Qualcomm

Enabling and Optimizing MariaDB on Qualcomm Centriq™ 2400 Arm-based Servers

World's First 10nm Server Processor

Sandeep Sethia

Staff Engineer
Qualcomm Datacenter Technologies, Inc.

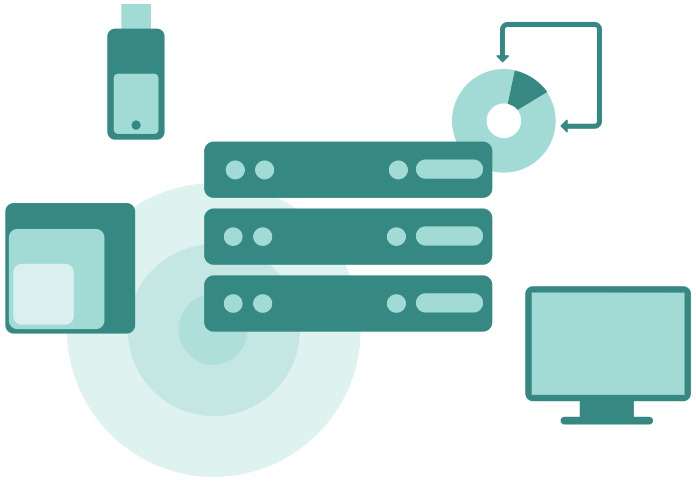
February 25, 2018

MariaDB Developers (Un)conference

Agenda

- Market Context & Background
- MariaDB on Qualcomm Centriq ™ 2400 Processor
  - Methods & Key Performance Indicators
  - Observed Performance Evaluation
  - Bottlenecks
- Next Steps: Perf Improvements & Code Clean-up
- Q & A

The Shift to the Cloud Requires New Business Models & Software Development Approaches



Traditional Enterprise

Monolithic | Stateful | OS or VM bound
Scale up | Silo’d

Cloud Environments

Microservices | Mix of stateless / stateful
Containerized | Scale out | Devops | Multi-tenant

Qualcomm Centriq™ 2400

World's first 10nm server processor

Qualcomm® Falkor™ CPU

5th-Generation Custom Core Design / ARMv8-Compliant

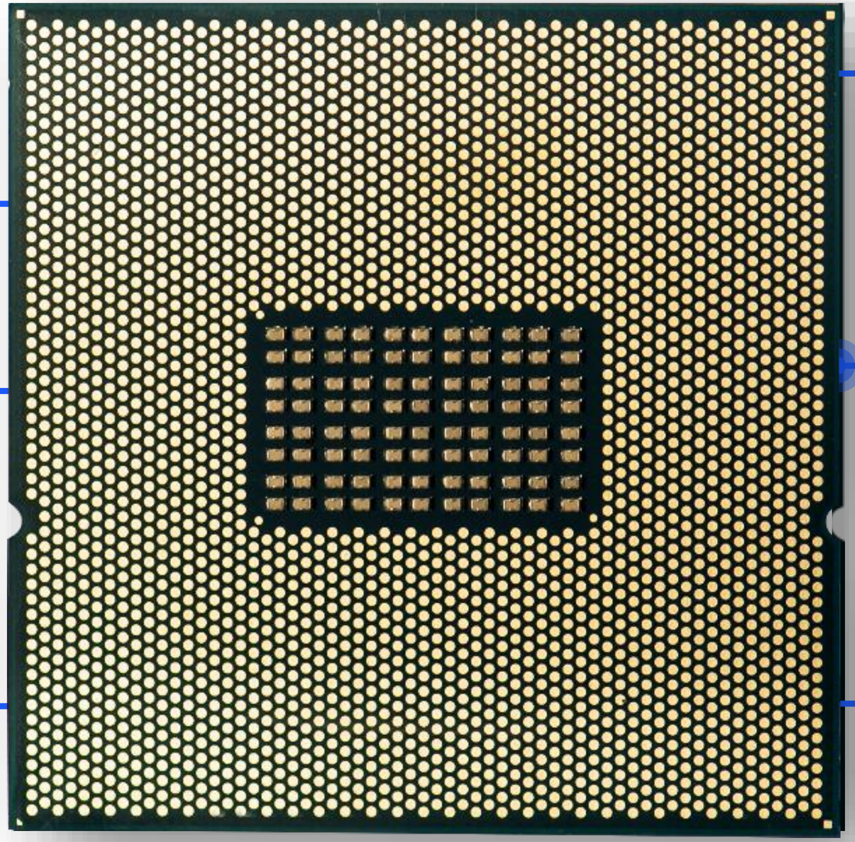
High core count

Up to 48 cores / Single-thread CPUs

Highly Integrated Server SoC

Distributed Architecture / Single Chip
Platform-level Solution / ARM SBSA Level 3 Compliant

Qualcomm Centriq™ 2400 SoC Overview



L3 cache

60 MB unified L3 w/ECC

DDR4 memory

6 Channels w/ECC
2667 MT/s
RDIMM, LRDIMM
1 or 2 DIMMs per Channel

PCIe Gen3

32 Lanes
6 Root Port Controllers

CPU subsystem

Qualcomm© Falkor™ cores
Up to 48 cores
Single-threaded CPU cores
2.6 GHz peak
Unified 512 KB L2 cache w/ECC

SoC

Integrated "south bridge" features
DMA, SATA, USB, I2C,
UART, SPI, GPIO
SBSA Level 3 Compliant

Package

55mm x 55mm LGA
Socketed

MariaDB®

Highly Threaded

Extensible Architecture

High Transactional Performance Leadership

Supports Qualcomm Centriq™ 2400

Highly Multi-core

Scale-out Architecture

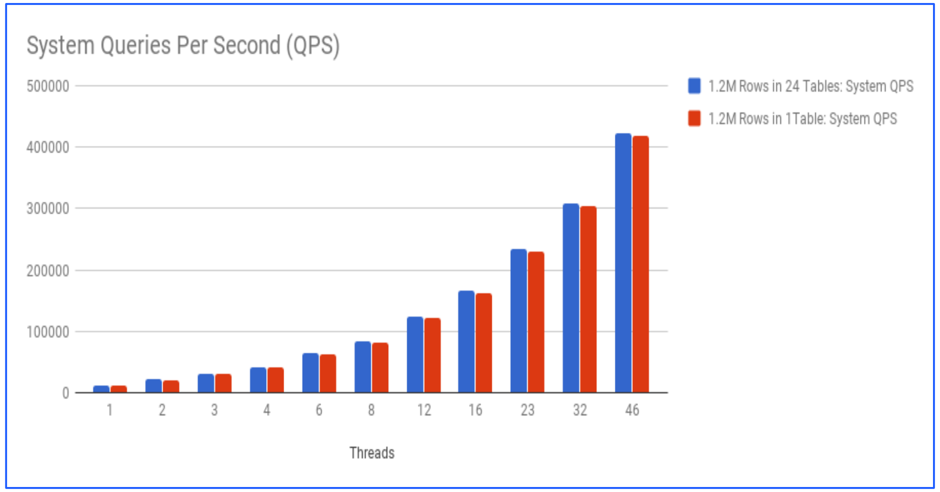
Performance per Thread per Watt Leadership

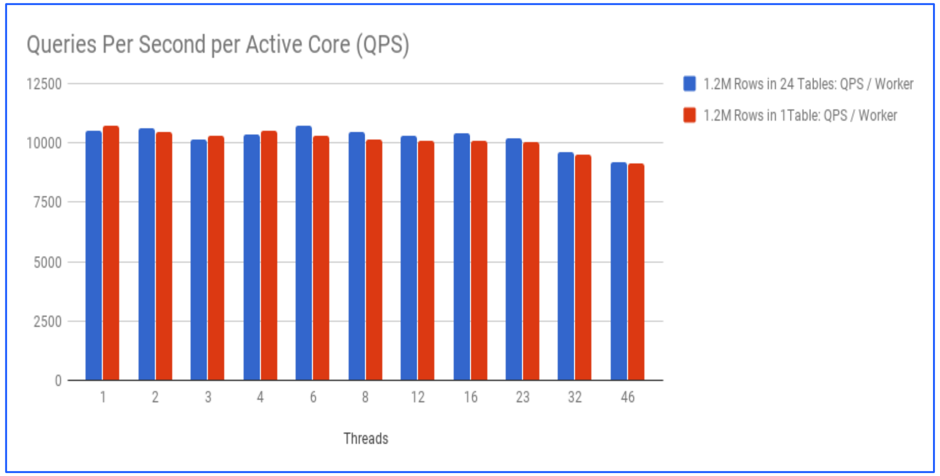
Supports MariaDB 10.3 (RC)

Qualcomm Centriq is a product of Qualcomm Datacenter Technologies, Inc

Evaluating MariaDB on Qualcomm Centriq™ 2400 Processor

MySQL Fork's Tested	MariaDB, Percona Server, MySQL Server
Code Path	Github master branch for any MySQL fork
Workload	Sysbench1.1 and MySQLSLAP
Platforms	Intel x86, Cavium AArch64, and Qualcomm Centriq 2400
Test Type	In-memory (tmpfs and ramdisk)
No. of Clients	1, 4, 8 …. 512 threads
OS	Ubuntu 16.0.4 and Centos7.3
C Compiler	Standard gcc compiler as per OS distribution
Query Type	Read only. Write only. Mixed.
MySQL Configuration	Buffer Pool Size: 30GB Buffer Pool Instance: 30 with 2GB log file size
Storage Engine	InnoDB
Database Size	20 tables with 100k records in each table (created using Sysbench)





Source: https://mariadb.com/resources/blog/mariadb-server-102-now-available-qualcomm-centriqtm-2400-server-processor

MariaDB's Highly-Scale, Highly-Parallel Database Architecture
Aligns Well with Qualcomm Centriq 2400 High-Core, High-Performance Architecture

MariaDB Scales Exceptionally Well on Centriq™ 2400 Processor

- Performance evaluation of write queries with Sysbench
- Currently, in event mutex code while trying to acquire lock in file: **storage/innobase/include/ib0mutex**

MariaDB Uses		MySQL Uses

<pre>bool try_lock()
{
        int32 oldval = MUTEX_STATE_UNLOCKED;
        return(my_atomic_cas32_strong_explicit(&m_lock_word, &oldval
                                                MUTEX_STATE_LOCKED,
                                                MY_MEMORY_ORDER_ACQUIRE,
                                                MY_MEMORY_ORDER_RELAXED));
}</pre>
	VS	<pre>bool tas_lock()
		{
		
		 return(TAS(&m_lock_word, MUTEX_STATE_LOCKED
		                    == MUTEX_STATE_UNLOCKED);
		
		}</pre>

MariaDB Scales Well on Qualcomm Centriq™ 2400 Processor

- In cases of high contention with several threads attempting atomic exchange, throughput degrades
- **Key Learning:** CAS (Compare and Swap) instruction results in better TPS at higher thread counts vs TAS (Test and Set) instruction in mySQL

CAS VS TAS							
Sysbench runs for 300 secs		Greater than 1 CAS is better					
Case	Threads	16	32	64	128	256	512
Oltp_update_index	TPS	1.1	2.61	2.8	2.2	2.1	2.12
Oltp_update_non_index	TPS	1.25	2.88	2.85	2.4	2.35	2.25
Oltp_write_only	TPS	1.4	2.61	3	2.5	2.1	2.12
Oltp_read_only	TPS	1	1	1	1	1	1

2x
Benefit

* Approximately 2x Benefits seen Using MariaDB (CAS) over MySQL (TAS) implementation on Qualcomm Centriq 2400.

Scaling Write Query Performance

- trx_set_rw_mode() is never called for read-only transactions, this is guarded by callers.
- Removing read calls from the "write only trx" critical section results in 5-10% scalability improvement in OLTP index update benchmark on all the platforms.
- Sample line of code change in file storage/innobase/trx/trx0trx.cc.

```
trx_set_rw_mode(
if (!trx->read_only) {    // Removing  the condition.
UT_LIST_ADD_FIRST(trx_sys->rw_trx_list, trx);
ut_d(trx->in_rw_trx_list = true);
} // Removing the braces.
               )
```

* Thanks to Sergey Vojtovich of MariaDB who recently upstreamed this patch.

Scaling Complex Join Queries -- Atomic Operations

- InnoDB uses excessive atomic operations in acquiring shared locks for read queries
- Using MySQLSLAP benchmark we simulated join between 2 on a long character primary key
- That is, the more concurrent threads you add, the faster you can execute the same amount of work (or the the higher throughput is). But ….
- **Key Learning:** After 8 threads, the time to complete the same task increases gradually.
- Basically issue is coming from the concurrent access to the same pages + having a join.
- Perf profiler show heavy usage of Compare and Swap instruction while acquiring read locks.

Scaling Complex Join Queries -- Atomic Operations

- Sample code from the function rw_lock_lock_word_decr in the file “storage/innobase/sync/sync0rw.cc”

```
while (local_lock_word > threshold)
{
        if (os_compare_and_swap_lint(&lock->lock_word,
                                     local_lock_word,
                          local_lock_word - amount))
          {
           return(true);
          }
 local_lock_word = lock->lock_word;
}
...)
```

- Other possible solutions (work in progress)
  - Disabling Adaptive hash index in .cnf improves the performances by 30%.
  - Queued spin lock in user space for r-w locks.

Code Optimization Opportunities in MariaDB

- Optimized InnoDB read-write transactions registry (trx_sys).

  a) It had to acquire trx_sys.mutex 4 times per oltp_update_index query

  b) All 4 trx_sys.mutex locks were eliminated. This was achieved by replacing global data structures protected by trx_sys.mutex

  c) This optimisation also required massive ReadView refactoring

- Optimized InnoDB mini transactions.

  a) Mini transaction was committed 5 times per oltp_update_index query. This effectively means log_sys.mutex was acquired 5 times

  b) First two mini transactions (write of the first undo log record and the creation of the undo log header) were combined into one

  c) This effectively reduced number of log_sys.mutex acqusitions from 5 to 4

Code Optimization Opportunities in MariaDB

- Relaxed memory barriers for Arm Platforms.

  a) From SEQ_CST to RELAXED in InnoDB spin wait used by mutexes and rwlocks
  b) From SEQ_CST to minimum required barriers in InnoDB rwlocks
  c) From SEQ_CST to RELAXED in InnoDB monitor routines

- Minor performance improvement: trx_t reference counter is now accessed atomically.

  a) Eliminates 3 atomic operations and 2 memory barriers

- Optimization of **ut_delay()** while doing a back off in spin lock code.

  a) Removing ut_rnd_interval( ) while calling delay function

```
ut_delay(ut_rnd_interval(0, max_delay)); Previous code

ut_delay(max_delay); Latest code pushed
```

Code Optimization Opportunities in MariaDB

- Preference of newer gcc atomic built ins over sync.

```
#if defined (HAVE_GCC_ATOMIC_BUILTINS)
<atomics>
#else if defined (HAVE_GCC_SYNC_BUILTINS)
 <sync>
```

- Add AArch64 optimized crc32c implementation .
    - These instructions will optimize the performance rather than uses table-based lookup.

- TTASFutex Lock release barrier (aarch64) .
    - For lock release use ATOMIC_RELEASE (os_wmb) instead of ATOMIC_ACQUIRE (os_rmb).

Summary

- Based on the Qualcomm Centriq™ 2400 Arm-based 48-core processor, MariaDB is a highly performant solution vis-à-vis other MySQL forms
- Details of our findings can be found in: https://jira.mariadb.org/browse/MDEV-14442
- Submissions and findings have also been posted here at MySQL: https://bugs.mysql.com/

Special thanks to David Thompson and Sergey Vojtovich of MariaDB for their involvement and support

Thank you

Follow us on:

For more information, visit us at:

www.qualcomm.com & www.qualcomm.com/blog