DataStax Enterprise
DSE Core Architecture & Modelling

Negib Marhoul
Solutions Engineer
negib.marhoul@datastax.com

DATASTAX
The power behind the moment.

Agenda

1	Topology and Architecture
2	Tuneable Consistency
3	Query Based Modelling

Design for failure and nothing will fail

Apache Cassandra™ Architecture

Cluster layer

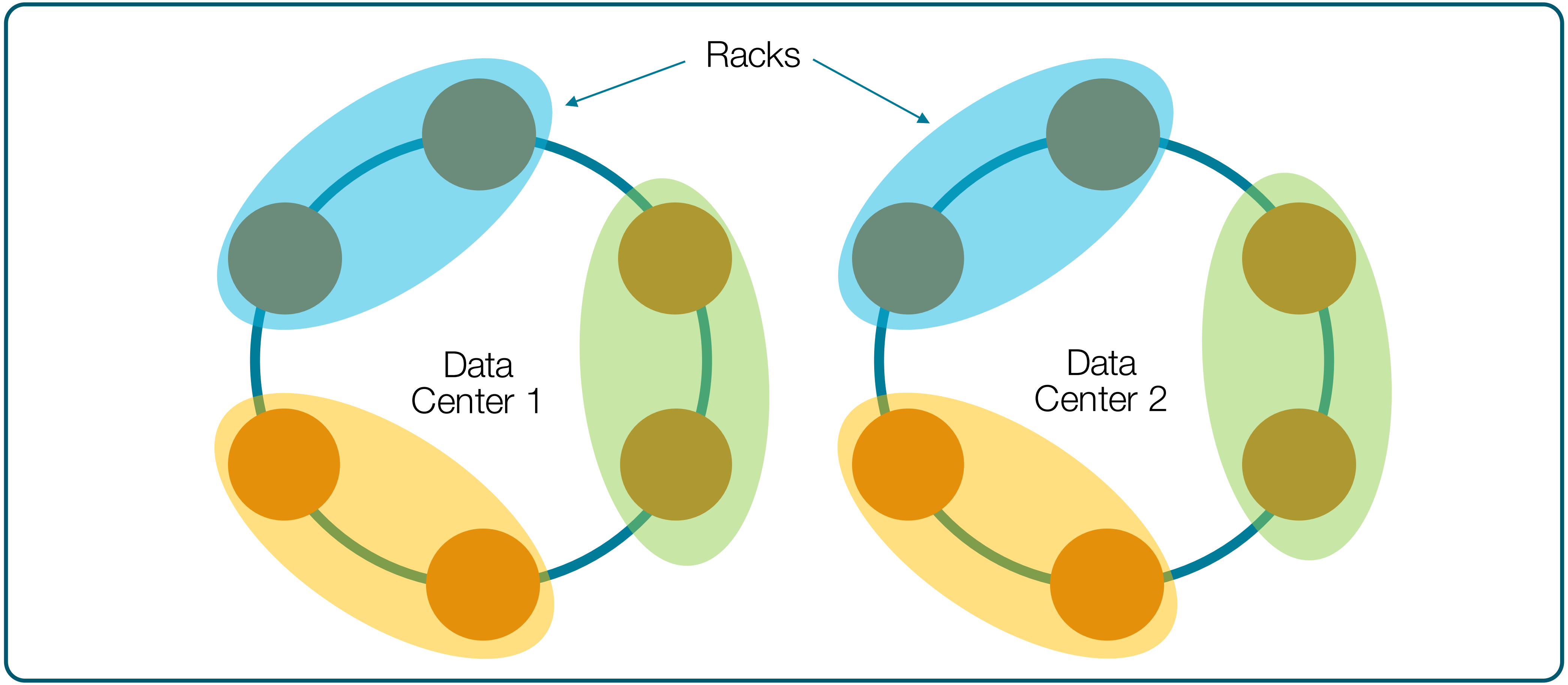
- Amazon DynamoDB paper
- masterless architecture

Data-store layer

- Google Big Table paper
- Columns / columns family

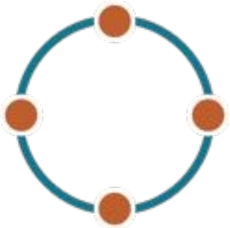
Topology

Cluster

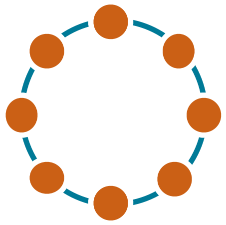


Master-less Always-On, Scalable, Distributed

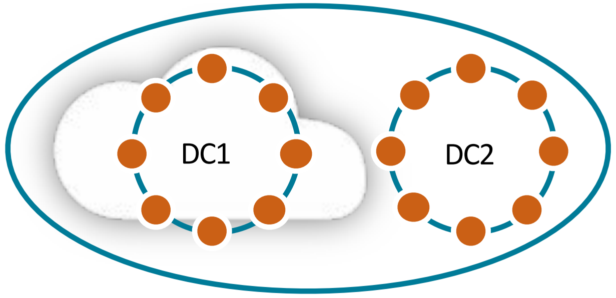
Continues Availability
- No master, Master Less
- Topology discovery
- Client topology awareness


AlwaysOn

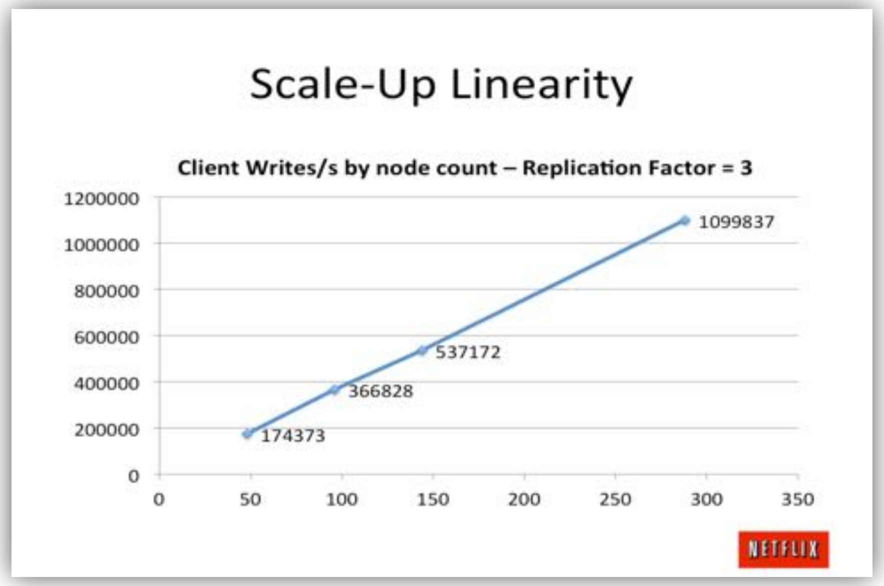
Linear Scalability
- Scale out
- tunable consistency
- Runs on Commodity hardware


Linear Scalability

Built-in data distribution
- Shared-Nothing Architecture
- coordination free
- Automatic data distribution

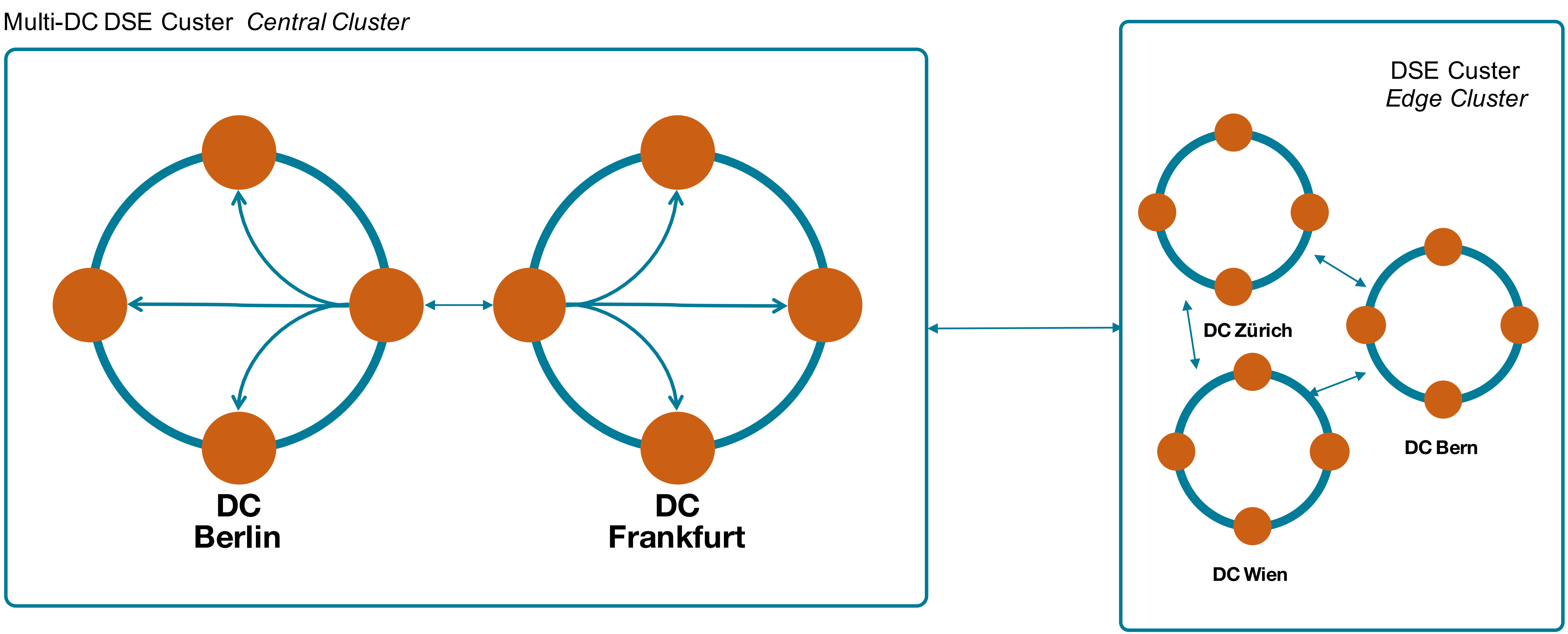


On Premise, Cloud or Hybrid



http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html

Distributed Cluster Or Hub’n Spoke Architecture

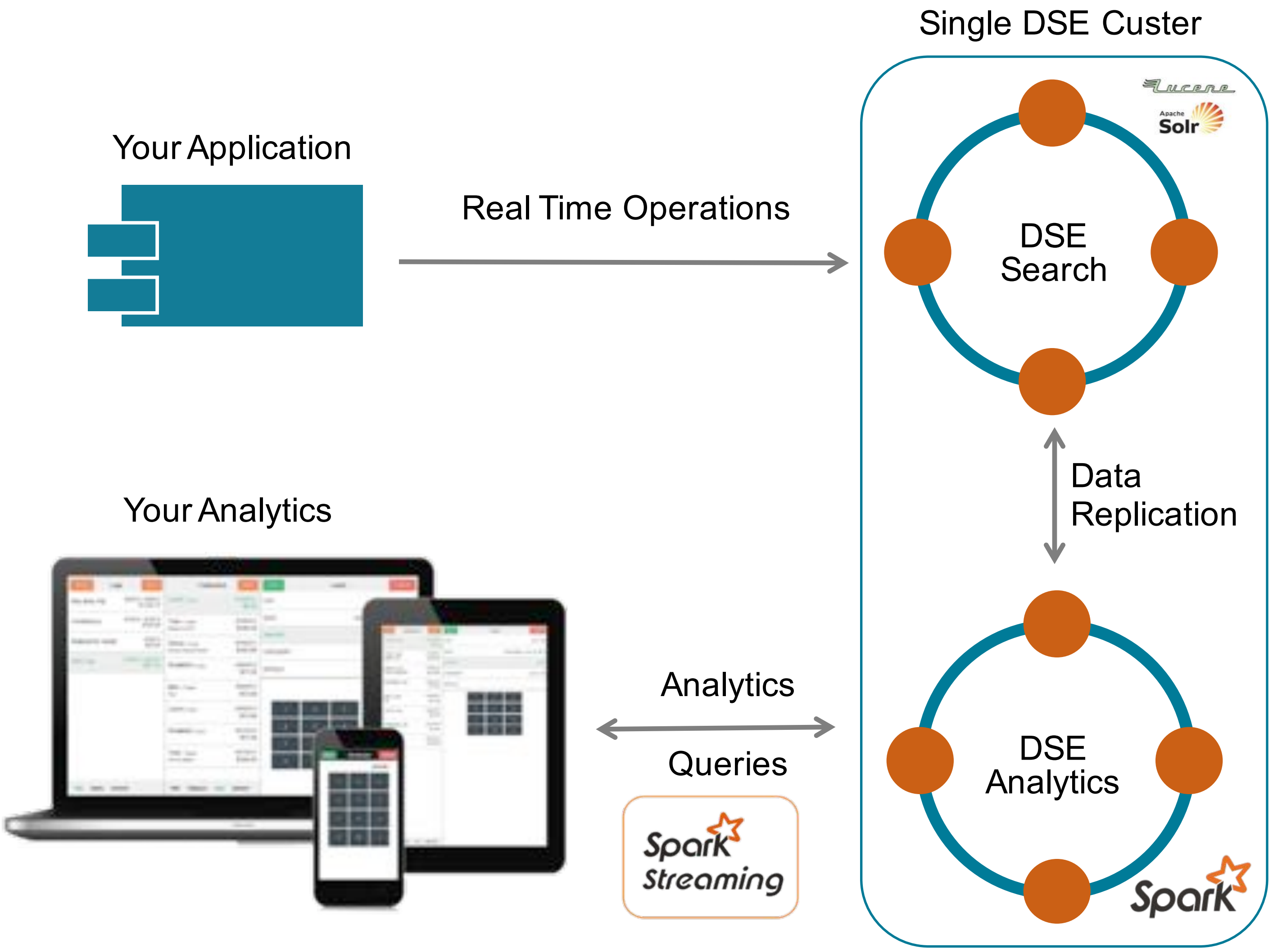


What is Advanced Replication

Advanced Replication supports:

- Many edge clusters replicating to a central hub
- Central hub replicating out to many edge clusters
- Consistent or sporadic connectivity – “store and forward”
- Active queries at the edge, as well as replicating to the hub
- Filter rules about which data is forwarded
- Prioritized streams for limited bandwidth situations

DSE Reference Architecture



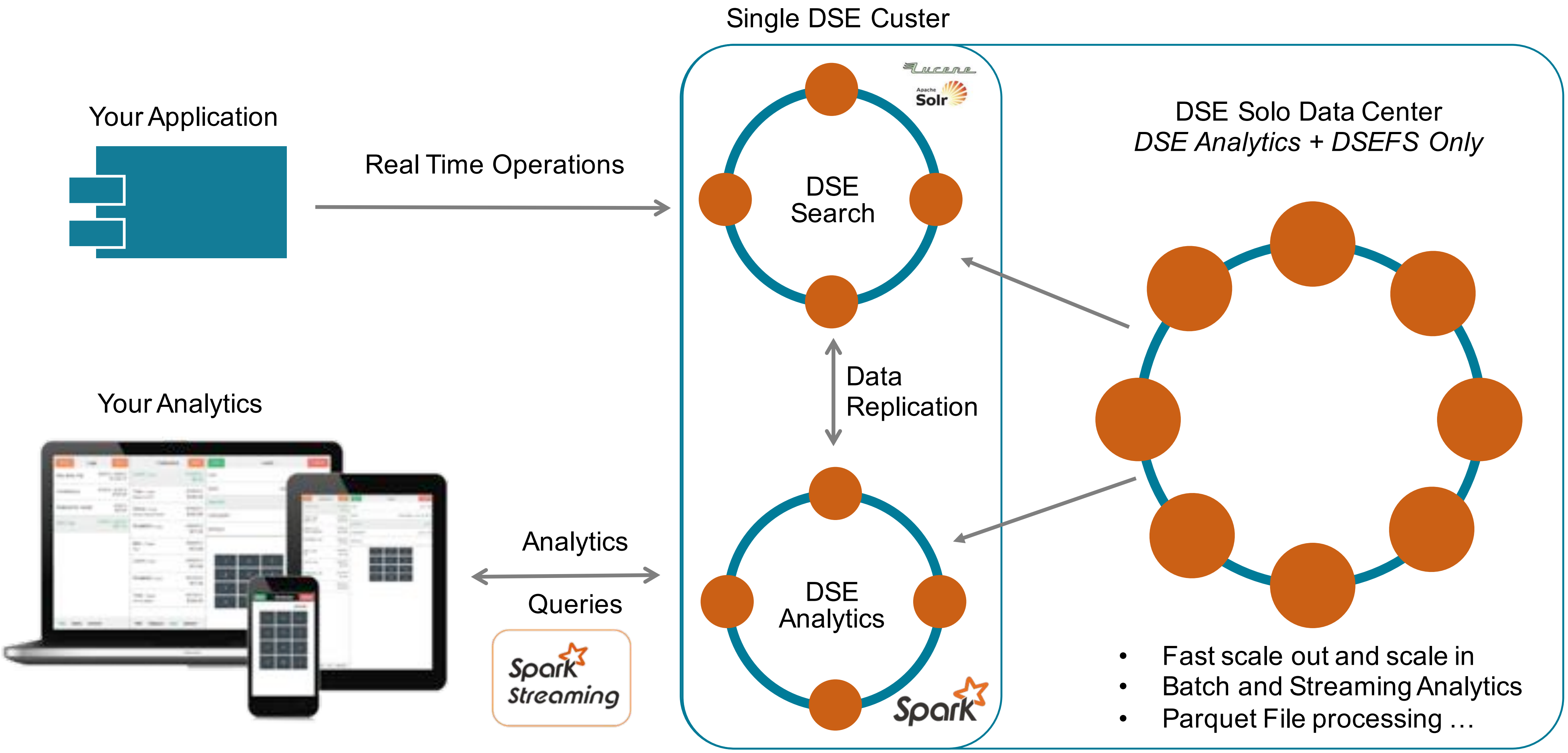
Streaming, ad-hoc, and batch

- High-performance
- Workload management
- SQL reporting

Compared to self-managed:

- No ETL
- True HA without Zookeeper

DSE Reference Architecture

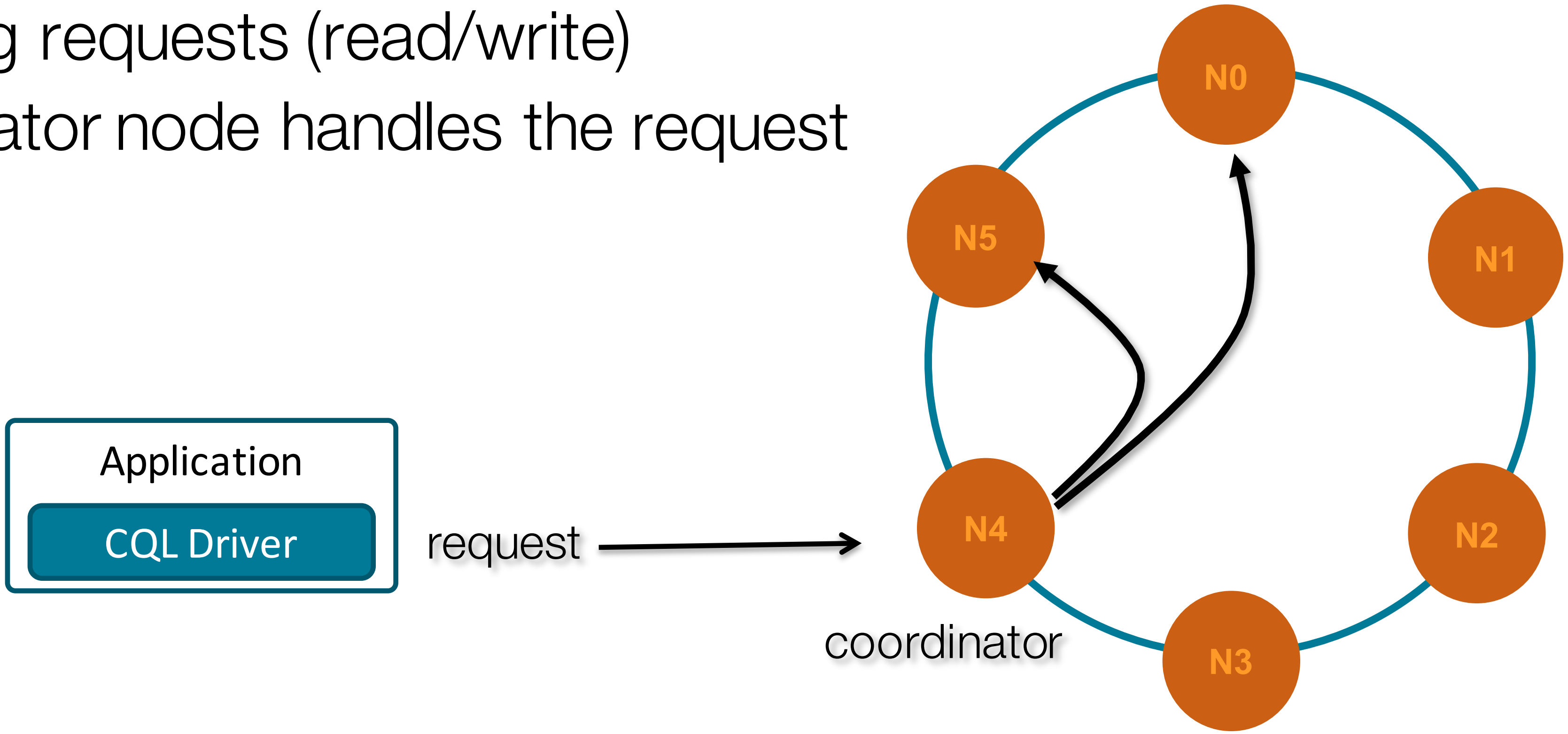


Tunable Consistency

Distributed Read and Write

Data Replication

- Incoming requests (read/write)
- Coordinator node handles the request




```
CREATE KEYSPACE dsbank WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1': '3'}
```

Every node can be coordinator → masterless

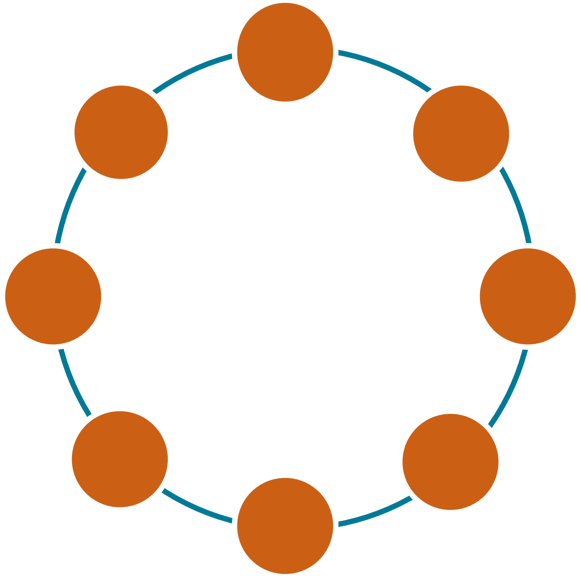
Tunable consistency

- Choose between strong and eventual consistency depending on the need
- Can be done on a per-operation basis, and for both reads and writes
- Handles multi-data center operations
- Light Weight transaction = ACID like

Client

Reads — Consistency Level — Writes

- ONE
- LOCAL_QUORUM
- LOCAL_ONE
- QUORUM
- ALL



Anti-Entropy and Consistency

Write time

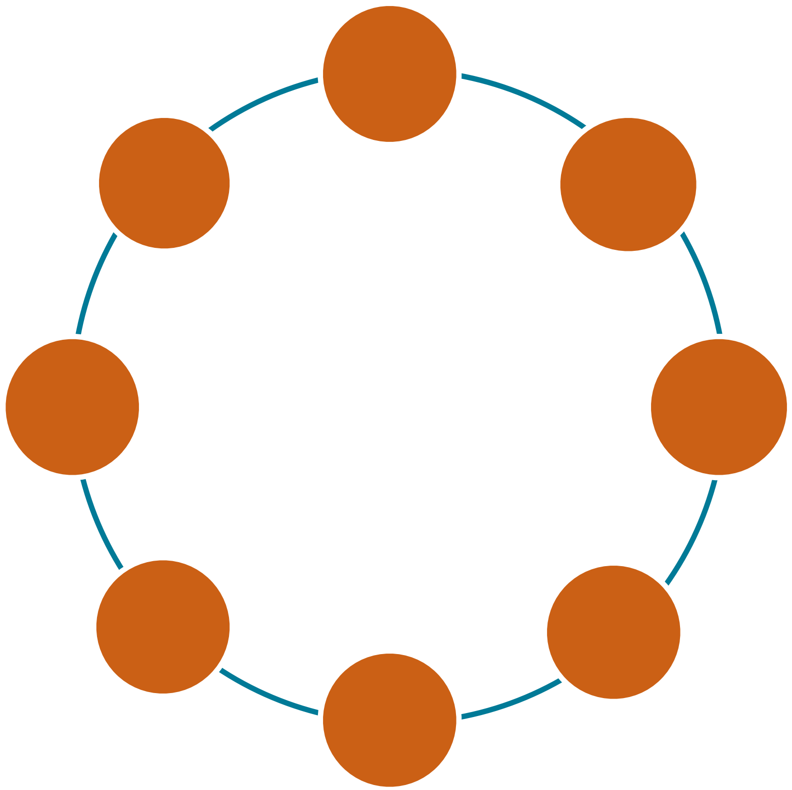
- Tunable Consistency
- Atomic batches
- Hinted handoff

Read Time

- Consistent reads
- Read Repair

Maintanance Time

- Node repair



Lab 1 : Accessing the cluster

Uniform Data Distribution

Query Based Modelling

Primary data model

Name	Column
Value	

Partition	Name	Name	Name	...	Row
	Value	Value	Value	...	

Partition	Wide Row Column		Wide Row Column		...		...		...		...		Wide Row
	Name	Name	Name	Name	Name	Name	Name	Name	Name	Name	Name	Name	
	Value	Value	Value	Value	Value	Value	Value	Value	Value	Value	Value	Value	

- Row-oriented, column structure
- Table: similar to an RDBMS table but more flexible/dynamic
  - A row in a table is indexed by its key

Modelling explicit partitioning

```
CREATE TABLE messaging.notifications(
    target_user text,
    notification_id timeuuid,
    notification_time timestamp,
    ...
    activity text,
    PRIMARY KEY (target_user, notification_time)
) WITH CLUSTERING ORDER BY (notification_id DESC)
```

Partition Key

Clustering Column

Query Optimized

```
SELECT * FROM notification where target_user =‘mike’ limit 1;

SELECT * FROM notification where target_user =‘mike’ AND notification_time >= 2017-11-01 10:00;
```

DATASTAX

Primary Key – Unique Identifier

```
PRIMARY KEY ((account_number), transaction_time)
```

Partition Key

– Required to satisfy a queries' predicate(s)
– Ensures row uniqueness
– Defines the location of the partition in the cluster
  • Hashed to ensure even data distribution
– Can be composed of multiple columns
  • "Composite / Compound Key"

Clustering Key

– Sorts data within each partition
  Defaults to ascending order
– Can Be composed of multiple columns

Modelling explicit partitioning

target_user	notification_id	notification_time	activity
nick	5321998c	2017-11-01 10:00	tom liked
nick	ea1c5d35	2017-11-02 11:00	jake commented
nick	321998c	2017-11-03 09:00	mike created account
mike	e1bd2bcb	2017-11-01 07:00	tom created account

```
SELECT * FROM notification where
target_user ='nick' AND
target_user ='mike' AND
notification_time >= 2017-11-01 07:00;
```

```
Sorted by
notification_time
```

nick	notification_time: 2017-11-01 10:00		notification_time: 2017-11-02 11:00		notification_time: 2017-11-03 09:00		....	notification_time: 2017-12-31 23:00	
	ntfcid: 5321998c	activity: tom liked	ntfcid: ea1c5d35	activity: jake commented	ntfcid: 5321998c	activity: mike created account			

Wide Row

```
Merged, Sorted and Stored Sequentially
```

Number cells per partition limited to 2 billion
Column max 2 GB , better 1 MB
Blob size 2 GB less then 1 MB recommended

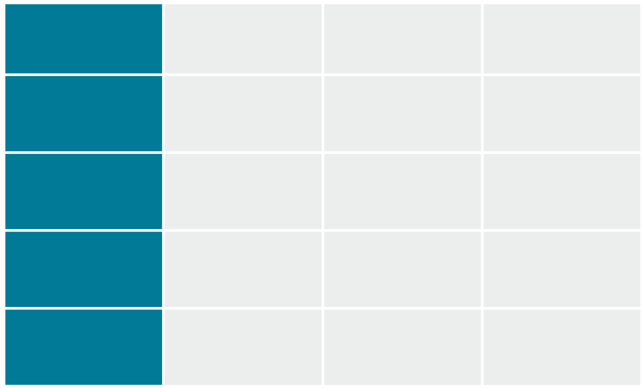
Tokens

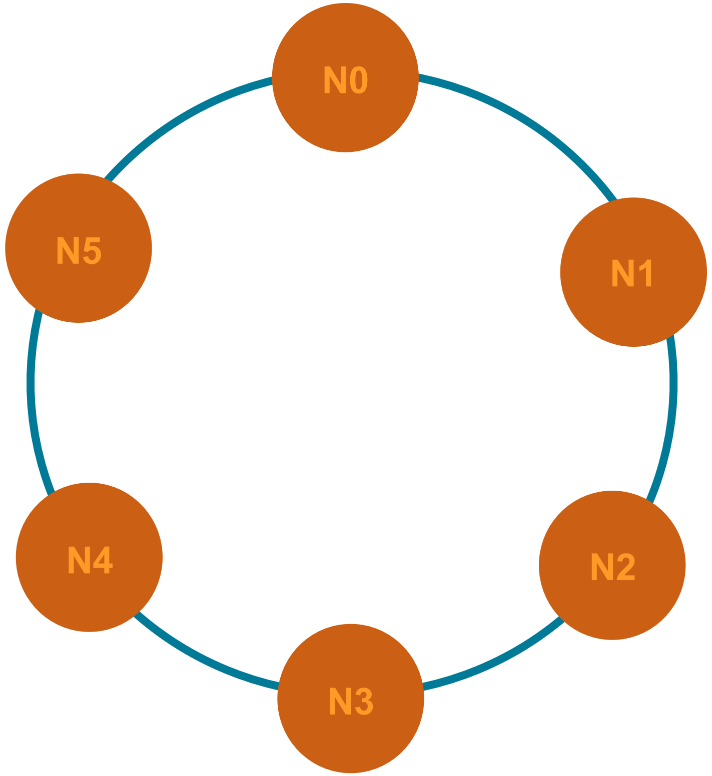
Token Range : - 2^{63} to 2^{63}

Data is partitioned after its partition key

A unique token is allocated to a partition

Token = *random* hash of #partition (murmer3)

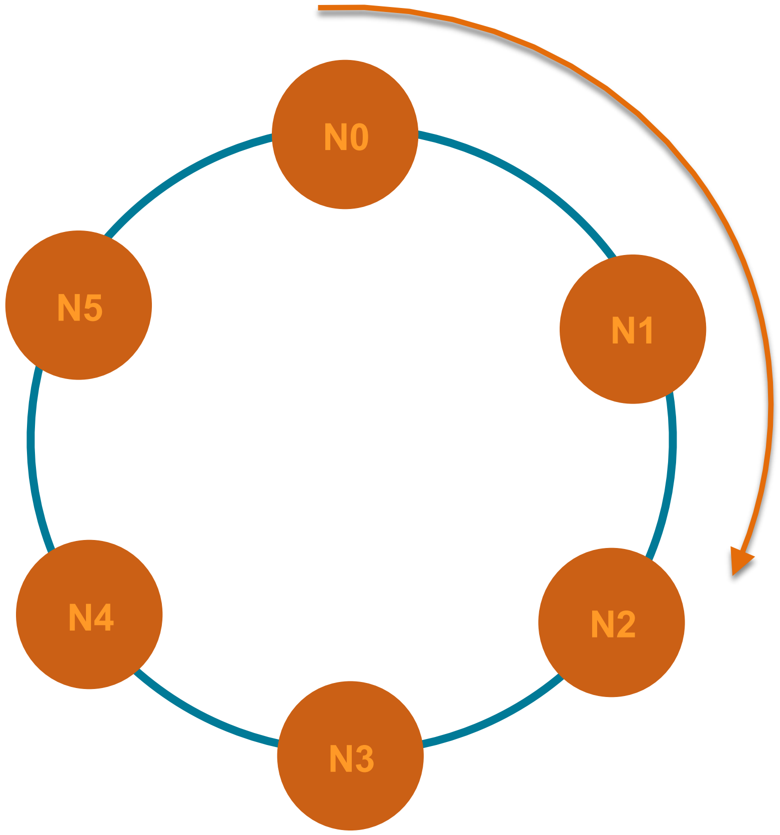




Data Distribution

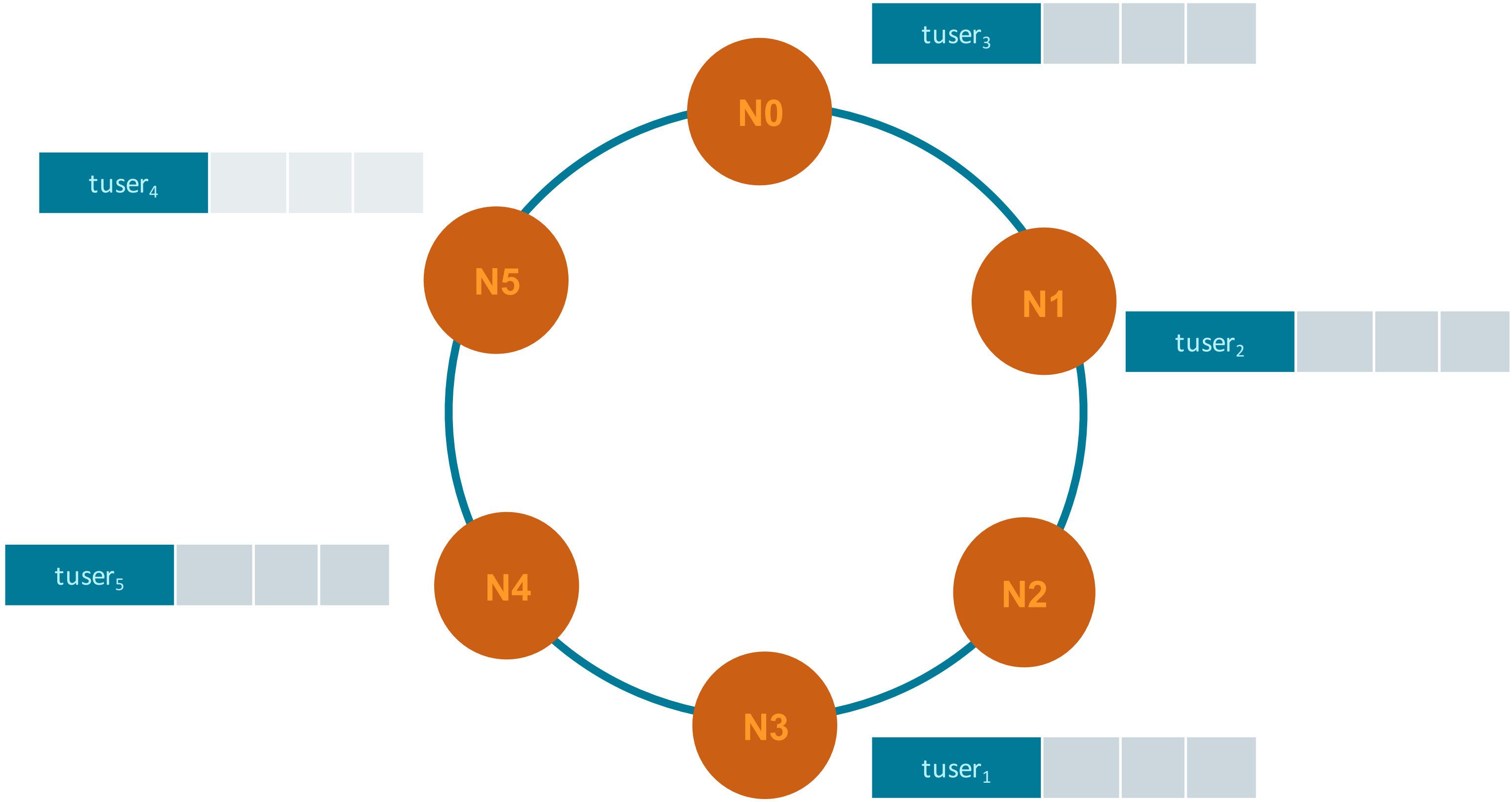
Token = hash of #partition → #node

Token1	target_user 1			
Token2	target_user 2			
Token3	target_user 3			
Token4	target_user 4			
Token4	target_user 5			



Data is evenly distributed and clock wise replicated

Automated Data Distribution Sharding



What's Stored With Each Column?

nick	notification_time: 2017-11-01 10:00		notification_time: 2017-11-02 11:00		notification_time: 2017-11-03 09:00	
	ntfcid: 5321998c	activity: tom liked	ntfcid: ea1c5d35	activity: jake commented	ntfcid: 5321998c	activity: mike created account

column name : "activity"
column value : "tom liked"
timestamp : 1353890782373000
TTL : 3600

- Last Write Win, cross cluster clock sync, e.g NTP

DATASTAX

Skinny vs. Wide rows

Compound Partition Key

```
PRIMARY KEY ((target_user,day), notification_time)
```

- Number cells per partition (2 billion max.)
- Faster operations and lower latency
- multiple gets per dataset if needed
- Equality select

```
SELECT * FROM notification where target_user =‘mike’ AND day IN (1,2,3);
```

Multiple Clustering Columns

```
PRIMARY KEY ((target_user), day, notification_time)
```

- Simulates 1-N relation ship
- wide rows
- Range selects

```
SELECT * FROM notification where target_user =‘mike’ AND day IN (1,2,3);
```

Basic Approach to Data Modeling

1. What queries are needed in your app?
2. What are your natural unique keys?
3. Is there ordering of the data needed to serve each query?
4. What are the groupings (1:M, M:M) in the data?
5. What filtering will your queries need?
6. Can events be stored in chronological order?
7. Does the data expire? Do large chunks of data expire together?

Alternatives to joins

- Collections
- Nested frozen Collections
- User Defined Types
- Nested Collections with UDTs
- JSON notation

notifications		
id	source_id	…

```
Select * from notifications
NATURAL JOIN sources
```

sources		
id	type	size

Collections
<Values>

User Defined Types			
Name	Name	Name	Name
Value	Value	Value	Value

Partition	Name	Name	Collection	User Defined Types		
	Value	Value	<Values>	Name	Name	Collections
				Value	Value	<Values>

Row

Alternatives to joins

```
CREATE TYPE source_type ( encoding text, size int, location text);

CREATE TABLE notifications(
    target_user text,
    notification_id timeuuid,
    notification_time timestamp,
    source map <text, frozen <source_type>>,
    activity text,
    PRIMARY KEY (target_user, notification_time)
) WITH CLUSTERING ORDER BY (notification_time DESC)


INSERT INTO notifications JSON '{
    "target_user ": “nick",
    "notification_id ": "5321998c",
    "notification_time ": “2017-11-01",
    “source": {
        “profile_pic": {
            “encoding": “jpeg",
            “size": 15,
            “location": “/"}}}';
```

DATASTAX

DSE Performance Basics

Can DSE be both bigger and faster? Yes it can.

More Throughput? More Data?
Use more nodes (scale out)
Do not use too big nodes (~~scale up~~)
Know Cassandra ops best practices Use OpsCenter to monitor, alert, repair

Faster Operations? Predictable Latency?
Check your data model and queries
Use asynchronous queries
Use prepared statements
Compaction tuning or maybe strategy

DATASTAX

Lab 2 : DSE Core and Operations

Questions?

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

DATASTAX