WiredTiger scalability (and RocksDB, too)

Benchmarks and observations

Vadim Tkachenko
Percona, CTO, Performance Enthusiast

Is performance important?

How the database can be more efficient

Getting more with less

In the end, it’s about money:

- Can we use less hardware/cloud instances?

Benchmark

Sysbench for MongoDB

https://github.com/Percona-Lab/sysbench/tree/dev-mongodb-support-1.0

What is sysbench?

Micro-synthetic benchmark

Primitive operations by primary key (collection _id)

Easy to setup and run

These operations are building blocks for more complex queries

Sysbench operations

Read-Only

- Point lookup
- Range select
- Range aggregation
- Distinct over range

Read-Write

- Read-only plus:
- Modify non-indexed field
- Modify indexed field
- Insert/delete row
  - Transactional for MySQL
  - Find-and-modify for MongoDB to guarantee atomic operation

Sysbench cons

Does not represent any real workload

But keep in mind:

It runs elementary operations that are building blocks for more complex operations

If there is problem in sysbench workload – most likely it will be a problem in real life

The reverse is not necessarily true…

Benchmark setup: hardware

Supermicro blade servers

CPU: Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz

Total RAM: 256GB

Client and Server identical, connected via 10GB network

Storage, SSD:

- Samsung 850 Pro, SATA – consumer grade, $500/TB
- Samsung SM863, SATA – server grade, $600/TB
- Samsung PM1725, NVMe, – high end, $1,300/TB
  - Big progress from $10,000+/TB, ~5 years ago

Benchmark setup: configuration

8 collections x 60mln documents each (~100GB of uncompressed data)

Threads from 1 to 1000

Storage engines: WiredTiger and MongoRocks (RocksDB)

Cachesize: from 4GB to 200GB

- Cgroup limits mongod memory with 2x cachesize
- Both WiredTiger and MongoRocks keep cached data in OS cache
  - Need special attention, as otherwise all data (100GB) gets cached in OS

Engines specifics

WiredTiger

- B-Tree based
  - Fast reads
  - Slow random writes
- LSM engine is not documented

MongoRocks

- Based on RocksDB – LSM tree
  - Fast random writes
  - Slower reads

Ways to scale

- Multiple user threads
- Increase memory
- Storage with better performance
- Multiple servers (scale out) – not in this talk

Benchmark methodology

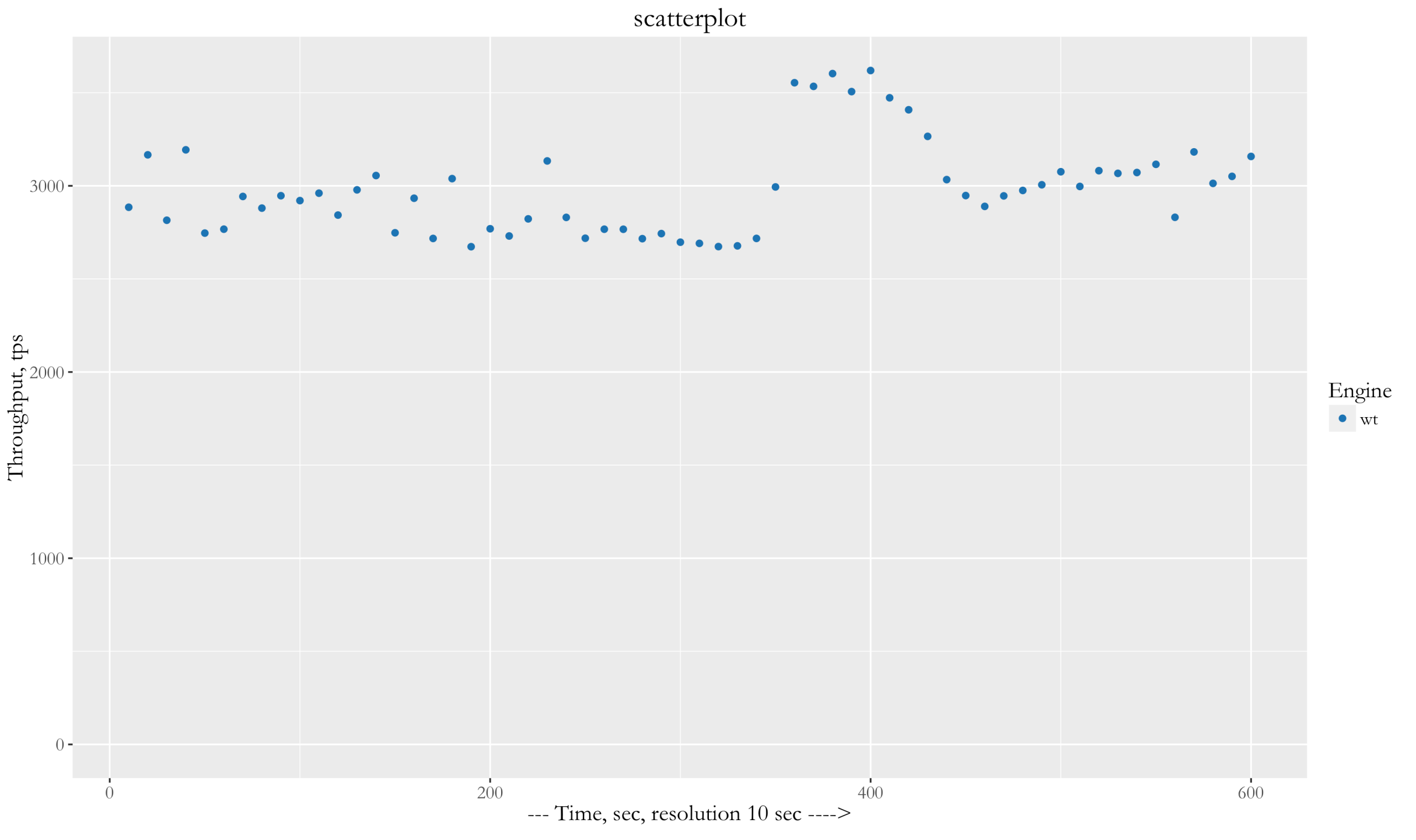
- Measurements with 10sec intervals
- Throughput and response time for the 10sec intervals
- It helps to see trends and stalls
- Benchmark length – 5-10 mins
- Threads in [1-1000] interval
- Cachesize in [4-200]GB interval
- A lot of data points

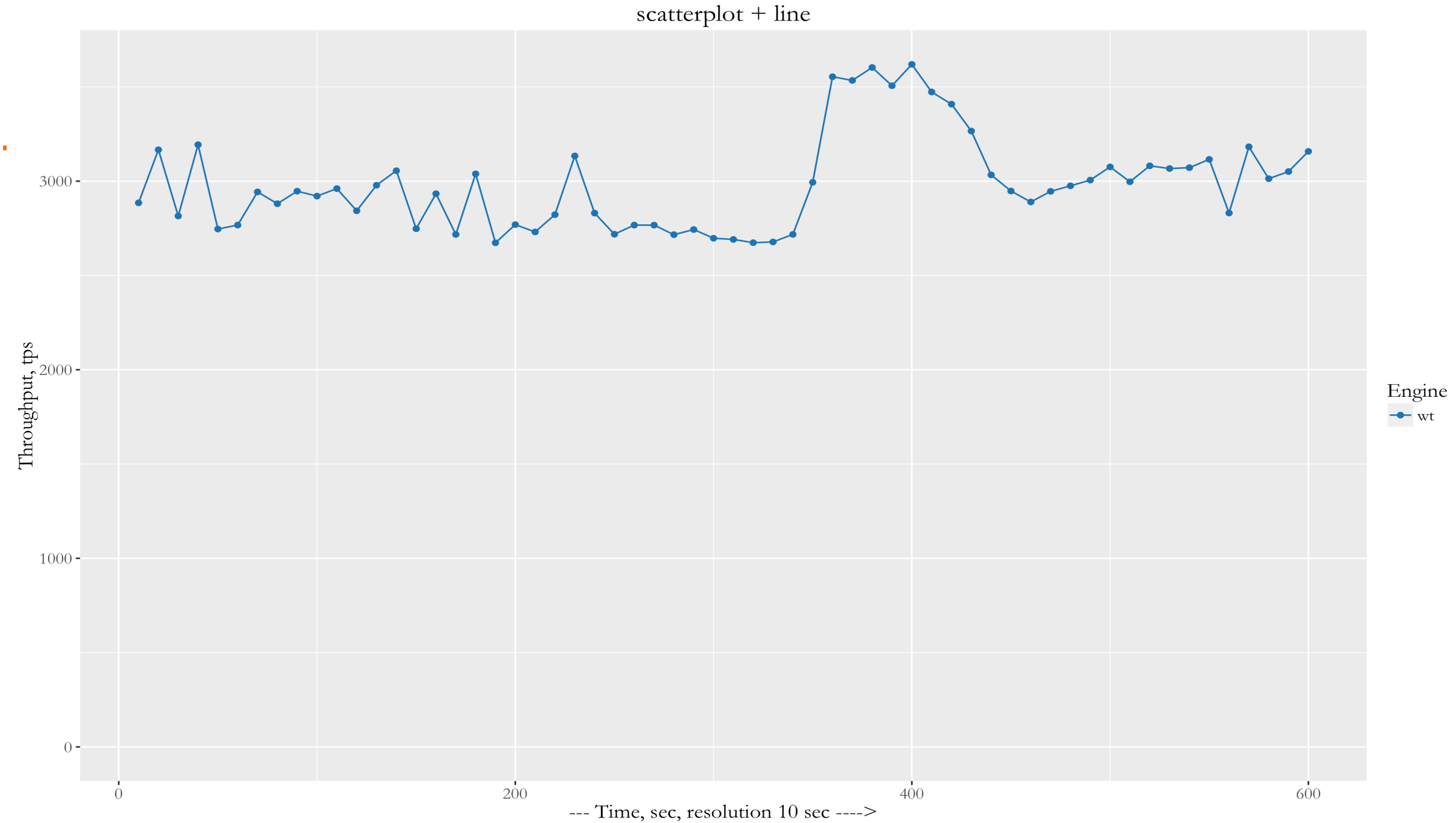
A single number is not interesting

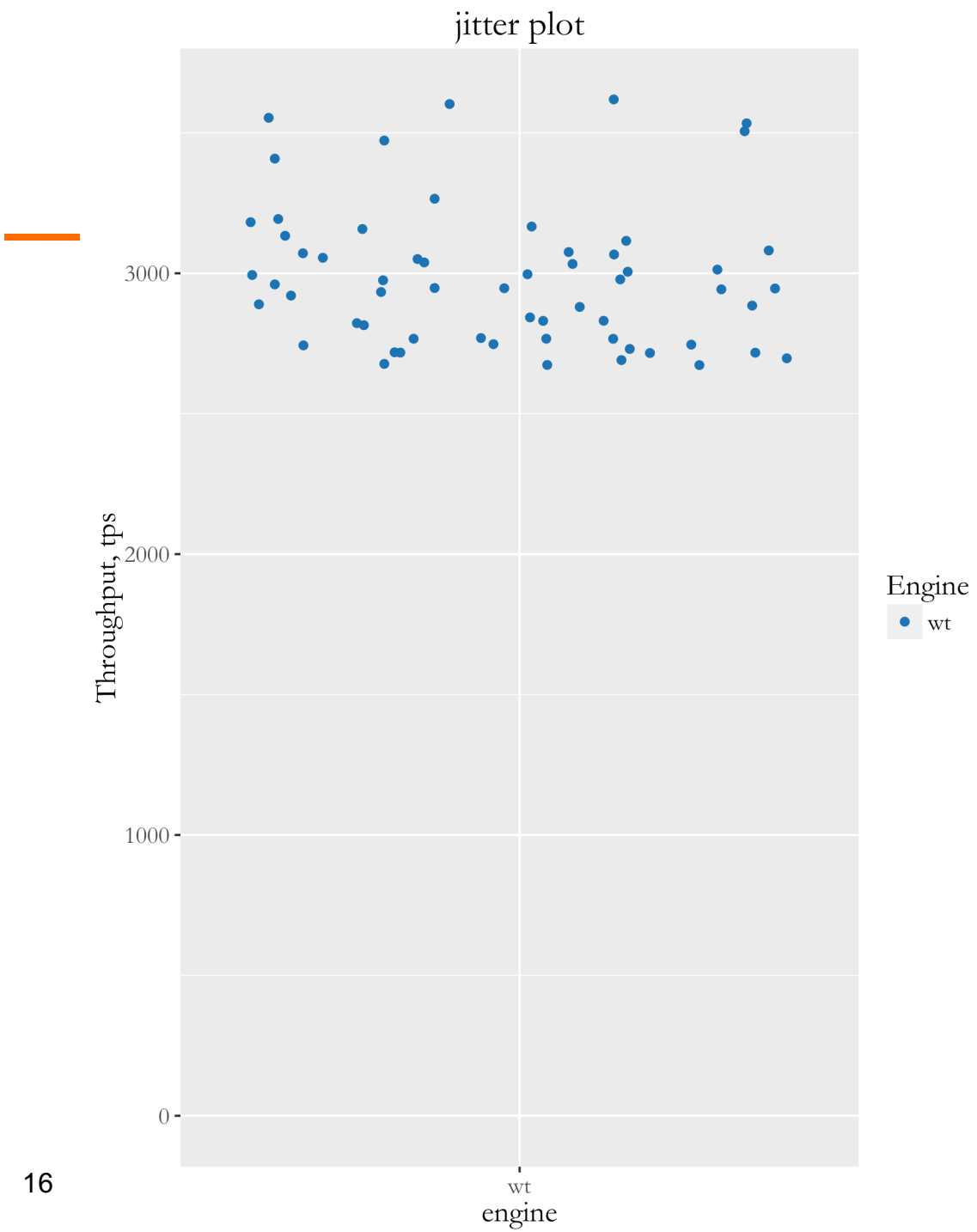
The trend is interesting

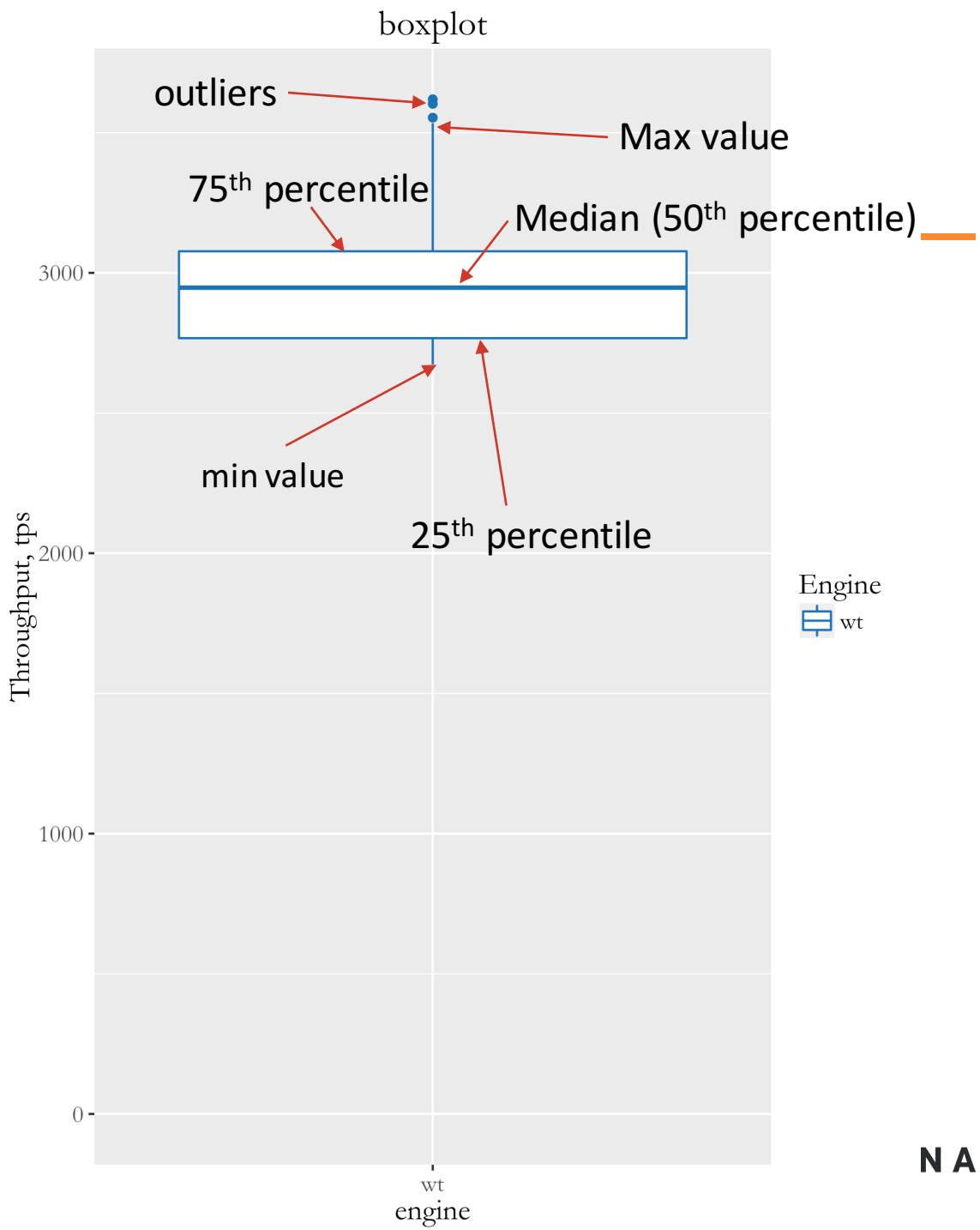
Ways to represent data

- Scatterplots
- Trend lines
- Jitters
- Boxplots


scatterplot
Throughput, tps
3000
2000
1000
0
0
200
400
600
--- Time, sec, resolution 10 sec ---->
Engine
wt


scatterplot + line
Throughput, tps
3000
2000
1000
0
0
200
400
600
--- Time, sec, resolution 10 sec ---->
Engine
wt


jitter plot
3000
2000
1000
0
Throughput, tps
wt
engine
Engine
wt

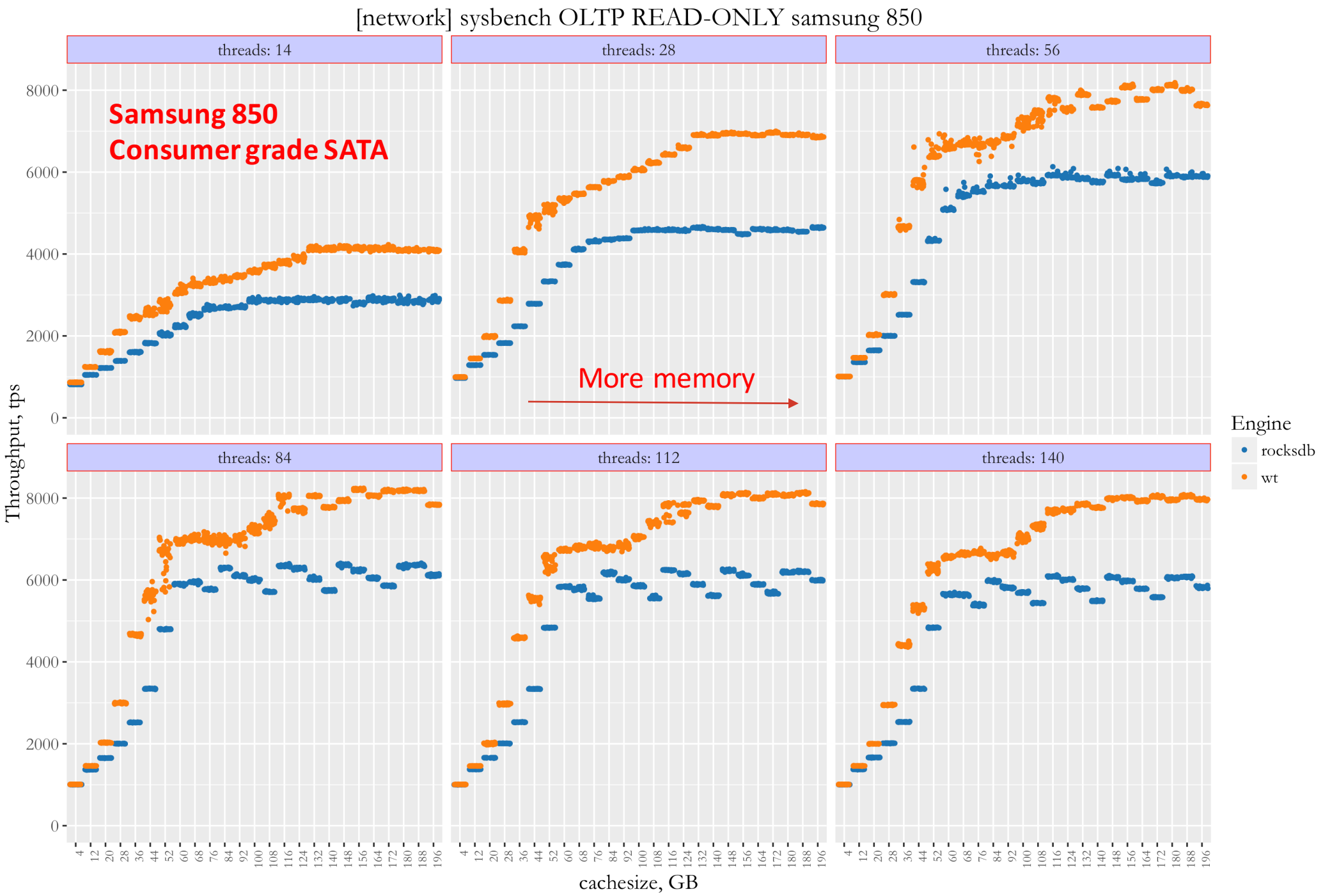

boxplot
outliers
Max value
75th percentile
Median (50th percentile)
min value
25th percentile
3000
2000
1000
0
Throughput, tps
wt
engine
Engine
wt

Results

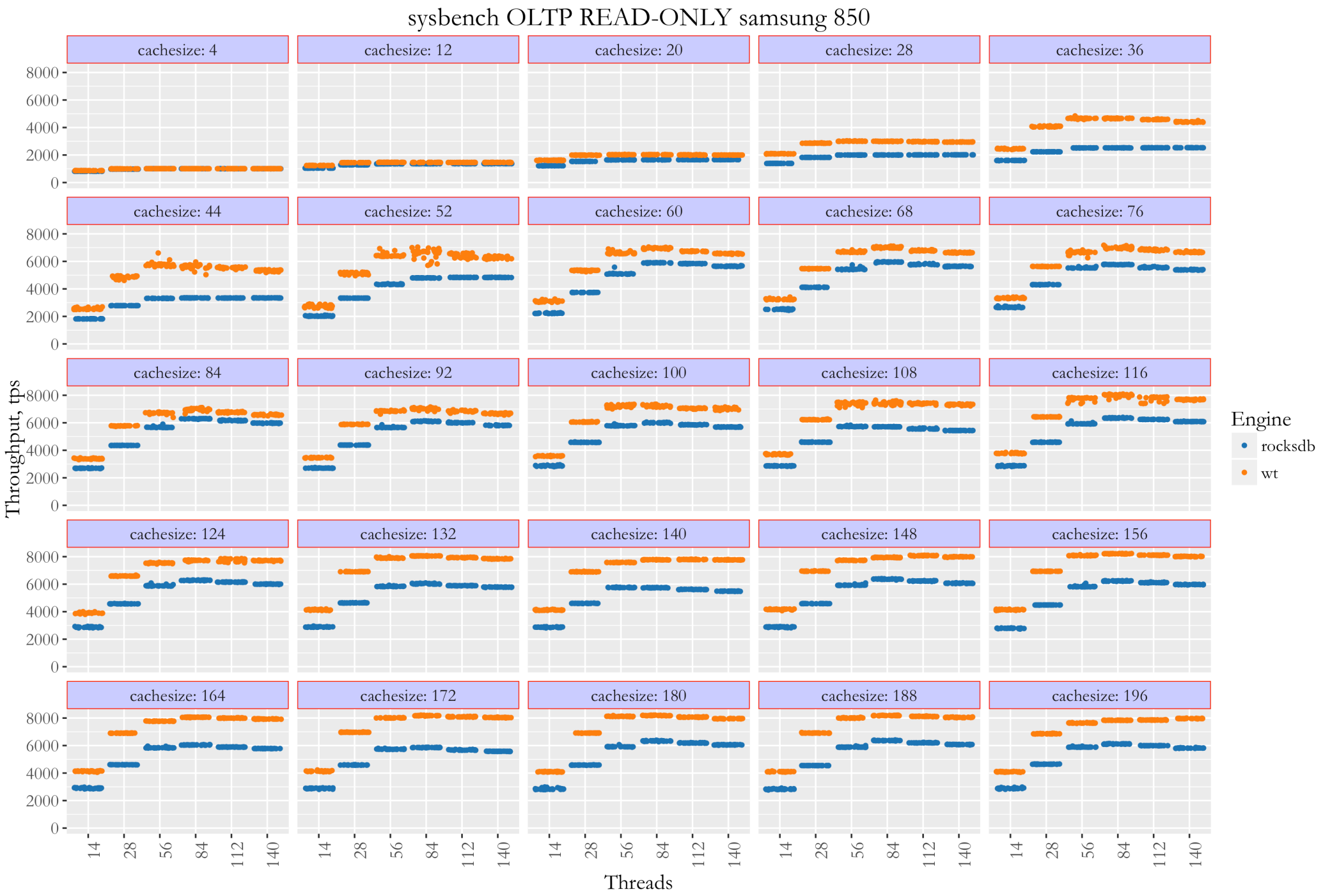
PERCONA

Read-only

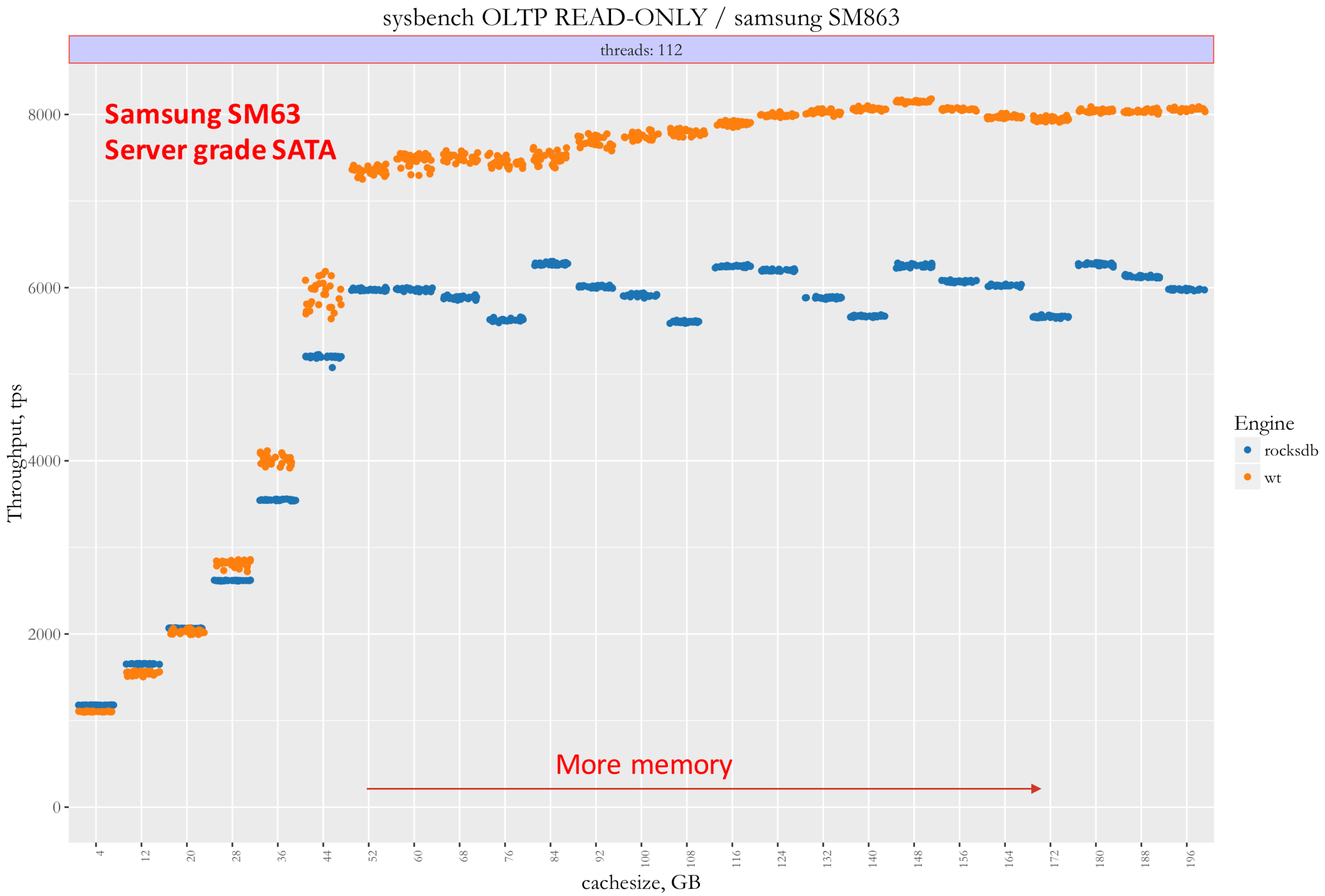
- Different cachesizes
  - Small cachesize shows ability to handle growing data
    - Pressure on IO-reads
  - Big cachesize shows the efficiency of internal structures
- Read-only use cases:
  - Reporting
  - Read-intensive components (blog, wiki etc)
- Problems in read-only scalability signal design problems
  - e.g., mutex abuse to access a shared structure


[network] sysbench OLTP READ-ONLY samsung 850
threads: 14
threads: 28
threads: 56
threads: 84
threads: 112
threads: 140
Samsung 850
Consumer grade SATA
More memory
Throughput, tps
cachesize, GB
Engine
rocksdb
wt

PERCONA


sysbench OLTP READ-ONLY samsung 850
cachesize: 4
cachesize: 12
cachesize: 20
cachesize: 28
cachesize: 36
cachesize: 44
cachesize: 52
cachesize: 60
cachesize: 68
cachesize: 76
cachesize: 84
cachesize: 92
cachesize: 100
cachesize: 108
cachesize: 116
cachesize: 124
cachesize: 132
cachesize: 140
cachesize: 148
cachesize: 156
cachesize: 164
cachesize: 172
cachesize: 180
cachesize: 188
cachesize: 196
Throughput, tps
0
2000
4000
6000
8000
14
28
56
84
112
140
Threads
Engine
rocksdb
wt

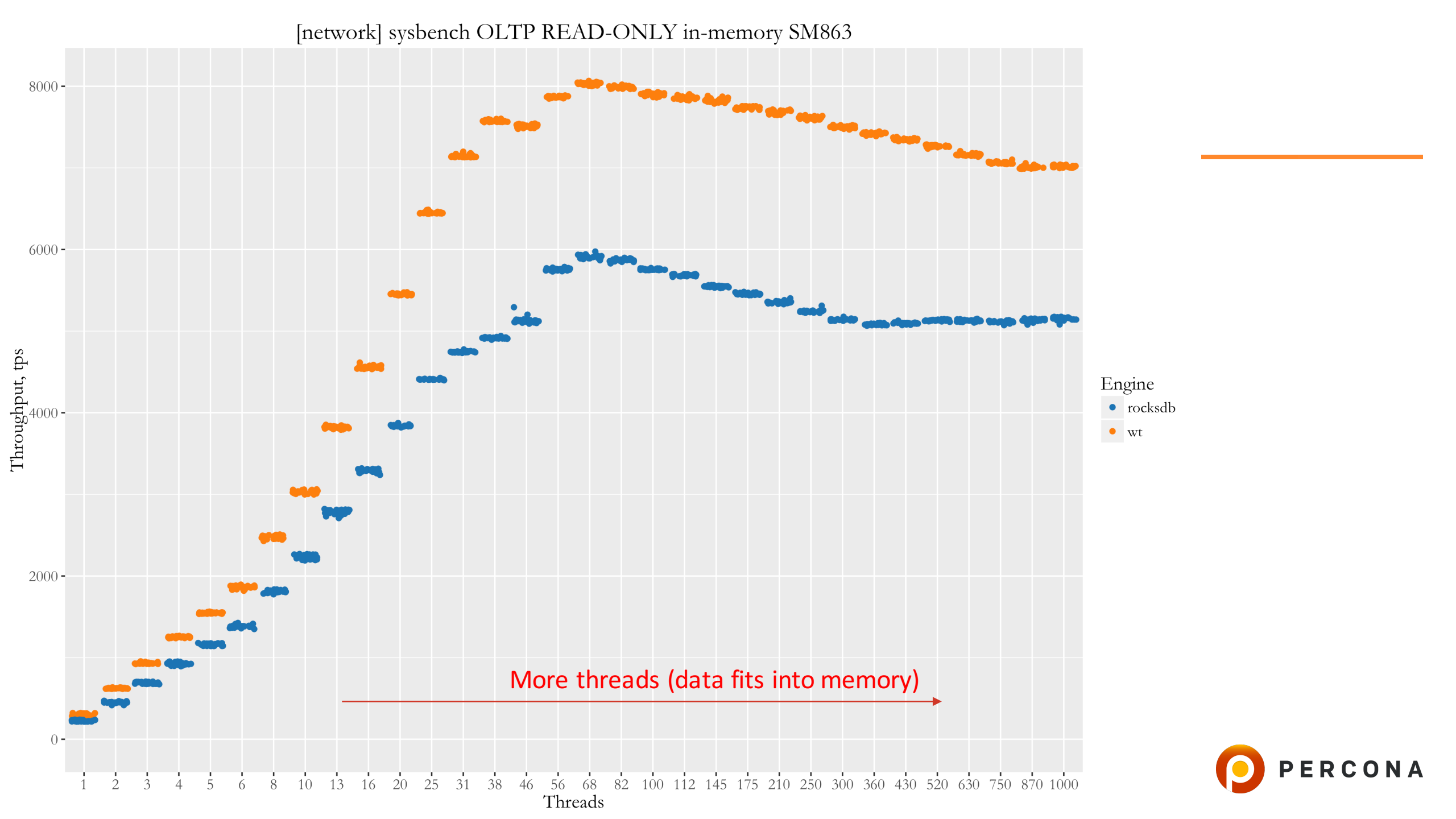
PERCONA


sysbench OLTP READ-ONLY / samsung SM863
threads: 112
Samsung SM63
Server grade SATA
More memory
Throughput, tps
cachesize, GB
Engine
rocksdb
wt

PERCONA

Multi-threaded scalability Read-Only

- Data fits into memory (200GB cachesize)


[network] sysbench OLTP READ-ONLY in-memory SM863
Throughput, tps
8000
6000
4000
2000
0
1 2 3 4 5 6 8 10 13 16 20 25 31 38 46 56 68 82 100 112 145 175 210 250 300 360 430 520 630 750 870 1000
Threads
More threads (data fits into memory)
Engine
rocksdb
wt

PERCONA

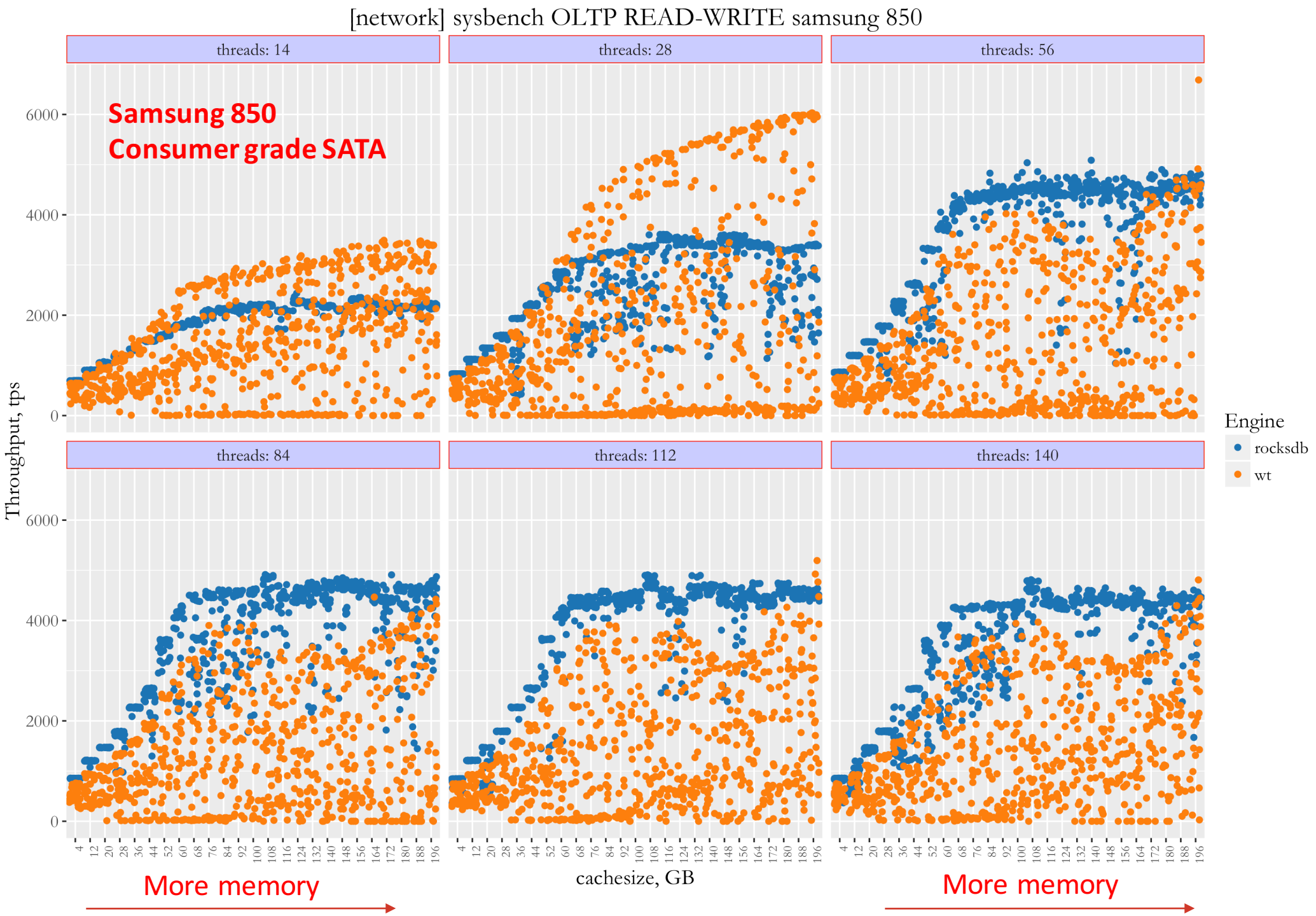
Read-only observation

- Both engines scale reasonable on read-only
- WiredTiger benefits from more memory, scales better with increasing threads
- WiredTiger seems suitable for read-intensive workloads, a lot of cached data

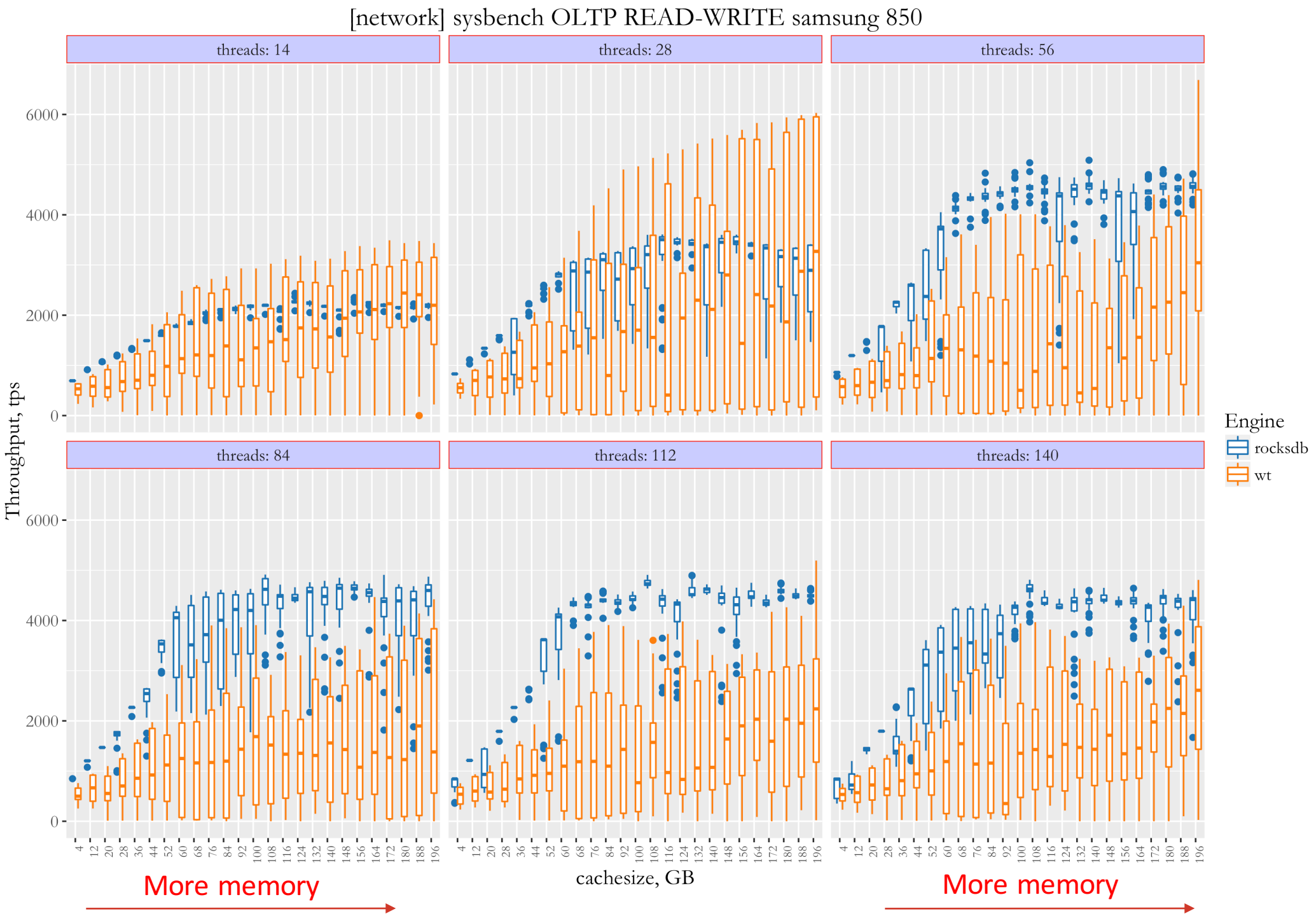
Read-Write results

Memory varies

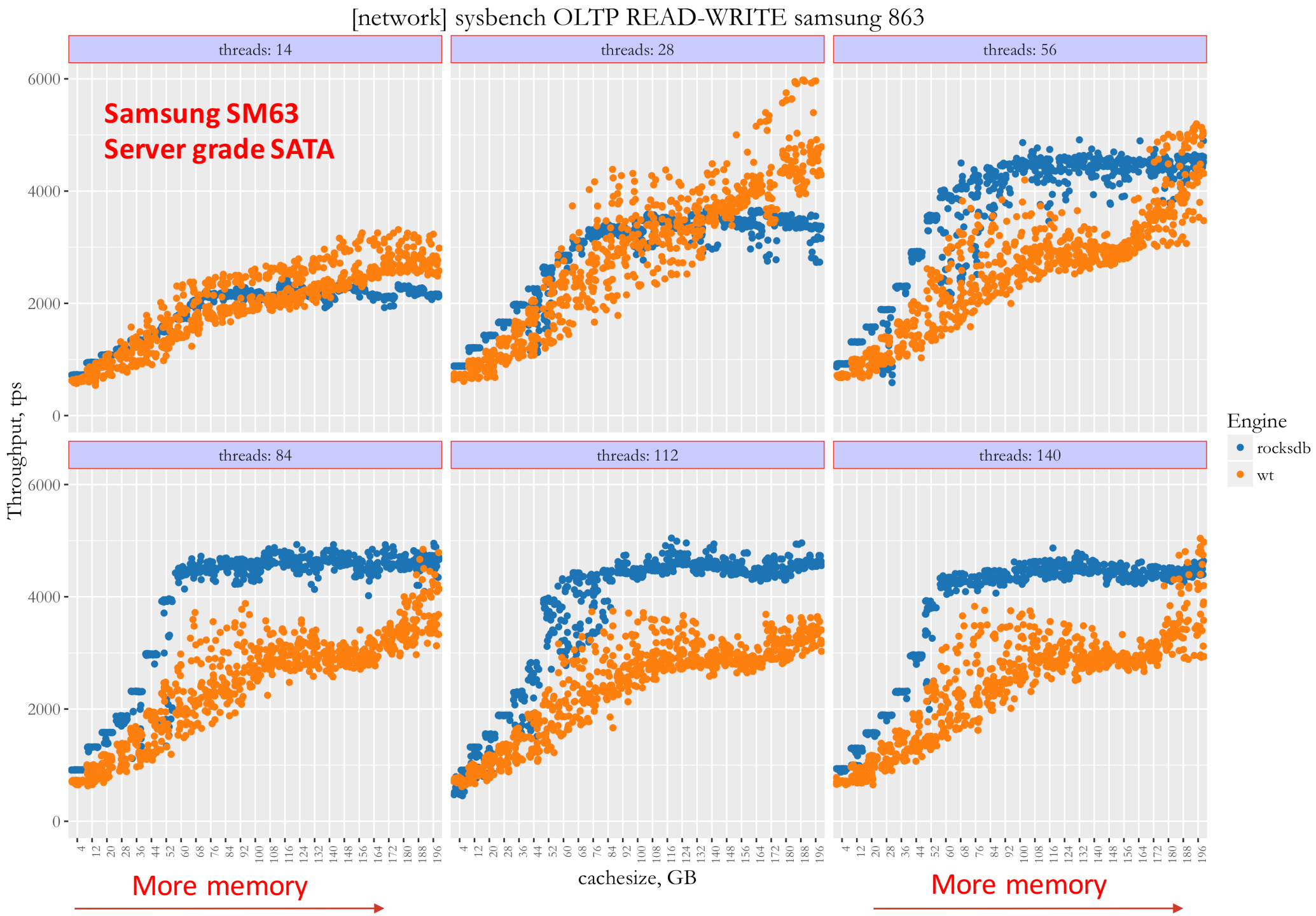
PERCONA


[network] sysbench OLTP READ-WRITE samsung 850
threads: 14
threads: 28
threads: 56
threads: 84
threads: 112
threads: 140
Samsung 850
Consumer grade SATA
Throughput, tps
0
2000
4000
6000
4
12
20
28
36
44
52
60
68
76
84
92
100
108
116
124
132
140
148
156
164
172
180
188
196
cachesize, GB
More memory
More memory
Engine
rocksdb
wt

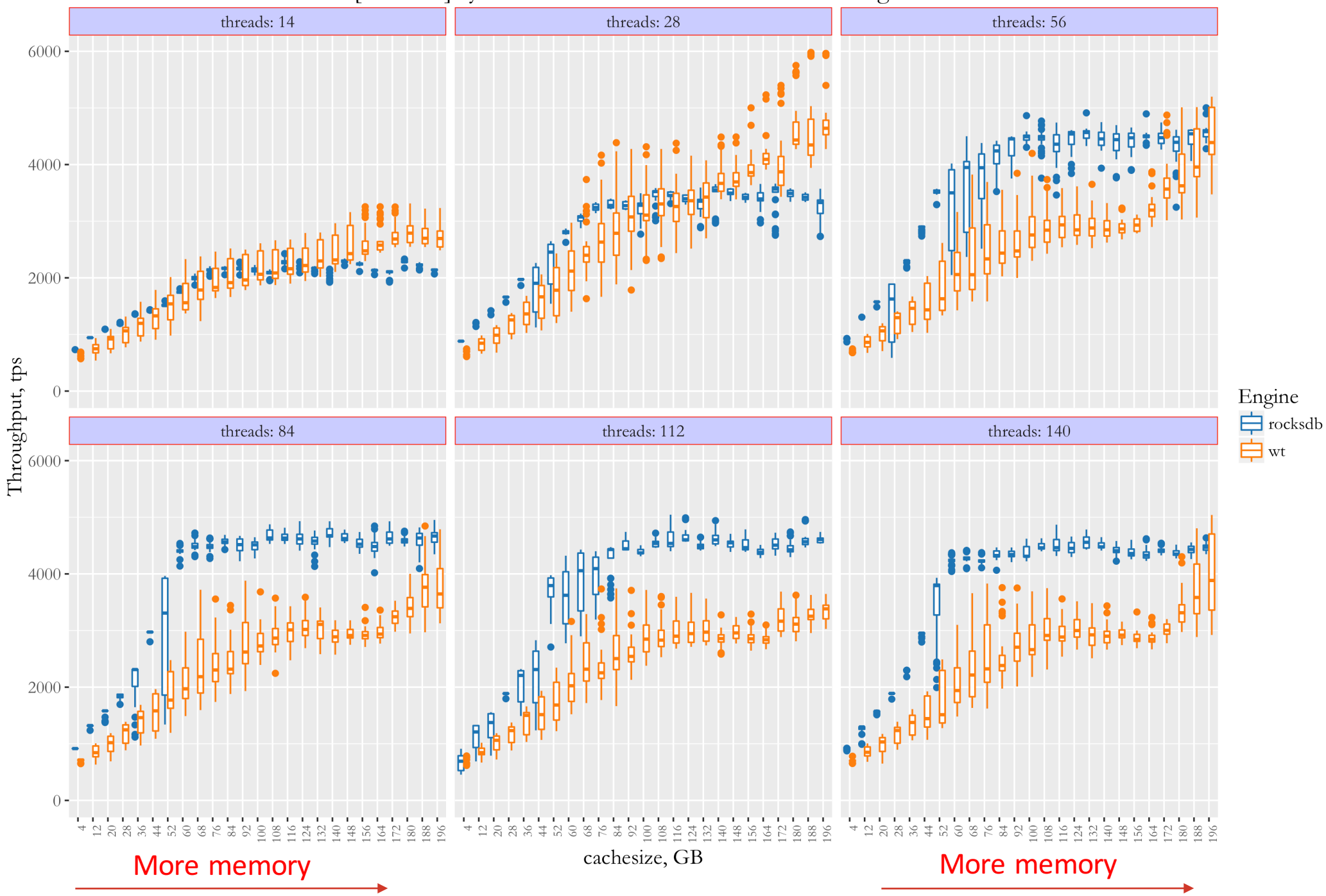
PERCONA


[network] sysbench OLTP READ-WRITE samsung 850
threads: 14
threads: 28
threads: 56
threads: 84
threads: 112
threads: 140
Throughput, tps
0
2000
4000
6000
4
12
20
28
36
44
52
60
68
76
84
92
100
108
116
124
132
140
148
156
164
172
180
188
196
cachesize, GB
More memory
More memory
Engine
rocksdb
wt

PERCONA


[network] sysbench OLTP READ-WRITE samsung 863
threads: 14
threads: 28
threads: 56
threads: 84
threads: 112
threads: 140
Samsung SM63
Server grade SATA
Throughput, tps
0
2000
4000
6000
4
12
20
28
36
44
52
60
68
76
84
92
100
108
116
124
132
140
148
156
164
172
180
188
196
cachesize, GB
More memory
More memory
Engine
rocksdb
wt

PERCONA


[network] sysbench OLTP READ-WRITE samsung 863
threads: 14
threads: 28
threads: 56
threads: 84
threads: 112
threads: 140
Throughput, tps
0
2000
4000
6000
4
12
20
28
36
44
52
60
68
76
84
92
100
108
116
124
132
140
148
156
164
172
180
188
196
cachesize, GB
More memory
More memory
Engine
rocksdb
wt

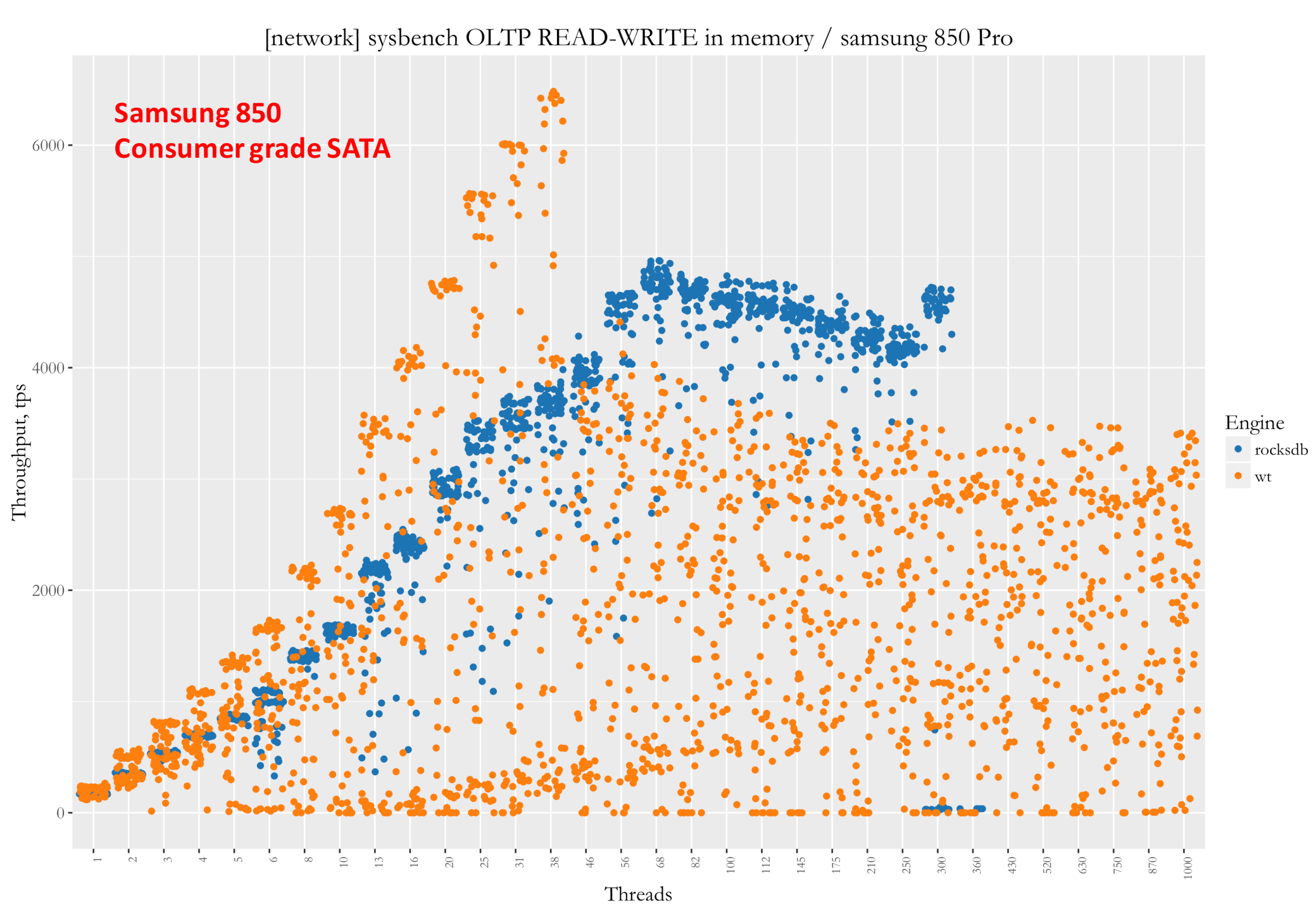
PERCONA

Read-Write

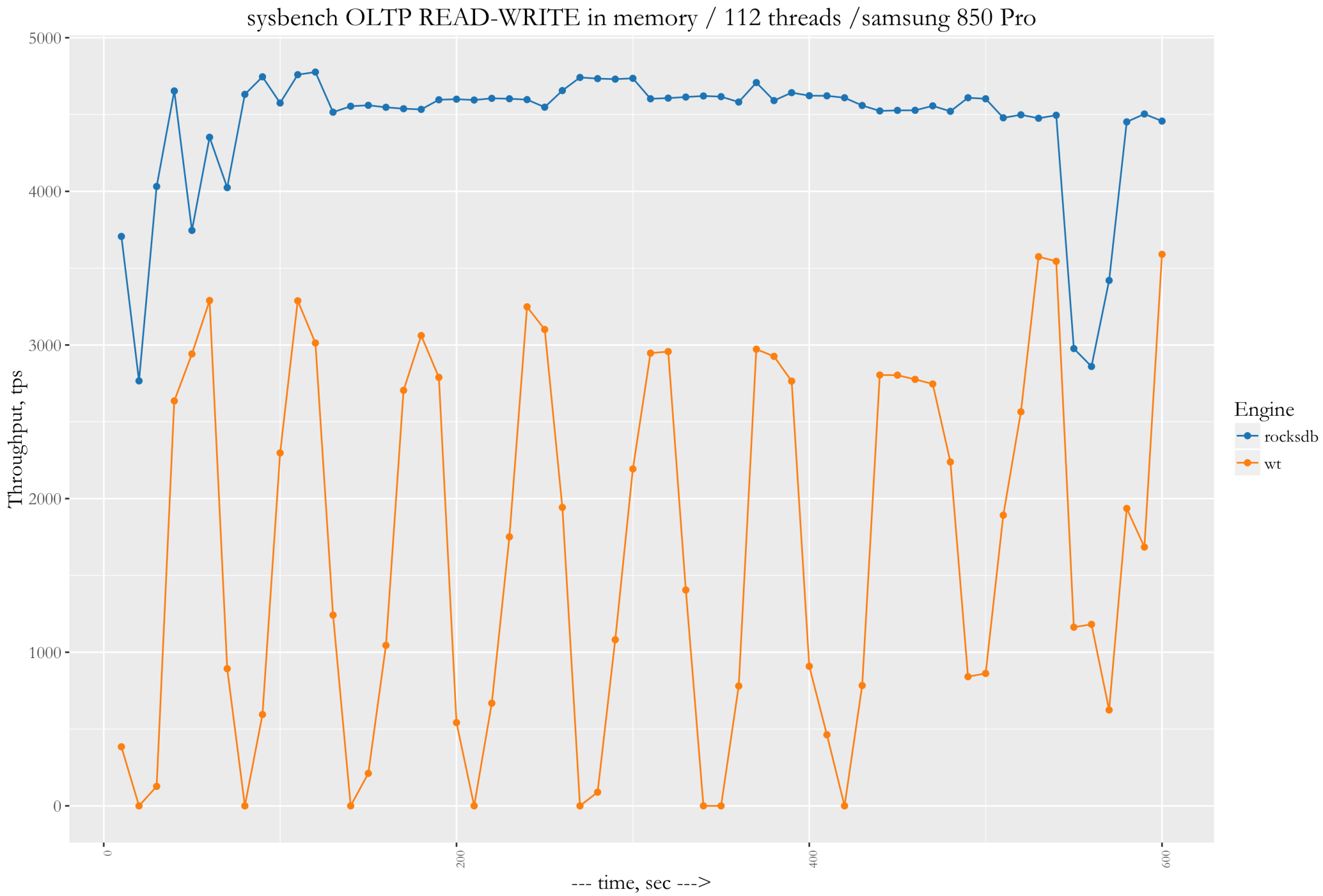
Threads scalability – in-memory workload

Cachesize 200GB

PERCONA


[network] sysbench OLTP READ-WRITE in memory / samsung 850 Pro
Samsung 850
Consumer grade SATA
Throughput, tps
6000
4000
2000
0
Threads
1
2
3
4
5
6
8
10
13
16
20
25
31
38
46
56
68
82
100
112
145
175
210
250
300
360
430
520
630
750
870
1000
Engine
rocksdb
wt

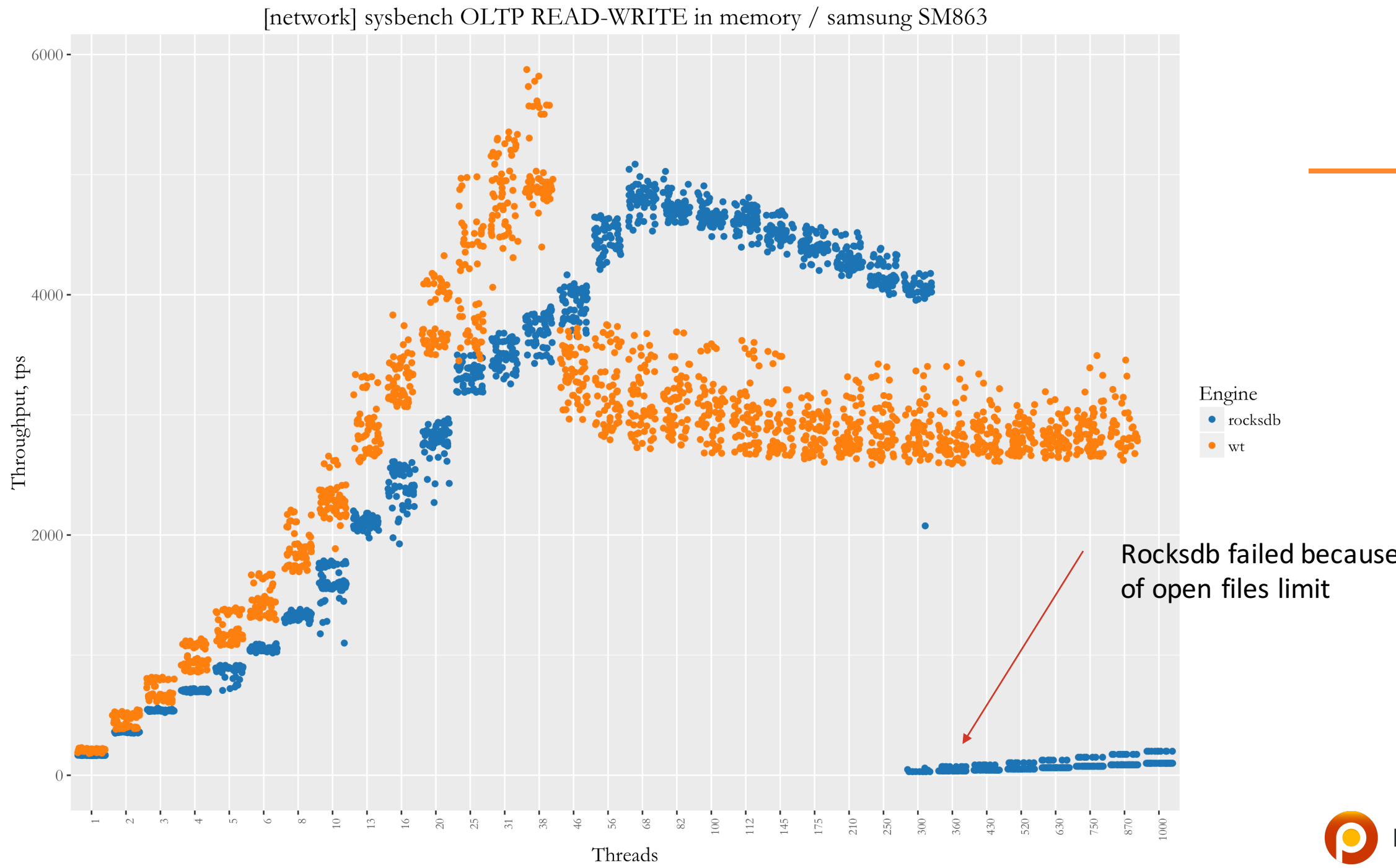
PERCONA


sysbench OLTP READ-WRITE in memory / 112 threads /samsung 850 Pro
Throughput, tps
5000
4000
3000
2000
1000
0
0
200
400
600
--- time, sec --->
Engine
rocksdb
wt

PERCONA

Let’s talk about checkpointing

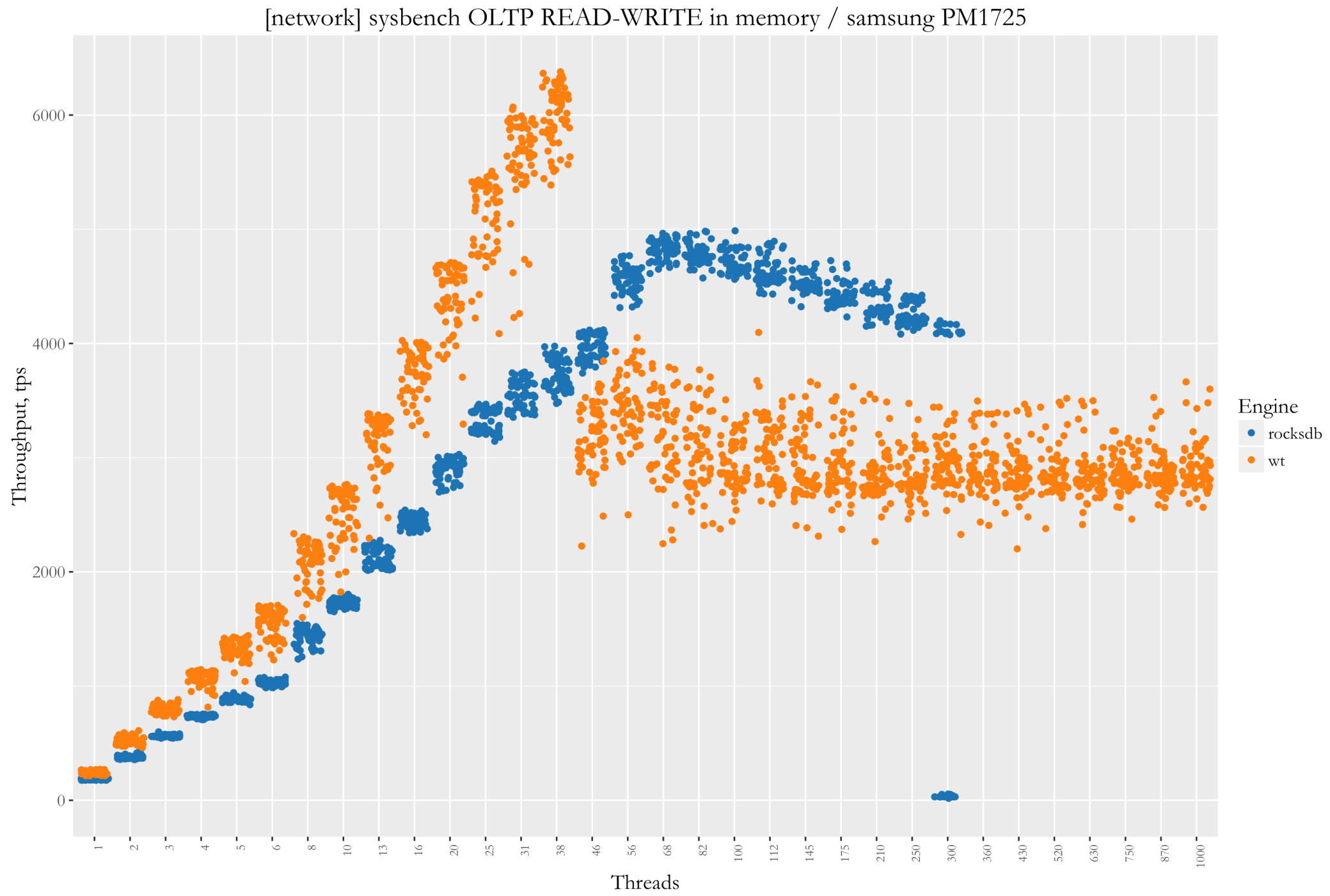
- Writes buffered in memory (not immediately landed on storage)
- To deal with crashes:
  - Write-Ahead-Log (WAL)
  - Redo-logs
  - Journals
- To limit size of log files in WiredTiger:
  - Time-based checkpointing (60 sec default)
  - syncdelay option
  - Or 2GB of log files (journals), whatever happens sooner
    - I am not sure if 2GB if configurable


[network] sysbench OLTP READ-WRITE in memory / samsung SM863
Throughput, tps
6000
4000
2000
0
Threads
1
2
3
4
5
6
8
10
13
16
20
25
31
38
46
56
68
82
100
112
145
175
210
250
300
360
430
520
630
750
870
1000
Engine
rocksdb
wt
Rocksdb failed because
of open files limit

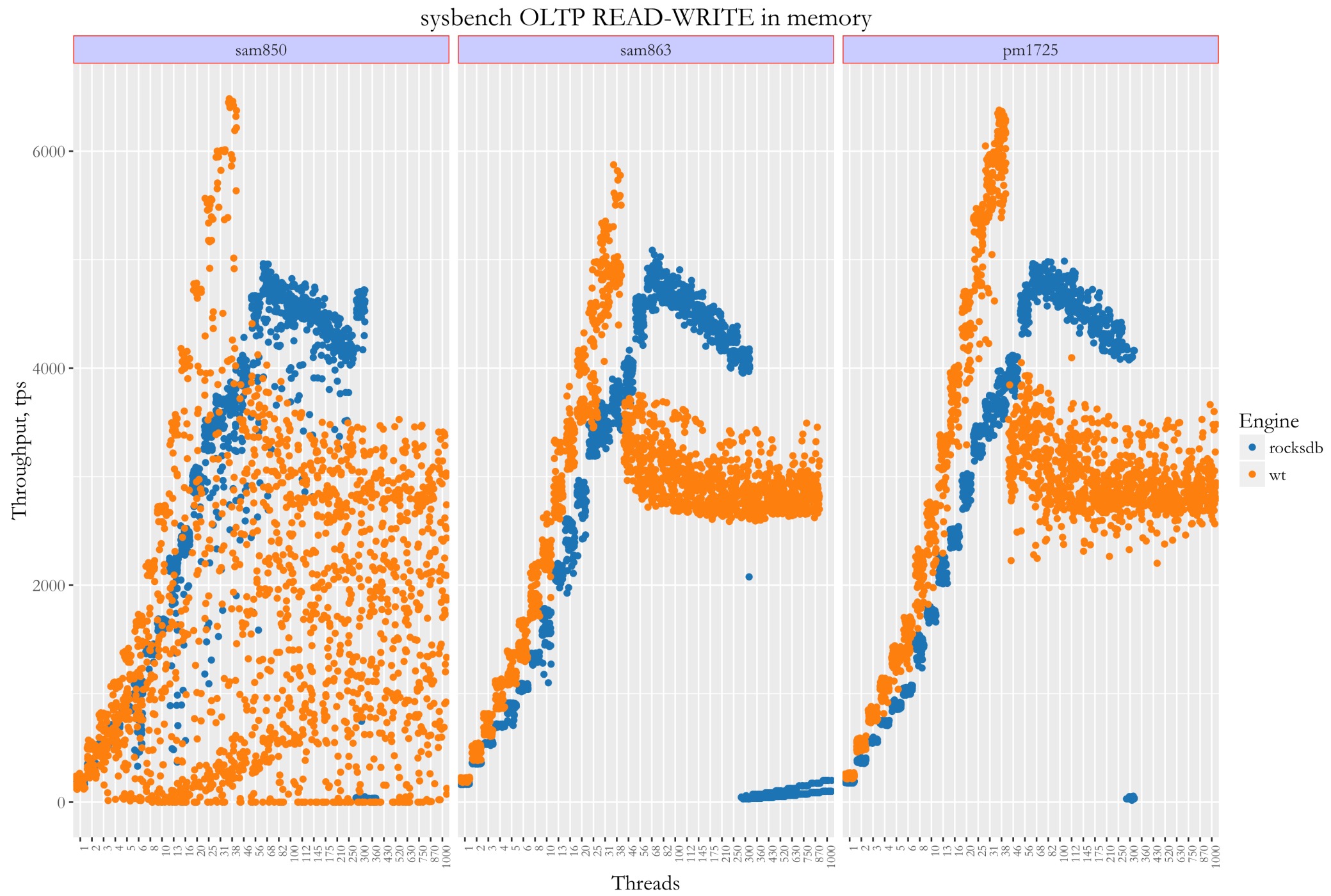
PERCONA

Let’s talk about open files in RocksDB

- RocksDB stores data in chunk files, fixed sizes
- It can result in large number of open files at give point of time
  - Be sure to increase the limit of open files `ulimit -n`
  - Not quite predictable how much will be required


[network] sysbench OLTP READ-WRITE in memory / samsung PM1725
Throughput, tps
0
2000
4000
6000
Threads
1
2
3
4
5
6
8
10
13
16
20
25
31
38
46
56
68
82
100
112
145
175
210
250
300
360
430
520
630
750
870
1000
Engine
rocksdb
wt

PERCONA


sysbench OLTP READ-WRITE in memory
sam850
sam863
pm1725
Throughput, tps
0
2000
4000
6000
Threads
1 2 3 4 5 6 8 10 13 16 20 25 31 38 46 56 68 82 100 112 145 175 210 250 300 360 430 520 630 750 870 1000
Engine
rocksdb
wt

PERCONA

Conclusions

- When data fits into memory:
  - RocksDB works well even on cheap storage
  - WiredTiger still requires storage that handles big volume writes well

Read-Write

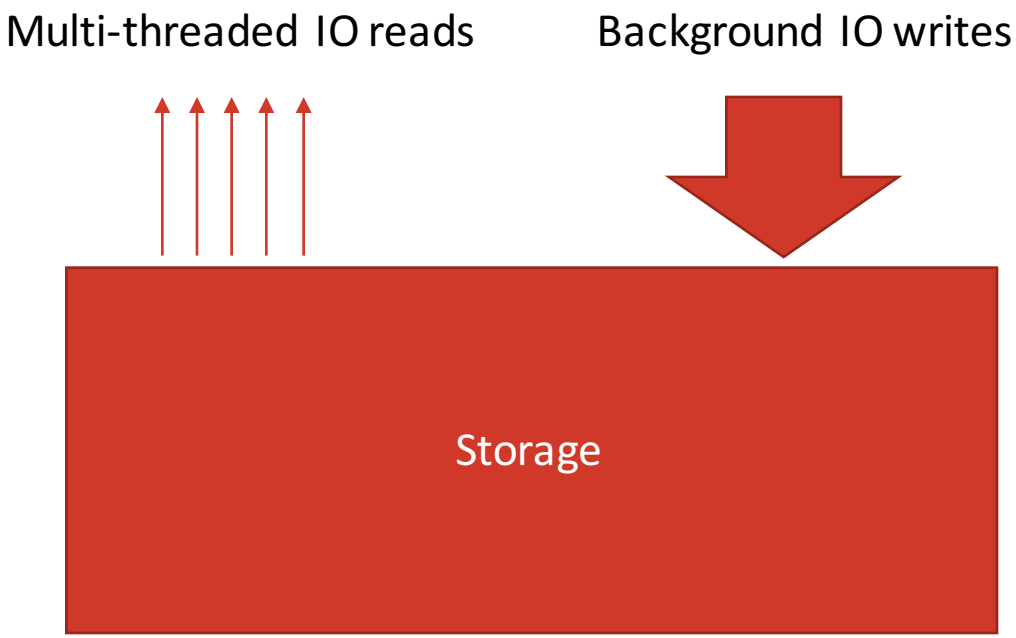
Threads scalability – IO-bound workload

Cachesize **20GB**

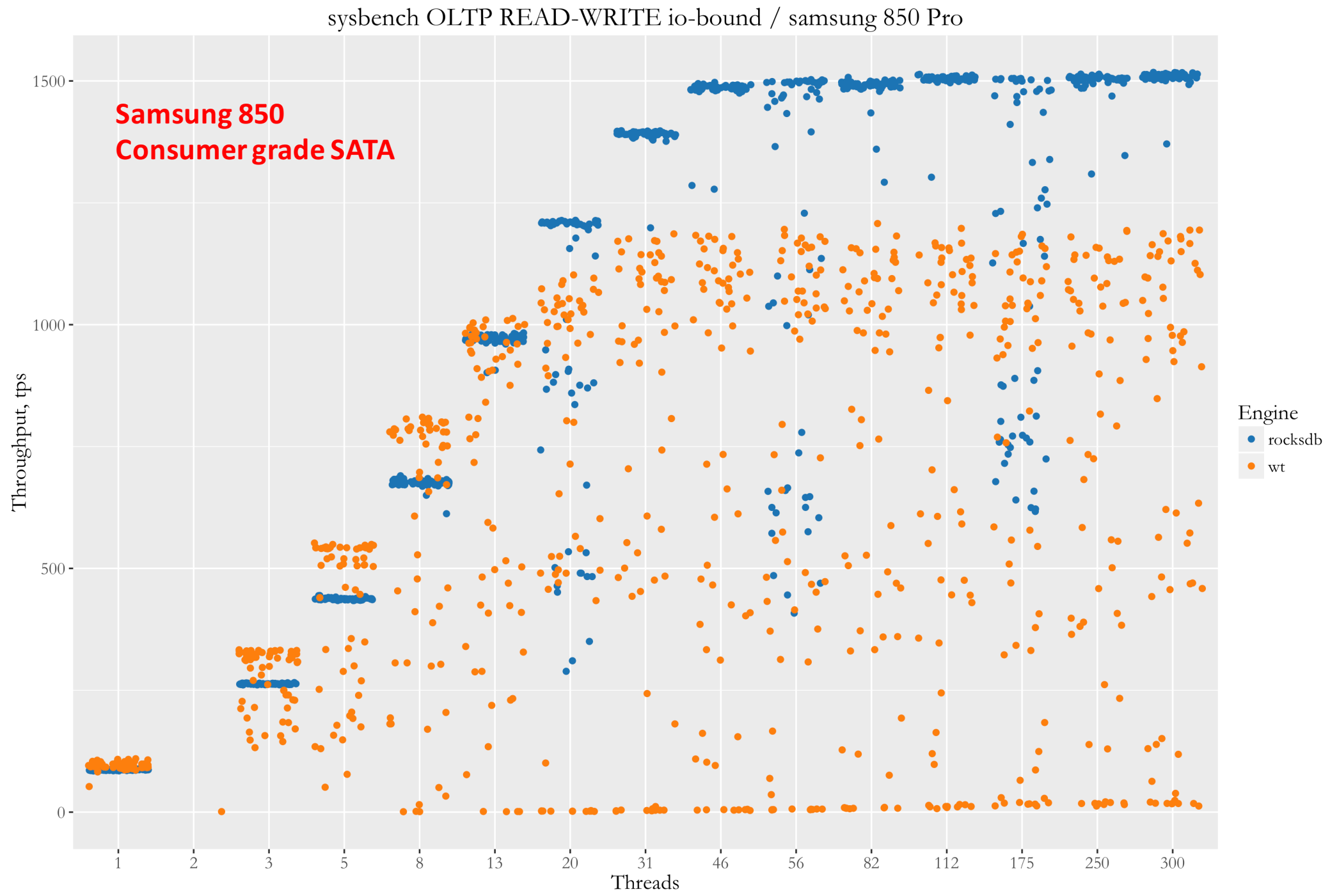
PERCONA

What to expect from this workload

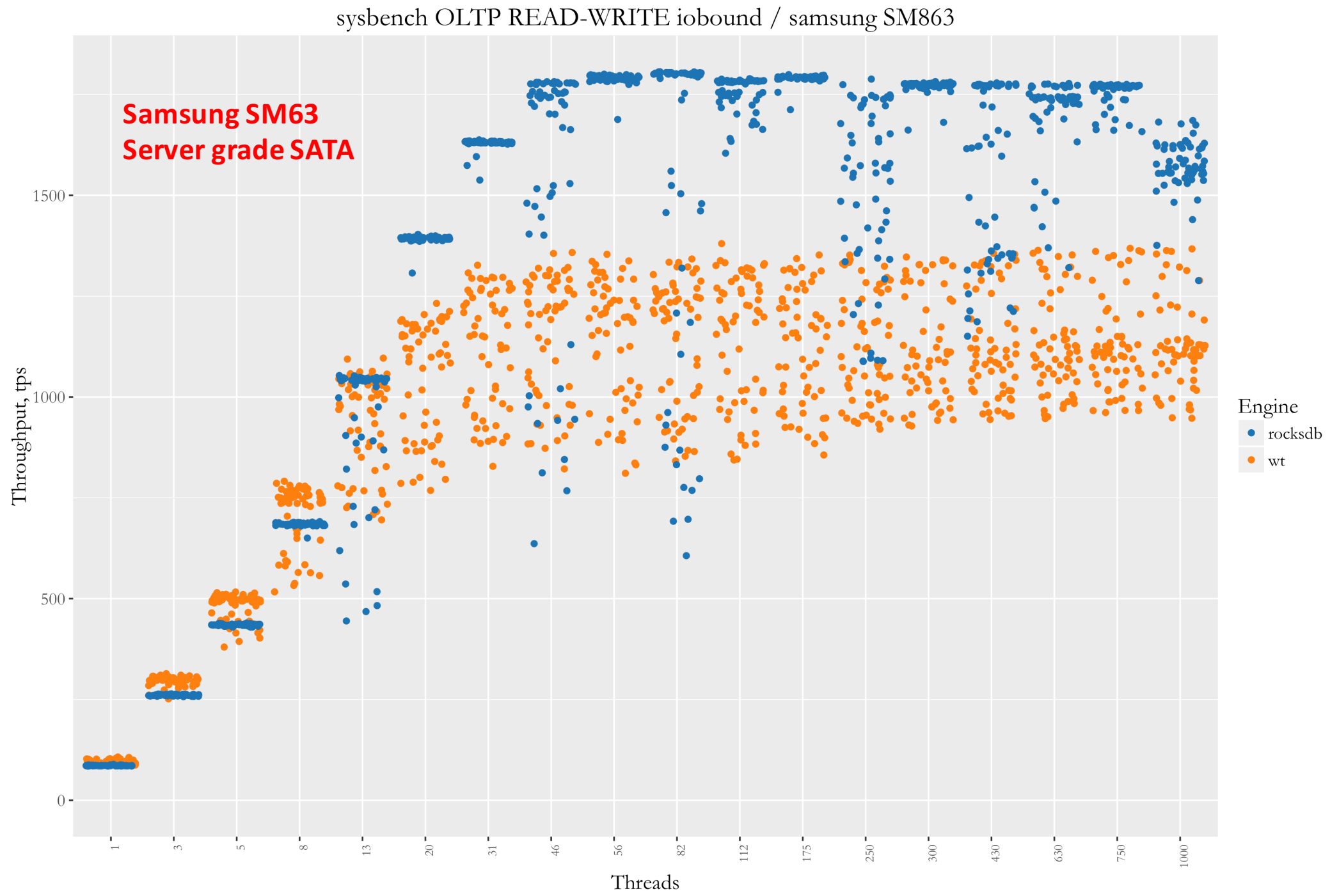
- Intensive background writes
- Response time to user query depends on IO read response time



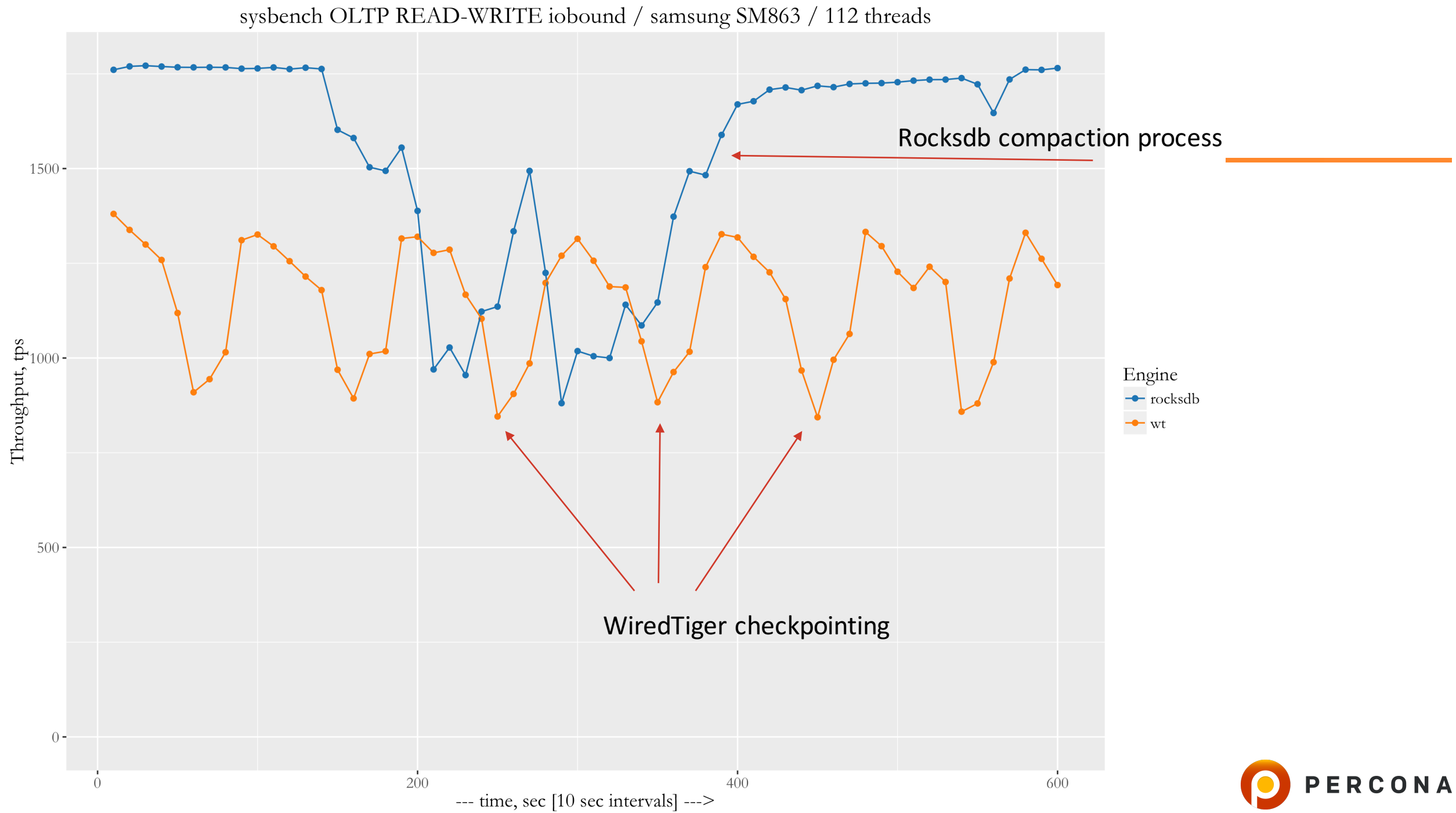
PERCONA


sysbench OLTP READ-WRITE io-bound / samsung 850 Pro
Samsung 850
Consumer grade SATA
Throughput, tps
1500
1000
500
0
1
2
3
5
8
13
20
31
46
56
82
112
175
250
300
Threads
Engine
rocksdb
wt

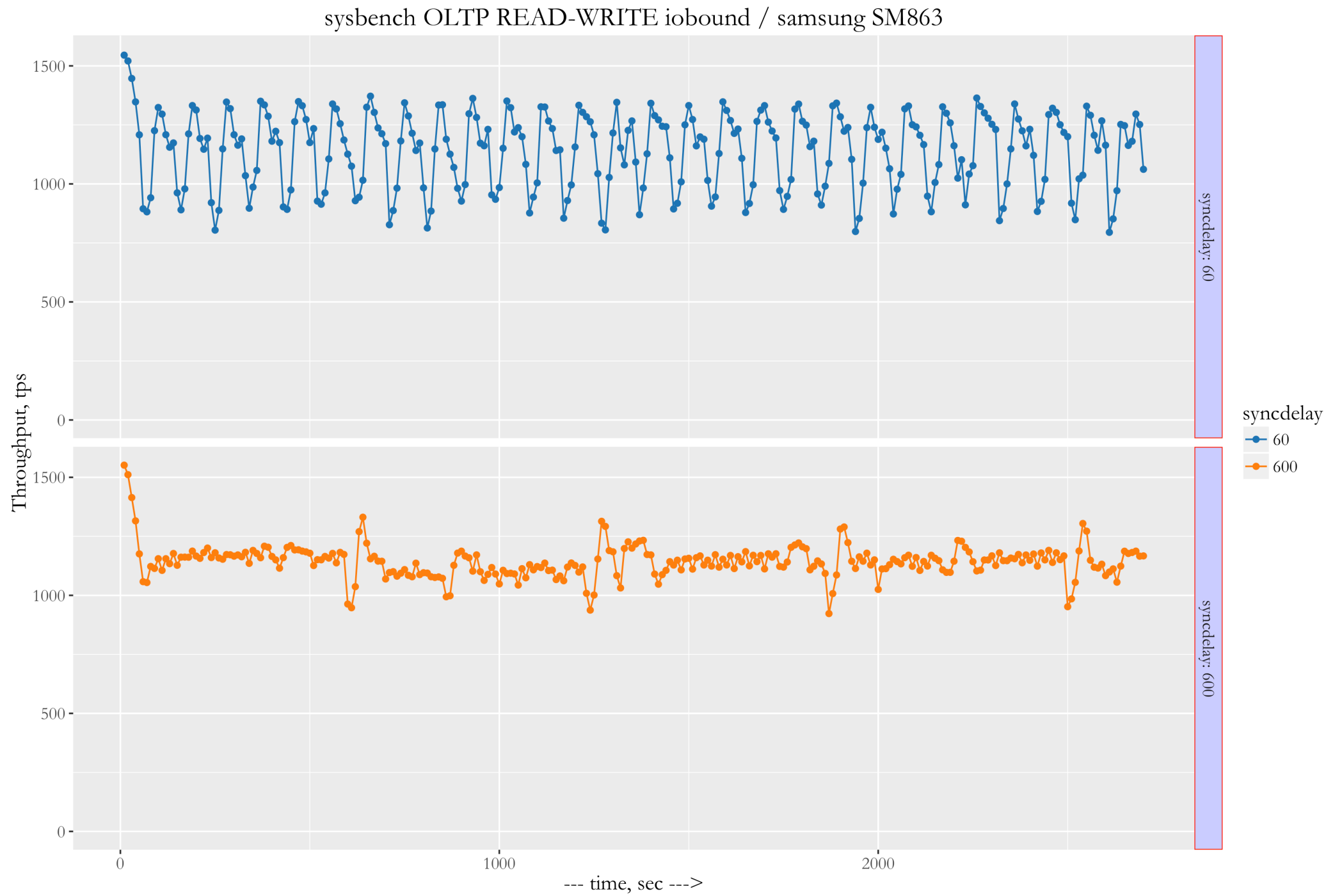
PERCONA


sysbench OLTP READ-WRITE iobound / samsung SM863
Samsung SM63
Server grade SATA
Throughput, tps
0
500
1000
1500
Threads
1
3
5
8
13
20
31
46
56
82
112
175
250
300
430
630
750
1000
Engine
rocksdb
wt

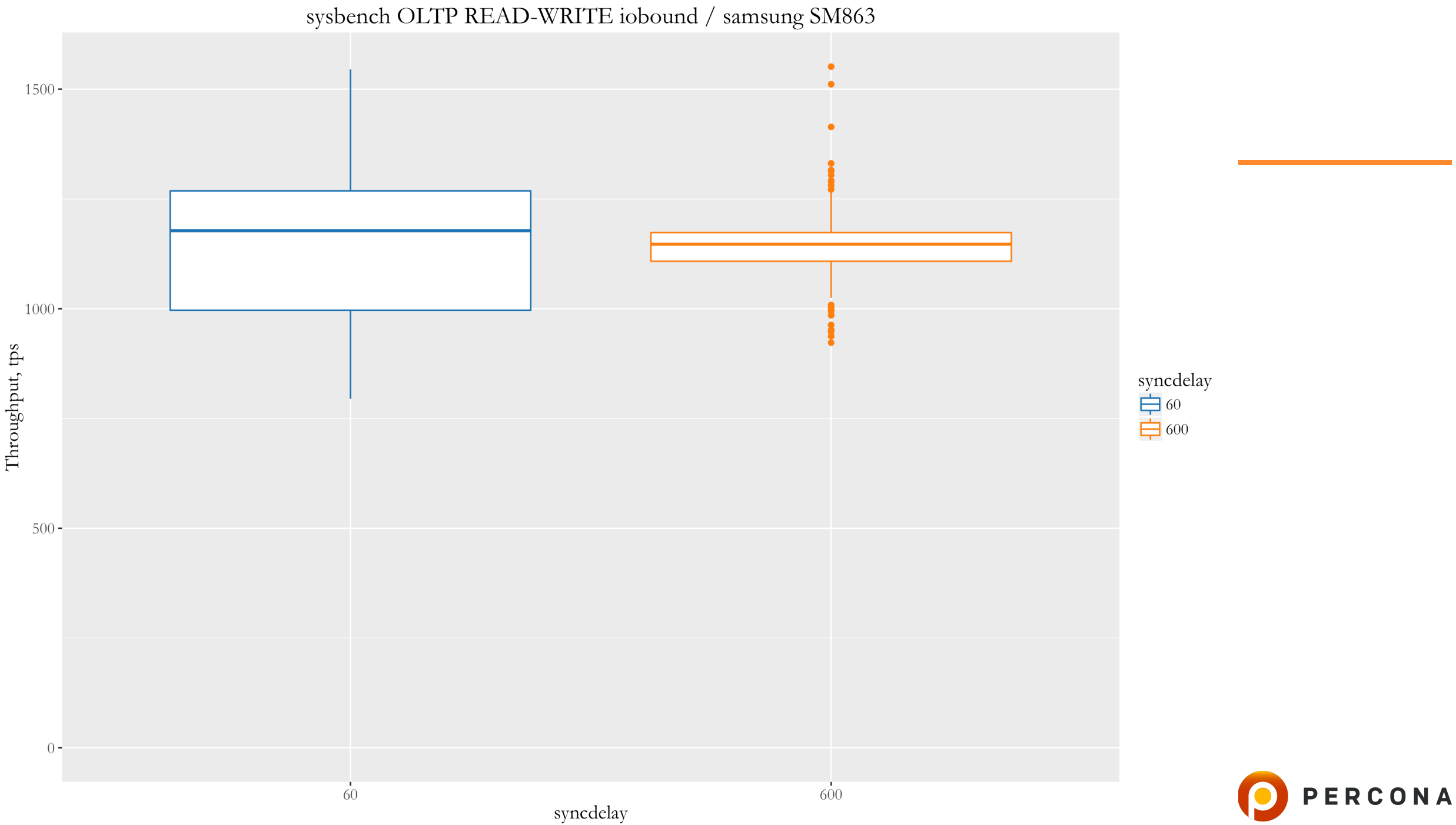
PERCONA


sysbench OLTP READ-WRITE iobound / samsung SM863 / 112 threads
Rocksdb compaction process
WiredTiger checkpointing
Throughput, tps
1500
1000
500
0
0
200
400
600
--- time, sec [10 sec intervals] --->
Engine
rocksdb
wt

PERCONA


sysbench OLTP READ-WRITE iobound / samsung SM863
Throughput, tps
1500
1000
500
0
syncdelay: 60
syncdelay: 600
0
1000
2000
--- time, sec --->
syncdelay
60
600

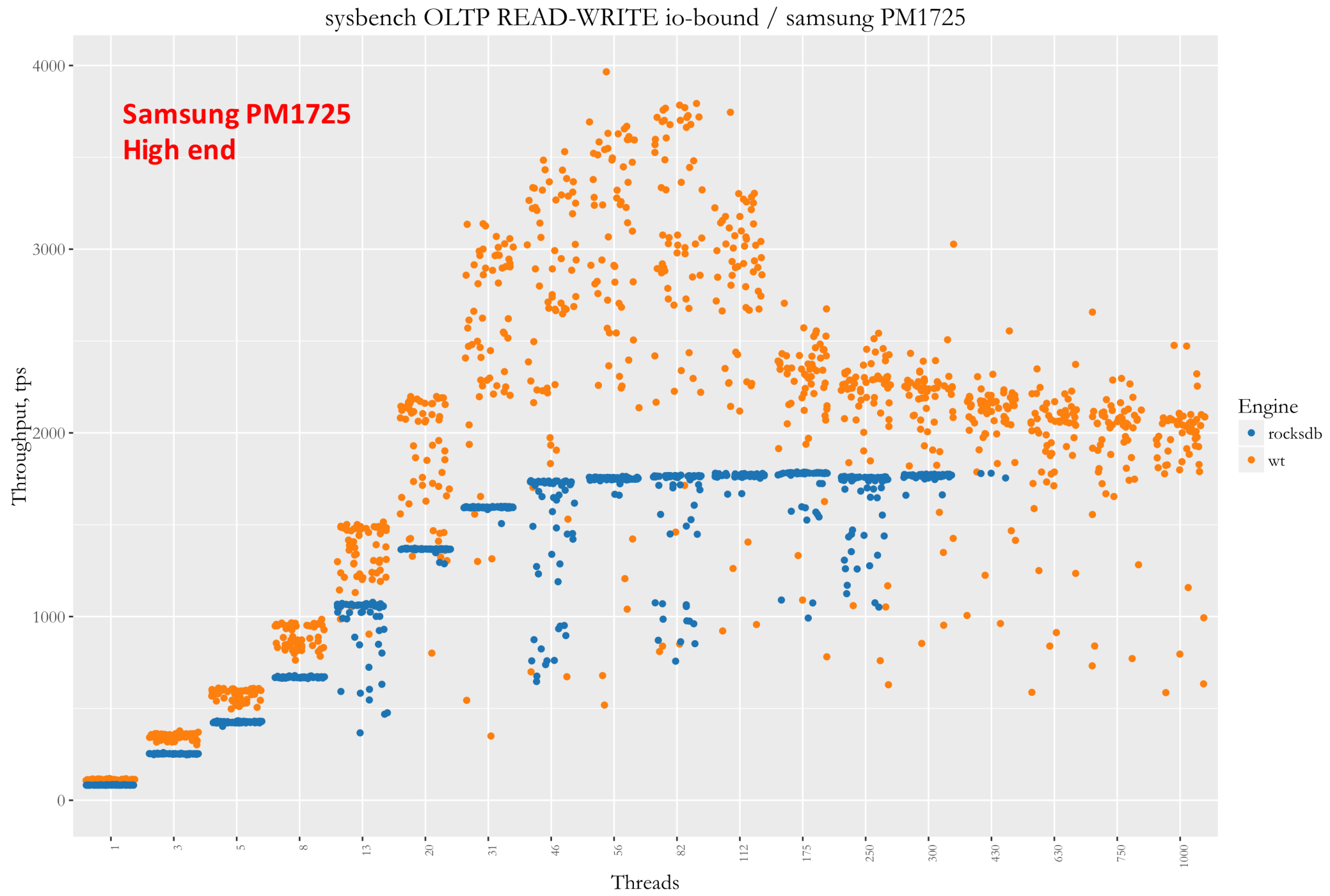
PERCONA


sysbench OLTP READ-WRITE iobound / samsung SM863
Throughput, tps
1500
1000
500
0
60
600
syncdelay
syncdelay
60
600

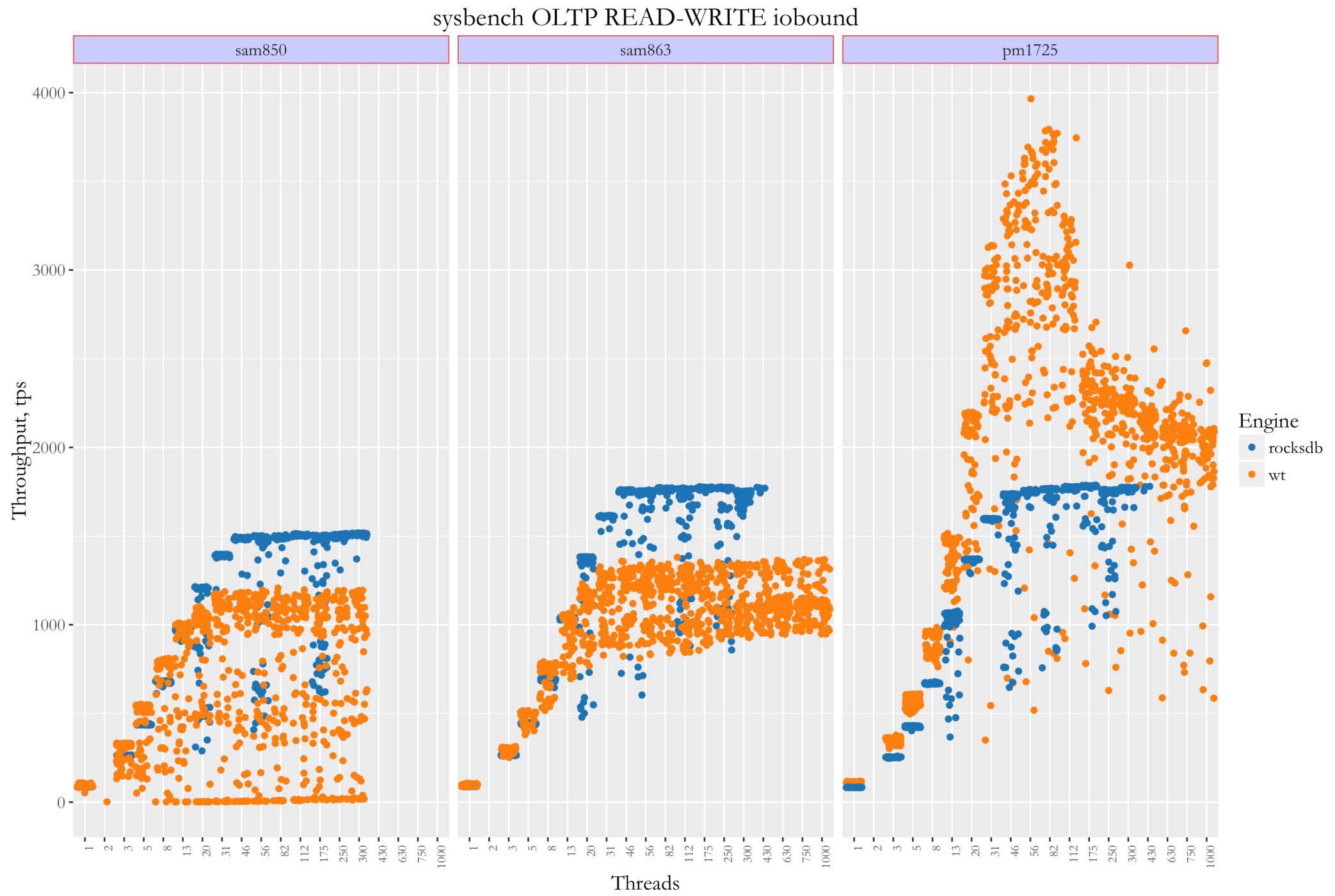
PERCONA

Crash Recovery time

- Syncdelay=60
  - To start mongod after a crash: ~25 mins
- Syncdelay=600
  - ~40 mins


sysbench OLTP READ-WRITE io-bound / samsung PM1725
Samsung PM1725
High end
Throughput, tps
4000
3000
2000
1000
0
Threads
1
3
5
8
13
20
31
46
56
82
112
175
250
300
430
630
750
1000
Engine
rocksdb
wt

PERCONA


sysbench OLTP READ-WRITE iobound
sam850
sam863
pm1725
Throughput, tps
4000
3000
2000
1000
0
1
2
3
5
8
13
20
31
46
56
82
112
175
250
300
430
630
750
1000
Threads
Engine
rocksdb
wt

PERCONA

Conclusions

- RocksDB works well with cheap storage
- WiredTiger benefits from faster storage
- To improve WiredTiger performance:
  - Typical recommendations for B-Tree based engines
    - More memory if data does not fit into memory
    - Storage with better response times
  - Playing with checkpoint intervals may help with variability, but keep in mind crash recovery times
- Send questions/feedback: @VadimTk, vadim@percona.com