Scaling a Web Application
(mostly PHP/MySQL and almost mostly harmless)

Duleepa “Dups” Wijayawardhana
MySQL Community Team
Who the hell am I?

- PHP/MySQL Developer since the last century :)
- MySQL Community Relations Manager in North America
- Former Web Developer for MySQL.com
- Various positions at BioWare Corp. 2001-2007
Who the hell am I? /2

PHP Expert

ME

MySQL Expert
Who the hell am I?/3
My Agenda

• All about Scaling
• Scaling your application
• Scaling your database, especially MySQL
Your Agenda

• Are you a Sysadmin?
• Are you a PHP Developer?
• Are you a MySQL DBA?
• Are you all of the above?
• Are you having scaling problems?
• Are you anticipating having scaling problems?
Your Agenda/2

• Do you think you have PHP Issues?
• Do you think you have MySQL Issues?
• Are you just here because all the other tutorials didn’t seem fun?
• Who is sitting here wondering if their server is even up right now and whether they even have a job after this finishes?
• Are you here to learn the one thing that will make your sites/applications scale unbelievably? The AMAZING, INCREDIBLE, silver bullet?
The Cheesy Answer

You.

Yes, you.
You control the Information Age.
Welcome to your world.
Famous scaling stories

- **World of Warcraft** on release, too much load, unprepared, system down.
- **Twitter**, initially continuous scaling problems
- **YouTube**, had to come up with new scaling methods to succeed.
- **LiveJournal**, created new technologies to stay up
- **Facebook**, expanded existing technologies to stay alive, incl. technologies from LiveJournal.
Not so famous scaling stories

• Your company site gets Slashdoted
• Your personal website gets quoted on CNN
• You accidentally post a video which goes viral
• You decide to run a competition for drunk people to phone in on St. Patrick’s Day and you run the site on a $5/month hosted web service and some New York magazine picks it up as a story...
St Patrick's Day Drunk Dial 2009

www.stpatsdrunkdial.com

Rate: ★★☆☆☆☆☆☆☆☆ 6 ratings
Views: 2,299

More From: stpatsdrunkdial

Related Videos
- Drunk Leprechaun yells at kids (St. Patrick's Day)
  11,140 views
  SoberFreeShow
- St Patricks Day 2009!!!!!!!
  245 views
  westfan69
- 'Erin Go Braith'
  1,003 views
  dynamiccvip
- A Saint Patricks Day 2009
  1,003 views
  facebook
Scaling in General/1

- Scaling Definition:
  - To make something *run better* on a given set of *parameters*
  - Something = ?
  - Run = ?
  - Better = ?
  - Parameters = ?
Scaling in General/2

- Scaling is an Art
- There is no one Answer
- But are you all knowing?
Scaling in General/3

• Scaling is about the art of asking questions

• The Right Questions will lead to The Right Answers (eventually)

Sorry - Try Again!!

Hmm.. Where can it be...
Scaling in General/4

- Some bad questions:
  - “how can I set up my server for faster responses?”
    - better: “what is slowing down my queries?”
  - “how can I merge these two results faster in Java?”
    - better: “how do I make a JOIN?”
When to Scale

• At the beginning?
• When you hit a problem?
• When your site is down?
Question 1: Who Cares?! 

- Seriously... who cares?
- Who’s complaining?
- What do they care about?
- Why do they care?
Question 2: How fast?

• How fast do you need it done?
  • Timelines are very important to asking the right scaling questions
• How fast do you need it to get?
  • Obviously you need to understand what’s required!
Question 3: How much?

• How much are you willing to spend?
• How much effort are you willing to expend?
Question 4: Is it Enough?

- Have you defined success criteria to any scaling exercise?
- Have you done enough?
- When will what you have done be done?
Real Life Can Teach!

• Typically you describe the potential of a website as needing to withstand a hurricane.

• Let’s look at a real hurricane for lessons!
Hurricane!

- Your website: Enjoying normal weather on the coast of Louisiana
Hurricane!\(^2\)

- Off the coast a Hurricane is forming (someone is preparing to slashdot you)
Hurricane! / 3

• What are the factors in surviving a really big Hurricane in real life?
  • Knowing its path
  • Knowing its relative strength
  • Strong pre-existing levies
  • Knowledge of the weakest points in your Levies
  • A good evacuation plan
  • A good implementation plan
Hurricane! 4

- What are the factors in surviving a really big Hurricane in real life?
- Preventing abuse/looting
- Damage control plans
- Good leadership / project management
- Fast Response Times
- Good Luck
Hurricane! /5

- Knowing trajectory / Knowing its relative strength
- Monitoring
- Strong pre-existing levies
- Solid pre-existing infrastructure which can scale
- A good evacuation/implementation plan / Know your weak points
  - Know your pain points in the application
  - Have a Scaling Plan
Hurricane!

- Preventing abuse/looting
  - Make sure that a scaling issue does not lead to a security nightmare
- Damage control plans
  - Have a good disaster recovery plan if your scaling plan is wrong
- Good leadership / project management / Fast Response Times
  - What more can I say? Preparedness leads to success!
Hurricane! / 7

FLEE FROM THE WRATH TO COME
System Profile?

- A list of each major system in an application
- Do it on a white-board with your systems engineers.
- If you do not understand each of your major systems, you are about to be in for a world of pain.
Sample System Profile/1

• **Step 1:** List your current systems from the point at which a user clicks the browser to when they receive the page on their screen.
  
  • **always** start from the customers perspective
  
  • who cares? The customer cares, you care about the customer!
Sample System Profile/2

• **Step 2:** List current system load scenarios. Do for each system with low, medium, high, super-high loads.
  
  • e.g. Web Servers, with low load 2 redundant servers enough, medium, 4, high, 8, super-high 12.

  • Identify dependencies.
Sample System Profile/3

• **Step 3: List application nodes.** Identify key application areas and the load scenarios you expect on each area.

  • Important to identify, database throughput, data transferred, bandwidth.

  • If you don’t have load information for your application, you need to get it.
Sample System Profile/4

- Step 4: Go back to the System Load Analysis. Now with an application analysis go back and see if you had the System Load analysis correct.
Step 5: **Prepare for the worst.** At this point you want to figure out what happens when things go completely out of bounds, can you scale each node?
Step 6: Create a plan for scaling. You know how each part of your application is going to be affected by traffic and load, now plan for it.

- Sounds simple.
- Tough reality.
How to identify pain points

- **Pain Point:** Any point in the flow that is critical to the application.
  - Web server uptime should not be a pain point.
  - A transaction system which allows only 10 concurrent connections when you expect to have hundreds... that’s a whole lot of pain.
  - Identify “Single Points of Failure”: SPoFs
How to identify pain points/2

- Load testers?
- Run monitoring solutions, especially on your database to see what is being hit (more later)
- Use debuggers such as xDebug for PHP and application profilers
- Profile your queries.
How to identify pain points/3

• Remember anything which slows the user’s experience is a pain point: do not forget about javascript and performance.
Scaling in general: Summary

- **Sacrifice smart**: Be prepared to sacrifice performance for availability.
- **Grow smart**: don’t expand without a plan.
- **Scale smart**: adding more servers is not necessarily scaling.
- **Be smart**: if you don’t have a plan, it will be worse.
Part 2: Application Scaling

- Typically this is looked at last.
- Should be looked at first.
- Application Architecture is key.
- Optimizing Frameworks is KEY!
- Ignoring Frameworks is.... your decision!
Step 1: Code Profiling

Typically ignored, tend to be easier to scale numbers of servers, databases i.e. resources.

Run an application through:

- Load testers
- Profilers (debuggers)
- Memory footprint
- Code audit
An example of profiling to help performance

A profile of mysql.com in April 2008 with xDebug and kCacheGrind
Load Testing

- Tons of tools
- Pylot (open source python-based tool) [http://www.pylot.org/](http://www.pylot.org/)
- Create test cases, be as realistic as possible
Application Scaling / 4
Load Testing

- Pylot Summary:

```
Pylot - Performance Results

Workload Model
  test duration (secs) 61
  agents 10
  rampup (secs) 0
  interval (milliseconds) 0

Results Summary
  requests 1552
  errors 0
  data received (bytes) 6208

<table>
<thead>
<tr>
<th>Response Time (secs)</th>
<th>Throughput (req/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg</td>
<td>avg</td>
</tr>
<tr>
<td>stdev</td>
<td>stdev</td>
</tr>
<tr>
<td>min</td>
<td>min</td>
</tr>
<tr>
<td>50th %</td>
<td>50th %</td>
</tr>
<tr>
<td>80th %</td>
<td>80th %</td>
</tr>
<tr>
<td>90th %</td>
<td>90th %</td>
</tr>
<tr>
<td>95th %</td>
<td>95th %</td>
</tr>
<tr>
<td>99th %</td>
<td>99th %</td>
</tr>
<tr>
<td>max</td>
<td>max</td>
</tr>
</tbody>
</table>
```

```
Load Testing - Zend Framework

Average 13.4 Requests/Second
Application Scaling/6
Load Testing - Zend Framework (Modified)

Avg. 15.7 Requests/Second
Application Scaling/7
Load Testing - No Framework

Graph showing requests per second vs. elapsed time in the test.
Step 2: Query Profiling

Application profiling and scaling should be done before doing server scaling.

A query which works at 100 MB table may fail completely at 300GB but may just be a bad query.

Tools analyzing the Slow Query Log and MySQL Query Analyzer can help.
Step 2: Query Profiling: Indexes

Check to see what are your best performing index.

Example, 100 million rows indexed by first_name(1), potentially randomly in a latin1 charset, you would have 100 million/26 = 3.8 million rows to search through. Ineffective index.

Adding a index on first_name is that a good solution?
Application Scaling/10

- **Step 2: Query Profiling: Indexes**
- Find the worst performing indexes
• **Step 2:** Query Profiling: Explain

• A table from PlanetMySQL

```sql
CREATE TABLE `entries` (  
  `entry_id` int(10) unsigned NOT NULL AUTO_INCREMENT,  
  `title` varchar(255) NOT NULL DEFAULT '',  
  ...  
  KEY `title` (`title`(1))  
);
```
Application Scaling/12

• An Explain *without* the title index

```sql
mysql> explain select title from entries ignore index (title) where title like 'a%' order by title ASC limit 10\G
id: 1
select_type: SIMPLE
table: entries
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 18459
Extra: Using where; Using filesort
```
• An Explain *with* the title index

```
mysql> explain select title from entries where title like 'a%' order by title ASC limit 10\G

id: 1
select_type: SIMPLE
table: entries
type: range
possible_keys: title
key: title
key_len: 5
ref: NULL
rows: 706
Extra: Using where; Using filesort
```
Results (Ignoring Index)

<table>
<thead>
<tr>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 'Simple' Protocol for Manual MySQL Slave Promotion to Master</td>
</tr>
<tr>
<td>A (round-about) story about Jeffry P. Bezos</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>A better backup procedure</td>
</tr>
</tbody>
</table>

10 rows in set (0.07 sec)
Results (With Index)

<table>
<thead>
<tr>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 'Simple' Protocol for Manual MySQL Slave Promotion to Master</td>
</tr>
<tr>
<td>A (round-about) story about Jeffry P. Bezos</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>A better backup procedure</td>
</tr>
</tbody>
</table>

10 rows in set (0.02 sec)
• Is 0.05 seconds a big deal?
• Depends on the number of queries you have!
• What about a query which takes only 0.01 seconds, is that a problem?
• Enter tools such as Query Analyzer!
Application Scaling/3
• **Step 3: Application Caching**

• Caching is your friend.

• Cache any and all output that you can.

• Examine your application for caching potential.

• Can reduce your use of databases dramatically and allow you to use cheap web server farms instead: Wikipedia, MySQL.com.
Application Scaling/18

• **Step 3: Caching: MySQL**

• MySQL Query Cache may not be your friend!

• If you configure your cache to be too big you might end up with a huge slow down in performance.


Step 4: Monitoring

Build in monitoring into the application.

If a person falls in the forest do you hear the PHP Fatal Error?

Monitoring an application can help fix performance issues and identify pain points as you expand an application.
Application Scaling/20

- **Step 5: Application Sharding**
  - Separate the application into multiple different types (video distribution, images etc.) Can increase performance.
  - Better use of SAN hardware/storage solutions by pushing most accessed content on faster disks.
• Step 6: **Improving AJAX**
• Profile with YSLOW, Firebug
• Consider CDNs
• Cache JS where possible
• If “web 2.0” DB gets more writes than reads sometimes -- DB Performance!
Part 3: Database Scaling (mostly MySQL)

• Different types of databases may help with performance depending on the job

• Database Architecture is critical

• Knowing your databases is even more so
Step 1: Memcached.

- Developed by LiveJournal to cache key/value pairs in RAM of clusters of servers.
- Incredibly fast.
- Typically cache data coming from a DB in Memcache and then query Memcache from the application.
• **Step 1**: **Memcached**.
  • It is not a “relational” database.
  • Help with replication lag.
  • Binary safe/cheap.
Database Scaling/3

- **Step 2**: Database Configuration
- Master/Slave
Database Scaling

- **Master/Slave**
- Good for high read/relatively low write applications.
- High content sites with limited user input (i.e., writes) are good examples.
- **Con:** There is replication lag, can be avoided using memcached.
Database Scaling

- Sharding
• Sharding

• Where you have many users and many writes and have the ability to shard on some ID/unique data.

• Pros: much easier to scale out; cheap to scale out; good redundancy with slaves

• Cons: much more complicated application; can be overkill, problems with data connections between shards.
• MySQL Cluster (NDB)

• Database where data is shared amongst many nodes to create a true cluster. Great for database uptime. Used by Telcos.
Database Scaling/8

- MySQL Proxy

- MySQL Proxy is not a solution for scaling for speed.

- Alpha solution where proxy talks to your MySQL servers.

- Proxy gives you some interesting possibilities and potentially a look at the future

- Spider Storage Engine...
• **Step 3: MySQL Storage Engine**

• Different Storage engines

• MyISAM, fast, no transaction, table-level locking. In general: good for read heavy, bad for write heavy web apps.

• InnoDB faster reads on primary key lookup, row-level locking, transaction. In general: good for write heavy web apps.
Step 4: The Query

- Test queries against a proper dataset size.
- In MySQL, use EXPLAIN often and understand how to use indexes.
- Sometimes a sub-selects may not be as fast as two selects. (smugmug!)
- Monitoring tools: Query Analyzer, Slow Query Log.
Step 5: Schema Normalization

- You can gain more performance by denormalizing your data.
- Cons: you have to build in more data checks. Data integrity can become compromised very fast.
- Maybe not the first thing to scale in an OLTP environment.
• **Step 6: Tuning the database**

  • The various parameters in a MySQL database can have dramatic impact on your performance.

  • e.g. `key_buffer_size`, this can improve using indexes, but if it is too big then your system slows down as it starts to page and write to disk.

  • Start tweaking and monitoring :) or seek professional advice!
Database Scaling/13

• **Step 7: MySQL 5.4**?
  
  • Performance version of MySQL
  
  • Scales much better vertically than any previous official release of MySQL.
Step 8: **Hardware time...**

- Improve performance by increasing RAM, faster Disk I/O depending on your situation and data served
- Split the DB to serve frequently accessed data from faster disks.
- Network issues?
- Move masters to faster machines, more RAM.
- OpenStorage? Flash Storage? SAN?
Last Words/Lessons Learned

• So does this all work? Some war stories :)  
• Remember there is no silver bullet!  
• Scaling Open Source means you have a million experts, ask your community, participate in the community!  
• Where to find this presentation? The MySQL Librarian!  
• Become part of the MySQL Community
Last Words/Lessons Learned

• How to find me:
  • dups@sun.com
  • dups@dups.ca
  • http://blogs.sun.com/dups
  • http://www.dups.ca
  • Facebook: /duleepa
  • Twitter: @dups
  • LinkedIn, Flickr

• Non Sun-Fun:
  • Tweet Rhapsody
    http://tweetrhapsody.com
  • St. Patrick’s Day
    Confessionals
    http://stpatsdrunkdial.com
Duleepa “Dups”
Wijayawardhana
dups@sun.com

Slides will be made available