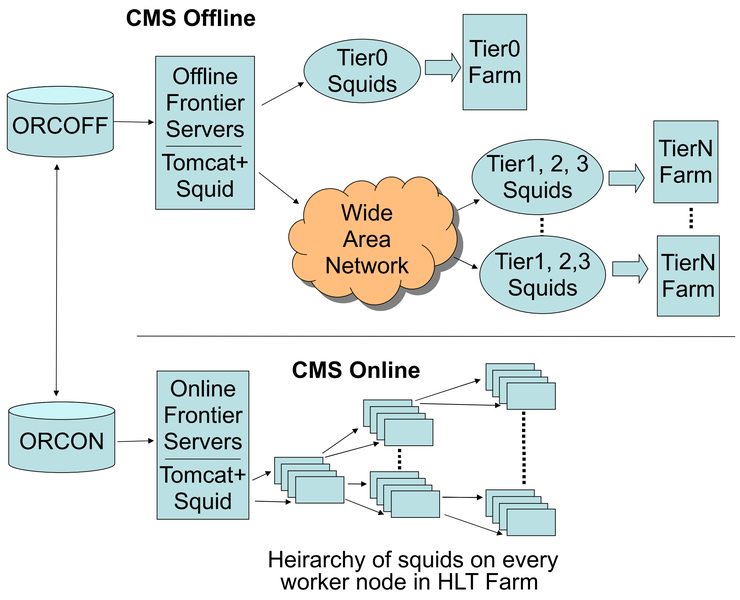
Greatly Improved Cache Update Times for Conditions Data with Frontier/Squid

Dave Dykstra, Lee Lueking
Computing Division, Fermilab, Batavia, IL

Introduction

The CMS detector project loads copies of conditions data to over a hundred thousand computer cores worldwide by using a software subsystem called Frontier. This sub-system translates database queries into http, looks up the results in a central database at CERN, and caches the results in an industry-standard http proxy/caching server called Squid. One of the most challenging aspects of any cache system is coherency, that is, ensuring that changes made to the underlying data get propagated out to all clients in a timely manner. Recently, the Frontier system was enhanced to drastically reduce the time for changes to be propagated everywhere without overloading servers. The propagation time is now as low as 15 minutes for some kinds of data and no more than 60 minutes for the rest of the data. This was done by taking advantage of an http and Squid feature called If-Modified-Since in which the Last-Modified time-stamp of cached data is sent back to the central server. The server responds to this with a very short message if data has not been modified, which is the case most of the time, and re-validates the cache. In order to use this feature, the Frontier server has to send the Last-Modified timestamp, but since modification times are not normally tracked by Oracle databases a PL/SQL program was developed to keep track of the modification times of database tables. We discuss the details of this caching scheme and the obstacles overcome including Oracle database bugs and Squid bugs.

Background – Frontier usage in CMS



In CMS Offline, client jobs on worker node farms at over 50 sites worldwide make http requests to their local Squids to load conditions data. The Squids in turn load data from the central Squids at CERN. The central Squids contact the tomcat on the same server which converts the http requests to Oracle database requests.

In CMS Online HLT, there is a Squid on every worker node, arranged in a hierarchy to load nearly 10,000 cores in under a minute.

Work supported by the U.S. Department of Energy under contract No. DE-AC02-07CH11359

The Problem – Cache Coherency

Every caching system must be able to track changes in the underlying data. Previously, this was managed in Frontier by separating queries into two types: (1) those that were expected to change and (2) those that weren't. Those that were expected to change were set to expire sooner than those that weren't. In CMS Offline, even the short times were still quite high, however: most were once per day. The long times were very high: a year. Setting the times much shorter would result in much higher delays and stress on the infrastructure, because the data had to be reloaded all the way from the database, even if it hadn't changed.

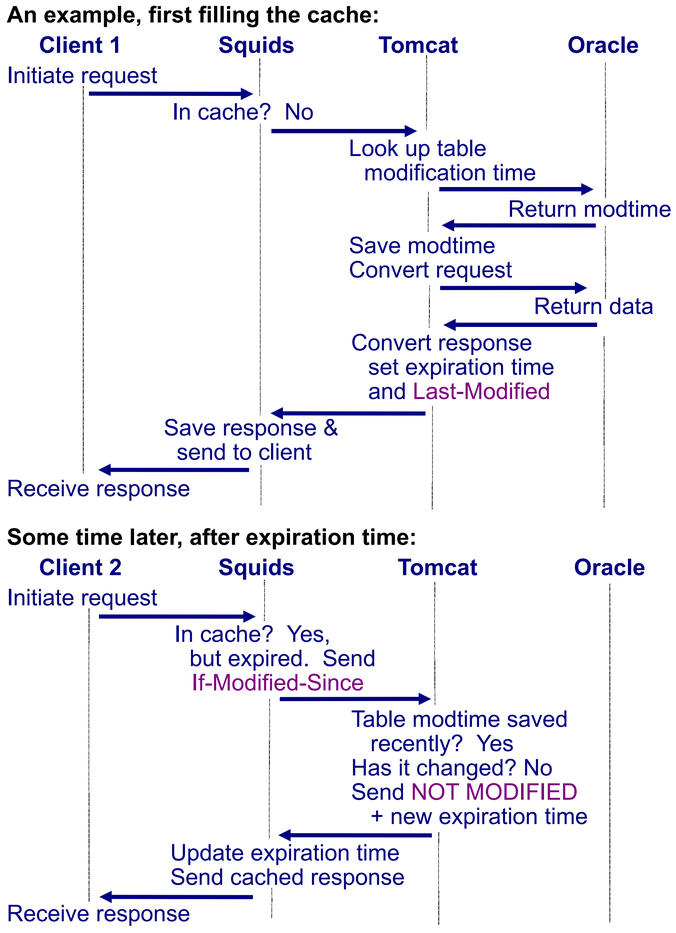
The problem with this previous approach was that there were often times when data would change in the database even though it wasn't supposed to. There were many occasions where caches had to be manually cleared.

In CMS Online, everything was set to be expired very frequently so there wasn't a coherency problem. However, all Conditions data had to be reloaded every time the Run was started. That took less than a minute, but during that time all collisions at the detector would be lost, so that was still too long: it was required to be less than 10 seconds. Fortunately, the new solution (described below) re-validates expired cached items so unless the data changes, it doesn't need to be completely reloaded and so is much faster.

The Solution – If-Modified-Since

The http protocol and Squid already had a solution: if a server supplies a Last-Modified time, and a query comes to Squid for an object in its cache that has expired, Squid automatically issues an If-Modified-Since request with that time.

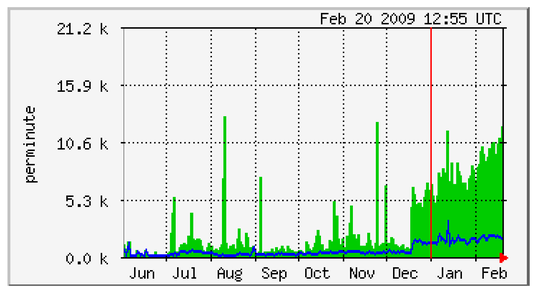
An example, first filling the cache:



So the long-distance traffic, and the work that Tomcat and Oracle have to do, is greatly reduced when the data is not modified. Instead, the response is a small NOT MODIFIED message. Tomcat saves the per-table modification times and re-uses them for up to 5 minutes (typically) so if there are many queries for the same table close together Oracle doesn't have to be contacted every time. These things allow us to greatly reduce the expiration times without overloading the servers, and so allow changes to be noticed much faster.

Effect on Frontier Infrastructure

The load on the 3 central Frontier servers at CERN did go up noticeably when we cut over in mid-December 2008 but they still have plenty of spare capacity. The chart below shows a plot of the combined number of requests per minute for the 3 central Squids over the last months (averaged per day):



The green area is the requests received from site Squids and the blue line is the requests passed on to tomcat. Site Squids often serve 40k+ requests per minute each so a few thousand requests per minute each is easy.

Getting Modification Times from Oracle DB

The most challenging part of this deployment was that the Oracle database does not normally track modification times. However, this capability was added using PL/SQL scripts. Three different approaches have been taken:

1. Use SQL triggers to track all table modifications and keep a timestamp in a LAST_MODIFIED_TIMES table in each account. This works, but DB administrators worried that the triggers would be too heavy of a load on the database servers during updates so it was never deployed.
2. Use Oracle's DBMS_CHANGE_NOTIFICATION package to update timestamps only upon DB COMMITs, and triggers to track table CREATEs and DROPs. At first there was a bug limiting the length of account plus table names, but Oracle fixed that, so this is the approach that was deployed. However, experience in production has shown a few more problems:
   - Sometimes it stops updating the timestamps.
   - The PL/SQL script for this has the inconvenience along with solution #1 of having to be installed in every account, and has to be reinstalled after an account's entire content is copied to another account.
   - Sometimes it interferes with streaming data from one server to another.
3. Use Oracle's DBMS_STATS package to put modification times into an ALL_TAB_MODIFICATIONS view, and copy the timestamps from there to a single table for the database. This method is being tested as of this writing in late February 2009. It requires periodically flushing the statistics, so a disadvantage is that it adds an additional 5 minute delay for changes to be noticed. The big advantage is that only has to be set up once per database.

Squid bugs

We also encountered two Squid bugs that affected this deployment, that were fixed in a very timely manner:

1. The oldest open bug in the Squid bug tracking system, bug #7 from August 2000, reported that http headers were not re-written in the cache when they are updated. It is necessary to update the cached Date header for this deployment when one Squid feeds another. Very fortunately for us, the latest stable Squid release 2.7 had finally fixed this 1.5 months before we ran into it.
2. During testing, we found a case where the Date header was still not updated. This was Squid bug #2430 and was fixed in October 2008 in Squid release 2.7STABLE5.