DataStax Enterprise 3.x

Realtime Analytics with Solr

Jason Rutherglen

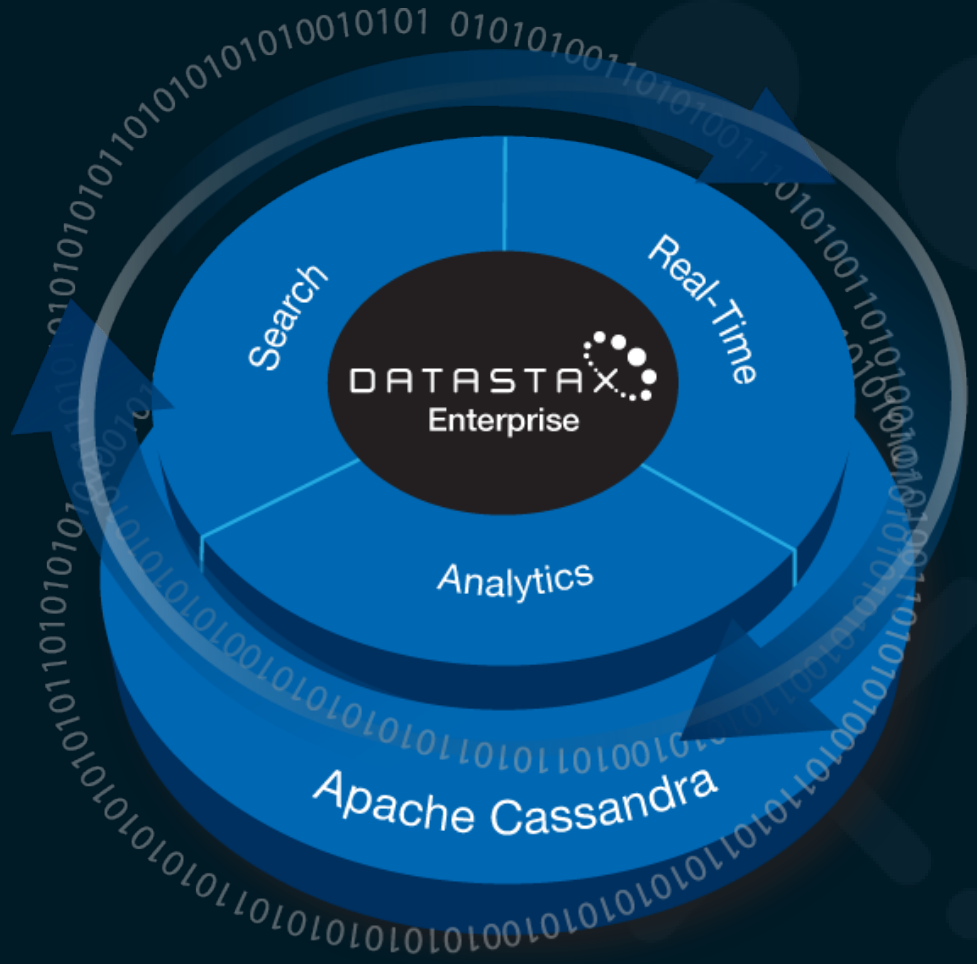
About the Presenter

- Big Data Engineer at DataStax
- Co-author of ‘Programming Hive’ and ‘Lucene and Solr: The Definitive Guide’ from O’Reilly

About DataStax

- The company behind Cassandra
- Sells DataStax Enterprise

DataStax Enterprise 3.x



DataStax Enterprise

- Single stack
- Cassandra
- Solr
- Hadoop
- Consulting
- Support

Cassandra at Netflix

- ZDNet article: “The biggest cloud app of all: Netflix”
- http://zd.net/ZHtrmW
- Built on Cassandra
- “According to Cockcroft, if something goes wrong, Netflix can continue to run the entire service on two out of three zones”

What is Big Data?

- Petabytes of growing data
- Hadoop is for batch work
- What are the solutions for realtime?

What is realtime?

- Near realtime
- 1000 millisecond latency

Why not relational databases?

- Cost of scaling to petabytes
- Physical limitations

Relational to Big Data

- Hadoop for batch
- Solr and Cassandra for realtime
- Gives most of relational capability at 1/10 the cost, scales linearly

Why Cassandra?

- Distributed database heavy lifting
- Simple dynamo model
- Executes replication tasks extremely well

Cassandra vs. HBase

- Cassandra is easier, code is readable
- Fewer moving parts
- Multi-datacenter replication
- Enables low level IO tuning

Cassandra vs. HBase

- HBase runs on HDFS
- HDFS is not designed for random access IO
- Multiple hacks / products to perform random access (MapR, HDFS Jiras)

Cassandra vs. HBase

- Cassandra is peer to peer, there is no single point of failure (SPOF)
- The HDFS name node is a single point of failure

HBase at Facebook

- Most of Facebook runs on MySQL
- Memcache front ends the reads

Batch Analytics

- Hive with Hadoop
- A vague dialect of SQL
- Requires Java for UDFs
- Relational Joins

Realtime Analytics

- Solr
  - SQL features except relational joins
  - Use Hive for relational joins
- CEP (Complex Event Processing)
  - Storm

CEP (Complex Event Processing)

- Storm, computes results on streaming data

Lucene

- Java inverted indexing library
- Text analytics is raw computation over linear sets of data
- High speed computation engine

Inverted Indexes

- Terms dictionary points to list document ids (integers)
- Tokenizes text
- Complete variety of computation on vectors of data

Solr

- Search server built around Lucene
- Adds faceting, distributed search
- Missed the cloud environment features of NoSQL systems for many years

Solr Cloud

- Solr Cloud is a Zookeeper based system
- New and probably not production ready
- Playing catch up

Elastic Search

- High overlap with Solr
- More mature than Solr Cloud
- Less distributed features than Cassandra

Cassandra Concepts

- Columns, column families, keyspaces
- Peer to peer
- Eventual consistency
- Implements basic Google BigTable model

Lucene and Cassandra

- Both implement a log structured merge tree file architecture

DataStax Enterprise with Solr

- Data is stored in Cassandra
- Data placement controlled by Cassandra
- Solr is a secondary index (only)

DataStax Enterprise with Solr

- Separation of church and state, eg, data and index

Indexing

- Indexing is a CPU intensive task
- Not IO bound because of multi-threading
- When a thread is flushing, other threads are indexing, CPU is saturated at all times

Queries

- IO bound, index needs to fit in RAM, then CPU bound
- Lucene enables multithreading queries
- Solr does not multithread queries

DataStax Enterprise with Solr

- Eventual consistency, each node has it's own Lucene index
- Lucene segment files are not replicated (like Solr Cloud and ElasticSearch)

Distributed Search Architecture

- Query requests are round robin'd across nodes automatically

DSE 3.0.1

- 3.0.1 is the current release of DataStax Enterprise

New Features in DSE 3.0.1

- Ease of re-indexing
- Re-index the entire cluster or per-node
- Re-indexing occurs when the Solr schema changes

New Features in DSE 3.0.1

- Solr Cloud requires re-indexing from an external data source such as a relational database

New Features in DSE 3.0.1

- DSE re-indexes directly from Cassandra
- No custom code is required for re-indexing

New Features in DSE 3.0.1

- View the heap memory usage of the field caches
- Perform capacity planning

New Features in DSE 3.0.1

- Multithreaded re-indexing and repair
- Adding a new Solr node is fast

New Features in DSE 3.0.1

- Kerberos and SSL security
- Security audit logging

DataStax Enterprise 3.1

- Near realtime: per-segment filters, facets, multivalue facets
- Solr 4.3

DataStax Enterprise 3.1

- vNodes
- Composite keys

Future

- Multi datacenter live Solr schema updates and re-indexing
- CQL -> Solr queries, makes porting SQL applications easy for SQL developers

Demo of Wikipedia

Real World Example: Tick Data

- Details about every trade
- Tick data generated real time and is quantitatively query-able
- Too big to query on in real time? Not anymore!

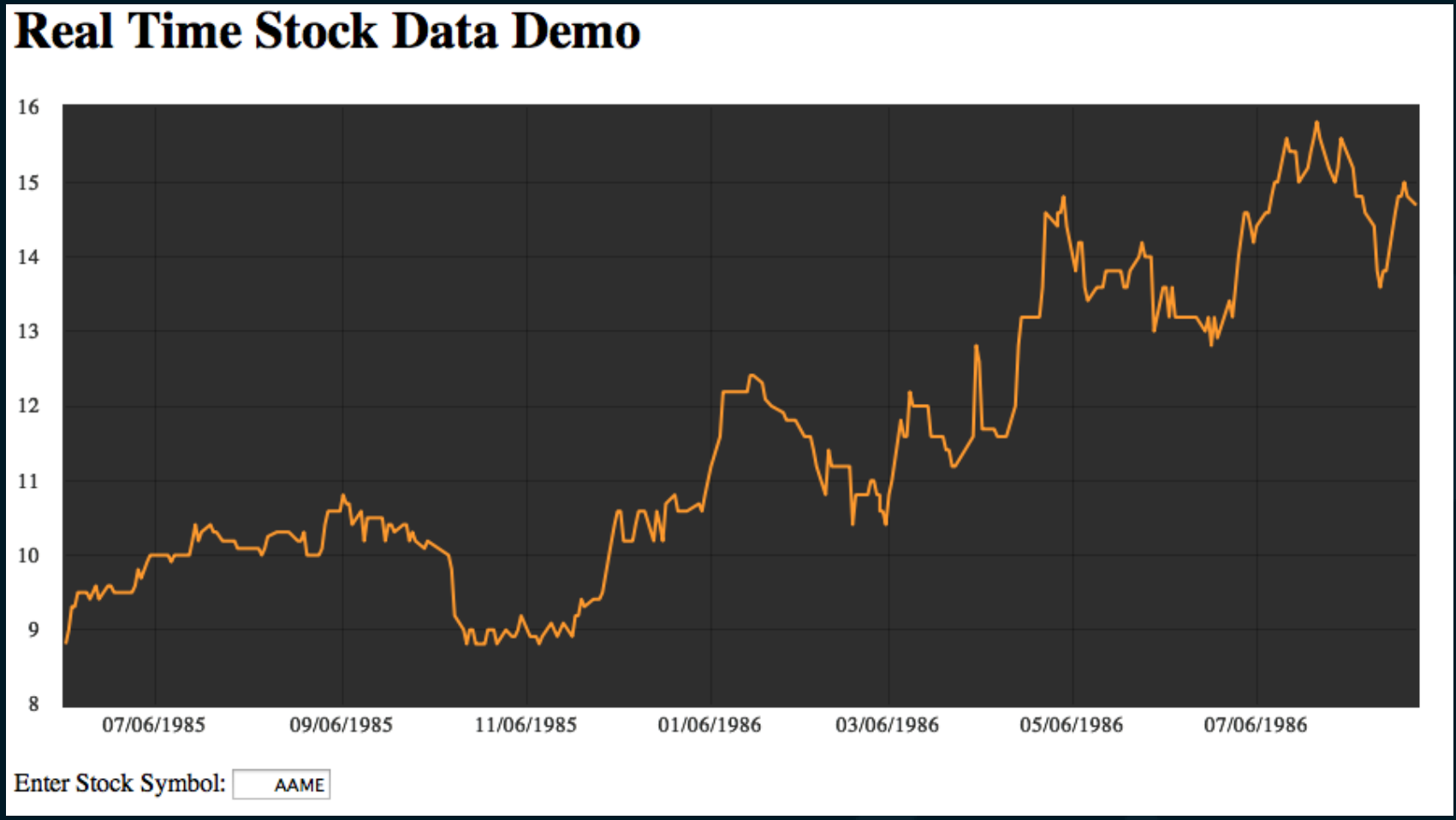
Tick Data - Moving Average

- Computing the moving stock price average in real time
- Comparing multiple moving averages for different stock_symbols
- Requires statistical analysis, group by companies, and faceting features

Tick Data Analytics - Ad Hoc Searches

- Read latest ticks for a given company
- Query ticks for companies in specific verticals during large events such as press releases
- Compute deviation of stock data over 5 years for groups of companies

Real Time Stocks Demo



Solr

General

Schema

- Like an SQL CREATE TABLE statement
- Defines field types
- Defines fields

Solr Config

- XML based configuration options for Solr

Soft Commit

- Commits new index segment to RAM
- Avoids ‘hard’ commit fsync

Auto Soft Commit

```
<!-- The default high-performance update handler -->
<updateHandler class="solr.DirectUpdateHandler2”>
  <autoSoftCommit>
    <maxTime>1000</maxTime> <!– Near Realtime of 1 second -->
  </autoSoftCommit>
</updateHandler>
```

Field Cache

- Loaded for sort and facet queries
- Uses heap space

SolrJ / HTTP

- Java based API for interacting with a Solr server
- DSE supports SolrJ/HTTP with no changes

Insert data with CQL

- Auto data type mapping
- Copy fields
- Dynamic fields

CQL with Solr Query

- Exists however is mainly useful for debugging
- Limited functionality, queries a single node

CQL Insert Example

```
INSERT INTO wikipedia (key, text)
VALUES ('1', 'when in rome')
```

SQL to Solr

How to convert applications

SQL to Solr

- Common to convert existing SQL applications to Big Data
- Focus on the application functionality

SQL to Solr

- Cassandra makes all distributed operations easy

SELECT WHERE

- SELECT * FROM wikipedia WHERE type = ‘pdf’
- q=type:pdf

SELECT columns

- SELECT title,text FROM wikipedia
- q=*:*
- fl=title,text

SELECT COUNT

- SELECT COUNT(*) FROM wikipedia WHERE type = ‘pdf’
- q=type:pdf
- Get the num found

SELECT ORDER BY

- SELECT * FROM stocks ORDER BY price ASC
- q=*:*
- sort=price asc

SELECT AVG

- SELECT AVG(price) FROM stocks
- q=*:*
- stats=true
- stats.field=price
- The average is called ‘mean’ in the Solr results

SELECT AVG GROUP BY

- SELECT AVG(price) FROM stocks GROUP BY symbol
- q=*:*
- stats=true
- stats.field=price
- stats.facet=symbol

SELECT WHERE LIKE

- SELECT * FROM wikipedia WHERE text LIKE ‘rom%’
- q=text:rom*


1010 0001 1100 1010 1010 1010 1010
0100010010100000 0100 10001 101
1010 0001 1100 1010 1010 1010 101010
10001001010000001 0100 10001 101 111
0000111010100010110100010001000
0100101000000 0100 1000101011100