apc@facebook

Brian M. Shire

PHP Tek 2007, Chicago
May 17th, 10:15-11:15 am

About Facebook

A social utility that connects you with people around you

Networks based around a workplace, region, high school or college

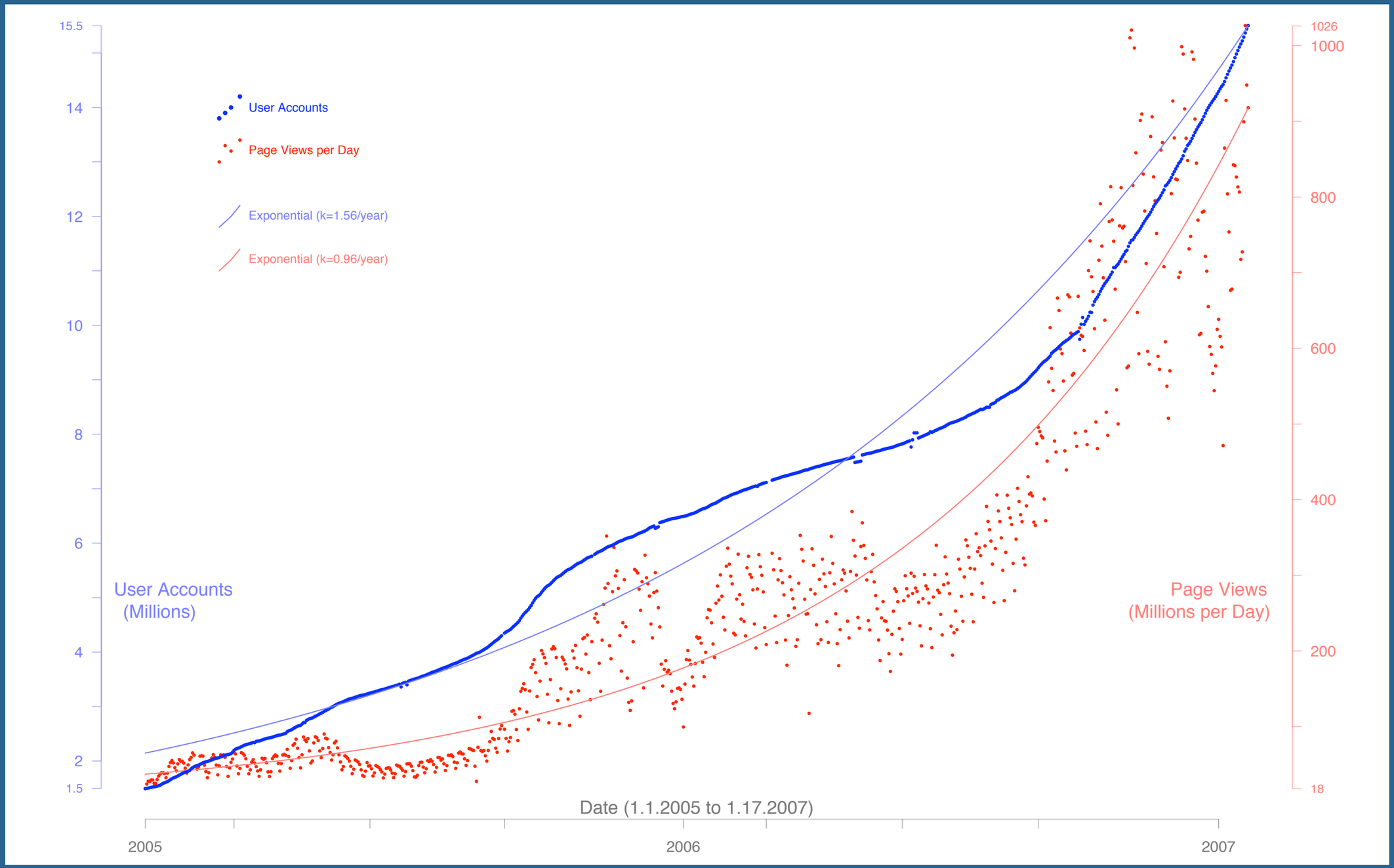
Typical uses:

- Look up people around you
- See what's going on with your friends
- Share information with people you know

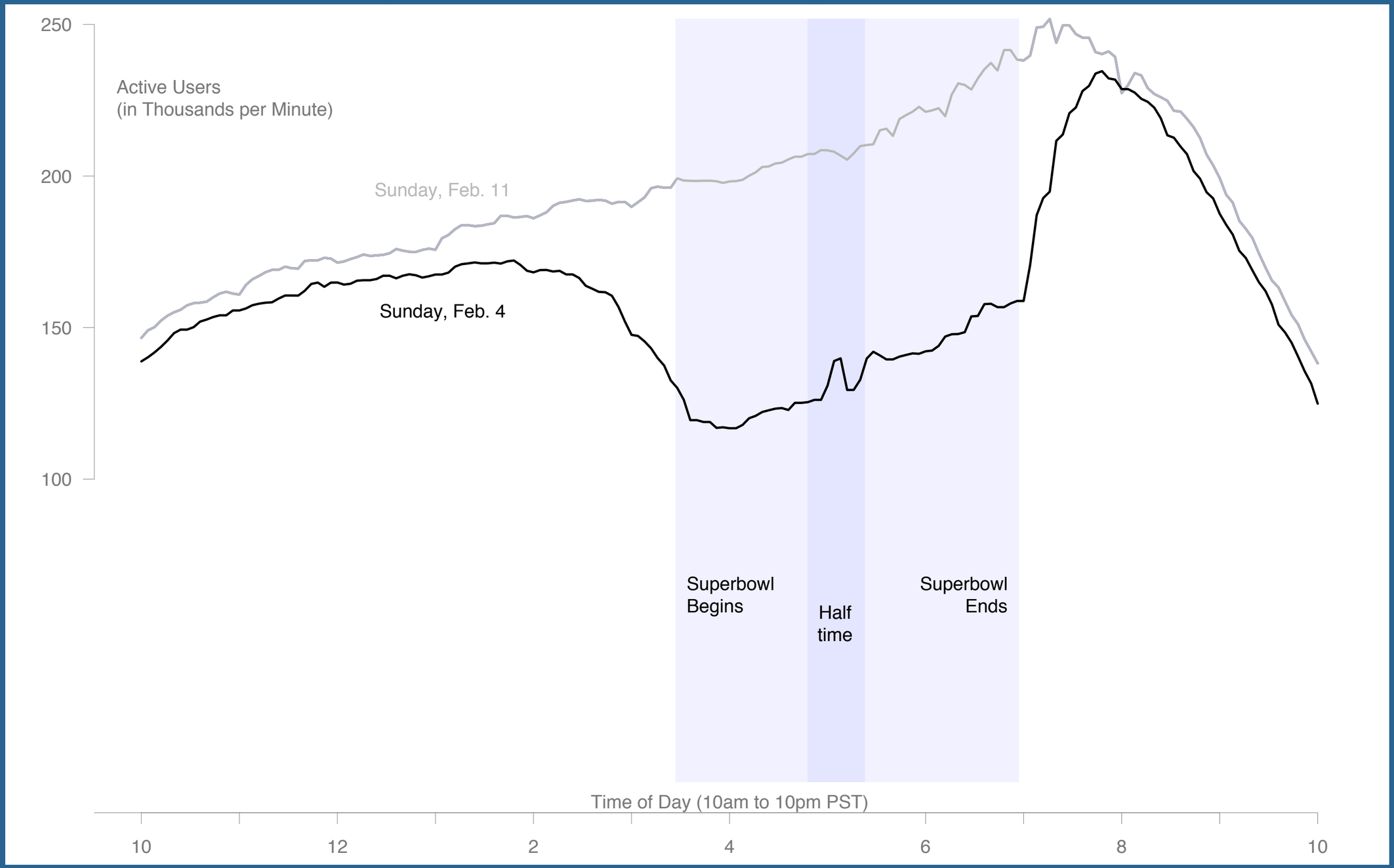
Facebook:

- 60+ Engineers
- 200+ Total Employees
- 23 Million active users in the last 30 days

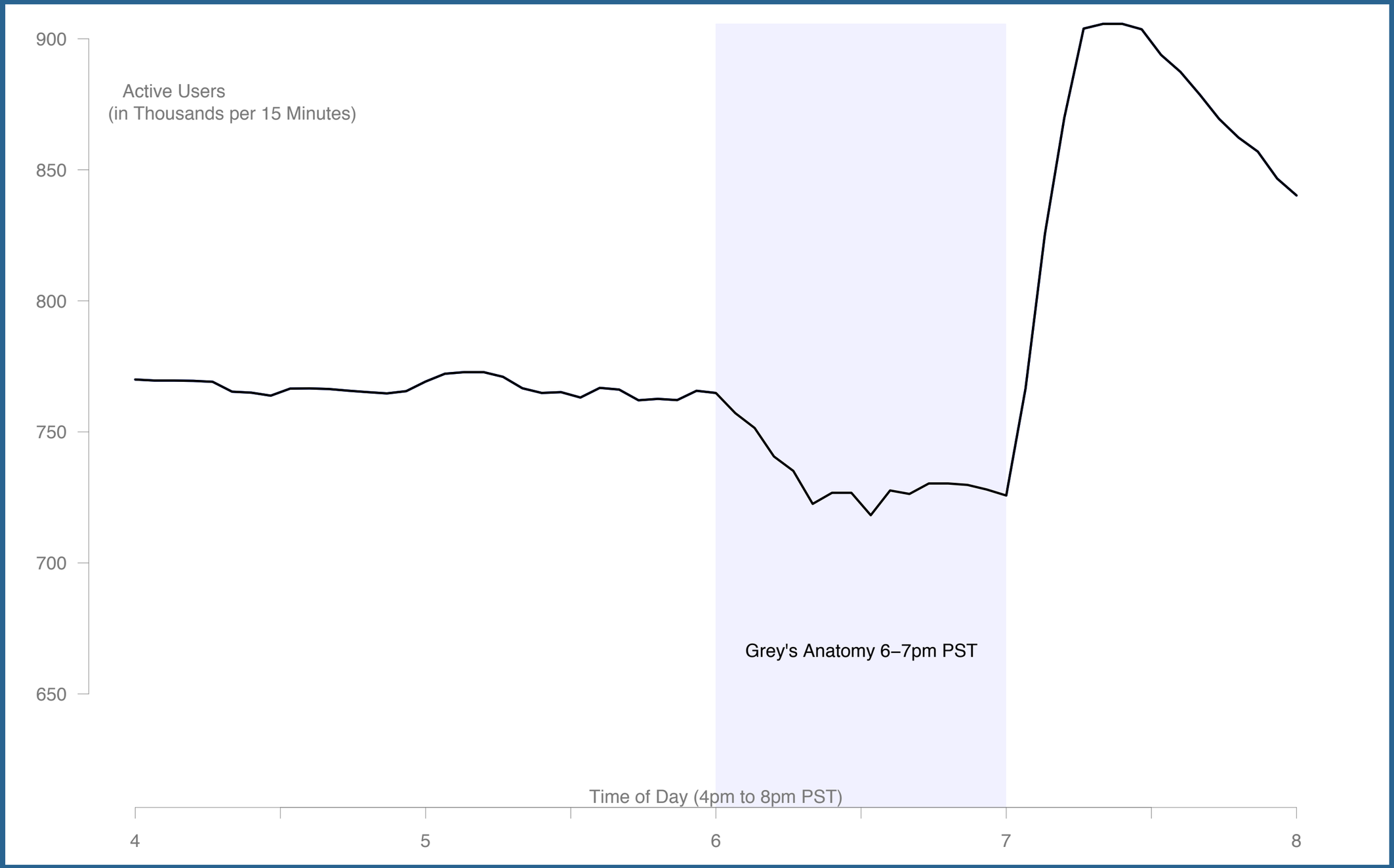
Facebook Growth



Super Bowl XLI: February 4th, 2007



Grey's Anatomy: February 22, 2007



The Open Source Advantage

Allows proprietary code alterations

Improvements contributed to open source community

Extended range of engineering possibilities

Discussions get voiced in a public forum

Some Facebook open source projects include **Thrift**, **phpsh** and the **Firefox Toolbar**.

http://developers.facebook.com/opensource.php

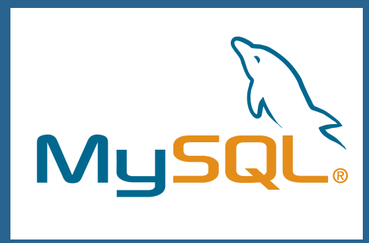
Facebook proudly uses:

PHP

Apache

Linux

GNU



MySQL

Vallgrind
Callgrind
Kcachegrind
APD
XDebug

OProfile
Memcached
Python
…

and the Alternative PHP Cache (APC)...

Alternative PHP Cache (APC)

http://pecl.php.net/packages/APC/

APC is a PHP intermediate opcode cache.

Provides a significant PHP interpreter performance increase.

Local user variable cache can also be used for application specific benefits.

Open source allows collaboration, bug fixes and optimizations.

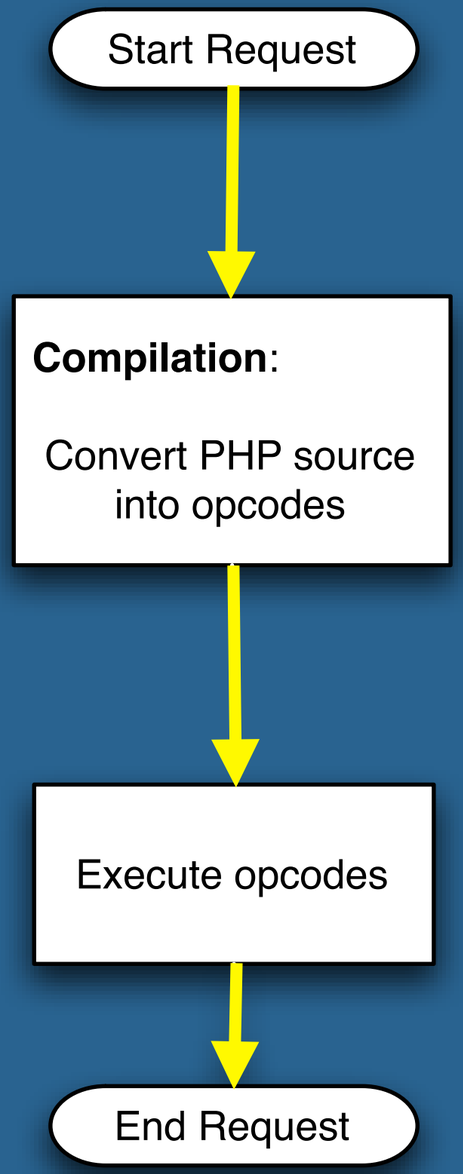
The APC Advantage

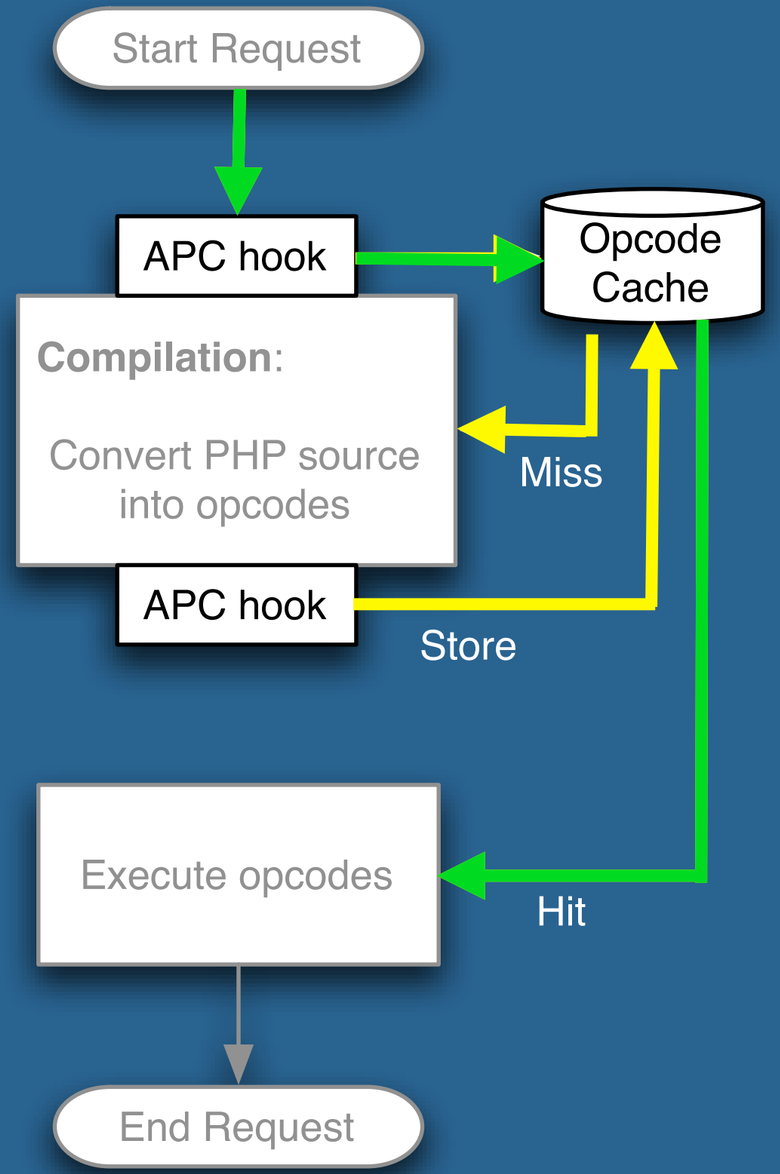
Requests per second provide a blunt measurement of performance gains on Facebook's profile.php page.

Configuration	Requests Per Second	
PHP	2.47 rps.	
APC User	5.24 rps.	2.12 x
APC User & File	22.01 rps.	8.91 x
Facebook	29.56 rps.	11.96 x

Measurements made using Apache Bench (ab) with 1000 total requests, concurrency of 40.
Executed on a dual Dual Core AMD Opteron 2.2Ghz, 8GB RAM.

PHP vs. APC


Start Request
Compilation:
Convert PHP source into opcodes
Execute opcodes
End Request


Start Request
APC hook
Opcode Cache
Compilation:
Convert PHP source into opcodes
Miss
APC hook
Store
Execute opcodes
Hit
End Request

From Code to Opcode

PHP Source

```
1 <?php
2 $output = 'Hello World!';
3 echo $output;
4 ?>
```

Opcodes

```
1 ASSIGN   !0   'Hello+World%21'
2 ECHO     !0
3 RETURN   1
4 ZEND_HANDLE_EXCEPTION
```

Opcode output generated with the Vulcan Logic Dissassembler (VLD)
by Derick Rethans (http://pecl.php.net/packages/VLD/)

From Code to Opcode

PHP Source

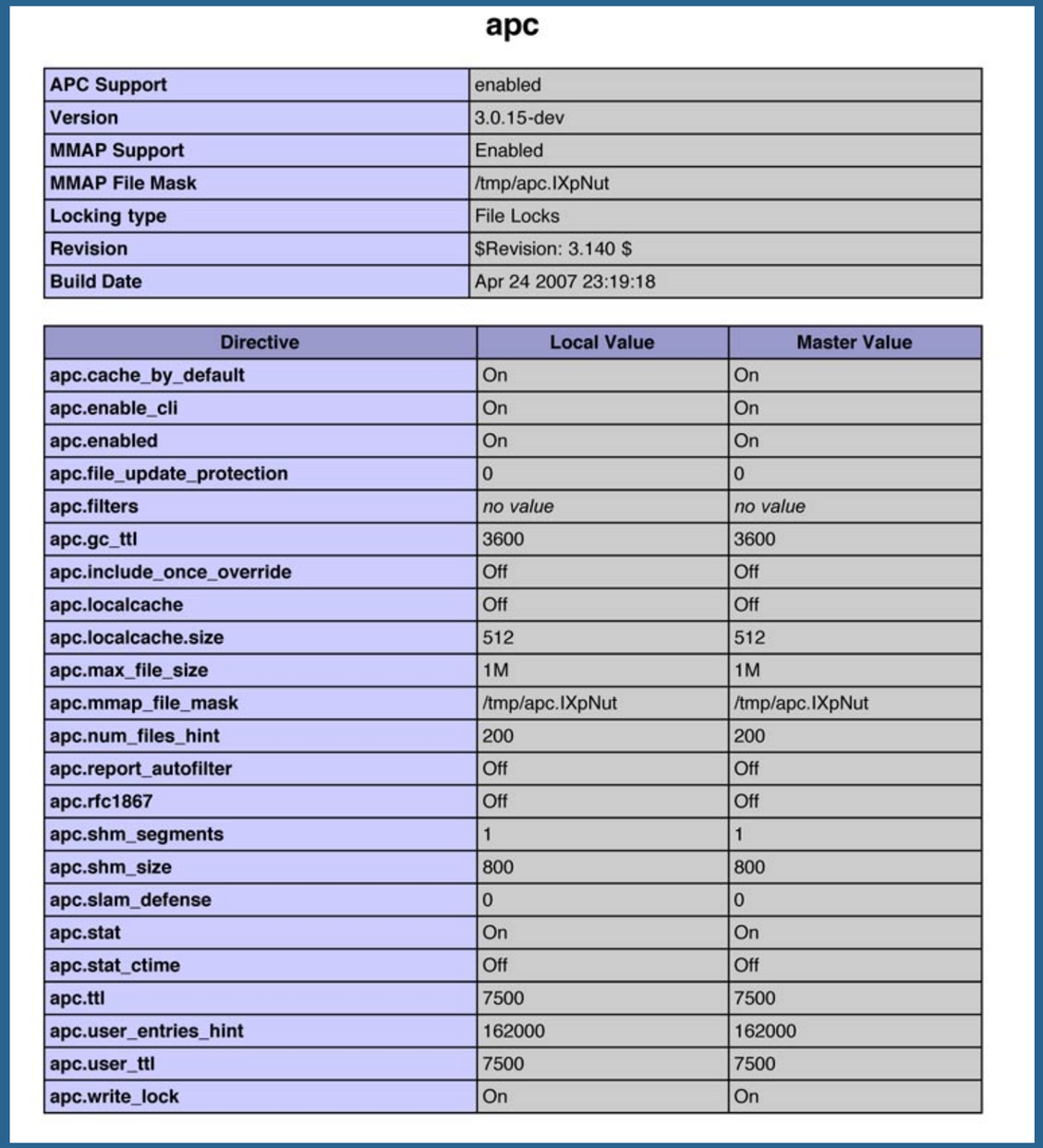
```
1 <?php
2 $output = 'Hello World';
3 if($_GET['exclaim']) {
4   $output .= '!';
5 } else {
6   $output .= '.';
7 }
8 echo $output;
9 ?>
```

Opcodes

```
 1 ASSIGN                    !0, 'HELLO+WORLD'
 2 FETCH_R      GLOBAL       $1, '_GET'
 3 FETCH_DIM_R               $2, $1, 'exclaim'
 4 JMPZ                      $2, ->6
 5 ASSIGN_CONCAT             !0, '%21'
 6 JMP                       ->7
 7 ASSIGN_CONCAT             !0, '.'
 8 ECHO                      $0
 9 RETURN                    1
10 ZEND_HANDLE_EXCEPTION
```

Opcode output generated with the Vulcan Logic Dissassembler (VLD) by Derick Rethans (http://pecl.php.net/packages/VLD/)

<? phpinfo(); ?>



apc

APC Support	enabled
Version	3.0.15-dev
MMAP Support	Enabled
MMAP File Mask	/tmp/apc.IXpNut
Locking type	File Locks
Revision	$Revision: 3.140 $
Build Date	Apr 24 2007 23:19:18

Directive	Local Value	Master Value
apc.cache_by_default	On	On
apc.enable_cli	On	On
apc.enabled	On	On
apc.file_update_protection	0	0
apc.filters	*no value*	*no value*
apc.gc_ttl	3600	3600
apc.include_once_override	Off	Off
apc.localcache	Off	Off
apc.localcache.size	512	512
apc.max_file_size	1M	1M
apc.mmap_file_mask	/tmp/apc.IXpNut	/tmp/apc.IXpNut
apc.num_files_hint	200	200
apc.report_autofilter	Off	Off
apc.rfc1867	Off	Off
apc.shm_segments	1	1
apc.shm_size	800	800
apc.slam_defense	0	0
apc.stat	On	On
apc.stat_ctime	Off	Off
apc.ttl	7500	7500
apc.user_entries_hint	162000	162000
apc.user_ttl	7500	7500
apc.write_lock	On	On

phpinfo() displays configuration information

Verify configuration settings have taken place

Debug configuration problems

Basic Configuration Options

apc.enable	**0**	Enable APC
apc.enabled_cli	**0**	Enable APC when running via the command line
apc.mmap_file_mask	**NULL**	Name of the file mask where as specified by mmap

Cache Size and Hints

apc.shm_segments	1
apc.shm_size	30

@ Facebook:

apc.shm_segments=1
apc.shm_size=648

One shared memory segment required

Size needs to hold all file and user entries

Monitor usage using apc.php

Extra space is required for deleted entries

Behaves poorly when memory is at capacity

apc.num_files_hint	1000
apc.user_entries_hint	4096

@ Facebook:

apc.num_files_hint=100
apc.user_entries_hint=640000

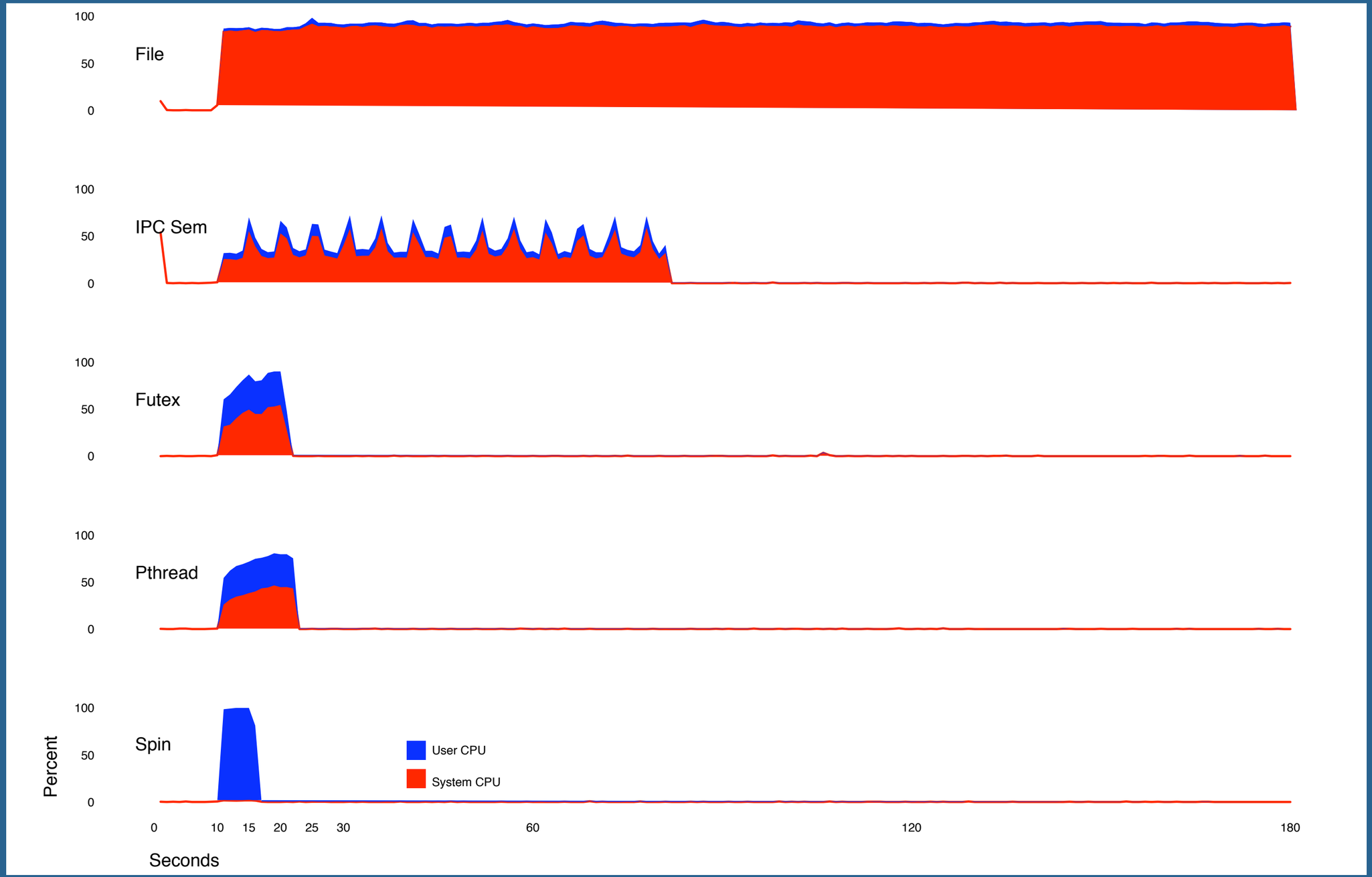
Hints optimize hash lookup tables

Maximum number of files or user entries

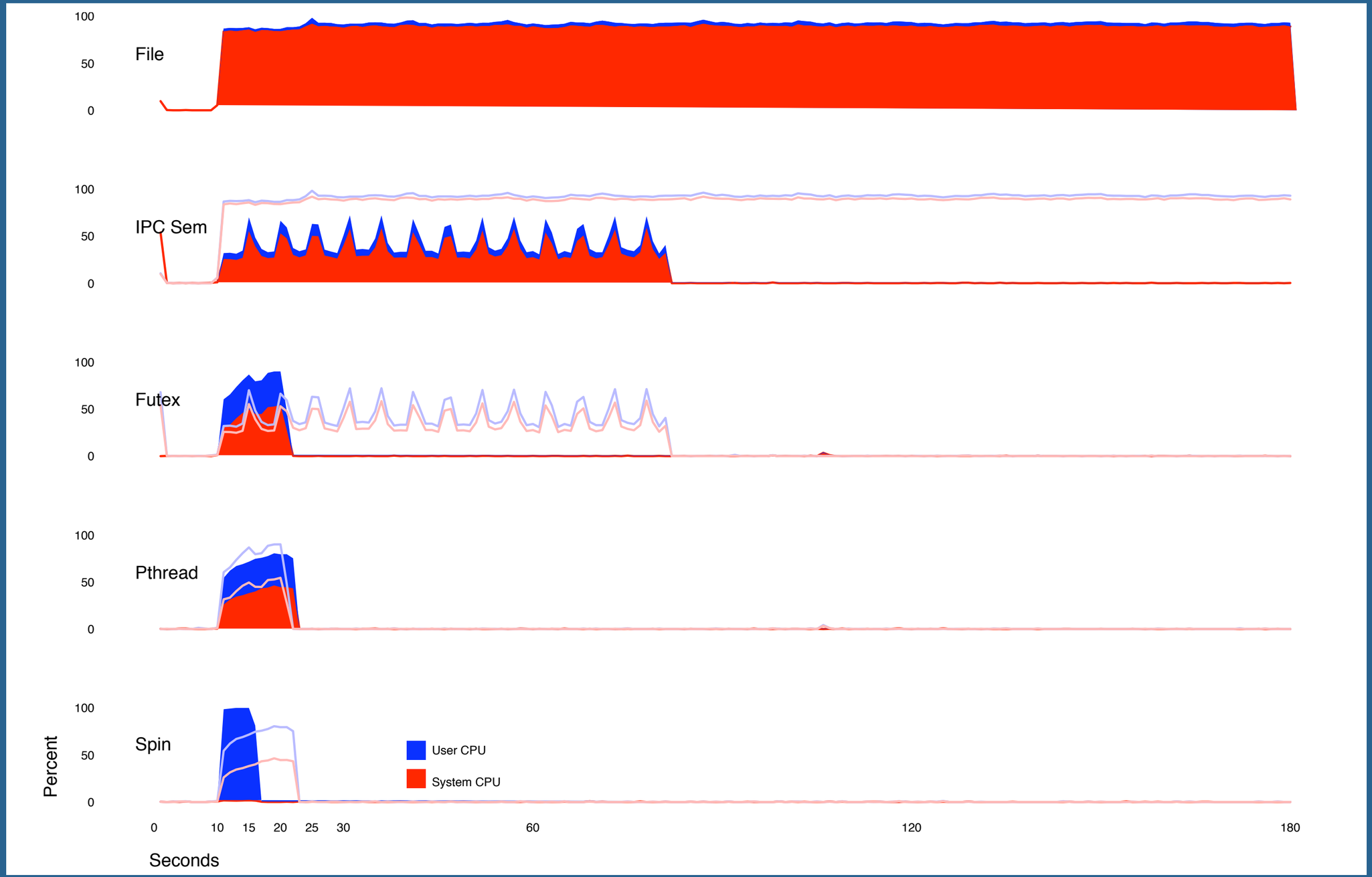
Locking Mechanisms

Locking type	
File Locks *Default*	Uses file locking operations Stable, not effecient
IPC Semaphore Locks	Faster alternative to file locks
Linux Futex Locks *EXPERIMENTAL*	Architecture specific, Linux Kernel 2.6.x or later Significant performance gain
pthread mutex Locks *EXPERIMENTAL* *Currently Used @ Facebook*	Better alternative to Linux Futex Locks Same performance gain More stable, more architecture support
spin Locks *EXPERIMENTAL*	Ported from the PostgreSQL project Runs in user space

Locking Performance



Locking Performance (Overlayed)



To Stat or Not To Stat?

apc.stat	TRUE

@ Facebook:
apc.stat=FALSE

APC stats files to determine if they've been updated

Disabling updates will increase performance

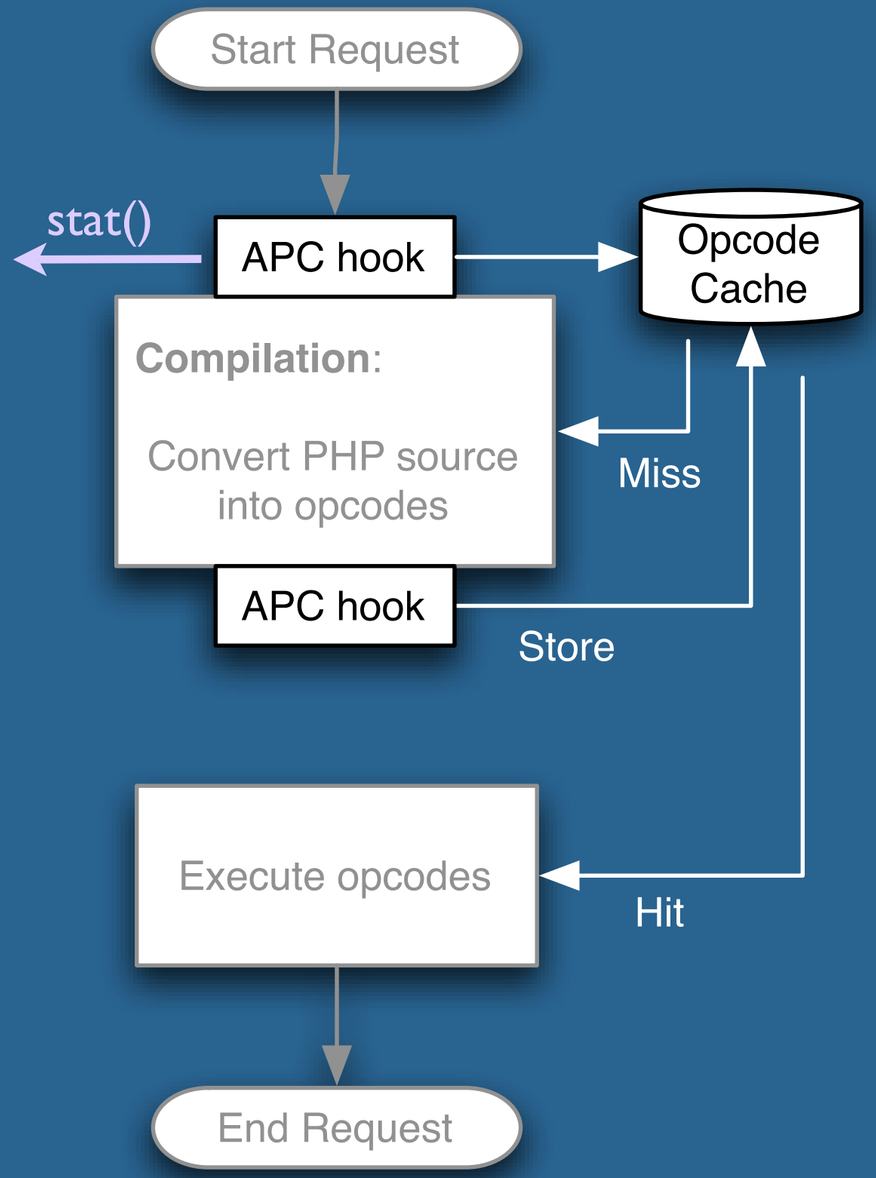
Requires restart or apc_cache_clear() to update

apc.stat_ctime	FALSE

@ Facebook:
apc.stat_ctime=FALSE

CVS, SVN and rsync backdate modified times

stat_ctime checks the creation time for updates



Slam Defenses & TTL

apc.slam_defense	**0**

Spreads load on startup by only caching given percent of requests

apc.write_lock	**1**

A non-blocking write lock provides a better solution to setting slam_defense

apc.file_update_protection	**2**

Read updated files after given delay to prevent loading incomplete files

apc.ttl	**0**
apc.user_ttl	**0**

"Time to Live"

Maximum time a cache entry can remain in cache.

apc.gc_ttl	**3600**

Inline garbage collector removes deleted entries from cache as soon as possible.

Entries still in use by a request will be removed after the gc_ttl has expired.

Filters

apc.max_file_size	**1M**	Limits the maximum file size that will be cached.
apc.filters	**NULL**	A regular expression that excludes files from being cached.
apc.cache_by_default	**1**	Setting to zero causes files to only cache if they match apc.filters.
apc.report_autofilter	**0**	Logs files excluded due to early/late binding issues

Recent Features

apc.include_once_override	**0**

EXPERIMENTAL

Optimizes include_once() calls

apc.rfc1867	**0**

EXPERIMENTAL

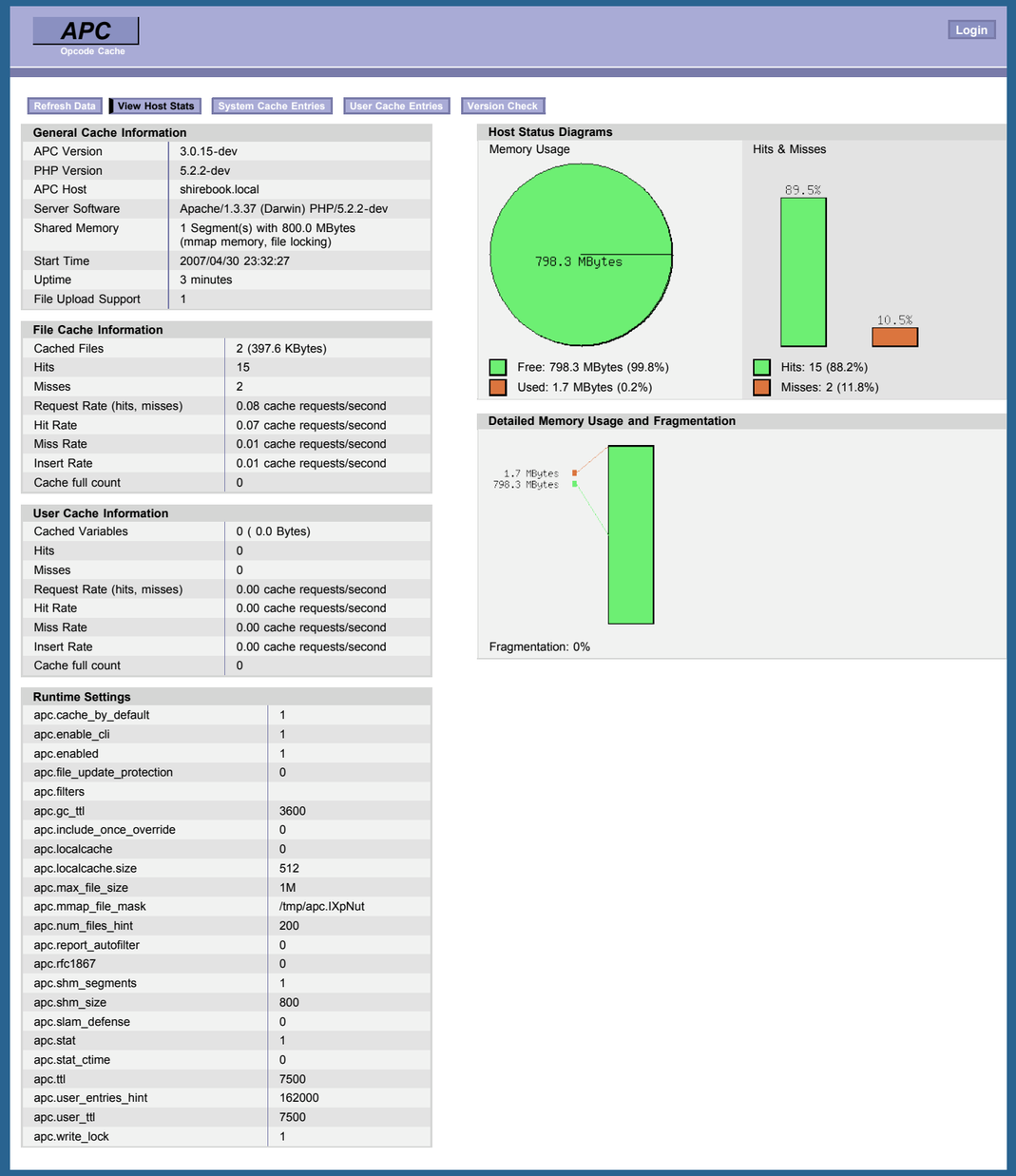
Upload progress support

apc.localcache	**0**
apc.localcache_size	**512**

EXPERIMENTAL

A process localized cache

apc.php



apc.php is located in the APC source directory

User and file cache browser

Graphs of hit rates and memory usage

API

http://us.php.net/manual/en/ref.apc.php

array **apc_cache_info**(*string* cache, *boolean* limited)	Cache information
array **apc_sma_info**(*boolean* limited)	Shared memory segment information.
boolean **apc_store**(*string* key, *mixed* value, *int* ttl)	Store key/value pair in cache.
boolean **apc_add**(*string* key, *mixed* value, *int* ttl)	Add key/value pair if key isn't in cache.
mixed **apc_fetch**(*string* key)	Fetch the value associated with key.
boolean **apc_delete**(*string* key)	Delete the value associated with key.
boolean **apc_clear_cache**(*string* cache)	Clear all entries from the cache.
boolean **apc_compile_file**(*string* file)	Compile given file and store in cache, bypass all filters.
boolean **apc_define_constants**(*string* key, *array* constants *bool* case_sensitive)	Define an array of key/value constants.
boolean **apc_load_constants**(string key, bool case_sensitive)	Assign each key/value constants in cache using define()

The User's Cache

User cache stores PHP variables across multiples requests on a per server basis.

Primary commands for utilizing the user cache:

boolean apc_store(*string* key, *mixed* value)

mixed apc_fetch(*string* key)

Optimize...

Application configuration

Statistics such as site usage, request types, error conditions, timing

Nth tier cache in addition to memcache or other caching service

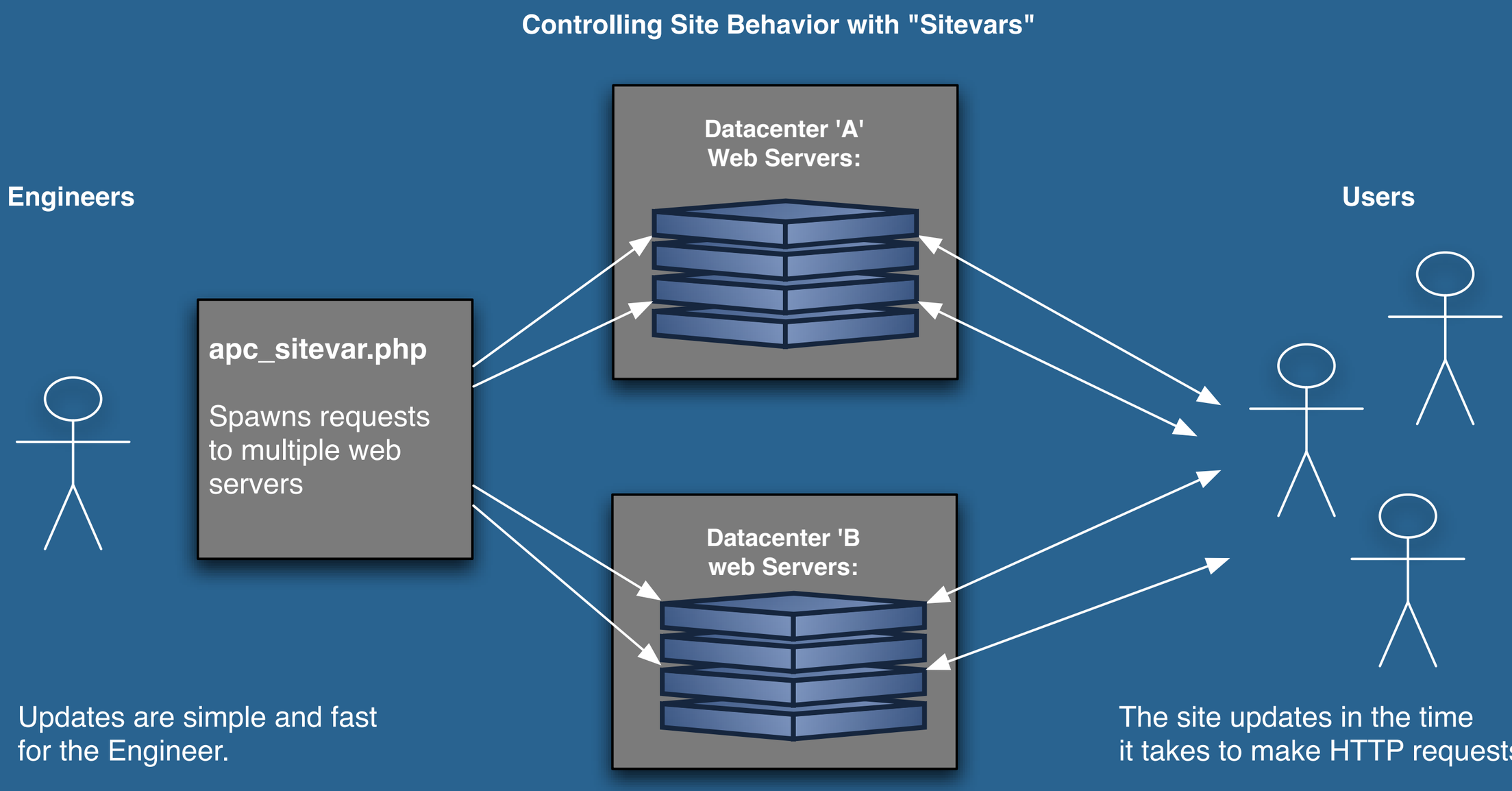
Database backed values that only change rarely like product listings

HTML or other output

Site behavior like new features, A/B tests, or rate controls

Site-wide Server Variables: “Sitevars”

Facebook controls site configuration and features via the user cache.



Optimus Prime

Facebook primes it's cache before each code push or server restart.

Ready to serve requests without delay due to compilation or updating cache values.

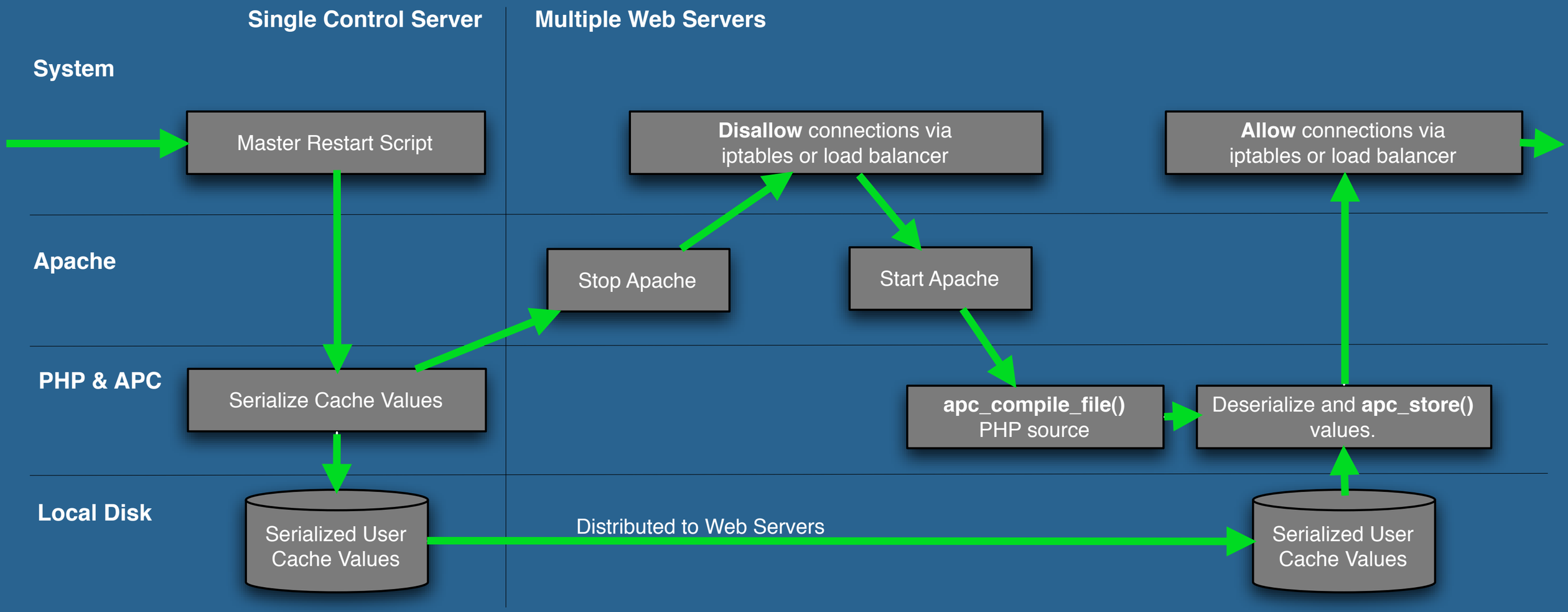
Handles immediate flood of requests with limited warm-up time.

Starts with same cache state limiting differences between servers.

boolean apc_store(*string* key, *mixed* value)

boolean apc_compile_file(*string* file)

APC Priming and Restart



Getting Started

Installation via “pecl install apc”
or source http://pecl.php.net/packages/APC/

Start with a basic apc.ini file:

apc.enabled=1
apc.shm_size=100M

Install apc.php under the DocumentRoot, configure the USER and PASS variables

Monitor apc.php for usage and adjust configuration

Try tuning some of the discussed settings to meet application needs

Add some APC user variables

Try different locking types

Try the apc.stat=0 setting

Measure changes in CPU usage and maximum requests per second

APC Developers

George Schlossnagle

Daniel Cowgill

Rasmus Lerdorf

Gopal Vijayaraghavan

Edin Kadribasic

Ilia Alshanetsky

Marcus Börger

Sara Golemon

Thank you!