Scaling for 1 Billion Queries per day

A journey into Friendster’s backend.

Presenter: Dathan Vance Pattishall

© 2005 Friendster Inc.
• Left Sun Microsystems to work at a startup in late 1998.
• Worked with a team to sustain 1 million queries per day using mySQL, Perl, PHP
• More than 17 million members staying in touch with one another and making new connections
• The largest social networking site on the web
Start at Friendster

• Tremendous number of users
• Unable to make drastic changes since uptime was just as important as speed
• No profiling
• No data model
• Inconsistent associations between tables, columns and usage.
Steps for re-factoring

- Assess the current problems
- Determine available hardware
- Don’t go after the biggest problems first
- Optimize without downtime
- Split load
Sample of Traffic on a box After-optimize
HE Diagram February - April 2004
Query Optimizations + New Pools
Gained 10 Mb of traffic after new C++ Graph

App Servers (cookie based stickiness)

Load Balancer

dbmsgmstr1
2u 3 effective spindles
RAID-10 System

Users

Messages

Users

Users

Flat Network

3u 6 effective spindle
RAID-10 System

dbmaster1

dbmsg

dbslowtriple

dbstmtn

dbfast

dbfriend

dbccompute

All

All

All except dbfriend-profile

All except dbfriend-profile

All Writes EXCEPT Messages

Users, Profile

Messages, Users

v
Migration to a new Platform

• 32-bit Intel to 64-bit AMD
• 15K SCSI Maxtor 36 GB Drives to a Hitachi SAN – with an experimental switch.
• RedHat 7.3 to RedHat AS + SuSe Kernel
• 2 GB address space to 4GB+
• First known implementation of this design
## IO Scheduler Problems

**Sysbench Tests with multiple configs**

<table>
<thead>
<tr>
<th>64 Bit Tests</th>
<th>64 Threads</th>
<th>64K Blocks</th>
<th>dbtest1: 2.6.6 - Opteron SAN qlogic card Dedicated 20 Disk RAID 5 Stripe</th>
<th>dbuser1: 2.6.6 - Opteron SAN Dedicated 20 Disk RAID 5 Stripe</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7.644 Mb/sec</td>
<td>1129.24 tps</td>
<td>9.311 Mb/sec</td>
<td>19.643 Mb/sec</td>
</tr>
</tbody>
</table>

**dbtest1:**
- 2.6.6
- Opteron
- Local Disk Test
- 4 disk RAID-0

**dbuser1:**
- 2.6.6
- Opteron
- SAN
- Dedicated 20 Disk
- RAID 5 Stripe

**dbtest1:**
- 2.6.6
- Opteron
- SAN qlogic card
- Dedicated 20 Disk
- RAID 5 Stripe

**dbuser1:**
- 2.6.6
- Opteron
- SAN
- Dedicated 20 Disk
- RAID 5 Stripe
## Proof that the scheduler in 2.6.6 is the problem

<table>
<thead>
<tr>
<th>64 Threads</th>
<th>64K Blocks</th>
<th>32 Bit Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linus Scheduler - Write</strong></td>
<td><strong>Linus Scheduler - Read</strong></td>
<td></td>
</tr>
<tr>
<td>dbtest2u1: 2.4.23aa2 Xeon Local 4 disk RAID-0</td>
<td>dbtest2u1: 2.4.23aa2 Xeon SAN 4 RAID Groups</td>
<td></td>
</tr>
<tr>
<td>16.83 Mb/sec 1077.44 tps</td>
<td>15.699 Mb/sec 1069.12 tps</td>
<td></td>
</tr>
<tr>
<td>dbuser4: 2.4.23aa2 Opteron SAN 32bit Groups</td>
<td>dbuser4: 2.4.23aa2 AN 32bit RAID Groups</td>
<td></td>
</tr>
<tr>
<td>19.3 Mb/sec 1237.5 tps</td>
<td>19.2 Mb/sec 1235 tps</td>
<td></td>
</tr>
<tr>
<td>13 Mb/sec 1005.45 tps</td>
<td>9.6 Mb/sec 1059.6 tps</td>
<td></td>
</tr>
</tbody>
</table>

| buser4: 2.4.23aa2 pteron SAN Himem | buser4: 2.4.23aa2 pteron 2bit Himem |
| buser4: 2.4.23aa2 pteron SAN Himem | buser4: 2.4.23aa2 pteron 2bit Himem |
| pteron | pteron |
| SAN | 2bit |
| Himem | Himem |
| 4 | 4 |

**Note:** The table shows performance metrics for different test cases involving different test subjects and RAID configurations.
Solution

- Run 2.4 without O_DIRECT and INNODB in 64-bit mode
- Thus the need for SuSe + RedHat
- ext3 has a bug in 2.4 that doesn’t allow O_DIRECT
How did we migrate to fewer servers?

- Dual x86-64 AMD Opterons
- 8 GB of RAM
- Faster disk (SAN)
- Optimized indexes
- Hardware load balancer in front of the databases
Distributing Load

- Created pools
- Isolated traffic
- Reduced management costs
Traditional 3-tier architecture with a twist: Load Balanced mySQL (no persistent connections)
Hit 1 billion Queries per Day

• Optimized Queries
  – Removed all sorts (example will follow)
  – Moved sorting query types into the application and added LIMITS.
  – Reduced ranges
  – Range on primary key
2 Queries are better than 1

SELECT COUNT(*) FROM messages where user=5435581 and folder='Inbox';

<table>
<thead>
<tr>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>438</td>
</tr>
</tbody>
</table>

SELECT id,...,DATE_FORMAT(date, '%M %e, %Y %l:%i %p') AS date,subject from messages WHERE user=5435581 and folder='Inbox' LIMIT 428, 10;

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>544791137</td>
<td>April 11, 2005 6:57 PM</td>
</tr>
<tr>
<td>544935034</td>
<td>April 11, 2005 10:31 PM</td>
</tr>
<tr>
<td>544979541</td>
<td>April 11, 2005 11:31 PM</td>
</tr>
<tr>
<td>545166664</td>
<td>April 12, 2005 3:43 AM</td>
</tr>
<tr>
<td>545708612</td>
<td>April 12, 2005 11:02 PM</td>
</tr>
<tr>
<td>546825155</td>
<td>April 14, 2005 7:26 AM</td>
</tr>
<tr>
<td>547668010</td>
<td>April 15, 2005 9:42 AM</td>
</tr>
<tr>
<td>547773882</td>
<td>April 15, 2005 3:54 PM</td>
</tr>
<tr>
<td>547943621</td>
<td>April 15, 2005 10:38 PM</td>
</tr>
<tr>
<td>548851490</td>
<td>April 17, 2005 8:20 AM</td>
</tr>
</tbody>
</table>
Looking to the Future

• Reduce query count
• Introduce automatic recovery from failure
• Hash / split query load based on users
• Like to see some cool things from mySQL.
• Assess: Define the issues
• Work with a team
• Benchmark → Make Change → Benchmark → Make Change
  (Cycle of Improvement)
• Stabilize: always have a plan to rollback
• Trust mySQL.
Contact info:
dathan@friendster.com,
dathanvp@yahoo.com

Database Stats for trending mySQL:
mySQL Details Stats Graphed -
http://sourceforge.net/projects/mysqlgraph/

http://dathan.blogs.friendster.com