Real Time analytics for Big Data: Facebook's New Realtime Analytics System

Recently, I was reading Todd Hoff's write-up on [FaceBook real time analytics system](http://highscalability.com/blog/2011/3/22/facebooks-new-realtime-analytics-system-hbase-to-process-20.html). As usual, Todd did an excellent job in summarizing [this video](http://www.facebook.com/video/video.php?v=707216889765&oid=9445547199&comments) from Engineering Manager at Facebook [Alex Himel](http://www.facebook.com/ahimel), Engineering Manager at Facebook.

In [this first post](http://natishalom.typepad.com/nati_shaloms_blog/2011/07/real-time-analytics-for-big-data-an-alternative-approach-to-facebooks-new-realtime-analytics-system.html), I’d like to summarize the case study, and consider some things that weren't mentioned in the summaries. This will lead to an architecture for building your own Realtime Time Analytics for Big-Data that might be easier to implement, using Facebook's experience as a starting point and guide as well as the experience gathered through a recent work with few of GigaSpaces customers. The [second post](http://natishalom.typepad.com/nati_shaloms_blog/2011/07/real-time-analytics-for-big-data-an-alternative-approach.html) provide a summary of that new approach as well as a pattern and a demo for building your own Real Time Analytics system..

**The Business Drive for real time analytics: Time is money**

The main drive for many of the innovations around realtime analytics has to do with competitiveness and cost, just as with most other advances.

For example, during the past few years financial organizations realized that inter-day risk analysis of their customers' portfolios translated to increased profit as they could react faster to profit and loss events.

The same applies to many of the online ecommerce and social sites. Knowing what your users are doing on your site in real time and matching what they do with more targeted information transforms into better conversion rate and better user satisfaction, which means more money in the end.

Todd provides similar reasoning to describe the motivation behind Facebook's new system:

Content producers can see what people like, which will enable content producers to generate more of what people like, which raises the content quality of the web, which gives users a better Facebook experience.

Why now?

**Real time analytics goes mainstream**

The massive transition to online and social applications makes it possible to track user patterns like never before. The correlation between the quality of data that providers track and their business success is closely related: for example, e-commerce customers want to know what their friends think about products or services, right in the middle of their shopping experience. If sites cannot keep up with their thousands of users in real-time, they can lose their customers to sites that can.

So while risk analytics in financial industry was still a fairly small niche of the analytics market the demand for real time analytics in Social, eCommrce and SaaS applications brought the demand for real time analytics to main-stream business under massive load.

No one has time for batch processing anymore.

**Technology advancement**

Newer infrastructures and technologies like tera-scale memory, nosql, parallel processing platforms, and cloud computing, provide new ways to process massive amount of data in shorter time and with lower cost.  As most of the current analytics systems weren’t built to take advantage of these new technologies and capabilities, they haven't been able to adopt to real time requirements without massive changes.

**Hadoop Map/Reduce doesn’t fit the real time business**

One of the hottest trends in the analytics space is the use of [Hadoop](http://hadoop.apache.org/) as almost the de-facto standard for many of the batch processing analytics application. While Hadoop (and Map/Reduce in general) does a pretty good job in processing massive logs and data through parallel batch processing, it wasn’t designed to serve the real time part of the business.

Strong evidence for that can be seen from the moves of those who were known as the “poster children” for Map/Reduce: Google and Yahoo both moved away from Map/Reduce. Google has moved to  [Google Percolator](http://www.theregister.co.uk/2010/09/24/google_percolator/)for its indexing service. Yahoo came-up with a new service[S4[clip_image001](http://blog.programmableweb.com/2010/12/31/yahoos-open-sourced-s4-could-be-a-real-time-cloud-platform/)](http://blog.programmableweb.com/2010/12/31/yahoos-open-sourced-s4-could-be-a-real-time-cloud-platform/) which was designed specifically for real time processing.

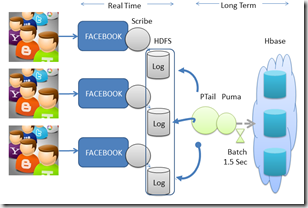
It is therefore not surprising that Facebook reached the same conclusion as it relate to Hadoop:

Hadoop/Hive..Not realtime. Many dependencies. Lots of points of failure. Complicated system. Not dependable enough to hit realtime goals.

 Facebook's Real Time Analytics system

According Todd,  Facebook evaluated a fairly long list of alternatives including as MySQL, an in-memory cache, Hadoop Hive/Map Reduce. I highly recommend reading the full details from [Todd's post](http://highscalability.com/blog/2011/3/22/facebooks-new-realtime-analytics-system-hbase-to-process-20.html).

I tried to outline Facebook architecture based on Todd's summary in the diagram below:

[](http://natishalom.typepad.com/.a/6a00d835457b7453ef014e898cf875970d-pi)

Every user activity triggers an asynchronous event, through AJAX – this event is logged in a tail log using Scribe.  Ptail is used to aggregate the different individual logs into a consolidated stream. The stream is batched in 1.5 sec groupings by Puma which stores the event batch into HBase. The real time logs are kept for a certain period of time and than get cleared from the system.

Obviously this description is a fairly simplistic view – the full details are provided in the original post.

Evaluating the Facebook Architecture

Facebook reasoning behind their technology evaluation seems acceptable for the most part, although there are some obvious concerns.

There were two things that caught my attention:

Memory Counters

(Facebook) felt in-memory counters, for reasons not explained, weren't as accurate as other approaches. Even a 1% failure rate would be unacceptable. Analytics drive money so the counters have to be highly accurate.

It sounds to me that the evaluation was based on memcached. By default, it's not highly available; a failure would result in loss of data. Obviously, that doesn’t apply to some other memory based solutions such as In Memory Data Grids (for example, [GigaSpaces](http://gigaspaces.com/" \t "_blank) and [Coherence](http://www.oracle.com/technetwork/middleware/coherence/index.html) both were designed for high resiliency.)

Cassandra vs HBase

The choice of [HBase](http://hbase.apache.org/" \t "_blank) over [Cassandra](http://cassandra.apache.org/) was also very interesting mainly since Cassandra was developed by another Facebook team to address write scalability.  The choice had to do with the write rate differences between the two alternatives at the time of evaluation:

HBase seemed a better solution based on availability and the write rate. Write rate was the huge bottleneck being solved.

[Eric Hauser](http://www.linkedin.com/in/erichauser)  posted a comment on this analysis which seem to indicate that this issue has been addressed with Cassandra:

When Facebook engineers started the project 6 months ago, Cassandra did not have distributed counters which is now committed in trunk. Twitter is making a large investment on Facebook for real-time analytics (see Rainbird). Write rate should be less of of bottleneck for Cassandra now that counter writes are spread out across multiple hosts. For HBase, every counter is still bound by the performance of single region server? A performance comparison of the two would be interesting

Eric's comment is indicative of how dynamic the NoSQL space is. I’d be interested in how different the technology selection would be now.

The Architecture

There are a few common principles that drive the architecture for this type of system:

* Events logging needs to be extremely fast, to minimize the latency impact on the site.
* Events need to be reliable, otherwise the entire system's accuracy is questionable and the data is devalued.
* The real time data in the form of logs is kept for a certain period of time (x hours or x days)
* Write scalability is key.
* Post processing can happens in batches, the size of the batch depends on how \*real-time\* we need this data to be.
* Writes to a backend database need to be done asynchronously.

Facebook seem to follow those same principles in their architecture while keeping the system at scale at a fairly impressive rate.

There are a few questions that are still open as I don’t have full visibility into their system:

* Are Ptail and Puma centralized components? If so, don't they pose a potential bottleneck? Based on Todd's summary and Alex' presentation, it seems that the way Facebook scaled their system is by splitting the PTail stream by categories of events so each type can be handled by a different data center.
* Puma batches event logs in memory before it writes them into Hbase – what happens if Puma fails before the batch is written?
* The solution seemed to be limited to handling simple counters. This seem to be a fairly severe limitation, as many systems need to produce more complex relationships even during the real time part of the system, as also indicated as part of the future enhancements, as shown:

"(We need to know) how many people acrossa time window liked a URL. Easy to do in MapReduce, hard to do with a naive counter solution.”

* In one point, it is noted that Facebook chose not to rely on memory for counters. However, throughout the description it seems that there is still strong reliance on keeping data within memory boundaries:

“(We) write extremely lean log lines. The more compact the log lines **the more can be stored in memory**..”

“(We) batch for 1.5 seconds on average. Would like to batch longer but they have so many URLs that they **run out of memory when creating a hashtable**”

Looking backward – wouldn’t it be better to store the data in-memory in the first place? Why add the extra architecture components, if you're able to make memory work for you? This is a crucial question, of course, because it focuses attention on memory availability. As mentioned, though, there are in-memory data grids that are designed for just this kind of situation.

* It is noted that Puma writes it batches to Hbase sequentially:

“(We) wait for last flush to complete for starting new batch to avoid lock contention issues...”

What if this rate is lower than the actual write rate?

Realtime.Analytics.next

Interestingly enough, I was asked to draw a solution for similar challenge for a voice recording system: collect lots data from various sources and process them in real time. The good news is that it's doable, as shown by Facebook's success.

The better news is that it's actually fairly easy. We can add rich query capabilities, elastic scaling, database and platform neutrality, evolution of data, and more without making things unnecessarily difficult - it's not so much that this is the next generation of realtime analytics as much as map/reduce and the hbase approach used by facebook is the *previous* generation.

I'll describe how it can be done more simply in the [next post](http://natishalom.typepad.com/nati_shaloms_blog/2011/07/Real%20Time%20Analytics%20for%20Big%20Data:%20An%20Alternative%20Approach).

References

* [Video: Realtime Analytics for Big Data: A Facebook Case Study](http://mrkn.co/forums/java/general/562.html)
* [Real Time Analytics for Big Data: An Alternative Approach](http://natishalom.typepad.com/nati_shaloms_blog/2011/07/real-time-analytics-for-big-data-an-alternative-approach.html)
* [Big data in real time is no fantasy](http://gigaom.com/cloud/big-data-in-real-time-is-no-fantasy/) (GigaOm)
* [Facebook's New Realtime Analytics System: HBase To Process 20 Billion Events Per Day](http://highscalability.com/blog/2011/3/22/facebooks-new-realtime-analytics-system-hbase-to-process-20.html)
* [Read/write scale without complete re-write](http://natishalom.typepad.com/nati_shaloms_blog/2011/06/readwrite-scale-without-complete-re-write.html)
* [How Map/Reduce and the Cloud are Affecting Analytics Applications](http://natishalom.typepad.com/nati_shaloms_blog/2009/10/how-mapreduce-and-the-cloud-are-affecting-analytics-applications.html)
* [Is MapReduce going mainstream?](http://natishalom.typepad.com/nati_shaloms_blog/2008/10/is-mapreduce-going-to-main-stream.html)
* [Memory is the New Disk for the Enterprise](http://natishalom.typepad.com/nati_shaloms_blog/2010/03/memory-is-the-new-disk-for-the-enterprise.html)