DataStax Enterprise
DSE Search & Analytics

Negib Marhoul
Solutions Engineer
negib.marhoul@datastax.com

Smart Data Movement

Data Write Path, Tiered Storage and DSEFS

DATASTAX

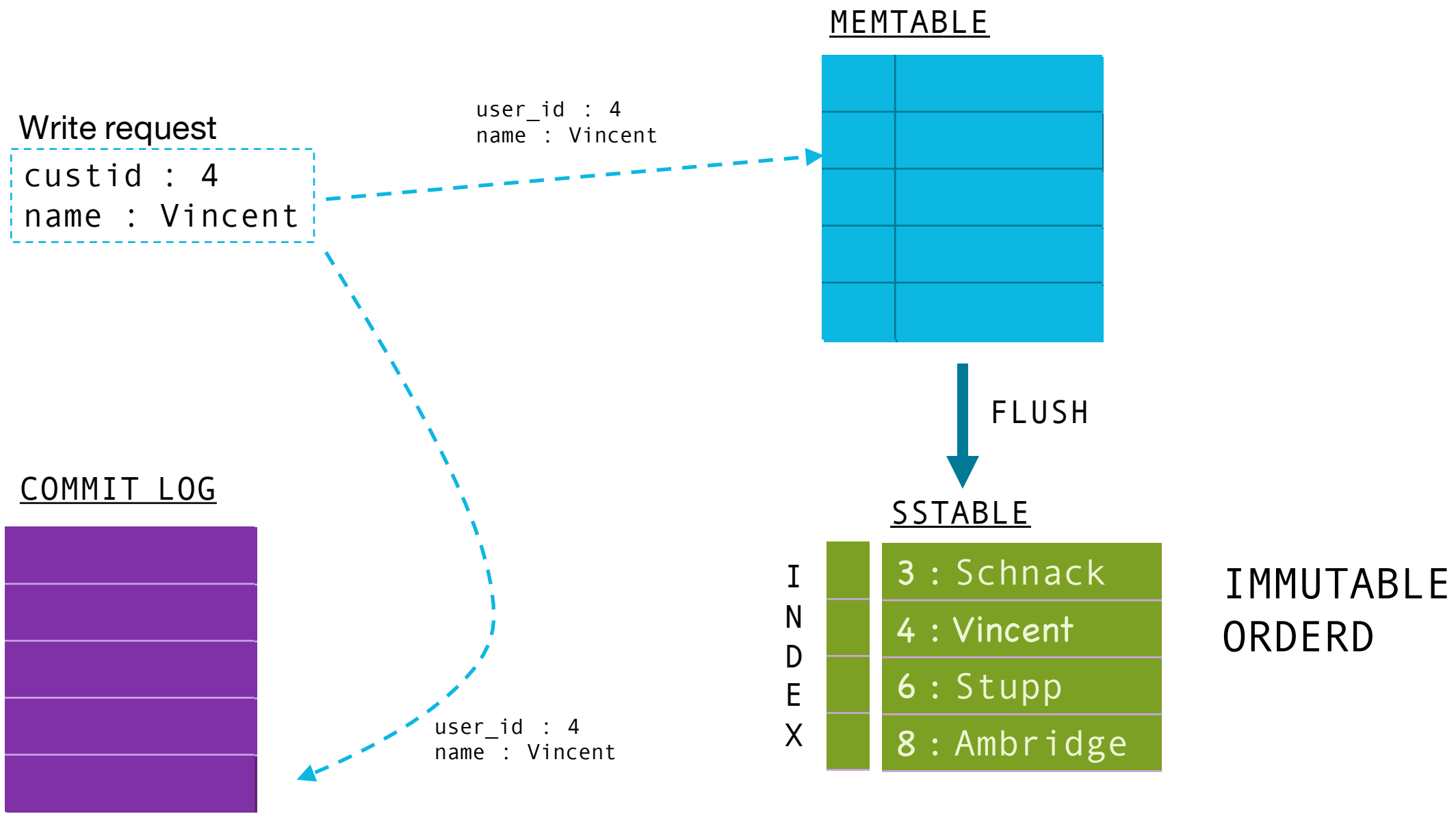
Apache Cassandra™ Architecture

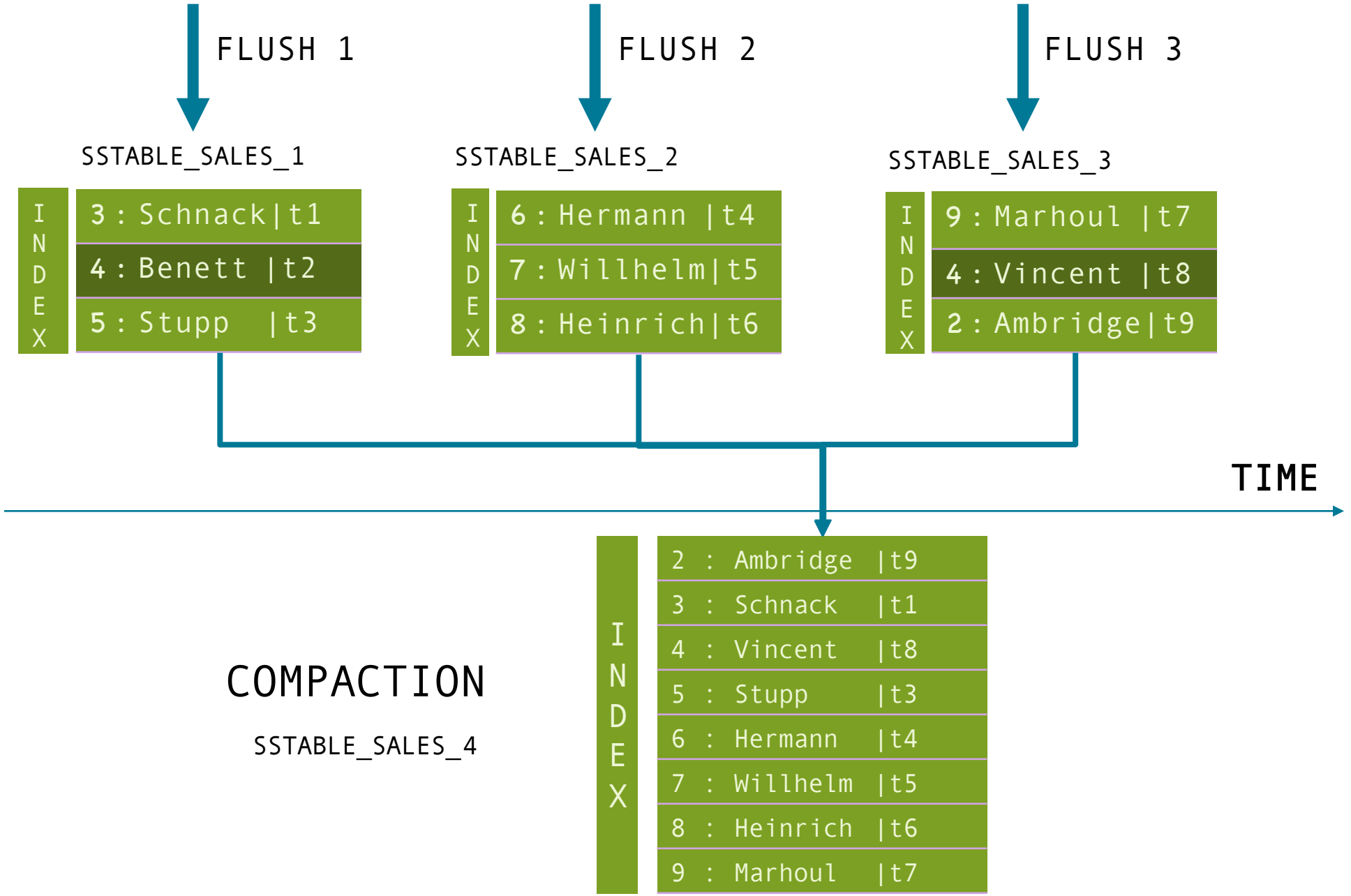
Cluster layer

- Amazon DynamoDB paper
- masterless architecture

Data-store layer

- Google Big Table paper
- Columns / columns family


MEMTABLE
Write request
custid : 4
name : Vincent
user_id : 4
name : Vincent
FLUSH
COMMIT LOG
SSTABLE
INDEX
3 : Schnack
4 : Vincent
6 : Stupp
8 : Ambridge
IMMUTABLE
ORDERD
user_id : 4
name : Vincent


FLUSH 1
FLUSH 2
FLUSH 3
SSTABLE_SALES_1
SSTABLE_SALES_2
SSTABLE_SALES_3
INDEX
3 : Schnack|t1
4 : Benett |t2
5 : Stupp |t3
INDEX
6 : Hermann |t4
7 : Willhelm|t5
8 : Heinrich|t6
INDEX
9 : Marhoul |t7
4 : Vincent |t8
2 : Ambridge|t9
TIME
COMPACTION
SSTABLE_SALES_4
INDEX
2 : Ambridge |t9
3 : Schnack |t1
4 : Vincent |t8
5 : Stupp |t3
6 : Hermann |t4
7 : Willhelm |t5
8 : Heinrich |t6
9 : Marhoul |t7

DATASTAX

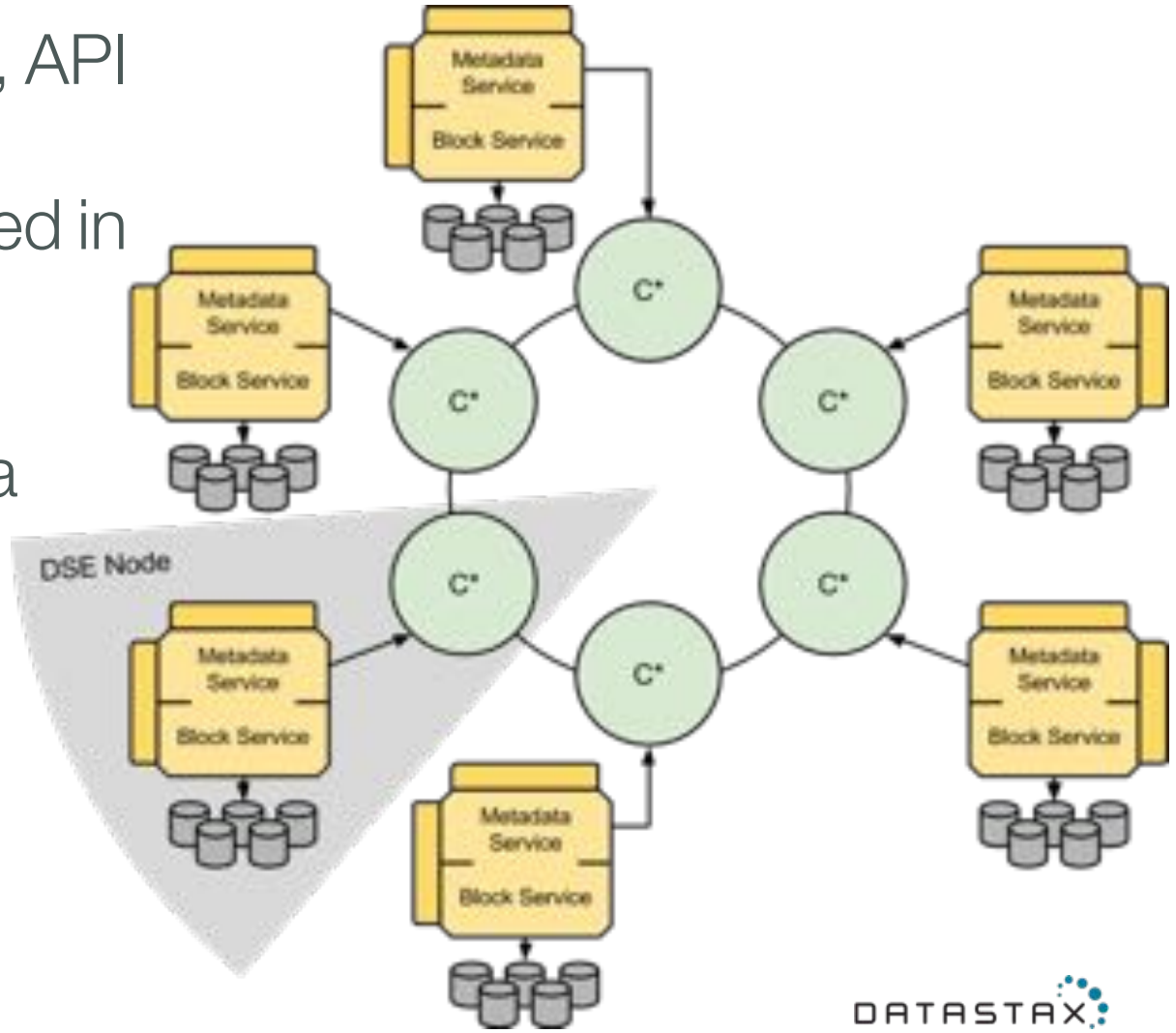
DSE Tiered Storage

- Able to automatically move data to different storage media based on defined criteria.
- Helps reduce storage costs by relegating lesser-used or older data to less expensive storage devices.
- Works on a granular per-row basis.

DSEFS

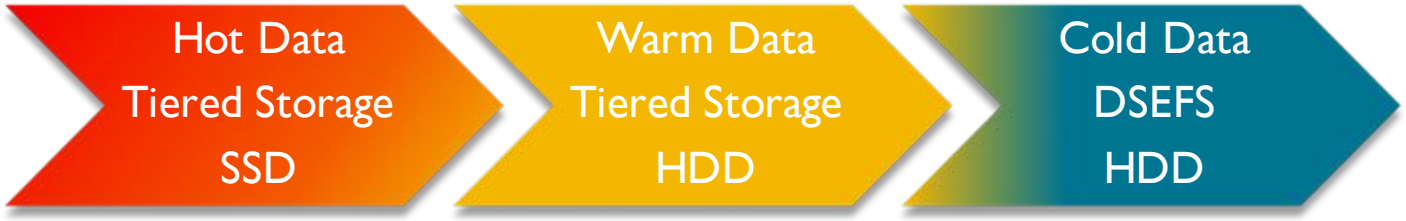
- Distributed file system, masterless, API compatible with HDFS
- Resiliency of metadata, being stored in Cassandra tables

- Cost effective, cold storage of data
- Staging
- Archiving
- Analytics (with Spark)



Storage Temperature Strategy

- Business value of data record per byte is low in IoT use cases
- Being able to optimize the cost of storage depending on the usage of the data is key
- A tiering / temperature management approach is a relevant response
  - **Hot Fast** storage using SSD for fresh data
  - Warm cost effective storage using HDD for older data (in-DB online archive)
  - Cold cheapest storage using file system for long term data (out-DB archive)
    - Can be used with Spark for analytical usages

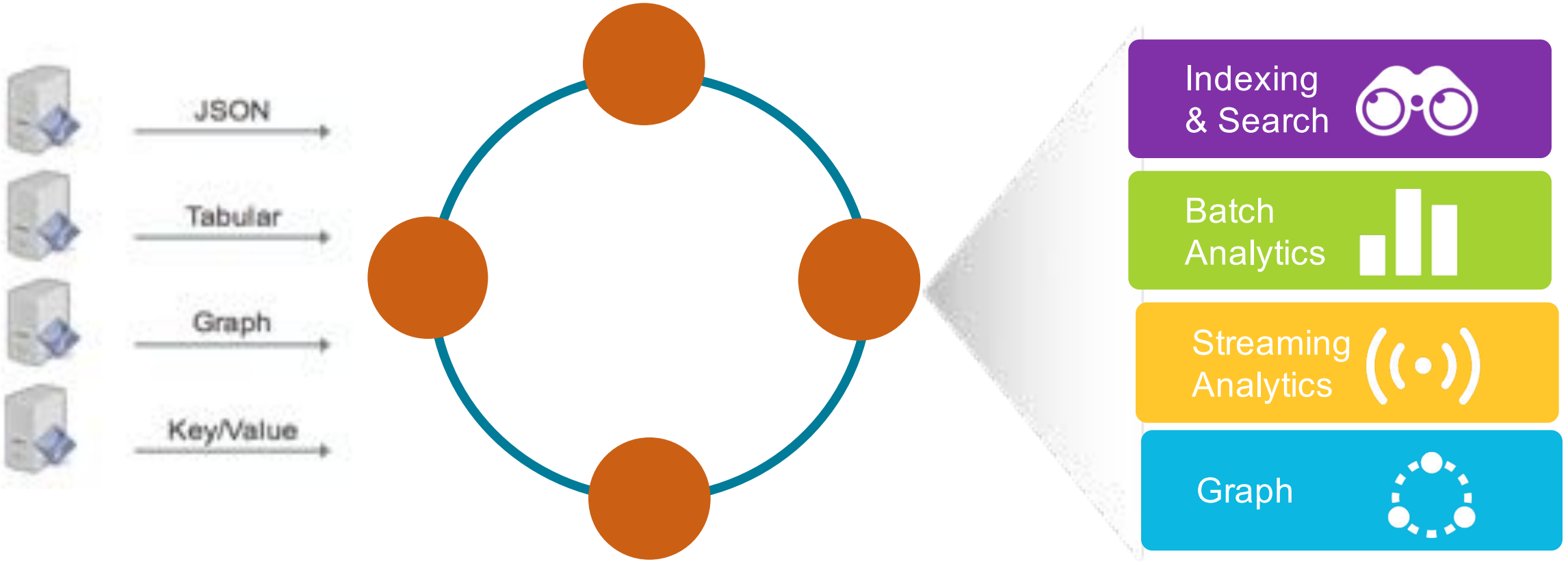


Information Search

Query by non-PK

DATASTAX

Integrated Multi-Model Platform



Information Search

Multi-criteria WHERE Constraints

- WHERE constraints with multiple columns
- No extra tables needed

Full Text Search

- **Wildcards** ? *, like or Lemmatisation
- **Faceting**, Slice and Dice

Live Indexing

- Real Time Search, High Index throughput

Geospatial queries

- Queries with coordinates and distances search

https://docs.datastax.com/en/dse/5.1/dse-dev/datastax_enterprise/search/tutorialsDemosTOC.html

https://github.com/phact/docker-silk-dse

DATASTAX

Query by non-PK

```
CREATE Search INDEX on notifications WITH COLUMNS activity;

SELECT * FROM notifications WHERE 'activity' LIKE 'cre%';
```

Range Queries (with open bounds possible)

```
SELECT * FROM taxi_trips
  WHERE 'pickup_dropoff_range' > '2017-02-01T12'
  AND  'pickup_dropoff_range' < '2017-02-02';
```

Filter operators, such as LIKE, IS NOT NULL, range, and =/!= are supported!

Geo-Spatial Search

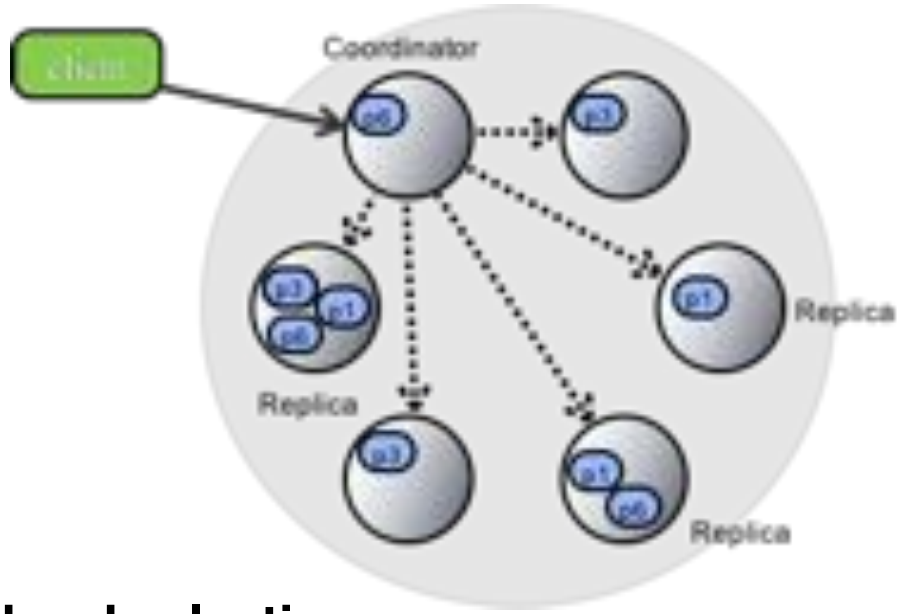
```
SELECT * FROM test WHERE solr_query=
'{"q":"*:*", "fq":"point:\"IsWithin(BUFFER(POINT(4.0 49.0), 10.0))\""}';
```

Querying through CQL

```
SELECT * FROM notifications WHERE 'activity' LIKE 'cre%';

activity
------------------------------
```

mike created account
tom created account

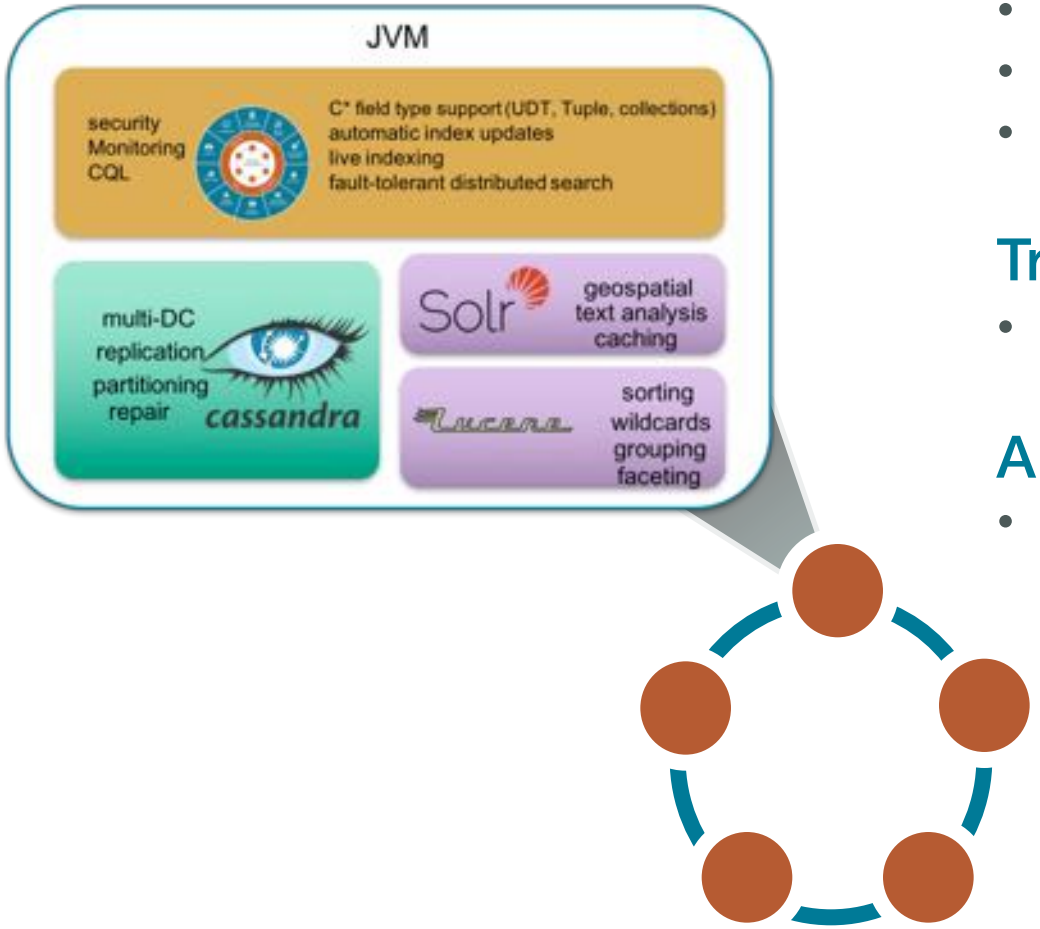


How many nodes to contact?

- We don't know the primary key
- Theory:
  - contact at least one replica for every token range

- Cassandra contacts all nodes
- **Our custom Solr SearchComponent does intelligent shard selection**

DSE Search Architektur



Integrated in the the same JVM

- Data Locality (TokenRanges) and Shared Memory
- High Available, no master needed
- Index is sharded to all nodes

Transparency

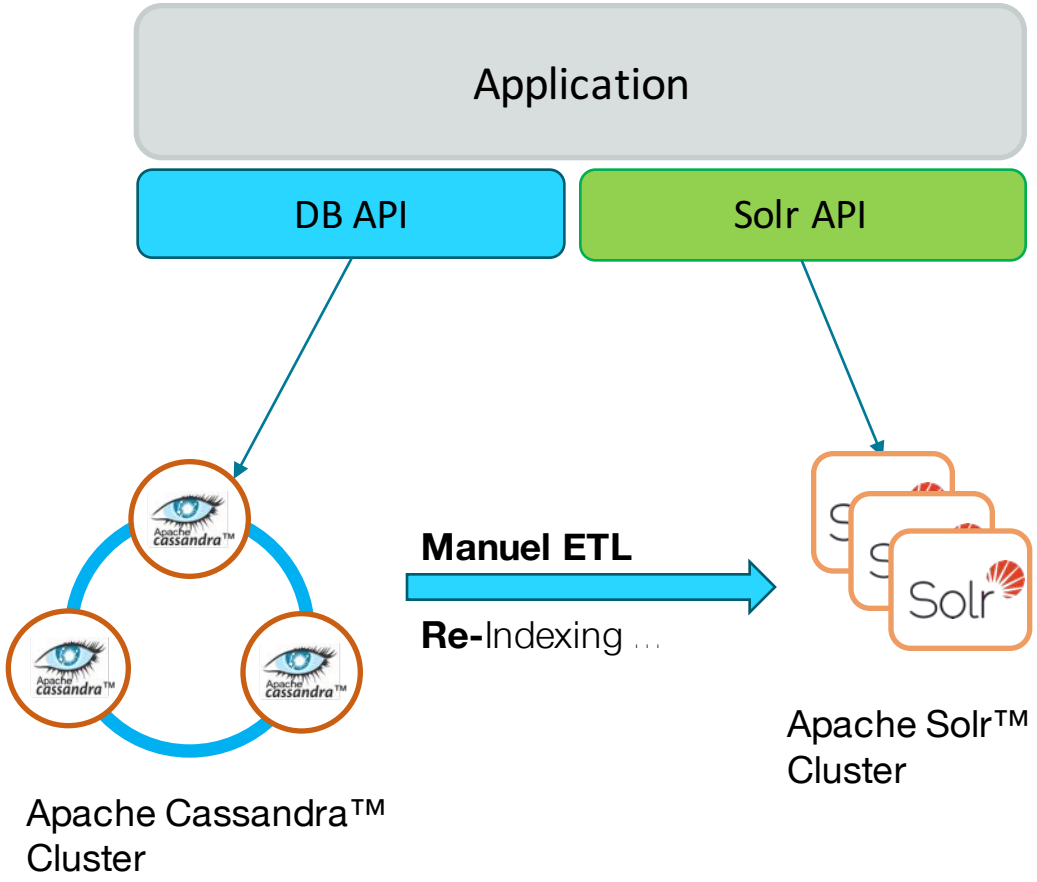
- Access via CQL or REST API

Automated indexing with INSERT / UPDATE

- No extra ETL Process needed

DATASTAX

The Open Source Way



Separated Search Cluster

- "Split Brain" risk, data inconsistency
- ETL to generate, update and re-create index

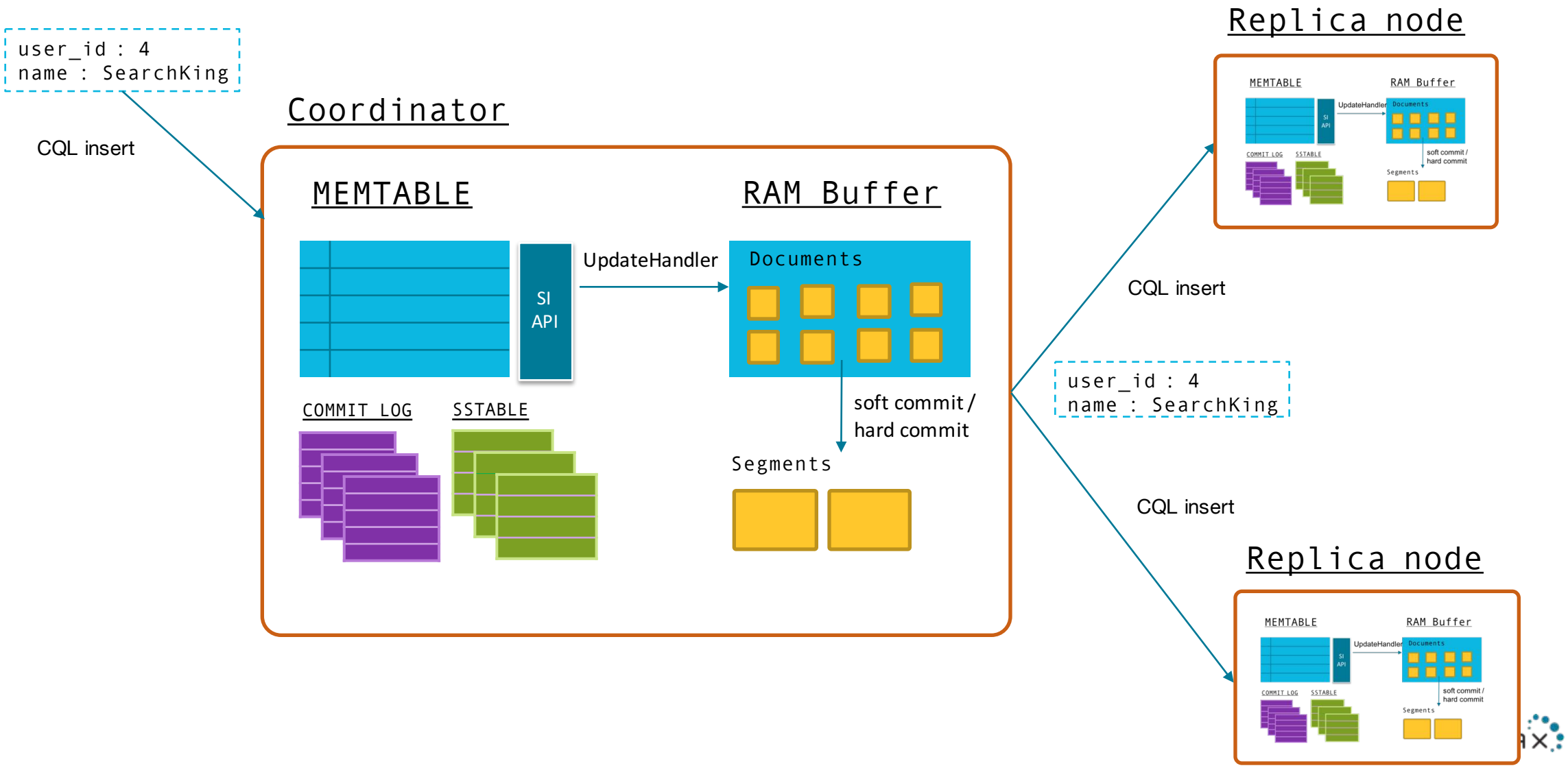
Complex application

Two seperated APIs, driver, read pathes

https://docs.datastax.com/en/datastax_enterprise/5.0/datastax_enterprise/srch/searchOssSolrDiff.html

DATASTAX

Inserting with DSE Search enabled



Advanced Data Analytics

DSE Analytics, AlwaysOn-SQL

DATASTAX

Advanced Analytics with Datastax Enterprise

- DataStax Enterprise integrates real-time and batch operational analytics capabilities with an **enhanced version of Apache Spark™**
- Enables ad-hoc reporting
- Personalization
- Predictive Analytics
- Process real-time streams of data

DATASTAX

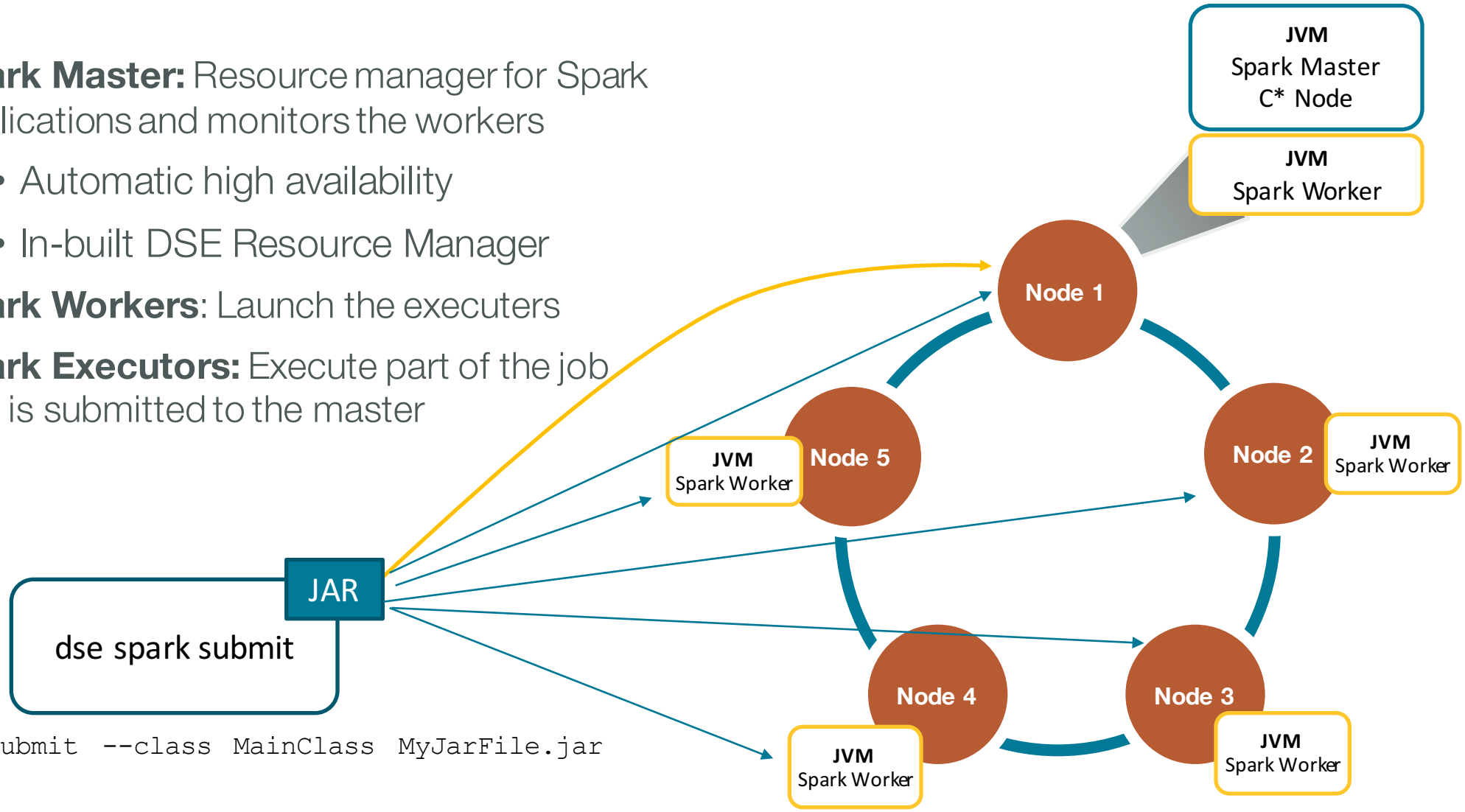
Leveraging Apache Spark through DataStax Enterprise Analytics

- Data model independent queries
- Cross-table operations (JOIN, UNION, etc.)
- Complex analytics (e.g. machine learning)
- Data transformation, aggregation, etc.
- Stream processing

DATASTAX

DSE Analytics Platform

- **Spark Master:** Resource manager for Spark applications and monitors the workers
  - Automatic high availability
  - In-built DSE Resource Manager
- **Spark Workers**: Launch the executers
- **Spark Executors:** Execute part of the job that is submitted to the master




```
dse spark-submit --class MainClass MyJarFile.jar
```

DATASTAX

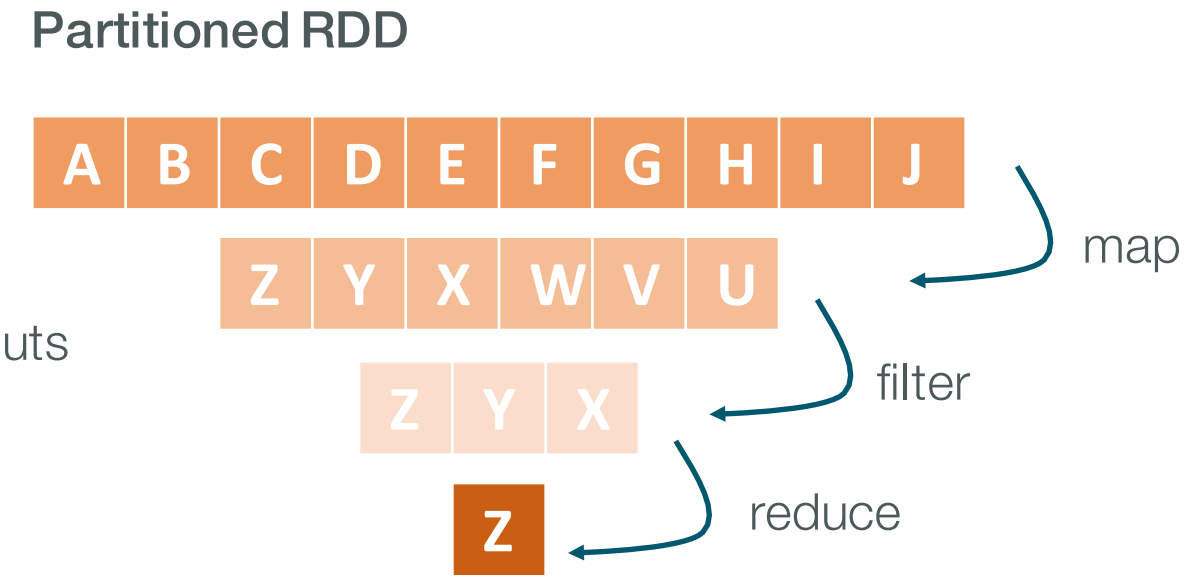
Other use cases than statistical reporting and ETL

- **Manufacturing & Internet of Things**: Real-time, adaptive analysis of machine data (e.g., sensors, control parameters, alarms, notifications, maintenance logs, and imaging results) from industrial systems (e.g., equipment, plant, fleet) for visibility into asset health, proactive maintenance planning, and optimized operations.
- **Marketing & Sales**: Analysis of customer engagement and conversion, powering real-time recommendations while customers are still on the site or in the store
- **Customer Service & Billing**: Analysis of contact center interactions, enabling accurate remote trouble shooting before expensive field technicians are dispatched
- **Information Technology**: Log processing to detect unusual events occurring in stream(s) of data, so that IT can take remedial action before service quality degrades
- **Risk Management**: Anomaly detection and root cause forensics, enabling fraud to be stopped while it happens

Spark data model RDD

- RDD = Resilient Distributed Dataset
- A collection with following qualities:
- immutable
- iterable
- serializable
- distributed
- parallel
- lazy

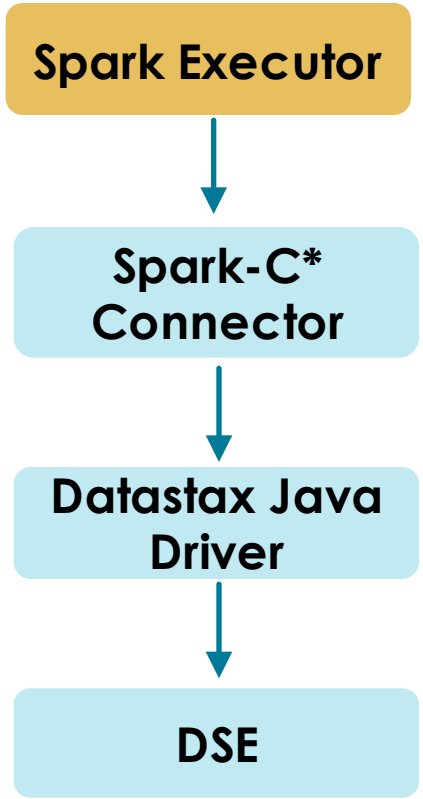
Immutable In and Outputs



Transformations are state less

Database Access with DataStax Driver

- DataStax Cassandra Spark driver
  - Implemented mostly in Scala
  - Scala + Java APIs
  - Does automatic type conversions




```
// Spark connection options
val conf = new SparkConf(true)
     .set("spark.cassandra.connection.host", "127.0.0.1")
     .set("spark.cassandra.auth.username", "cassandra")
     .set("spark.cassandra.auth.password", "cassandra")
val sc = new SparkContext("spark://127.0.0.1:7077", "myapp", conf)


// Read from DSE and add partitioner with primary key
val rdd = sc.cassandraTable("my_keyspace", "my_table").byKey("pk","cc")


// Save to DSE
rdd.saveToCassandra("my_keyspace", "my_table", SomeColumns("key", "value"))
```

DATASTAX

RDD – Resilient Distributed Dataset

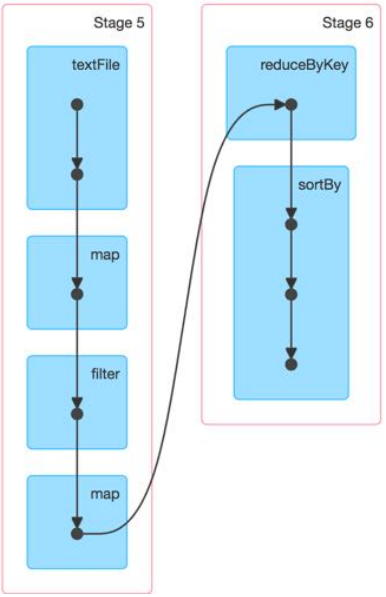
- Sparks RDD is the Data abstraction layer for the distributed data processing
- RDDs are **stateless , immutable and partitioned data collection**s, which are distributed across many machines (cluster)

```
val myfile = sc.textFile("/de_zip_codes.csv“);
```

Partition 1	Partition 2	Partition 3	Partition 4
93336,Bayern,Eichstätt93349,Bayern,Eichstätt84405,Bayern,Erding84416,Bayern,Erding84424,Bayern,Erding	36037,Hessen,Fulda 36039,Hessen,Fulda 36041,Hessen,Fulda 35759,Hessen,Lahn-Dill-Kreis, 35764, Hessen,Lahn-Dill-Kreis35767	83607,Bayern,Miesbach83624,Bayern,Miesbach83626,Bayern,Miesbach83627,Bayern,Miesbach83629, Bayern,Miesbach83666,Bayern,Miesbach	92726,Bayern,Neustadta.d.Waldnaab92727,Bayern,Neustadta.d.Waldnaab92729,Bayern,Neustadta.d.Waldnaab95

```
myfile.filter(_._3 == "Hessen")
     .map(record => (record._5,1)).
     .reduceByKey( (a,b) => (a + b)).
     .sortBy(-_._2).
     .take(10)
```

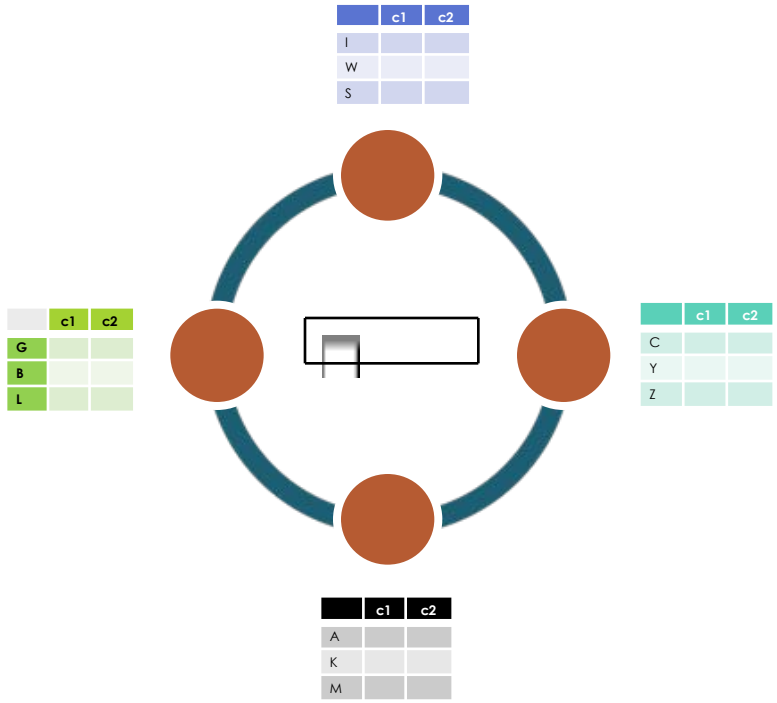
Directed Acyclic Graph (DAG)



- **Resilience** : Spark's RDDs dependencies address fault tolerance by using a **lineage graph**
- **Lazy Evaluation**: transformations performed on RDDs without actually spending compute time on them
- **RDD functions and data structure are opaque** to Spark => general-purpose compute engine

DATASTAX

Data Locality

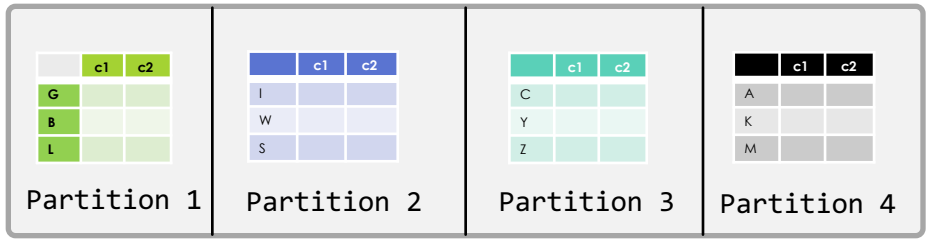


- DSE Analytics respects data locality
- No need for ETL between separated clusters
- Spark Master HA

Every Spark task uses a CQL-like query to fetch data for a given token range:

```
SELECT "key", "value"FROM "keyspace"."table“
    WHERE
        token("key") >  384023840238403 AND
        token("key") <= 38402992849280
ALLOW FILTERING
```

In Memory: Distributed on all available nodes

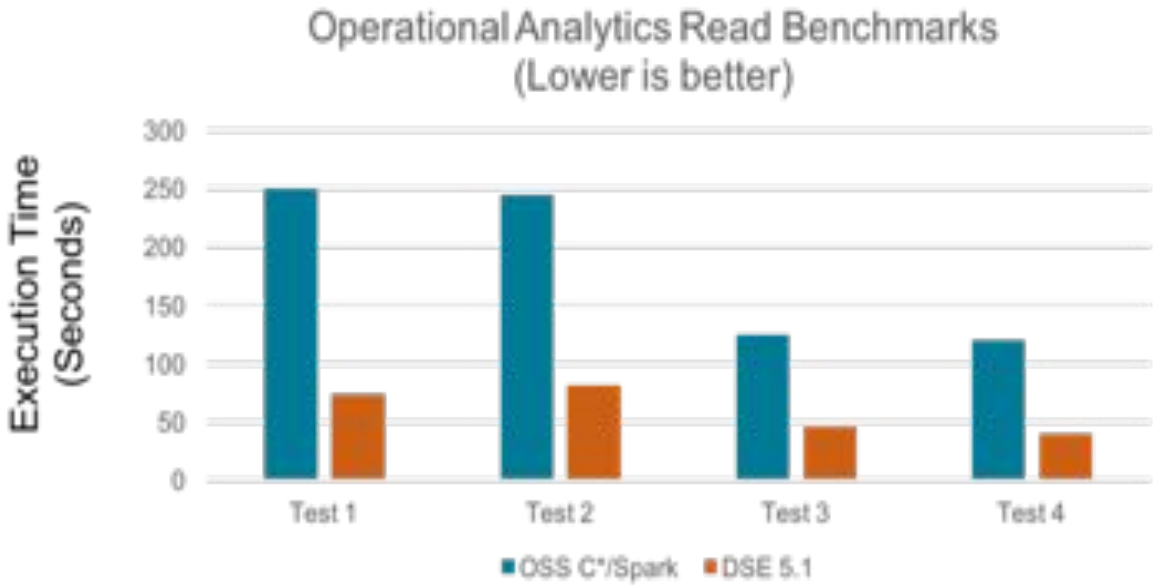


DATASTAX

Improving Spark Read Performance with Continuous Paging

- Beneficial for large scan queries e.g. analytical queries
- Any operation that has reading from Apache Cassandra™ as it's bottleneck will benefit from continuous paging
- The default is disabled **`spark.dse.continuous_paging_enabled`**
- The server continuously prepare new result pages in response to a query
- No cycle of communication between the DSE Server and DSE Java Driver
- DataStax Enterprise feature only

Test: selecting all columns or some columns, with or without a clustering-column predicate, and we see a 2.5 – 3.5x performance improvement



https://www.datastax.com/dev/blog/dse-continuous-paging-tuning-and-support-guide

DATASTAX

Pushdown Predicate and Integration with DSE Search

- SearchAnalytics mode allow you to create analytics queries that use DSE Search indexes
- improves performance by reducing the amount of data that is processed

Push Down Predicate with DSE Search solr query

```
val table = sc.cassandraTable("states_statistics","de_zip_code")
val result = table.select("zip","city")
.where("solr_query='cite:He*'")
.take(10)
```

Push Down Predicate with DataFrames

```
val table1 = spark.read.cassandraFormat("department","hr"))
.load()
```

DATASTAX

DataFrames and Datasets

- Higher Level API of structured distributed data
- DataSets are structured, typesafe objects
- DataFrames equivalent to tables in relational DBs
- Uses Query optimizations and predicate pushdown
- Enables better optimizations through Spark
- Faster serialization and less memory consuming

Scala query

```
spark.table("zip").
     filter("state = 'Hessen'").
     groupBy("city").
     count().
     orderBy(desc("count")).
     limit(10).show()
```

SQL Query

```
spark.sql("select count(zip) z, city c
          FROM zip
          WHERE State = 'Hessen'
          GROUP BY city
          ORDER BY z desc
          LIMIT 10").show()
```

Apache Spark @Scale: A 60 TB+ production use case

https://code.facebook.com/posts/1671373793181703/apache-spark-scale-a-60-tb-production-use-case/

Common question: How can we migrate data to DSE?

- Files
- DSEFS
- Cassandra
- Kafka
- JDBC/ODBC
- DataStax Enterprise

```
val departments = sqlContext.read.format("jdbc")
     .option("url", "jdbc:oracle:thin:hr/hr@localhost:1521/orcl")
     .option("driver", "oracle.jdbc.OracleDriver")
     .option("dbtable", "departments")
     .option("partitionColumn", "DEPARTMENT_ID")
     .option("numPartitions", "4")
     .load()
```

JDBC/ODBC Example : https://github.com/simonambridge/Oracle_to_Cassandra

Writing to Cassandra

```
departments.write.format("org.apache.spark.sql.cassandra")
     .options(Map( "table" -> "department", "keyspace" -> "hr"))
     .save()
```

DATASTAX

Common Issues

Too few partitions gives less concurrency

- Need “reasonable number” of partitions
  - Lower bound: At least ~2x number of cores in cluster
  - Upper bound: Ensure tasks take at least 100ms

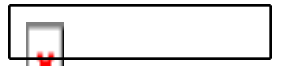
1. Ensure enough partitions for concurrency, **check #task in the Spark UI** or `rdd.partions.size`
   1. spark.cassandra.input.split.size
   2. spark.storage.memoryFraction
2. Minimize memory consumption (esp. of sorting and large keys in groupBys)

Use the CassandraPartioner to avoid shuffling

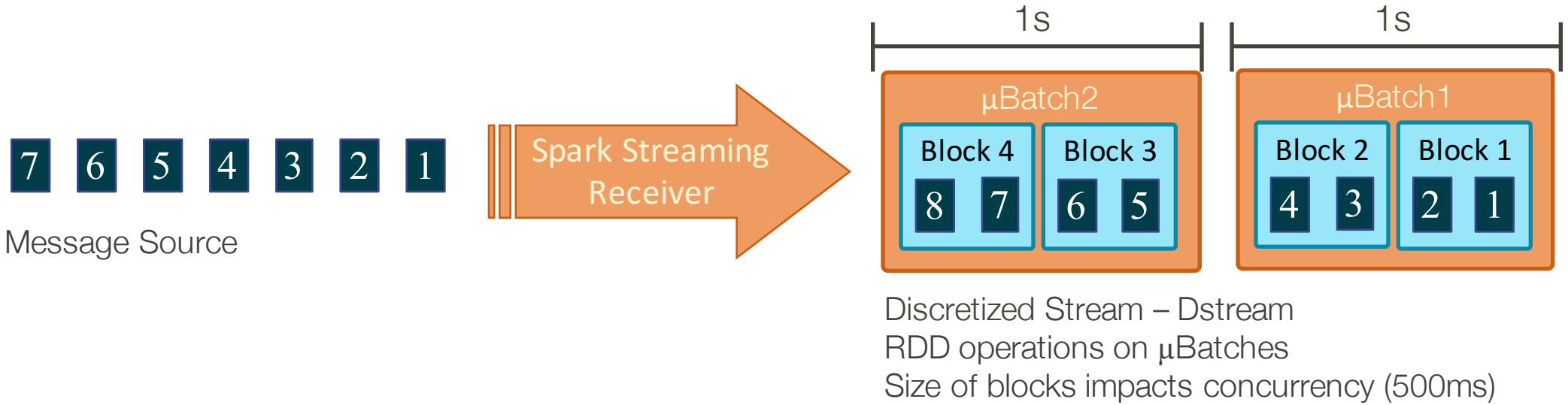
https://github.com/datastax/spark-cassandra-connector/blob/master/doc/FAQ.md

Troubleshooting Spark

https://www.datastax.com/dev/blog/common-spark-troubleshooting



Data Stream Processing at your Fingertips



Discretized Stream – Dstream
RDD operations on μBatches
Size of blocks impacts concurrency (500ms)

Ingest Data from different sources:

- DSEFS
- S3
- Kafka
- MQTT
- Flume

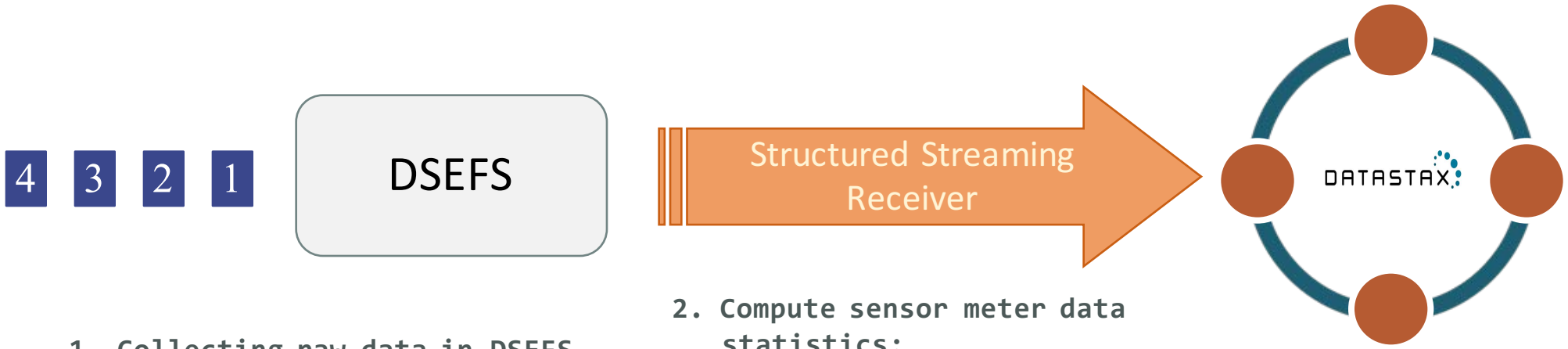
```
val ssc = new StreamingContext(sc, Seconds(1))
val lines = ssc.textFileStream(args(0))
val readingStream = lines.map(_.split(",")).
map(reading => (reading(0),1)).
reduceByKey( (a,b) => (a + b))
```

Structured Stream processing at your Fingertips

- Spark structured streaming API is, it's just DataFrames
- Continuous query: Unifies streaming, interactive and batch queries
- Spark will automatically figure out how to incrementally process only what has arrived since the last time we did some processing
- Online training of a model
- Event time, windowing, sessions, sources & sinks

DATASTAX

Demo with DSEFS source



1. Collecting raw data in DSEFS

2. Compute sensor meter data statistics:
   - Cumulative count
   - Average
   - Min
   - max

3. Write time series raw data and aggregated statistics to DSE

DEMO

Use Cases

DATASTAX

DataStax Helps Macquarie Deliver Amazing Customer Experience

THE CHALLENGE: Drive digital transformation initiatives to enhance customer experience.

- Macquarie uses **DSE** to drive their **customer experience** initiative.
- Consolidated data from many disparate systems delivers **360º, real-time** customer visibility.
- Their world-class consumer banking app utilizes **real-time analytics** and **full text search.**
- Customers now have **better insight** into how they spent their money and where.
- Transformed from no retail presence to a digital consumer banking juggernaut in less than 2 years.

MACQUARIE

GE PROVIDES PREDICTIVE MAINTENANCE INNOVATION WITH DATASTAX

THE CHALLENGE:

Collect sensor data from millions of devices from around the world and manage trillions of transactions per day.

- GE offers the first Industrial Cloud Platform called **GE Predix.** Datastax is part of the data services layer within the platform.
- DSE will collect sensor data from millions of devices from around the world to help GE provide **predictive maintenance** to their customers and increase **operational efficiencies**.
- Predix manages **trillions of transactions** per day. DSE was recognized as the only solution that could support this scale and data center replication.

WALMART DELIVERS THE NEXT GENERATION OF CUSTOMER EXPERIENCE WITH DATASTAX

THE CHALLENGE: Transform its strategy to avoid downtime on Black Friday and prevent food spoilage in grocery section of stores.

- Walmart serves nearly 260 million customers each week via their eCommerce websites and apps and 11,504 stores in 28 countries. With revenue of $486 billion in 2015, Walmart is transforming their strategy to deliver the next generation of customer experience.

- Product Catalog - DSE stores 100s of millions of Walmart products in their ecommerce site, and has helped them **avoid downtime for four Black Fridays in a row** since migrating from Oracle.
- IOT - DSE tracks chiller data to **prevent food spoilage**.

Walmart

DMC – DataStax Managed Cloud

White Glove Management Service

DATASTAX

Introducing DataStax Managed Cloud

A Fully Managed, Secure Architecture

- DSE on AWS for production workloads
- 24x7x365 coverage, lights-out management
  – by the engineering and support experts at mar
- Optimized for your apps
- Architecture advisory services, guidance, and be

Onboarding

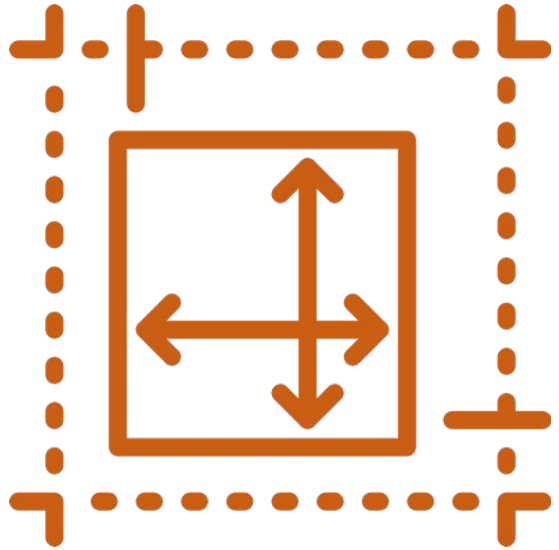
Ramping you up for take-off

- DataStax Solution Architect
  - Orients you to the cloud console and platform
  - Reviews your apps for platform, data model, and access patterns
- Cluster
  - Tested and sized for right instances
  - Production provisioned
  - Alerting configured

Provisioning

Custom fit

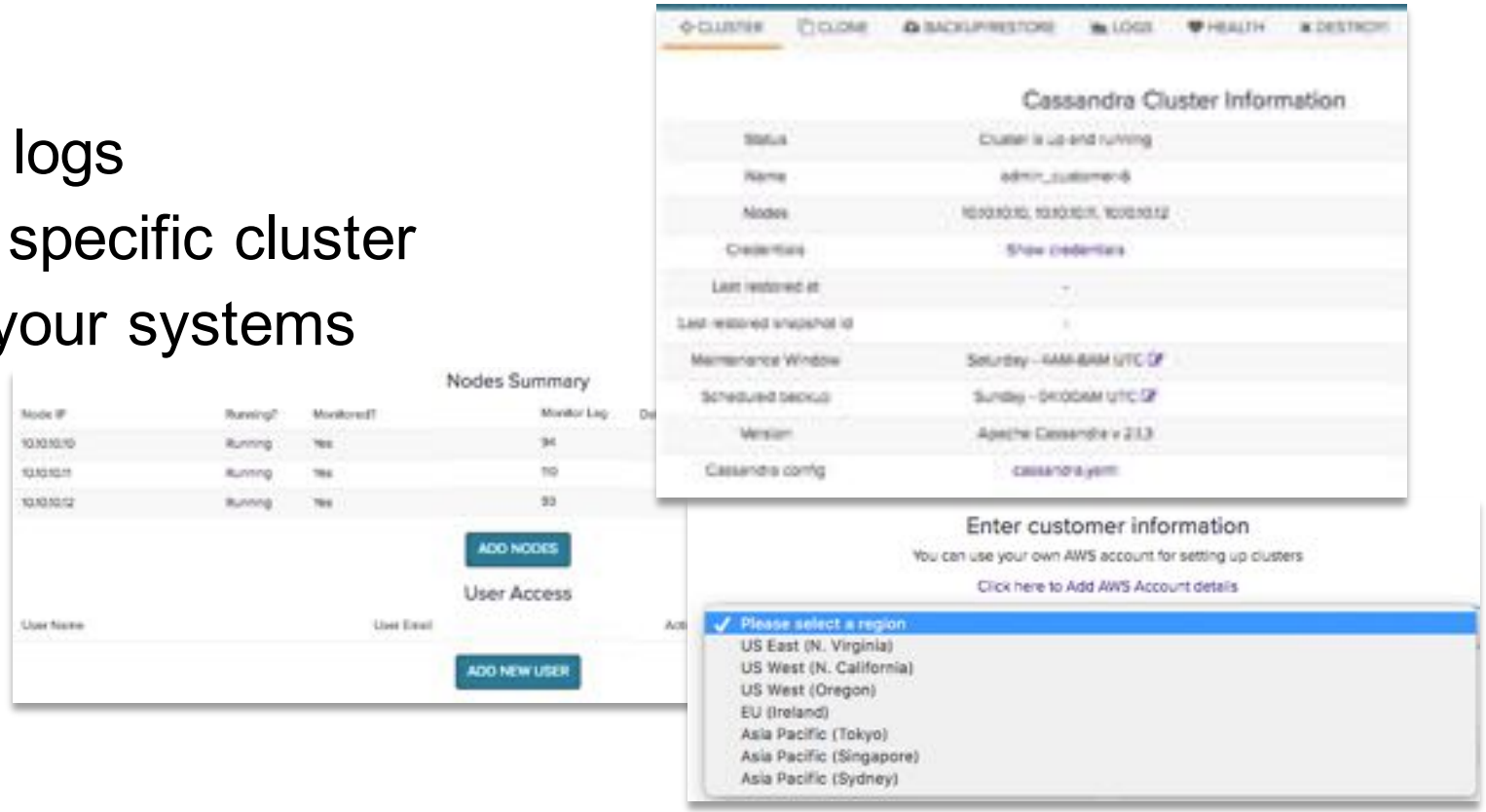
- Provisioning into your VPC
  - Environment setup, installation, and configuration
- Production
  - Best-practices blueprints
  - Configured for optimal performance
  - Tuned to your apps, lifecycle-managed
- Non-production (optional)
  - Ad-hoc, provisioning by end-user and lifecycle management via console or API



Cloud Console

Your “home base” to view your cluster

- View cluster metrics and logs
- Easily submit a ticket on specific cluster
- Console integrates with your systems via RESTful API

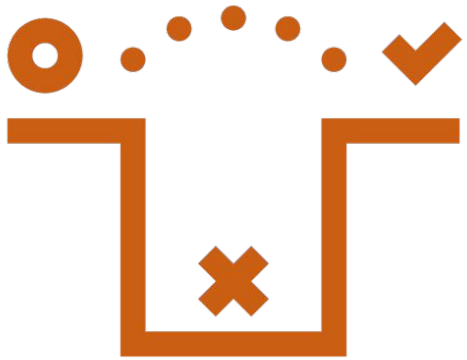


DATASTAX

Proactive intervention/alerting

Finding problems – before they exist

- Automated alerts against exceeded thresholds
  - Based on years of expertise
- Avoiding issues through
  - Automated fixes
  - Engineer action
- Fault-tolerance at multiple levels
  - Intervention still required for recovery – and handled for you
- Bursting included



Backups & restores

Restore with confidence

- Protect against
  - Data loss due to application bugs
  - Operator error
- Backups moved to cloud object storage for easy recovery
- Scheduled restore exercises – and restore on request

Upgrades support/service requests

Easy upgrades

- Aggressive bug fixes, if needed
  - Leveraging DataStax engineering for fast time to resolution
- The latest and greatest features – that fit your schedule

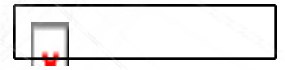
DATASTAX

Negib Marhoul
Solutions Engineer
negib.marhoul@datastax.com

Thank you

Günther Schnack
Enterprise Sales Manager
PLZ: 7 – 9 & 5
+44 79 49 448 969
gunther.schnack@datastax.com

Jerome Ruiz
Regional Channel Director
jerome.ruiz@datastax.com



Negib Marhoul
Solutions Engineer
negib.marhoul@datastax.com

Thank you

Giscard Venn
Enterprise Sales Manager
PLZ: 0 - 4 & 6
+49 173 4516618
giscard.venn@datastax.com

Jerome Ruiz
Regional Channel Director
jerome.ruiz@datastax.com