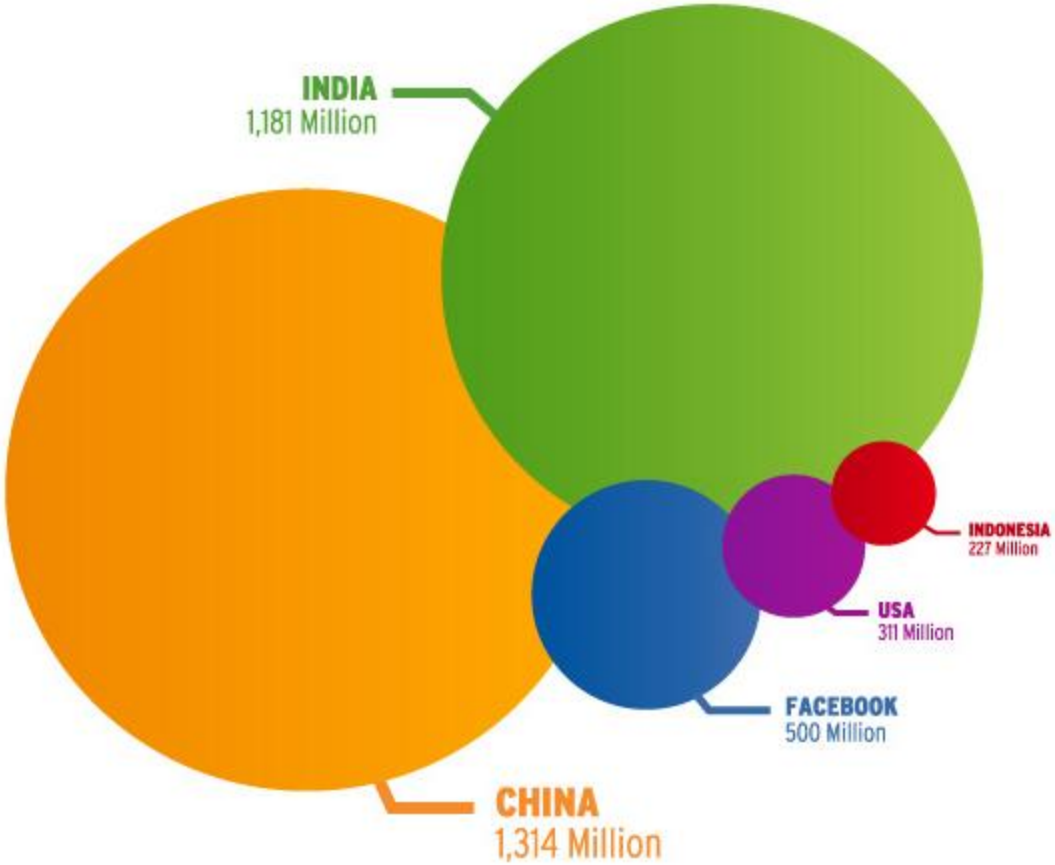
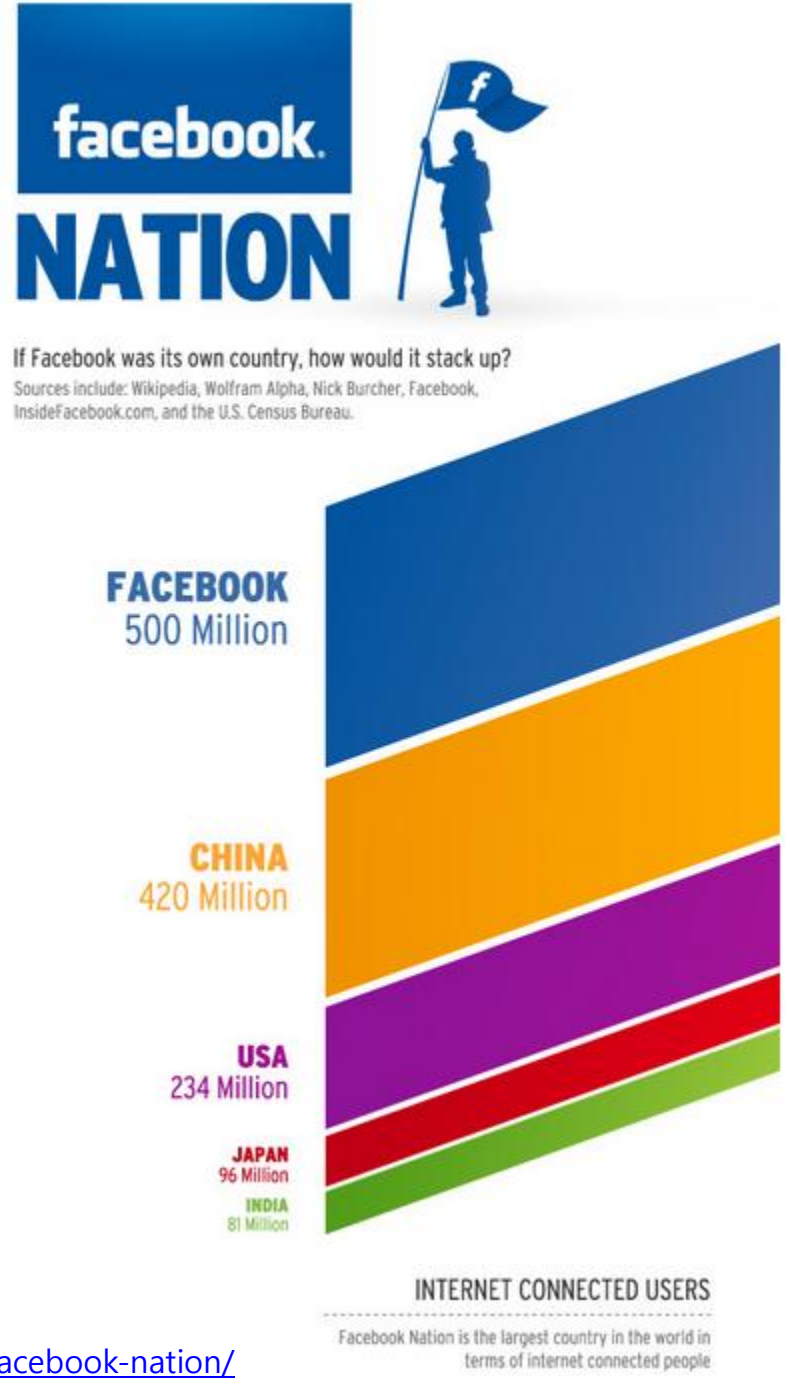
facebook

facebook Architecture for (over) 600M Users

’2011 1st Issue of TechLab Report
Neowiz Games. TechLab
http://rainblue.kr
kikiki.blue@gmail.com

http://kr.blog.yahoo.com/lhd1487





Worldwide Facebook Users

642 613 700

POPULATION

Facebook Nation is the 3rd largest country in the world

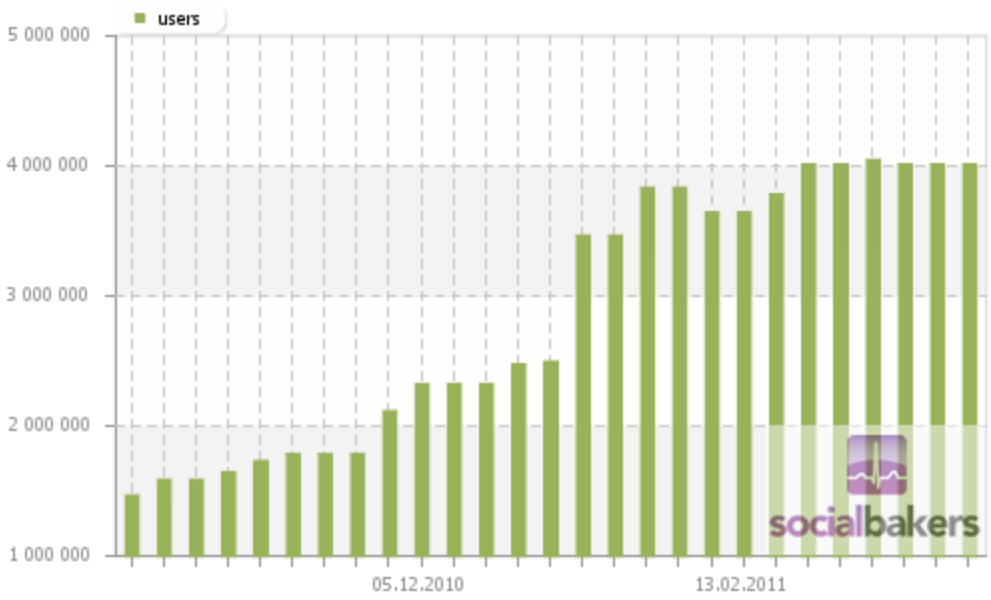
TOP5 Countries on Facebook

1.	United States	152 189 880
2.	Indonesia	35 174 940
3.	United Kingdom	28 940 400
4.	Turkey	26 417 820
5.	Philippines	22 651 600

출처 : http://blog.fliptop.com/facebook-nation/

Facebook Users in Korea

Total Facebook Users:	**4 010 980**	Penetration of population:	**8.25%**
Position in the list:	**27.**	Penetration of online pop.:	**10.17%**
Average CPC(Cost per Click):	**$0.21**	Average CPM(Cost Per Mile):	**$0.07**



출처 : http://www.socialbakers.com

Social Network Users in Korea

View data for: South Korea

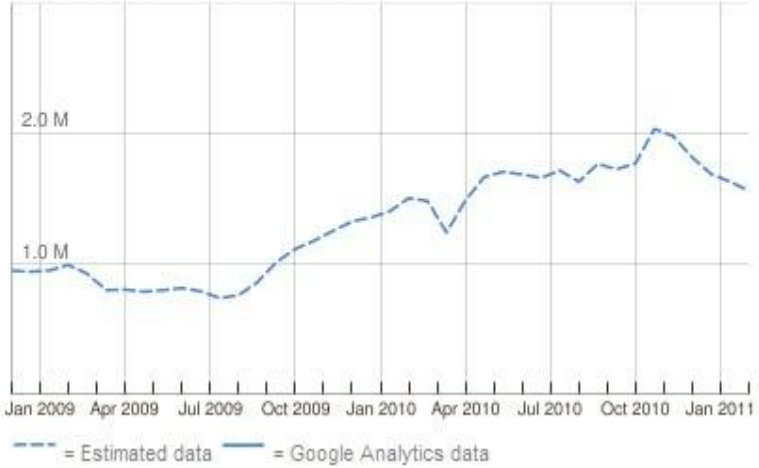
Cyworld - 2010년 11월을 기점으로 traffic / unique visitor도 감소중

Data: January 2011

Traffic statistics All traffic statistics are estimates

	Country	Worldwide
Unique visitors (estimated cookies)	8.9M	11M
Unique visitors (users)	16M	16M
Reach	42.9%	1.0%
Page views	950M	1B
Total visits	120M	130M
Avg visits per visitor	7.3	8.1
Avg time on site	10:40	10:50

Daily Unique Visitors (cookies)



View data for: South Korea

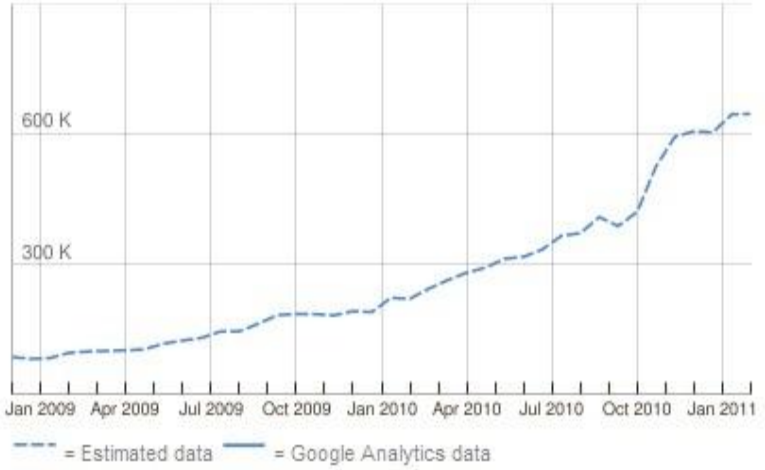
Facebook - 2011년 1월, 740만의 unique visitors, 지속적인 상승곡선

Data: January 2011

Traffic statistics All traffic statistics are estimates

	Country	Worldwide
Unique visitors (estimated cookies)	4.6M	1.9B
Unique visitors (users)	7.4M	600M
Reach	20.1%	39.1%
Page views	1.4B	840B
Total visits	45M	24B
Avg visits per visitor	6.1	41
Avg time on site	20:00	25:00

Daily Unique Visitors (cookies)



출처 : http://iblur.tistory.com/328

Games applications Facebook Statistics

« Previous | 1 | 2 | 3 | 4 ... 29 ... 57 ... 84 ... 112 | Next »

Last 6 months | Last 3 months | Last month | Last 2 weeks | Last week

Category: Games (5 600)

#	App	Developer Category	DAU	MAU Growth
1.	CityVille	Zynga Games » Other	20 183 455	89 724 427 -3.76%
2.	FarmVille	Zynga Games » Virtual World	13 342 728	48 147 282 -3.72%
3.	Texas HoldEm Poker	Zynga Games » Card	7 096 165	36 416 395 -2.36%
4.	Mafia Wars	Zynga Games » Action & Arcade	2 157 035	17 256 788 +36.58%
5.	FrontierVille	Zynga Games » Virtual World	4 740 354	16 577 664 -12.34%
6.	Café World	Zynga Games » Virtual World	3 017 524	13 449 954 -7.40%
7.	Bejeweled Blitz	PopCap Games Games » Action & Arcade	3 636 670	11 631 797 +5.93%
8.	Ravenwood Fair	Lolapps Inc. Games » Virtual World	1 257 400	11 534 052 +1.72%
9.	Pet Society	Electronic Arts Games » Virtual World	1 422 822	9 673 450 -11.32%
10.	Birthday Cards	RockYou! Games » Virtual World	528 240	9 410 777 +26.54%

http://www.socialbakers.com/facebook-applications/category/8-games

"High Performance at Massive Scale – Lessons learned at Facebook"

Jeff Rothschild - Vice President of Technology Facebook
Calit2 Auditorium - University of California, San Diego

http://cns.ucsd.edu/lecturearchive09.shtml#Roth

Web 2.0 Summit 09:, “High Order Bit: The Infrastructure of Facebook”

Mike Schroepfer - Vice President of Technology Facebook

http://www.youtube.com/watch?v=iddTbLo5s1M
http://channy.creation.net/blog/759

facebook

검색

Facebook Engineering님의 노트

Facebook Engineering님에 관한 노트

노트 검색

친구의 노트

페이지의 노트

내 노트

작성 중인 노트

나에 대한 노트

친구 프로필 및 페이지 바로 가기

구독하기

Facebook Engineering님의 노트

노트 ▸ Facebook Engineering님의 노트

+ 노트 작성

One Mobile Site to Serve Thousands of Phones

작성: Lee Byron · 2011년 4월 1일 금요일

Building for the mobile Web is a big challenge. You have to plan for thousands of different devices with varying capabilities, screen sizes, keyboards, CSS and JavaScript support, underlying technologies, and browser bugs.

Today we're excited to start rolling out a major upgrade to m.facebook.com that delivers the best possible mobile Web experienc...

전체 노트 보기 · 좋아요 · 댓글 달기

541명이 좋아합니다.

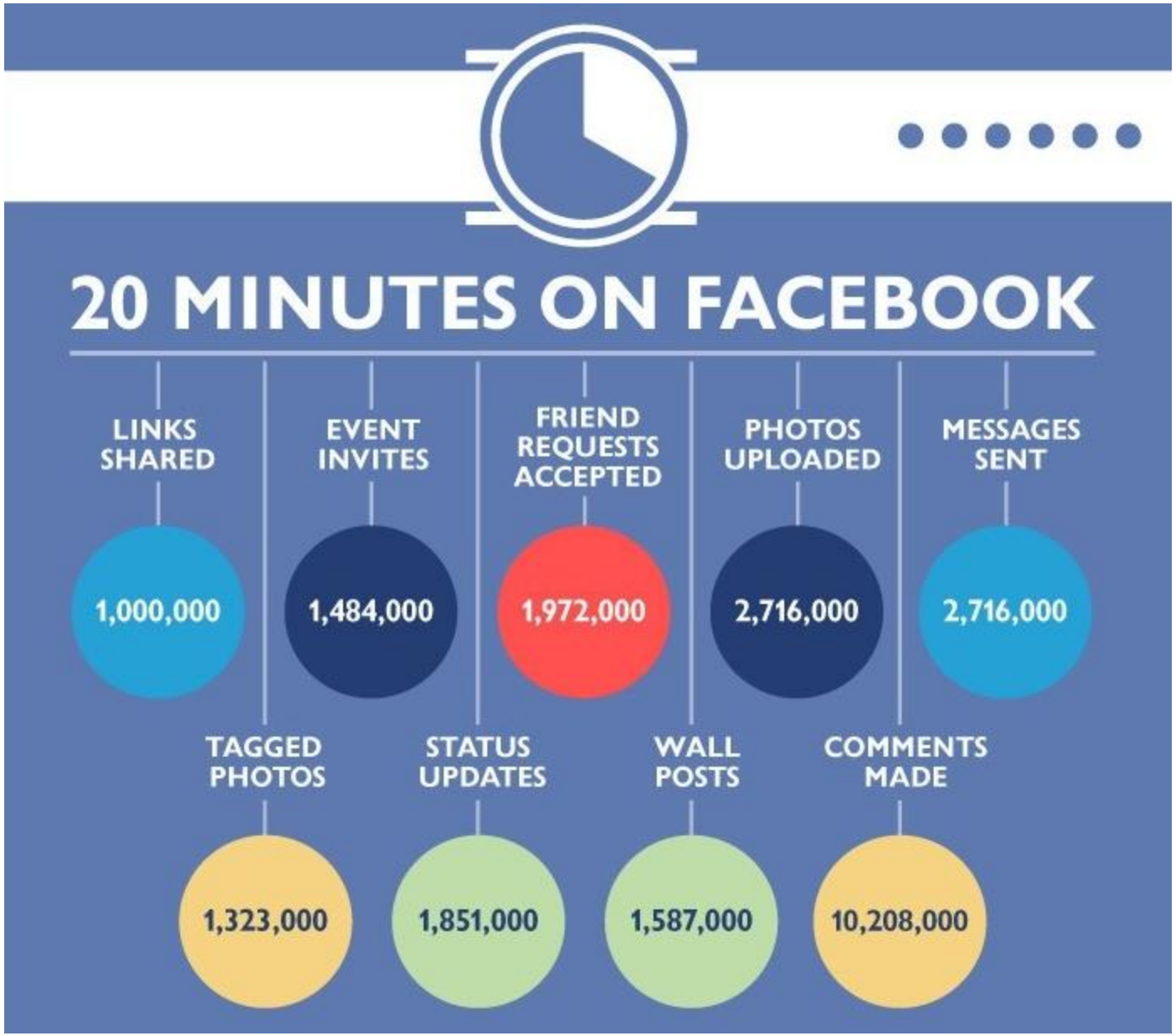
87개의 댓글 모두 보기

댓글을 입력하세요...

HipHop for PHP: More Optimizations for Efficient Servers

작성: Xin Qi · 2011년 3월 31일 목요일

Facebook switched all its production servers to HipHop in early 2010, also releasing the project's source code at that time. At the time of the switch, HipHop reduced our average CPU usage by 50%, the six months after its release saw an additional 1.8x performance improvement, and in the past six months the team in conjunction with the open source ...



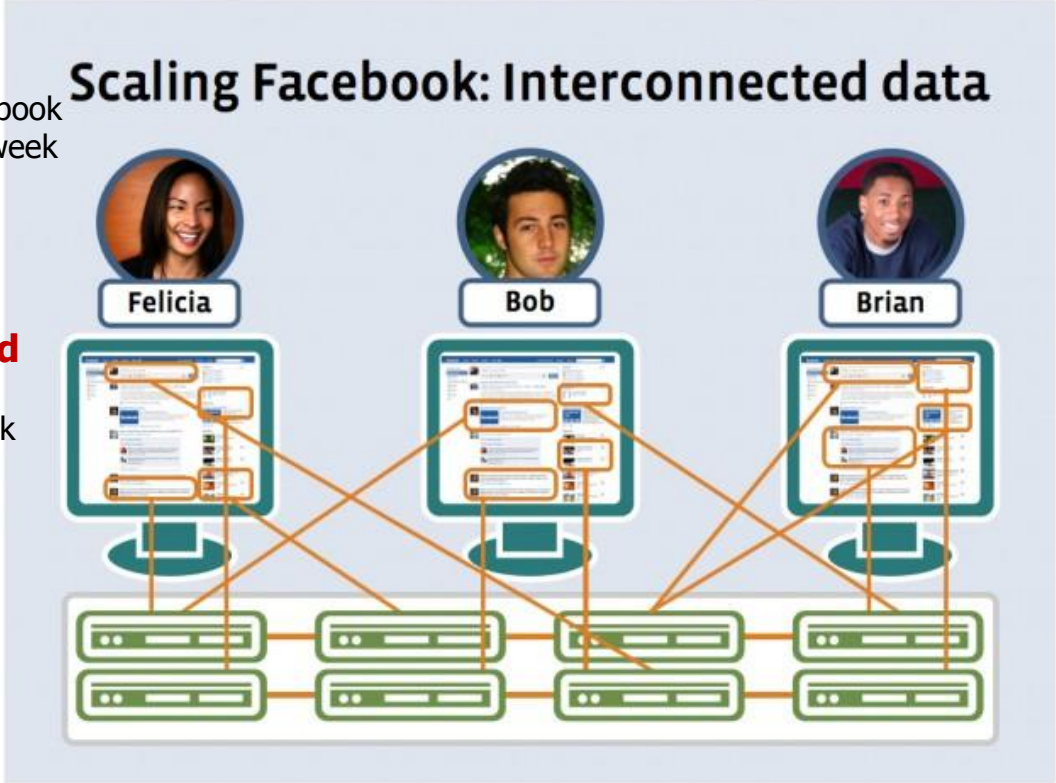
출처 : http://blog.daum.net/amorfati2005/53

• 500 million active users
• 100 billion hits per day
• 50 billion photos
• 2 trillion objects cached, with hundreds of millions of requests per second
• 130TB of logs every day

- 전통적인 웹사이트에서는
  장비를 추가하고 DB를 나누어서 해결
- 하지만 Facebook은 서로 연결된
  데이터 구조이기에
  전통적 방식으로는 불가능하다

- Users spend 8 billion minutes online everyday using Facebook
- There are some 2 billion pieces of content shared every week on the service
- Users upload 2 billion photos each month
- There are over 20 billion photos now on Facebook
- **During peak times,**
  Facebook serves 1.2 million photos a second
- Yesterday alone, Facebook served 5 billion API calls
- There are 1.2 million users for every engineer at Facebook

2009년 기준 !!

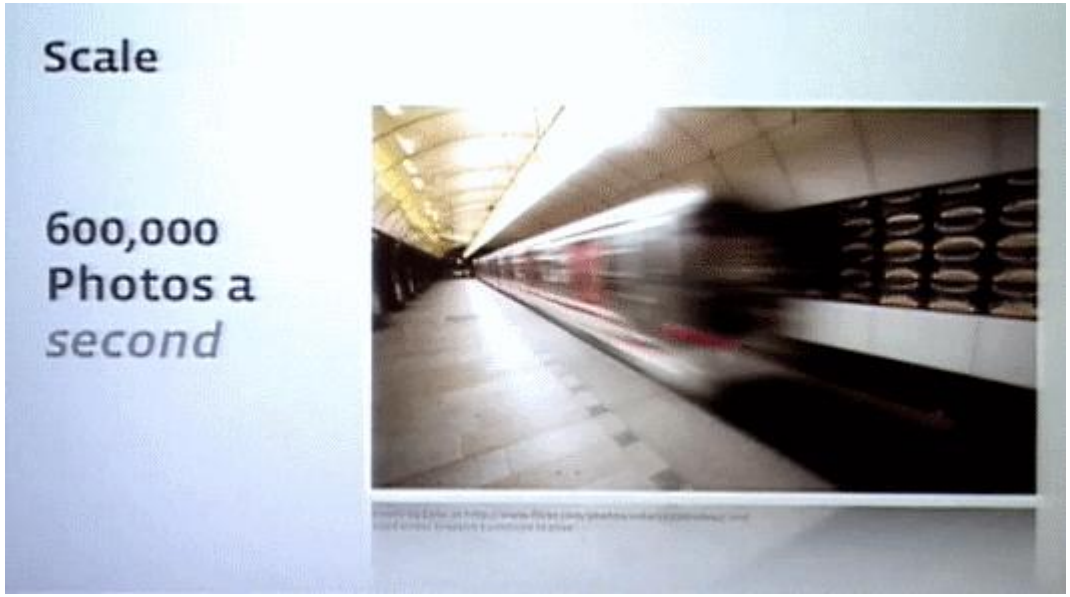


File Storage at facebook

File Storage at facebook

haystack: efficient storage of billions of photos

Facebook에는 약 20빌리언(200억)장의 사진이 보존되고 있으며 각각 4 종류의 해상도로 보존되고 있기 때문에 전체 800억장의 사진이 보존되고 있다고 말할 수 있다. 이것은 사진을 전부 이으면 지구를 10회 둘러 쌀 만한 면적이 있다.

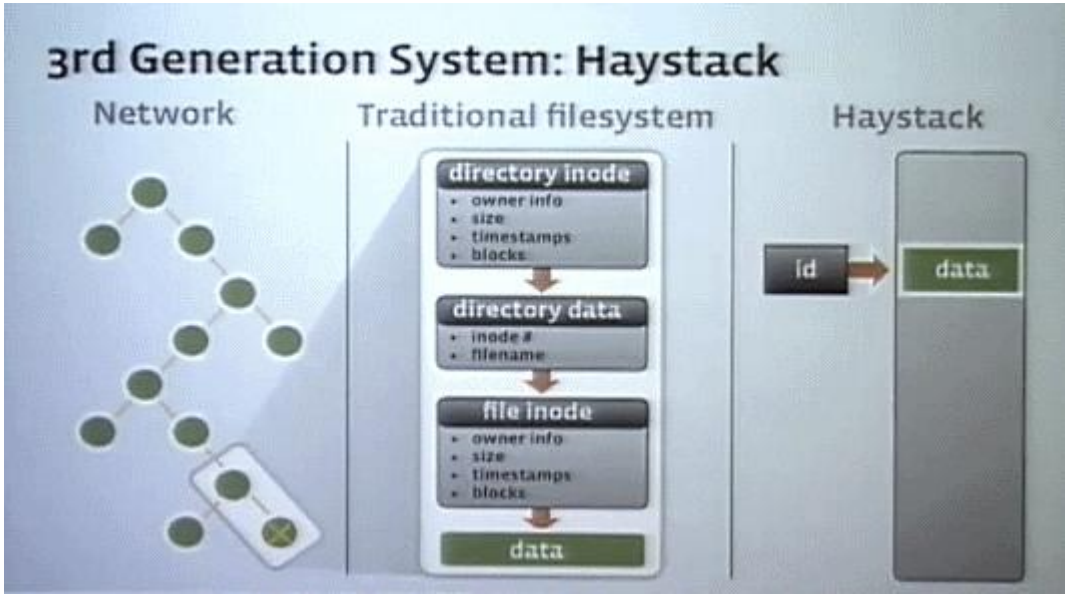


File Storage at facebook

Traditional Filesystem (Posix Standard) Metadata

=> **It's Too Many & Big. We couldn't cache it at all !!**

- File length
- Device ID
- Storage block pointers
- File owner
- Group owner
- Access rights on each assignment: read, write execute
- Change time
- Modification time
- Last access time
- Reference counts



Scaling: First you do it the easy way

- Upload tier: handles uploads, scales the images, stores on NFS tier Serving tier: Images are served from NFS via HTTP
- NFS Storage tier built from commercial products
- Filesystems aren't really good at supporting large numbers of files
- Metadata is too large to fit in memory, and thus many disk operations required for each file read
- Limited by I/O not storage density

Then you optimize

Cachr: Cache the high volume smaller images to offload the main storage systems

- Only 300M images in 3 resolutions
- Distribute these through a CDN to reduce network latency
- Cache them in memory at origin for scalability, redundancy, and performance

NFS file handle cache: Eliminates some of the NFS storage tier metadata overhead

We can do better

File Storage at facebook

Haystack : a generic HTTP-based object store containing needles that map to stored opaque objects

- **HTTP server** : simple evhttp server provided with the open source libevent library

- **Photo Store Server** : responsible for accepting HTTP requests and translating them to the corresponding Haystack store operations (Load all meta data on Memory)

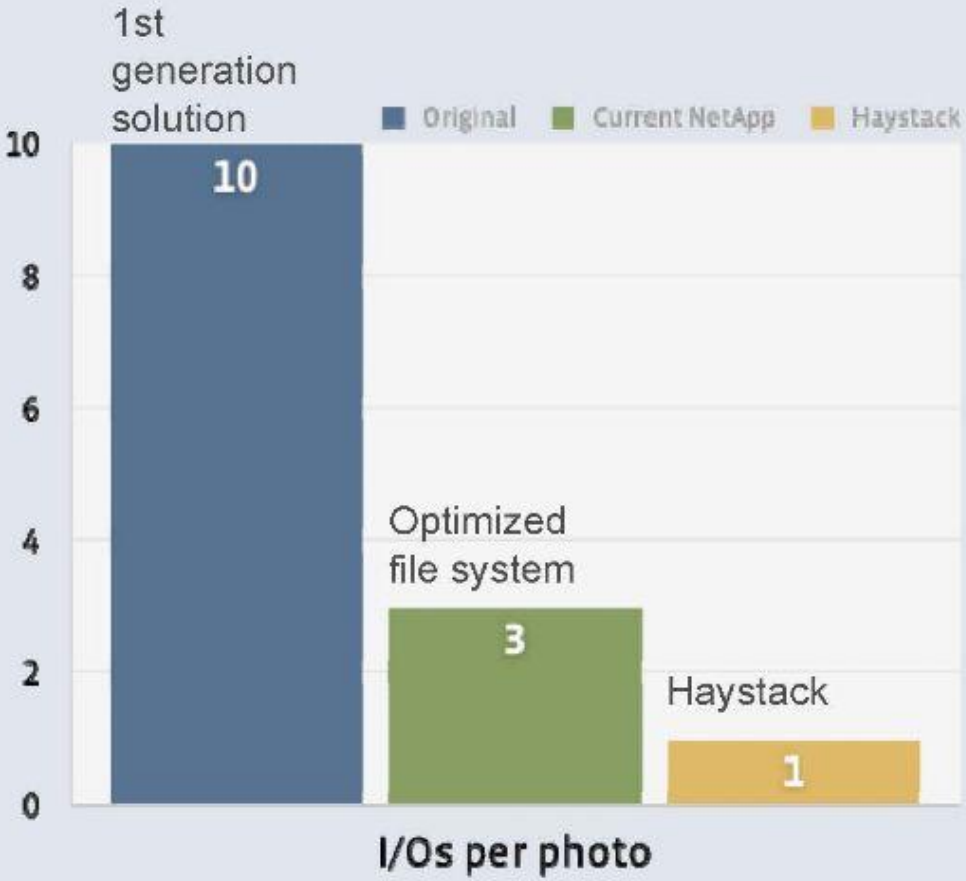
- **Haystack Object Store** : Index(meta data) + Data

- **Filesystem(XFS)** : Haystack object stores are implemented on top of files stored in a single filesystem created on top of the 10TB volume.

- **Storage(Blade server)** : 2 x quad-core CPUs, 16GB – 32GB memory, hardware raid controller with 256MB – 512MB of NVRAM cache, 12+ 1TB SATA drives

The efficiency of Haystack

Compared with other systems



But It’s not opensource yet !!

Pomegranate - Storing Billions And Billions Of Tiny Little Files
Hadoop Archive: File Compaction for HDFS

Web Tier at facebook

Presentation Layer: PHP

- Simple to learn: small set of expressions and statements
- Simple to write: loose typing and universal "array"
- Simple to read: similar syntax to C++ and Java
- Simple to debug: no need to re-compile
- One of the shortest Hello Worlds


```
<?PHP print "Hello World!" ?>
```

Web Tier at facebook

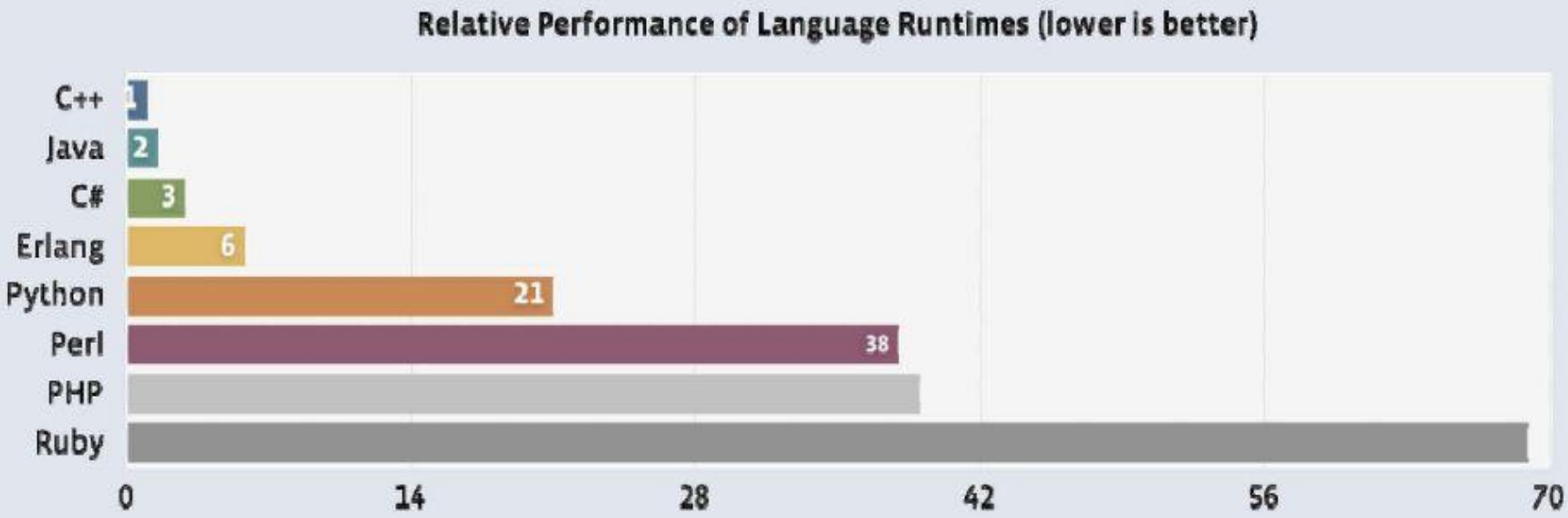
But this comes at a cost

High CPU and memory consumption

Interoperability with C++ Challenging

Language doesn't encourage good programming in the large

Initialization cost of each page scales with size of code base



Optimizing PHP

Op-code optimization

APC improvements

- Lazy loading
- Cache priming

Custom extensions

- Memcache client extension
- Serialization format
- Logging, Stats collection, Monitoring
- Asynchronous event-handling mechanism

Web Tier at facebook

- HipHop is a source code transformer which transforms PHP source code into highly optimized C++ and then compiles it using g++.

- It will be useful to companies running very large PHP infrastructures who do not wish to rewrite complex logic within C or C++

- 50% reduction in CPU usage than Apache + PHP

- Facebook's API tier can serve twice the traffic using 30% less CPU

- It has embedded simple webserver on top of libevent.

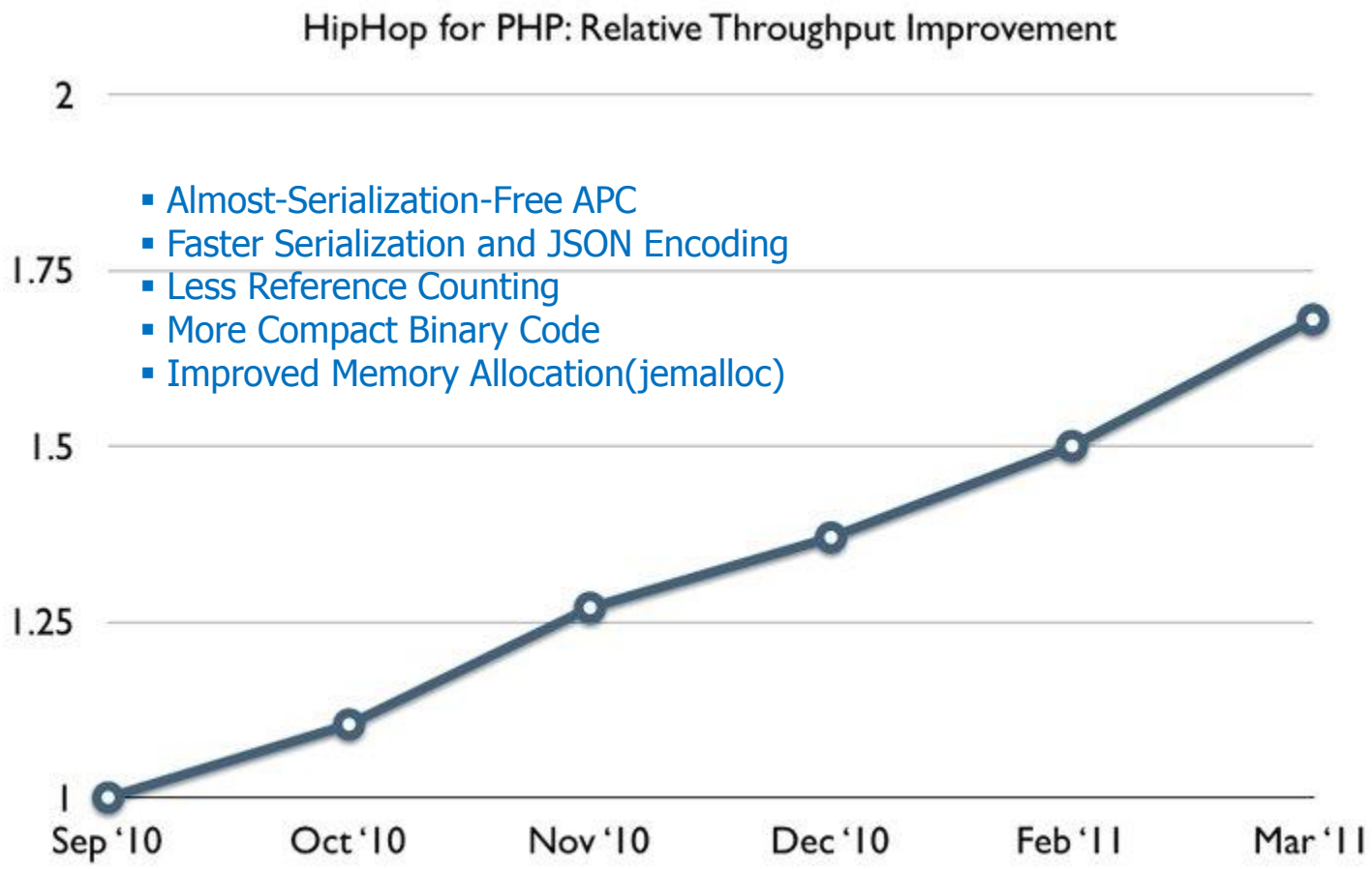
  90% of Apaches disappear in history.

HipHop for PHP

Reference

- Drupal
- MediaWiki
- phpBB
- WordPress (WordPress has become 2.7x faster)

Web Tier at facebook



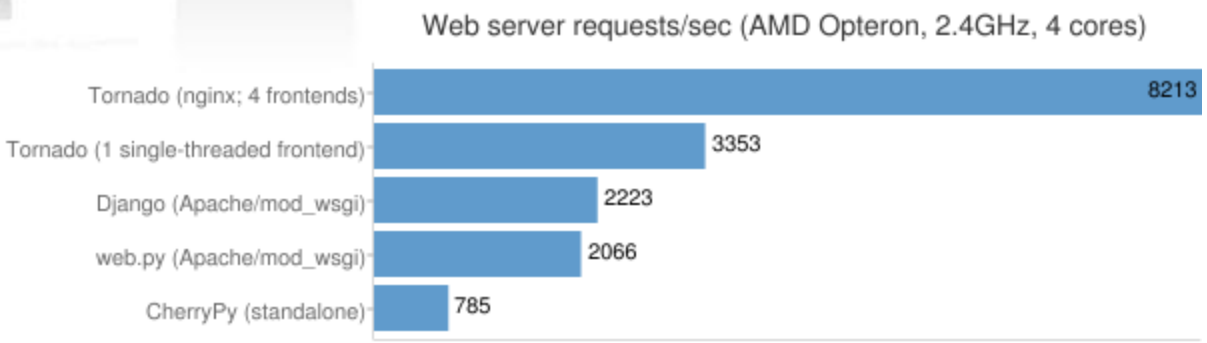
- Almost-Serialization-Free APC
- Faster Serialization and JSON Encoding
- Less Reference Counting
- More Compact Binary Code
- Improved Memory Allocation(jemalloc)

Throughput : the average rate of Web requests successfully served over a given period of time

Web Tier in facebook

Tornado is a relatively simple, **non-blocking web server** framework written in Python.

It is designed to handle thousands of simultaneous connections, making it ideal for real-time Web services.



Web Tier at facebook

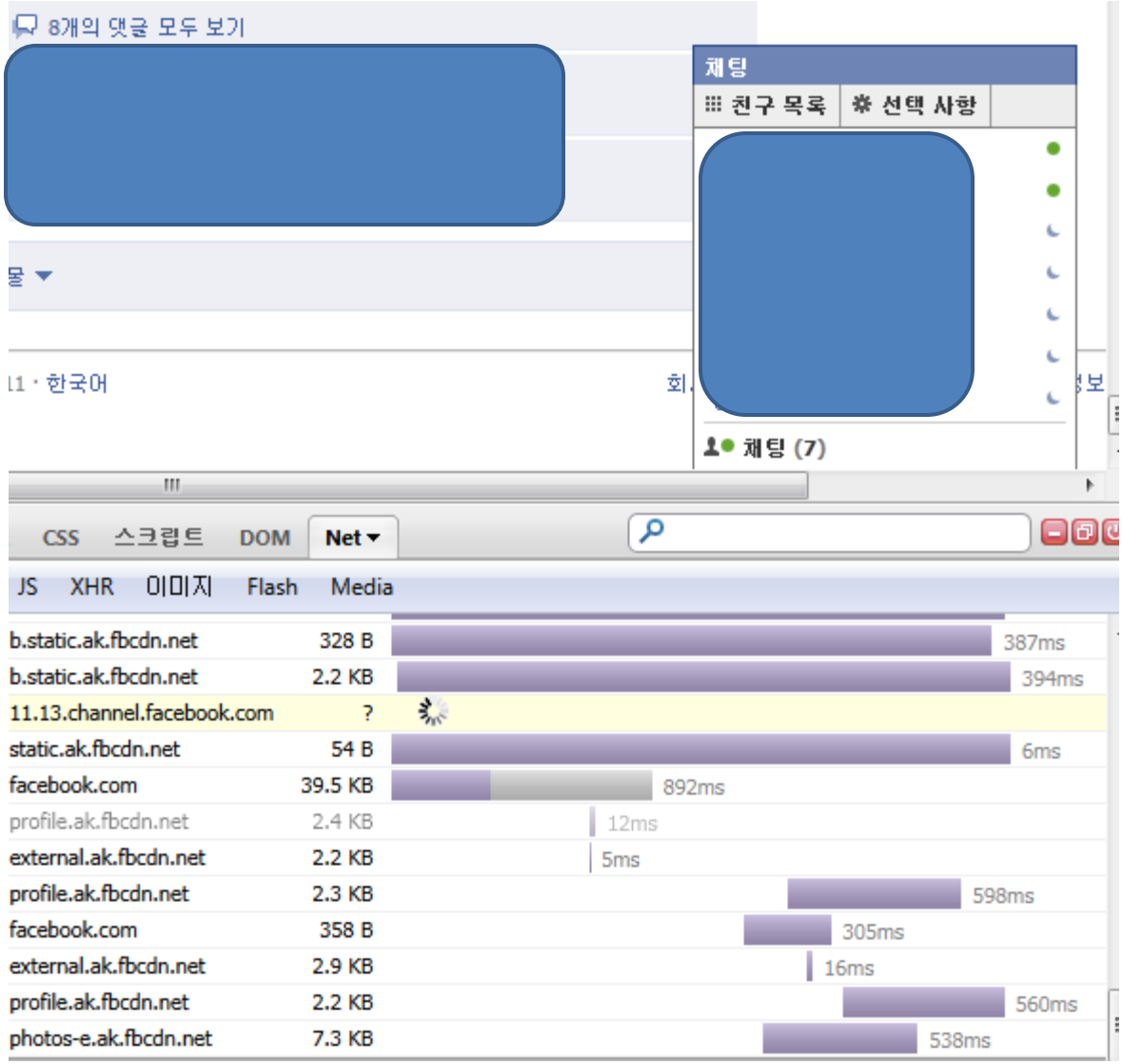
Node's goal is to provide an easy way to build scalable network programs.

Node is similar in design to and influenced by systems like Ruby's Event Machine or Python's Twisted. Node takes the event model a bit further—it presents the event loop as a **language construct** instead of as a library

Chat , Message at facebook

Chat Service in facebook

Real-time presence notification is biggest challenge. Not sending messages



Chat Service at facebook

Real-time presence notification is biggest challenge. Not sending messages

Each Facebook Chat user now needs to be notified whenever one of his/her friends
(a) takes an action such as sending a chat message or loads a Facebook page (if tracking idleness via a last-active timestamp) or
(b) transitions between idleness states (if representing idleness as a state machine with states like "idle-for-1-minute", "idle-for-2-minutes", "idle-for-5-minutes", "idle-for-10-minutes", etc.).
Note that approach (a) changes the sending a chat message / loading a Facebook page from a one-to-one communication into a **multicast** to all online friends, while approach (b) ensures that users who are neither chatting nor browsing Facebook are nonetheless generating server load.

http://www.facebook.com/note.php?note_id=14218138919
http://www.facebook.com/note.php?note_id=51412338919

Chat Service at facebook

Facebook Chatting Sub-Systems

For Facebook Chat, we rolled our own subsystem for **logging chat messages (in C++)** as well as **an epoll-driven web server (in Erlang)** that holds online users' conversations in-memory and serves the long-polled HTTP requests. Both subsystems are clustered and partitioned for reliability and efficient failover. **Why Erlang**? In short, because the problem domain fits Erlang like a glove. **Erlang is a functional concurrency-oriented language with extremely low-weight user-space "processes", share-nothing message-passing semantics, built-in distribution, and a "crash and recover" philosophy proven by two decades of deployment on large soft-realtime production systems.**

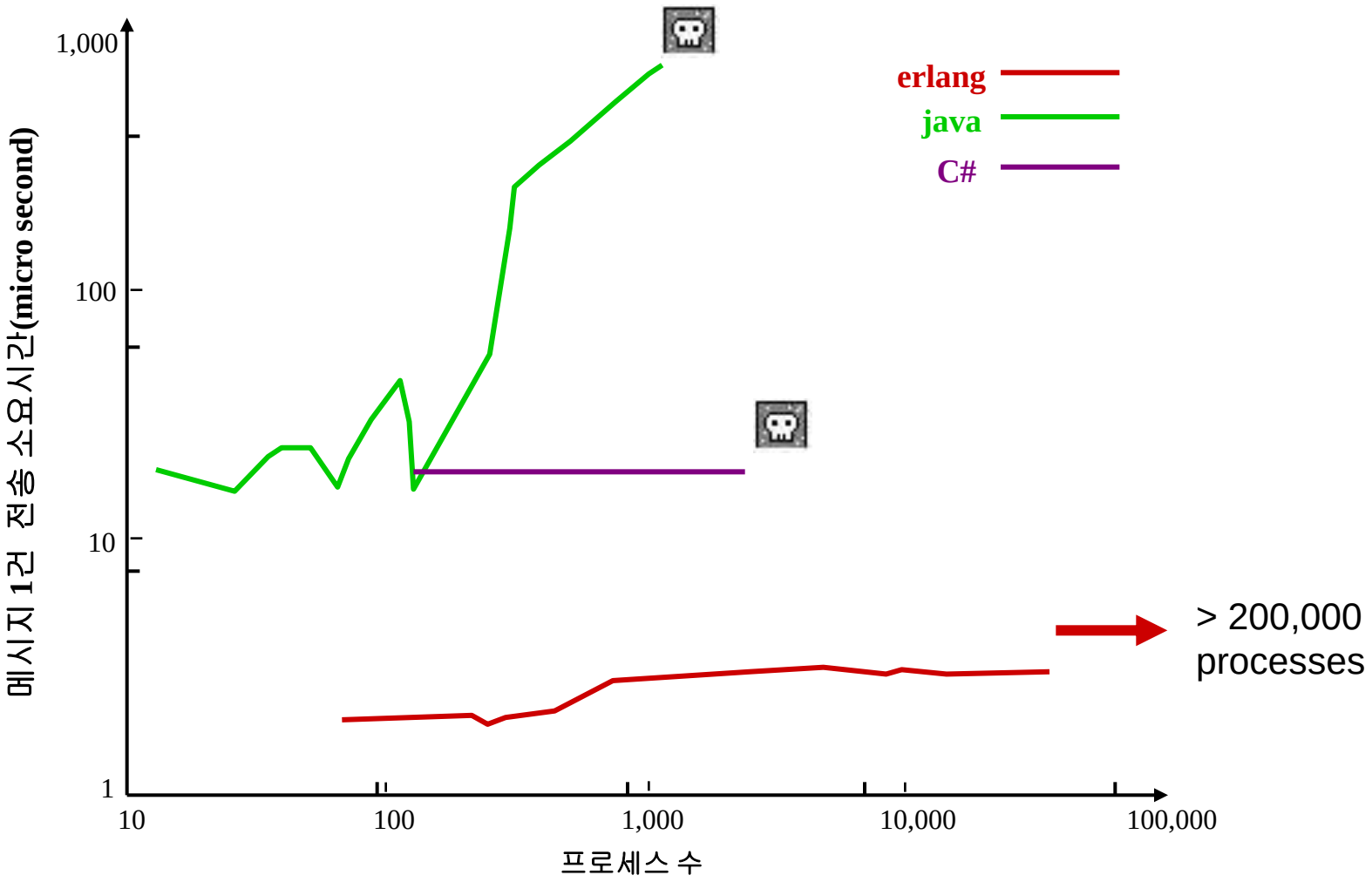
http://www.facebook.com/note.php?note_id=14218138919&id=9445547199&index=2

Erlang? Why?

- 멀티코어에 적합한 프로그래밍 모델
  - CPU가 늘어날수록 속도도 빨라진다.
- 함수형 언어
- Light weight process(경량 프로세스)
- 변하지 않는 변수
- 속도와 과부하 문제에 대한 탁월한 해결력
- 무정지(Fault-tolerant) 애플리케이션

출처 : 2008-2009 Neowiz Devlelopers' Day

Chat Service at facebook



Source:
Joe Armstrong
SICS

출처 : 2008-2009 Neowiz Devlelopers' Day

Chat Service at facebook

Another Challenge, Communicate with other parts of system

Thrift is a software framework for scalable cross-language services development. It combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml.

	Lang Features	Protocol Support				Transports		Servers			Clients		OS Support		
Language	Unions	Binary	Dense	Compact	JSON	Framed	SSL	Basic	Non- blocking	HTTP	Basic	HTTP	Win	OSX	Linux
Action Script 3 (as3)															
C Glib (c_glib)		0.6				0.6		0.6			0.6				0.6
C++(cpp)							0.7			0.4			THRIFT-757		
C# (csharp)					0.5		THRIFT-181			THRIFT-322					
Erlang (erl)															
Haskell (hs)															
Java (java)		0.2		0.2	0.2	0.2	0.5	0.2	0.2	0.4	0.2				
JavaScript (js)					0.3		0.3					0.3			
Node.js (js:node)		0.6				0.6			0.6		0.6				
Objective C (cocoa)															
OCaml (ocaml)															
Perl (perl)															
PHP (php)															
Python (py)							0.7								
Ruby (rb)															
Smalltalk(st)															

Chat Service Trio's Challenge
9 months after launching Service

1. Erlang Channel Server
    1. String problem of Erlang makes Memory footprint big.
    2. Garbage collecting when wait for new message.
    3. We could monitor channel server at runtime.
        1. debug & load new code on the fly => Erlang's feature

2. C++ Logger
    1. Memory Fragmentation

3. Presence Server
    1. Update user info from channel server data
        1. It's far simpler than the Thrift example service bundled.
    2. Collecting user presence to Presence server
    3. Use zlib for compressing data to solve network bandwidth problem

4. Facebook Chat Now Available Everywhere
    1. They start supporting Jabber(XMPP)

Social Messaging Service in facebook

새로운 메시지 기능을 소개합니다

하나의 대화 속에서 SMS, 채팅, 이메일을 함께 즐기세요.

모든 메시지를 한 곳에

한 곳에서 Facebook 메시지, 채팅 대화 내역과 SMS 를 한꺼번에 확인하세요.

- 선택 사항인 Facebook 이메일 주소를 활성화하면 이메일도 포함할 수 있습니다
- 개인 정보 설정을 통해 메시지를 보낼 수 있는 사용자 범위를 직접 지정할 수 있습니다

전체 대화 히스토리

각 친구들과 나누었던 모든 대화를 하나의 대화에서 확인할 수 있습니다.

- 제목이나 다른 형식을 따를 필요가 없습니다
- 따분하고 재미없는 대화에서 쉽게 벗어나세요

내가 원하는 메시지

친구한테 온 메시지를 먼저 확인하세요.

- 모르는 사람이 보낸 메시지나 다량 발송 메일은 기타 폴더에 저장됩니다
- 스팸은 자동으로 가려집니다

더 자세한 정보를 원하시면 다음 링크를 참조하세요. 새로운 메시지에 대한 FAQ

초대장 요청

over 350 million users sending over 15 billion messages per month

http://blog.facebook.com/blog.php?post=452288242130
http://blog.hankyung.com/kim215/blog/3653251

The Underlying Technology of Messages

Eventual Consistency

Performance

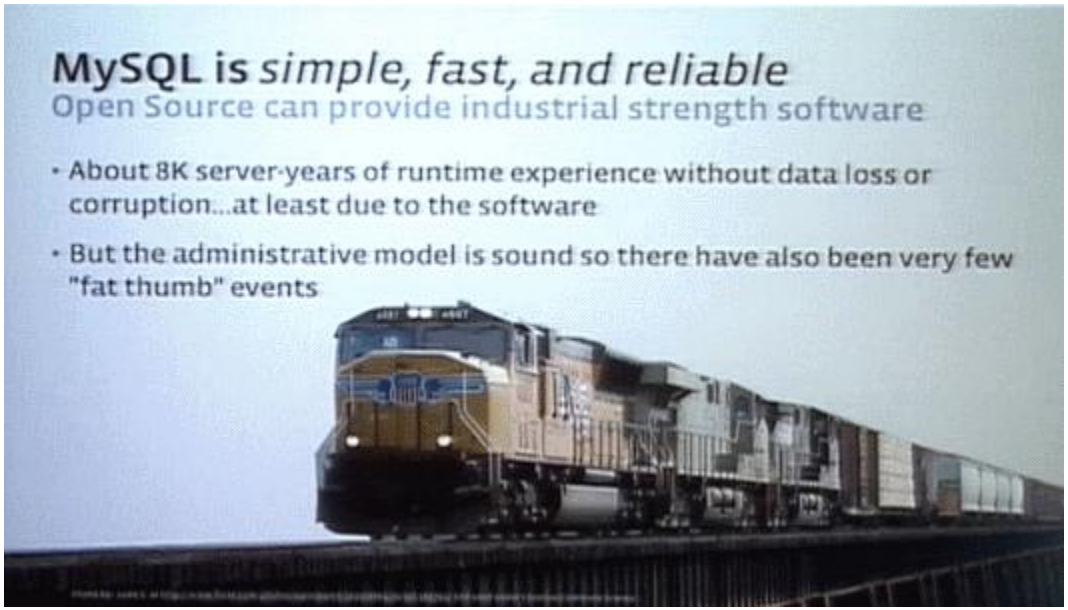
Haystack

http://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919
http://zookeeper.apache.org/
http://hbase.apache.org/

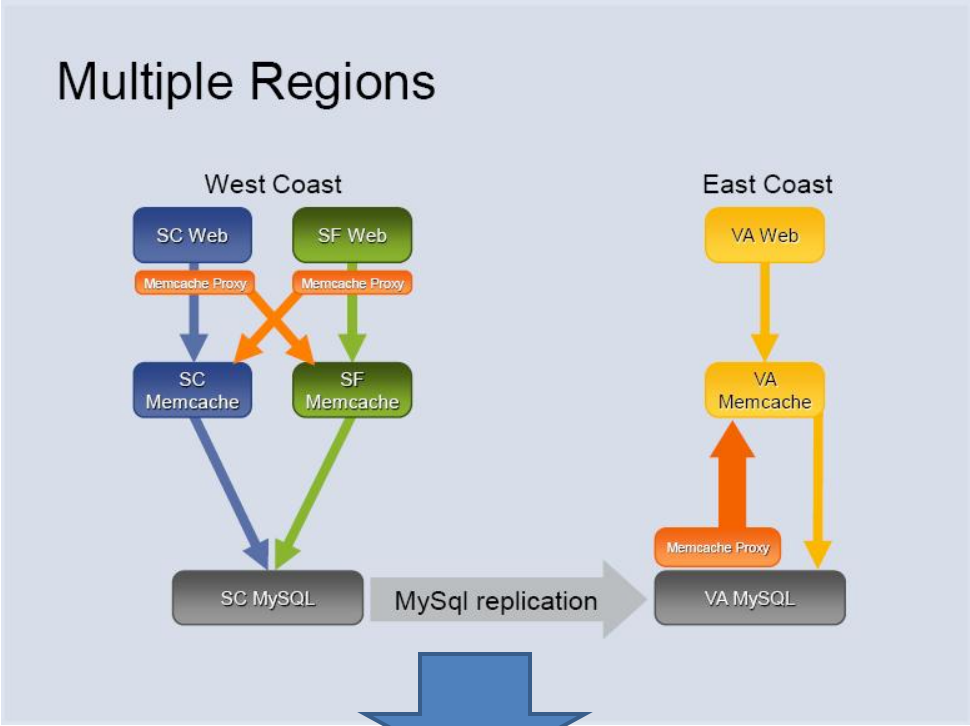
Data Store at facebook

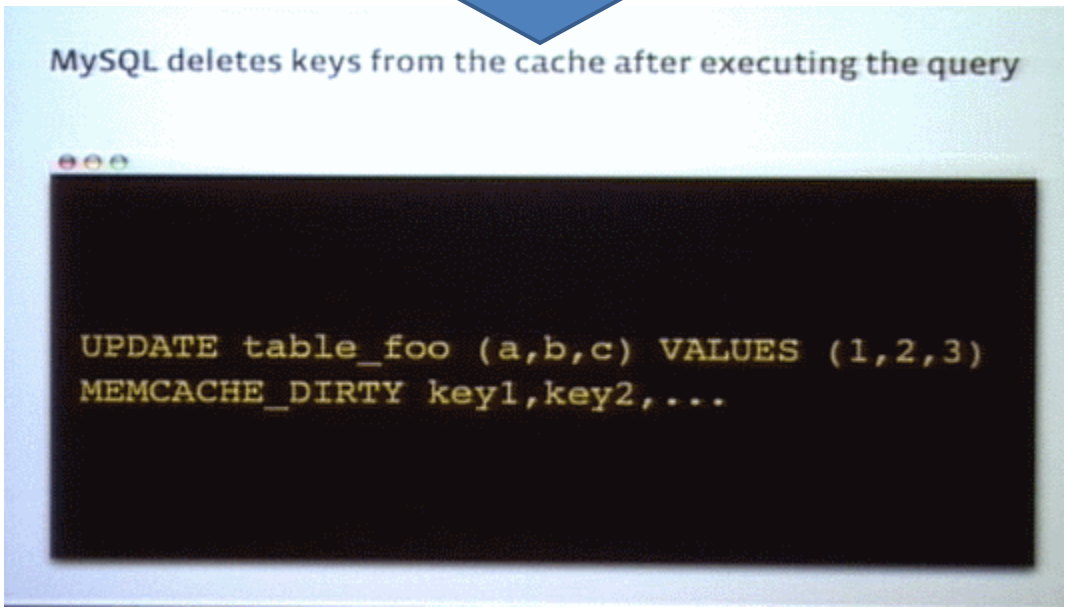
We use MySQL as a Simple Data Storage(Key-value).
It's a good RDB, but We do not use that as RDB.

- Logical Migration은 어렵다.
- 많은수의 Logical DB를 생성하고, 다양한 물리노드로 load balancing을 한다
- Scale at Web-Tier
    - Business 로직 처리는 Front-end 에서
    - No-Join, Compute on Web-tier
- Non-Static data 는 DB에 두지마라
    - 많이 참조되는 Static Data는 DB에 두지마라 (Cache하라)
- Service나 Memcache를 이용해서 Global Query를 하라(?)



Memcached & MySQL at facebook



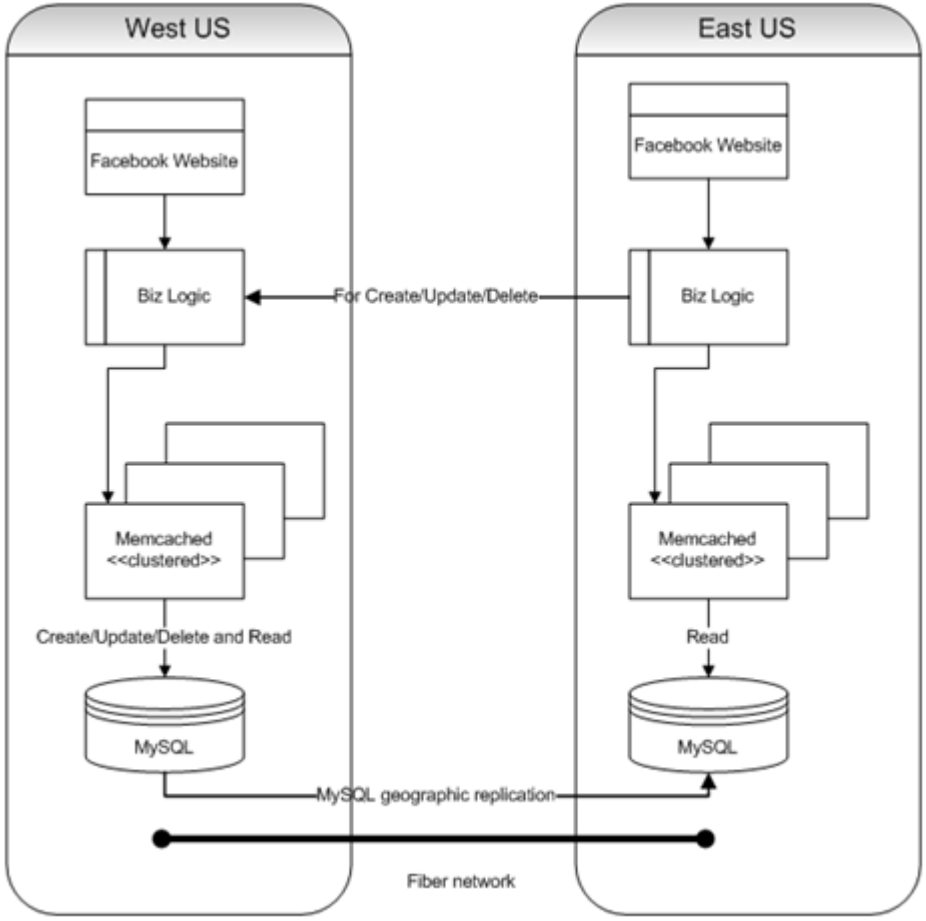


Memcached & MySQL at facebook

Global Load Balancing(IP기반)

미국 서부와 아시아권

유럽



http://bcho.tistory.com/416

Memcached at facebook

1. Per-connection memory buffer
    1. Facebook implemented per-thread shared connection buffer pool for TCP/UDP
2. Choose UDP for reduce network traffic / Multi-get
    1. UDP Socket Lock (Linux kernel use single socket lock for multiple thread)
        1. separate UDP sockets for transmitting replies
3. On Linux, network interrupt is delivered to one cores => all cores receive soft interrupt. And some NIC card has high rate of interrupts
    1. Solve that combination of interrupt driven and polling driven network I/O
    2. Now every core do their work!!
4. Memcached's stat use global lock
    1. moving stats collection per-thread and aggregating results on-demand
5. contention on the lock that protects each network device's transmit queue
    1. changed the dequeue algorithm
    2. scale memcached to 8 threads on an 8-core system

http://www.facebook.com/note.php?note_id=39391378919&id=9445547199&index=0
https://github.com/facebook/memcached

Memcached at facebook

1. 800대의 Memcached 운용중 (2008년 12월)

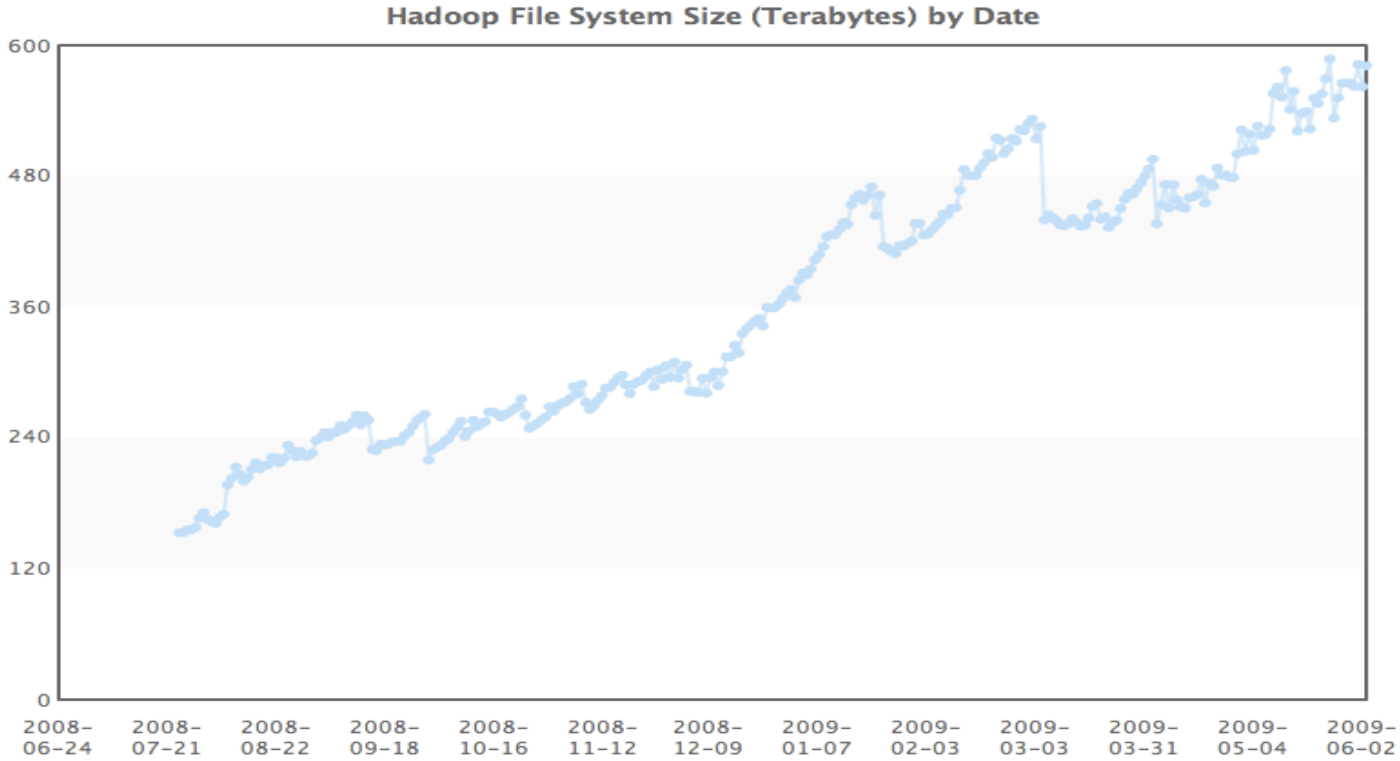
2. 28 terabytes of memory (2008년 12월)
    1. 초당 12억건의 Request 처리중

3. Original memcached can serve 50,000 UDP requests/s
    1. 200,000 UDP requests/s with 173 microseconds.
    2. 300,000 UDP requests/s with high latency.

http://www.facebook.com/note.php?note_id=39391378919&id=9445547199&index=0
https://github.com/facebook/memcached

Data Warehousing at facebook

Data: How much?



- 200GB per day in March 2008
- 2+TB(compressed) raw data per day in April 2009
- 4+TB(compressed) raw data per day today (2009, 11)

Scribe :

A **Thrift service for distributed logfile collection**. Scribe was designed to run as a daemon process on every node in your data center and to forward log files from any process running on that machine back to a central pool of aggregators. Because of its ubiquity, a major design point was to make Scribe consume as little CPU as possible.

Scribe is a server for aggregating streaming log data. It is designed to scale to a very large number of nodes and be robust to network and node failures. There is a scribe server running on every node in the system, configured to aggregate messages and send them to a central scribe server (or servers) in larger groups. **If the central scribe server isn't available the local scribe server writes the messages to a file on local disk and sends them when the central server recovers**. The central scribe server(s) can write the messages to the files that are their final destination, typically on an nfs filer or a distributed filesystem, or send them to another layer of scribe servers.

What is HIVE?

- A system for managing and querying structured data built on top of Hadoop
  - Map-Reduce for execution
  - HDFS for storage
  - Metadata in an RDBMS
- Key Building Principles:
  - SQL as a familiar data warehousing tool
  - Extensibility - Types, Functions, Formats, Scripts
  - Scalability and Performance
  - Interoperability

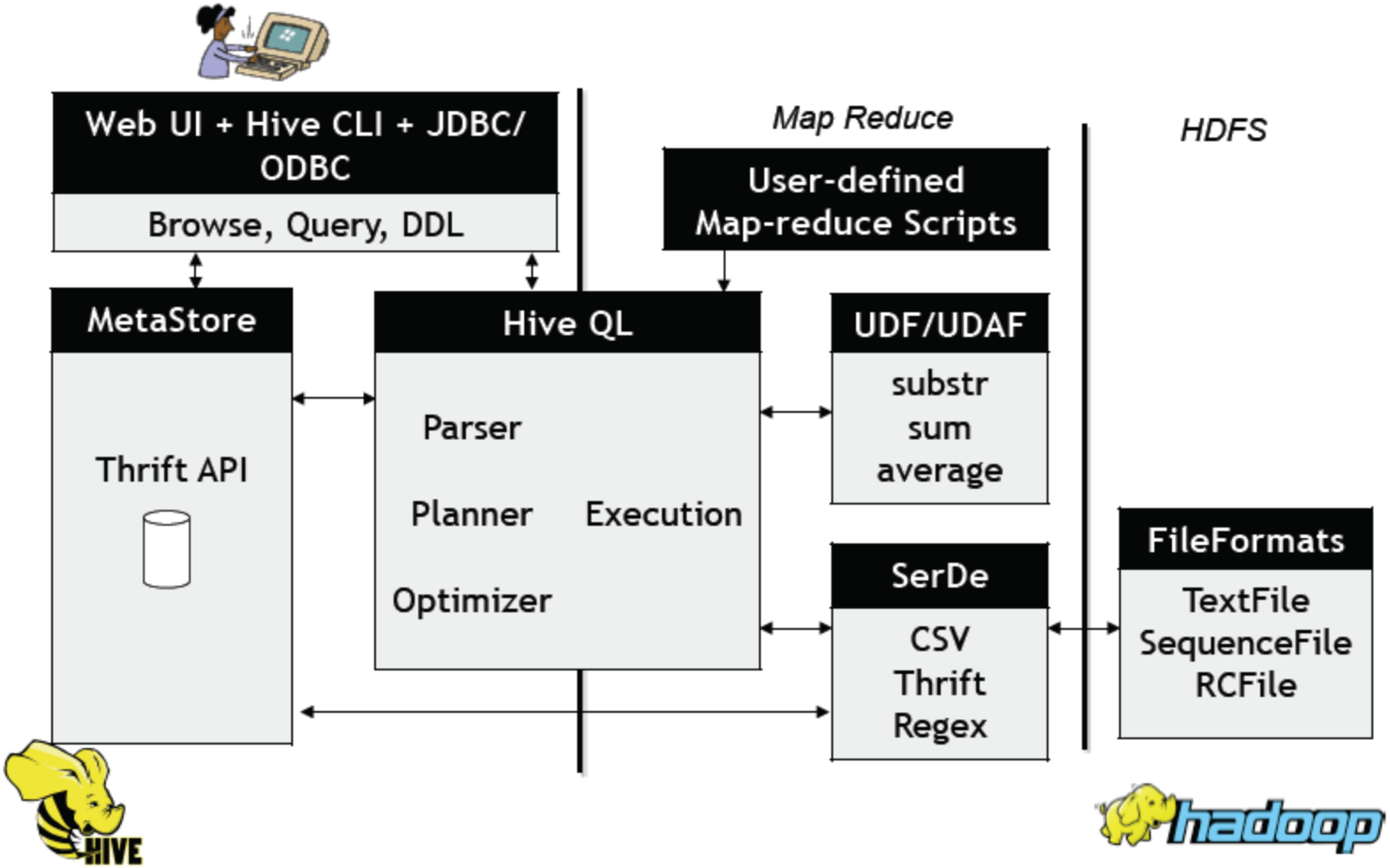
Why SQL on Hadoop?

```
hive> select key, count(1) from kv1 where key > 100 group by
   key;
```

vs.

```
$ cat > /tmp/reducer.sh
uniq -c | awk '{print $2"\t"$1}'
$ cat > /tmp/map.sh
awk -F '\001' '{if($1 > 100) print $1}'
$ bin/hadoop jar contrib/hadoop-0.19.2-dev-streaming.jar -input /user/hive/warehouse/kv1 -
    mapper map.sh -file /tmp/reducer.sh -file /tmp/map.sh -reducer reducer.sh -output /tmp/
    largekey -numReduceTasks 1
$ bin/hadoop dfs –cat /tmp/largekey/part*
```

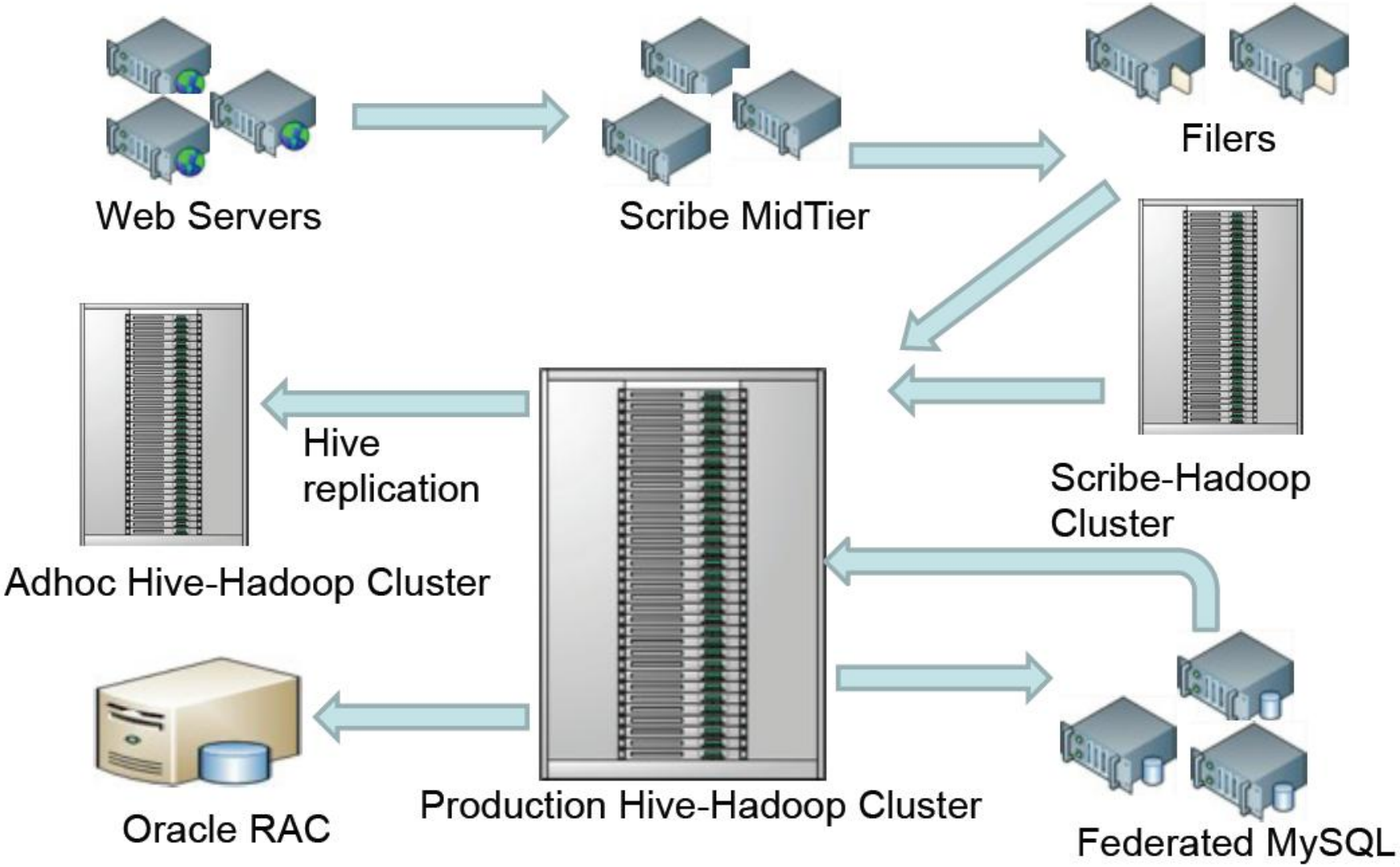
Hive Architecture



Data Usage

- **Statistics per day:**
  - 4 TB of compressed new data added per day
  - 135TB of compressed data scanned per day
  - 7500+ Hive jobs on production cluster per day
  - 80K compute hours per day
- **Barrier to entry is significantly reduced:**
  - New engineers go though a Hive training session
  - ~200 people/month run jobs on Hadoop/Hive
  - Analysts (non-engineers) use Hadoop through Hive

Data Flow Architecture at Facebook



Culture of facebook

Culture of facebook

Importance of culture

Hackathon

http://mimul.com/pebble/default/2010/05/23/1274600280000.html

Culture of facebook

1. 개발 과정에서 **Product Manager**의 권한은 미약하고, 오히려 개발자들이 자기 프로젝트에 관심을 가지도록 하기 위해 **개발자들을 설득, 로비**하는 일이 잦다.

2. 모든 엔지니어는 입사 후 **4주~6주 과정의 부트 캠프**를 거친다. 부트 캠프에서 페이스북 시스템의 버그를 직접 수정하면서 배우고, 시니어 엔지니어들의 강의를 듣는다. 약 10% 정도의 엔지니어들이 이 과정을 완료하지 못하며 이 경우 권고 사직 대상이 된다.

3. 엔지니어는 **백엔드부터 프론트엔드까지** 보통 혼자서 다 구현을 한다. 클라이언트, UI 일은 기술적 난이도가 낮고 반복이 많아 엔지니어들이 기피한다. 뉴스피드나 광고 알고리즘, memcache 최적화 등 백엔드 구현을 선호한다.

3. 모든 변경된 **코드는 의무적으로 내부 리뷰**를 거친다. **주커버그**는 뉴스피드 관련 코드는 직접 리뷰를 한다. 최소 1명 이상의 리뷰를 거치지 않은 코드를 릴리스해서는 안된다.

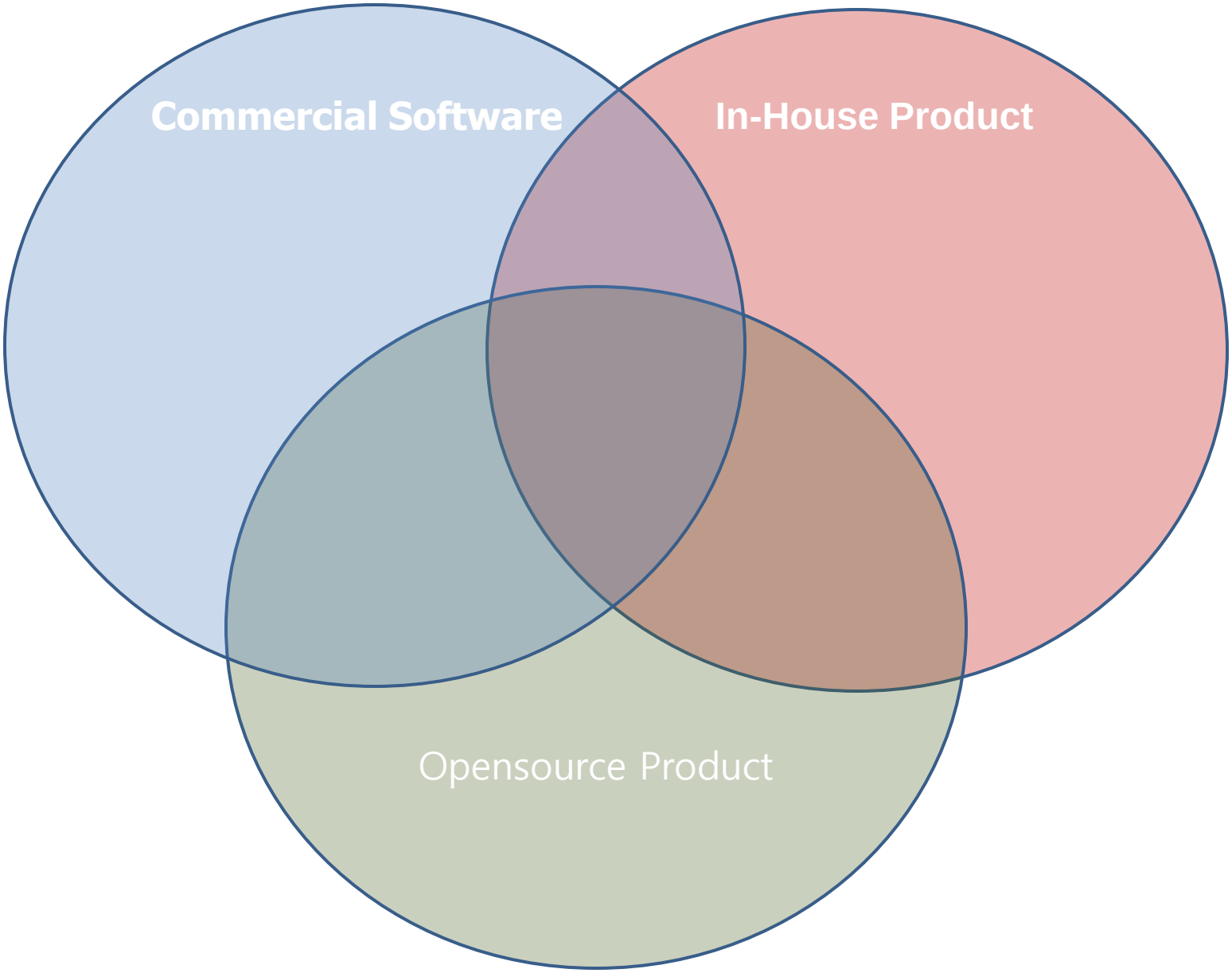
4. 페이스북에는 **전문 QA가 없다**. 모든 개발자가 테스트, 버그 수정, 유지보수까지 자기 코드에 대해 책임진다. 내부 개발자가 사용해보는 테스트는 있다. 서비스 런칭을 위해 거쳐야 하는 자동화된 테스트도 있다. 하지만 기본적으로는 개발자가 UI 구현부터 테스트까지 모두 책임을 지는 구조이다.
"대부분의 엔지니어들은 버그 없는 코드를 작성할 줄 안다"는 믿음도 일부 있는 분위기.

5. 매주 화요일 릴리스하며 그 주중에 릴리스 대상 코드를 커밋한 개발자는 릴리스 시작 시 on-site(특정 IRC 채널)에서 대기해야 한다. 그렇지 않으면 공개 창피당한다. 9레벨 릴리스 중 문제가 생기면 개발자가 수정 후 1레벨부터 다시 시작한다.

6. SVN을 통해 혼나거나 프로젝트를 자주 지연시킨 엔지니어는 해고된다.(매우 뛰어난 퍼포먼스만 허용. 보통 채용 6개월 이내에 해고) 버그나 사이트 다운, 실수 등의 문제로 혼난다고 해서 해고되지는 않는다. 다만 이런 공개 창피를 통해 이슈를 해결하기 위해 매우 열중하며 이 과정에서 모두가 함께 배운다.

http://logonjava.blogspot.com/2011/01/facebook-sw.html

Culture of facebook




고맙습니다