNTier deployment for delivering CMS conditions data.

sharing through a Round Robin DNS configuration and failover through client retries. If one of the Launchpad servers is down, it is removed from the Round Robin DNS dynamically. The Squids are configured in accelerator mode and communicate only to the FroNTier Tomcats so no access control is required. This is referred to as "wide open FroNTier" and allows new sites to begin using the service with no registration required, a convenient approach given the very large number of potential users. The FroNTier servlets run under the standard Tomcat server, and there is typically a servlet for development, integration and production activities. Each servlet is configured to connect to an instance of the Oracle database.

There are many configuration options available for the squids and the cache management and sharing greatly affects the performance of the system. Peer caching was initially used to share cached objects between the launchpad squids. It was realized, however, that in some cases when the same object is being requested by many clients at nearly the same time there can be a race condition during which the database is accessed directly more than expected. A squid option called "collapsed forwarding" was added in Squid v 2.6 that resolves this problem, but is incompatible with peer caching and it was chosen instead.

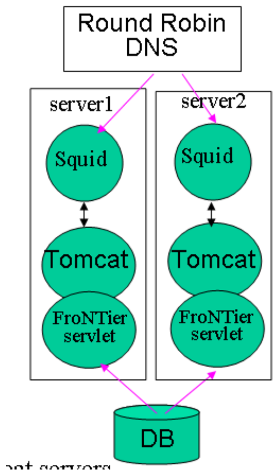

Figure 3. FroNTier central launchpad operated at CERN.

4. Cache Coherency

One important issue that must be dealt with in the system is that of cache coherency. Since the Squid caches are populated automatically and have no connection to the state of the data in the central oracle server it is possible for the cache to become stale. For the most part, this problem is resolved by CMS policy that all objects put in to the database are never changed. If conditions data is updated in the database a completely new tag identifier is stipulated and the previous data is unchanged. The tag and the information about the time region of validity, or Interval of Validity (IOV) are referred to as the metadata. The actual conditions data itself in the POOL-ORA repository is called the payload. The relationship between the metadata and the payload is illustrated in Fig. 4. By decree, data in conditions IOV can **not** change, therefore caching is consistent. However, new IOV's and payloads can be appended in order to extend the IOV range and this is inconsistent with caching.

After considering many ideas, CMS has adopted the solution that all cached objects have expiration times. There are two categories of object expiration:

- **Short-lived:** Metadata objects, including the pointers to payload objects, may change in short time period, and
- **Long-lived:** Payload objects which never change.

The amount of time for short-lived and long-lived objects are adjusted according to where in the CMS data model the data is being used. For example, in the online the calibrations change quickly as new data is added for upcoming runs. At Tier-0, calibrations change on the order of a few hours as new runs appear for reconstruction and at Tier 1 and above conditions data may be stable for weeks. The value of the short and long expirations is controlled at the central FroNTier server, so they can easily be tuned as needed.

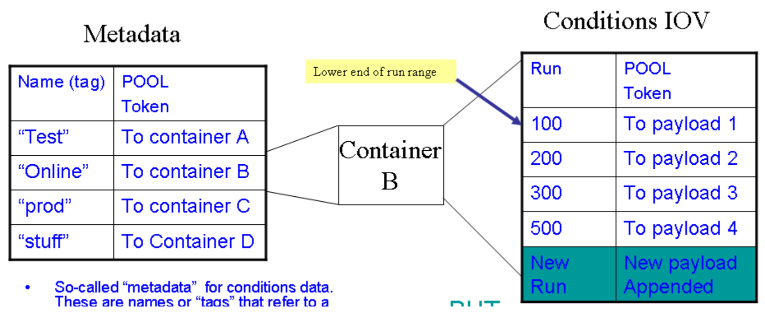

Figure 4. CMS organization of Conditions data and metadata.

5. Deployment and Operation

The number of sites with working FroNTier Squids installed is charted in Fig. 5. Deploying squids for CMS began in late 2005 when 10 centers were used for initial testing. In 2006 additional sites were prepared between May and October in preparation for the Computing, Software and Analysis Challenge (CSA06). More sites are being deployed throughout the summer and fall of 2007 in preparation for the ongoing CSA and perhaps as many as 20-30 sites will be added. There are several options for installing the squids but a tarball and scripts provided by CMS is the most popular. Very few problems have been experienced with the installation procedures CMS provides.

The system has been heavily used by CMS for the last year. The daily requests to the FroNTier launchpad and information delivered by it are charted in Figures 6 and 7. These charts

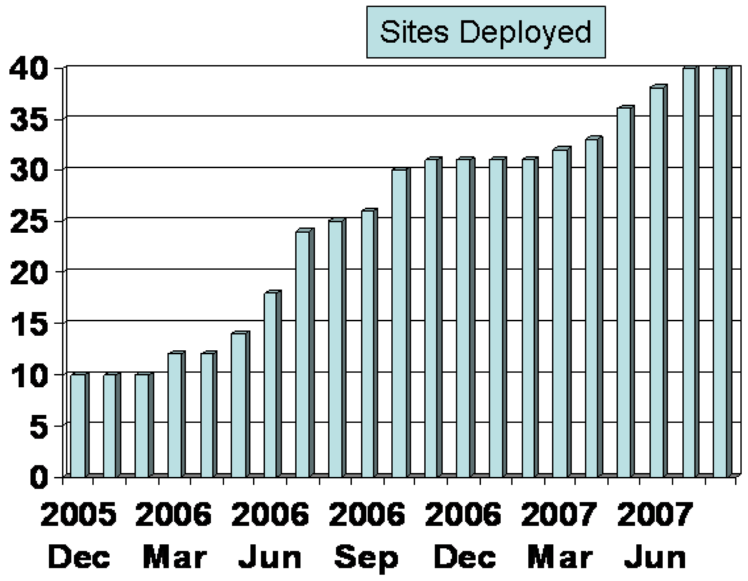

Figure 5. FroNTier Squid deployment at CMS Tier-0, 1, and 2 sites.

reflect the total activity of production, development, and testing. The number of daily requests varies significantly but a peak day was observed in April when over 100 Million were served. This was anomalous and about two orders of magnitude above what is expected, nevertheless it verifies that the system operates stably even under extreme load. During the monitoring period charted there was at least one day with 100 GB of data delivered to remote sites. Of course in this system the requests observed on the launchpad represent only a small fraction of the total client requests satisfied mostly by remotely deployed squid caches.

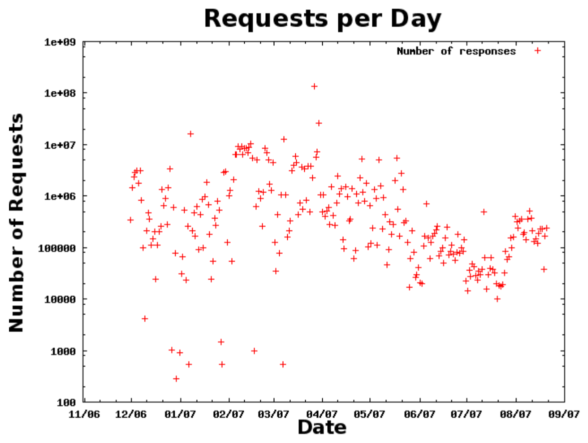

Figure 6. FroNTier central launchpad daily requests satisfied since December 2006.

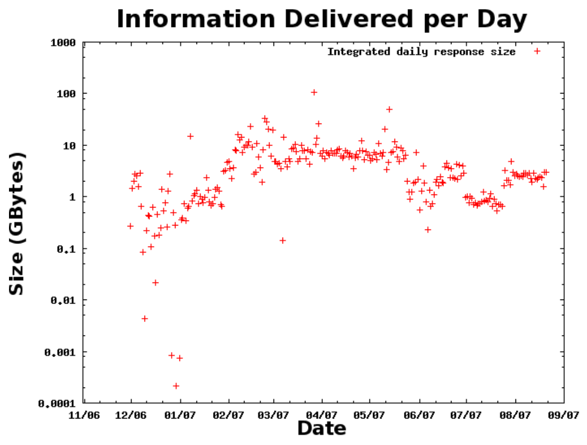

Figure 7. FroNTier central launchpad daily data delivered since December 2006.

There are several approaches being used to monitor the system's health and operation. The squid logs produced on the launchpad are mined for statistics on requests origins, number of requests, amount of data transferred, and response times for requests. All squids in the system are polled every five minutes through their SNMP interface to collect and record information

Table 1. Requirements for conditions data delivery for High Level Trigger and Tier-0 Prompt reconstruction Farm. Entries marked with an "*" indicate worst-case expectations.

Parameter	HLT	Tier0
Number of Nodes	2000	1000
Number of Processes	16k	3k
Startup	< 10 sec all clients	< 100 sec per client
Client Access	Simultaneous	Staggered
Cache Load	< 1 Min	N/A
Tot Obj Size	100 MB*	150 MB*
New Objects	100% / run*	100% / run*
Number of Squids	1 per node	Scalable (2-8)

about the number of objects served, throughput, size of the cache, and cache hit performance. In addition, as part of the Site Availability Monitoring (SAM), GRID jobs are automatically submitted several times a day to all deployed sites to validate the integrity of the entire FroNTier system. The success or failure of the SAM tests are collected and are available with other site tests on SAM web pages.

6. Specific Challenges

Within CMS there are two specific areas that represent challenges to the FroNTier system; the online High Level Trigger (HLT) filter farm and the offline Tier-0 prompt reconstruction farm. Requirements for these two areas are summarized in Table 1. In the case of the HLT all the conditions information needs to be loaded simultaneously into 16k processes running on 2k hardware nodes in less than 10 seconds. In the worst case it is estimated that the amount of conditions data needed is around 100 MB, and all of this could be refreshed every new run which amounts to delivering 1.8 TB of data to the clients. In the case of the prompt reconstruction farm of order 3k processes start in a non-simultaneous, or staggered, fashion. Although the requirements are less stringent for the offline case the environment is less controlled than online, and slightly different solutions are needed in the two cases.

6.1. Hierarchy of Squids

The solution developed for the HLT farm is to deploy a squid on each node, and a hierarchy of squids to feed these squids. With the squid cache on each node loaded, the simultaneous start of the 8 processes there is easily accomplished in under 10 seconds. An additional requirement on this approach is to pre-load all the squid caches, in preparation for a run, in less than a minute. Testing has been conducted using a small part of the online HLT farm to confirm that these requirements can be satisfied. The configuration chosen has 6 tiers of squids with each tier fanning out to four squids in the next tier. Three tiers of squids feed the 50 racks of nodes via one non-blocking GBit network, and 3 tiers of squids within each rack feed the 40 constituent nodes over separate non-blocking GBit networks. The current bottleneck is the conversion from the database to HTTP in the FroNTier server, but this portion of the load time is still under the one minute requirement.

6.2. Parallel load-balanced Squids

For the prompt reconstruction farm the time required to access 150 MB of conditions data should be less than 1 % of the time for the entire job. This might translate into approximately one minute and can easily be satisfied by a set of parallel squids configured with Round Robin DNS load-sharing. It is believed that as many as eight parallel servers can be used effectively with this type of load-sharing and the system is now being tested with four. In cases where many farm jobs start together there may be bursts of accesses that slow the response to some degree.

On multi-processor/multi-core machines, CPU resources are under utilized because a single Squid process utilizes only one CPU. Multiple squids can be run on each machine employing multiple network interfaces to greatly improve the effectiveness of each machine. Two approaches have been examined, 1) Multi-homed where the machine looks like multiple nodes and uses multiple network interfaces, and 2) Bonded interfaces where multiple network interfaces are used together to increase the network throughput. The bonded approach was implemented at Fermilab because it works best with the specific load balancing configuration being used there. With this configuration, one multi-processor server using two GBit network interfaces and running two squids, 200 MBps throughput can be achieved. Some sites, for now, prefer simply adding more machines without special configurations to reduce network configuration issues.

7. Summary

FroNTier is used by CMS for all accesses to conditions data and significant operational experience has been gained over the last year. The ease of deployment, stability of operation, and high performance make the FroNTier approach well suited to both online and offline environments. The use of standard software, such as Squid and various monitoring tools, make the system reliable, highly configurable and easy to maintain. There are currently 40 squid sites being monitored and many more expected.

Recent testing has demonstrated that the requirements needed for the online environment used by the CMS High Level Trigger filter farm can be met by pre-loading a hierarchy of squid. Meeting the needs of the offline prompt reconstruction farm can also be met, and at Fermilab running multiple Squids processes on a single server machine has been demonstrated to more effectively utilize the resources available on modern servers.

8. Acknowledgments

We would like to thank DOE and NSF for support of the project under which this work was conducted. This effort is the result of many within CMS working together to anticipate the needs of the Experiment. The CMS Computing and Offline Software organization has provided the overall data model and software framework in which FroNTier us used. We greatly thank the numerous individuals within CMS and CERN IT who have contributed to its deployment, testing, and operation.

References

[1] POOL-ORA home page: http://pool.cern.ch/
[2] D. Duellmann, et. al.,"Development Status and Plans for the LCG Common Database Access Layer CORAL," Proceedings of this conference. Also see the CORAL home page: http://pool.cern.ch/coral/
[3] Tomcat home page: http://tomcat.apache.org/
[4] Squid home page: http://www.squid-cache.org/
[5] SNMP home page: http://www.snmplink.org/
[6] MRTG home page: http://oss.oetiker.ch/mrtg/
[7] AWStats home page: http://awstats.sourceforge.net/