

Web Application Clustering with MySQL & Squid

Bård Farstad <bf@ez.no>

Zak Greant <zak@ez.no>

The Us Slide

- Bård Farstad
 - Co-founder of eZ systems
 - Director of Emerging Technology
 - Guitar player
 - Fish wrangler
- Zak Greant
 - Managing Director, eZ systems North America
 - (Bad) Guitar player
 - Has no fish

The You Slide

- What do you do?
 - Developers?
 - Team leaders?
 - DBAs?
 - Managers?
- What size of sites do you run?
 - Tens, hundreds, thousands of concurrent users per minute
 - Mostly read, mixed read/write, high write

We're not in Kansas any more

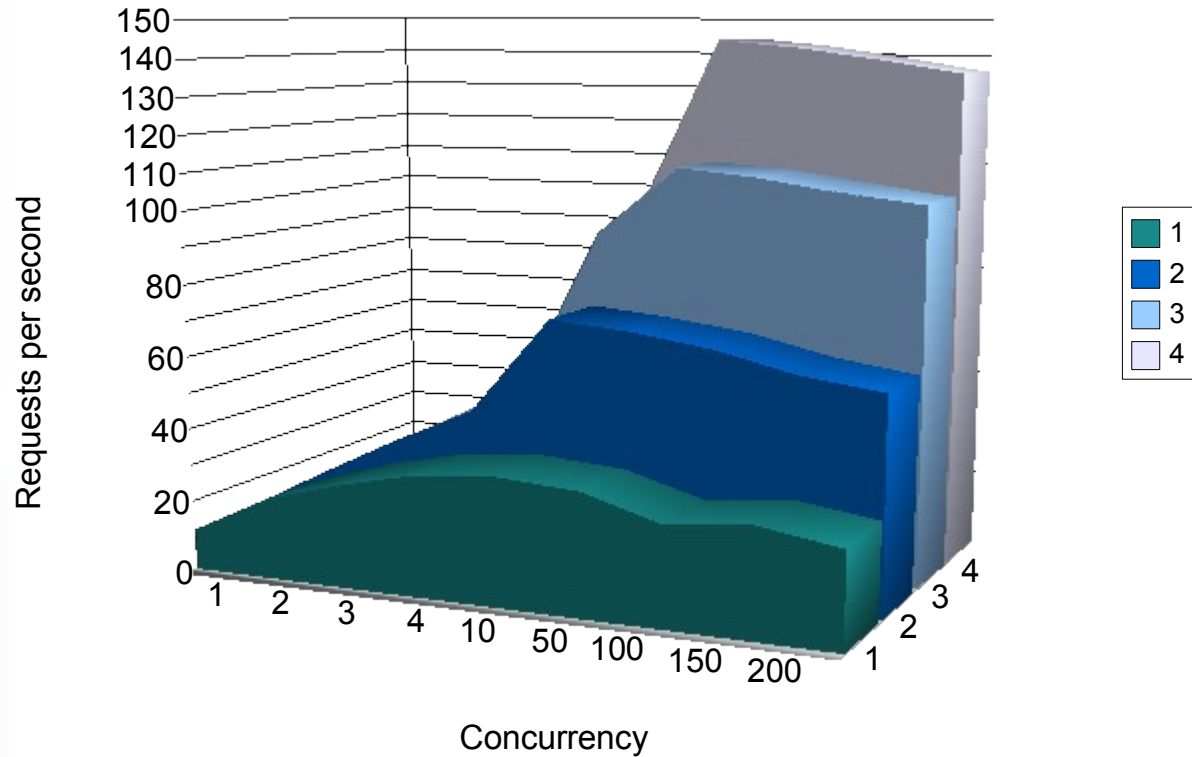
- Mid-sized enterprises
- Corporate intranets (1000+ accts.)
- Popular websites (300k pages/day)
- Processor-intensive sites (personalization, permissions, multi-lingual, forums, ratings, XML processing, etc.)
- Commodity hardware – mid-end hardware

YMMV

- Many ways to scale
- This method works for:
 - High 1000's of concurrent users
 - Mixed read and write
 - Lots of processing (PHP)
 - 1 load balancer
 - 3 web servers (scale here)
 - 1 Squid server
 - 1 master DBMS server
 - 1 failover DBMS server (scale here)

Results from eZ publish

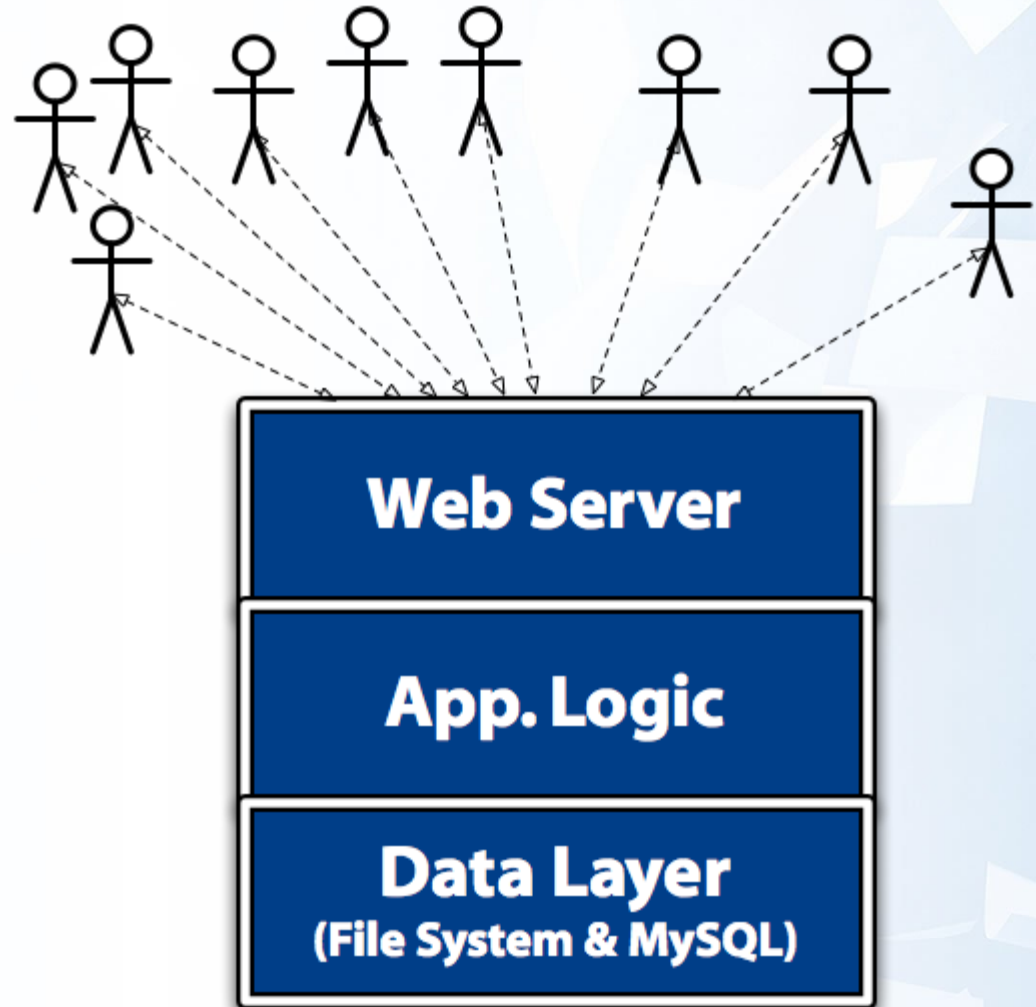
Performance with different setups from
1-4 servers with concurrency of 1-200



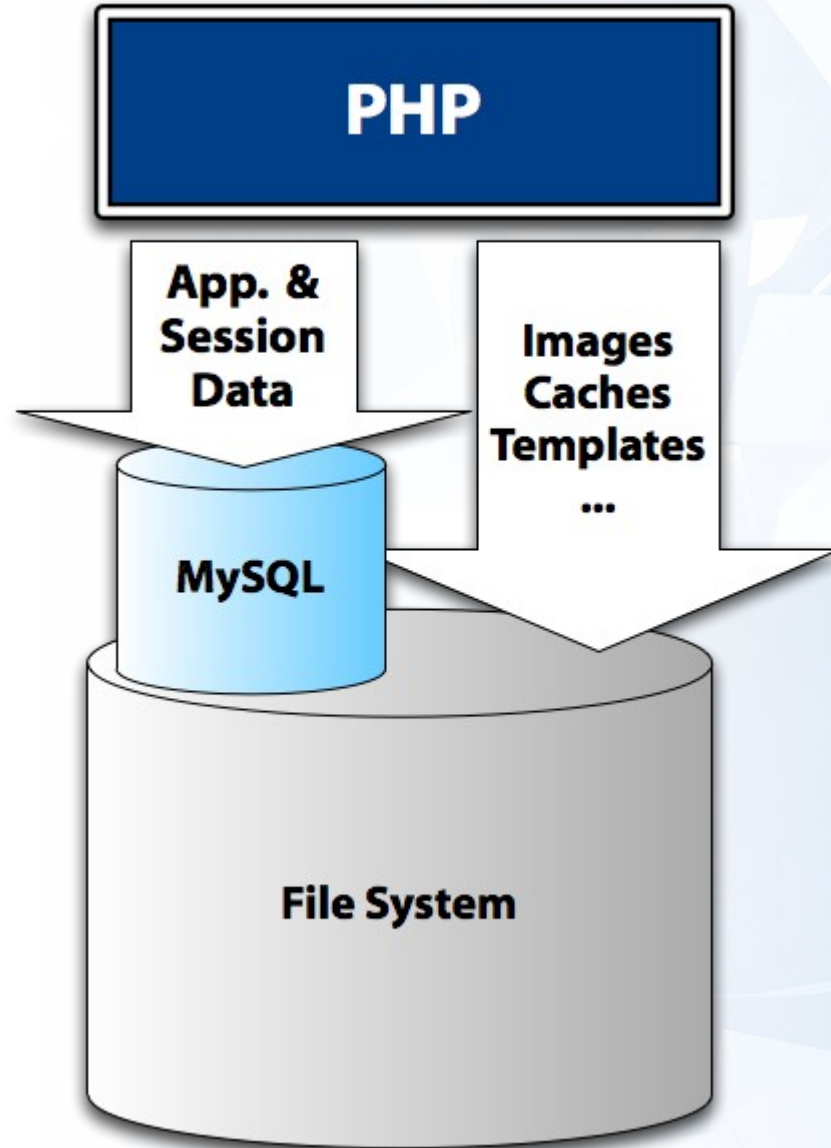
Enables these things

- Near-linear scaling on multiple servers as load grows
- Higher availability through some redundancy
- Excluding some race conditions
- Simplifying administration
- Backup
- Staging and troubleshooting
- Geo-based load balancing (r/o)
- Mostly OS-independent

Typical Single Server Setup



Traditional Storage Architecture



Typical Solutions for Scaling

- Separate web server from DBMS server
- Increase the number of web servers to handle load
- ... but then you need to sync data on file system:
 - rsync
 - NFS or other network file systems
 - Manual syncing
- Works, but..

This works, but ...

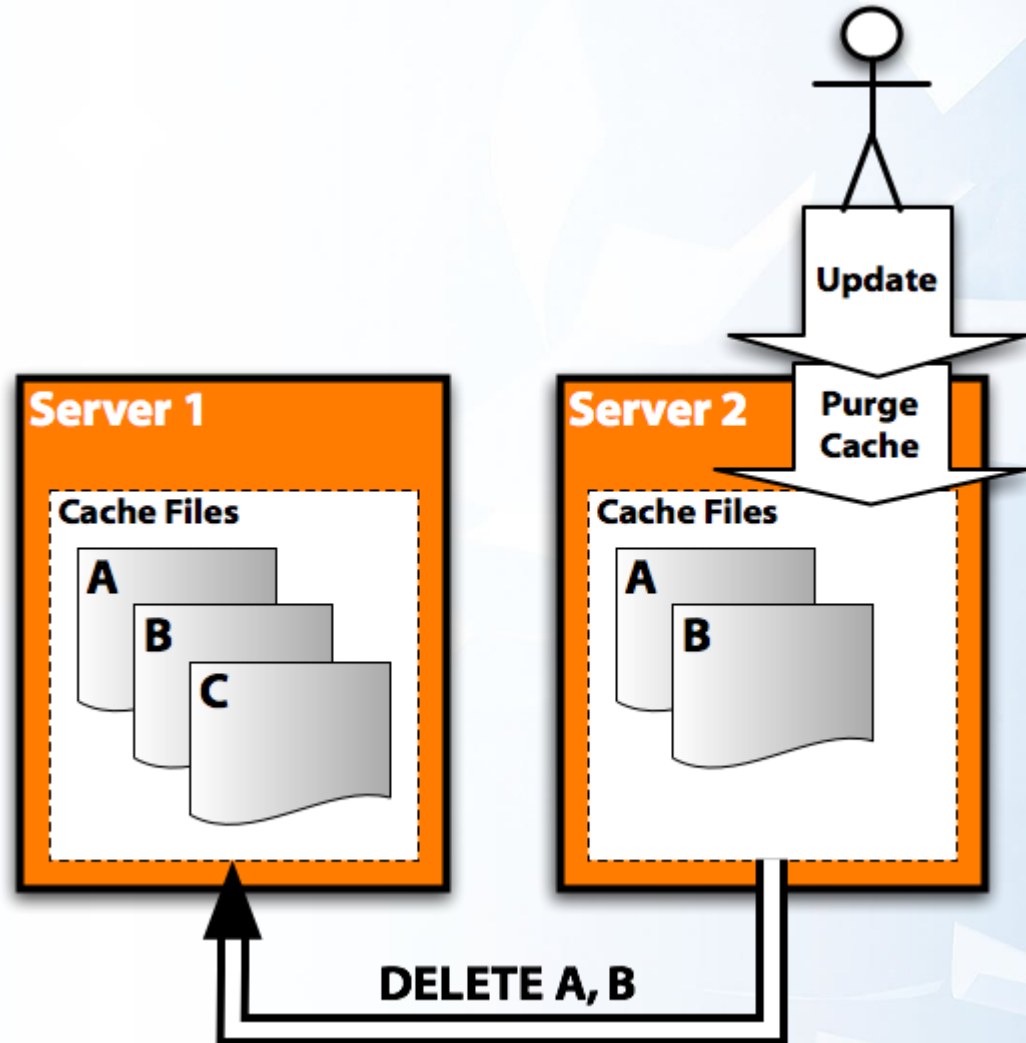
- No canonical data source
- Data stored in MySQL, File System, SQLite, etc.
- Esp. troublesome for application-level caches.
- More complexity = more admin needed = more problems

Web Application Caching

- Partial caching of page content – static and/or transformed



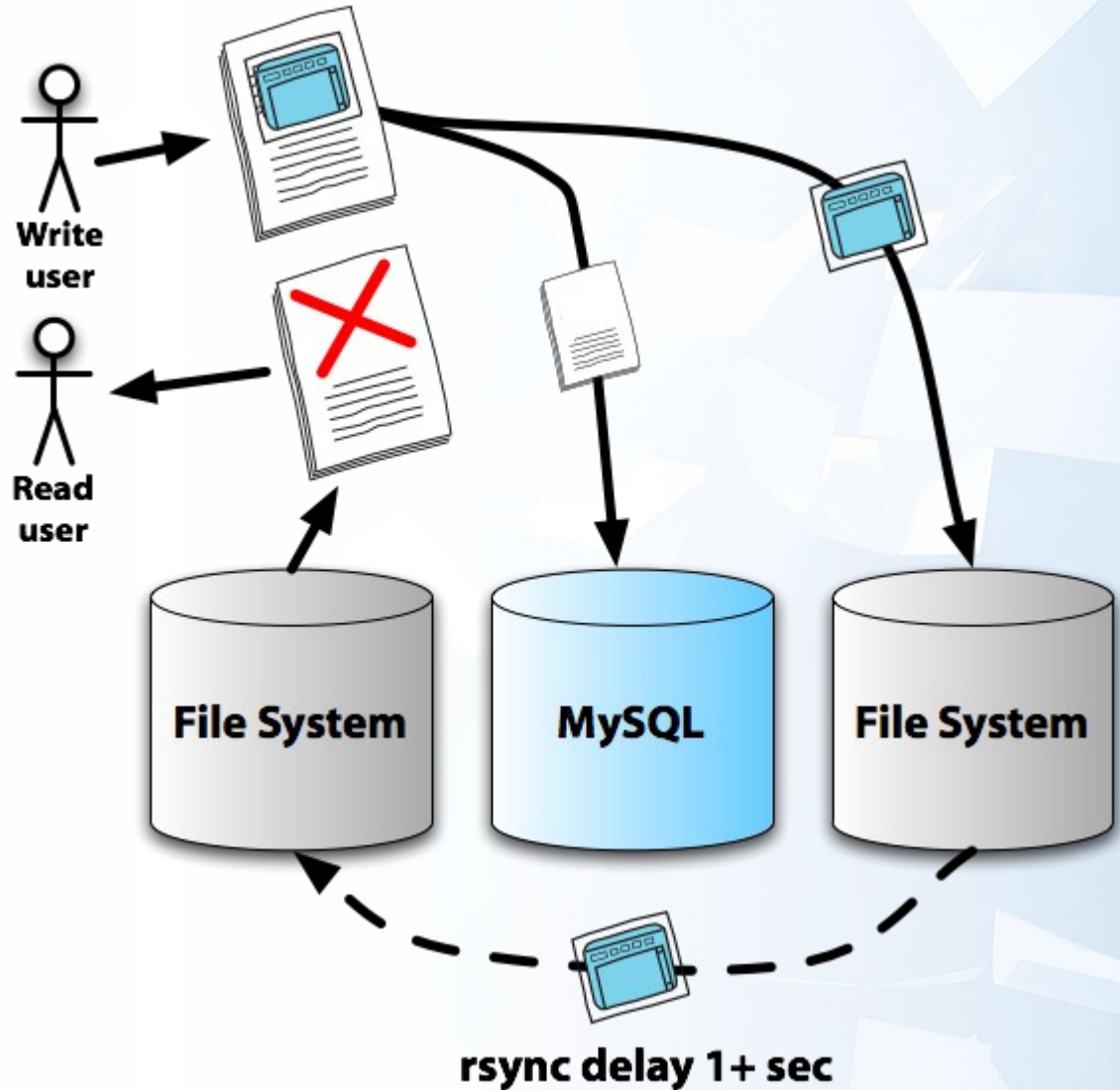
Problem: Cache Corruption



On-demand caching vs. On-create

- On-create cache
 - More storage (sometimes infinite)
 - More CPU at publish time
 - High hit rate
- On-demand cache
 - Less storage, CPU, hit rate
 - Hit rate drops per server in pool
 - No distribution of caching effort
- Note
 - Some cached data is more expensive to share than make.

Synch. Lag and Race Conditions



A Solution

- Canonical storage: MySQL
 - Cache files
 - Images
 - Other stuff
- Use transactions
 - Guaranteed syncing of data
 - No race conditions
- Easy backups
- Cross-platform

Simpler Administration

- Add new node by cloning
- No data stored on application nodes
- No synchronization needed on file system level

Images in database

- Serve via Apache rewrite rules
- Slower than serving from file system
- Possibility to do fine-grained permission control

```
Rewriterule ^/images/.*  
                /index_image.php [L]
```

HTTP headers for caching

- Make Squid cache-able, if public
- Pragma no cache headers for non-public

```
header("Pragma: ");  
header("Cache-Control: ");  
header("Expires: ".  
    gmdate('D, d M Y H:i:s',  
           time() + 600) .  
    ' GMT');
```

Large files in database

- MySQL handles storage of large files (MB not GB)
- (Much) slower than file system
- Use Squid reverse proxy cache for speed
- Alternatively use disk cache on web server
 - Timestamp verification on web server against DBMS
 - Use Apache rewrite again

Cache Files in Database

- Typical size 1-3KB
- MySQL uses query cache
- Not noticeably slower than local filesystem

Table Structure

- Metadata table

```
CREATE TABLE ezdbfile (  
  id          MEDIUMINT(8) UNSIGNED NOT NULL auto_increment,  
  name       VARCHAR(255) NOT NULL DEFAULT '',  
  size       BIGINT(20)    UNSIGNED NOT NULL,  
  mtime      INT(11)       NOT NULL DEFAULT '0',  
  ...  
) ENGINE=InnoDB;
```

- Data table

```
CREATE TABLE ezdbfile_data (  
  id          MEDIUMINT(8) unsigned NOT NULL auto_increment,  
  masterid   MEDIUMINT(8) unsigned NOT NULL default '0',  
  filedata   BLOB NOT NULL,  
  ...  
) ENGINE=InnoDB;
```

- Use chunks if blob size limited by max-packet-size

PHP API: eZClusterFileHandler

- Store file in database:

```
require_once('kernel/classes/ezclusterfilehandler.php');  
  
$fh = eZClusterFileHandler::instance();  
$fh->fileStoreContents('var/foo.txt',  
    $data );
```

- Get data

```
$fh->fileFetch('var/foo.txt');  
$data = $fh->fileFetchContents('var/foo.txt');
```

- Delete data

```
$fh->fileDelete('var/foo.txt');
```

Making Application Cache-able

- Proper headers needs to be sent
- Use unique location for images and files
- Application needs to send Squid purge cache requests

Load balancer

- Distributes request to web servers
- Various distribution schemes
 - Round-robin
 - Load
 - Random
 - ...
- Includes heartbeat monitoring
- We use keepalived

MySQL Replication

- Use one master database for write
- Slave servers to handle read requests
 - Can be used for guaranteed read only requests
- Failover of master database

More information

- ez.no
 - Articles
 - Library
 - Benchmarks

Questions



bf@ez.no
zak@ez.no