Drupal Back End Performance Optimization for large web sites

Khalid Baheyeldin

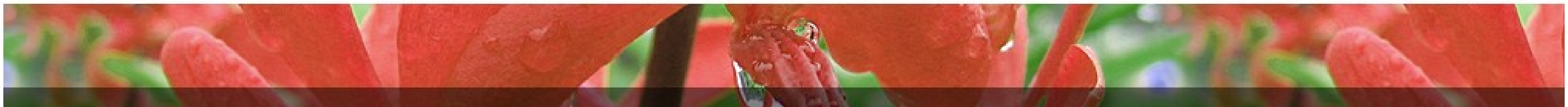
March 5, 2009
DrupalCon Washington, DC

2bits

About Khalid

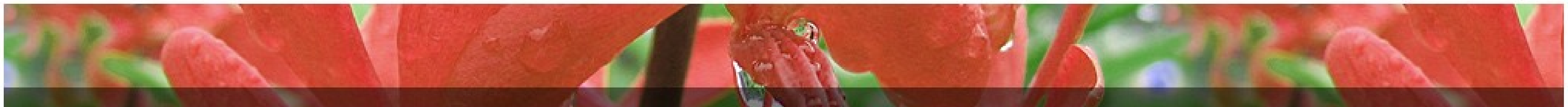
- Software development and consulting for 24 years
- Drupal addict since 2003
- Core contributions
  - Site off-line maintenance feature
  - Logging and alerts (syslog)
  - Reverse Proxy
  - Other patches ...
- Member of
  - Drupal Association (General Assembly)
  - security/infrastructure teams
- Co-founder of 2bits.com, Inc.
- Blog at http://baheyeldin.com

- Contributed modules (37+?)
  - Userpoints
  - Nagios
  - Second Life
  - Adsense
  - Job search
  - Favorite nodes
  - Flag content
  - Nudge
  - Stock API and module
  - Currency API and module
  - Custom Error
  - Image watermark
  - Site menu
  - Email logging and alerts
  - Technorati
  - Referral
  - Nodevote



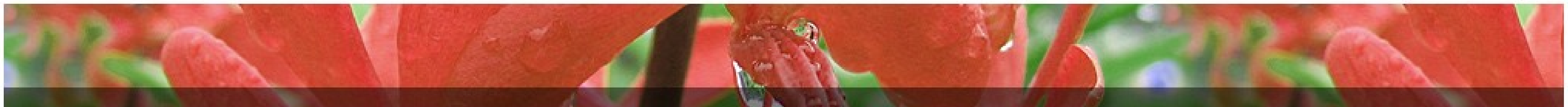
About 2bits.com

- Founded 1999
- Based in Waterloo, Ontario (Canada)
- Active member of the Drupal community since 2003
- 37+ contributed modules on drupal.org
- Listed on Drupal.org's service providers section
- Maintain modules that run on drupal.org (donations, feature, lists, fee, ...)
- Event sponsorship (DrupalCon, DrupalCamps)



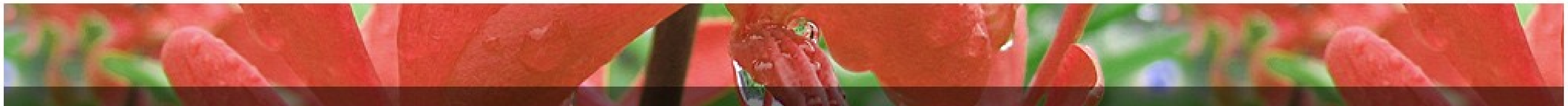
2bits.com Services

- Clients mainly in the USA and Canada, as well as in Europe
- Performance tuning and optimization
- Drupal site monitoring
- Development/Customization of modules
- Subcontracting development projects (developers' developer)
- Server provisioning, installation, upgrades
- Automated backups



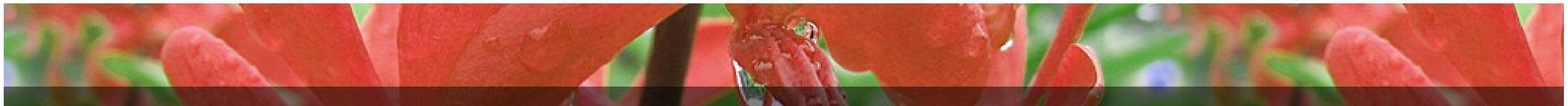
Agenda

- Introduction
- The LAMP Stack
  - Linux, Apache, MySQL, PHP
- Drupal
  - Database queries
  - Modules
  - Caching
- Measurement and monitoring tools
- Case studies
- Questions, discussion



Definitions

- Performance
- Scalability
- High Availability
- Load Balancing
- Performance Assessment/Analysis
- Performance Optimization/Tuning



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
- Most often, everyone *wants* them **all**, but don't *need* them, yet willing to pay for **none**!

Diminishing Returns

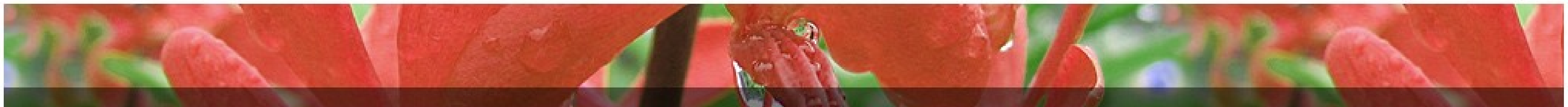
- Often, there are some “low hanging fruit”, easy to pick, that provide noticeable improvement with relatively little effort
- After that, it gets harder and harder to achieve more performance (more effort, less return)
  - More infrastructure (split server, multiple web head)
  - Patching of Drupal
  - Re-architecting the application (e.g. CCK, Views)

Diagnosis

- A proper diagnosis is essential for any solutions
- Otherwise, you are running blind
- Like a doctor who says "let us try medicine A, and surgery B, as well as procedure C, and see *maybe* things will get better" **without** lab tests and examinations!
- Must be based on proper data
- Analysis of the data collected

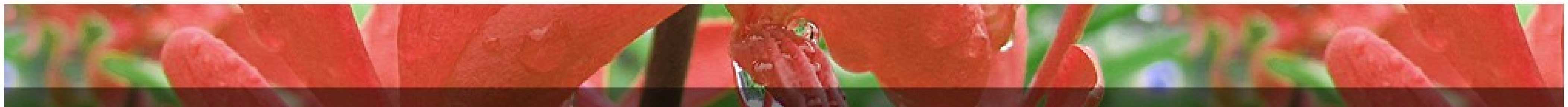
Validation

- Validate the results on a test server
- Copy the site (MySQL dump and tar archive, maybe without images)
- Re-create the site
- Measure again and see if the relative times are about the same
- Avoid “wild goose chase”



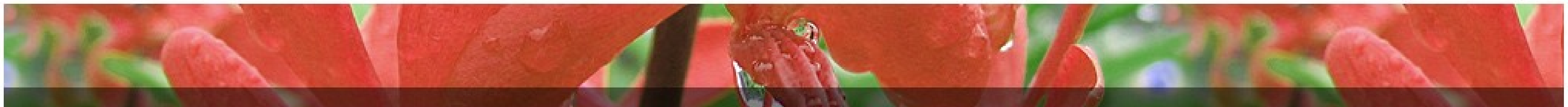
Hardware

- Physical server matters
  - Dedicated
  - VPS
- Multiple cores are the norm now
- 4 are better than 2, and 8 and better than 4
- Lots of RAM (caching the file system and the database, as much as possible)
- Multiple disks if you can for different file systems
- Always mirrored!
- Not applicable to shared hosting



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

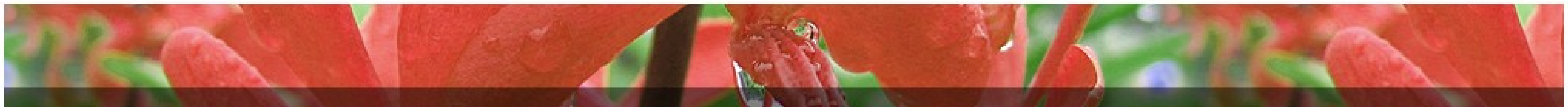
- Most commonly used stack for hosting Drupal and similar applications
  - Linux
  - Apache
  - MySQL
  - PHP
- Most of this presentation applies to *BSD as well. Parts apply to Windows (anyone use it?).

Linux

- Use a proven stable distro (Debian stable, Ubuntu Server LTS, CentOS)
- Use recent versions
- 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 Java, No PostgreSQL if you are only using MySQL, ...etc.)

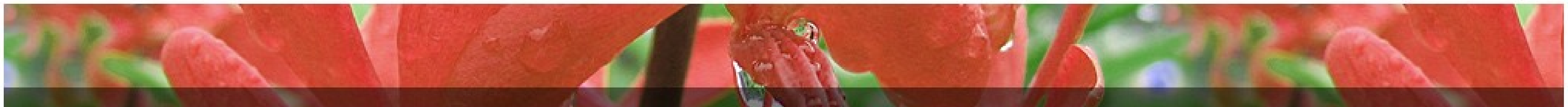
Linux (cont'd)

- Balance “compile your own” vs. upgrades
- Compile your own
  - Pros: Full control on specifc 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



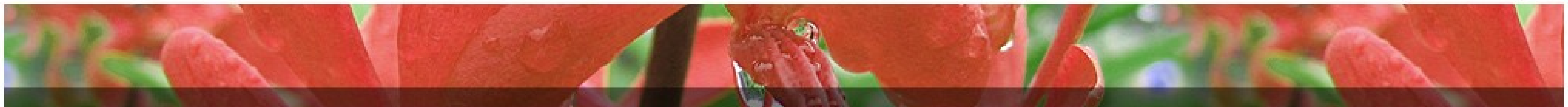
Apache

- Most popular, most supported, most stable and feature rich
- Cut the fat
  - Enable only mod_php and mod_rewrite (as a start)
  - Disable everything else (mod_python, mod_perl, ...)



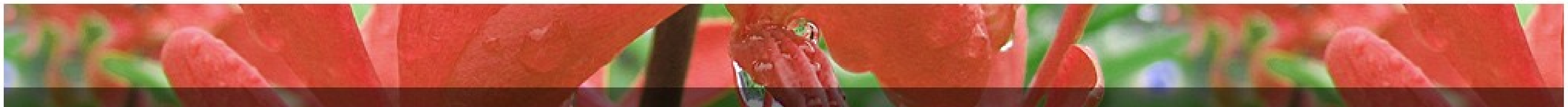
Apache

- MaxClients (prevent swapping/thrashing)
  - 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!)
- MaxRequestsPerChild
  - To terminate the process faster, and free up memory
- KeepAlive
  - Should be low (~ 3 seconds)
- mod_gzip/deflate
  - Compress HTML, CSS, JS, ...



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)

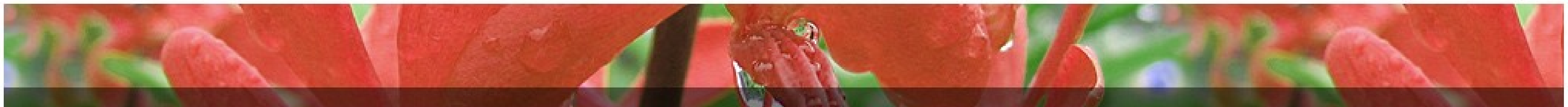


Apache Alternatives

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

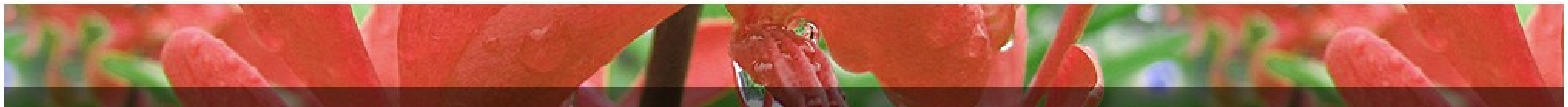
MySQL

- 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 (good for heavily hit tables, such as sessions, watchdog, ...)
  - Oracle owns the engine now ...



MySQL Engines

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

MySQL tuning

- Query cache
  - Probably the most important thing to tune
- Table cache
  - Also important
- Key buffer
- InnoDB (e.g. sessions, watchdog, ...)
- Temp tables on Linux tempfs (in memory)

MySQL replication

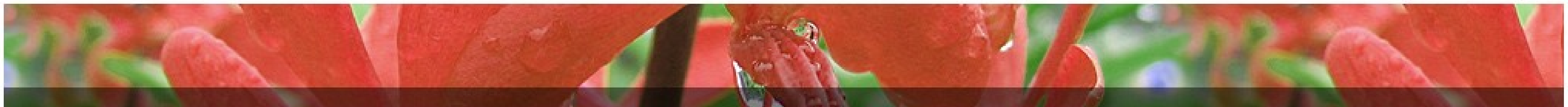
- Now in use on drupal.org
  - INSERT/UPDATE/DELETE go to the master
  - SELECTs go the slave
- Noticable improvement
- Patch here http://drupal.org/node/147160
- Beware of complexity (code and infrastructure)

PHP

- Use a recent version
  - 5.2 minimum for Drupal 7.x, and many 6.x contribs
- Install an Op-code cache / Accelerator
  - eAccelerator
  - APC
  - Xcache
  - Zend (commerical)

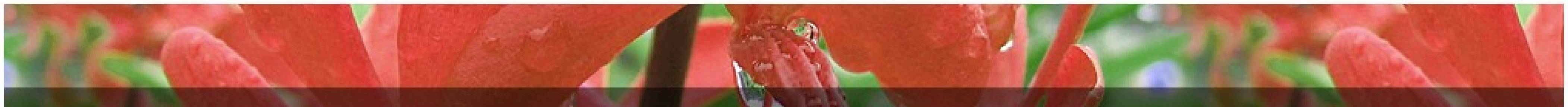
Op-code caches

- 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
- APC vs. eAccelerator vs. Xcache benchmark on 2bits.com
- Drawbacks (for other than APC)
  - Other than APC, they may crash often
  - Use logwatcher to auto restart Apache

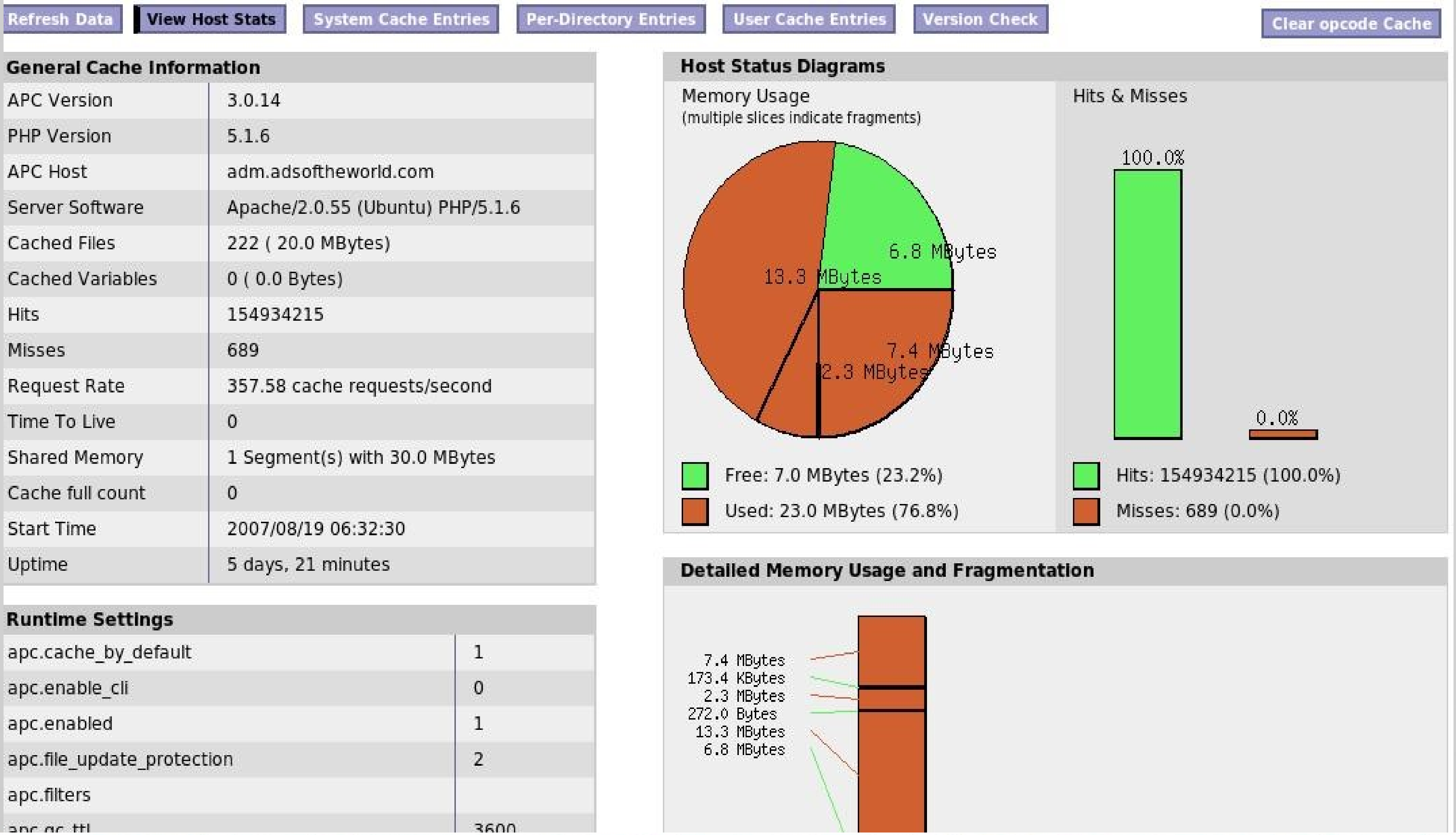


Unless ...

- Accelerators will not help in certain cases
  - When it is not just code execution
  - Network connections (Web 2.0 widgets, emails, some ads)
  - Sorting of arrays
  - Heavy database access
  - Combinations of the above
  - tagadelic, node access modules, admin_menu, forum, tracker)



APC admin



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%)

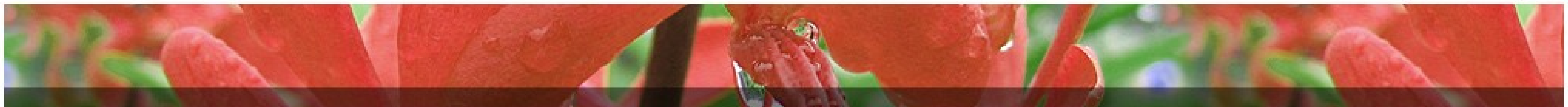
Hits & Misses

🟩 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 100MB (but depends on modules installed, Apache modules, ...)



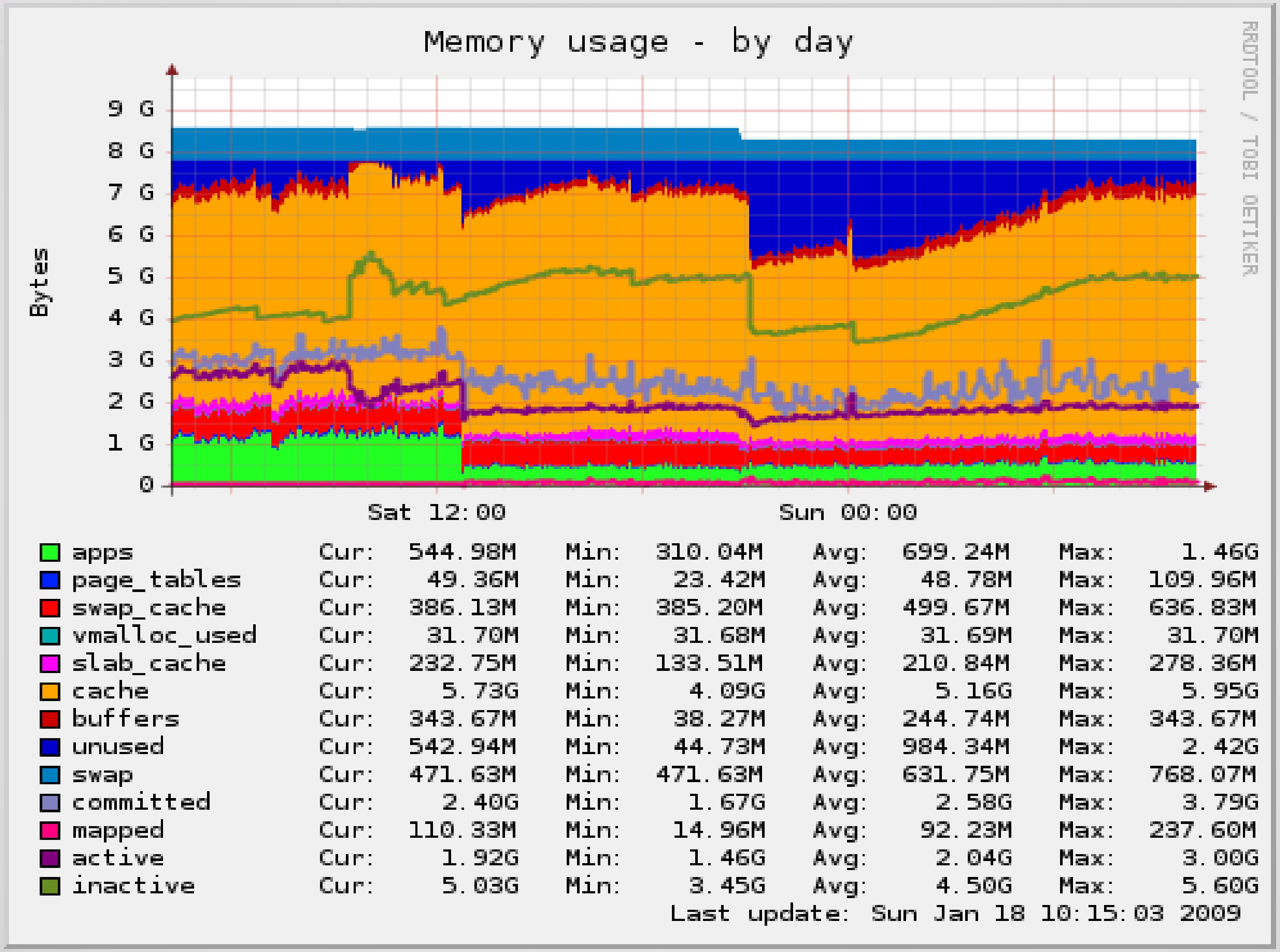
PHP as CGI

- 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
- Better separation of permissions (e.g. Shared hosting)
  - If you have one server and one Linux user, permissions may not be an issue.
- Of late, Apache with fcgid has proven to be stable as well as better on memory usage (major savings).

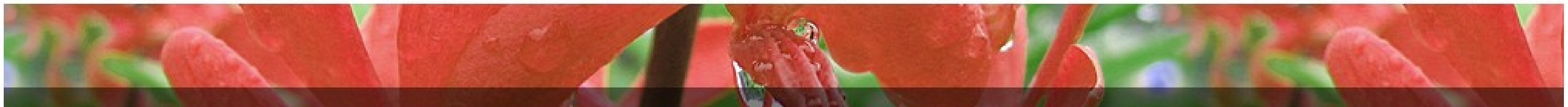
mod_php vs. fcgid



```
🟩 apps          Cur:  544.98M   Min:  310.04M   Avg:  699.24M   Max:    1.46G
🟦 page_tables   Cur:   49.36M   Min:   23.42M   Avg:   48.78M   Max:  109.96M
🟥 swap_cache    Cur:  386.13M   Min:  385.20M   Avg:  499.67M   Max:  636.83M
🟦 vmalloc_used  Cur:   31.70M   Min:   31.68M   Avg:   31.69M   Max:   31.70M
🟪 slab_cache    Cur:  232.75M   Min:  133.51M   Avg:  210.84M   Max:  278.36M
🟧 cache         Cur:    5.73G   Min:    4.09G   Avg:    5.16G   Max:    5.95G
🟥 buffers       Cur:  343.67M   Min:   38.27M   Avg:  244.74M   Max:  343.67M
🟦 unused        Cur:  542.94M   Min:   44.73M   Avg:  984.34M   Max:    2.42G
🟦 swap          Cur:  471.63M   Min:  471.63M   Avg:  631.75M   Max:  768.07M
🟪 committed     Cur:    2.40G   Min:    1.67G   Avg:    2.58G   Max:    3.79G
🟥 mapped        Cur:  110.33M   Min:   14.96M   Avg:   92.23M   Max:  237.60M
🟪 active        Cur:    1.92G   Min:    1.46G   Avg:    2.04G   Max:    3.00G
🟩 inactive      Cur:    5.03G   Min:    3.45G   Avg:    4.50G   Max:    5.60G
                                   Last update: Sun Jan 18 10:15:03 2009
```

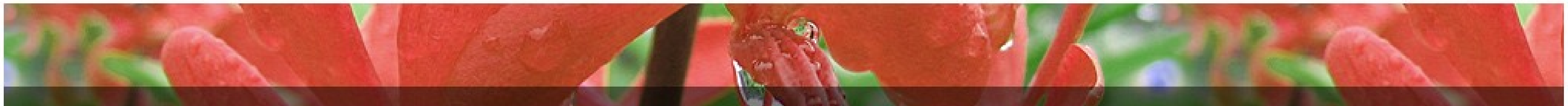
Other ways for PHP

- Roadsend PHP compiler
  - Compiles PHP to native code!
  - http://code.roadsend.com/pcc
- PHC (incomplete, Parrot spinoff)
  - http://www.phpcompiler.org/
- Caucho Quercus
  - Implementation of PHP written in Java!
  - http://quercus.caucho.com/



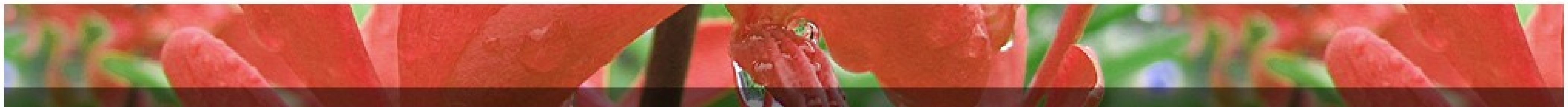
Drupal

- Mainly database intensive (100s of queries per page)
- Can be CPU bound (certain modules, resource starved hosts, ...)
- Can be a memory intensive (lots of modules, or if untuned)
- 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



Drupal (cont'd)

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

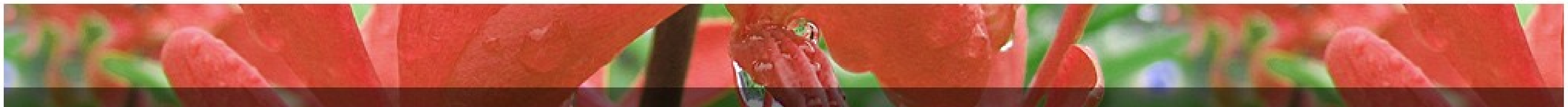


Module calls network?

- Does your module do stuff over the network?
- For every page view?
  - Email 2,000 users on node/comment submit (og!)
  - Call web 2.0 widgets (e.g. Digg this)?
- Don't!
- Cache the data
- Use job_queue
- Or queue_mail module

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.



CDN

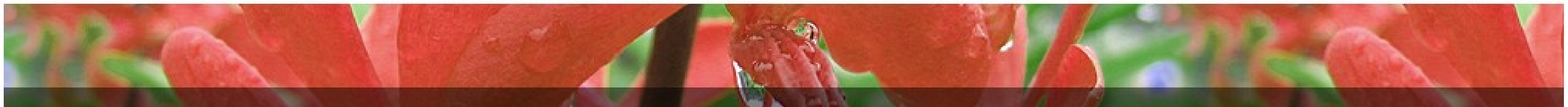
- Content Delivery Network
  - Servers in different locations (e.g. Europe, US East coast and US West cost)
  - Monthly fees, as well as volume fees.
  - Pricing varies wildly
  - Proximity based, user requests fullfilled from nearest servers
  - Akamai, Panther Express

Caching Reverse Proxy

- Squid Cache
  - Stores static files (css, js, images)
  - Needs a patch for HTML (i.e. Drupal generated pages)
    - on 2bits.com for Drupal 6.x
  - Vast performance improvement
    - Requests never reach the web server, let alone PHP or the database!
  - Intermediate proxies still an issue
- Varnish
  - Newer than Squid

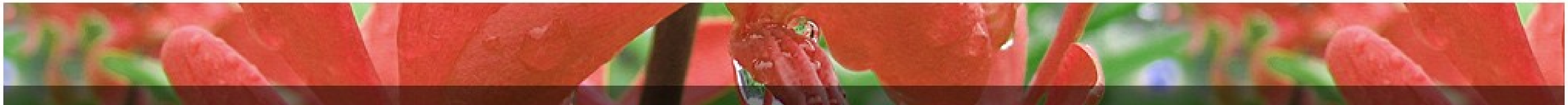
Drupal caching

- 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



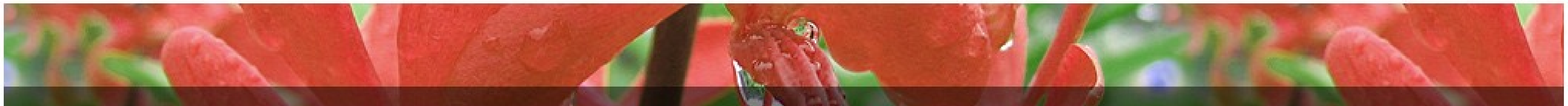
Drupal caching (cont'd)

- Certain parts of cache are always on and cannot be turned off (but see later)
  - Filter
  - Menu
  - Variables
  - Forms



Boost

- Drupal module
- Creates HTML for pages and stores it in files
- Requires changes to .htaccess and symlinks
- Usable on shared hosts as well as VPS/Ded.
- Vastly enhances the ability to handle traffic spikes
- Make sure you TRUNCATE sessions when installing, otherwise you will see stale pages
- Can leave dangling symlinks in the file system



Drupal caching (cont'd)

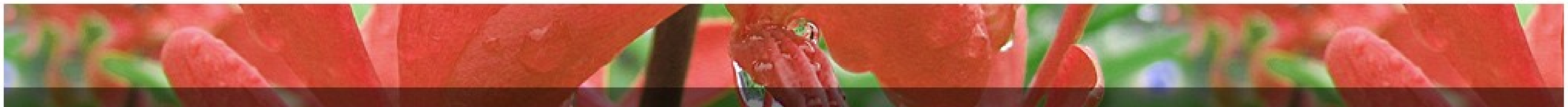
- If you use Squid as a cache, then those may not apply
- Consider other caching modules that use files
  - FS Fastpath
    - Still some of Drupal's PHP is executed
  - File Cache
    - Useful for shared hosting
    - Uses flat files to store the cached objects outside the DB
    - Available in cache router module too

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)

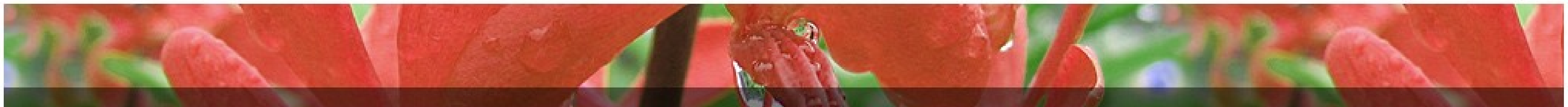
Block caching

- Contrib module for Drupal 5.x
- In core since Drupal 6 (but less configurability)
- Eliminates the overhead of generating blocks for each page view
- 64% improvement (Drupal 6.x)

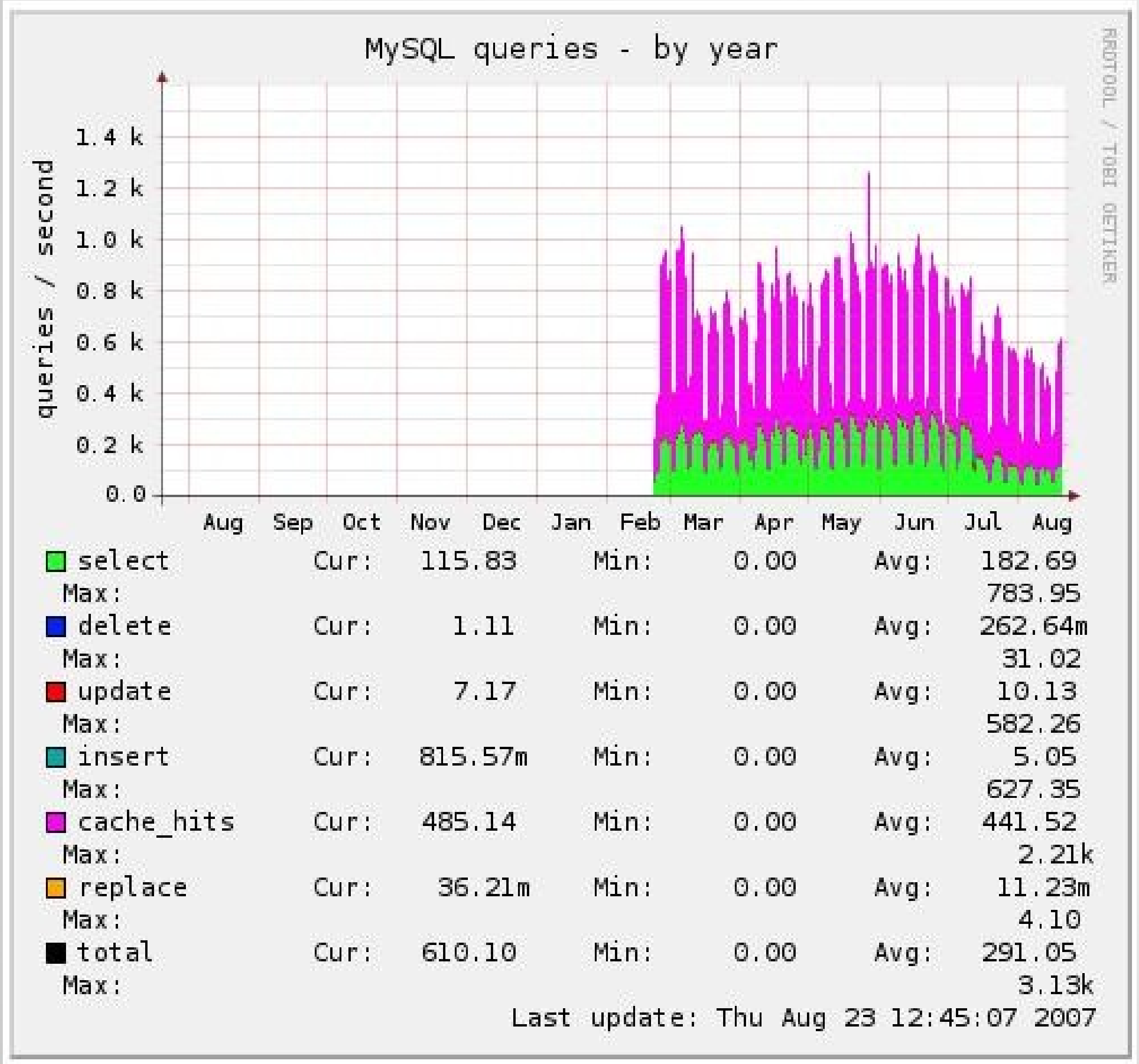


memcached

- 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.x
- Should be seamless for Drupal 6.x (at least core)



memcached (cont'd)



```
🟩 select      Cur:  115.83    Min:   0.00    Avg:  182.69
 Max:                                                783.95
🟦 delete      Cur:    1.11    Min:   0.00    Avg:  262.64m
 Max:                                                 31.02
🟥 update      Cur:    7.17    Min:   0.00    Avg:   10.13
 Max:                                                582.26
🟦 insert      Cur:  815.57m   Min:   0.00    Avg:    5.05
 Max:                                                627.35
🟪 cache_hits  Cur:  485.14    Min:   0.00    Avg:  441.52
 Max:                                                 2.21k
🟧 replace     Cur:   36.21m   Min:   0.00    Avg:   11.23m
 Max:                                                  4.10
⬛ total       Cur:  610.10    Min:   0.00    Avg:  291.05
 Max:                                                 3.13k
                          Last update: Thu Aug 23 12:45:07 2007
```

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

memcached (cont'd)

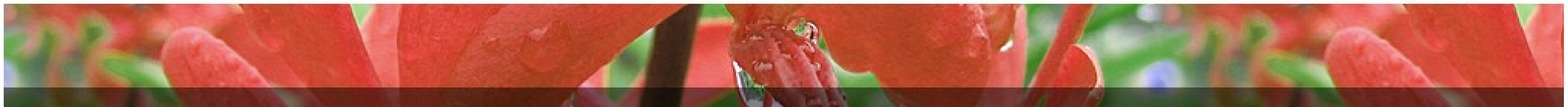
- Watch out for:
  - Must start Apache after memcached restart
- Also:
  - Gets complex as you add instances
  - Gets more compelx as you add instances on other servers

Advanced Caching

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

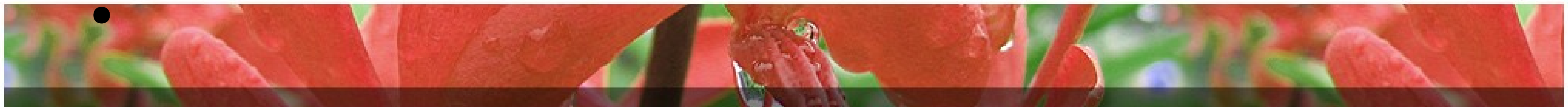
Slow modules

- 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!
- Tagadelic with free tagging (many 1,000s)
- Admin_menu (adds up to 500ms)
- Node Access modules with large number of nodes (10,000 or more)
-



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

atop

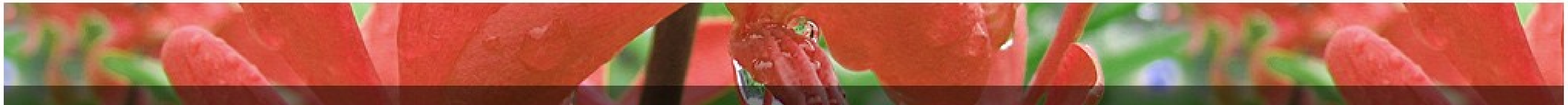
- ATS Top
- Different format and info
- Shows network stats
- Runs a collection daemon in the background

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

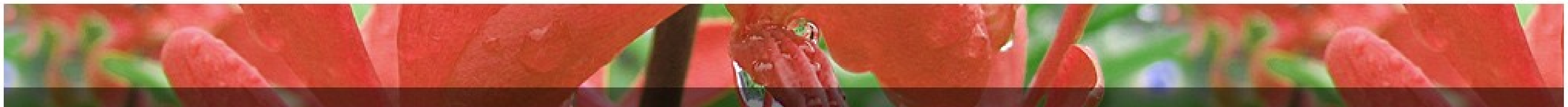
- Reads and analyses Apache's access log
- Shows all/recent hits
  - Request per second, KB/sec, KB/req
  - 2xx, 3xx, 4xx, 5xx
- List of requests being served
- Good to detect crawlers
- To run it use:
  - apachetop -f /var/log/access.log

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?

Other MySQL tools

- Mysqlreport
  - Displays statistics
  - No recommendations
- MySQL DB tuning primer
  - A shell script that reads variables from MySQL
  - 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
  - Also mysqlsla script

Stress testing

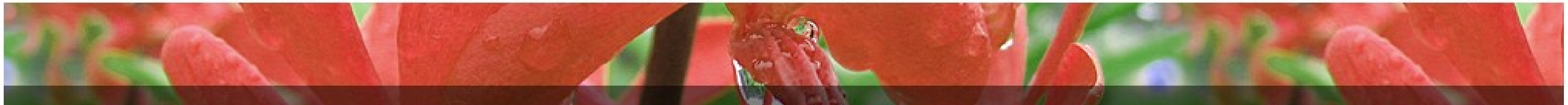
- How much requests per second can your site handle?
- Are you ready for a digg?
- Do you know your performance and bottlenecks before you deploy? or after?
- The challenge is finding a realistic workload and simulating it
- If you find bottlenecks, submit patches

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

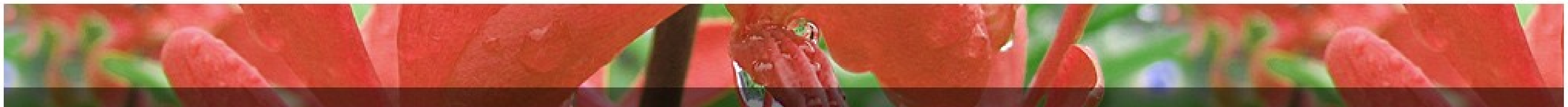
Stress testing (cont'd)

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



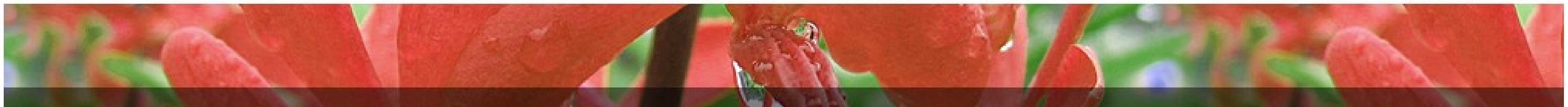
Graphical Monitoring

- 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 (e.g. We wrote one for php-cgi when running fcgid)
- Cacti
  - Similar features



Nagios

- A monitoring platform
  - Alerts by email, XMPP, SMS, ...
- New module for Drupal (5.x and 6.x)
  - Alerts about many things
    - Pending core and contrib releases (security!)
    - Database schema updates
    - File directory permissions
    - Performance
    - Much more
    - API too!

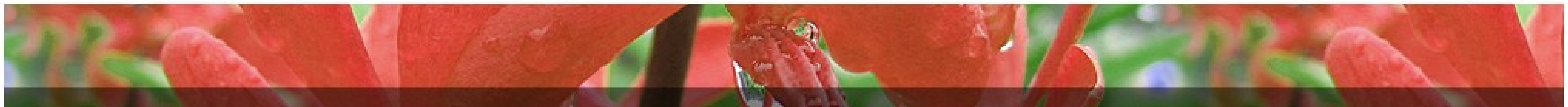


Web site statistics

- 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)

Site Statistics

- Do you know how many page views per days your site gets? (not just visits!)
- Google Analytics
  - Measures humans only (javascript)
  - Does not count access to feeds
  - Nor search engine and spam bots
- Awstats
  - Measures everything (also bandwidth!)
  - Relies on Apache's logs

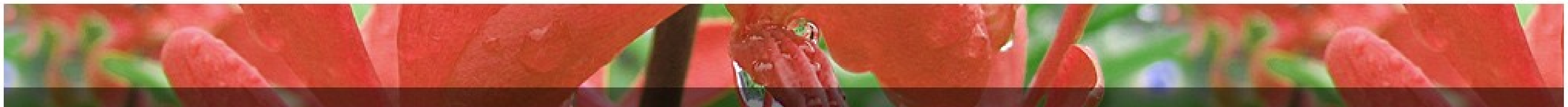


Drupal tools

- 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

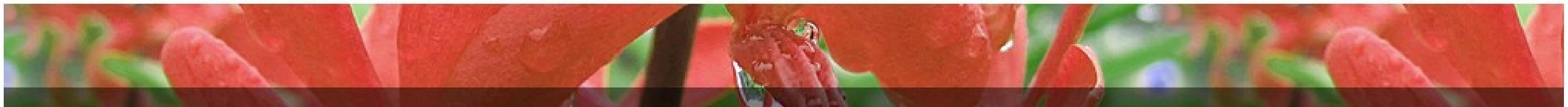
Performance Logging

- Started as an independent project by 2bits
- Now part of Devel (5.x, 6.x and 7.x, in -dev)
- Aims at collecting info for analysis of performance
  - Which pages use most queries
  - Which pages use most time to generate
  - Average and maximums
  - Logs to database (dev/test) or APC (ok for live sites)
  - Can be combined with stress testing (ab/siege)



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



Case studies

In real life action ...

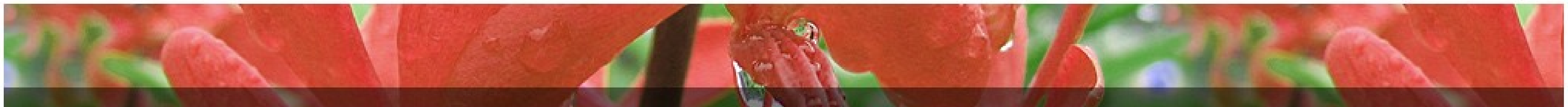
Case 1: Million pages

Can Drupal do 1,000,000 page views a day?

“Yes we can!”

How?

- Dedicated server (single server in this case)
- Lean site (no views, no CCK, no locale, no statistics, but has votingAPI, fivestar, subscriptions)
- Memcache is a live saver
- APC
- Fcgid instead of mod_php (saves memory)



Case 2: Slow forums

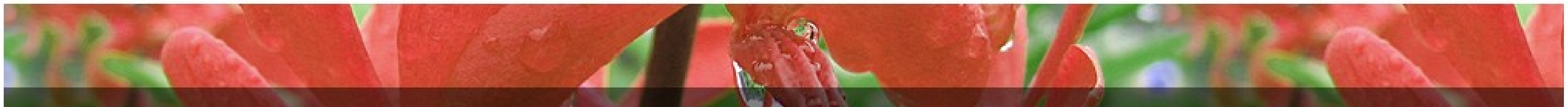
- Node access (taxonomy access lite in this case) was used to make some forums private
- Not needed in that case

Case 3: 10s of seconds

- A site which tools tens of seconds to process the submission of a node
- Og was sending emails to 1,000s of users instantly
- Used job_queue module

Case4 :Authenticated

- Intranet application for 200 concurrent users (96,000 in users table)
- Could only do 30!
- Using CREATE TEMPORARY TABLE on each user's home page
- 2bits.com was able to scale it in our labs to 200 users
  - Use cache_get()/set() for this query
  - InnoDB for sessions, watchdog, accesslog tables



Case 5: Hangs

- Page loading “hangs”
- The site used a digg widget that did a fsock_open() call, and digg's API host was down!
- Using netstat, we knew the IP address and saw many connections
- Removed ...

Case 6: Slow LAN

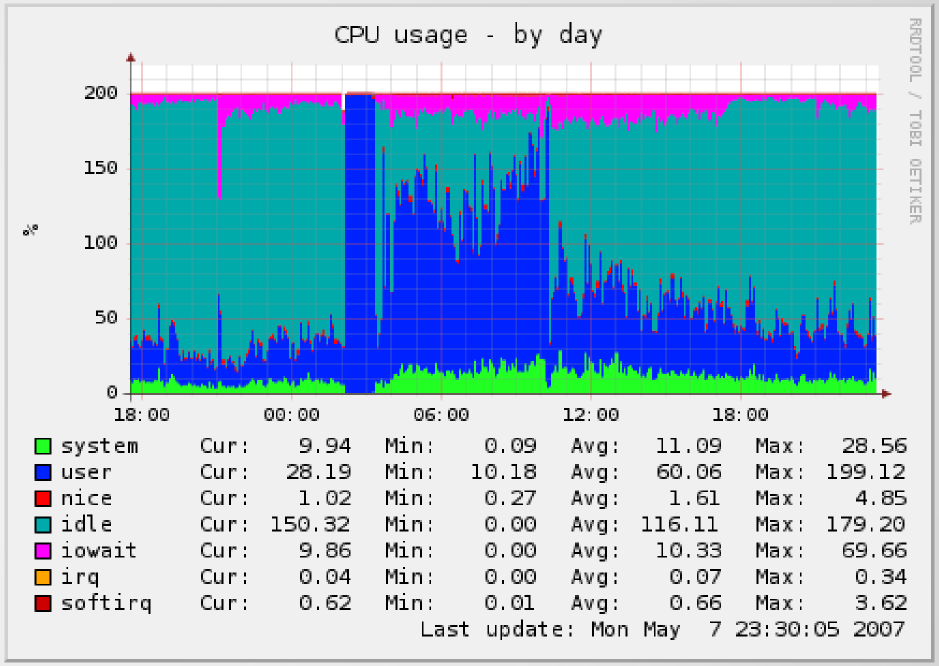
- Site was 10 – 12 seconds
- Two VPS's, one for web the other for database
- 10Mbps connection! Not enough, very high query time
- Increased to 60Mbps, much better ...

Case 7: Crawler!

- Site is very slow
- A crawler was hitting it repeatedly
- Turned out to a worm
- Use apachetop
- Use iptables to block the Ips
- Freed up resources instantly

CPU 100%

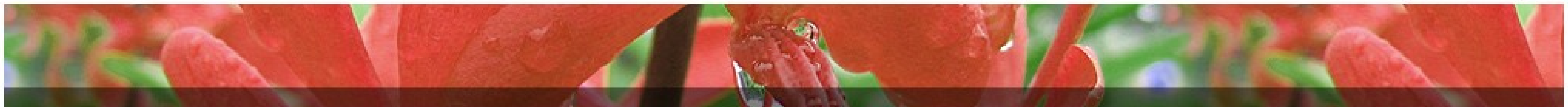
- What was it?
- Was OK for a day
- eAccelerator (svn303 + PHP 5)
- Note CPU utilization (100%, then high, then dropped low when good version used)



```
🟩 system    Cur:    9.94  Min:   0.09  Avg:  11.09  Max:  28.56
🟦 user      Cur:   28.19  Min:  10.18  Avg:  60.06  Max: 199.12
🟥 nice      Cur:    1.02  Min:   0.27  Avg:   1.61  Max:   4.85
🟦 idle      Cur:  150.32  Min:   0.00  Avg: 116.11  Max: 179.20
🟪 iowait    Cur:    9.86  Min:   0.00  Avg:  10.33  Max:  69.66
🟧 irq       Cur:    0.04  Min:   0.00  Avg:   0.07  Max:   0.34
🟥 softirq   Cur:    0.62  Min:   0.01  Avg:   0.66  Max:   3.62
                          Last update: Mon May  7 23:30:05 2007
```

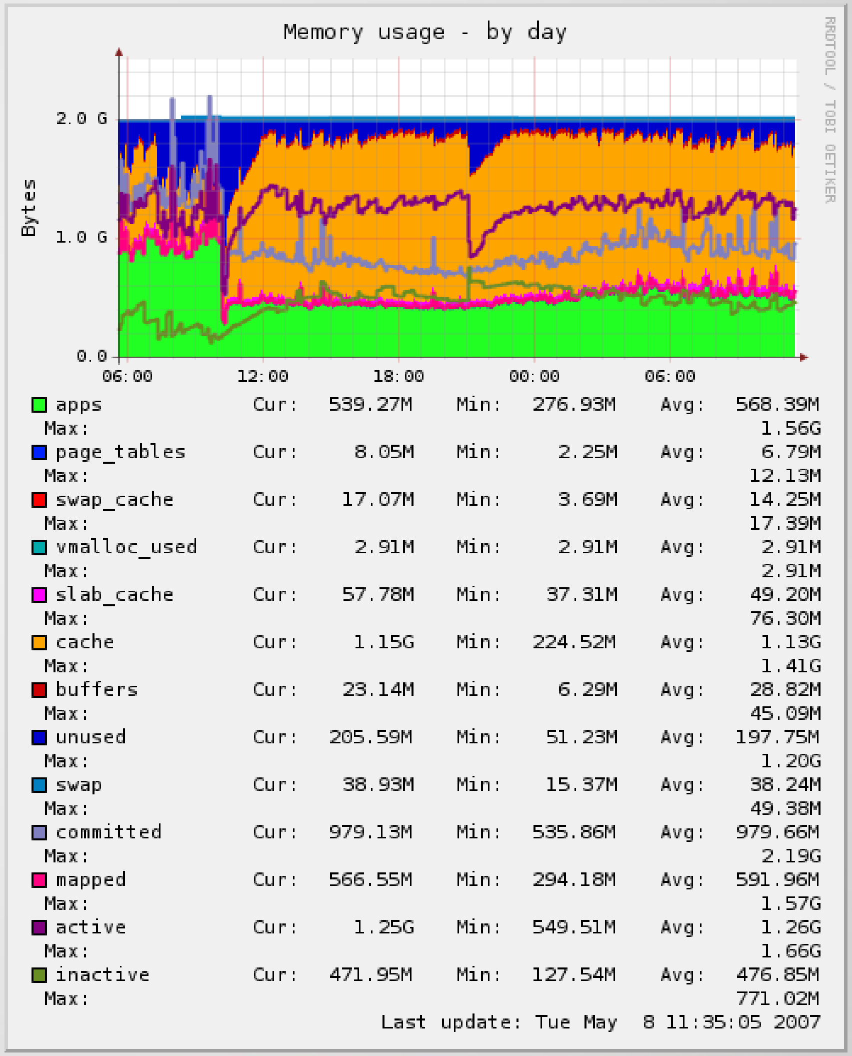
Memory

- 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)



```
🟩 apps          Cur:  539.27M   Min:  276.93M   Avg:  568.39M
  Max:                                                 1.56G
🟦 page_tables   Cur:    8.05M   Min:    2.25M   Avg:    6.79M
  Max:                                                12.13M
🟥 swap_cache    Cur:   17.07M   Min:    3.69M   Avg:   14.25M
  Max:                                                17.39M
🟦 vmalloc_used  Cur:    2.91M   Min:    2.91M   Avg:    2.91M
  Max:                                                 2.91M
🟪 slab_cache    Cur:   57.78M   Min:   37.31M   Avg:   49.20M
  Max:                                                76.30M
🟧 cache         Cur:    1.15G   Min:  224.52M   Avg:    1.13G
  Max:                                                 1.41G
🟥 buffers       Cur:   23.14M   Min:    6.29M   Avg:   28.82M
  Max:                                                45.09M
🟦 unused        Cur:  205.59M   Min:   51.23M   Avg:  197.75M
  Max:                                                 1.20G
🟦 swap          Cur:   38.93M   Min:   15.37M   Avg:   38.24M
  Max:                                                49.38M
🟪 committed     Cur:  979.13M   Min:  535.86M   Avg:  979.66M
  Max:                                                 2.19G
🟥 mapped        Cur:  566.55M   Min:  294.18M   Avg:  591.96M
  Max:                                                 1.57G
🟪 active        Cur:    1.25G   Min:  549.51M   Avg:    1.26G
  Max:                                                 1.66G
🟩 inactive      Cur:  471.95M   Min:  127.54M   Avg:  476.85M
  Max:                                               771.02M
                           Last update: Tue May  8 11:35:05 2007
```

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
- Private gigabit LAN if you have two servers
- Occasionally you will have stubborn crawlers though
- Or even a DDoS
- Or worse, extortion
- Can eat up resources, including network

DDoS

- “Distributed Denial of Service”
  - Aggressive crawlers
  - Worms probing for vulnerabilities
- Sap the energy from your site
- External problem
- Diagnosis: Look in the web server log or use Apachetop
- Solution: use iptables to block the addresses

Further reading

Drupal Performance Tuning and Optimization section

http://2bits.com

Discussion

Questions?

Comments?