MariaDB

MariaDB, the new branch of MySQL

Froscon
2012-08-26

Michael “Monty” Widenius
MariaDB hacker
monty@askmonty.org
http://mariadb.com/

Things addressed in talk

1) Why MariaDB was created
2) What new in the different MariaDB releases
3) Why next release is called MariaDB 10.0
4) Reasons to switch to MariaDB

Why MariaDB was created

“Save the People, Save the Product”

- To keep the MySQL talent together
- To ensure that a free version of MySQL always exists
- To get one community developed and maintained branch
- Work with Drizzle, Percona (and other MySQL forks/branches) to share knowhow and code.
- Work with MySQL users & companies to together develop MariaDB further

The origin of Maria (DB)

Introducing Maria

Role of Monty Program Ab

- Main driver of MariaDB development and infrastructure.
- Be a home for the core MariaDB developers
- We only do (paid for and free) development on MariaDB and MySQL and 3 level support (bug fixes and very hard cases).
- Some companies with MySQL expertise internally have signed direct support contracts with Monty Program Ab; All other support are done through partners.
- Aim is to not try do 'everything' and take business from partners (as MySQL AB did).
- Monty Program Ab has 30+ partners involved in MySQL and MariaDB
- Several companies are now sponsoring features for MariaDB!
    - Monty Program Ab sponsors MariaDB development with 50 % of our developers time!

*MariaDB **server** is a branch of MySQL*

- User level (data, API, replication..) compatible with MySQL
  - Drop in replacement
  - More plugins, more features, faster, better code quality.
- GPL-only license. C Client library with FOSS exception.
- More open development
  - Source in public repository on launchpad
  - Active external contributors
  - All development plans public on askmonty.org
- Current state
  - MariaDB 5.2 was released as stable in November 2010
  - MariaDB 5.3 was released as stable in April 2012
  - MariaDB 5.5 was released as stable in April 2012
  - MariaDB 10.0 will be released as alpha in September

The MariaDB releases

- MariaDB 5.1 (based on MySQL 5.1)
    - Better build & test system, code cleanups, community patches, new storage engines, table elimination.
- MariaDB 5.2 (based on MariaDB 5.1)
    - Community features that did not go into 5.1:
        - Virtual columns
        - Extended User Statistics
        - Segmented MyISAM key cache (faster multi user!)
- MariaDB 5.3 (based on MariaDB 5.2)
    - Optimizer features (faster subquerier, joins etc)
    - Microsecond, dynamic columns, faster HANDLER etc.
    - Better replication (group commit, more options)
- MariaDB 5.5 (based on MariaDB 5.3 and MySQL 5.5)

Feedback plugin

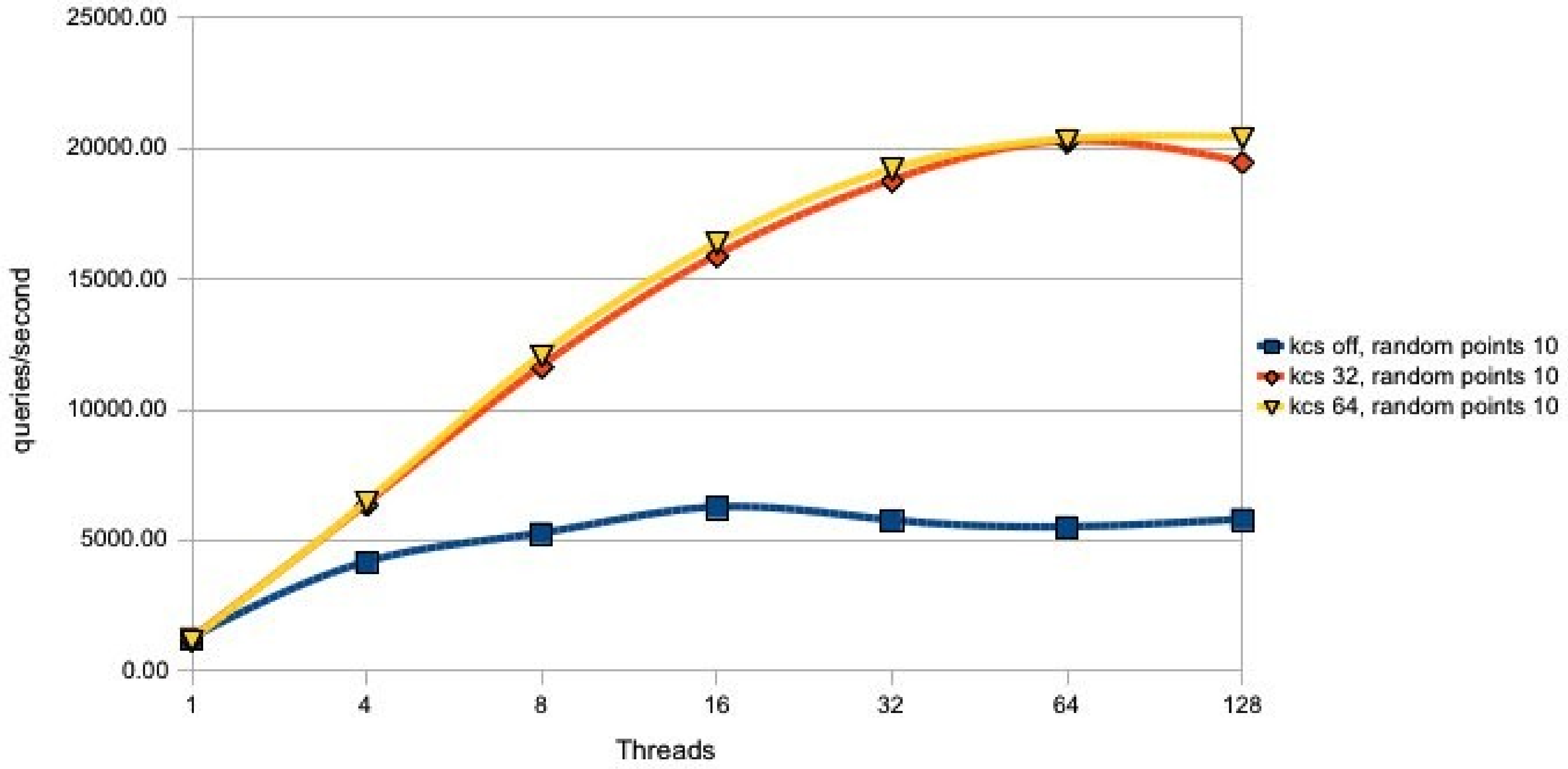
- All recent MariaDB versions has the feedback plugin
- Enable by adding “plugin-load=feedback.so” and --enable-feedback "enable-feedback” to the [mysqld] section in my.cnf.
- Feedback plugin will automatically send a report (basicly SHOW STATUS)a few minutes after a startup and once a week
- This information is used to decide what features should be developed/expanded upon
- For more information see http://kb.askmonty.org/en/user-feedback-plugin/
- For statistics see http://mariadb.org/feedback_plugin/

Major new features in MariaDB 5.2

- SphinxSE: Text search within MariaDB
  - Built-in Sphinx client which allows MariaDB to talk to searchd, run search queries, and obtain search results.
- Virtual columns
  - Columns that are an expression and are calculated on retrieval.
- Extended User Statistics
  - Client, User, Index and Table statistics.
- Segmented MyISAM key cache (see separate slide)
- Pluggable Authentication
- Storage-engine-specific CREATE TABLE
- Very fast 'copying to temp table' phase (speeds up GROUP BY and other complex queries).
- **Group commit** & better recovery for the **Aria** engine.
  - Speeds up multi-user inserts.

Blue means developed by the community

MyISAM Segmented key cache



- Blue line is without segmented key cache.
- Solves one of the major read bottlenecks for MyISAM
- We see up to 250% performance gain depending on the amount of concurrent users.
- Fix applies to all MyISAM usage with many readers!

What's new in MariaDB 5.3

This is the biggest redesign of the MariaDB optimizer in 10 years and it will finally makes subqueries usable in MariaDB.

- Faster subqueries
  - Back porting and extending subquery optimization from MySQL 6.0
  - No materialization for many kinds of subqueries or VIEW's in the FROM clause. SELECT * from (SELECT ....)
  - Caching of subquery results

In applicable cases, you can get 10x – 100x speedups.

- Faster joins (of big tables) thanks to
  - Multi-Read-Range (MRR) access (better than in MySQL 5.6)
  - Batch key access (BKA)
  - Index condition pushdown
  - Classic Hash joins

What's new in MariaDB 5.3

- Microsecond support for NOW(), CAST() and timestamp, time, and datetime columns.
- Windows speed improvements
- Asynchronous IO in XtraDB is redesigned and is now faster, due to the use of IO completion ports.
- A new Windows MSI installer. Includes a GUI-tool, HeidiSQL.
- Lots of new status variables that helps finds out what's wrong
- Progress reports for ALTER TABLE and LOAD DATA INFILE (and some other admin commands)

What's new in MariaDB 5.3

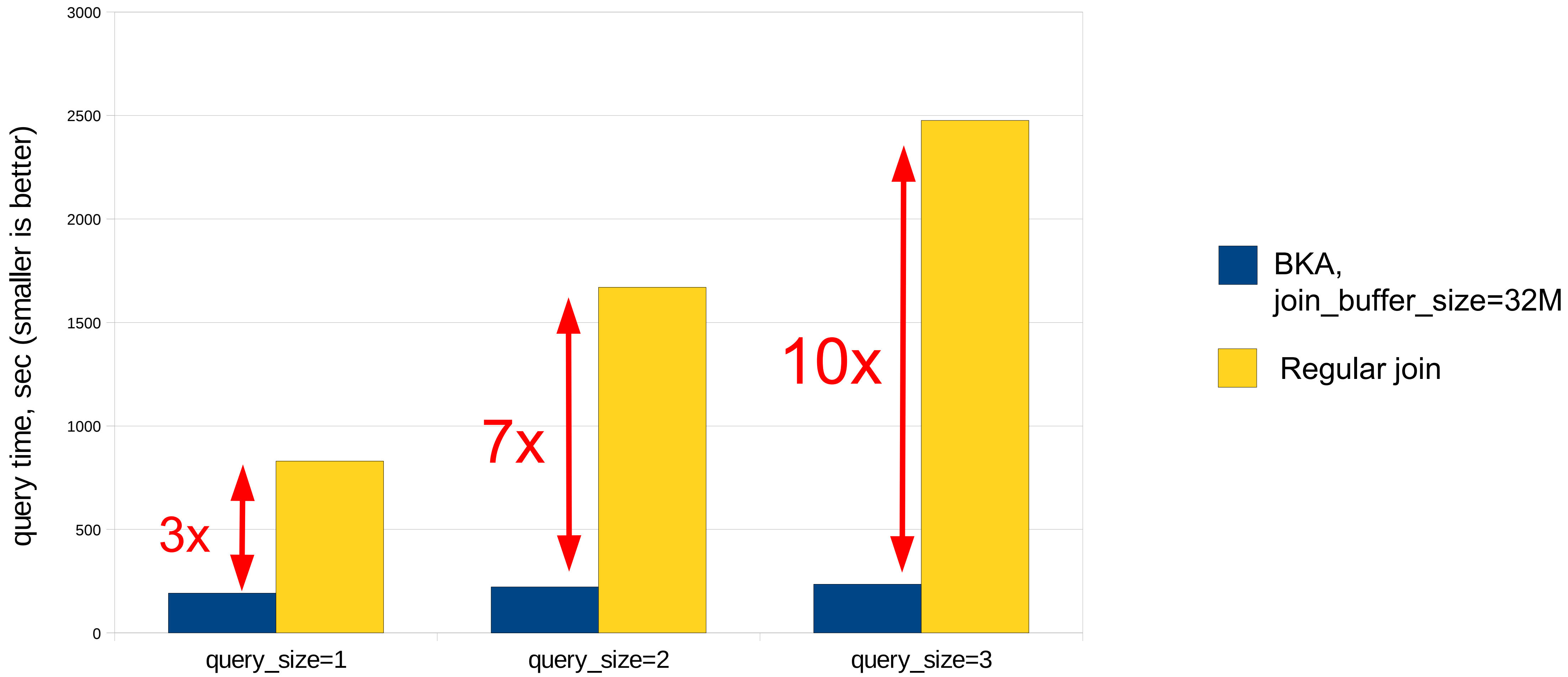
Some common sub queries that are now significantly faster:

- No materialization or materialization with keys:
  - SELECT anything FROM (SELECT ....) AS a WHERE a=...
- Caching of common values (Good if outer_ref has a few values)
  - SELECT (SELECT ... WHERE outer_ref=xxx) FROM ...
- Transformations
  - SELECT * FROM big_table WHERE big_table.col IN (SELECT anything FROM small_table) ->

  Reorders SELECT:s to use sub query as driving table
- Materialization with keys in temporary table also for WHERE
  - SELECT ... WHERE a [NOT] IN (SELECT not-a-key ...)

New Batched Key Access Speedups

- Join benchmark with BKA



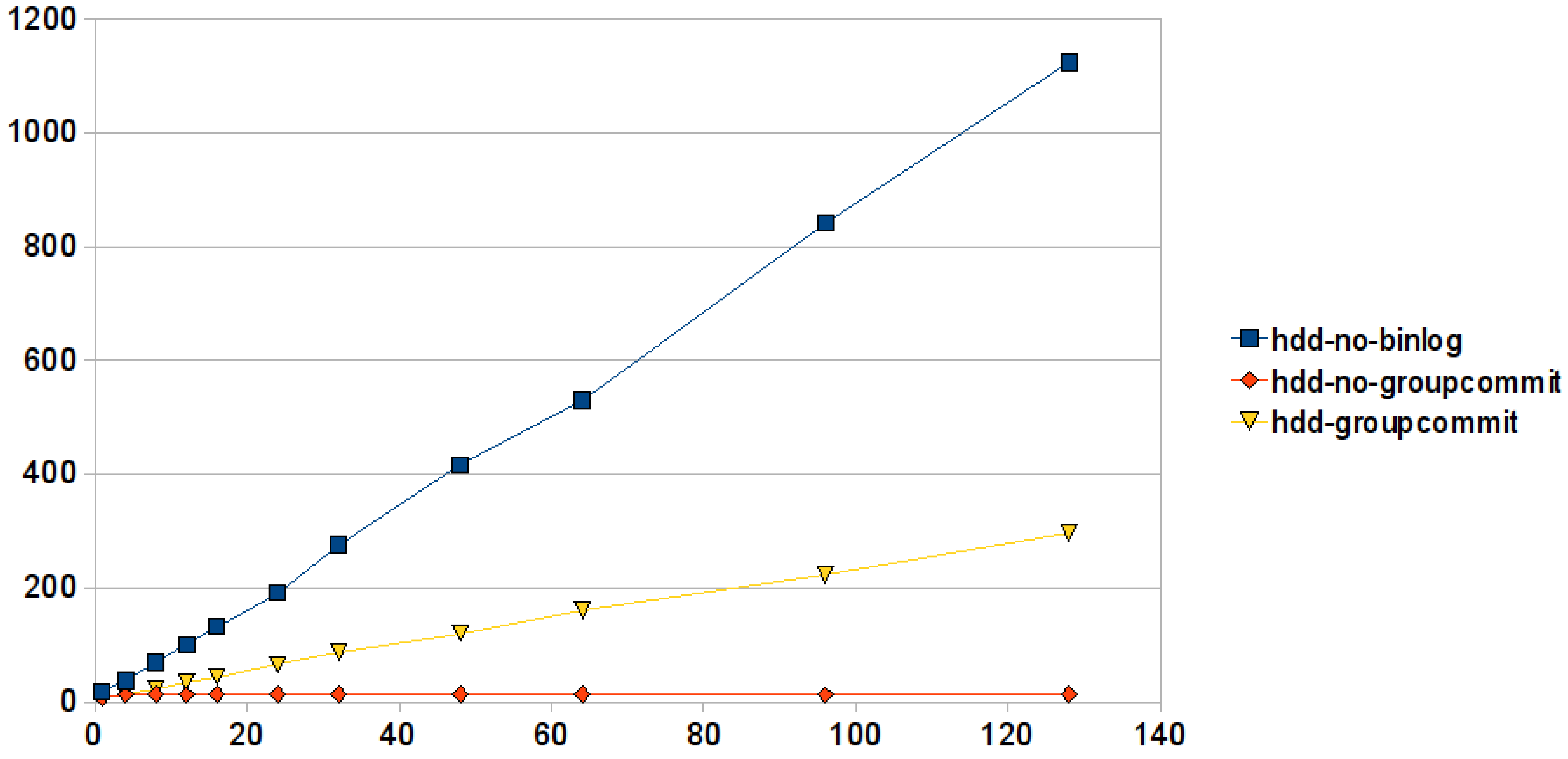
```
select max(l_extendedprice) from orders, lineitem where
o_orderdate between $DATE1 and $DATE2 and
l_orderkey=o_orderkey
```

What's new in MariaDB 5.3

- Full **microsecond** support. This includes TIMESTAMP, TIME DATETIME types, NOW() and all CAST and TIME related functions, replication etc.
- **Group commit** between binary log and storage engines
    - FASTER and safer replication
- Progress report (with PUSH method) for ALTER TABLE, LOAD DATA INFILE, REPAIR, OPTIMIZE & ANALYZE.
- Precise GIS operations.
- Windows installer that includes Windows GUI-tool (HeidiSQL)
- Lots of small optimizations, code cleanups, better error messages and bug fixes.
- For full list, see http://kb.askmonty.org/en/what-is-mariadb-53

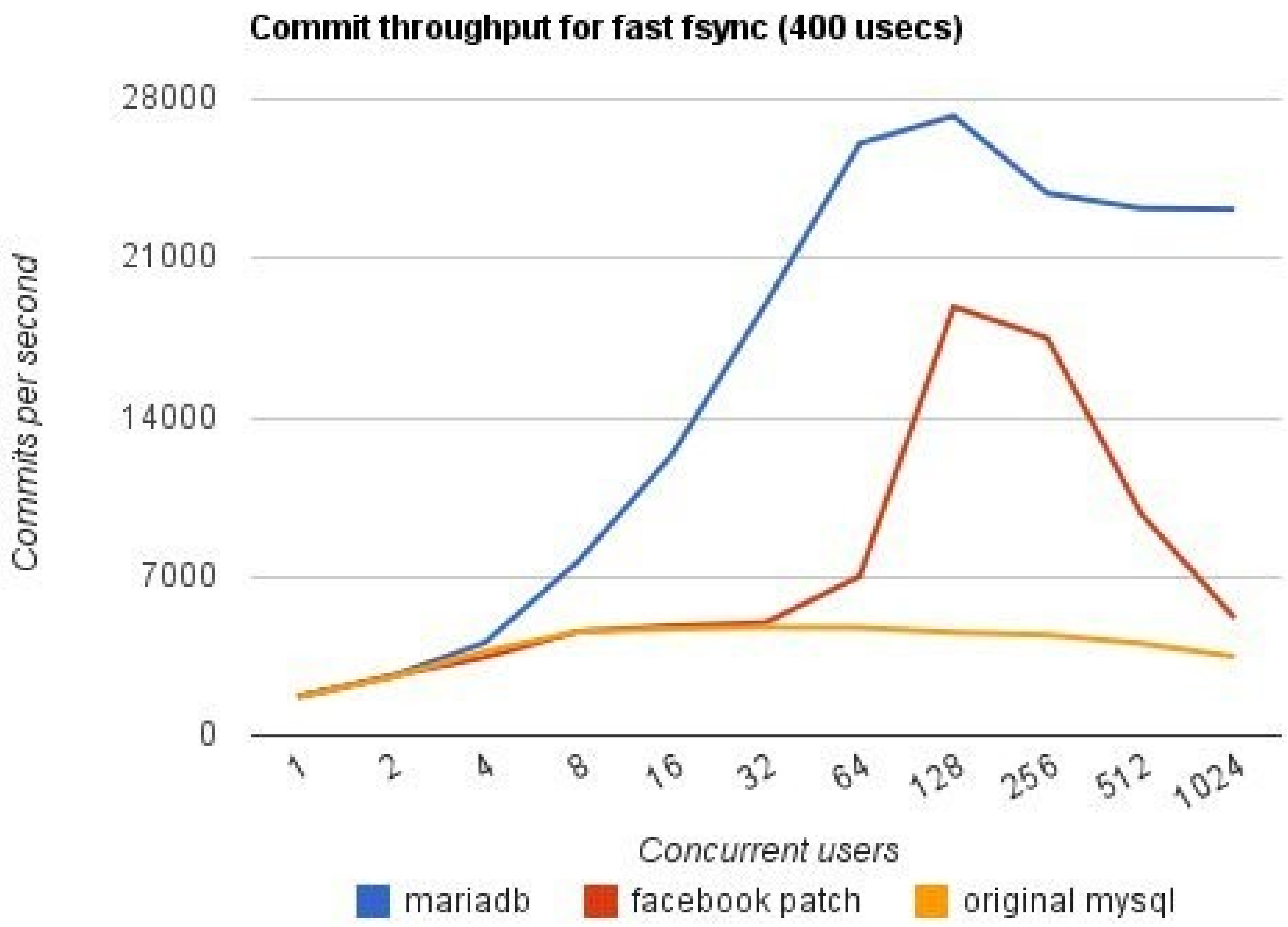
Group commit scales well!

Commits per second vs. number of connections, RAID 1 HDD



- Yellow line shows group commit performance
- Now get scalability, only pay the cost of the 3 * fsync()

Group commit, verified



Source: Marc Callaghan's facebook blog for a server with 400 microsecond fsync latency

MariaDB 5.3 and NoSQL

The main reasons for using NoSQL are:

  - Handling of unstructured data (not everything is table and fixed number of columns)
  - Faster replication (usually with 'unconventional' shortcuts)

- The same way MySQL with it's storage engine interface can handle both transactional and datawarehousing , we are extending MariaDB to be a bridge between SQL and NoSQL.

- MariaDB 5.3 has now even better “NoSQL” support:
  - Faster HANDLER commands; HANDLER READ now also work with prepared statements.
  - HandlerSocket compiled in (Direct access to InnoDB)
  - Dynamic columns (each row can have different set of columns)

HANDLER READ improvements

- Streamlined HANDLER READ interface
- Added support for prepared statements
- Added support for MEMORY tables.

Effect is:

- All HANDLER READ calls are now 7% faster
- 20-50% speedup when using prepared statements and better concurrency.
- You can now get up to 530,000 queries/second trough SQL with NO-SQL commands (60% of HandlerSocket).

Stephane Varoqui's blog:
http://varokism.blogspot.com/2011/01/20-to-50-improvement-in-mariadb-53.html

SQL doesn't solve all common problems

The (web) store problem:
All items needs: ID, Type, Price, Country, Manufacturer)

A T-Shirt has the following additional properties:
Size, color...
A computer has the following additional properties:
CPU, MHz, memory, Watt...

There is no easy way to store many different types into a relational database!
(It will not work by having one table/types as joins becomes impossible to manage).

Dynamic columns in MariaDB 5.3

- With dynamic columns all extra columns are stored in one or many packed blobs, maintained by the database.
- You can instantly add more columns, remove or query them for a row.
- You can access columns in the server or retrieve the full blob to the client and manipulate it there.
- You can use virtual columns to create indexes on some values.
    - True indexes for dynamic columns is planned for later.
- Implemented trough functions to enable use by ODBC etc.
- First implementation uses an integer to access columns.

Dynamic columns in MariaDB 5.3

- Simple set of functions (available in server and client):
    - COLUMN_CREATE(column_nr, value,[column_nr,....])
    - COLUMN_ADD(blob,column_nr, value, [column_nr,...])
    - COLUMN_DELETE(blob, column_nr, column_nr...);
    - COLUMN_EXISTS(blob, column_nr);
    - COLUMN_LIST(blob, column_nr);
    - COLUMN_GET(blob, column_nr, type);

As a proof of concept we are working to create an experimental storage engine for **Cassandra** where we use dynamic columns as a bridge.

Subquery strategies in 5.3

- Flattening into JOINs (semijoin):
  - Duplicate elimination, Pull out, Loose scan, First match, Materialization-scan
- Materialization for non-correlated subqueries
  - Subqueries with NOT, GROUP BY, ORDER BY, in the SELECT clause
  - efficient NULL-aware materialized execution
- Derived tables
  - Derived tables are merged as views
  - Late materialization of derived tables and views
  - Dynamic indexing
- Subquery caching
- Cost-based/manual choice of most strategies
- Fast EXPLAIN with subqueries

Subquery flattening

- Basic idea
  - Transform subquery predicates into JOINs

  -
    ```
    SELECT ...
    FROM outer_table
    WHERE outer_col IN (SELECT inner_col
                        FROM inner_table
                        WHERE subq_where)
          AND outer_where;

    SELECT …
    FROM outer_table [SEMI]JOIN inner_table
    WHERE subq_where AND outer_where;
    ```
  - No:
    - GROUP
    - Aggregation
    - Disjunction in the WHERE

Subquery flattening

- Basic idea
  - Transform subquery predicates into JOINs

  -
    ```
    SELECT ...
    FROM outer_table
    WHERE outer_col IN (SELECT inner_col
                        FROM inner_table
                        WHERE subq_where)
          AND outer_where;

    SELECT …
    FROM outer_table [SEMI]JOIN inner_table
    WHERE subq_where AND outer_where;
    ```
  - No:
    - GROUP
    - Aggregation
    - Disjunction in the WHERE

Pullout strategy

Query: orders from customers with negative balance:

```
SELECT * FROM orders
WHERE o_custkey IN
      (SELECT c_custkey FROM customer
       WHERE c_acctbal < -500);
```

MariaDB 5.2 (any MySQL): 45 sec (slow)

```
+--+-----------+--------+-----+-----------+----+-------+------------------------+
|id|select_type|table   |type |key        |ref |rows   |Extra                   |
+--+-----------+--------+-----+-----------+----+-------+------------------------+
| 1|PRIMARY    |orders  |index|i_o_custkey|NULL|1493631|Using where; Using index|
| 2|SUBQUERY   |customer|range|c_acctbal  |NULL|  10536|Using where; Using index|
+--+-----------+--------+-----+-----------+----+-------+------------------------+
```

MariaDB 5.3: 0.43 sec (faster ~ 100x)

```
+--+-----------+--------+-----++----------+--------------------------+-----+------------------------+
|id|select_type|table   |type |key        |ref                       |rows |Extra                   |
+--+-----------+--------+-----+-----------+--------------------------+-----+------------------------+
| 1|PRIMARY    |customer|range|c_acctbal  |NULL                      |10536|Using where; Using index|
| 1|PRIMARY    |orders  |ref  |i_o_custkey|dbt3sf1.customer.c_custkey|    7|Using index             |
+--+-----------+--------+-----+-----------+--------------------------+-----+------------------------+
```

Optimizations comparison

Feature	MariaDB 5.3/5.5	MySQL 5.5	MySQL 5.6
Index Condition Pushdown (ICP)	Yes		Yes
Disk-sweep Multi-range read (DS-MRR)	Yes		Yes
DS-MRR with Key-ordered retrieval	Yes		
Index_merge / Sort_intersection	Yes		
Cost-based choice of range vs. index_merge	Yes		
ORDER BY ... LIMIT <small_limit>	(In 10.0)		Yes
Use extended (hidden) primary keys for innodb/xtradb	5.5		
Batched key access (BKA)	Yes		Yes
Block hash join	Yes		
User-set memory limits on join buffers	Yes		
Apply early outer table ON conditions	Yes		
Null-rejecting conditions tested early for NULLs	Yes		

Optimizations comparison

Feature	MariaDB 5.3/5.5	MySQL 5.5	MySQL 5.6
Subquery: In-to-exists	Yes	Yes	Yes
Subquery: Semi-join	Yes		Yes
Subquery: Materialization	Yes		Yes
Subquery: NULL-aware Materialization	Yes		
Subquery: Cost choice of materialization vs. in-to-exists	Yes		
Subquery: Cache	Yes		
Subquery: Fast explain with subqueries	Yes		
Delayed materialization of derived tables / materialized views	Yes		Yes
Instant EXPLAIN for derived tables	Yes		Yes
Derived Table with Keys optimization	Yes		Yes
Fields of merge-able views and derived tables used in equality optimizations	Yes		

Optimizations comparison

Feature	MariaDB 5.3/5.5	MySQL 5.5	MySQL 5.6
LIMIT ROWS EXAMINED rows_limit	5.5		
Systematic control of all optimizer strategies	Yes		Partial
Explain for DELETE, INSERT, REPLACE, and UPDATE			Yes
EXPLAIN in JSON format			Yes
More detailed and consistent EXPLAIN for subqueries	Yes		

What's new in MariaDB 5.5

- Significantly more efficient thread pool
- Non-blocking client API Library (MWL#192)
- @@skip_replication option (MWL#234)
  - Run some statements without replication
- SphinxSE updated to version 2.0.4 (+.
- Extended Keys support for XtraDB and InnoDB
- New LIMIT ROWS EXAMINED optimization.
  - Limits max number rows examined for a query
- Variables replicate_do_*, replicate_ignore_*, and replicate_wild_* have been made dynamic
- New plugin to log SQL level errors.
- Updates to performance schema are not replicated

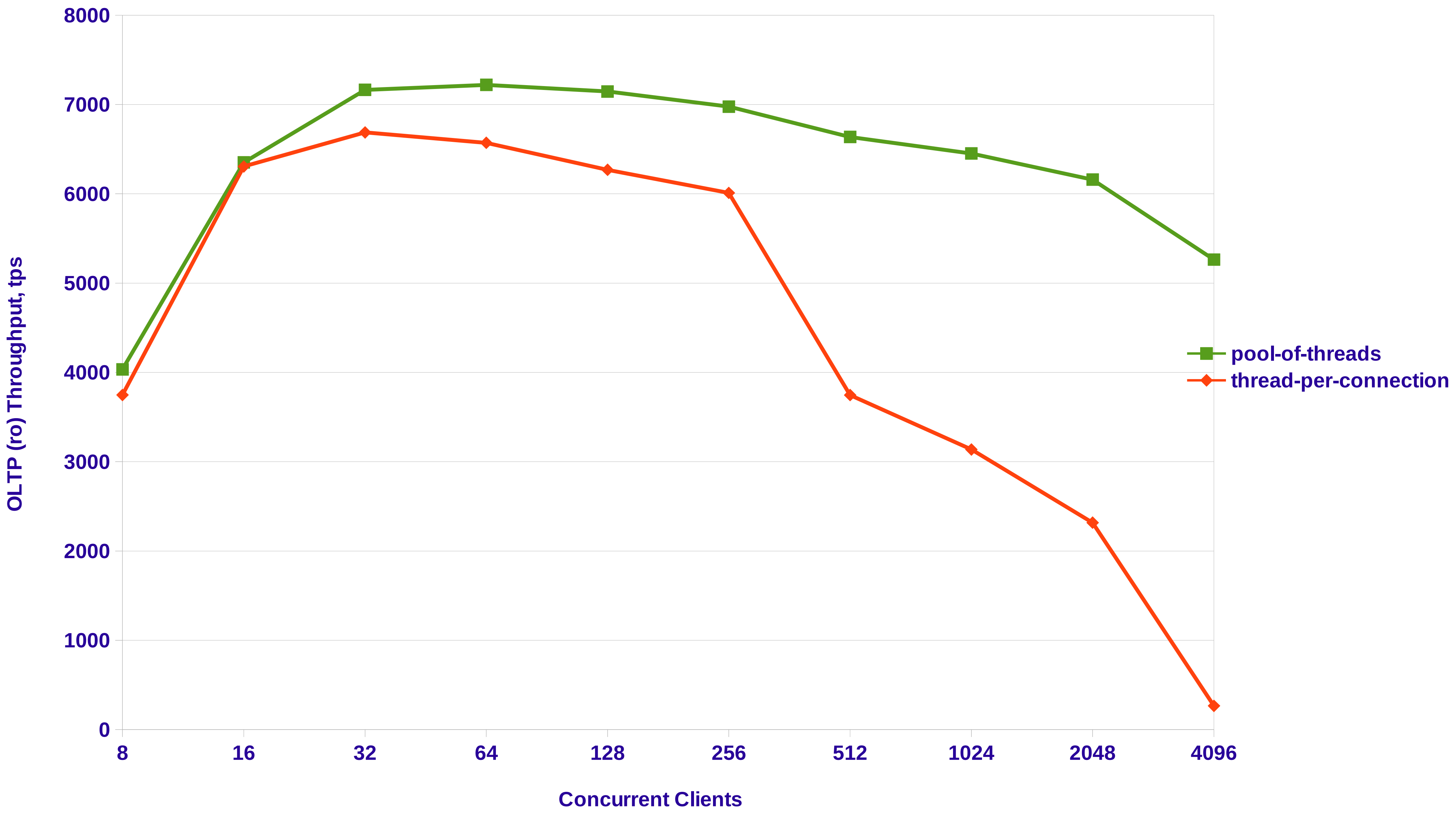
New thread pool for 5.5

Thread pools solves a couple of problems:

- Allows you to limit the number of worker threads at your machines peek performance.
- More fair scheduler; Less query time distribution
- If too many queries, machine can run at 1% of peek performance

- New tread pool for 5.5: (Tested with 24 CPU's):
  - Always better on Windows
  - Better one Linux than thread-per-connection after 1024 connections
  - Much less performance degradation when more connections (60% performance instead of 1%)

New thread pool for 5.5



MariaDB-galera

We are working closely with Codership to release MariaDB 5.5 with Galera (a multi-master solution).

MariaDB 5.5 galera tree already exist. We should have binaries next week.

Why MariaDB 10.0

- MariaDB 5.5 already have most (+ a lot more) of the optimizer features of MySQL 5.6
- MariaDB 5.5 is already a superset of MySQL 5.5. MySQL 5.6 will only have a fraction of the MariaDB 5.5 new features.
- A full merge of MySQL 5.6 into MariaDB 5.6 is a one year project as a lot of the code has to be completely rewritten.
  - Features and usable code are removed, either intentionally or by mistake
  - New code is way to complex (you can do same thing in a fraction of the code)
  - Lots of new introduced bugs we have to get rid of.
  - It's clear that some of the new MySQL programmers doesn't understand the current code (see Kristian Nielsen's blog)
  - A lot of the new code is re-factoring we don't want to have.

→ Better to do the merge in 3 steps into 10.0, 10.1 and 10.2

MariaDB 10.0 trees

- Work on 10.0 is already started and one can follow or participate on it by looking at the trees in launchpad:
  - 10.0-base tree holds new and rewritten features
  - 10.0 tree will be created during next week (as soon as all test passes for my development 10.0-monty tree).

We plan to release an alpha version of MariaDB 10.0 in September.
10.0, 10.1 and 10.2 are planned to be released as GA with ½ years interval's.

What are planned for MariaDB 10.0 ?

Things already back ported from MySQL 5.6:

- All InnoDB changes (done)
- Performance schema changes (done)
- Read only transaction (significant InnoDB optimization) (done)

5.6 features that are reimplemented:

- Better error message (with system error string) (done)
- NOW() as default value for datetime (in progress)
- Global transaction ID for replication

New features:

- SHOW EXPLAIN (see what other thread is doing) (done)
- Multi source (one slave can have many masters)
- Faster ALTER TABLE with UNIQUE index (in progress)
- Even faster group commit (in progress)

What are planned for MariaDB 10.x ?

This list of proposed features is still work in progress, contact us if you want to get something added / ensure something is done!

- GIS
    - OpenGIS compliance
    - Deeper integration of GIS with optimizer
- More online operations
    - Analyze table
    - ALTER ONLINE TABLE
- Compatibility & usability
    - IPv6
    - Query logging and summary per query
    - Audit for specific users

What is planned for MariaDB 10.x ?

- Replication
  - Extend group commit to have on sync per group commit
  - Global transaction id
  - Parallel applying of binary log in slave
  - ALTER TABLE's will be applied in parallel
- Statistics and monitoring
  - Better EXPLAIN
  - Persistent table statistics
  - Log all SQL errors
  - Progress indicator for SELECT
- Store engine for Cassandra/HBASE (in progress)

What is planned for MariaDB 10.x ?

- Optimizer
  - Implement UNION ALL without usage of a temporary table
  - Grace HASH join and Sort merge join (Need sponsors)
- Performance
  - Better multi CPU performance above 16 cores
  - More scalable query cache under higher concurrency
  - Faster VIEW (don't open & parse view for every query)
- Easy of use
  - VARCHAR and BLOB support for memory tables (Sponored: Will be done in September)
  - Table functions

For full list, see http://kb.askmonty.org/v/plans-for-56

Conclusions

- MariaDB has 20 man years of more development than MySQL (and the gap will continue growing).
- MariaDB is maintained by the people that originally created MySQL and has the best knowledge of the MySQL code.
- MariaDB is binary compatible with MySQL, so its trivial to replace MySQL with MariaDB (minutes).
- Reasons to switch to MariaDB
  - Faster queries thanks to XtraDB (InnoDB plugin fork from Percona), a better optimizer and replication and better code.
  - Open source development: Anyone can be part of the development at all stages. Dev meetings are public (next in Dublin in November!)
  - More features, including critical ones like microsecond and dynamic column support.
  - Less risk as MariaDB will not remove features like MySQL is doing (thread pool, storage engines, safemalloc (developer feature), older OS etc)

Want to test MariaDB 5.5?

One of the easiest way to test MariaDB 5.5 for free is to signup at Dogado Jelastic service

http://www.dogado.de/virtualisierung/jelastic-java-cloud-hosting/uebersicht.html

Questions ?

For questions later, use the public MariaDB email list at maria-discuss@lists.launchpad.net or #maria on Freenode.