



MySQL Performance and Scalability

Peter Zaitsev
CEO, Percona Inc
Percona Live, Washington, DC
11 January 2012

MySQL Performance and Scalability

- Basic Definitions
- Ways to Get good Performance and Scalability
- Getting Most of Single Box
- Using Multiple Boxes

What do we really need ?

- Users Care about **Response time** of their Requests to **Application**
- Performance might be poor even for single user
 - Lets Call it Response time Bound
- Often problems happen when Many users Operate
 - Let Call it Throughput Bound
- Note. Users care about predictable performance.
 - Predictable means low variance
 - Average or mean performance numbers are meaningless

Growth of the Application

- Increase in Number of Users
 - Increase in User Interactions might be even faster
 - And as such about number of “request arrivals”
- Increase of the Data Volume/database size
- Increase in Query Complexity
 - Each Query complexity may growth with increase in data size.
- Application complexity Growths
 - “Features” become more elaborate requiring more queries
 - Or more complicated queries to the database.

What do we want ?

- Maintain or Improve Performance of User Interactions
- As our System Grows and Changes
- Do this with Reasonable Cost

Did you say Cost ?

- Yes. We want to make system cost effective
- We want cost per user/user interaction be low
 - And dropping over time
- What is Cost ?
 - We care about Total Cost. Also known as TCO
 - Which includes much more than cost of hardware/software
 - Accounts for development/change complexity
 - Risks and costs of downtime etc

Basic Definitions

- Performance
 - System Response Quickly
- Scalability
 - I can maintain/improve performance by using More powerful system or multiple systems
- Efficiency – how efficient the system is ?
 - As in cost per user

Scalability and Efficiency

- There are no technologies with Infinite Scalability
 - Per Amdahl's Law and Universal Scalability Law by Neil Gunter
- More Scalable systems are usually not as cost effective.
 - Having very scalable and cost effective system is a Holy Grail
- Few Applications need Google/Facebook Scale
 - You need to know the scale of your application
 - To pick efficient solutions which provides you good scalability

Scalability

- **Scale Vertically (Scale UP)**
 - Using more powerful System for your MySQL Instance
 - “Simple” software; Expensive Hardware
- **Scale Horizontally (Scale out)**
 - Use many servers
 - Can use low cost servers
 - But more complicated software and operations
- **Often you Scale Up and Out at the same time.**

Getting Good Performance and Scalability

- **Rule #1** – Do not look at it as purely Database Problem
 - Lots of optimization opportunities outside of MySQL
- **Rule #2**
 - Remember the best way to optimize something is stop doing it.
- **Rule #3**
 - Mind Network Roundtrips
 - Both internal and external
- **Rule #4**
 - Queuing is best on higher level

Note Purely Database Problem

- Make sure application uses reasonable algorithms
 - And reasonable queries as result
- Consider Caching and Buffering
 - Varnish, memcached etc
- Supplement MySQL with Other tools
 - Memcache, Redis, Gearman, MongoDB, Hadoop etc
- Be open minded about application functionality
 - Very small changes may give huge performance wins

Stop doing something you do not need

- Do not read rows you do not need
- Do not fetch columns you do not need
 - Though simplicity might advise opposite here
- Do not fetch same data more than once
- Make sure your queries do not join tables they do not need
 - Can be especially problem with VIEWS
- Make sure queries only traverse rows they need
 - Use proper indexes to achieve this.

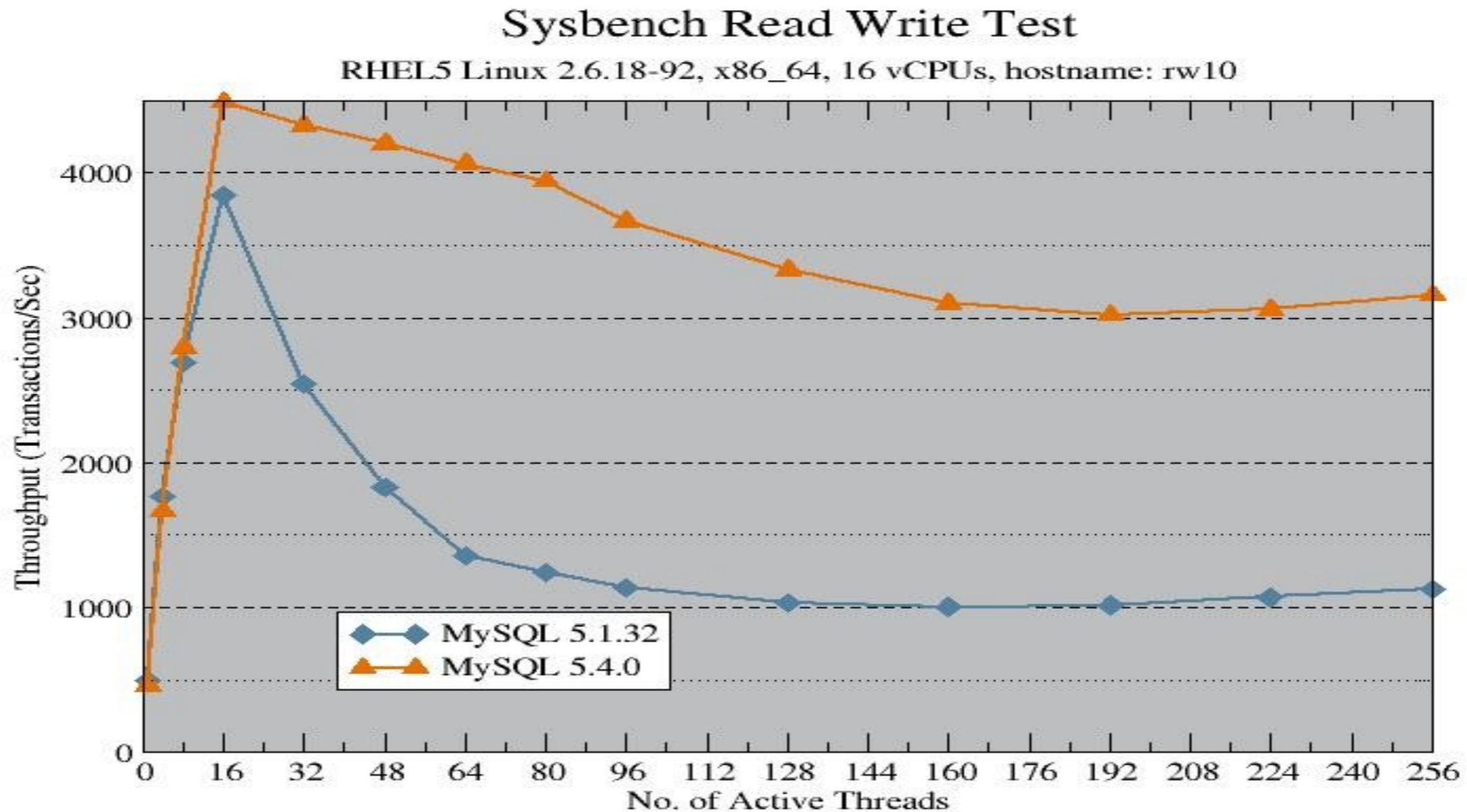
Network Round trips

- Network is Expensive
 - Remote Network is 10-100ms (and often unpredictable)
 - Local network is 0.3ms (for 1Gbit network)
- Page performing 1000 of separate memcache lookups
 - $0.3\text{ms} * 1000 = 0.3\text{sec}$ for network traffic alone
- Additional local caching can make sense
- Process data in sets
 - Memcache `multi_get()`, `SELECT ... IN (1,2,3,4,5)`
- Concurrency is your friend
 - Perform multiple network IO operations in parallel

Queuing

- Each resource has optimal concurrency
 - Pushing concurrency higher just makes things slower, better to queue
 - Imagine single CPU, need to process 100 tasks each cost 1 sec of CPU time. What better parallel or one after another ?
- Queue can be used to achieve optimal concurrency
 - Queue on higher level means more resources already busy in processing
 - Internal Queueing in MySQL means connection is created, various buffers are allocated, possible row level locks etc.
 - Queuing on load balancer is great

Random Scalability Graph



Realities of Single Server

- You can get a lot even from “Commodity” Server
 - 64 Cores (with HT)
 - 512+ GB of Memory
 - 2TB+ of High Performance Flash Storage
- On MySQL It can mean
 - 200K+ of simple queries/sec
 - 5.000.000+ rows traversed/sec
 - 20.000+ updates per second
 - 50.000+ storage IO/s per second.

Lets do some Math

- Lets assume user interaction does
 - 100 read queries + 10 write queries
 - Reading 2500 rows;
- Can get 2000 user interactions/sec
- This is 80M interaction/day
 - Allowing for double the traffic at the peak

Getting Great Performance from Single Server

- Right MySQL Server version
- Right Hardware
- Right MySQL Configuration
- Schema and Queries

Right MySQL Server

- A lot of Optimizations in New Versions
- MySQL 5.5 or Percona Server is best
 - If you're running High End Hardware or High Concurrency
- MySQL 5.6 looks to be even better
- Use InnoDB
 - MyISAM does not scale well with table locks and internal contention

Right Hardware

- Fast CPUs more important than many cores
 - Especially considering MySQL Replication single thread
 - 2 socket servers is sweet spot for MySQL
- Plenty of Inexpensive Memory
 - Memory chips are inexpensive until certain size
 - Memory frequency/type has very small impact
- Consider Flash if you have intensive IO
- Good Network Communication is Must
 - Single 1Gbit port might not cut it any more

Common Hardware Mistakes

- Getting Many slow cores
 - ... and being bound by MySQL single thread replication
 - CPUs for 4 sockets with many cores tend to have lower frequencies.
- Saving on Memory
 - Going 16GB to 64GB increased throughput 10x for one customer
- Getting stuck with conventional storage
 - If you're buying external disk shelf to get more IOPS you likely just need Flash.

MySQL Configuration

- Few Options Matter in Most Cases
 - innodb_buffer_pool_size
 - Innodb_buffer_pool_instances
 - innodb_log_file_size
 - innodb_flush_method=O_DIRECT
 - innodb_log_buffer_size
 - innodb_file_per_table
 - innodb_flush_log_at_trx_commit
- As soon as you get main options right you should get 95% of possible performance in 95% of cases
- [Http://tools.percona.com](http://tools.percona.com) - can get you started.

Queries and Schema

- This is where the most possibilities for DBA come from
- They really come together
 - Many query optimizations may need Schema changes
- Learn MySQL EXPLAIN statement
- Profiling queries with SHOW STATUS
- SHOW PROFILES
- Understanding PERFORMANCE SCHEMA
 - Brand new in MySQL 5.5
- Analyzing Workload via Slow Query Log

Slow Query Log Analyses

- Use Percona Server for Most advanced features
- Use `long_query_time=0`
 - Use query sampling if you have to
- Can find queries which combined take the most time
 - Likely causing most load on the server
- As well as Queries which are slow
- Pt-query-digest is the tools we use the most
 - Check out tutorial at <http://bit.ly/tCNtgF>

Slow Query Log Entry Sample

```
# User@Host: cust_event_user[cust_event_user] @ localhost []
# Thread_id: 8923660 Schema: customers_eventum Last_errno: 0 Killed: 0
# Query_time: 0.002723 Lock_time: 0.000044 Rows_sent: 1 Rows_examined: 3422 Rows_affected: 0 Rows_read: 0
# Bytes_sent: 259 Tmp_tables: 1 Tmp_disk_tables: 1 Tmp_table_sizes: 1017
# InnoDB_trx_id: 15B51C6B
# QC_Hit: No Full_scan: Yes Full_join: No Tmp_table: Yes Tmp_table_on_disk: Yes
# Filesort: No Filesort_on_disk: No Merge_passes: 0
# InnoDB_IO_r_ops: 0 InnoDB_IO_r_bytes: 0 InnoDB_IO_r_wait: 0.000000
# InnoDB_rec_lock_wait: 0.000000 InnoDB_queue_wait: 0.000000
# InnoDB_pages_distinct: 29
SET timestamp=1325884197;
SELECT
```

```
    DISTINCT contacts.customers_id, contacts.name, contacts.email
FROM
    customers_eventum.eventum_issue,
    customers.customers,
    customers.contacts
WHERE
    iss_prj_id = 2 AND
    iss_id IN (16604) AND
    customers.id = iss_customer_id AND
    contacts.customers_id = customers.id AND
    is_dead = 0
```

```
-- File: send.php    Line: 61    Function: sendEmail    request_id: e3a4fa6dccd4832e54687b547c68b319ca6455eb    session_id:
ker0l4hanvq4qvpk5jvr6gf7v4;
```

Pt-query-digest Workload Summary

```
# 45.9s user time, 110ms system time, 46.34M rss, 136.47M vsz
# Current date: Fri Jan 6 11:47:12 2012
# Hostname: sl2.percona.com
# Files: slave-slow.log
# Overall: 109.28k total, 365 unique, 44.46 QPS, 0.11x concurrency _____
# Time range: 2012-01-06 11:06:12 to 11:47:10
# Attribute      total    min    max    avg    95%    stddev  median
# =====
# Exec time      271s    1us    3s    2ms    725us  31ms    57us
# Lock time      2s      0      179ms 21us   52us   549us   17us
# Rows sent      957.82k 0      3.08k 8.98   6.98   129.59  0
# Rows examine   128.74M 0      201.42k 1.21k 65.89  14.09k  0
# Rows affecte   2.36k   0      9      0.02   0      0.15    0
# Rows read      7.83M   0      17.92k 75.09  6.98   482.67  0
# Bytes sent     202.56M 0      9.56M  1.90k  2.76k  66.99k  234.30
# Merge passes   0        0      0      0      0      0      0
# Tmp tables     12.09k  0      3      0.11  0.99   0.31    0
# Tmp disk tbl   8.08k   0      2      0.08  0.99   0.26    0
# Tmp tbl size   704.06M 0      5.00M  6.60k  1.04k  77.76k  0
# Query size     38.26M  6      1.34M  367.12 685.39 11.99k  44.60
# InnoDB:
# IO r bytes     0        0      0      0      0      0      0
# IO r ops       0        0      0      0      0      0      0
# IO r wait      0        0      0      0      0      0      0
# pages distin  102.47k  0      523   11.68  27.38  14.59   0.99
# queue wait     0        0      0      0      0      0      0
# rec lock wai   0        0      0      0      0      0      0
# Boolean:
# Filesort      20% yes, 79% no
# Full join     0% yes, 99% no
# Full scan     13% yes, 86% no
# QC Hit        0% yes, 99% no
# Tmp table     11% yes, 88% no
# Tmp table on  7% yes, 92% no
```

Workload Profile

Profile

```
# Rank Query ID      Response time Calls R/Call Apdx V/M Item
# =====
# 1 0x3E474EA62A0BCB14 220.3137 81.4% 491 0.4487 1.00 0.01 SELECT customers_eventum.eventum_user
# 2 0x06D241D5B72720A9 10.5684 3.9% 94 0.1124 1.00 0.01 SELECT customers_eventum.eventum_support_email_body
# 3 0x0F22B9AEE81F695D 8.7109 3.2% 2160 0.0040 1.00 0.00 SELECT contacts
# 4 0x8079A317BD20666B 5.4872 2.0% 57 0.0963 1.00 0.00 SELECT customers_eventum.eventum_user
# 5 0x8C454172AB539241 4.4759 1.7% 273 0.0164 1.00 0.00 SELECT customers_eventum.eventum_user
# 6 0x871E2FB00FE4A755 2.9693 1.1% 2 1.4847 0.75 2.67 DELETE egw_app_sessions
# 7 0x833687FB3D9FC279 1.6593 0.6% 10 0.1659 1.00 0.11 INSERT customers_eventum.eventum_support_email_body
# 8 0xF14E3F65D71E5D8E 1.5138 0.6% 82 0.0185 1.00 0.01 SELECT customers_eventum.eventum_user
# 9 0xF6F65CD386F5652A 1.1433 0.4% 1613 0.0007 1.00 0.00 SELECT customers_eventum.eventum_issue_custom_field
# 10 0x6973AB8A670DFCC7 0.8394 0.3% 1535 0.0005 1.00 0.00 SELECT customers_eventum.eventum_issue_custom_field
# MISC 0xMISC 13.1134 4.8% 102958 0.0001 NS 0.0 <355 ITEMS>
```

Query Profile part1

```
# Query I: 0.20 QPS, 0.09x concurrency, ID 0x3E474EA62A0BCB14 at byte 73987359
# Scores: Apdex = 1.00 [1.0], V/M = 0.01
# Query_time sparkline: |   ^   |
# Time range: 2012-01-06 11:06:20 to 11:47:09
# Attribute  pct total  min  max  avg  95% stddev median
# =====
# Count      0  491
# Exec time   81  220s 403ms 688ms 449ms 609ms 64ms 412ms
# Lock time   11  270ms 53us 179ms 550us 89us 8ms 80us
# Rows sent   1 11.47k 0 109 23.92 76.28 26.77 9.83
# Rows examine 73 95.11M 198.25k 198.68k 198.35k 192.13k 0 192.13k
# Rows affecte 0 0 0 0 0 0 0 0
# Rows read   0 11.77k 0 108 24.55 76.28 27.39 10.84
# Bytes sent  1 2.73M 1.37k 18.83k 5.70k 15.20k 4.61k 3.52k
# Merge passes 0 0 0 0 0 0 0 0
# Tmp tables  3 491 1 1 1 1 0 1
# Tmp disk tbl 5 491 1 1 1 1 0 1
# Tmp tbl size 3 26.82M 0 254.64k 55.94k 182.98k 62.60k 22.45k
# Query size  1 624.73k 1.27k 1.27k 1.27k 1.26k 0.00 1.26k
# InnoDB:
# Boolean:
# Filesort 100% yes, 0% no
# Full scan 100% yes, 0% no
# Tmp table 100% yes, 0% no
# Tmp table on 100% yes, 0% no
```

Query Profile Part2

```
# Databases  customers_eventum
# Hosts      localhost
# Last errno 0
# Users      cust_event_user
# Query_time distribution
# 1us
# 10us
# 100us
# 1ms
# 10ms
# 100ms #####
# 1s
# 10s+
# Tables
# SHOW TABLE STATUS FROM `customers_eventum` LIKE 'eventum_user'\G
# SHOW CREATE TABLE `customers_eventum`.`eventum_user`\G
# EXPLAIN /*!50100 PARTITIONS*/
SELECT ....
```

Using More than One Server

- Scaling Reads
- Scaling Writes
- Maintaining Memory fit ratio needed
 - Fitting 1TB working set might be expensive on one server
- Operational Needs
 - Large databases might be inconvenient to deal

Using Multiple Servers Choices

- Replication
- Functional Partitioning
- Sharding
- MySQL Cluster (NDB)
- Percona XtraDB Cluster (Galera)
- Emerging Technologies

MySQL Replication

- Used in Majority of MySQL Applications
- If not for Scaling than for Availability
- Replication is Asynchronous
 - Careful if you're reading from Slaves
- Replication does not scare writes
 - In fact it limits them as standard replication is single threaded
- Data duplication makes cache and storage use inefficient.
- For many applications Replication can take you 5x compared to single server.

Functional Partitioning

- Split Functions of Application and put them on different servers.
- Or at least separate tables you do not join on different boxes
- Typically hard to find many independent parts you can split easily
- Rarely can give more than 2-3x gain
- Yet good to do anyway as sharing server by many different applications adds complexity
 - What if one application wants MySQL 5.5 while other can't work with MySQL 5.5 ?

Sharding

- “Manual Partitioning”
- Splitting data by certain key and placing on different servers
 - Typically something like user_id
- Can be trivial for some applications
 - When data for different users is independent
 - A lot of SaaS applications are these ways
 - Until you do not have user large enough single server can't handle them
- Can be really tough for other applications
 - May be benefiting to delay sharding until you really need it
- Typically used together with Replication

MySQL Cluster

- Based on NDB Storage engine
 - Different from Innodb in many respects
- Automatically partitions data across many nodes
 - And keep multiple replicated copies
- Originally design for real time in memory telecom applications
 - Gradually finds it way to be more useful for conventional applications
- Requires high speed reliable interconnect to work well.

Percona XtraDB Cluster

- Based on “Galera” Replication Technology
 - Beta technology at this point
- Uses Conventional Innodb Tables
- Synchronous Replication
 - Can read from the slave safely
- Can replicate data in parallel
 - No single thread replication bottleneck
- Needs One Round Trip per transaction commit
 - Places less demands on network
- Does not partition data
 - Does not help to scale writes dramatically

Emerging Technologies

- Clustrix
 - “NewSQL” solution. Appliance Based. Proprietary database technology which speaks MySQL Protocol. Very Cool
- SchoonerSQL
 - Commercial Software based on MySQL
 - Synchronous replication; Cluster Management; Flash Based
- Tungsten Replicator
 - External Replication for MySQL
 - Can replicated from MySQL to Different databases
 - Tungsten Enterprise
 - Commercial solution for Automated Failover etc

Emerging Technologies

- ScaleDB
 - Oracle RAC like technology for MySQL (Beta)
- GenieDB
 - Plugable storage engine for MySQL
- Xeround
 - MySQL based solution for Dynamically scalable cloud
- ScaleBase
 - Automatically sharding middleware (Beta)
- DbShards (Beta)
- ScaleArc

Summary

- We defined Performance and Scalability which is important how to reach it
- We looked at Ways to Achieve High Performance Scalability inside MySQL at Application level
- We looked how you can get most of your MySQL server
- And we looked what approaches and technologies can take you beyond single MySQL Server if it is not enough.

Percona Live DC Sponsors



MySQL Conference & Expo 2012

Presented by Percona Live

The Hyatt Regency Santa Clara & Santa Clara Convention Center

April 10th-12th, 2012

Tutorials & Initial Breakout Sessions Announced

<http://www.percona.com/about-us/pressreleases/percona-live-mysql-conference-and-expo-heads-to-santa-clara-california-april-10-12-2012/>

Featured Speakers

Mark Callaghan (Facebook), Jeremy Zawodny (Craigslist), Marten Mickos (Eucalyptus Systems)

Sarah Novotny (Blue Gecko), Peter Zaitsev (Percona), Baron Schwartz (Percona)

Learn More at www.percona.com/live/mysql-conference-2012/

Want More MySQL Training?

Percona Training in Washington DC Next Week

January 16th-19th, 2012

MySQL Workshops

MySQL Developers Training - Monday

MySQL DBA Training - Tuesday

MySQL InnoDB / XtraDB - Wednesday

MySQL Operations – Thursday

Use Discount code DCPL30 to save 30%

Visit <http://bit.ly/dc-training>

Upcoming Percona MySQL Training

Washington, DC – January 16, 2012

Vancouver, Canada - February 6, 2012

Frankfurt, Germany - February 13, 2012

Irvine, CA – February 14, 2012

Denver, CO - February 20, 2012

San Francisco, CA - March 12, 2012

Austin, TX - March 19, 2012

Visit <http://percona.com/training>



Your Name
Your Email Address

We're Hiring!
www.percona.com/about-us/careers/



PERCONA
LIVE

www.percona.com/live