**Facebook Engineering: What is Facebook's architecture?**

This is a follow-up question to [Internet Infrastructure: How does Quora's architecture differ from Facebook's?](http://www.quora.com/Internet-Infrastructure/How-does-Quoras-architecture-differ-from-Facebooks).

**7 ANSWERS**

[**ASK TO ANSWER**](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

**[Michaël Figuière](http://www.quora.com/Micha%C3%ABl-Figui%C3%A8re" \t "_blank)**, Engineer at DataStax

2.1k upvotes by [Olaoluwa 'Ola' Okelola](http://www.quora.com/Olaoluwa-Ola-Okelola" \t "_blank) (Facebook Engineer), [Andrew 'Boz' Bosworth](http://www.quora.com/Andrew-Boz-Bosworth) (I have been an engineer at Facebook forever), [Keith Adams](http://www.quora.com/Keith-Adams) (Engineer at Facebook), [Tom Cook](http://www.quora.com/Tom-Cook) (Facebook TechOps '07-'11), [Ozgur Demir](http://www.quora.com/Ozgur-Demir" \t "_blank) (I worked for Facebook between Oct 2007 - Feb 20...), [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

From various readings and conversations I had, my understanding of Facebook's current architecture is:

* Web front-end written in PHP. Facebook's HipHop Compiler [1] then converts it to C++ and compiles it using g++, thus providing a high performance templating and Web logic execution layer.
* Because of the limitations of relying entirely on static compilation, Facebook's started to work on a HipHop Interpreter [2] as well as a HipHop Virtual Machine which translate PHP code to HipHop ByteCode [3].
* Business logic is exposed as services using Thrift [4]. Some of these services are implemented in PHP, C++ or Java depending on service requirements (some other languages are probably used...)
* Services implemented in Java don't use any usual enterprise application server but rather use Facebook's custom application server. At first this can look as wheel reinvented but as these services are exposed and consumed only (or mostly) using Thrift, the overhead of Tomcat, or even Jetty, was probably too high with no significant added value for their need.
* Persistence is done using MySQL, Memcached [5], Hadoop's HBase [6]. Memcached is used as a cache for MySQL as well as a general purpose cache.
* Offline processing is done using Hadoop and Hive.
* Data such as logging, clicks and feeds transit using Scribe [7] and are aggregating and stored in HDFS using Scribe-HDFS [8], thus allowing extended analysis using MapReduce
* BigPipe [9] is their custom technology to accelerate page rendering using a pipelining logic
* Varnish Cache [10] is used for HTTP proxying. They've prefered it for its high performance and efficiency [11].
* The storage of the billions of photos posted by the users is handled by Haystack, an ad-hoc storage solution developed by Facebook which brings low level optimizations and append-only writes [12].
* Facebook Messages is using its own architecture which is notably based on infrastructure sharding and dynamic cluster management. Business logic and persistence is encapsulated in so-called 'Cell'. Each Cell handles a part of users ; new Cells can be added as popularity grows [13]. Persistence is achieved using HBase [14].
* Facebook Messages' search engine is built with an inverted index stored in HBase [15]
* Facebook Search Engine's implementation details are unknown as far as I know
* The typeahead search uses a custom storage and retrieval logic [16]
* Chat is based on an Epoll server developed in Erlang and accessed using Thrift [17]
* They've built an automated system that respond to monitoring alert by launching the appropriated repairing workflow, or escalating to humans if the outage couldn't be overcome [18].

About the resources provisioned for each of these components, some information and numbers are known:

* Facebook is estimated to own more than 60,000 servers [18]. Their recent datacenter in Prineville, Oregon is based on entirely self-designed hardware [19] that was recently unveiled as Open Compute Project [20].
* 300 TB of data is stored in Memcached processes [21]
* Their Hadoop and Hive cluster is made of 3000 servers with 8 cores, 32 GB RAM, 12 TB disks that is a total of 24k cores, 96 TB RAM and 36 PB disks [22]
* 100 billion hits per day, 50 billion photos, 3 trillion objects cached, 130 TB of logs per day as of july 2010 [22]

[1] *HipHop for PHP*: [http://developers.facebook.com/b...](http://developers.facebook.com/blog/post/358)   
[2] *Making HPHPi Faster*: [http://www.facebook.com/note.php...](http://www.facebook.com/note.php?note_id=10150336948348920)   
[3]*The HipHop Virtual Machine*: [http://www.facebook.com/note.php...](http://www.facebook.com/note.php?note_id=10150415177928920)   
[4] *Thrift*: <http://thrift.apache.org/>   
[5] *Memcached*: <http://memcached.org/>   
[6] *HBase*: <http://hbase.apache.org/>   
[7] *Scribe*: <https://github.com/facebook/scribe>   
[8] *Scribe-HDFS*: [http://hadoopblog.blogspot.com/2...](http://hadoopblog.blogspot.com/2009/06/hdfs-scribe-integration.html)   
[9] *BigPipe*: [http://www.facebook.com/notes/fa...](http://www.facebook.com/notes/facebook-engineering/bigpipe-pipelining-web-pages-for-high-performance/389414033919)   
[10] *Varnish Cache*: <http://www.varnish-cache.org/>   
[11] *Facebook goes for Varnish*: [http://www.varnish-software.com/...](http://www.varnish-software.com/customers/facebook)   
[12] *Needle in a haystack*: efficient storage of billions of photos:[http://www.facebook.com/note.php...](http://www.facebook.com/note.php?note_id=76191543919)   
[13] *Scaling the Messages Application Back End*: [http://www.facebook.com/note.php...](http://www.facebook.com/note.php?note_id=10150148835363920)   
[14] *The Underlying Technology of Messages*: [https://www.facebook.com/note.ph...](https://www.facebook.com/note.php?note_id=454991608919)   
[15] *The Underlying Technology of Messages Tech Talk*:[http://www.facebook.com/video/vi...](http://www.facebook.com/video/video.php?v=690851516105)   
[16] *Facebook's typeahead search architecture*: [http://www.facebook.com/video/vi...](http://www.facebook.com/video/video.php?v=432864835468)   
[17] *Facebook Chat*: [http://www.facebook.com/note.php...](http://www.facebook.com/note.php?note_id=14218138919)   
[18] *Who has the most Web Servers?*: [http://www.datacenterknowledge.c...](http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/)   
[19] B*uilding Efficient Data Centers with the Open Compute Project*:[http://www.facebook.com/note.php...](http://www.facebook.com/note.php?note_id=10150144039563920)   
[20] *Open Compute Project*: <http://opencompute.org/>   
[21] *Facebook's architecture presentation at Devoxx 2010*:[http://www.devoxx.com](http://www.devoxx.com/)   
[22] *Scaling Facebook to 500 millions users and beyond*:[http://www.facebook.com/note.php...](http://www.facebook.com/note.php?note_id=409881258919)

**[Nati Shalom](http://www.quora.com/Nati-Shalom" \t "_blank)**, Wrote a series of patterns for... [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

24 upvotes by [Ravi Dronamraju](http://www.quora.com/Ravi-Dronamraju), [Amit Chaudhary](http://www.quora.com/Amit-Chaudhary), [Uday Kalbhor](http://www.quora.com/Uday-Kalbhor-1" \t "_blank), [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

Facebook Real Time Analytics system is based on Scribe to log all incomming links from like and comments request on a user page. Store them into HDFS than pull them out using Puma and store them on HBase in batches.  
  
I wrote a detailed post outlining the Facebook Real Time analytics architecture here:

Real Time analytics for Big Data: Facebook's New Realtime Analytics System  - <http://ht.ly/8OGHD>

This post include references to vidoe cast and other useful references on that regard.

**[Saumye Srivastava](http://www.quora.com/Saumye-Srivastava" \t "_blank)**, Quoraddict. Tech-Enthusiast. D... [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

39 upvotes by [Frank Kelly](http://www.quora.com/Frank-Kelly-1), [Purav Shah](http://www.quora.com/Purav-Shah" \t "_blank), [Jack Webb](http://www.quora.com/Jack-Webb-1), [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

At the scale that Facebook operates, a lot of traditional approaches to serving web content breaks down or simply aren’t practical. The challenge for Facebook’s engineers has been to keep the site up and running smoothly in spite of handling close to **half a billion active users**. This article takes a look at some of the software and techniques they use to accomplish that.  
  
Facebook’s scaling challenge  
Before we get into the details, here are a few factoids to give you an idea of the scaling challenge that Facebook has to deal with:

* Facebook serves **570 billion page views per month** (according to Google Ad Planner).
* There are more photos on Facebook than all other photo sites combined (including sites like Flickr).
* More than **3 billion photos** are uploaded every month.
* Facebook’s systems serve **1.2 million photos per second**. This doesn’t include the images served by Facebook’s CDN.
* More than **25 billion pieces of content** (status updates, comments, etc) are shared every month.
* Facebook has more than **30,000 servers** (and this number is from last year!)

**Software that helps Facebook scale**  
In some ways Facebook is still a LAMP site (kind of), but it has had to change and extend its operation to incorporate a lot of other elements and services, and modify the approach to existing ones.  
For example:

* Facebook still uses PHP, but it has built a compiler for it so it can be turned into native code on its web servers, thus boosting performance.
* Facebook uses Linux, but has optimized it for its own purposes (especially in terms of network throughput).
* Facebook uses MySQL, but primarily as a key-value persistent storage, moving joins and logic onto the web servers since optimizations are easier to perform there (on the “other side” of the Memcached layer).

Then there are the custom-written systems, like Haystack, a highly scalable object store used to serve Facebook’s immense amount of photos, or Scribe, a logging system that can operate at the scale of Facebook (which is far from trivial).  
But enough of that. Let’s present (some of) the software that Facebook uses to provide us all with the world’s largest social network site.  
  
**Memcached**  
Memcached is by now one of the most famous pieces of software on the internet. It’s a distributed memory caching system which Facebook (and a ton of other sites) use as a caching layer between the web servers and MySQL servers (since database access is relatively slow). Through the years, Facebook has made a ton of optimizations to Memcached and the surrounding software (like optimizing the network stack).  
Facebook runs thousands of Memcached servers with tens of terabytes of cached data at any one point in time. It is likely the world’s largest Memcached installation.  
  
**HipHop for PHP**  
PHP, being a scripting language, is relatively slow when compared to code that runs natively on a server. HipHop converts PHP into C++ code which can then be compiled for better performance. This has allowed Facebook to get much more out of its web servers since Facebook relies heavily on PHP to serve content.  
A small team of engineers (initially just three of them) at Facebook spent 18 months developing HipHop, and it is now live in production.  
  
**Haystack**  
Haystack is Facebook’s high-performance photo storage/retrieval system (strictly speaking, Haystack is an object store, so it doesn’t necessarily have to store photos). It has a ton of work to do; there are more than 20 billion uploaded photos on Facebook, and each one is saved in four different resolutions, resulting in more than 80 billion photos.  
And it’s not just about being able to handle billions of photos, performance is critical. As we mentioned previously, Facebook serves around 1.2 million photos *per second*, a number which doesn’t include images served by Facebook’s CDN. That’s a staggering number.  
  
**BigPipe**  
BigPipe is a dynamic web page serving system that Facebook has developed. Facebook uses it to serve each web page in sections (called “pagelets”) for optimal performance.  
For example, the chat window is retrieved separately, the news feed is retrieved separately, and so on. These pagelets can be retrieved in parallel, which is where the performance gain comes in, and it also gives users a site that works even if some part of it would be deactivated or broken.  
  
**Cassandra**  
Cassandra is a distributed storage system with no single point of failure. It’s one of the poster children for the NoSQL movement and has been made open source (it’s even become an Apache project). Facebook uses it for its Inbox search.  
  
**Scribe**  
Scribe is a flexible logging system that Facebook uses for a multitude of purposes internally. It’s been built to be able to handle logging at the scale of Facebook, and automatically handles new logging categories as they show up (Facebook has hundreds).  
  
**Hadoop and Hive**  
Hadoop is an open source map-reduce implementation that makes it possible to perform calculations on massive amounts of data. Facebook uses this for data analysis (and as we all know, Facebook has massive amounts of data). Hive originated from within Facebook, and makes it possible to use SQL queries against Hadoop, making it easier for non-programmers to use.  
Both Hadoop and Hive are open source (Apache projects) and are used by a number of big services, for example Yahoo and Twitter.  
  
**Thrift**  
Facebook uses several different languages for its different services. PHP is used for the front-end, Erlang is used for Chat, Java and C++ are also used in several places (and perhaps other languages as well). Thrift is an internally developed cross-language framework that ties all of these different languages together, making it possible for them to talk to each other. This has made it much easier for Facebook to keep up its cross-language development.  
Facebook has made Thrift open source and support for even more languages has been added.  
  
**Varnish**  
Varnish is an HTTP accelerator which can act as a load balancer and also cache content which can then be served lightning-fast.  
Facebook uses Varnish to serve photos and profile pictures, handling billions of requests every day. Like almost everything Facebook uses, Varnish is open source.  
  
**Other things that help Facebook run smoothly**  
  
**Gradual releases and dark launches**  
Facebook has a system they called Gatekeeper that lets them run different code for different sets of users (it basically introduces different conditions in the code base). This lets Facebook do gradual releases of new features, A/B testing, activate certain features only for Facebook employees, etc.  
Gatekeeper also lets Facebook do something called “dark launches”, which is to activate elements of a certain feature behind the scenes before it goes live (without users noticing since there will be no corresponding UI elements). This acts as a real-world stress test and helps expose bottlenecks and other problem areas before a feature is officially launched. Dark launches are usually done two weeks before the actual launch.  
  
  
**Profiling of the live system**  
Facebook carefully monitors its systems (something we here at Pingdom of course approve of), and interestingly enough it also monitors the performance of every single PHP function in the live production environment. This profiling of the live PHP environment is done using an open source tool called XHProf.  
Gradual feature disabling for added performance  
If Facebook runs into performance issues, there are a large number of levers that let them gradually disable less important features to boost performance of Facebook’s core features.  
The things we didn’t mention  
We didn’t go much into the hardware side in this article, but of course that is also an important aspect when it comes to scalability. For example, like many other big sites, Facebook uses a CDN to help serve static content. And then of course there is the huge data center Facebook is building in Oregon to help it scale out with even more servers.  
And aside from what we have already mentioned, there is of course a ton of other software involved. However, we hope we were able to highlight some of the more interesting choices Facebook has made.  
  
Source: [Exploring the software behind Facebook, the world's largest site](http://royal.pingdom.com/2010/06/18/the-software-behind-facebook/)

**Anonymous**

14 upvotes by [James Thornton](http://www.quora.com/James-Thornton), [Tom Robinson](http://www.quora.com/Tom-Robinson), [Zhijin Li](http://www.quora.com/Zhijin-Li" \t "_blank), [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

* Data Warehousing and Analytics Infrastructure at Facebook:[http://borthakur.com/ftp/sigmodw...](http://borthakur.com/ftp/sigmodwarehouse2010.pdf" \t "_blank)
* Apache Hadoop Goes Realtime at Facebook: [http://borthakur.com/ftp/Realtim...](http://borthakur.com/ftp/RealtimeHadoopSigmod2011.pdf)  (For more see [Dhruba Borthakur](http://www.quora.com/Dhruba-Borthakur)'s blog: <http://hadoopblog.blogspot.com/>  )
* Scalable Memory Allocation using jemalloc (by [Jason Evans](http://www.quora.com/Jason-Evans)):[http://www.facebook.com/notes/fa...](http://www.facebook.com/notes/facebook-engineering/scalable-memory-allocation-using-jemalloc/480222803919)  (also [Who wrote jemalloc, and what motivated its creation and implementation?](http://www.quora.com/Who-wrote-jemalloc-and-what-motivated-its-creation-and-implementation) )
* [Tornado (web framework)](http://www.quora.com/Tornado-web-framework): <https://github.com/facebook/tornado>  (blog post by [Bret Taylor](http://www.quora.com/Bret-Taylor): [http://bret.appspot.com/entry/to...](http://bret.appspot.com/entry/tornado-web-server)  )
* [Facebook Infrastructure: How is Facebook's search speed so fast?](http://www.quora.com/Facebook-Infrastructure/How-is-Facebooks-search-speed-so-fast)

**Anonymous**

12 upvotes by [Mustafa Kirac](http://www.quora.com/Mustafa-Kirac), [Yalım K. Gerger](http://www.quora.com/Yal%C4%B1m-K-Gerger), [Daniel Schildt](http://www.quora.com/Daniel-Schildt), [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

In regards to Michaël Figuière's excellent response, the only thing I can see that is no longer true is their use of Cassandra.  It is no longer in use inside of Facebook.  This got flagged as needing to be a comment to another answer, but you can't comment on an answer as anonymous.   So until Quora changes that policy, this will have to stand alone as an answer.

**[Simon Gardner](http://www.quora.com/Simon-Gardner" \t "_blank)**, 23 years developing software f... [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

5 upvotes by [Julie Prentice](http://www.quora.com/Julie-Prentice), [Heriberto Acosta](http://www.quora.com/Heriberto-Acosta), [Aditi Modak](http://www.quora.com/Aditi-Modak" \t "_blank), [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

Here's a very interesting presentation by a Facebook engineer about how their architecture has evolved. [Evolution of Code Design at Facebook](http://www.infoq.com/presentations/Evolution-of-Code-Design-at-Facebook)

**[Kr Ace Kumar Ramaraju](http://www.quora.com/Kr-Ace-Kumar-Ramaraju" \t "_blank)**, Created web apps in PHP,Flask,... [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

8 upvotes by [Wang Jun](http://www.quora.com/Wang-Jun-2), [Toby Thain](http://www.quora.com/Toby-Thain), [Riyad Parvez](http://www.quora.com/Riyad-Parvez), [(more)](http://www.quora.com/Facebook-Engineering/What-is-Facebooks-architecture)

Facebook Chat is implemented in Erlang.  
Facebook uses Tornado non blocking web server too.