Performance Tuning and Optimization for Large Drupal Sites

Jeremy Andrews (Tag 1 Consulting)
Khalid Baheyeldin (2bits.com)
Konstantin Käfer (NowPublic)
Scott Mattoon (Sun)
Narayan Newton (OSU Open Source Lab)
Steve Rude (Achieve Internet)
David Strauss (Four Kitchen Studios)

Scalability vs. Performance

- Performance
    - Page load time
- Scalability
    - Simultaneous users
    - Size of database (including content, users)

Symptoms of Performance and Scalability Problems

- Often overlapping
- Symptoms of performance problems
  - Slow page load times
- Symptoms of scalability problems
  - Slow site during times of high usage
  - Site slows down as amount of content and users increases

Finding the Problem

- 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

Operating System

UNIX and Linux

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

Linux

- 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, ...etc.)

Operating System

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

“Compile Your Own” vs. Upgrades

- Compiling your own
  - Pros: full control of version, compilation options
  - Cons: more work to do security upgrades
- Using packages
  - Pros: less work to upgrade security releases
  - Cons: whatever version your operating system has

Web Server

Apache and PHP

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

PHP

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

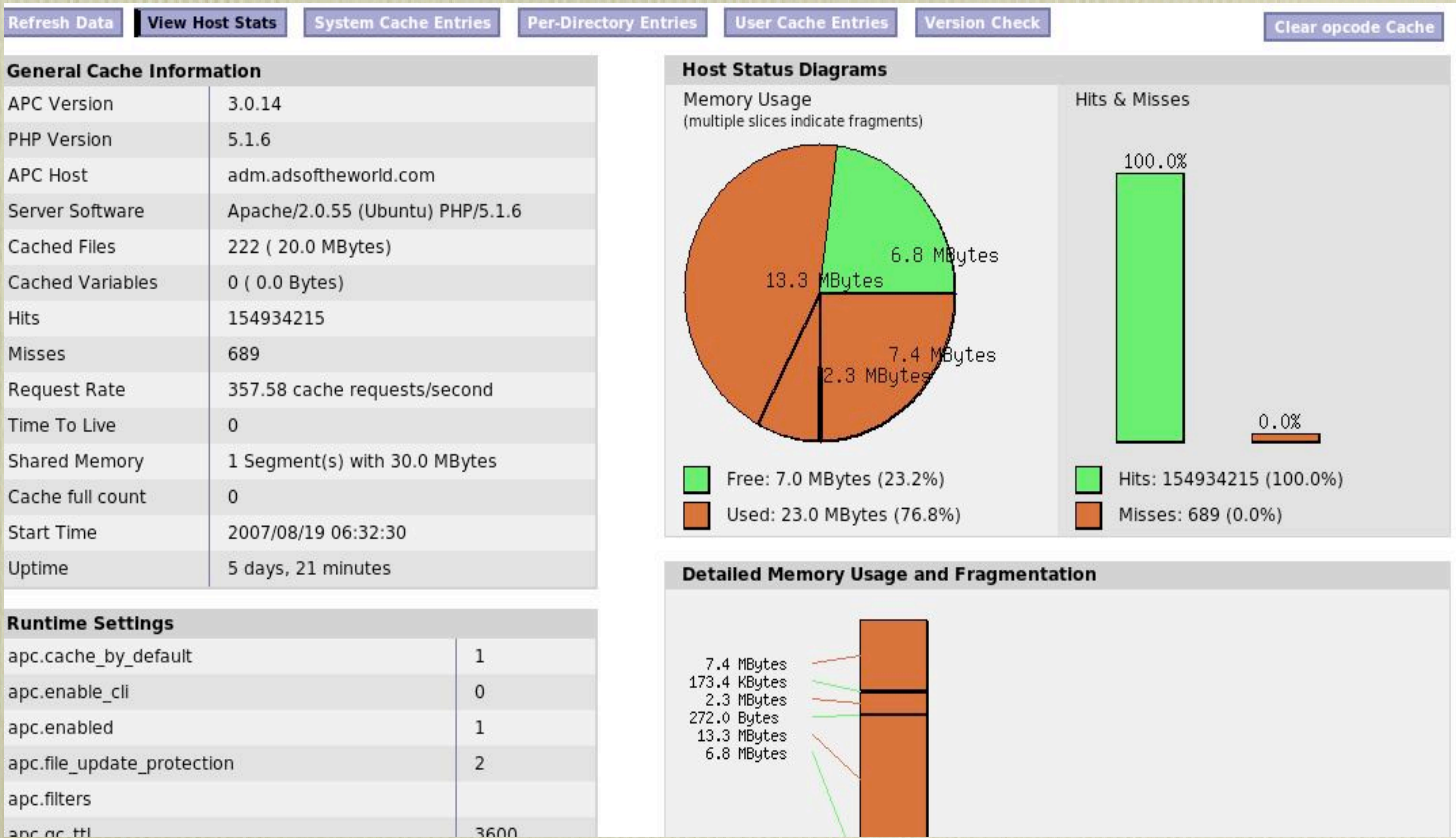
Op-code caches

- 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

Op-code caches (cont'd)

- 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

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 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!

FastCGI

- 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 PHP Options

- 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 PHP Options

- 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/

Find Hampster Wheels with DTrace

Dtrace provider for PHP

“DTrace is one of those tools that makes you wonder how you did anything without it before you'd heard of it.

Why is it better than strace and similar tools? It's non-invasive, fast, scriptable and extensible.”

- Wez Furlong

Find Hampster Wheels with DTrace

Questions you can answer with DTrace

Which functions are being called by Drupal?

```
# dtrace -n function-entry'{printf("called %s() in %s at line %d\n",\
  copyinstr(arg0), copyinstr(arg1), arg2)}' -q
```

How many times is a function called?

```
# dtrace -n function-entry'{@[copyinstr(arg0)] = count()}'
```

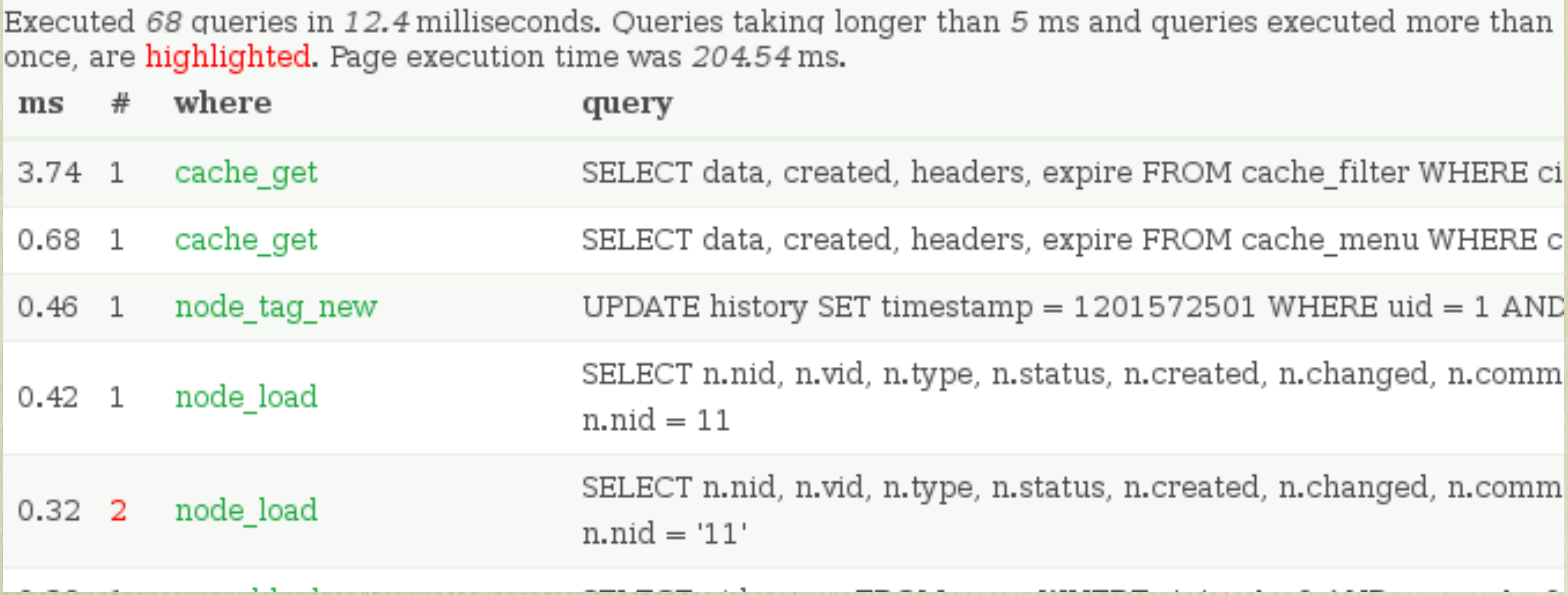
What's the file name and line number count:

```
# dtrace -n function-entry'{@[copyinstr(arg1)] = lquantize(arg2, 0, \
  5000)}'
```

Database Server

MySQL

Drupal devel module



Executed *68* queries in *12.4* milliseconds. Queries taking longer than 5 ms and queries executed more than once, are highlighted. Page execution time was *204.54* ms.

ms	#	where	query
3.74	1	cache_get	SELECT data, created, headers, expire FROM cache_filter WHERE ci
0.68	1	cache_get	SELECT data, created, headers, expire FROM cache_menu WHERE c
0.46	1	node_tag_new	UPDATE history SET timestamp = 1201572501 WHERE uid = 1 AND

0.42	1	node_load	SELECT n.nid, n.vid, n.type, n.status, n.created, n.changed, n.comm
			n.nid = 11

0.32	2	node_load	SELECT n.nid, n.vid, n.type, n.status, n.created, n.changed, n.comm
			n.nid = '11'

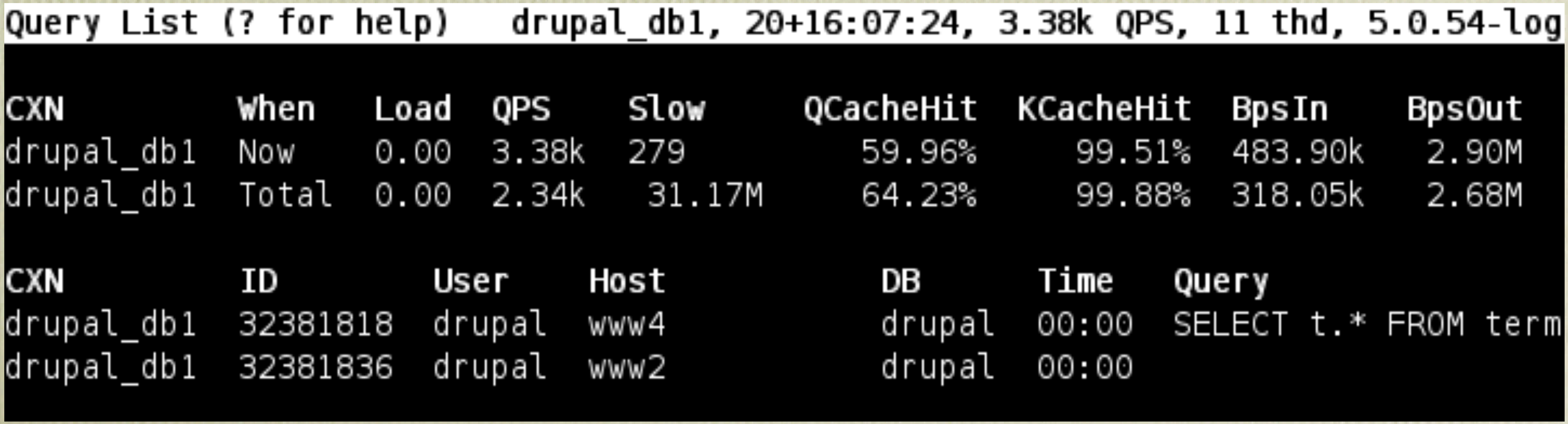
- Time in database versus total execution time.
- Slow queries, duplicate queries.
- http://drupal.org/project/devel

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?

innotop

- Inspired by mytop, much more powerful
- Originally InnoDB specific, has become much more
- Server groups, buffer statistics, replication status
- http://innotop.sourceforge.net



```
Query List (? for help)   drupal_db1, 20+16:07:24, 3.38k QPS, 11 thd, 5.0.54-log

CXN          When    Load  QPS    Slow      QCacheHit  KCacheHit  BpsIn      BpsOut
drupal_db1   Now     0.00  3.38k  279          59.96%     99.51%  483.90k     2.90M
drupal_db1   Total   0.00  2.34k   31.17M      64.23%     99.88%  318.05k     2.68M

CXN          ID        User    Host           DB      Time   Query
drupal_db1   32381818  drupal  www4           drupal  00:00  SELECT t.* FROM term
drupal_db1   32381836  drupal  www2           drupal  00:00
```

mysqlreport

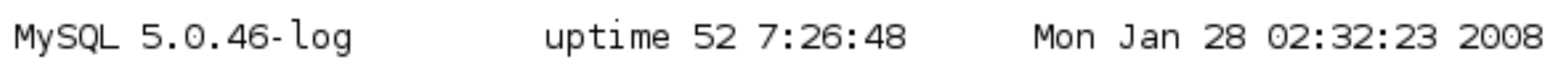
- mysqlreport
  - Perl shell script
  - Displays statistics
  - Does not make recommendations

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 mysqlsla script

Tuning with mysqlreport (1)

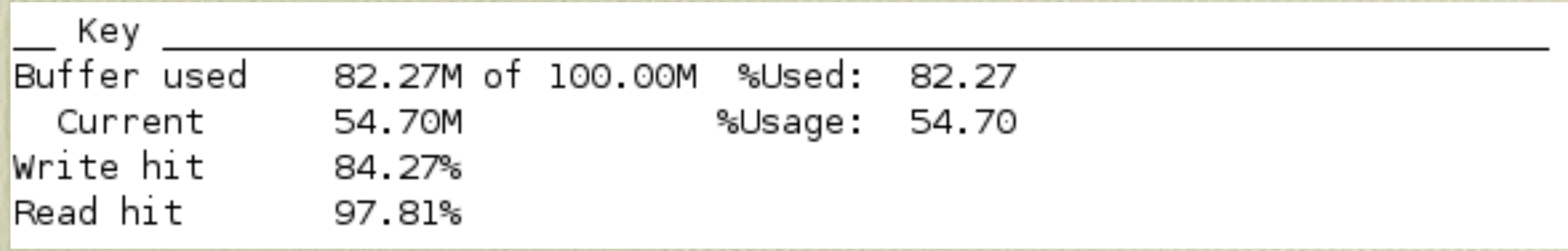
- Header:



```
MySQL 5.0.46-log          uptime 52 7:26:48       Mon Jan 28 02:32:23 2008
```

  - version, uptime

- Key (MyISAM)

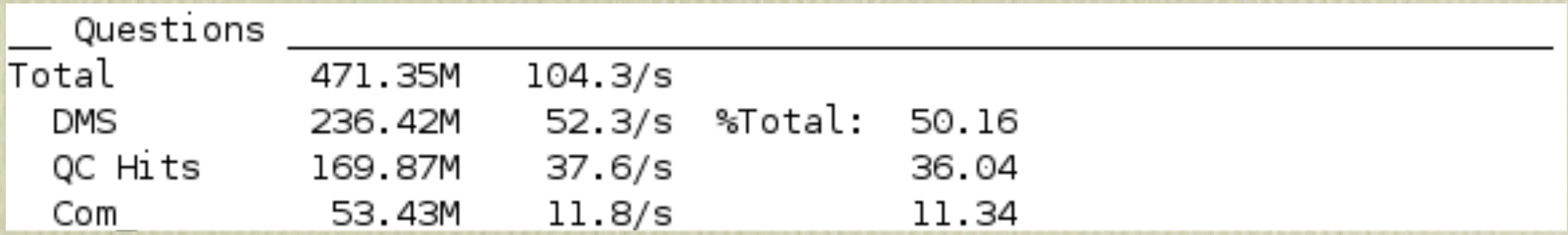


```
__ Key _________________________________________________________________
Buffer used     82.27M of 100.00M  %Used:   82.27
  Current       54.70M            %Usage:   54.70
Write hit       84.27%
Read hit        97.81%
```

  - Buffer used, high water mark
  - Current, actual usage
  - Write / Read hits rates are ration of hard drive : RAM

Tuning with mysqlreport (2)

- Questions: SQL queries & MySQL protocol

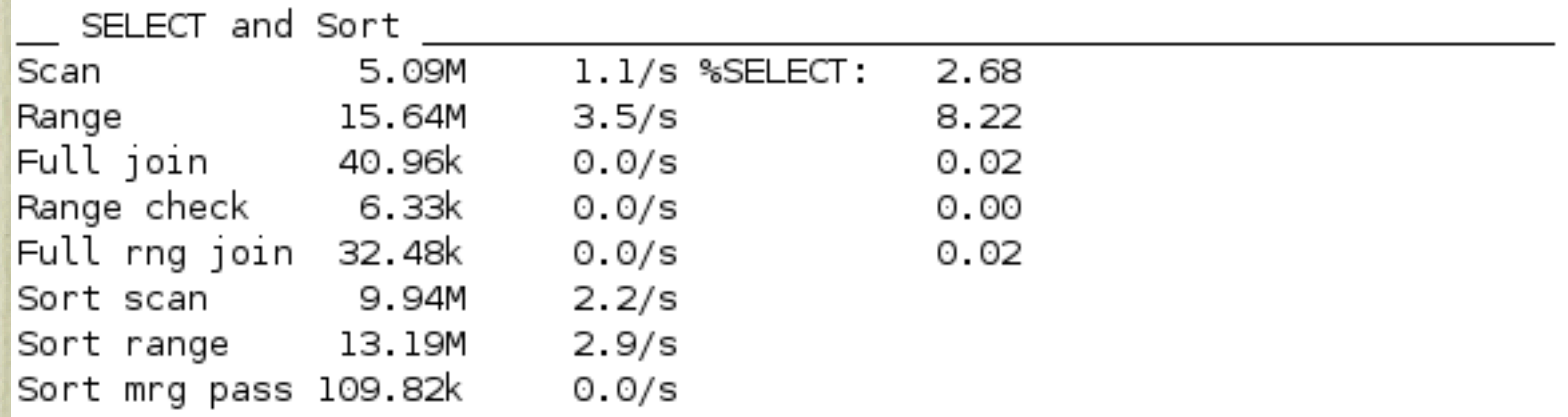


```
__ Questions ___________________________________________________________
Total         471.35M   104.3/s
  DMS         236.42M    52.3/s  %Total:  50.16
  QC Hits     169.87M    37.6/s           36.04
  Com_         53.43M    11.8/s           11.34
```

- DMS, Data Manipulation Statements (includes: SELECT, INSERT, UPDATE, DELETE)
- QC Hits, query cache
- Com, MySQL commands & protocol
- Below, break down of each

Tuning with mysqlreport (3)

- SELECT and sort

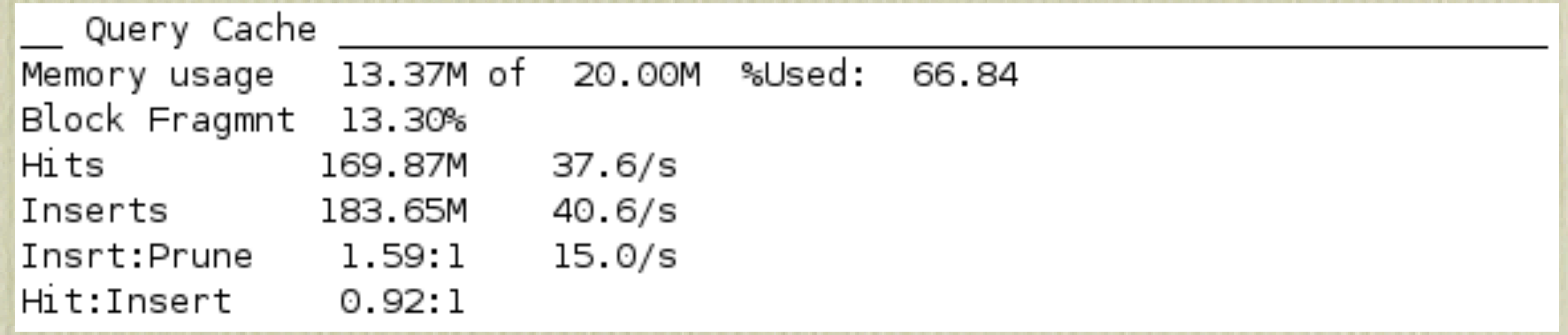


```
__ SELECT and Sort _____________________________________________________
Scan            5.09M     1.1/s %SELECT:    2.68
Range          15.64M     3.5/s             8.22
Full join      40.96k     0.0/s             0.02
Range check     6.33k     0.0/s             0.00
Full rng join  32.48k     0.0/s             0.02
Sort scan       9.94M     2.2/s
Sort range     13.19M     2.9/s
Sort mrg pass 109.82k     0.0/s
```

  - Scan: entire table
  - Full join: multiple full tables (tunable: join_buffer_size *)
  - Sorts: monitor ”SHOW STATUS LIKE Sort_merge_passes” (tunable: sort_buffer_size *)
  - * per-connection memory allocations

Tuning with mysqlreport (4)

- Query cache

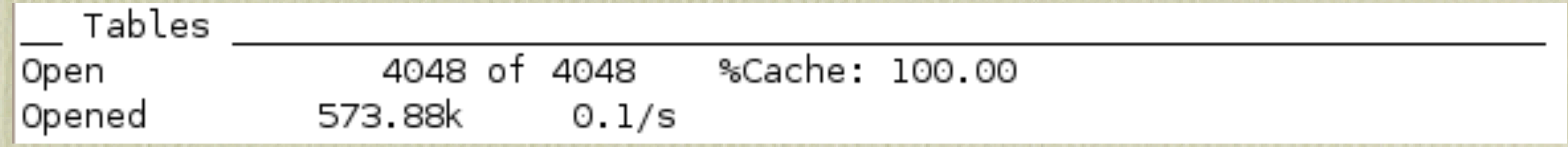


```
__ Query Cache _________________________________________________
Memory usage   13.37M of   20.00M  %Used:   66.84
Block Fragmnt  13.30%
Hits          169.87M    37.6/s
Inserts       183.65M    40.6/s
Insrt:Prune    1.59:1    15.0/s
Hit:Insert     0.92:1
```

- If too big (64M+), can cause MySQL server freezes
- Fragmented? Increase memory (query_cache_size), or decrease tunable: query_cache_min_res_unit (2k is a good place to start with Drupal)
- Review Insert:Prune and Hit:Insert
- Can control QC with SQL_NO_CACHE and SQL_CACHE

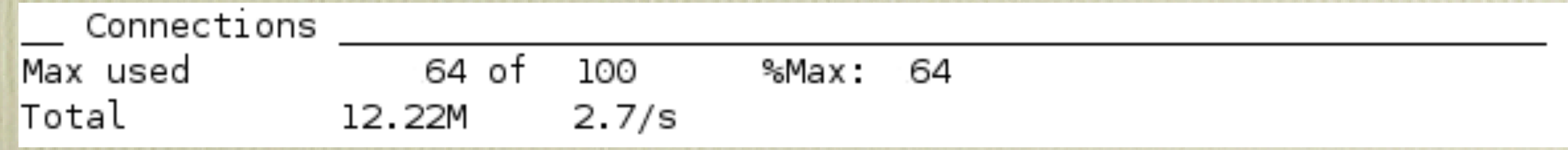
Tuning with mysqlreport (5)

- Tables



```
__ Tables _______________________________________________________________
Open             4048 of 4048    %Cache: 100.00
Opened        573.88k     0.1/s
```

  - Same table can be opened by many threads
  - More than 1/s?   Increase tunable: table_cache
  - Monitor memory usage closely when increasing

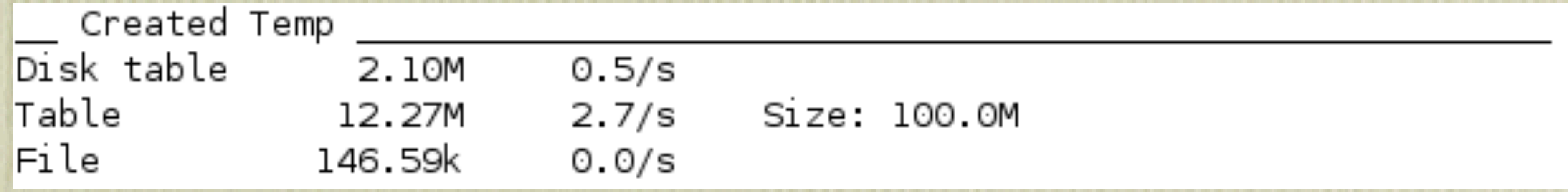


```
__ Connections __________________________________________________________
Max used           64 of   100      %Max:   64
Total          12.22M     2.7/s
```

- Connections

Tuning with mysqlreport (6)

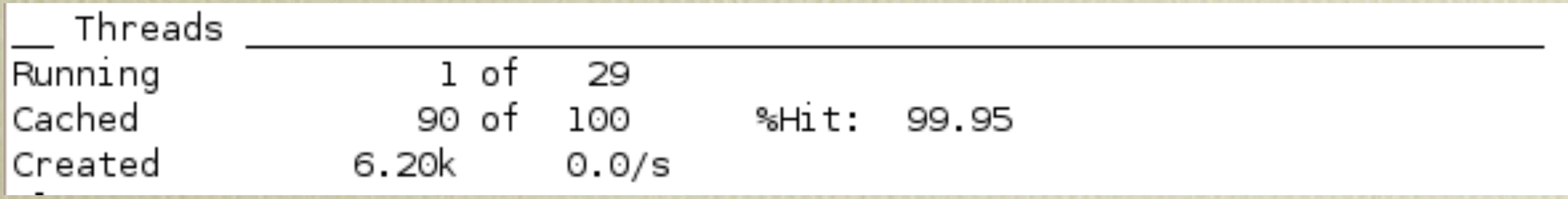
- Created Temp



```
__ Created Temp _________________________________________________
Disk table         2.10M      0.5/s
Table             12.27M      2.7/s     Size: 100.0M
File             146.59k      0.0/s
```

  - Want most as table (RAM)
  - Tunables: tmp_table_size, max_heap_table_size *(per-connection memory allocations)*
  - Queries with blobs and large text fields won't fit in RAM

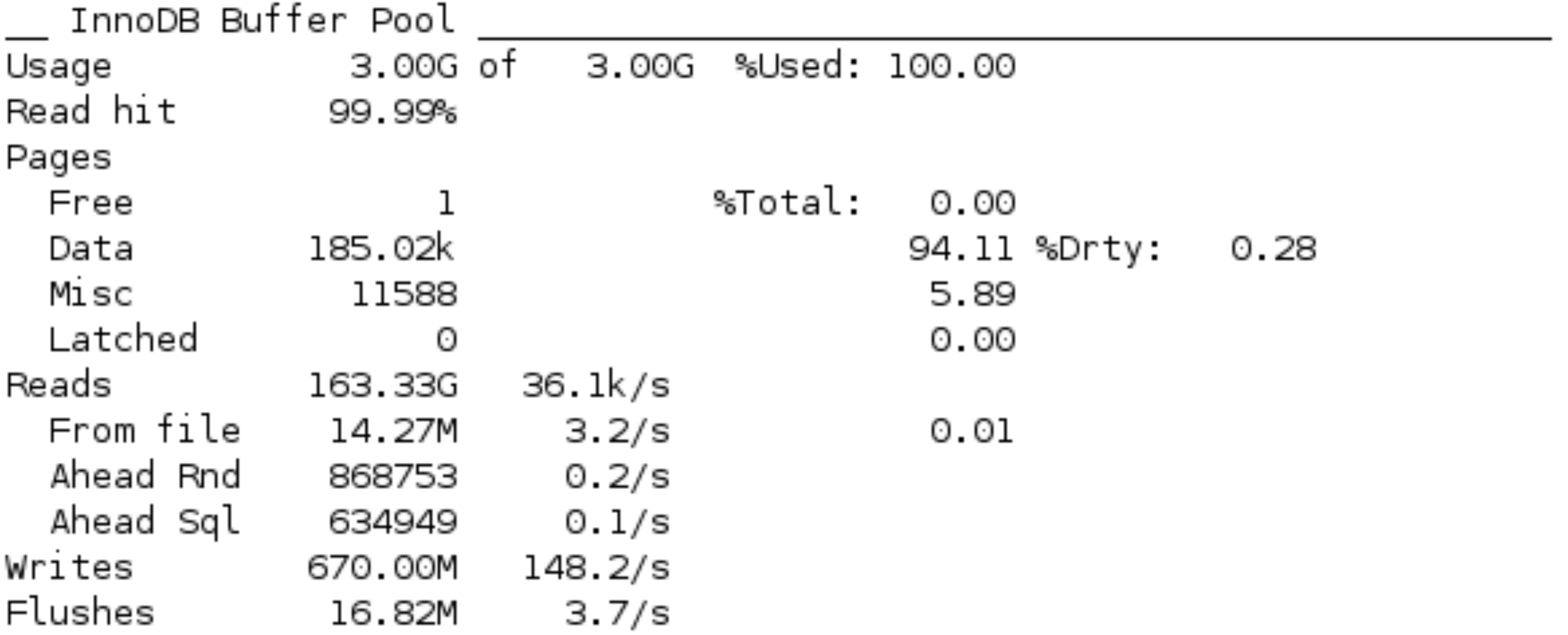
- Threads



```
__ Threads _____________________________________________________
Running                  1 of    29
Cached                  90 of   100      %Hit:   99.95
Created              6.20k     0.0/s
```

Tuning with mysqlreport (7)

- InnoDB Buffer Pool

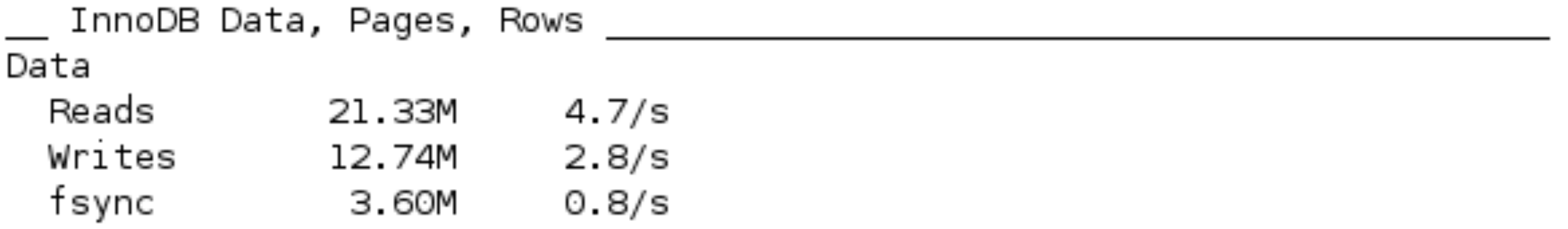


```
__ InnoDB Buffer Pool __________________________________________________
Usage           3.00G of   3.00G  %Used: 100.00
Read hit       99.99%
Pages
  Free              1            %Total:   0.00
  Data        185.02k                     94.11 %Drty:   0.28
  Misc          11588                      5.89
  Latched           0                      0.00
Reads         163.33G   36.1k/s
  From file    14.27M     3.2/s            0.01
  Ahead Rnd    868753     0.2/s
  Ahead Sql    634949     0.1/s
Writes        670.00M   148.2/s
Flushes        16.82M     3.7/s
```

  - Rule of thumb: 70% of available memory
  - Increase: over 80% used, lower than .1 read ratio, significant reads from file
  - Tunable: innodb_buffer_pool_size

Tuning with mysqlreport (8)

- InnoDB Data, Pages, Rows



```
__ InnoDB Data, Pages, Rows ____________________________________________
Data
  Reads         21.33M     4.7/s
  Writes        12.74M     2.8/s
  fsync          3.60M     0.8/s
```

- Ratio of Writes to fsync?
- InnoDB defaults to ACID
- Can you afford data loss?
- Tunable: innodb_flush_log_at_trx_commit (1 = flush every write, 0 = flush every second)

InnoDB Specific Tuning

- Already mentioned buffer_pool size and transaction flushing
- Data File Settings
    - File Per Table
    - Autoextend fragmentation

InnoDB Specific Tuning

- Transaction Isolation
  - Serializable
  - Repeatable-Read (Default)
  - Read-Committed
  - Read-Uncommitted
- MySQL 5.1 improves performance for Read-Committed

InnoDB at the OSL

- The Open Source Lab (Drupal, Apache, Linux, etc) migrated most clients and central servers to InnoDB in early 2007
- Some clients resisted...most didn't notice
- Summer 2007: db3.osuosl.org hardware failure
  - No data loss. No data corruption

InnoDB And Drupal.org

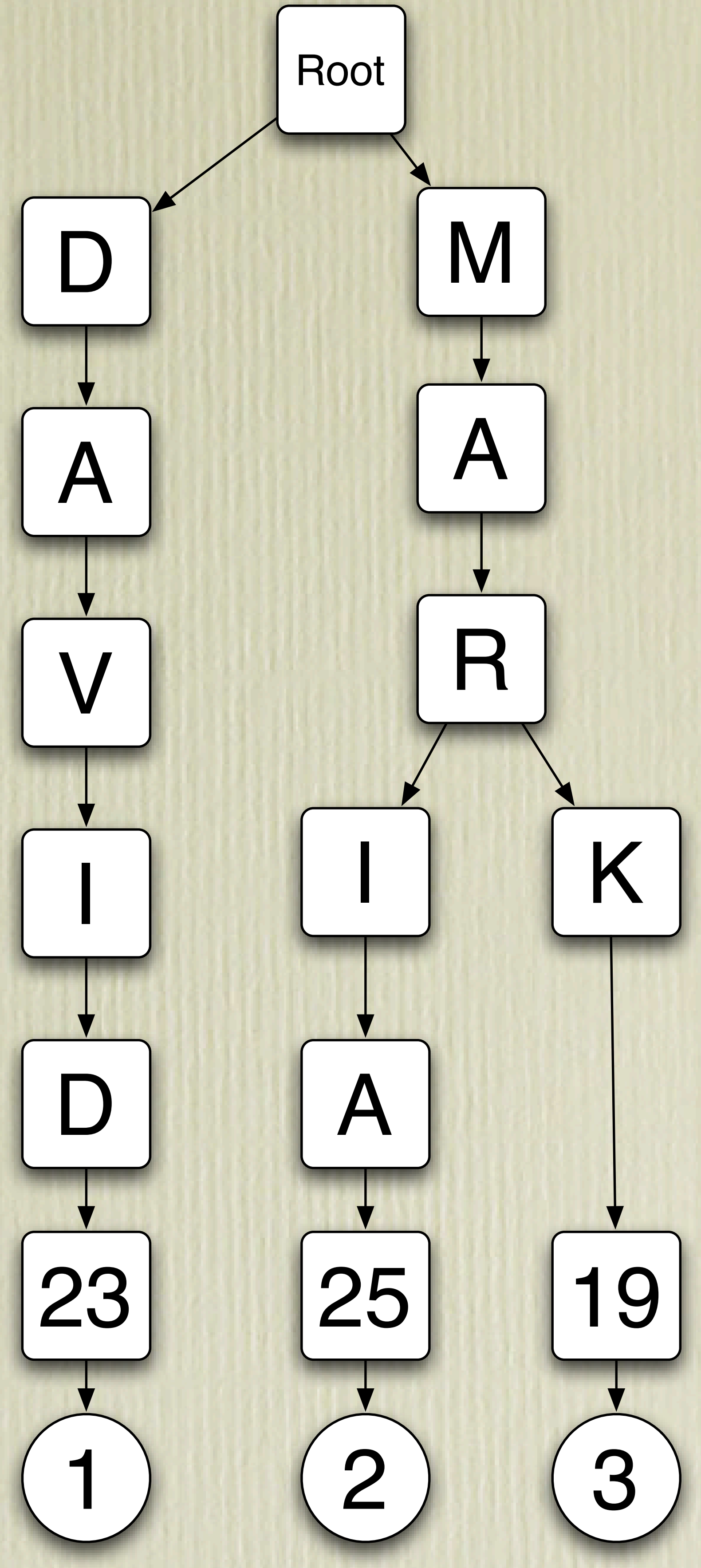
- Was converted Spring 2007
- Issues
  - Memory footprint
  - Clustered indexes
  - Search queries
  - When is concurrency bad?

Anatomy of an Index

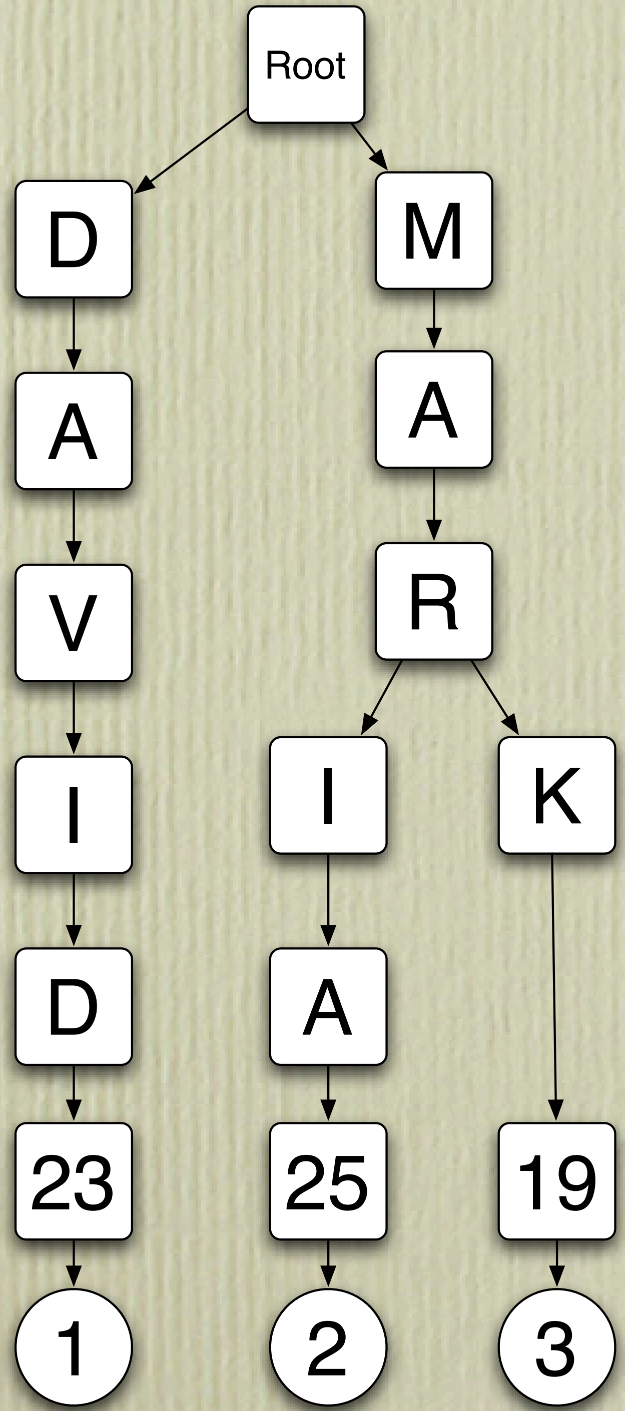
id	*name*	*age*	*evil*
1	David	23	1
2	Maria	25	0
3	Mark	19	0

```
CREATE TABLE IF NOT EXISTS `people` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `name` varchar(32) NOT NULL,
  `age` smallint(5) unsigned NOT NULL,
  `evil` tinyint(1) unsigned NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `name` (`name`,`age`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8
AUTO_INCREMENT=4 ;

INSERT INTO `people` (`id`, `name`, `age`, `evil`)
VALUES
(1, 'David', 23, 0),
(2, 'Maria', 25, 0),
(3, 'Mark', 19, 0);
```



What Does This Index Do?



- Allows the following queries to run efficiently
  - SELECT age FROM {people}
  - SELECT * FROM {people} WHERE name LIKE “Mar%”
  - SELECT * FROM {people} ORDER BY name, age
  - SELECT * FROM {people} WHERE name = “Maria” ORDER BY age

MySQL EXPLAIN

```
mysql> EXPLAIN SELECT * FROM people;
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | people | ALL  | NULL          | NULL | NULL    | NULL |    3 |       |
+----+-------------+--------+------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT age FROM people;
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
| id | select_type | table  | type  | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | people | index | NULL          | name | 100     | NULL |    3 | Using index |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM people WHERE name LIKE "Mar%";
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
| id | select_type | table  | type  | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | people | range | name          | name | 98      | NULL |    1 | Using where |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------------+
1 row in set (0.01 sec)
```

MySQL EXPLAIN (cont’d)

```
mysql> EXPLAIN SELECT * FROM people ORDER BY name, age;
+----+-------------+--------+-------+---------------+------+---------+------+------+-------+
| id | select_type | table  | type  | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | people | index | NULL          | name | 100     | NULL |    3 |       |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM people ORDER BY name, age;
+----+-------------+--------+-------+---------------+------+---------+------+------+-------+
| id | select_type | table  | type  | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | people | index | NULL          | name | 100     | NULL |    3 |       |
+----+-------------+--------+-------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM people WHERE age = 23;
+----+-------------+--------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+--------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | people | ALL  | NULL          | NULL | NULL    | NULL |    3 | Using where |
+----+-------------+--------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

Tips on Index Usage

- MySQL can use an index prefix almost as well as a complete index.
- Order of index consumption (left to right)
  - WHERE clause
  - ORDER BY clause
  - SELECT fields
- If an index contains all the data for a query, MySQL may skip reading the table data.

More Tips

- With certain statistical situations, MySQL may use indexes in non-optimal ways that still surpass table scans in speed.
- All ORDER BY criteria must be either ASC or DESC.
- It's hard to do things fast with OR criteria.
  - Denormalization is often an option.
- Don't depend on MySQL using more than one index per table in a query.
- Don't put WHERE conditions on one table and ORDER BY conditions on another.

Engine: MyISAM

- Locks
  - INSERTs are table-level (with one exception)
  - UPDATEs are table-level
  - DELETEs are table-level
  - SELECTs wait on write locks
- Other notes
  - TRUNCATE is O(1) within MySQL
  - INSERT is non-locking and O(1) within MySQL
    - The primary key is autoincrement
    - No UNIQUE indexes
    - This special case is possible because no checking is necessary for key conflicts with existing rows

Engine: InnoDB

- Locks
    - UPDATEs are row-level
    - INSERTs are table-level
    - DELETEs are row-level
    - SELECTs use MVCC, so they never wait on locks
        - Multi-version concurrency control
        - Keeps revisions around as necessary to satisfy read requests without waiting for concurrent data changes to complete
- Other notes
    - Even though INSERTs are table-level, they can buffered in the transaction system

Engine: Quantum

- Locks
  - Qubit-level locking
- Other notes
  - Highly concurrent
  - Zero-latency replication
    (Compile option --with-plugins=entanglement)
  - Experimental
  - Can be monitored using qtop
  - Display log files using cat schroedinger.log

Replication

- When it’s the answer
  - You have SELECT queries that don't require access to perfectly fresh data
  - You want a server to run backups from without burdening the master
  - You want a hot standby in case of failure of the master
- When it’s not the answer
  - If you have high levels of lock contention and you're looking for performance benefits
    - Replication simply runs the data-modification queries from the master server on the slaves
    - So, the exact same locking behavior happens on the slaves

Data Caching

memcache, APC, Squid

Squid

- What is Squid?
  - Caching Proxy
  - Moonlights as a Reverse Proxy, a.k.a. HTTP accelerator
- Why Squid?
  - Can (Possibly) Cache Dynamic Pages
  - Caches Static Content
  - Serves as a connection pool
  - Simple and easy to setup

Squid-At the OSL

- The Open Source Lab And Squid
  - Big Squid users
  - Introduced into the lab by Scott Kveton
  - We started with little funding. Squid was the only way our servers could survive slashdot/digg for quite awhile
- Sites that depend on Squid for high-traffic stability
  - osuosl.org
  - mozillazine.org
  - Every host on our shared webserver (60 community sites)

Squid-Drupal.org

- Deployed during summer 2007 by Eric Searcy
  - Caches most static files, allowing us to not have a static.drupal.org host
  - Was a “piece of the puzzle”, but didn't solve our scaling issues
  - Isn't a silver bullet for Drupal deployments

Squid-Limitations with Drupal

- Drupal sends inaccurate headers
  - Cache-Control headers tell Squid what to cache and for how long
  - Drupal for the most part tells it “not to”
- Static vs Dynamic Cache Rates Due to this
  - Static Files: 318828/11710 Hits/Misses = 96.5%
  - Dynamic Pages: 34551/420339 Hits/Misses = 7%

Squid - As HTTP Accelerator

- Squid Configuration
  - Very well documented configuration file
  - Simple ACL's based on Ports, Header Information
  - Your webserver (Apache) is your “cache_peer”
  - You listen on “http_port”, with the accel or the vhost option
- Apache configuration
  - Apache listens on port 8080 for incoming connections from Squid
  - Tweak KeepAlive to take advantage of persistent connections from Squid

Squid-Issues

- Initial configuration makes it seem like a “fire and forget” solution...
- Not so much
  - Serving stale files
  - Set-Cookie header caching
  - “Zero Sized Reply”
  - Scalability (lack of)
  - Differentiation between logged in and anonymous users hitting the cache
- Varnish: http://varnish.projects.linpro.no/

Static Page Cache

- Several contrib modules utilize Drupal’s “Early page cache”
  - Modules:
    - Boost – http://drupal.org/project/boost
    - Fastpath FSCache http://drupal.org/project/fastpath_fscache
  - Pros:
    - Very fast
    - No DB overhead (uses static html)
  - Cons:
    - Not good for dynamic sites with a lot of fresh content

Drupal 5.x – The Good ☺

- Current 5.x implementation
  - Pluggable ‘cache.inc’
    - Easily
      r
      e
      place cache.inc with any type of cache you can dream up. (File, Memory, etc)
  - Caches full pages for Anonymous Users
    - woot!
  - Allows for minimum cache lifetime
    - woot!

Drupal 6.x – The future is now!

- New 6.x Caching features
  - Block-level caching!

Example:

```
<?php
/**
* Implementation of hook_block().
*/
function mymodule_block($op = 'list', $delta = 0, $edit = array()) {
if ($op == 'list') {
$blocks[0]['info'] = t(’Super Rad Block!');
$blocks[0]['cache'] = BLOCK_CACHE_PER_PAGE | BLOCK_CACHE_PER_ROLE;
return $blocks;
}
}
?>
```

Drupal 6.x – The future is now!

- New 6.x Caching features
  - Block-level caching!

    Modes available:

  - BLOCK_CACHE_PER_ROLE (default)
  - BLOCK_CACHE_PER_USER
  - BLOCK_CACHE_PER_PAGE
  - BLOCK_CACHE_GLOBAL
  - BLOCK_NO_CACHE

Drupal 6.x – The future is now!

- New 6.x Caching features
  - Core cache serialization!
    - You no longer have to worry about serializing your data when using the Drupal cache.

Drupal 6.x – The future is now!

- New 6.x Caching features
  - Core cache serialization!
    - You no longer have to worry about serializing your data when using the Drupal cache.
  - A clear cache button!
    - “...and there was much rejoicing....yay.”

How to cache

- Using caching in your code

Example:

```
function tagadelic_helper_tag_listing() {
// Lets first try to get the page from our cache.
$cache = cache_get('page:tagadelic_helper_tag_listing');
if ($cache) {
$output = $cache->data;
}
else {
// If we can't get it from cache then we will run the query.
$limit = 100;
// BFQ - This takes a long time to process
$result['popular'] = db_query("SELECT COUNT(*) AS count, d.tid, d.name, d.vid …”, $limit);

$tags['popular'] = tagadelic_build_weighted_tags($result['popular'], 3);
$tags['popular'] = tagadelic_sort_tags($tags['popular']);

// Last theme the content into a string ready for output.
$output = theme('tagadelic_helper_tag_listing', $tags, 'taxonomy/term/', 'filter0');

// Cache page for one hour.
cache_set('page:tagadelic_helper_tag_listing', 'cache', $output, time() + 3600);
}
return $output;
}
```

Pluggable Cache

- Drupal offers “pluggable” cache.inc

Pluggable Cache

- Drupal offers “pluggable” cache.inc
  - Modules that implement “pluggable” cache.inc:
  - APC – http://drupal.org/project/apc
  - Xcache – http://drupal.org/project/xcache
  - Memcache – http://drupal.org/project/memcache

Pluggable Cache

- APC
  - Pros:
    - Uses very fast web server memory (no network delay)
    - Extremely fast
    - Simple to implement, no complex setup

Pluggable Cache

- APC
  - Cons:
    - No shared memory
      - Not good for multiple web servers due to cache redundancy and wasted memory
    - Has to
      k
      eep track of each bin itself which slows it down a bit. (working on fix for this)
    - Still beta quality code.

Pluggable Cache

- XCache
  - Pros:
    - Uses very fast web server memory (no network delay)
    - Fast
    - Multithread safe
    - Easy to implement

Pluggable Cache

- XCache
  - Cons:
    - – No shared memory
      - » Not good for multiple web servers due to cache redundancy and wasted memory
    - – Has to
      k
      eep track of each bin itself which slows it down a bit. (working on fix for this)
    - – Still beta quality code.
    - – No support for complex data types, you have to serialize the data yourself

Pluggable Cache

- Memcache
  - Pros:
    - Stable code (Drupal module used on FastCompany, LifetimeTV, NowPublic.com and lots

      more

      ,

      memcache daemon used on Facebook, Slashdot, Wikipedia, Livejournal, etc)
    - Shared memory – No cache duplication with multiple web servers
    - Setup multiple instances, so entire cache flushes clear only their own bins
    - Very fast hash lookups
    - Takes care of serializing the data for you

Pluggable Cache

- Memcache
  - Cons:
    - Uses network to connect, so some network delay can occur if not running locally
    - Complex to setup and manage

Load Testing

Evaluating the Entire Stack

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

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

Front End Performance

- Another bottleneck: CSS, JS and graphic files
- Can be > 90% of the load time

- Components: CSS, JS, UI graphic files
- Sometimes > 100 components on a page

YSlow

- Firefox addon for measuring front end performance
- Inspects website and looks for:
    - Amount of components == HTTP requests
    - Use of a CDN
    - HTTP headers (Cachability, ETags)
    - Use of GZip
    - Location of components on the page
    - Redirects, DNS lookups, CSS exps, duplicate scripts, ...

Drupal Modules

- cdn.module
  - Pushes files to a CDN (e.g. CacheFly)
- sf_cache.module
  - Intelligent CSS/JS aggregation in bundles
  - Separate hosts for CSS, JS, UI graphics
- "High Performance Web Sites" (0596529307) by Steve Souders (affiliated with YSlow)

Questions?

Supplemental

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
```

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
```

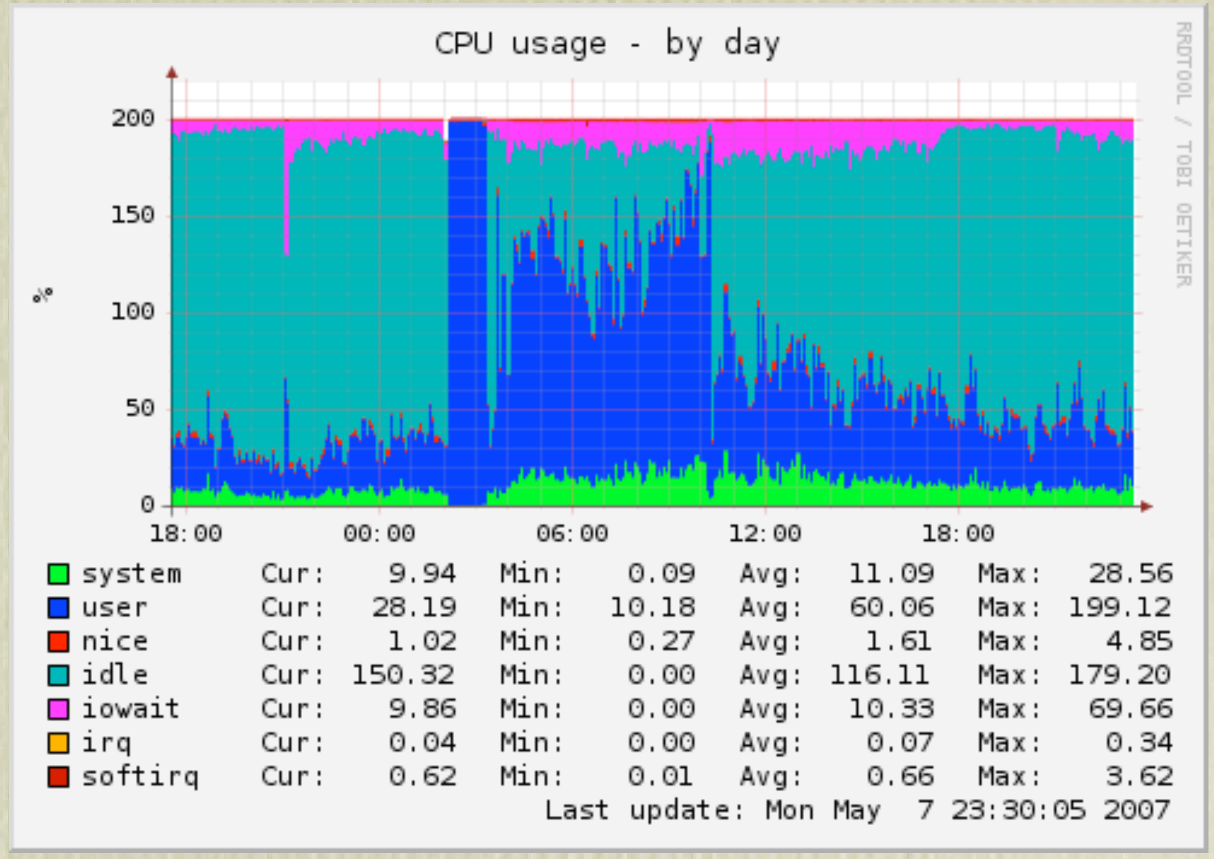
CPU 100%

- What was it?

- eAccelerator (svn303 + PHP 5)

- Attempt to get over PHP crashes

- Note CPU utilization (100%, then high, then
  drop
  p
  ed low when good version used)



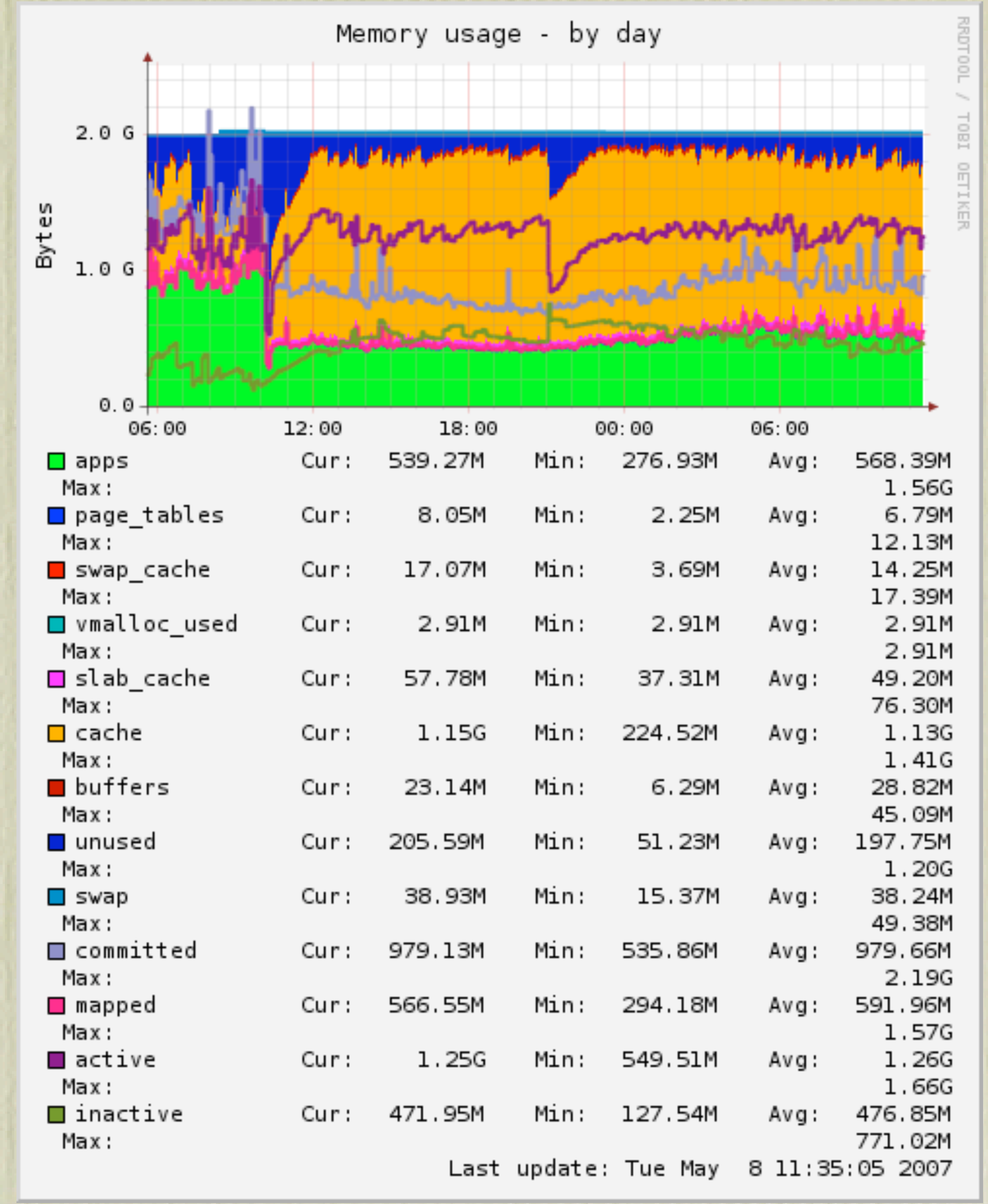
```
🟩 system    Cur:   9.94  Min:  0.09  Avg:  11.09  Max:  28.56
🟦 user      Cur:  28.19  Min: 10.18  Avg:  60.06  Max: 199.12
🟥 nice      Cur:   1.02  Min:  0.27  Avg:   1.61  Max:   4.85
🟦 idle      Cur: 150.32  Min:  0.00  Avg: 116.11  Max: 179.20
🟪 iowait    Cur:   9.86  Min:  0.00  Avg:  10.33  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