# HDB++: HIGH AVAILABILITY WITH

# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

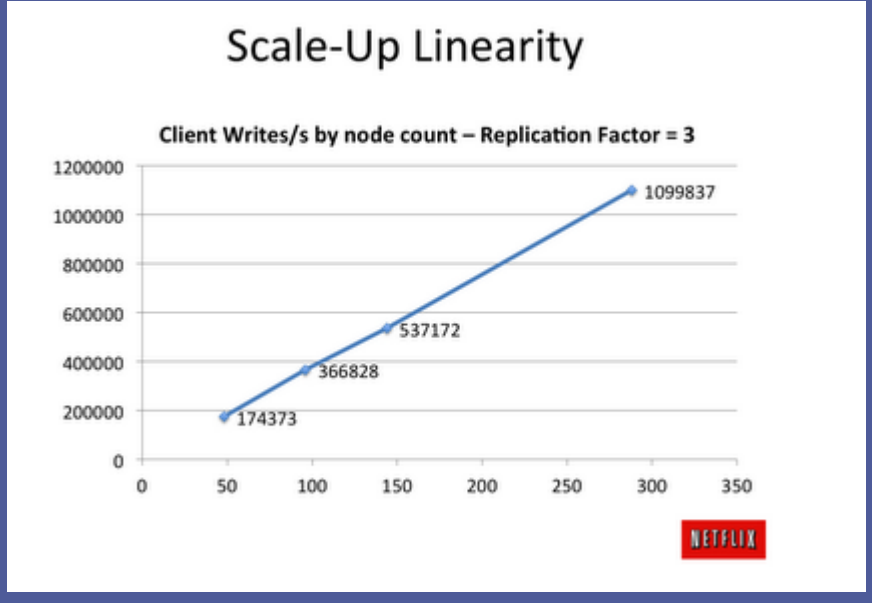
# WHAT IS CASSANDRA?

- Mythology: an excellent Oracle not believed.
- A massively scalable open source NoSQL (Not Only SQL) database
- Created by Facebook
- Open Source since 2008
- Apache license, 2.0, compatible with GPLV3

# WHAT IS CASSANDRA?

- Peer to peer architecture
- No Single Point of Failure
- Replication
- Continuous Availability
- Multi Data Centers support
- 100s to 1000s nodes
- Java
- High Write Throughput
- Read efficiency

# WHAT IS CASSANDRA?




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

# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

# WHO IS USING CASSANDRA?

# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

# CASSANDRA QUERY LANGUAGE

- **CQL: Cassandra Query Language**
- **Very similar to SQL**
- **But restrictions and limitations**
- JOIN requests are forbidden
- No subqueries
- String comparisons are limited (when not using SOLR)
  `select * from my_table where mystring like '%tango%';`
- No OR operator
- Can only apply a WHERE condition on an indexed column (or primary key)

# CASSANDRA QUERY LANGUAGE

- **Collections (64K Limitation):**
- list
- set
- map
- **TTL**
- **INSERT = UPDATE (UPSERT)**
- **Doc:**
  **http://www.datastax.com/documentation/cql/3.1/cql/cql_intro_c.html**
- **cqlsh**

# CASSANDRA QUERY LANGUAGE

```
CREATE TABLE IF NOT EXISTS att_scalar_devdouble_ro (
    att_conf_id timeuuid,
    period text,
    data_time timestamp,
    data_time_us int,
    value_r double,
    quality int,
    error_desc text,
    PRIMARY KEY ((att_conf_id ,period),data_time,data_time_us)
)
WITH comment='Scalar DevDouble ReadOnly Values Table‘;
```

# CASSANDRA QUERY LANGUAGE

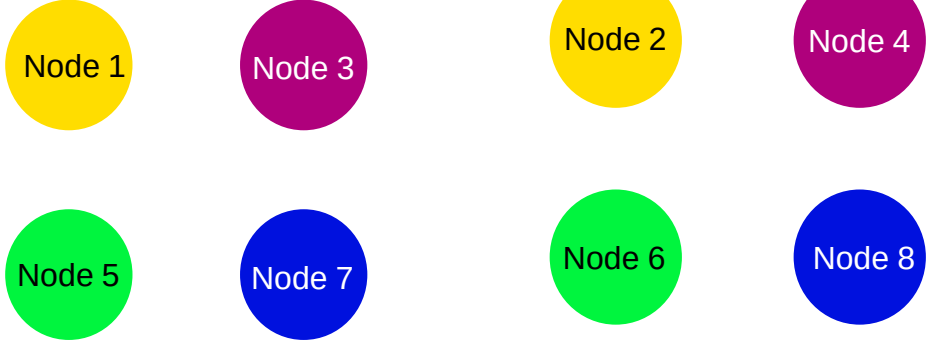
```
CREATE TABLE IF NOT EXISTS att_scalar_devdouble_ro (
    att_conf_id timeuuid,
    period text,
    data_time timestamp,
    data_time_us int,
    value_r double,
    quality int,
    error_desc text,
    PRIMARY KEY ((att_conf_id ,period),data_time,data_time_us)
);
```

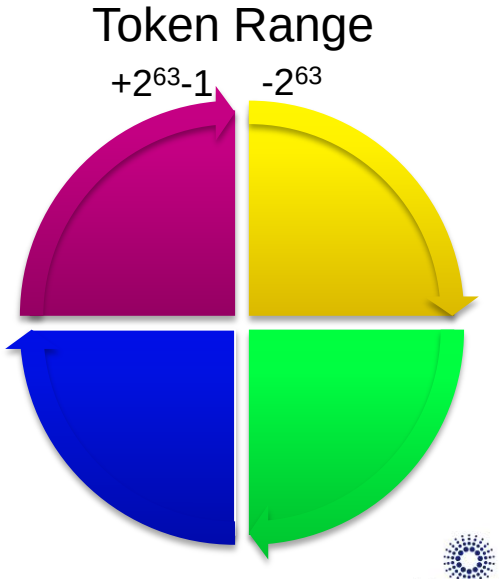

# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

# CASSANDRA ARCHITECTURE

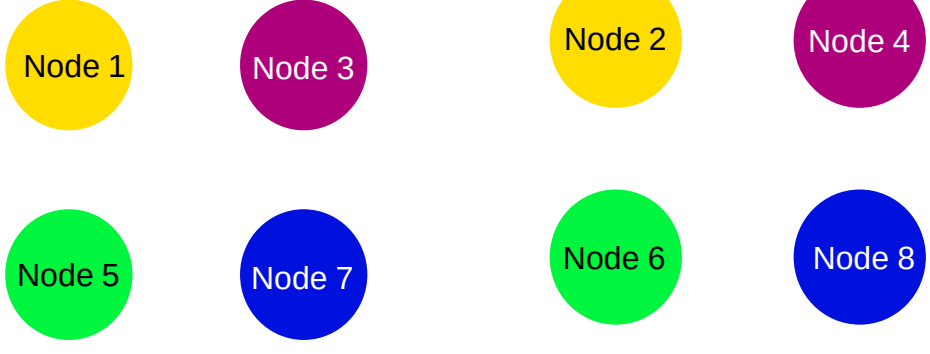
- **Node: one Cassandra instance (Java process)**

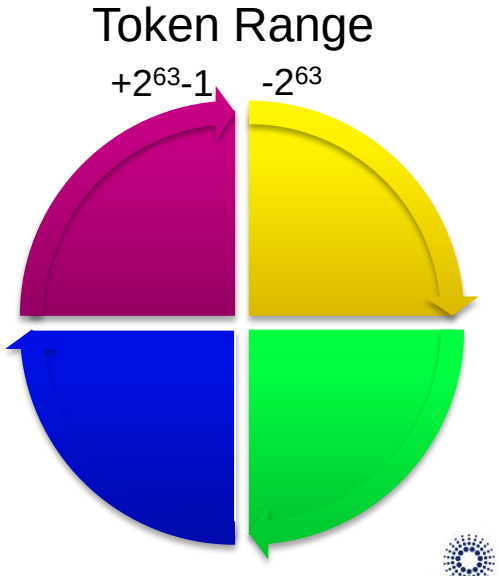


# CASSANDRA ARCHITECTURE

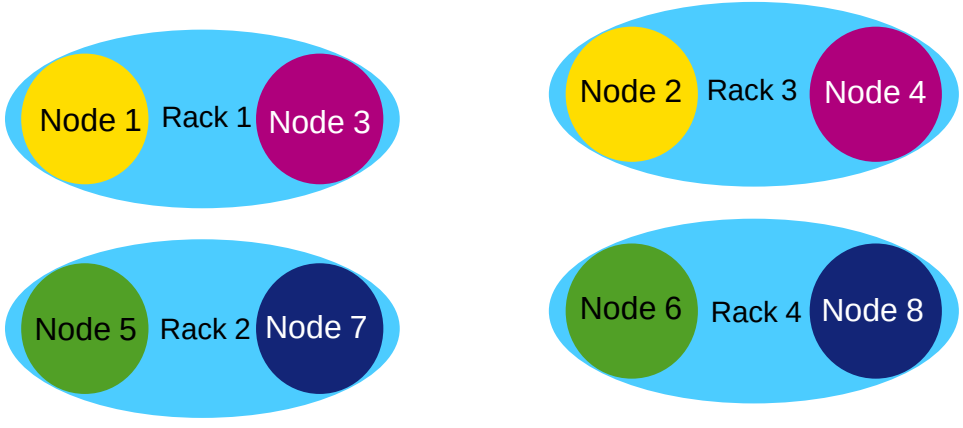
- **Partition: ordered and replicable unit of data on a node identified by a token**
- **Partitioner (based on mumur3 algorithm by default) will distribute the data across the nodes.**

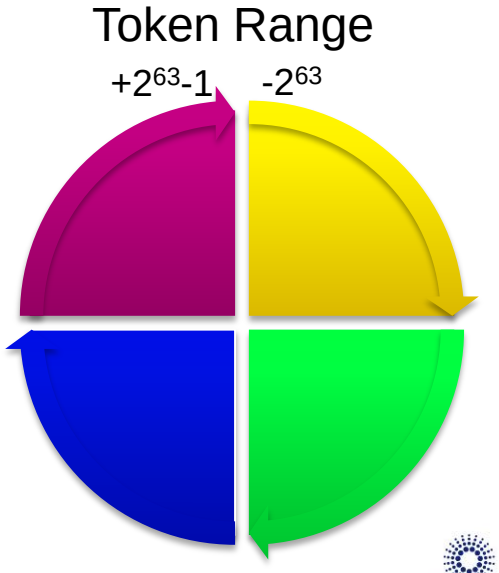


# CASSANDRA ARCHITECTURE

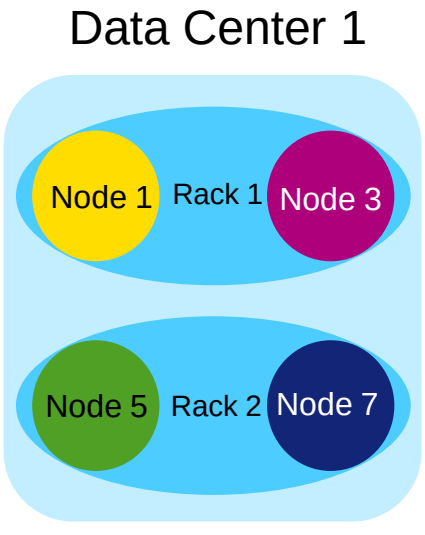
- **Rack**: logical set of nodes

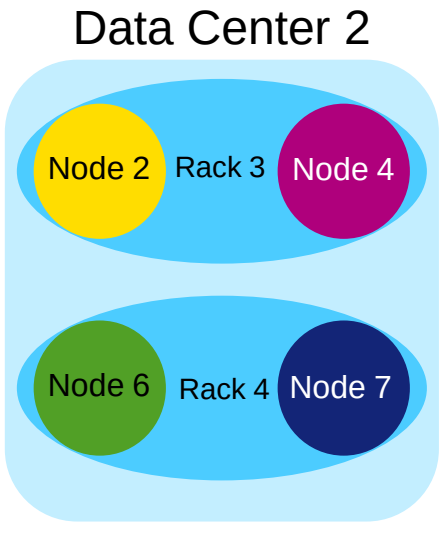


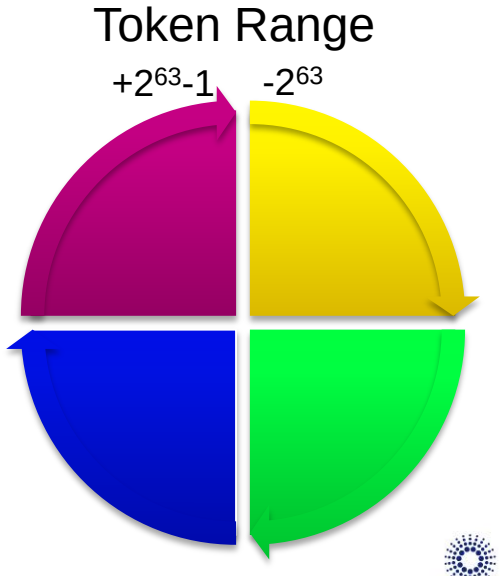



# CASSANDRA ARCHITECTURE

- **Data Center: logical set of racks**

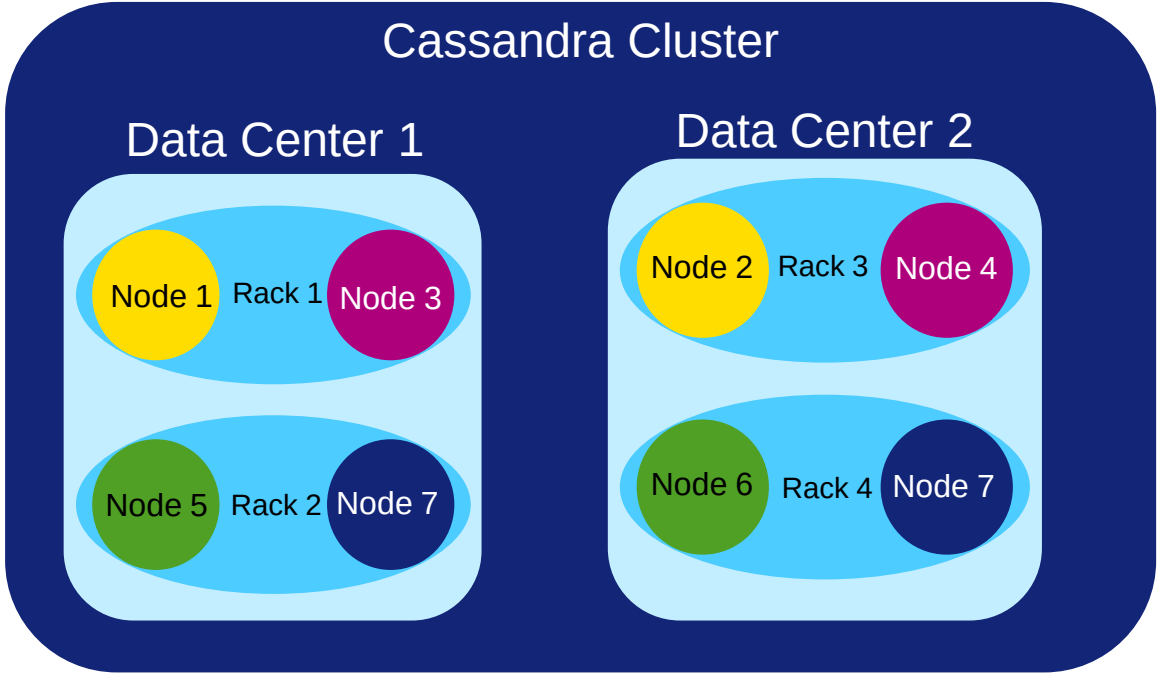


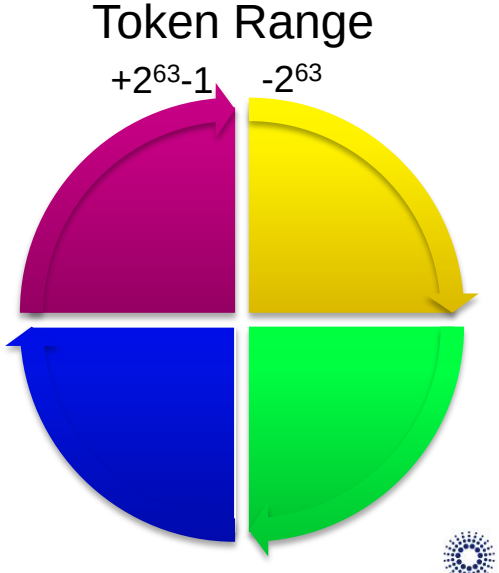



# REQUEST COORDINATION

- **Cluster: full set of nodes which maps to a single complete token ring**



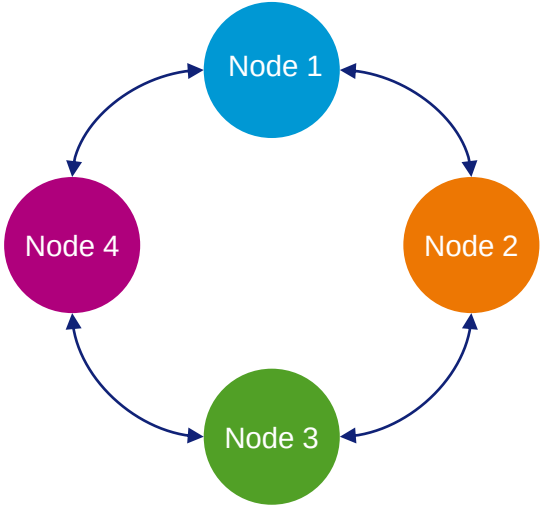



# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

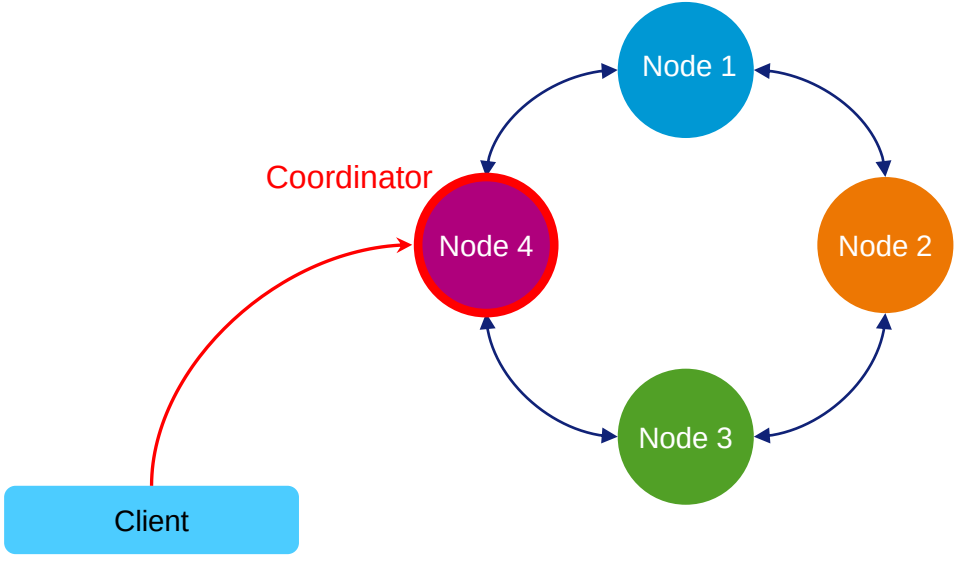
# REQUEST COORDINATION

- **Coordinator: the node chosen by the client to receive a particular read or write request to its cluster**



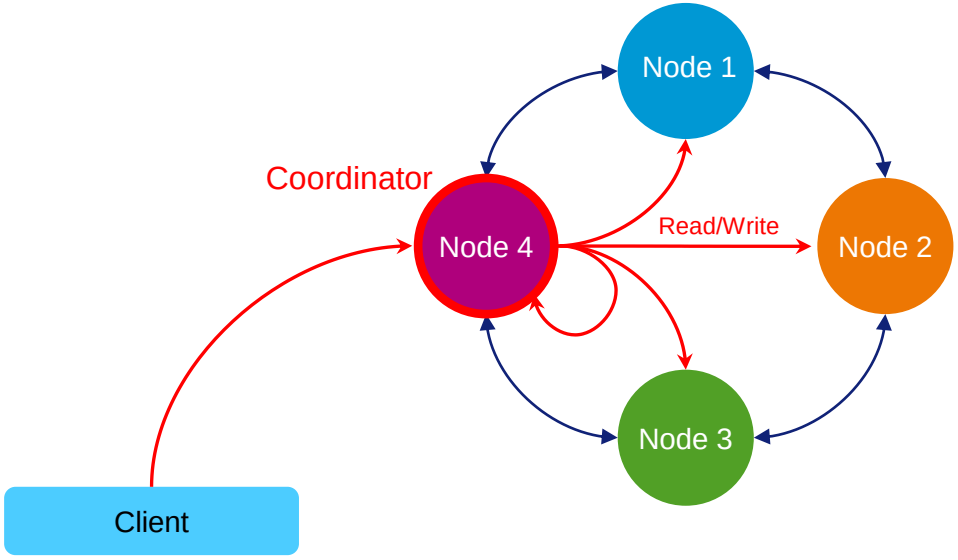
# REQUEST COORDINATION

- **Coordinator: the node chosen by the client to receive a particular read or write request to its cluster**



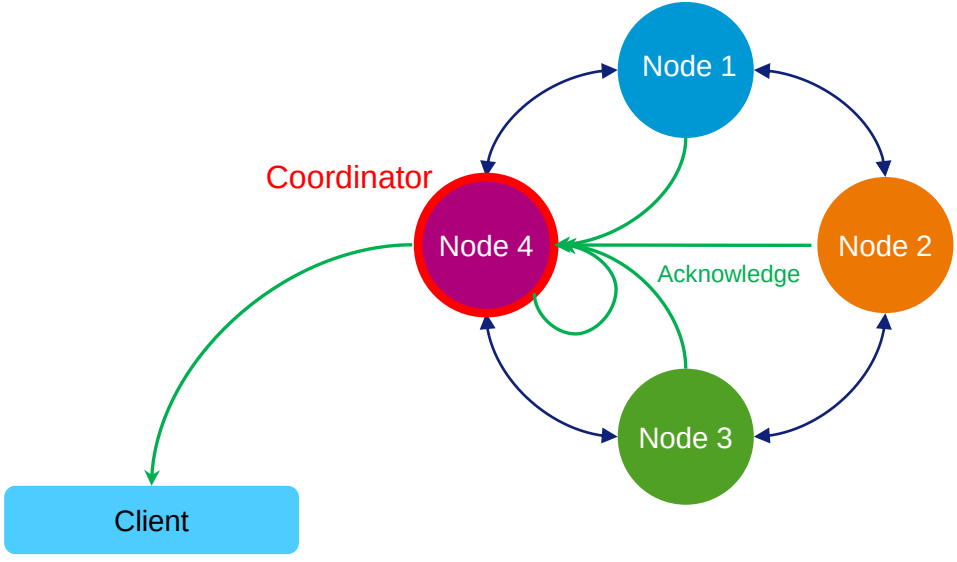
# REQUEST COORDINATION

- **Coordinator: the node chosen by the client to receive a particular read or write request to its cluster**



# REQUEST COORDINATION

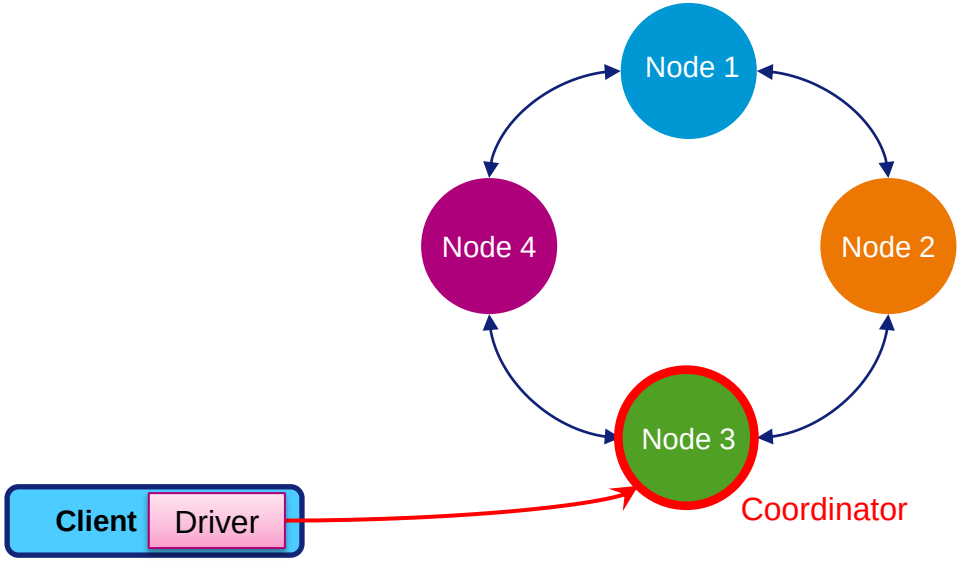
- Any node can coordinate any request
- Each client request may be coordinated by a different node




**No Single Point of Failure**

# REQUEST COORDINATION

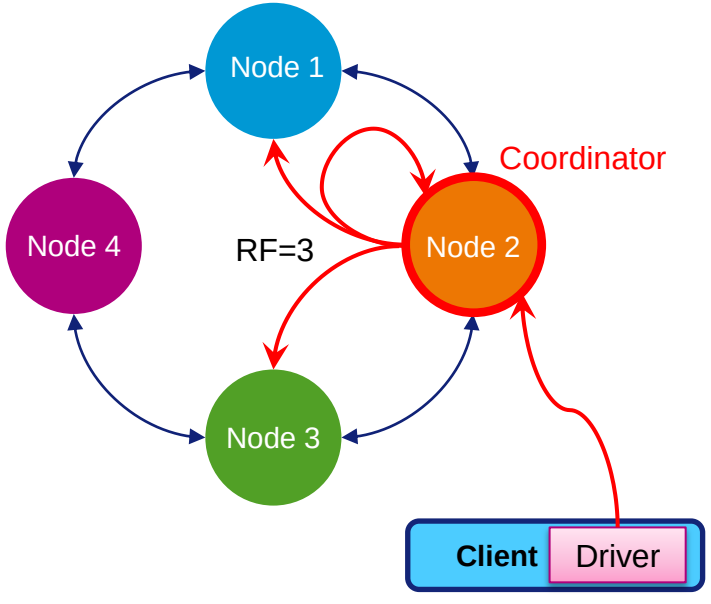
- The Cassandra driver chooses the coordinator node
- Round-Robin pattern, token-aware pattern
- Client library to manage requests
- Many open source drivers for many programming languages




Java
Python
C++
Node.js
C#
Perl
PHP
Go
Clojure
Ruby
Scala
R (GNU S)
Erlang
Haskell
ODBC
Rust

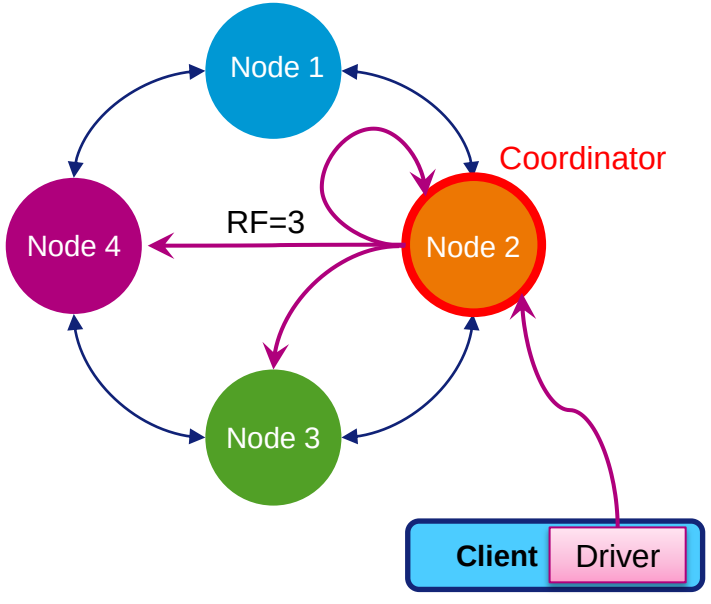
# REQUEST COORDINATION

- The coordinator manages the replication process
- Replication Factor (RF): onto how many nodes should a write be copied
- The write will occur on the nodes responsible for that partition
- 1 ≤ RF ≤ (#nodes in cluster)
- Every write is time-stamped



# REQUEST COORDINATION

- The coordinator manages the replication process
- Replication Factor (RF): onto how many nodes should a write be copied
- The write will occur on the nodes responsible for that partition
- 1 ≤ RF ≤ (#nodes in cluster)
- Every write is time-stamped



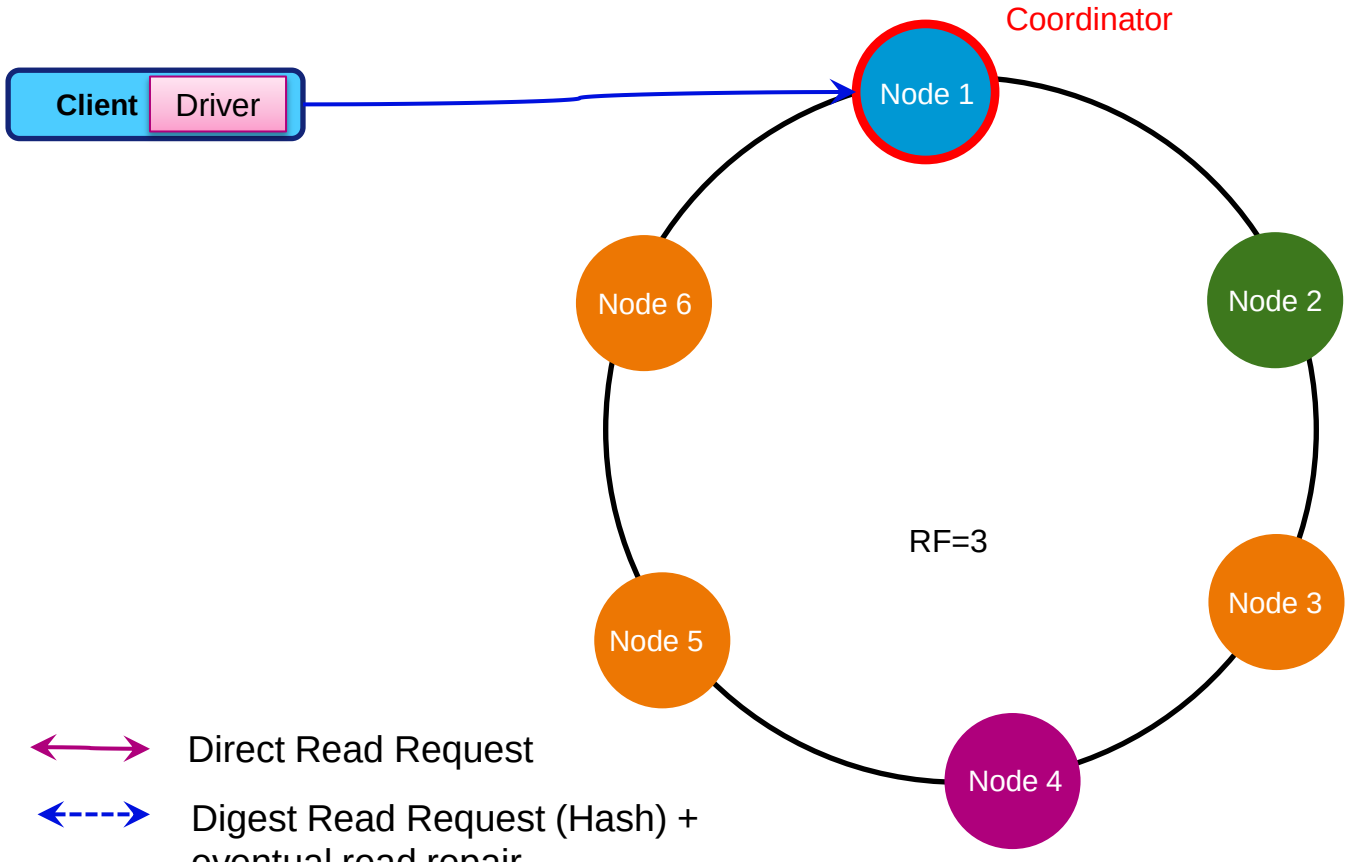
# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

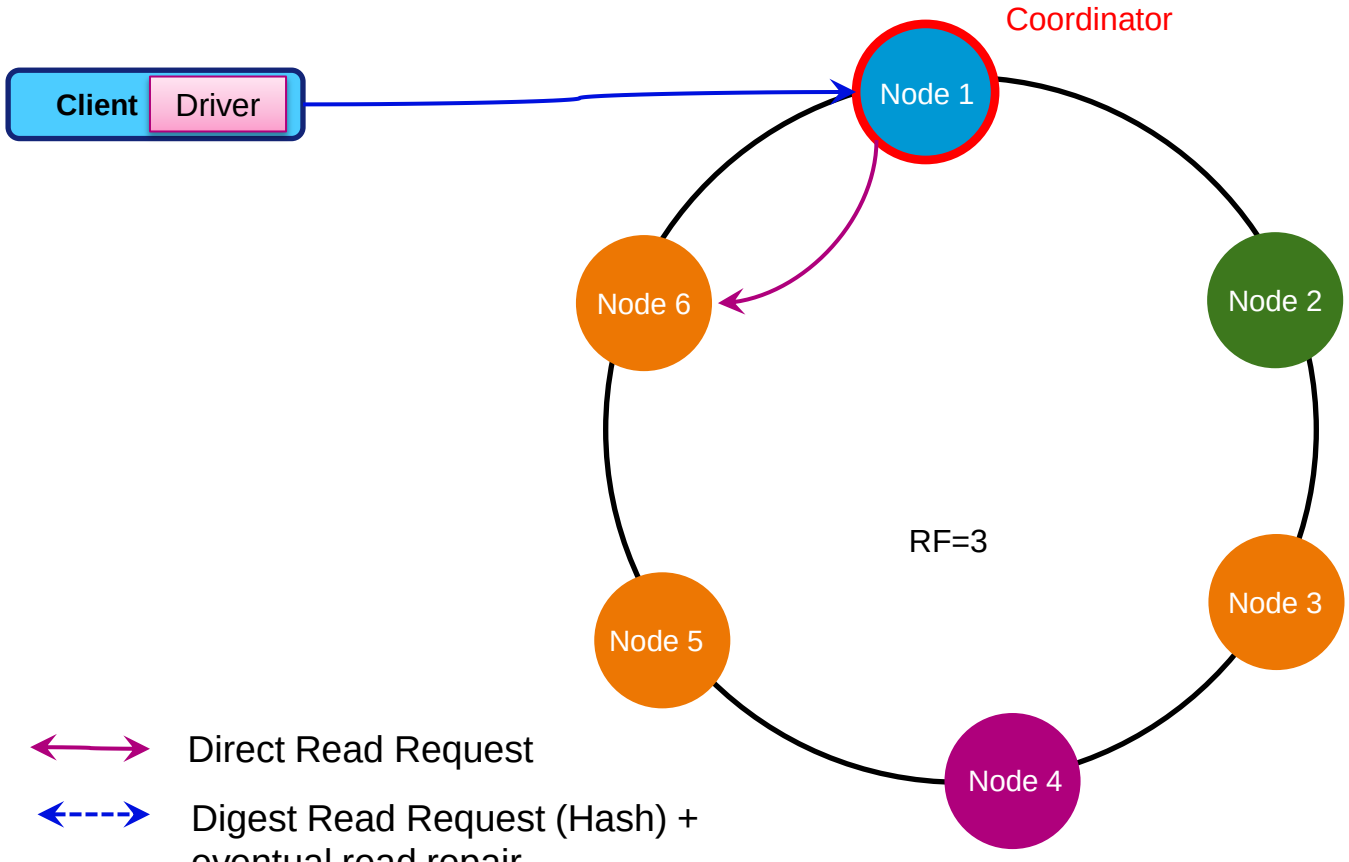
# CONSISTENCY

- **The coordinator applies the Consistency Level (CL)**
- **Consistency Level (CL): Number of nodes which must acknowledge a request**
- **Examples of CL:**
- ONE
- TWO
- THREE
- ANY
- ALL (Not recommended)
- QUORUM (= RF/2 + 1)
- EACH_QUORUM
- LOCAL_QUORUM
- **CL may vary for each request**
- **On success, the coordinator notifies the client (with most recent partition data in case of read request)**

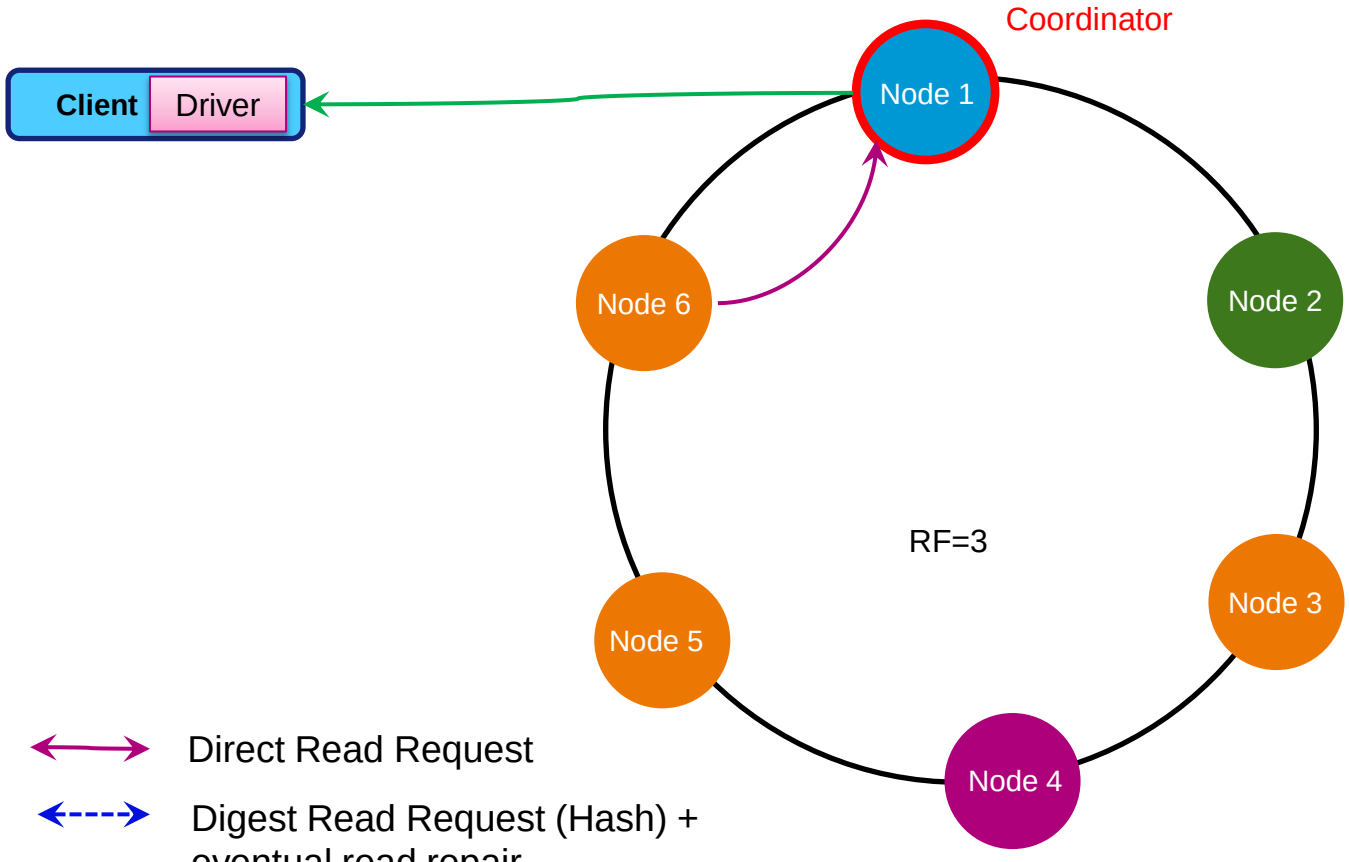
# CONSISTENCY ONE - READ - SINGLE DC



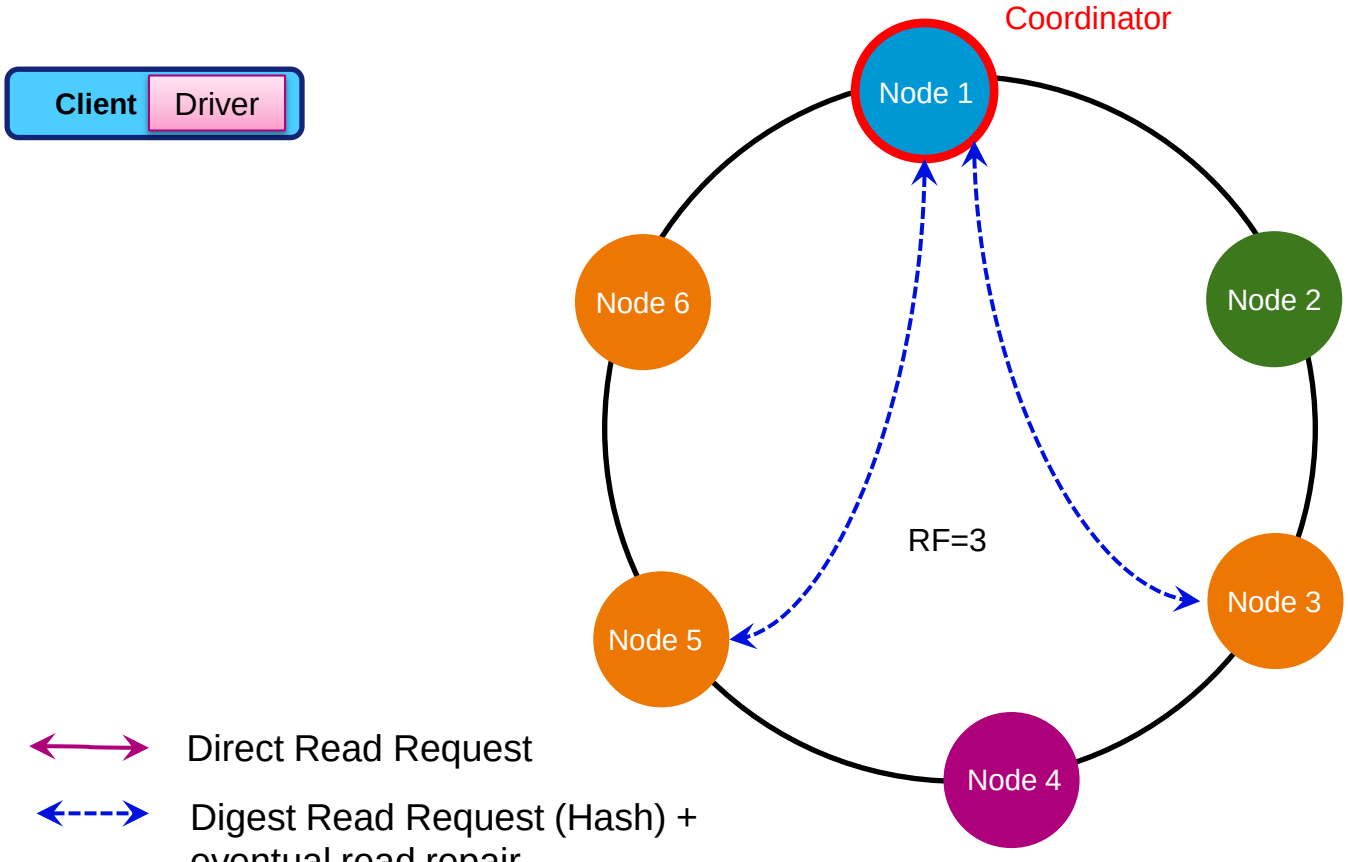
# CONSISTENCY ONE - READ - SINGLE DC



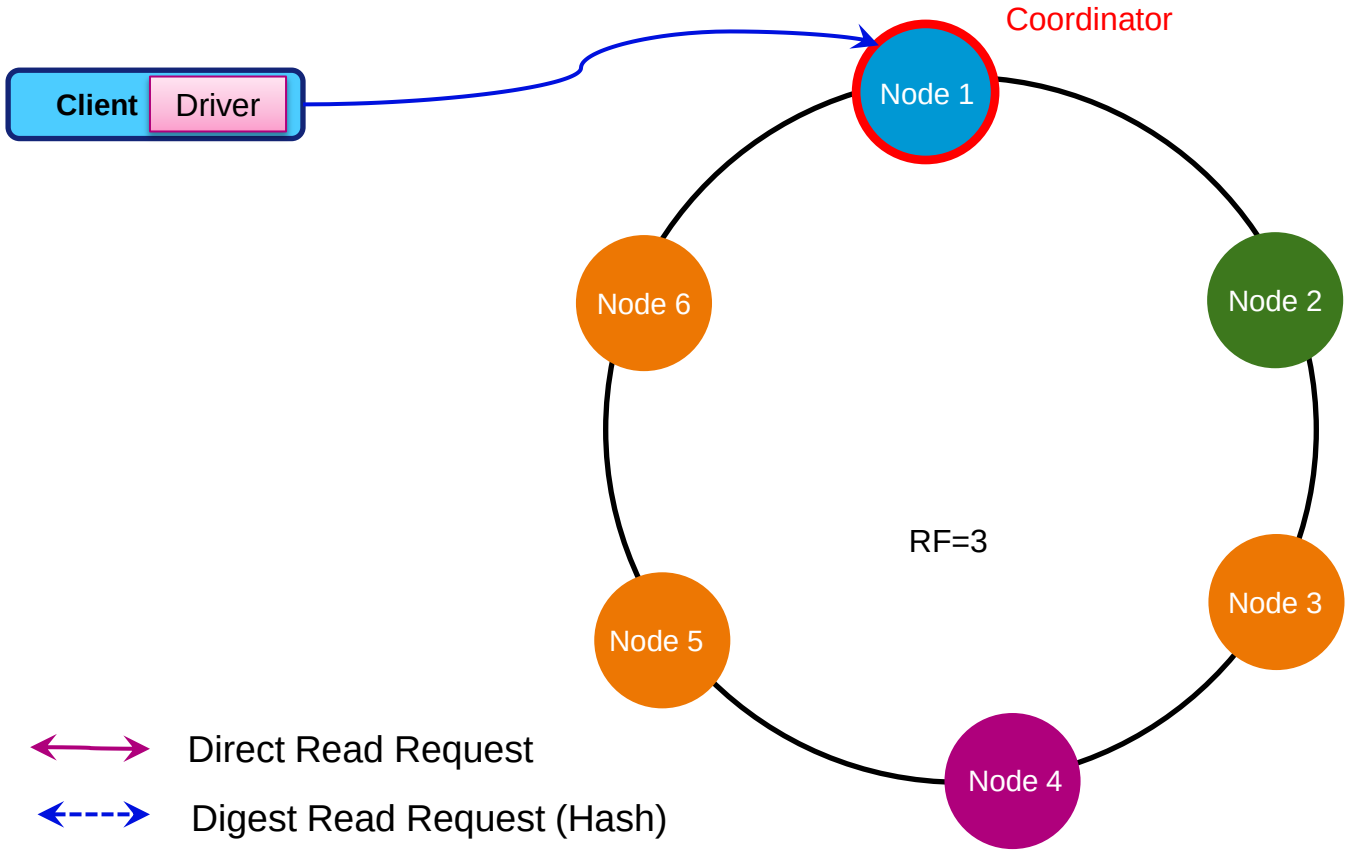
# CONSISTENCY ONE – READ - SINGLE DC



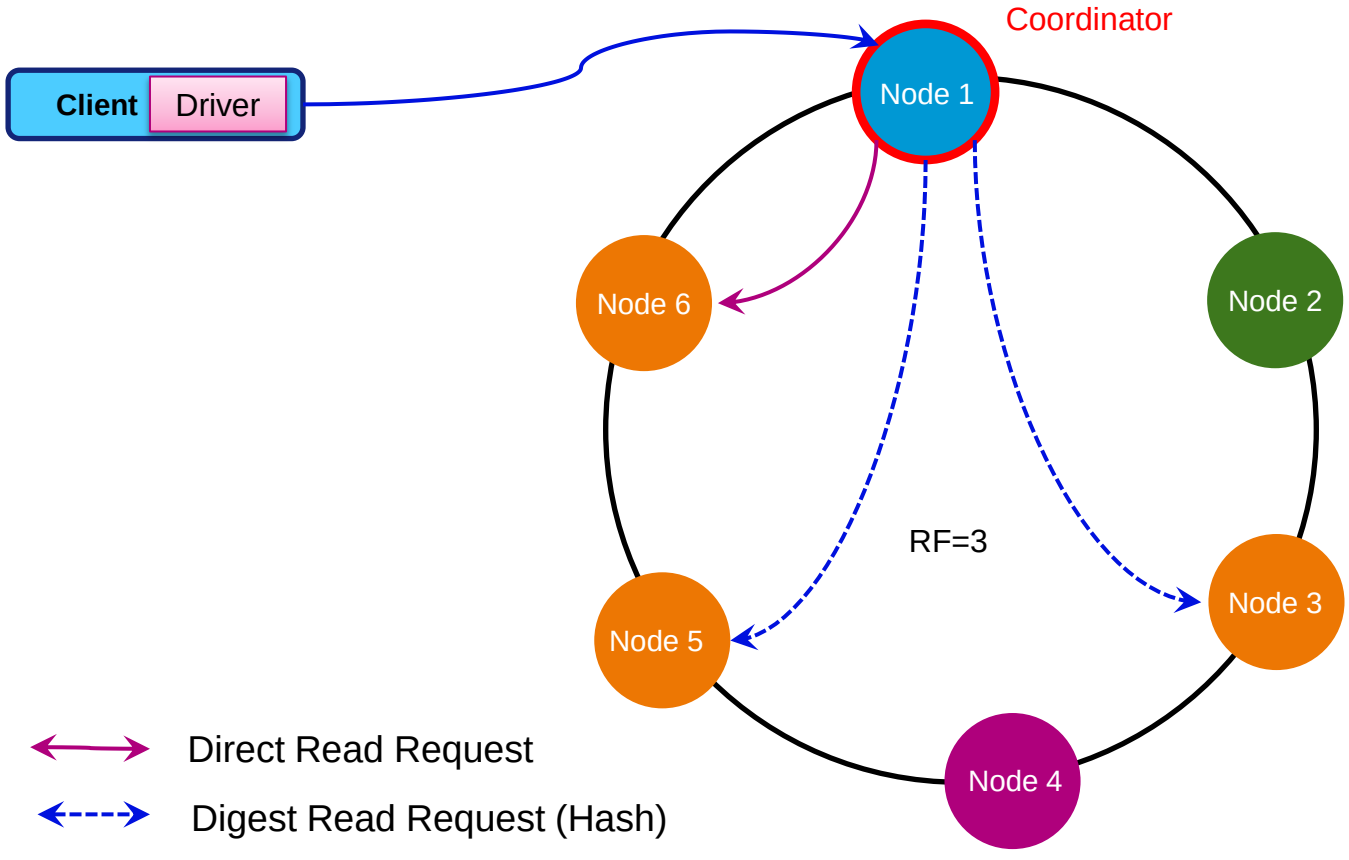
# CONSISTENCY ONE - READ - SINGLE DC



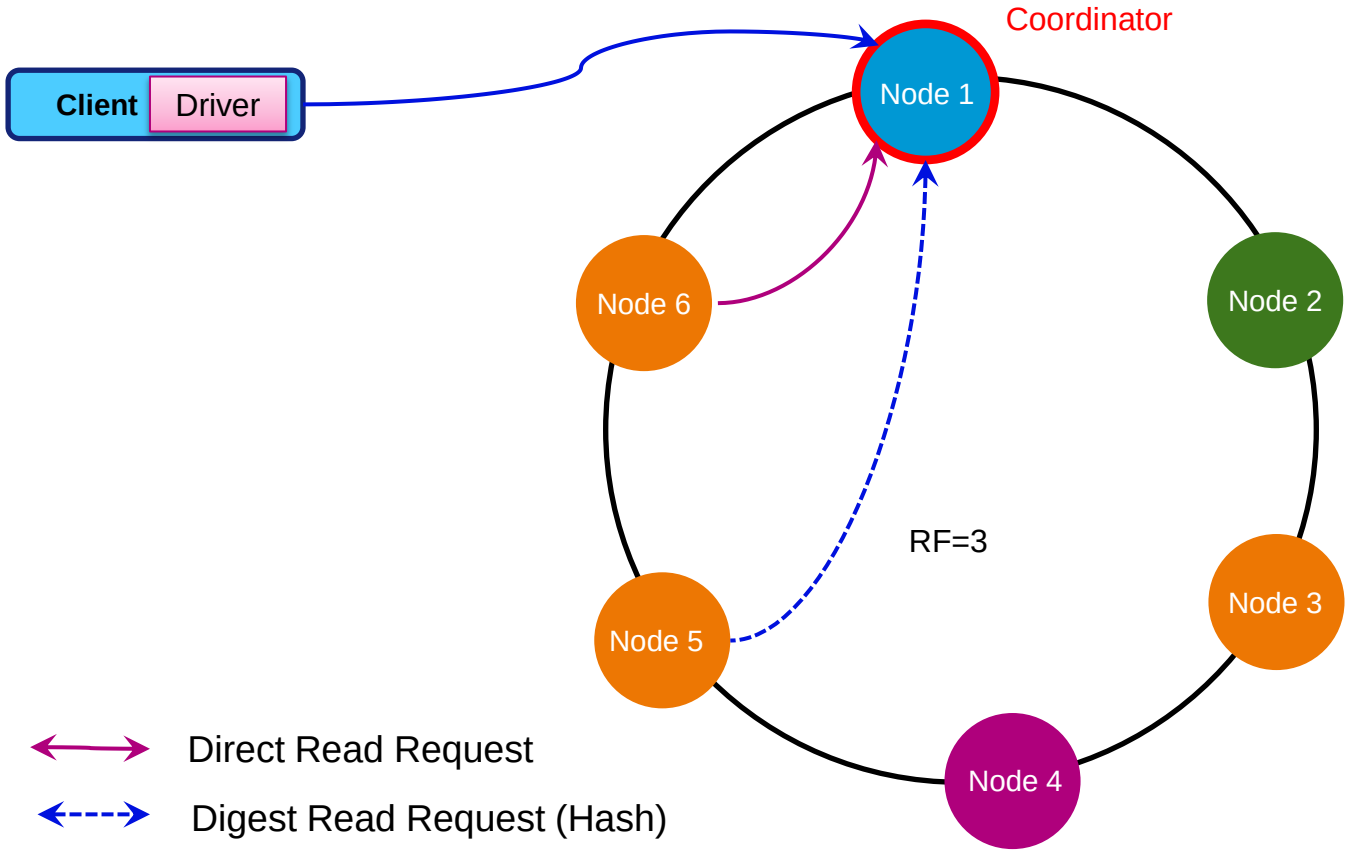
# CONSISTENCY QUORUM – READ - SINGLE DC



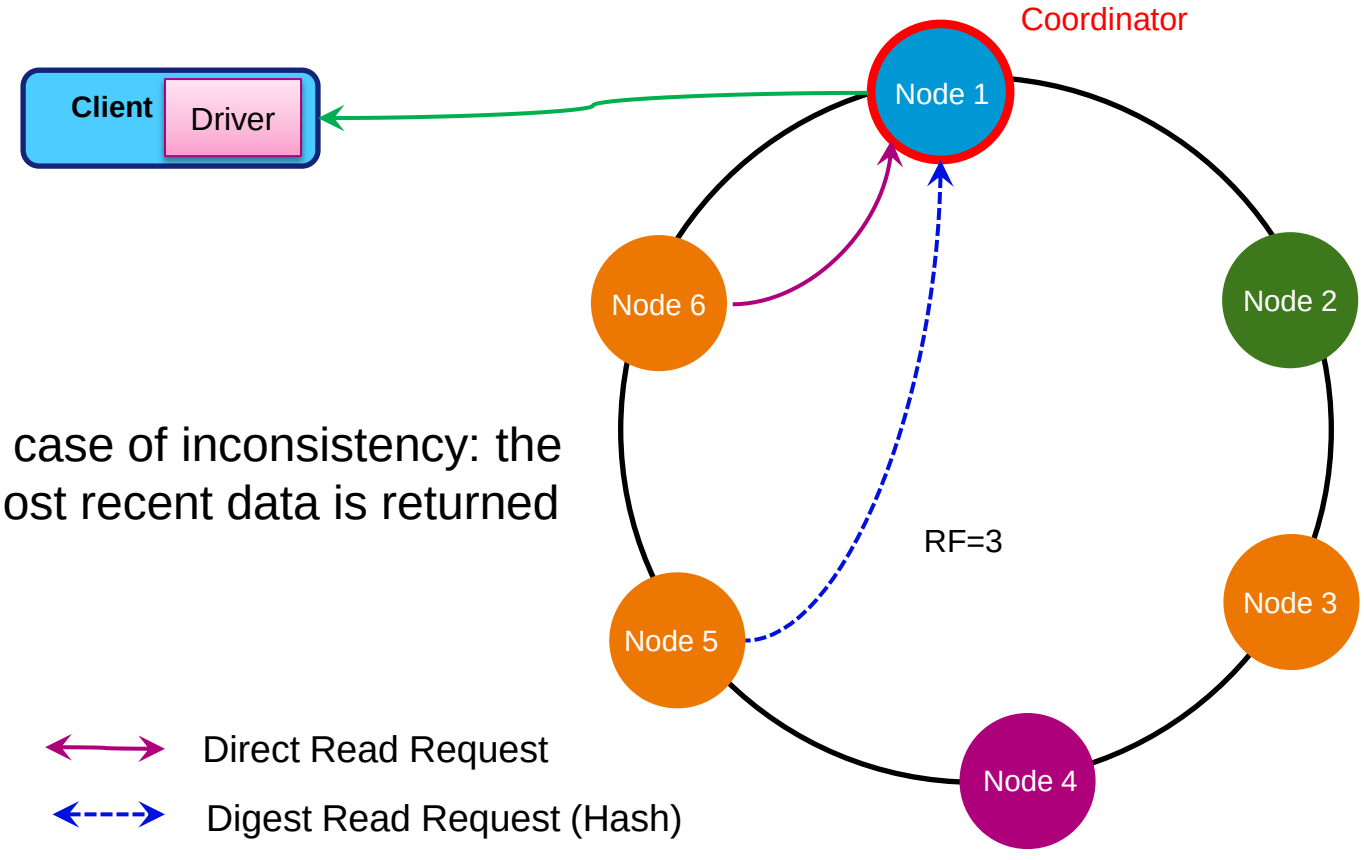
# CONSISTENCY QUORUM – READ - SINGLE DC



# CONSISTENCY QUORUM – READ - SINGLE DC

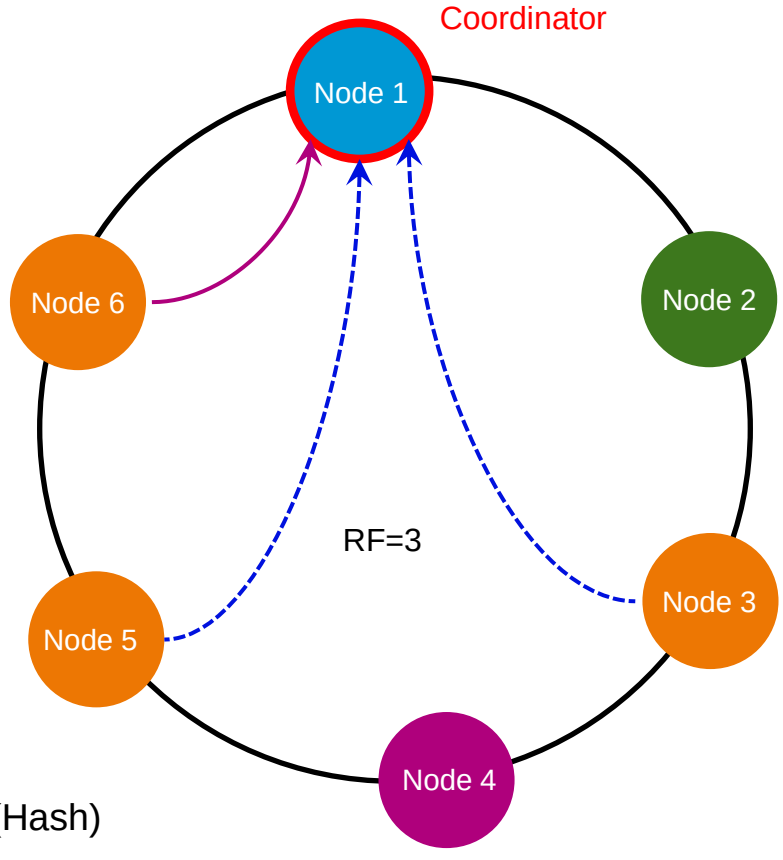


# CONSISTENCY QUORUM – READ - SINGLE DC




In case of inconsistency: the most recent data is returned

# CONSISTENCY QUORUM – READ - SINGLE DC

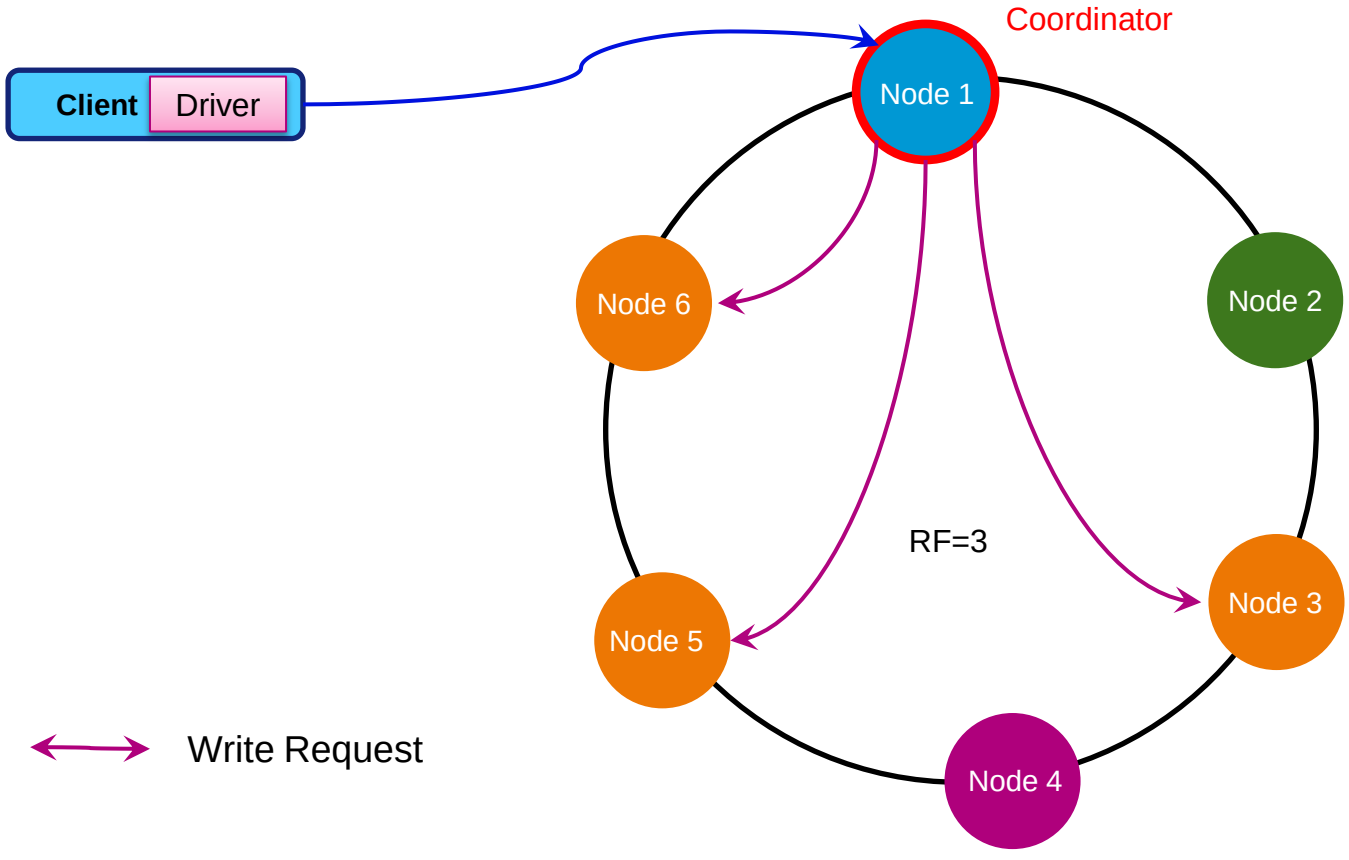




Read repair if needed

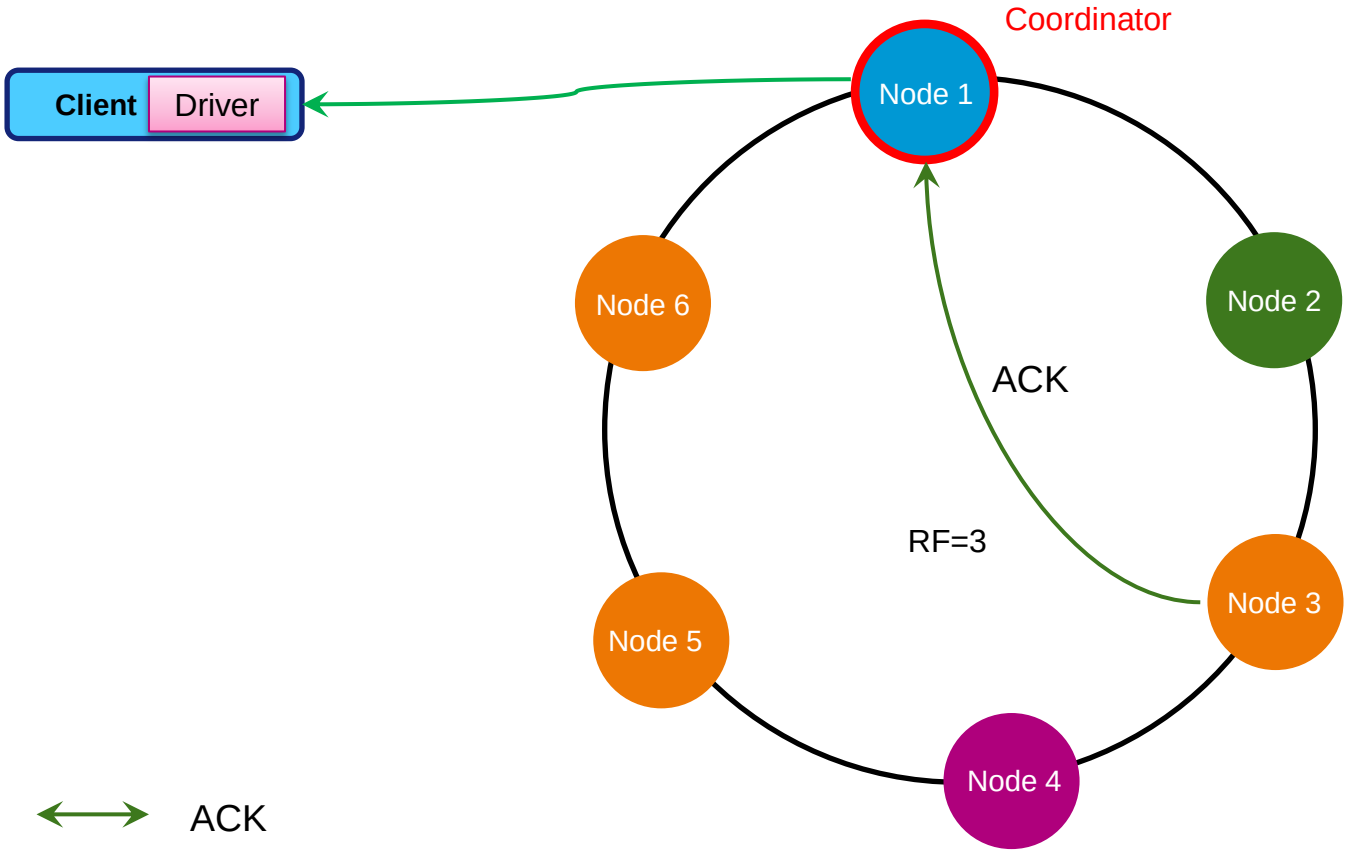
Direct Read Request

Digest Read Request (Hash)

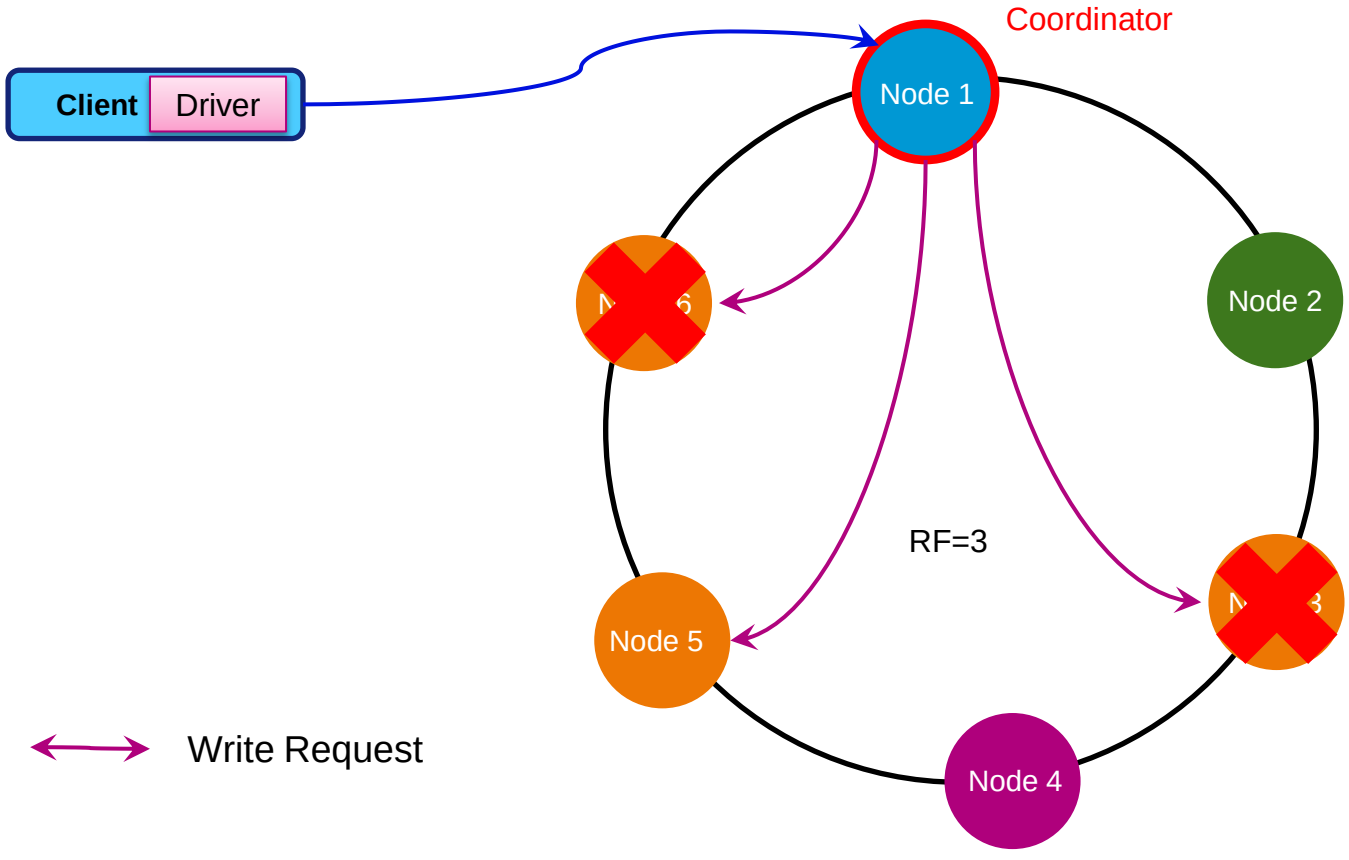
# CONSISTENCY ONE – WRITE - SINGLE DC



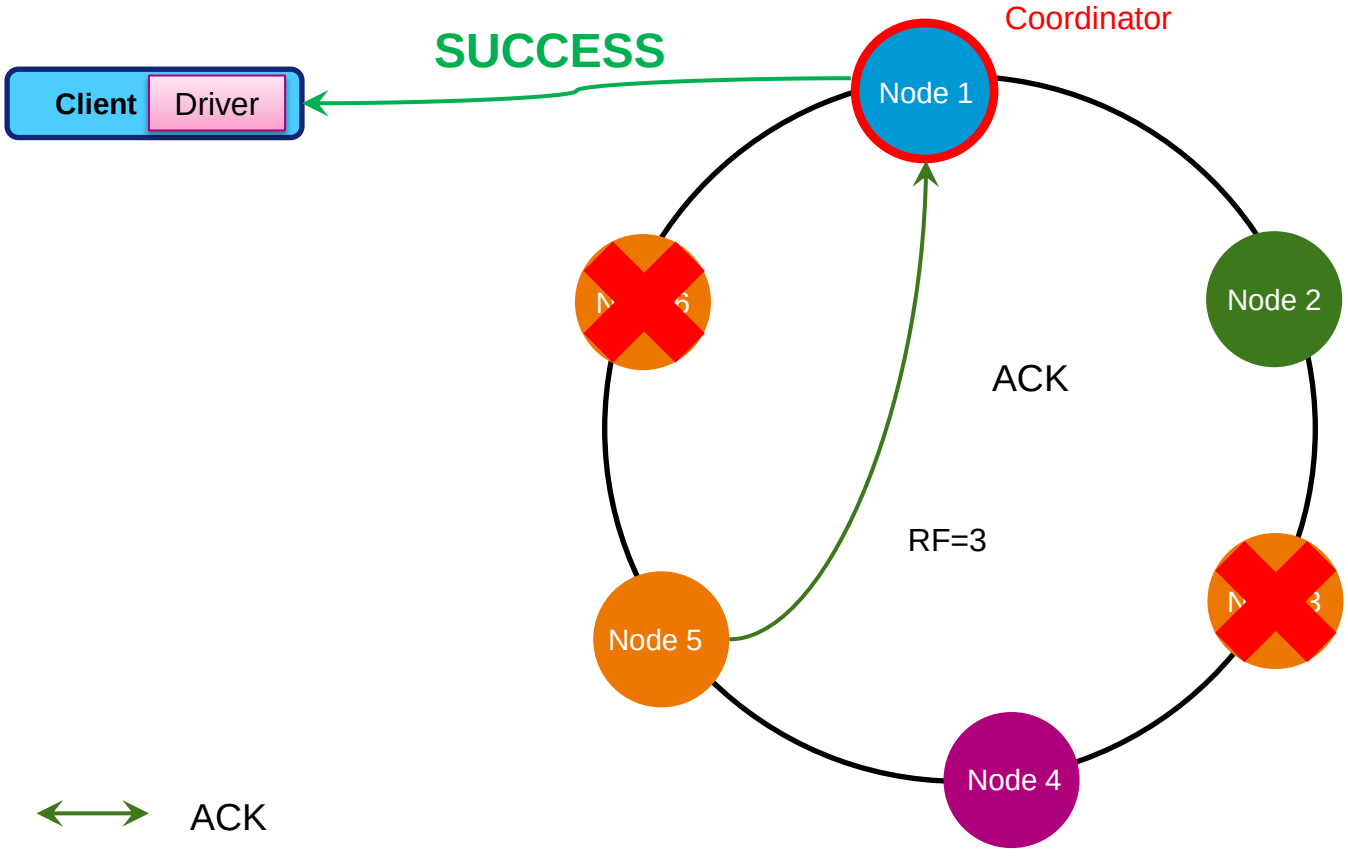
# CONSISTENCY ONE – WRITE - SINGLE DC



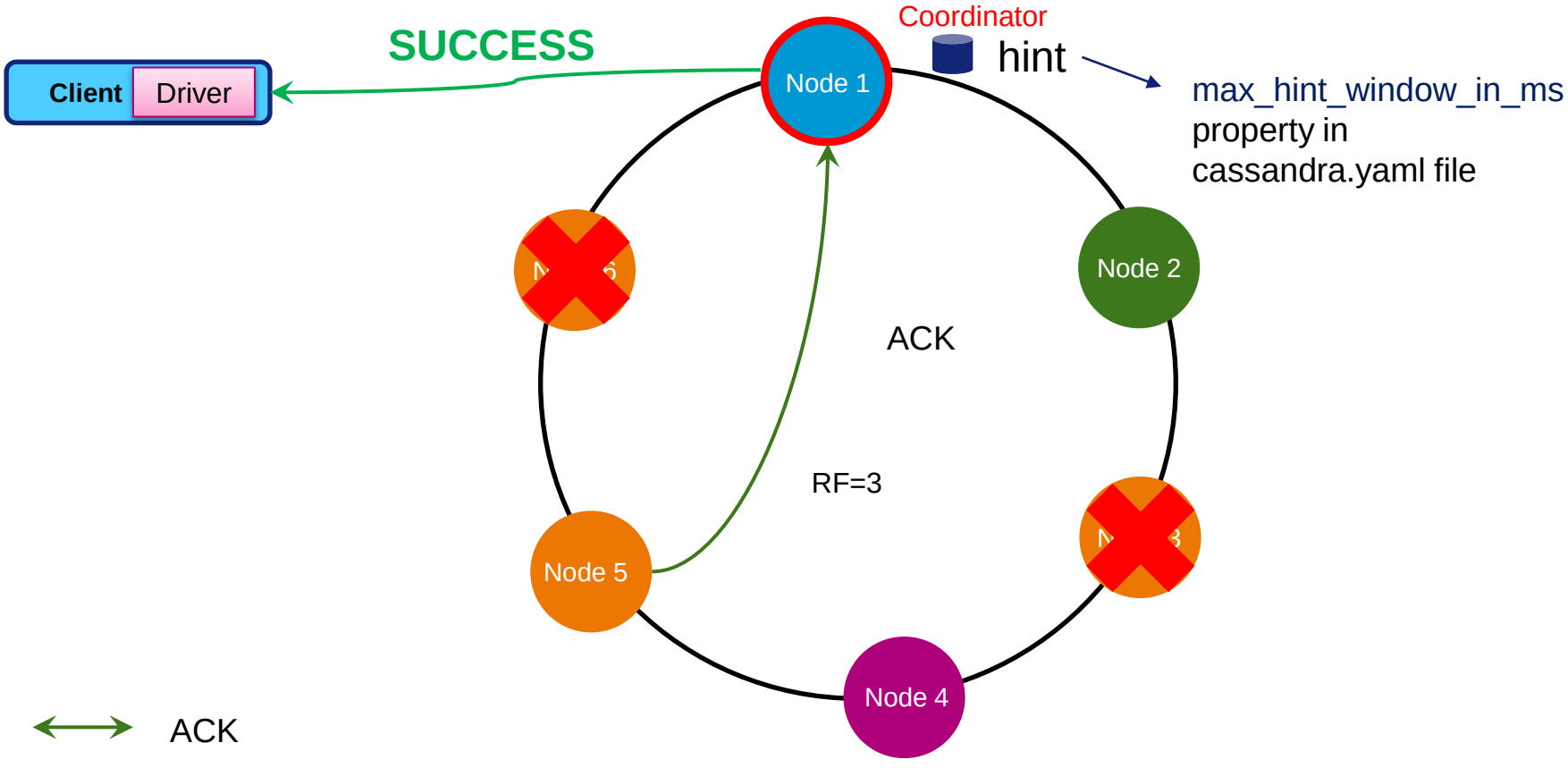
# CONSISTENCY ONE – WRITE - SINGLE DC



# CONSISTENCY ONE – WRITE - SINGLE DC

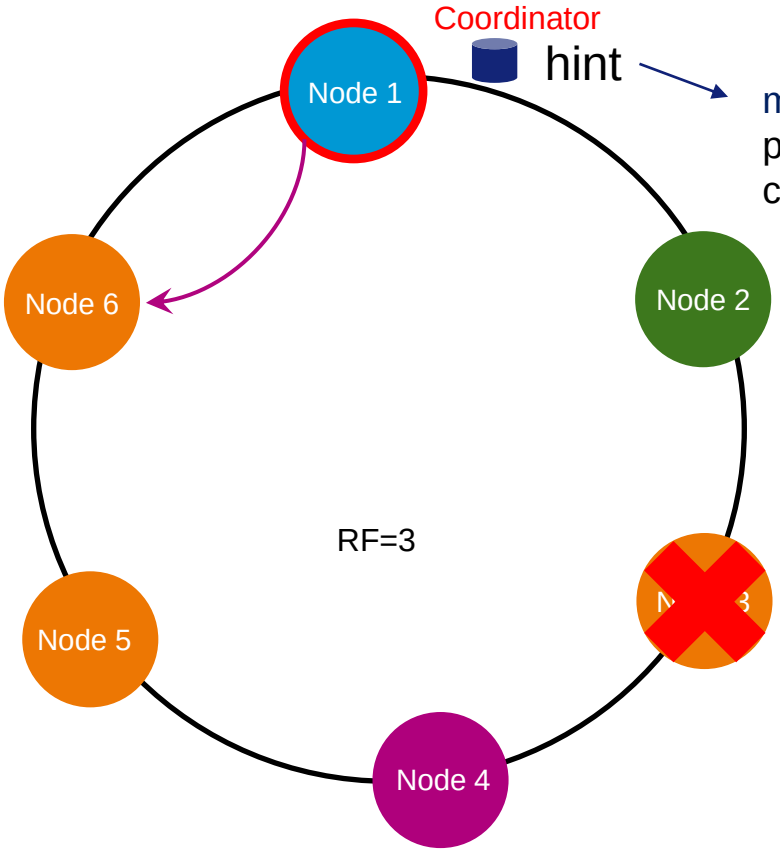


# CONSISTENCY ONE – WRITE - SINGLE DC




Hinted handoff mechanism

# CONSISTENCY ONE – WRITE - SINGLE DC

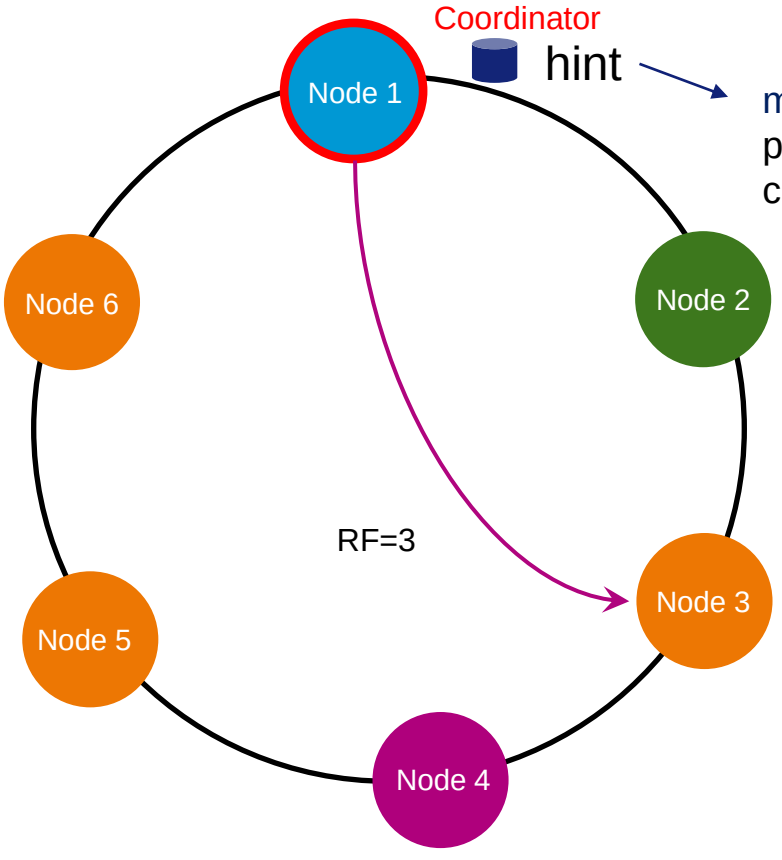




max_hint_window_in_ms property in cassandra.yaml file

Write Request

Hinted handoff mechanism

# CONSISTENCY ONE – WRITE - SINGLE DC

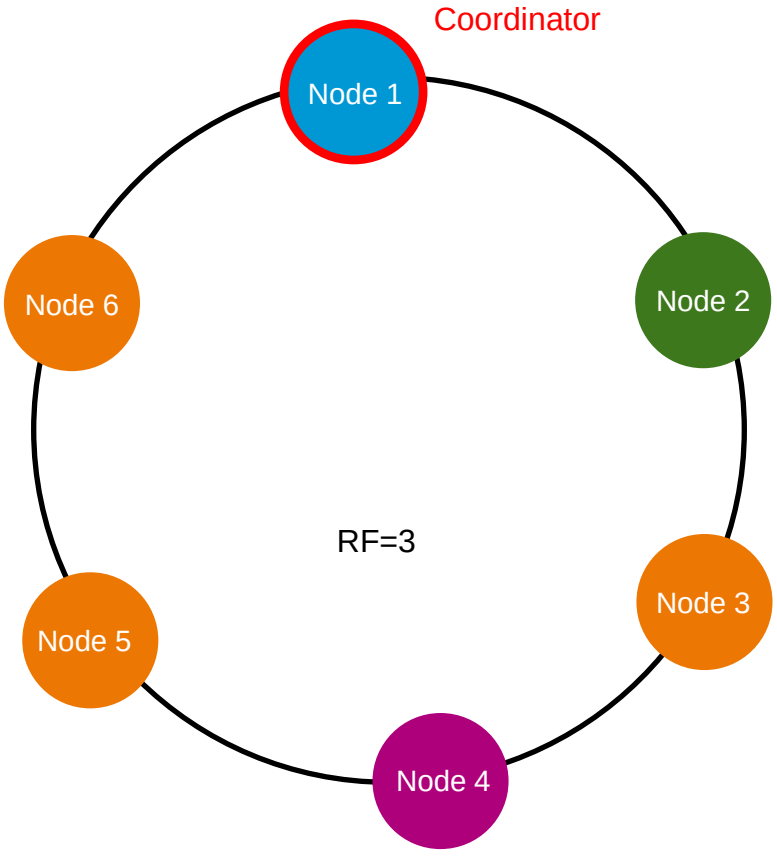




max_hint_window_in_ms property in cassandra.yaml file

Write Request

Hinted handoff mechanism

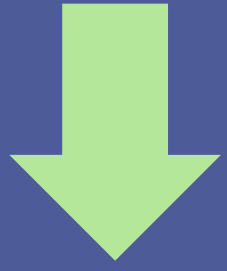
# CONSISTENCY ONE – WRITE - SINGLE DC


Hinted handoff mechanism

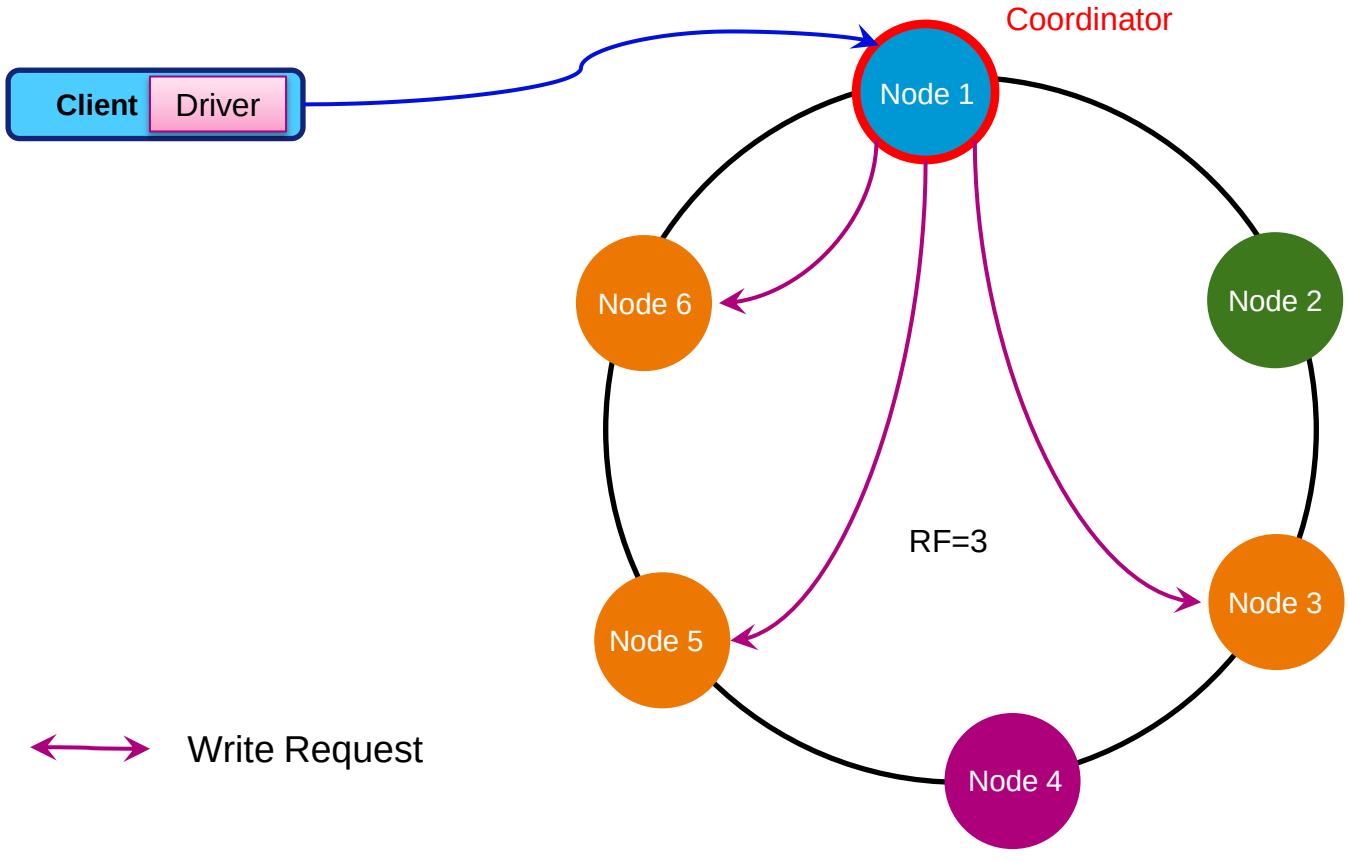
# CONSISTENCY

**if node downtime > max_hint_window_in_ms**

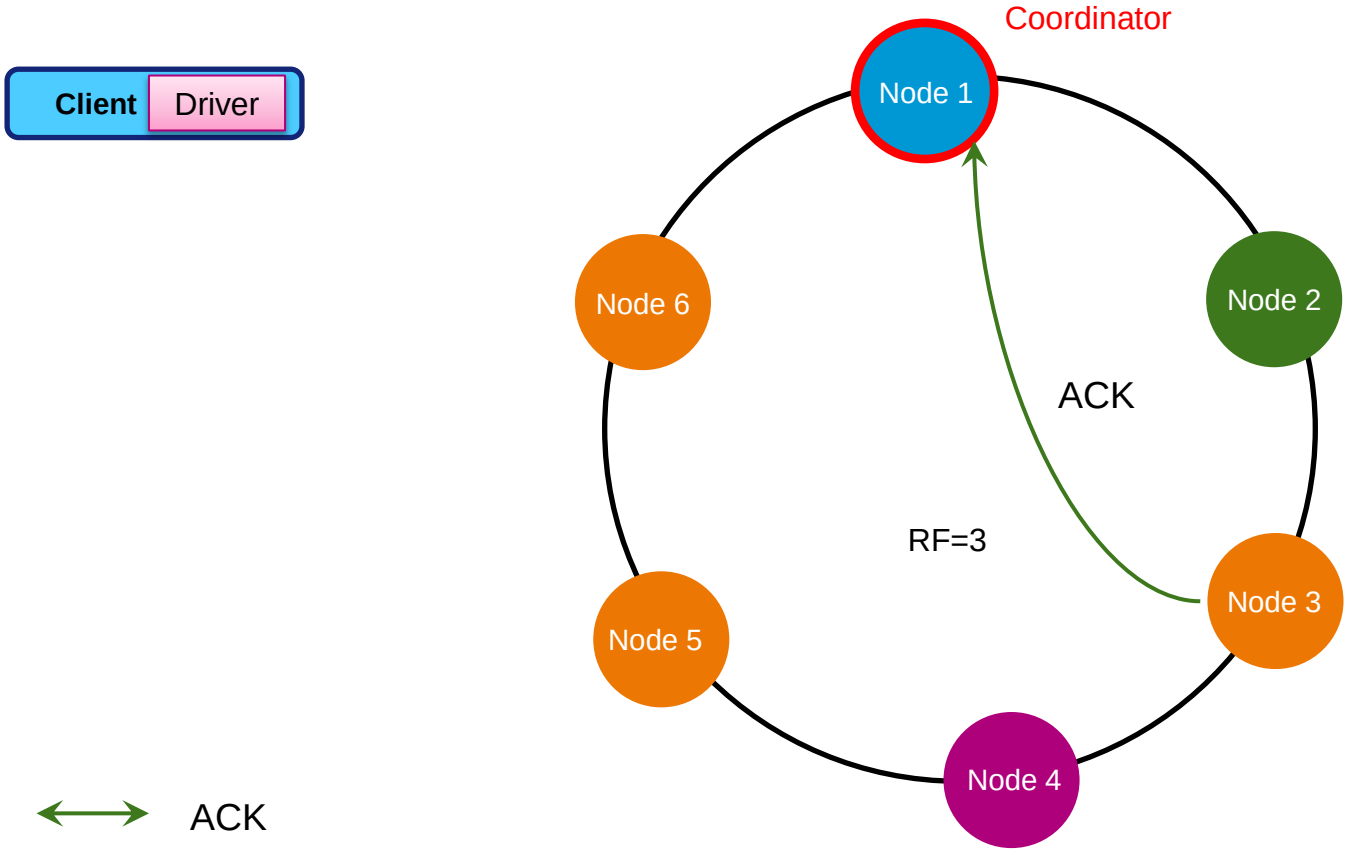


**Anti-entropy node repair**

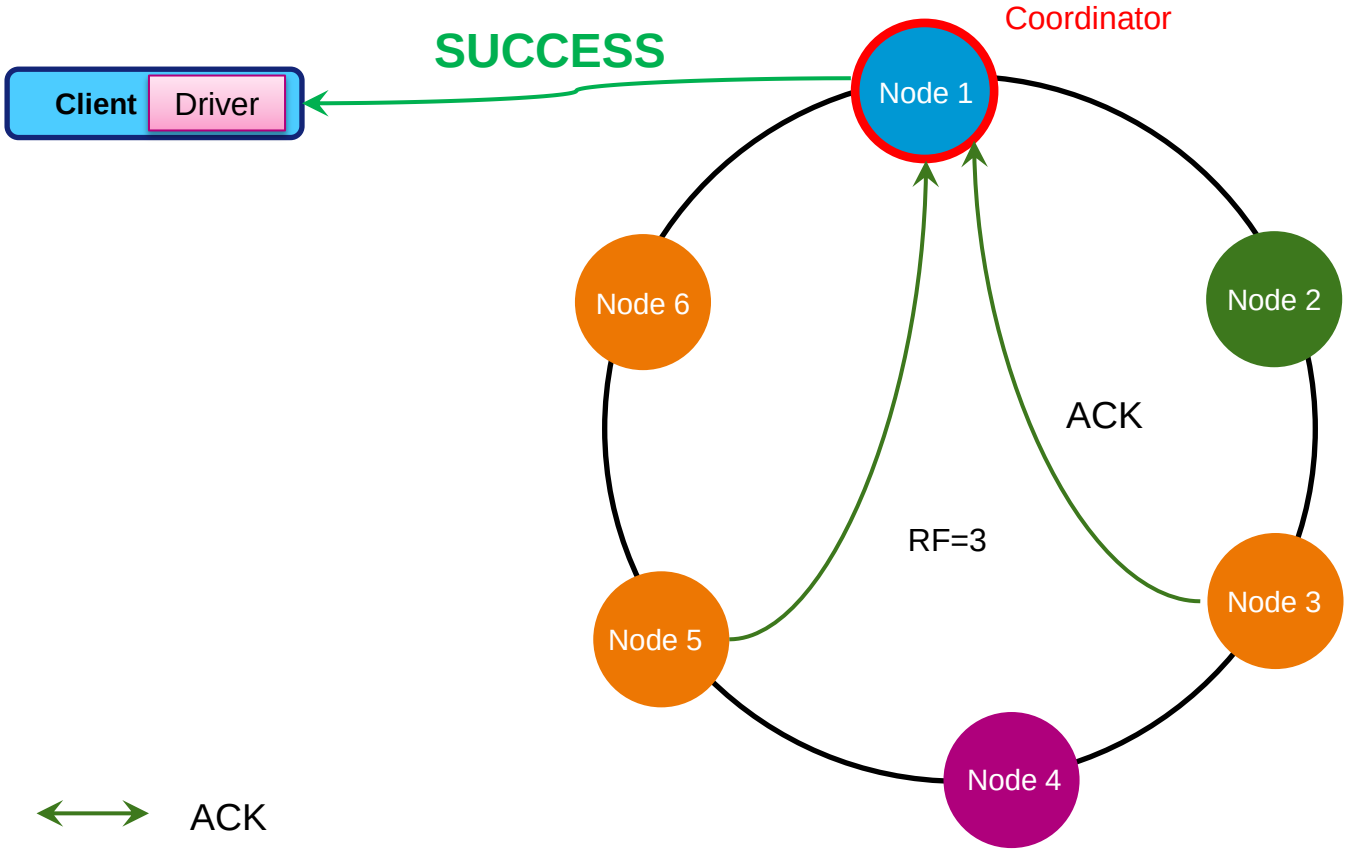
# CONSISTENCY QUORUM – WRITE - SINGLE DC



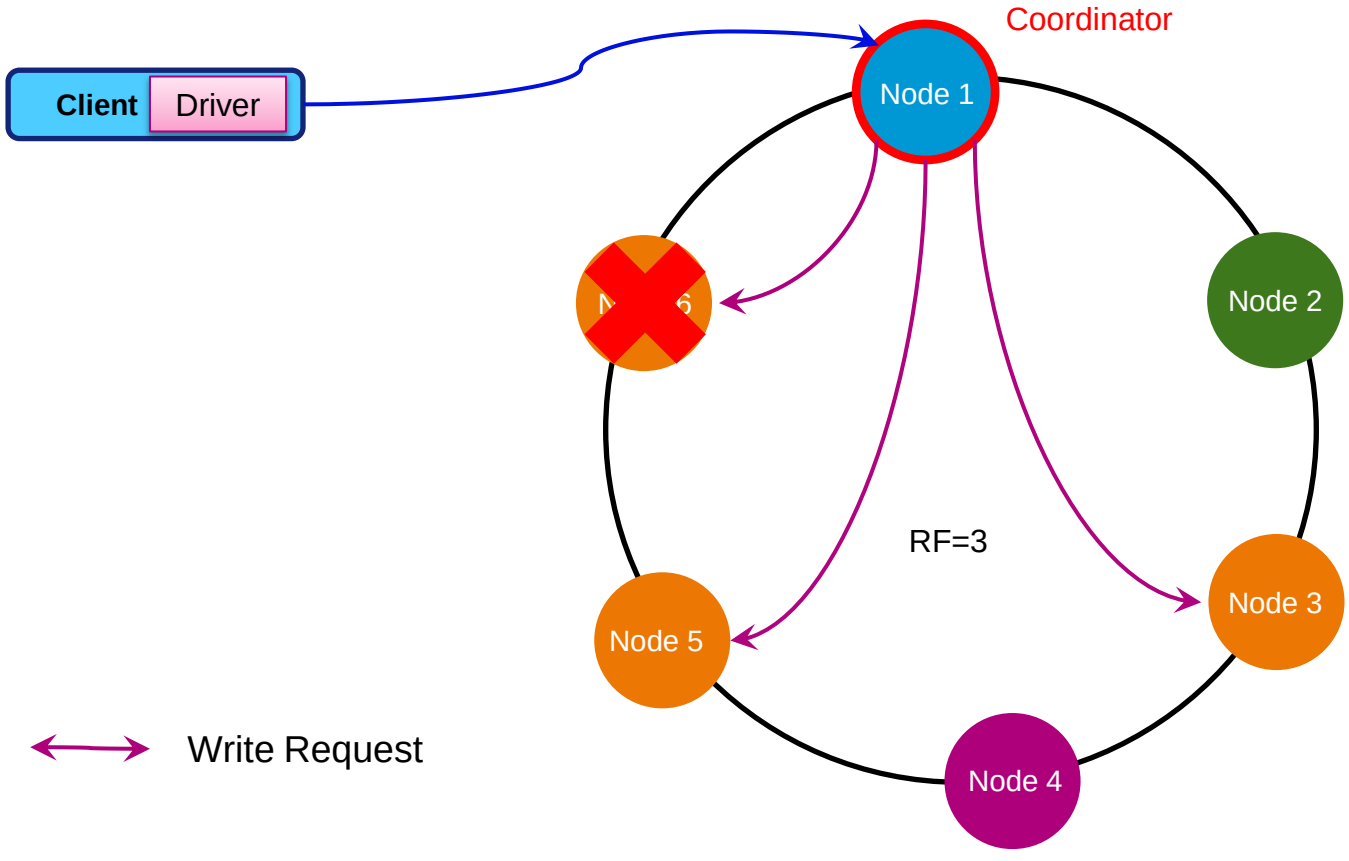
# CONSISTENCY QUORUM – WRITE - SINGLE DC



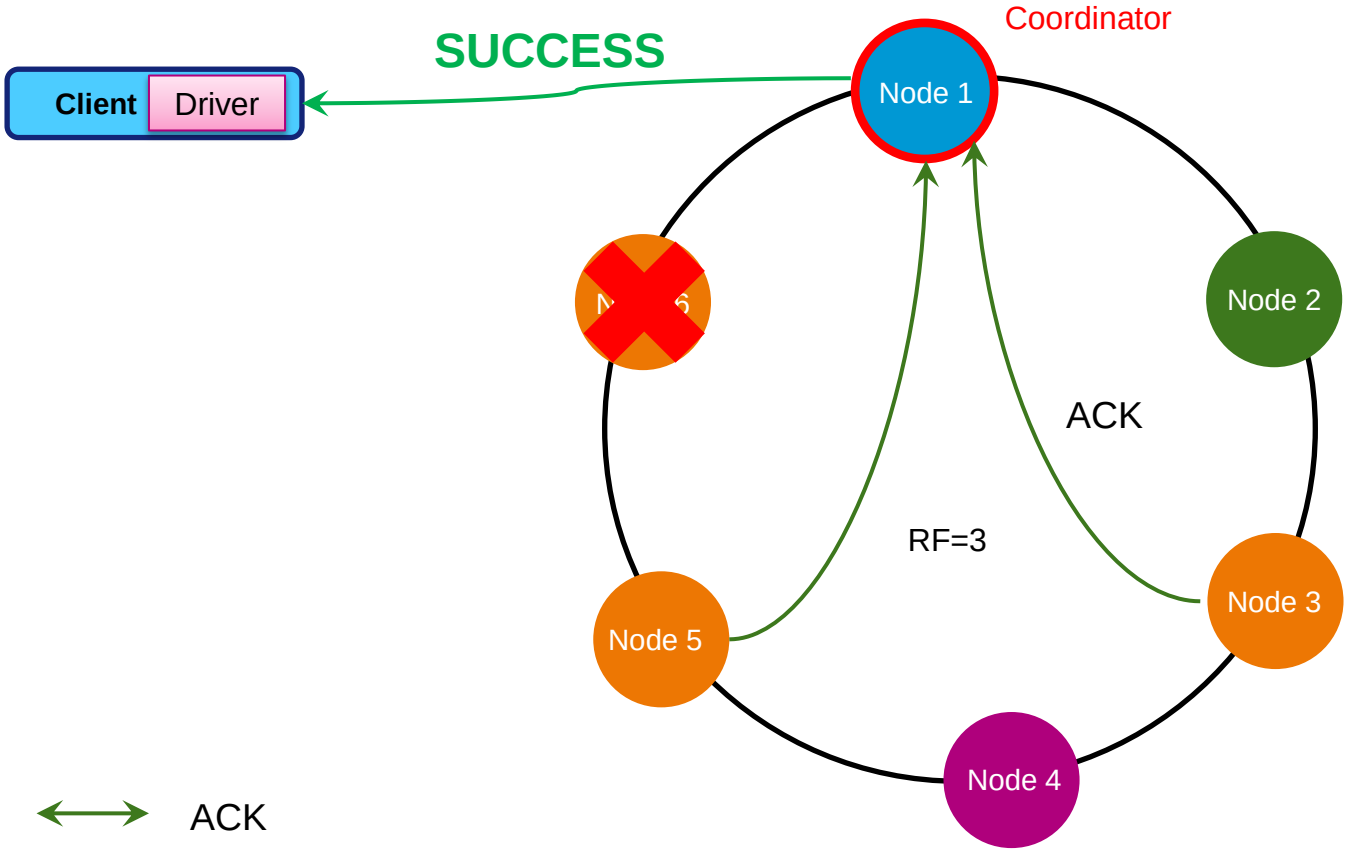
# CONSISTENCY QUORUM – WRITE - SINGLE DC



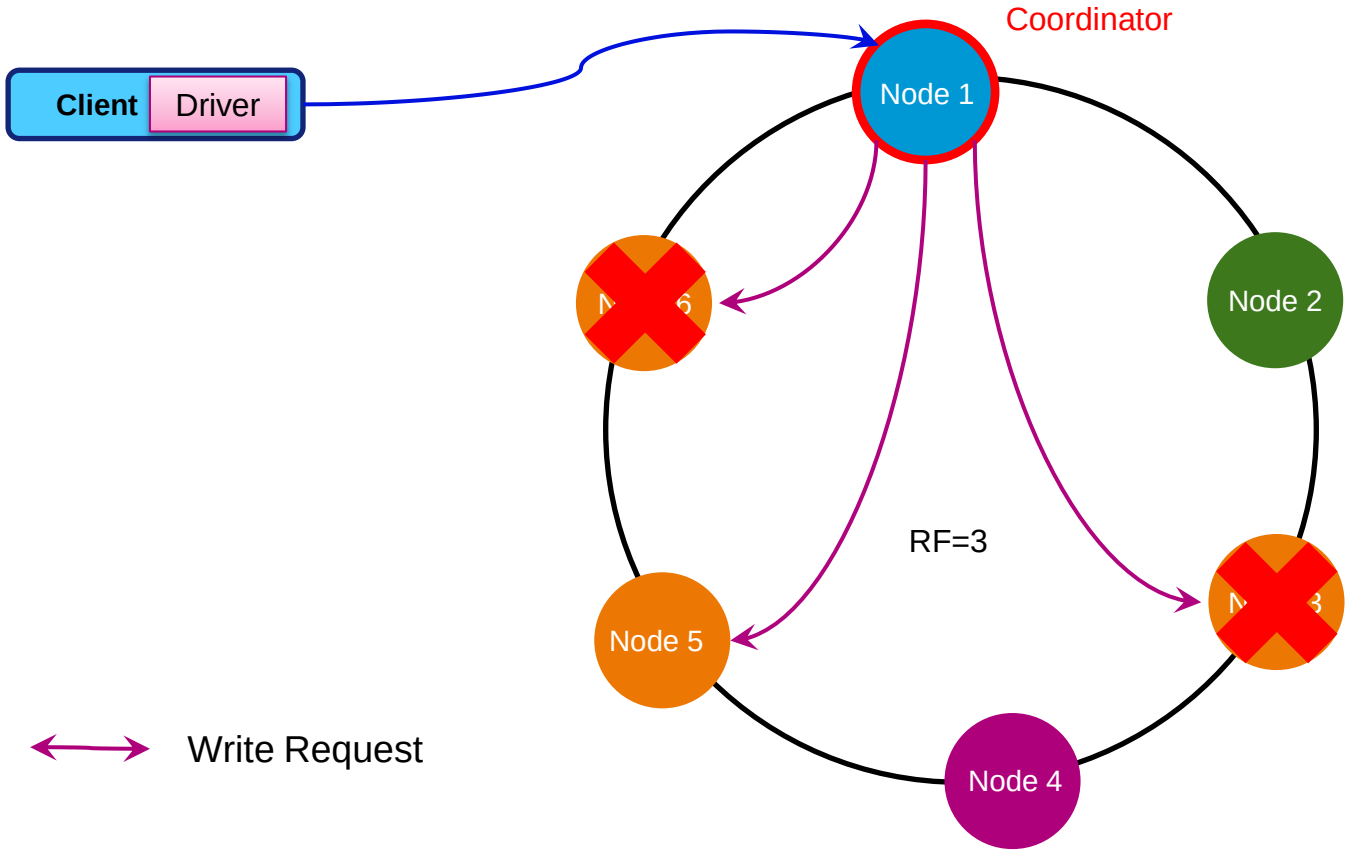
# CONSISTENCY QUORUM – WRITE - SINGLE DC



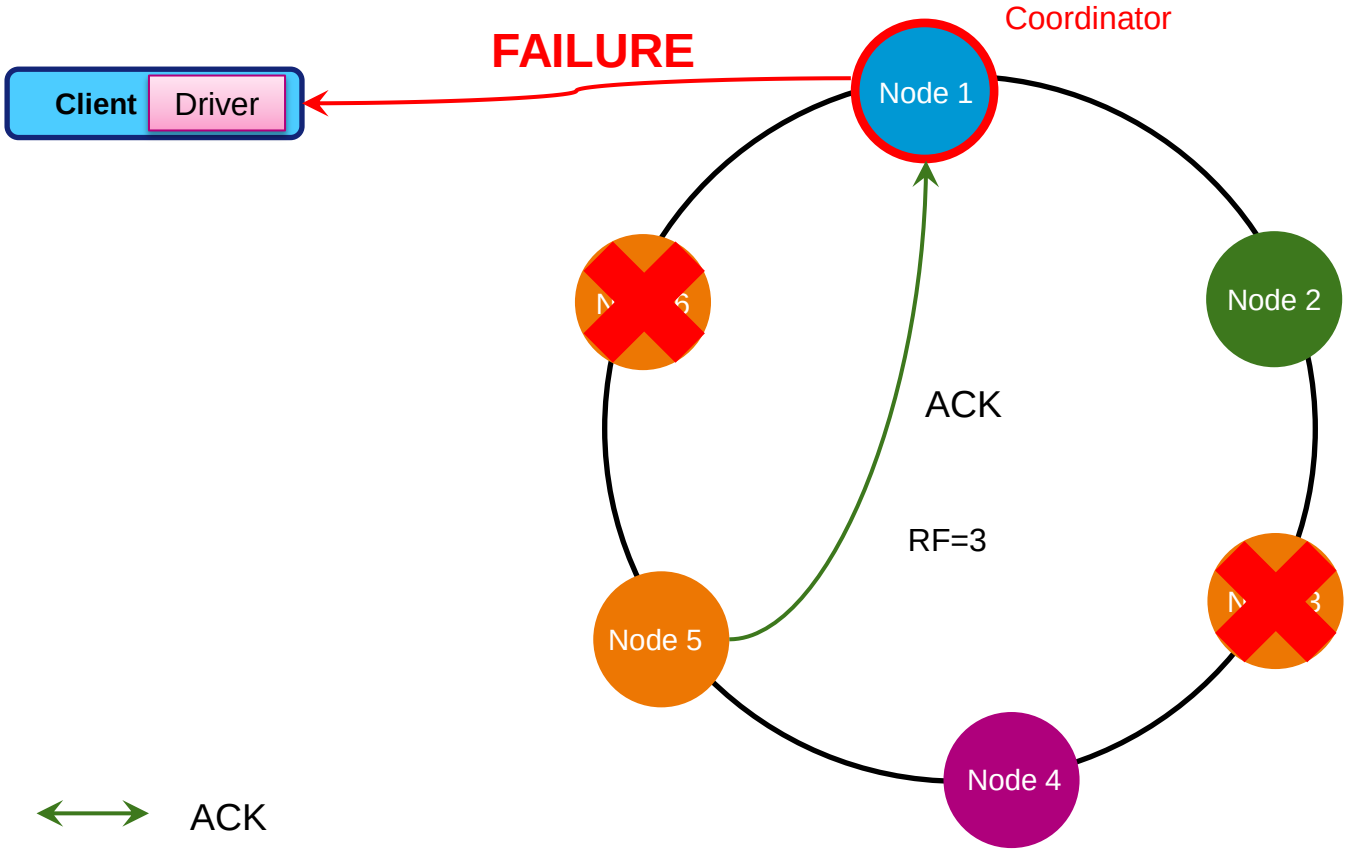
# CONSISTENCY QUORUM – WRITE - SINGLE DC



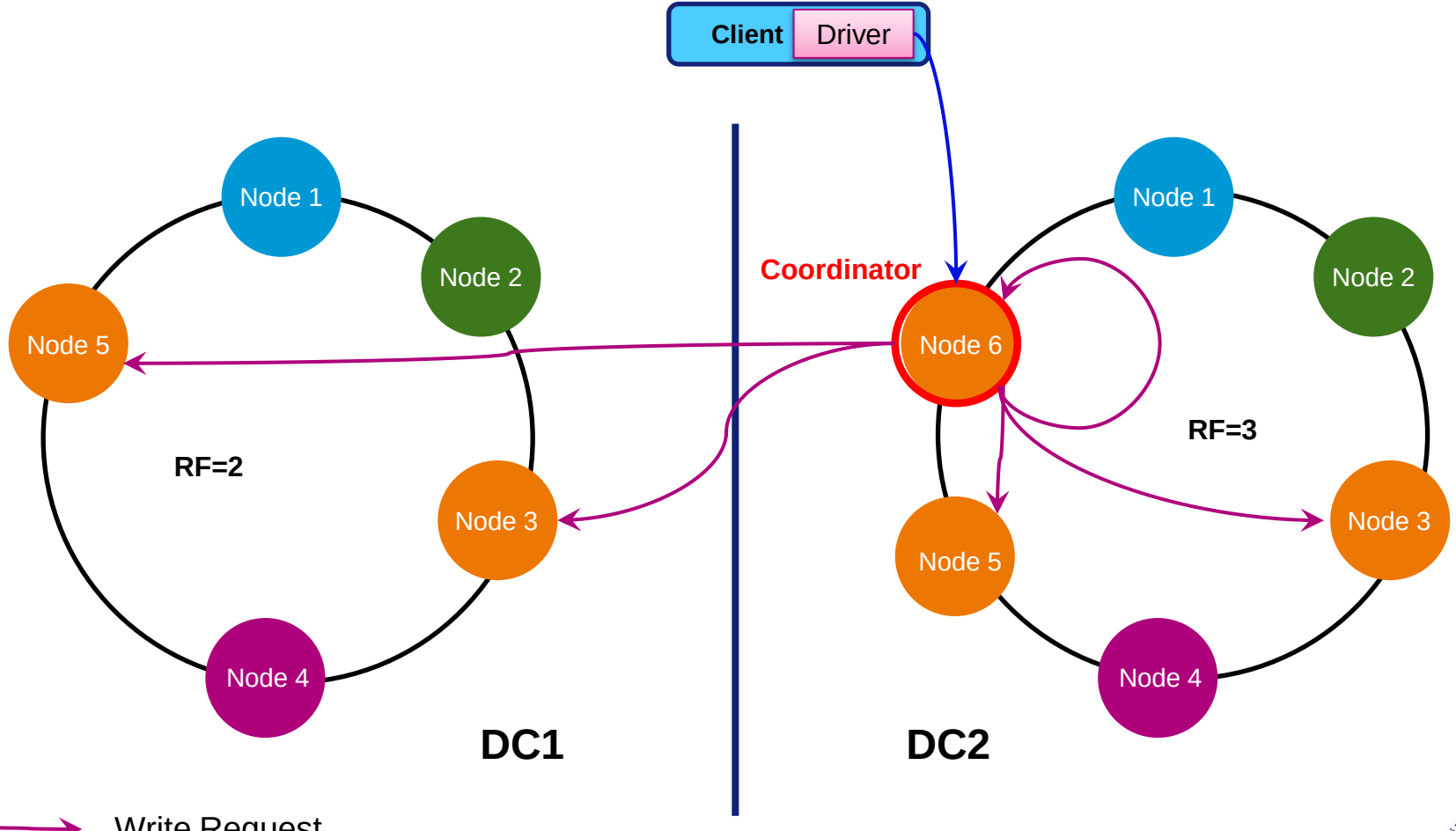
# CONSISTENCY QUORUM – WRITE - SINGLE DC



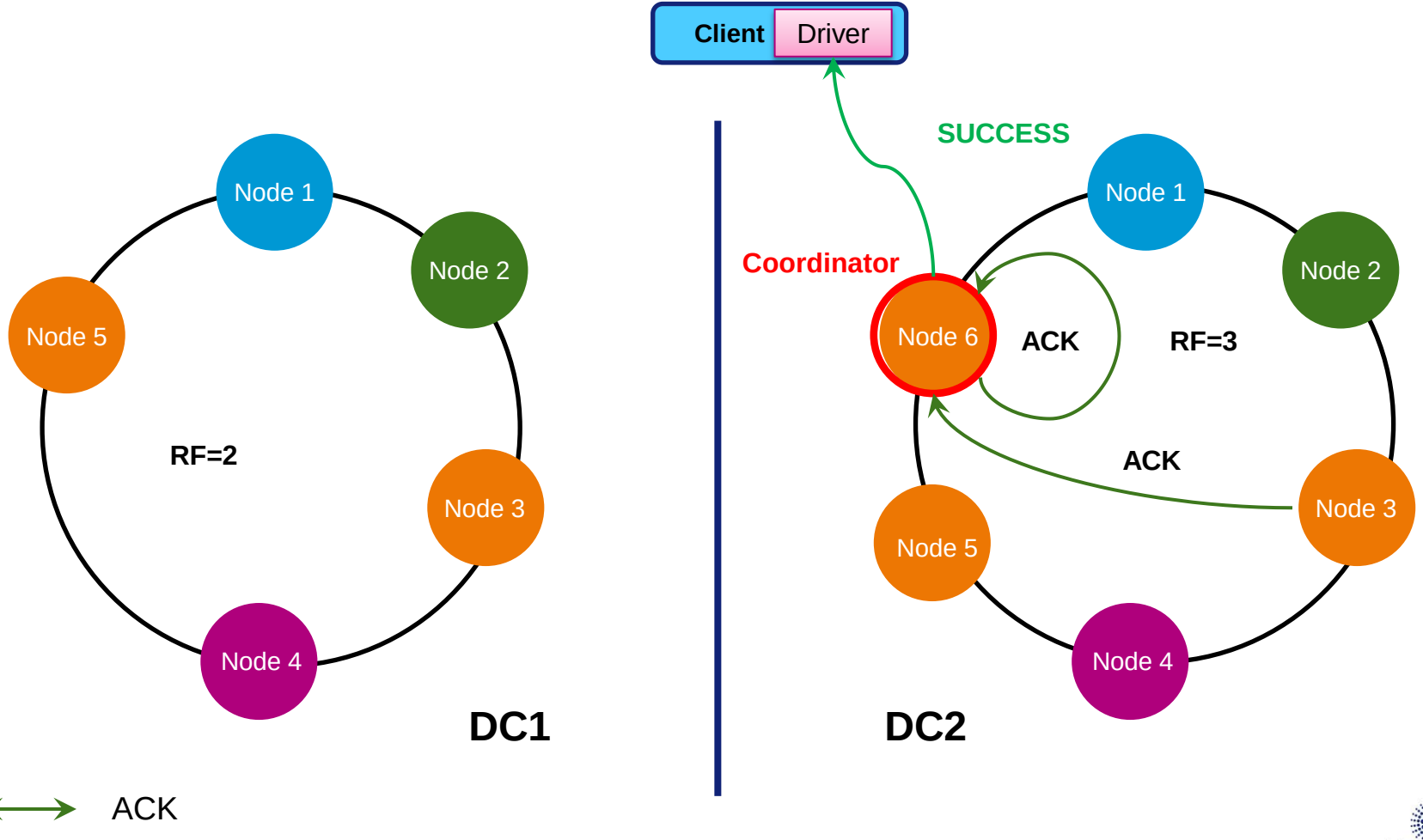
# CONSISTENCY QUORUM – WRITE - SINGLE DC



# CONSISTENCY LOCAL QUORUM – WRITE - MULTI DC



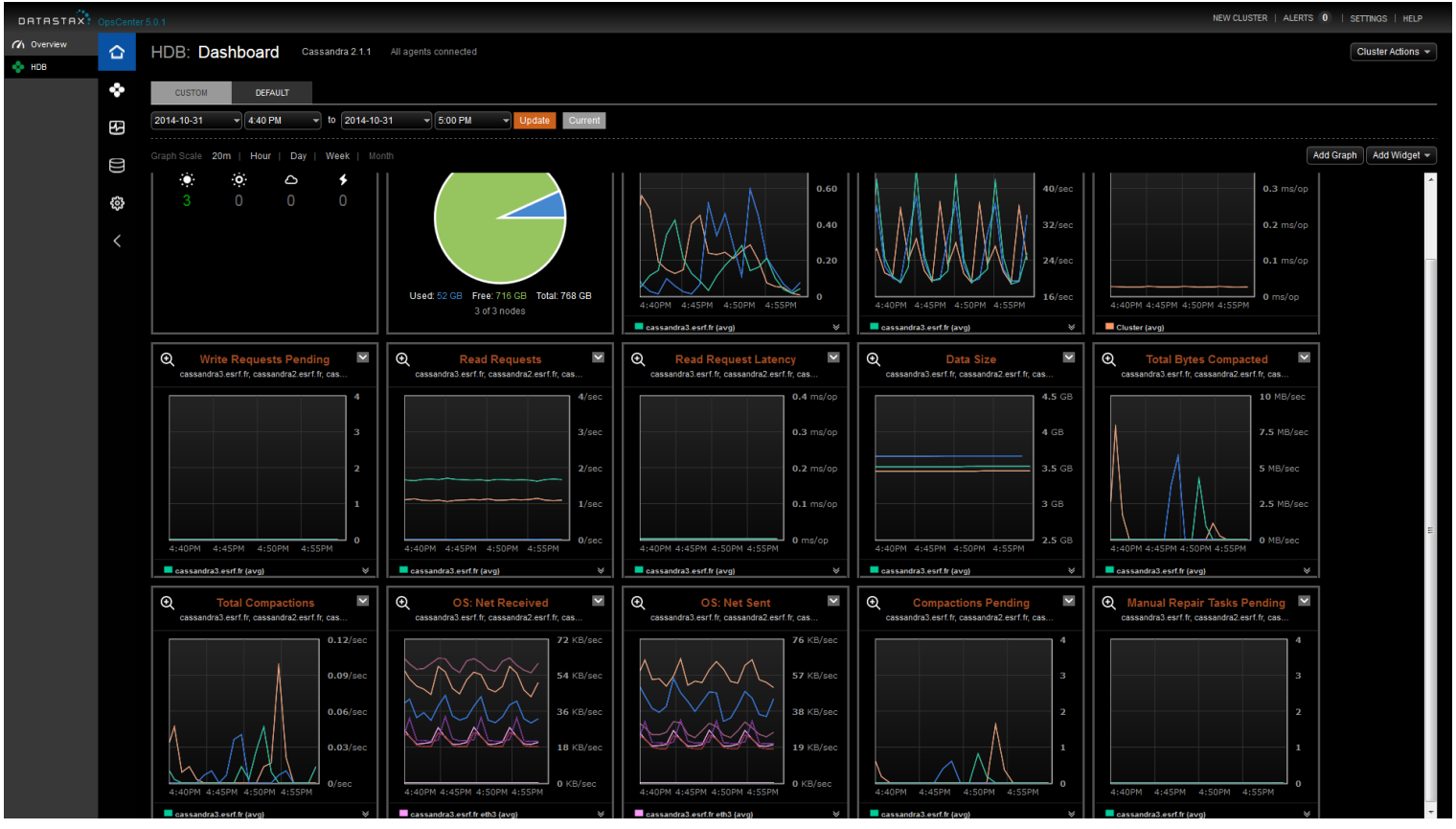
# CONSISTENCY LOCAL QUORUM – WRITE - MULTI DC



# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

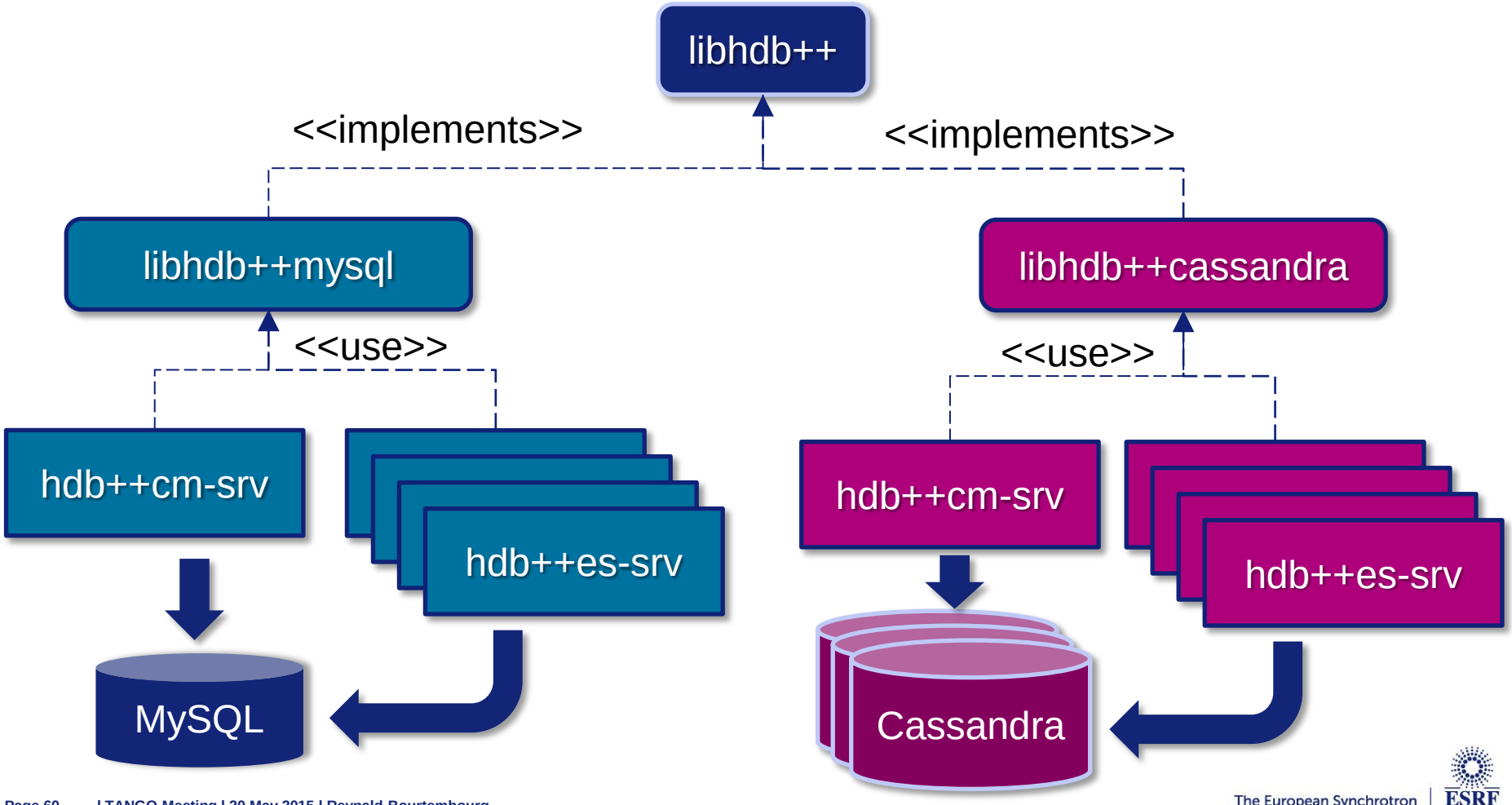
# MONITORING TOOL: OPSCENTER



# OVERVIEW

- What is Cassandra (C*)?
- Who is using C*?
- CQL
- C* architecture
- Request Coordination
- Consistency
- Monitoring tool
- HDB++

# HDB++


libhdb++
<<implements>>
<<implements>>
libhdb++mysql
libhdb++cassandra
<<use>>
<<use>>
hdb++cm-srv
hdb++es-srv
hdb++cm-srv
hdb++es-srv
MySQL
Cassandra

# CONCLUSION: C* PROS

- High Availability
  - SW upgrade with no downtime
  - HW failure
- Linear Scalability
  - Need more performances? => Add nodes
- Big community with industrial support

# CONCLUSION: C* PROS

- Can use Apache Spark for analytics (distributed processing)
- List, Set, Map data types (tuples and user defined types soon)
- Tries not to let you do actions which do not perform well
- Backups = snapshot = hard links => very fast (+Replication)
- Difficult to lose data
- Good fit for time series data

# CONCLUSION: C* CONS

- **Requires more total disk space and machines**
- **sstable (C* data files) format can change from one version to another**
- **No easy way to come back to a previous version once the sstables have been converted to a newer version**
- **Cannot rename keyspaces or tables easily** (not foreseen in CQL)
- **Tedious to modify existing partitions** (Needs to duplicate the data at some point in the process)

# CONCLUSION: C* CONS

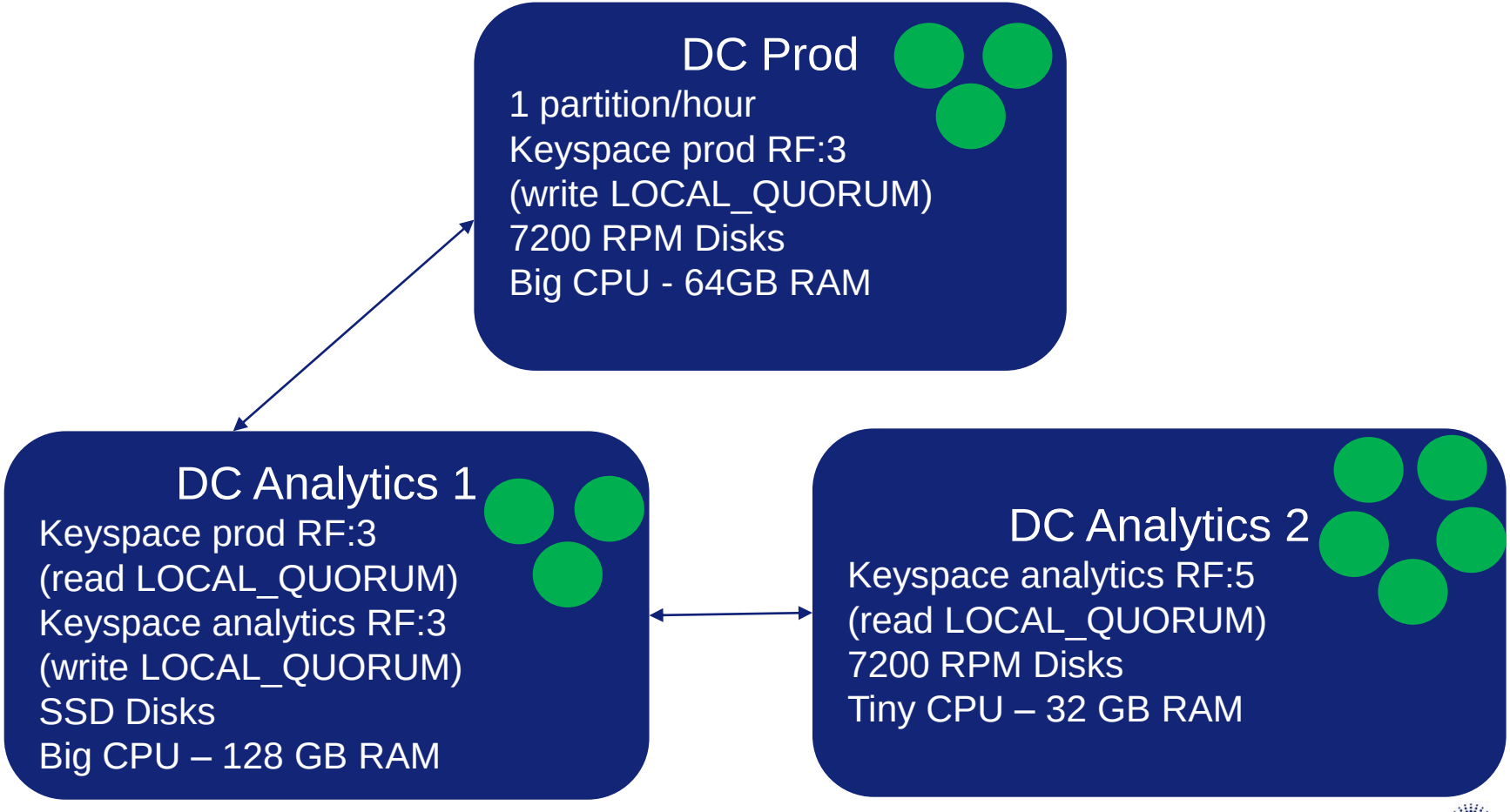
- **Different way of modeling**
- **Not designed for huge read requests**
- **Can be tricky to tune to avoid long GC pauses**
- **Maintenance: Need to run *nodetool repair* regularly if some data are deleted to avoid resurrections** (CPU intensive operation)
- **Can take quite some time to redeem disk space after deletion in some specific cases.**

# THE END

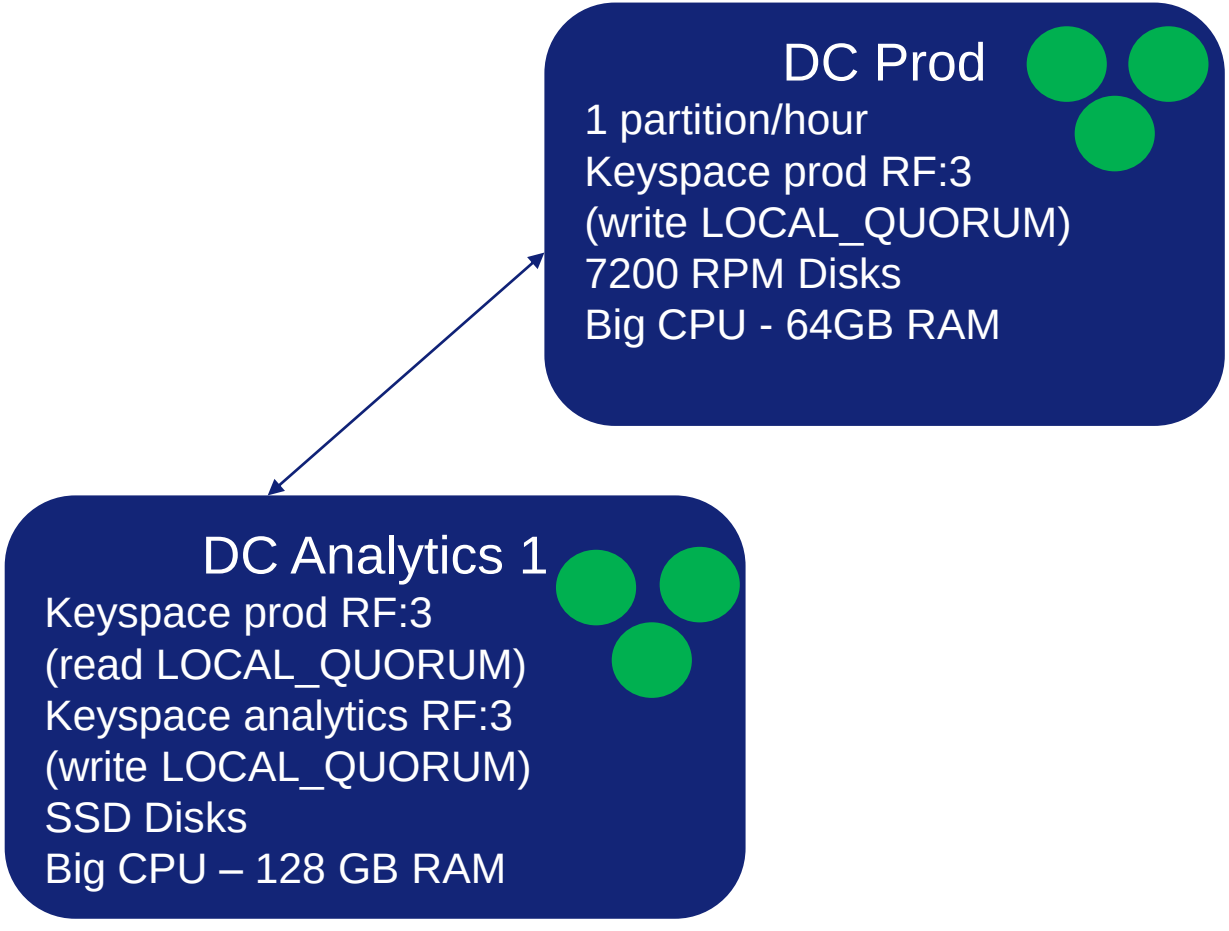
# USEFUL LINKS

- http://cassandra.apache.org
- Planet Cassandra (http://planetcassandra.org)
- Datastax academy (https://academy.datastax.com)
- Cassandra Java Driver getting started (https://academy.datastax.com/demos/cassandra-java-driver-getting-started)
- Cassandra C++ Driver: https://github.com/datastax/cpp-driver
- Datastax documentation (http://www.datastax.com/docs)
- Users mailing list: user-subscribe@cassandra.apache.org
- #Cassandra channel on IRC (http://webchat.freenode.net/?channels=#Cassandra)

# CASSANDRA FUTURE DEPLOYMENT



# CASSANDRA FUTURE DEPLOYMENT



# CASSANDRA'S NODE-BASED ARCHITECTURE

# BASIC WRITE PATH CONCEPT

# BASIC READ PATH CONCEPT