Diagnosing and speeding up a slow Drupal web site
A Case Study

Khalid Baheyeldin
January 15, 2009
http://2bits.com

Agenda

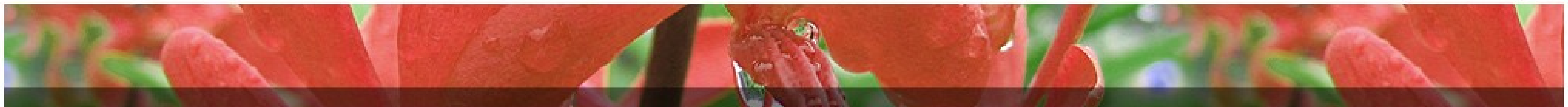
- Case study
  - The problem
  - Diagnosis
  - Analysis
  - Solution
- Background info where relevant
- Discussion

The Problem

- Getting a good description of the problem is the first step
- Bad
  - “It does not work” (What is “it” and what is “work”?)
  - “We have problems” (What “problem”?)
- Good
  - “Site is slow when we have lots of visitors”
  - “Site is slow all the time”
  - “Server is under heavy load (CPU and memory)”

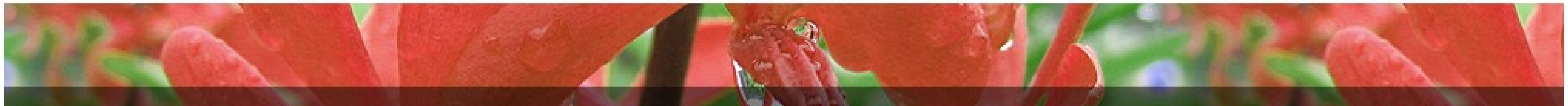
Case Study

- Vancouver Island Paragliding (viparagliding.com)
- Built by Vibe Computing
- Pushed off shared hosting because of resource usage
- Went to VPS (managed)
- Still “feels sluggish” despite that



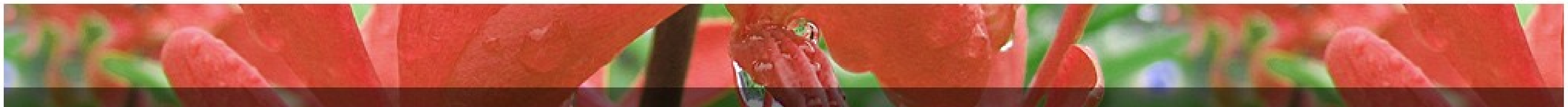
Hosting Environment

- Virtual Private Server (VPS)
- Managed (can't change things yourself, have to pay for changes)
- Hosted on Dual Core Intel Core2 @ 1.2GHz
- 512MB
- SSH available, but in a jailshell
- What does `<?php phpinfo();` say?



Quantify the problem

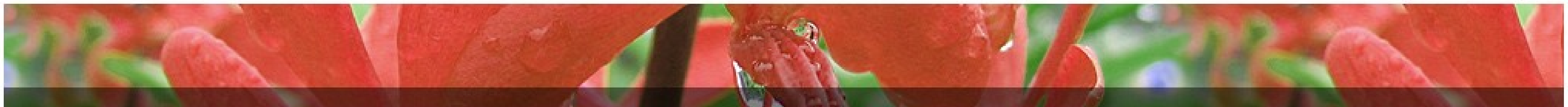
- Backend or Front End?
  - Backend
    - Hardware
    - Database
    - PHP
    - Drupal
  - Front End
    - CSS/JS
    - Images
    - Number of HTTP requests per page



Back End: Devel

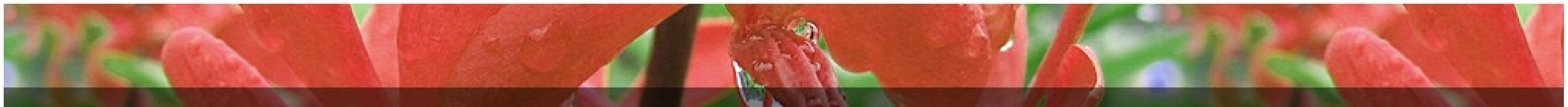
- Use Devel module
  - How many milliseconds per page?
  - How many millseconds in the database?

Page execution time was **1064.54 ms**. Executed **389** queries in **108.58** milliseconds.



Devel (Cont'd)

- Total page time
    - Simple sites: 300ms to 500ms
  - More complex sites: 600ms to 800ms
  - Over 1000 ms is definitely a red flag!
- Number of queries
  - 400-500 should be OK (what they are also)
  - Over 500 means the site is complex
- Time for Queries
  - Absolute and relative to total page time

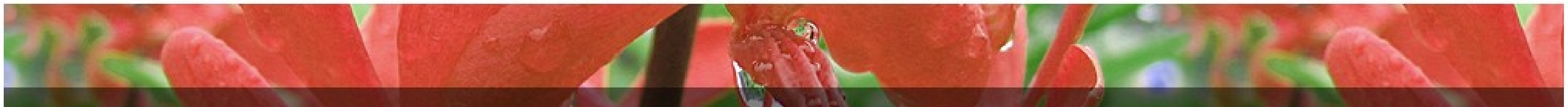


Performance Logging

- 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)

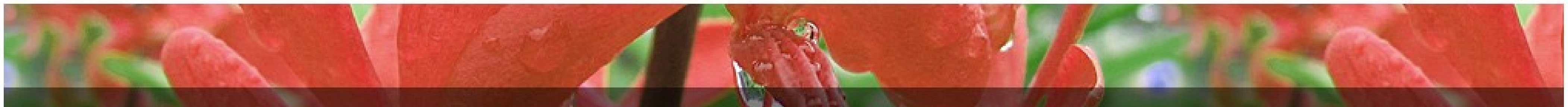
Diagnosis

- 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



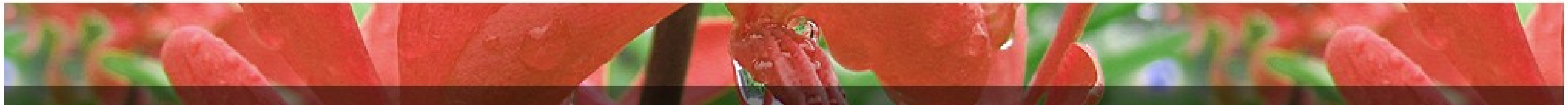
Findings

- For this particular site, at this point in time:
  - Database is not the bottleneck (good news)
  - 90% of the time spend outside the database
  - Real cause: 119 modules enabled! (open buffet binge)
  - Chances are very good that a PHP op-code cache/ accelerator will help (APC, eAccelerator, Xcache)



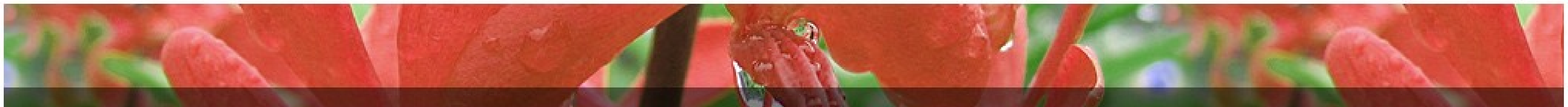
Accelerators

- Also knows as “code caches”
- Parses and tokenizes scripts and stores the result in memory (or file(s)), and uses them for future requests
- Saves memory, and CPU execution
- Translates into less time per page request
- Cannot be used with PHP in CGI mode, since it forks a new process for each request
- Can be used with FastCGI/fcgid, but they have their own issues
- Free ones: APC, eAccelerator, XCache



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)



Validation

- Validate the results on 2bits' test server
- Copy the site (MySQL dump and tar archive, without images)
- Re-create the site
- Measure again and see if the relative times are about the same

Page execution time was **794.75 ms**. Executed **397** queries in **65.41** milliseconds.

Actions

- Enable APC on test server

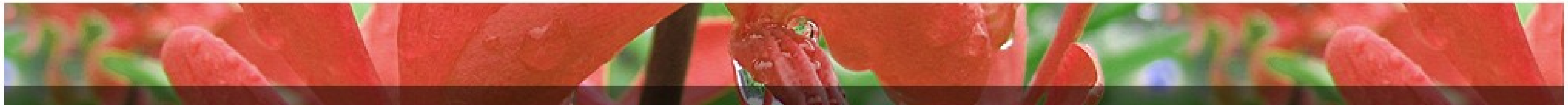
  Page execution time was **469.99** ms. Executed **397** queries in **62.77** milliseconds.

  Significant improvement!

- Disabled admin_menu module
  - Saves about 150 ms per page

  Page execution time was **306.86 ms**. Executed 380 queries in 62.21 milliseconds.

- Enable page caching
  - Page times less than 100 ms overall



Results

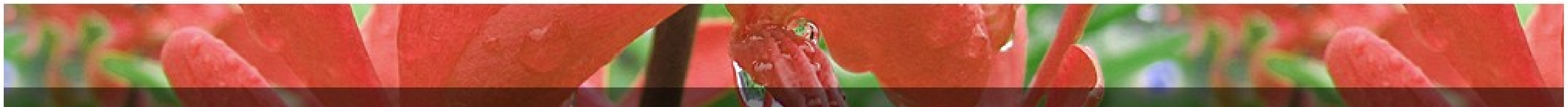
- On the live site
  - Install APC (@ $50 per hour support request, remember “managed” VPS?)
  - Had to disable Zend Optimizer
  - IonCube Loader (encoded PHP) left alone
  - Outstanding improvement!

  Page execution time was **243.84 ms**. Executed 314 queries in 47.11 milliseconds.

  - About 1/4th of the original time

Site Profile

- Complexity
  - Many modules: more code and more database queries
  - Over use of modules
  - Over use of CCK/Views/Panels
  - Makes upgrading problematic too
- Visitor types
  - Mainly Anonymous or logged in?
  - Anonymous is easy to solve (page cache, memcache, ...etc.)



Site Statistics

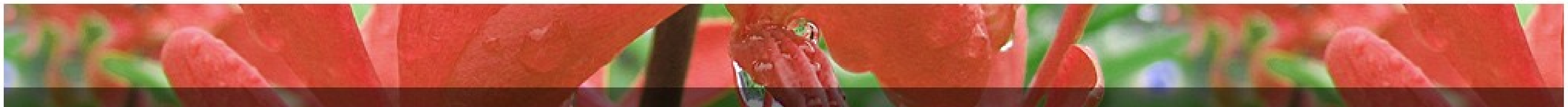
- Do you know how many page views per days your site gets? (not just visits!)
- Google Analytics
  - Measures humans only (javascript)
  - Does not count access to feeds
  - Nor search engine and spam bots
- Awstats
  - Measures everything (also bandwidth!)
  - Relies on Apache's logs

Resource Utilization

- Shared hosting tout bandwidth and disk space
  - What matters more is CPU and memory
- What is the utilization on your server?
  - If you don't know you are in the dark. How can you justify/recommend a new server?
- Munin
  - Graph over time
  - CPU, memory, disk, apache, mysql, and much more
- Cacti

Other tools

- top and htop
  - Shows what is running now, and an “at a glance” view of utilization
- vmstat 5
  - Shows snapshots every X seconds
- Free -m
  - Memory usage (-/+ buffers line)
- netstat
  - Open network connections

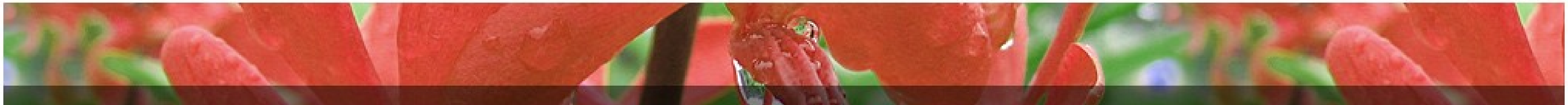


Apache

- MaxClients
  - To prevent swapping when you are on Digg
- MaxRequestsPerChild
  - To terminate the process faster, and free up memory
- KeepAlive
  - Should be low (~ 3 seconds)
- Not the only web server around (lighttpd, ngnix)
  - Only FastCGI mode

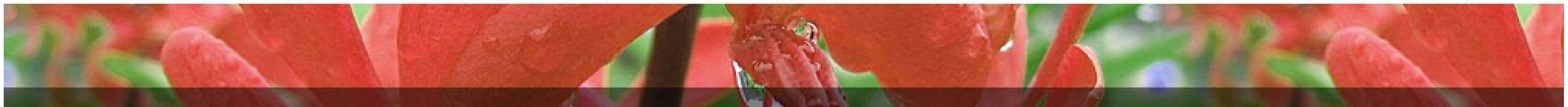
Database

- MySQL tuning
  - MyISAM vs. InnoDB
  - Often needed on large sites
  - Query cache must be enabled
  - Slow query log, and tools to analyze it
  - EXPLAIN on long running statements
- PostgreSQL
  - Slower in general due to ACID



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

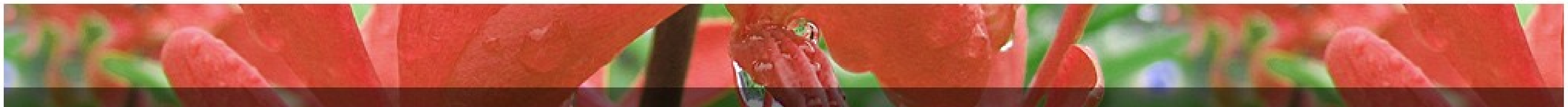


Custom Patching

- Various areas, depending on sites
  - Delaying session last access writes (in 6.x core now)
  - Path lookup whitelist

Front End

- Requires the Web Developer Extension
- YSlow FireFox Extension
- Shows you a score card
  - Background images in theme
  - Number of HTTP requests (.css, .js, images)
- Not all recommendations may be practical, but at least you know where time is eaten up



High End Sites

- Splitting the servers
  - One for database, and one or more for web server/ PHP
  - With a Load Balancer in front
  - database replication, master/slave
  - More complexity and sysadmin load, so don't jump into it without some forethought

Memcached

- Object cache daemon
- Distributed (more than one server)
- Amazing scalability, specially for anonymous users
- Requires patching for 5.x
- Two modules:
  - Memcache
  - CacheRouter

Caching Reverse Proxy

- Squid Cache
  - Stores static files (css, js, images)
  - Needs a patch for HTML
    - 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

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

Diminishing Returns

- Often, there are some low hanging fruit that can be gained quickly with little effort
  - e.g. APC in this case study
- 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)
- Same for front end tuning. Getting an “A” in Yslow will cost you!

Final Result

“I am stoked!” -- Mark D.

Further reading

Drupal Performance section at

http://2bits.com

Discussion

Questions?

Comments?