S15 Large Scale Tests

This page captures information about the large scale tests we ran on the IN2P3 cluster during Summer 2015.

For comparisons, here is a link to the previous large scale test that we run: https://dev.lsstcorp.org/trac/wiki/db/Qserv/in2p3_300

Data Set

table	row count	.MYD size [TB]	.MYI size [TB]
Object	1,889,695,615	2.45	0.06
Source	34,886,017,763	17.13	2.05
ForcedSource	172,081,115,270`	5.85	4.61

Total MySQL data dir size: 33.2 TB
So for Object and Source we are at the ~10% of DR1 level. We have more Forced Sources than 10% of DR1.

Hardware

- 50 nodes, DELL PowerEdge R620
- 2 x Processors Intel Xeon E5-2603v2 @ 1.80 Ghz 4 core
- 10 Mo 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

Timing summary

Data on 24 nodes (for comparison, DR1 is expected to be on 92 nodes).

Short queries

- single object selection by id: 0.09 sec
- small spatial area selection from Object: 0.33 sec

Full table scans, single query at a time

- Object ~4 min
- Source ~18 min
- ForcedSource ~15 min

Full table joins, single query at a time

- Object x Source: ~23 min
- Object x ForcedSource: ~ 21 min

Concurrent scans

- 2 Object scans ~ 8 min, 5 Object scans ~16-20 min (this shows that our shared scan code has issues, scheduled to be solved in W16)

Heavy load test 50 LV + 5 HV

- 50 low volume and 5 high volume queries (3 scans for Object, 1 scan for Source, 1 Object-Source joins), all running simultaneously with appropriate sleep in between queries to enforce the mix we were aiming
- During 24 hours we completed:
  - 431,597 low volume queries (consistent with the baseline: ~10 sec per query, or 432,000 queries in 24h)
  - 73 Object scans (consistent with baseline: ~1h per query, or 72 in 24h)
  - 3 Source scans (consistent with baseline: ~8h per query, or 3 in 24h)
  - 3 Object-Source joins (consistent with the baseline ~8h per query, or 3 in 24h)
  - overall size of results was 6.5 GB (~16 KB per query on average)
- Average times:
  - low volume queries: 0.91 sec (per baseline, should be under 10 sec)
  - Object scan: 15 min (per baseline, should be under 1 hour)
  - Source scan: 56 min (per baseline, should be under 8 hours)
  - Object-Source join 57 min (per baseline, should be under 8 hours)
- Observations:
  - io bound during the time when scans happen at the same time. Disks 85-90% busy (~750 MB/sec seen)
  - the aggregate load on the cluster: https://confluence.lsstcorp.org/download/attachments/35815659/in2p3_5HV50LV.png?api=v2

Heavy load test 100 LV + 10 HV

- 100 low volume and 10 high volume queries (6 scans for Object, 2 scan for Source, 2 Object-Source joins), all running simultaneously with appropriate sleep in between queries to enforce the mix we were aiming
- During 24 hours we completed:
  - 861,608 low volume queries
  - 144 Object scans
  - 6 Source scans
  - 8 Object-Source joins
- Average times:
  - low volume queries: 5.1 sec
  - Object scan: 22 min
  - Source scan: 1 h 33 min
  - Object-Source join 1 h 22 min
- Observations:
  - the aggregate load on the cluster: https://confluence.lsstcorp.org/download/attachments/35815659/in2p3_10HV100LV.png?api=v2
- Notes: during the last part of that test we were reloading data on the node ccqserv101, which was impacted how the average load on the cluster looked like

Notes and Observations

- Concurrently greatly improved. This was the very first time we ever successfully ran more than 3-4 simultaneous queries (we run up to 110).
- Robustness greatly improved. This was the very first time we ran continuously without any failure for 24 hours (it could have ran for longer, we just limited it to 24h)
- Latency reduced 100x. In previous tests average time to complete low-volume query was above 1 sec. Some of the latest improvements involve reducing latency, in particular the overhead of query dispatch and sending back result data. The improvements are clearly visible, we were able to demonstrate 90 milisec response time for individual low-volume queries - 100x better than before.
- Work to do:
  - When we run 5 full scan queries, some low volume queries get stuck, and wait to be scheduled for a long time (minutes), this

needs to be optimized. But overall things balance out because full scan queries end before planned time and there is quite time, when low volume queries can catch up.
- Single-table scared scans are not working well. This will be fixed in W16
- Multi-node shared scans are not working. We did not implement it yet. The plan is to implement this in W16
- The tests revealed problem with handling large results on the master node: when a query involves multi-GB results, our master node will currently use excessive amount of memory and CPU. (Some of the tests we ran produced result sets up to 46 GB over the period of 30 hours). The uncovered issue will be addressed in FY16 (**DM-3495** - X16 Large Results **DONE**)

Sample Queries

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

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*
> *{quote}*

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, decl_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 qserv_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>*

Raw output from selected individual queries

Numbers for 24h scaling tests not shown due to size of the output

Counts

```
select count(*) from Object;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 1889695615 |
+----------------+
1 row in set (47.75 sec)
```

```
select count(*) from Source;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 34886017763 |
+----------------+
1 row in set (40.99 sec)
```

```
select count(*) from ForcedSource;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 172081115270 |
+----------------+
1 row in set (48.33 sec)
```

Short-running queries

```
SELECT ra, decl FROM Object WHERE deepSourceId = 3306154155315676;
+------------------+-------------------+
| ra | decl |
+------------------+-------------------+
| 346.444574155259 | -20.0756000206646 |
+------------------+-------------------+
1 row in set (0.09 sec)
```

```
SELECT ra, decl FROM Object WHERE qserv_areaspec_box(0.95, 19.171, 1.0, 19.175);
+-------------------+------------------+
| ra | decl |
+-------------------+------------------+
| 0.952155934104298 | 19.1739644910299 |
| 0.951022182881938 | 19.1744018550878 |
| 0.979879729932035 | 19.1721286203352 |
| 0.978531748948322 | 19.173622354719 |
| 0.975277403624571 | 19.1717082593989 |
| 0.965659553702501 | 19.1732402376328 |
| 0.960765770111898 | 19.1728325244272 |
| 0.956040810381224 | 19.1748876675009 |
| 0.954389385192787 | 19.1715837046997 |
| 0.970953770462485 | 19.1732960324755 |
| 0.988995842261423 | 19.172924537295 |
| 0.98748403175534 | 19.1744384618428 |
| 0.990599073289862 | 19.1748218268107 |
| 0.989373097950412 | 19.1741759125297 |
| 0.995062781391914 | 19.1726058129962 |
| 0.993584927322364 | 19.174694023095 |
| 0.994098536926311 | 19.171425377618 |
| 0.997942570312296 | 19.1749796823199 |
| 0.987602654004053 | 19.1743333663937 |
| 0.988982091888198 | 19.1729311723649 |
+-------------------+------------------+
20 rows in set (0.33 sec)
```

Full table scans

```
select count(*) from Object where y_instFlux > 5;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 0 |
+----------------+
1 row in set (4 min 7.61 sec)
```

```
select min(ra), max(ra), min(decl), max(decl) from Object;
+--------------+------------------+-------------------+------------------+
| MIN(QS1_MIN) | MAX(QS2_MAX) | MIN(QS3_MIN) | MAX(QS4_MAX) |
+--------------+------------------+-------------------+------------------+
| 0 | 359.999999921199 | -87.8823524031432 | 45.5294117096401 |
+--------------+------------------+-------------------+------------------+
1 row in set (4 min 4.24 sec)
```

```
select count(*) from Source where flux_sinc between 1 and 2;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 3539300 |
+----------------+
1 row in set (18 min 8.09 sec)
```

```
select count(*) from Source where flux_sinc between 2 and 3;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 3589961 |
+----------------+
1 row in set (17 min 57.38 sec)
```

```
select count(*) from ForcedSource where psfFlux between 0.1 and 0.2;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 67769638 |
+----------------+
1 row in set (14 min 58.61 sec)
```

Joins

```
select count(*) from Object o, Source s WHERE o.deepSourceId=s.objectId AND s.flux_sinc BETWEEN 0.13 AND
0.14;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 35179 |
+----------------+
1 row in set (23 min 1.44 sec)
```

```
select count(*) FROM Object o, ForcedSource f WHERE o.deepSourceId=f.deepSourceId AND f.psfFlux BETWEEN
0.13 AND 0.14;
+----------------+
| SUM(QS1_COUNT) |
+----------------+
| 6749369 |
+----------------+
1 row in set (21 min 31.38 sec)
```

Near neighbor

```
select count(*)
from Object o1, Object o2
where qserv_areaspec_box(90.299197, -66.468216, 98.762526, -56.412851) and scisql_angSep(o1.ra, o1.decl,
o2.ra, o2.decl) < 0.015;

+----------------+

| SUM(QS1_COUNT) |

+----------------+

|       96795152 |

+----------------+

1 row in set (11 min 16.02 sec)
```

Shared scans

Two scans on Object, both finished in ~8.5 min or so. Startup was staggered.

```
QTYPE_FTSObj: 505.703582048  SELECT COUNT(*) FROM Object WHERE y_instFlux > 5
QTYPE_FTSObj: 505.837508917  SELECT MIN(ra), MAX(ra) FROM Object WHERE decl > 3
```

Five scans on Object finished in 16-20 min. Startup was staggered.

```
QTYPE_FTSObj: 990.450098038 SELECT MIN(ra), MAX(ra) FROM Object WHERE decl > 3
QTYPE_FTSObj: 1168.69941115 SELECT MIN(ra), MAX(ra) FROM Object WHERE decl > 3
QTYPE_FTSObj: 1180.72830892 SELECT COUNT(*) FROM Object WHERE y_instFlux > u_instFlux
QTYPE_FTSObj: 1178.19018197 SELECT COUNT(*) FROM Object WHERE y_instFlux > 5
QTYPE_FTSObj: 1173.29835892 SELECT MIN(ra), MAX(ra) FROM Object WHERE z_apFlux BETWEEN 1 and 2
```

Five scans on Object, without staggering, not much difference:

```
QTYPE_FTSObj: 738.438729763 SELECT COUNT(*) FROM Object WHERE y_instFlux > 5
QTYPE_FTSObj: 1162.67162609 left 2437.32837391 SELECT MIN(ra), MAX(ra) FROM Object WHERE decl > 3
QTYPE_FTSObj: 1169.67710209 left 2430.32289791 SELECT COUNT(*) FROM Object WHERE y_instFlux > 5
QTYPE_FTSObj: 1171.61784506 left 2428.38215494 SELECT COUNT(*) FROM Object WHERE y_instFlux > 5
QTYPE_FTSObj: 1171.95623493 left 2428.04376507 SELECT COUNT(*) AS n, AVG(ra), AVG(decl), chunkId FROM Object GROUP BY chunkId
```

Five scans: four on Object, one on Source: ~1h10 min per scan

```
QTYPE_FTSObj: 4237.70917988 SELECT MIN(ra), MAX(ra) FROM Object WHERE decl > 3
QTYPE_FTSObj: 4262.98238802 SELECT COUNT(*) FROM Object WHERE y_instFlux > 5
QTYPE_FTSObj: 4263.39259911SELECT COUNT(*) FROM Object WHERE y_instFlux > 5
QTYPE_FTSObj: 4263.39338088 SELECT COUNT(*) FROM Object WHERE y_instFlux > 5
QTYPE_FTSSrc: 4264.03135395 SELECT COUNT(*) FROM Source WHERE flux_sinc BETWEEN 1 AND 2
```