Facebook Architecture: Breaking it Open (01/03/2012)
http://www.slideshare.net/AditiTechnologies/facebook-architecture-breaking-it-open
A good summary of Facebook technology stacks, from front end (php) to back end (memcache, haystack, message, HBase, etc).

Article: Exploring the software behind Facebook (06/18/2010)

http://royal.pingdom.com/2010/06/18/the-software-behind-facebook/

Very nice review of Facebook technologies.

Slides: Facebook Technology Stack (2011)
http://www.slideshare.net/meet.hak/facebook-technology-stack (25 slides)

Video: Scale at Facebook (Aditya Agarwal, 2010, 47 min presentation + 14 min Q&A)

http://www.infoq.com/presentations/Scale-at-Facebook

Facebook architecture:

- PHP improvements (Hip Hop)
- How Newsfeed works (from 21st min)
- Memcache (distributed index)
- MySQL (permanent storage): a key-value (id-data) system, no joins in production.
- Evolution of photo storage (from NFS to Cachr to Haystack)

Many slides are identical to the slides in the video on 11/22/2008:
http://www.slideshare.net/mysqlops/facebook-architecture

Facebook Culture: (from 42nd min)

- Move fast and break things
- Huge impact with small teams
- Be bold and innovate

Blog Article: Scale at Facebook (10/29/2009)
http://www.25hoursaday.com/weblog/2009/10/29/FacebookSeattleEngineeringRoadShowMikeShroepferOnEngineeringAtScaleAtFacebook.aspx
Nice summary of facebook technologies. (Read it after watching the video above.)

"An example of a dedicated internal service at Facebook is MultiFeed. This is the service which takes the tens of thousands of updates from a user's friends and whittles them down to the forty five updates that are shown on the homepage when a user logs in. When a user publishes an update, it is both written to MySQL and published to a their memcache cluster using Scribe. There is a cache of the fifty most recent items each user has performed always stored in memory. When a user hits the home page, an aggregator takes the list of the user's friends and then queries a bunch of leaf nodes for activities their friends have performed. Each leaf node does some filtering before returning data to the aggregator which in turn

does some filtering of its own before returning a list of story IDs to the calling application. The application then looks up the actual items from memcache given their story IDs and then renders the home page.

"Multifeed is a custom distributed system which takes the tens of thousands of updates from friends and picks the 45 that are either most relevant or most recent. Bob updates status to DB and then Scribe pushes this to a Multifeed server which is a cache of the most recent 50 items the user has performed. This caches data for every user in the system. When a user shows up, the user hits an aggregator which takes the list of friends and then each leaf node does some filtering then gets back to the aggregator which then does some filtering and returns story IDs. Then the aggregator fills out the data from memcache given the story ID."

Facebook blog: Scaling out (08/20/2008)
https://www.facebook.com/notes/facebook-engineering/scaling-out/23844338919
Details on the maintaining consistencies between the data center in California and Virginia.

- **Cache Consistency**: We made a small change to MySQL that allows us to tack on extra information in the replication stream that is updating the slave database. We used this feature to append all the data objects that are changing for a given query and then the slave database "sees" these objects and is responsible for deleting the value from cache after it performs the update to the database.
- **Page routing**

FB Blog: Keeping Up (12/21/2007)
https://www.facebook.com/notes/facebook/keeping-up/7899307130

- Use datacenter in Virginia for read-only access.
- Only about 10% of our traffic causes a modifying operation.
- MySQL has a great replication feature that allows us to, in real time, stream all the modifications happening on a California MySQL server to another one in Virginia.
- Replication happens so fast, even across the country, that the Virginia servers are almost never more than 1 or 2 seconds behind the California servers.
- Figure out a way for memcached servers to replicate data concurrently with the MySQL databases: embed extra information into the MySQL replication stream to properly update memcached in Virginia.
- Almost 30% of our traffic has been served from Virginia.

MITBBS Article

http://www.mitbbs.com/article_t/JobHunting/32676563.html
There will be one interview that focuses on architecture. These interviews focus on:

- systems: think distributed systems
- APIs: focused on building/implementing a structure/product.

One example of a question:
How to build a chat system that handles millions of concurrently connected users?
Be sure to be very thorough in your explanation, we are generally looking for a boxes and arrows diagram on the whiteboard.

A couple of things to focus on in this interview:

- Communication is key, you will be steering the conversation, and it will be up to you to understand the problem and ask clarifying questions.
- Our engineers will be focusing on your familiarity with complex systems.

Some topics you should be familiar with:

- Concurrency (threads, deadlock, starvation, consistency, coherence)
- Networking (IPC, TCP/IP)
- Abstraction (understanding how OS, filesystem, and database works)
- Real-world performance (relative performance RAM, disk, your network, SSD)
- Availability and Reliability (durability, understanding how things can fail)
- Data storage (RAM vs. durable storage, compression, byte sizes)
- CAP Theorem
- Byte math

Note that we're not looking for you to be an expert in ALL of these, but you should know enough of them to weigh design considerations and know when to consult an expert.
For practice:

- Work with a fellow engineer on mock design sessions;
- Dig into the implementation and performance of an open source system, understand things like how the system stores data on disk and how it compacts data;
- Be familiar with how databases and operating systems work;
- Practice on a whiteboard.

Quora: What is Facebook's Architecture?
http://www.quora.com/What-is-Facebooks-architecture

* Making HPHPi Faster (10/18/2011)

https://www.facebook.com/note.php?note_id=10150336948348920

HPHPi: HipHop interpreter

* HipHop Virtual Machine (12/09/2011)

https://www.facebook.com/note.php?note_id=10150415177928920

HPHPc: HipHop compiler

* Scribe:

https://github.com/facebookarchive/scribe

- Server for aggregating log data streamed in real-time from many other servers.
- A scalable framework useful for logging a wide array of data.
- Built on top of Thrift.

* Scribe-HDFS:

http://hadoopblog.blogspot.com/2009/06/hdfs-scribe-integration.html

* BigPipe

https://www.facebook.com/notes/facebook-engineering/bigpipe-pipelining-web-pages-for-high-performance/389414033919 (06/04/2010)

http://www.cubrid.org/blog/dev-platform/faster-web-page-loading-with-facebook-bigpipe/

- Implemented entirely in PHP and JavaScript.
- By overlapping the web server's generation time with the browser's rendering time, we can not only reduce the end-to-end latency but also make the initial response of the web page visible to the user much earlier, thereby significantly reducing user perceived latency.
- BigPipe breaks the page generation process into several stages:
    1. Request parsing: web server parses and sanity checks the HTTP request.
    2. Data fetching: web server fetches data from storage tier.
    3. Markup generation: web server generates HTML markup for the response.
    4. Network transport: the response is transferred from web server to browser.
    5. CSS downloading: browser downloads CSS required by the page.
    6. DOM tree construction and CSS styling: browser constructs DOM tree of the document, and then applies CSS rules on it.
    7. JavaScript downloading: browser downloads JavaScript resources referenced by the page.
    8. JavaScript execution: browser executes JavaScript code of the page.

  The first three stages are executed by the web server, and the last four stages are executed by the browser. Each pagelet must go through all these stages sequentially, but BigPipe enables several pagelets to be executed simultaneously in different stages.

Related: Making Facebook 2x faster (02/18/2010)

https://www.facebook.com/note.php?note_id=307069903919

TTI: Time-to-Interact

* *Varnish Cache*: HTTP accelerator

“Facebook uses Varnish to serve billions of requests every day to users around the world. We like its simple architecture, which is designed for modern operating systems and find that it does not consume much CPU while handling heavy loads. Varnish is our favored HTTP cache and we use it heavily; whenever you load photos and profile pictures of your friends on Facebook, there's a very good chance that Varnish is involved."

http://www.varnish-cache.org/

http://en.wikipedia.org/wiki/Varnish_(software)#Architecture

- Stores data in virtual memory and leaves the task of deciding what is stored in memory and what gets paged out to disk to the operating system. This helps avoid the situation where the operating system starts caching data while it is moved to disk by the application.
- Heavily threaded, with each client connection being handled by a separate worker thread. When the configured limit on the number of active worker threads is reached, incoming connections are placed in an overflow queue; when this queue reaches its configured limit incoming connections will be rejected.

* **Who has the most Web Servers? (July 2013)**

http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers

* **Building Efficient Data Centers with the Open Compute Project (04/07/2011)**

https://www.facebook.com/note.php?note_id=10150144039563920

http://www.opencompute.org/

* **Scaling Facebook to 500 millions users and beyond (07/21/2010)**

https://www.facebook.com/note.php?note_id=409881258919

Facebook culture

HS: Facebook At 13 Million Queries Per Second Recommends: Minimize Request Variance (11/04/2010)

http://highscalability.com/blog/2010/11/4/facebook-at-13-million-queries-per-second-recommends-minimiz.html

Facebook Architecture (12/09/2012)

http://blog-bhaskaruni.blogspot.in/2012/12/facebook-architecture.html

Stackoverflow: Facebook Architecture

http://stackoverflow.com/questions/3533948/facebook-architecture

---

Video: Hacker Way: Facebook's High Performance Server Infrastructure

(04/30/2014, 50 min, not worth the time)

https://www.youtube.com/watch?v=zBV9TkTXxg0
HHVM: Hip Hop Virtual Machine
Hack: Enhanced PHP feature.

Video: Facebook Infrastructure (2013, 39 min**,** not really worth the time.**)**
http://www.infoq.com/presentations/scaling-operations-facebook
Pedro Canahuati (Facebook SRE) describes how Facebook's operations maintain their infrastructure, including challenges faced and lessons learned: prioritizing calls, managing technical debt, incident management.

Video: Evolution of Facebook Code Design (Nick Schrock, 03/11/2011, 60 min)

http://www.infoq.com/presentations/Evolution-of-Code-Design-at-Facebook

49 min of presentation and 11 min of Q & A.

- Improvements of php that calls memcache (FBObject and Preparable in PHP)
- Ent: Facebook Entities, Facebook's business object layer (usage in PHP)

Code in PHP, not really worth the time!

Video: Moving Fast at Scale (Robert Johnson, 12/04/2009, 60 min)

http://www.infoq.com/presentations/Facebook-Moving-Fast-at-Scale

40 min presentation and 20 min of Q & A. Talked about some general software engineering principles. Not really worth the time.

Video: Facebook Architecture (Aditya Agarwal, 11/22/2008, a little outdated)

http://www.infoq.com/presentations/Facebook-Software-Stack (61 min)

http://www.slideshare.net/mysqlops/facebook-architecture (slides)
Aditya Agarwal discusses Facebook's architecture, the software stack used, presenting the advantages and disadvantages of its major components: LAMP (PHP, MySQL), Memcache, Thrift, Scribe. (dhu: Part 4 talked about Newsfeed.)
SMC: Service Management Console