Cassandra @ macy's

Andrew Santoro *Development Manager*
Peter Connolly *Application Architect*
Ivan Ukolov *Architect/Implementation Lead*

April 2015

Agenda

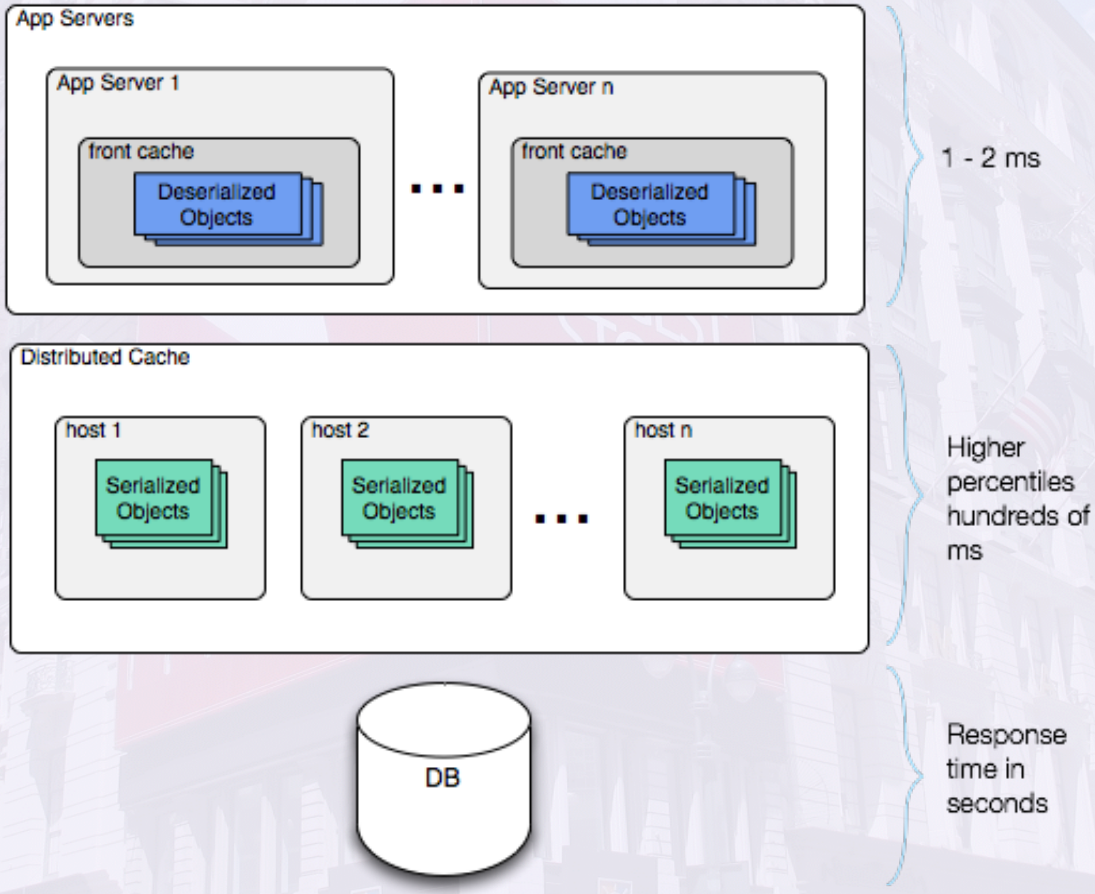
- Background/Problem Statement
- Options
- Migration
- Data Models
- Performance
- Retrospectives
- Future plans
- Video

Background & Problem Statement

About Us

- Real-time RESTful data services application
- Serve product, inventory and store data to website, mobile, store, and partner applications
- Not “Big Data” or Analytics
- Growth in business lead us to want:
  - 10x growth in data
  - Move from a read-mostly model to one which could handle near-real-time updates
  - Move into multiple data centers

Scaling Problems

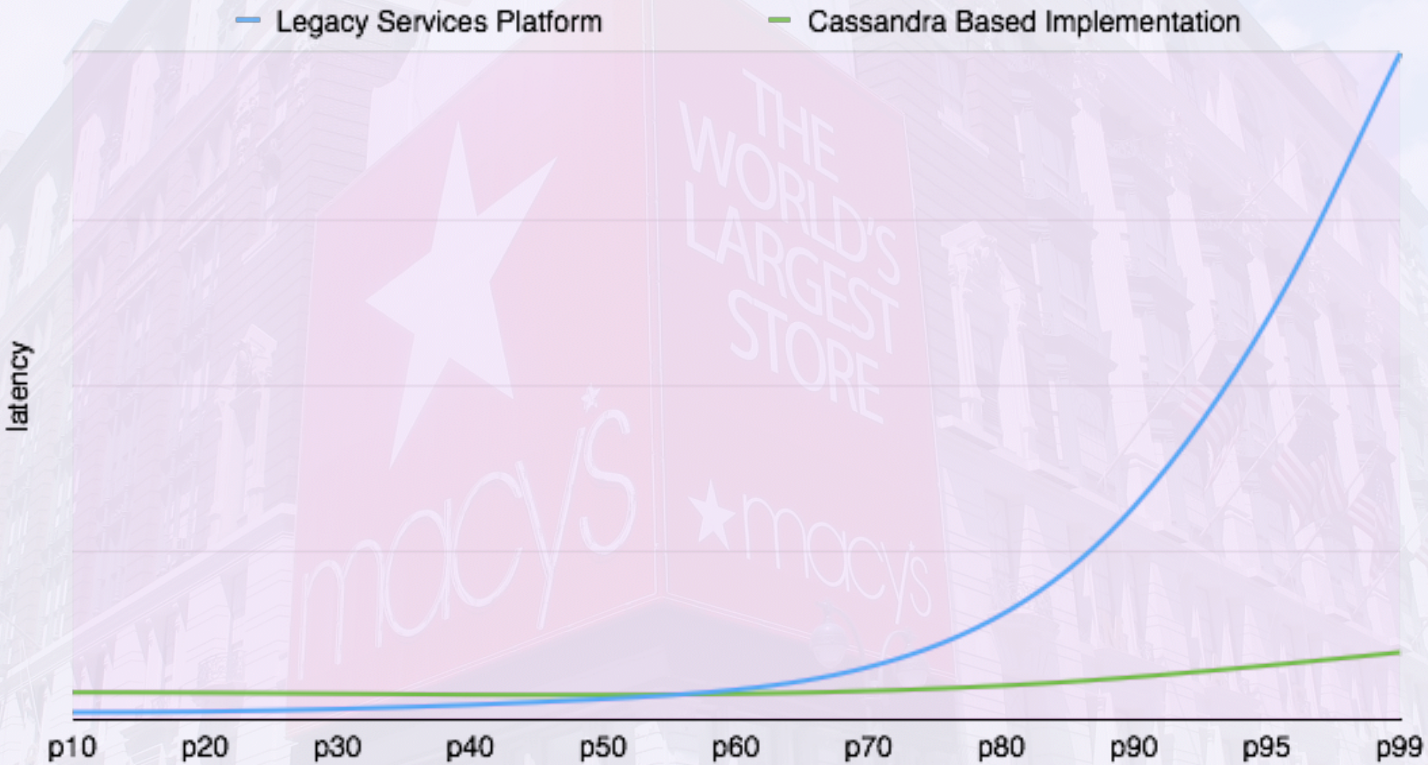


macy's

Operational Problems

- Prewarming step was a prerequisite to satisfy performance SLA
- Maintained second data grid for failover scenario
    - hard to keep those consistent
    - complicated release process

Performance Goals for 10x Data



Options Explored

Options Evaluated

- Denormalized Relational DB: DB2
- Document DB: MongoDB
- Columnar DB: Cassandra, Couchbase, ActiveSpaces
- Graph: Neo4J
- Object: Versant

Options Short List

- MongoDB
  - Feature-rich, JSON document DB
- Cassandra
  - Scalable, true peer-to-peer
- ActiveSpaces
  - TIBCO Proprietary
  - Existing relationship with TIBCO
  - In-memory key/value datagrid

POC Environment

- Amazon EC2
  - 5 servers
  - Same servers used for each test
  - Had vendors assist with setup and execution of benchmarks
  - Modeled benchmarks after retail inventory use cases
  - C3.2xlarge instances
  - Baseline of 148MM inventory records

POC Results - Summary

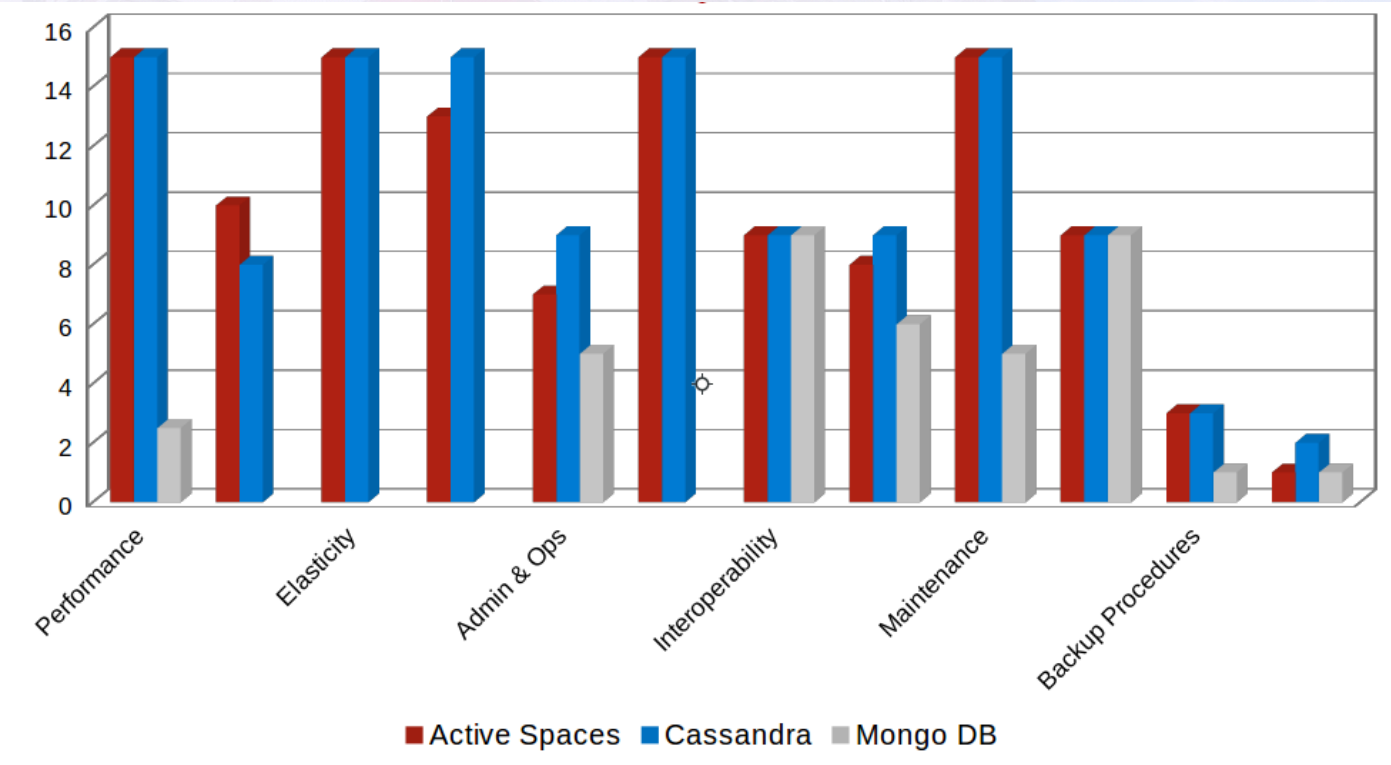
Criteria	Active Spaces	Cassandra	Mongo DB
Performance	15	15	2.5
Scalability	10	8	0
Elasticity	15	15	0
Fault Tolerance	13	15	0
Admin and Operation	6	9	5
Robustness	15	15	0
Interoperability	9	9	9
Functional Maturity	6	9	6
Maintenance	15	15	5
Ease of Consumption	9	9	9
Backup Procedures	3	3	3
Recovery Procedures	3	3	1
Data Security	2	6	1
Aggregate Score	**121**	**131**	**41.5**

macy's

POC Results - Summary

- **Cassandra & ActiveSpaces**
  - Very close
- **MongoDB**
  - Failed tests

YMMV!
Your mileage may (will probably) vary



macy's

POC Results - Initial Load

Operation	TIBCO ActiveSpaces	Apache Cassandra	10Gen MongoDB
Initial Load ~148MM Records Same datacenter (availability zone)	72,000 TPS (32 nodes) 34 min 98,000 TPS (40 nodes) 25 min CPU: 62.4% Memory: 83.0% I/O, Disk 1: 36%	65,000 TPS (5 nodes) 38 min CPU: 31% Memory: 59% I/O, Disk 1: 42% I/O, Disk 2: 14%	20,000 TPS (5 nodes) ?? min (Did not complete) processed ~23MM records

macy's

POC Results - Upsert (Writes)

Operation	TIBCO ActiveSpaces	Apache Cassandra	10Gen MongoDB
Upsert (Writes) ActiveSpaces sync writes vs. Cassandra async	4,000 TPS 3.57 ms Avg Latency CPU: 0.6% Memory: 71% I/O: 18% (disk 1) I/O: 17% (disk 2)	4,000 TPS 3.2 ms Avg Latency CPU: 3.7% Memory: 77% I/O: 0.3% I/O: 2.2%	(Did not complete) tests failed

macy's

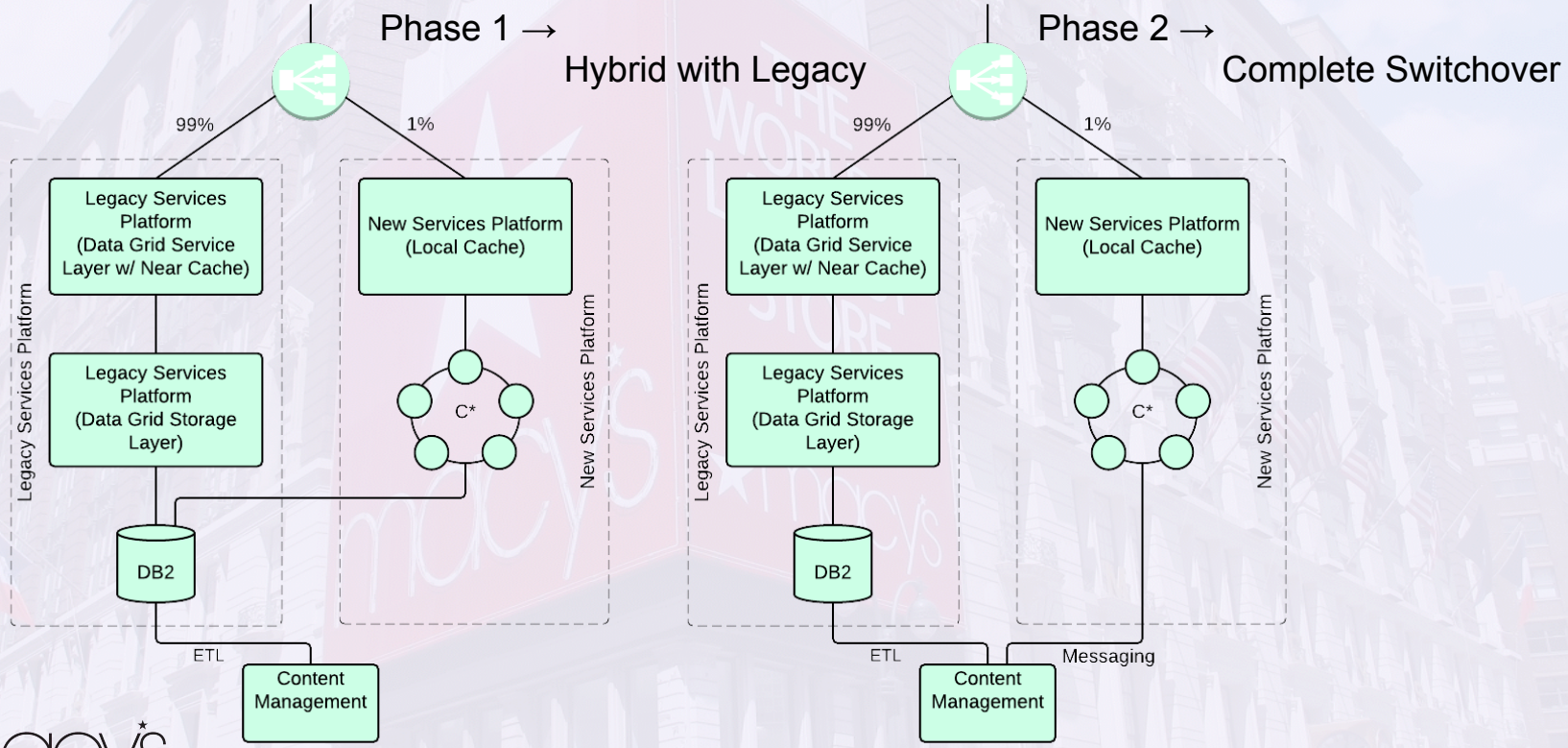
POC Results - Read

Operation	TIBCO ActiveSpaces	Apache Cassandra	10Gen MongoDB
Read	400 TPS 2.54 ms Avg Latency CPU: 0.06% Memory: 62.4% I/O: 0%	400 TPS 3.23 ms Avg Latency CPU: 0.02% Memory: 47% I/O: 3.7%	(Did not complete) tests failed

macy's

Migration Path

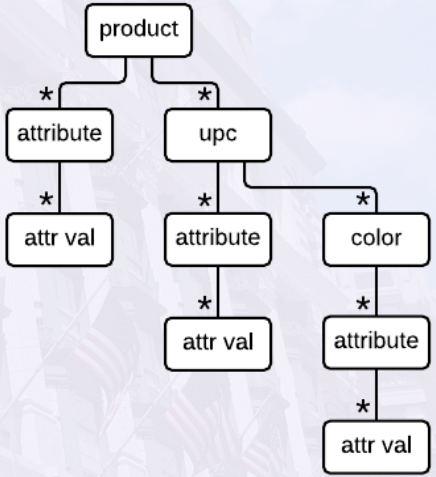
Past & Present



Data Models

Data Model Requirements

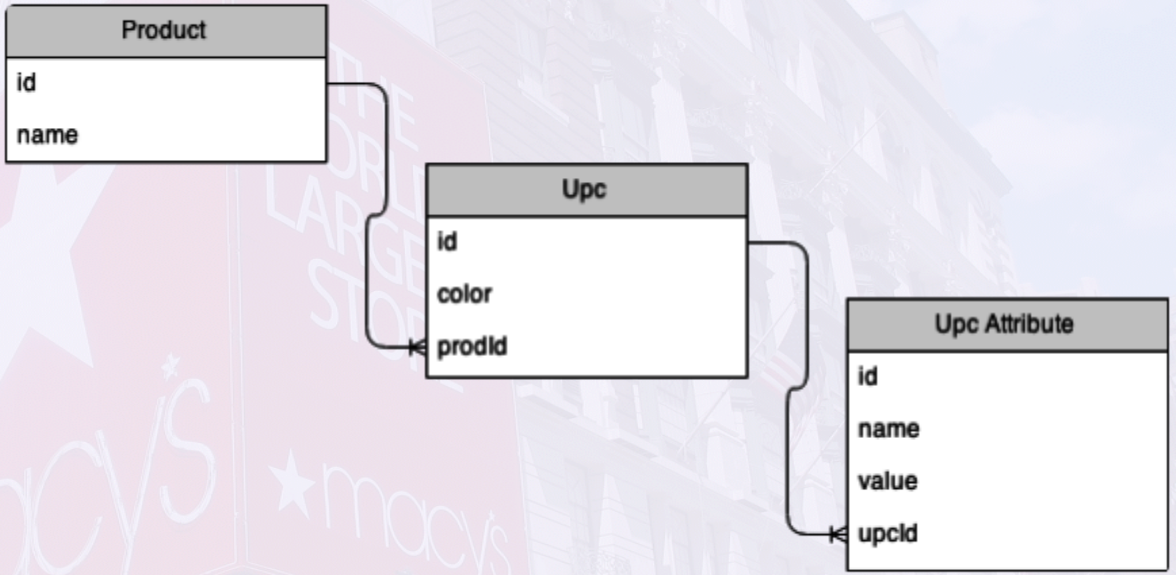
- Model hierarchical domain objects
- Aggregate data to minimize reads
- No reads before writes
- Readable by 3rd party tools (cqlsh, Dev Center)



macy's

Data Model based on Lists

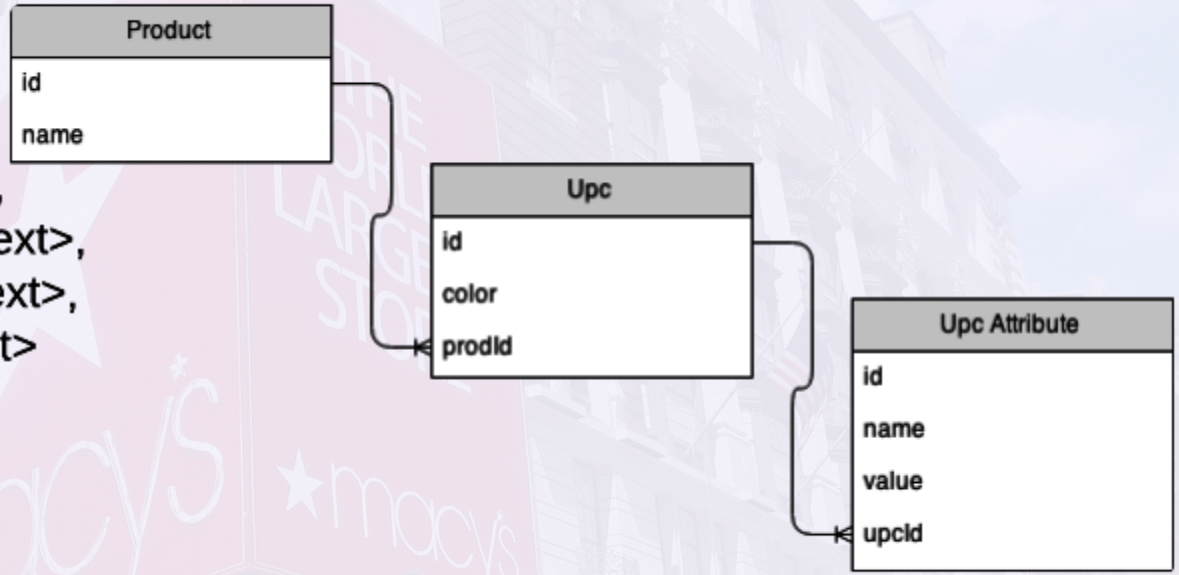
```
CREATE TABLE product (
    id int PRIMARY KEY,
    name text,
    upcId list<int>,
    upcColor list<text>,
    upcAttrId list<int>,
    upcAttrName list<text>,
    upcAttrValue list<text>,
    upcAttrRefId list<int>
);
```



id	name	upcId	upcColor	upcAttrId	upcAttrName	upcAttrValue	upcAttrRefId
11	Nike Pants	[22, 33]	['White', 'Red']	[44, 55, 66]	['ACTIVE', 'PROMOTION', 'ACTIVE']	['Y', 'Y', 'N']	[22, 22, 33]

Data Model based on Maps

```
CREATE TABLE product (
    id int PRIMARY KEY,
    name text,
    upcColor map<int, text>,
    upcAttrName map<int, text>,
    upcAttrValue map<int, text>,
    upcAttrRefId map<int, int>
);
```




```
id | name        | upcColor                  | upcAttrName                                                                  | upcAttrValue                | upcAttrRefId
---+-------------+---------------------------+------------------------------------------------------------------------------+-----------------------------+-------------------------
 11 | Nike Pants | {22: 'White', 33: 'Red'} | {44: 'ACTIVE', 55: 'PROMOTION', 66: 'ACTIVE'} | {44: 'Y', 55: 'Y', 66: 'N'} | {44: 22, 55: 22, 66: 33}
```

macy's

Collection Size Limits

- Native protocol v2
  - collection max size is 64K
  - collection element max size is 64K
- Native protocol v3
  - the collection size and the length of each element is 4 bytes long now
  - still 64K limit for set element max size and map key max size

https://github.com/apache/cassandra/blob/trunk/doc/native_protocol_v3.spec

Collections Internal Storage

```
CREATE TABLE collections (
        my_key int PRIMARY KEY,
        my_set set<int>,
        my_map map<int, int>,
        my_list list<int>,
);
```

```
INSERT INTO collections
(my_key, my_set, my_map, my_list)
VALUES ( 1, {1, 2}, {1:2, 3:4}, [1, 2]);
```

```
SELECT * FROM collections ;
 my_key | my_list | my_map   | my_set
------------+---------+--------------+--------
          1 |    [1, 2] | {1: 2, 3: 4} | {1, 2}
```

macy's

Collections Internal Storage

RowKey: 1

=> (name=, value=, timestamp=1429162693574381)

=> (name=my_list:**c646b7d0e3fa11e48d582f4261da0d90**, value=**00000001**, timestamp=1429162693574381)

=> (name=my_list:**c646b7d1e3fa11e48d582f4261da0d90**, value=**00000002**, timestamp=1429162693574381)

=> (name=my_map:**00000001**, value=**00000002**, timestamp=1429162693574381)

=> (name=my_map:**00000003**, value=**00000004**, timestamp=1429162693574381)

=> (name=my_set:**00000001**, value=, timestamp=1429162693574381)

=> (name=my_set:**00000002**, value=, timestamp=1429162693574381)

Data Model based on Compound Key & JSON

```
CREATE TABLE product (
      id int,                    -- product id
      upcId int,                 -- 0 means product row,
                                 -- otherwise upc id indicating it is upc row
      object text,               -- JSON object
      review text,               -- JSON object
      PRIMARY KEY(id, upcId)
);
```

id	upcId	object	review
11	0	{"id" : "11", "name" : "Nike Pants"}	{"avgRating" : "5", "count" : "567"}
11	22	{"color" : "White", "attr" : [{"id": "44", "name": "ACTIVE", "value": "Y"}, ...]}	null
11	33	{"color" : "Red", "attr" : [{"id": "66", "name": "ACTIVE", "value": "N"}]}	null

Compound Key Internal Storage

```
CREATE TABLE compound_keys (
  part_key int,
  clust_key text,
  data1 int,
  data2 int,
  PRIMARY KEY (part_key, clust_key)
);
```

```
INSERT INTO compound_keys
(part_key, clust_key, data1, data2)
VALUES ( 1, 'clust_key value 1', 1, 2);

INSERT INTO compound_keys
(part_key, clust_key)
VALUES ( 1, 'clust_key value 2');
```

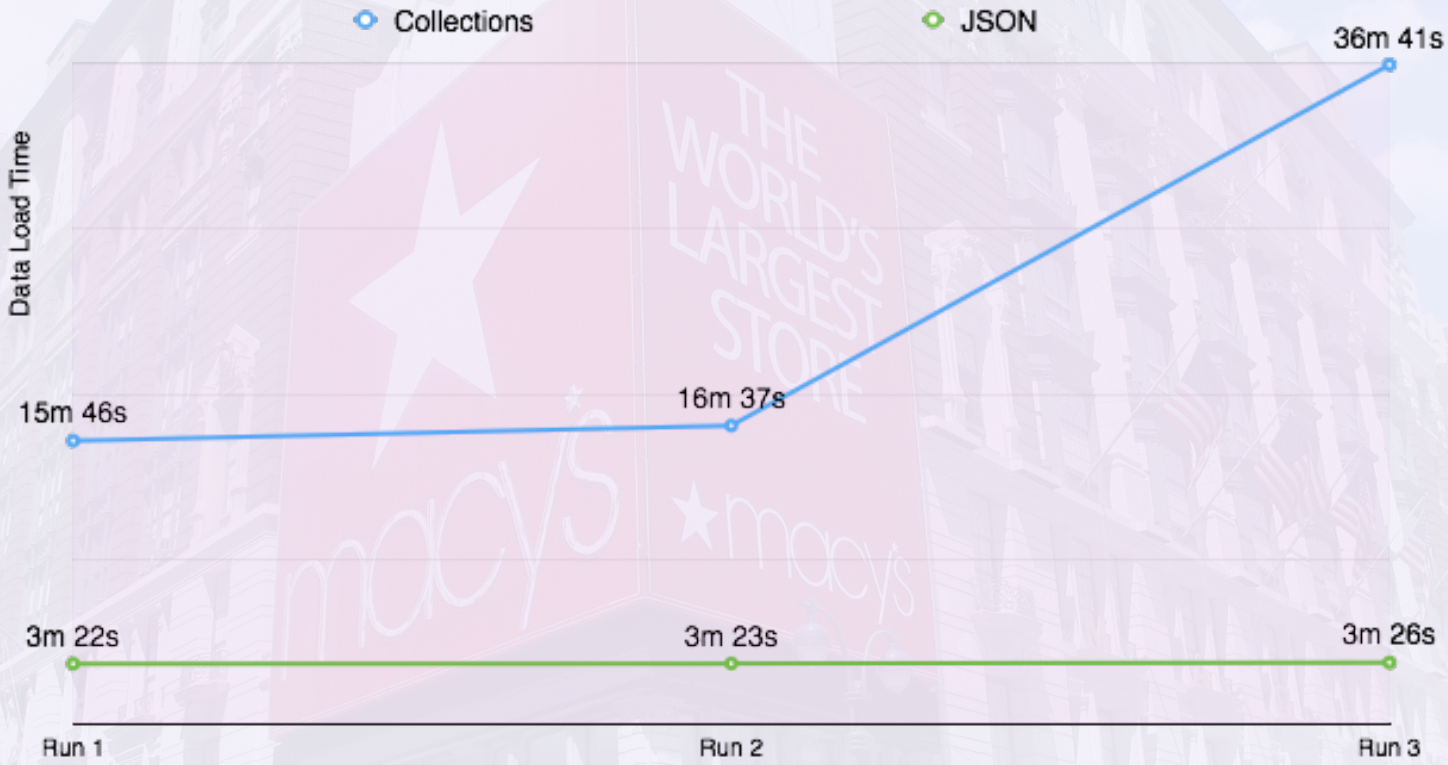
```
part_key | clust_key           | data1 | data2
-------------+---------------------+-------+-------
           1 | clust_key value 1 |     1 |     2
           1 | clust_key value 2 |  null |  null
```

macy's

Compound Key Internal Storage

RowKey: 1
=> (name=**clust_key value 1:**, value=, timestamp=1429203000986594)
=> (name=**clust_key value 1:data1**, value=**00000001**, timestamp=1429203000986594)
=> (name=**clust_key value 1:data2**, value=**00000002**, timestamp=1429203000986594)
=> (name=**clust_key value 2:**, value=, timestamp=1429203000984518)

Data Load Performance Comparison



July 2013

Column Storage Overhead

- regular column size = column name + column value + 15 bytes
- counter / expiring = +8 additional bytes

name : 2 bytes (length as short int) + byte[]

flags : 1 byte

timestamp : 8 bytes (long)

value : 4 bytes (length as int) + byte[]

http://btoddb-cass-storage.blogspot.ru/2011/07/column-overhead-and-sizing-every-column.html

https://issues.apache.org/jira/browse/CASSANDRA-4175

JSON vs primitive column types

- Significantly reduces storage overhead because of better ratio of payload / storage metadata
- Improves throughput and latency
- Supports complex hierarchical structures
- But it loses in partial reads / updates
- Complicates schema versioning

Data Model for Secondary Indices

```
CREATE TABLE product_by_upc (
    upcId int,
    prodId int,
    PRIMARY KEY (upcId)
);
```

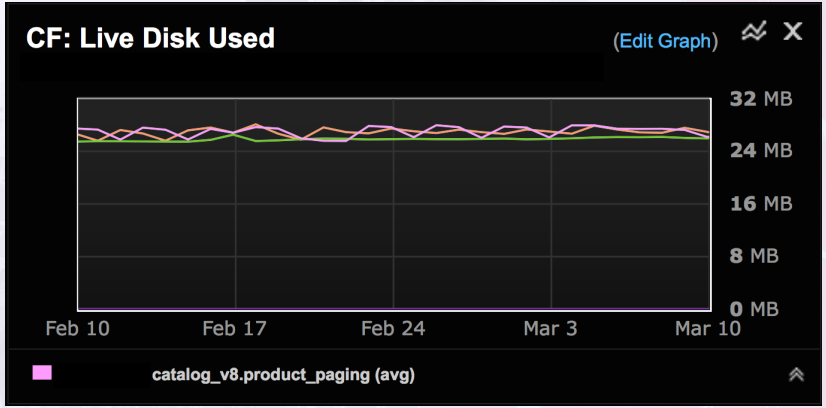
Pagination

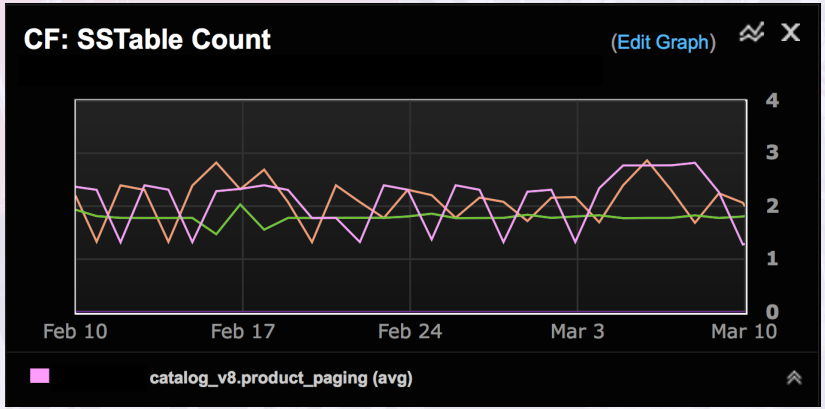
base_service_url?filter=active&_limit=100&_offset=last_prod_id

```
CREATE TABLE product_paging (
      filter text,
      prodId int,
      PRIMARY KEY (filter, prodId)
);
```

```
SELECT prodId FROM product_paging
WHERE filter='active' AND prodId >last_prod_id limit 100
```

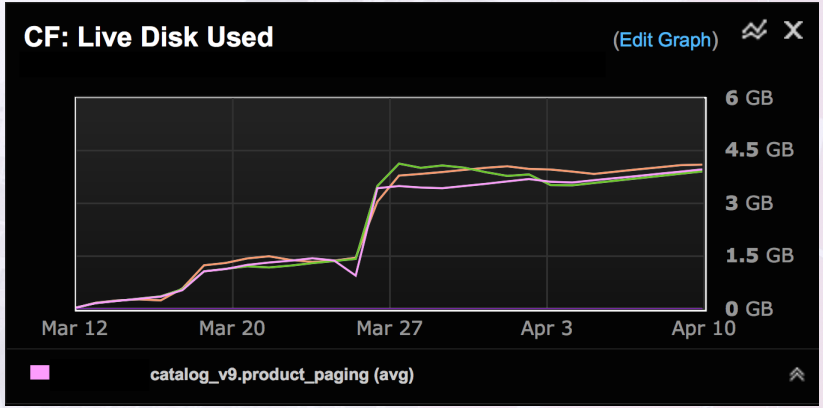
Pagination - SSTable Stats

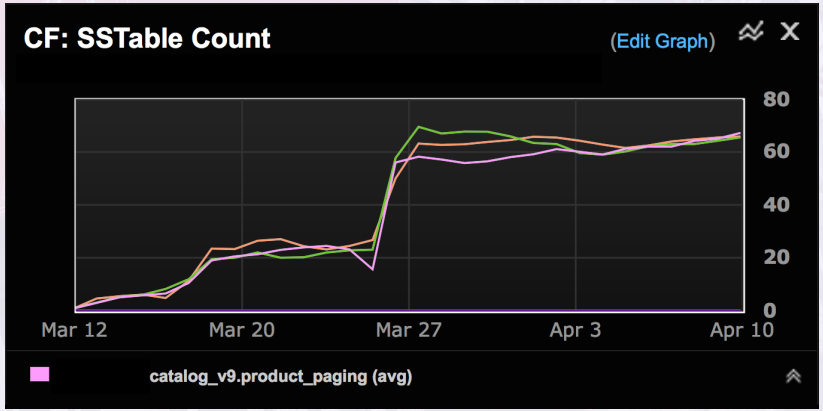




macy's

Pagination - SSTable Stats after DELETE





macy's

Compaction Issue

```
##################################################################
##### nodetool compactionstats Status at 03-03-2015 14:00:01 #####
##################################################################
        compaction type        keyspace           table       completed           total      unit  progress
             Compaction       catalog_v9   product_paging             110       196860609     bytes     0.00%
```

INFO [CompactionExecutor] 2015-04-07 02:36:44,564 CompactionController **Compacting large row** catalog_v9/product_paging (**2030276075** bytes) **incrementally**

Performance

Production Environment

- Datastax Enterprise 4.0.3 (Cassandra 2.0.7)
  - migrating to DSE 4.6.5 (Cassandra 2.0.14) in May
- 6 DSE Servers
- Server Specs
  - Red Hat Enterprise Linux Server 6.4
  - 8 Core Xeon HT w/196GB RAM
  - 2 x 400GB SSD RAID-1 Mirrored

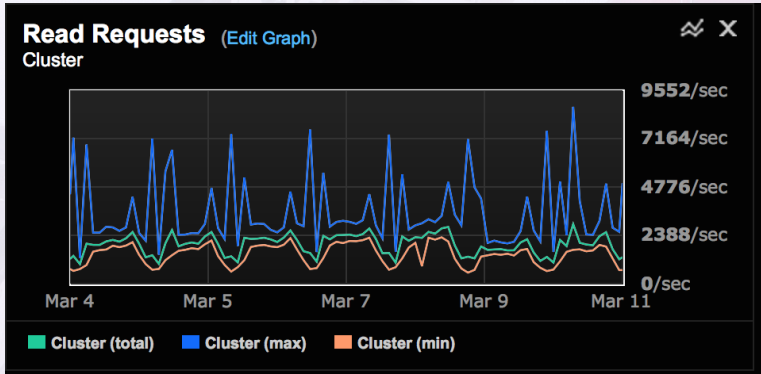
Cassandra Config

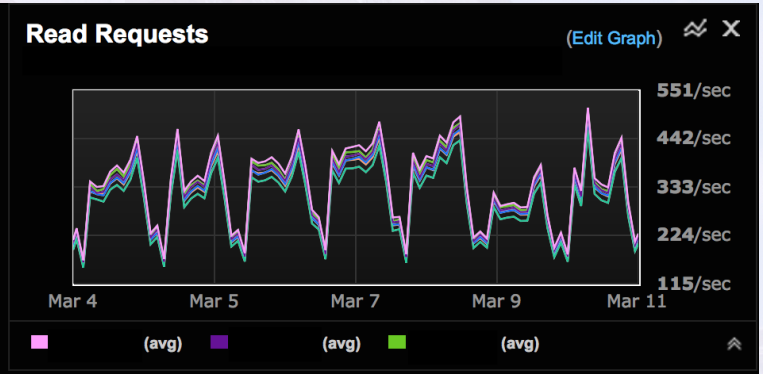
- NetworkTopologyStrategy with RF 3
- Murmur3Partitioner
- PropertyFileSnitch
- Virtual nodes
- Key cache 100MB
- Concurrent reads 32
- Concurrent writes 32
- 8GB Heap, 400MB Young Gen
- SizeTieredCompactionStrategy for bulk load data
- LeveledCompactionStrategy for real time updates

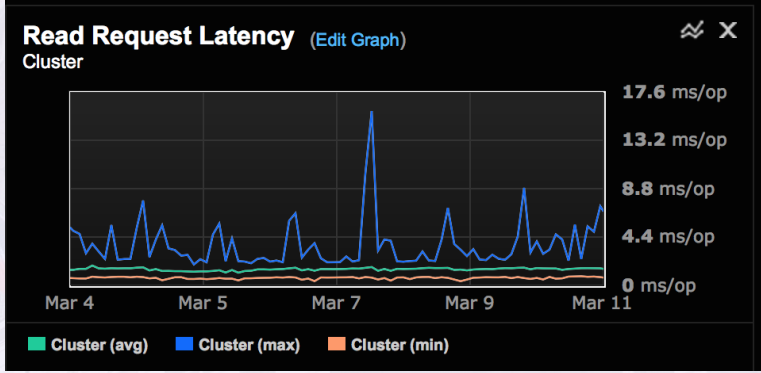
Client App Config

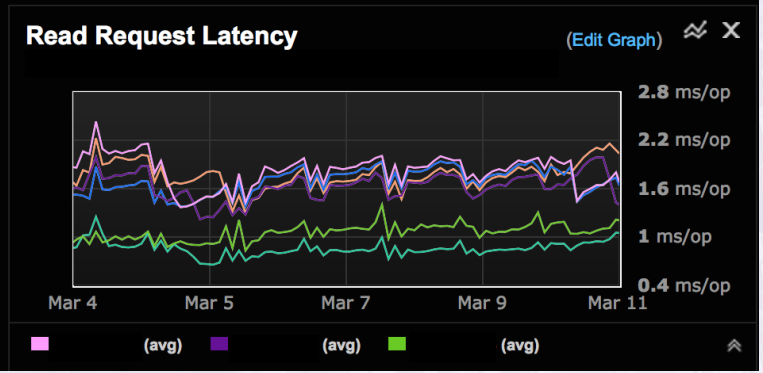
- Datastax Java Driver version 2.1.4
- CQL 3
- Write CL is LOCAL_QUORUM
- Read CL is ONE

Read Requests



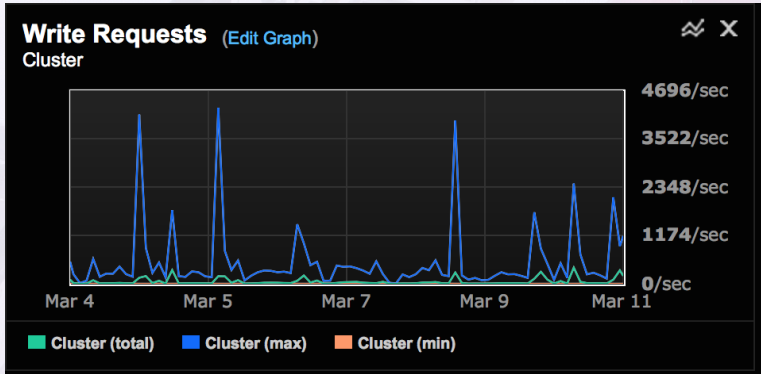


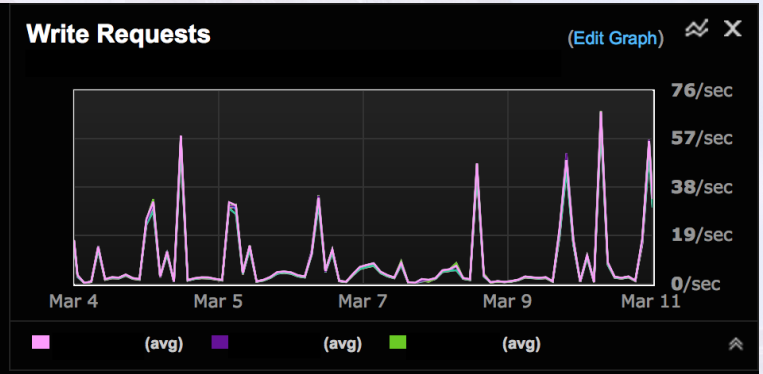


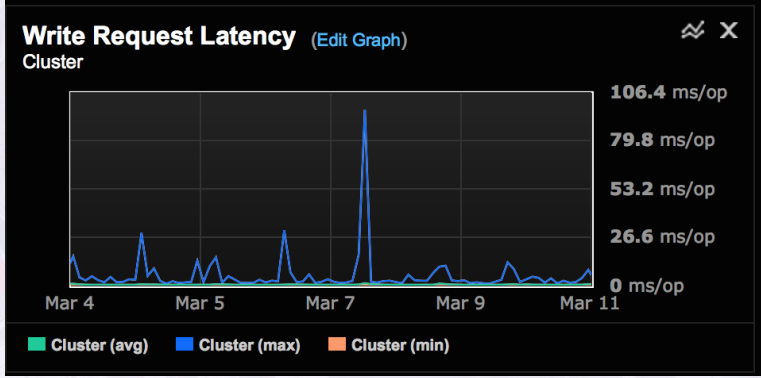


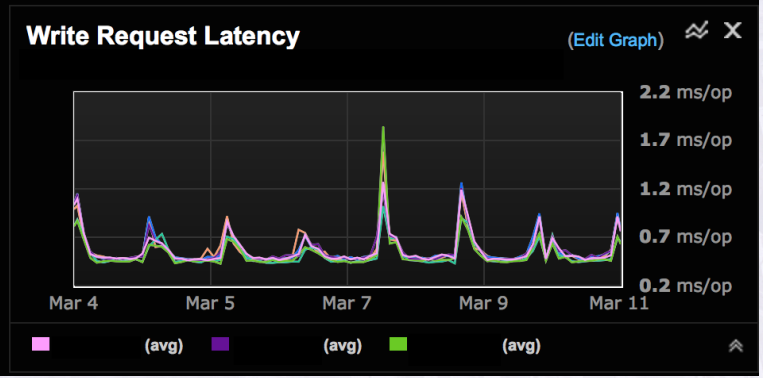
macy's

Write Requests



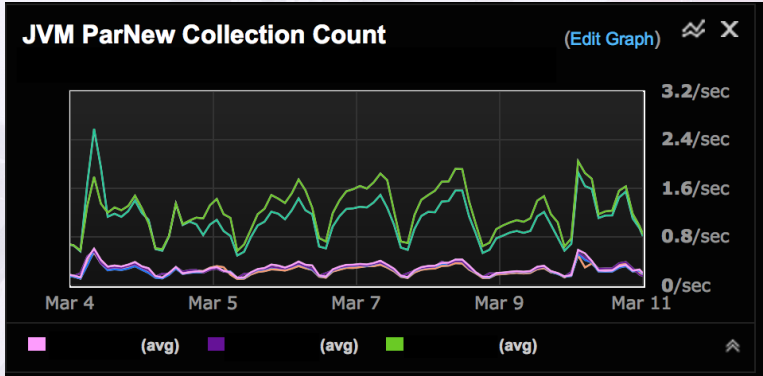


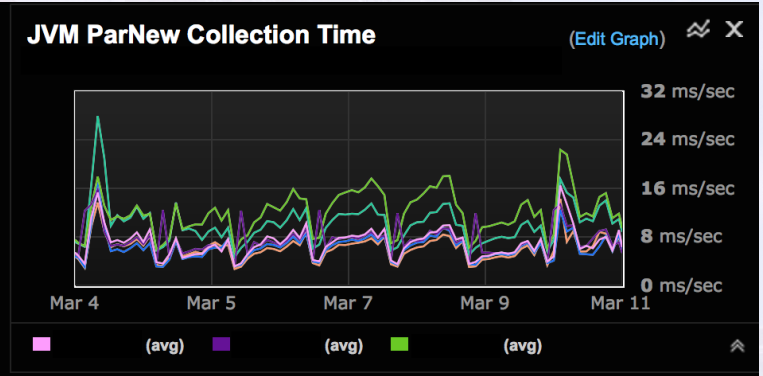


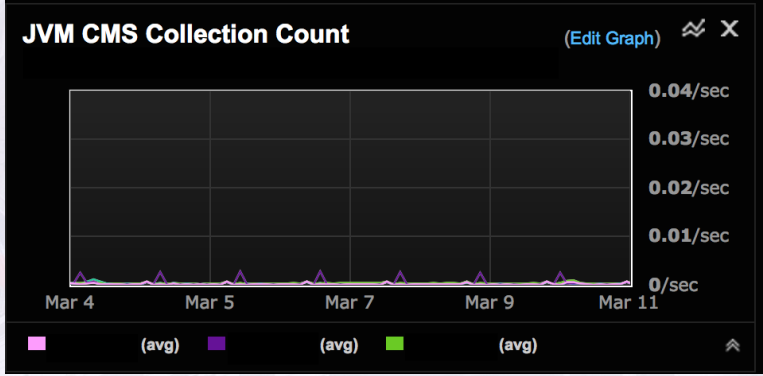


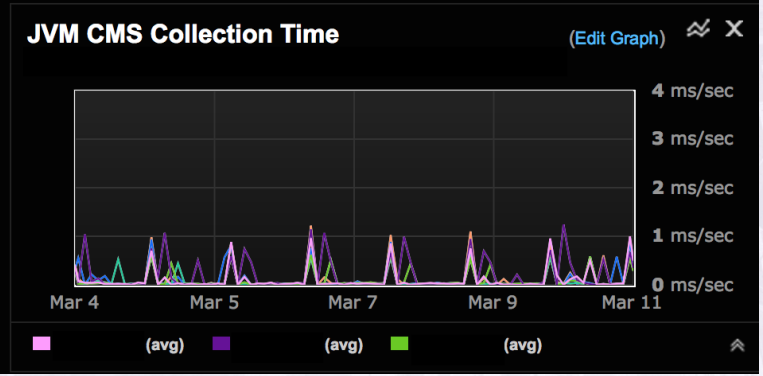
macy's

Garbage Collection



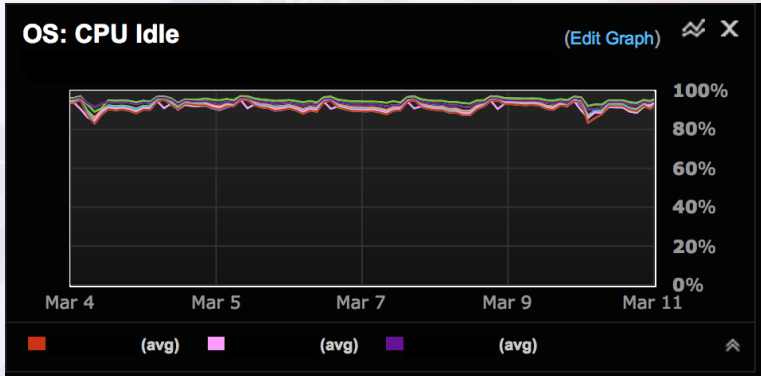


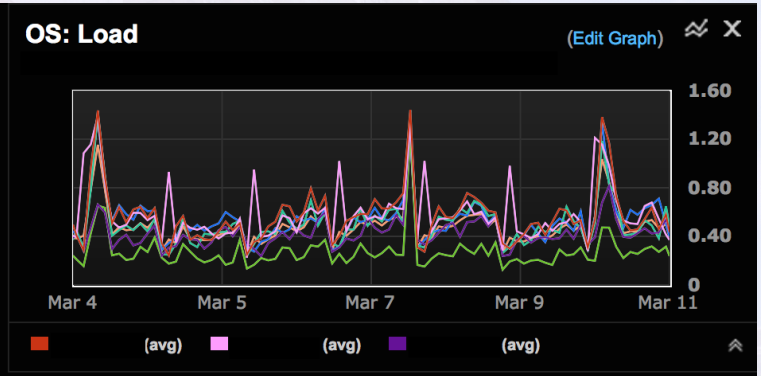


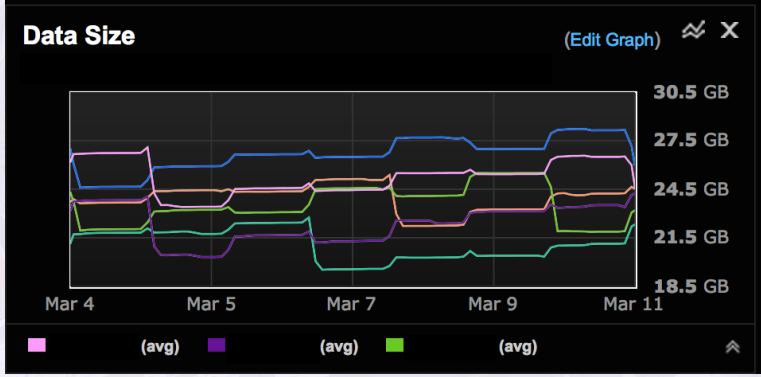


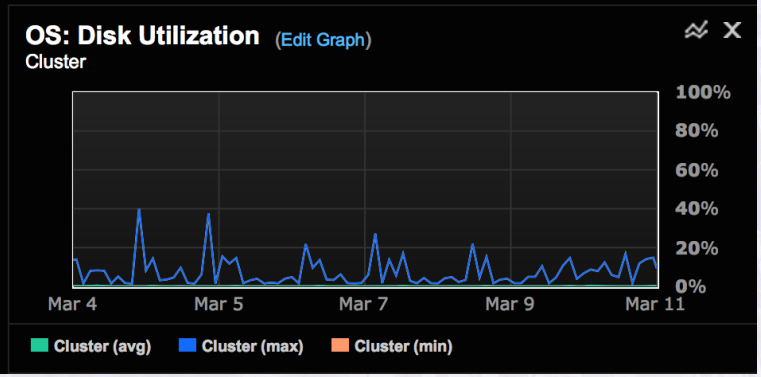
macy's

CPU & Disk









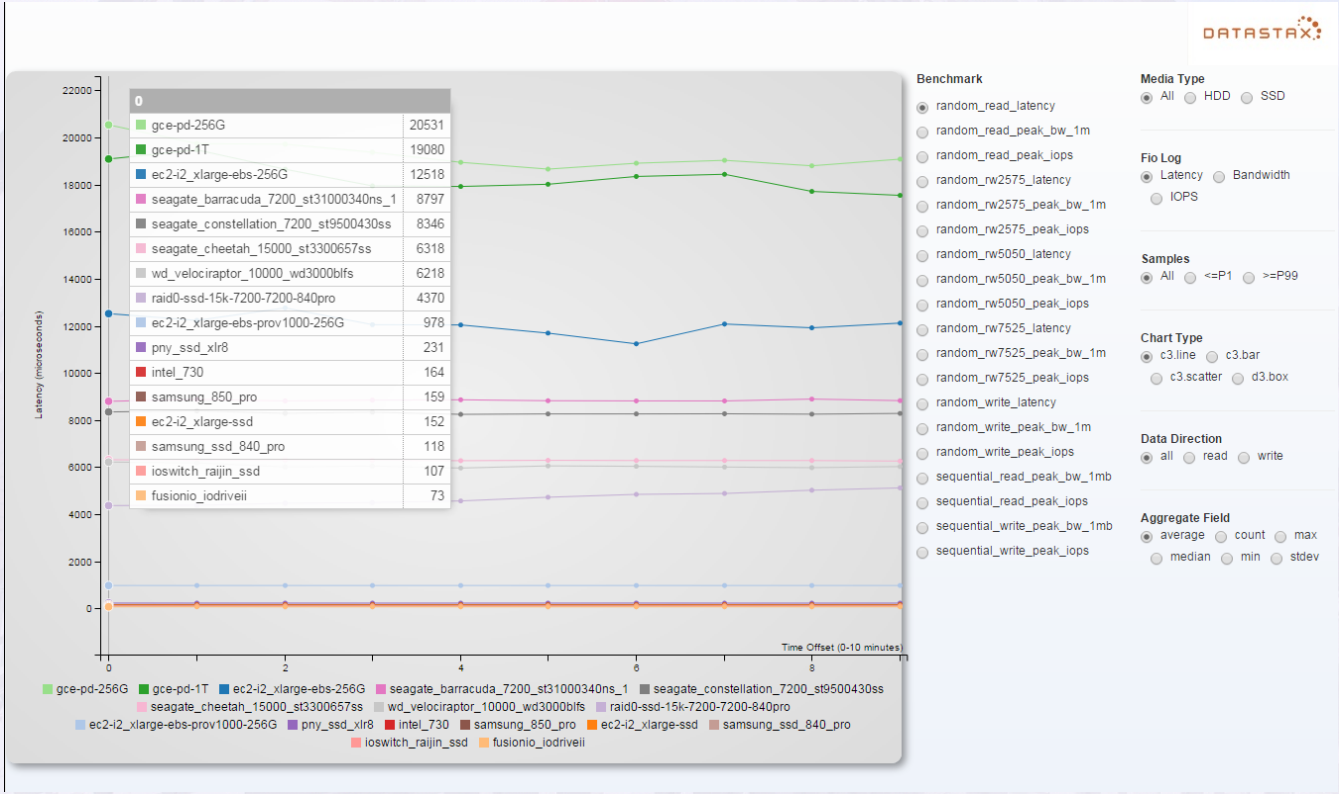
macy's

Disk Performance

Disk Performance

- DataStax provided measurement app
  - Hats off to Al Tobey & Patrick McFadin
  - Ran it internally
  - Compared it against other vendors
- Result:
  - SAN **latency** is significantly slower!
  - Don't be fooled by claims of high IOPS
- Action:
  - Use locally-mounted SSD storage!

Disk Performance



macy's

http://tobert.org/disk-latency-graphs/

Retrospectives

What Worked Well

- Documentation is good
- CQL3 easy to learn and migrate to from SQL background
- Query tracing is helpful
- Stable: Haven’t lost a node in production
- Able to reduce our reliance on caching in app tier
- Cassandra Cluster Manager (https://github.com/pcmanus/ccm)

Problems encountered with Cassandra

- CQL queries brought down cluster
- Delete and creating a keyspace ….
- Lengthy compaction
- Need to understand underlying storage
- OpsCenter Performance Charts

Our own problems

- Small cluster ⇒ **all** servers must perform well
- Lack of versioning of JSON schema
- How to handle performant exports
  - 5% CPU -> 30%
- Non-primary key access
- Under-allocated test environments

One concern…

Patrick McFadin

Buddy Pine/”Syndrome”

Secret Identity???

Future Plans

Future Plans

- Upgrade DSE 4.0.3 → 4.6.5
  - Cassandra 2.0.7 → 2.0.14
- DSE Spark Integration for reporting
- Multi-Datacenter
- Using Mutagen for schema management
- JSON schema versioning
- RxJava for asynchronous operations
- Spring Data Cassandra

Stuff we'd like to see in DSE

- DSE Kibana
  - Augment Spark's query capability with an interactive GUI
- Replication listeners
  - Detect when replication is complete

Video


We’re hiring ...
ecommerce.macysjobs.com
3rd & Folsom
macy's.com
macy's

Questions?