Facebook Performance Caching

Lucas Nealan
DC PHP Conference
November 7, 2007, 16:00 – 17:00

Facebook

Social Platform

Sharing and communicating efficiently

6th most trafficked site in the U.S.*

* ComScore 2007

Facebook Stats

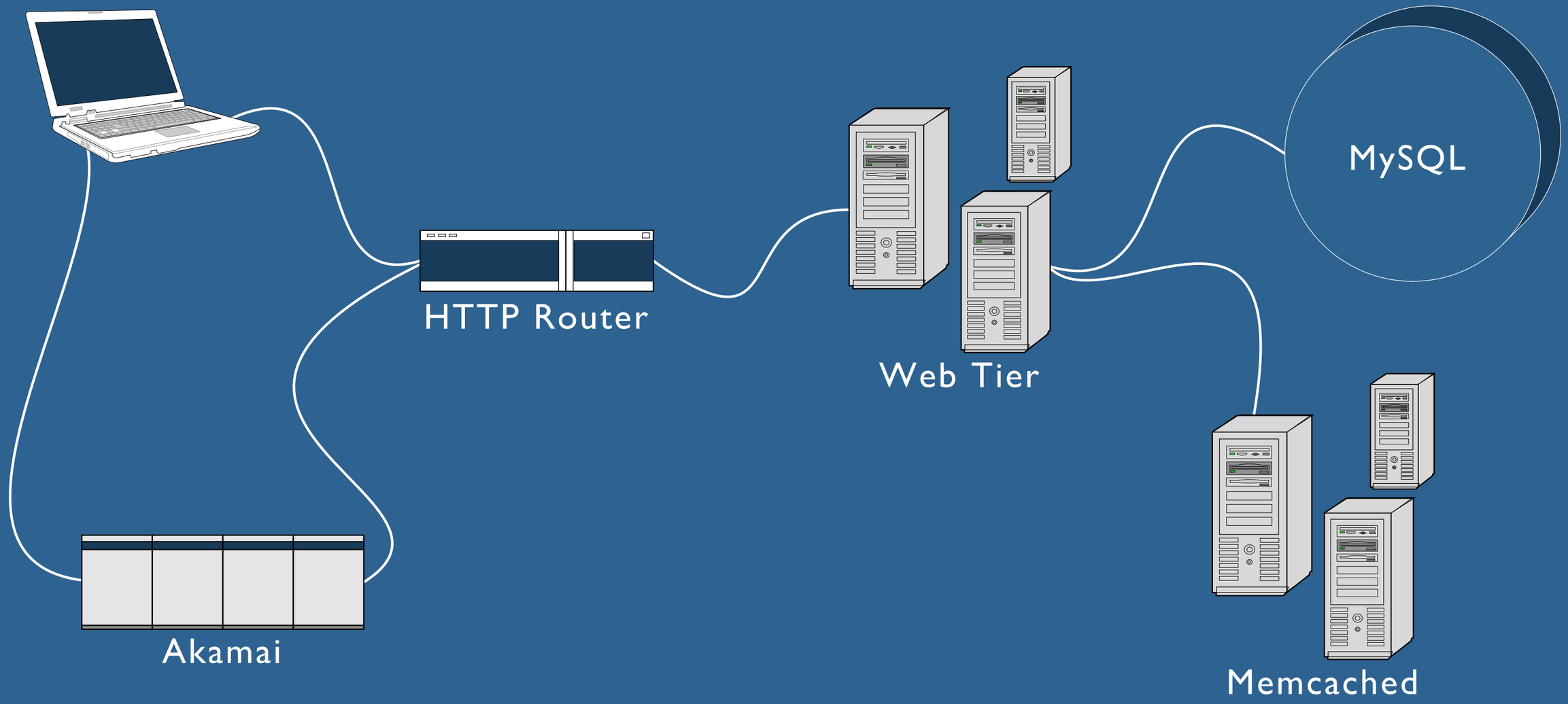
Over 52 million active users

~ 50 pages per user daily

250,000 new users daily

Over 7,000 platform applications

Facebook Architecture



Complexity

Connecting to all Database is impossible

Large codebase

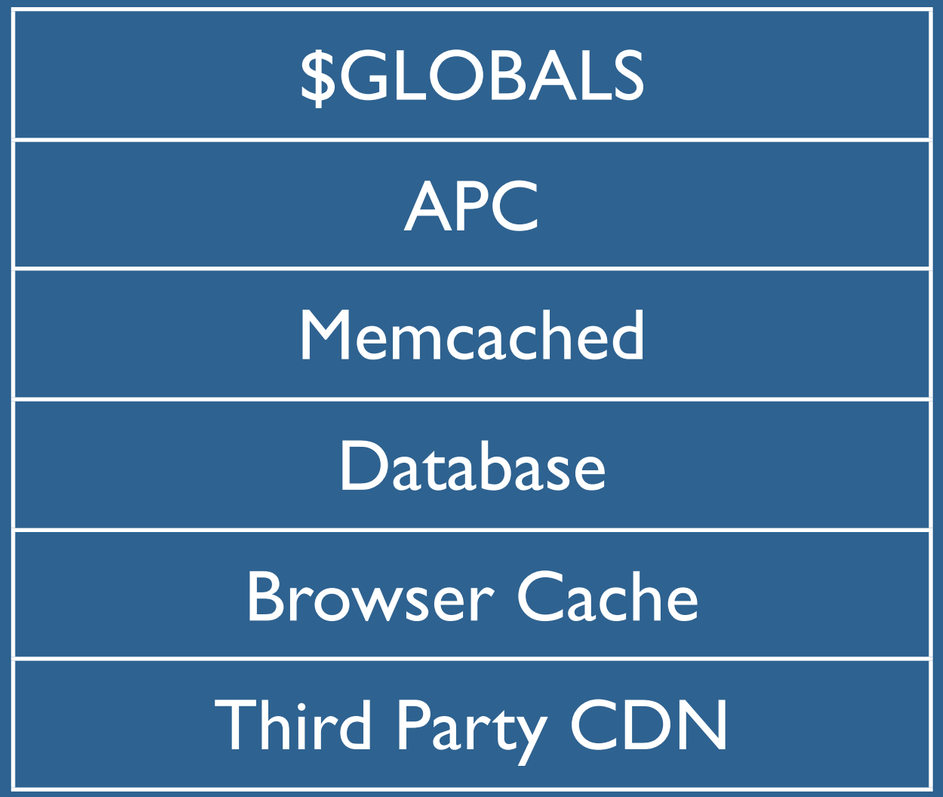
Scaling affects resources in many ways

- Memory consumption
- Socket connection limits

Cache retrieval is ~ 10% cpu-user of most pages

What are the Benefits of Caching?

Caching Layers



Globals Cache

```
function cache_get($id, $key, $apc=false) {
  if (isset($GLOBALS['CACHE']["$key:$id"])) {
    $cache = $GLOBALS['CACHE']["$key:$id"]));
    $hit = 1;
  } elseif ($apc && (($cache = apc_fetch("$key:$id")) !==
             false) {
    $hit = 1;
  } else {
    ... // fetch from memcached
    if ($apc) apc_store("$key:$id", $cache);
  }
  if ($hit) $GLOBALS['CACHE']["$key:$id"] = $cache;
  return $hit ? $cache : NULL;
}
```

Globals Cache

Avoids unnecessary APC and Memcached requests

Automatic via abstraction

But still has function call cost overhead

```
foreach($ids as $id) {
  if (isset($GLOBALS['CACHE']["profile:$id"])) {
    $profile = $GLOBALS['CACHE']["profile:$id"];
  } else {
    $profile = cache_get($id, 'profile');
  }
}
```

APC

Opcode caching

- Hundreds of included libraries
- Thousands of functions

Variable caching

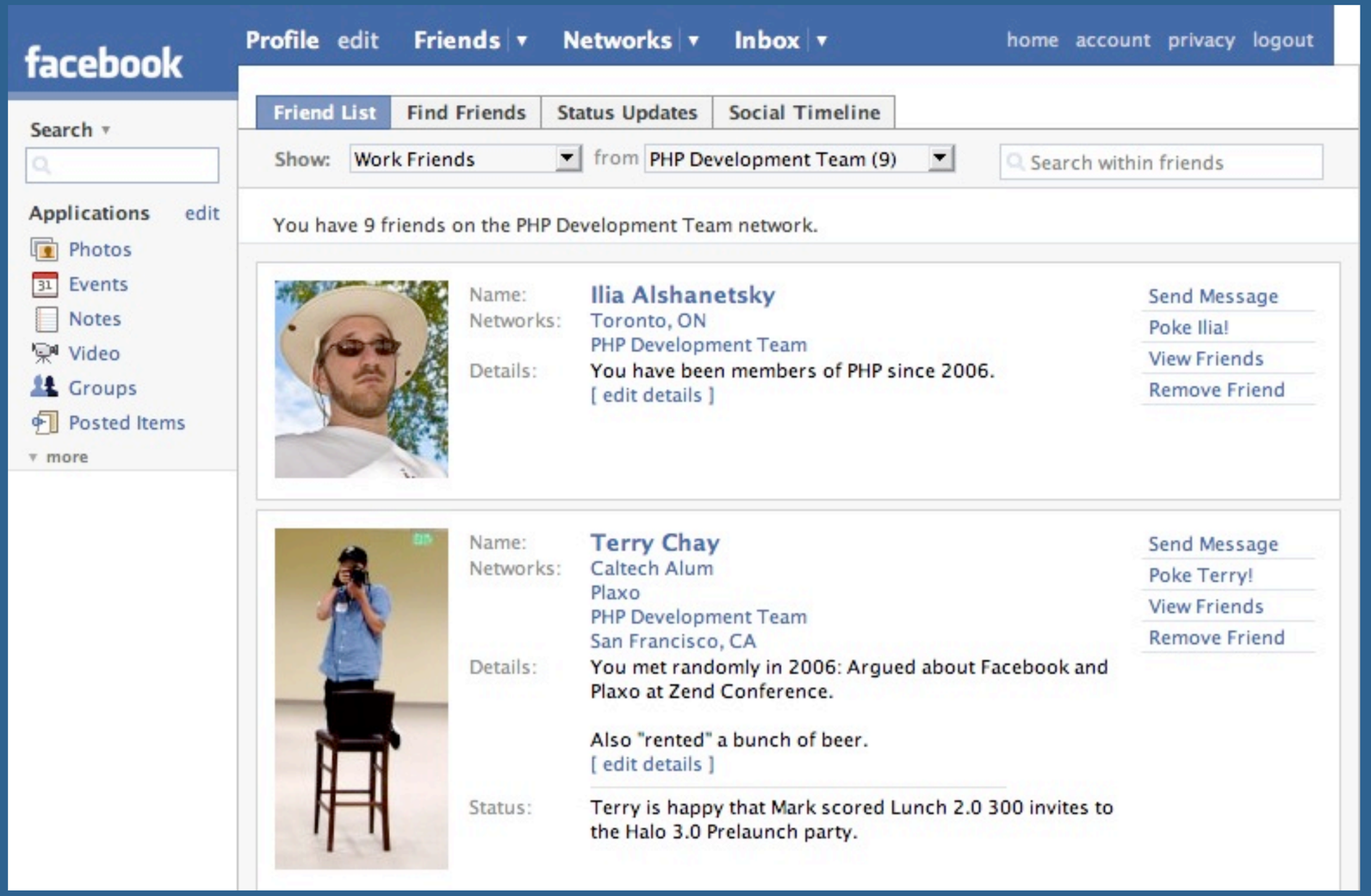
- Hundreds of MB's of data

APC ~~User~~ Cache

Non-user specific data

- Network information
- Database information
- Useragent strings
- Hot application data
- Site variables

Friends



Normal	4050ms
APC	135ms
apc.stat=0	128ms

APC Opcode Priming

```
$path    = '/path/to/source';
$exclude = $path.'/www/admin/';
$expr    = '.*\\.php';
$files = split(' ', exec("find -L $path -regex '$expr'
                          | grep -v '$exclude' | xargs"));

// prime php files
foreach($files as $file) {
  apc_compile_file($file);
}
```

APC+SVN Client Cache Busting

```
function get_static_suffix($file) {
  global $ROOT;
  if ($version = cache_get($file, ‘sv’, 1) === null) {
    $version = trim(shell_exec(“svn info $ROOT/$file | grep
    ‘Changed Rev’ | cut -c 19-”));

    apc_store(“sv:$file”, $version);
  }

  return '?'.$version;
}
```

APC + Useragent Strings

Useragent string parsing is inefficient in PHP

Cache parsed useragents in APC for the first 10 minutes

Hit rate of over 50%

Pear implementation available soon:

- PEAR::Net_Useragent_Detect_APC

Site Variables

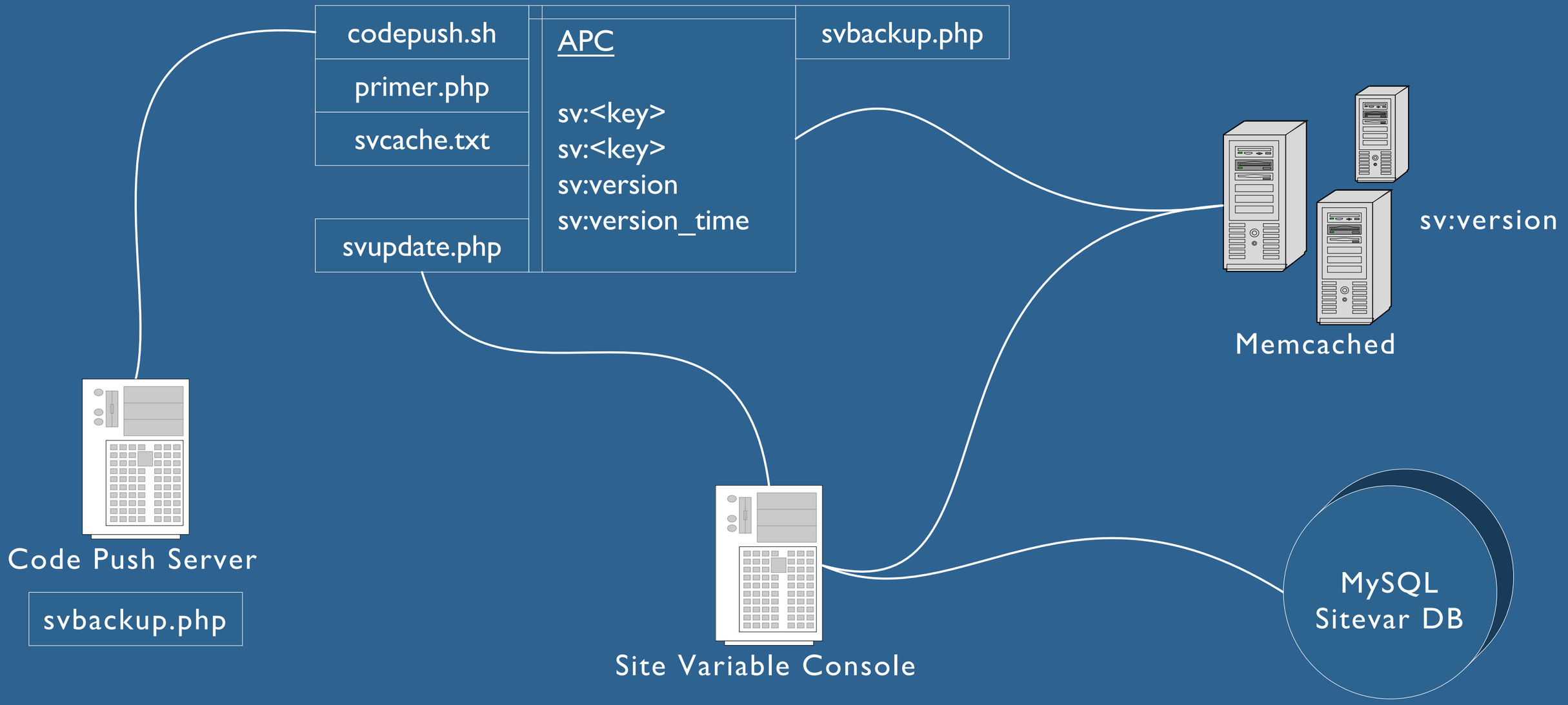
Enable/Disable site features across all servers

Configure memcached cluster IP's

Configure product features

Version memcached keys for invalidation

Site Variables



Memcached

Distributed object cache

Facebook currently utilizes > 400 memcached hosts

With > 5TB in memory cache

Facebook contributions:

- UDP Support
- Performance Enhancements

Many choices in opensource clients

What to cache?

User Specific Data

- Long profile
- Short profile
- Friends
- Applications

Key versioning

- sp:6:10030226

Cache Retreival

Create Wrapper functions:

- `cache_get($id, <key>, <miss>, $apc, $timeout);`
- `cache_get_multi($ids, <key>, <miss>, $apc);`

Cache key callback function:

```
function profile_key($id) {
  global $VERSION_SP; // primed site variable
  return “sp:$version:$id”;
}
```

Cache Multiget

Sort keys into buckets of servers

For each server

- obtain connection
- send requests

For each server

- read responses
- deserialize non-scalar types

```
foreach($keys as $key => $kdata) {
  $host = get_host($key); // hash
  $keys_hosts[$host][$key] = $kdata;
}

function get_host($key) {
  global $servers;

  $prefix = $key[0].$key[1].$key[2];
  $prefix = isset($servers[$prefix) ?
            $prefix : ‘wildcard’;

  $hash = crc32($key);
  $host = $servers[$hash %
            count($servers[$prefix]);
}
```

Cache Multiget

Sort keys into buckets of servers

For each server

- obtain connection
- send requests

For each server

- read responses
- deserialize non-scalar types

```
if (isset($cache_sock[$host])) {
  return $cache_sock[$host];
}

list($ip, $port) = explode(':', $host);

while($try < 5 && !$sock) {
  $sock = pfsockopen($ip, $port ...);
  $try ++;
}

stream_set_write_buffer($sock, 0);

$cmd = "get $keys\r\n";
fwrite($sock, $cmd);
```

Profile Multigets

Profile info

Profile installed platform applications

Viewer installed applications

Platform application data

List of friends

Privacy data

Cache Dispatching

```
cache_get_multi($ids, <key>, <miss>, $apc,
                $pending=false, &$pending_arr);

cache_dispatch();
```

Combine requests for data from the same memcache server

- Up to 10% performance improvement

Execute code while the kernel buffers the memcache response

Profile Multigets

```
<?php
include_once(...);

parse_arguments();
check_permissions();
check_friends();
check_friend_status();
• get_profile();
render_basic_information();
get_friend_details();
render_minifeed();
render_wall();
• get_photos();
count_photos();
• get_applications();
render_menu_actions();
• get_friends();
render_friends();
• get_networks();
render_networks();
render_applications();
```

```
<?php
include_once(...);

• get_profile(true);
• get_photos(true);
• get_applications(true)
• get_friends(true)
• get_networks(true)
cache_dispatch();

parse_arguments();
check_permissions();
check_friends();
check_friend_status();
• get_profile();
render_basic_information();
get_friend_details();
render_minifeed();
render_wall();
get_photos();
count_photos();
get_applications();
render_menu_actions();
get_friends();
render_friends();
get_networks();
render_networks();
render_applications();
```

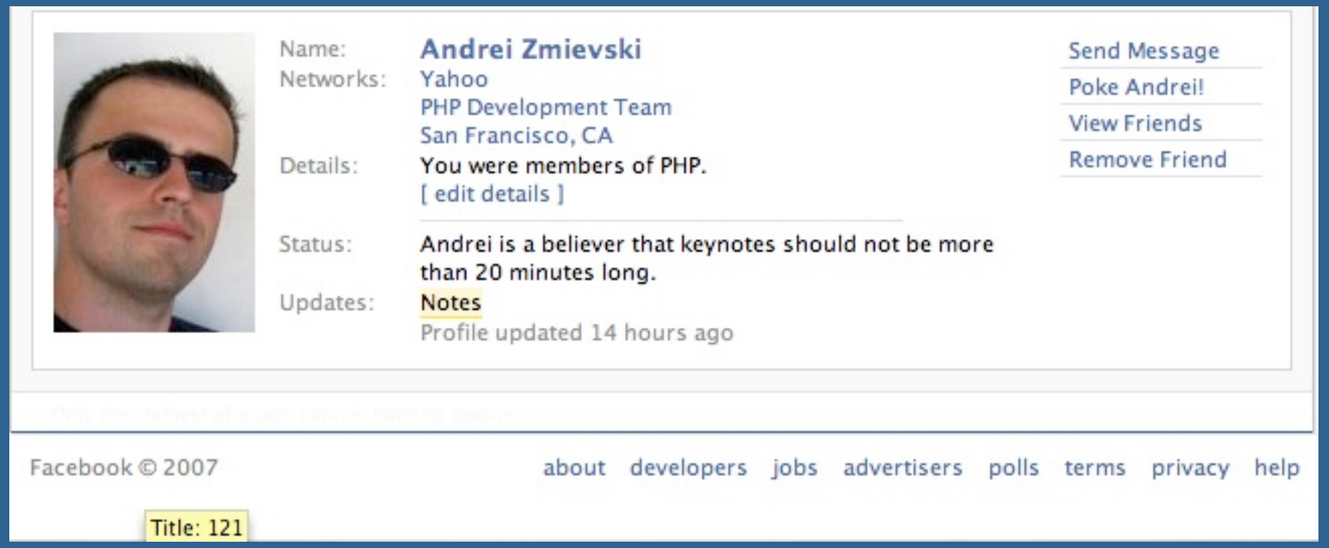
Lets make it even faster

Memcached PHP extension

- Reduced PHP function calling overhead
- Socket blocking in C instead of userspace PHP
- UDP support

Friends Again

Memcached extension runs ~ 10% faster realtime than in PHP userspace



Userspace	131ms
Extension	115ms
Extension w/ UDP	122ms

Serialization

Facebook internal serialization

- Profiles store 38% less memory in memcache
- Improves network throughput and realtime I/O

Incl.	Self	Called	Function
1.36	1.36	293	unserialize
1.04	1.04	293	fb_thrift_unserialize

Compression

- Investigating LZO compression for even more space savings

UDP Memcached

TCP Limitations

- Maximum simultaneous connect count of ~ 250k
- Impedes scalability of memcached clusters

Requires a very stable network environment

Occasional misses are acceptible

Reduces kernel buffer memory usage on clients and servers

Supported in Facebook Memcache Extension coming soon

A Dirty Problem

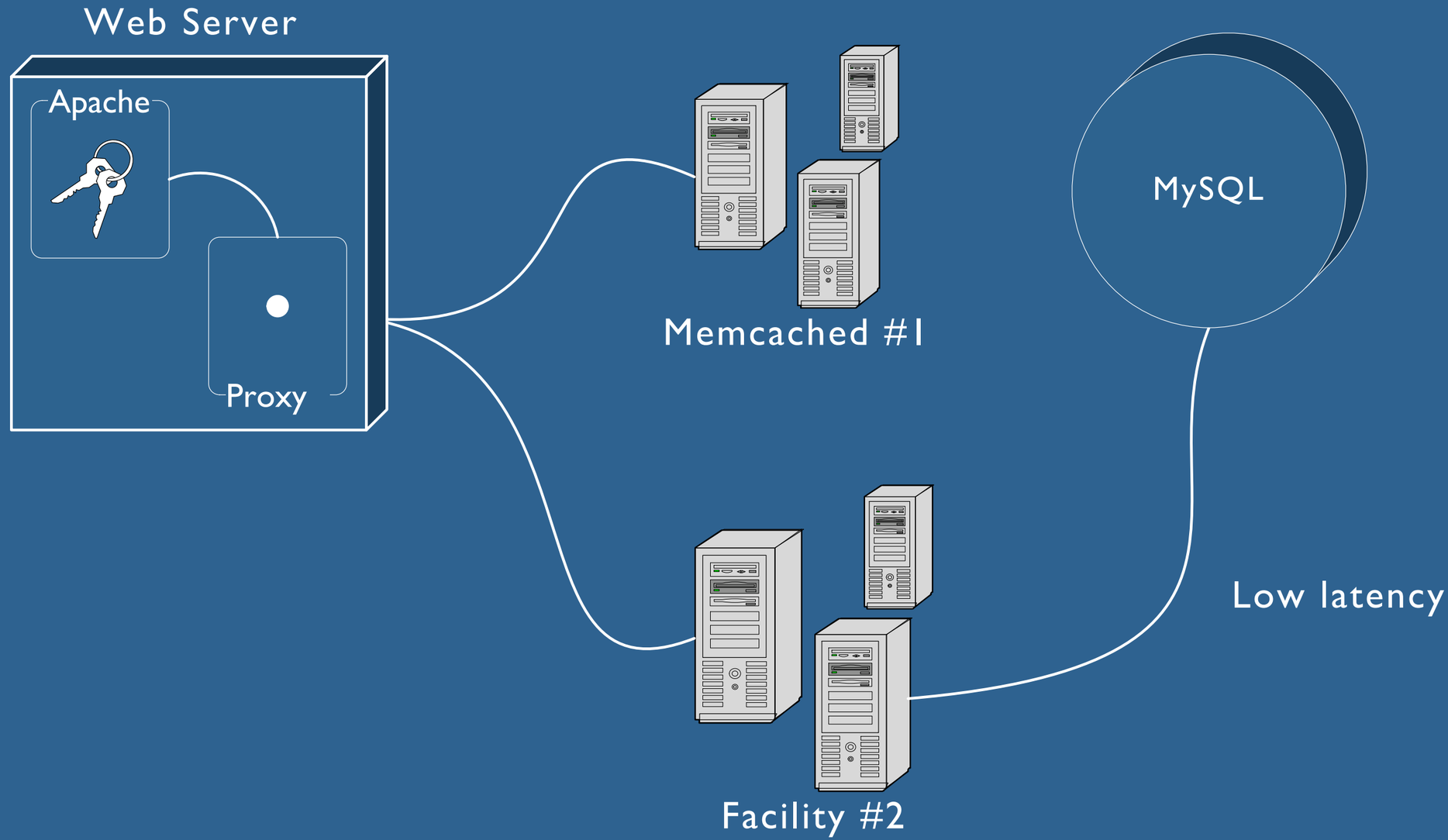
Wrapper functions

- `cache_dirty($id, $key);`

Actively dirty cache entries as users change data

Dirty entries between multiple hosting facilities via proxy

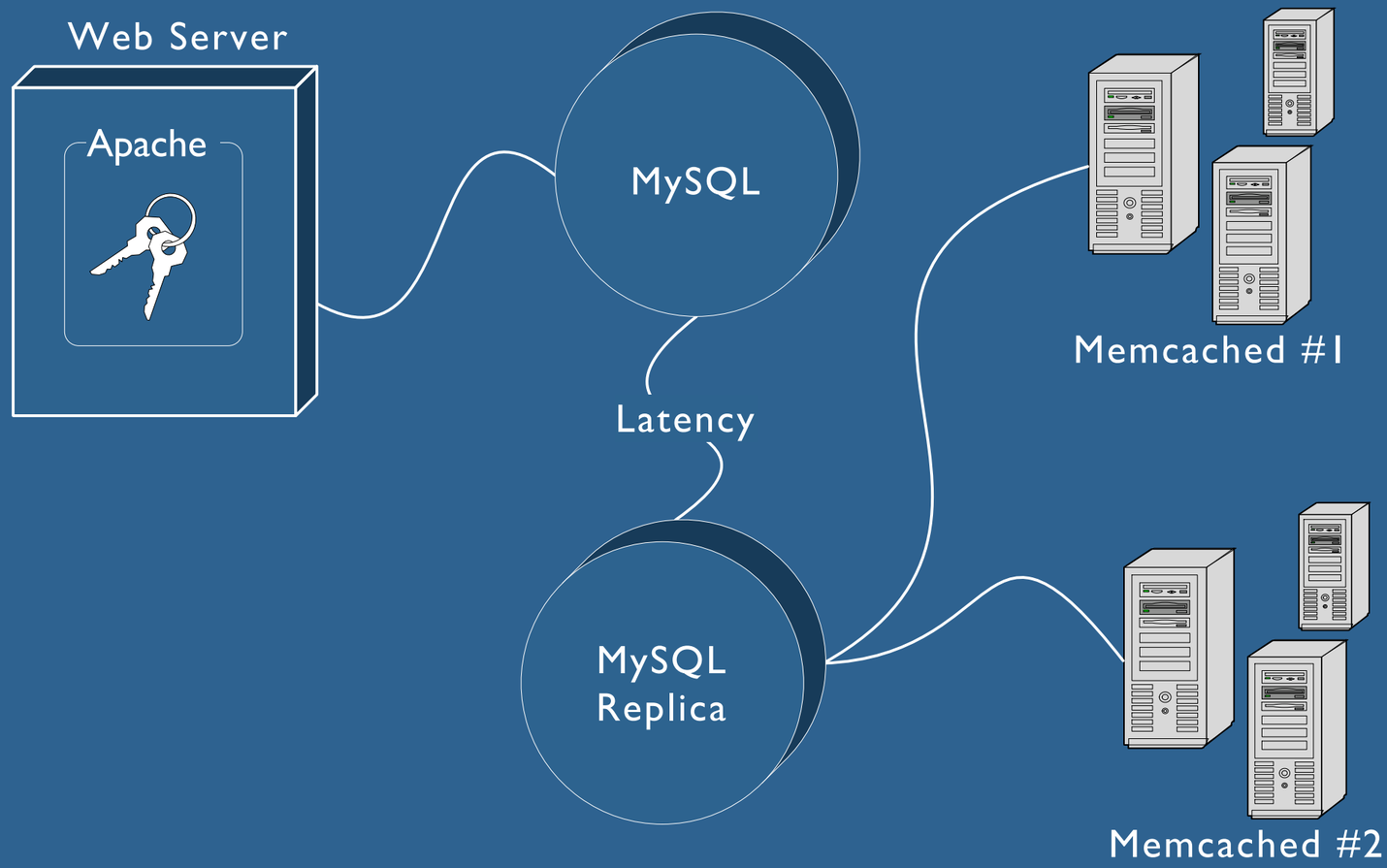
Memcached Proxy



Latency

Proxy deletes only work with low latency between facilities

When facilities are further apart deletes need to be smarter



Presention online

http://sizzo.org/talks/

lucas@facebook.com

PHP Development and the Facebook Platform

Dave Fetterman

Tomorrow 10:30 – 11:15
Room “A”