ISEngineering

Technische Universität Berlin

Enterprise Computing: Cloud Storage Part III

Marco Peise

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Agenda

1. Intro Cloud Storage

2. Systems in detail
   a) Dynamo
   b) GFS, BigTable

3. Other systems
   c) Cassandra
   d) MongoDB

Cassandra

TOPICS

1. Cassandra Architecture & Operation (lecture)
2. Cassandra Data Model (exercise)
3. Cassandra Query Language (exercise)

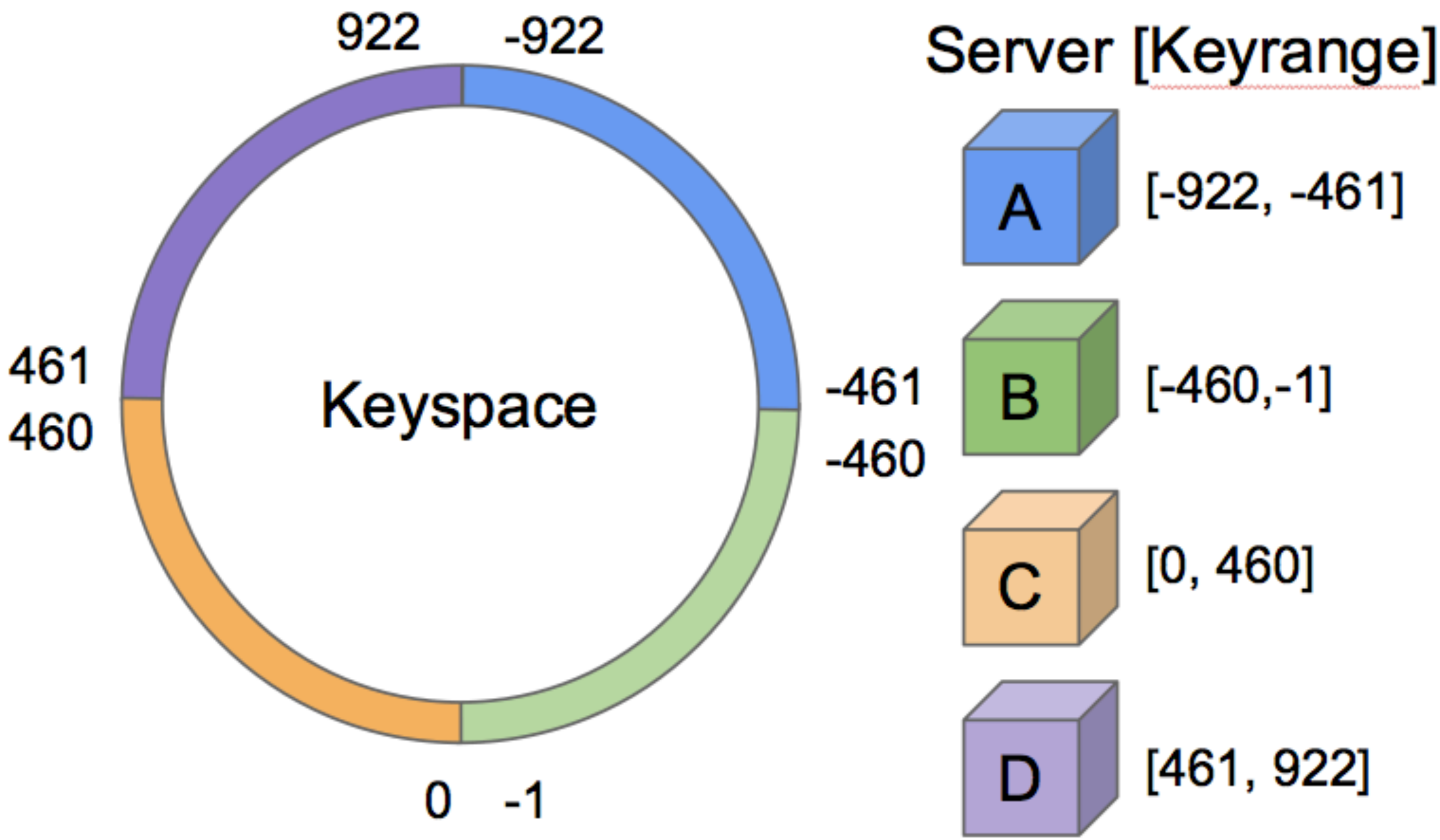
Reference:
Avinash Lakshman and Prashant Malik. 2010. Cassandra: a decentralized structured storage system. SIGOPS Oper. Syst. Rev. 44, 2 (April 2010), 35-40.

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture

- Data Partitioning & Distribution
  - Partitioners
  - Virtual Nodes
- Scaling a Cluster
- Data Replication
- Network Toplogy (Snitches)
- Server-to-Server Communication (Gossip)
  - Membership
  - Failure detection
- Client-Server Communication
- Local Persistence

Cassandra Architecture: Data Partitioning & Distribution



Cassandra Architecture: Partitioners

- ByteOrderedPartitioner (not recommended)
  - Plus: You can scan across lexically ordered keys
  - Minus: bad load balancing, hotspots, etc.
- RandomPartitioner (default before 1.2)
  - The RandomPartition distributes data evenly across the nodes using an MD5 hash value of the row key. The possible range of hash values is from 0 to 2127 -1.
- Murmur3Partitioner (default since 1.2)
  - The Murmur3Partitioner uses the MurmurHash function. This hashing function creates a 64-bit hash value of the row key. The possible range of hash values is from -263 to +263.

Source: http://www.datastax.com/docs/1.2/cluster_architecture/partitioners

Cassandra Architecture: Partitioners

MD5 hash

Max	5e02739678..
Susi	a9a0198010..
Jason	f4eb27cea7..
Devon	78b421309e..

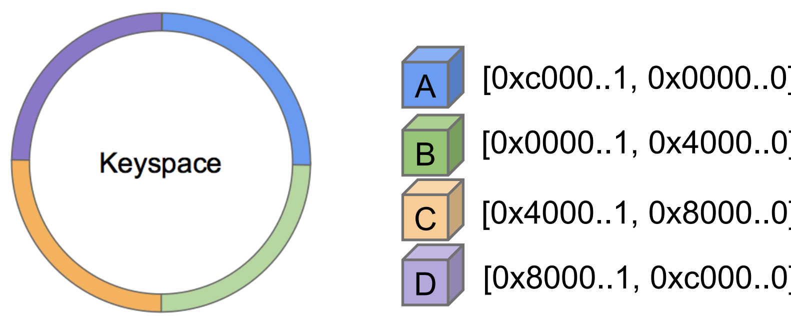
MD5 hash operation yields a 128-bit number for keys of any size

Source: http://www.datastax.com/docs/1.2/cluster_architecture/partitioners

Cassandra Architecture:
Data Partitioning & Distribution

	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

Max	**5e02739678..**
Susi	**a9a0198010..**
Jason	**f4eb27cea7..**
Devon	**78b421309e..**



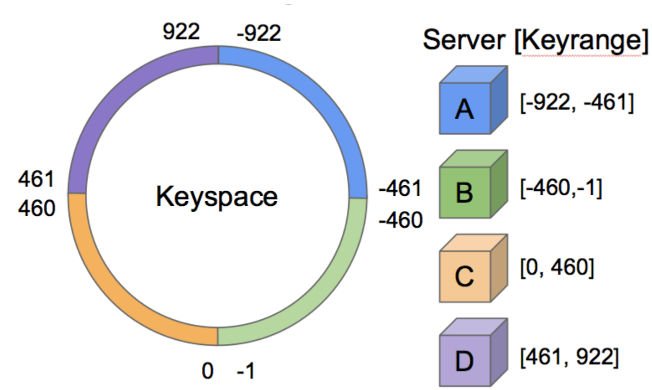
Technische Universität Berlin

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Data Partitioning & Distribution

	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

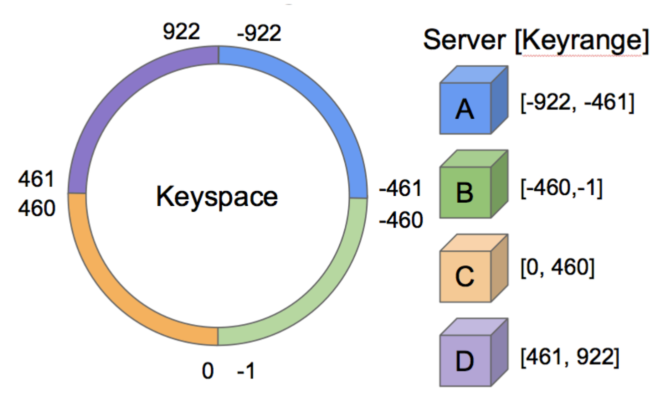
Max	**5e02739678..**
Susi	**a9a0198010..**
Jason	**f4eb27cea7..**
Devon	**78b421309e..**



Cassandra Architecture: Data Partitioning & Distribution

	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

Max	**5e02739678..**
Susi	**a9a0198010..**
Jason	**f4eb27cea7..**
Devon	**78b421309e..**



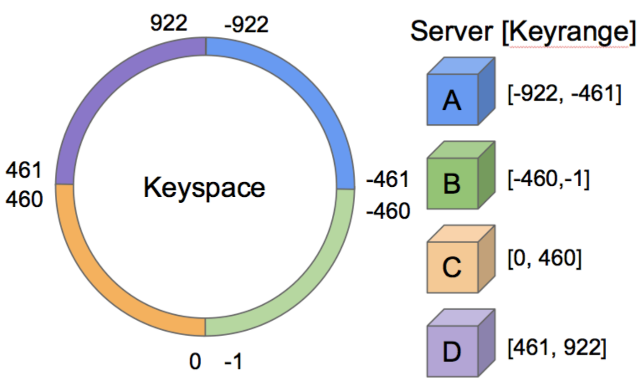
ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Data Partitioning & Distribution

Technische Universität Berlin

	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

Max	5e02739678..
Susi	a9a0198010..
Jason	f4eb27cea7..
Devon	78b421309e..

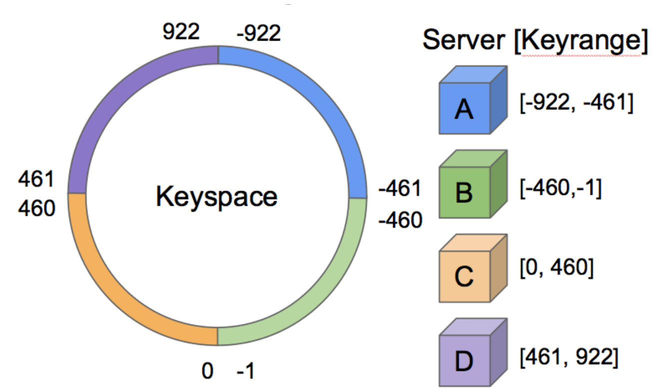


ISEngineering
Wirtschaftsinformatik – Information Systems Engineering

Cassandra Architecture:
Data Partitioning & Distribution

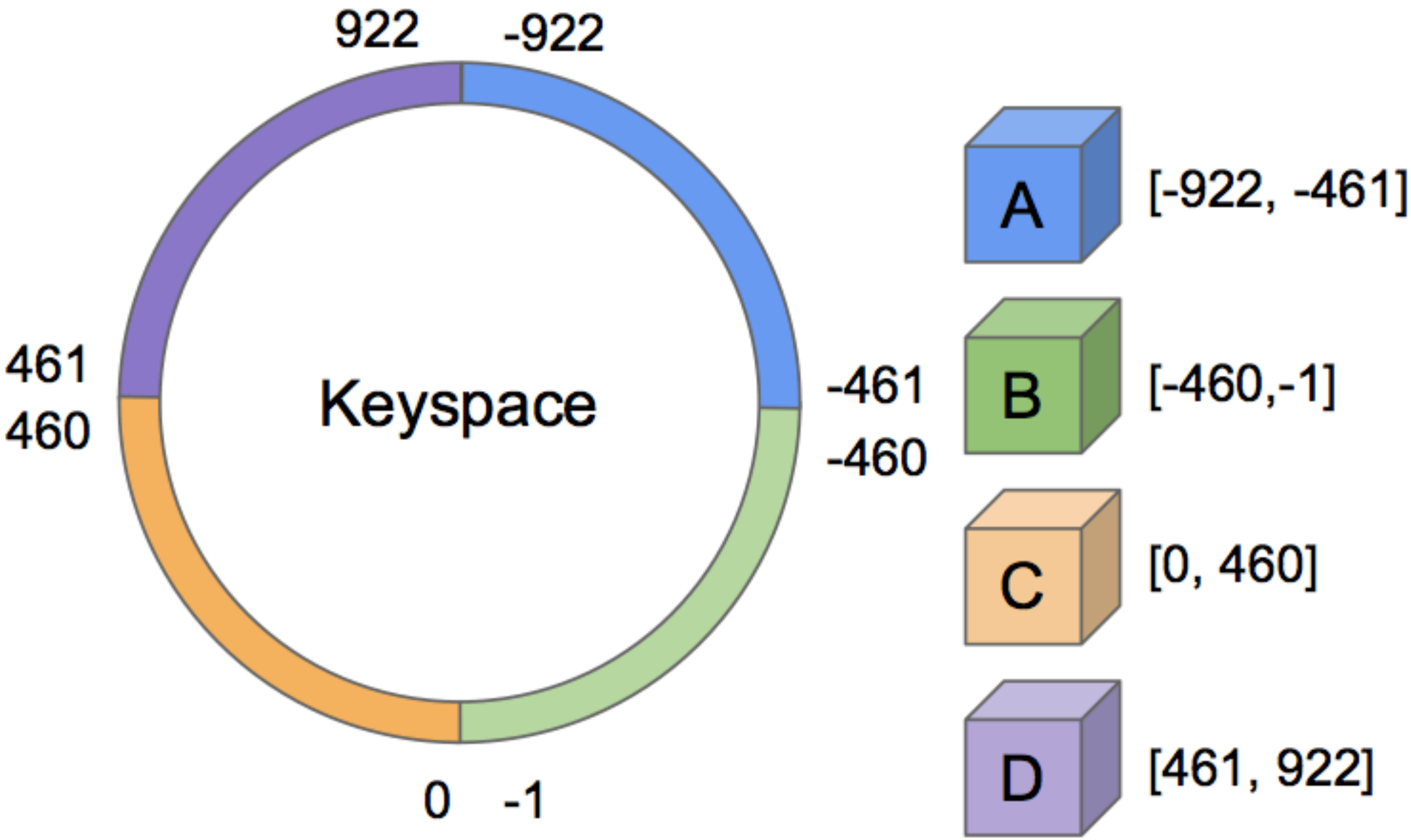
	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

Max	5e02739678..
Susi	a9a0198010..
Jason	f4eb27cea7..
Devon	78b421309e..

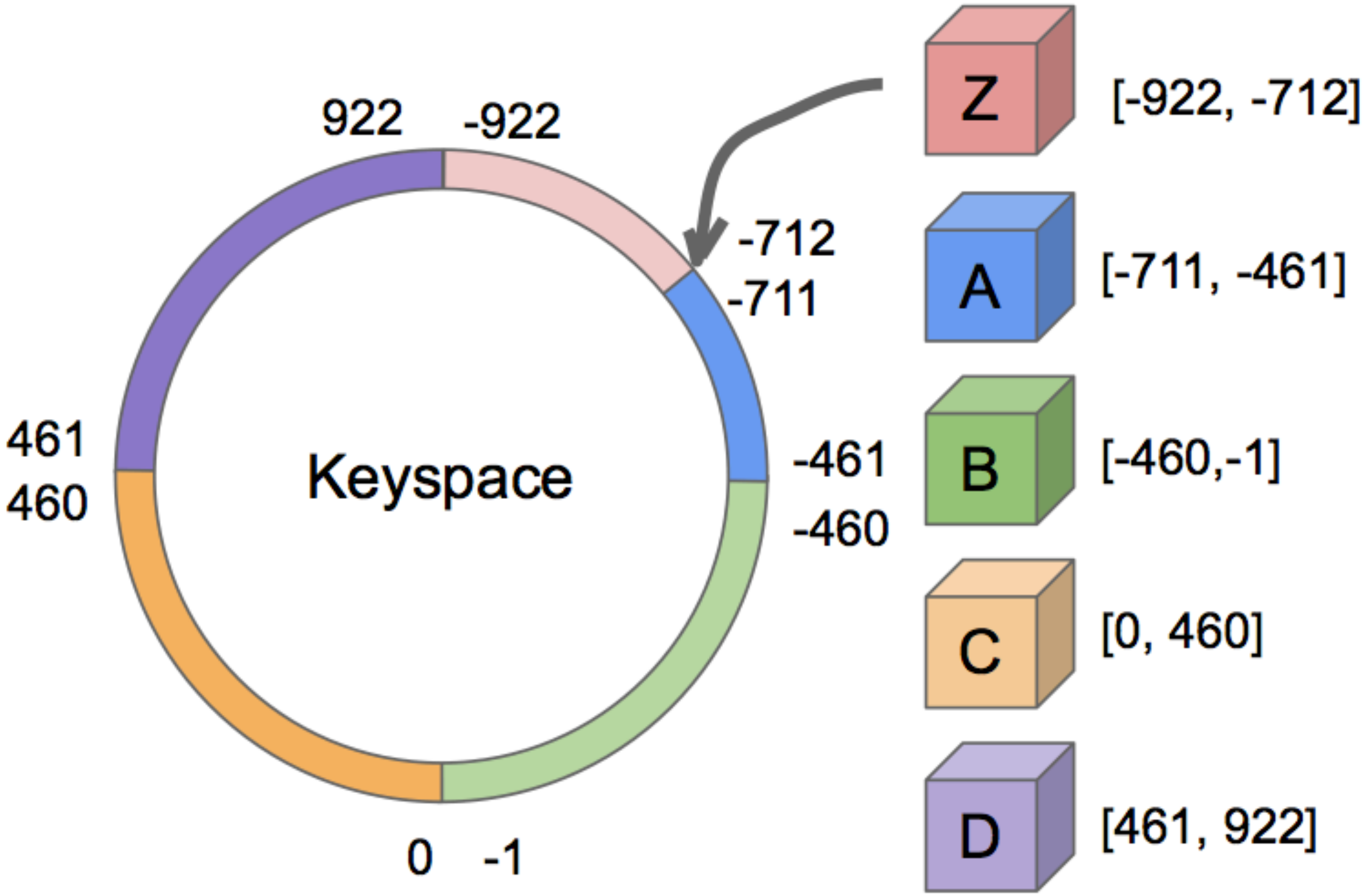


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Scaling

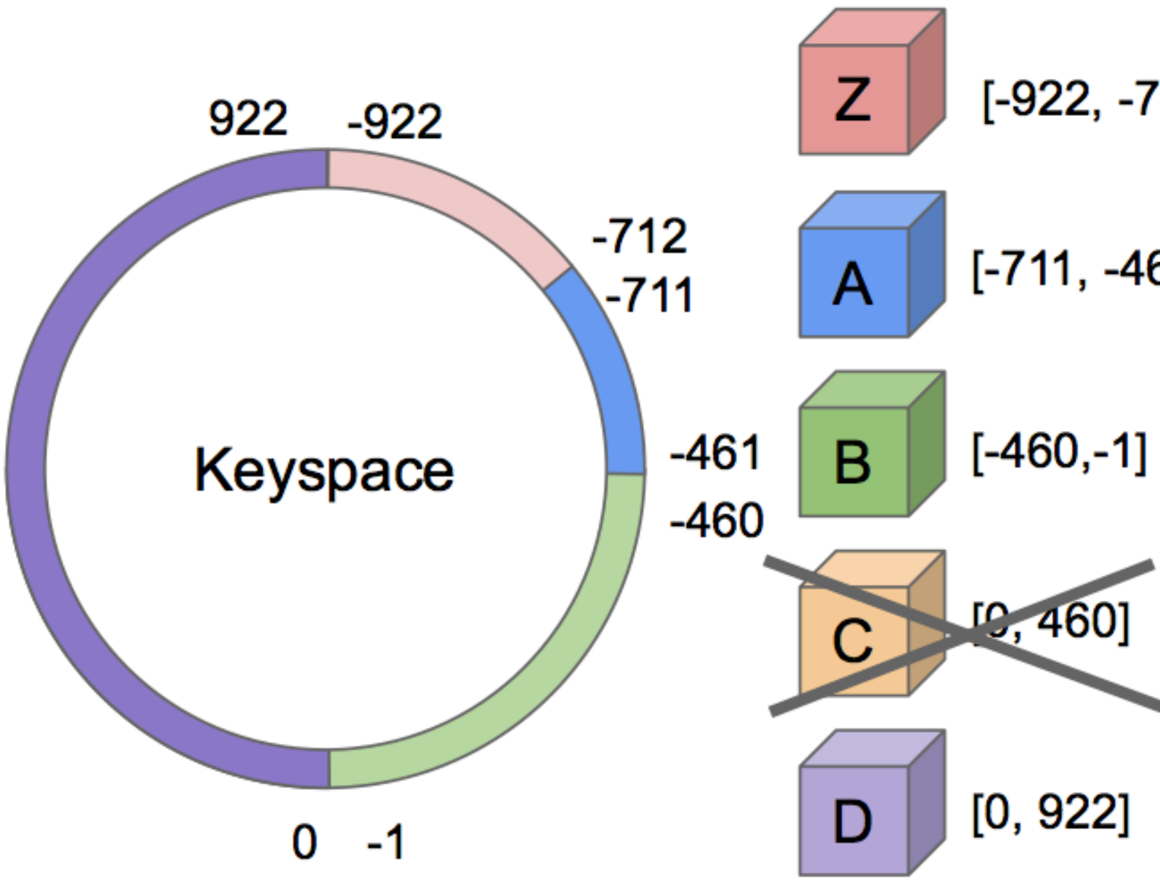


Cassandra Architecture: Scaling



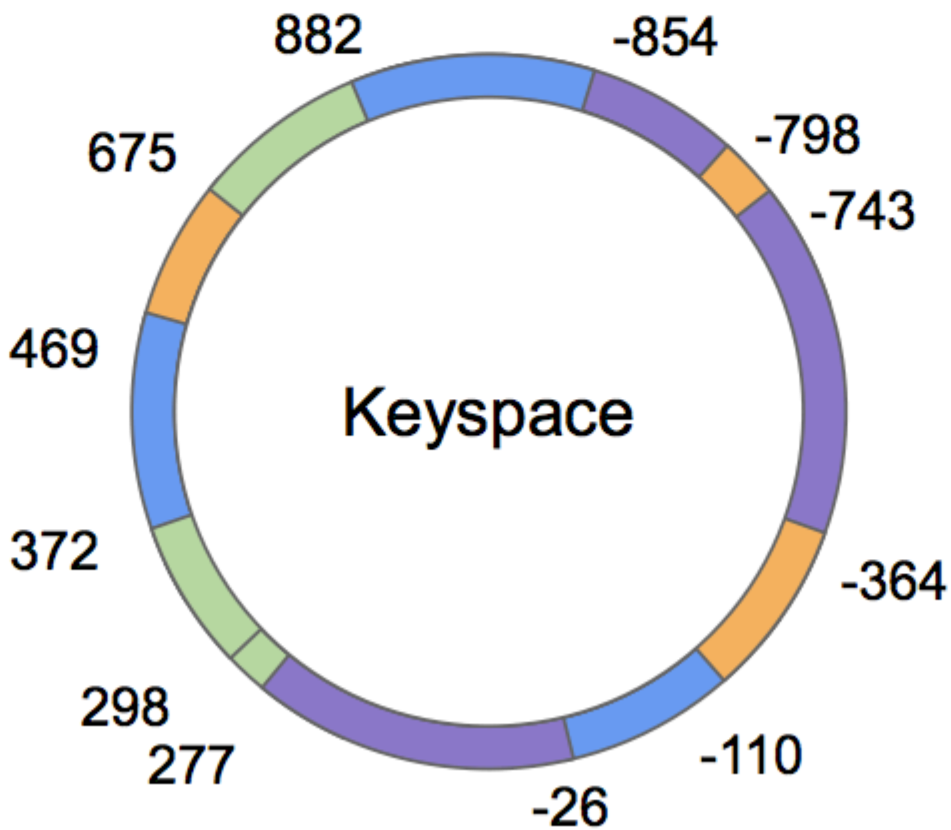
ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Scaling



ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Virtual Nodes

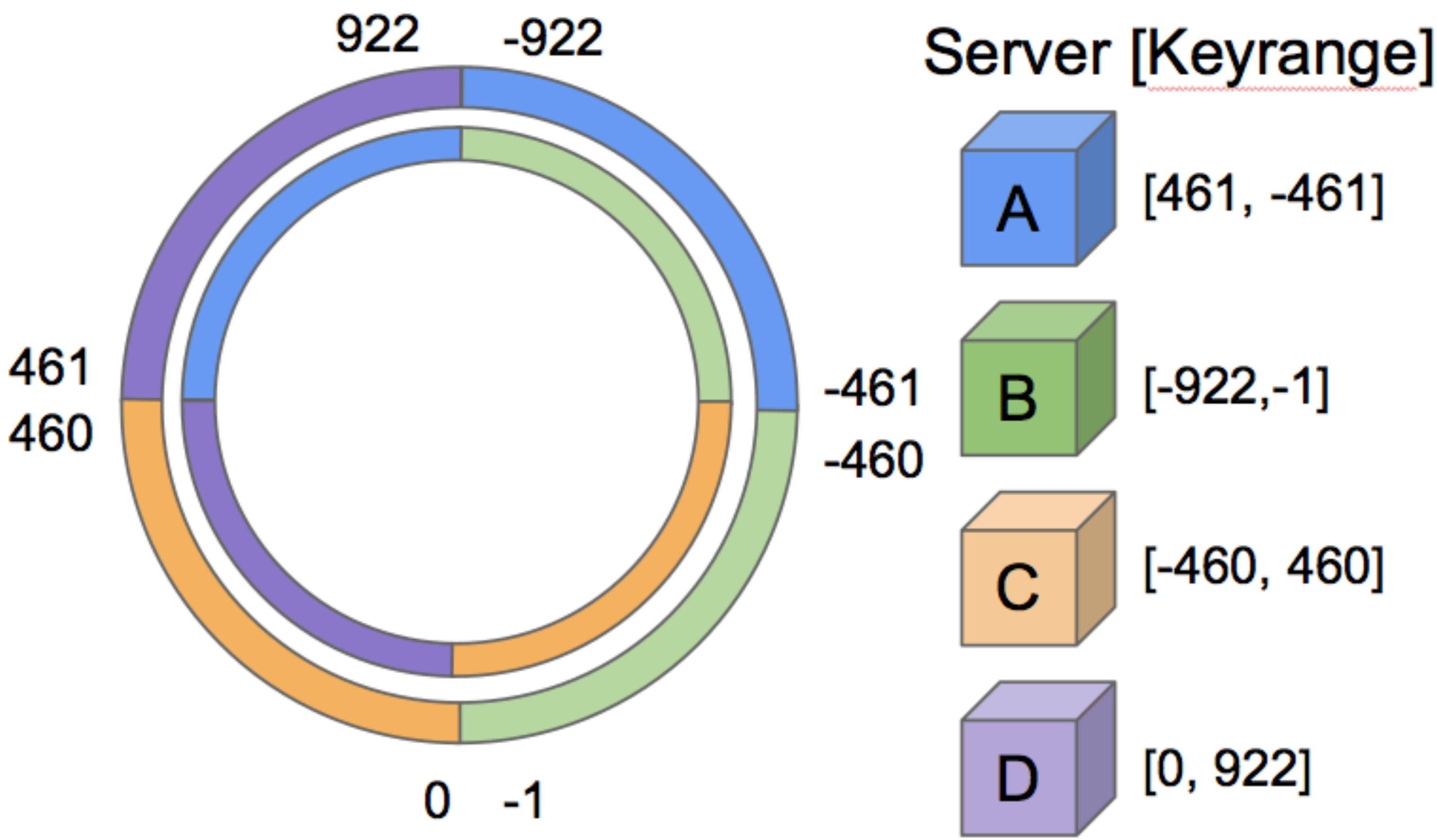


Server [Keyrange]

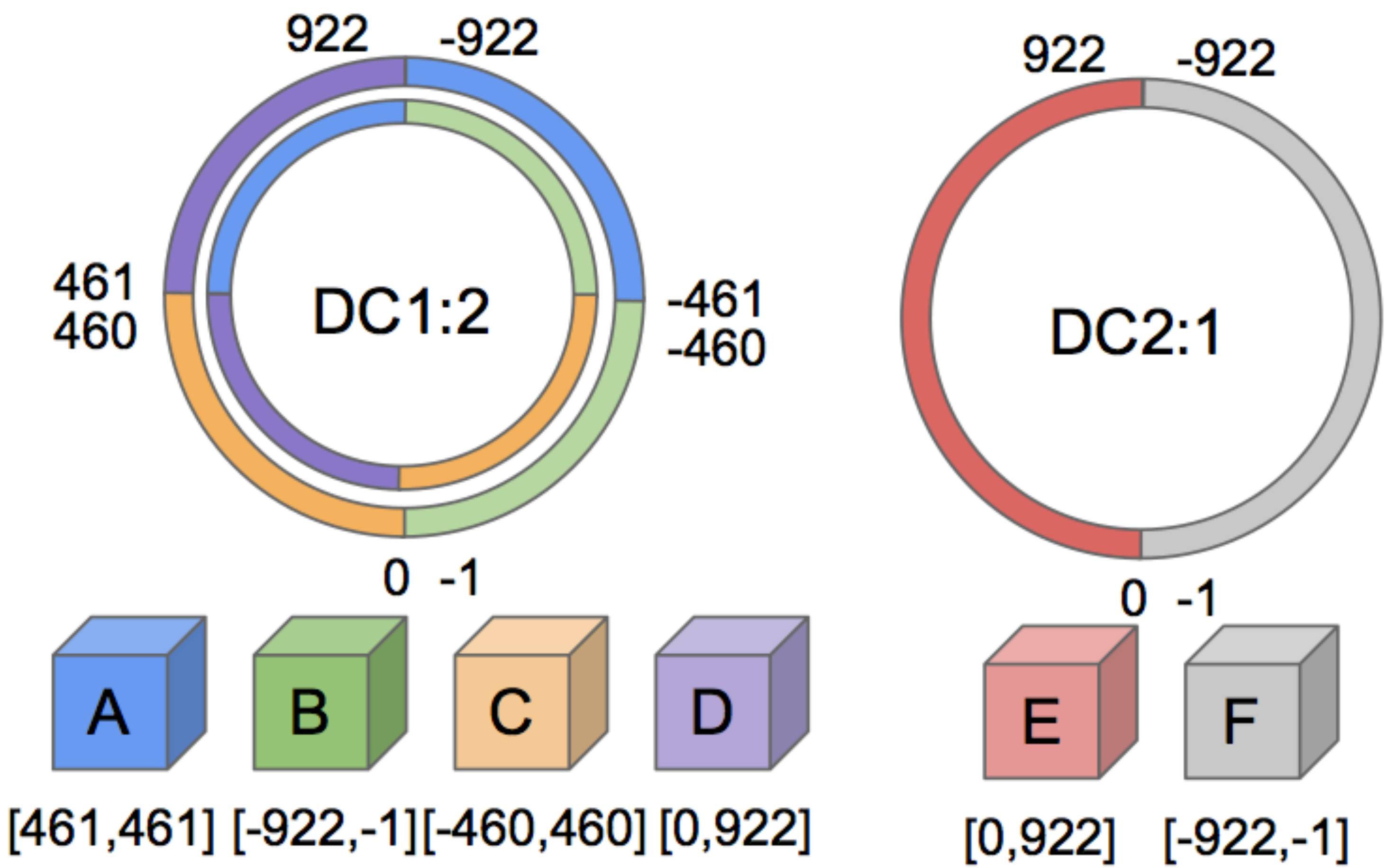
A	[882, -854] [-110, --26] [372, 469]
B	[675, 882] [227, 298] [298, 372]
C	[-798, -743] [-364, -110] [469, -675]
D	[-854, -798] [-743, -364] [-26, 277]

ISEngineering
Wirtschaftsinformatik – Information Systems Engineering

Cassandra Architecture: Data Replication



Cassandra Architecture: Multi-DC Data Replication



ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Network Topology

- Snitch function: to determine which datacenters and racks are both written to and read from
- Selected Snitches
  - **Dynamic snitching**
  - **SimpleSnitch (default)**
  - **RackInferringSnitch**
  - **PropertyFileSnitch**
  - **GossipingPropertyFileSnitch**
  - Ec2Snitch | Ec2MultiRegionSnitch
  - GoogleCloudSnitch | CloudstackSnitch
- Set the `endpoint_snitch` property in *cassandra.yaml*

Cassandra Architecture: Network Topology – Dynamic snitching

- dynamic snitch is enabled by default and is recommended for use in most deployments
- **monitors read latency** and, when possible, **routes requests away from poorly-performing nodes**
- Problem: many replicas to ask for the actual data
- Solution: monitoring the performance of reads from the various replicas and choosing the best one based on this history
- however in absence of information, the dynamic snitch can’t react > read interval > badness_threshold permitting

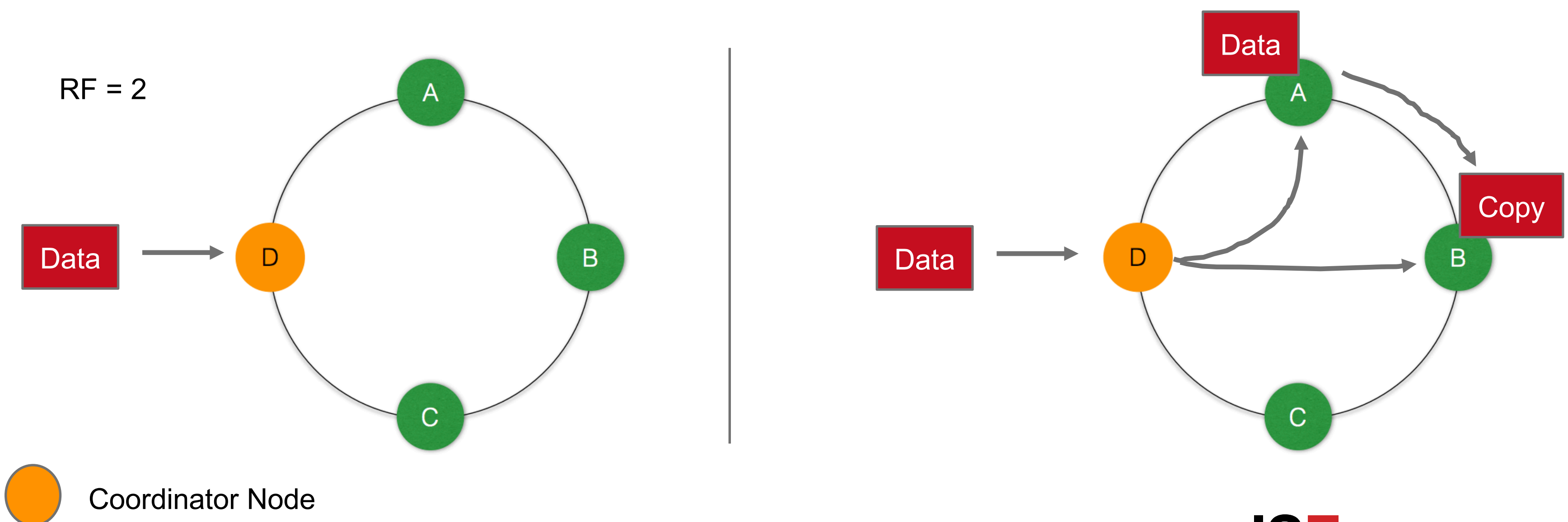
Cassandra Architecture: Network Topology – Dynamic Snitching

- Properties in the *cassandra.yaml*

```
dynamic_snitch_update_interval_in_ms: 100
dynamic_snitch_reset_interval_in_ms: 600000
dynamic_snitch_badness_threshold: 0.1
```

Cassandra Architecture: Network Topology - SimpleSnitch

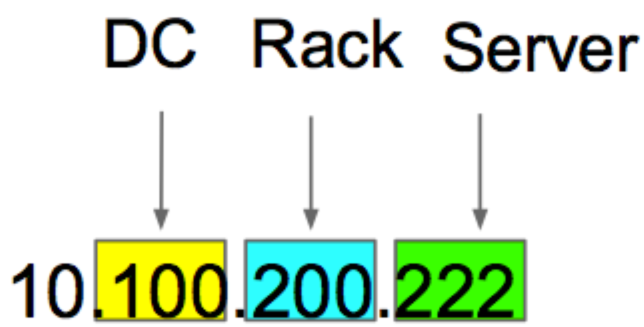
- SimpleSnitch does not recognize data center or rack information
- Only useful for small single-DC deployments
- define the keyspace to use SimpleStrategy and specify a replication factor



ISEngineering
Wirtschaftsinformatik – Information Systems Engineering

Cassandra Architecture:
Network Topology - RackInferringSnitch

- Assumes the network topology from the node's IP address

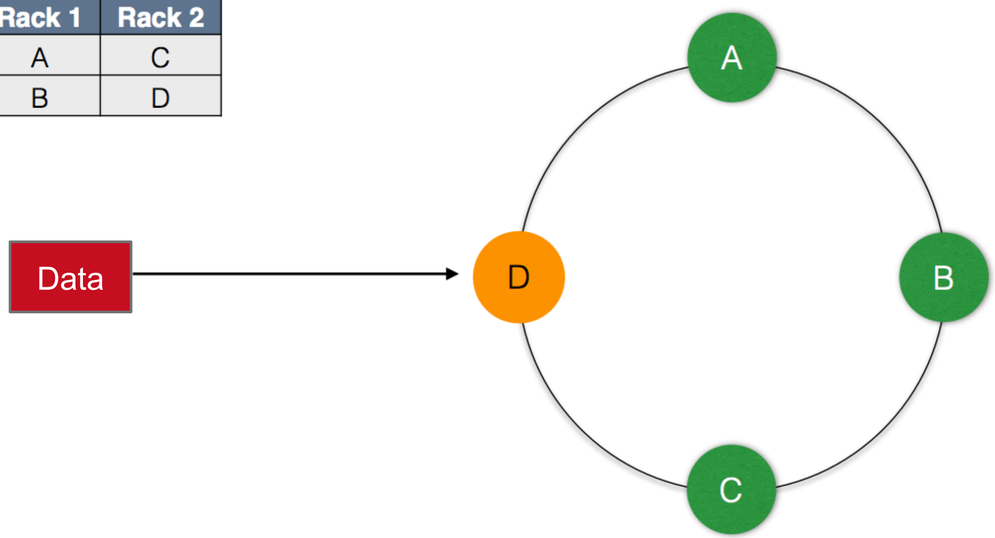


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Network Topology - RackInferringSnitch

- RF = DC1:2

DC1	
Rack 1	Rack 2
A	C
B	D



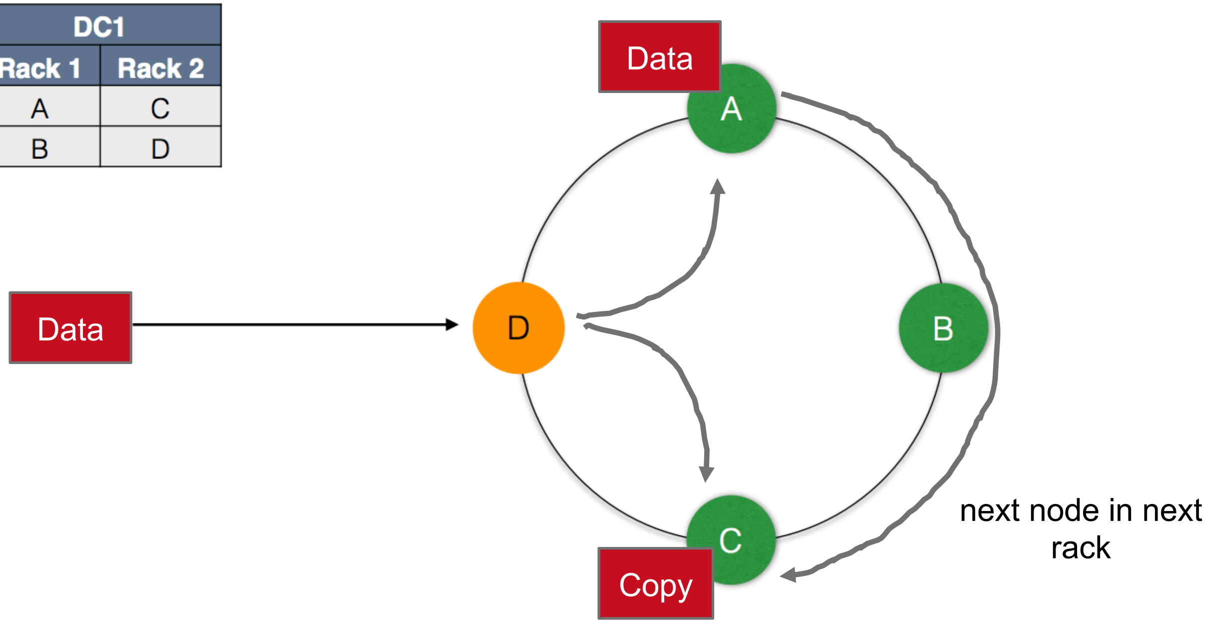
Coordinator Node

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture:
Network Topology - RackInferringSnitch

- RF = DC1:2

DC1	
Rack 1	Rack 2
A	C
B	D

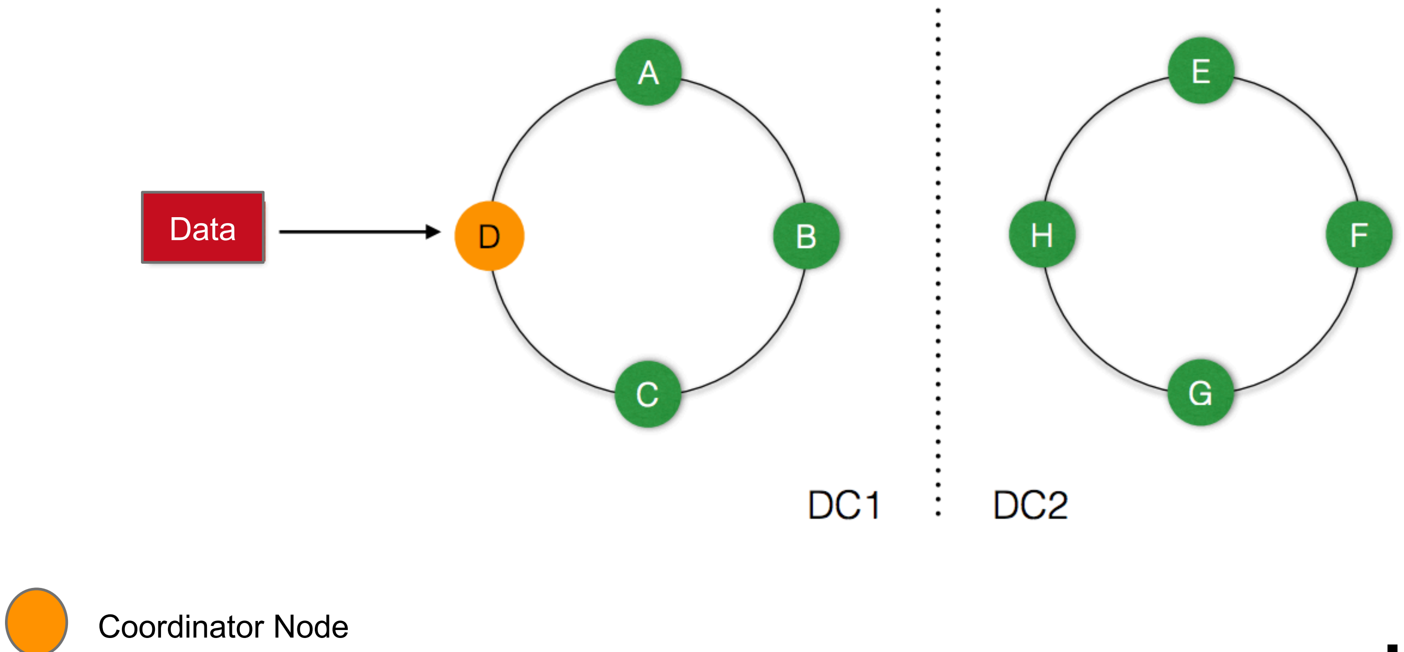


Coordinator Node

Cassandra Architecture: Network Topology - RackInferringSnitch

- RF = DC1:2, DC2:2

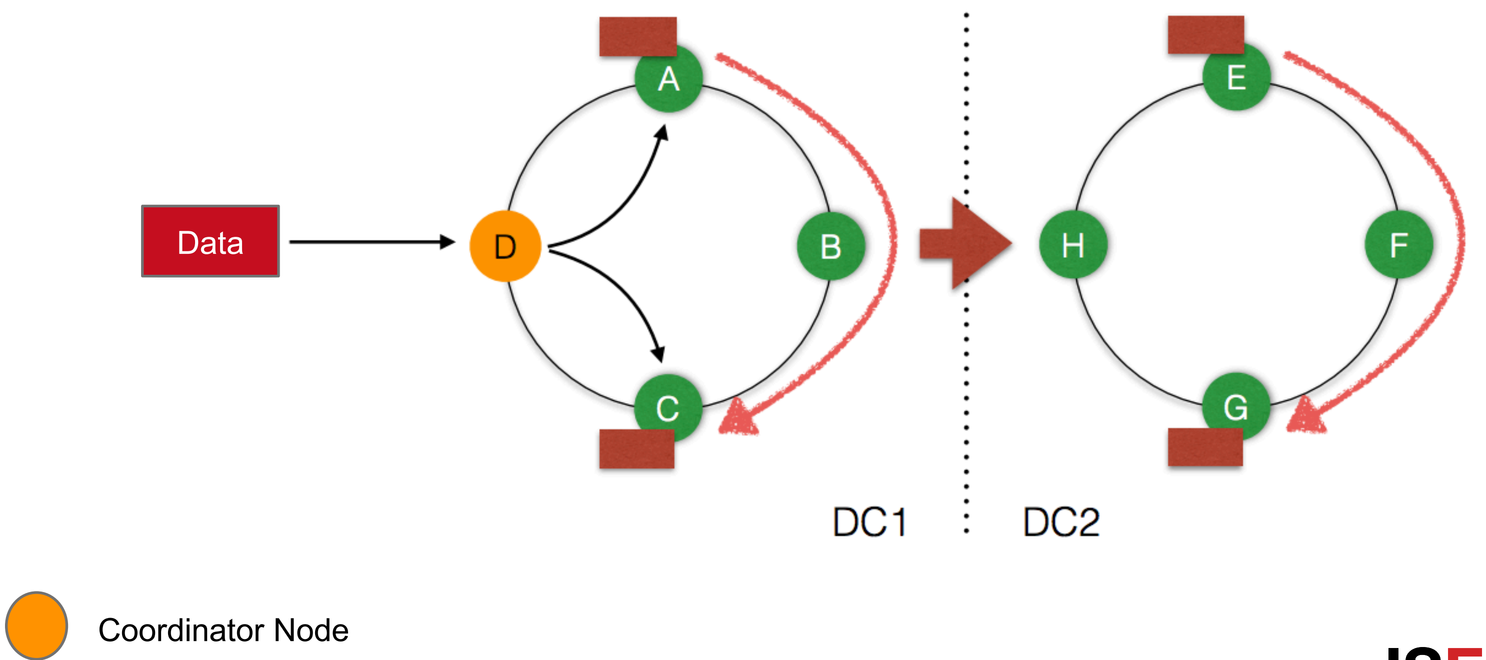
DC1		DC2	
Rack 1	Rack 2	Rack 1	Rack2
A	C	E	G
B	D	F	H



Cassandra Architecture: Network Topology - RackInferringSnitch

- RF = DC1:2, DC2:2

DC1		DC2	
Rack 1	**Rack 2**	**Rack 1**	**Rack2**
A	C	E	G
B	D	F	H



Cassandra Architecture: Network Topology - PropertyFileSnitch

- Uses *conf/cassandra-topology.properties* file to infer data center and rack information
- Useful if cluster layout is not matched by IP addresses or if you have complex grouping requirements
- Example properties file:

```
# Data Center One
175.56.12.105=DC1:RAC1
120.53.24.101=DC1:RAC2

# Data Center Two
110.56.12.120=DC2:RAC1
50.17.10.203=DC2:RAC2
```

Cassandra Architecture: Network Topology-GossipingPropertyFileSnitch

- Each node sets its own data center and rack info via *conf/cassandra-rackdc.properties* file.
- The info is propagated to other nodes via gossip. Fits nicely the P2P style of Cassandra.
- Example properties file:

```
dc=DC1
rack=RAC1
```

Cassandra Architecture: Gossip

- Cassandra uses a gossip protocol to exchange information between servers in a cluster in a peer-to-peer fashion
  - The gossip process runs every second on each Cassandra server
  - Each server sends its state in a message to other servers in the cluster
  - Each gossip message has a version. Old gossip state information on a server is overwritten.

Cassandra Architecture: Failure Detection

Heartbeat Failure Detection

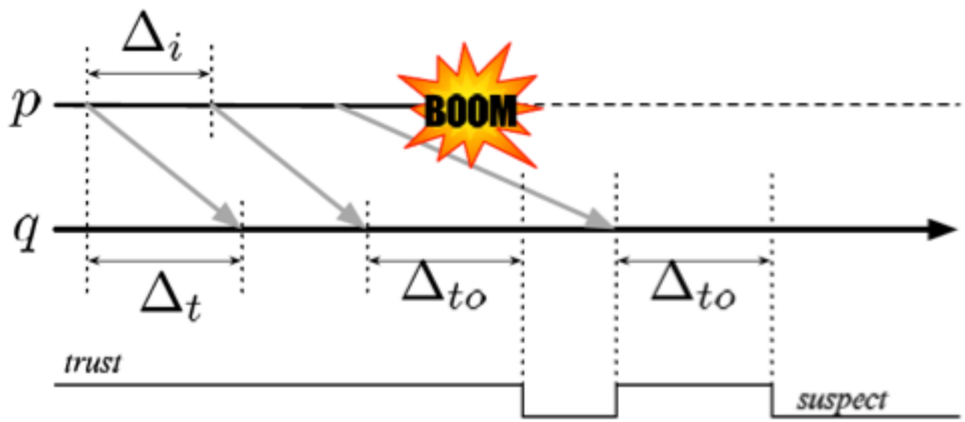
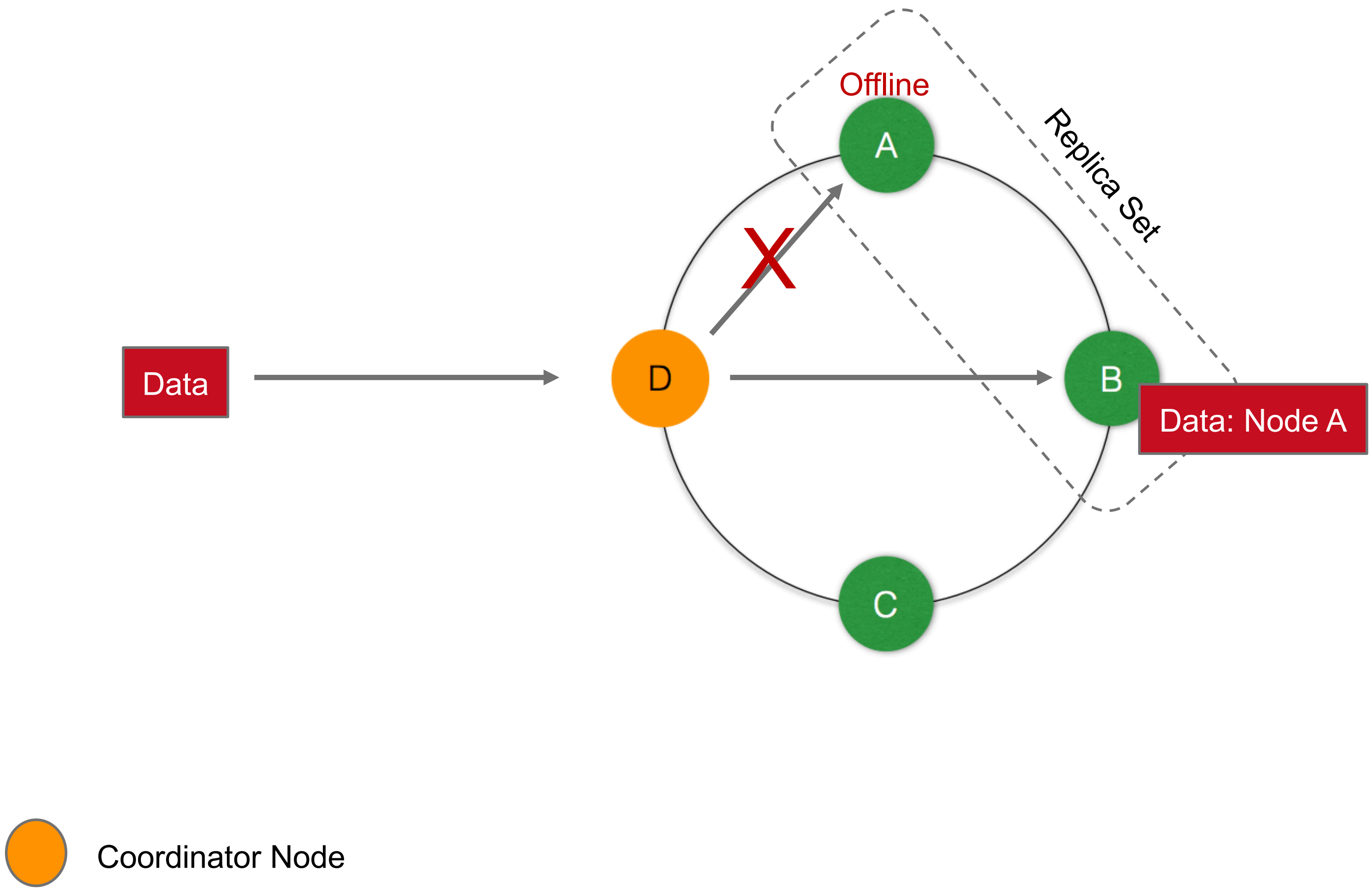


Fig. 1. Heartbeat failure detection and its main parameters.

Naohiro Hayashibara, Xavier Defago, Rami Yared, and Takuya Katayama. 2004. The Phi Accrual Failure Detector. In Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS '04). IEEE Computer Society, Washington, DC, USA, 66-78.

Cassandra Architecture: Temporary Failures - Hinted Handoff

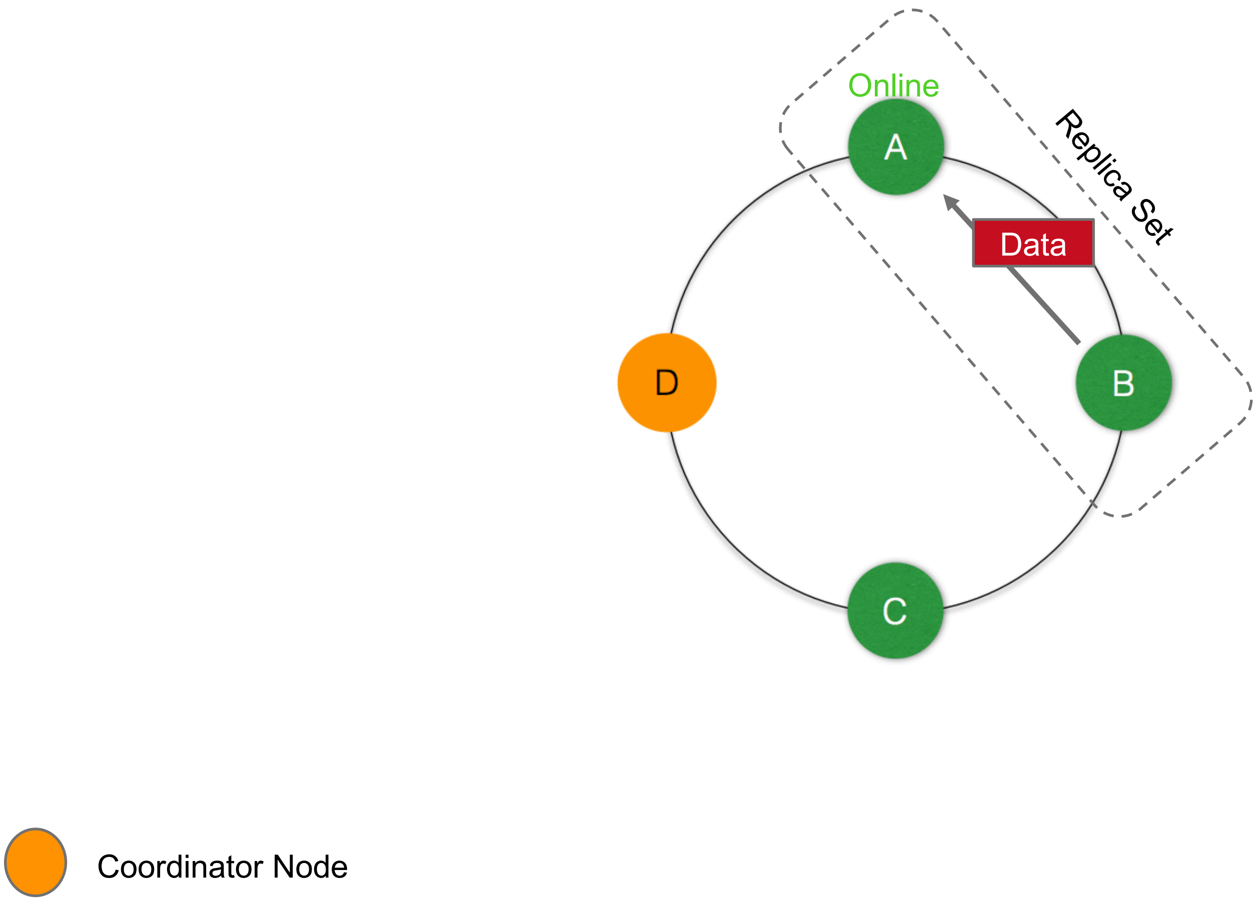
- another node covers for the offline one



ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Temporary Failures - Hinted Handoff

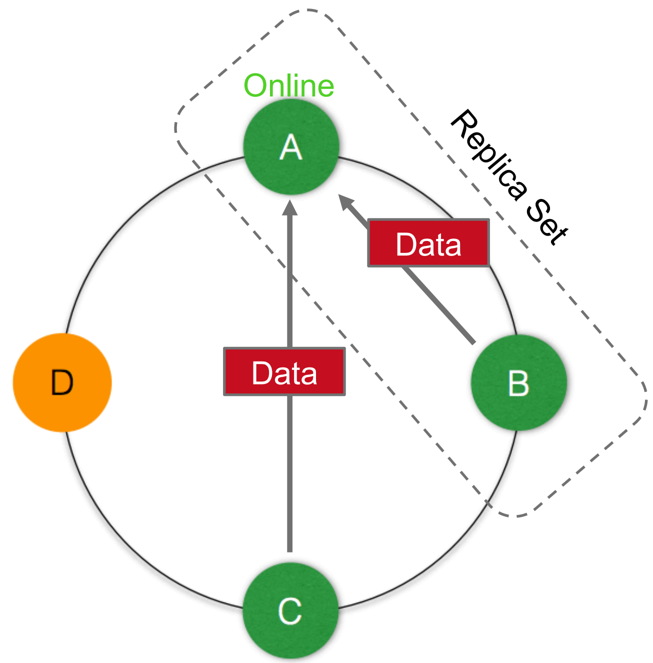
- another node covers for the offline one



ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Permanent Failures – Manual Repair

- repairing through replica wide repair tool



#NodeA:~ nodetool repair

Coordinator Node

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

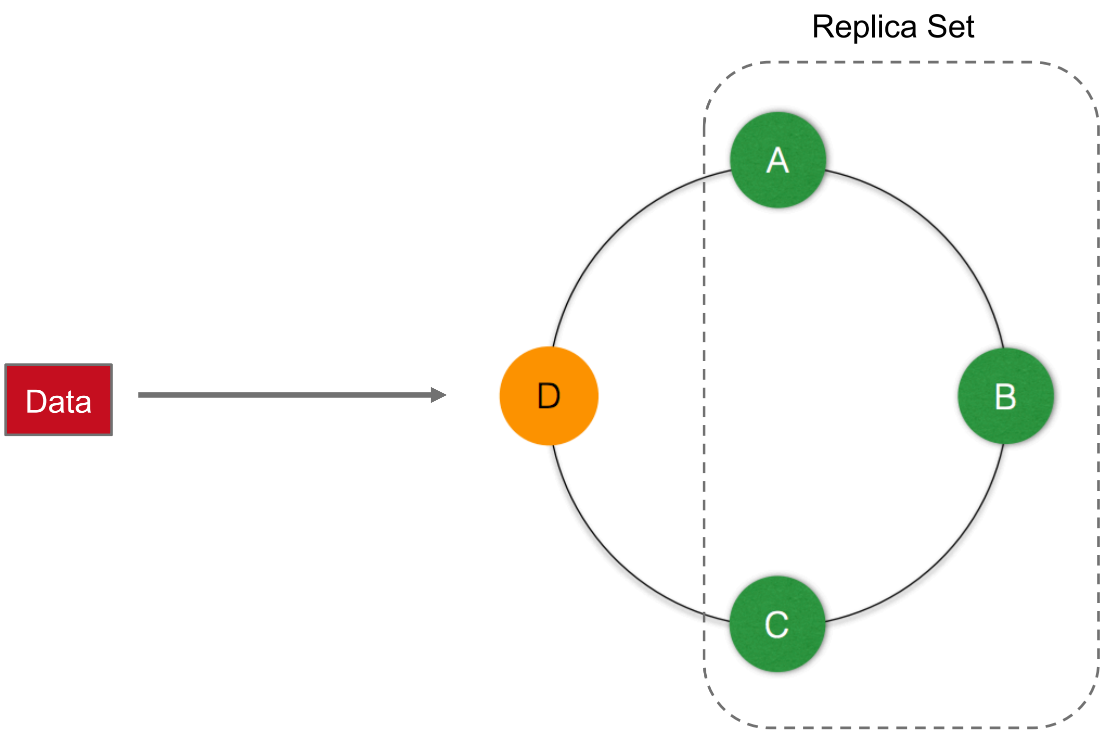
Cassandra Architecture: Consistency

Level	Description
ALL	All Nodes from Replica reply
QUORUM	Majority of Nodes from Replica Set reply
ONE	Minimum one Node from Replica set reply

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

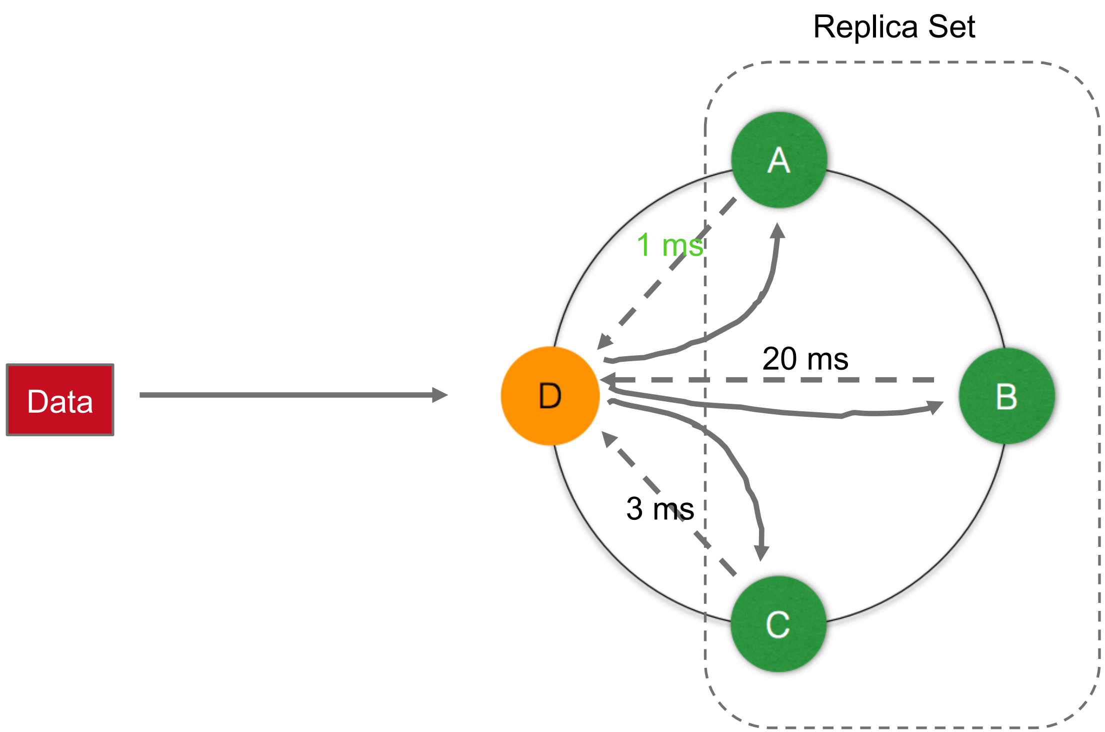
Cassandra Architecture: Consistency Level = ONE

write



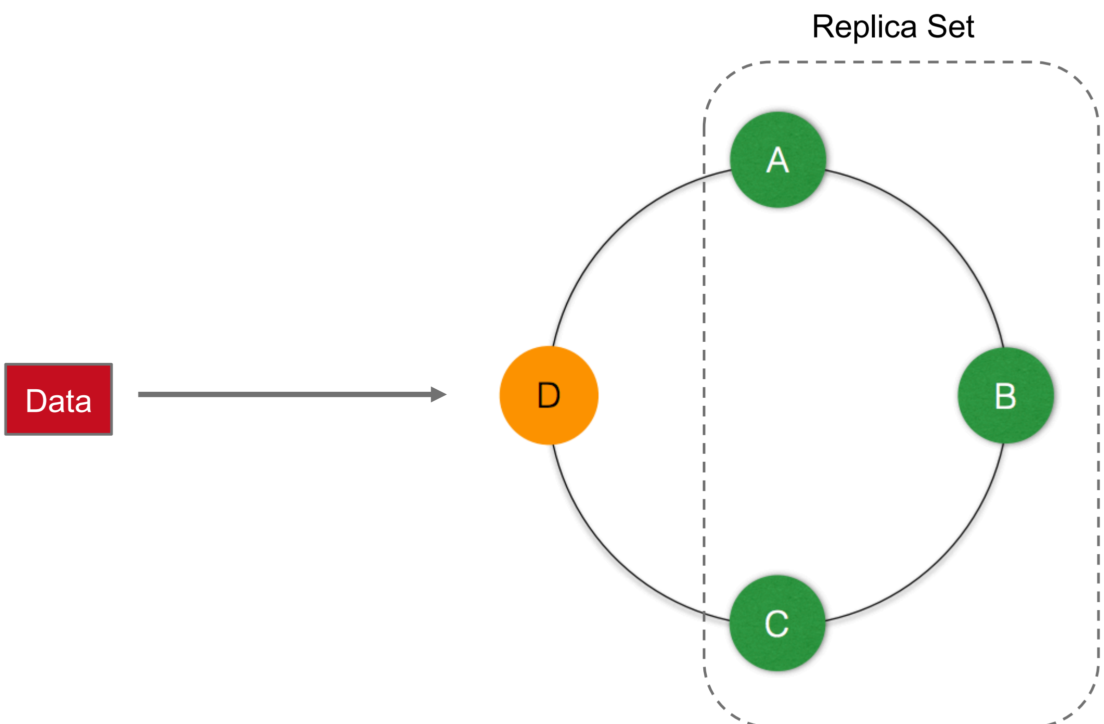
Cassandra Architecture: Consistency Level = ONE

write



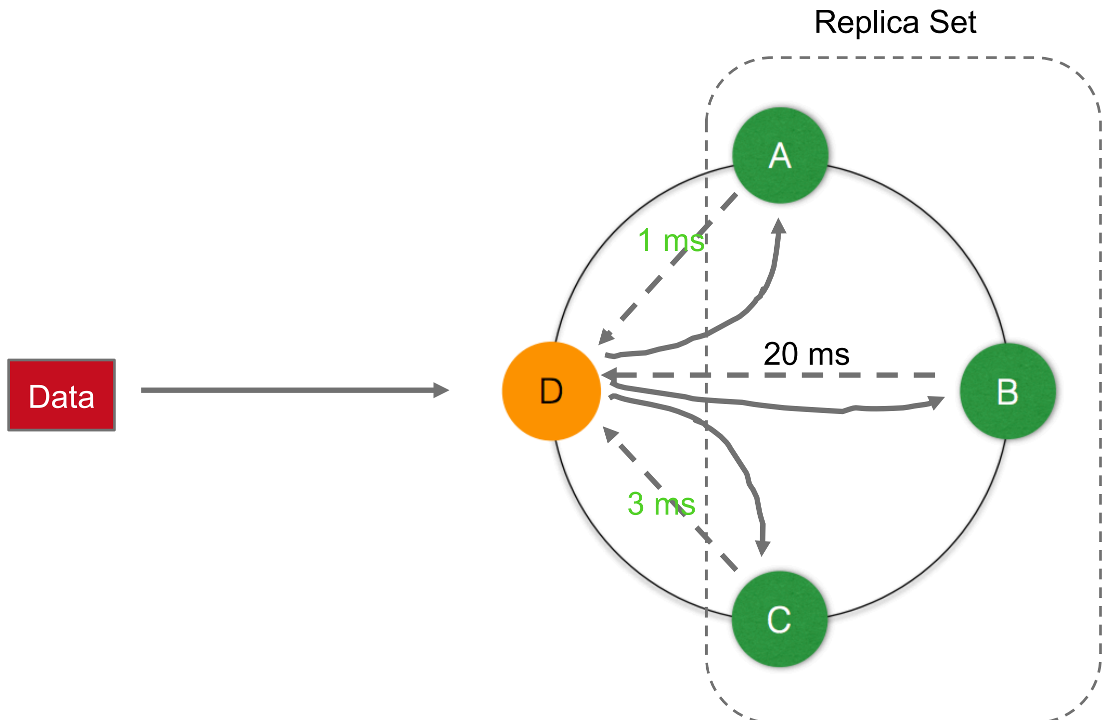
Cassandra Architecture: Consistency Level = QUORUM

write



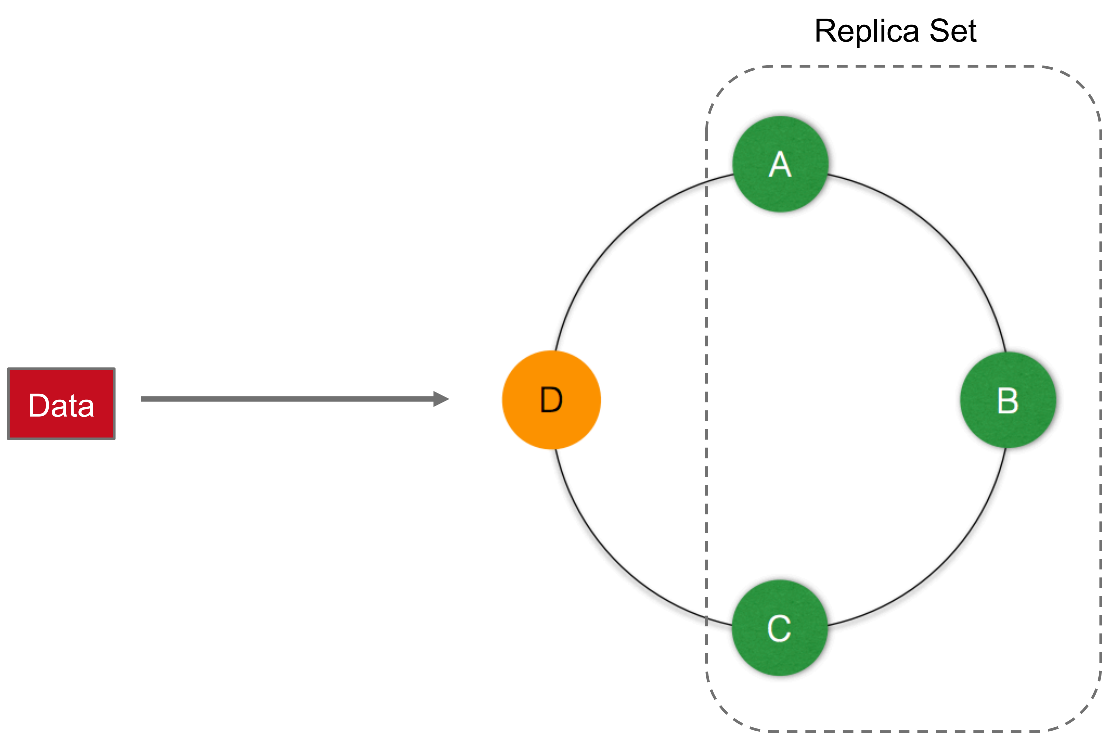
Cassandra Architecture: Consistency Level = QUORUM

write



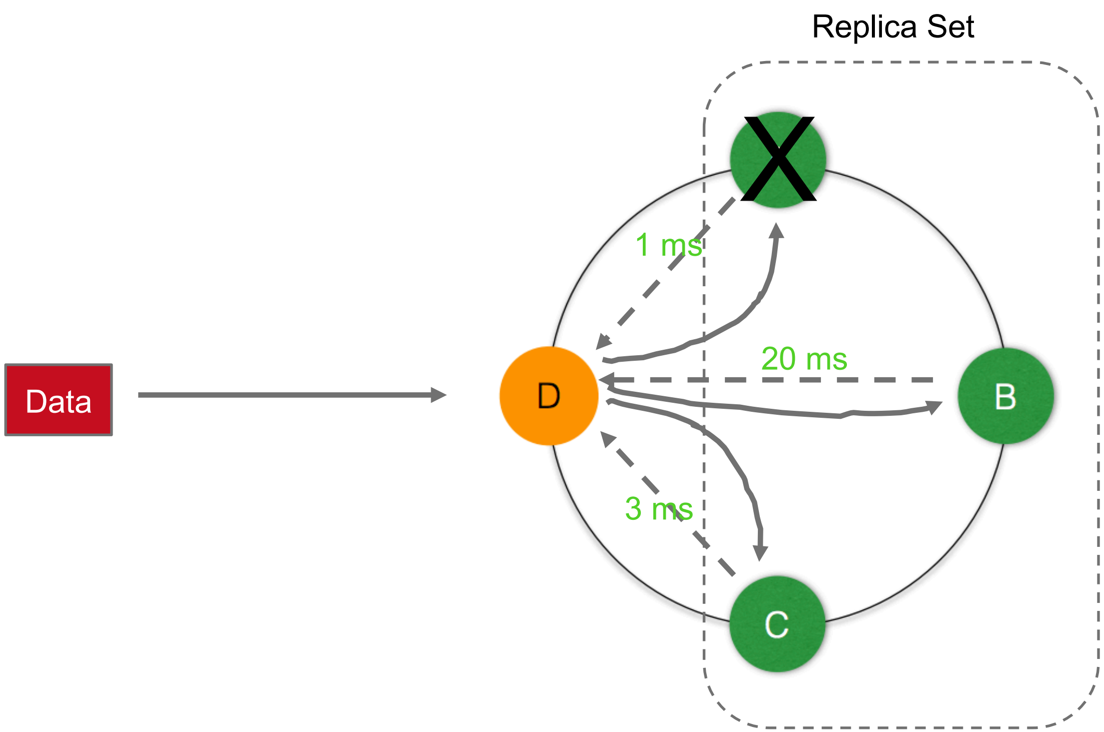
Cassandra Architecture: Consistency Level = ALL

write



Cassandra Architecture: Consistency Level = ALL

write

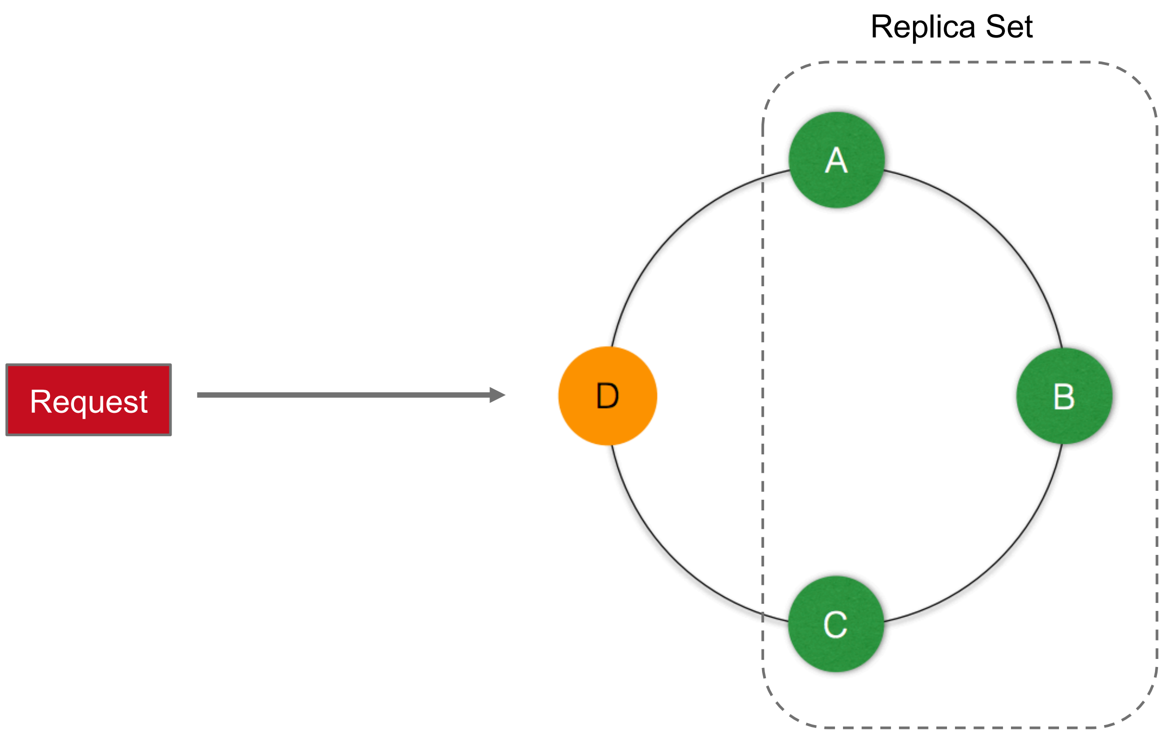


What happens if A is Offline?

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

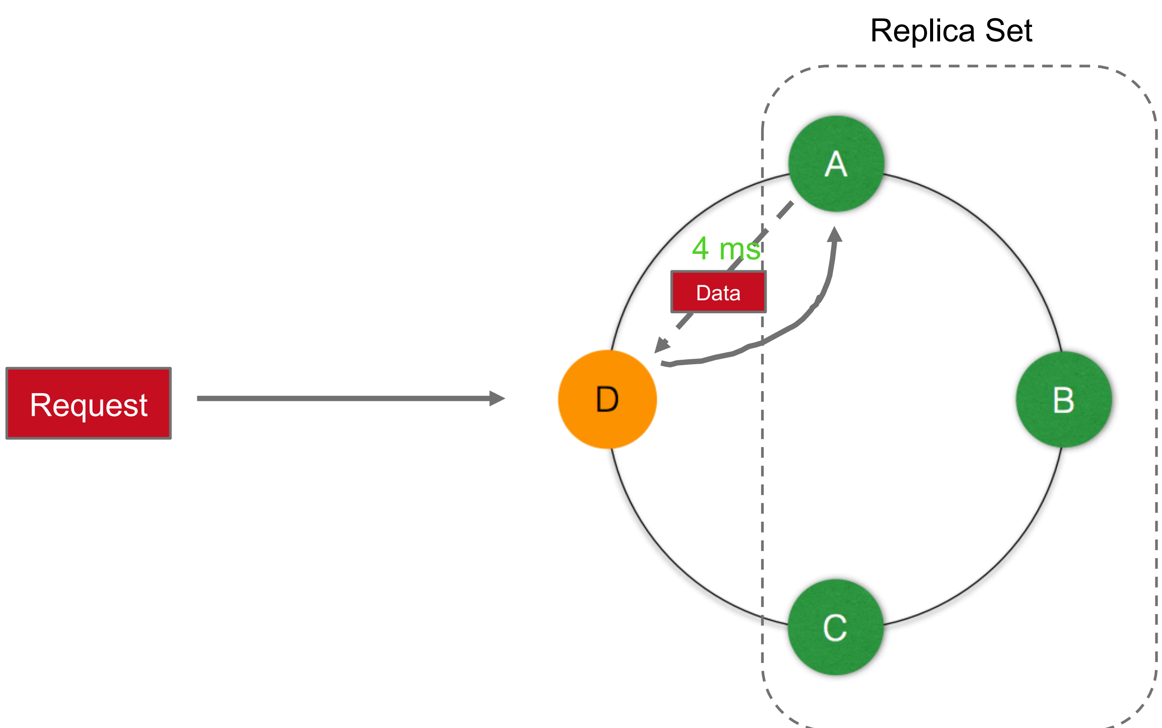
Cassandra Architecture: Consistency Level = ONE

read



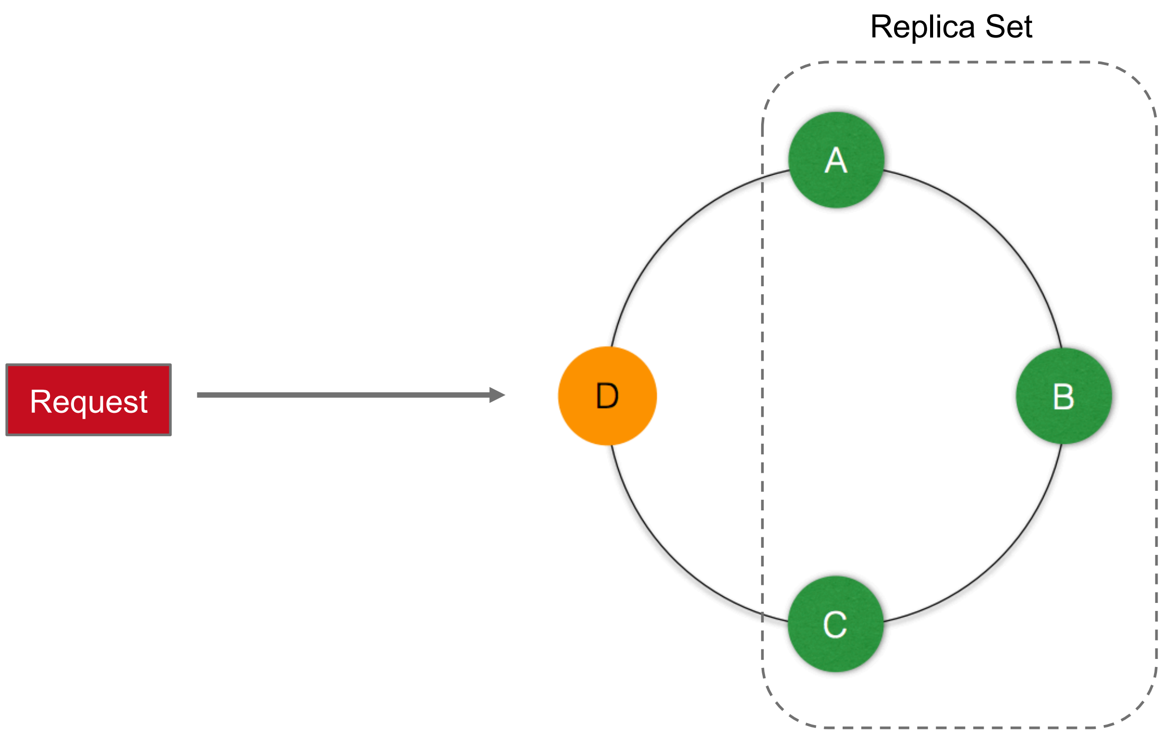
Cassandra Architecture: Consistency Level = ONE

read



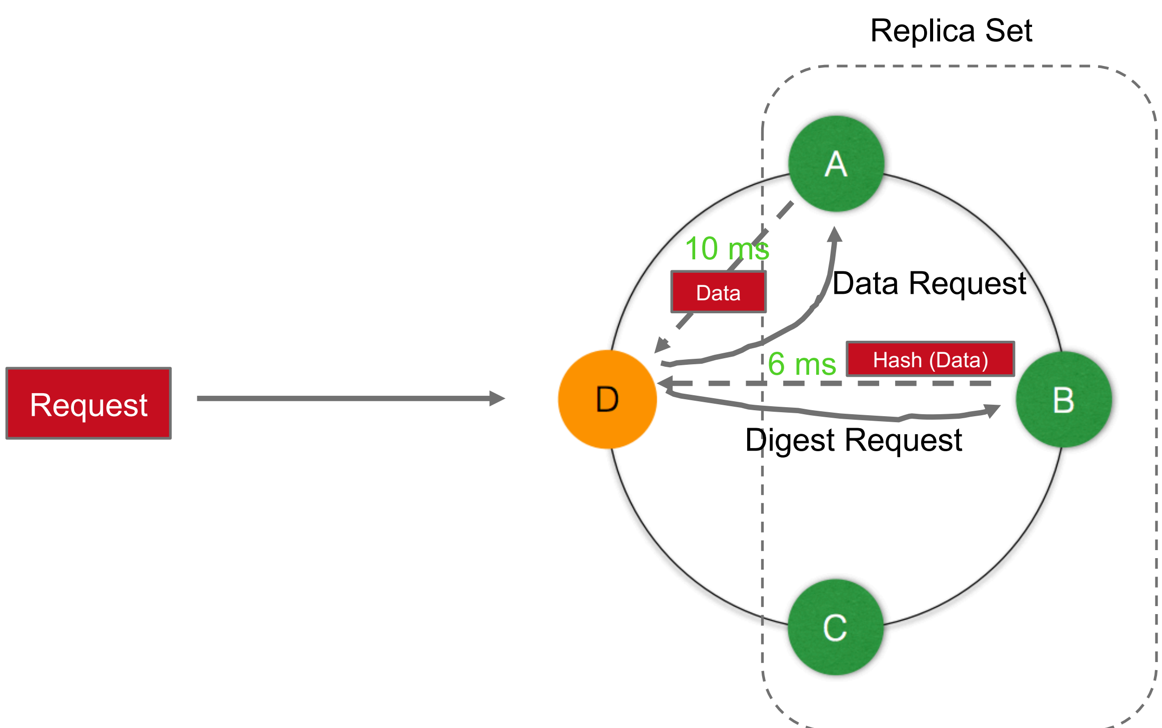
Cassandra Architecture: Consistency Level = QUORUM

read



Cassandra Architecture: Consistency Level = QUORUM

read



Cassandra Architecture: Consistency Compromise

QUORUM

ALL

ONE

Latency

Consistency

Technische Universität Berlin

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Consistency Compromise

Latency

Consistency

Quorum

Cassandra Architecture: Consistency Compromise

Latency Consistency

ALL

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra Architecture: Consistency Compromise

Latency Consistency

ONE

Cassandra Architecture: Local Persistence

Write

- Append to commit log for durability (recoverability)
- Update of in-memory, per-column-family Memtable

If Memtable crosses a threshold

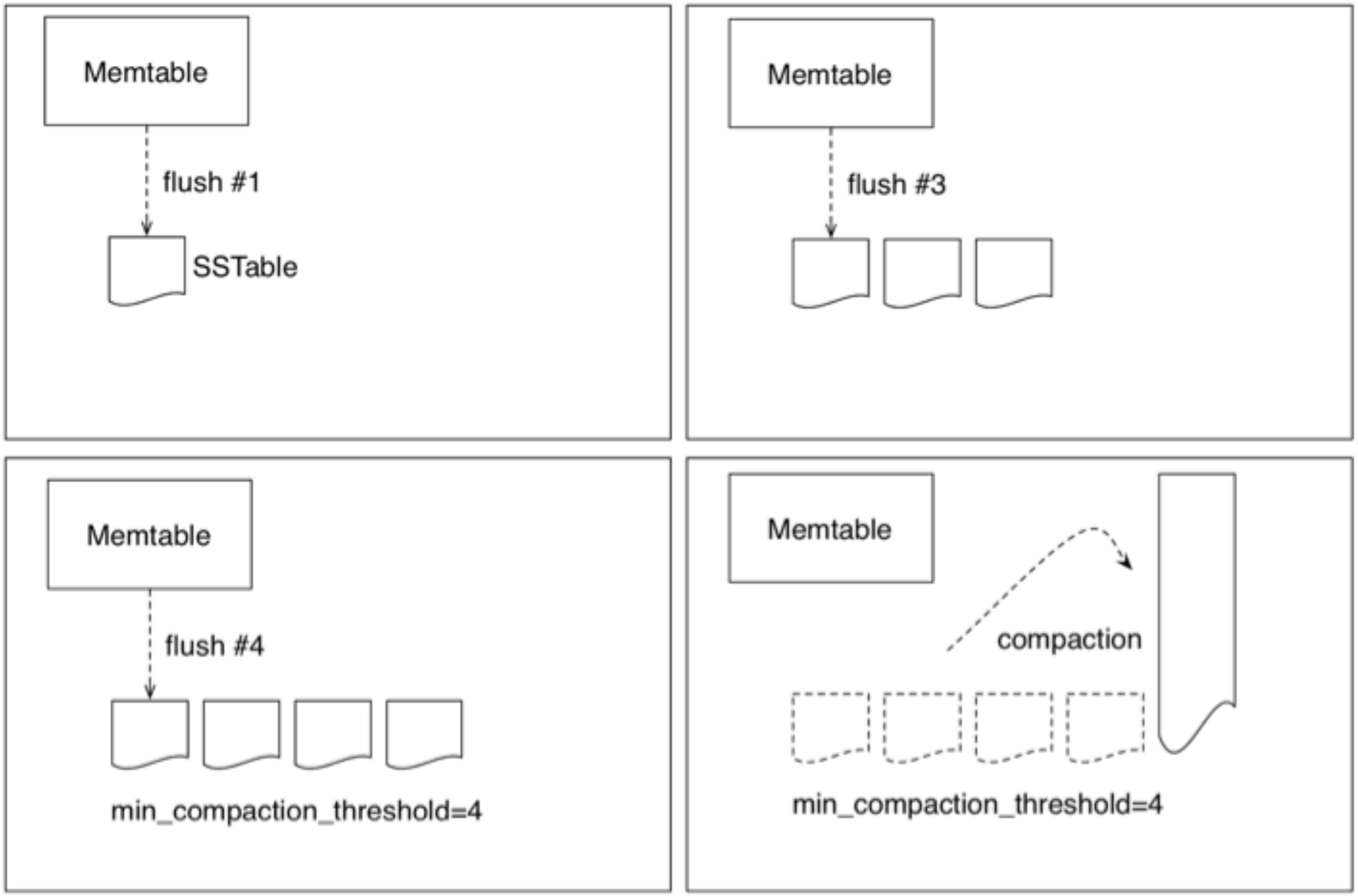
- Sequential write to disk (SSTable).
- Merge SSTables from time to time (compactions)

Cassandra Architecture: Local Persistence

Read

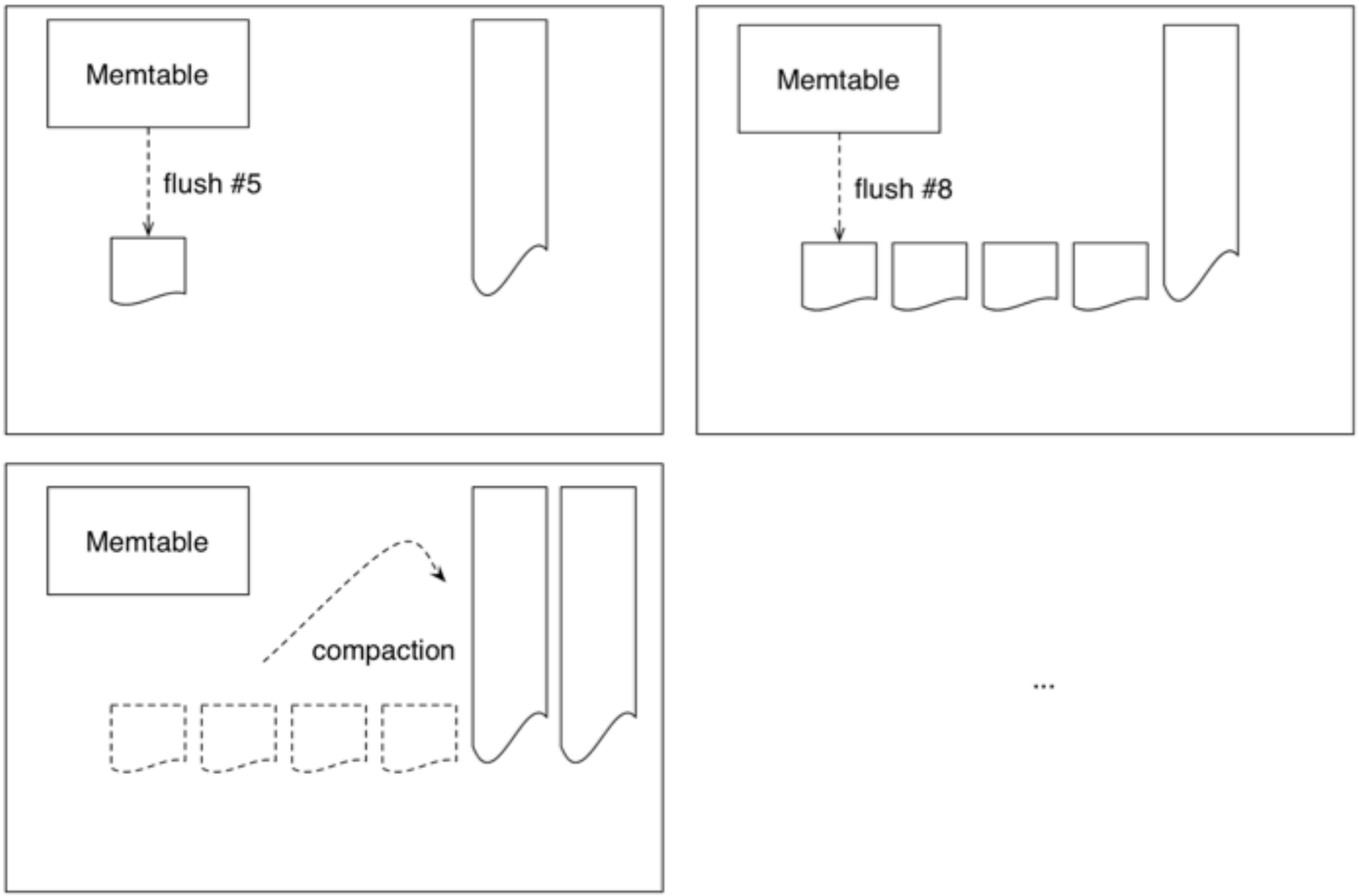
- Query in-memory Memtable
- Check in-memory bloom filter
  - Used to prevent unnecessary disk access.
  - A bloom filter summarizes the keys in a file.
  - False Positives are possible
- Check column index to jump to the columns on disk as fast as possible.
  - Index for every 256K chunk.

Cassandra Architecture: Local Persistence

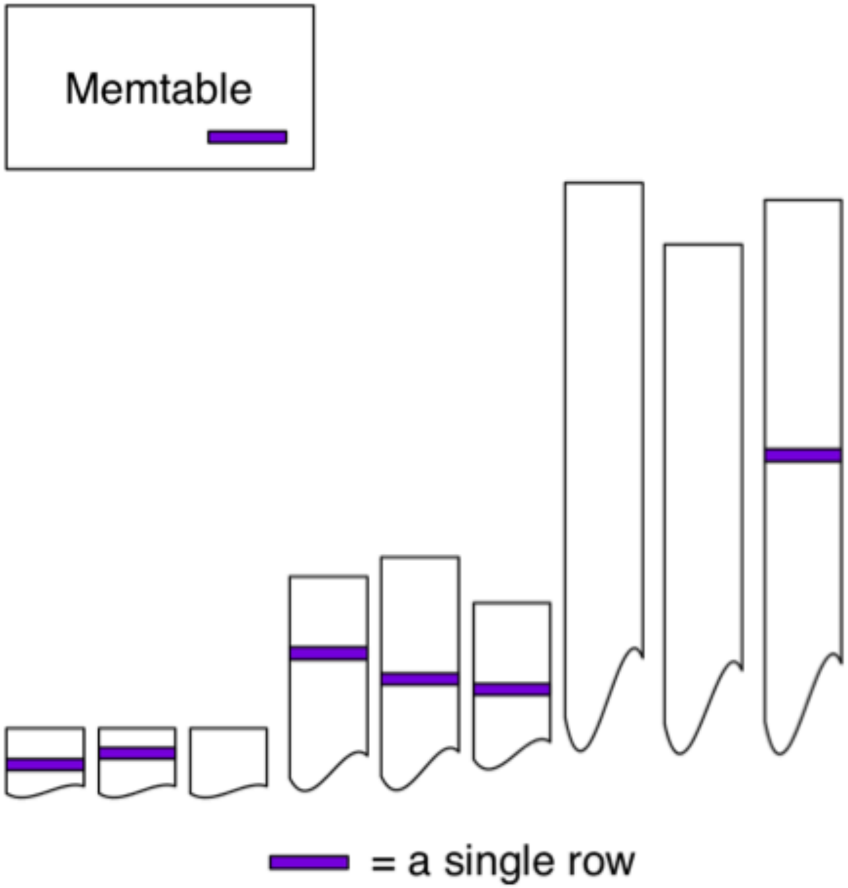


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

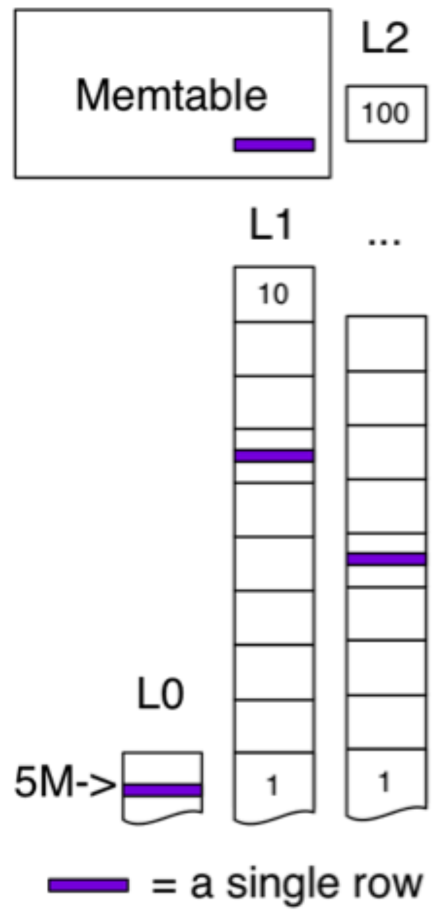
Cassandra Architecture: Local Persistence



Cassandra Architecture:
Local Persistence – Size Tiered Compactions



Cassandra Architecture: Local Persistence – Leveled Compactions



Summary

Cassandra implements a scalable and highly available replication architecture very similar to Amazon Dynamo and a write-optimized storage engine very similar to BigTable.

Cassandra has evolved since 2010 from an open source project into a robust database solution.

- Main contributor: http://www.datastax.com
- New features (more than Dynamo + BigTable)
  - Cassandra Query Language (CQL), “lightweight transactions”, etc.
  - Architecture improvements (Leveled Compactions, etc.)
  - Integration with other systems, such as Hadoop

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Cassandra

TOPICS

1. Cassandra Architecture & Operation (lecture)
2. Cassandra Data Model (exercise)
3. Cassandra Query Language (exercise)

Reference:
Avinash Lakshman and Prashant Malik. 2010. Cassandra: a decentralized structured storage system. SIGOPS Oper. Syst. Rev. 44, 2 (April 2010), 35-40.

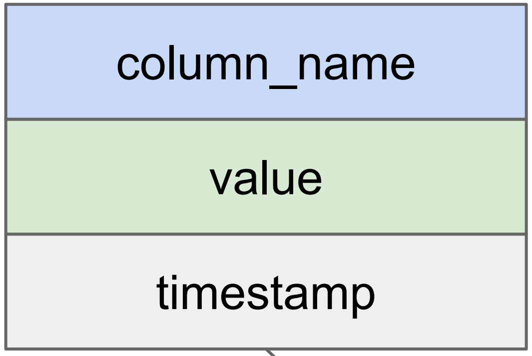
ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Data Model

- Keyspace (Database)
- Column Family (Table)
- Keys and Columns

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

A column



the timestamp field is
used by Cassandra for conflict
resolution: “Last Write Wins”

Column family (Table)

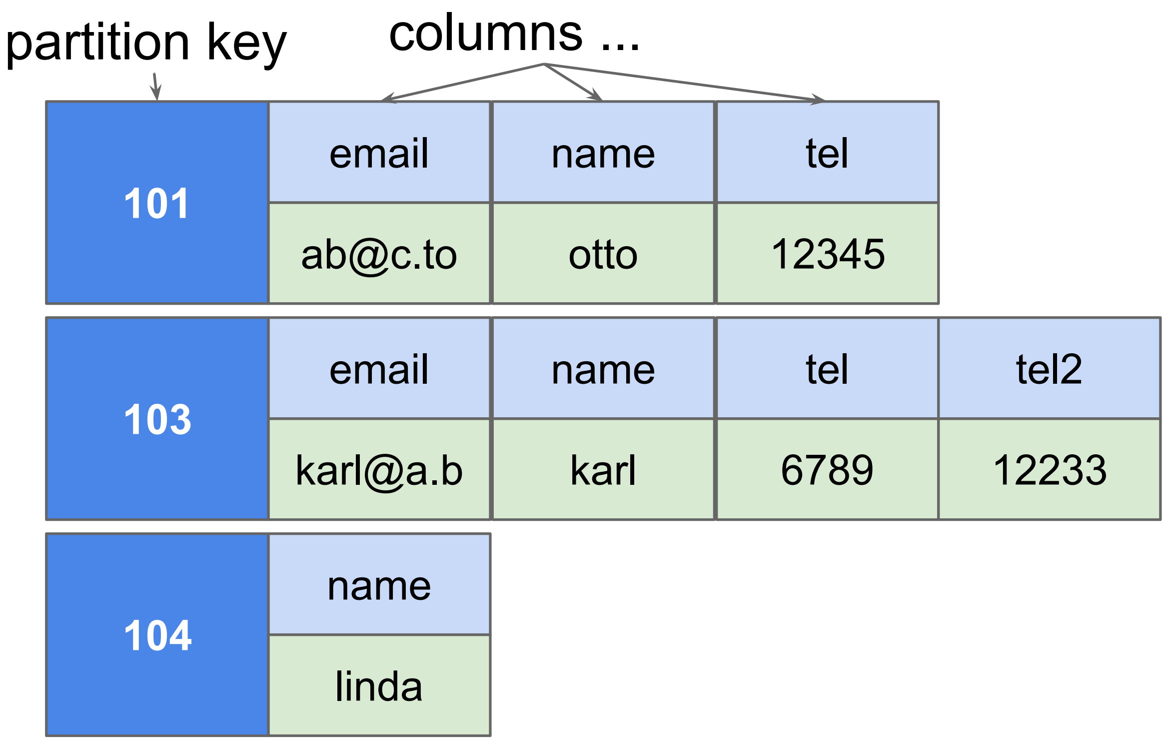


Table with standard PRIMARY KEY

```
CREATE TABLE messages (
  msg_id timeuuid PRIMARY KEY,
  author text,
  body text
 );
```


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Table: Tweets

PRIMARY KEY
= msg_id

9990	author	body
	otto	Hello World!
9991	author	body
	linda	Hi, Otto

Table with compound PRIMARY KEY

```
CREATE TABLE timeline (
  user_id uuid,
  msg_id timeuuid,
  author text,
  body text,
  PRIMARY KEY (user_id, msg_id)
);
```


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

"Wide-row" Table: Timeline

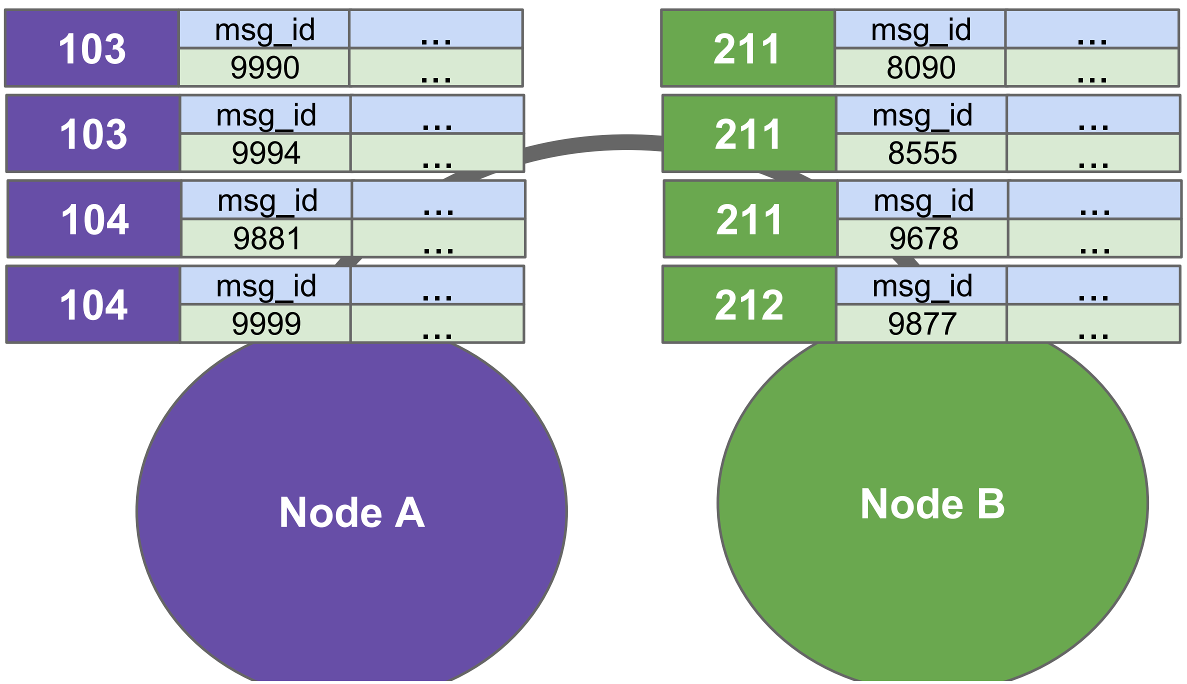
PRIMARY KEY = user_id + msg_id

partition key column

103	msg_id	author	body
	9990	otto	Hello World!

103	msg_id	author	body
	9991	linda	Hi @otto

Timeline Table is partitioned by user and locally clustered by msg



ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Comparison: RDBMS vs. Cassandra

RDBMS Data Design

Users Table

user_id	name	email
101	otto	o@t.to

Tweets Table

tweet_id	author_id	body
9990	101	Hello!

Followers Table

id	follows_id	followed_id
4321	104	101

Cassandra Data Design

Users Table

user_id	name	email
101	otto	o@t.to

Tweets Table

tweet_id	author_id	name	body
9990	101	otto	Hello!

Follows Table

user_id	follows_list
104	[101,117]

Followed Table

id	followed_list
101	[104,109]

Intro: CLI, CQL2, CQL3

Technische Universität Berlin

- CQL is “SQL for Cassandra”
- Cassandra CLI deprecated, CQL2 deprecated
- CQL3 is default since Cassandra 1.2

```
$ cqlsh
```

- Pipe scripts into cqlsh

```
$ cat cql_script | cqlsh
```

- Source files inside cqlsh

```
cqlsh> SOURCE '~/cassandra_training/cql3/
               01_create_keyspaces';
```


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

CQL3

- Create a keyspace
- Create a column family
- Insert data
- Alter schema
- Update data
- Delete data
- Apply batch operation
- Read data
- Secondary Index
- Compound Primary Key
- Collections
- Consistency level
- Time-To-Live (TTL)
- Counter columns
- sstable2json utility tool

Create a SimpleStrategy keyspace

- Create a keyspace with `SimpleStrategy` and `"replication_factor"` option with value `"3"` like this:

```
cqlsh> CREATE KEYSPACE <ksname>
          WITH REPLICATION =
          {'class':'SimpleStrategy',
          'replication_factor':3};
```


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Create a NetworkTopologyStrategy keyspace

Create a keyspace with `NetworkTopologyStrategy` and strategy option `"DC1"` with a value of `"1"` and `"DC2"` with a value of `"2"` like this:

```
cqlsh> CREATE KEYSPACE <ksname>
          WITH REPLICATION = {
            'class':'NetworkTopologyStrategy',
            'DC1':1,
            'DC2':2
          };
```

Create a Table “users”

- Connect to the "twotter" keyspace.

```
cqlsh> USE twotter;
```

- Create new column family (Table) named "users".

```
cqlsh:twotter> CREATE TABLE users (
                   id int PRIMARY KEY,
                   name text,
                   email text
                   );
cqlsh:twotter> DESCRIBE TABLES;
cqlsh:twotter> DESCRIBE TABLE users;
```

*we use int instead of uuid in the exercises for the sake of readability

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Create a Table “messages”

Technische Universität Berlin

- Create a new Table named "messages" with the attributes "posted_on", "user_id", "user_name", "body", and a primary key that consists of "user_id" and "posted_on".

```
cqlsh:twotter> CREATE TABLE messages (
   posted_on bigint,
   user_id int,
   user_name text,
   body text,
   PRIMARY KEY  (user_id, posted_on)
);
```

*we use bigint instead of timeuuid in the exercises for the sake of readability

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Insert data into Table “users” of keyspace “twotter”

```
cqlsh:twotter>
INSERT INTO users(id, name, email)
VALUES (101, 'otto', 'otto@abc.de');

cqlsh:twotter> ... insert more records ...

cqlsh> SOURCE
  '~/cassandra_training/cql3/03_insert';
```

Read data

```
cqlsh:twotter> SELECT * FROM users;

     id  | email          | name
    -----+----------------+-------
     105 |    g@rd.de     |  gerd
     104 | linda@abc.de | linda
     102 |        null    |  jane
     106 | heinz@xyz.de | heinz
     101 |  otto@abc.de |  otto
     103 |        null    |  karl
```


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Update data

```
cqlsh:twotter> UPDATE users
               SET email = 'jane@smith.org'
               WHERE id = 102;

 id   | email                | name
-----+----------------+-------
  105 |    g@rd.de       |  gerd
  104 |   linda@abc.de | linda
  102 | jane@smith.org |  jane
  106 |   heinz@xyz.de | heinz
  101 |  otto@abc.de    |  otto
  103 |      null        |  karl
```

Delete data

- Delete columns

```
cqlsh:twotter> DELETE email
                 FROM users
                 WHERE id = 105;
```

- Delete an entire row

```
cqlsh:twotter> DELETE FROM users
                 WHERE id = 106;
```


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Delete data

```
id  | email            | name
-----+-----------------+-------
105 |      null        |  gerd
104 |    linda@abc.de  | linda
102 | jane@smith.org   |  jane

101 |   otto@abc.de    |  otto
103 |      null        |  karl
```


ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Alter Table Schema

```
cqlsh:twotter>
ALTER TABLE users ADD password text;

cqlsh:twotter>
ALTER TABLE users
  ADD password_reset_token text;
```

* Given its flexible schema, Cassandra's CQL ALTER finishes much quicker than RDBMS SQL ALTER where all existing records need to be updated.

Alter Table Schema

id	email	name	password	password_reset_token
107	j@doe.net	john	null	null
108	michael@abc.de	michael	null	null
104	linda@abc.de	linda	null	null
102	jane@smith.org	jane	null	null
101	otto@abc.de	otto	null	null
103	null	karl	null	null

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering