

Performance Tuning and Optimization for large LAMP sites

Khalid Baheyeldin

Ontario Linux Fest

Toronto

October 13, 2007





Agenda



- Introduction
- The LAMP Stack
 - Linux, Apache, MySQL, PHP
- Drupal
 - Database queries
 - Modules
 - Caching
- Measurement and monitoring tools
- Examples and case studies of what can go wrong?
- Questions, discussion





About 2bits



- Based in Waterloo, Ontario
- Active member of the Drupal community since 2003
- Member of security and infrastructure teams
- 25+ modules on drupal.org
- Listed on Drupal.org's service providers section
- Maintain modules that run on drupal.org (donations, feature, lists, ...)
- Google Summer of Code mentoring (2005, 2006, 2007)





2bits Services



- Clients mainly in the USA and Canada
- Performance tuning and optimization
- Subcontracting development projects
 - Customization of existing modules
 - Development of new modules
- Installation, upgrades
- Automated backups





About Khalid



- Developing for computers for way too long (22 years), Drupal since 2003
- Core contributions
 - Site maintenance feature
 - Logging and alerts in Drupal 6
 - Many patches
- Member of
 - Drupal security team
 - webmasters team
 - infrastructure team
- Co-founder of 2bits
- Blog at <http://baheyeldin.com>
- Contributed modules
 - Adsense
 - Userpoints
 - Nodevote
 - Job search
 - Favorite nodes
 - Flag content
 - Stock API and module
 - Custom Error
 - Currency
 - Image watermark
 - Site menu
 - Email logging and alerts
 - Second Life
 - Technorati
 - Click thru
 - Referral





Definitions



- Performance
 - Can be many things
 - Short response time (milliseconds per page)
 - High throughput (rate of processing work, e.g. page per second)
 - Low utilization of computing resources (CPU, memory, disk, network)
- Performance analysis





Definitions



- Scalability
 - Can the “system” handle more units of work?
 - Horizontal (more servers working in parallel)
 - Vertical (more units of work per server)





Definitions



- High availability
 - Relates to downtime (lack of ...)
 - Uptime of the application exceeds target (measured in 9's, e.g. 5 9s = 99.999)
- Load balancing
 - More than one server
 - Can be for fault tolerance/high availability, or for performance too





Definitions



- Performance Optimization/Tuning
 - Improving system performance from what it is at present
 - Optimal utilization of current resources





Define your goals



- Define your objectives and goals first
 - Do you want faster response to the end user per page?
 - Do you want to handle more page views?
 - Do you want to minimize downtime?
- Each is different, but they can be related
- Everyone wants them **all**, with the cost of **none**!





Hardware



- Physical server matters
 - Dedicated
 - VPS
- Not applicable to shared hosting
- Dual Opterons kick ass
- Lots of RAM (caching the file system and the database, as much as possible)
- Multiple disks if you can
- Always mirrored!





Multiple Servers



- One database server + multiple web servers
- Can use DNS round robin for load leveling
- Or proper load balancers (commercial, free)
- Even a reverse proxy (squid, like drupal.org uses)
- Do it only if you have the budget
 - Complexity is expensive (running cost)
 - Tuning a system can avoid (or delay) the split





The LAMP stack



- Most commonly used stack for hosting Drupal and similar applications
 - Linux
 - Apache
 - MySQL
 - PHP
- Most of this presentation applies to *BSD as well. Parts apply to Windows (anyone use it?).





Linux



- Use a proven stable distro (Debian, Ubuntu)
- Use recent versions (no Fedora Core 4 please)
- Use whatever distro your staff has expertise in
- Be a minimalist, avoid bloat
 - Install only what you need
 - (e.g. No X11, no desktop, No PostgreSQL if you are only using MySQL, no Java, ...etc.)





Linux (cont'd)



- Balance “compile your own” vs. upgrades
- Compile your own
 - Pros: Full control on specific versions
 - Cons: not easy (more work) to do security upgrades
- Using deb/rpm
 - Pros: easy to upgrade security releases, less work
 - Cons: whatever versions your distro has





Apache



- Most popular, supported and feature rich
- Cut the fat
 - Enable only `mod_php` and `mod_rewrite` (as a start)
 - Disable everything else (java, python)
 - May need extended status for Munin
- Tune MaxClients
 - Too low: you can't serve a traffic spike (Digg, Slashdot)
 - Too high: your memory cannot keep up with the load, and you start swapping (server dies!)





Apache (cont'd)



- KeepAlive
 - 5 to 10 seconds.
 - More than that, it ties up processes
- AllowOverrides
 - You can set to None and move Drupal's .htaccess contents to vhosts
 - Less filesystem accesses
- mod_gzip/mod_deflate
 - Compromise of CPU usage vs. Bandwidth usage





Apache Alternatives



- lighttpd (lighty)
 - Popular with Ruby on Rails
 - 1MB per process
 - Recently: reports of really bad memory leaks
- nginx
 - New comer
 - More stable than lighty (no leaks)





Apache Alternatives



- Only run PHP as Fast CGI
- Both lighttpd and nginx run that way
- Separate processes
- Covered later





MySQL



- Most popular database for Drupal
- Not the best database from the technology point of view (ACID, transactions, concurrency), but still adequate for the job
- Various pluggable engines





MySQL Engines



- MyISAM
 - Faster for reads
 - Less overhead
 - Poor concurrency (table level locking)
- InnoDB
 - Transactional
 - Slower in some cases (e.g. `SELECT COUNT(*)`)
 - Better concurrency
 - Oracle owns the engine now ...





MySQL Engines



- Two new engines, owned by MySQL AB
 - Falcon. Not mature enough to match InnoDB, benchmarks show it is still slow, but promising
 - SolidDB.
- PBXT
 - PrimeBase XT





MySQL tuning



- Query cache
 - Probably the most important thing to tune
- Table cache
 - Also important
- Key buffer





MySQL replication



- Master/Slave implementation
- Now in use on drupal.org
 - INSERT/UPDATE/DELETE go to the master
 - SELECTs go the slave
- Noticable improvement
- Patch here <http://drupal.org/node/147160>





PHP



- Use a recent version
- Install an Op-code cache / Accelerator
 - eAccelerator
 - APC
 - Xcache
 - Zend (commerical)





Op-code caches



- Benefits
 - Dramatic speed up of applications, specially complex ones like Drupal
 - Significant decrease in CPU utilization
 - Considerable decrease in memory utilization
 - The biggest impact on a busy site
- Drawbacks
 - May crash often
 - Use logwatcher to auto restart Apache





Op-code caches (cont'd) **2bits**

- Findings
 - eAccelerator uses the least memory and provides the most speed
 - Barely maintained (start to lag behind)
 - APC recent versions are more stable
- APC vs. eAccelerator benchmark on 2bits
- Find the right combination for your setup
 - PHP version
 - op-code cache version





APC admin



- Refresh Data
- View Host Stats**
- System Cache Entries
- Per-Directory Entries
- User Cache Entries
- Version Check
- Clear opcode Cache

General Cache Information

APC Version	3.0.14
PHP Version	5.1.6
APC Host	adm.adsoftheworld.com
Server Software	Apache/2.0.55 (Ubuntu) PHP/5.1.6
Cached Files	222 (20.0 MBytes)
Cached Variables	0 (0.0 Bytes)
Hits	154934215
Misses	689
Request Rate	357.58 cache requests/second
Time To Live	0
Shared Memory	1 Segment(s) with 30.0 MBytes
Cache full count	0
Start Time	2007/08/19 06:32:30
Uptime	5 days, 21 minutes

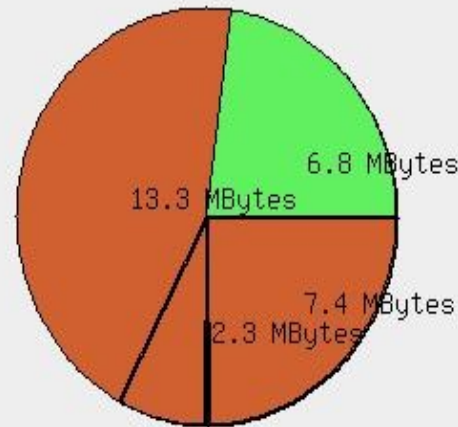
Runtime Settings

apc.cache_by_default	1
apc.enable_cli	0
apc.enabled	1
apc.file_update_protection	2
apc.filters	
apc.gc_ttl	3600

Host Status Diagrams

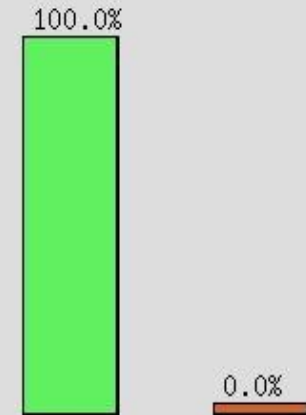
Memory Usage

(multiple slices indicate fragments)



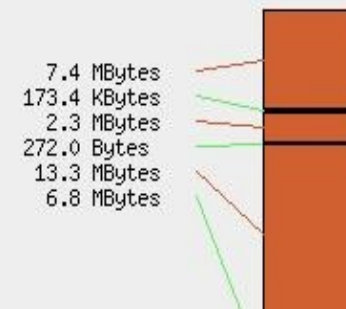
- Free: 7.0 MBytes (23.2%)
- Used: 23.0 MBytes (76.8%)

Hits & Misses



- Hits: 154934215 (100.0%)
- Misses: 689 (0.0%)

Detailed Memory Usage and Fragmentation





mod_php



- Normally, Apache mod_php is the most commonly used configuration
- Shared nothing
 - No state retained between requests
 - Less issues
 - Most tested and supported
- Stay with mod_php if you can.
- Can be as low as 10-12MB per process
- Saw it as high as mid 20s+ (but depends on modules installed)





PHP as CGI



- CGI is the oldest method from the early 90s.
- Forks a process for each request, and hence very inefficient.
- Some hosts offer it by default (security) or as an option (e.g. running a specific PHP version).
- Don't use it!





Fast CGI



- FCGI is faster than CGI (uses a socket to the PHP process, not forking)
- Mostly with Lighttpd and nginx, since it is the only way to run PHP for those servers, but also with Apache
- There are some cases (e.g. drupal.org itself)
- Better separation of permissions (e.g. Shared hosting)
- If you have one server and one Linux user, permissions may not be an issue.





Other ways for PHP



- Non-Zend
- Roadsend PHP compiler
 - Compiles PHP to native code!
 - Source is available, requires Scheme to build
 - <http://code.roadsend.com/pcc>
- PHC
 - Not yet complete, but has a Parrot spinoff
 - <http://www.phpcompiler.org/>





Other ways for PHP



- Caucho Quercus
 - Implementation of PHP written in Java!
 - Benchmarks say it is as fast as PHP with an op-code cache
 - <http://quercus.caucho.com/>





Drupal



- Mainly database bound
- Can be CPU bound (certain modules, resource starved hosts, ...)
- Bottlenecks are worked on as they are found by the community
- Some modules known to be slow (more on it)
- Not all sites affected by all bottlenecks





Watchdog



- Avoid errors (404s on graphics, favicon)

```
TIME STATE      INFO
24  updating  DELETE FROM watchdog WHERE timestamp < 1176392718
24  Locked    INSERT INTO watchdog (uid, type, message, severit
19  Locked    INSERT INTO watchdog (uid, type, message, severit
14  Locked    INSERT INTO watchdog (uid, type, message, severit
11  Locked    INSERT INTO watchdog (uid, type, message, severit
6   Locked    INSERT INTO watchdog (uid, type, message, severit
```

- Optional in Drupal 6 (syslog as an option)





Sessions



- Heavily used in high traffic sites

TIME	STATE	INFO
28	Locked	UPDATE sessions SET uid = 0, hostname = '212.154.
28	Copying to t	SELECT ... FROM sessions WHERE timestamp >= 11776
28	Locked	SELECT ... FROM users u INNER JOIN sessions s ON
27	Locked	UPDATE sessions SET uid = 0, hostname = '222.124.
27	Locked	UPDATE sessions SET uid = 0, hostname = '201.230.
27	Locked	SELECT ... FROM users u INNER JOIN sessions s ON
27	Locked	SELECT ... FROM users u INNER JOIN sessions s ON
27	Locked	SELECT ... FROM users u INNER JOIN sessions s ON
27	Locked	SELECT ... FROM users u INNER JOIN sessions s ON
27	Locked	SELECT ... FROM users u INNER JOIN sessions s ON





Sessions (cont'd)



- New patch in Drupal 6 limits writes to one per X minutes, not for each access





Drupal (cont'd)



- Disable modules that you do not need.
- Make sure cron runs regularly
- Enable throttle
 - Be wary about throttle and cache





Media files



- Large video and audio ties up resources for a long time
- Specially to slow connections, or unstable ones (users try to download again and again)
- Serve them from a separate box
 - <http://example.com> for PHP
 - <http://media.example.com> for video/audio
 - Video modules already supports this (but you have to manually FTP the videos)
- Use a content delivery network (CDN) e.g. Akamai.





Drupal caching



- For anonymous visitors only
- Does not affect authenticated users
- Enable page caching
 - May expire too often on a busy site, causing slow downs!
 - Set the cache expiry minimum (Drupal 5 and later)
- Aggressive caching can have some implications, but gives better performance





Drupal caching (cont'd)



- Certain parts of cache are always on and cannot be turned off (but see later)
 - Filter
 - menu
 - variables
- Filter cache
 - We turned it off on a busy site
 - Patch or pluggable cache





Drupal caching (cont'd)



- If you use Squid as a cache, then those may not apply
- Consider other caching modules that use files
 - FS Fastpath
 - Still some of Drupal's PHP is executed
 - Boost
 - Uses rewrite rules, so better performance
 - A Drupal 5.x version came out a week or so ago
 - File Cache
 - Useful for shared hosting
 - Uses flat files to store the cached objects outside the DB





Pluggable caching



- Using \$conf variable in settings.php
 - 'cache_include' => './includes/yourcache.inc'
- Allows you to have a custom caching module
- Developers tip: can be used to disable cache for development (stub functions that do nothing)





Block caching



- Contrib module for Drupal 5.x
- Now in core for Drupal 6
- Eliminates the overhead of generating blocks for each page view
- 64% improvement (Drupal 6)





memcached



- Distributed object caching in memory
- Written by Danga for livejournal
- No disk I/O (database or files)
- Can span multiple servers (over a LAN)
- Give it a lot of RAM
- Uses Drupal pluggable caching
- Requires patches and schema changes for Drupal 5





memcached (con't)



- Status

127.0.0.1:11211

pid	4827
uptime	524688
time	1187952839
version	1.1.12
rusage_user	279.360000
rusage_system	804.110000
curr_items	19057
total_items	1515820
bytes	143970240





memcached (con't)



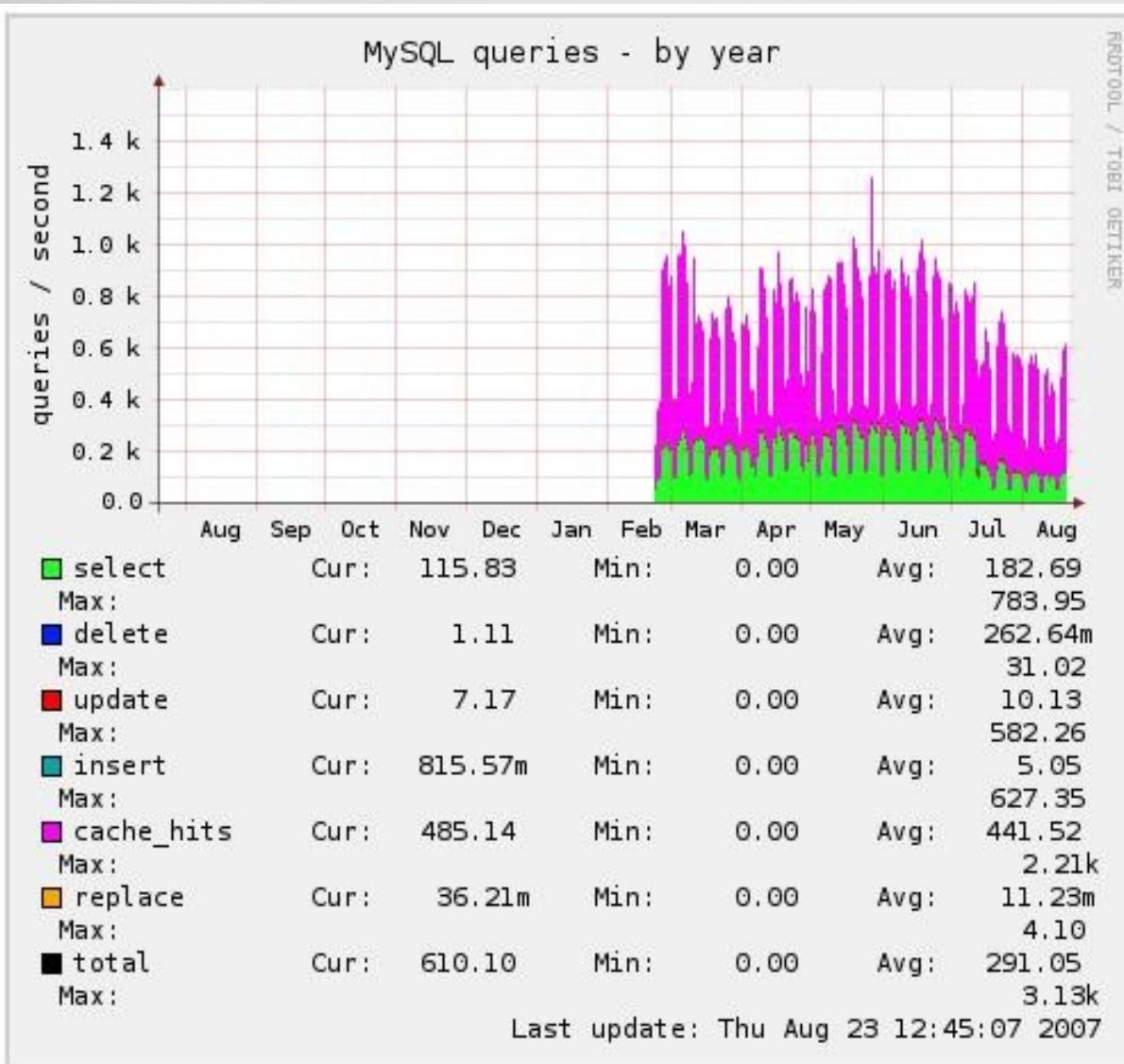
- Status (cont'd)

curr_connections	76
total_connections	87873
cmd_get	10711779
cmd_set	1515820
get_hits	9179288
get_misses	1532491
bytes_read	16696949055
bytes_written	265519514970
limit_maxbytes	201326592





memcached (cont'd)



- How much of an effect does memcache has?
- See how many SELECTs were reduced in early July compared to earlier month!





memcached (cont'd)



- Watch out for:
 - Must start Apache after memcached restart
- Also:
 - Gets complex as you add instances
 - Gets more complex as you add instances on other servers





Advanced Caching



- Contributed module, set of patches
- For authenticated users
 - block_cache
 - comment_cache
 - node_cache
 - path_cache
 - search_cache
 - taxonomy_cache





Slow modules



- Statistics module
 - Adds extra queries
 - Even slower on InnoDB (COUNT(*) slow)
 - Disable Popular Content block
- gsitemap (XML sitemap)
 - Had an extra join, patch accepted
 - Can't handle more than 50,000 nodes
 - Exhausted memory
 - New version rewritten to use a flat file





Slow modules (cont'd)



- Aggregator2
 - Uses body field (text) to store an ID
 - Joins on it
 - Abandoned!
- Many more ...





Drupal.org slowdowns



- Long running pain
- In a nutshell
 - Use InnoDB instead of MyISAM (mixed results)
 - Use Squid reverse proxy
 - Forum: Remove next/previous forum topic
 - Forum: block caching patch for Recent forum block
 - Tracker: use UNION query
 - Use master/slave patch for query (big difference)
- Details here <http://drupal.org/node/163216>





Measure and Monitor



- How do you know you have a problem?
 - Wait till users complain (site is sluggish, timeouts)?
 - Wait till you lose audience? Loss of interest from visitors?
- Different tools for various tasks





Top



- Classic UNIX/Linux program
- Real time monitoring (i.e. What the system is doing NOW, not yesterday)
- Load average
- CPU utilization (user, system, nice, idle, wait I/O)
- Memory utilization
- List of processes, sorted, with CPU and memory
- Can change order of sorting, as well as time interval, and many other things





htop



- Similar to top
- Multiprocessor (individual cores)
- Fancy colors





vmstat



- From BSD/Linux
- Shows aggregate for the system (no individual processes)
- Shows snapshot or incremental
- Processes in the run queue and blocked
- Swapping
- CPU user, system, idle and io wait
- First line is average since last reboot





netstat



- Shows active network connections (all and ESTABLISHED)
- `netstat -anp`
- `netstat -anp | grep EST`
- Remember that delivering content to dialup users can be slow, because the other end is slow





apachetop



- Reads and analyses Apache's access log
- Shows all/recent hits
 - Request per second, KB/sec, KB/req
 - 2xx, 3xx, 4xx, 5xx
- List of requests being served
- To run it use:
 - `apachetop -f /var/log/access.log`





mtop, mytop



- mtop / mytop
 - Like top, but for MySQL
 - Real time monitoring (no history)
 - Shows slow queries and locks
- If you have neither
 - SHOW FULL PROCESS LIST
 - mysqladmin processlist
 - run from cron?





mysqlreport / db tuning



- Mysqlreport
 - Perl shell script
 - Displays statistics
 - No recommendations
- Db tuning
 - A shell script that reads variables from MySQL
 - Annoying use of colors
 - Useful recommendations





Slow Query Log



- Has to be enabled in my.cnf
- Lists queries taking more than N seconds
- Very useful to identify bottlenecks
- Best way to interpret it:
 - Use `mysql_slow_log_parser` script





Stress testing



- How much requests per second can your site handle?
- Are you ready for a digg?
- Do you know your performance and bottlenecks before you deploy? or after?
- The challenge is finding a realistic workload and simulating it
- If you find bottlenecks, submit patches





Stress testing (cont'd)



- ab/ab2 (Apache benchmark)
 - `ab -c 50 -n10000 http://example.com`
 - Requests per second
 - Average response time per request
 - Use `-C` for authenticated sessions
 - <http://httpd.apache.org/docs/2.0/programs/ab.html>





Stress testing (cont'd)



- Siege
 - Another HTTP Server load test tool
 - <http://www.joedog.org/JoeDog/Siege>
- Jmeter
 - Written in Java
 - Desktop
 - <http://jakarta.apache.org/jmeter/>





Graphical monitoring



•Munin

- Nice easy to understand graphs.
- History over a day, week, month and year
- CPU, memory, network, Apache, MySQL, and much more
- Can add your own monitoring scripts

•Cacti

- Similar features





Web site statistics



- Definitions

- Hits (every page, graphic, video, css, js file)
- Page views (e.g. a node, a taxonomy list)
- Visits
- Unique visits (advertisers care about this)





Web site statistics



•Awstats

- Free perl script to analyse your Apache logs
- Measures humans as well as crawlers
- Measures hits, page views, operating system, browser, search engines, referring sites, and more

•Google Analytics

- Measures humans only (javascript)
- Focus on visits, new visitors
- Map overly, and much more





Drupal tools



- Devel module
 - Total page execution
 - Query execution time
 - Query log
 - Memory utilization
- Trace module
 - More for debugging, but also useful in knowing what goes on under the hood





Drupal tools (cont'd)



- Loadtest module
 - Google Summer of Code 2007
 - Load testing of Drupal
 - Measures timings for discrete components
 - Need to write simpletest-like tests
 - Has a project page on drupal.org





What can go wrong?



- CPU usage is too high
- Memory over utilization
- Too much disk I/O
- Too much network traffic





CPU



- Find out who is using the CPU?
- Find out which type (user, system, wait I/O)





CPU



- If it is an Apache process, the op-code cache will help (APC, eAccelerator), unless you have a bad module.
- If it is MySQL, then some of that is normal (intensive queries. lots of queries), otherwise
 - tune the indexes (OPTIMIZE TABLE ...)
 - split the server to two boxes (web and db).
 - Tune the query cache
- If it is something else, and consistent, then consider removing it.





CPU 100%



- Output from Top

top - 10:16:58 up 75 days, 59 min, 3 users, **load average: 152.70, 87.20, 46.98**

Tasks: **239 total, 157 running**, 81 sleeping, 0 stopped, 1 zombie

Cpu(s):100.0%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st

Mem: 2075932k total, 1558016k used, 517916k free, 13212k buffers

Swap: 1574360k total, 49672k used, 1524688k free, 442868k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
659	www-data	21	0	61948	14m	4060	R	3	0.7	0:14.35	apache2
960	www-data	20	0	62084	14m	4076	R	3	0.7	0:10.51	apache2
989	www-data	20	0	62036	14m	4052	R	3	0.7	0:09.95	apache2

.... hundreds of them





CPU 100%



- Vmstat output

```
# vmstat 15
```

```
procs -----memory-----      ----cpu----
 r  b   swpd   free   buff  cache   us   sy  id  wa
152  0   40868 1190640 13740 465004  22   6  71  2
153  0   40868 1190268 13748 464996 100  0  0  0
155  0   40868 1189740 13756 464988 100  0  0  0
154  0   40868 1189540 13768 465044 100  0  0  0
```

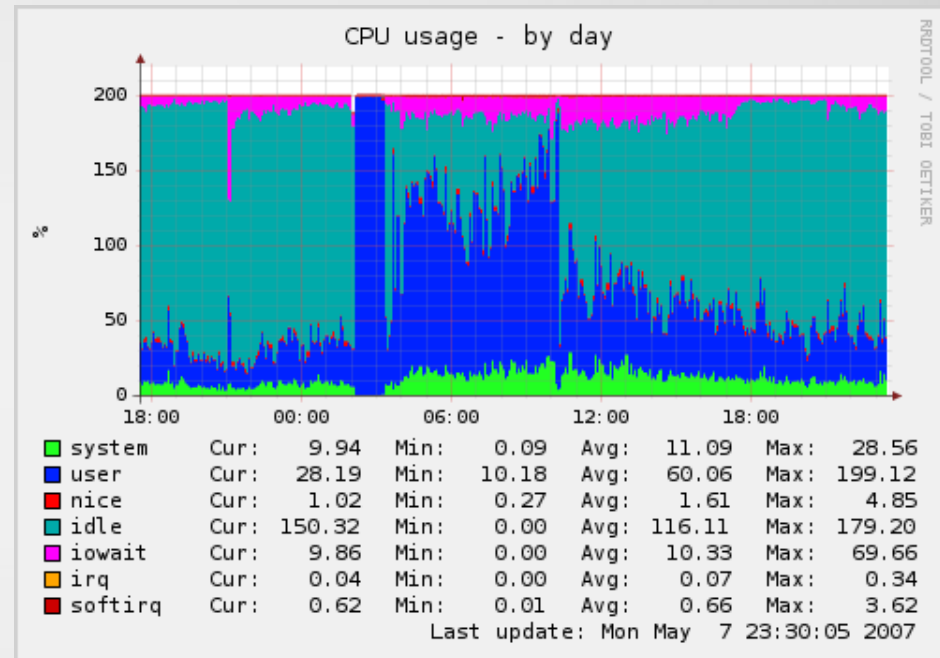




CPU 100%



- What was it?
- eAccelerator (svn303 + PHP 5)
- Attempt to get over PHP crashes
- Note CPU utilization (100%, then high, then dropped low when good version used)





Memory



- Swapping means you don't have enough RAM
- Excessive swapping (thrashing) is server hell!
- Reduce the size of Apache processes (no SVN DAV)
- Reduce the number of Apache processes (MaxClients)
- Turn off processes that are not used (e.g. Java, extra copies of email servers, other databases)
- Buy more memory! Cost effective and worth it.

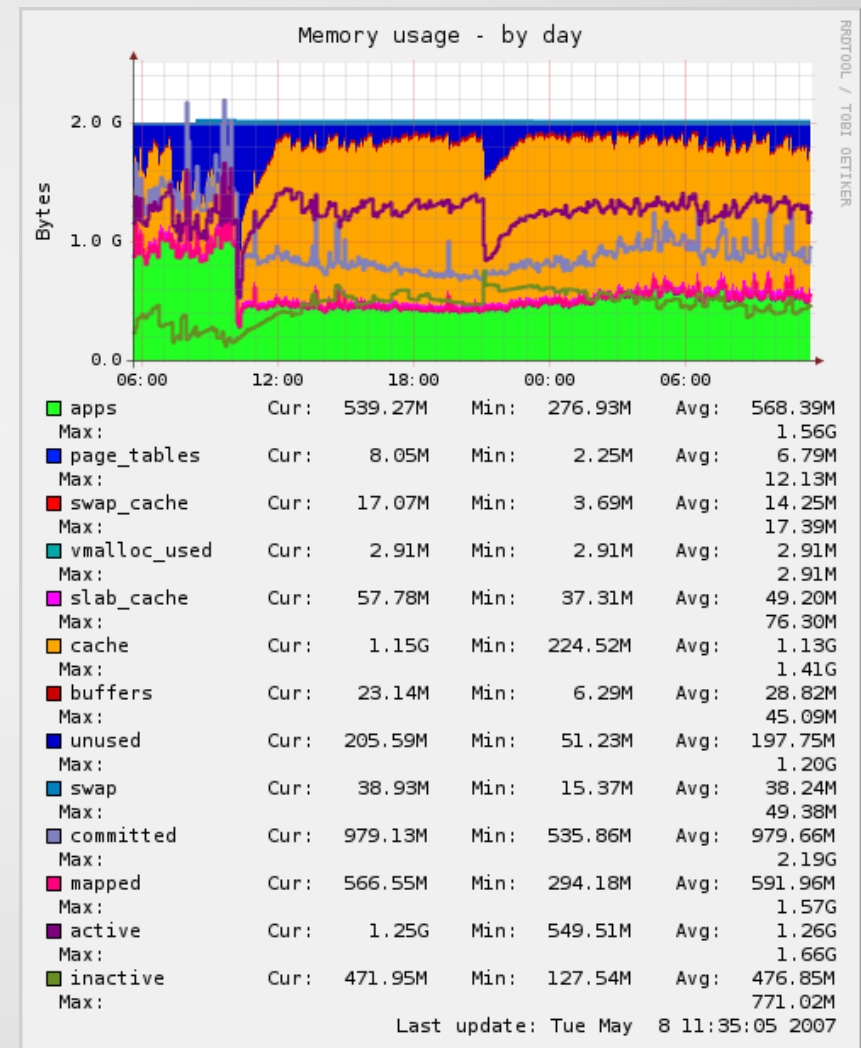




Memory



- Impact on memory usage when there is no op-code cache vs. with an op-code cache (eAccelerator in this case)





Disk I/O



- First eliminate swapping if get hit by it.
- Get the fastest disks you can. 7200 RPM at a minimum.
- Turn off PHP error logging to `/var/log/*/error.log`
- Consider disabling watchdog module in favor of syslog (Drupal 6 will have that option), or hack the code
- Optimize MySQL once a week, or once a day





Network



- Normally not an issue, but make sure you have enough bandwidth
- Occasionally you will have stubborn crawlers though
- Or even a DDoS
- Or worse, extortion
- Can eat up resources, including network





Digg front page?



- On Good Friday, a web site we manage was on Digg's front page.
- Survived the digg well.
- Another server (untuned) got digged twice and died





Resources and Links



- General

- <http://2bits.com/articles/drupal-performance-tuning-and-optimization-for-large-web-sites.html>
- http://www.lullabot.com/articles/performance_and_scalability_seminar_slides

- Apache

- <http://httpd.apache.org/docs/2.0/misc/perf-tuning.html>

- MySQL

- <http://www.mysqlperformanceblog.com/>
- <http://dev.civicaactions.net/moin/CodeSprint/SanFransiscoMarch2007/PerformanceAndScalabilitySeminar>





Conclusion



- Questions?
- Comments?
- Discussions?

