Bases de données spécialisées

Cassandra, the server side story

2025-2026
Giovanni Bernardi, gioXYZirif.fr
http://www.irif.fr/~gio/index.xhtml

Slow reponse time implies

???

Slow reponse time implies being out of business

16 Aug 2023

Twitter appears to delay links by five seconds to sites Elon Musk dislikes

Warm-up

Why weak consistency ?

Why weak consistency ?

Theorem (CAP, Gilbert, Lynch 2002)

It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:

- *Availability*
- *Atomic Consistency*

in all fair executions (including those in which messages are lost).

Further material:

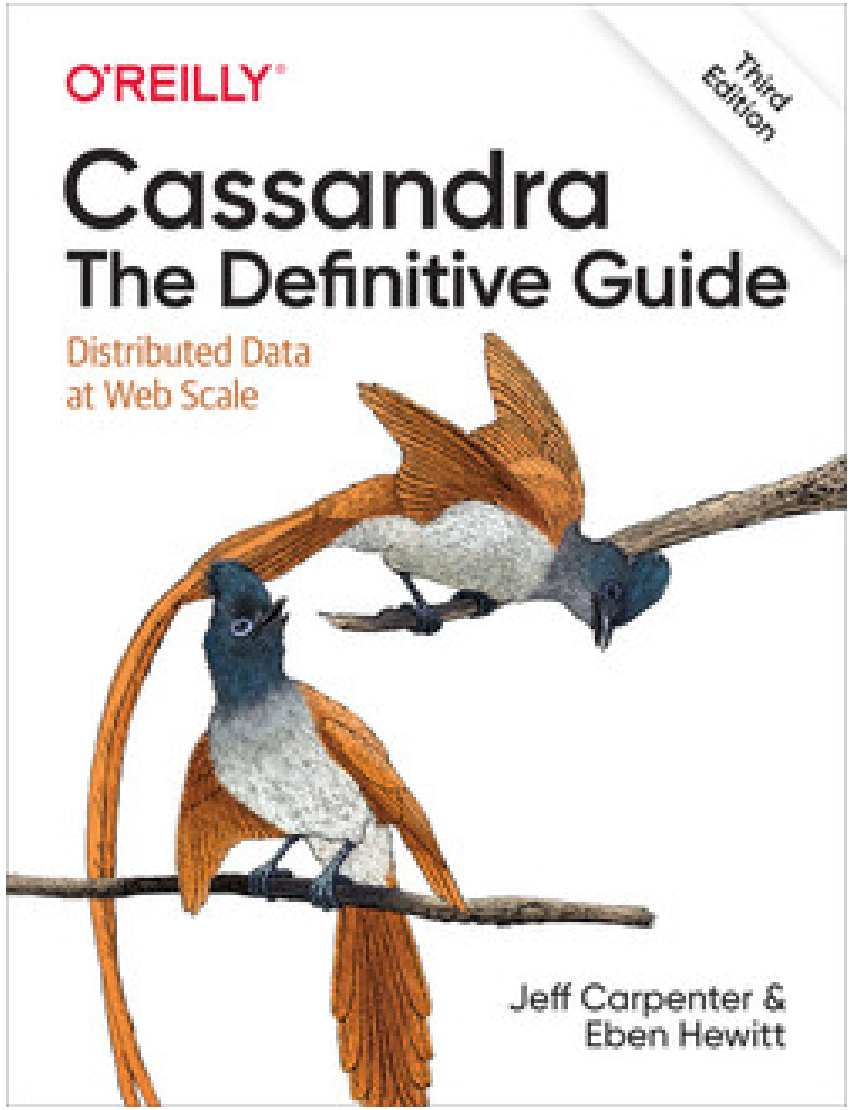
- Account by IBM at `https://www.ibm.com/cloud/learn/cap-theorem`
- Perspectives on the CAP Theorem, S. Gilbert, N. A. Lync
- A Critique of the CAP Theorem, M. Kleppmann

What is an anomaly ?

TPC-C

wiki


cassandra


O'REILLY
Third Edition
Cassandra
The Definitive Guide
Distributed Data at Web Scale
Jeff Carpenter & Eben Hewitt

What is Cassandra ? 1/5

Apache Cassandra is an open source, **distributed**, **decentralized**, *elastically scalable,* **highly-available**, *fault-tolerant, tuneably consistent, row-orieted database.*

Cassandra bases its distribution design on Amazon's Dynamo and its data model on Google's Bigtable, with a query language similar to SQL.

Cassandra: The Definitive Guide
J. Carpenter, E. Hewitt

What is Cassandra ? 2/5

Cassandra is a **distributed** *storage system*
for managing very large amounts of structured data
spread out across many commodity servers,
while providing **highly-available** *service*
with **no single point** *of failure.*

Cassandra - A decentralized Structured Storage System,
A. Lakshman, P. Malik
ACM SIGOPS 2010

What is Cassandra ? 3/5

Apache Cassandra is a massively scalable NoSQL database.

Noteworthy features:

Replication
: built-in, customizable, and easy to configure

Data consistency
: tunable across a database cluster,
it can be handled on a per operation basis

White paper by Datastax at

`http://www.odbms.org/wp-content/uploads/2014/06/WP-IntroToCassandra.pdf`

What is Cassandra ? 4/5

> *Cassandra is a distributed* **key value store** *[. . .] optimized for write heavy workload while maintaining a good performance for reads.*

Noteworthy objectives:

- cross-data center replication
  to achieve good read efficiency
- fault tolerance
- scaling of platform by adding commodity servers
  no ad-hoc machines

See `https://medium.com/coinmonks/cassandra-distributed-key-value-store-optimized-for-write-heavy-workloads-77f69c01388c`

What is Cassandra ? 5/5

Cassandra is a **highly-scalable partitioned** *row store.*
Rows are organized into tables with a required primary key.

Partitioning Cassandra can distribute your data across multiple machines in an application-transparent matter. Cassandra will automatically repartition as machines are added and removed from the cluster.

Row store Cassandra organizes data by rows and columns. The Cassandra Query Language (CQL) is a close relative of SQL.

See official README

Analogies with RDBMS

Relational Model	Cassandra Model
Database	Keyspace
Table	Column Family
Primary key	Row key
Column name	Column name/key
Column value	Column value

Built-in replication

```
CREATE KEYSPACE     myStorage
WITH replication = {'class':'SimpleStrategy',
                    'replication_factor' :  3 };
```

replication_factor
=
number of copies of every row
=
3

Built-in replication

More realistic example

```
CREATE KEYSPACE properStorage
WITH replication = {'class':'NetworkTopologyStrategy',
                    'datacenter1' :  3
                    'datacenter2' :  2 };
```

replication_factor

=

??

Built-in replication

More realistic example

```
CREATE KEYSPACE properStorage
WITH replication = {'class':'NetworkTopologyStrategy',
                    'datacenter1' :  3
                    'datacenter2' :  2 };
```

$$\texttt{replication\_factor} = \sum \texttt{replication\_factor} \text{ per datacenter} = 3 + 2$$

Key-value store

A column family has the Java type

```
Map<RowKey,
        SortedMap<ColumnKey, ColumnValue>>
```

- In relational DB all the rows in a table have the same amount of columns, even when they contain no data.
- In a key-value store the rows of a same table may have different amount of columns.

Cassandra is **not** a general-purpose database

recall that this is a course on

Bases de données **spécialisées**

Cassandra

- deals badly with updates and deletes,
- it does not perform joins,
- it offers neither foreign keys nor referential integrity,
- works best in append-only scenarios, for instance
  - time-series data
  - event sourcing architecture[1]

  example: on-line messaging how often do you modify messages ?

A down to earth description of pros and cons at

`https://blog.pythian.com/cassandra-use-cases/`

[1] Event Sourcing is a pattern for data storage, where instead of storing the current state of any entity, all past changes to that state are stored.

Is Cassandra valued in industry ?

According to Dice, Cassandra is in the top 10 paying skills in

... [2] 2014 2015 2016 2017 2019

In the 2018 report it is the 11^{th} skill, and in the 2021 report it is 30^{th}.

Take the above data with a grain of salt...

- Can you find analogous data for the Eurasian job market ?
- Con you find data about more recent years ?

[2] I have not checked older reports.

@ Facebook

June 20, 2008

> *This month, we're rolling out Inbox search, a tool which allows you to search through all of your messages either by name of the person who sent it, or by a keyword that shows up in the text. . . .*
>
> https://www.facebook.com/notes/facebook/inbox-search/20387467130/

- Cassandra: a decentralized structured storage system, ACM SIGOPS, '10
- Cassandra: structured storage system on a P2P network, PODC '09

A. Lakshman, P. Malik

@ Netflix

Nov 02, 2011

Netflix is using Cassandra on AWS as a key infrastructure component of its globally distributed streaming product. Cassandra scales linearly far beyond our current capacity requirements, and very rapid deployment automation makes it easy to manage.

https://netflixtechblog.com/

benchmarking-cassandra-scalability-on-aws-over-a-million-writes-per-second-39f45f066c9e

- Talk by A. Cockcroft at Cassandra Summit 2012

@ Spotify

January 9, 2015

When we started shopping around for different storage solutions for UPS[3] we desired a solution which could:

- *Scale horizontally*
- *Support replication – preferably cross-site*
- *Have low latency even at the cost of consistency since we aren't performing transactions*
- *Have a decent ability to model different data schemas for different use-cases of entity metadata*
- *...*

Only Cassandra fit the bill for all these requirements.

https://engineering.atspotify.com/2015/01/09/personalization-at-spotify-using-cassandra/

[3] User Profile Store

@ Discord

January 13, 2017

In July, we announced 40 million messages a day, in December we announced 100 million, and as of this blog post we are well past 120 million. [. . .]
This is a lot of data that is ever increasing in velocity, size, and must remain available. How do we do it? Cassandra!

Our requirements:
Linear scalability, Automatic failover, Low maintenance, Open source.

https://discord.com/blog/how-discord-stores-billions-of-messages

https://www.meetup.com/
cassandra-paris-meetup/

Remaining part of this lecture

General overview of how the Cassandra server works

Theory

- Introduce necessary terminology
- Sketch the main ideas / features
- Point to the relevant literature
- Point to the relevant code

Cassandra is open source 

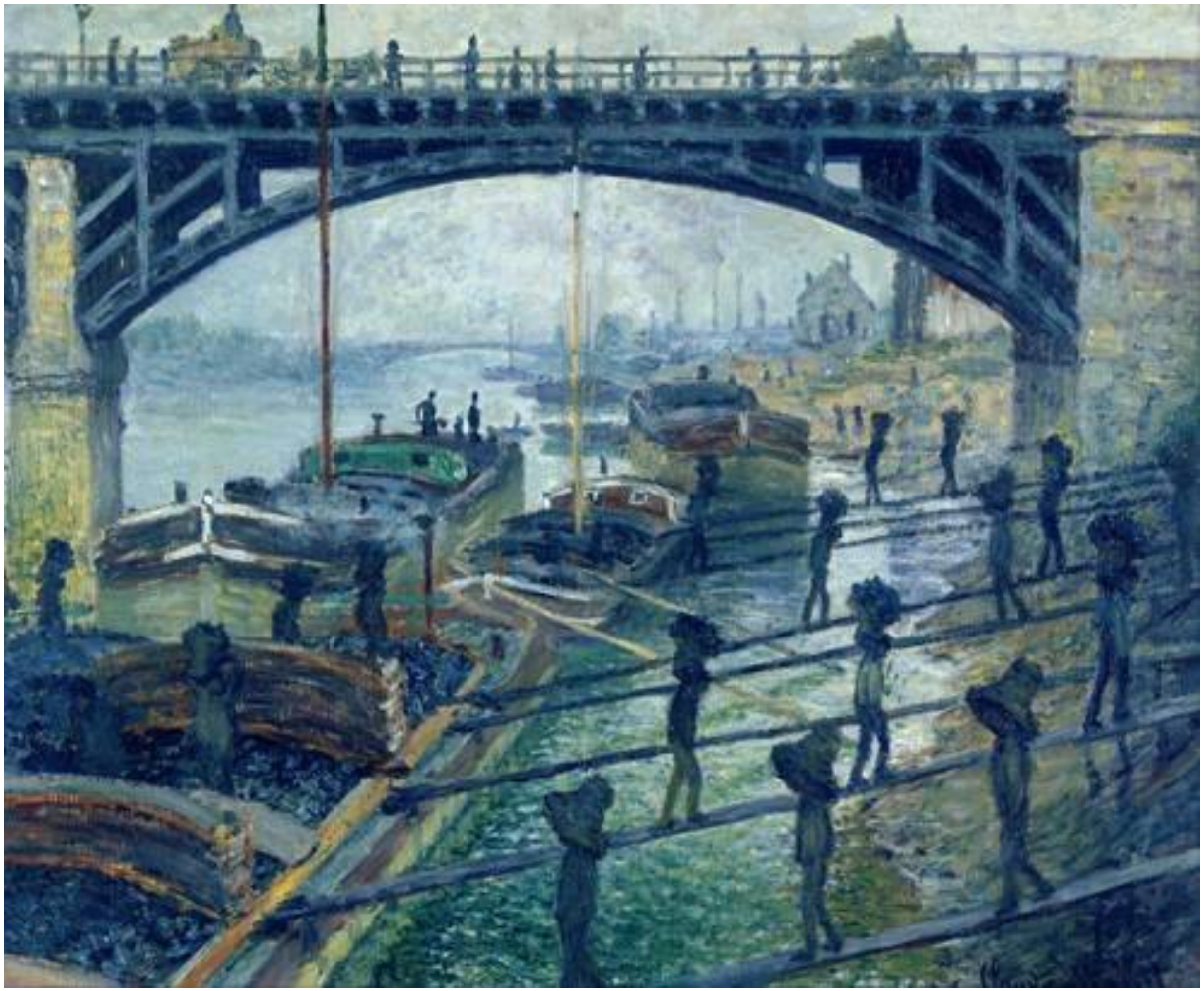
Practice

- Minimal examples of Tools to manage a node
- Point to further documentation

THEORY

WHY THEORY ?

Technology changes

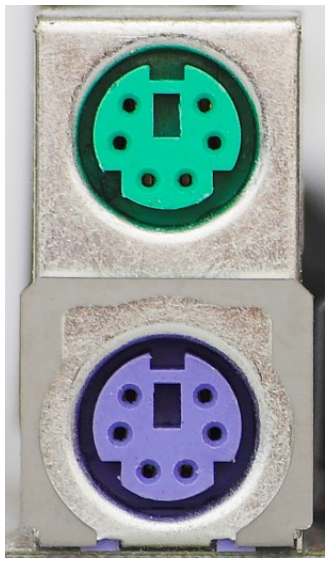


Les Déchargeurs de charbon, Claude Monet, 1875

Technology changes

world's first HDD

Technology changes



Theory is forever

- Who proved that primes are infinite, and when?
- Which book introduced the Fibonacci series, and when ?
- What was rediscovered in 1994 ?

[4] Filius Bonacci

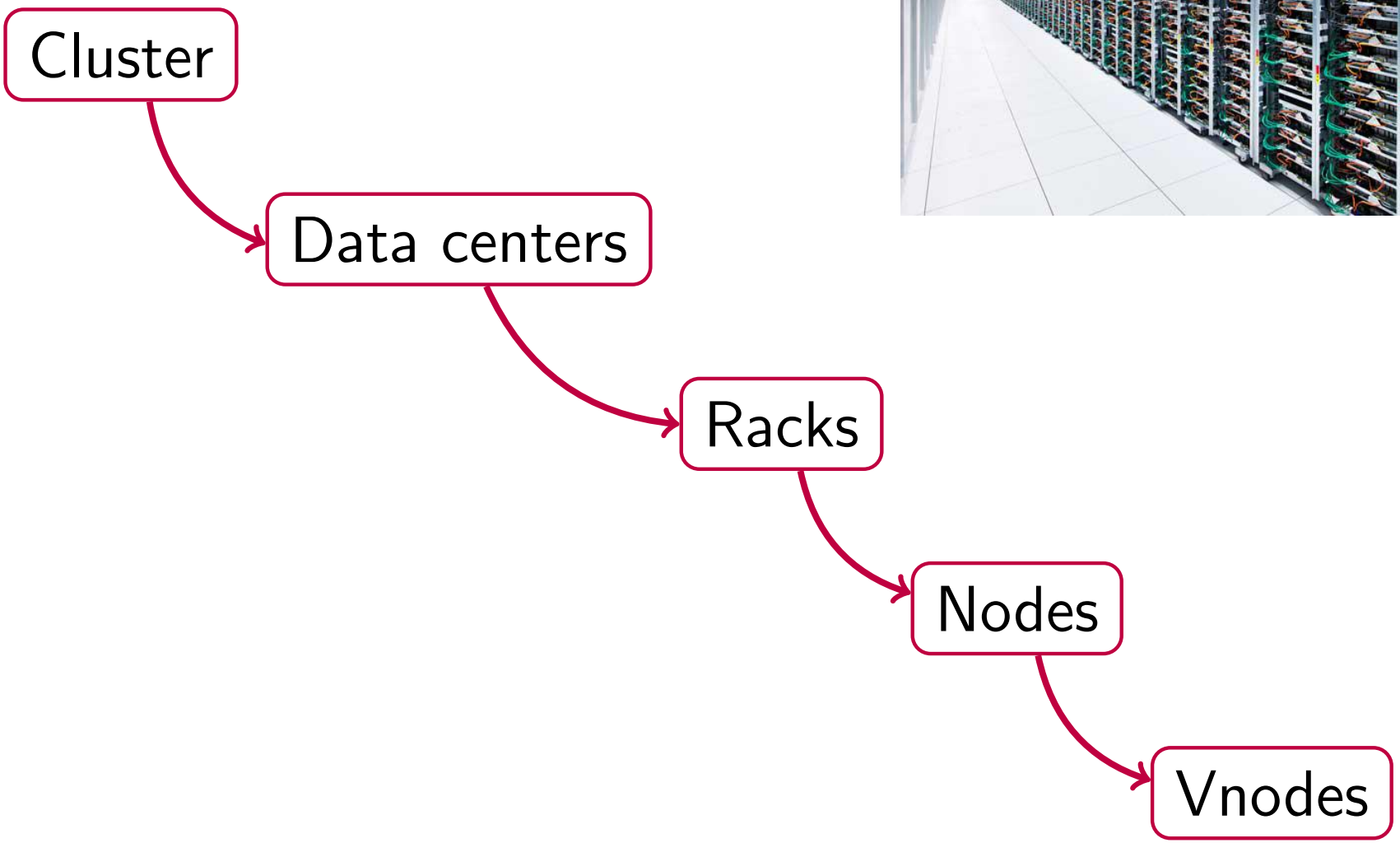
Theory is forever

- Who proved that primes are infinite, and when?
  Elements, Euclid, 300 BC
- Which book introduced the Fibonacci series, and when ?
  Liber Abaci, Leonardo da Pisa[4], 1202 AD
- What was rediscovered in 1994 ?
  The trapezoidal rule
  Already used in Babylon before 50 BC

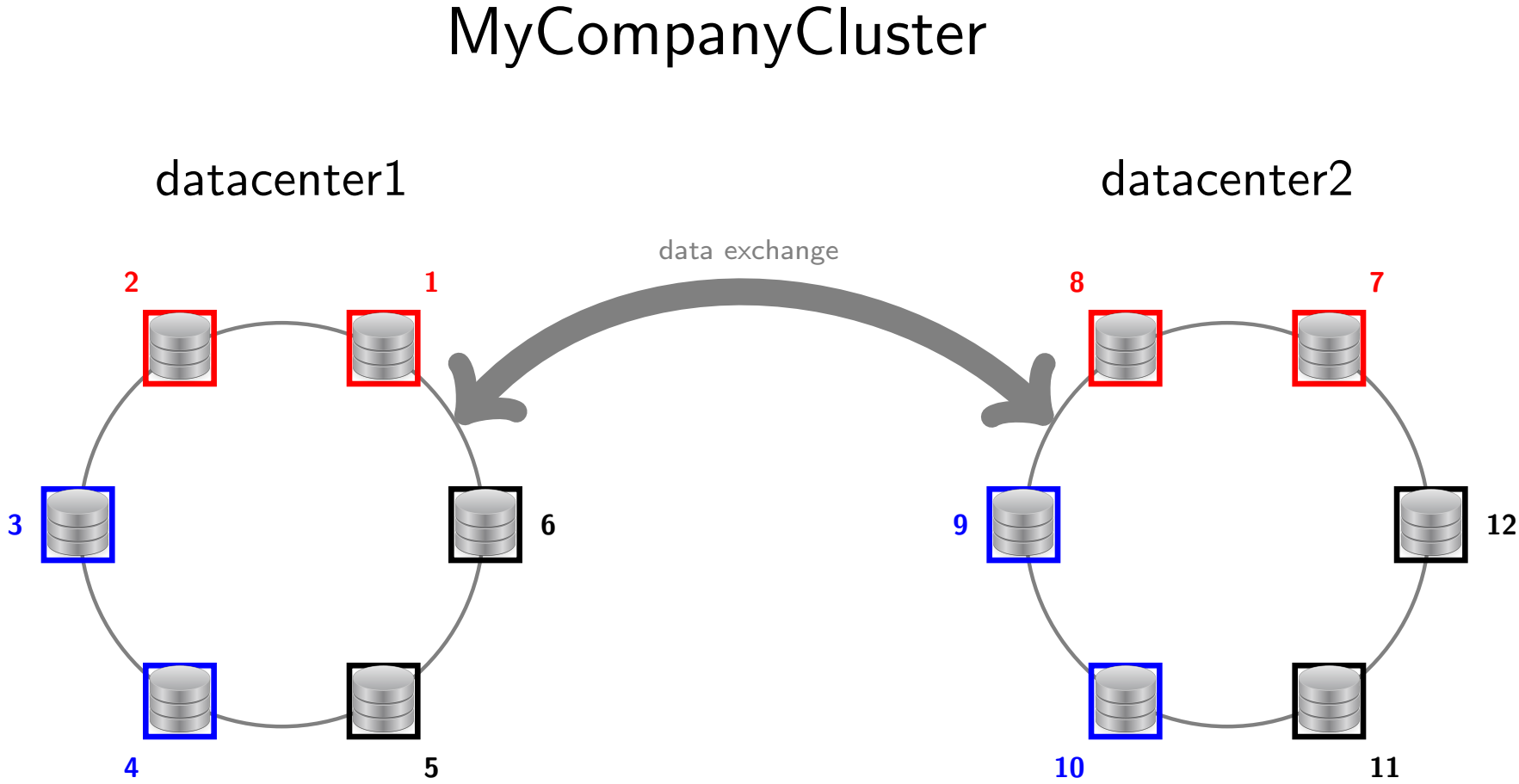
[4] Filius Bonacci

Node organisation, physical

Top - down



Node organisation, **logical standpoint**



Notice the same node colours in every data center
they represent the same rack ids: rack1, rack2, rack3.

Cluster

Set of all the servers in the storage system

- Spans usually different physical locations / dacenters
- Contains the definitions of the keyspaces — databases
- Every keyspace has a replication factor wrt datacenters
  in the same cluster
- i.e. no cross-cluster replication

Data center

Synonym of "replication group"

- Logical grouping of nodes configured together with a view to replication
- Non-empty set of racks — at least one rack per datacenter

Rack

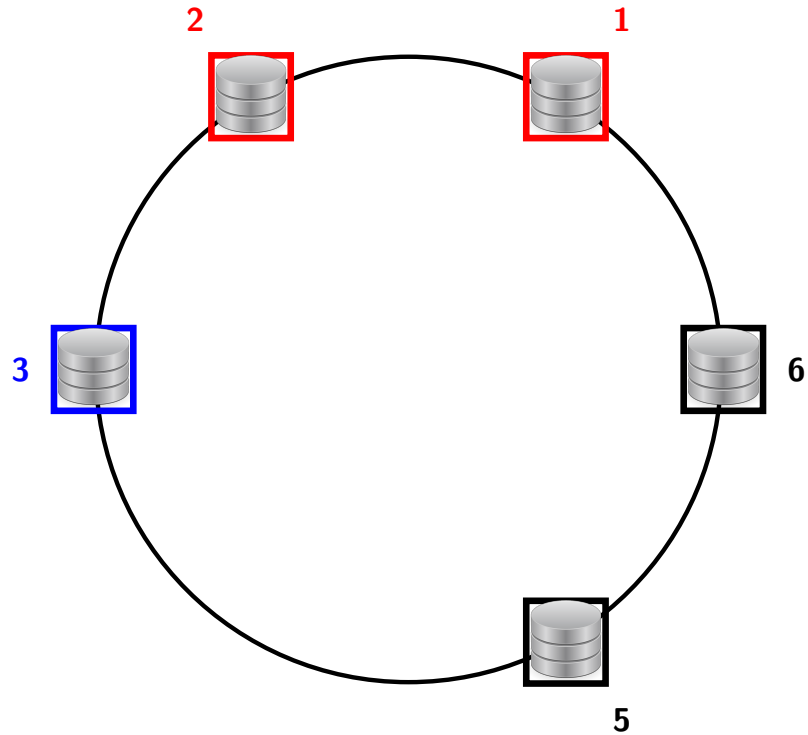
- Logical set of nodes

Node aka Endpoint

- A Cassandra instance that runs in a machine and that
- manages a range of tokens — default: 256
- runs virtual nodes — more on virtual nodes later

How does a node join a cluster ?

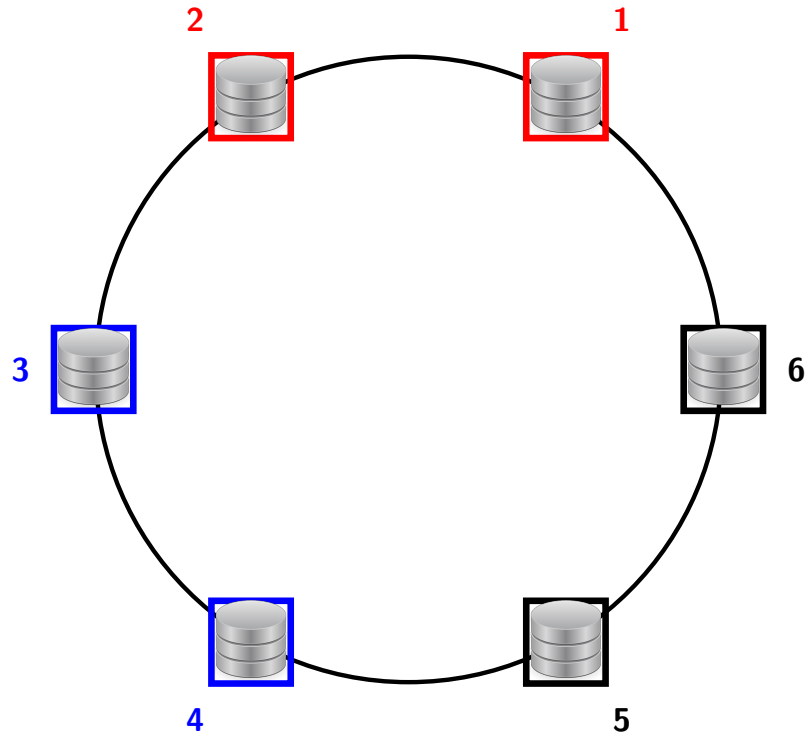
Node bootstrap



part of Source code: StorageService.java

see initServer()

Node bootstrap

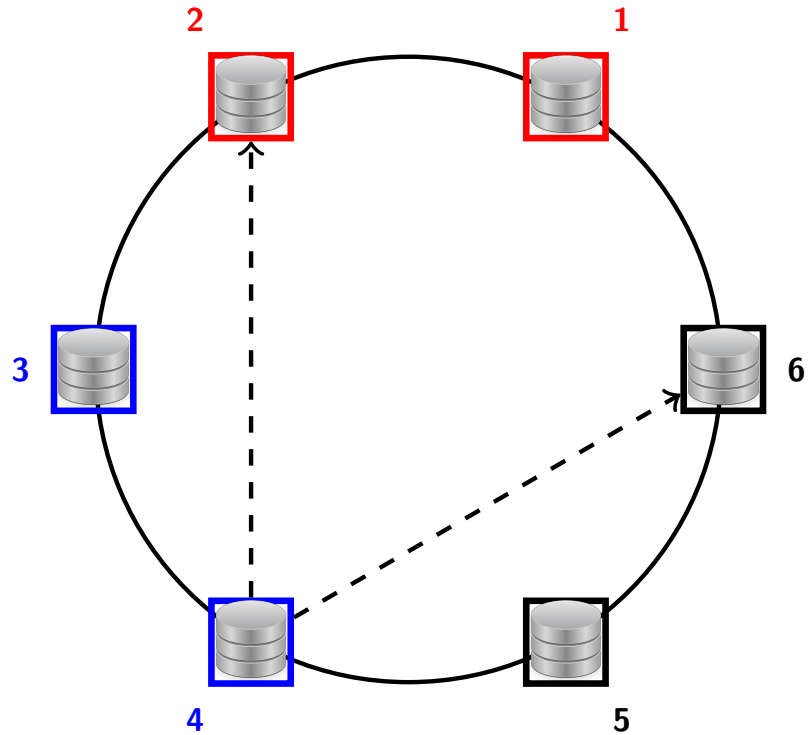


Node 4 goes on-line, it is in state JOINING.

part of Source code: StorageService.java

see initServer()

Node bootstrap



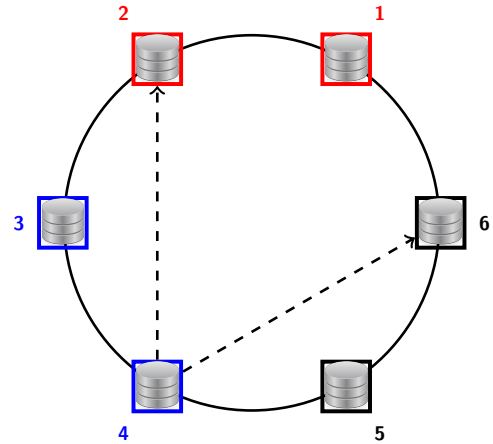
Node 4 contacts at least one seed node
to learn the state of the cluster, and the schemas of the keyspaces.

part of Source code: StorageService.java see initServer()

Node bootstrap

Seed node

A node in the cluster
contacted by nodes joining the cluster
to bootstrap the gossip protocol



The designation of "seed" has no purpose beyond bootstrapping

any node can be made seed at any time

What if there is no seed node ?

- a cluster can boot and operate without seeds
- in this case no new nodes can be added to the cluster.

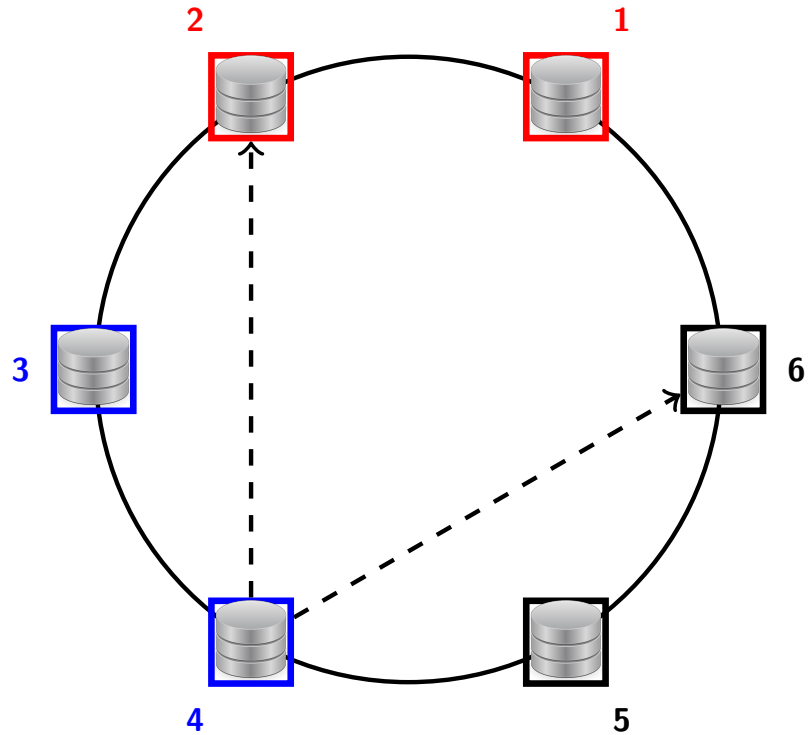
seeds are not points of failure

Recommended usage of seeds

1. pick at least two nodes per data center as seeds
2. sync the seed list to every node

see file `cassandra.yaml`

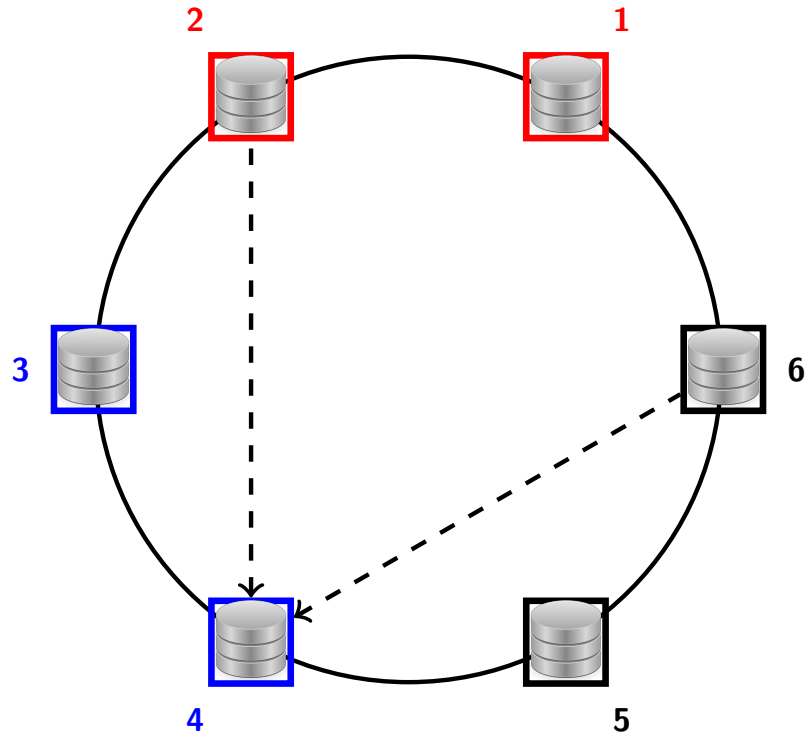
Node bootstrap



Node 4 contacts at least one seed node
to learn the state of the cluster, and the schema.

part of Source code: StorageService.java see initServer()

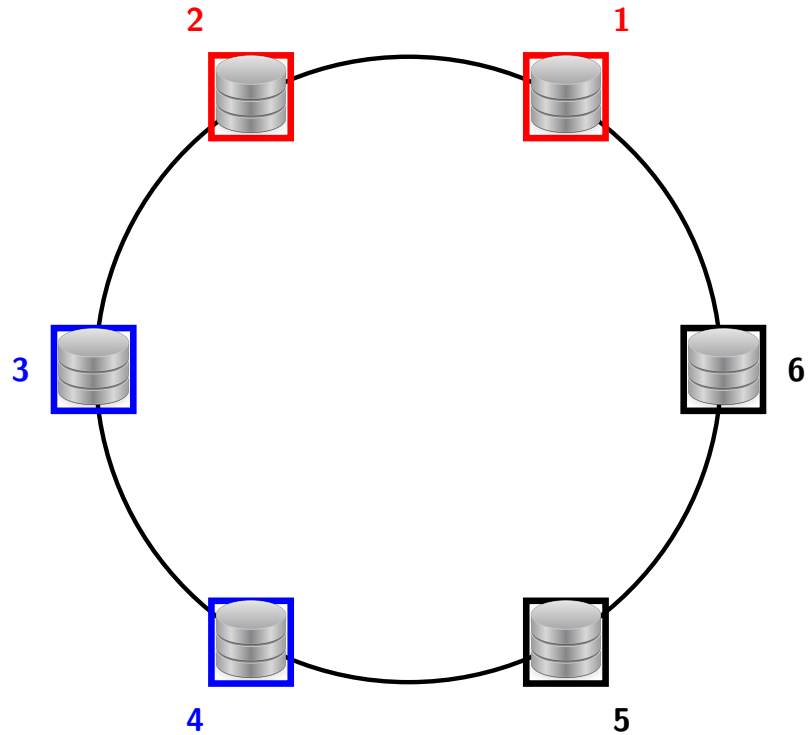
Node bootstrap



Node 4 receives information on existing nodes, their tokens, and the schema.

part of Source code: StorageService.java see initServer()

Node bootstrap

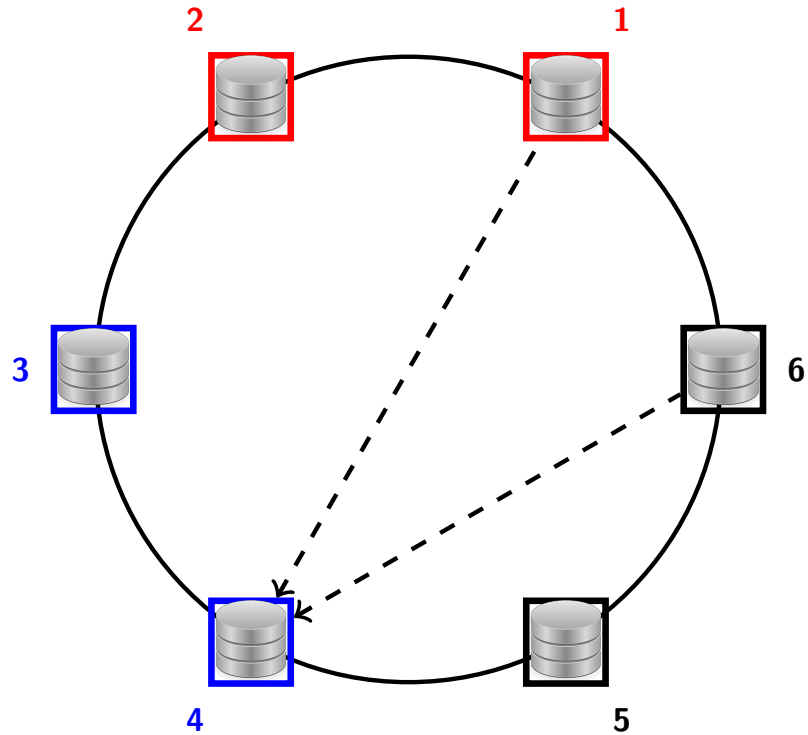


Node 4 computes the token range that it is responsible for.

part of Source code: StorageService.java

see initServer()

Node bootstrap

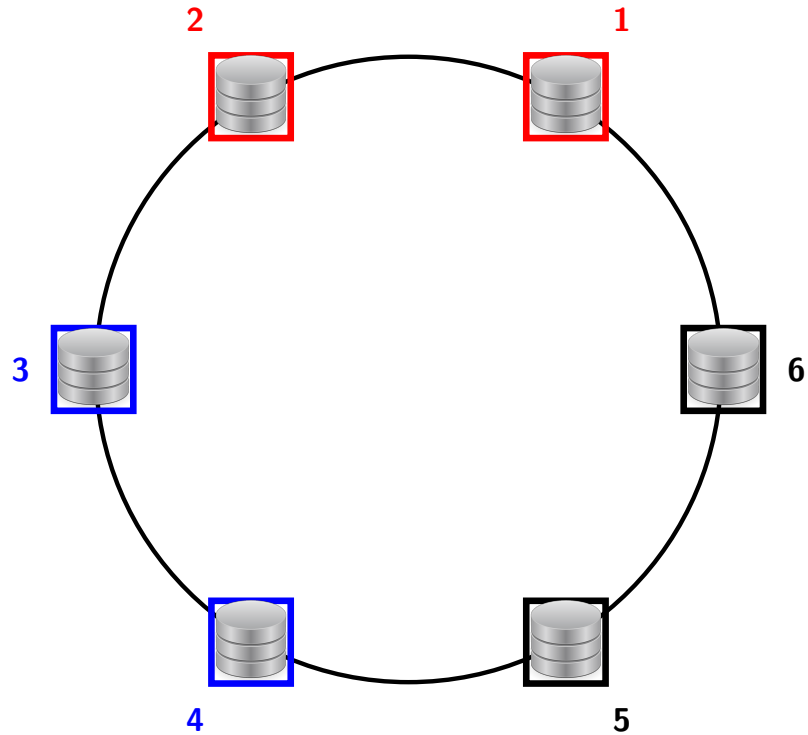


Former owners stream data to node 4.

part of Source code: StorageService.java

see initServer()

Node bootstrap

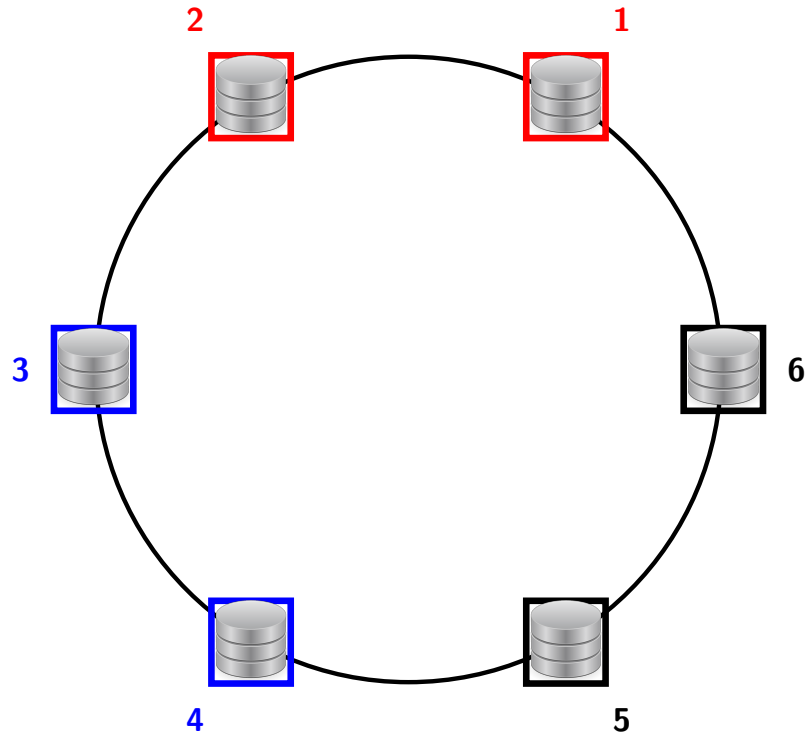


Node 4 transitions from JOINING to NORMAL state.

part of Source code: StorageService.java

see initServer()

Node bootstrap



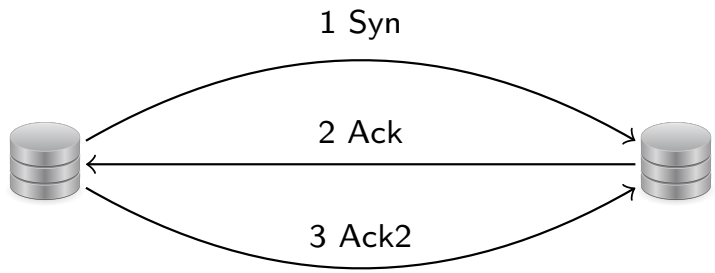
Node 4 begins listening for CQLSH requests on port 9042.

part of Source code: StorageService.java see initServer()

How does the internode communication work ?

The Gossip protocol

- executed every second in every node
- used for peer discovery and (meta)data propagation
- A node gossips with up to three other nodes in the cluster; in particular one randomly picked **live** node.
- 3-way handshake



Source code:

Gossiper.java — see class GossipTask, line 243

GossipDigest.java — local knowledge of a node

MessagingService.java — internode communication

The Gossip protocol

crucial for scalability

gossip causes only a linear amount of network traffic

crucial for availability

p2p: gossip is completely decentralized

problem

How does a node know if another node is alive ?

Importance of failure detection

and another impossibility result

To detect the crash of other processes is a fundamental issue in distributed systems.

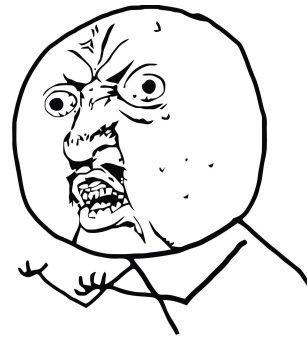
[. . .] several distributed agreement problems, such as Consensus, cannot be solved deterministically in asynchronous systems if even a single process might crash [. . .] in such a system a crashed process cannot be distinguished from a very slow one.

N. Hayashibara, X. Défago, R. Yared, T. Katayama '04

For more details see

Impossibility of distributed consensus with one faulty process
M.J. Fischer, N.A. Lynch, M.S. Paterson
JACM, 1985

https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf



On distributed computing

Impossibility of distributed consensus with one faulty process
M.J. Fischer, N.A. Lynch, M.S. Paterson
JACM, 1985 - **Dijkstra Prize 2001**

https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf

Unreliable Failure Detectors for Reliable Distributed Systems
T.D. Chandra, S. Toueg
PODS 1996 - **Dijkstra Prize 2010**

https:

//ecommons.cornell.edu/server/api/core/bitstreams/c483a842-954a-4c92-a657-04b7a6964215/content

Failure detection in Cassandra

In Cassandra every node has a *status*, either UP or DOWN.

Failure detection is the mechanism by which a node locally determines the status of every other node.

The failure detection in Cassandra is a modified version of the technique presented in

The φ Accrual Failure Detector,
N. Hayashibara, X. Défago, R. Yared, T. Katayama
Tech. Report 2004

https://dspace.jaist.ac.jp/dspace/bitstream/10119/4784/1/IS-RR-2004-010.pdf

Failure detection in Cassandra

Rough idea

Pick a node A, and suppose it monitors a live node B.[5]

1. A records the time intervals (aka heartbeat-interval Δ_i) at which it receives gossip messages from B.
2. A computes the distribution of the Δ_i and a *suspicion level* φ

   The Exponential distribution seems a better approximation than the Normal distribution.

   Math implementation is subtle, see CASSANDRA-2597

   The value of φ represents the likelihood of A being wrong about the state of B.
3. The higher the φ, the higher the confidence that B has failed.
4. If φ is higher than the `phi_convict_threshold` then node B is marked DOWN. See line 359 in FailureDetector.java

- FailureDetector.java, EndpointState.java
- Official documentation check also Dynamo

[5] A node is live at initialisation. EndpointState.java#79, `isAlive = true;`

How is data distributed across nodes ?

Data partitioning

- The act of distributing data across nodes.
- A partition is a division of a logical database or its constituent elements into distinct independent parts. see wiki

Cassandra achieves horizontal scalability by partitioning all data stored in the system using a hash function.
Each partition is replicated to multiple physical nodes, often across failure domains such as racks and even data centers.

Data partitioning

A partitioner is meant to distribute data in an even way across nodes, to avoid bottlenecks.

Even way ?

Intuitive example: if a cluster has four nodes,
every node should contain **???**% of the data.

In practice we may want more powerful nodes to manage more data than less powerful ones.

Furthermore, bringing nodes on-line and off-line should not create complications.

Data partitioning

A partitioner is meant to distribute data in an even way across nodes, to avoid bottlenecks.

Even way ?

Intuitive example: if a cluster has four nodes,
every node should contain 25% of the data.

In practice we may want more powerful nodes to manage more data than less powerful ones.

Furthermore, bringing nodes on-line and off-line should not create complications.

Cassandra partitioners

Murmur3Partitioner: uniform distribution based on Murmur3 hash (default)

RandomPartitioner: uniform distribution based on MD5 hash

ByteOrderedPartitioner: lexical distribution based on key bytes (legacy only)

Why default is not MD5 ?

Cassandra does not need a cryptographic hash no security concerns
Using the Murmur3Partitioner results in a 3-5 times improvement in performance.

Source code:

- Murmur3Partitioner.java,
- RandomPartitioner.java

Token

A token is the result of applying a hash function.

Partitioner	token type	min. value	max. value
Murmur3Partitioner	`long`	-2^{63}	$+2^{63}-1$
RandomPartitioner	`BigInteger`	0	$2^{127}-1$

Recall that when joining a cluster (during bootstrap)
a node is assigned a token range it is responsible for.

When setting-up a Cassandra node

- Using vnodes: you do not need to calculate the tokens.
- Not using vnodes: you must calculate the tokens to assign to the `initial_token` parameter.

Partitioning

What is hashed ?

- Every row has a Row Key

  `Map<RowKey, SortedMap<ColumnKey, ColumnValue>>`

- A Row Key can be a tuple — aka Compound key

  ```
  CREATE TABLE my_table (user uid, item iid, time
    timestamp, PRIMARY KEY (user, item, time))
  ```

- The first element of a Row Key is the Partition Key

  ```
  CREATE TABLE my_table (user uid, item iid, time
    timestamp, PRIMARY KEY (user, item, time))
  ```

Partitioning

Rough idea

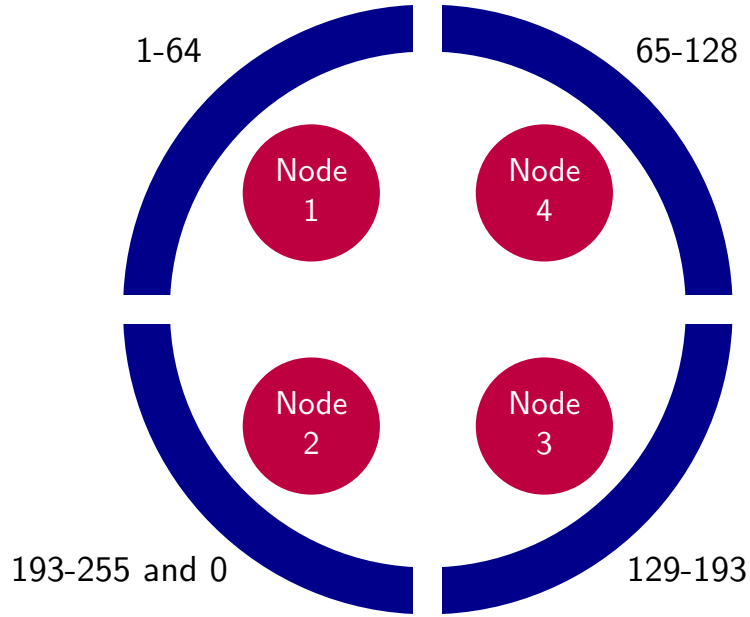
Given a Row Key $(pk, c_1, c_2, \ldots)$, the partitioner hashes the Partition Key pk to compute a token t,

$$t = hash(pk)$$

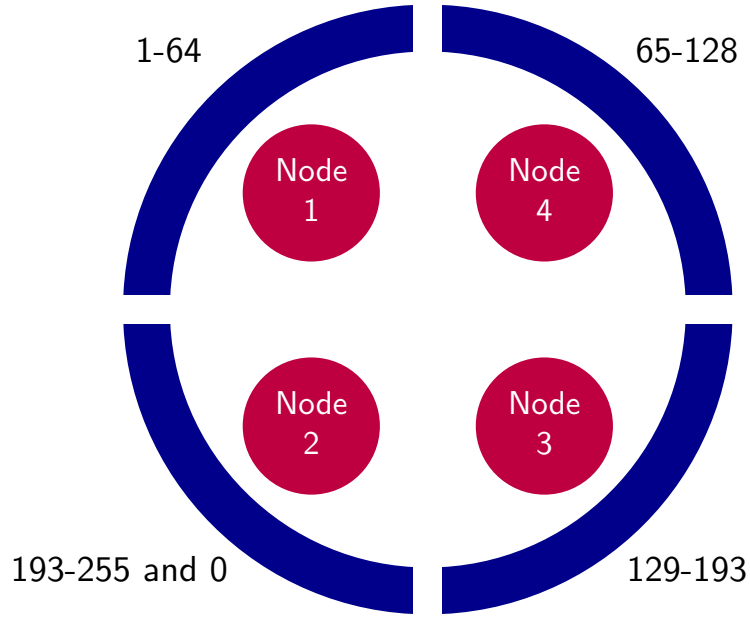
The row will be stored in the node responsible for the range of tokens than contains t, and replicated in a clockwise fashion to its sibling nodes depending on the Replication Factor.

Further details:

- Official documentation
- Figure 2 in Dynamo paper
- Chord algorithm, I. Stoica et al., ACM SIGCOMM '01
- TokenMetadata.java, EndpointsForToken.java, IPArtitioner.java#82

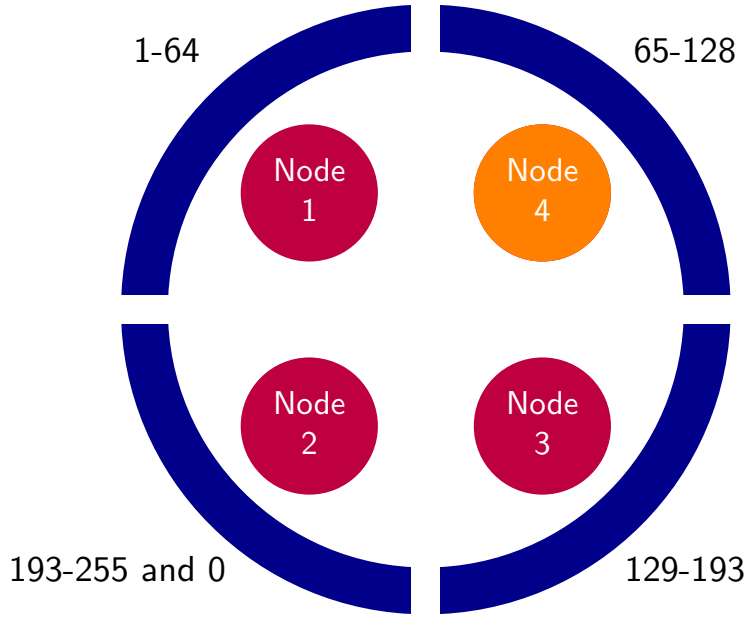


A client asks Cassandra to write the row (69000, Lyon)
in a keyspace with replication factor 2.
The server



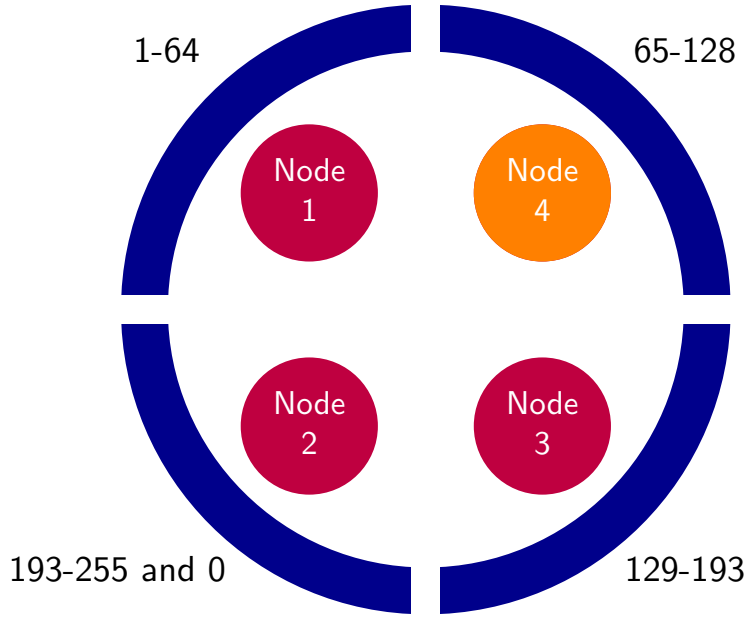
A client asks Cassandra to write the row (69000, Lyon) in a keyspace with replication factor 2.
The server

1. computes the token, $hash(69000) = 67$



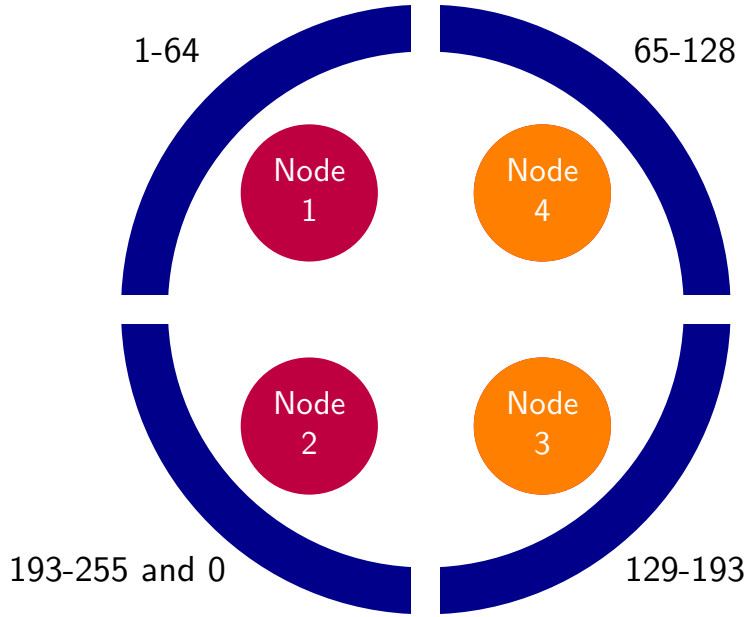
A client asks Cassandra to write the row (69000, Lyon) in a keyspace with replication factor 2.
The server

1. computes the token, $hash(69000) = 67$
2. finds the node responsible for token 67, Node 4



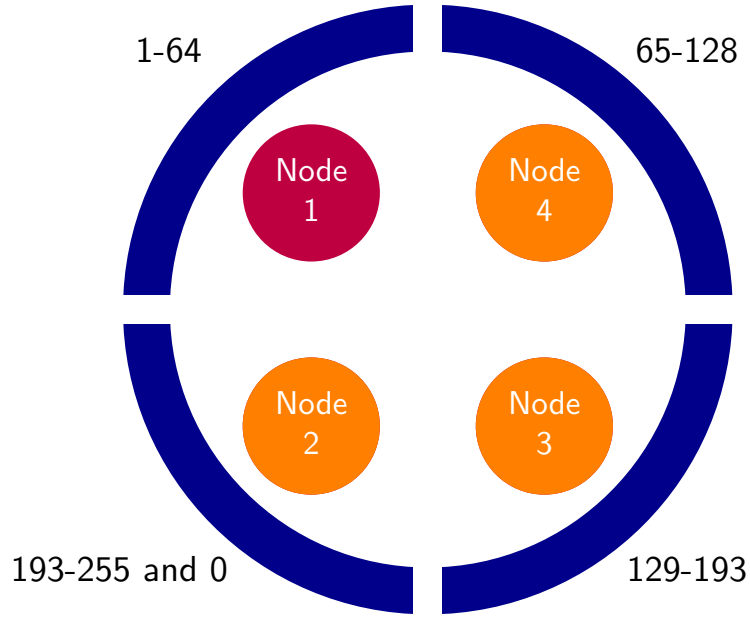
A client asks Cassandra to write the row (69000, Lyon) in a keyspace with replication factor 2.
The server

1. computes the token, $hash(69000) = 67$
2. finds the node responsible for token 67, Node 4
3. writes the (69000, Lyon) in Node 4



A client asks Cassandra to write the row (69000, Lyon) in a keyspace with replication factor 2.
The server

1. computes the token, $hash(69000) = 67$
2. finds the node responsible for token 67, Node 4
3. writes the (69000, Lyon) in Node 4
4. and in the next node in clockwise order, Node 3



A client asks Cassandra to write the row (69000, Lyon)
in a keyspace with replication factor 3.
The server

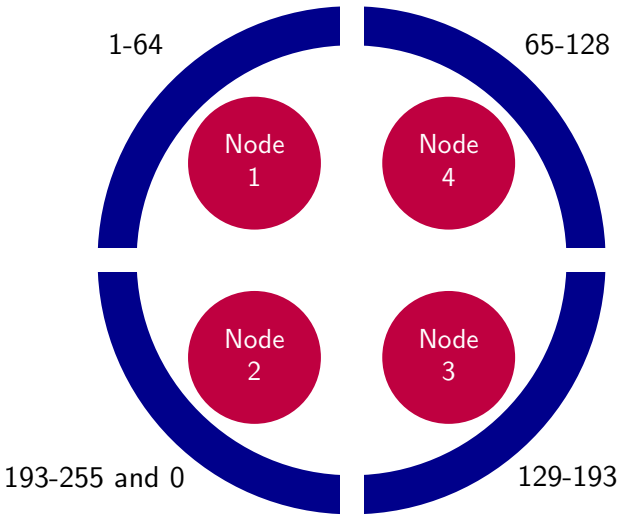
1. computes the token, $hash(69000) = 67$
2. finds the node responsible for token 67, Node 4
3. writes the (69000, Lyon) in Node 4
4. and in the next 2 nodes in clockwise order, Node 3 and Node 2

Read operations are implemented by the methods:

`read`	StorageProxy.java#L2149
`readRegular`	StorageProxy.java#L2529
`fetchRows`	StorageProxy.java#L2625
`legacyReadWithPaxos`	StorageProxy.java#L2323
`read`	Paxos.java#L888

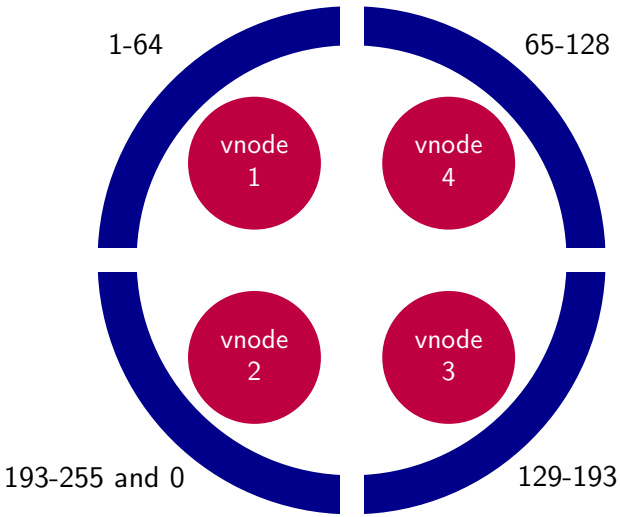
Virtual nodes

logical token ring



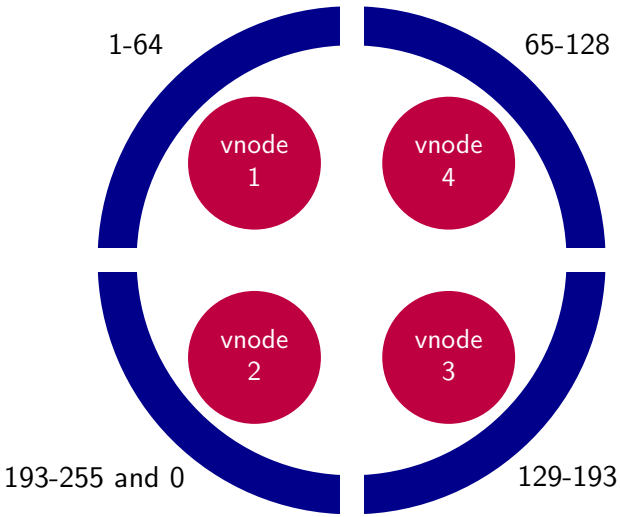
Virtual nodes

logical token ring

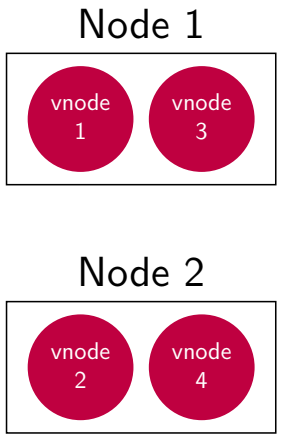


Virtual nodes

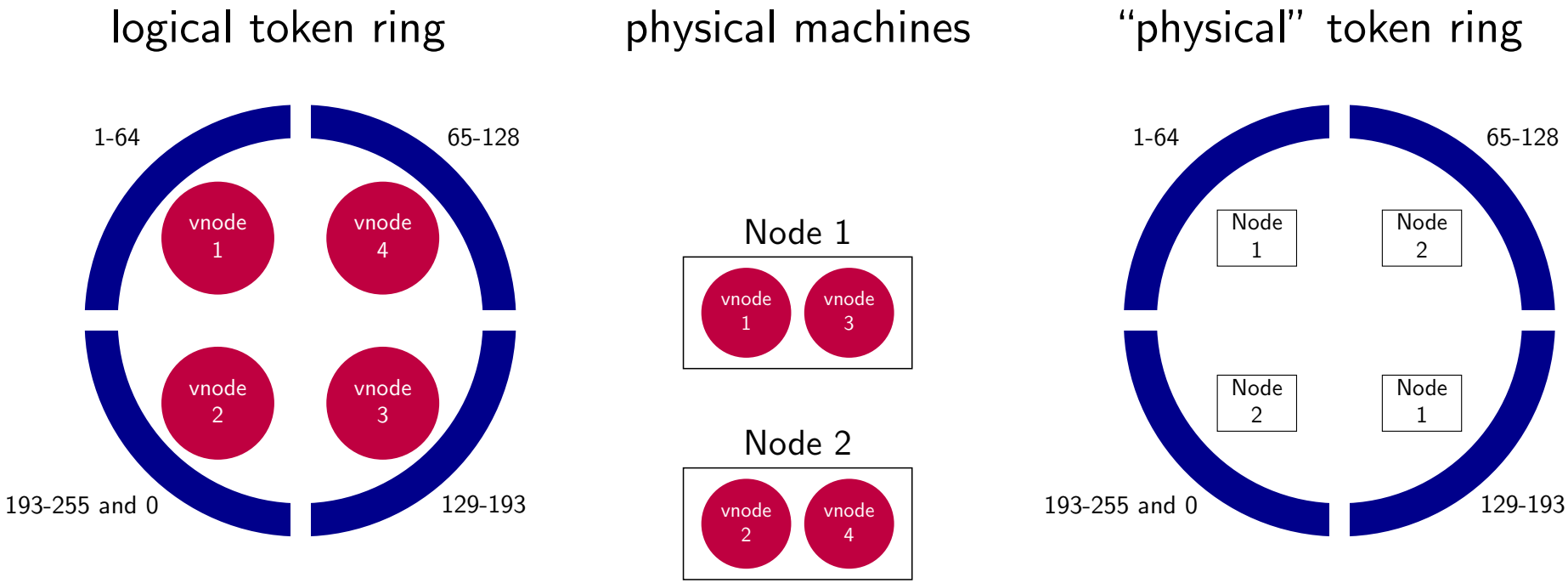
logical token ring



physical machines



Virtual nodes



Decoupling of token ranges and nodes

- automatic token ranges generation / assignment
- simplifies replacing, adding, removing a node thanks to replicas
- even distribution of data:
  different nodes can have different amounts of vnodes

Virtual nodes

- Partitions are replicated across multiple vnodes
- Each copy of a partition is called a replica

How many vnodes per physical machine ?

Datastax reports that 8 vnodes distributes the workload between systems with a ˜10% variance and has minimal impact on performance.

Within a datacenter

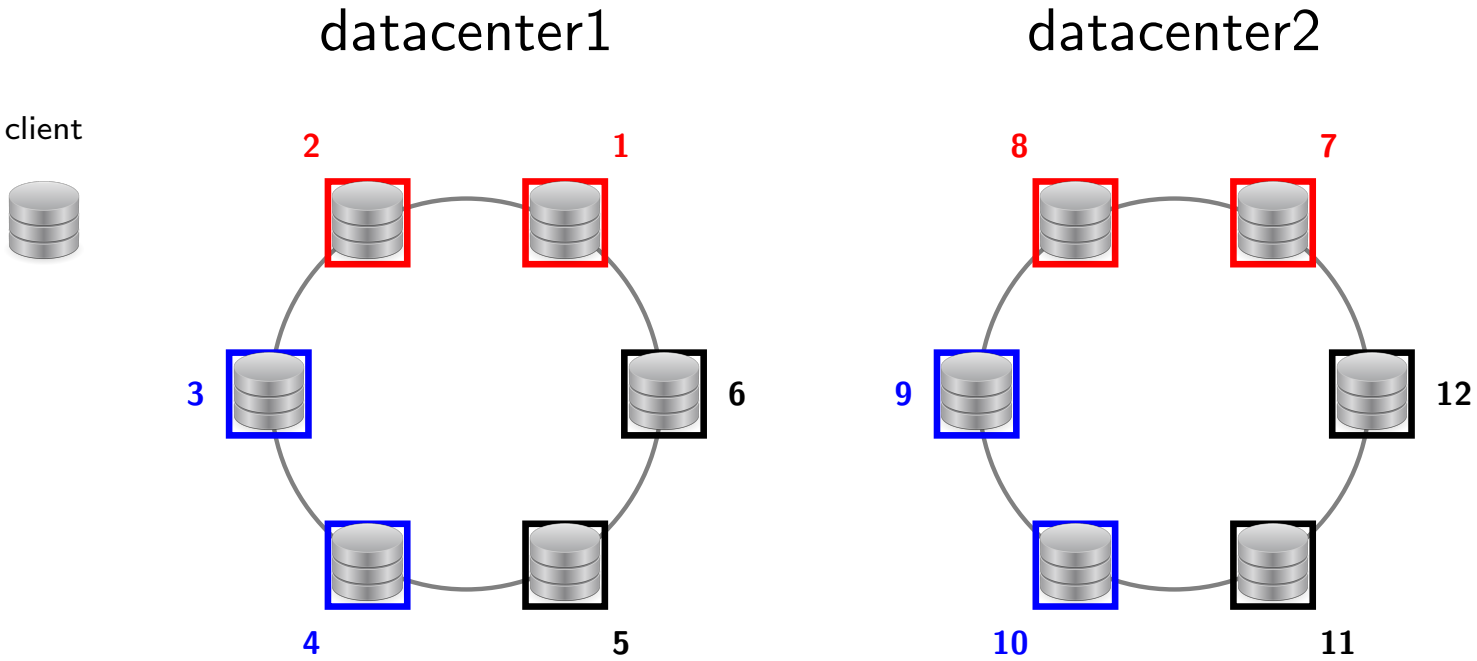
- either all the nodes are vnode-enabled
- or none is

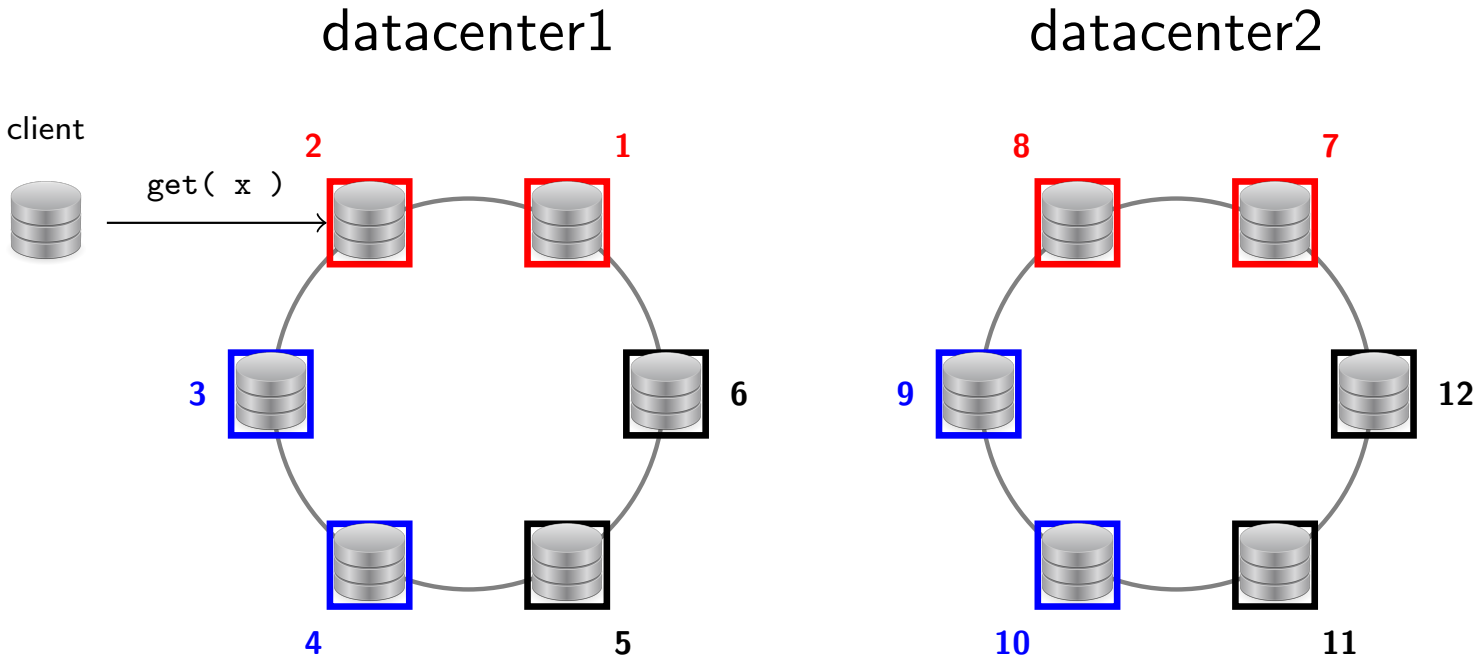
Data centers architecture, though, can vary.

Further details

- Datastax discussion on vnodes and replication
- Motivation for vnodes: V is for vnodes

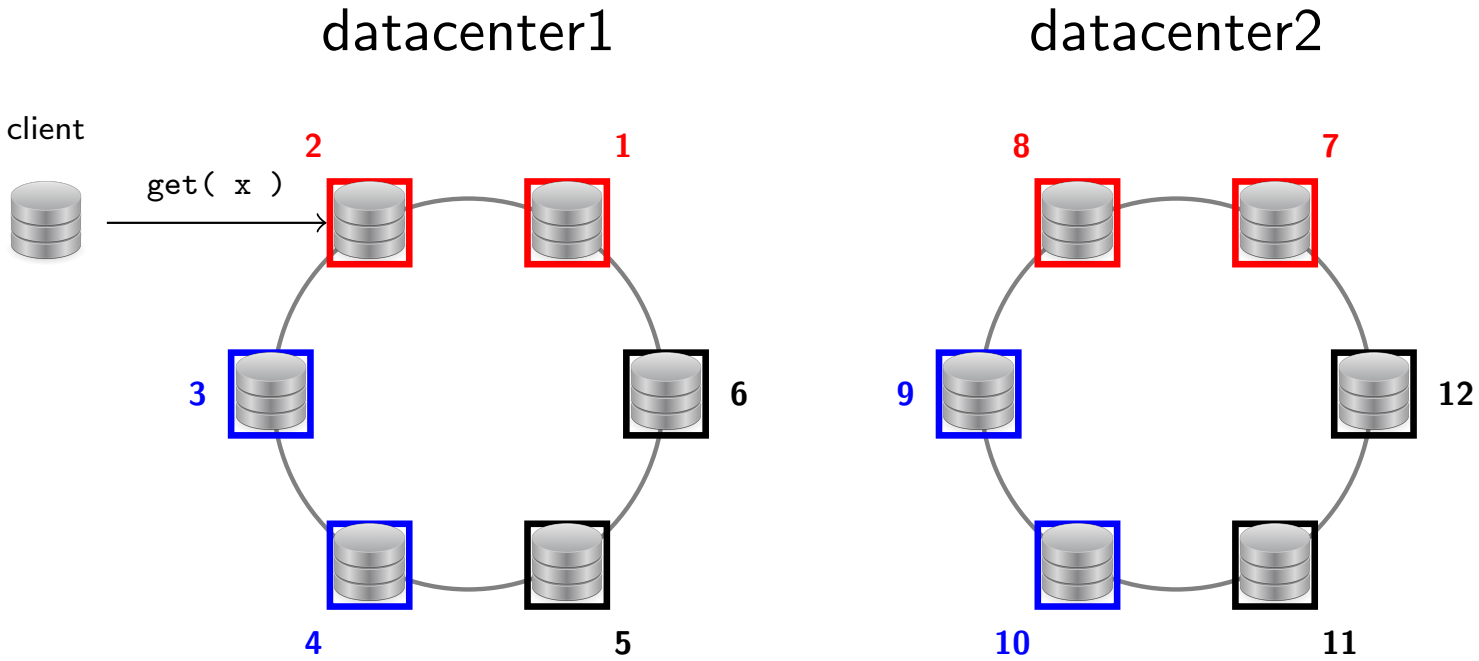
How does a client read data ?


datacenter1
datacenter2
client
2
1
8
7
3
6
9
12
4
5
10
11



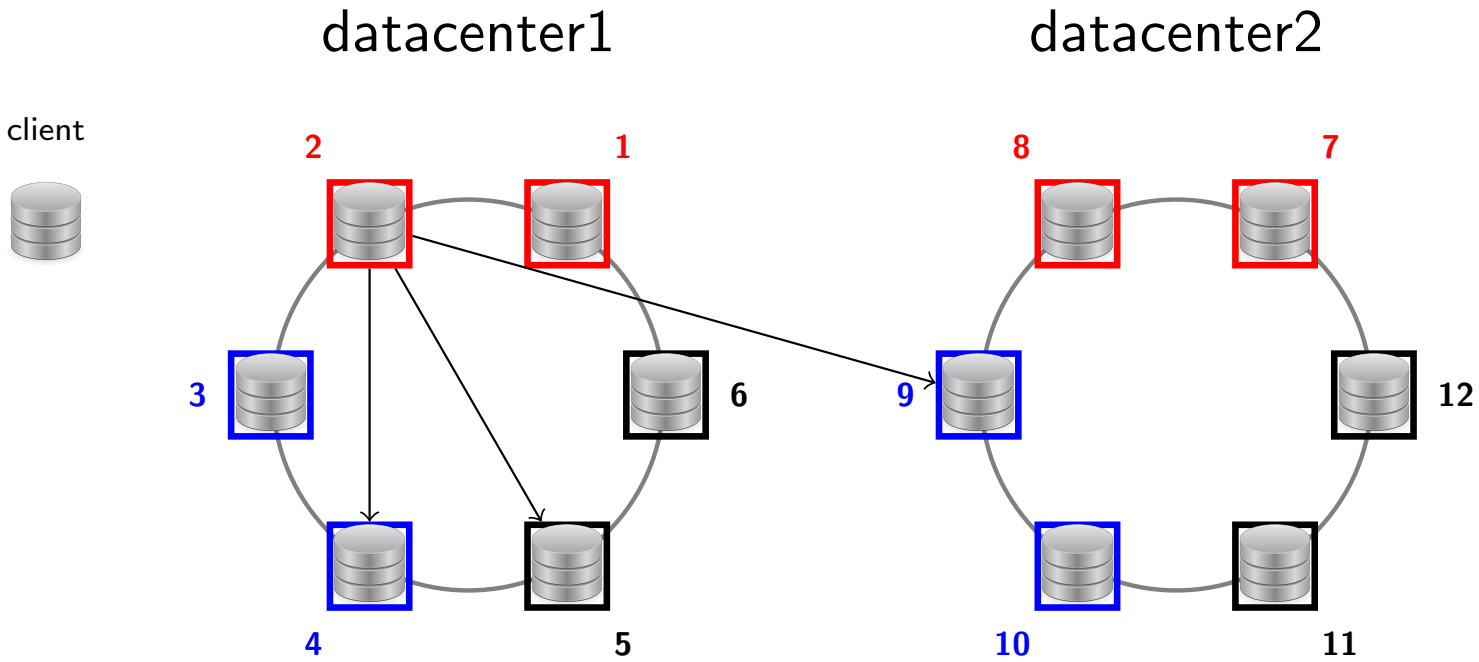
Node 2 coordinator for the request

1. computes the node / partition that contains `x`



Node 2 coordinator for the request

1. computes the node / partition that contains `x`
2. computes which replicas contains `x`

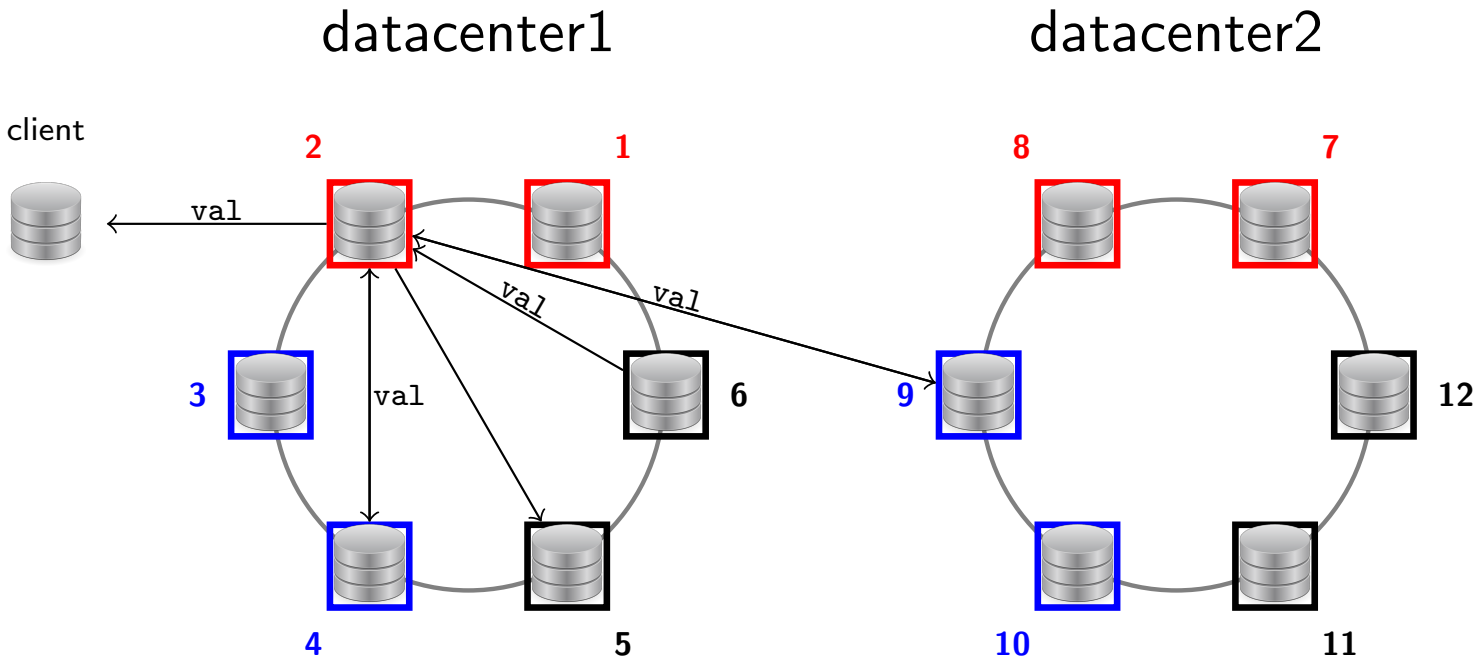


Node 2 coordinator for the request

1. computes the node / partition that contains `x`
2. computes which replicas contains `x`
3. requests in parallel `x` to

enough nodes to meet the desired consistency level

```
we'll only send read requests to enough replicas to satisfy
the consistency level (see StorageProxy.java#L2084)
```



Node 2 coordinator for the request

1. computes the node / partition that contains `x`
2. computes which replicas contains `x`
3. requests in parallel `x` to

   enough nodes to meet the desired consistency level

   `we'll only send read requests to enough replicas to satisfy the consistency level (see StorageProxy.java#L2084)`
4. waits for the answers and answer the client

Read consistency levels

Levels	Coordinator waits for a response from
ALL	all replicas; fails if a replica does not respond
QUORUM	a quorum of replicas from all data centers.
LOCAL_QUORUM	a quorum of replicas in the coordinator data center.
ONE	Immediately return the record held by the first node that respond
TWO	Immediately return the record held by the first nodes that respond
THREE	Immediately return the record held by the first nodes that respond
LOCAL_ONE	the closest replica in the coordinator datacenter.
SERIAL	...
LOCAL-SERIAL	...

A complete description of the levels at

`https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlConfigConsistency.html`

These classes contain the details of the `read` requests to satify the required consistency level:

AbstractReadExecutor.java
ReplicaPlan.java

The Snitch[6]

An endpoint snitch gathers information about the network topology, to determine which nodes to read / write from.

- `SimpleSnitch`
  Not rack aware. Unsuitable for multiple datacenters deployement.
- `PropertyFileSnitch`
  Relies on (detailed) information in `cassandra-topology.properties`.
- `GossipingPropertyFileSnitch`
  Nodes exchange information about the datacenter and the rack via the gossip protocol. The info is defined in the files `cassandra-rackdc.properties` and `cassandra-topology.properties`.
- `Ec2Snitch`, `Ec2MultiRegionSnitch`, `GoogleCloudSnitch`, `AlibabaSnitch`, `...`

[6] Snitch = mouchard.

The Snitch

- The snitches are in `org.apache.cassandra.locator`,
- they implement the interface `IEndpointSnitch.java`,
- by extending the class `AbstractNetworkTopologySnitch.java`.
- The official documentation at `https://cassandra.apache.org/doc/latest/operating/snitch.html`

Let's try a query (according to theory)

Recall the keyspace declaration

```
CREATE KEYSPACE properStorage
WITH replication = {'class':'NetworkTopologyStrategy',
                    'datacenter1' : 3
                    'datacenter2' : 2 };
```

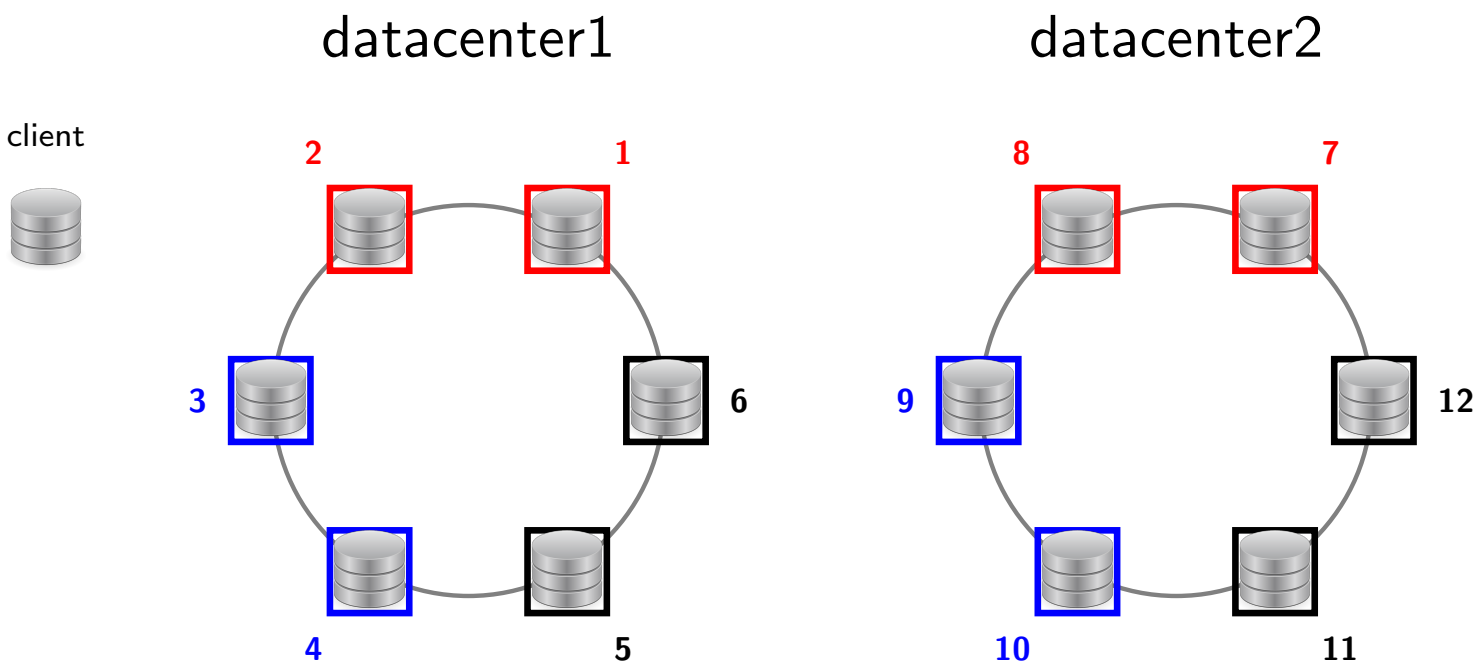
and consider the queries

```
CONSISTENCY QUORUM;
SELECT * FROM cities WHERE pobox = 96000;
```

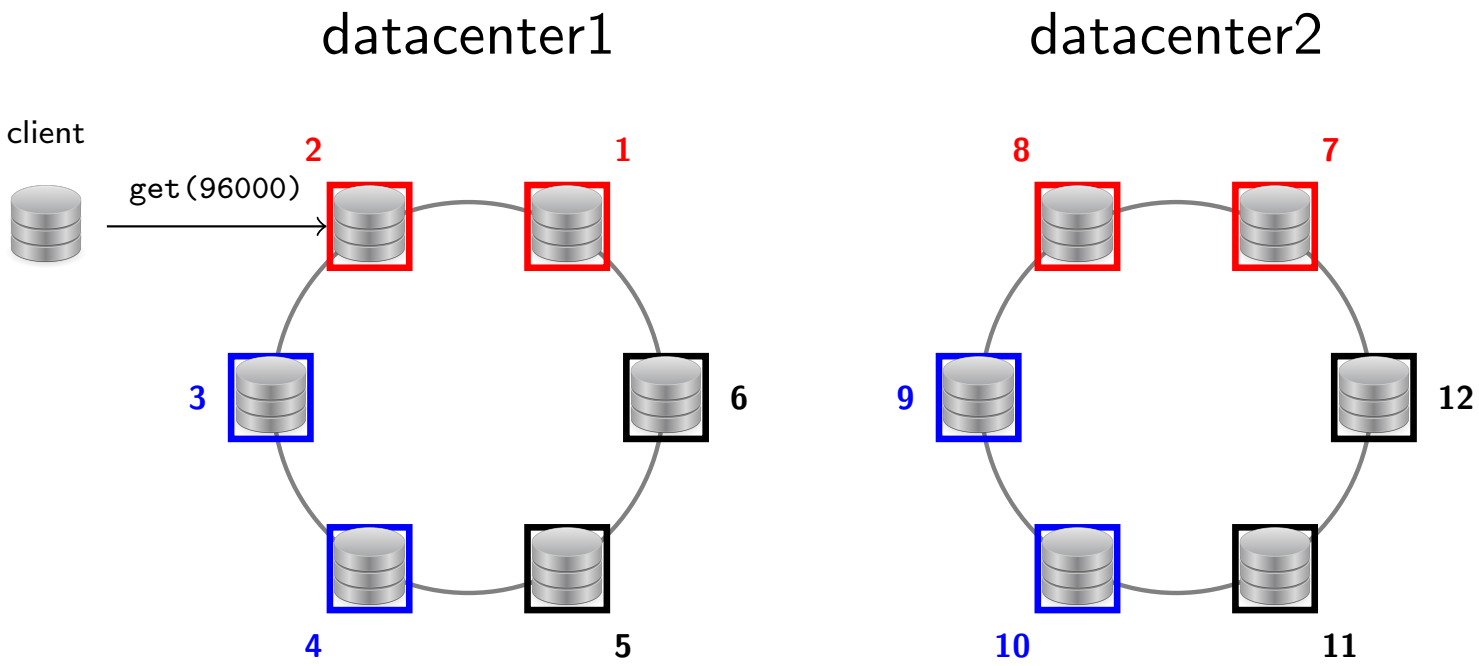
How many replicas must answer for quorum ?

$quorum = floor(\texttt{replication\_factor}/2) + 1 = 3$

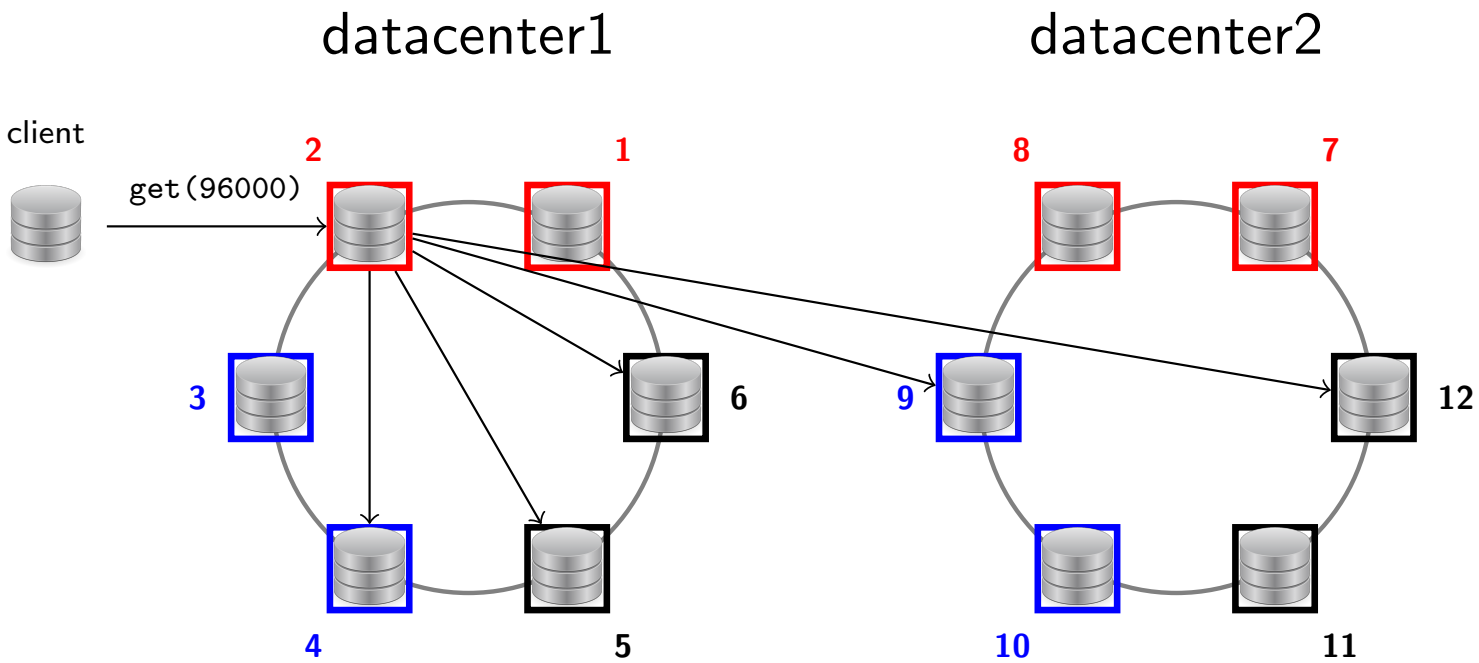
QUORUM



QUORUM

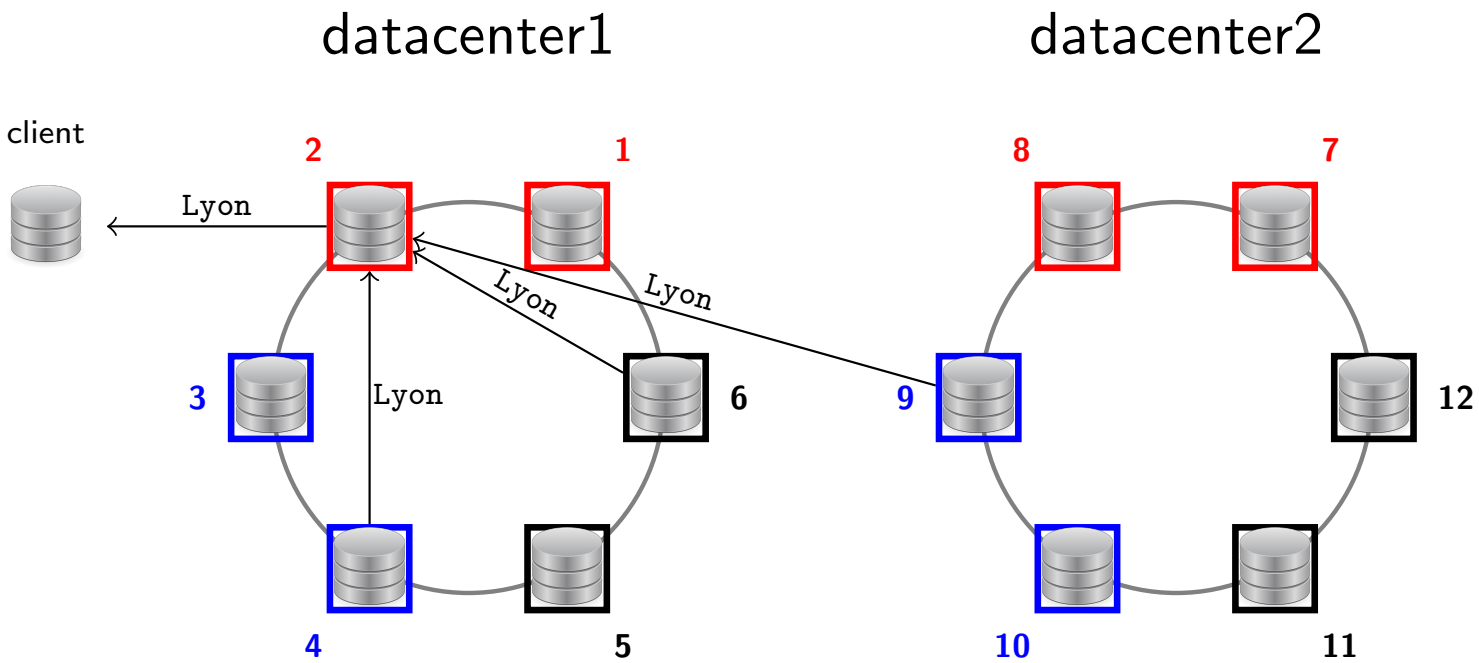

datacenter1
datacenter2
client
get(96000)
1
2
3
4
5
6
7
8
9
10
11
12

QUORUM



Coordinator asks data to 5 replicas

QUORUM

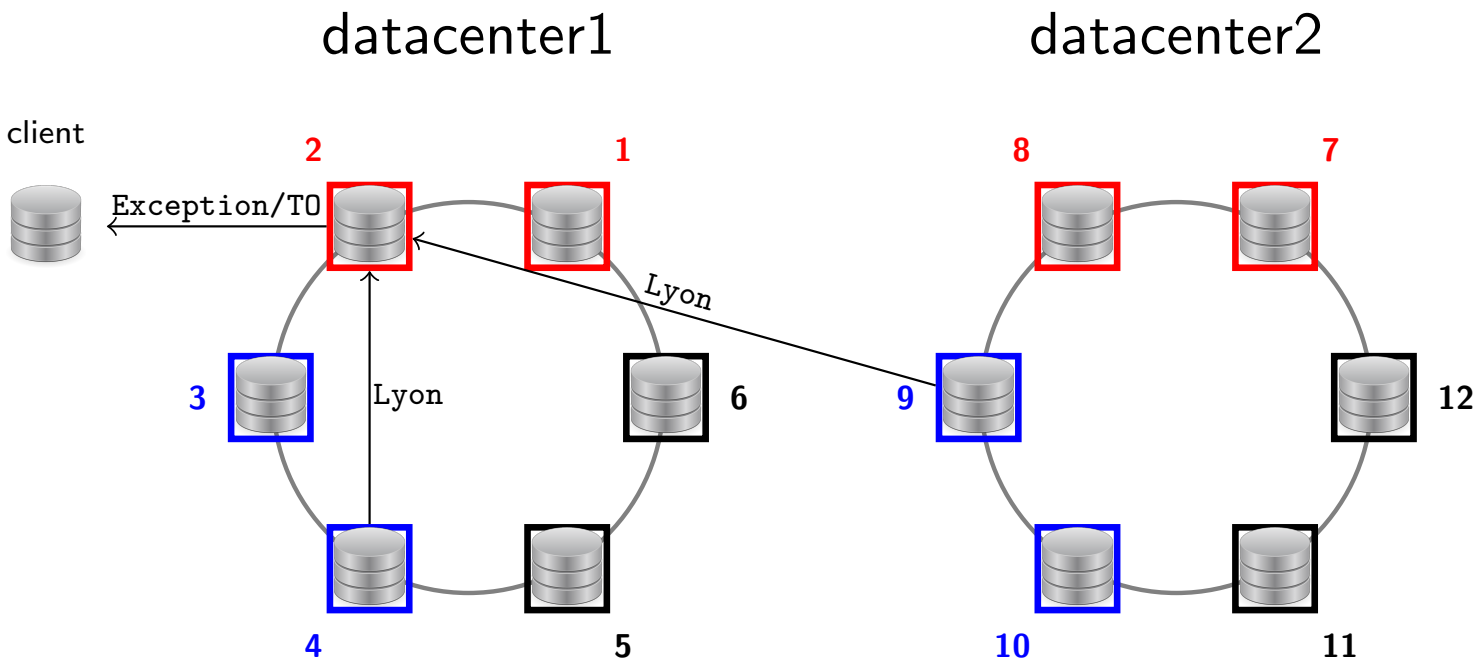


Coordinator asks data to 5 replicas

- 3 answers: return data to client

quorum is 3

QUORUM

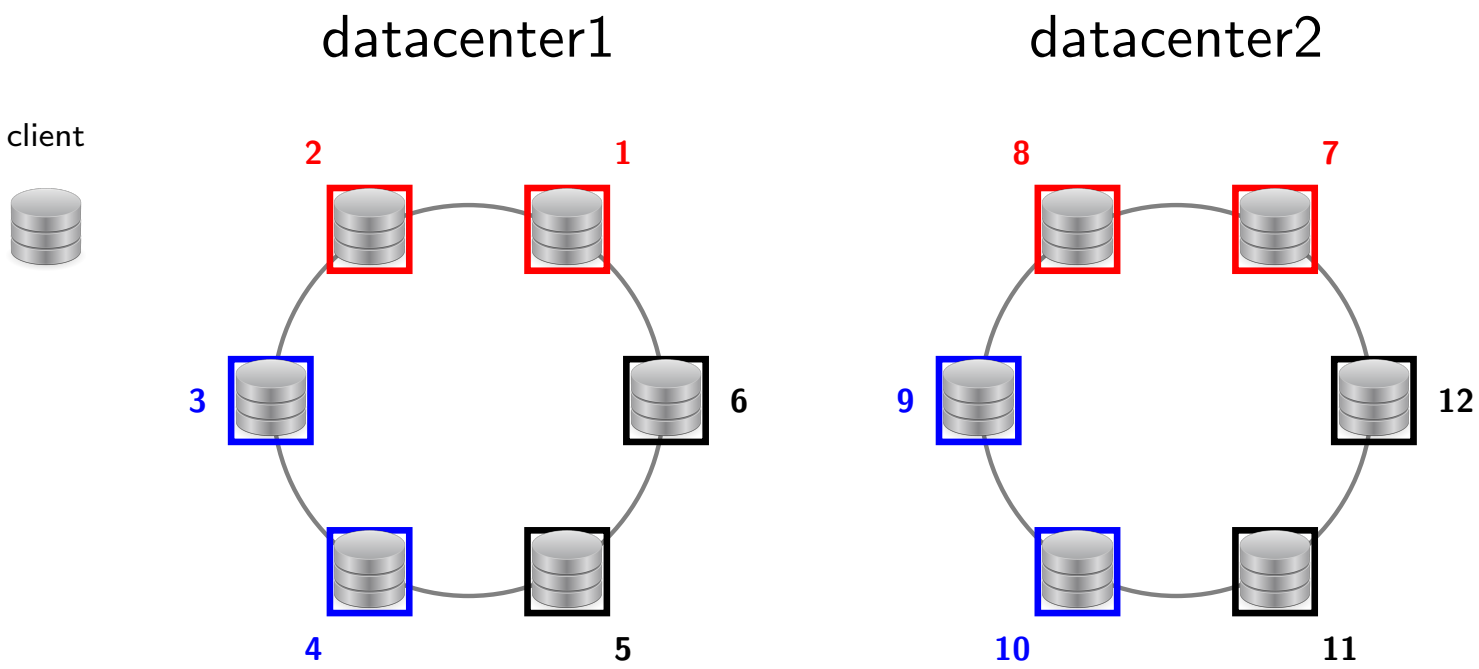


Coordinator asks data to 5 replicas

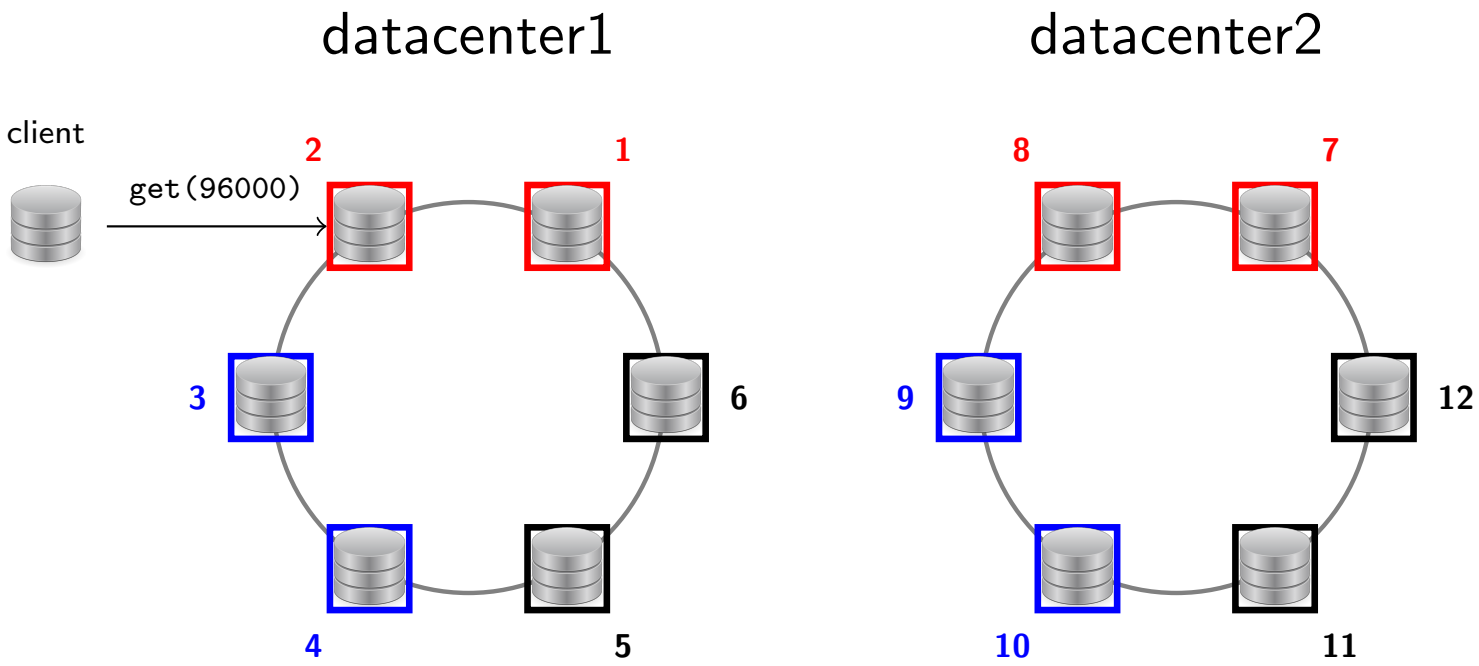
- 3 answers: return data to client
- less than 3 answers: `Exception` or time out

quorum is 3

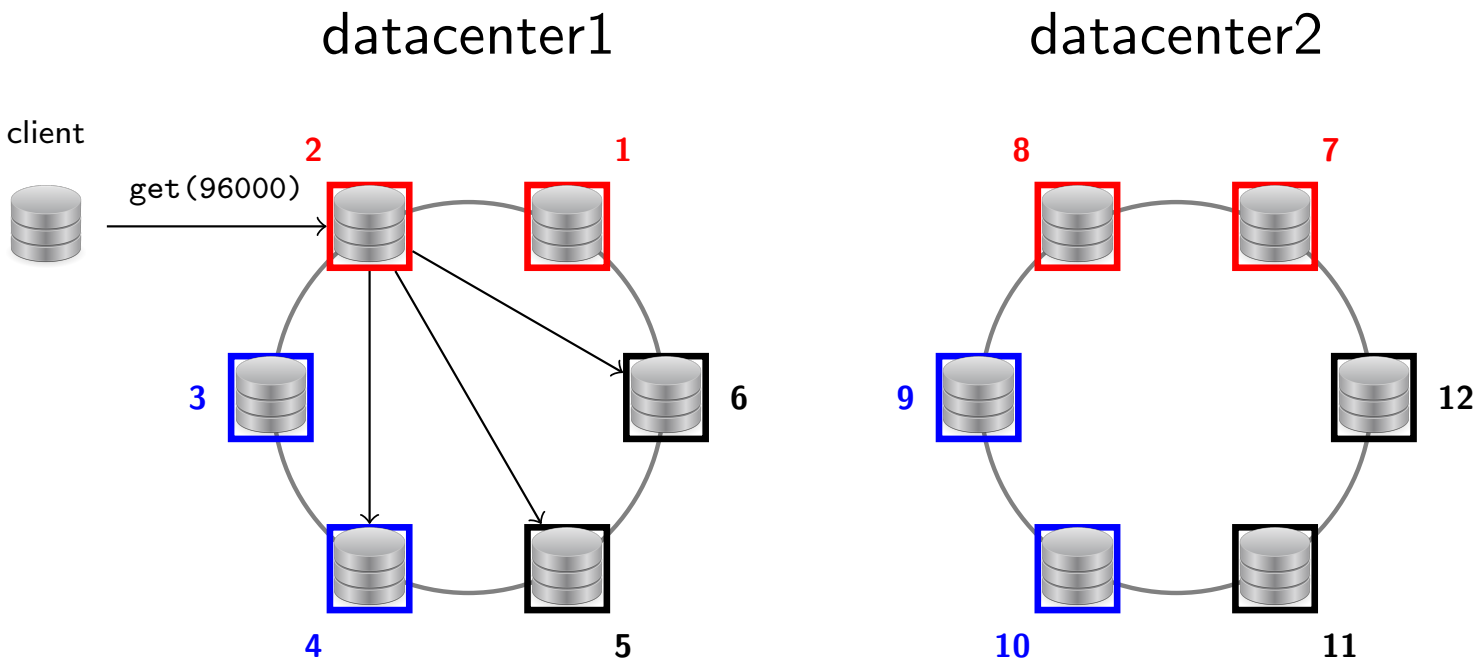
LOCAL_QUORUM



LOCAL_QUORUM

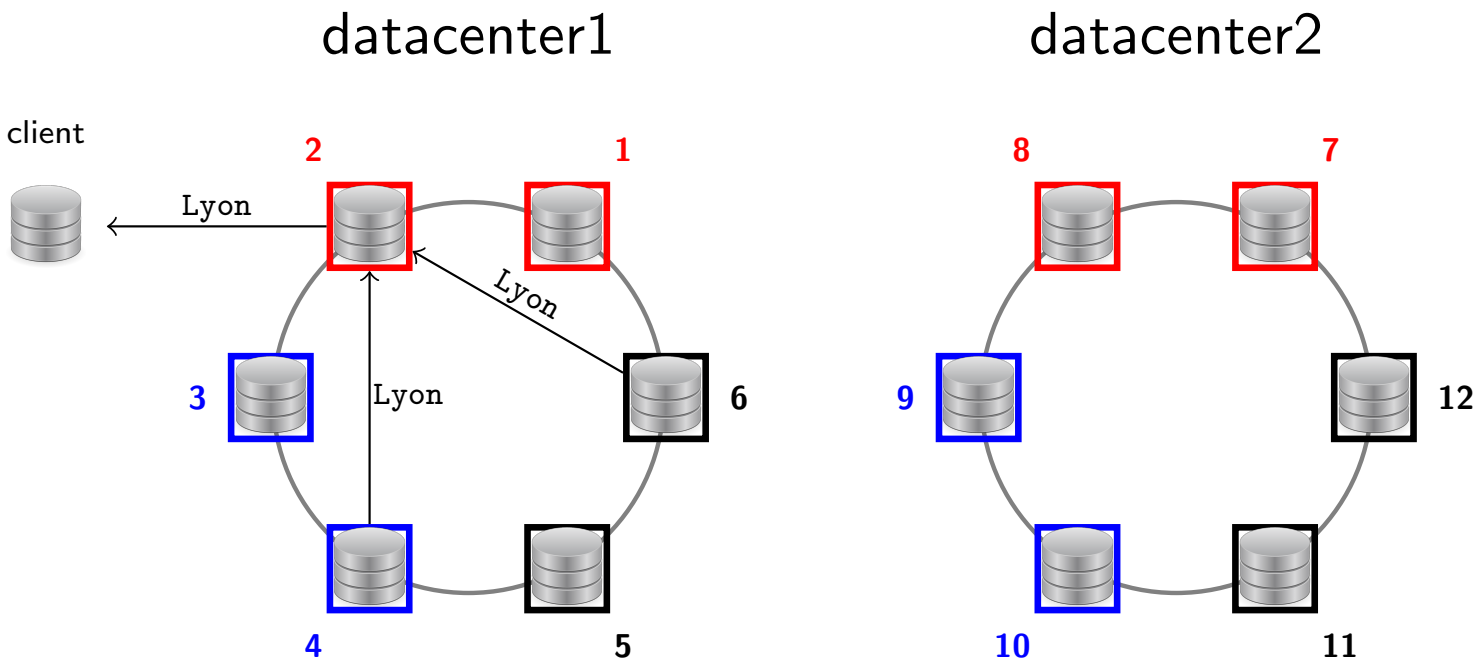


LOCAL_QUORUM



Coordinator asks data to 3 replicas in its datacenter

LOCAL_QUORUM

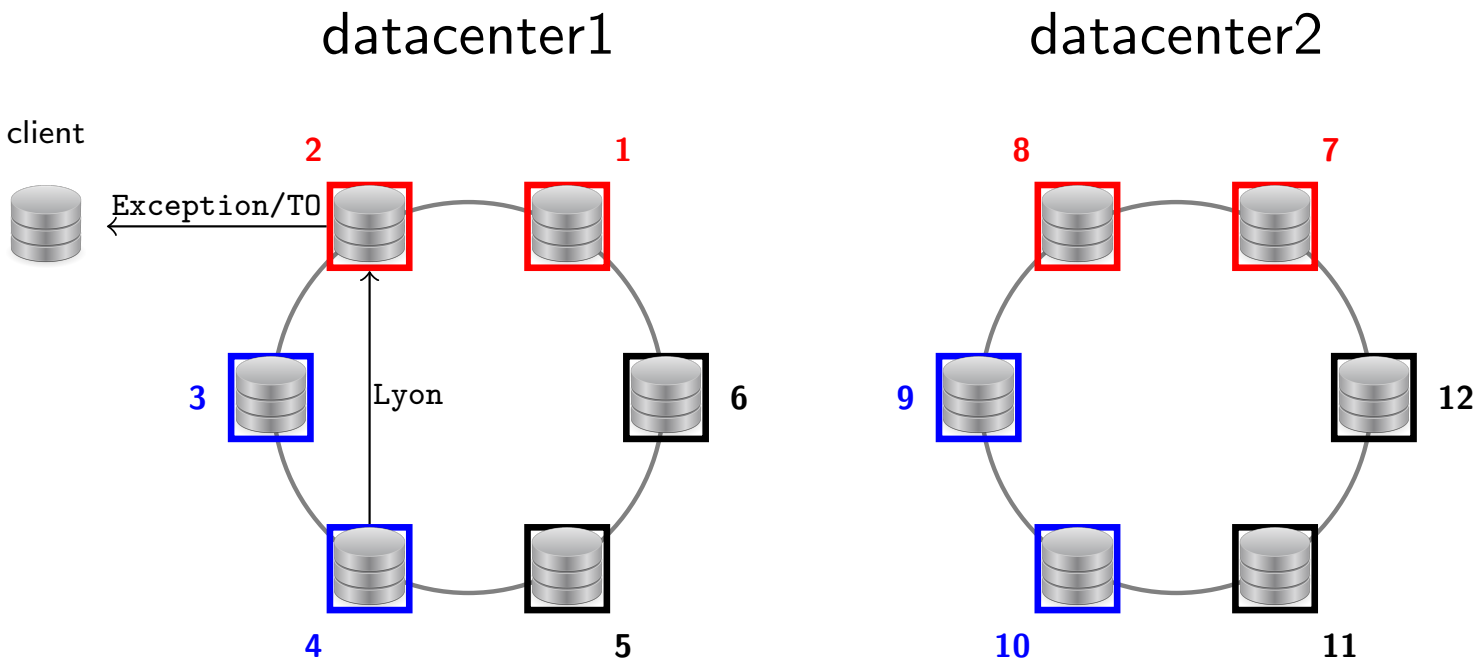


Coordinator asks data to 3 replicas in its datacenter

- 2 answers: return data to client

quorum for dc1 is 2

LOCAL_QUORUM

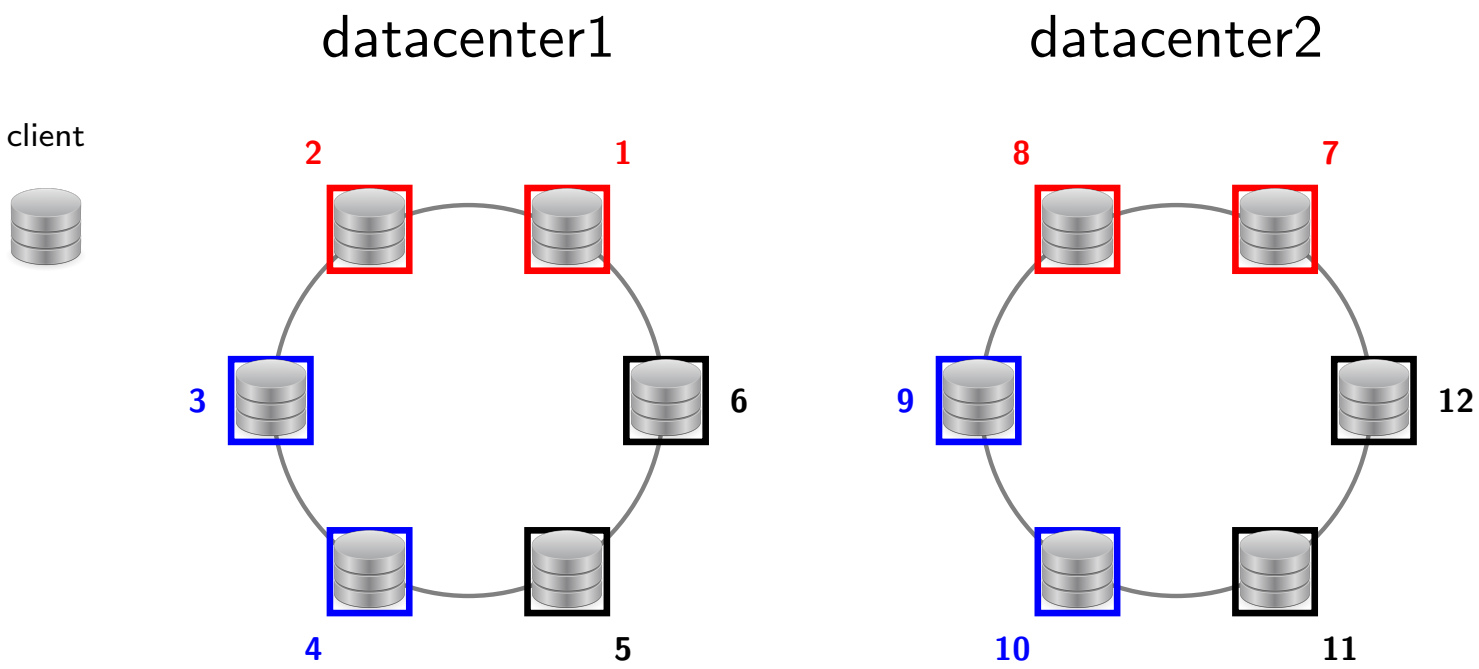


Coordinator asks data to 3 replicas in its datacenter

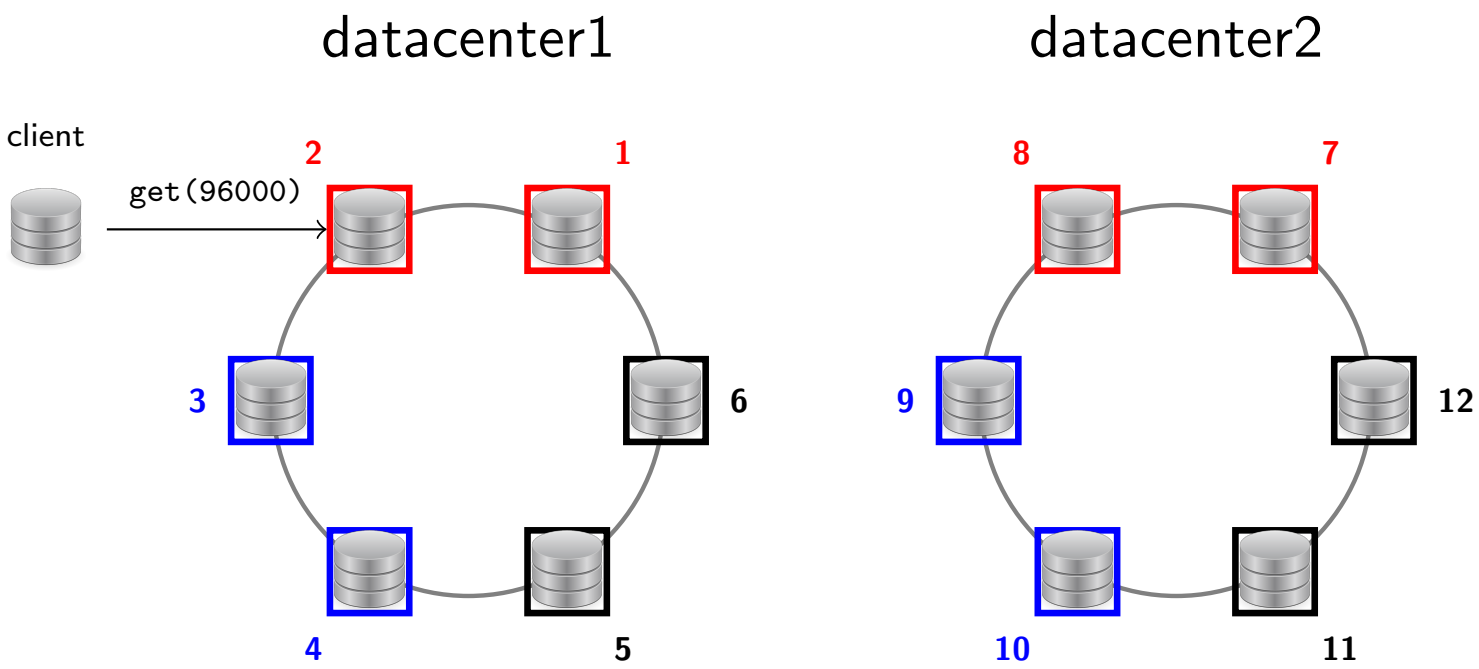
- 2 answers: return data to client
- less than 2 answers: `Exception` or time out

quorum for dc1 is 2

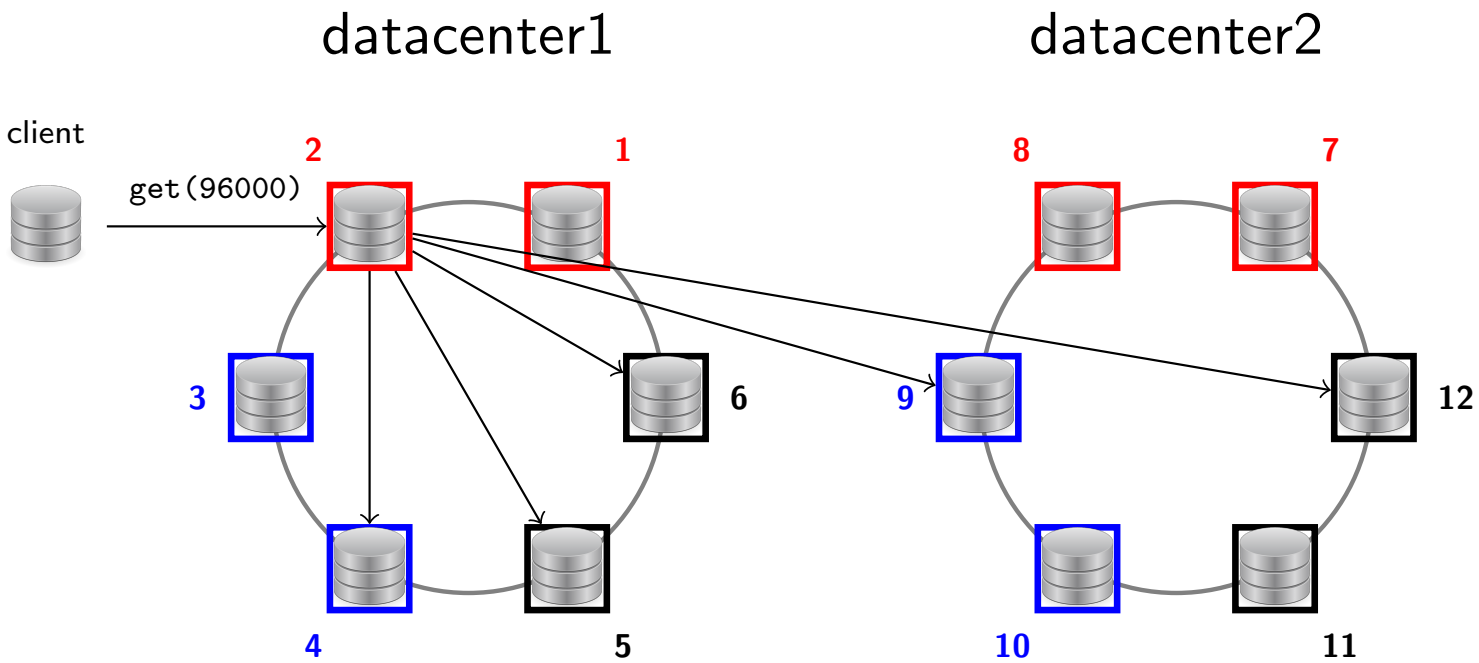
ONE



ONE

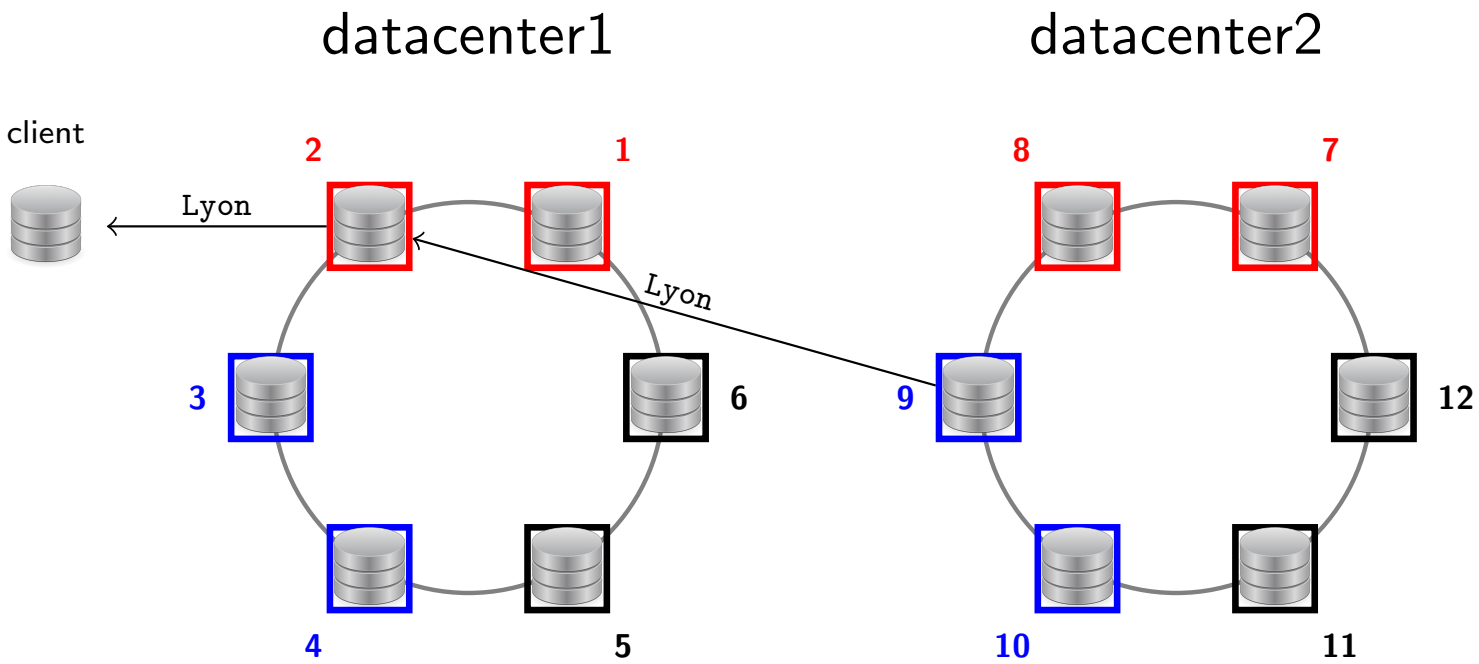

datacenter1
datacenter2
client
get(96000)
1
2
3
4
5
6
7
8
9
10
11
12

ONE



Coordinator asks data to 5 replicas

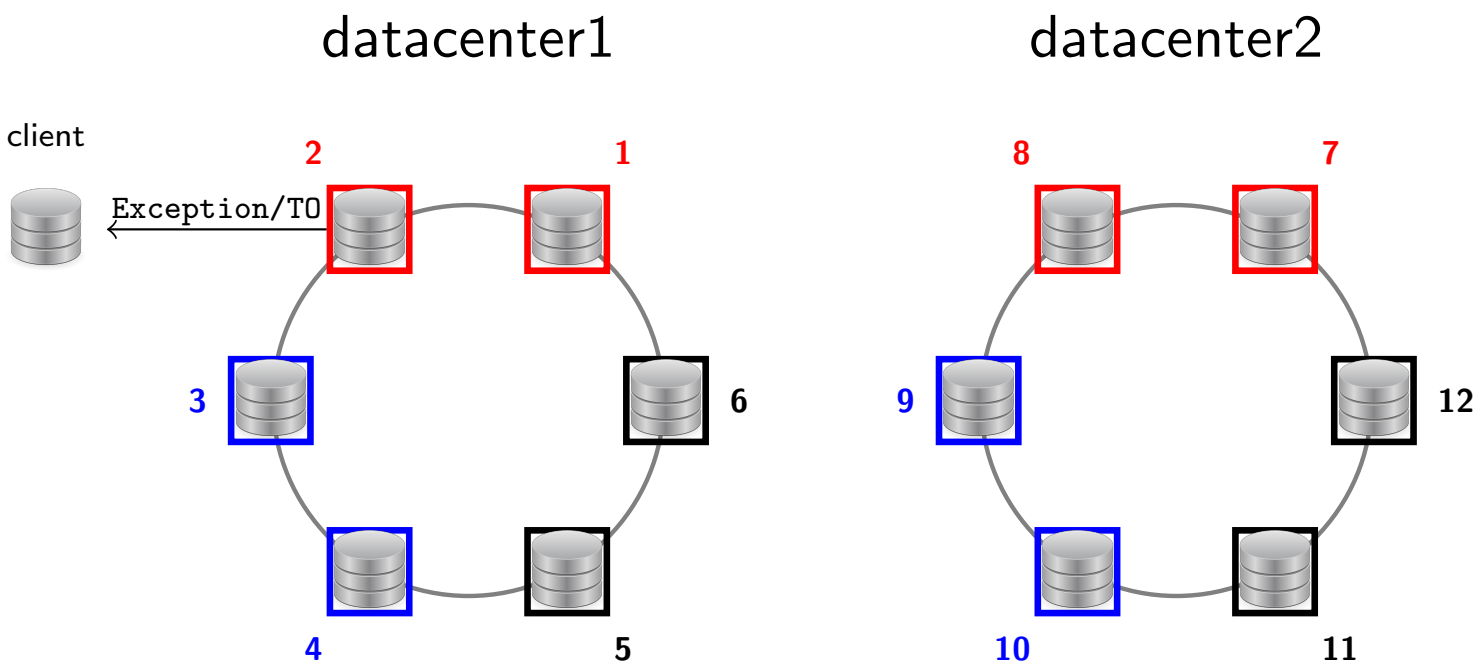
ONE



Coordinator asks data to 5 replicas

- 1 answer: return data to client

ONE



Coordinator asks data to 5 replicas

- 1 answer: return data to client
- no answer: `Exception` or time out

Analogous logic for write operations

LOCAL_SERIAL is replaced by ANY

The write path is

- implemented in the method `performWrite` of the class `StorageProxy.java`
- discussed wrt a real-world application in the Netflix post at this URL,

`https://netflixtechblog.com/`

`benchmarking-cassandra-scalability-on-aws-over-a-million-writes-per-second-39f45f066c9e`

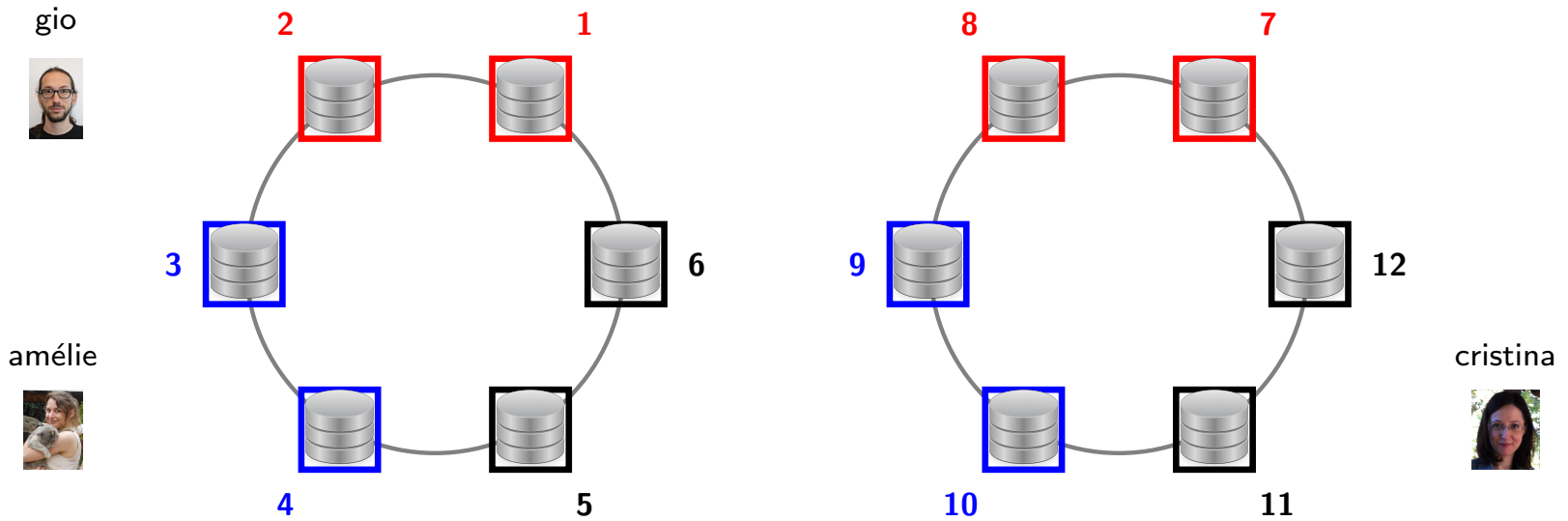
Tuneable consistency

ALL
QUORUM
LOCAL_QUORUM
ONE
TWO
THREE
LOCAL_ONE
SERIAL
LOCAL-SERIAL

Let's take a break and order some

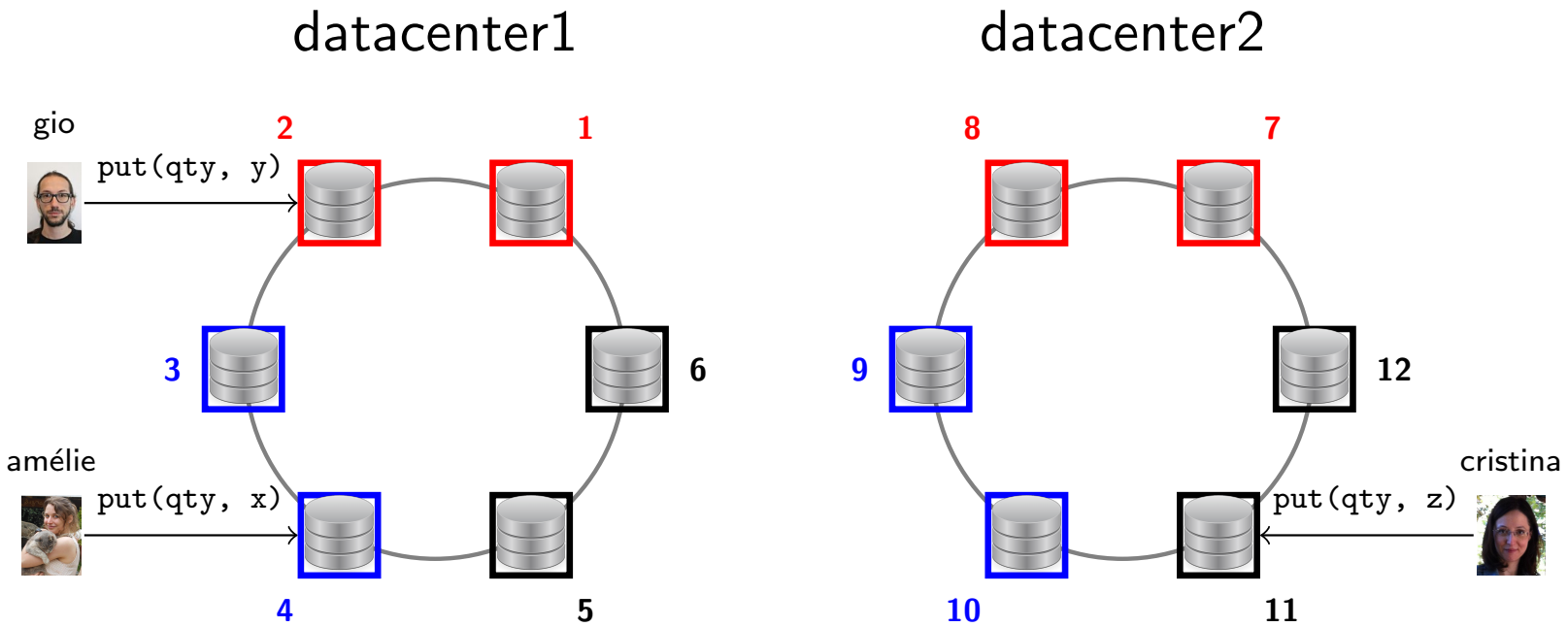
Cassandra eventually consistent

Let's order 



Cassandra eventually consistent

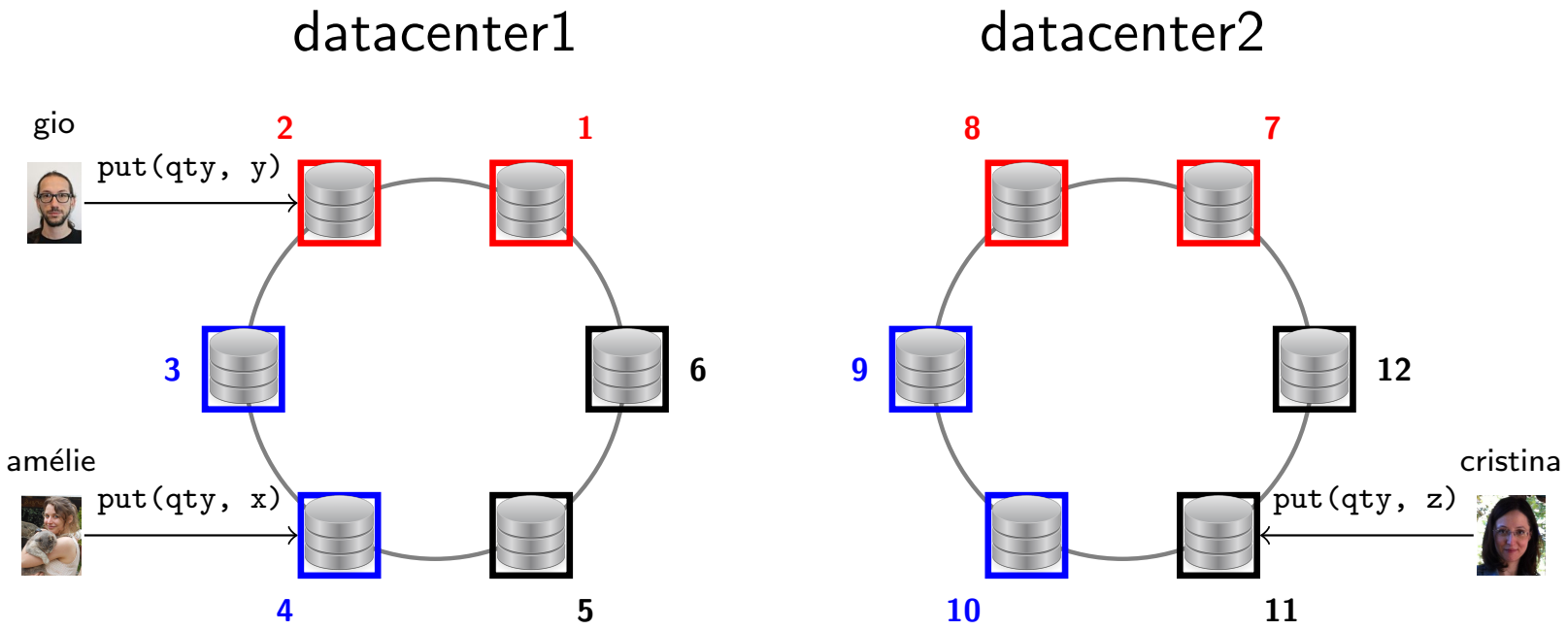
Let's order 



Which is the value of qty ?

Cassandra eventually consistent

Let's order 



Which is the value of qty ?

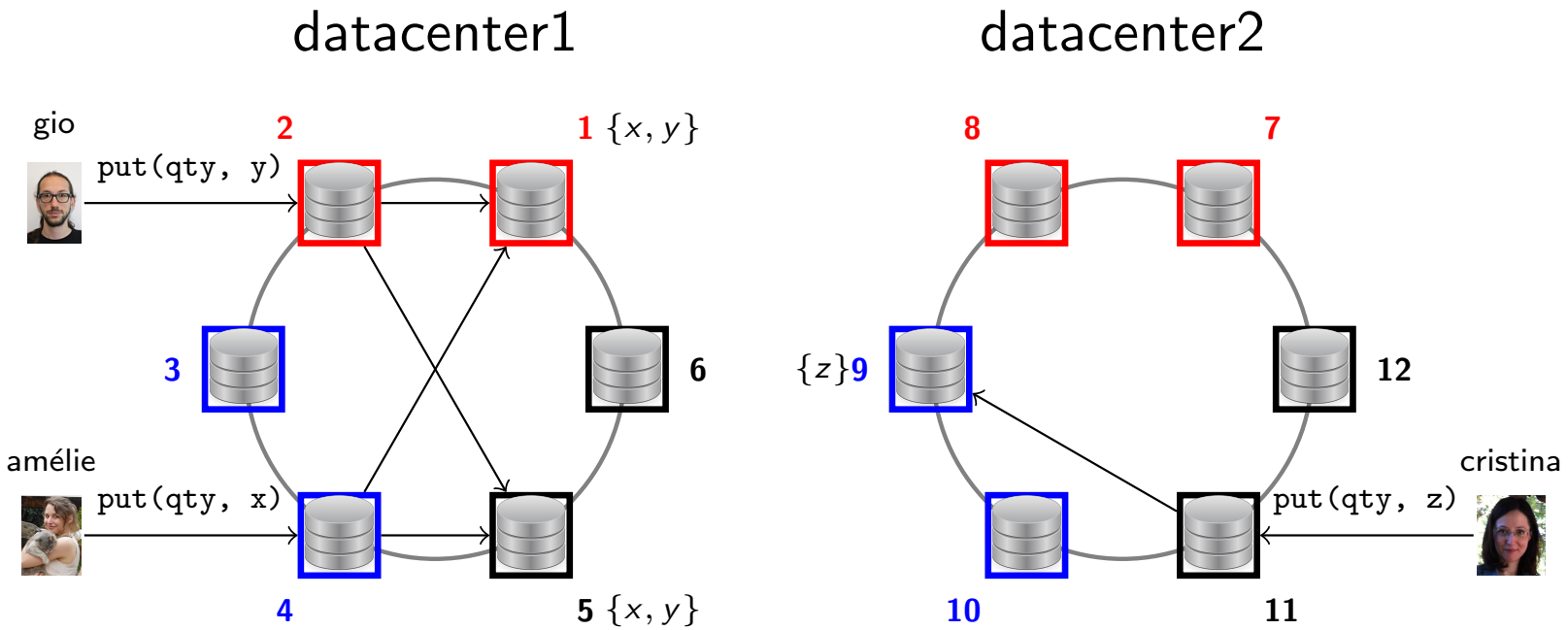
It depends

- on the consistency level of the operations reads, writes
- on <u>when</u> the reads take place;

<u>eventually</u> the replicas converge to one value (which one?)

Cassandra eventually consistent

Let's order 



Which is the value of qty ?

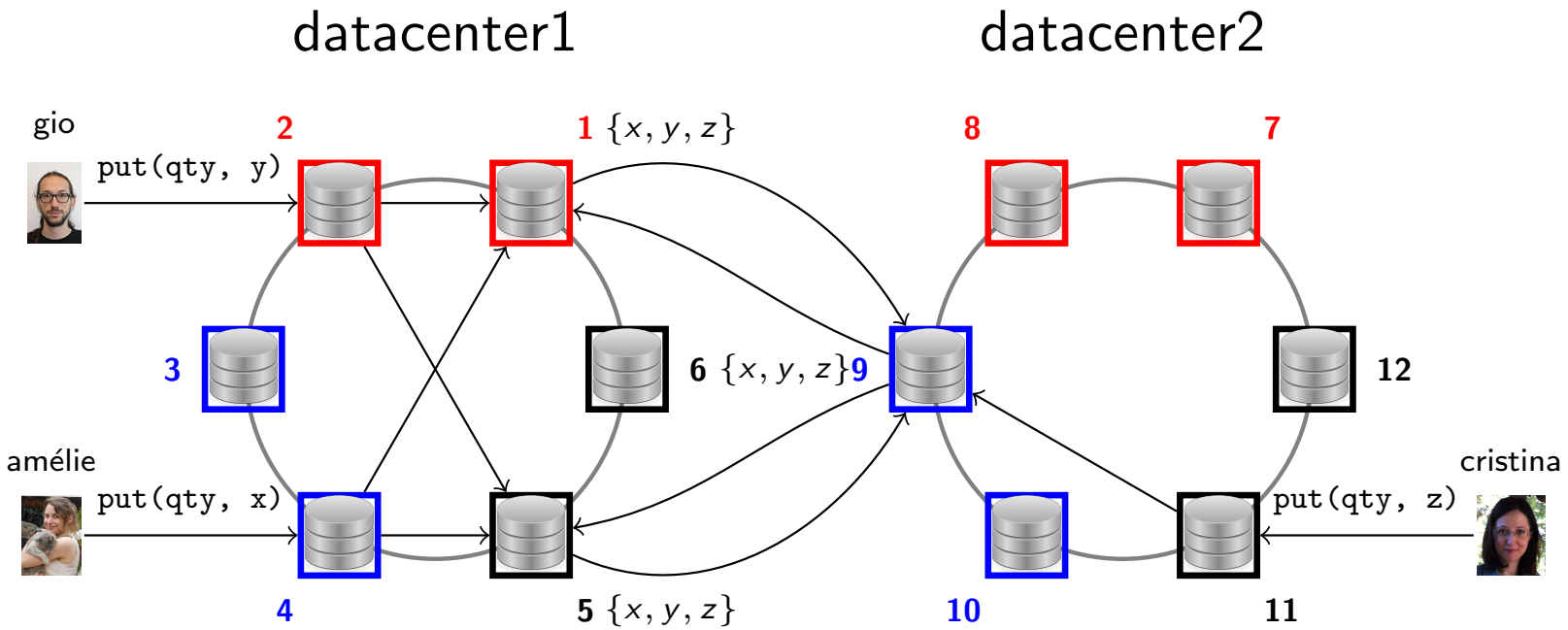
It depends

- on the consistency level of the operations reads, writes
- on <u>when</u> the reads take place;

<u>eventually</u> the replicas converge to one value (which one?)

Cassandra eventually consistent

Let's order 



Which is the value of qty ?

It depends

- on the consistency level of the operations reads, writes
- on <u>when</u> the reads take place;

<u>eventually</u> the replicas converge to one value (which one?)

Cassandra eventually consistent

How many  have we ordered ?

Which is the value of `qty` ?

eventually
replicas know all the possible values for `qty`
$\{x, y, z\}$

final value depends on the
conflict resolution policy

Cassandra eventually consistent

Conflict resolution policy

Last write wins

implemented via

Conflict-free Replicated Data Types

Details on CRDTS is out of scope,
contact me if you are interested

Very interesting material

- Talk by M. Shapiro on SEC and CRDTS
- Results of testing Cassandra via Jepsen

PRACTICE

back

Questions we will address

Server side

How to set-up and run a cassandra node ?

Client side

How to interact with the storage system ?

Next week

Install Cassandra

official instructions

`https://cassandra.apache.org/doc/latest/getting_started/installing.html`

Local set-up

Download and extract a tgz file or clone the official git repository

`git clone https://github.com/apache/cassandra.git`

- Everything in directory `apache-cassandra-*`
- Great to modify the code and run your home-made version

System-wide set-up

Install cassandra package via `apt-get` | `yum` | `...`

- Requires being a sudoer
- Requires dealing with package sources and their GPG keys
- Spreads cassandra files in the filesystem rightfully so!

Install Cassandra

System-wide, in Ubuntu

Steps tested on Ubuntu 22.04.

1. `$ echo "deb [signed-by=/etc/apt/keyrings/apache-cassandra.asc] https://debian.cassandra.apache.org 41x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list`
   `deb https://debian.cassandra.apache.org 41x main`
2. `$ curl -o /etc/apt/keyrings/apache-cassandra.asc https://downloads.apache.org/cassandra/KEYS`
3. `$ sudo apt-get update`
4. `$ sudo apt-get install cassandra`

- Executable script: `/usr/sbin/cassandra`
- jar files: `/usr/share/cassandra/`

WARNING

```
update-rc.d:  warning:  start and stop actions are no
longer supported; falling back to defaults
```

Install Cassandra

If server started by the system *before* you configure the node

1. stop the server
   ```
   $ sudo service cassandra stop
   ```
2. apply brute force
   ```
   $ sudo rm -rf /var/lib/cassandra/*
   ```
3. prepare configuration files

Prerequisite

As we have seen Cassandra is written in java,
to execute it you need a working JVM.
Make sure you run JVM version $>= 8$.

Useful commands:

- `$ java -version`
- `$ sudo update-alternatives --config java` in Ubuntu

Configuration

before starting Cassandra

Decide

- a unique cluster name
- a naming convention for each data center and rack

For instance:

cluster	MyWorldWideService
data centers	datacenterEU, datacenterCANADA
racks	rack1, rack2, rack3

Choose the names carefully: renaming a data center after a node bootstrap is not an easy task.

Note

The rack names must be exactly the same in each data center.

Configuration

before starting Cassandra

We have to give each Cassandra node many information

- Which cluster, data center, rack does the node belong to ?
- Which are the seed nodes ?
- Which ports to use ?
  - 7001: Cassandra TLS/SSL inter-node cluster communication
  - 7199: Cassandra JMX monitoring port (only enabled on localhost for security reasons, you must use SSH to connect the machine).
  - 9042: CQL native client port
- Which gossip protocol to use ?
- Authentication options
- Possibly configure the memory usage

Configuration

The relevant files are located in `/etc/cassandra/`

- `cassandra.yaml` main config file
  contains the cluster name, the seed list, ...
- `cassandra-env.sh` environment configuration
  memory bounds can be configured in this file (more to come) ...
- **cassandra-rackdc.properties**
- **cassandra-topology.properties**
  Gossip configuration: data center, rack
- **jvm-server.options** JVM flags used by cassandra
  Set-up location of config file, num. of processors (more to come) ...
- logback.xml logging configuration

The standard log file is `/var/log/cassandra.log`

How to put two nodes in the same cluster ?

Sketch

- `cassandra.yaml`:
  `cluster_name: 'Punk'`
  `seeds: ...`
  `num_tokens: ...`
- cassandra-rackdc.properties:
  `dc=DataCenterName`
  `rack=RackName`

- `cassandra.yaml`:
  `cluster_name: 'Punk'`
  `seeds: ...`
  `num_tokens: ...`
- cassandra-rackdc.properties:
  `dc=DataCenterName`
  `rack=RackName`

The file cassandra-rackdc.properties is used by the GossipingPropertyFileSnitch. If you use a different snitch protocol you may need to use the `cassandra-topology.properties`

How to set num. virtual nodes in a Cassandra node ?

- `cassandra.yaml`:

⋮

`num_tokens: 4`

Example

```
Datacenter: SophieGermainInfo
==========
Address    Rack         Status  State   Load       Address    ...  Owns     Token
                                                                             7836813856671159227
127.0.0.1  office_4021  Up      Normal  98.46 KiB  127.0.0.1  ...  100.00%  -8560640244036590679
127.0.0.1  office_4021  Up      Normal  98.46 KiB  127.0.0.1  ...  100.00%  -2634741523618209362
127.0.0.1  office_4021  Up      Normal  98.46 KiB  127.0.0.1  ...  100.00%  -2582772119120308142
127.0.0.1  office_4021  Up      Normal  98.46 KiB  127.0.0.1  ...  100.00%  7836813856671159227
```

How to start Cassandra

System-wide set-up

- Via system tools
  - `$ sudo service cassandra start | stop | restart`
  - `$ sudo systemctl start | stop | restart cassandra`

- Executing by hand the starting script
  `$ sudo -u cassandra /usr/sbin/cassandra -f`

  Useful options

  `-f` runs the process in **f**oreground
  `-p` writes the **P**ID of the process in a file

Quick check

After starting cassandra, perform the following "checks"

1. `$ ps aux | grep cassandra`

   Quiz
   - Which user owns the process?
   - Which program runs cassandra ?
   - What is -XX used for ?

2. `$ nodetool info` more on `nodetool` later

WARNING

do **not** use the command

`$ sudo service cassandra status`

!!!

The output of this command is misleading if you are not familiar with it.

Which class is run ?

CassandraDaemon

so let's check
CassandraDaemon.java

```
public static void main(String[] args)
 {
     instance.activate();
 }
```

Complexity is split among

- Declaration of `instance`
- Calls to `setup()` and `start()`

How to manage a cassandra node?

`nodetool`

synopsis simplified:

```
$ nodetool [-p port] [-h host] command
```

commands to get info

info gossipinfo status ring netstats getendpoints

commands to manipulate a node

decommission flush system repair cleanup

Source code: NodeTool.java

Show the information about one single node

```
user@host: $ nodetool info
 ID                        :aa08c4a8-63af-434b-a42a-f178134685c3
 Gossip active             :true
 Thrift active             :false
 Native Transport active:true
 Load                      :236.81 KiB
 Generation No             :1597354312
 Uptime (seconds)          :35958
 Heap Memory (MB)          :292.55 / 1924.00
 Off Heap Memory (MB)      :0.00
 Data Center               :datacenter1
 Rack                      :rack1
 Exceptions                :0
 Key Cache                 :entries 22, size 1.82 KiB, capacity 96 MiB,...
 Row Cache                 :entries 0, size 0 bytes, capacity 0 bytes,...
 Counter Cache             :entries 0, size 0 bytes, capacity 48 MiB,...
 Chunk Cache               :entries 12, size 768 KiB, capacity 449 MiB,...
 Percent Repaired          :100.0%
 Token                     :(invoke with -T/--tokens to see all 256 tokens)
```

Unique snitch and partitioner per cluster

```
user@host: $ nodetool describecluster
Cluster Information:
Name:  Punk_is_not_dead
Snitch: org.apache.cassandra.locator.GossipingPropertyFileSnitch
DynamicEndPointSnitch:  enabled
Partitioner: org.apache.cassandra.dht.Murmur3Partitioner
Schema versions:
    5b547ad6-544b-3208-b432-9a9778c9b9da:  [127.0.0.1]
```

Show status+state and minimal info about every node

```
user@host: $ nodetool status
Datacenter:  datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
 --   Address     Load          Tokens   Owns (effective)   ...
 UN   127.0.0.1   236.81 KiB    256      100.0%             ...
```

- datacenter1 = the name of the data center the node is in
- UN = the node is UP
- 256 = the number of tokens managed by the node
- aa...c3 = the unique node ID within the cluster
- rack1 = the name of the rack the node is in

Show status+state and minimal info about every node

```
user@host: $ nodetool status
Datacenter:  datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
 ...   Host ID                                      Rack
 ...   aa08c4a8-63af-434b-a42a-f178134685c3        rack1
```

- datacenter1 = the name of the data center the node is in
- UN = the node is UP
- 256 = the number of tokens managed by the node
- aa...c3 = the unique node ID within the cluster
- rack1 = the name of the rack the node is in

```
user@host: $ nodetool ring
 Datacenter:  SophieGermainInfo
 ==========
 Address     Rack          Status   State    Load          ...
                                                          ...
 127.0.0.1   office_4021   Up       Normal   98.46 KiB     ...
 127.0.0.1   office_4021   Up       Normal   98.46 KiB     ...
 127.0.0.1   office_4021   Up       Normal   98.46 KiB     ...
 127.0.0.1   office_4021   Up       Normal   98.46 KiB     ...
```

```
user@host: $ nodetool ring
 Datacenter:  SophieGermainInfo
 ==========
 Address     ...   Owns      Token
                             7836813856671159227
 127.0.0.1   ...   100.00%   -8560640244036590679
 127.0.0.1   ...   100.00%   -26347415236182093 62
 127.0.0.1   ...   100.00%   -25827721191203081 42
 127.0.0.1   ...   100.00%   7836813856671159227
```

Show the information about the gossip protocol

```
user@host: $ nodetool gossipinfo
 localhost/127.0.0.1
 generation:1597354312
 heartbeat:37720
 STATUS:15:NORMAL,-1050938293068292307
 LOAD:37703:147192.0
 SCHEMA:31779:a6d18c1a-892e-33d0-9326-0cb653f9d953
 DC:7:datacenter1
 RACK:9:rack1
 RELEASE_VERSION:5:3.11.7
 RPC_ADDRESS:4:127.0.0.1
 NET_VERSION:2:11
 HOST_ID:3:aa08c4a8-63af-434b-a42a-f178134685c3
 RPC_READY:18:true
 TOKENS:14:<hidden>
```

getendpoints

```
$ nodetool [-p port] [-h host] getendpoints keyspace
                        table key
```

Till 2022, everything worked:

```
user@host: $ nodetool getendpoints demo data -8751398437351544090
```

getendpoints

```
$ nodetool [-p port] [-h host] getendpoints keyspace
                        table key
```

Till 2022, everything worked:

```
user@host: $ nodetool getendpoints demo data -8751398437351544090
127.0.0.1
```

Nowadays

```
user@host: $nodetool -Dcom.sun.jndi.rmiURLParsing=legacy
getendpoints demo students -693433537833096
error:  For input string:  "-693433537833096" [...]

user@host: $nodetool -Dcom.sun.jndi.rmiURLParsing=legacy
getendpoints demo students -6934
127.0.0.1
```

A good account of the situation is here:

https://stackoverflow.com/questions/30514237/what-node-does-cassandra-store-data-on

```
user@host: $ nodetool netstats
Mode:  NORMAL
Not sending any streams.
Read Repair Statistics:
Attempted:  0
Mismatch (Blocking):  0
Mismatch (Background):  0
 Pool Name         Active   Pending   Completed   Dropped
 Large messages       n/a         0           0         0
 Small messages       n/a         0           0         0
 Gossip messages      n/a         0           0         0
```

For a detailed example of a real heavy use-case of `nodetool`, read this post:

`https://thelastpickle.com/blog/2011/12/15/Anatomy-of-a-Cassandra-Partition.html`

WARNING: the `cassandra-cli` is deprecated and removed since version 2.2 of cassandra.

Further instructions for `nodetool`

decommission Makes a live node stream its data to the next node on the ring to replicate appropriately. This assigns the ranges that the node was responsible for to other nodes.

flush system Writes one or more tables from the RAM into a disk.

repair Repairs inconsisent data in one or more nodes in a cluster when **all** involved replicas are up and accessible. If one replica is DOWN, the repair process halts (an error occurs). Non-trivial task: if inconsistent data is detected by comparing the Merkle trees, the out-of-date data is amended by streaming it from the nodes with the newer data.

cleanup Remove keyspaces and partition keys no longer belonging to a node. This is usefull after adding a new node into a ring and re-distributing the data to it. If no keyspace is specified, this command cleans all keyspaces no longer belonging to a node.

How to change snitch protocol/ data center / rack ?

1. `$ nodetool decommission`
2. Stop cassandra
3. Modify relevant config files:
   - cassandra.yaml
   - cassandra-rackdc.properties or cassandra-topology.properties
4. start cassandra with the options

   `-Dcassandra.ignore_dc=true, -Dcassandra.ignore_rack=true`
5. `$ nodetool repair`
6. `$ nodetool cleanup`

Typical issue: `CassandraDaemon.java:775 - Cannot start node if snitch's data center (SophieGermainInfo) differs from previous data center (dc1). Please fix the snitch configuration, decommission and rebootstrap this node or use the flag -Dcassandra.ignore_dc=true.`

How to change snitch protocol/ data center / rack ?

brute force approach

1. stop cassandra
2. `$ sudo rm -rf /var/lib/cassandra/*`
3. Modify relevant config files:
   - cassandra.yaml
   - cassandra-rackdc.properties or cassandra-topology.properties
4. start cassandra

Further reading

Cassandra: The Definitive Guide, 3rd Edition, O'Reilly 2020
J. Carpenter, E. Hewitt

- Chapter 3: Installing Cassandra
- Chapter 6: The Cassandra Architecture
- Chapter 9: Writing and Reading Data

That's the story.
Thank you for the attention

Questions?

TROUBLESHOOTING

Troubleshooting

Cassandra version 3.11.8

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
```

Troubleshooting

Cassandra version 3.11.8

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
...
intx ThreadPriorityPolicy=42 is outside the allowed range [ 0 ...1 ]
Improperly specified VM option 'ThreadPriorityPolicy=42'
Error:  Could not create the Java Virtual Machine.
Error:  A fatal exception has occurred.  Program will exit.
host@user:~$
```

Troubleshooting

Cassandra version 3.11.8

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
...
intx ThreadPriorityPolicy=42 is outside the allowed range [ 0 ...1 ]
Improperly specified VM option 'ThreadPriorityPolicy=42'
Error:  Could not create the Java Virtual Machine.
Error:  A fatal exception has occurred.  Program will exit.
host@user:~$
```

Alternative solutions

- Run JVM version 8 (exactly the 8!)
- Change to value of `ThreadPriorityPolicy` in the file `jvm-server.options`
- Further details can be found here

Troubleshooting

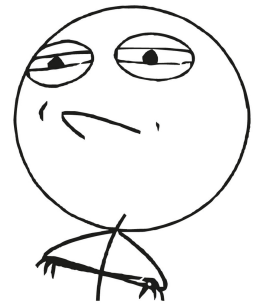
```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
```

Troubleshooting

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
Killed
host@user:~$
```

Troubleshooting

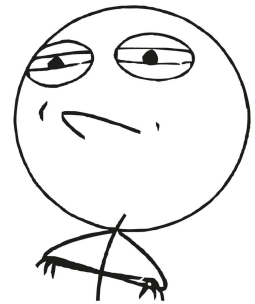
```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
Killed
host@user:~$
```



Troubleshooting

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
Killed
host@user:~$
```

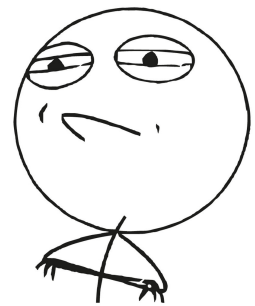
Where are the system log files ?



Troubleshooting

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
Killed
host@user:~$
```

Where are the system log files ? `/var/log/`

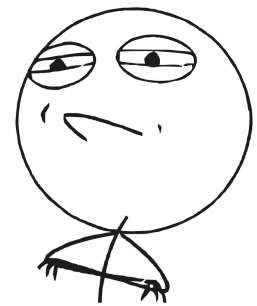


```
host@user:~$ tail -n 2 /var/log/kern.log
Aug 5 14:48:19 alfortville kernel:  [1099672.209415]
Out of memory:  Killed process 27993 (java)
total-vm:3555460kB, anon-rss:1983072kB, file-rss:0kB, shmem-rss:0kB
Aug 5 14:48:19 alfortville kernel:  [1099672.404307]
oom_reaper:  reaped process 27993 (java),
now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB
```

Troubleshooting

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
Killed
host@user:~$
```

Where are the system log files ? `/var/log/`



```
host@user:~$ tail -n 2 /var/log/kern.log
Aug 5 14:48:19 alfortville kernel:  [1099672.209415]
Out of memory:  Killed process 27993 (java)
total-vm:3555460kB, anon-rss:1983072kB, file-rss:0kB, shmem-rss:0kB
Aug 5 14:48:19 alfortville kernel:  [1099672.404307]
oom_reaper:  reaped process 27993 (java),
now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB
```

Further material

- Taming the OOM killer
- LinuxMM: OOM_Killer

Troubleshooting

Set manually the amount of memory used by cassandra:
modify the file `jvm-server.options`

- -Xms minimum amount of memory used by the JVM
- -Xmx maximum amount of memory used by the JVM

  Either set both of them, or none of them !

See JVM doc

Troubleshooting

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
⋮
ERROR [COMMIT-LOG-ALLOCATOR] ...
JVMStabilityInspector.java:196 -
Exiting due to error while processing commit log
during initialization.
org.apache.cassandra.io.FSWriteError:
java.nio.file.AccessDeniedException:
/var/lib/cassandra/commitlog/CommitLog-7-1732370642008.log
host@user:~$
```

Troubleshooting

```
host@user:~$ sudo -u cassandra /usr/sbin/cassandra -f
⋮
ERROR [COMMIT-LOG-ALLOCATOR] ...
JVMStabilityInspector.java:196 -
Exiting due to error while processing commit log
during initialization.
org.apache.cassandra.io.FSWriteError:
java.nio.file.AccessDeniedException:
/var/lib/cassandra/commitlog/CommitLog-7-1732370642008.log
host@user:~$
```

Idea behind solutions

Make sure the user running the process has sufficient rights on the FS to write data and log files. `chmod`, `chown`, `sudo -u`, `cassandra-env.sh`

Troubleshooting

Cassandra version 3.11.8

```
host@user:~$ nodetool status
```

Troubleshooting

Cassandra version 3.11.8

```
host@user:~$ nodetool status
nodetool:  Failed to connect to '127.0.0.1:7199' -
URISyntaxException:  'Malformed IPv6 address at index 7:
```

Troubleshooting

Cassandra version 3.11.8

```
host@user:~$ nodetool status
nodetool:  Failed to connect to '127.0.0.1:7199' -
URISyntaxException:  'Malformed IPv6 address at index 7:
```

Alternative solutions

- Use the option
  `-Dcom.sun.jndi.rmiURLParsing=legacy`
  in the command line
- Add `-Dcom.sun.jndi.rmiURLParsing=legacy`
  to the environment variable `JAVA_TOOL_OPTIONS`
- Upgrade to Cassandra 3.11.13