

# Database Optimizations

- Jeremy Andrews

# Database Optimizations

- Jeremy Andrews, Drupal background:



- statistics.module
- throttle.module
- pager.inc
- file cache
- 
- banner.module/ ad.module
- spam.module
- dba.module

# Database Optimizations

- Jeremy Andrews, Drupal background:
  - installer work
  - improved cache system
  - built highly available ASP infrastructure
  - developed ASP provisioning system



# Overview

- MySQL Monitoring
- MySQL Storage Engines
- Caches
- Physical Optimizations
- SQL Level Optimizations
- Replication

# MySQL Monitoring

- SHOW PROCESSLIST
- SHOW STATUS
- mytop
- mysqlreport
- Drupal SQL report
- top M, vmstat, iostat, sar
- *slow query* log
- *no index* log

# MySQL Storage Engines

- MyISAM
- InnoDB
- Archive
- Falcon

# MySQL Storage Engines

## MyISAM

- MySQL default storage engine
- Tables stored in three files:
  - table.frm (format)
  - table.MYD (MYData)
  - table.MYI (MYIndex)
- Can be very fast (not ACID compliant)
- *myisamchk* (`--myisam-recover`)

# MySQL Storage Engines

## MyISAM

- Limitations
  - not transactional
  - relies on OS I/O cache for reads and writes
  - performance lost on massive tables

# MySQL Storage Engines

## MyISAM

- Tuning

- *key\_buffer\_size*: set to 30-40% of available RAM, caches indexes. Data is cached by OS, so leave RAM for that. (Even if not using MyISAM tables, allocate some space for temporary tables.)
- *table\_cache*: opening a table includes modifying MYI header to mark table as in use. Best to set large enough to keep all tables open. Each connection requires new entry, so busier sites need to increase this.
- *thread\_cache*: Monitor `Threads_Created`, if growing quickly increase this value. Start around 16. (`SHOW STATUS LIKE 'Thread%'`)
- *query\_cache\_size*: Monitor usage and hit ratio. Start around 32M to 512M and adjust based on monitoring. (`SHOW STATUS LIKE 'Qcache%'`)
- *max\_connections*, *wait\_timeout*, *connect\_timeout* (Linux can support ~500-1000 simultaneous connections, depending on RAM and resources. MySQL defaults to 100 connections. Idle connections default to 8 hours to timeout. See `PROCESSLIST`.)
- Monitor, monitor, monitor. Websites are not static.

# MySQL Storage Engines

## InnoDB

- Transactional tables supporting rollback and commit capabilities (ACID = Atomicity, Consistency, Isolation, Durability)
- Supports foreign key restraints
- Can be significantly faster than MyISAM (ie, buffered indexes are hashed, ultra fast lookup)
- Supports row-level-locking
- Automatic crash recovery
- Support for hot backups (commercial tool)
- Fast table recovery

# MySQL Storage Engines

## InnoDB

- Limitations

- Data physically stored in key order, versus the order added such as MyISAM (potentially increases performance of reads and decreases performance of writes)
- True ACID compliance requires flush to disk at least once per transaction, equates to maximum of roughly 200 update transactions per second (can be optimized away with hardware, ie use a disk controller with write caching and battery backup)
- AUTOCOMMIT and LOCK TABLES issue.
- Tables take 2-3 times more space (partially because index are not compressed as with MyISAM)
- Tuning more important than with MyISAM (for the advanced caching and synchronous IO support)

# MySQL Storage Engines

## InnoDB

- Limitations

- Live backups require commercial “*InnoDB Hot Backup*” tool (\$470/year, or \$1190 for lifetime)
- Oracle acquired Innobase Oy October 2005

# MySQL Storage Engines

## InnoDB

- Tuning

- *innodb\_buffer\_pool\_size*: Set to 70-80% of available memory, though for smaller non-growing datasets won't need as much. Monitor overall memory usage, don't make so big MySQL swaps.
- *innodb\_log\_file\_size*: Larger size generally increases performance, but increases recovery times. Start from 64M to 512M and monitor.
- *innodb\_log\_buffer\_size*: Flushed every second, so don't waste too much RAM on it. Important for sites seeing lots of UPDATE spikes.
- *innodb\_flush\_logs\_at\_trx\_commit*: Value 0 means flush only once every second. Value 1 (default) means flush each transaction. Value 2 means do not flush to disk, flush only to OS cache. With 0, MySQL crash can cause up to 1 second loss of data. With 2, requires OS crash to lose data.

# MySQL Storage Engines

## Falcon

- **Features:**

- Multi Version Concurrency Control (MVCC) eliminates need for table level and row level locking.
- Optimized for multi-core CPUs and 64-bit architectures
- Transaction-safe, ACID compliant
- B-Tree indexes
- Data is stored in compressed format on disk
- Intelligent disk management, auto space reclamation
- Intelligent data and index caching
- Written by Jim Starkey based on Netrastructure code base, owned by MySQL AB

- **Alpha** (beta expected Q3)

- **Current benchmarks suggest** its got a long ways to go, lagging well behind MyISAM and InnoDB in performance.

# MySQL Storage Engines

## Archive

- **Overview:**

- Allows live compression (versus offline compression with MyISAM using *myisampack*), allows inserts into compressed tables (MyISAM packed tables are read only), provides better compression.
- Only supports SELECT and INSERT, easy to audit.
- Does not support indexes.
- Not in community edition of MySQL, compile from source

# Caches

- Query Cache
- File cache / memcached

# Caches

## Query Cache

- Disabled by default, enable in my.cnf

```
[mysqld]
query_cache_size=64M
```
- Size of cache depends on use cases.
  - Too small a cache can result in too frequent of invalidations, reducing the query cache hit rate
  - Large cache with infrequent invalidations can result in MySQL stalls.
- `SHOW VARIABLES LIKE 'query%';`
- `SHOW STATUS LIKE 'Qcache%';`
- Query Cache is byte sensitive (“*SELECT \* FROM table*” is different than “*select \* from table*” and from “*SELECT \* FROM table*”)

# Caches

## File / Memory Cache

- Pluggable *cache.inc*
- File Cache / `fastpath_fcache`
- Memcached (Robert)

# Physical Optimizations

## Scaling hardware:

### 1. RAM:

1. ideally all keys should fit in RAM
2. watch total memory usage, swap kills

### 2. Drives:

1. as table size increases, number of seeks increases. If tables get REALLY big or if you don't have enough RAM, bound by disk seek speeds
2. Mount filesystem with *async* and *noatime* options.
3. RAID (striping for performance, mirroring for reliability, ideally only mirror important data, only stripe ie temp tables)
4. Spread tables over multiple drives (can be done with symbolic links), if using replication put binary logs on own drive.

### 3. CPU:

1. MySQL uses 64-bit integers internally, so 64-bit CPU helpful.
2. Monitor CPU usage (top, vmstat, sar)

# Physical Optimizations

- Ethernet speeds impact remote databases.
  - Gigabit ethernet recommended for reducing latency more than for increasing bandwidth.
- If scaling single large site, possible to use database prefixing to split tables amongst multiple servers.

# SQL Optimizations

- Monitor Slow Query Log and No Index Log
- Enable devel module, monitor query times
- Use EXPLAIN to understand slow queries
- Indexes
  - <http://dev.mysql.com/doc/refman/5.0/en/mysql-indexes.html>

# Replication

- How it works
- Backups
- Performance / Load Balancing
- High Availability

# Replication

## How it works

- **Binary Logs**

- contain all queries that update or could update data (ie *INSERT*, *UPDATE*, *CREATE*, *DROP*, *DELETE*, but not *SELECT* or *SHOW*)
- used for replication, as well as for incremental backups
- slows down performance about 1%, benefits outweigh costs
- *max\_binlog\_size*, *RESET MASTER*, *PURGE MASTER LOGS*

- **Master** (read/write) **versus Slave** (read)

- **Configuring**

- *GRANT REPLICATION SLAVE ON \*.\* TO 'replicant'@'slave IP' IDENTIFIED BY 'password';*
- my.cnf:
  - *server-id = 1* (must be unique on each host)
  - *log-bin = /var/log/mysql/hostname-bin.log* (required on master, in HA configuration also on slave)

# Replication

## How it works

- **Configuring (cont.)**

- backup Master server

- *mysql> FLUSH TABLES WITH READ LOCK*

- *mysql> RESET MASTER*

- *mysql> SHOW MASTER STATUS* (note File and Position)

- *\$ mysqldump -user=USERNAME -password=PASSWORD --extended-insert --all-databases --master-data > master\_backup.sql*

- *mysql> UNLOCK TABLES*

- import Master backup into slave

- *\$ mysql -user=USERNAME -password=PASSWORD < master\_backup.sql*

- start replication on Slave

- *mysql> CHANGE MASTER TO MASTER\_HOST='master IP', MASTER\_USER='replicant', MASTER\_PASSWORD='password', MASTER\_LOG\_FILE='log\_file', MASTER\_LOG\_POS='position';*

- *mysql> START SLAVE*

- *mysql> SHOW SLAVE STATUS*

# Replication

## How it works

- **Verify it's working**
  - “Binlog Dump” thread on master
    - SHOW PROCESSLIST
  - Check slave status
    - SHOW SLAVE STATUS
  - Two threads on slave
    - SHOW PROCESSLIST
  - Review MySQL error log

# Replication Backups

- Complete live copy of all data
- Full backups with STOP SLAVE
- No overhead on MASTER server
- Incremental backups with binary logs

# Replication

## Performance / Load Balancing

- MASTER can handle reads and writes
- SLAVE can handle reads
- Hack `_db_query()`, use `preg_match` to detect reads (ie `SELECT`, `SHOW`) and direct these queries to SLAVE servers. Direct all other queries to MASTER.

# Replication

## High Availability

- **MASTER <-> MASTER configuration.**
  - not officially supported by MySQL
  - not officially supported by Drupal core
  - very easy to break and end up with divergent masters that are not keeping in sync
- **Master -> Slave, with automatic failover**
  - uCarp, or similar technologies
  - enable Binary logging on MASTER and SLAVE
  - SLAVE becomes MASTER on first write
  - Use full backup of SLAVE after failover to set up old MASTER as new SLAVE

# Questions?