DATA MODEL SCORECARD

## Keyspace: Assessment

## Database: workoutdb

8 of 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NUMBER | RESULT | MEASUREMENT | PASSING VALUE | SIGNIFICANCE | WHEN MEASURED  (TestRef or Screenshot) |
| 1 | PASS | Pending compactions with steady “read only” load | < 15 at all times | High number of pending compactions has a negative impact on read performance because of I/ O contention between SSTable and Compaction processing | Test #12 |
| 2 | PASS | Disk utilization with steady “writes only” load | < 40% most of the time. Never higher than 70% | We use writes only because reads primarily come from cache | Test #15 |
| 3 | PASS | Scalability  Steady state work load run against 3, 6 and 9 nodes | Throughput increases, but latency does not. | Although it is widely acknowledged that Cassandra scales in a linear fashion, this test provides concrete proof that the current project follows this pattern | Unable to run this test due to  Time constraints |
| 4 | PASS | TPSTATS  Steady state load | No blocked processes or dropped messages | Dropped or blocked processes indicate a bottleneck in hardware or configuration tuning issues | Test #12 |
| 5 | PASS | Write latency with steady state load | Flat or downward trend | Increasing latency indicates Commit log or Memtable could be overrun with write requests | Test #12 |
| 6 | PASS | CMS collections  (Concurrent Mark Sweep)  Steady state (any work load) | No OldGen activity lasting > 5 min | OldGen garbage collection is expensive. If heap is not released consistently, excessive CMS activity can cause OutOfMemoryExceptions in Casssandra | All tests |
| 7 | FAIL | Datastax Enterprise Preflight Test Suite and Version check | No ERRORs in results | Mismatched versions of Cassandra between nodes negatively impacts performance.  Additionally the Preflight test suite will check basic configurations | [kratos@ip-172-31-28-94 pfc]$ `sudo ./preflight\_check`  INFO Blockdev readahead values are within reason.  INFO vm.max\_map\_count set correctly.  ERROR 12 missing limits found:  ERROR \* soft nofile 32768  ERROR \* hard nofile 32768  ERROR root soft nofile 32768  ERROR root hard nofile 32768  ERROR \* soft memlock unlimited  ERROR \* hard memlock unlimited  ERROR root soft memlock unlimited  ERROR root hard memlock unlimited  ERROR \* soft as unlimited  ERROR \* hard as unlimited  ERROR root soft as unlimited  ERROR root hard as unlimited  ERROR Limits check did not complete successfully.  WARNING "\* soft nproc 1024" should be set to "\* soft nproc 10240" in '/etc/security/limits.d/90-nproc.conf' if on CentOS, RHEL, or OEL Systems.  ERROR Nproc check did not complete successfully.  WARNING Use --ssd or --nossd to signal ssd usage.  ERROR SSD check did not complete successfully.  INFO Swap disabled.  WARNING Please ensure '/vol00/cassandra/data' and '/vol00/cassandra/commitlog' are linked to seperate devices.  ERROR Yaml check did not complete successfully.  WARNING Use --device to signal devices to benchmark.  ERROR Disk benchmarks check did not complete successfully. |
| 8 | FAIL | Row sizes in cfhistogram | Graduated column sizes (no one partition/row greatly outsizes the others) | A single partition that is much larger than any others is a sign of problems with client application and/or data model | Each table showed data hot spots in different  Test runs. Especially assessment\_by\_type &  Assessment\_by\_date tables.  Several latency histograms had “two humps”,  Which is another documented indicator of data  Model issues. (see artifacts section for  Details)  Test #7, 9, 11, 12, 14 |
| 9 | PASS | Cassandra Anti-Patterns | Avoidance of Cassandra anti-patterns in Table DDL, main application Cassandra clients or application queries run with Trace enabled  Please see this link for details> | AntiPatterns fall into different categories:   * Deployment * Data model * Access Patterns   Emphasis placed on examining application source code, access patterns and data model would provide the most value | Anti-patterns:  client side joins  nonUse of PreparedStatements  nonUse of async feature  multi-partition queries  misuse of batches (logged, unlogged)  Distributed mutable state  Excessive usage of Deletes  C\* used as a queue  Secondary Indexes  Inappropriate read or write retry policy  -----  Todo: create a central reference  to document these |
| 10 | PASS | Parnew Garbage collection | Should not last more than 1000 ms per 1 sec interval | Excessive ParNew collection indicates overload of the write paths and can lead to unresponsiveness | All test runs |

## Learning and Key Observations:

* Data Model Scalability: Design has excellent scaling characteristics.
* Write latency on per table basis ranges from 17-20 microseconds / op
* 12-node Write Capacity: 5000/sec
* 12-node Read Capacity: 1400/sec
* Under stress, neither CPU nor I/O is a bounding constraint. Our primary bounding constraint is Compaction. This process tends to get behind when loads increase.
* Under heavy write load, assessments\_by\_date consistently has lower SSTable count than the other two tables
* Cfhistogram measurements of each table showed consistent “two-hump” latency profiles. This is considered a indicative of problems with the data model and/or tuning.

## RECOMMENDATIONS

Increase tribal knowledge of anti-patterns, as well as ability to recognize them.

Create better isolation of application keyspaces. Ideally we should have a separate, ephemeral cluster of Cassandra nodes to use for these performance tests. This would provide better control over the state of the data and prevent cross-contamination of cluster activity between applications that share the same Cassandra cluster.

Continue improving test tools and practices.

## NEXT STEPS

Follow up on failure items from Preflight Check. - [NICE-563](https://jira.nike.com/browse/NICE-563)

url> <http://docs.datastax.com/en/cassandra/1.2/cassandra/install/installRecommendSettings.html>

cfhistograms revealed “hot spot” data in cfhistograms for assessments\_by\_type, assessments\_by\_date tables. Find root cause and propose a fix - NICE-564

Iterate on development of the Nike \*dbperf test tool. We received feedback from the team on this iteration of ways to improve the tool and make it more user-friendly. – [NICE-524](https://jira.nike.com/browse/NICE-524)

Find root cause of “two hump” latency in cfhistograms - [NICE-565](https://jira.nike.com/browse/NICE-565)

Find root cause for OpsCenter. It has not been operating normally since the most recent stress write test was run. -- [NICE-567](https://jira.nike.com/browse/NICE-567)

Document best practices for testing Cassandra (based on field experience and research) - [NICE-526](https://jira.nike.com/browse/NICE-526)

Assessment Data Model Test

ARTIFACTS

## 1) Tables

CREATE TABLE IF NOT EXISTS assessment\_keyspace.assessments\_by\_date (user\_id text, start\_time timestamp, assessment\_json text, PRIMARY KEY (user\_id, start\_time)) WITH CLUSTERING ORDER BY (start\_time DESC) AND compaction={'class': 'LeveledCompactionStrategy'} AND compression={'sstable\_compression': 'LZ4Compressor'};

CREATE TABLE IF NOT EXISTS assessment\_keyspace.assessments\_by\_type (user\_id text, assessment\_type text, start\_time timestamp, assessment\_json text, PRIMARY KEY (user\_id, assessment\_type, start\_time)) WITH CLUSTERING ORDER BY (assessment\_type ASC, start\_time DESC) AND compaction={'class': 'LeveledCompactionStrategy'} AND compression={'sstable\_compression': 'LZ4Compressor'};

CREATE TABLE IF NOT EXISTS assessment\_keyspace.assessments\_by\_type\_id (user\_id text, assessment\_type text, assessment\_type\_id text, start\_time timestamp, assessment\_json text, PRIMARY KEY (user\_id, assessment\_type, assessment\_type\_id, start\_time)) WITH CLUSTERING ORDER BY (assessment\_type ASC, assessment\_type\_id DESC, start\_time DESC) AND compaction={'class': 'LeveledCompactionStrategy'} AND compression={'sstable\_compression': 'LZ4Compressor'};

## 2) CQL queries

POST /assessments/me

INSERT INTO assessments\_by\_date (user\_id, start\_time, assessment\_json) VALUES ('1552abe7-ba99-4124-854b-e25580ca0a65', 143949023, '{full assessment json posted here}');

INSERT INTO assessments\_by\_type (user\_id, assessment\_type, start\_time, assessment\_json) VALUES ('1552abe7-ba99-4124-854b-e25580ca0a65', 'FUEL\_PRINT', 143949023, '{full assessment json posted here}');

INSERT INTO assessments\_by\_type\_id (user\_id, assessment\_type, assessment\_type\_id, start\_time, assessment\_json) VALUES ('1552abe7-ba99-4124-854b-e25580ca0a65', 'FUEL\_PRINT', '1.1', 143949023, '{full assessment json posted here}');

GET /assessments/me/{id}

// when traced in cqlsh, these should show as “single partition” queries

SELECT \* FROM assessments\_by\_date WHERE user\_id = ‘123456789' AND start\_time = 143949023;

SELECT \* FROM assessments\_by\_type WHERE user\_id = ‘123456789' AND assessment\_type = 'FUEL\_PRINT';

SELECT \* FROM assessments\_by\_type\_id WHERE user\_id ‘123456789' AND assessment\_type = 'FUEL\_PRINT' and assessment\_type\_id = '1.1';

GET /assessments/me

// range queries - Cassandra is very good at these types, provided a sorted column is used in ‘where’ clause

SELECT \* FROM assessments\_by\_date WHERE user\_id = '1552abe7-ba99-4124-854b-e25580ca0a65' AND start\_time < 143949023 LIMIT 1;

SELECT \* FROM assessments\_by\_type WHERE user\_id = '1552abe7-ba99-4124-854b-e25580ca0a65' AND assessment\_type = 'FUEL\_PRINT' AND start\_time < 143949023 LIMIT 1;

SELECT \* FROM assessments\_by\_type\_id WHERE user\_id = '1552abe7-ba99-4124-854b-e25580ca0a65' AND assessment\_type = 'FUEL\_PRINT' AND assessment\_type\_id = '1.1' AND start\_time < 143949023 LIMIT 1;

## 3) Keyspace definition

CREATE KEYSPACE assessment\_keyspace WITH replication = {'class':'NetworkTopologyStrategy', 'us-west-2':'3'};

## 4) JVM arguments (shortened for brevity)

|  |  |
| --- | --- |
| Steady Write ( #12)  userCount=1000  CassandraConsistencyLevelRead LOCAL\_ONE \  --CassandraConsistencyLevelWrite LOCAL\_QUORUM \  --dbConnPoolSize 10 \  --metricsReporterSecondsBetweenReports 500 \  --NumberOfUserOpsRunEachThread 1000 \  --RatioUserReads 0 \  --RatioUserWrites 100 \  --resultSetRowLimit 10 \  --simultaneousThreads 50 \ | Stress Write (#16)  userCount=50  CassandraConsistencyLevelRead LOCAL\_ONE \  --CassandraConsistencyLevelWrite LOCAL\_QUORUM \  --dbConnPoolSize 10 \  --metricsReporterSecondsBetweenReports 500 \  --NumberOfUserOpsRunEachThread 1000000 \  --RatioUserReads 0 \  --RatioUserWrites 100 \  --resultSetRowLimit 10 \  --simultaneousThreads 400 \  --userCount $userCount |
| Steady Read (#17)  userCount=50  CassandraConsistencyLevelRead LOCAL\_ONE \  --CassandraConsistencyLevelWrite LOCAL\_QUORUM \  --dbConnPoolSize 10 \  12--metricsReporterSecondsBetweenReports 500 \  --NumberOfUserOpsRunEachThread 1000000 \  --RatioUserReads 0 \  --RatioUserWrites 100 \  --resultSetRowLimit 10 \  --simultaneousThreads 8 \  --userCount $userCount | Stress Read (#13)  userCount=10000  CassandraConsistencyLevelRead LOCAL\_ONE \  --CassandraConsistencyLevelWrite LOCAL\_QUORUM \  --dbConnPoolSize 10 \  12--metricsReporterSecondsBetweenReports 500 \  --NumberOfUserOpsRunEachThread 10 \  --RatioUserReads 0 \  --RatioUserWrites 100 \  --resultSetRowLimit 10 \  --simultaneousThreads 400 \  --userCount $userCount |

## 5) Test Tool Output

# Nodetool Cfhistograms

* “double humps” are an indicator of data model problems.
* Another indicator of issues is when a single row / partition contains a really large number of bytes.

Found in Test Run #2, 5, 6, 7, 11, 12, 14

**assessment\_keyspace/assessments\_by\_type\_id histograms**

Write Latency (microseconds)

3 us: 1

4 us: 37

5 us: 181

6 us: 649

7 us: 1524

8 us: 2334

10 us: 6621

12 us: 8572

14 us: 10792

17 us: 18878 ### one hump is OK. Two indicates a problem

20 us: 14266

24 us: 10715

29 us: 5260

35 us: 1067

42 us: 157

50 us: 55

60 us: 14

72 us: 2

86 us: 1

103 us: 0

124 us: 20

149 us: 218

179 us: 127

215 us: 8

258 us: 0

310 us: 0

372 us: 0

446 us: 0

535 us: 0

642 us: 1

assessments\_by\_date

Read Latency (microseconds)

No Data

Partition Size (bytes)

3311 bytes: 9198

3973 bytes: 0

4768 bytes: 102

5722 bytes: 0

6866 bytes: 0

8239 bytes: 0

9887 bytes: 121

11864 bytes: 0

14237 bytes: 143

17084 bytes: 0

20501 bytes: 124

24601 bytes: 113

29521 bytes: 89

35425 bytes: 77

42510 bytes: 90

51012 bytes: 29

61214 bytes: 25

73457 bytes: 6

88148 bytes: 1 ### hot data spot – single partition containing large number of bytes

105778 bytes: 0

126934 bytes: 0

152321 bytes: 0

182785 bytes: 0

219342 bytes: 0

263210 bytes: 0

315852 bytes: 0

379022 bytes: 0

454826 bytes: 0

545791 bytes: 9

654949 bytes: 255

785939 bytes: 208

943127 bytes: 6

assessment\_keyspace/assessments\_by\_type histograms

Partition Size (bytes)

3311 bytes: 9198

3973 bytes: 0

4768 bytes: 175594

5722 bytes: 0

6866 bytes: 0

8239 bytes: 0

9887 bytes: 185958

11864 bytes: 0

14237 bytes: 132363

17084 bytes: 0

20501 bytes: 70624

24601 bytes: 30305

29521 bytes: 10819

35425 bytes: 3282

42510 bytes: 1158

51012 bytes: 43

61214 bytes: 9

73457 bytes: 0

88148 bytes: 11

105778 bytes: 99

126934 bytes: 569

152321 bytes: 1309

182785 bytes: 1815

219342 bytes: 582

263210 bytes: 96

315852 bytes: 0

379022 bytes: 0

454826 bytes: 0

545791 bytes: 0

654949 bytes: 0

785939 bytes: 0

943127 bytes: 0

1131752 bytes: 5

1358102 bytes: 345

1629722 bytes: 77

1955666 bytes: 0

2346799 bytes: 0

2816159 bytes: 0

3379391 bytes: 0

4055269 bytes: 0

4866323 bytes: 0

5839588 bytes: 1

7007506 bytes: 50