Setting up a LAMP Server

Morgan Tocker, Support Engineer
MySQL AB, Brisbane, Australia, morgan@mysql.com

January 24 Dunedin, NZ
Agenda

- Install Debian
- Install MySQL 5
- Install PHP
- Testing
- Install E-Accelerator
- General Optimisation overview
- Backing up your data
- Restoring backups
- Q&A
Install Debian

- I prefer a light / base install approach -- for whatever reason;
  - Security
  - I know what's installed – it's easier to rebuild.
Current Versions of MySQL

• MySQL 5.0.18 GA
  – Stable since Oct 05
  – Brings triggers, stored procedures, pluggable storage engines, views, information schema, XA, greedy optimizer magic!
Current Versions (cont..)

- MySQL 4.1.16 GA
  - Subselects, multiple character sets, prepared statements
Current Versions (cont..)

- MySQL 4.0.x, 3.23.x – previous releases.
  - You possibly don't want these.
Current Versions (cont..)

• MySQL 5.1.6 - Alpha
  – Reaching feature freeze about now
  – Partitioning, more storage engine magic, multi-master replication*, xml (xpath), time scheduling.
Eek!

• MySQL 4 to 4.1:
  – Timestamps
  – password
  – character sets
• MySQL 4.1 to 5.0:
  – White space in function names
  – trailing space in varschar
  – sql 'modes'
Where to get

• Debian's packages
  – they are a bit behind
• Community Edition binaries from mysql.com
  – Layout breaks the Debian file system guidelines (/usr/local/* is not meant to have files), but I like it!
• Compile your own
  – not recommend for greatest stability
• MySQL Network's certified binaries
  – The team I work for.
Install PHP

- I like PHP5. <insert bias> That's what you want.
- PHP5 contains new OO goodness.
- PHP5.1 is fast.
- some things broke were fixed – which possibly slowed adoption.
Where to get PHP

- Debian package
- dotdeb.org && dexter repositories
  - http://www.dotdeb.org/
- Compile your own.
Testing PHPMyAdmin

morgo@morguntu:~$ ab -n 100 localhost/phpmyadmin
This is ApacheBench, Version 2.0.41-dev <$Revision: 1.141 $> apache-2.0
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Benchmarking localhost (be patient).....done
[..]
Document Path: /phpmyadmin/
Document Length: 1778 bytes

Concurrency Level: 1
Time taken for tests: 8.468725 seconds
Complete requests: 100
Failed requests: 78
  (Connect: 0, Length: 78, Exceptions: 0)
Write errors: 0
Total transferred: 240040 bytes
HTML transferred: 177540 bytes
Requests per second: 11.81 [#/sec] (mean)
Time per request: 84.687 [ms] (mean)
Time per request: 84.687 [ms] (mean, across all concurrent requests)
Transfer rate: 27.63 [Kbytes/sec] received
Connection Times (ms)

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>mean[+/-sd]</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>Processing</td>
<td>63</td>
<td>84</td>
<td>39.3</td>
<td>65</td>
</tr>
<tr>
<td>Waiting</td>
<td>29</td>
<td>80</td>
<td>38.8</td>
<td>65</td>
</tr>
<tr>
<td>Total</td>
<td>63</td>
<td>84</td>
<td>39.3</td>
<td>65</td>
</tr>
</tbody>
</table>

Percentage of the requests served within a certain time (ms)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>65</td>
</tr>
<tr>
<td>66%</td>
<td>65</td>
</tr>
<tr>
<td>75%</td>
<td>66</td>
</tr>
<tr>
<td>80%</td>
<td>150</td>
</tr>
<tr>
<td>90%</td>
<td>158</td>
</tr>
<tr>
<td>95%</td>
<td>165</td>
</tr>
<tr>
<td>98%</td>
<td>172</td>
</tr>
<tr>
<td>99%</td>
<td>205</td>
</tr>
<tr>
<td>100%</td>
<td>205 (longest request)</td>
</tr>
</tbody>
</table>
Installing E-Accelerator

• An opcode cache
  – Similar to APC/Zend Performance Suite/Turk MMCache

• PHP is interpreted
  – Before it is run, it is parsed into opcodes
  – E-accelerator caches that in memory, cutting out a step.

• Performance varies
  – If network I/O, database are your bottleneck, then tough.
  – I've seen x5 improvement
Re-benchmarking

Concurrency Level: 1
Time taken for tests: 3.837926 seconds
Complete requests: 100
Failed requests: 70
  (Connect: 0, Length: 70, Exceptions: 0)
Write errors: 0
Total transferred: 240052 bytes
HTML transferred: 177552 bytes
Requests per second: 26.06 [#/sec] (mean)
Time per request: 38.379 [ms] (mean)
Time per request: 38.379 [ms] (mean, across all concurrent requests)
Transfer rate: 60.97 [Kbytes/sec] received
General Optimisations I'm familiar with

• Slow query log
• Query cache
• Thread cache
• Improve index performance
  – EXPLAIN
• Change schema
  – PROCEDURE ANALYZE();
• my-huge.cnf etc.
Backi ng up your MySQL Data

• Three methods I'd like to discuss;
• 1. Backing up the datadir
  – Inexpensive if the server is shutdown (just copy raw files)
  – Hard to do a partial / PITR recovery.
• 2. Exporting an SQL dump of the data
  – Can be done as a single transaction, many options.
  – Can be backwards compatible to earlier versions, or compatible with other DBMS (4.1 added compatibility options)
• 3. setting up a quick replication system
  • backing up off the slave
  • no huge I/O overhead of backup on master
  • 'hot spare' in event of failure.
Backing up the datadir

- `cp /usr/local/mysql/data/* [somewhere]`
Exporting the SQL Dump of the data

- mysql dump -u ted -password=bonza -all-databases > sqldump.sql
Off a slave

• on the master:
• mysqladmin -u ted --password=bonzafifty2 --master-data=1 --all-databases > sqldump.sql
• mysql > GRANT REPLICATION SLAVE on *.* TO 'repl'@'%.mydomain.com' IDENTIFIED by 'ihaveaweakpassword';
Off a slave (cont..)

- *on the slave:*
- `mysql > CHANGE MASTER TO MASTER_HOST='master', MASTER_PASSWORD='yep';`
- `shell > mysql < sqldump.sql`
- `mysql> start slave;`
- `mysql > show slave status;`
Restoring from a Backup

• PITR (Point in time recovery)
  – `mysqlbinlog --start-position=x --stop-position=x binlogname > sqldump.sql`
  – `mysqlbinlog --start-datetime=yyyy-mm-dd --stop-datetime=yyyy-mm-dd binlogname > sqldump.sql`

• Recovering a single database that someone bollocks'ed
  – `mysql -o mysql < sqldump.sql`

• Recovering from datadir backup.
  – replace the files.
Q&A