

Drinking from the Fire Hose

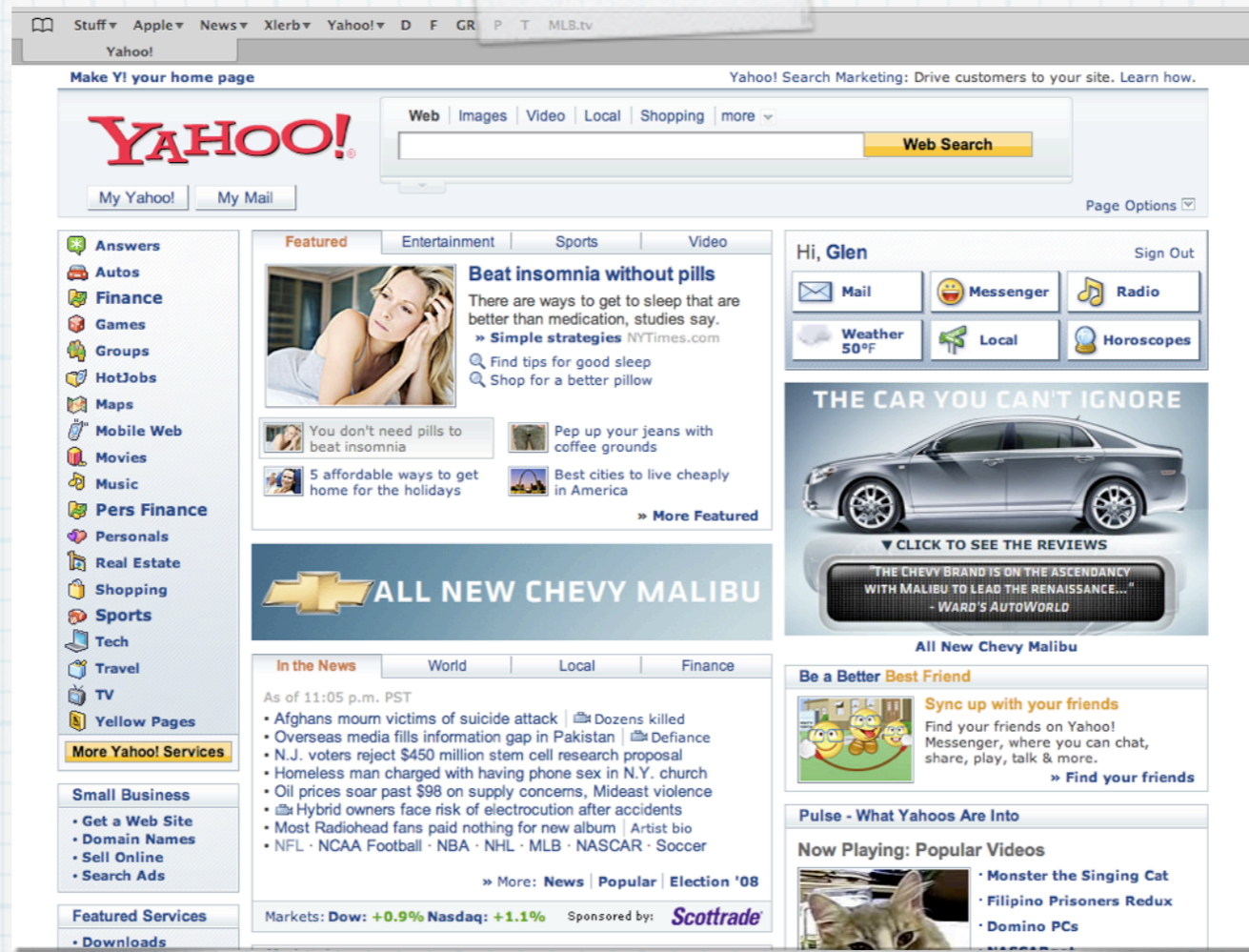
Optimizing Service-Oriented Architectures
for High Performance

Glen Campbell

Engineering Manager, Yahoo! Lifestyles
(We're hiring! See me later!)



The most-visited page on
the Internet, 1997



The most-visited page on
the Internet, 2007

What we're talking about today

- * Websites
- * Architectures
- * PHP Coding
- * Caching

What we're not talking about

- * Web Applications
- * Java
- * Coffee
- * The metaphysical questions surrounding man's being, sense of purpose, and ultimate destiny. Or college football.

Scalability

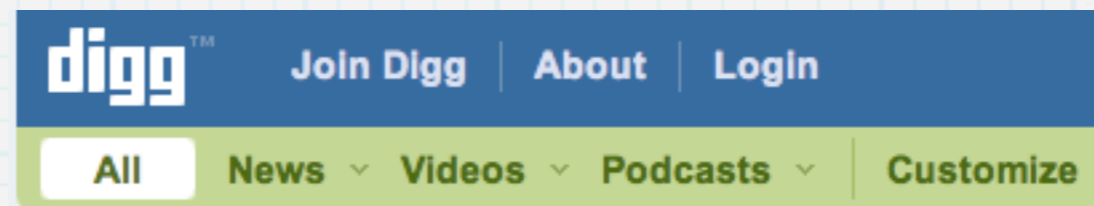
The ability of a system to continue to perform (or gracefully degrade) under situations of extreme stress.

What are we talking about?

* The Slashdot Effect



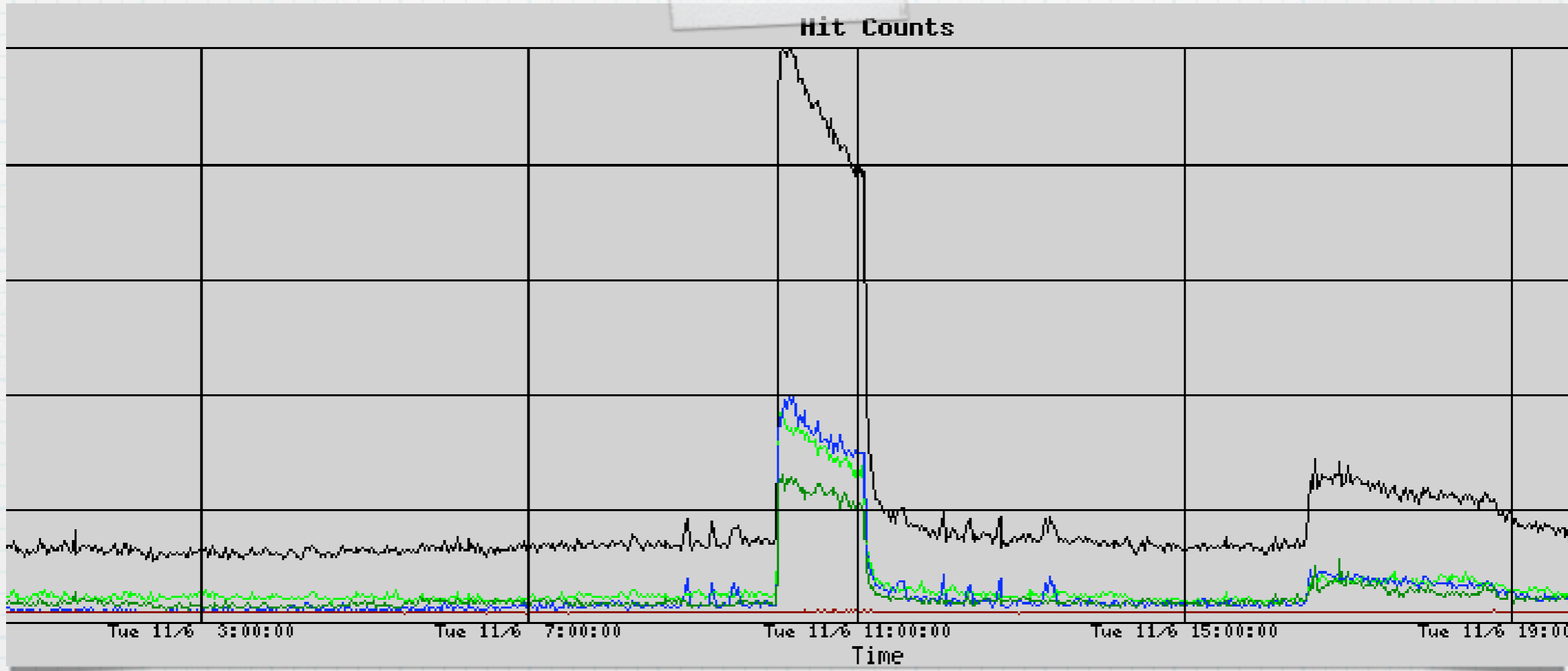
* The Digg Effect



* The Yahoo! Effect



[Redacted]



The Y! Effect

An Aside

Yahoo! Tech—the first SOA website at Yahoo!

About Yahoo! Tech

- * Launched May 1, 2006
- * First new site launched by the Yahoo! Media group in over five years
- * First Yahoo! site to use a complete service-oriented architecture
- * #1/#2 site in the Technology/Gadgets niche

Integrations

- * Content-Management System
- * Partner Articles
- * Tech News
- * Video
- * Tech Blogs
- * Yahoo! Answers
- * UPES
- * Yahoo! Shopping
- * Ratings & Reputation

Some Definitions

- * **SOA—Service-Oriented Architecture.** A structure for delivering content via services (as opposed to via databases or static content).
- * **REST—Representational State Transfer.** An “architectural style” for implementing services using an existing HTTP infrastructure.

What is REST?

- * Resources are represented by URLs
- * Actions use HTTP methods (GET, POST, PUT, DELETE)
- * Some resources:
 - * Representational State Transfer
<http://www.ics.uci.edu/~fielding/pubs/dissertation/>

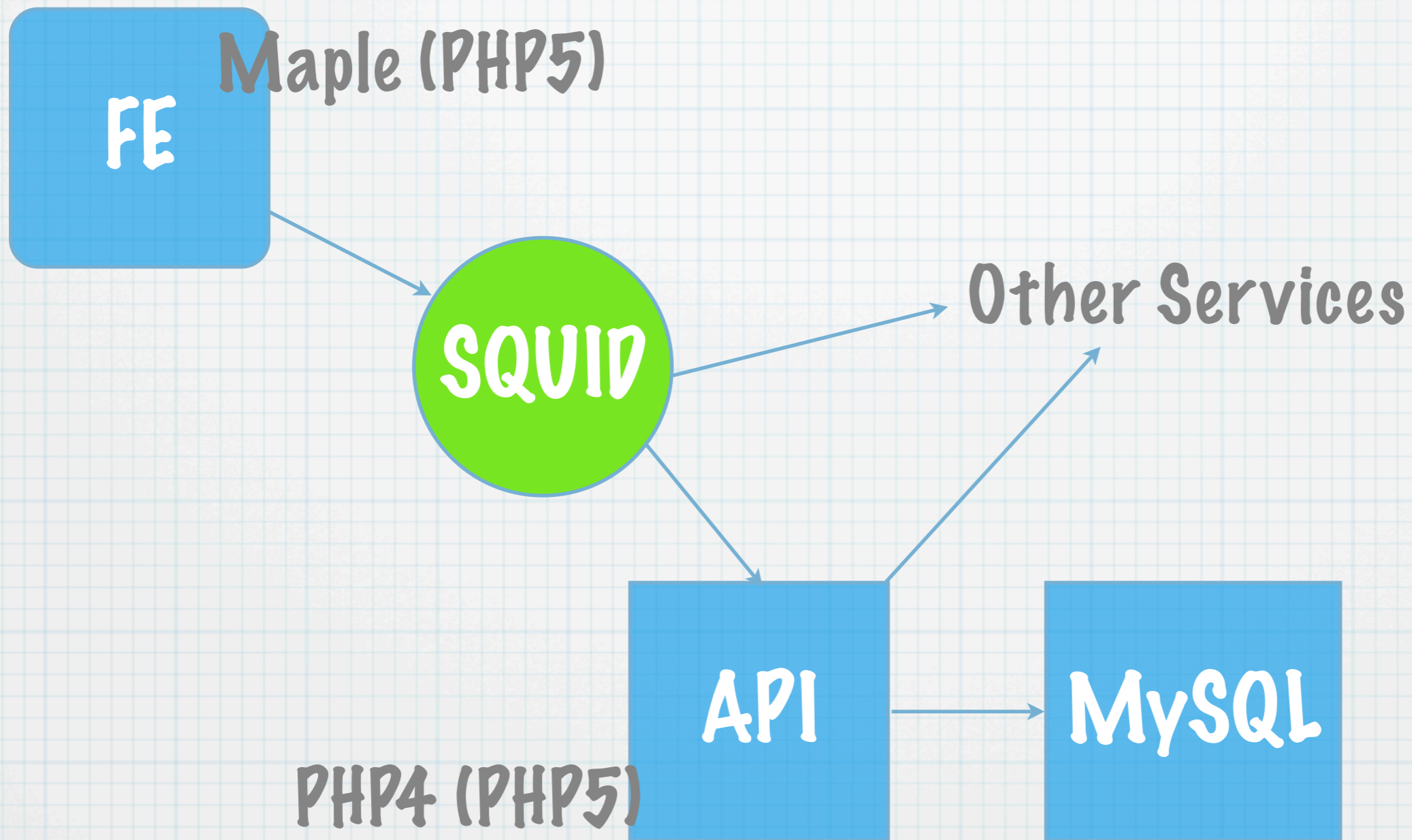
The Data Transfer Stack

- * Time to read 1K of data:
 - * CPU cache = 80ns
 - * PC-3200 DDR RAM = 350ns
 - * SATA hard disk = 17 μ s
 - * 1000Mbit network = 20 μ s (plus latency)

Implications of SOA

- * Content is stored on the network
- * Content is non-local, and therefore slow
- * Content is highly redundant, and therefore cacheable
- * Network optimization
- * Parallelism

Current Architecture



Lessons Learned

Tips and Tricks

Parallelism

- * For services, the latency is in the network; using parallel requests can save time
- * The multi-curl extension to PHP:

```
// create both cURL resources
$ch1 = curl_init();
$ch2 = curl_init();
// set URL and other appropriate options
curl_setopt($ch1, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);
//create the multiple cURL handle
$mh = curl_multi_init();
//add the two handles
```

Parallelism

- * Preceding example is not optimal; still waits until all services are fetched before proceeding
- * Our presentation layer begins rendering a component as soon as the service finishes retrieving data

HTTP Cache

- * REST services are cacheable
- * Using HTTP allows any HTTP-compatible caching intermediary or proxy (e.g., squid)
- * Services should use Expires: **OR** Cache-Control:
headers:
Cache-Control: max-age=600

How to use a caching proxy in PHP (curl)

```
// retries a single URL via CURL

function fetch_curl($url, $timeout=5)
{
    $ch = curl_init($url);

    curl_setopt($ch, CURLOPT_HEADER, 0);

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    curl_setopt($ch, CURLOPT_TIMEOUT, $timeout);

    curl_setopt($ch, CURLOPT_PROXY, '127.0.0.1:3128');
```

How much effect does caching have?

- * Yahoo! Tech, without caching:
0.6 requests/second per server
- * With squid cache:
15.5 requests/second per server
- * 2,400% increase!
- * Or, in hardware terms, 16 servers instead of 400

Squid

- * Open-source, industry-standard HTTP proxy
- * Highly configurable
- * Single-threaded (sigh)
- * Handles ~7,000 requests/second on average hardware

How to make Squid faster

- * Give it more RAM
- * Use `heap LFUDA` cache replacement policy (keeps frequently-accessed objects in cache longer)
- * Give it more RAM
- * Use `refresh_stale_hit`

Other things you can do with Squid

- * Clustering (`cache_peer`) effectively doubles the throughput of the cache (think multi-threaded)
- * Always use timeouts, if possible (never wait for anything)

Network Interface Optimization

- * Um, gigabit anyone?
- * Seriously, use as fast a NIC as you can afford.

PHP-specific

- * Use absolute paths

```
require '/usr/local/lib/php/something.inc';
```

instead of relative

```
require 'something.inc';
```

- * Significant performance improvements

PHP-specific, 2

* **CURL sends `Pragma: no-cache` by default, ensuring that your content is never cached**

* **To avoid this:**

```
curl_setopt($ch,  
            CURLOPT_HTTPHEADER, array('Pragma:'));
```

PHP-specific, 3

- * **Avoid** `open_basedir`
- * **Ok, it's a security risk, but it's also a performance hit**
- * **Weigh the options and decide**

XML performance optimization

- * Never parse an XML document twice
- * Pass a DOM handle or SimpleXML object around

Plan for failure

- * Always specify a timeout
- * Always have behavior planned in case of a timeout
- * Tune your timeouts for optimal effect

Questions and Answers

Speak up! I can't hear you in the back!

More information

- * Download these slides:
<http://files.broadpool.com/dcphp2007/>
- * Email me:
glen.campbell@mac.com
- * Read my blogs:
<http://broadpool.com>
<http://dailyfunnies.org>
<http://suburbanredneck.org>
<http://techbreakfast.org>

Did I mention that
we're hiring PHP
programmers?