DRUPAL BACKEND PERFORMANCE AND SCALABILITY

Ashok Modi – DrupalCampLA 2011

About me

- Systems Analyst at California Institute of the Arts.
- Working with Drupal since 2006.
  - View my profile at http://drupal.org/user/60422
  - View my thoughts at http://btmash.com
- Strong interest in server optimization.

About presentation

- Will reference some of the sites I've worked on
  - Main CalArts site with (too many) modules kept under control.
  - CalArts (Photo Site with about 150k pieces of content)
  - Zimmer Twins (1M content, ½ M users)
- There is a **lot** of material to go through.
  - Avoid talking about software that has successor.
  - May have to speed through some areas (or are you ok with staying around for longer?)
- Doesn't really apply to shared hosting
  - Only a part of this presentation (code related optimizations?) might apply.
- Have a question? Ask!
- Have something to share? Come on up!
  - I am certain there are some very knowledgeable folk in the room…

Resources

- Khalid Baheyeldin
  - http://2bits.com
- Drupal.org infrastructure group
  - http://groups.drupal.org/high-performance
- Peter Zaitsev
  - http://www.mysqlperformanceblog.com/
- Lullabot
  - http://lullabot.com/ideas/blog
- Community ☺

Goals

- Define your objectives and goals first
  - Do you want a faster response to the end user per page?
  - Do you want to handle more page views?
  - Do you want to minimize downtime?
- Each is related…but different
- Gets harder and harder to achieve more performance
  - More infrastructure
  - Patching / Hacking Drupal
  - Revisions to architecture (Fields, Views)

Diagnosis

- Proper diagnosis is essential before proposing and implementing a solution.
- Based on proper data.
- Analysis of collected data.
  - Few possible paths of optimization.

Validation

- Avoid the ‘wild goose chase’.
- Validate results on a test server.
- Replicate the data on a development server.
  - Backup and migrate will help.
  - Migrate can also help.
- Recreate the site.
- Gather a time difference between test and production server.
- Measure again and see if relative times remain the same.

Points of optimization

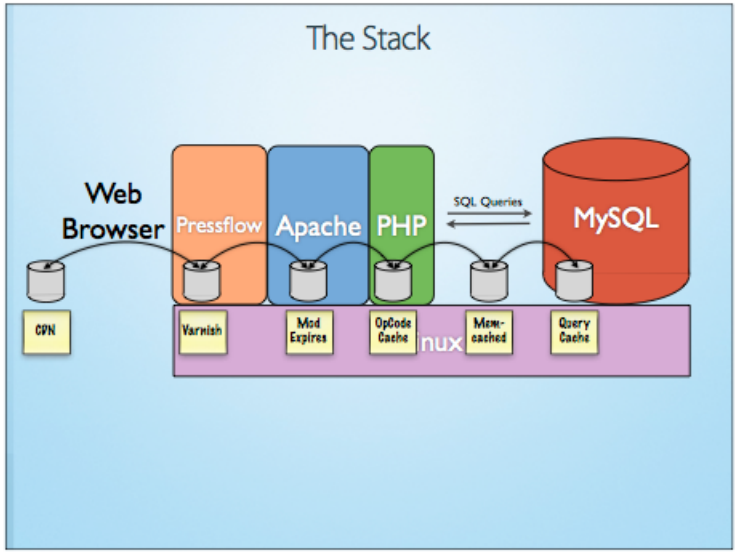
- Introduction.
- Tools to measure and diagnose issues.
- Speed optimizations.

Introduction – Hardware

- Physical server matters
  - Dedicated
  - VPS
  - Cloud-based
    - Anyone here to argue it doesn't work?
- Multiple cores are the norm
  - 32 > 16 > 8 > 4 > 2 > 1
- Lots of RAM (caching the file system and db as much as possible)
- Multiple disks (split of the server to various disks and/or servers)
  - SSD is **much** faster than a reg. HDD.
  - Look into https://github.com/facebook/flashcache
  - FusionIO also looks very promising.
  - Tuning DB on SSD is different from tuning DB on HDD

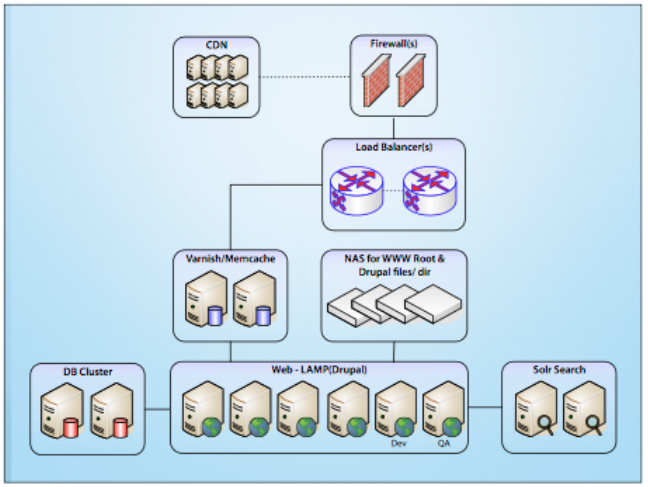
LAMP Stack

- Traditionally most commonly used stack for hosting Drupal and similar applications.
  - **L**inux
  - **A**pache
  - **M**ySQL
  - **P**HP
- A chunk of the presentation will focus on the above.
  - Though we need an acronym involving V(arnish), A(pachesolr), M(emcache), M(ongo), N(ginX)
  - VLAMNMP? Or VAN LAMMP?
- Not discussing Windows.
  - Anyone host Drupal on Windows Server? Would like to know more.



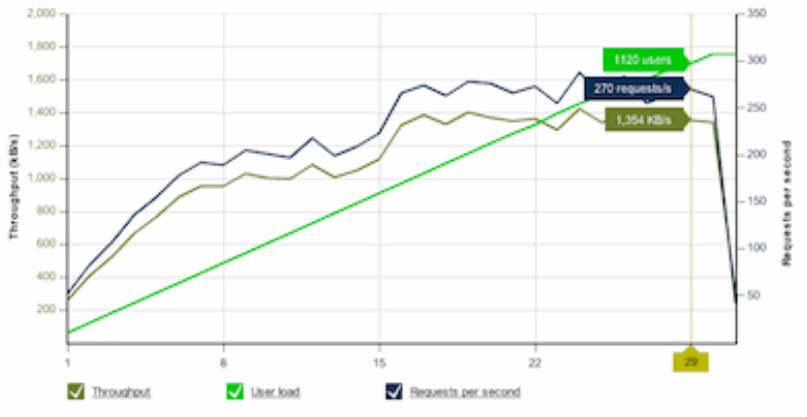
Multiple Servers

- Master DB Server, multiple web servers.
  - Use a load balancer (or something like HAProxy or some other DNS round robin)
  - Set up slave database servers for SELECT queries.
- Do it only if you have the budget and resources.
  - Complexity is expensive (running cost, maintenance time)
  - Tuning a system can avoid or delay a split.
  - A site by 2bits runs on one server.
    - Handles 3+ million hits per day.
    - Read more at http://goo.gl/XueVY



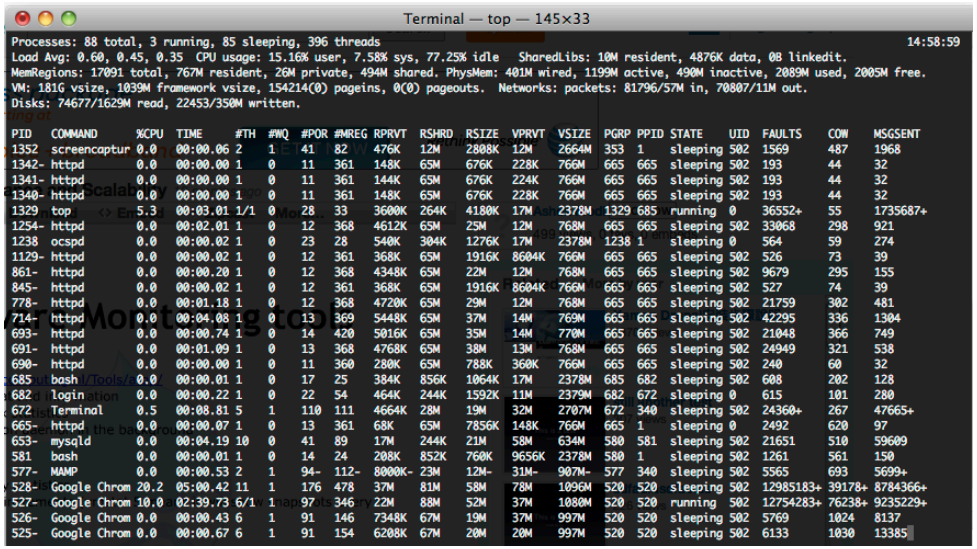
Testing tools

- Apache Benchmark (DEMO)
  - ab –n 100 –c 10 http://www.example.com
  - ab –n 10 –c 10 –C PHPSESSID=<sessid> http://www.example.com
  - Do 10 concurrent requests for up to 100 requests.
  - Average response time per second.
  - How many requests handled per second.
- Jmeter
  - Similar to Apache Benchmark.
  - Can natively use on Windows.
  - Can test POST functionality.
- LoadStorm
  - Web service to test site.
  - Will give pretty graphs in real time.



Console Monitoring tools

- Top
  - Real time monitoring.
  - Load average.
  - CPU utilization.
  - Memory usage.
  - List of processes.



- htop
  - Similar to top but for multiple cores.
  - Faster.

Console Monitoring tools (cont'd)

- atop
  - Shows network statistics.
  - Runs a collection daemon in the background.
- vmstat
  - Report memory statistics
- netstat
  - Shows active network connections
  - netstat –an
  - netstat –an | grep EST

Graphical Monitoring tools

- Cacti
  - http://www.cacti.net
  - Available as a package on Ubuntu, Debian (various other *nix/bsd flavors)
  - Easy to understand graphs.
  - Displays history over day, week, month, year.
  - Graphs available to display stats for CPU, memory, network, Apache, MySQL
  - Many others written by others available online.
- Munin
  - http://munin.projects.linpro.no
  - Very similar to Cacti (doesn't require a db)
  - Can create own monitoring scripts.
- Nagios
  - Heard its very powerful (alerts by email, sms, etc).
  - Drupal module for integration.
- Panopta and New Relic offer hosted monitoring

Linux / BSD

- Use a proven stable distribution (Debian Stable, Ubuntu LTS, RHEL, CentOS)
- Use recent versions
- *Use whatever distro your staff has expertise with
- Try to avoid bloat
  - Don't install PostGreSQL if you are only using MySQL, no desktop server, java, etc.
- Balance compiling own programs versus using packages.
  - Old versions of GD on older versions of Debian, Ubuntu.
- Compiling provides full control.
  - Can be a pain to upgrade.

Apache

- Most popular, supported, and feature rich.
- Stable.
- Can also be enabled with too many unnecessary modules.
  - mod_proxy, cgi_module, etc may be unnecessary.
  - Smaller process = more users can access site
  - apachectl -M – Display all enabled apache modules.
- apachetop
  - Reads and analyses Apache access logs.
  - Good to detect crawlers.
  - ‘apachetop –f /path/to/site/access.log

Apache Optimizations

- MaxClients (prevent swapping / thrashing)
  - Too low – cannot serve enough clients
  - Too high – you run out of memory and you start swapping. Server dies and you cannot serve **any** clients.
- MaxRequestsPerChild
  - Tune it to terminate the process faster and free up memory.
- KeepAlive
  - Keep it enabled.
  - New connects will not get opened constantly.
- mod_gzip/deflate
  - Serve more content quickly.

NginX

- http://nginx.net
- Quite stable.
- More lightweight than Apache.
- Reasonably easy to set up.
  - http://wiki.nginx.org/Drupal for drupal settings ☺
- Seeing some promising results in our dev environment so far…
- Run PHP as FastCGI process.
  - Can also do this with Apache (current method for DrupalCampLA website).
    - Also uses less memory.

Varnish

- HTTP accelerator
- Set it up as a reverse proxy to send the call to Apache if it cannot server something itself
- Serve anonymous page requests and static files
  - D6 core will not get served anonymous – use pressflow.
  - D7 and varnish play nicely.
- Requires some tuning
- Used on Drupal.org
  - Riot Games uses it (@dragonwize might be able to say some more if he's in the room?)
  - Serve millions of pages of content with little impact on server.
- Look at http://drupal.org/project/varnish
- http://goo.gl/8l7gl has some configuration info.
- http://goo.gl/9xQDz has a better explanation ☺

Varnish (cont'd)

- Define IP/port in backend <name> for each web server.
  - Define multiple backends for different servers
  - backend b1 {.host="127.0.0.1"; .port="8082"}
- Use director to group backends for round robin.
- return (pass) // do not cache any checks you make.
  - If (req.url ~ "^.*/ajax/.*$") { return (pass); }
- return (lookup) // lookup in cache or pass it on to backend to get cached.
- unset beresp.http.Set-Cookie; // Unset cookies; What actually allows caching.
- Lullabot's article: http://goo.gl/7JFrP
- Basic setup for D7: http://goo.gl/I7601
  - Tested on own blog…handled about 3k requests per second (couldn't figure out way to break it). Previously handled 30 – 50 requests per second.

MySQL

- Most popular database for Drupal.
- Not necessarily the best database but still does a good job.
- Easy to set up, lots to tune.
  - Less to tune in D6 but D7 requires tuning, even for small sites.
- Various pluggable engines (InnoDB, Archive)
- Forks
  - MariaDB
  - Percona
  - Drizzle
- MySQL 5.5 is a big difference.
  - More to tune.
  - http://goo.gl/hU8tW

MySQL Monitoring

- mtop / mytop
  - Like top but for MySQL
  - Real time monitoring (no history)
  - Shows slow queries and locks.
  - If you have neither – SHOW FULL PROCESSLIST
- Mysqlreport
  - http://hackmysql.com/mysqlreport
  - Reports on server – no recommendations (documentation on site explains a lot)
- Slow Query Log
  - Can be enabled in you're my.cnf file
  - List queries more than N seconds
  - List queries without indexes.
    - Helped identify bottlenecks (one involved a bad count(*) query which I removed)
  - mysql_slow_log_parser script (http://goo.gl/4ZHCT)

MySQL Engines

- MyISAM
  - Fast reads
  - Less overhead
  - Poor concurrency
- InnoDB
  - Transactional
  - Slower in some cases (SELECT COUNT(…))
  - Lots of settings to analyze and change
  - Better concurrency.
- Forks
  - Percona comes with XtraDB (replacement for InnoDB).
  - Maria comes with XtraDB.
  - Both currently looking to be better options than InnoDB (use Google Patches).
  - Same tuning settings.

MySQL Tuning

- There are many (**many**) settings that could be tuned.
- Talk about the ones most likely to give the largest benefits. (D7 Focused)
- innodb_buffer_pool_size
  - Very important.
  - Set **up to** 80% of memory allocated for DB to this.
    - If db is small, memory could be used elsewhere.
- innodb_log_file_size
  - Important for sites with large write workloads
  - 64 – 512M
- innodb_flush_log_at_trx_commit
  - By default, each update transaction flushes the log which is expensive.
  - Set to 0 (write to buffer but no flushes on transaction) or 2 (flush cache instead of disk; both still flush disk/sec) so flushes happen to OS cache.
    - Will lose 1-2 seconds of data if set to 0 if OS crashes. Will have data loss only with full OS crash if set to 2.

MySQL Tuning (cont'd)

- table_cache
  - Opening tables can be expensive.
  - Keeps tables open in cache.
  - 1024 is good place to start.
- thread_cache
  - If you have a lot of quick connections, increase the value.
- query_cache_size
  - Will cache query results.
  - Generally 32M – 512M
- Use mysqlreport to get an idea of what settings to tune.
- Use mysqltuner to help guide you in right direction.
  - http://mysqltuner.pl
  - Read http://mysqlperformanceblog.com

MySQL Replication

- Used on Drupal.org
  - INSERT/DELETE/UPDATE queries go to master
  - SELECT queries go to slave(s)
- Provide noticeable improvements.
- D7 supports replication.
  - For D6, Pressflow is best bet.
- Beware of complexity (connection bet. Master/slave goes down, bad things happen).
- Did extensive tuning on Zimmer Twins
  - Noticeable improvement despite lack of querying slave server.
  - Removed slave server for good.

MongoDB

- Public release in 2009
- Document-oriented
  - 'No-SQL'
  - b.collection.insert|add|update({parameters});
- Retrieve subsets (for certain fields / objects)
- Manages collections of objects in JSON-like format.
- { "username" : "bob", "address" : { "street" : "123 Main Street", "city" : "Springfield", "state" : "NY" } }
- Currently supports up to 64 indexes.
  - 1 for ascending order, -1 for descending order.
  - b.collection.ensureIndex({username: 1, address.state: 1});
- Nested fields can also be indexed.
- Supports master-slave replication. Has automated database sharding.
  - Easily create a cluster.

MongoDB (cont'd)

- Module for Drupal exists!
  - http://drupal.org/project/mongodb
  - Cache, Field Storage, Blocks, Queues, Sessions, Watchdog currently supported.
  - Does a lot of the heavy lifting.
  - Very fast (Examiner.com uses it and has disabled page caching despite high load).
- For anything exported into MongoDB, previous SQL queries will need to be modified so they become mongo queries.
- For entities, use entityfieldquery
  - Anything in this format will actually allow you to switch between SQL and Mongo (and anything else) without changing code.
  - Look at http://drupal.org/project/efq_views for promising work into even more flexible views ☺

PHP

- Use a recent, stable release.
  - D7 requires 5.2.x, as do a few 6.x contributed modules.
  - D8 will require 5.3.x (yes, it's a ways away ☺).
- Install an Op-code cacher / accelerator.
- Useful in bringing down memory usage for a site.
  - eAccelerator
  - APC
  - Xcache
  - Zend optimizer (commercial)
- Anyone try HipHop?

Running PHP

- mod_php
  - Standard php module used by apache.
  - Few issues.
  - Well tested and supported.
  - Can be a resource hog.
- FastCGI (PHP-FPM)
  - Can be used with NginX, Apache.
  - Runs as a separate process.
  - More stable with lower memory usage than mod_php.
  - Trickier to install (lots of good doc. online).

Debugging PHP

- Xdebug
  - http://xdebug.org
  - Display traces on error conditions.
  - Trace functions.
  - Profile PHP Scripts
  - Lots of docs online for installing.
- kCacheGrind
  - Provides a visualization on bottlenecks in code.

Op-code Caching

- Benefits
  - Lowered memory usage.
  - Significant decrease in CPU utilization.
  - Usage on http://www.zimmertwins.com lowered memory usage per process from 20M down to less than 4M
  - Usage on http://calarts.edu lowered memory usage per process from 45M down to less than 10M.
- Drawbacks
  - May crash
  - May require restarts after updating code (apc.stat = 0)

Op-code Caching (cont'd)

- Op-code caching will not work in all circumstances
  - Network connections.
  - Sorting arrays.
  - DB queries
- Bad modules are bad.

Drupal

- Database intensive.
- Can be a resource hog.
- Memory intensive.
  - D7 > D6 > D5
- Site may not be affected by bottleneck.
- Quick Tips
  - Disable unnecessary modules.
    - If performance is such a concern, sites *can* be made without Fields / CCK (ZimmerTwins was such a site)
    - Views UI, Field UI, Rules UI, <module> UI on production.
  - Make sure cron runs regularly.

Drupal Tools

- Devel
  - http://drupal.org/project/devel
  - Total page execution.
  - Query execution time.
  - Query log.
  - Memory utilization.
  - Can be combined with Stress Testing.
- Trace
  - http://drupal.org/project/trace
  - Use for debugging.
  - Traces output, invocations, warnings.
  - Filter by Query Type.

Module calls over network

- Email users (og)
- Call web widgets / APIs (youtube, twitter, facebook)
- Cache as much data as possible.
  - Use helper modules to aid with reducing bog.
  - Queues.
  - SMTP Mail module.

Drupal Caching

- Helpful in not querying / processing same bits of content over and over.
- Especially for anonymous users all of whom may be viewing the same content on your site.
- **Many** caches in core.
  - Bootstrap
  - Block
  - Field
  - Filter
  - Form
  - Image
  - Menu
  - Page (only for anonymous)
- Many from contrib modules (like views, rules, media)

Useful contrib caching modules

- EntityCache
  - http://drupal.org/project/entitycache
  - Caches all core entities (node, user, taxonomy) on entity_load()
  - Stays in cache until expiry or until content is updated/deleted.
- Boost
  - http://drupal.org/project/boost
  - Creates HTML for pages and stores it in files.
  - Requires changes to .htaccess file
  - Does not load up Drupal once content is cached to file for anon. users.
  - Can also use module to display site while in maintenance mode.
  - Varnish has mostly replaced this module (though they could play with each other) on sites not in shared hosting.
- Views content cache
- Block cache alter
- Performance hacks

Pluggable caching

- Use $conf variable in settings.php
  - $conf['cache_backends'][]= ‘./path/to/cache_first_mechanism.inc’;
  - $conf['cache_backends'][]= ‘./path/to/cache_second_mechanism.inc’;
  - $conf[‘cache_class_<bin>’] = ‘CACHECLASS’;
  - $conf[‘cache_default_class’] = ‘SECONDCACHECLASS’;
- Allows you to use a custom caching module.
- Can even use this to completely disable caching (DO NOT USE ON A PRODUCTION SITE!)
- Contrib modules
  - Cache Router (D6)
  - APC (D7)
    - Very fast.
    - Limited to caching on one web server (cache cannot get reused over multiple servers – can be good or bad.)

Memcached

- Distributed object caching in memory.
- Written by danga for LiveJournal
- Lives in memory
- Can span multiple servers.
- Seamless for D6, D7
  - D7 is still undergoing some changes.
- Hash own requirements (apache needs to be restarted, have to clear old caches)
- Slower than APC, but scalable.
- Takes load off DB server (yay!)

Search Mechanism

- Drupal core search
  - Need I say anything?
  - Search API looks to be a much more promising (not to mention flexible!) option.
  - Pluggable system to support various types of backends.
    - Search API MongoDB, Search API Xapian
  - Supports Views.
  - Proposal to include in D8 (core conversation at DC London).
- LuceneAPI
  - Not as fast as ApacheSolr.
  - Easy to setup.
- Google CSE might also be a good fit.

ApacheSolr

- ApacheSolr
  - Very fast.
  - Easy to configure on *nix systems.
    - Requires a server on which Java can be installed.
  - More and more companies offering Solr as a Service (take load off your systems altogether).
  - Available as Drupal Module.
    - Apachesolr (very mature)
    - Search API apachesolr (very promising)
  - Views Plugin so even drive non-search related pages using Solr!

Other options

- Using an optimized distribution
- Pressflow (for D6)
  - http://fourkitchens.com/pressflow-makes-drupal-scale
  - Only supports MySQL
  - Cleanly supports reverse proxies such as Varnish
  - Optimized for PHP5
- Pressflow for D7 is currently identical to D7
  - Talks about abandoning MySQL in favor of a MongoDB/ Cassandra DB architecture.
  - Many of the improvements made for Pressflow are in D7 core.
- Keep in mind that a faster Drupal Core won't save you from contrib modules behaving badly.

Other options (cont'd)

- Patching Drupal / Contrib
  - 'Hack core'
  - Need to know what you are doing.
  - Sometimes necessary.
  - Create a patches directory where all the changes you make for a core/contrib file can be tracked and easily applied on updates.
  - Create own module and do any necessary schema updates / alters from there.

Past experiences from Drupal Core

- User login on zimmertwins.com was painfully slow (5+ seconds per user)
  - Gist of problem: DB not using index on username due to lower()
  - Bug had been around since 2006.
  - Solution: Modified patch on D.O for site.
  - User login time down to 0.1 seconds.
  - Pressflow avoids this by not allowing case-insensitve login.
- Comments did not have index on user ID
  - Created index on user id as an update from my module.
  - If added in future, can remove my version of index.
  - Loading for comment by user no longer an issue.

Advice for developers

- Take advantage of caching.
- Use memory wisely.
  - Unset the variable if you don't have a need for it later.
  - Save variable to memory for future use so processing isn't done multiple times (see drupal_static()).
- Take advantage of AHAH functionality
  - Fewer queries.
  - Not reloading the page.
  - Saving bandwidth.
- Learn to use jQuery (same as above).
- Pay close attention to what

Possibly related sessions

- Note: Some have passed (but check out their screencasts)
- Drupal on the Cloud
  - Patrick Wall
- Drupal Development Q&A
- Drupal 7 Q&A
- Building APIs
  - Adam Gregory
- Professional Staging and Deployment
  - Christefano
- Understanding Ctools
  - Helior Colorado

Questions

- Have a question?
- Want to talk more about performance?
  - Let's talk after ☺
- Think you can help with Drupal performance?
  - http://goo.gl/I3PN2
- Thank you ☺