PALOMINODB

Proven Database Excellence

Understanding Performance through Measurement, Benchmarking and Profiling

Presented by:
René Cannaò @rene_cannao
Senior Operational DBA
www.palominodb.com

WHY RUN BENCHMARK?

Looking for bottlenecks

- Hardware
  - Disk, memory, network, etc
- Operating System
  - File System, memory management, drivers, scheduler , etc
- RDBMS
  - SQL Layer , Storage engine layer
- Schema design
  - Indexes, tables structure, data-types
- Queries
  - Badly written queries, wrong logic
- Application

Looking for bottlenecks

- How the various parts of the system interact?
  - Disk IO, RAM, RBMS, etc
- Measure quantities:
  - How do they perform?
  - Where is the most time spent?
  - Which components are busiest?
- Finding the causes of bottlenecks:
  - Why?

Measurement is a requirement to

- choose the right hardware and the right software
- understand behaviors
- understand performance
- correctly tune and configure
- identify advantages and disadvantages
- spot weak points

Your organization may not be interested in the above!

Measurement is a requirement to

- Plan for growth
- Plan for special events
- Understand scalability

Your organization may be interested in the above!

What to benchmark

- Full-stack
  - Benchmark all the infrastructure, web servers, applications, caching layers, databases, network, external resources, etc
  - Gives a better overview
  - Very difficult to implement
  - Results are complex to understand, and may be misleading
- Single-component
  - Benchmark of only a single component of the infrastructure
  - Easier to implement
  - May ignore the real bottleneck
  - Results are easier to understand

Database benchmarks

Single component benchmark

Compare:

- different OS configuration
- different hardware
- different DB systems
- different parameters
- different schema
- different queries
- different workload

Benchmark guidelines

- Define pre-benchmark goals
- Understand the workload to reproduce
- Record everything, even what seems useless, HW info, configuration, versions, table size , …
- Create baselines
  - Every benchmark is a baseline for future ones
- Define post-benchmark targets
- Change only one thing at the time, and rerun the benchmark

TYPES OF BENCHMARK

Performance testing types

- Stress testing
- Load testing
- Endurance testing
- Spike testing

Stress testing

- System tested to its limit
  - Extreme load is used
  - Defines upper boundary
  - Defines capacity

Load testing

- System tested against an expected load
  - Number of users
  - Number of transactions
  - Dataset size

Endurance testing

- Similar to load testing
- Emulating extended time
- Measure stability on long term

Spike testing

- Sudden change in workload
- Defines the system's behavior in such circumstances. Possible cases:
  - The spike is handled with no impact
  - Performance will degrade
  - System will halt/fail/crash

Performance testing goals

- Must be defined before starting any benchmark
- Verify if the system meets defined criteria
- Compare two systems
- Isolate the component of the system that has poor performance

Performance testing goals

- High throughput
- Low latency
- Stability when faced with scalability
- Stability when faced with concurrency

Throughput

- The most widely used measured quantity
- Number of successful transactions per time unit (usually second or minute)
- Normally it focuses only on average result
- Widely used in marketing for comparisons
- Important for Stress Testing

Latency

- Total execution time for a transaction to complete
- Min / Avg / Max response time
- Important in Load Testing

Stability when faced with scalability

- Measures how system performs when scaling
- Finds weak spots before they become a serious problem
- Useful for capacity planning
- Endurance Testing

Stability when faced with concurrency

- Measures how system performs when number of threads/connections changes
- Defines if it is useful to impose an upper limit in number of threads and introduce queues
- Spike Testing

HOW TO RUN A BENCHMARK

Test conditions

How to avoid common mistakes

Run benchmarks on a realistic setup

- Similar/identical hardware
  - CPU, memory, network, IO subsystem
- Same software configuration
- Similar dataset size
- Similar distribution of data
- Similar access pattern
  - Avoid query and data caching
  - Reproduce access distribution
- Compatible number of threads
  - Multi-users and multi-servers

Record everything

- It is better to have useless information than to miss important ones
- Document all the steps to re-run the benchmark
- Configuration : hardware, software version and parameters, data size, data set
- Collect hardware statistics (CPU , disk , network , memory)
- Collect software statistics (GLOBAL STATUS, INNODB STATUS)
- Organize the outputs

Isolate the benchmark setup

- Run benchmarks on a system that is not used for other purposes
- Avoid shared environment (Virtualization, or SAN)
- Ensure that no other applications are using the network
- Identical conditions at starting point (warmup)
- (if necessary) Restart the system before each benchmark
- Disable what can interfere with the benchmark (crons)

Automate benchmark

- Simplify the recurrent execution
- Avoid human mistakes
- Improved documentation
- Easy to reproduce
- Easy to analyze

Analysis of results

- Process all the data collected
- Identify which data provide useful information
- Answer the questions defined in your goals
- Document the result
- Conclusion
- Define targets

BENCHMARK TOOLS

BENCHMARK TOOLS: mysqlslap

mysqlslap

- Load emulator client
- It executes in 3 stages:
  - It creates the tables structure and load data
  - Run tests
  - Cleanup

mysqlslap (examples)

```
$ mysqlslap --only-print --auto-generate-sql
DROP SCHEMA IF EXISTS `mysqlslap`;
CREATE SCHEMA `mysqlslap`;
use mysqlslap;
CREATE TABLE `t1` (intcol1 INT(32) ,charcol1 VARCHAR(128));
INSERT INTO t1 VALUES (...);
INSERT INTO t1 VALUES (...);
....
SELECT intcol1,charcol1 FROM t1;
INSERT INTO t1 VALUES (...);
SELECT intcol1,charcol1 FROM t1;
INSERT INTO t1 VALUES (...);
...
DROP SCHEMA IF EXISTS `mysqlslap`;
```

mysqlslap (examples)

```
$ mysqlslap --only-print --auto-generate-sql | awk '{print $1"
    "$2}' | sort | uniq -c | sort -n -r
   105 INSERT INTO
     5 SELECT intcol1,charcol1
     2 DROP SCHEMA
     1 use mysqlslap;
     1 CREATE TABLE
     1 CREATE SCHEMA
```

- By default, it creates and drop the database `mysqlslap`
- --create-schema=value to specify a different database name
- Each thread performs 100 INSERT during the loads data stage (default for --auto-generate-sql-write-number)
- Each thread performs as many INSERT and SELECT during the test stage (default for --auto-generate-sql-load-type)

mysqlslap (examples)

```
$ mysqlslap --only-print --auto-generate-sql --number-of-
  queries=100 | awk '{print $1}' | egrep '(INSERT|SELECT)' |
  sort | uniq -c | sort -n -r
    154 INSERT INTO
     45 SELECT intcol1,charcol1

$ mysqlslap --only-print --auto-generate-sql --concurrency=5
  --number-of-queries=100 | awk '{print $1}' | egrep '(INSERT|
  SELECT)' | sort | uniq -c | sort -n -r
    154 INSERT INTO
     45 SELECT intcol1,charcol1

$ mysqlslap --only-print --auto-generate-sql --iteration=5
  --number-of-queries=100 | awk '{print $1}' | egrep '(INSERT|
  SELECT)' | sort | uniq -c | sort -n -r
    770 INSERT INTO
    225 SELECT intcol1,charcol1
```

mysqlslap (examples)

```
$ mysqlslap --only-print --auto-generate-sql --number-of-
  queries=100 --auto-generate-sql-write-number=10000 | awk
  '{print $1}' | egrep '(INSERT|SELECT)' | sort | uniq -c |
  sort -n -r
  10054 INSERT
     45 SELECT
```

```
$ mysqlslap --only-print --auto-generate-sql --number-of-
  queries=100 -auto-generate-sql-write-number=10000
  --auto-generate-sql-load-type=read | awk '{print $1}' |
  egrep '(INSERT|SELECT)' | sort | uniq -c | sort -n -r
   9999 INSERT
    100 SELECT
```

mysqlslap (examples)

```
~$ mysqlslap --auto-generate-sql --concurrency=1,2,3 -iteration=10 \
   --number-of-queries=100 -auto-generate-sql-write-number=100000
Benchmark
Average number of seconds to run all queries: 4.522 seconds
Minimum number of seconds to run all queries: 4.216 seconds
Maximum number of seconds to run all queries: 4.648 seconds
Number of clients running queries: 1
Average number of queries per client: 100

Benchmark
Average number of seconds to run all queries: 3.025 seconds
Minimum number of seconds to run all queries: 2.737 seconds
Maximum number of seconds to run all queries: 3.227 seconds
Number of clients running queries: 2
Average number of queries per client: 50

Benchmark
Average number of seconds to run all queries: 2.618 seconds
Minimum number of seconds to run all queries: 2.338 seconds
Maximum number of seconds to run all queries: 2.746 seconds
Number of clients running queries: 3
Average number of queries per client: 33
```

mysqlslap (examples)

```
$ mysqlslap --auto-generate-sql --concurrency=4 \
  --engine=myisam,innodb --number-of-queries=100 \
  --iteration=10 --auto-generate-sql-write-number=100000

Benchmark
Running for engine myisam
Average number of seconds to run all queries: 2.232 seconds
Minimum number of seconds to run all queries: 2.003 seconds
Maximum number of seconds to run all queries: 2.319 seconds
Number of clients running queries: 4
Average number of queries per client: 25

Benchmark
Running for engine innodb
Average number of seconds to run all queries: 5.332 seconds
Minimum number of seconds to run all queries: 5.314 seconds
Maximum number of seconds to run all queries: 5.370 seconds
Number of clients running queries: 4
Average number of queries per client: 25
```

mysqlslap (examples)

```
$ mysqlslap --auto-generate-sql --concurrency=4 \
  --engine=myisam,innodb --number-of-queries=1000 \
  --iteration=10 --auto-generate-sql-write-number=10000

Benchmark
Running for engine myisam
Average number of seconds to run all queries: 2.059 seconds
Minimum number of seconds to run all queries: 1.937 seconds
Maximum number of seconds to run all queries: 2.169 seconds
Number of clients running queries: 4
Average number of queries per client: 250

Benchmark
Running for engine innodb
Average number of seconds to run all queries: 5.604 seconds
Minimum number of seconds to run all queries: 5.560 seconds
Maximum number of seconds to run all queries: 5.659 seconds
Number of clients running queries: 4
Average number of queries per client: 250
```

mysqlslap (examples)

```
SET GLOBAL concurrent_insert=0;

$ mysqlslap --auto-generate-sql --concurrency=4 -engine=myisam,innodb \
  --number-of-queries=1000 --iteration=10 --auto-generate-sql-write-number=10000

Benchmark
Running for engine myisam
Average number of seconds to run all queries: 3.256 seconds
Minimum number of seconds to run all queries: 3.210 seconds
Maximum number of seconds to run all queries: 3.317 seconds
Number of clients running queries: 4
Average number of queries per client: 250

Benchmark
Running for engine innodb
Average number of seconds to run all queries: 5.615 seconds
Minimum number of seconds to run all queries: 5.585 seconds
Maximum number of seconds to run all queries: 5.699 seconds
Number of clients running queries: 4
Average number of queries per client: 250

SET GLOBAL concurrent_insert=1;
```

mysqlslap (examples)

```
mysqlslap --auto-generate-sql --concurrency=4 \
  --engine=memory,myisam,innodb --iteration=10 \
--number-of-queries=1000 -auto-generate-sql-write-number=10000 --auto-
  generate-sql-add-autoincrement

CREATE TABLE `t1` (id serial,intcol1 INT(32) ,charcol1 VARCHAR(128));


Running for engine memory
Average number of seconds to run all queries: 0.035 seconds

Running for engine myisam
Average number of seconds to run all queries: 0.038 seconds

Running for engine innodb
Average number of seconds to run all queries: 0.032 seconds


100x times faster?
InnoDB faster than MEMORY and MyISAM?
```

mysqlslap (examples)

Increasing the duration of the test

```
$ mysqlslap --auto-generate-sql --concurrency=4
  --engine=memory,myisam,innodb --iteration=10 \
    --number-of-queries=10000 --auto-generate-sql-write-
  number=10000 --auto-generate-sql-add-autoincrement

Running for engine memory
Average number of seconds to run all queries: 0.430 seconds

Running for engine myisam
Average number of seconds to run all queries: 0.467 seconds

Running for engine innodb
Average number of seconds to run all queries: 0.327 seconds
```

InnoDB wins!

mysqlslap (examples)

Add more fields:

```
$ mysqlslap mysqlslap -a -c4 -y4 -x4
  --engine=memory,myisam,innodb -i10 --number-of-
  queries=10000 --auto-generate-sql-write-number=10000
  --auto-generate-sql-add-autoincrement

Running for engine memory
Average number of seconds to run all queries: 0.504 seconds

Running for engine myisam
Average number of seconds to run all queries: 0.526 seconds

Running for engine innodb
Average number of seconds to run all queries: 0.468 seconds
```

InnoDB still wins, but has slow down a lot.

mysqlslap (examples)

Increase concurrency:

```
$ mysqlslap mysqlslap -a -c8 -y4 -x4
  --engine=memory,myisam,innodb -i10 --number-of-
  queries=10000 --auto-generate-sql-write-number=10000
  --auto-generate-sql-add-autoincrement

Running for engine memory
Average number of seconds to run all queries: 0.526 seconds

Running for engine myisam
Average number of seconds to run all queries: 0.596 seconds

Running for engine innodb
Average number of seconds to run all queries: 0.657 seconds
```

InnoDB is the slowest

mysqlslap (examples)

Workload: read primary key

```
$ mysqlslap mysqlslap --auto-generate-sql-load-type=key -a
  -c8 -y4 -x4 --engine=memory,myisam,innodb -i10 --number-of-
  queries=10000 --auto-generate-sql-write-number=10000 -auto-
  generate-sql-add-autoincrement

Running for engine memory
Average number of seconds to run all queries: 0.278 seconds

Running for engine myisam
Average number of seconds to run all queries: 0.328 seconds

Running for engine innodb
Average number of seconds to run all queries: 0.324 seconds
```

mysqlslap (examples)

Workload: update primary key

```
$ mysqlslap mysqlslap --auto-generate-sql-load-type=update -a -c8
  -y4 -x4 -engine=blackhole,memory,myisam,innodb -i10 --number-of-
  queries=10000 --auto-generate-sql-write-number=10000 -auto-
  generate-sql-add-autoincrement

Running for engine blackhole
Average number of seconds to run all queries: 0.251 seconds

Running for engine memory
Average number of seconds to run all queries: 0.724 seconds

Running for engine myisam
Average number of seconds to run all queries: 0.939 seconds

Running for engine innodb
Average number of seconds to run all queries: 0.518 seconds
```


BENCHMARK TOOLS: sysbench

sysbench

Multi-threaded benchmark tool for:

- File I/O performance
- Scheduler performance
- Memory allocation and transfer speed
- POSIX threads implementation performance
- Database server performance (OLTP)

sysbench

Intially developed for MySQL , but then extended

Available at:

- https://code.launchpad.net/~sysbench-developers/sysbench/0.5
- http://sysbench.sourceforge.net/
- http://dev.mysql.com/downloads/benchmarks.html

Download:

- bzr branch lp:sysbench
- http://sourceforge.net/projects/sysbench/ (0.4.12)
- http://downloads.mysql.com/source/sysbench-0.4.12.5.tar.gz

Sysbench
installation

```
$ bzr branch lp:sysbench
$ cd sysbench
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
```

sysbench
CPU test

Very simple test to process prime numbers:

```
$ sysbench --test=cpu --cpu-max-prime=20000 run
Running the test with following options:
Number of threads: 1
Primer numbers limit: 20000

General statistics:
    total time:                          54.5050s
    total number of events:              10000
    total time taken by event execution: 54.4840s
    response time:
         min:                                  5.20ms
         avg:                                  5.45ms
         max:                                 45.02ms
         approx.  95 percentile:               5.69ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   54.4840/0.00
```

sysbench
CPU test

Prime numbers, multi-threads:

```
$ sysbench --num-threads=4 --test=cpu --cpu-max-prime=20000 run
Running the test with following options:
Number of threads: 4

Primer numbers limit: 20000

General statistics:
    total time:                          16.5565s
    total number of events:              10000
    total time taken by event execution: 66.1771s
    response time:
         min:                                  5.89ms
         avg:                                  6.62ms
         max:                                101.86ms
         approx.  95 percentile:               8.43ms

Threads fairness:
    events (avg/stddev):           2500.0000/136.15
    execution time (avg/stddev):   16.5443/0.00
```

sysbench
CPU test

Prime numbers, multi-threads:

```
$ sysbench --num-threads=8 --test=cpu --cpu-max-prime=20000 run
Running the test with following options:
Number of threads: 8

Primer numbers limit: 20000

General statistics:
    total time:                          16.1659s
    total number of events:              10000
    total time taken by event execution: 129.1411s
    response time:
         min:                                  5.89ms
         avg:                                 12.91ms
         max:                                 49.95ms
         approx.  95 percentile:              25.30ms

Threads fairness:
    events (avg/stddev):           1250.0000/54.65
    execution time (avg/stddev):   16.1426/0.01
```

sysbench
threads test (scheduler)

Lock mutex => Yield CPU => Run queue => Unlock mutex

```
$ sysbench --num-threads=64 --test=threads run
Running the test with following options:
Number of threads: 64

General statistics:
    total time:                          3.4055s
    total number of events:              10000
    total time taken by event execution: 217.2030s
    response time:
         min:                                  0.56ms
         avg:                                 21.72ms
         max:                                162.65ms
         approx.  95 percentile:              59.37ms

Threads fairness:
    events (avg/stddev):           156.2500/10.98
    execution time (avg/stddev):   3.3938/0.01
```

sysbench
mutexs test

Threads trying to acquire the same set of mutexs

Measure mutexes implementation.

```
$ sysbench --num-threads=128 --test=mutex --mutex-locks=100000 --mutex-loops=100
   --mutex-num=2048 run

Running the test with following options:
Number of threads: 128

General statistics:
    total time:                          10.4538s
    total number of events:              128
    total time taken by event execution: 1274.8660s
    response time:
         min:                                  8069.55ms
         avg:                                  9959.89ms
         max:                                 10453.19ms
         approx.  95 percentile:               10436.39ms

Threads fairness:
    events (avg/stddev):           1.0000/0.00
    execution time (avg/stddev):   9.9599/0.52
```

sysbench
memory test

Sequential read or write memory

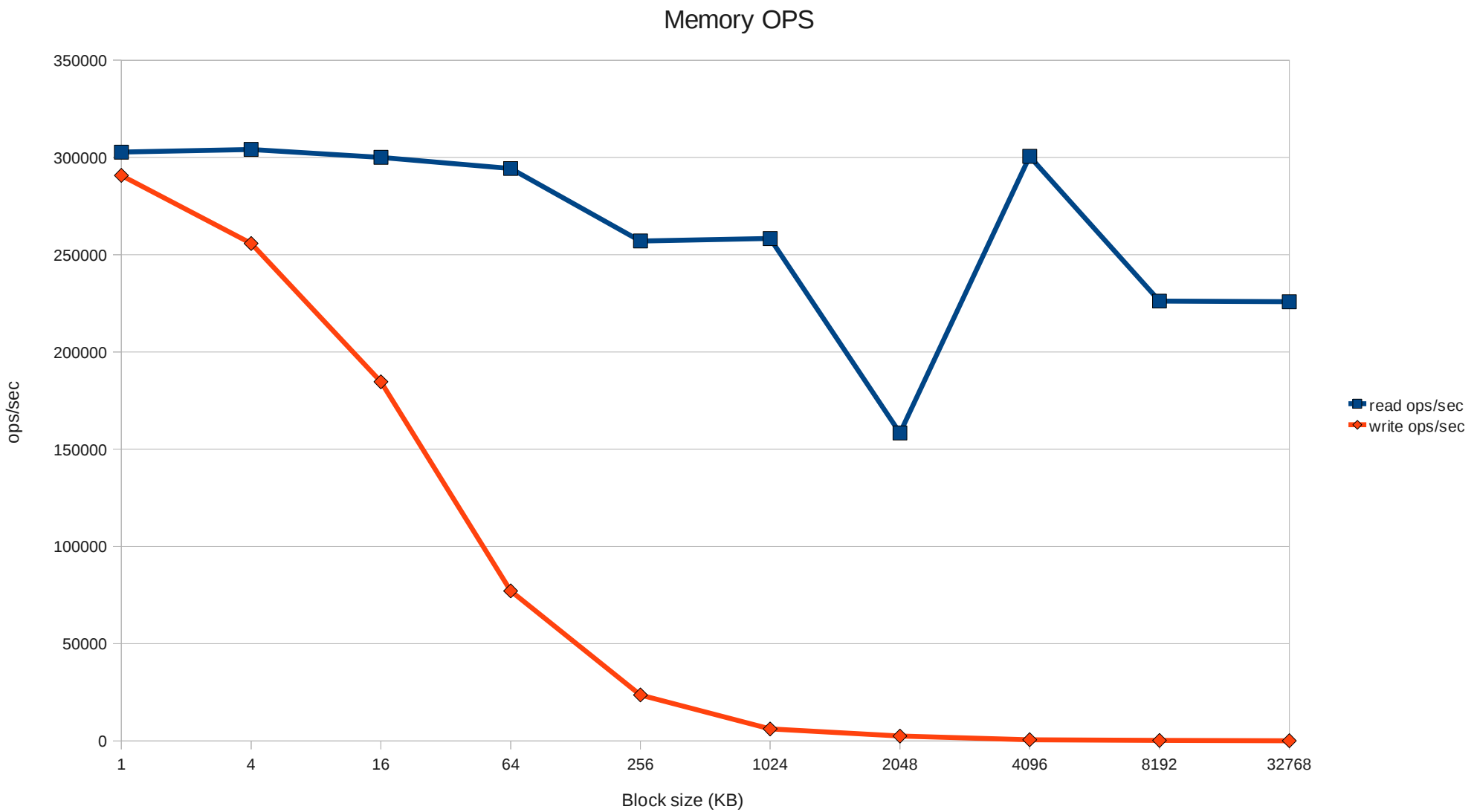
```
$ sysbench --num-threads=1 --test=memory run –memory-block-size=16k
  --memory-total-size=4G --memory-oper=read
Running the test with following options:
Number of threads: 1

Operations performed: 262144 (302035.23 ops/sec)
4096.00 MB transferred (4719.30 MB/sec)

General statistics:
    total time:                          0.8679s
    total number of events:              262144
    total time taken by event execution: 0.3994s
    response time:
         min:                                  0.00ms
         avg:                                  0.00ms
         max:                                  0.11ms
         approx.  95 percentile:               0.00ms

Threads fairness:
    events (avg/stddev):           262144.0000/0.00
    execution time (avg/stddev):   0.3994/0.00
```

sysbench
memory test



sysbench
CPU test

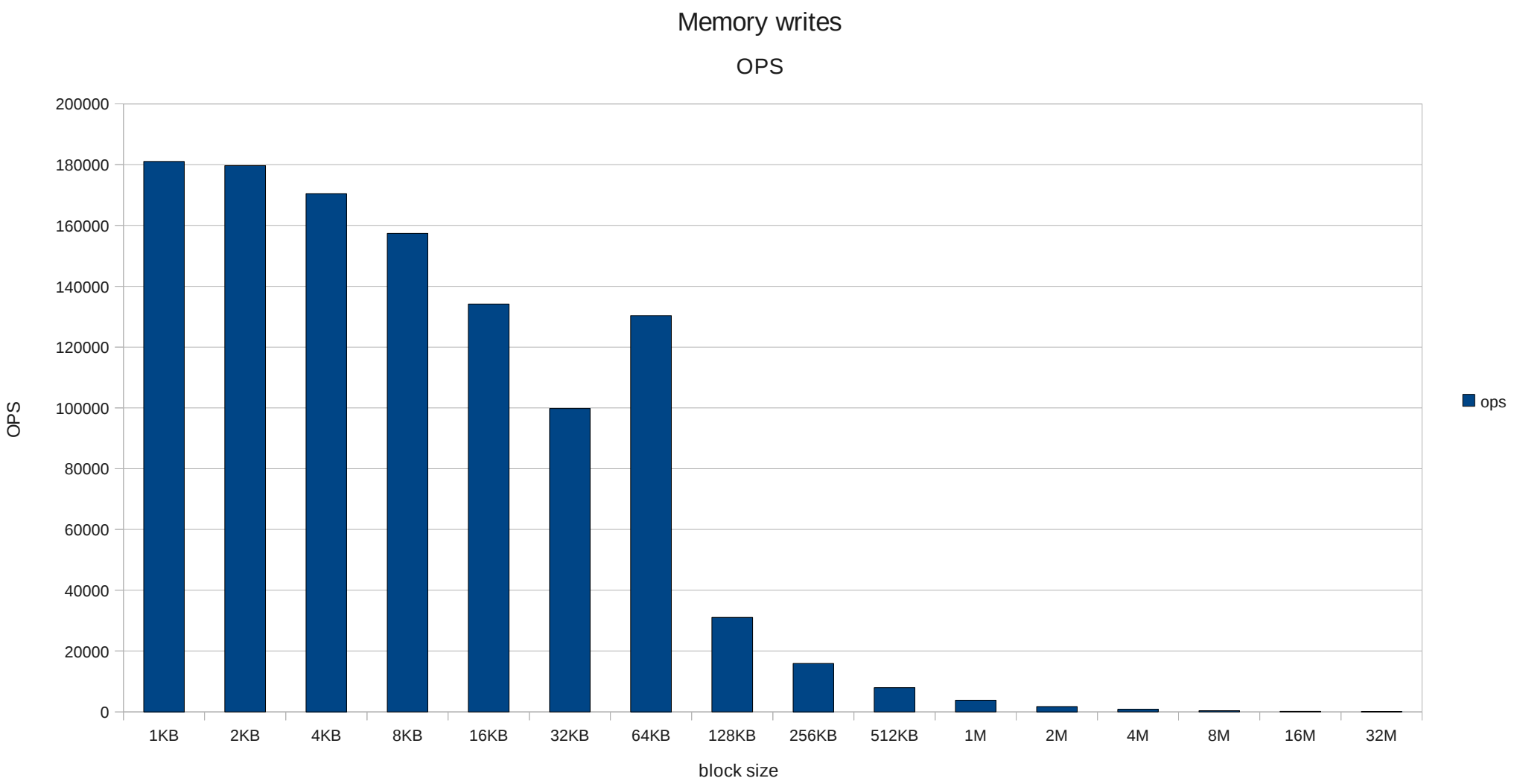
Numbers lie!

The previous test reports reads at 1.8TB/s
with 8MB block size.

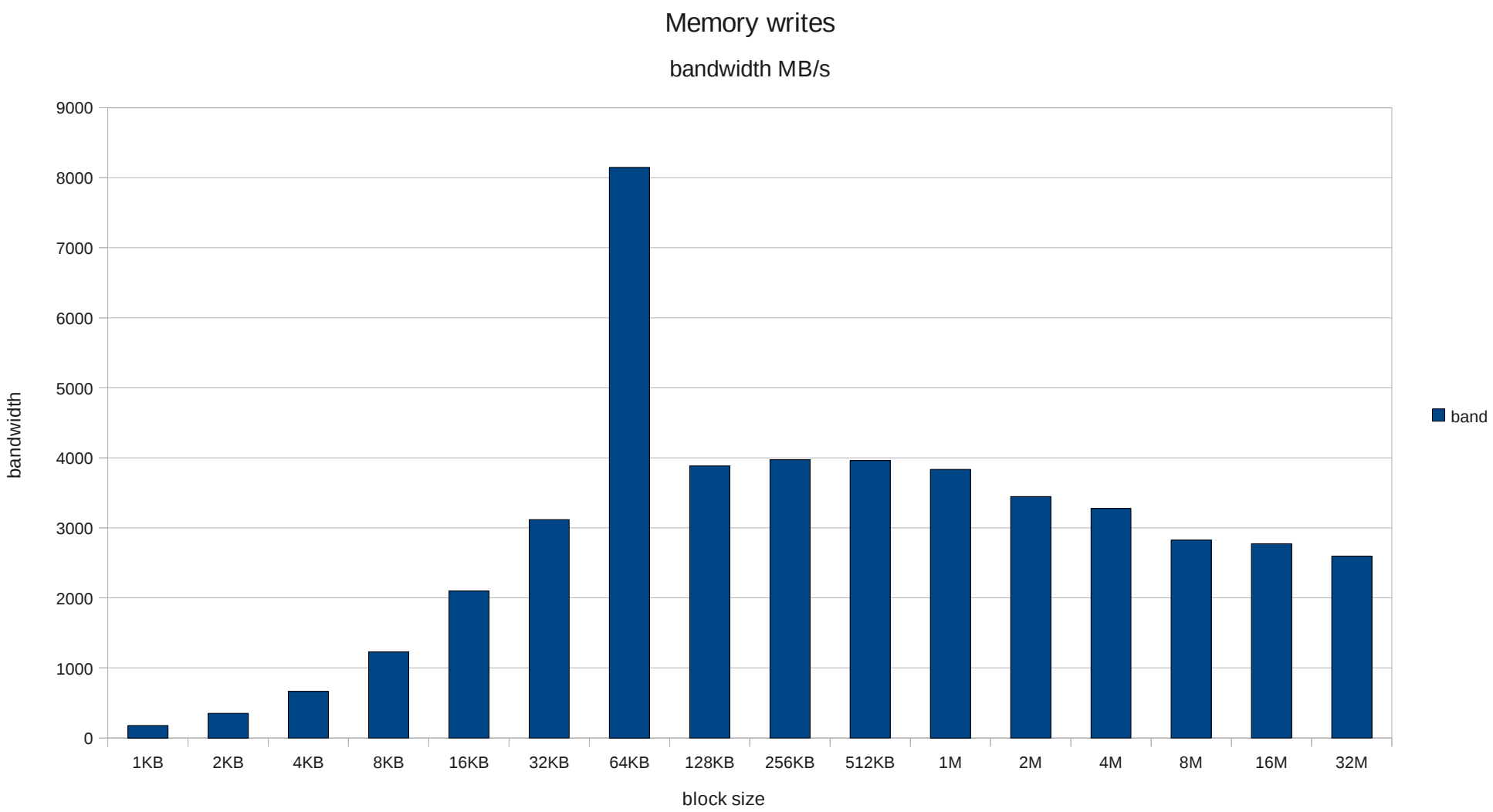
With 32MB block size, reads go to 7.2TB/s

Constant OPS is the real bottleneck.

sysbench
memory test



sysbench
memory test



sysbench
fileio test

2 stages test:

- prepare

  files are created on filesystem

- run

  test is executed on previously created files

sysbench
fileio test

Supported I/O tests: (--file-test-mode)

- seqwr : sequential write
- seqrewr : sequential rewrite
- seqrd : sequential read
- rndrd : random read
- rndwr : random write
- rndrw : random read/write

sysbench
fileio test

Other options:

- --file-num : number of files
- --file-block-size
- --file-total-size : all files combined
- --file-io-mode : sync , async, mmap
- --file-extra-flags (odirect)
- --file-fsync-freq : frequency of fsync()
- --file-fsync-all
- --file-rw-ratio : R/W ratio in rndrw mode

sysbench
fileio test

Prepare the files:

```
$ sysbench --test=fileio --file-num=4
   --file-total-size=1G prepare


4 files, 262144Kb each, 1024Mb total
Creating files for the test...
Extra file open flags: 0
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
1073741824 bytes written in 40.32 seconds (25.40 MB/sec).
```


Sysbench - fileio test

```
sysbench --test=fileio --file-num=1 --file-total-size=2G --file-fsync-all=on
   --file-test-mode=seqrewr --max-time=100 –file-block-size=4096
    --max-requests=0 run

Number of threads: 1
Extra file open flags: 0
1 files, 2Gb each
2Gb total file size
Block size 4Kb
Calling fsync() after each write operation.
Using synchronous I/O mode
Doing sequential rewrite test

Operations performed:  0 Read, 524288 Write, 524288 Other = 1048576 Total
Read 0b  Written 2Gb  Total transferred 2Gb  (52.616Mb/sec)
13469.70 Requests/sec executed

Test execution summary:
    total time:                          38.9235s
    total number of events:              524288
    total time taken by event execution: 38.2213
    per-request statistics:
         min:                                  0.06ms
         avg:                                  0.07ms
         max:                                  8.60ms
         approx.  95 percentile:               0.12ms
```

Sysbench - fileio test

```
sysbench --test=fileio --file-num=29 --file-total-size=290G --file-fsync-all=on
  --file-test-mode=rndrw --max-time=100 --file-block-size=16384 --max-requests=0 run

Number of threads: 1
Extra file open flags: 0
29 files, 10Gb each
290Gb total file size
Block size 16Kb
Number of random requests for random IO: 0
Read/Write ratio for combined random IO test: 1.50
Calling fsync() after each write operation.
Using synchronous I/O mode
Doing random r/w test

Operations performed:  449824 Read, 299883 Write, 299883 Other = 1049590 Total
Read 6.8638Gb  Written 4.5759Gb  Total transferred 11.44Gb  (117.14Mb/sec)
 7497.05 Requests/sec executed

Test execution summary:
    total time:                          100.0003s
    total number of events:              749707
    total time taken by event execution: 98.9223
    per-request statistics:
         min:                                  0.01ms
         avg:                                  0.13ms
         max:                                  7.34ms
         approx.  95 percentile:               0.16ms
```

Sysbench - fileio test

```
sysbench --num-threads=8 --test=fileio --file-num=29 –file-total-size=290G
    --file-fsync-all=on --file-test-mode=rndrw --max-time=100 –file-rw-ratio=4
    --file-block-size=16384 --max-requests=0 run

Number of threads: 8
Extra file open flags: 0
29 files, 10Gb each
290Gb total file size
Block size 16Kb
Number of random requests for random IO: 0
Read/Write ratio for combined random IO test: 4.00
Calling fsync() after each write operation.
Using synchronous I/O mode
Doing random r/w test

Operations performed:  1734215 Read, 433552 Write, 433552 Other = 2601319 Total
Read 26.462Gb  Written 6.6155Gb  Total transferred 33.077Gb  (338.71Mb/sec)
21677.60 Requests/sec executed

Test execution summary:
    total time:                          100.0003s
    total number of events:              2167767
    total time taken by event execution: 796.3734
    per-request statistics:
         min:                                  0.01ms
         avg:                                  0.37ms
         max:                                 15.18ms
         approx.  95 percentile:               1.77ms
```

Sysbench - fileio test

```
sysbench --num-threads=16 --test=fileio --file-num=29 –file-total-size=290G
    --file-fsync-all=on --file-test-mode=rndrw --max-time=100 –file-rw-ratio=4
    --file-block-size=16384 --max-requests=0 run

Number of threads: 16
Extra file open flags: 0
29 files, 10Gb each
290Gb total file size
Block size 16Kb
Number of random requests for random IO: 0
Read/Write ratio for combined random IO test: 4.00
Calling fsync() after each write operation.
Using synchronous I/O mode
Doing random r/w test

Operations performed:  2028938 Read, 507238 Write, 507238 Other = 3043414 Total
Read 30.959Gb  Written 7.7398Gb  Total transferred 38.699Gb  (396.26Mb/sec)
25360.46 Requests/sec executed

Test execution summary:
    total time:                          100.0051s
    total number of events:              2536176
    total time taken by event execution: 1594.7509
    per-request statistics:
         min:                                  0.01ms
         avg:                                  0.63ms
         max:                                 17.41ms
         approx.  95 percentile:               1.84ms
```

Sysbench - fileio test

Conclusions:

More threads => more throughput

- 1 thr : 117.14MB/sec
- 8 thrs : 338.71 MB/s (2.89x)
- 16 thrs : 396.26 MB/s (3.41x)

More threads => less predictable response time

- 1 thr : avg 0.13ms , 95% 0.16ms
- 8 thrs : avg 0.37ms , 95% 1.77ms
- 16 thrs : avg 0.63ms , 95% 1.84ms

Sysbench - OLTP

OLTP workload

- on multiple tables (Percona extension)
- --oltp-test-modes :
  - simple : very simple queries, PK lookup
  - complex : transactions
  - nontrx : non transactional queries
- --oltp-read-only
- --oltp-skip-trx

Sysbench - OLTP

Delays

- --oltp-reconnect-mode :
  - session : never disconnect
  - query : after each query
  - transaction : after each transaction
  - random : randomly
- --oltp-connect-delay
- --oltp-user-delay-min
- --oltp-user-delay-max

Sysbench - OLTP

```
$ mysql -e "DROP DATABASE IF EXISTS sbtest"
$ mysql -e "CREATE DATABASE sbtest"
$ sysbench –test=tests/db/oltp.lua
   --oltp-tables-count=4 --oltp-table-size=500000
  --mysql-table-engine=innodb --mysql-user=root
  --mysql-password=password prepare

Creating table 'sbtest1'...
Inserting 500000 records into 'sbtest1'
Creating table 'sbtest2'...
Inserting 500000 records into 'sbtest2'
Creating table 'sbtest3'...
Inserting 500000 records into 'sbtest3'
Creating table 'sbtest4'...
Inserting 500000 records into 'sbtest4'
```

Sysbench - OLTP

```
$ sysbench --test=tests/db/oltp.lua --oltp-tables-count=4 --oltp-table-size=500000
  --oltp-test-mode=simple run

Number of threads: 1
OLTP test statistics:
    queries performed:
        read:                            140000
        write:                           40000
        other:                           20000
        total:                           200000
    transactions:                        10000  (217.87 per sec.)
    deadlocks:                           0      (0.00 per sec.)
    read/write requests:                 180000 (3921.61 per sec.)
    other operations:                    20000  (435.73 per sec.)
General statistics:
    total time:                          45.8995s
    total number of events:              10000
    total time taken by event execution: 45.7667s
    response time:
         min:                                  2.62ms
         avg:                                  4.58ms
         max:                               1273.69ms
         approx.  95 percentile:               4.86ms
Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   45.7667/0.00
```

Sysbench - OLTP

```
$ sysbench --test=tests/db/oltp.lua -oltp-tables-count=16
  --oltp-table-size=500000 --oltp-test-mode=simple -num-threads=16 run

Number of threads: 16
OLTP test statistics:
    queries performed:
        read:                            140000
        write:                           40000
        other:                           20000
        total:                           200000
    transactions:                        10000  (777.78 per sec.)
    deadlocks:                           0      (0.00 per sec.)
    read/write requests:                 180000 (13999.96 per sec.)
    other operations:                    20000  (1555.55 per sec.)
General statistics:
    total time:                          12.8572s
    total number of events:              10000
    total time taken by event execution: 204.7219s
    response time:
         min:                                  3.07ms
         avg:                                 20.47ms
         max:                                178.74ms
         approx.  95 percentile:              44.42ms
Threads fairness:
    events (avg/stddev):           625.0000/10.07
    execution time (avg/stddev):   12.7951/0.01
```

Sysbench - OLTP

```
$ sysbench --test=sysbench/sysbench/sysbench/tests/db/oltp.lua --oltp-tables-count=4
  --oltp-table-size=500000 --oltp-test-mode=complex –mysql-user=root
   --mysql-password=password --num-threads=16 –oltp-point-selects=1000
    --oltp-range-size=1000 --oltp-index-updates=1000 --max-time=300 run

Number of threads: 16
OLTP test statistics:
    queries performed:
        read:                            5803120
        write:                           2015732
        other:                           7585
        total:                           7826437
    transactions:                        1805   (5.98 per sec.)
    deadlocks:                           3975   (13.18 per sec.)
    read/write requests:                 7818852 (25916.23 per sec.)
    other operations:                    7585   (25.14 per sec.)

General statistics:
    total time:                          301.6972s
    total number of events:              1805
    total time taken by event execution: 4815.5190s
    response time:
         min:                                  383.54ms
         avg:                                 2667.88ms
         max:                                18964.65ms
         approx.  95 percentile:              6059.93ms

Threads fairness:
    events (avg/stddev):           112.8125/6.89
    execution time (avg/stddev):   300.9699/0.48
```

Sysbench - OLTP

Type of queries:

- Point queries:
  SELECT c FROM sbtest WHERE id=N
- UPDATEs on index column:
  UPDATE sbtest SET k=k+1 WHERE id=N
- UPDATEs on non-index column:
  UPDATE sbtest SET c=N WHERE id=M
- Range queries:
  SELECT c FROM sbtest WHERE id BETWEEN N AND M
- Range SUM():
  SELECT SUM(k) FROM sbtest WHERE id between N and M
- Range ORDER BY:
  SELECT c FROM sbtest WHERE id between N and M ORDER BY c
- Range DISTINCT:
  SELECT DISTINCT c FROM sbtest WHERE id BETWEEN N and M ORDER BY c

BENCHMARK TOOLS: DBT2

DBT2

- is an OLTP transactional performance test
- it simulates a wholesale parts supplier where several workers access a database, update customer information and check on parts inventories
- DBT-2™ is a fair usage implementation of the TPC's TPC-C™ Benchmark specification

Source:

http://sourceforge.net/apps/mediawiki/osdldbt/index.php?title=Main_Page#dbt2

DBT2

- İt is one of the most popular benchmark tool for MySQL
- It seriously lacks of any good documentation
- Website: http://osdldbt.sourceforge.net

DBT2 : Perl modules

- Required Perl modules:
  - Statistics::Descriptive
  - Test::Parser
  - Test::Reporter
- Install Perl modules with:

```
$ sudo cpan Statistics::Descriptive
$ sudo cpan Test::Parser
$ sudo cpan Test::Reporter
```

Note: it won't return any compiling errors if these packages are missing.

DBT2 :Download and compile

Download page: http://sourceforge.net/projects/osdldbt/files/dbt2/

$ wget
http://downloads.sourceforge.net/project/osdldbt/dbt2/0.40/dbt2-0.40.tar.gz
$ tar -zxf dbt2-0.40.tar.gz
$ cd dbt2-0.40
$ ./configure –with-mysql[=/usr/local/mysql]
$ make

Note: If you don't have installed MySQL or any of its variants through package manager (rpm/deb) , you can specify a path for –with-mysql .
Ex: unpack MySQL tarball in /usr/local/mysql

DBT2 : stages

- Generate data
- Load data
- Run benchmark

DBT2 :Generate data

Data for the test is generated by datagen

Usage: datagen -w # [-c #] [-i #] [-o #] [-s #] [-n #] [-d <str>]
-w # : warehouse cardinality
-c # : customer cardinality, default 3000
-i # : item cardinality, default 100000
-o # : order cardinality, default 3000
-n # : new-order cardinality, default 900
-d <path> : output path of data files
--sapdb : format data for SAP DB
--pgsql : format data for PostgreSQL
--mysql : format data for MySQL

Ex:
$ mkdir /tmp/dbt2-w10
$ datagen -w 10 -d /tmp/dbt2-w10 –mysql

DBT2 : Load data

Load data using build_db.sh

options:

-d <database name>
-f <path to dataset files>
-g (generate data files)
-s <database socket>
-h <database host>
-u <database user>
-p <database password> *
-e <storage engine: [MYISAM|INNODB]. (default INNODB)>
-l <to use LOCAL keyword while loading dataset> *
-v <verbose>
-w <scale factor>

build_db.sh -w 3 -d dbt2 -f /tmp/dbt2-w3 -s /var/run/mysqld/mysqld.sock -h localhost -u root -p password -e INNODB -v -l

(*) See next slide

DBT2 : Load data (cont.)

Series of gotcha :

Documentation say to use scripts/mysql/mysql_load_db.sh : this script doesn't exist!

Instead, use script/mysql/build_db.sh

Doesn't load the files!

In build_db.sh replace:
> while getopts "d:h:f:gs:e:m:u:p:vw:" opt; do
with:
< while getopts "d:h:f:gs:e:m:u:p:vlw:" opt; do

The database password is not accepted via -p but via -x

DBT2 : Run the test

Run test using run_workload.sh

usage: run_workload.sh -c <number of database connections> -d
<duration of test> -w <number of warehouses>

other options:

-d <database name. (default dbt2)> *
-h <database host name. (default localhost)> *
-l <database port number>
-o <enable oprofile data collection>
-s <delay of starting of new threads in milliseconds>
-n <no thinking or keying time (default no)>
-u <database user>
-x <database password>
-z <comments for the test>

(*) See next slide

DBT2 : Run the test (cont.)

Series of gotcha :

Documentation say to use scripts/run_mysql.sh : this script doesn't exist!
Instead use scripts/run_workload.sh

Options:

-h doesn't specify the database hostname, but prints the help! Use -H instead
-d is not the database name, but the duration of the test in seconds

It fails to run unless you also run the follow:
$ export USE_PGPOOL=0

Socket /tmp/mysql.sock is hardcoded in the script. If your server run on a different socket the easiest way to fix it is to symlink /tmp/mysql.sock to your socket , ex:
$ ln -s /var/run/mysqld/mysqld.sock /tmp/mysql.sock

DBT2 : Run the test (cont.)

Example:
./run_workload.sh -c 10 -d 900 -n -w 10 -s 60 -u root -x password

```
                                        Response Time (s)
  Transaction      %    Average :    90th %        Total        Rollbacks      %
------------  -----  ---------------------  -----------  ---------------  -----
    Delivery   3.99      0.313 :     0.493        14301                0   0.00
   New Order  45.18      0.074 :     0.102       161981             1635   1.02
Order Status   3.93      0.054 :     0.077        14103                0   0.00
     Payment  42.90      0.059 :     0.083       153816                0   0.00
 Stock Level   4.00      0.053 :     0.076        14324                0   0.00
------------  -----  ---------------------  -----------  ---------------  -----

10592.50 new-order transactions per minute (NOTPM)
15.0 minute duration
0 total unknown errors
2 second(s) ramping up
```

BENCHMARK TOOLS: tpcc-mysql

tpcc-mysql

- It s an OLTP transactional performance test
- benchmark tool developed by Percona
- It generates TPC-C workload

tpcc-mysql

get the software

- Hosted on launchpad at: https://code.launchpad.net/~percona-dev/perconatools/tpcc-mysql
- Get the branch:

```
$ bzr branch lp:~percona-dev/perconatools/tpcc-mysql
```

- Compile:

```
$ cd tpcc-mysql/src
$ make
$ cd ..
```

tpcc-mysql

prepare the database

```
mysqladmin create tpcc10
mysql tpcc10 < create_table.sql
mysql tpcc10 < add_fkey_idx.sql
```

tpcc-mysql

load the data

./tpcc_load localhost tpcc10 root pass 10

Options are:

- hostname
- dbname
- dbuser
- dbpass
- number of warehouses

tpcc-mysql

start the benchmark

```
tpcc_start -h localhost -d tpcc10 -u root -p pass
   -w 10 -c 32 -r 10 -l 600
```

Options:
-h : hostname
-d : dbname
-u : dbuser
-p : password
-w : warehouses
-c : connections
-r : rampup (warmup time)
-l : benchmark duration

From README, incorrect: tpcc_start localhost tpcc1000 root "" 1000 32 10 10800

tpcc-mysql

start the benchmark

```
***************************************
*** ###easy### TPC-C Load Generator ***
***************************************
option h with value 'localhost'
option d with value 'tpcc10'
option u with value 'root'
option p with value 'password'
option w with value '10'
option c with value '32'
option r with value '10'
option l with value '600'
<Parameters>
     [server]: localhost
     [port]: 3306
     [DBname]: tpcc10
       [user]: root
       [pass]: password
  [warehouse]: 10
 [connection]: 32
     [rampup]: 10 (sec.)
    [measure]: 600 (sec.)
```

tpcc-mysql

start the benchmark

```
RAMP-UP TIME.(10 sec.)

MEASURING START.

 10, 2612(0):1.669|4.155, 2608(0):0.337|3.301, 260(0):0.180|0.242, 261(0):1.812|3.709, 260(0):4.304|6.928
 20, 2624(0):1.641|4.083, 2624(0):0.327|3.414, 263(0):0.186|3.136, 262(0):1.760|1.809, 266(0):4.230|6.371
 30, 2538(0):1.654|4.826, 2537(0):0.335|0.681, 253(0):0.183|3.247, 254(0):1.792|4.287, 250(0):4.151|4.577
 40, 2620(0):1.648|3.867, 2624(0):0.324|1.795, 263(0):0.185|2.332, 262(0):1.766|4.450, 264(0):4.078|6.530
 50, 2559(0):1.654|4.286, 2557(0):0.331|0.426, 256(0):0.189|0.198, 256(0):1.742|3.451, 253(0):5.538|6.709
 60, 2568(0):1.651|4.571, 2568(0):0.325|0.407, 256(0):0.176|0.300, 257(0):1.764|2.748, 258(0):4.231|6.058
 70, 2501(0):1.653|4.265, 2504(0):0.335|0.428, 251(0):0.174|0.415, 250(0):1.952|3.246, 250(0):5.176|14.204
 80, 2491(0):1.655|4.352, 2495(0):0.333|3.008, 249(0):0.184|3.041, 250(0):1.832|3.091, 249(0):4.979|6.495
 90, 2591(0):1.642|3.938, 2586(0):0.329|2.834, 258(0):0.177|0.191, 258(0):1.783|4.552, 262(0):4.341|7.202
100, 2384(0):1.673|4.406, 2364(0):0.336|0.636, 239(0):0.190|0.238, 241(0):1.775|3.547, 243(0):4.334|5.786
…
…
600, 2482(0):1.655|4.431, 2484(0):0.332|3.333, 249(0):0.177|0.188, 248(0):1.814|4.535, 249(0):4.570|6.611

STOPPING THREADS.............................
```

tpcc-mysql

benchmark results

```
<Raw Results>
  [0] sc:125242  lt:3  rt:0  fl:0
  [1] sc:125252  lt:0  rt:0  fl:0
  [2] sc:12522  lt:0  rt:0  fl:0
  [3] sc:12525  lt:0  rt:0  fl:0
  [4] sc:12526  lt:0  rt:0  fl:0
 in 600 sec.

<Raw Results2(sum ver.)>
  [0] sc:125251  lt:3  rt:0  fl:0
  [1] sc:125254  lt:0  rt:0  fl:0
  [2] sc:12524  lt:0  rt:0  fl:0
  [3] sc:12525  lt:0  rt:0  fl:0
  [4] sc:12527  lt:0  rt:0  fl:0
```

tpcc-mysql

benchmark results

```
<Constraint Check> (all must be [OK])
 [transaction percentage]
        Payment: 43.48% (>=43.0%) [OK]
   Order-Status: 4.35% (>= 4.0%) [OK]
       Delivery: 4.35% (>= 4.0%) [OK]
    Stock-Level: 4.35% (>= 4.0%) [OK]
 [response time (at least 90% passed)]
      New-Order: 100.00%  [OK]
        Payment: 100.00%  [OK]
   Order-Status: 100.00%  [OK]
       Delivery: 100.00%  [OK]
    Stock-Level: 100.00%  [OK]

<TpmC>
                12524.500 TpmC
```

MONITORING TOOLS: GATHER OS STATISTICS

OS Statistics

- Monitor what OS is doing
- Monitor what the HW is doing
  - CPU
  - Memory
  - Swap
  - Network
  - IO
- Monitor running processes

OS Statistics

Tools are OS-specific. Common on Linux :

- vmstat
- iostat
- ps
- top / atop / htop
- sar
- strace
- vnstat
- ifstat

OS Statistics - vmstat

Example:

```
shell> vmstat 10 8
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 5  2      0 1798008  92568 9738680    0    0   565  1850 9910 10207 10  1 88  1
 2  0      0 1797864  92580 9742456    0    0   390  1398 9891 10160 10  1 89  1
 1  0      0 1794728  92580 9746048    0    0   554  1890 9730 10020 10  1 88  1
 4  0      0 1789768  92580 9750536    0    0   685  2029 9961 10388 10  1 87  1
 1  0      0 1787252  92584 9753788    0    0   387  1741 9773 10140 10  1 88  1
 6  0      0 1780868  92584 9757940    0    0   410  1972 9552 9994  9  1 89  1
 9  0      0 1778524  92588 9761180    0    0   509  1790 10065 10540 11  1 87  1
 1  0      0 1774904  92592 9764996    0    0   685  1837 10096 10608 11  1 87  1
```

OS Statistics - vmstat

- procs (processes)
  - r : waiting for CPU
  - b : blocked on uninterruptible sleep (waiting for IO)
- memory (1KB blocks)
  - swpd : swapped memory
  - free : free memory
  - buff : used as buffer
  - cache : used for cache
- swap (1KB blocks)
  - si : swapped in from disk
  - so : swapped out to disk

OS Statistics – vmstat

- io (1KB blocks)
  - bi : blocks received from block devices
  - bo : blocks sent to block devices
- system
  - in : interrupts
  - cs : context switches
- cpu
  - us : in user space
  - sy : in kernel code
  - id : idle
  - wa : waiting for IO
  - (if virtualization is enabled) st : stolen from a virtual machine

OS Statistics - iostat

Example:

```
shell> iostat -k -d -x 10 3 /dev/sd?
Device:  rrqm/s  wrqm/s    r/s    w/s   rkB/s    wkB/s avgrq-sz avgqu-sz  await  svctm  %util
sda        0.50    6.31  91.33  78.02 1092.28  2162.86    38.44     0.04   0.21   1.01  17.12
sdb        0.00    0.09   0.25   2.98   26.99   175.11   124.99     0.02   4.89   0.74   0.24

Device:  rrqm/s  wrqm/s    r/s    w/s   rkB/s    wkB/s avgrq-sz avgqu-sz  await  svctm  %util
sda        0.00    1.50  58.60  97.00  358.40  2520.40    37.00     0.42   2.70   0.93  14.40
sdb        0.00    0.00   0.00   2.40    0.00   240.75   200.62     0.01   2.67   0.33   0.08

Device:  rrqm/s  wrqm/s    r/s    w/s   rkB/s    wkB/s avgrq-sz avgqu-sz  await  svctm  %util
sda        0.00    1.50  54.00  61.10  351.60  1556.70    33.16     0.25   2.13   1.46  16.80
sdb        0.00    0.00   0.00   2.20    0.00   185.05   168.23     0.00   0.91   0.18   0.04
```

OS Statistics - iostat

rrqm/s : read requests merged per second

wrqm/s : write requests merged per second

r/s : read requests per second

w/s : write requests per second

rKB/s : KBs read per second

wKB/s : KBs written per second

avgrq-sz : average size (in sectors) of requests

avgqu-sz : average queue length of requests

await : average time (milliseconds) for requests to be served (queue time + service time)

svctm : average service time (milliseconds) for requests to be served

%util :percentage of CPU time during the requests : device saturation

OS Statistics - iostat

Example:

```
shell> iostat -k -d -x 10 3 /dev/sd?
Device: rrqm/s  wrqm/s   r/s   w/s   rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
sdd       0.00    0.30  0.00 541.90   0.00 246928.30   911.34   111.64  205.75   1.85 100.00

Device: rrqm/s  wrqm/s   r/s   w/s   rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
sdd       0.00    1.10  0.00 543.10   0.00 247461.60   911.29   112.11  205.99   1.84 100.02

Device: rrqm/s  wrqm/s   r/s   w/s   rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
sdd       0.00    0.80  0.00 541.40   0.00 246080.00   909.05   111.97  206.90   1.85 100.01
```

OS Statistics - top

Example:

```
top - 05:05:25 up 264 days,  7:28, 13 users,load average: 4.03, 4.28, 4.13
Tasks: 333 total,   1 running, 328 sleeping,   0 stopped,   4 zombie
Cpu(s): 19.4%us,  0.5%sy,  0.0%ni, 79.1%id,  0.3%wa,  0.3%hi,  0.4%si,  0.0%st
Mem:  49508976k total, 48101200k used,  1407776k free,    94028k buffers
Swap: 23438824k total,        0k used, 23438824k free, 10128656k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
28384 mysql     20   0 35.5g  34g 7620 S  388 73.7  49961:09 mysqld
    1 root      20   0 10312  336  208 S    0  0.0   1:38.12 init
    2 root      15  -5     0    0    0 S    0  0.0   0:00.24 kthreadd
    3 root      RT  -5     0    0    0 S    0  0.0  15:56.46 migration/0
    4 root      15  -5     0    0    0 S    0  0.0   0:21.74 ksoftirqd/0
    5 root      RT  -5     0    0    0 S    0  0.0   1:59.25 watchdog/0
```

OS Statistics - ifstat

Example:

```
shell> ifstat -i eth1 5 8
       eth1
 KB/s in  KB/s out
 4091.66   6565.15
 3828.57   5750.80
 3772.95   5725.60
 3624.42   5597.97
 3838.65   5936.50
 3813.17   5773.01
 3424.67   5264.98
 3981.78   6020.65
```

QUERY PROFILING:
Slow Query Log

Query profiling

- Analysis of slow query log
- PROFILE
- EXPLAIN

Slow query log

- By default logs on file
- Logs SQL statements that:
  - Run for more than long_query_time
  - Do not use indexes (log_queries_not_using_indexes)
- Other options:
  - log_slow_admin_statements
  - log_slow_slave_statements
  - (Percona) log_slow_verbosity (query_plan / innodb / profiling)
  - (Percona) log_slow_filter (full_scan / tmp_table / filesort_on_disk / etc)

http://dev.mysql.com/doc/refman/5.5/en/slow-query-log.html
http://www.percona.com/doc/percona-server/5.5/diagnostics/slow_extended_55.html

Slow query log

```
# Time: 120325  2:29:54
# User@Host: web[web] @ web1 [192.168.1.138]
# Thread_id: 1217328  Schema: dbname_prod  Last_errno: 0  Killed: 0
# Query_time: 1.248839  Lock_time: 0.001044  Rows_sent: 98  Rows_examined:
    146  Rows_affected: 0  Rows_read: 1
# Bytes_sent: 215048  Tmp_tables: 0  Tmp_disk_tables: 0  Tmp_table_sizes: 0
# InnoDB_trx_id: 71BE9460
# QC_Hit: No  Full_scan: No  Full_join: No  Tmp_table: No  Tmp_table_on_disk: No
# Filesort: No  Filesort_on_disk: No  Merge_passes: 0
#   InnoDB_IO_r_ops: 9  InnoDB_IO_r_bytes: 147456  InnoDB_IO_r_wait: 1.240737
#   InnoDB_rec_lock_wait: 0.000000  InnoDB_queue_wait: 0.000000
#   InnoDB_pages_distinct: 43
SET timestamp=1332667794;
SELECT ....
```


Profiling from slow query log

- SET GLOBAL long_query_time=0;
- Force all the connections to close;
- Move / delete the current slow query log;
- FLUSH LOGS;
- SET GLOBAL slow_query_log_file = 'slow.log';
- Leave it running for hours
- Keep an eye on disk space
- SET GLOBAL long_query_time=2;
- Analyze the slow query log with pt-query-digest

pt-query-digest

- Sophisticated log analysis tool
- Analyze queries execution logs
- Generates reports
- Review queries
- Replay logs

http://www.percona.com/doc/percona-toolkit/2.0/pt-query-digest.html

pt-query-digest

- SQL profiling of production system
- Analysis of benchmark results

REPLAY QUERIES:
pt-log-player

pt-log-player

- It is not a benchmark tool
- It is a load simulator
- It splits slow query log into files
- It replays the queries in sessions

pt-log-player : split

```
mkdir ./sessions

pt-log-player --split-random --session-files 32 \
    --filter '$event->{arg} =~ m/^select/i'  \
    --base-dir ./sessions slow.log
```

pt-log-player : play

```
mkdir ./results

pt-log-player --play ./sessions –threads=16 \
  --base-dir ./results h=dbserver 2> /dev/null

Found 32 session files.
Running processes...
All processes are running; waiting for them to finish...
Process 15 finished with exit status 0.
….
All processes have finished.
```

Analysis with pt-query-digest

```
pt-query-digest ./results/* --limit 100% > result.txt
```

Analysis with pt-query-digest

```
# 12281.9s user time, 43.8s system time, 28.95M rss, 70.60M vsz
# Current date: Mon Mar 26 03:59:38 2012
# Hostname: localhost
# Files: ./results/session-results-5572.txt, ...
# Overall: 72.87M total, 617 unique, 0 QPS, 0x concurrency _________________
# Attribute          total     min     max     avg     95%  stddev  median
# ============     ======= ======= ======= ======= ======= ======= =======
# Exec time        195839s    51us    322s     3ms     2ms   112ms   185us
# Query size         8.09G      17 107.37k  119.28  246.02  314.29   97.36

# Profile
# Rank Query ID           Response time    Calls   R/Call   Apdx V/M   Ite
# ==== ================== ================ ======= ======== ==== ===== ===
#    1 0x556DDA838C37A12A 61313.0978 31.3%  335402   0.1828 1.00  0.01 SELECT tablename1
   tablename2
#    2 0x97EA420234E19A02 47876.5685 24.4%  335402   0.1427 1.00  0.01 SELECT tablename1
#    3 0x4AA832C76224C657  9850.8987  5.0%  117116   0.0841 1.00 54.17 SELECT newspapers
   newspaper_subscribers articles
#    4 0xB594EEFDB69D3394  9550.8752  4.9%  343188   0.0278 1.00  0.00 SELECT tablename3
#    5 0x9989A980DB6EAA1F  4578.3801  2.3%  148360   0.0309 1.00  0.52 SELECT
   article_comments
```

Analysis with pt-query-digest

```
# Query 1: 0 QPS, 0x concurrency, ID 0x556DDA838C37A12A at byte 672169 ___
# This item is included in the report because it matches --limit.
# Scores: Apdex = 1.00 [1.0], V/M = 0.01
# Query_time sparkline: |    _^_ |
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          0  335402
# Exec time     31  61313s    81ms      1s   183ms   230ms    33ms   171ms
# Query size     1 115.89M     361     364  362.30  363.48    1.50  346.17
# String:
# Databases    dbname
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms  #
# 100ms  ###############################################################
#    1s  #
#  10s+
# Tables
#    SHOW TABLE STATUS FROM `dbname` LIKE 'tablename1'\G
#    SHOW CREATE TABLE `dbname`.`tablename1`\G
#    SHOW TABLE STATUS FROM `dbname` LIKE 'tablename2'\G
#    SHOW CREATE TABLE `dbname`.`tablename2`\G
# EXPLAIN /*!50100 PARTITIONS*/
select eo.*, ca.value cavalue from tablename1 eo left join tablename2 ca on eo.caid=ca.id and eo.from_cid = ca.cid
    where eo.expired = 0 and ca.cid = 62 and eo.hidden_until < unix_timestamp() and eo.value > 0 and eo.created >
    unix_timestamp(now() - interval 5 day) order by eo.aer limit 0, 10\G
```


Analysis with pt-query-digest

- Analysis of pt-log-player result files reports:
  - Execution time
  - Count
  - Distribution
- Analysis is done client side

pt-log-player

+

slow query log

+

pt-query-digest

pt-log-player and slow query log

- Enable slow query log on server
- Run pt-log-player
- Process the slow query log with pt-query-digest
- Provides more useful information

pt-log-player and slow query log

- SET GLOBAL long_query_time=0;
- pt-log-player --play ./sessions –threads=16 --base-dir ./results h=dbserver 2> /dev/null
- SET GLOBAL long_query_time=2;
- pt-query-digest slow.log > slow.digest.txt

pt-query-digest

```
# Files: slow.log
# Overall: 72.97M total, 691 unique, 5.85k QPS, 15.47x concurrency ________
# Time range: 2012-03-25 17:28:34 to 20:56:20
# Attribute          total     min     max     avg     95%  stddev  median
# ============     ======= ======= ======= ======= ======= ======= =======
# Exec time        192899s     1us    322s      3ms      2ms   114ms   144us
# Lock time          3095s       0      4s    42us    60us     1ms    36us
# Rows sent        352.08M       0  27.01k    5.06   14.52   32.87    0.99
# Rows examine     107.01G       0  29.64M   1.54k  223.14  16.89k    0.99
# Rows affecte           0       0       0       0       0       0       0
# Rows read        352.08M       0  27.01k    5.06   14.52   32.87    0.99
# Bytes sent        44.14G       0  16.58M  649.44   1.20k   4.04k  420.77
# Tmp tables         1.26M       0       4    0.02       0    0.13       0
# Tmp disk tbl     225.35k       0       1    0.00       0    0.06       0
# Tmp tbl size       1.09T       0   7.99M  15.98k       0 172.19k       0
# Query size         8.11G      16 107.37k  119.28  246.02  314.34   97.36
# InnoDB:
# IO r bytes       615.94M       0 291.12M    8.85       0  36.39k       0
# IO r ops          38.50k       0  18.20k    0.00       0    2.26       0
# IO r wait              0       0       0       0       0       0       0
```

pt-query-digest

```
# Profile
# Rank Query ID           Response time    Calls   R/Call   Apdx V/M   Ite
# ==== ================== ================ ======= ======== ==== ===== ===
#    1 0x556DDA838C37A12A 61280.8789 31.8%  335349   0.1827 1.00  0.01 SELECT tablename1
   tablename2
#    2 0x97EA420234E19A02 47850.7893 24.8%  335355   0.1427 1.00  0.01 SELECT tablename1
#    3 0x4AA832C76224C657  9843.0476  5.1%  117092   0.0841 1.00 54.22 SELECT newspapers
   newspaper_subscribers articles
#    4 0xB594EEFDB69D3394  9529.5770  4.9%  343133   0.0278 1.00  0.00 SELECT tablename3
#    5 0x9989A980DB6EAA1F  4569.4989  2.4%  148326   0.0308 1.00  0.52 SELECT
   article_comments
```


pt-query-digest query #1

```
# Query 1: 26.90 QPS, 4.92x concurrency, ID 0x556DDA838C37A12A at byte
   38055152
# This item is included in the report because it matches --limit.
# Scores: Apdex = 1.00 [1.0], V/M = 0.01
# Query_time sparkline: |    _^_ |
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          0  335349
# Exec time     31  61281s    81ms      1s    183ms    230ms     33ms    171ms
# Lock time      0     25s    50us    94ms     73us     84us    157us     69us
# Rows sent      0   3.06M       0      10    9.56    9.83    2.02    9.83
# Rows examine  45  48.78G 151.60k 161.98k 152.53k 158.07k   1.97k 150.54k
# Rows affecte   0       0       0       0       0       0       0       0
# Rows read      0   3.06M       0      10    9.56    9.83    2.02    9.83
# Bytes sent     1 523.84M     934   1.73k   1.60k   1.61k  150.88   1.61k
# Tmp tables     0       0       0       0       0       0       0       0
# Tmp disk tbl   0       0       0       0       0       0       0       0
# Tmp tbl size   0       0       0       0       0       0       0       0
# Query size     1 115.87M     361     364  362.30  363.48    1.50  346.17
```


pt-query-digest query #1

```
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms  #
# 100ms  ##################################################
#    1s  #
#  10s+
# Tables
#    SHOW TABLE STATUS FROM `dbname` LIKE 'tablename1'\G
#    SHOW CREATE TABLE `dbname`.`tablename1`\G
#    SHOW TABLE STATUS FROM `dbname` LIKE 'tablename2'\G
#    SHOW CREATE TABLE `dbname`.`tablename2`\G
# EXPLAIN /*!50100 PARTITIONS*/
select eo.*, ca.value cavalue from tablename1 eo left join tablename2 ca on
   eo.caid=ca.id and eo.from_cid = ca.cid where eo.expired = 0 and ca.cid =
   62 and eo.hidden_until < unix_timestamp() and eo.value > 0 and eo.created
   > unix_timestamp(now() - interval 5 day) order by eo.aer limit 0, 10\G
```


EXPLAIN query #1

```
mysql> EXPLAIN select eo.*, ca.value cavalue from tablename1 eo left join
   tablename2 ca on eo.caid=ca.id and eo.from_cid = ca.cid where eo.expired =
   0 and ca.cid = 62 and eo.hidden_until < unix_timestamp() and eo.value > 0
   and eo.created > unix_timestamp(now() - interval 5 day) order by eo.aer
   limit 0, 10;
+----+-------------+-------+--------+---------------+---------+---------
   +---------------+--------+----------------------------+
| id | select_type | table | type   | possible_keys | key     | key_len |
   ref             | rows   | Extra                      |
+----+-------------+-------+--------+---------------+---------+---------
   +---------------+--------+----------------------------+
|  1 | SIMPLE      | eo    | ALL    | caid          | NULL    | NULL    |
   NULL            | 161517 | Using where; Using filesort |
|  1 | SIMPLE      | ca    | eq_ref | PRIMARY,cid   | PRIMARY | 4       |
   dbname.eo.caid |      1 | Using where                |
+----+-------------+-------+--------+---------------+---------+---------
   +----------   ---+--------+----------------------------+
2 rows in set (0.00 sec)
```

EXPLAIN query #1

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: eo
         type: ALL
possible_keys: caid
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 161517
        Extra: Using where; Using filesort
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: ca
         type: eq_ref
possible_keys: PRIMARY,cid
          key: PRIMARY
      key_len: 4
          ref: dbname.eo.caid
         rows: 1
        Extra: Using where
```

EXPLAIN:
a closer look to its output

EXPLAIN

- Explains the execution plan
- Works only for SELECT statements
- Returns one row for each table in the query
  - also subqueries and UNION

EXPLAIN output : id

- Indicates which SELECT is referred
- Distinguish nested SELECT

EXPLAIN output : select_type

Defines the type of SELECT:

- SIMPLE : no subqueries or unions
- PRIMARY : outer SELECT
- UNION : not first SELECT in a union
- UNION RESULT : result of a union
- DERIVED : subquery in FROM clause
- DEPENDANT UNION : union is dependent on the outer query
- DEPENDANT SUBQUERY : subquery is dependent on the outer query
- UNCACHEABLE SUBQUERY : subquery is evaluated for each matching row of the outer query

EXPLAIN output : table

- shows the name of the table
- order of the tables is the order of the JOINs
- Indexes affect JOINs order

EXPLAIN output : type

Defines the type of join:

- system : table has only 1 row
- const : only 1 row matching
- eq_ref : only 1 row matching for this referenced table (ex. Unique key)
- ref : all rows with matching index values are read
- ref_or_null : as “ref”, but also for NULL values
- index_merge : more than one index is used, and result merged
- unique_subquery : the subquery is replaced a single value from unique index
- index_subquery : similar to unique_subquery, but without unique index
- range : rows retrieved using a index range scan
- index : similar to ALL , but only the index tree is scanned
- ALL : full table scan

EXPLAIN output : possible_keys

- Lists of keys the optimizer can choose to perform the join
- If NULL : no relevant indexes

EXPLAIN output : key

- The key that the optimizer has chosen
- If the key is not listed in possible_keys:
  - No suitable indexes for rows lookup and the index is a covering index

EXPLAIN output : key_len

- Length of a single value in the index
- In case of multi-column index returns the length used

EXPLAIN output

- ref: Column or constraint used to compare the index in “ref”
- rows: estimated number of rows that MySQL will examine
- Extra : additional information

EXPLAIN output : Extra

- Distinct : looking only for DISTINCT values
- Impossible WHERE : WHERE clause is always false
- Using filesort : extra pass for sorting rows
- Using index : column information retrieved from the index and not from the row
- Using temporary : temporary tables are created
- Using where : WHERE clause used to filter rows
- Etc etc

… back to pt-query-digest

EXPLAIN query #1

```
EXPLAIN select eo.*, ca.value cavalue from tablename1 eo left join tablename2 ca on
   eo.caid=ca.id and eo.from_cid = ca.cid where eo.expired = 0 and ca.cid = 62 and
   eo.hidden_until < unix_timestamp() and eo.value > 0 and eo.created >
   unix_timestamp(now() - interval 5 day) order by eo.aer limit 0, 10;
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: eo
         type: ALL
possible_keys: caid
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 161517
        Extra: Using where; Using filesort
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: ca
         type: eq_ref
possible_keys: PRIMARY,cid
          key: PRIMARY
      key_len: 4
          ref: dbname.eo.caid
         rows: 1
        Extra: Using where
```

CREATE TABLE

```
CREATE TABLE `tablename1` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `cid` int(10) unsigned NOT NULL,
  `caid` int(10) unsigned NOT NULL,
  `value` decimal(10,2) unsigned DEFAULT '1.00',
  `from_cid` tinyint(3) unsigned NOT NULL DEFAULT '1',
  `aer` decimal(10,3) unsigned DEFAULT '1.000',
  `created` int(10) unsigned NOT NULL,
  `expired` int(11) NOT NULL,
  `hidden_until` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `cid` (`cid`),
  KEY `aid` (`caid`),
) ENGINE=InnoDB AUTO_INCREMENT=188988 DEFAULT CHARSET=latin1

CREATE TABLE `tablename2` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `citid` int(10) unsigned NOT NULL,
  `cid` tinyint(3) unsigned NOT NULL DEFAULT '1',
  `value` decimal(14,6) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `citid` (`citid`,`cid`),
  KEY `cid` (`cid`)
) ENGINE=InnoDB AUTO_INCREMENT=9171691 DEFAULT CHARSET=latin1
```

ADD INDEX

```
ALTER TABLE tablename1 ADD INDEX (expired, from_cid);
```

```
CREATE TABLE `tablename1` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `cid` int(10) unsigned NOT NULL,
  `caid` int(10) unsigned NOT NULL,
  `value` decimal(10,2) unsigned DEFAULT '1.00',
  `from_cid` tinyint(3) unsigned NOT NULL DEFAULT '1',
  `aer` decimal(10,3) unsigned DEFAULT '1.000',
  `created` int(10) unsigned NOT NULL,
  `expired` int(11) NOT NULL,
  `hidden_until` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `cid` (`cid`),
  KEY `aid` (`caid`),
  KEY `expired` (`expired`,`from_cid`)
) ENGINE=InnoDB AUTO_INCREMENT=188988 DEFAULT CHARSET=latin1
```

New EXPLAIN query #1

```
EXPLAIN select eo.*, ca.value cavalue from tablename1 eo left join tablename2 ca on
   eo.caid=ca.id and eo.from_cid = ca.cid where eo.expired = 0 and ca.cid = 62 and
   eo.hidden_until < unix_timestamp() and eo.value > 0 and eo.created >
   unix_timestamp(now() - interval 5 day) order by eo.aer limit 0, 10;
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: eo
         type: ref
possible_keys: caid,expired
          key: expired
      key_len: 5
          ref: const,const
         rows: 660
        Extra: Using where; Using filesort
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: ca
         type: eq_ref
possible_keys: PRIMARY,cid
          key: PRIMARY
      key_len: 4
          ref: dbname.eo.caid
         rows: 1
        Extra: Using where
```

pt-query-digest query #2

```
# Query 2: 26.90 QPS, 3.84x concurrency, ID 0x97EA420234E19A02 at byte
   319276140
# This item is included in the report because it matches --limit.
# Scores: Apdex = 1.00 [1.0], V/M = 0.01
# Query_time sparkline: |    _^  |
# Time range: 2012-03-25 17:28:34 to 20:56:19
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          0  335355
# Exec time     24  47851s    50ms   828ms    143ms    180ms     29ms    141ms
# Lock time      0     20s    42us     8ms     59us     69us     19us     57us
# Rows sent      0 327.50k       1       1       1       1        0        1
# Rows examine  45  48.48G 151.58k 151.58k 151.58k 151.58k        0 151.58k
# Rows affecte   0       0       0       0       0       0        0        0
# Rows read      0 327.50k       1       1       1       1        0        1
# Bytes sent     0  20.53M      64      65   64.20   62.76     0.00    62.76
# Tmp tables     0       0       0       0       0       0        0        0
# Tmp disk tbl   0       0       0       0       0       0        0        0
# Tmp tbl size   0       0       0       0       0       0        0        0
# Query size     0  65.82M     205     206  205.80  202.40        0  202.40
```


EXPLAIN query #2

Before ADD INDEX:

```
EXPLAIN select ceil(count(1)/10) maxPages from tablename1 where from_cid =
   62 and value >= 1 and expired = 0 and hidden_until <= unix_timestamp() and
   created > unix_timestamp(now() - interval 5 day)
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: eo
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 161517
        Extra: Using where
```

EXPLAIN query #2

After ADD INDEX:

```
EXPLAIN select ceil(count(1)/10) maxPages from tablename1 where from_cid =
   62 and value >= 1 and expired = 0 and hidden_until <= unix_timestamp() and
   created > unix_timestamp(now() - interval 5 day)
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: eo
         type: ref
possible_keys: expire
          key: expire
      key_len: 5
          ref: const,const
         rows: 660
        Extra: Using where
```


TRENDING

Cacti

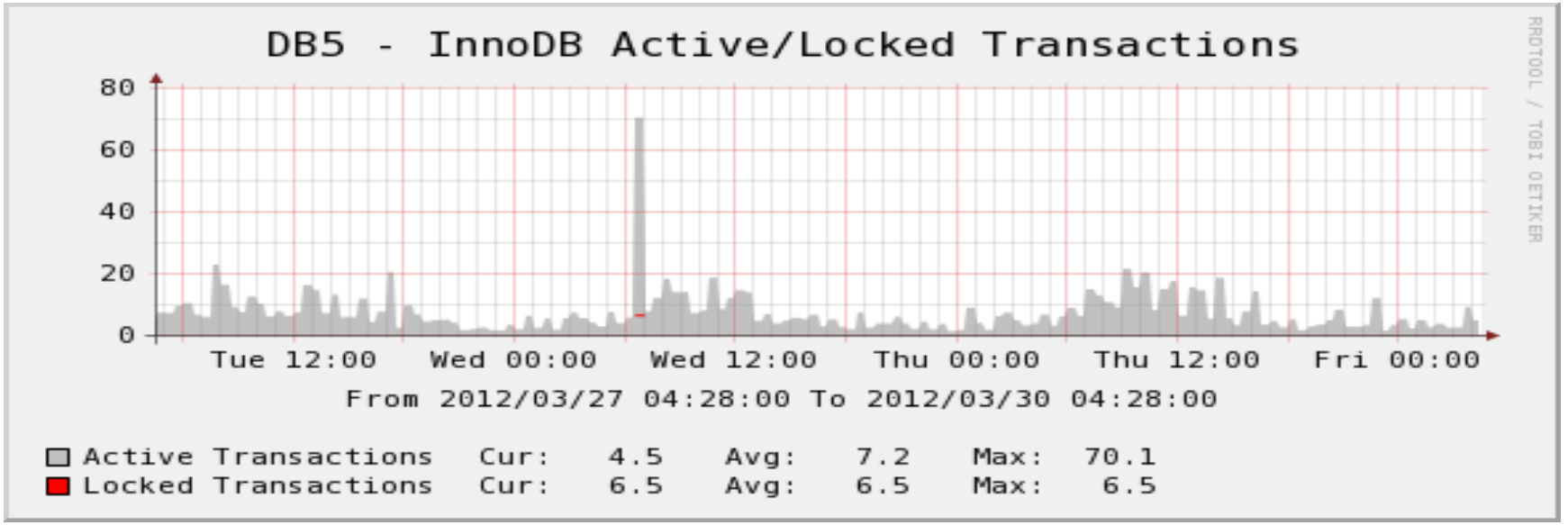
- Data logging
- RRDTool (round-robin database)
- Trending
- Track history
- Useful to spot change in workload

Percona MySQL Monitoring Template for Cacti

- Plugin for Cacti
- Formely known as “MySQL Cacti Template”
- Avaliable at :

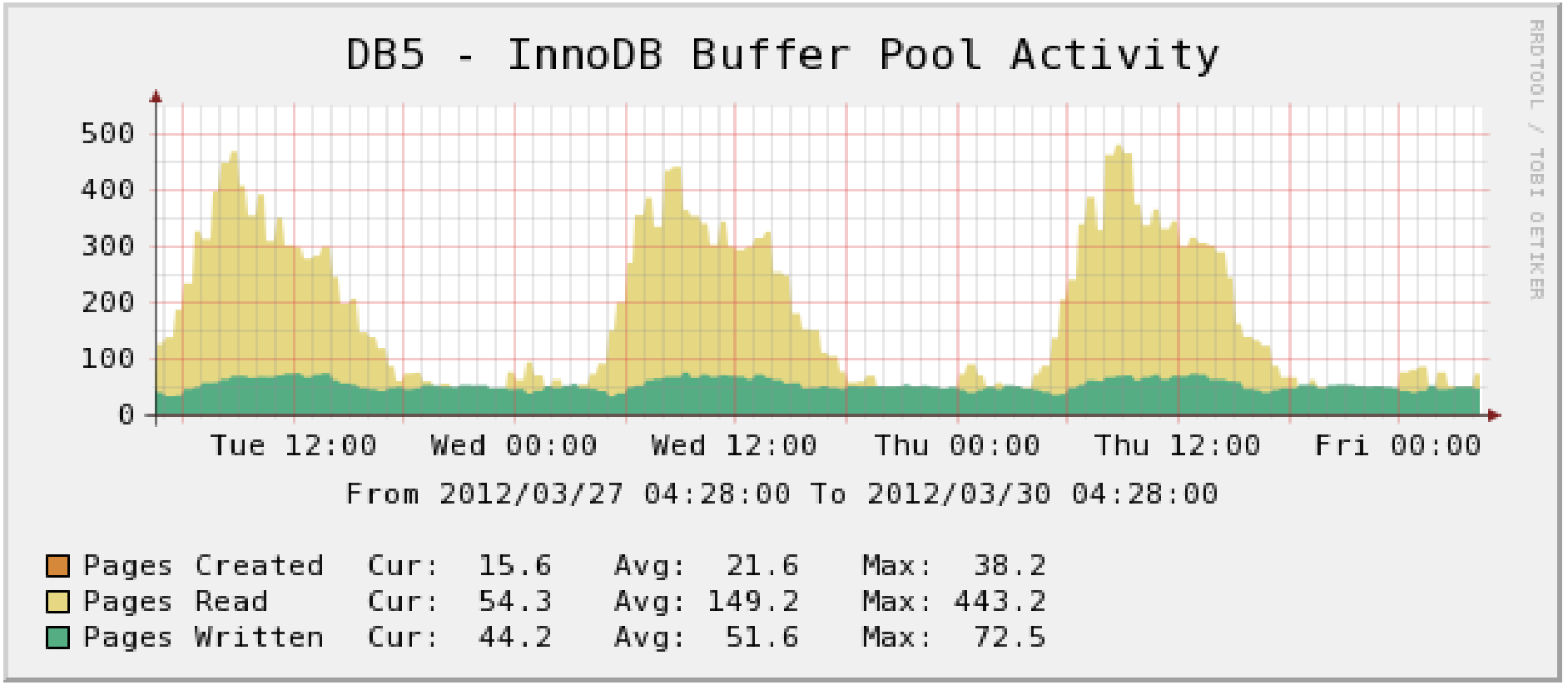
http://www.percona.com/software/percona-monitoring-plugins/

InnoDB Active/Lock Transactions



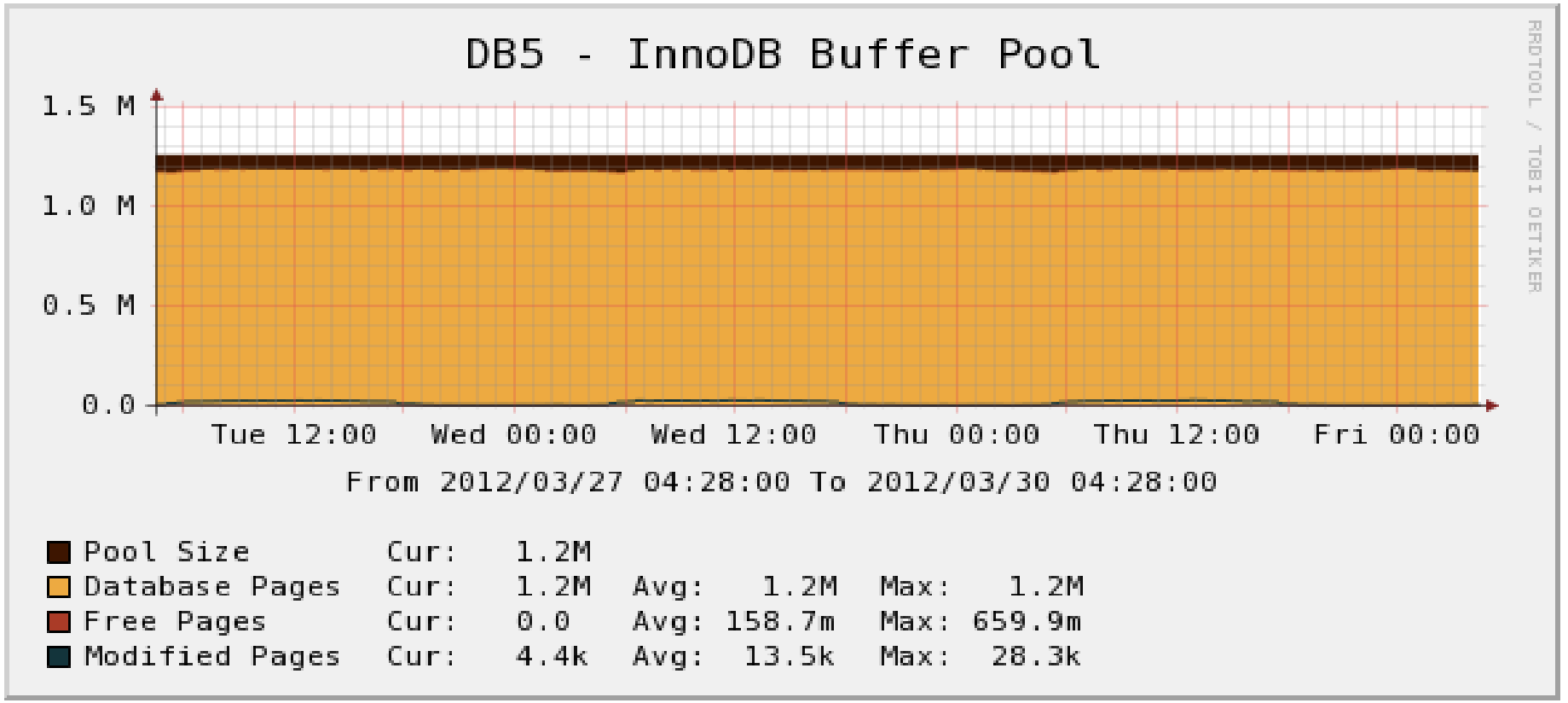
- Active : between BEGIN and COMMIT
- Locked : in LOCK WAIT status

InnoDB Buffer Pool Activity



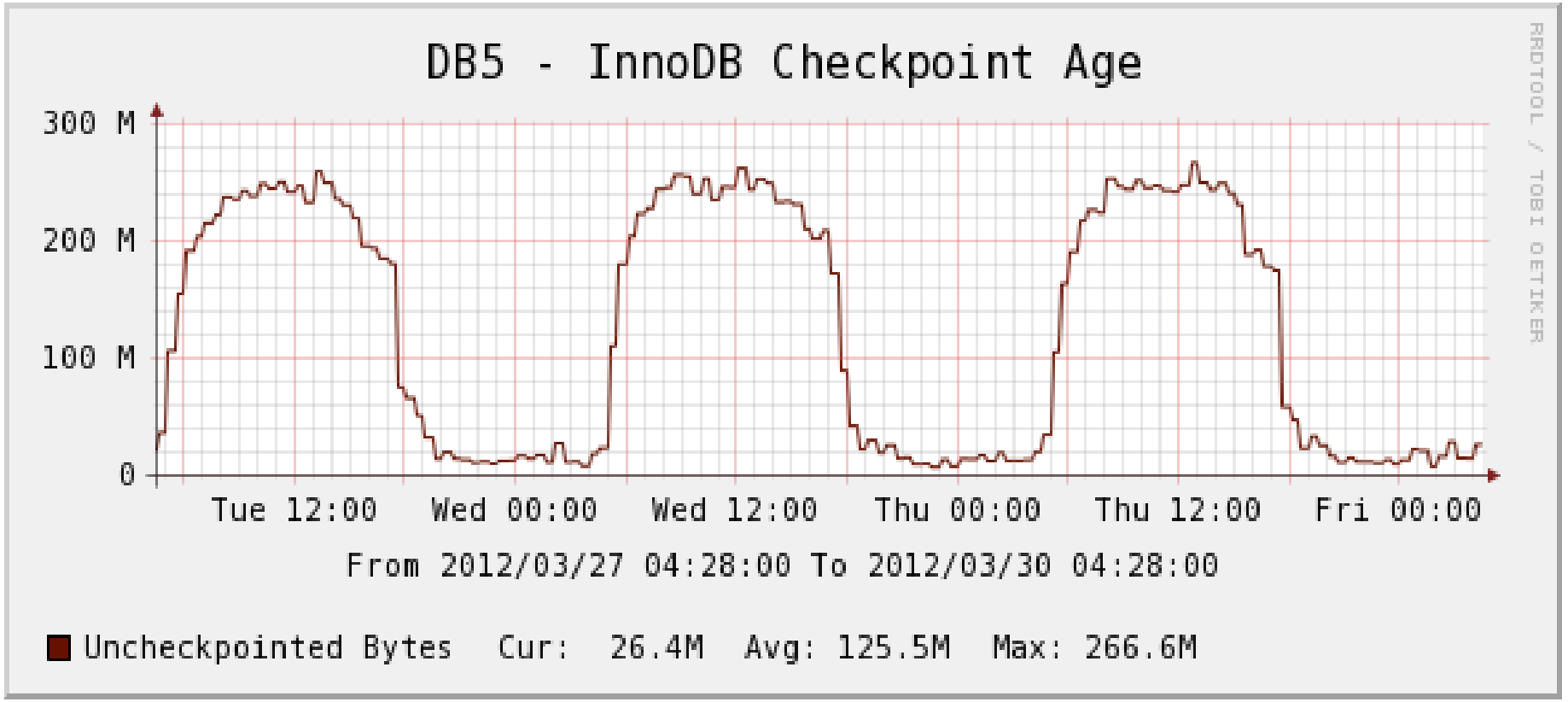
- Pages activity

InnoDB Buffer Pool



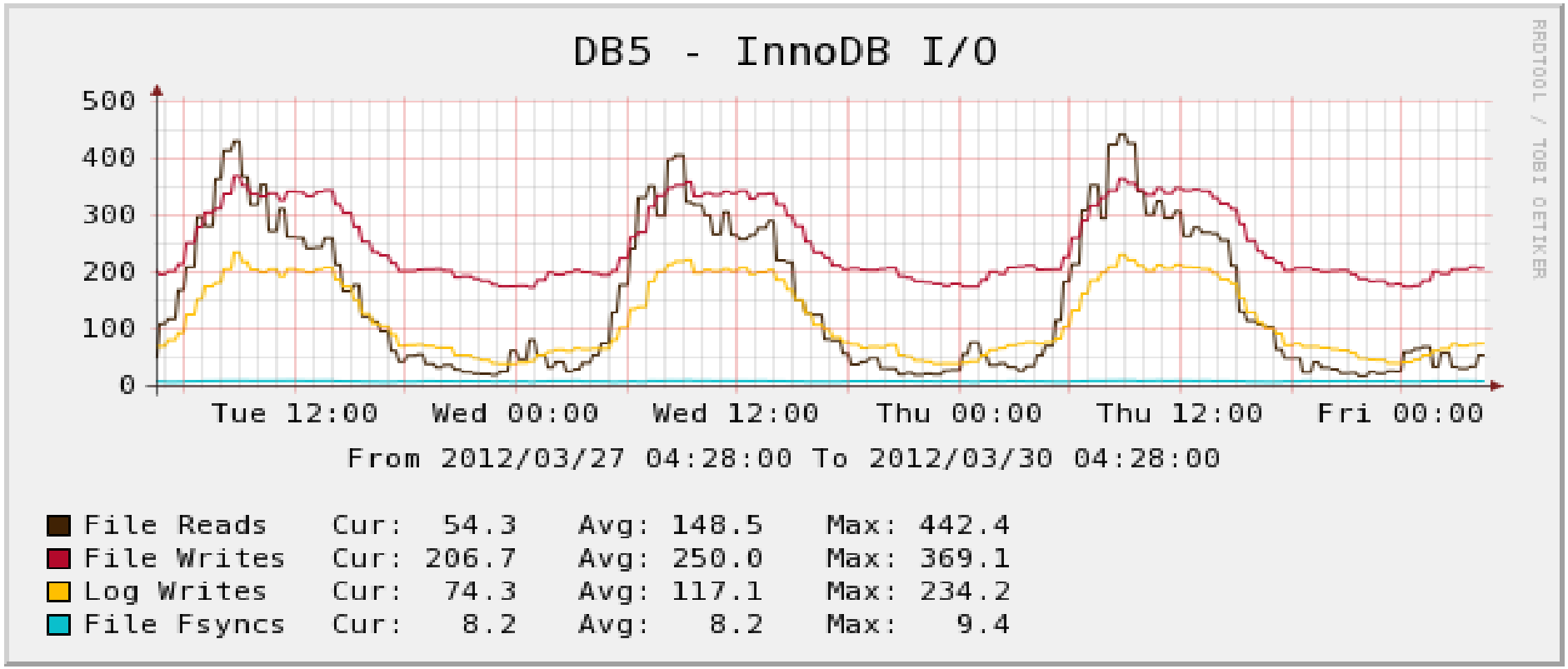
- Pages status

InnoDB Checkpoint Age



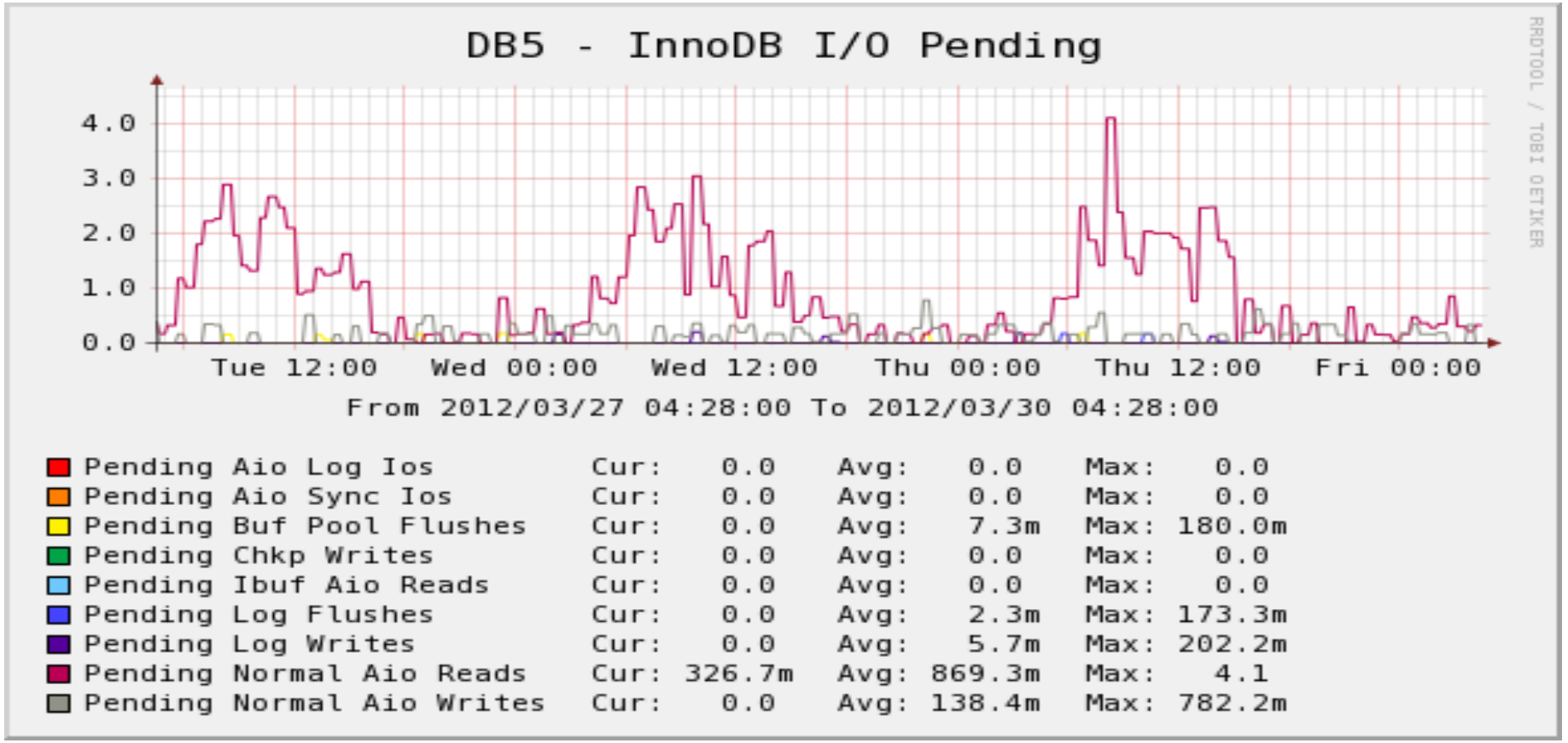
- Uncheckpointed bytes in the REDO Log

InnoDB I/O



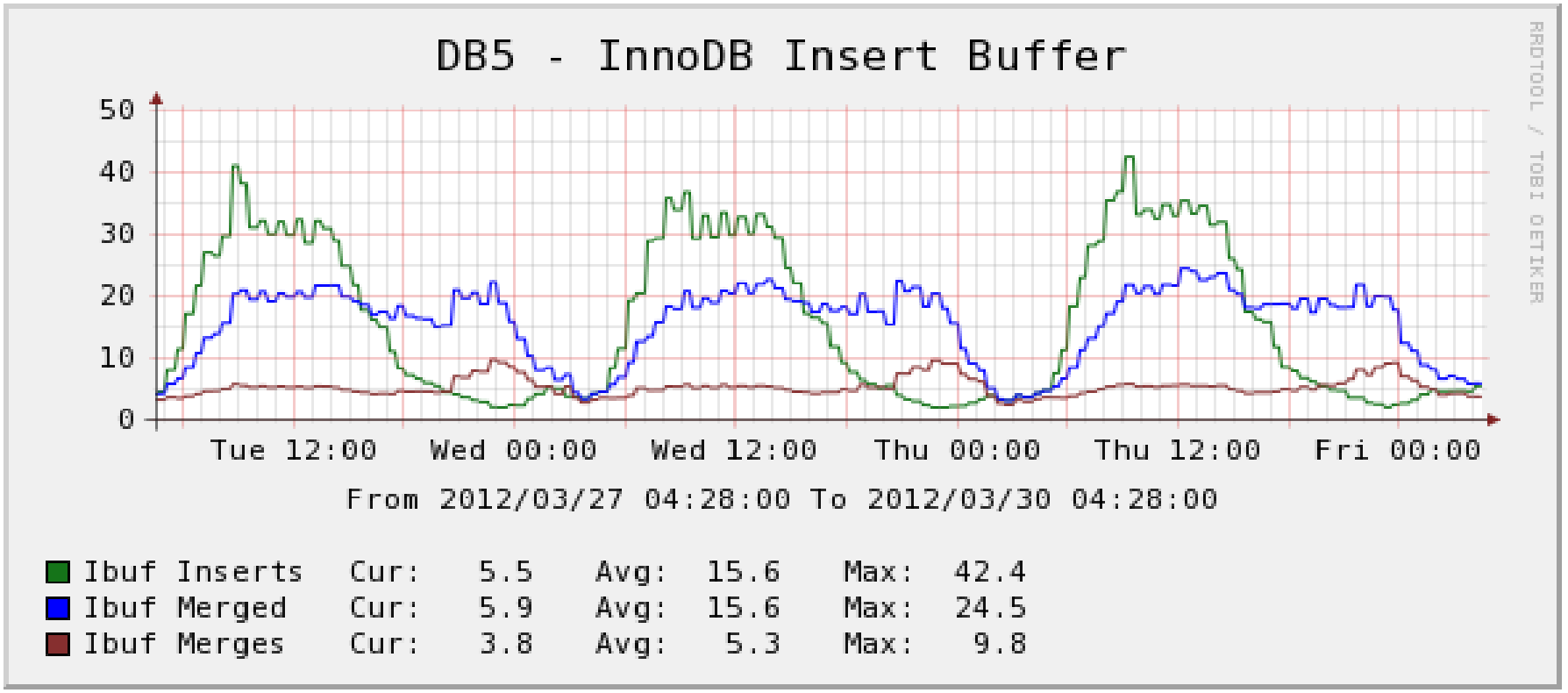
- I/O activity
  - Tablespaces reads and writes
  - Log activity

InnoDB I/O Pending



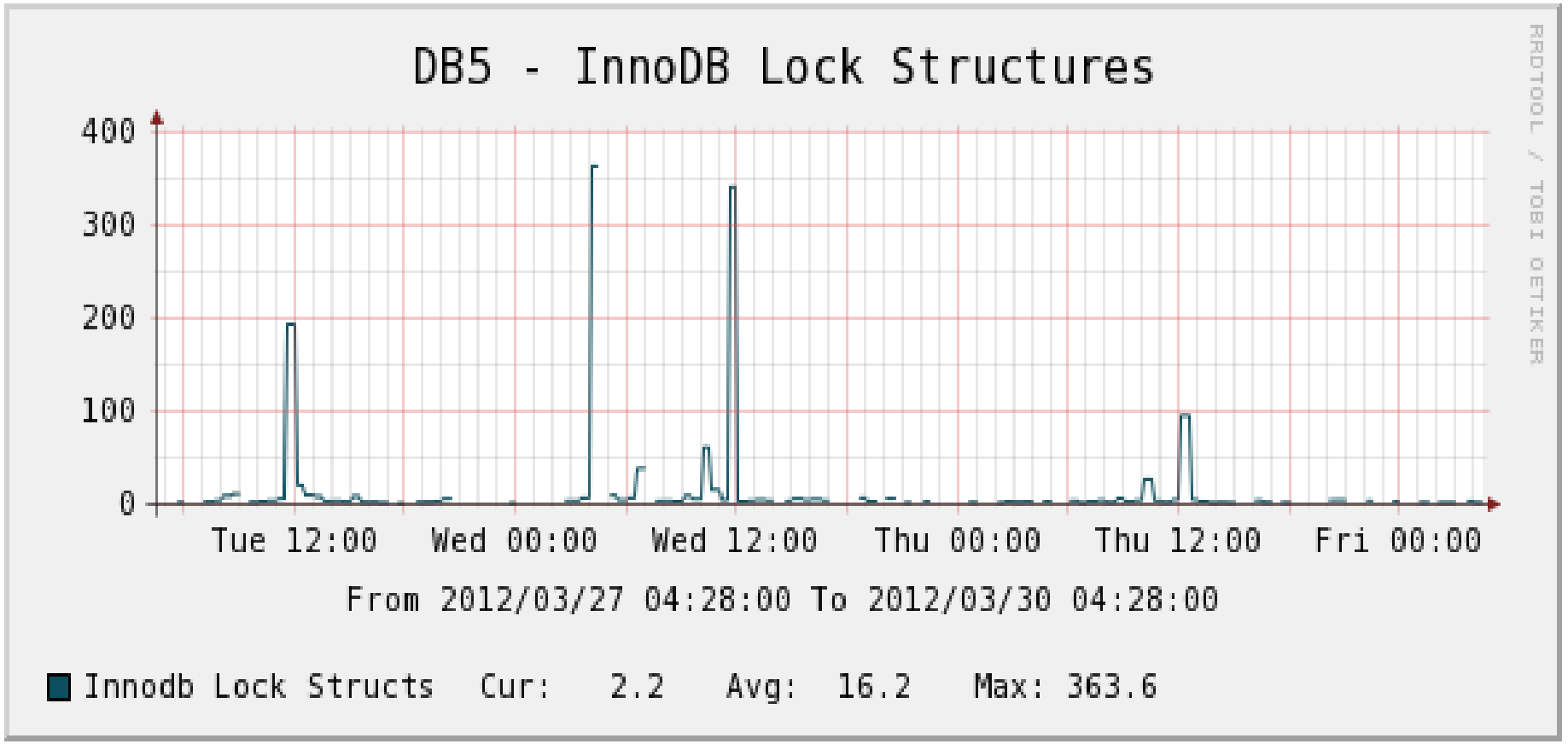
- Pending IO in the various InnoDB files

InnoDB Insert Buffer



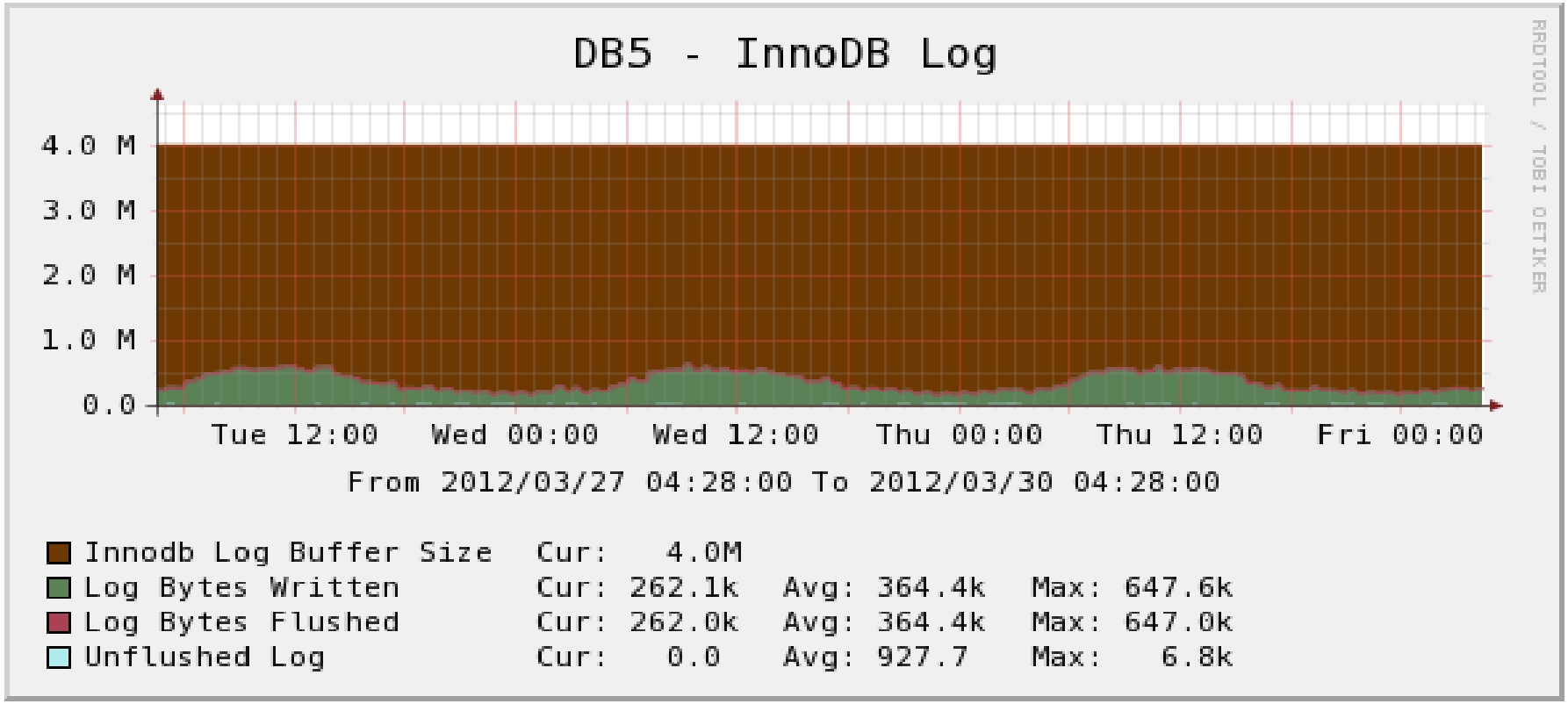
- Cache for secondary index records update

InnoDB Lock Structures



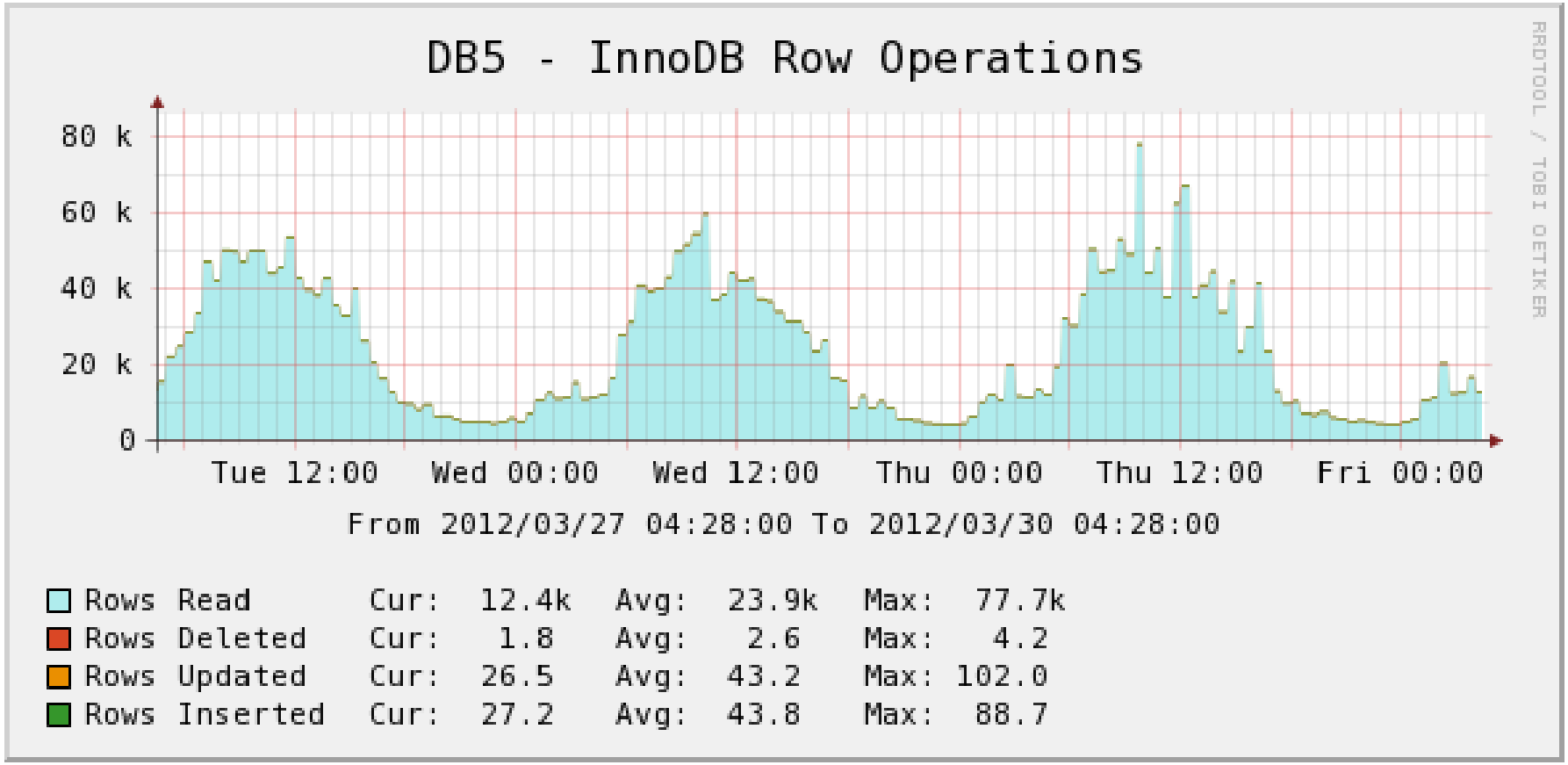
- Number of lock structures in transactions

InnoDB Log



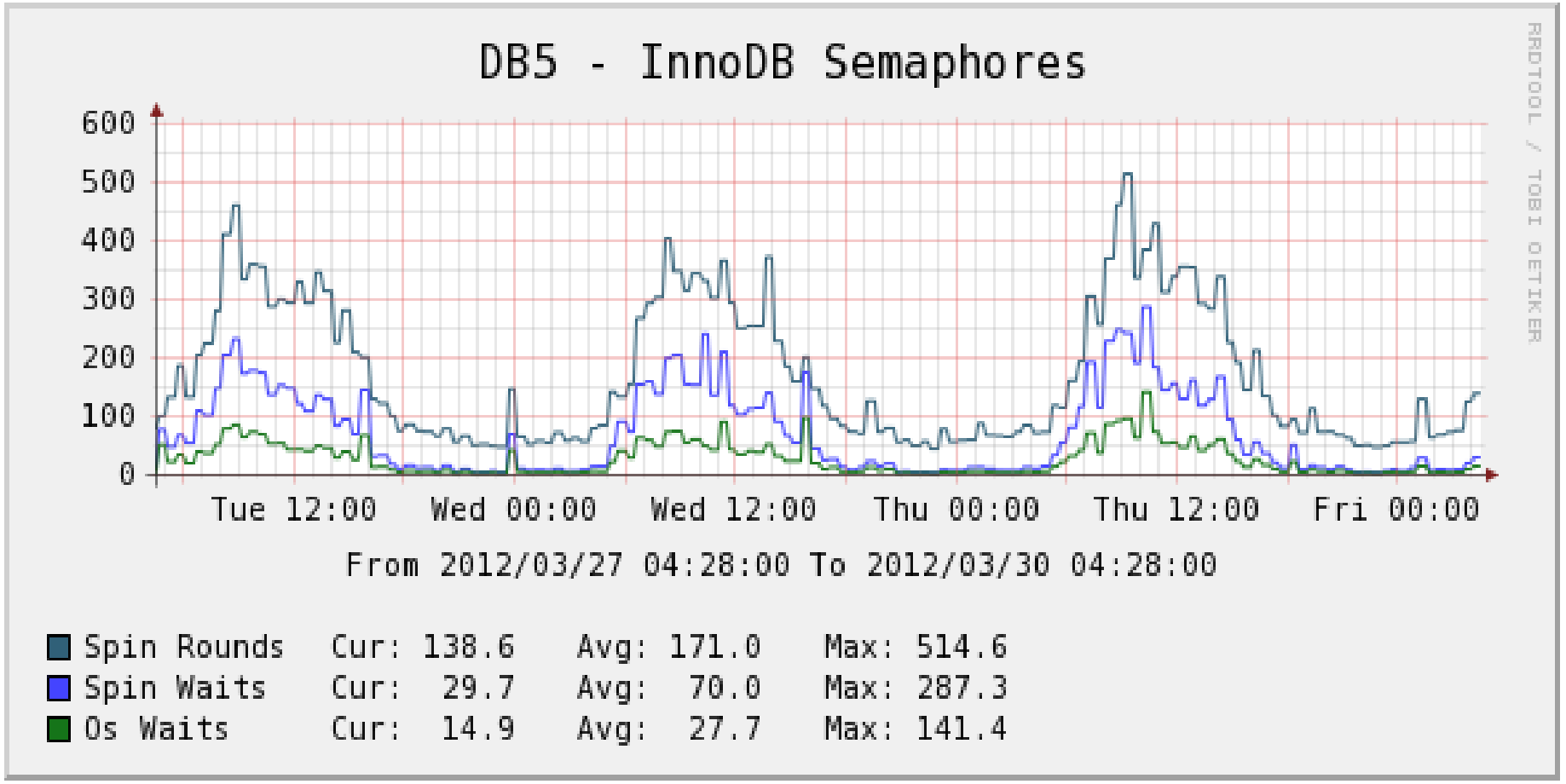
- Activity in InnoDB Log Buffer

InnoDB Row Operations



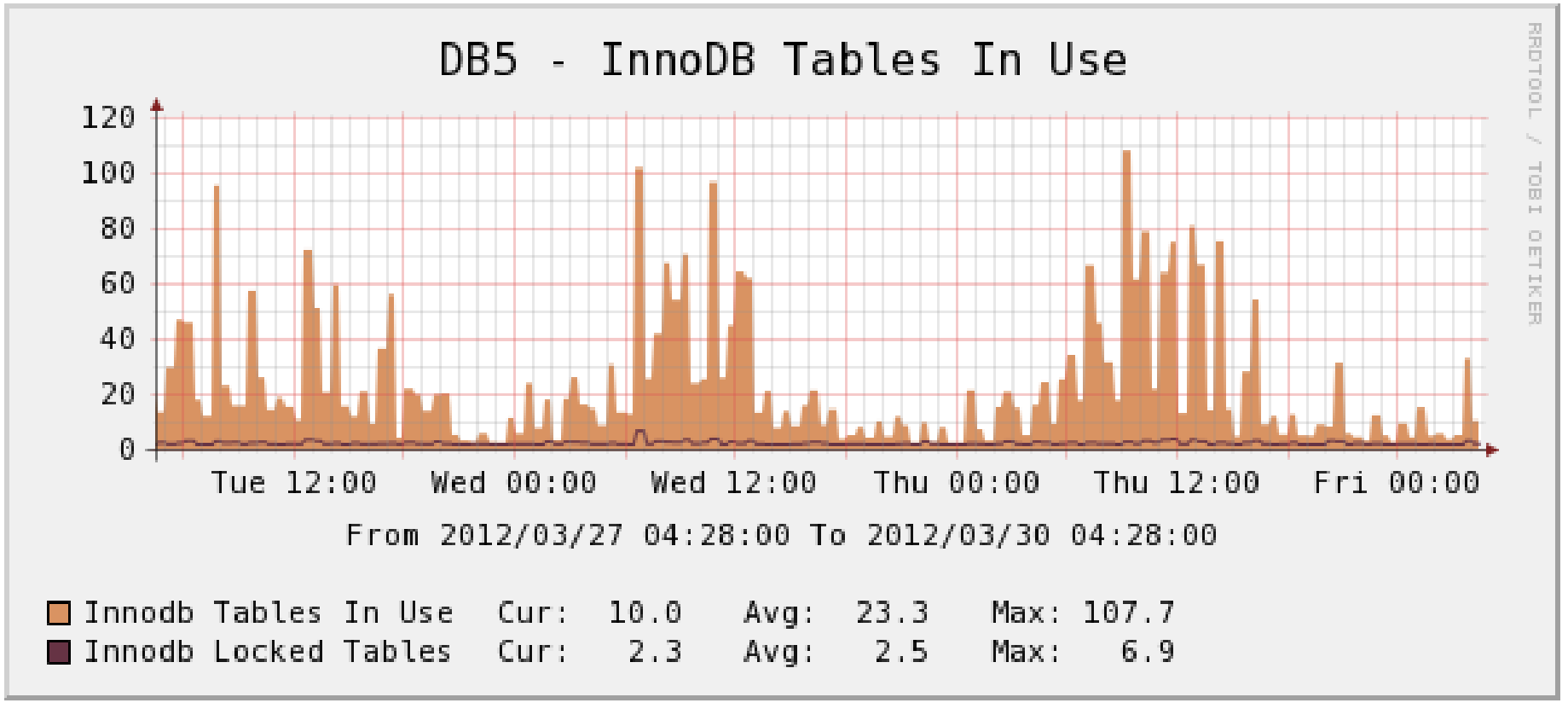
- Read, deleted, updated, inserted

InnoDB Semaphores



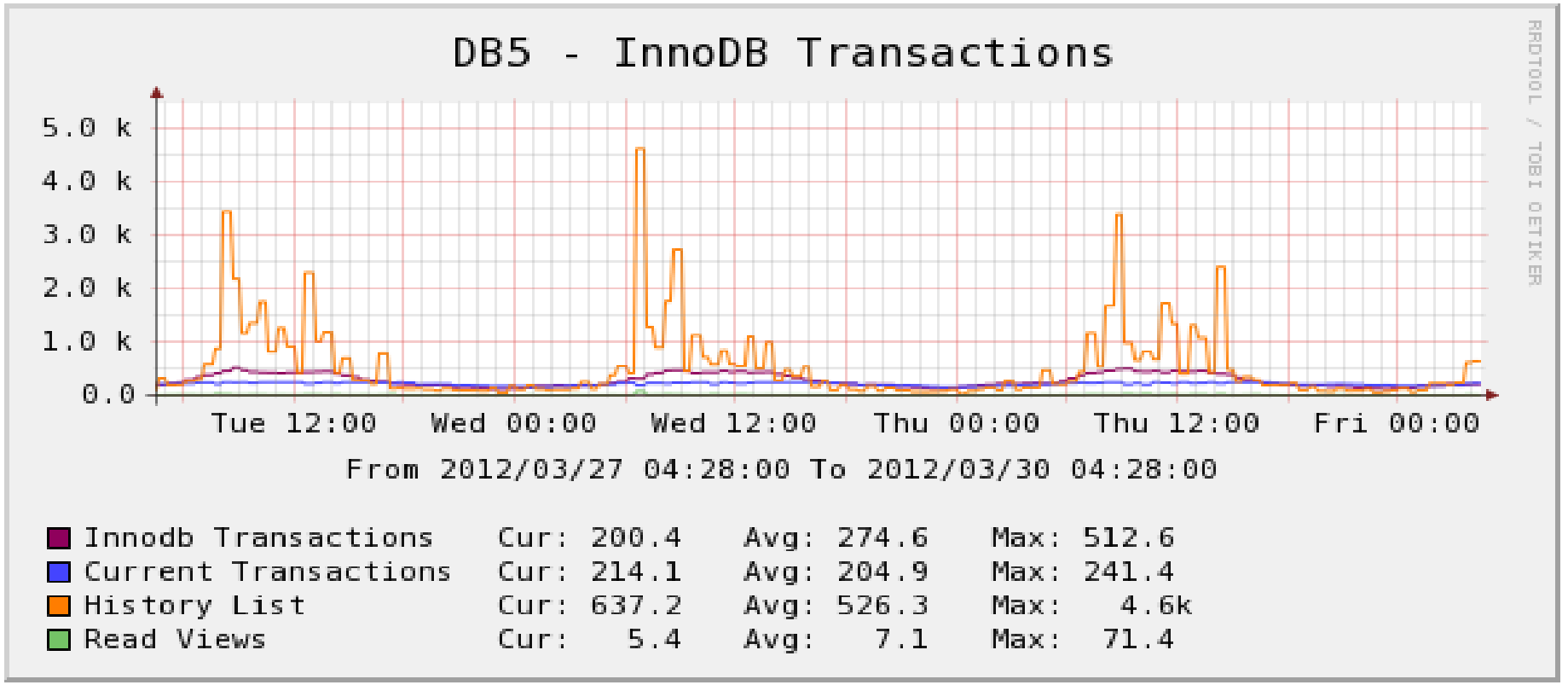
- Symptom of poor scalibility on concurrency

InnoDB Tables In Use



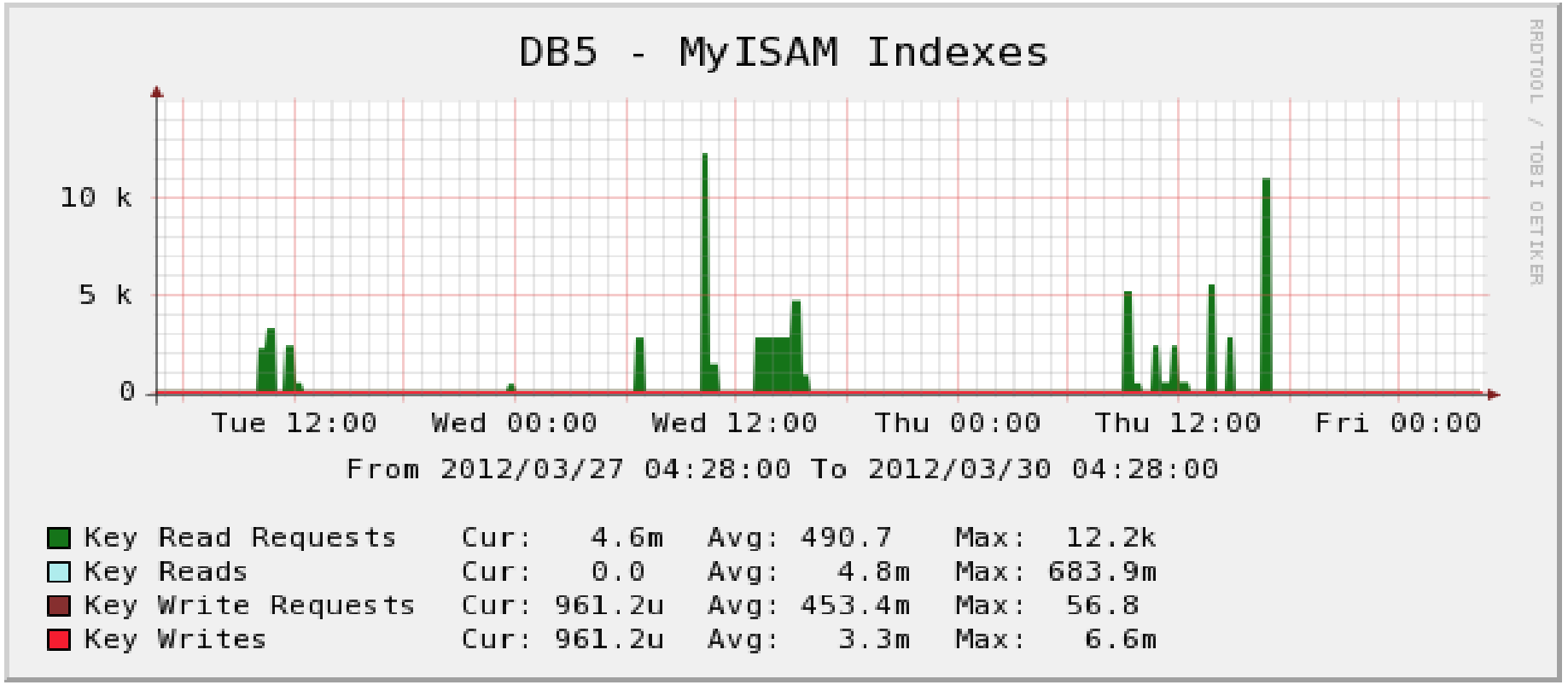
- Number of tables in use, and locked

InnoDB Transactions



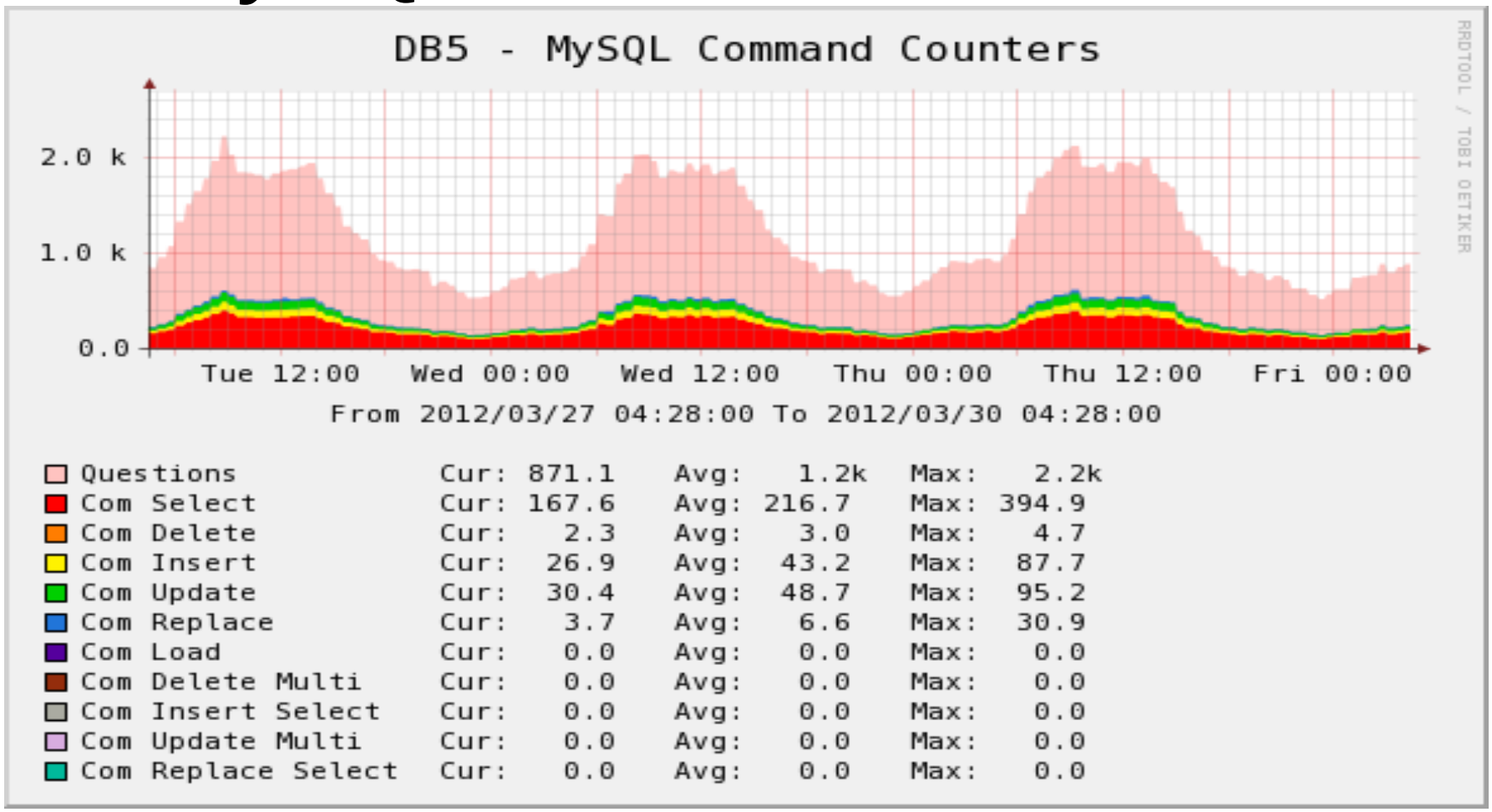
- Number of transactions
- History list (unpurged transaction)

MyISAM Indexes



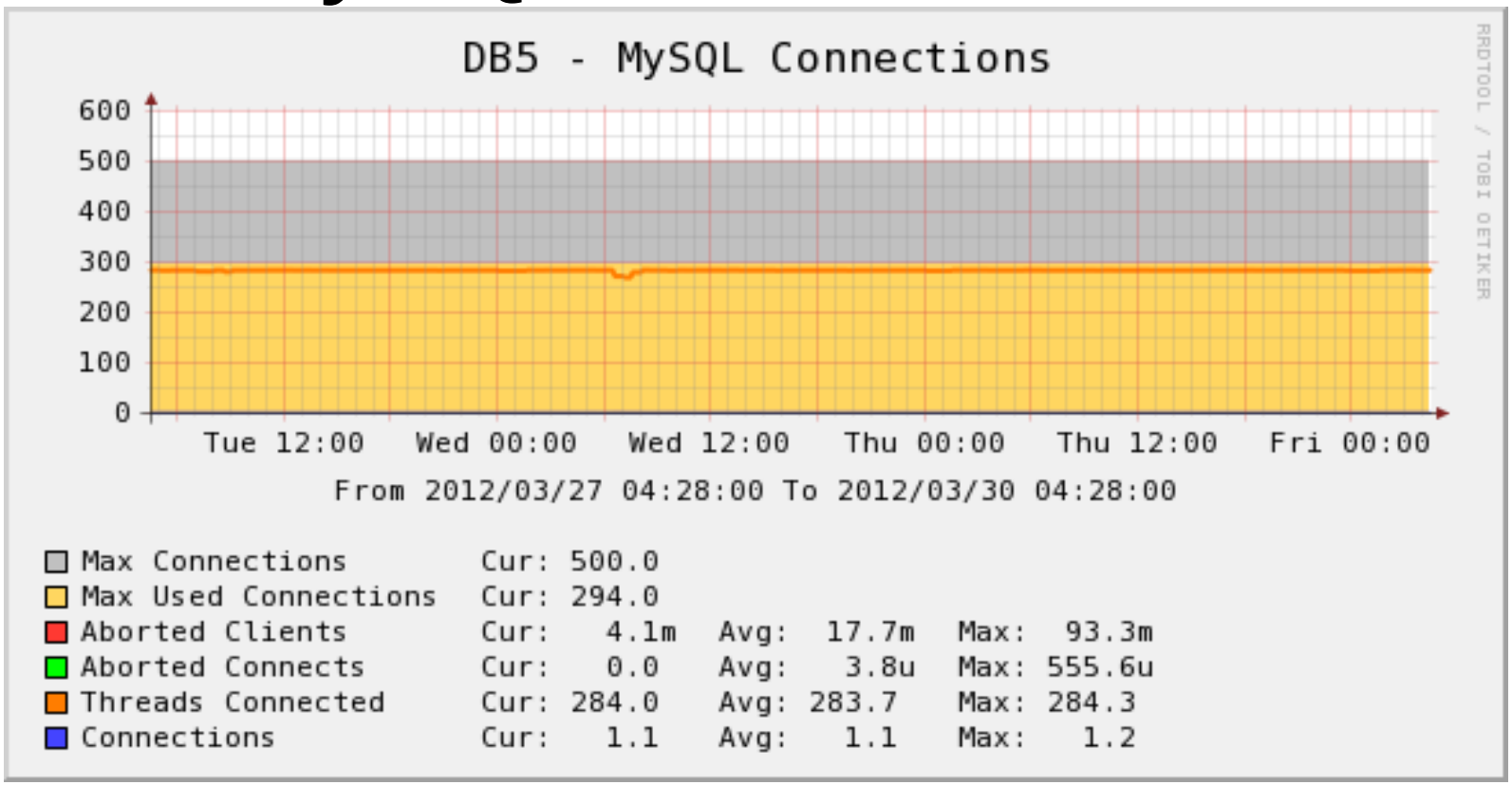
- Reads/Writes of MyISAM indexes blocks
- Key buffer activity

MySQL Command Counters



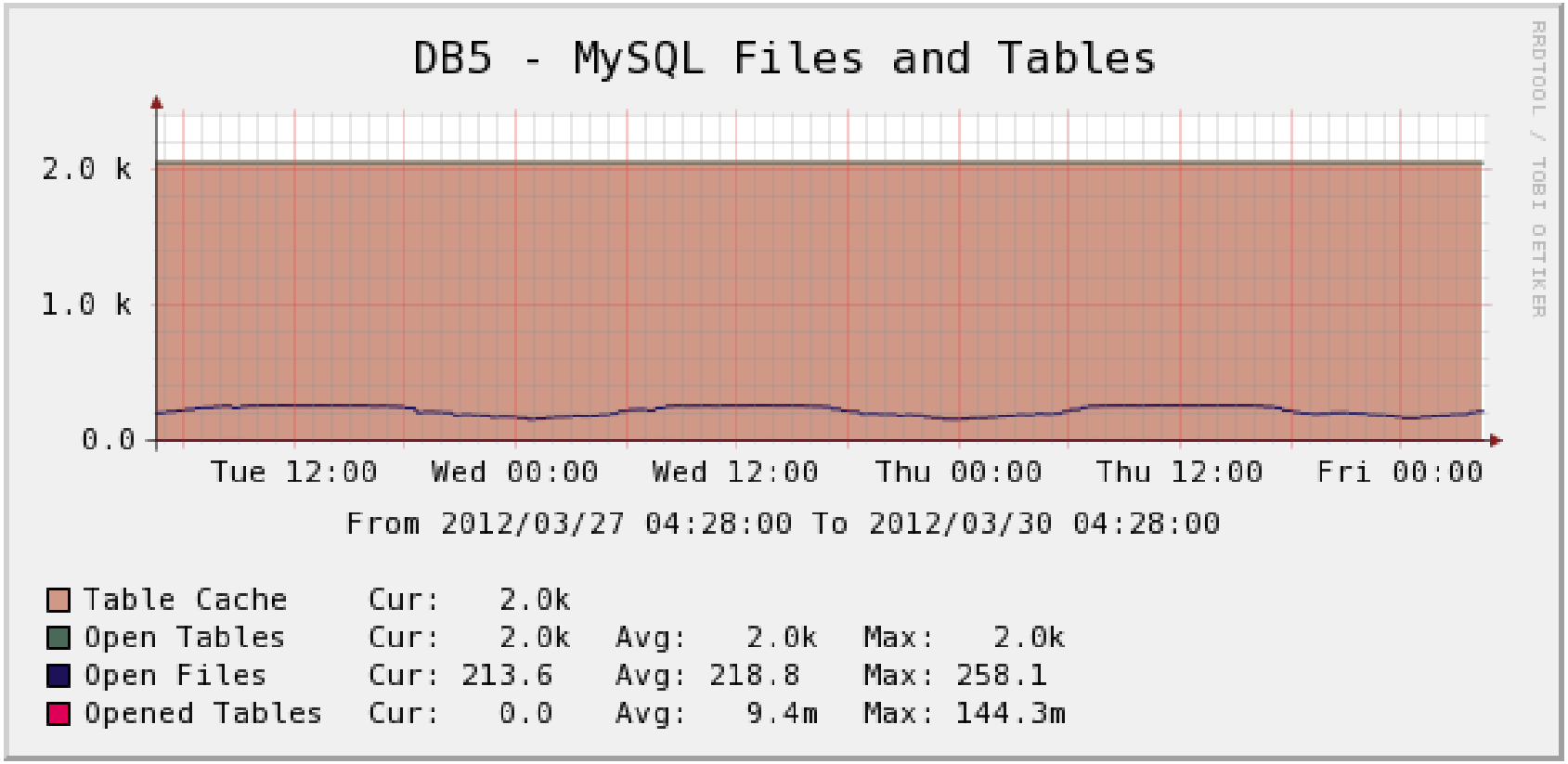
- Type of commands running

MySQL Connections



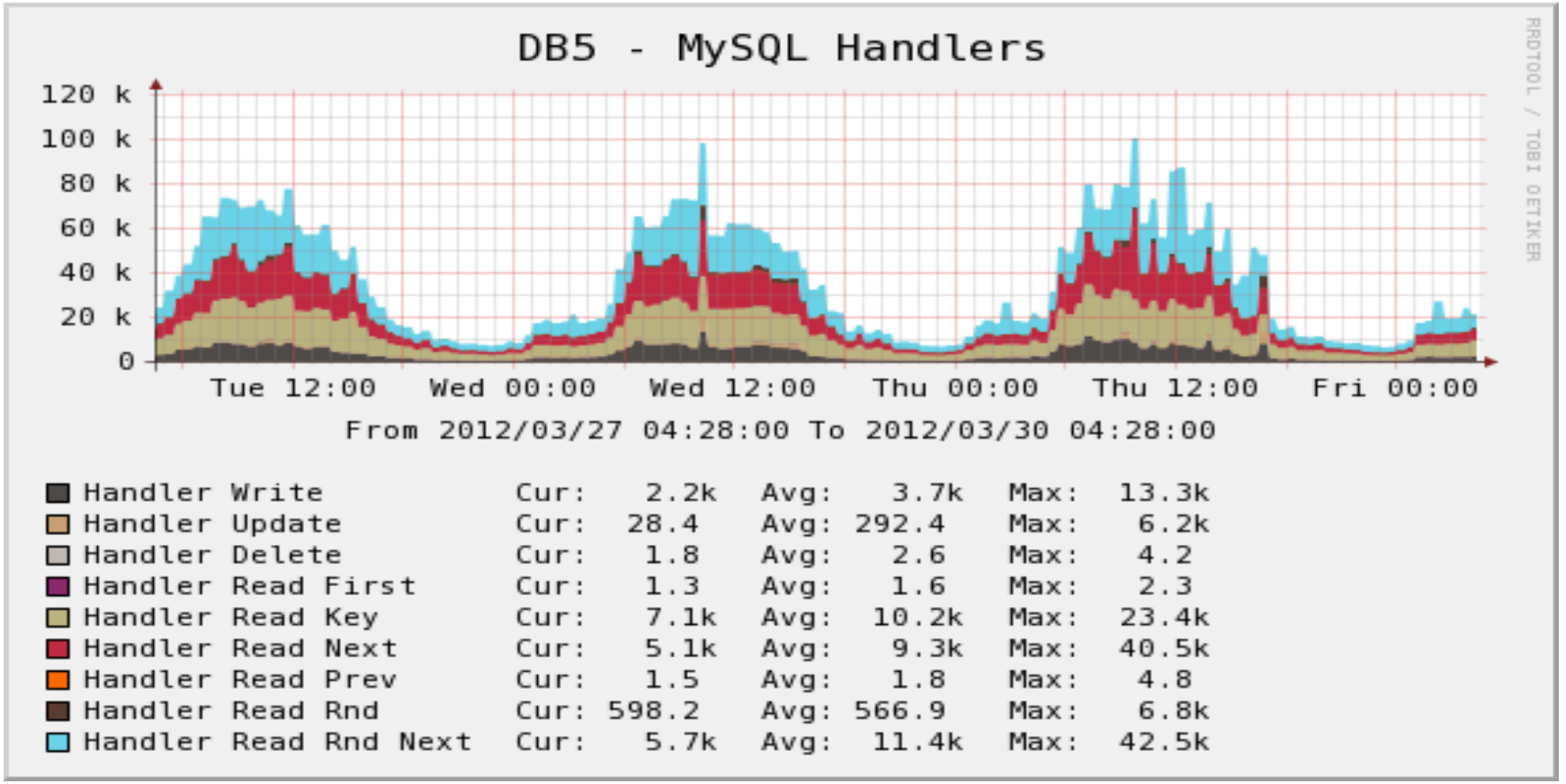
- Max (used) connections
- Aborted Clients/Connections
- Current / New connections

MySQL Files and Tables



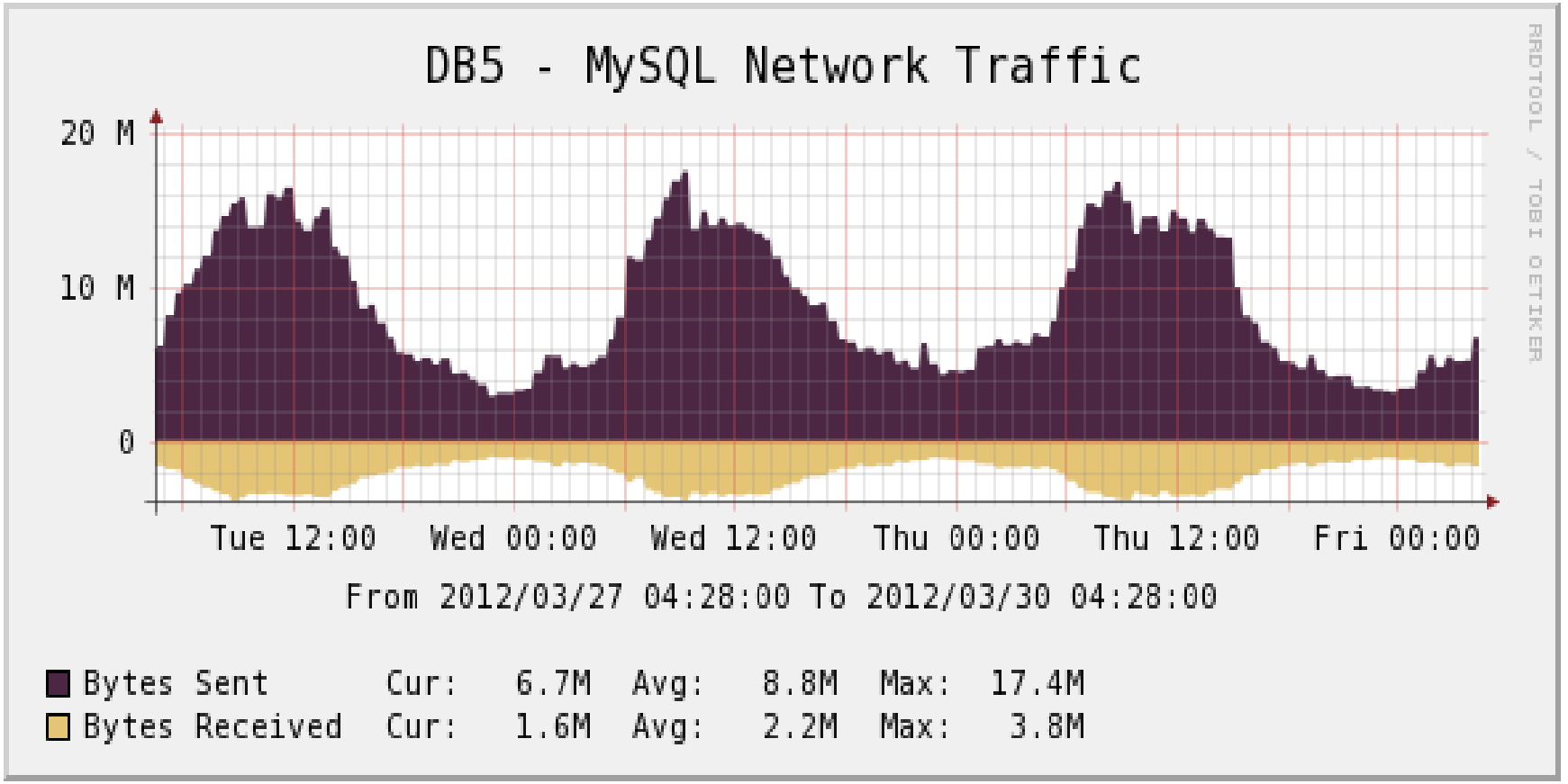
- File Handlers status
- Table cache activity

MySQL Handlers



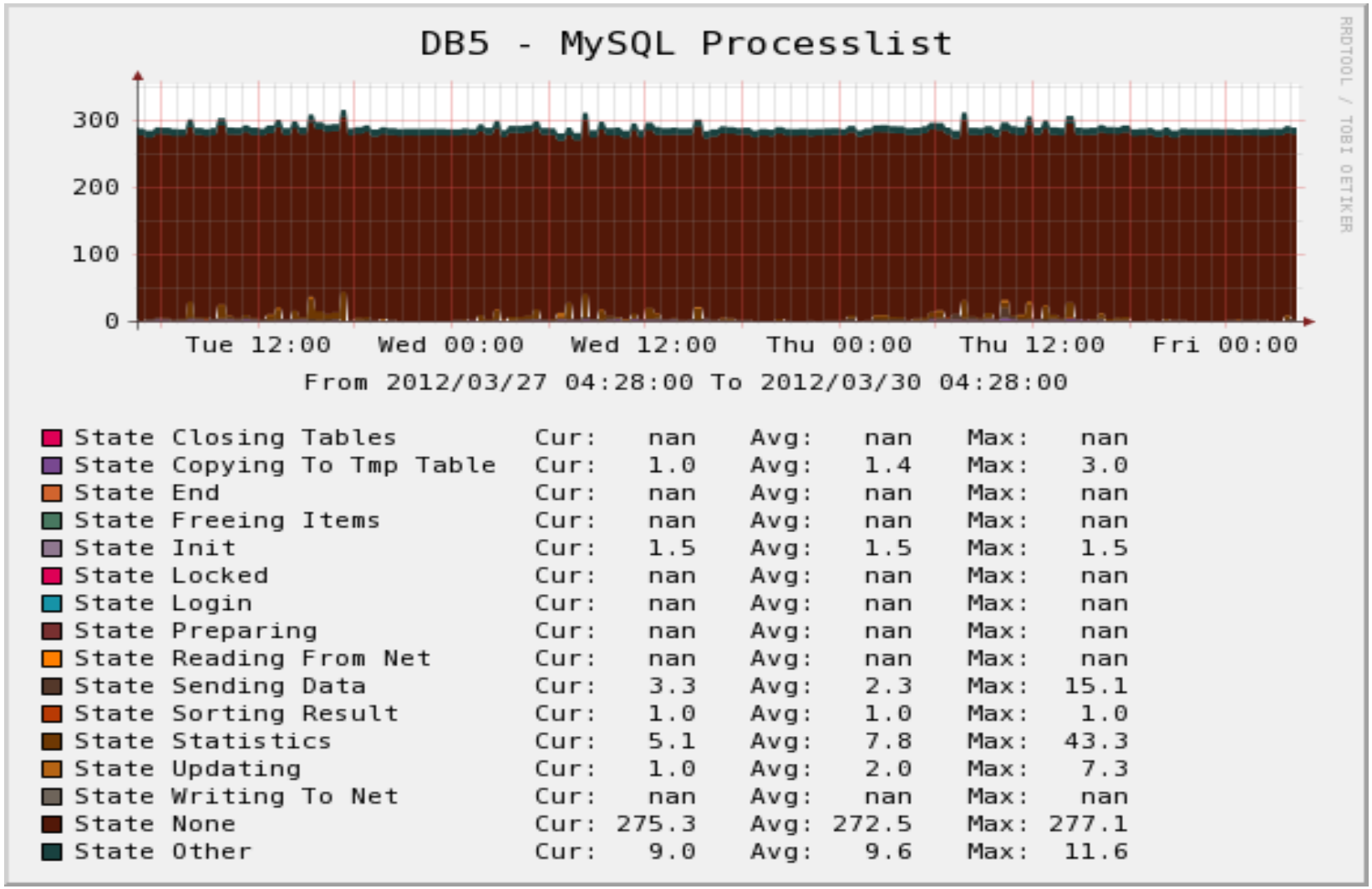
- Storage engine handlers

MySQL Network Traffic



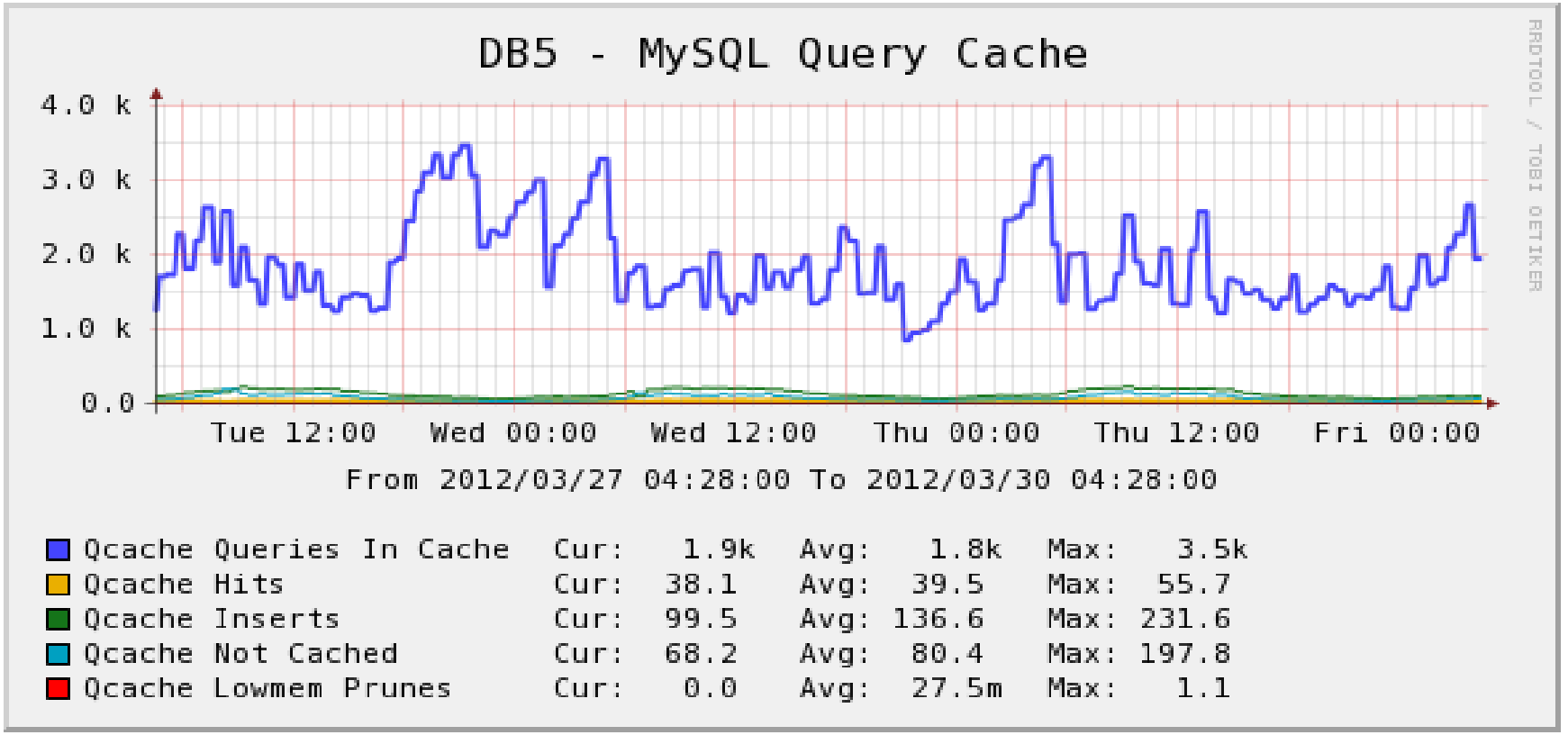
- Bytes sent/received to/from MySQL

MySQL Processlist



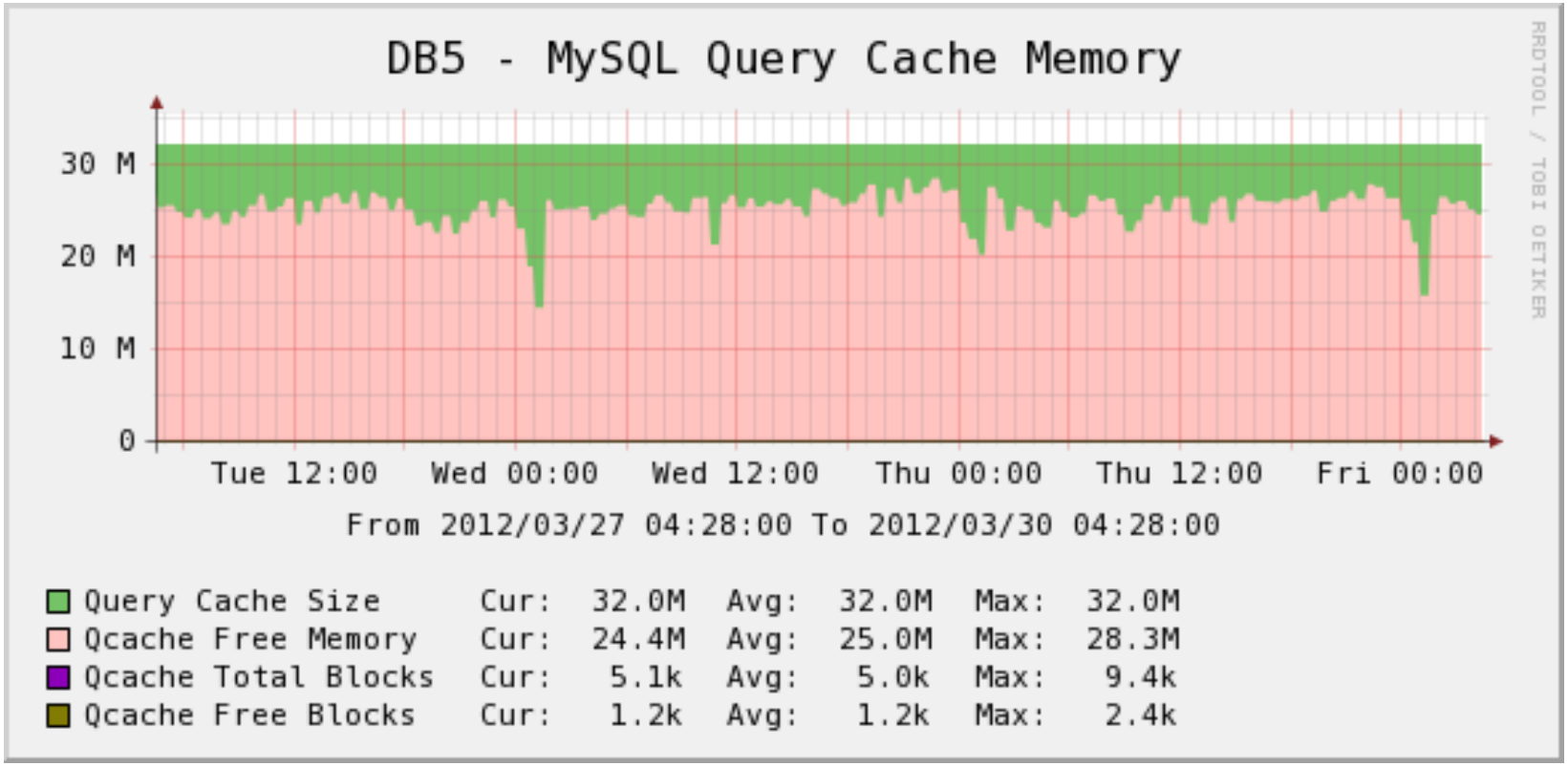
- State resulting from SHOW FULL PROCESSLIST

MySQL Query Cache



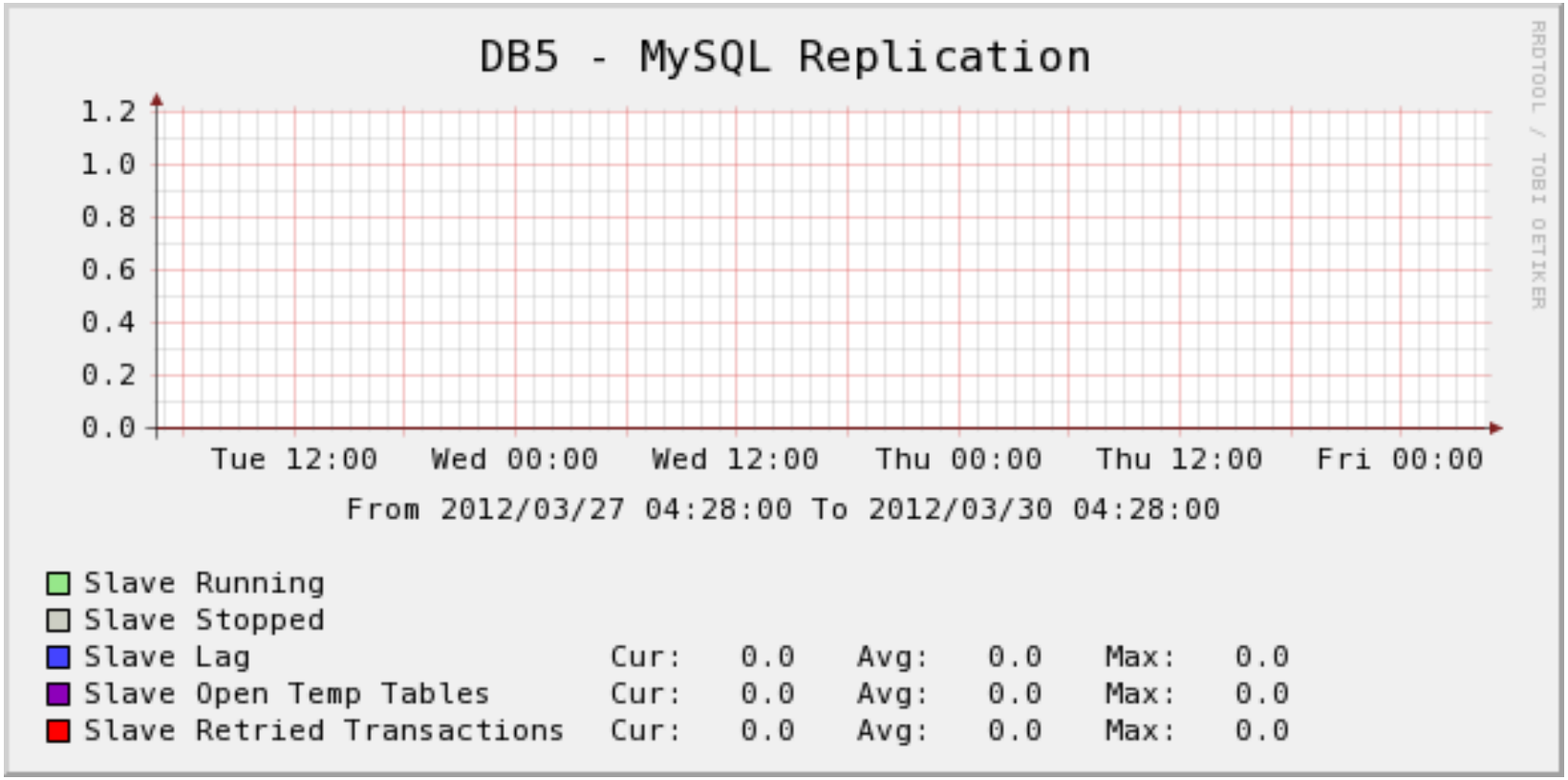
- Query Cache statistics
- Hits/Inserts ratio

MySQL Query Cache Memory



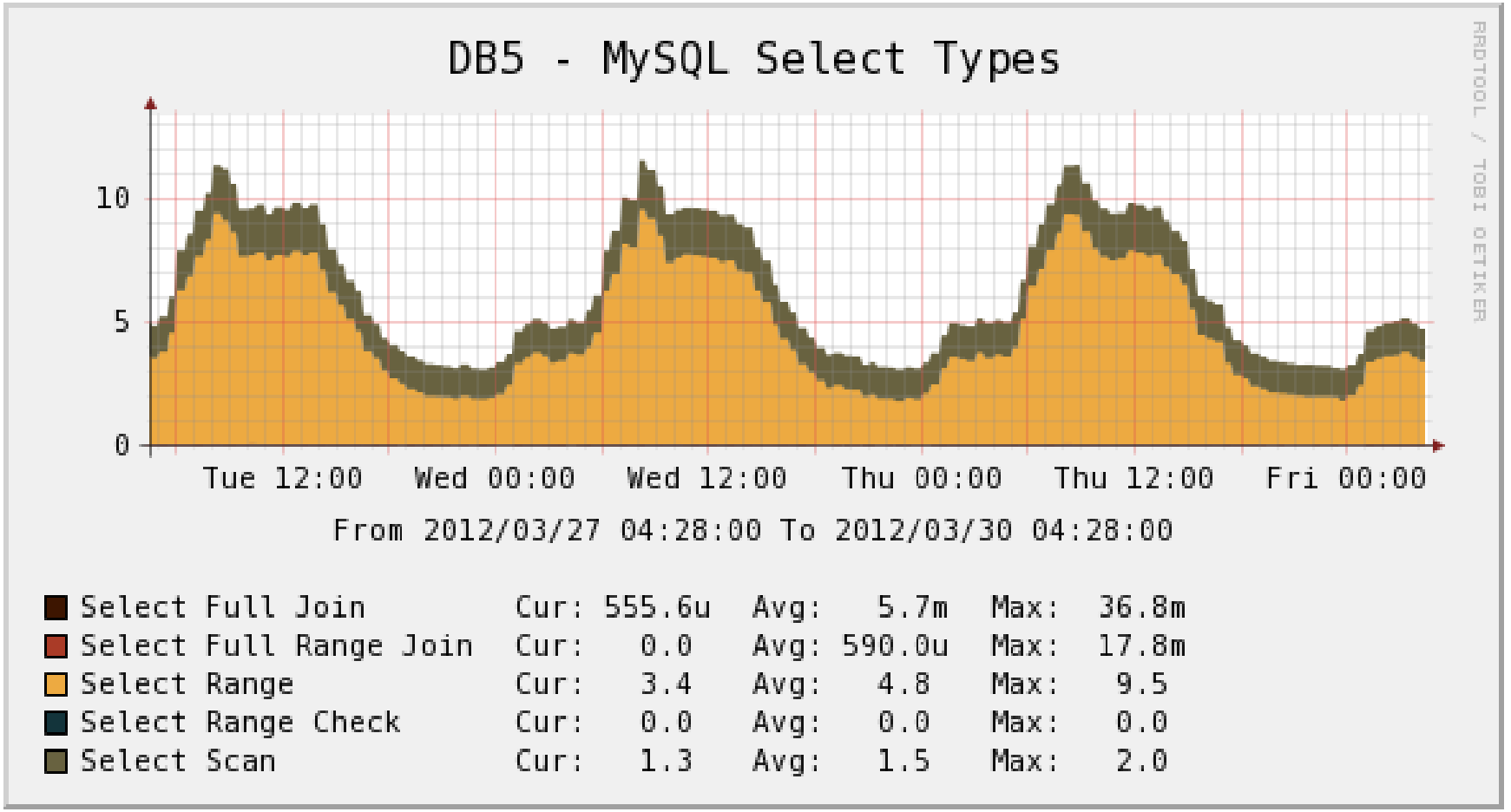
- Query Cache memory utilization

MySQL Replication



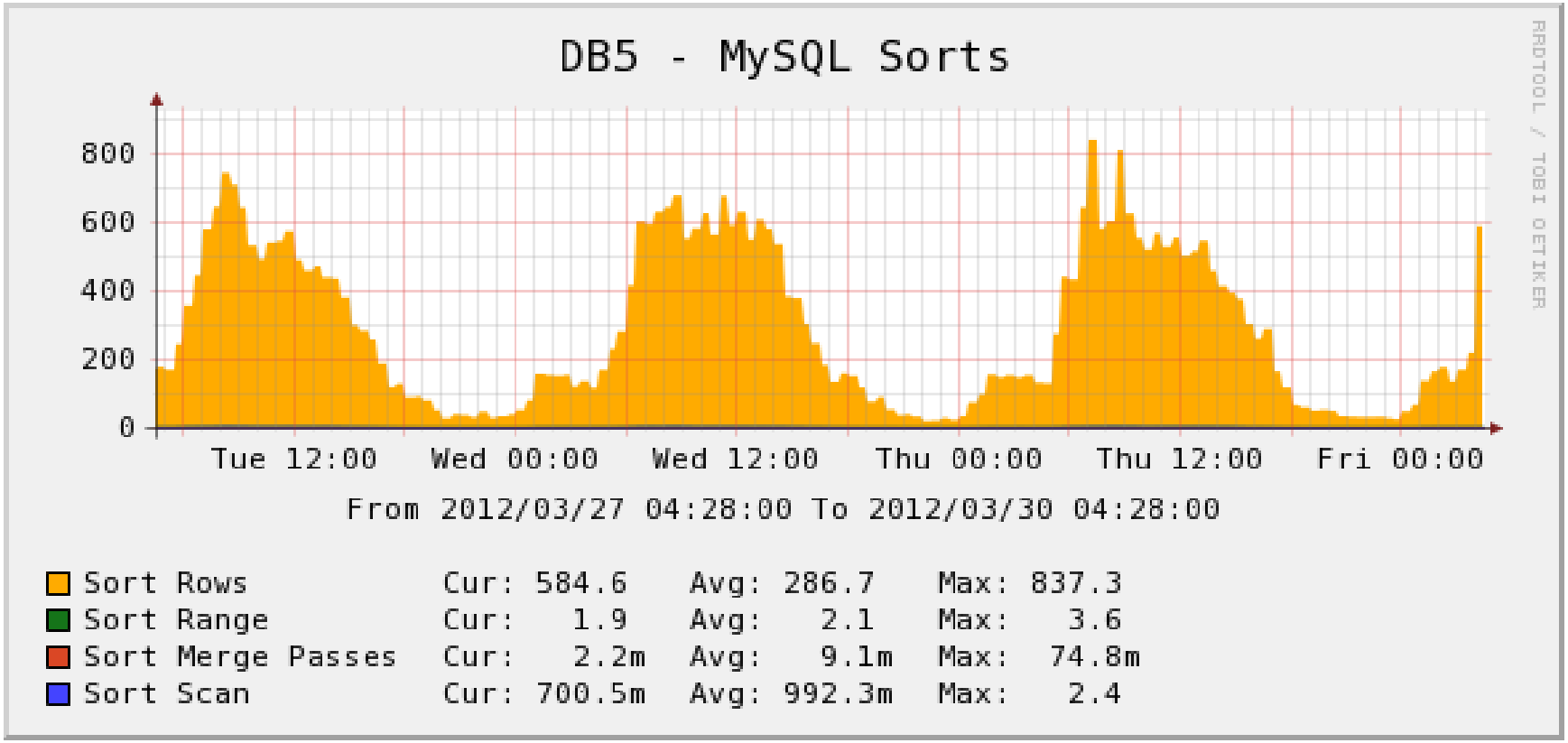
- MySQL (asynchronous) Replication lag

MySQL Select Types



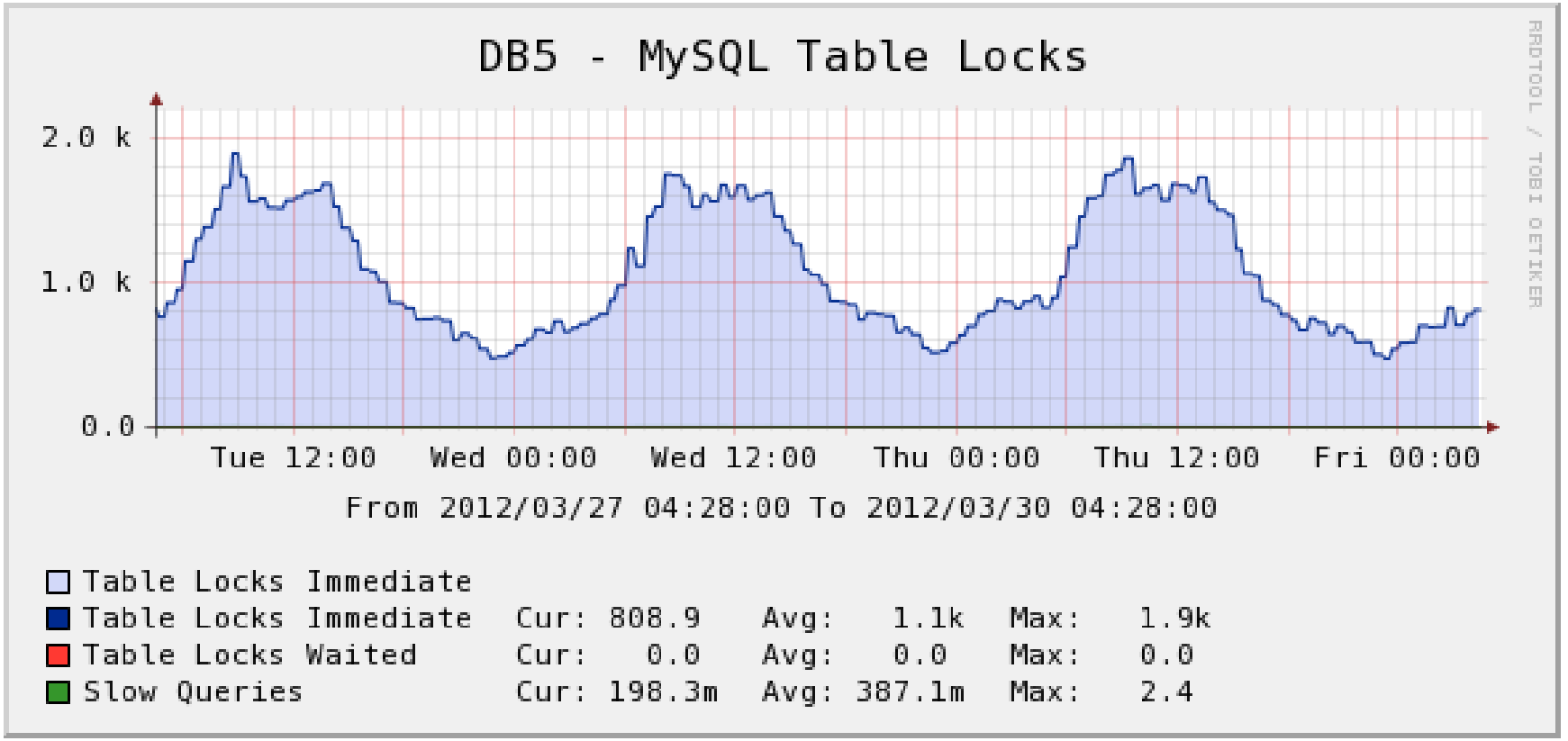
- Type of join in SELECT

MySQL Sorts



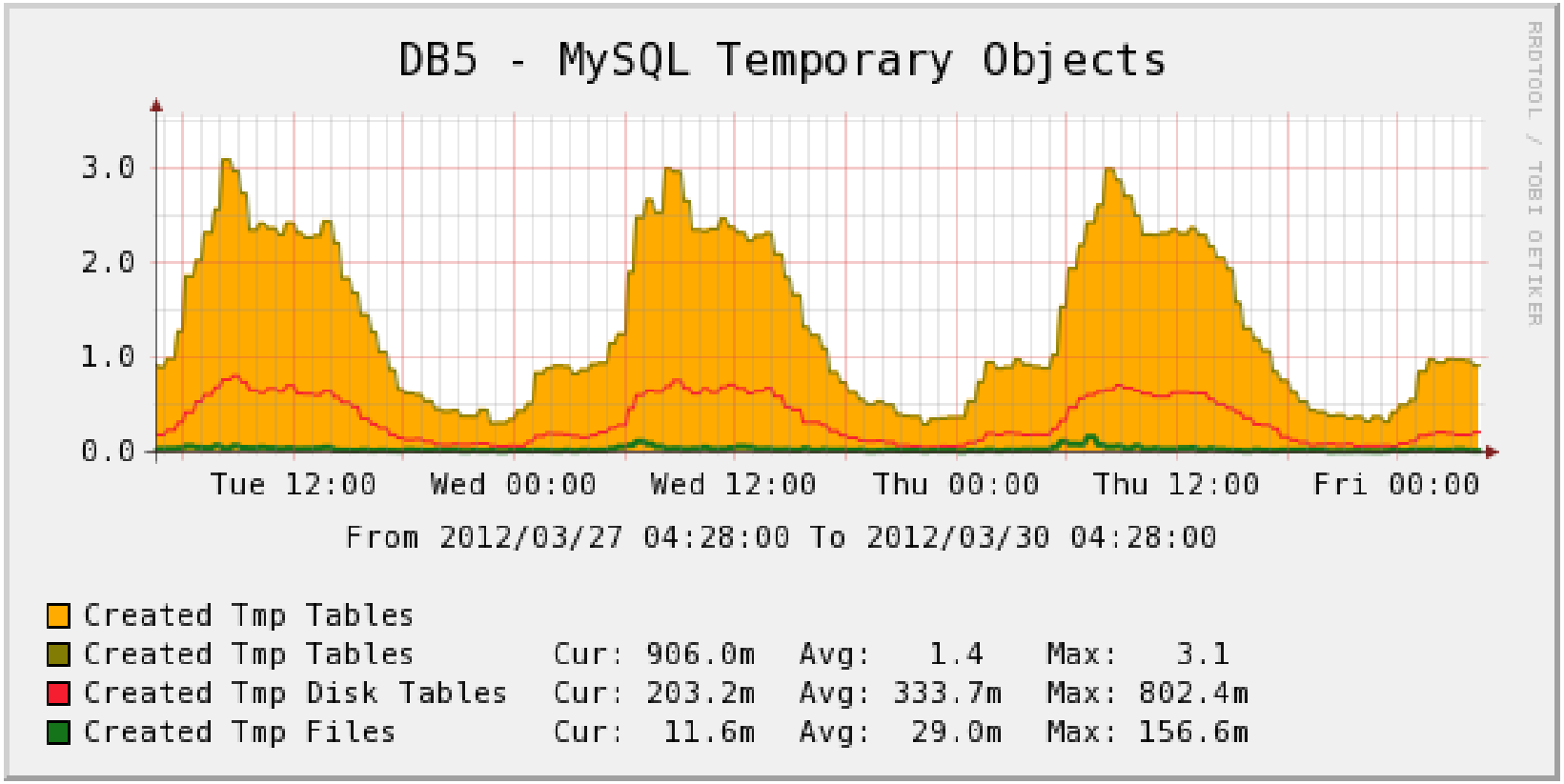
- Rows sorted
- Merge passes
- Range/Scan queries

MySQL Table Locks



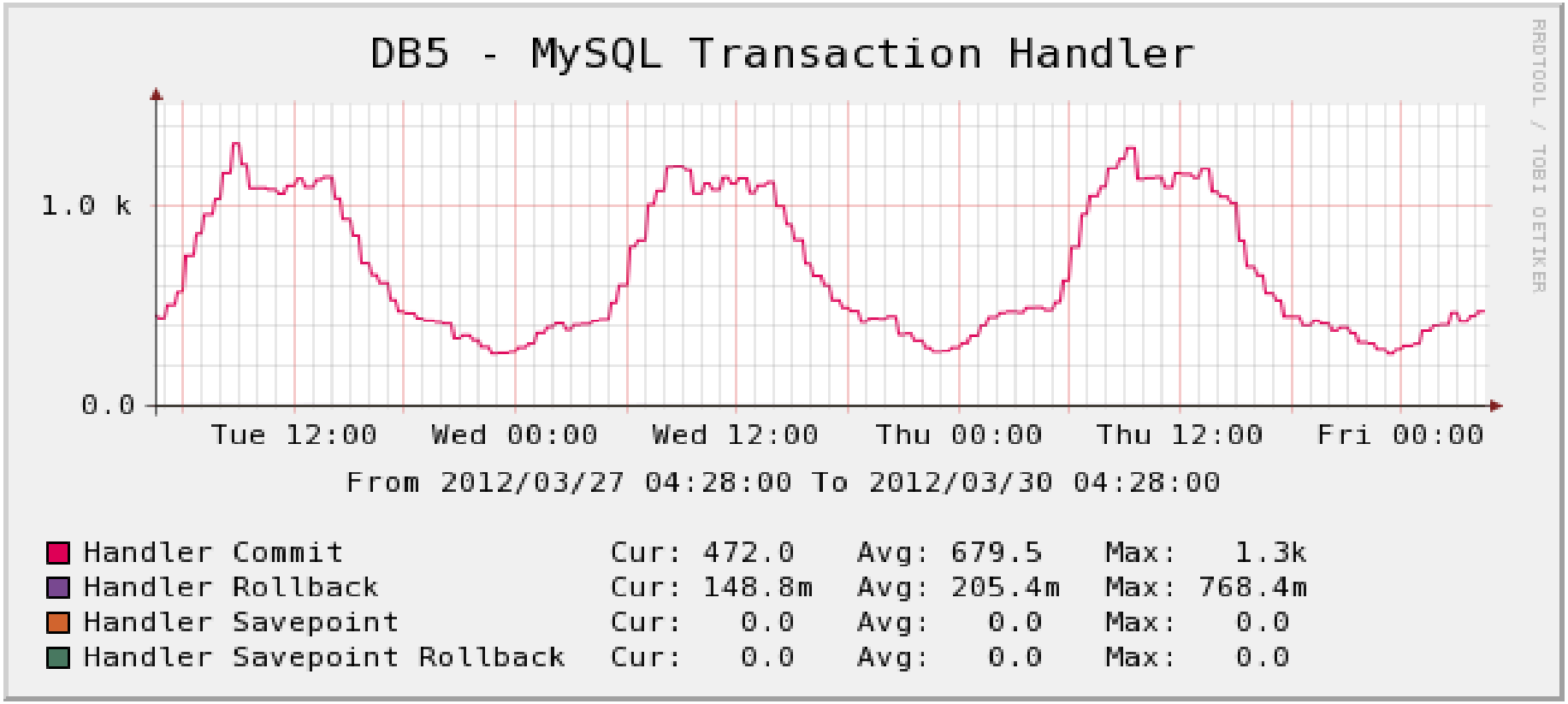
- Table level locking
  - MyISAM vs InnoDB
- Slow queries

MySQL Temporary Objects



- Temporary tables
  - In memory vs on disk
- Temporary files

MySQL Transaction Handler



- COMMIT
- ROLLBACK
- SAVEPOINT