EXHIBIT 4

« How Can the Large Hadron Collider Withstand One Petabyte of Data a Second? | Main | How did Google Instant become Faster with 5-7X More Results Pages? »

Google's Colossus Makes Search Real-Time By Dumping MapReduce

SATURDAY, SEPTEMBER 11, 2010 AT 5:50AM

As the Kings of scaling, when Google changes its search infrastructure over to do something completely different, it's news. In Google search index splits with MapReduce, an exclusive interview by Cade Metz with Eisar Lipkovitz, a senior director of engineering at Google, we learn a bit more of the secret scaling sauce behind Google Instant, Google's new faster, real-time search system.

The challenge for Google has been how to support a real-time world when the core of their search technology, the famous MapReduce, is batch oriented. Simple, they got rid of MapReduce. At least they got rid of MapReduce as the backbone for calculating search indexes. MapReduce still excels as a general query mechanism against masses of data, but real-time search requires a very specialized tool, and Google built one. Internally the successor to Google's famed Google File System, was code named Colossus.

Details are slowly coming out about their new goals and approach:

- Goal is to update the search index continuously, within seconds of content changing, without rebuilding the entire index from scratch using MapReduce.
- The new system is a "database-driven, Big Table–variety indexing system."
- When a page is crawled, changes are made incrementally to the web map stored in BigTable, using something like database triggers.
- A compute framework executes code on top of BigTable.

The use of triggers is interesting because triggers are largely ignored in production systems. In a relational database triggers are integrity checks that are executed on a record every time a record is written. The idea is that if the data is checked for problems before it is written into the database, then your data will always be correct and the database can be a pure source of validated facts. For example, an account balance could be checked for a negative number. If the balance was negative then the write would fail, the transaction aborted, and the database would maintain a correct state.

RECENT POSTS

Sponsored Post: Couchbase, Tokutek, Logentries, Booking, Apple, MongoDB, BlueStripe, AiScaler, Aerospike, LogicMonitor, AppDynamics, ManageEngine, Site24x7

How the AOL.com Architecture Evolved to 99.999% Availability, 8 Million Visitors Per Day, and 200,000

In practice there are many problems with triggers:

- Destroys performance. Since triggers happen on every write they slow down the write path to the extent that database performance is often killed. Triggers also take record locks which makes contention even worse.
- Checking is distributed. Not all integrity checks can be made from data inside the database so checks that are database oriented are put in the triggers and checks, that say access a 3rd party system, are kept in application code. So now we have checks in multiple places, which leads to the update anomalies in code that ironically, relational databases are meant to prevent in data.
- Checking logic is duplicated. Checking in the database is often too late. A user needs to know immediately when they are doing something wrong, they can't wait until a transaction is committed to the database. So what ends up happening is checking code is duplicated, which again leads to update anomalies.
- Not scalable. The database CPU becomes dedicated to running triggers instead of "real" database operations, which slows down the entire database.
- Application logic moves into triggers. Application code tends to move into triggers over time since triggers happen on writes (changes), in a single common place, they are a very natural place to do all the things that need to be done to data. For example, it's common to emit events about data changes and perform other change sensitive operations. It's easy to imagine building secondary indexes from triggers and synchronizing to backup systems and other 3rd party systems. This makes the performance problems even worse. Instead of application code being executed in parallel on horizontally scalable nodes, it's all centralized on the database server and the database becomes the bottleneck.

So, triggers tend not to be used in OLTP scenarios. Applications contain the same logic and it's up to release policies and testing to make sure nothing falls through the cracks.

This isn't to say you don't want to put logic in triggers, you do. Triggers are ideal in an event oriented world because the database is the one central place where changes are recorded. The key is to make triggers efficient.

It sounds like Google may have made a specialized database where triggers are efficient by design. We know hundreds of thousands of pages are being crawled simultaneously. When the pages are written to the database that's the perfect opportunity perform calculations and updates. The compute framework would make it efficient to perform operations in parallel on pages as they come in so that processing isn't completely centralized on each node.

One can imagine that the in-memory data structures that existed on the MapReduce nodes have been extracted in some form and reified within BigTable. What we have is a sort of Internet DOM, analogous to the browser DOMs that have made it possible to have such incredibly powerful browser UIs, but for the web. I imagine programming the web for Google has become something like programming a browser DOM. I could be completely wrong of course, but I explored this idea a while ago in All the world is a DOM.

There's an interesting quote at the end of the article: "*We're in business of making*

Login
Register
All Time Favorites
Useful Products
Useful Papers
Useful Strategies
Useful Blogs
Useful Books
Useful Conferences
Book Store

High Scalability RSS
High Scalability Comments RSS

Requests Per Second
Stuff The Internet Says On Scalability For February 14th, 2014
Snabb Switch - Skip the OS and Get 40 million Requests Per Second in Lua
Paper: Network Stack Specialization for Performance
13 Simple Tricks for Scaling Python and Django with Apache from HackerEarth
Stuff The Internet Says On Scalability For February 7th, 2014
Little's Law, Scalability and Fault Tolerance: The OS is your bottleneck. What you can do?
Sponsored Post: Logentries, Booking, Apple, MongoDB, BlueStripe, AiScaler, Aerospike, LogicMonitor, AppDynamics, ManageEngine, Site24x7
How Google Backs Up the Internet Along With Exabytes of Other Data

ALL TIME FAVORITES
More Favorites...
YouTube Architecture
Plenty Of Fish Architecture
Google Architecture

searches useful, we're not in the business of selling infrastructure."

These could actually be the same goal. It's a bit silly to have everyone in the world crawl the same pages and build up yet another representation of the Web. Imagine if the Web was represented by an Internet DOM that you could program, that you could write to, and that you could add code to just like Javascript is added to the browser DOM. Insert a node into the Internet DOM, like a div in an HTML page, and it would just show up on the Web.

This Internet DOM could be shared and all that backbone bandwidth and web site CPU reclaimed for something more useful. Google is pretty open, but they may not be that open.

Todd Hoff | 11 Comments | Permalink | Share Article Print Article Email Article

Reader Comments (11)

If you are looking to build something similar, check out the open source elasticsearch. It was built with the mentioned architecture in mind (sorry for the "plug", I am the creator of elasticsearch).

September 11, 2010 | Shay Banon

Interesting. But Google Instant is independent of realtime search, isn't it? I thought Instant was a new UI that shows results before you hit the Search button, with the same back end search.

September 11, 2010 | Brian M

I suspect you're taking the statement "like triggers" too literally. Incrementally updating results is known to be very useful for maintaining real-time result sets. Look up "materialized views".

September 11, 2010 | Allan MacKinnon

Thanks Todd, great post!

I think the idea of the "internet DOM" is very powerful. Try to imagine the next step. Node's don't only represent documents on the web, but objects in the world. Each object can be a part of the world DOM. In DOM you also have events, so you can listen to events that happen on objects in the DOM. You can listen for events and trigger new events. Perhaps your alarm clock will listen to changes in your work schedule and your toaster and friends will listen to your alarm clock. Maybe an inventory management system will listen for price-change events and execute orders accordingly. The world object DOM with events is a very powerful computational network that works very similar to a human brain.

How Twitter Stores 250 Million Tweets A Day Using MySQL

Scaling Twitter: Making Twitter 10000 Percent Faster

Flickr Architecture

Amazon Architecture

How I Learned to Stop Worrying and Love Using a Lot of Disk Space to Scale

Stack Overflow Architecture

Facebook

An Unorthodox Approach to Database Design : The Coming of the Shard

Building Super Scalable Systems: Blade Runner Meets Autonomic Computing in the Ambient Cloud

Are Cloud Based Memory Architectures the Next Big Thing?

Latency is Everywhere and it Costs You Sales - How to Crush it

How will memristors change everything?

DataSift Architecture: Realtime Datamining At 120,000 Tweets Per Second

Useful Scalability Blogs

Scaling Traffic: People Pod Pool of On Demand Self Driving Robotic Cars who Automatically Refuel from Cheap Solar

VoltDB Decapitates Six

This is far ahead of the curve, though. Google seems to finally have figured out how to navigate the "document" graph in realtime. The next difficulty level is the social graph, where we don't know yet who'll take the cake. Only then will the object graph play a role. Exciting :)

September 11, 2010 | Jonas Huckestein

Sounds closer to replication than triggers? They're just updating the copy in Bigtable?

September 12, 2010 | Greg Linden

Brian M:

It's more than just a new UI. If you think about it, instant requires a bunch of new back-end infrastructure. How do you log "views" on the server end if they're instant? You need to generate the candidate list, but you only know retrospectively whether the user hits the 3-second threshold to count as a view.

For things like sponsored search advertising, where an estimate of how likely someone is to click on an ad is very important for ensuring you show higher-quality ads, this is actually a very big deal.

September 13, 2010 | jon

Google document data is dimensionally simplistic and it should not be compared to a standard OLTP or even relational database. This is not meant to say they don't do an amazing job at what they do but when an entire core data model consists of one table that happens to be extremely easy to partition...well...LUCKY!!!

September 14, 2010 | Jason

Yahoo's YQL presents the web like one big DOM :)

September 14, 2010 | mike503

Each call to the database has a certain overhead. You can gain performance if you can limit the number of calls to the database by using something like stored procs or triggers or MapReduce.

I think that Google has to much data to process it all in some kind of client or middle tier so they process it in the database itself. I don't think I tell something new here.

I don't think that triggers lock records, at least not in Oracle and the code in triggers does do real database work. And triggers are not fired when the transaction is committed but when the inserts, update and deletes are happening. Or just before and after the inserts, update and deletes are happening.

The big disadvantage of triggers is the lack of maintainability, it all becomes too cyclic and the flow

SQL Urban Myths And Delivers Internet Scale OLTP In The Process
The Canonical Cloud Architecture
Justin.Tv's Live Video Broadcasting Architecture
Why Are Facebook, Digg, And Twitter So Hard To Scale?
What The Heck Are You Actually Using NoSQL For?
Playfish's Social Gaming Architecture - 50 Million Monthly Users And Growing
The Updated Big List Of Articles On The Amazon Outage
More Favorites...

becomes difficlut to understand when you have a lot of triggers. Oracle (Tom Kyte) doesn't advice the use of triggers but advices the use of stored procs or packages of stored procs. Oracle's parallelized pipelined functions offer the same functionality as MapReduce: http://www.dbms2.com/2010/02/11/google-mapreduce-patent/

I think that triggers are more used for journaling (storing audit information) than for business logic checks.

Using stuff like MapReduce, stored procs or triggers becomes indeed much harder (slower) when your data is distributed in all different kind of stores. But if you write your stored procs in for instance Java you are not limited in your possibilities.

September 15, 2010 | RC

As a poster above mentioned, I think the difference is more likely to be the difference between re-executing a query over all of the source data when it changes as opposed to incrementally updating a materialized view as the underlying data changes.

Incremental maintenance of materialized views : http://www.csd.uoc.gr/~hy562/Papers/maintenance.pdf

So instead of batching up and processing n web page changes through map-reduce and having multiple 'layers' of web index, a single index can be maintained as changes are detected :

```
onPageChange(old_page, new_page)
{
Calculate index entry diff(old_page, new_page);
Apply diff to index (remove old entries, change some existing entries,
add new entries)
}
```

This could be a similar amount of work per page change as required in the batch approach, but doesn't require multiple layered generations of index and periodic re-index-the-web jobs. I imagine that the number of changed pages is << the number of static pages for any time unit, so the total computation could be less.

To be efficient it requires the abllity to perform random access writes to the index, which is presumably where BigTable helps. Also to be worthwhile, the index changes need to be pushed out to all the replicas as the changes occur, rather than as a batch job.

Lower-latency maintenance of the search index is decoupled from higher throughput of searches as required for the Instant Api. However, both put more pressure on the underlying index storage infrastructure.

September 15, 2010 | Frazer Clement

I wonder why they haven't built a system yet to allow website owners to notify Google directly of content

changes. That would remove all that outdated crawling (polling) replacing it with notifications. Then make that free with some daily quotas so that small websites would still be real-time indexed + add payment options to raise those quotas for large commercial websites. In the end we'll have a realtime web search funded by websites themselves.

September 19, 2010 | Max

Post A New Comment

Enter your information below to add a new comment.

Author:

Author Email (optional):

Author URL (optional):

Post: ↓ | ↑

Some HTML allowed: <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <code> <i> <strike>

Comment Moderation Enabled

Your comment will not appear until it has been cleared by a website editor.

Notify me of follow-up comments via email.

Create Post Preview Post