LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

Qserv Fall 16 Large Scale Tests/KPMs

Vaikunth Thukral

DMTR-16

Latest Revision: 2017-05-09

Abstract

This document contains results of the large scale tests at 20% DR1 capacity that were run on CC-IN2P3 cluster during February-March 2017.
For comparisons, the previous large scale test can be found in DMTR-13.

Contents

Qserv Fall 16 Large Scale Tests/KPMs

1 Dataset Information

Table	Row Count	.MYD size [TB]	.MYI size [TB]
Object	3,777,968,880	4.6	0.1
Source	69,744,689,072	33.6	3.6
ForcedSource	344,034,271,380	11.0	8.7

Total MySQL data dir size: 62.4 TB

DR1 numbers are available in Document-16168 under “Data Releases”

Object and Source are now at ~20% of DR1 level and ForcedSource has more than 20% of DR1.

2 Hardware

- 50 nodes:
  - DELL PowerEdge R620 (Dell Spec Sheet) for nodes 1-25
  - DELL PowerEdge R630 (Dell Spec Sheet) for nodes 26-50
- 2 x Processors Intel Xeon E5-2603v2 @ 1.80 Ghz 4 core
- 10 MB cache, 6.4 GT/s, 80W
- Memory 16 GB DDR-3 @ 1600MHz (2x8GB)
- 2 x hard drive 250GB SATA 7200 Rpm 2,5” - hotplug => OS
- 8 x hard drive 1 TB Nearline SAS 6 Gbps 7200 Rpm 2,5”
- hotplug => DATA
- 1 x card RAID H710p with 1 GB nvram

- 1 x card1 GbE 4 ports Broadcom® 5720 Base-T
- 1 x card iDRAC 7 Enterprise

3 Testing methodology and summary

The cluster is divided into two sets of 25 nodes each, and each sub-cluster has the 20% DR1 dataset loaded in. These KPM tests were performed on the second half, on nodes `ccqserv125 - ccqserv149` (DR1 is expected to be on 92 nodes)

Queries are selected from a query pool that is divided into "types" representing different usage patterns as well as expected support for baseline requirements. Threads in the count of number of simultaneous queries to be tested are spawned, each managing an open connection to the qserv proxy and repeatedly picking random queries for its designated pool while sleeping when required.

The actual program that we used to drive the testing can be found at: runQueries.py

3.1 Short queries

- Single object selection by ID: 0.18 sec

```
SELECT * FROM Object WHERE deepSourceId = 16968353272299750
```

- Small spatial area selection from Object: 0.52 sec

```
SELECT COUNT(*) FROM Object WHERE qserv_areaspec_box(42.247928,
    38.874077, 42.273855, 39.034748)
```

3.2 Full table scans, single query at a time

- Object: ~7 min

```
SELECT ra, decl, u_psfFlux, g_psfFlux, r_psfFlux FROM Object
    WHERE y_shapeIxx BETWEEN 20 AND 20.2
```

- Source: ~41 min

```
SELECT COUNT(*) FROM Source WHERE flux\_sinc BETWEEN 1 AND 1.1
```

- ForcedSource: ~26 min

```
SELECT COUNT(*) FROM ForcedSource WHERE psfFlux BETWEEN 0.1 AND 0.2
```

3.3 Full table joins, single query at a time

- Object x Source: ~56 min (See section 4 for details)

```
SELECT o.deepSourceId, s.objectId, s.id, o.ra, o.decl
    FROM Object o, Source s WHERE o.deepSourceId=s.objectId
    AND s.flux_sinc BETWEEN 0.3 AND 0.31
```

- Object x ForcedSource: ~38 min

```
SELECT o.deepSourceId, f.psfFlux FROM Object o, ForcedSource f
    WHERE o.deepSourceId=f.deepSourceId
    AND f.psfFlux BETWEEN 0.13 AND 0.14
```

3.4 Concurrent scans and stress test

- 2 Object scans: ~8 min
- 5 Object scans: ~8 min (this shows that shared scans implemented in W16 work as intended)
- Able to run up to 35 (30 on Object, 3 on Source and 2 on ForcedSource) scans together — Object scans < 1h and Source scans < 1.5h
- Attempt to run 50 scans causes the proxy to fail, solution is targeted in multi-czar deployment in the future

3.5 *Typical load test 60 LV + 10 HV (+10% if possible)*

- 60 low volume and 10 high volume queries (4 scans for Object, 1 scan for Source, 1 scan for ForcedSource, 2 Object-Source joins, 1 Object-ForcedSource join and 1 NearNeighbor

query), all running simultaneously with appropriate sleep in between queries to enforce the mix we are aiming for. We also introduce a new metric to quantify system performance, **Q**uery **T**hrough**P**ut, or **QTP**

- During 24 hours we completed:
  - 571,029 Low Volume queries finished — Baseline: ~10 sec per query, 432,000 queries in 24h
    * Achieved Low Volume QTP: **396 queries/minute**
  - 108 Object scans — Baseline: ~1h per query, or 96 in 24h
  - 2 Source scans — Baseline: ~12h per query, or 2 in 24h
  - 2 ForcedSource scans — Baseline: ~12h per query, or 2 in 24h
  - 4 Object-Source joins — Baseline: ~12h per query or 4 in 24h
  - 2 Object-ForcedSource joins — Baseline: ~12h per query or 2 in 24h
  - 27 NearNeighbor queries — Baseline: ~1h per query, or 24 in 24h
    * Achieved High Volume QTP: **6 queries/hour**
- Average output:
  - Overall size of Low Volume results was ~70GB: **128 kB/query**
  - Overall size of High Volume results was ~7GB: **48 MB/query**
- Average times:
  - Low Volume queries **3.23 sec/query** — Baseline: should be under 10 sec. See Fig. 1.
  - Object scans **15.15 min/query** — Baseline: should be under 1 hour
  - Source scans **2.7 hr/query** — Baseline: should be under 12 hours
  - ForcedSource scans **2.7 hr/query**— Baseline: should be under 12 hours
  - Object-Source joins **2.7 hr/query** — Baseline: should be under 12 hours
  - Object-ForcedSource joins **2.7 hr/query** — Baseline: should be under 12 hours
  - NearNeighbor queries **16.6 min/query** — Baseline: should be under 12 hours. See Fig. 2.
- Observations:

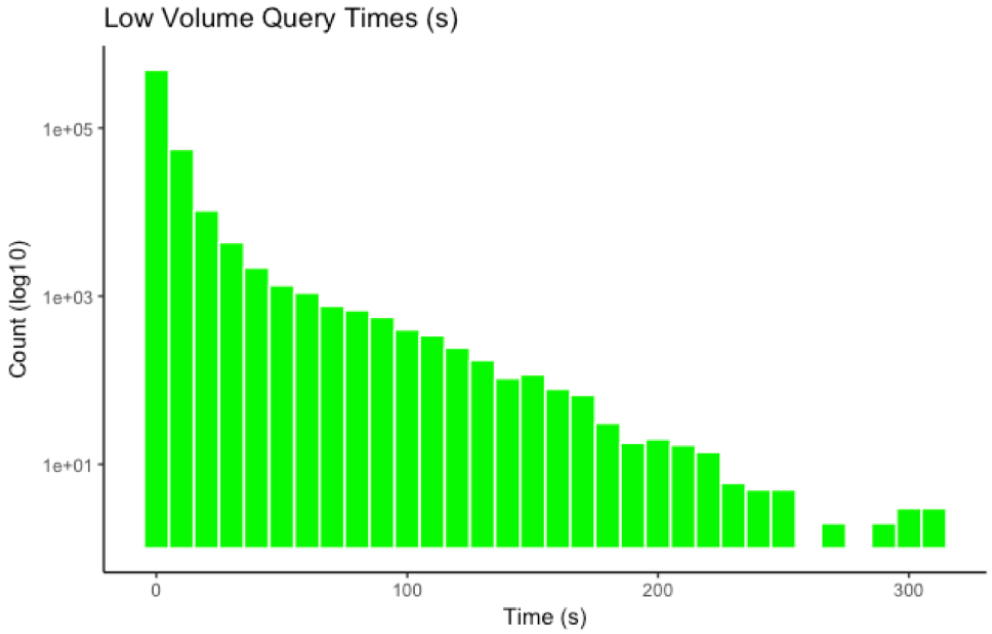

FIGURE 1: Histogram for all Low Volume query times, note the log scale on the Y axis.

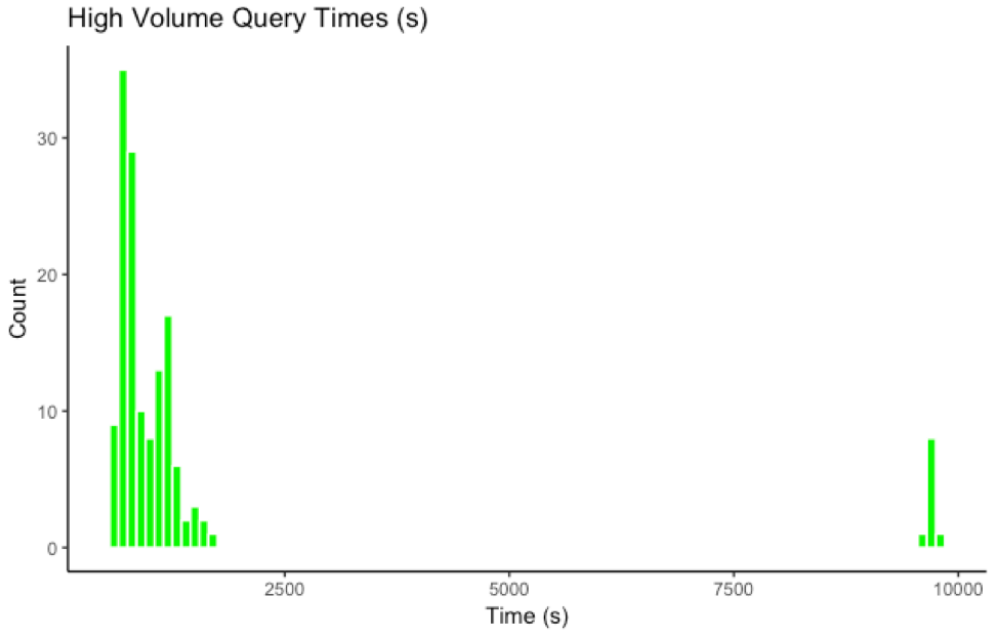

FIGURE 2: Histogram for all High Volume query times, note the clear difference in Object and Source scans.

  - Tests were able to run exactly 10% faster than the attempted mix for Low Volume queries and Object scans (due to the short time frame of these queries)
  - Scan scheduling implemented in W16 works with scans on each table type running in approximately the same amount of time
    * For correct scan scheduling and query planning, mysql statistics have to be generated. Future work for this is planned
    * Issues were uncovered with returning large result sizes. Future testing will take this into account
    * Due to order-of-magnitude increase in output size, average times for Low Volume queries has increased due to tests running from one submit node
    * For these issues, see DM-9757, DM-10360, and DM-10366.

3.6 *Heavy load test 100 LV + 20 HV (+10% if possible)*

- 100 low volume and 20 high volume queries (8 scans for Object, 2 scans for Source, 2 scans for ForcedSource, 4 Object-Source joins, 2 Object-ForcedSource join and 2 NearNeighbor queries), all running simultaneously with appropriate sleep in between queries to enforce the mix we are aiming for
- During 24 hours we completed:
  - 814,989 Low Volume queries
  - 129 Object scans
  - 4 Source scans
  - 4 ForcedSource scans
  - 8 Object-Source joins
  - 2 Object-ForcedSource joins
  - 54 NearNeighbor queries
- Average Output:
  - Overall size of Low Volume results was ~130GB: **160 kB/query**
  - Overall size of High Volume results was ~8GB: **38 MB/query**
- Average times:

  - Low Volume queries **9.98 sec/query**
  - Object scans **24 min/query**
  - Source scans **4.44 hr/query**
  - ForcedSource scans **4.44 hr/query**
  - Object-Source joins **4.44 hr/query**
  - Object-ForcedSource joins **4,44 hr/query**
  - NearNeighbor queries **24 min/query**

- Observations:
  - Did not finish expected number of Object scans
  - Average size of High Volume query results lowered due to fewer completed Object scans
  - Low Volume queries still finished under the baseline of 10 seconds

4 Notes and Observations

- *Concurrency maintained:* With double the data from S15 we successfully ran simultaneous queries up to and more than the baseline requirements for 20% DR1
- *Robustness improved:* Although the tests were run for 24 hours in S15 too, these test runs have been performed multiple times and continuously over weeks without any Qserv services failing or needing restarts
- *Shared Scan implementation:* Qserv is able to handle successively larger number of scans on the same table and perform them for a similar cost/time.

5 Query Templates Used

The actual query pool used in these tests is available from the `git` repository associated with this document. A sample of the query types used in each category is listed below.

- Trivial query that retrieves one row, using index

```
SELECT * FROM Object WHERE objectId = <objId>
```

- Counts

```
SELECT COUNT( * ) FROM Object

SELECT COUNT( * ) FROM Source

SELECT COUNT( * ) FROM ForcedSource
```

- Spatially restricted query, small area of sky, should return small number of rows (say <100)

```
SELECT COUNT( * )
FROM Object
WHERE ra_PS BETWEEN 1 AND 2
AND decl_PS BETWEEN 3 AND 4
```

- Full table scan, use some column in WHERE that is not indexes, make sure the number of results returned is sane (eg thousands, not millions)

```
SELECT objectId, ra_PS, dec\_PS, <few other columns>
FROM Object
WHERE fluxToAbMag(iFlux_PS) - fluxToAbMag(zFlux_PS) > 4
```

- Aggregation

```
SELECT COUNT(*) AS n,
AVG(ra_PS),
AVG(decl_PS), chunkId
FROM Object
GROUP BY chunkId
```

- Near neighbor

```
SELECT COUNT(*)
FROM Object o1, Object o2
WHERE qserv_areaspec_box(-5,-5,5,-5)
AND scisql_angSep(o1.ra_PS, o1.decl_PS, o2.ra_PS, o2.decl_PS) < 0.1
```

- Joins

```
SELECT o.objectId, s.sourceId, ra_PS, decl_PS, <few other columns>
FROM Object
JOIN SOURCE USING (objectId)
WHERE fluxToAbMag(iFlux_PS) - fluxToAbMag(zFlux_PS) > 4
AND <some restriction from source table>
```

References

[1] **[DMTR-13]**, Becla, J., 2015, *Qserv Summer 15 Large Scale Tests*, DMTR-13, URL https://ls.st/DMTR-13

[2] **[Document-16168]**, LSST Systems Engineering, 2014, *LSST Key System Parameters Summary*, Document-16168, URL https://ls.st/Document-16168