Care and Feeding of a MySQL Database for Linux Administrators

Dave Stokes
MySQL Community Manager
David.Stokes@Oracle.Com

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Simple Introduction

This is a general introduction to running a MySQL database server(s) for Linux Administrator

Simple Introduction

This is a general introduction to running a MySQL database server(s) for Linux Administrator

Database servers have needs different than SMTP, HTTP, or other servers

Simple Introduction

This is a general introduction to running a MySQL database server(s) for Linux Administrator

Database servers have needs different than SMTP, HTTP, or other servers

Hardware choices are critical! (do not go cheap)

Simple Introduction

This is a general introduction to running a MySQL database server(s) for Linux Administrator

Database servers have needs different than SMTP, HTTP, or other servers

Hardware choices are critical! (do not go cheap)

Tuning to 80% efficiency is relatively easy

Simple Introduction

This is a general introduction to running a MySQL database server(s) for Linux Administrator

Database servers have needs different than SMTP, HTTP, or other servers

Hardware choices are critical! (do not go cheap)

Tuning to 80% efficiency is relatively easy (last 20% is tricky)

Session Overview

1. Basics of a database server
2. Hardware
3. MySQL Configuration
4. Monitoring Operations
5. Backups
6. Replication
7. Indexes
8. Tuning
9. Q/A

How does a Database server work

<u>Client</u> <u>Server</u>

SELECT phone

FROM friends

WHERE name = ‘Joe’;

How does a Database server work

<u>Client</u>

SELECT phone

FROM friends

WHERE name = 'Joe';

<u>Server</u>

PARSE

find Joe in friends table in memory

return phone

How does a Database server work

<u>Client</u>

SELECT phone

FROM friends

WHERE name = ‘Joe’;

. . .

Process phone data

<u>Server</u>

PARSE

find Joe in friends table in memory

return phone

How does a Database server work

<u>Client</u>

SELECT phone

FROM friends

WHERE name = ‘Joe’;

. . .

Process phone data

<u>Server</u>

PARSE

find Joe in friends table in memory

return phone

What was that about memory???

ORACLE

Rule #1

- Databases love data in memory

Rule #1

- Databases love data in memory

Corollary #1 – getting data in/out of memory will cause you nightmares!

What if it is not in memory?

MySQL

Please give me the data from the city table

OS

What if it is not in memory?

MySQL

Please give me the data from the city table

→

OS

Get inode

What if it is not in memory?

<u>MySQL</u>

Please give me the data from the city table

→

<u>OS</u>

Get inode

Ask disk for data

What if it is not in memory?

MySQL

Please give me the data from the city table

OS

Get inode

Ask disk for data

Get data into buffer

What if it is not in memory?

<u>MySQL</u>

Please give me the data from the city table →

<u>OS</u>

Get inode

Ask disk for data

Get data into buffer

← Hand buffer off

Load data into memory

What if it is not in memory?

<u>MySQL</u>

Please give me the data from the city table

$\Rightarrow$

<u>OS</u>

Get inode

Ask disk for data

Get data into buffer

Hand buffer off

$\Leftarrow$

Load data i

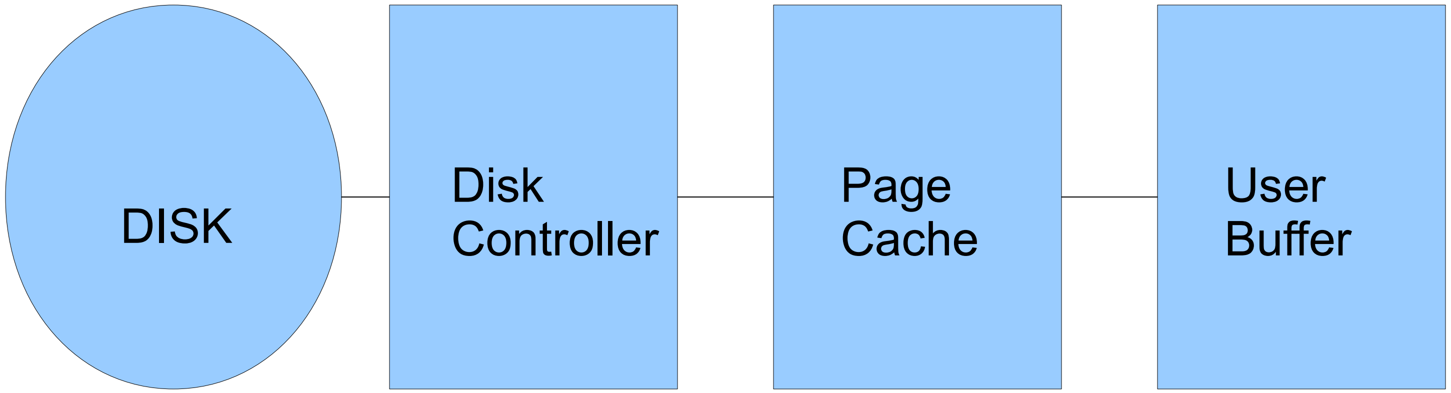
Much longer than just reading from memory

Rule #2

Databases have to do unpredictable queries, random I/O, and sequential scans so slow I/O kills performance

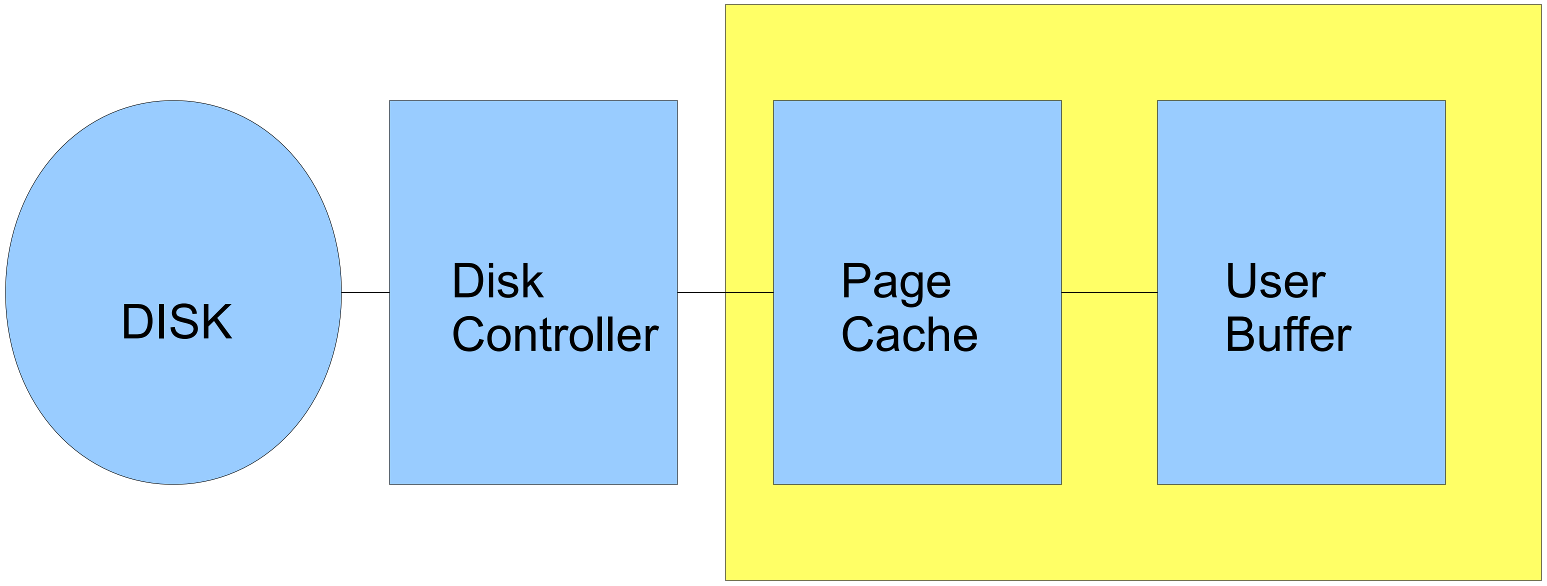
Buffers

What happens when you read a file into memory



Buffers

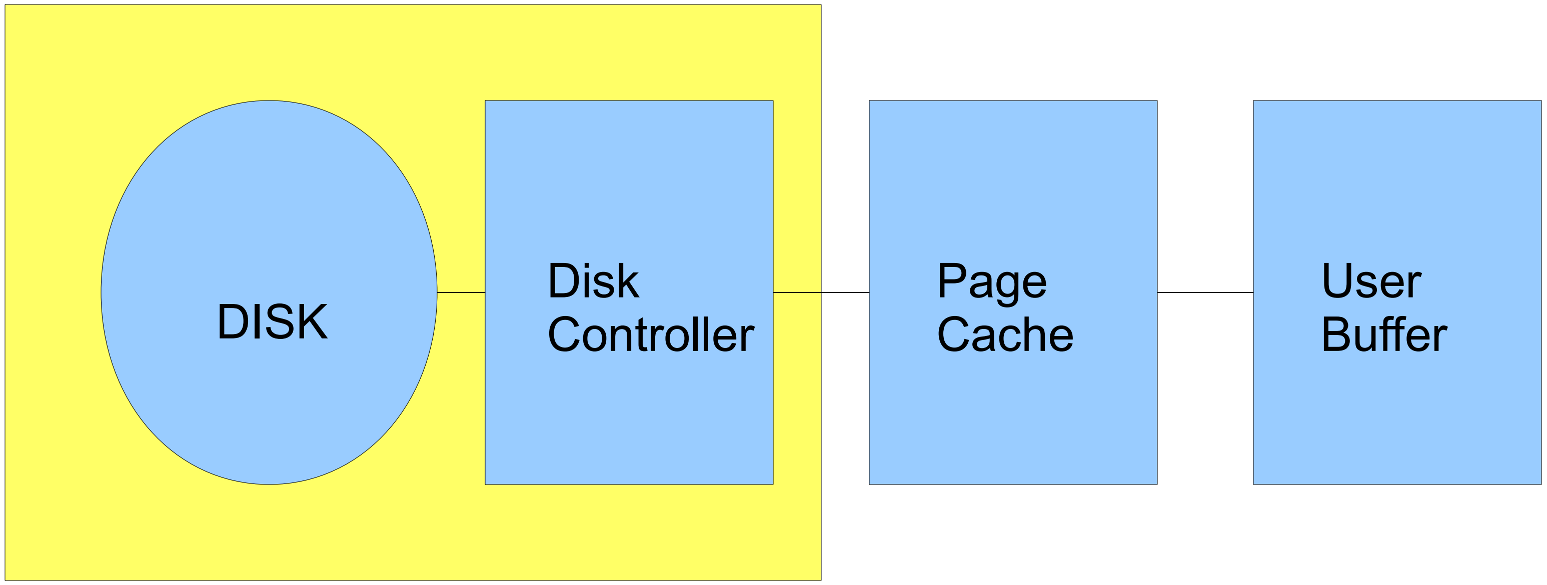
Memory Lookup – 100 nanoseconds, 12GB/sec



Buffers

Memory Lookup – 100 nanoseconds, 12GB/sec

DISK seek – 10 milliseconds, 760MB/sec



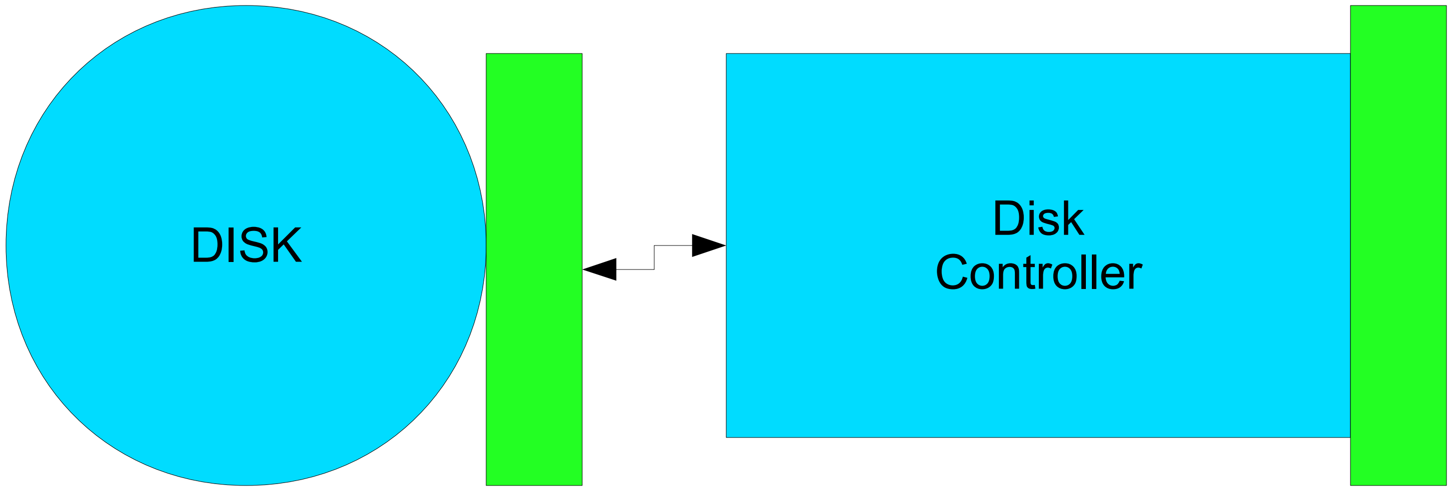
Disk Reads

A disk read is 100,000 slower than reading memory

- For comparison
  - 100,000 minutes is about 69.5 days
  - 100,000 feet is about 19 miles
  - 100,000 kilometers is 15.7 times around Earth's equator

Cache

Write*back* and write*through* caches

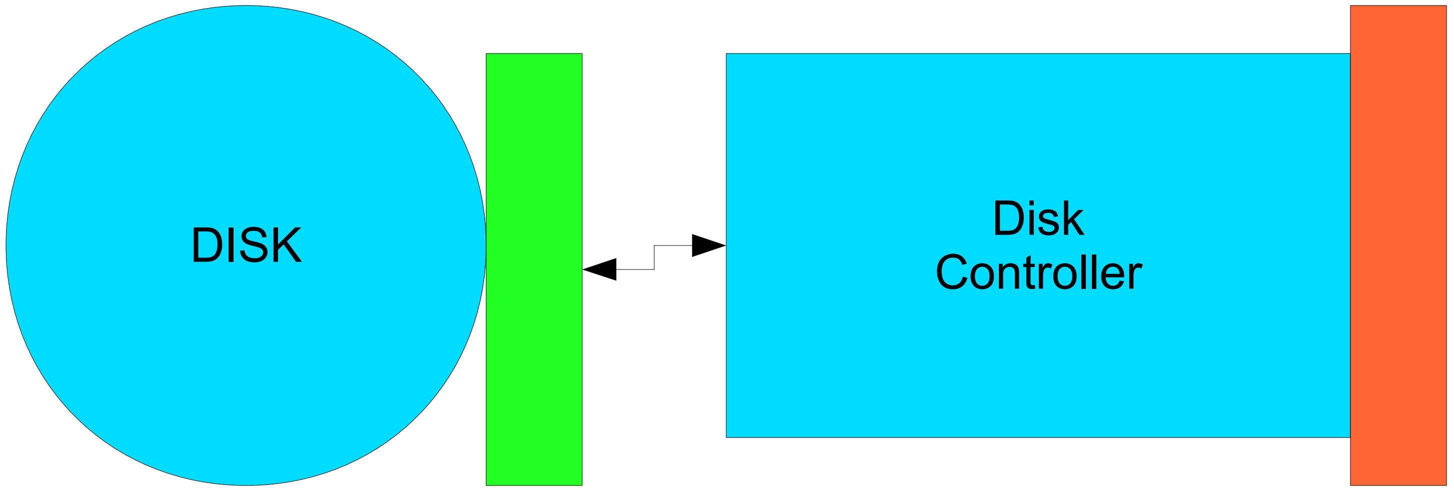


Writethrough on disk

Writethrough on controller

Cache

Recommended setup



Writethrough on disk

Writeback on controller

SSD

Use standard RAID controller

- SSD as writeback cache
- Battery backed
- $2.5/GB verses .075/GB
- Very fast seek time
- Density

Hardware recommendations

1. Memory – lots of it, ecc

Hardware recommendations

1. Memory – lots of it, ecc
2. DISKs – more spindles, high speed, fast controllers, RAID 10, write back cache, and XFS/ZFS/ext4 not ext2/3

Hardware recommendations

1. Memory – lots of it, ecc
2. DISKs – more spindles, high speed, fast controllers, RAID 10, write back cache, and XFS/ZFS/ext4 not ext2/3
3. Write-through caches with battery backup units for disks must be monitored, and have life span much longer than planned outages

Hardware recommendations

1. Memory – lots of it, ecc
2. DISKs – more spindles, high speed, fast controllers, RAID 10, write back cache, and XFS/ZFS/ext4 not ext2/3
3. Write-through caches with battery backup units for disks must be monitored, and have life span much longer than planned outages
4. CPUs, Core less important (spend money on Memory and IO)

Quick Security Warning!

MySQL login security is primitive.

- Database *mysql* has *users* table
- 'jsmith'@'co.com' or 'fred'@'10.10.%'
  - Matches *host*, then *user*, then *password* fields
    - Be explicit
- Proxy and Pluggable logins on the way

MySQL privilege security

- GRANT functions or access
- Can get Byzantine quickly
- Use a GUI

When *root* is the root of your *root* problem OR stingy is your best friend

Linux server has root

- Highly privileged
- Dangerous

MySQL daemon has root

- Highly privileged
- Dangerous
- Understand MySQL priv system and be stingy
- Proxy and plug-in adapters soon
- Really limit access, not just play at it

Linux root is not the same as MySQL root
but **both can be dangerous to you!!!**

ORACLE

Installation

1. Use pre built MySQL packages

Installation

1. Use prebuilt MySQL packages
2. Don’t double up with other services

Installation

1. Use prebuilt MySQL packages
2. Don’t double up with other services
3. Supplied configuration files are OLD!

Installation

1. Use prebuilt MySQL packages
2. Don’t double up with other services
3. Supplied configuration files are OLD!
4. Move logs to different disk than data

Installation

1. Use prebuilt packages
2. Don’t double up with other services
3. Supplied configuration files are 𝔒𝔏𝔇!
4. Move logs to different disk than data
5. Spread data over different drives

Installation

1. Use prebuilt packages
2. Don’t double up with other services
3. Supplied configuration files are OLD!
4. Move logs to different disk than data
5. Spread data over different drives
6. Backups are necessary – and practice recovery!

Monitoring Operations

1. Slow query log -- not all long running queries are bad

Monitoring Operations

1. Slow query log -- not all long running queries are bad
2. Log queries not using indexes

Monitoring Operations

1. Slow query log -- not all long running queries are bad
2. Log queries not using indexes
3. Use monitoring software – MEM, Innotop, Cacti, Munin, Nagios, etc – and pay attention to it

Monitor!!!

Monitoring Operations

1. Slow query log -- not all long running queries are bad
2. Log queries not using indexes
3. Use monitoring software – MEM, Innotop, Cacti, Munin, Nagios, etc – and pay attention to it
4. More in tuning ….

Backups

Backups are usually some sort of disk snap shot or serializing data to a file

Backups

Backups are usually some sort of disk snap shot or serializing data to a file

The more the better but you need to know steps to recover dropped table, lost databases, or mangled data

Backups

Backups are usually some sort of disk snap shot or serializing data to a file

The more the better but you need to know steps to recover dropped table, lost databases, or mangled data.

Use data replication to a slave and then backup slave

Backups

Backups are usually some sort of disk snap shot or serializing data to a file

The more the better but you need to know steps to recover dropped table, lost databases, or mangled data.

Use data replication to a slave and then backup slave

Be paranoid!!!!!

Replication

Replication for MySQL is the binary log for the master being copied to a slave. The slave then updates its copy of the data

Replication

Replication for MySQL is the binary log for the master being copied to a slave. The slave then updates its copy of the data

Two types:

1. Asynchronous – server does not check changes sent to slave before proceeding

Replication

Replication for MySQL is the binary log for the master being copied to a slave. The slave then updates its copy of the data

Two types:

1. Asynchronous – server does not check changes sent to slave before proceeding
2. Semi Synchronous – server checks that slave received changes before proceeding

Replication -- threads

Currently single threaded – 5.6 will fix that

Replication -- network

Network latency will affect MySQL replication. So plan network topology to minimize bandwidth competition with other systems/services.

Replication -- network

Network latency will affect MySQL replication. So plan network topology to minimize bandwidth competition with other systems/services.

Slaves do not need to be as fast as the master but try to keep things reasonably close

Replication -- network

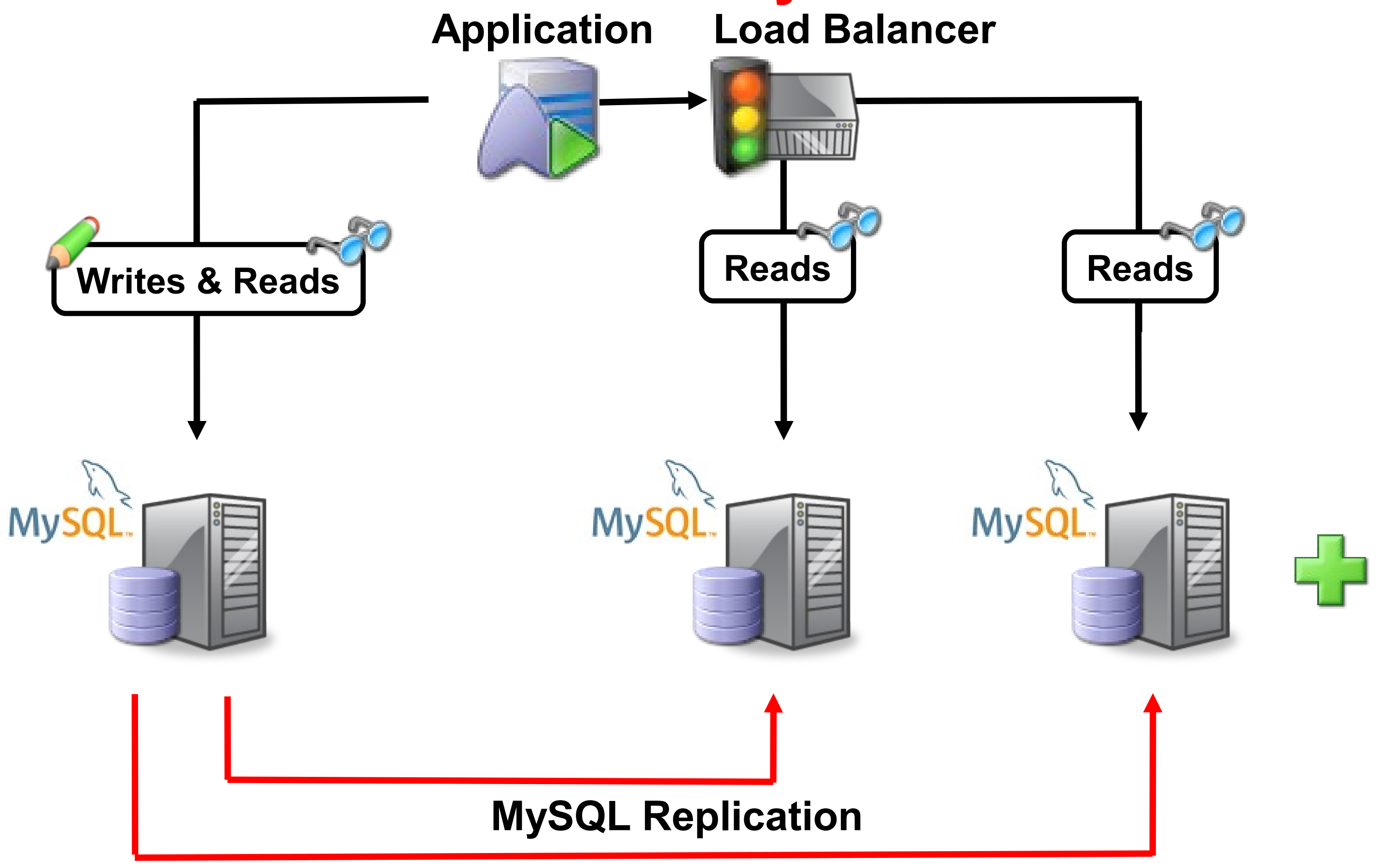
Network latency will affect MySQL replication. So plan network topology to minimize bandwidth competition with other systems/services.

Slaves do not need to be as fast as the master but try to keep things reasonably close

Do not have to replicate all tables/databases to all slaves. Cut down on traffic by replicating what is needed!

MySQL Database

Replication Enables Scalability

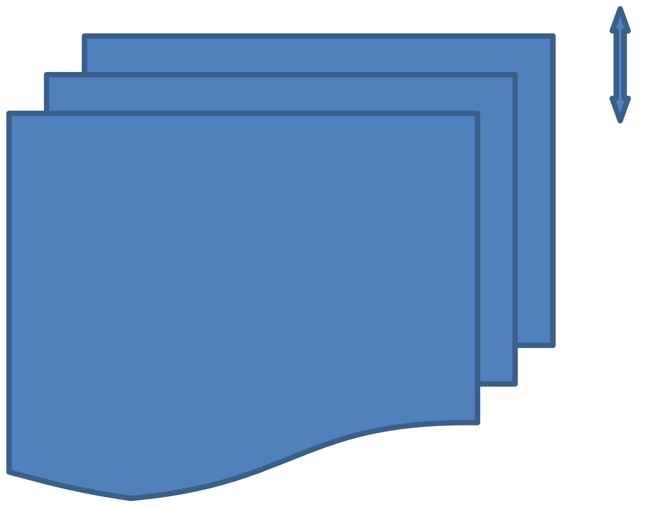


- Write to one master
- Read from many slaves, easily add more as needed
- Perfect for read/write intensive apps

Indexes are good

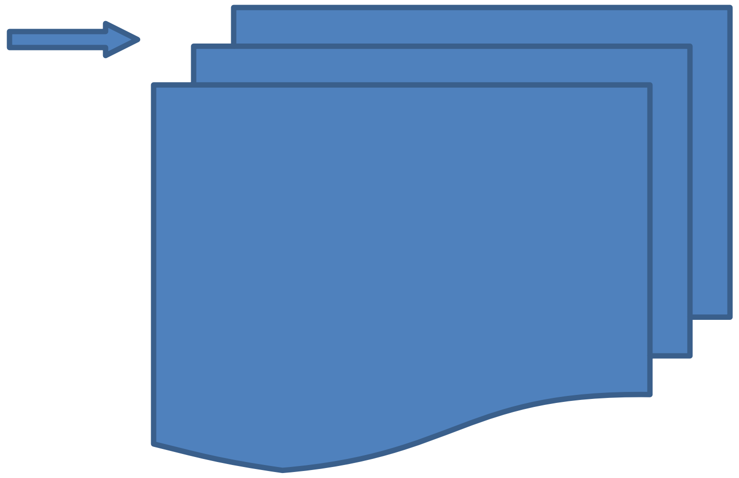
Without Index

DB needs to scan entire table or table scan



With Index

DB can go right to record



Indexes, the bad

- Overhead -- space, speed, maintenance

Indexes, the bad

- Overhead -- space, speed, maintenance
- Not a panacea – does not cure all problems

Indexes, the bad

- Overhead -- space, speed, maintenance
- Not a panacea – does not cure all problems
- Will not help if you need to perform a table scan

Indexes, the bad

- Overhead -- space, speed, maintenance
- Not a panacea – does not cure all problems
- Will not help if you need to perform a table scan
- Composite indexes can be tricky – YearMonthDay usually better than DayMonthYear

Tuning to 80%

- Use InnoDB

Tuning to 80%

- Use InnoDB
- Set *`innodb_buffer_pool_size`* to 70-80% of memory

Tuning to 80%

- Use InnoDB
- Set *`innodb_buffer_pool_size`* to 70-80% of memory
- Use XFS/ZFS/ext4

Tuning to 80%

- Use InnoDB
- Set *`innodb_buffer_pool_size`* to 70-80% of memory
- Use XFS/ZFS/ext4
- Partition data -- divide and conquer

Tuning to 80%

- Use InnoDB
- Set `innodb_buffer_pool_size` to 70-80% of memory
- Use XFS/ZFS/ext4
- Partition data -- divide and conquer
- Architect your data

Tuning to 80%

- Use InnoDB
- Set `innodb_buffer_pool_size` to 70-80% of memory
- Use XFS/ZFS/ext4
- Partition data -- divide and conquer
- Architect your data
- Review your SQL statements

Don't hurt yourself

Some RAID controllers have a battery learning cycle when BBU goes write-back. Schedule during off-time!

Minimize time for most frequent queries

Keep on top of upgrades

- 5.5 20% faster than 5.1

Tuning Past 80%

Will depend on *your* data

Tuning Past 80%

Will depend on *your* data

Lots of Tuning Help Available

Tuning Past 80%

Will depend on *your* data

Lots of Tuning Help Available

- *High Performance MySQL* – Schwartz et al
- *MySQL Administrator's Bible* – Cabral

Tuning Past 80%

Will depend on *your* data

Lots of Tuning Help Available

- *High Performance MySQL* – Schwartz et al
- *MySQL Administrator's Bible* – Cabral
- OurSQL podcast

Tuning Past 80%

Will depend on *your* data

Lots of Tuning Help Available

- *High Performance MySQL* – Schwartz et al
- *MySQL Administrator's Bible* – Cabral
- OurSQL podcast
- Performance forum on Forums.MySQL.Com
- Planet.MySQL.com & MySQL.Com

Tuning Past 80%

Will depend on *your* data

Lots of Tuning Help Available

- *High Performance MySQL* – Schwartz et al
- *MySQL Administrator's Bible* – Cabral
- OurSQL podcast
- Performance forum on Forums.MySQL.Org
- Planet.MySQL.com & MySQL.com

Skilled DBA

Tuning Past 80%

Will depend on *your* data

Lots of Tuning Help Available

- *High Performance MySQL* – Schwartz et al
- *MySQL Administrator's Bible* – Cabral
- OurSQL podcast
- Performance forum on Forums.MySQL.Org
- Planet.MySQL.COM & MySQL.Com

Skilled DBA

- Defined as obsessive professional looking to save nanoseconds off queries, possess current backups, helps developers rewrite queries, plans for future, and watches buffer hits rates compulsively.

Big hint

Seek to optimize the system as a whole. Often the database is blamed for systemic slowness when other components are at fault.

General Last hints

SQL

- Fetch needed data only, no **SELECT ***
- Use EXPLAIN
- Think in data sets, not nested loops
- Set benchmark times
- Use smallest data type possible
- Rewrite subqueries to joins

Mysqld

- Pay attention to new technologies, updates
- Know which buffers are per-session, global
- Do not ignore log files

Q&A

David.Stokes@oracle.com - http://NorthTexasMySQL.org