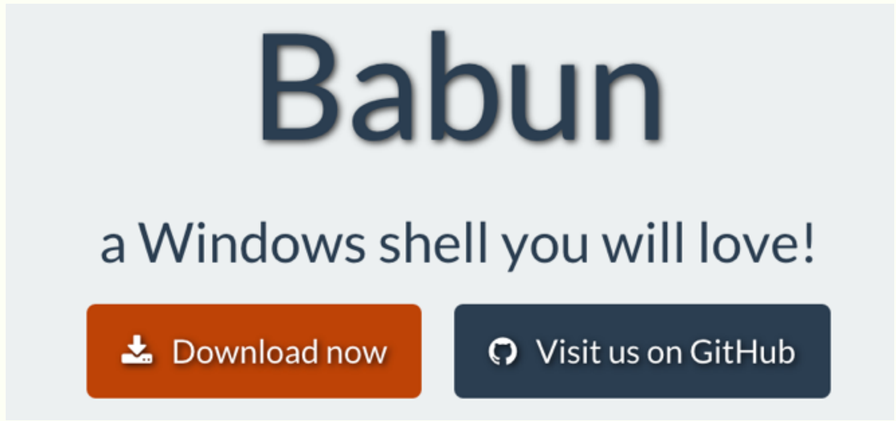
NOSQL DATABASES YOU HAVE TO KNOW

@TOM BUJOK

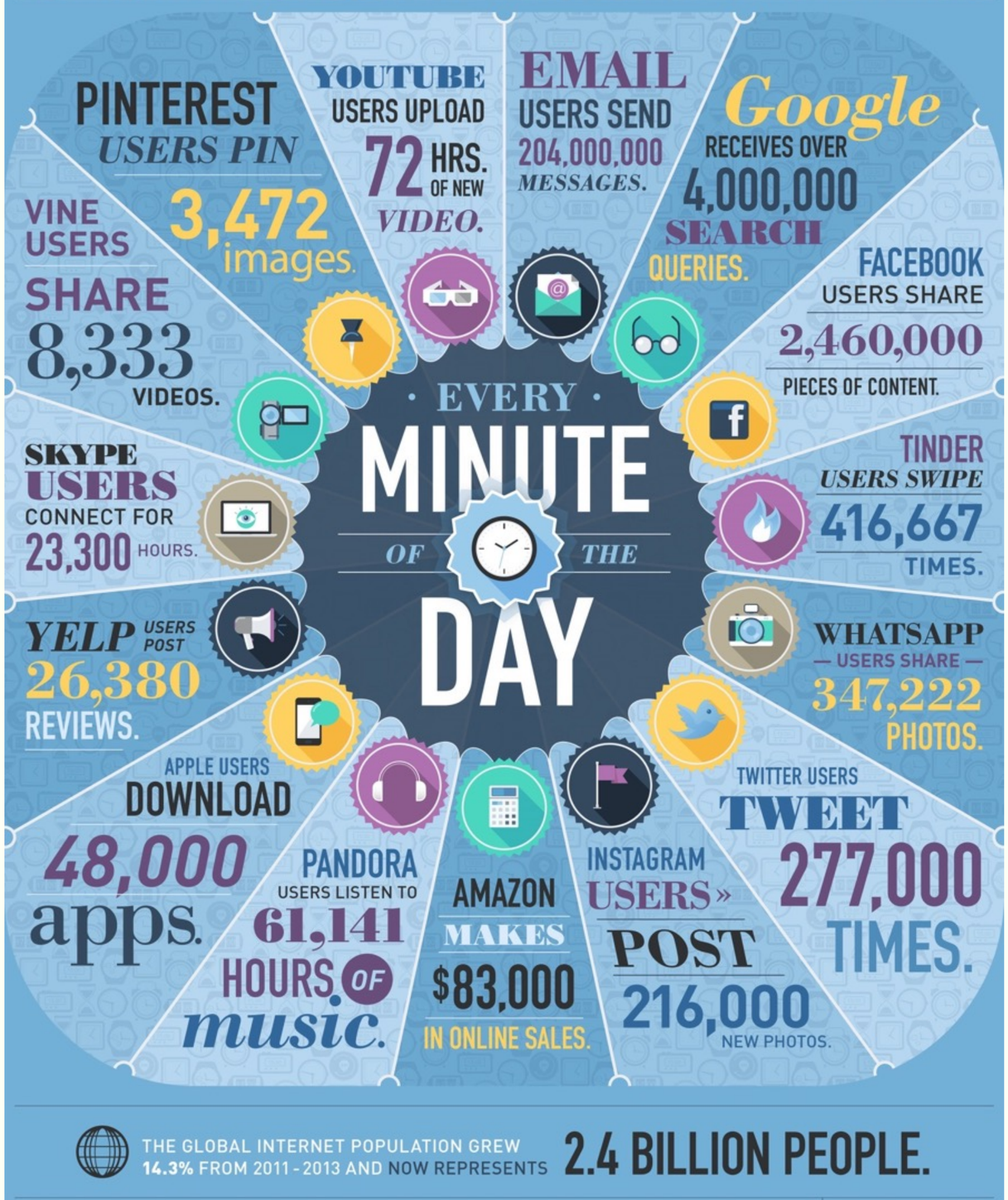
ABOUT ME

TOMEK BUJOK
HAZELCAST
@TOMBUJOK
WWW.REFICIO.ORG

hazelcast



(NOT) YET ANOTHER NOSQL TALK



https://www.domo.com/learn/data-never-sleeps-2

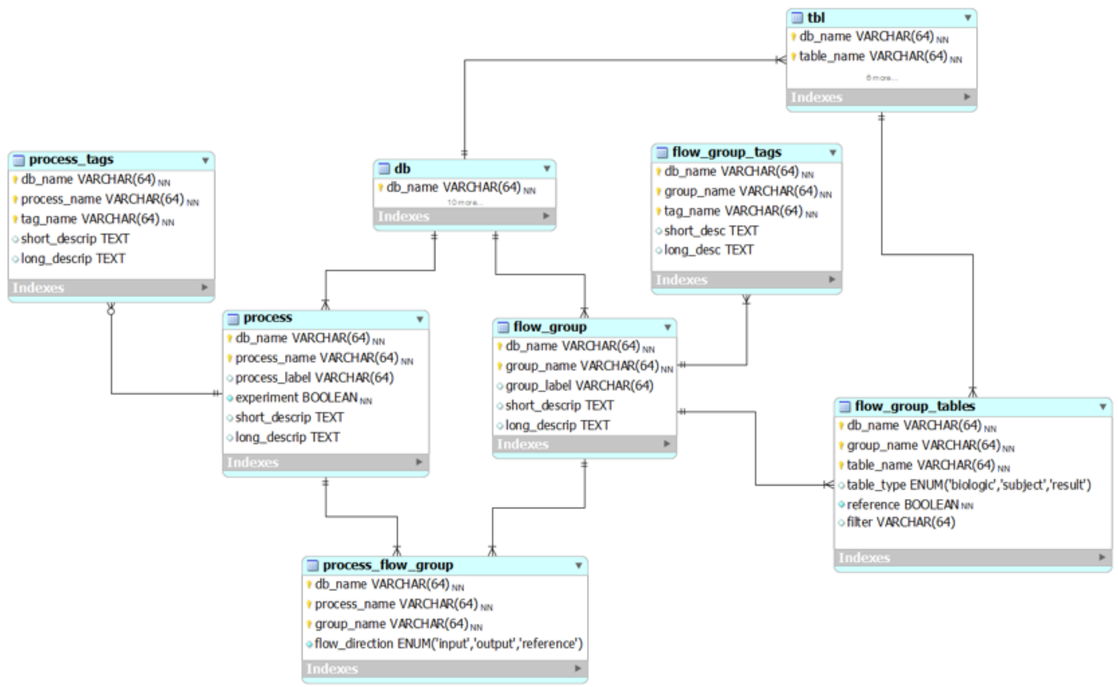
FACEBOOK SCALE

- 1+ BILLION DAILY ACTIVE USERS (04.2015)
- 600 TB DAILY ASYNCHRONOUS REPLICATION BETWEEN DATA CENTERS (04.2014)
- 300+ PB IN TOTAL (04.2014)
- BILLIONS OF QUERIES PER SECOND (09.2014)

http://www.webpronews.com/936-million-users-get-on-facebook-at-least-once-a-day-2015-04 https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/ https://research.facebook.com/blog/1522692927972019/facebook-s-top-open-data-problems/

CAN RDBMS HANDLE IT?

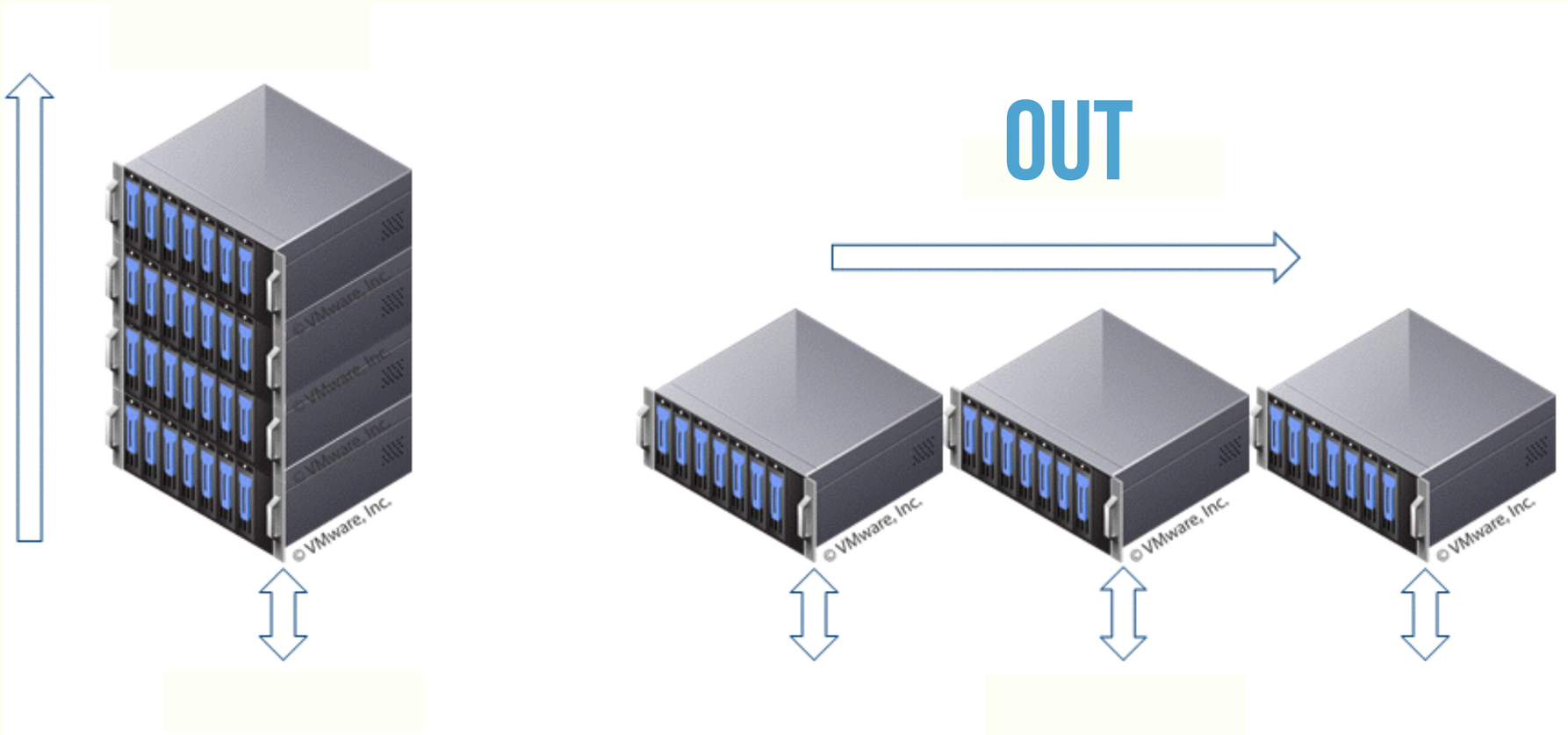
- RELATIONAL MODEL
- ACID
  - ATOMICITY - ALL OR NOTHING
  - CONSISTENCY - ALWAYS VALID STATE
  - ISOLATION - CONCURRENT TX -> STATE AS IF TX EXECUTED SERIALLY
  - DURABILITY - NO DATA LOSS AFTER TX COMMITED



http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode

RDBMS “PROBLEMS”?

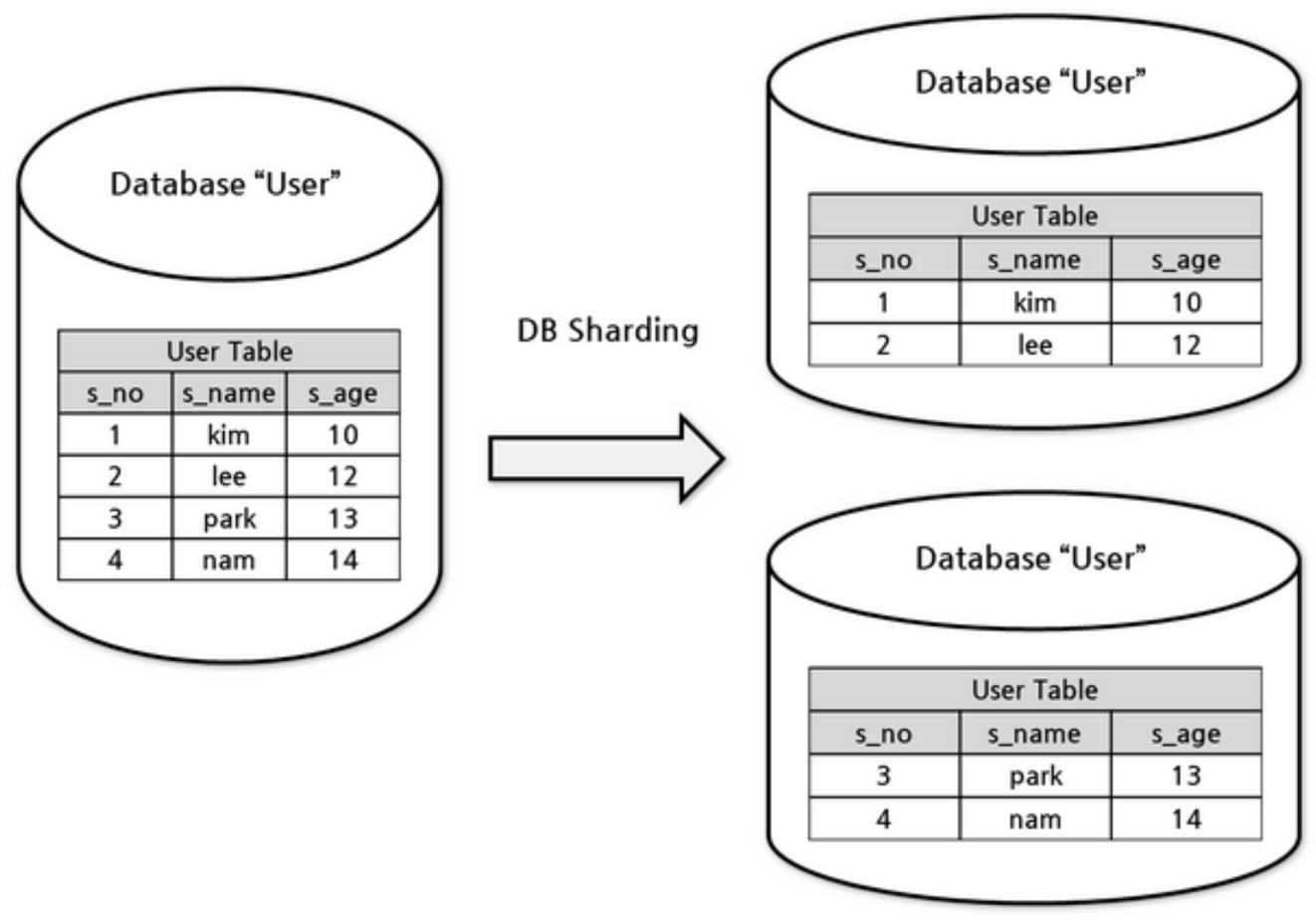
SCALING UP VS. SCALING OUT



http://vinfrastructure.it/wp-content/uploads/2012/08/Storage-Scaleout-Scalein.png

SHARDING / PARTITIONING

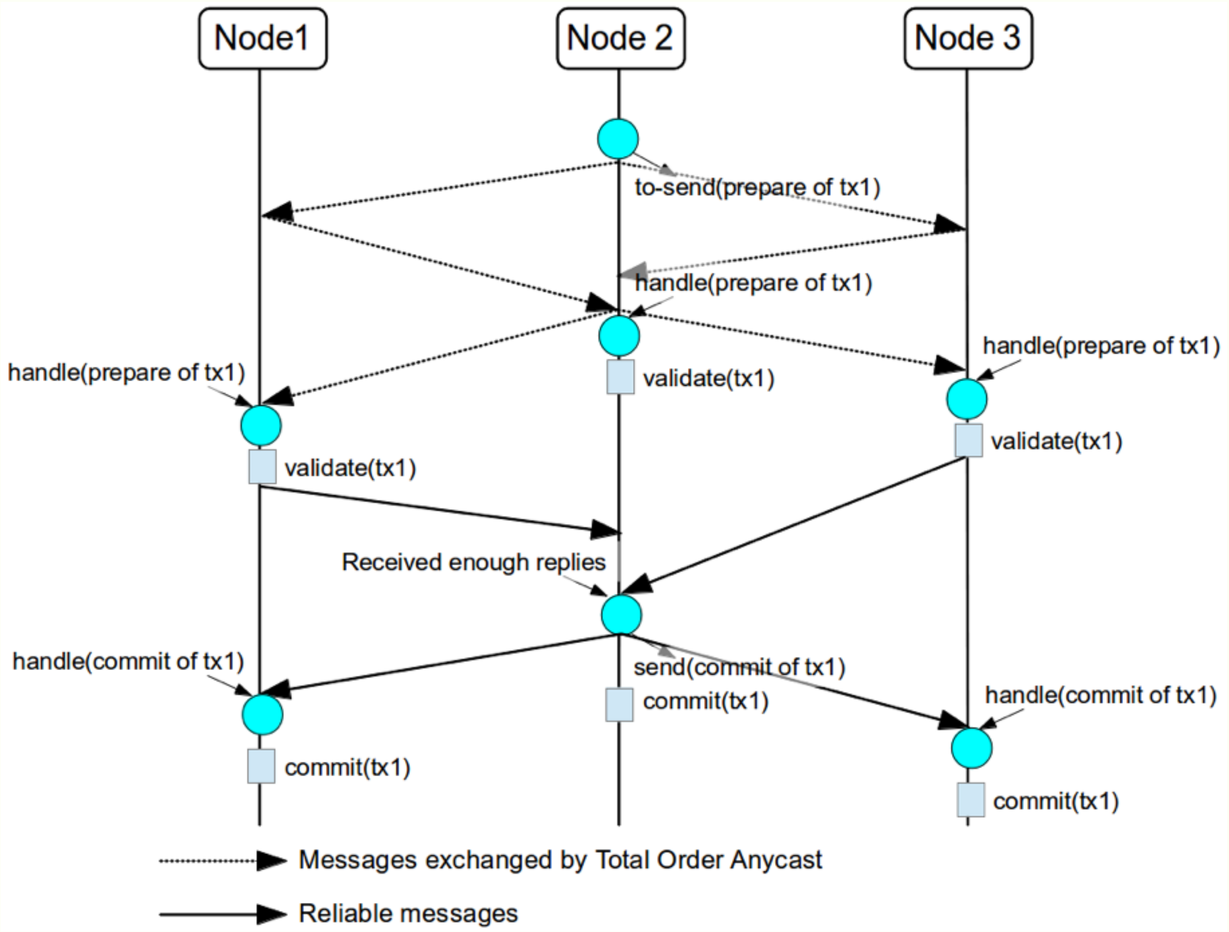
- DISTRIBUTED JOINS?
- DISTRIBUTED REFERENTIAL?
  INTEGRITY?



http://www.cubrid.org/manual/91/en/_images/image39.png

DISTRIBUTED TRANSACTIONS

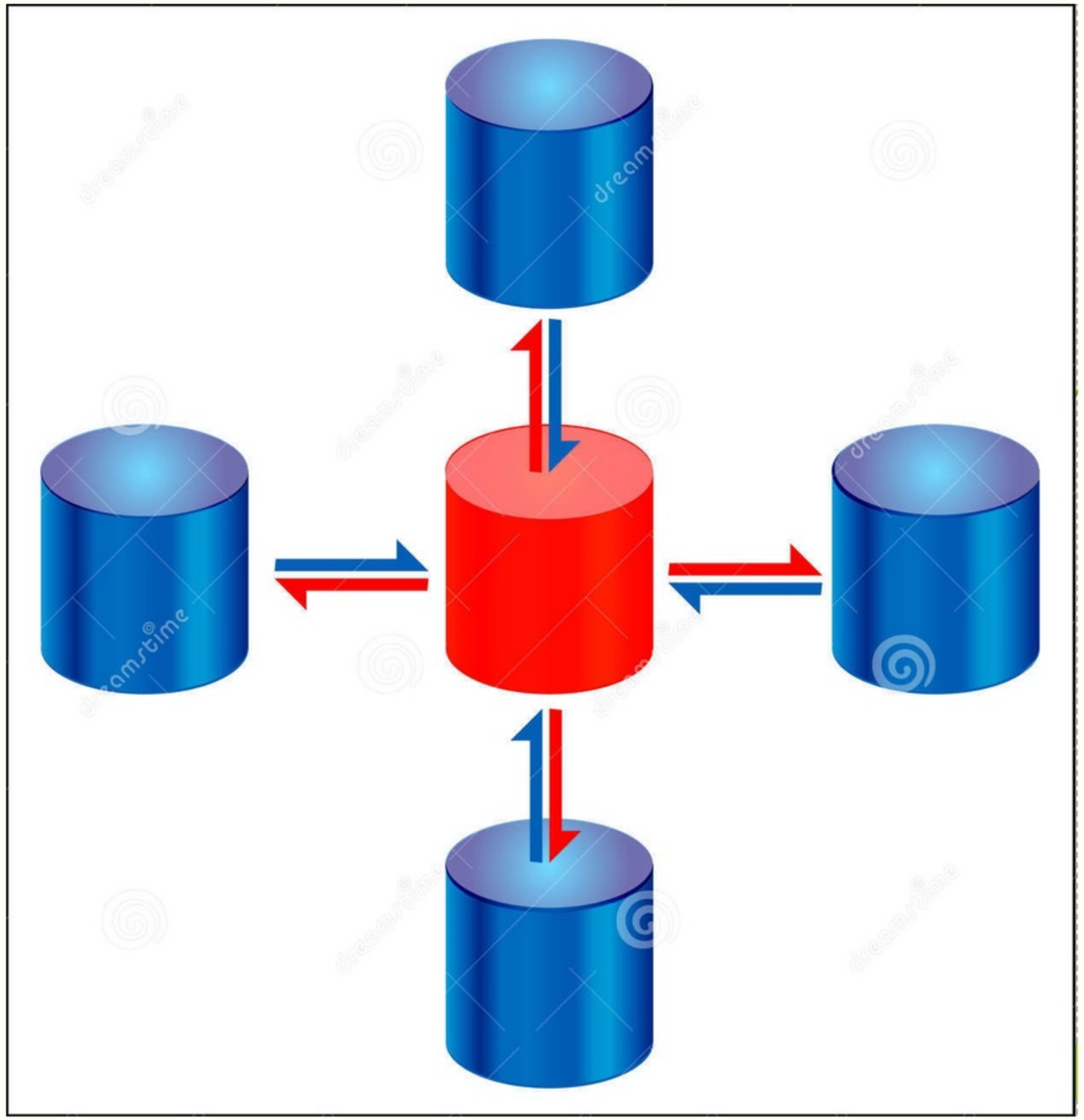
- ATOMICITY
  - LOCKS FOR THE WHOLE DURATION?



http://infinispan.org/docs/6.0.x/user_guide/images/fig6.png

REPLICATION

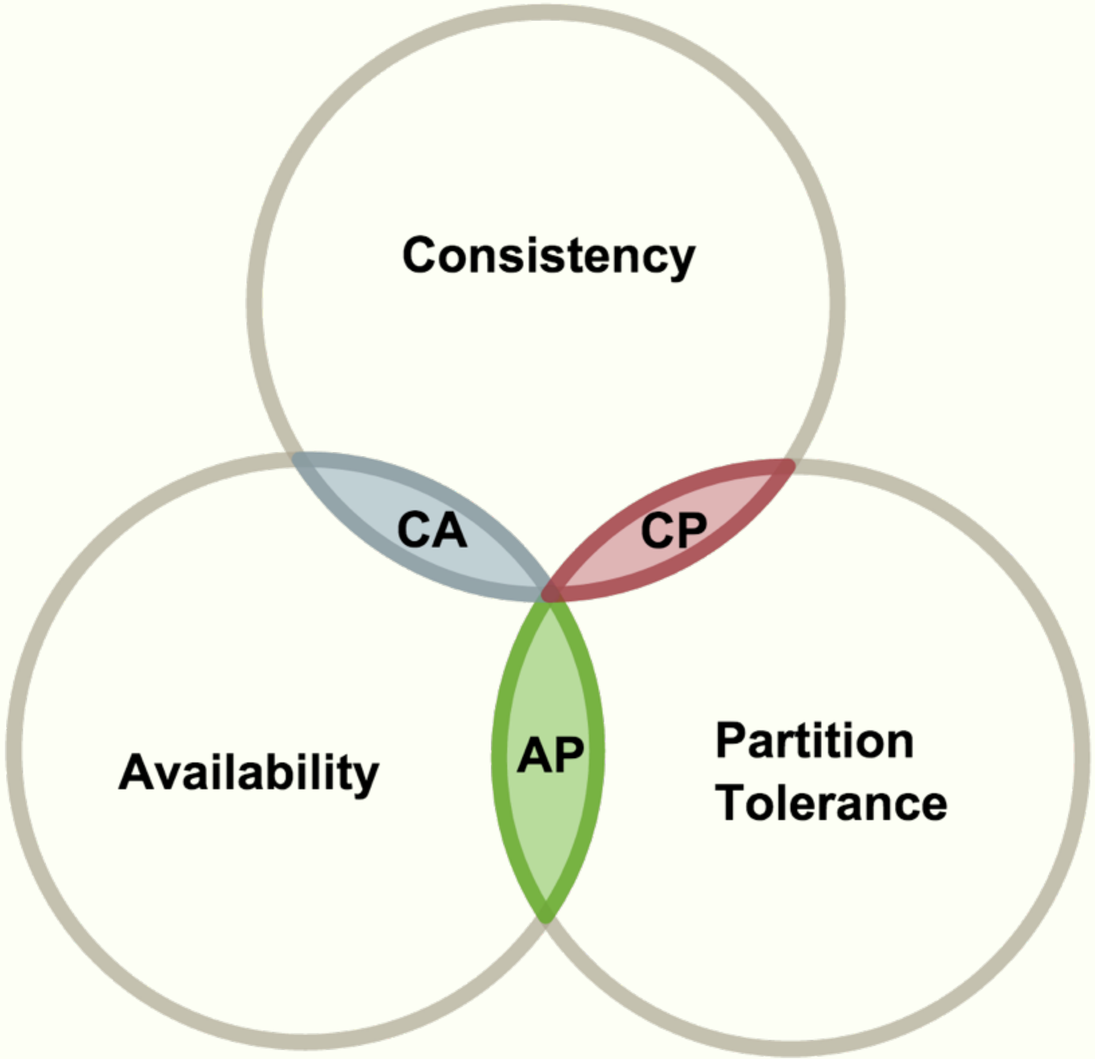
- CONSISTENCY -> HA AMONG REPLICAS
  - ONLINE REPLICATION -> HIGH LATENCY
  - POST-WRITE REPLICATION -> TRADEOFFS



http://thumbs.dreamstime.com/z/database-replication-7383674.jpg

CAP THEOREM

- IN DISTRIBUTED SYSTEMS -> P IS OBLIGATORY
- IN CASE OF PARTITION
  - C VS A IS THE TRADE-OFF!
  - NOT A BINARY DECISION!



http://berb.github.io/diploma-thesis/original/resources/cap.svg

RDBMS TOTALLY SUCK!

http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode

FACEBOOK ARCHITECTURE

http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode

FACEBOOK ARCHITECTURE – MYSQL!

- NO JOINS OR LARGE SCANS
- FAVOR READ AND WRITE AVAILABILITY OVER ATOMICITY OR CONSISTENCY
- ASYNCHRONOUS REPLICATION BETWEEN DATA CENTERS (EVENTUAL CONSISTENCY)
- MULTI-SHARD TRANSACTIONS PROHIBITED
- TAO = DATA STORE FOR THE SOCIAL GRAPH (MYSQL + MEMCACHED)

https://research.facebook.com/blog/1522692927972019/facebook-s-top-open-data-problems/

RDBMS TOTALLY SUCK?

PRANKED!

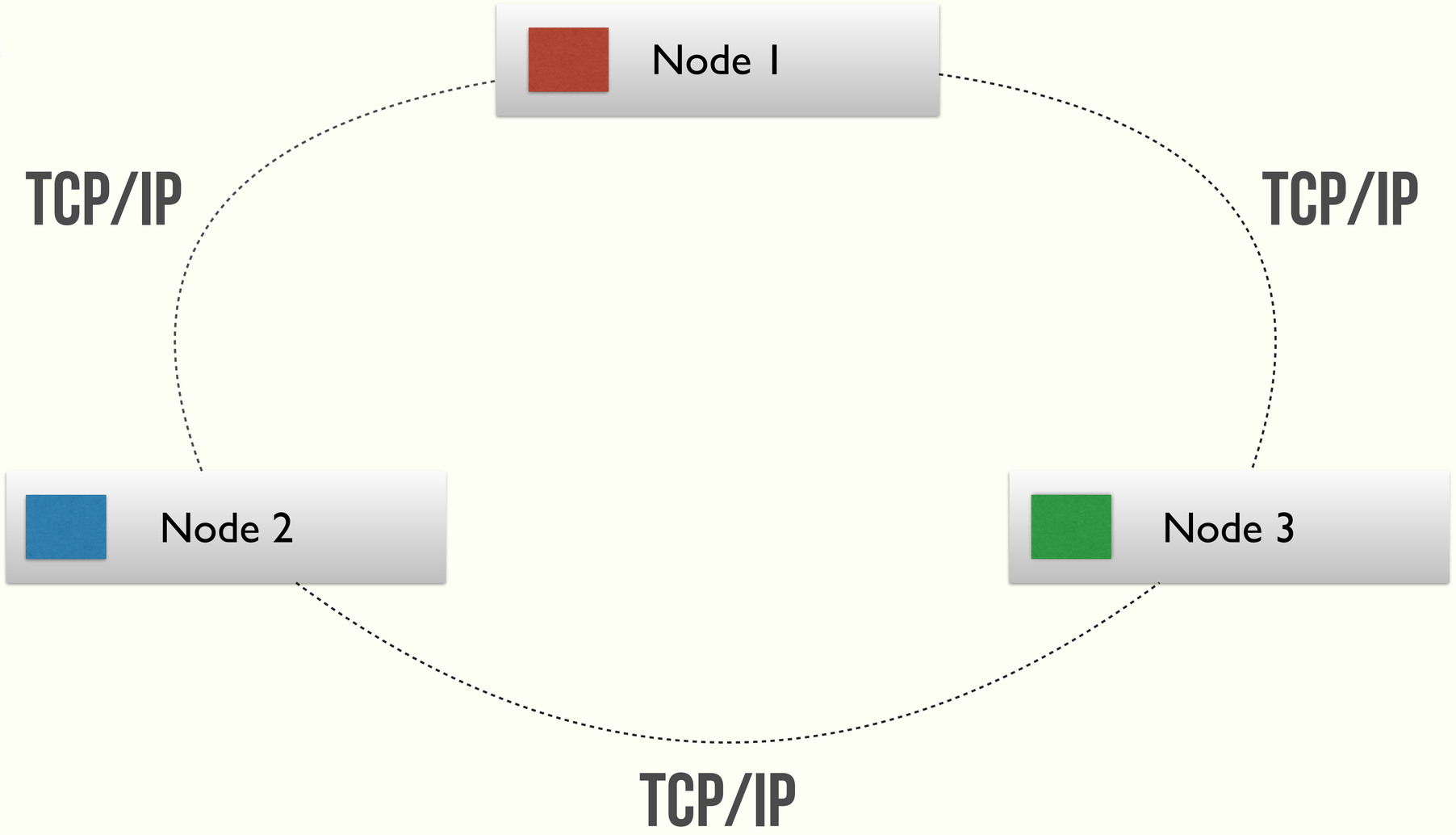

NOSQL

HAZELCAST

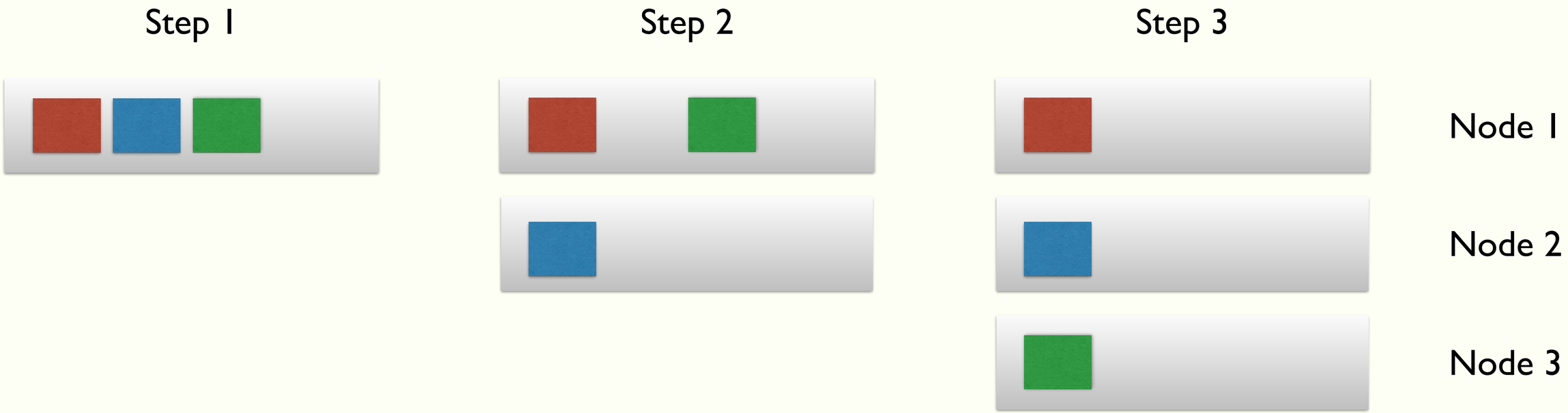
- LEADING OPEN-SOURCE IN-MEMORY DATA / COMPUTE GRID
- JCACHE PROVIDER
- KEY-VALUE (IMAP, JCACHE)
- JUST A JAR!

TOPOLOGY

- PEER-TO-PEER



PARTITIONING / SHARDING

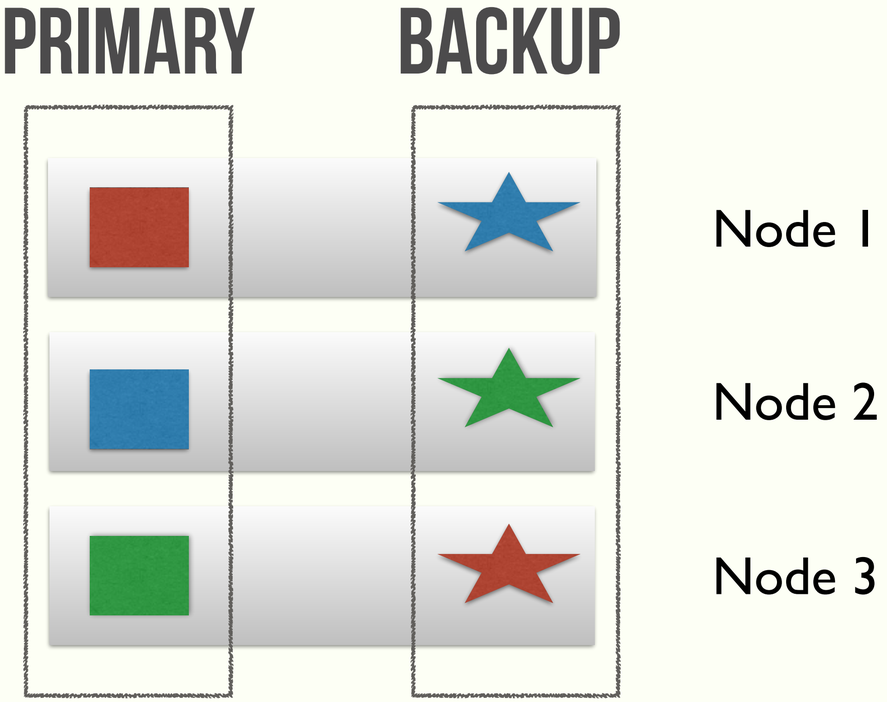


... 271 PARTITIONS

PARTITION_ID = HASH (KEY) % PARTITION_COUNT

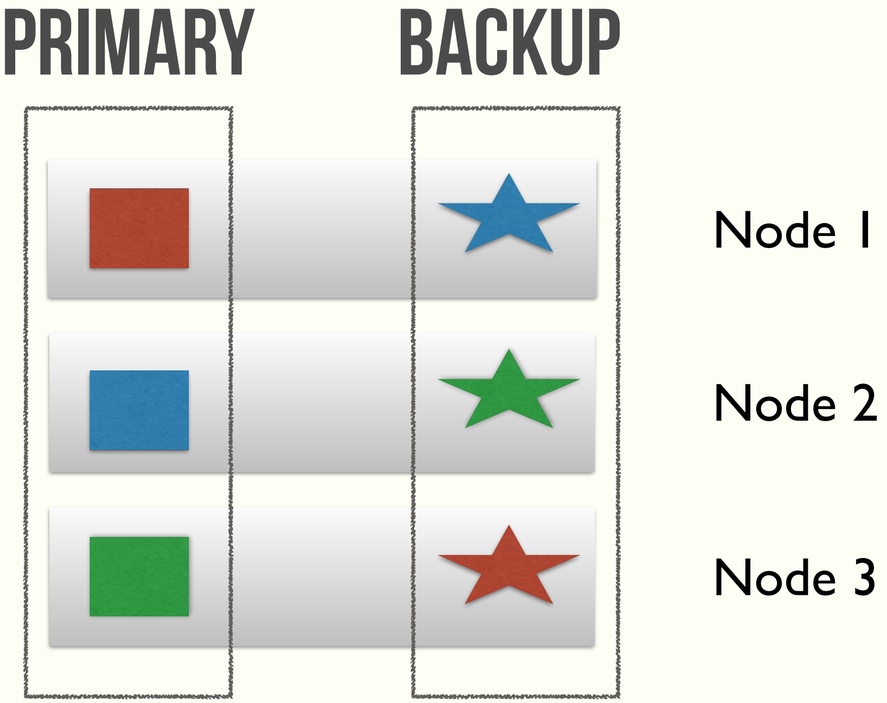
REPLICATION

- PRIMARY VS. BACKUP
  - SYNC
  - ASYNC
- BACKUP COUNT
  - PER DATA STRUCTURE



WRITING

- PUT -> SERIALISABLE ON PARTITION THREAD
- OPTIMISTIC "LOCKING"
  - REPLACE OPERATION
- PESSIMISTIC LOCKING
  - LOCK AND UNLOCK

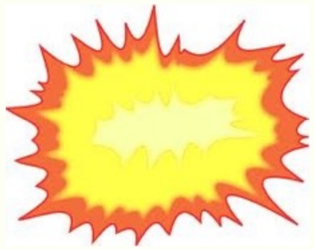


FAILOVER

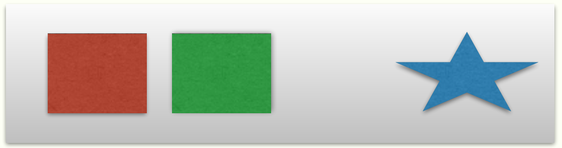
hazelcast

Step 1

Step 2



Step 3

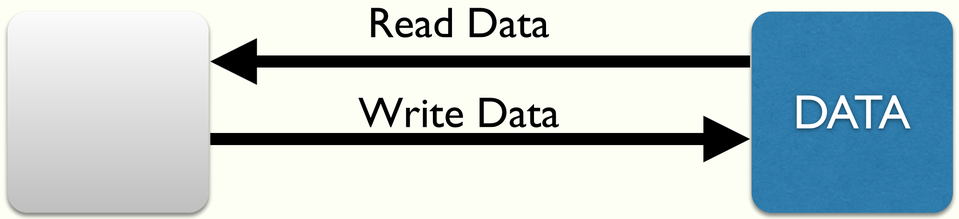


Node 1

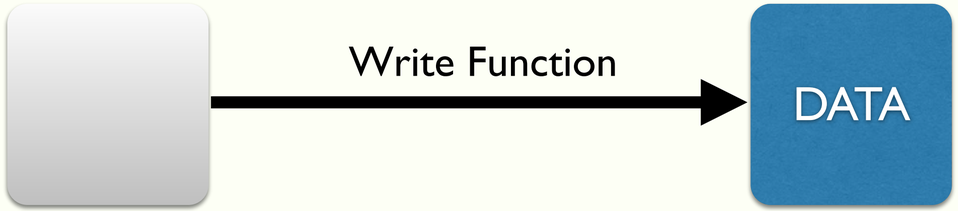
Node 2

Node 3

DATA PROCESSING

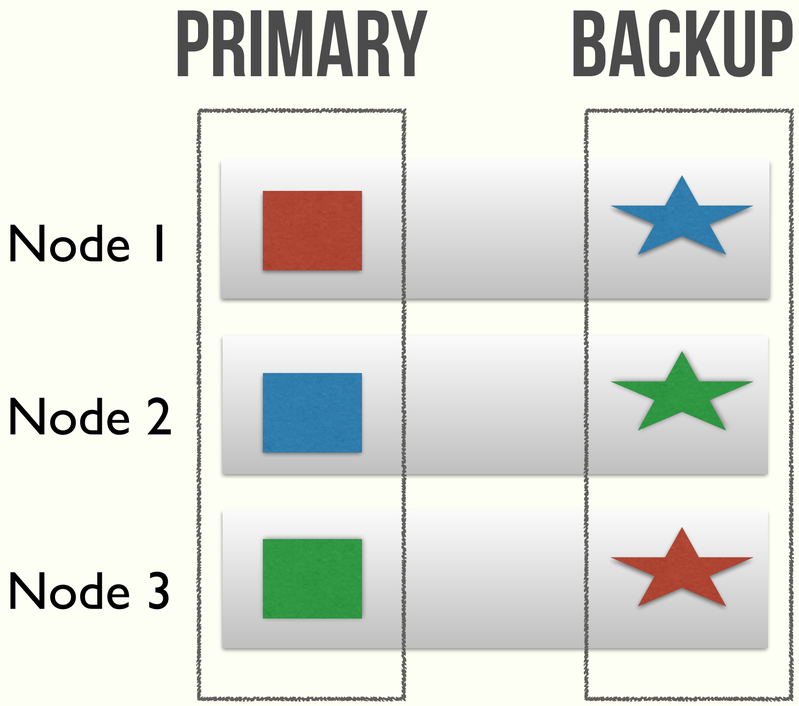


BAD: SEND DATA TO FUNCTION



GOOD: SEND FUNCTION TO DATA

ENTRY PROCESSOR



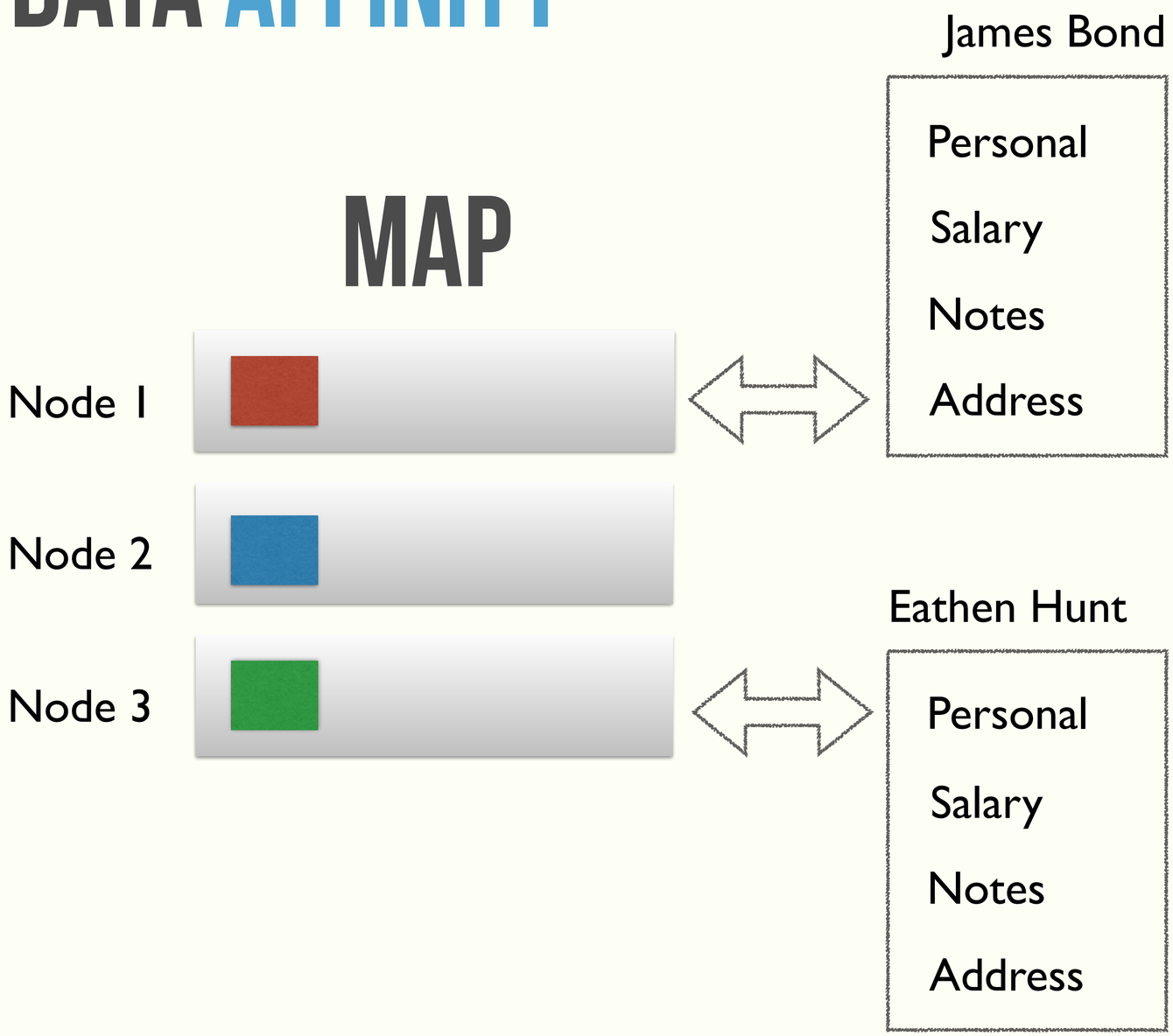

```
class SalaryProcessor
        extends AbstractEntryProcessor<> {

    int amount;

    SalaryProcessor(int amount) {
        this.amount = amount;
    }

    @Override
    public Object process(Map.Entry<> entry) {
        entry.getValue().salary += amount;
        return null;
    }
}
```

DATA AFFINITY

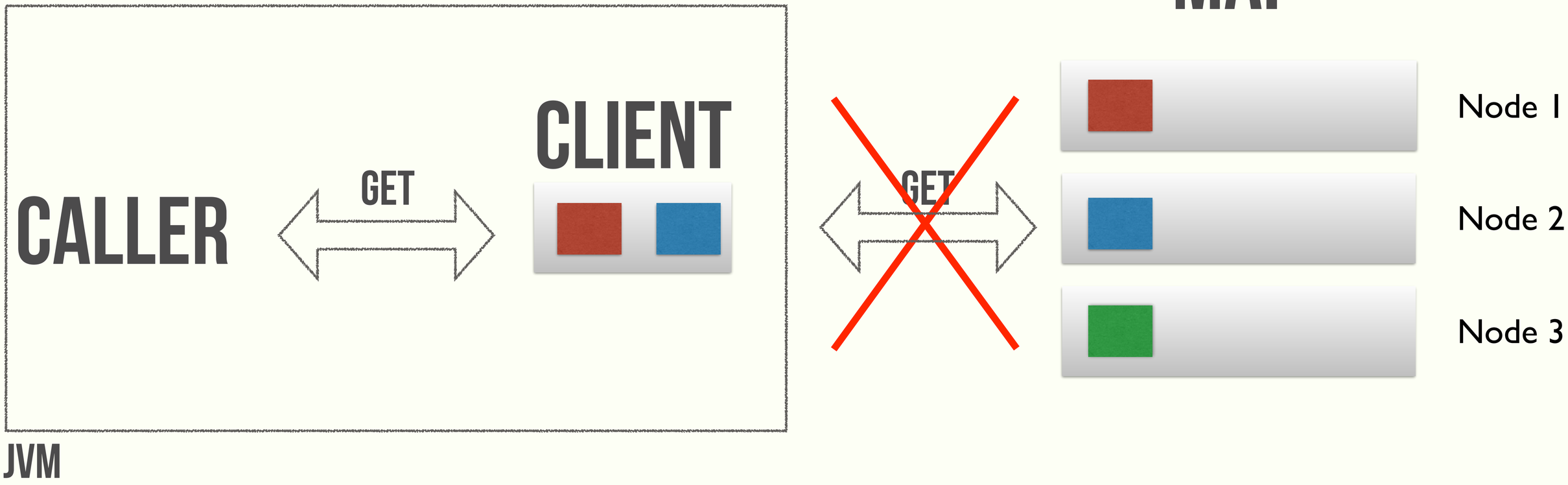



```
public class SalaryKey implements
  Serializable, PartitionAware {

  private final long employeeId;
  private final long salaryId;

  @Override
  public Object getPartitionKey()
  {
    return employeeId;
  }
```

WITH NEAR CACHE



QUERYING - PREDICATES

```
public Set<Employee> getWithNameOrAge( int age, double salary ) {

  Predicate age = Predicates.between( "age", 25, 35 );
  Predicate salary = Predicates.lessThan( "salary", 50000 );
  Predicate query = Predicates.or( age, salary );

  // SqlPredicate query = new SqlPredicate(
  //         "(age BETWEEN 25 AND 35) OR (salary < 50000)"
  // )

  return employees.values( query );

}
```

HAZELCAST USED BY OPEN-SOURCE

- VERT.X
- APACHE CAMEL, SHIRO, KARAF
- MULE ESB
- WSO2
- ALFRESCO

OVERVIEW

- DOCUMENT ORIENTED
- JSON-STYLE DOCUMENTS
  - MAX 16MB
- DYNAMIC (NO) SCHEMAS
- AD-HOC QUERY LANGUAGE

mongoDB

Collection

Document

```
db.users.insert(
                {
                    name: "sue",
                     age: 26,
                  status: "A",
                  groups: [ "news", "sports" ]
                }
               )
```

Document

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

insert

Collection

```
{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }
```

users

http://docs.mongodb.org/manual/_images/crud-insert-stages.png

DOCUMENTS

- BSON DOCUMENT
- DOCUMENT

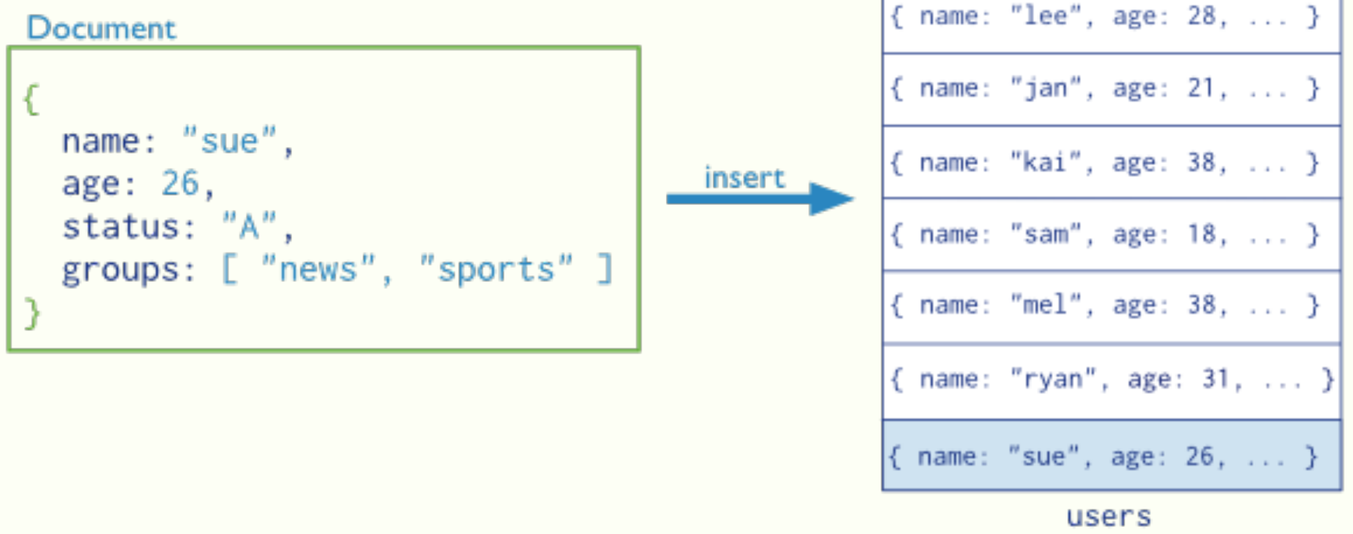
```
{
  "a" : "MongoDB",
  "b" : [ 1, 2 ]
}
```

```
new BsonDocument()
        .append("a", new BsonString("MongoDB"))
        .append("b", new BsonArray(
                Arrays.asList(new BsonInt32(1), new BsonInt32(2)))
        );
```

```
new Document().append("a", "MongoDB")
              .append("b", Arrays.asList(1, 2));
```

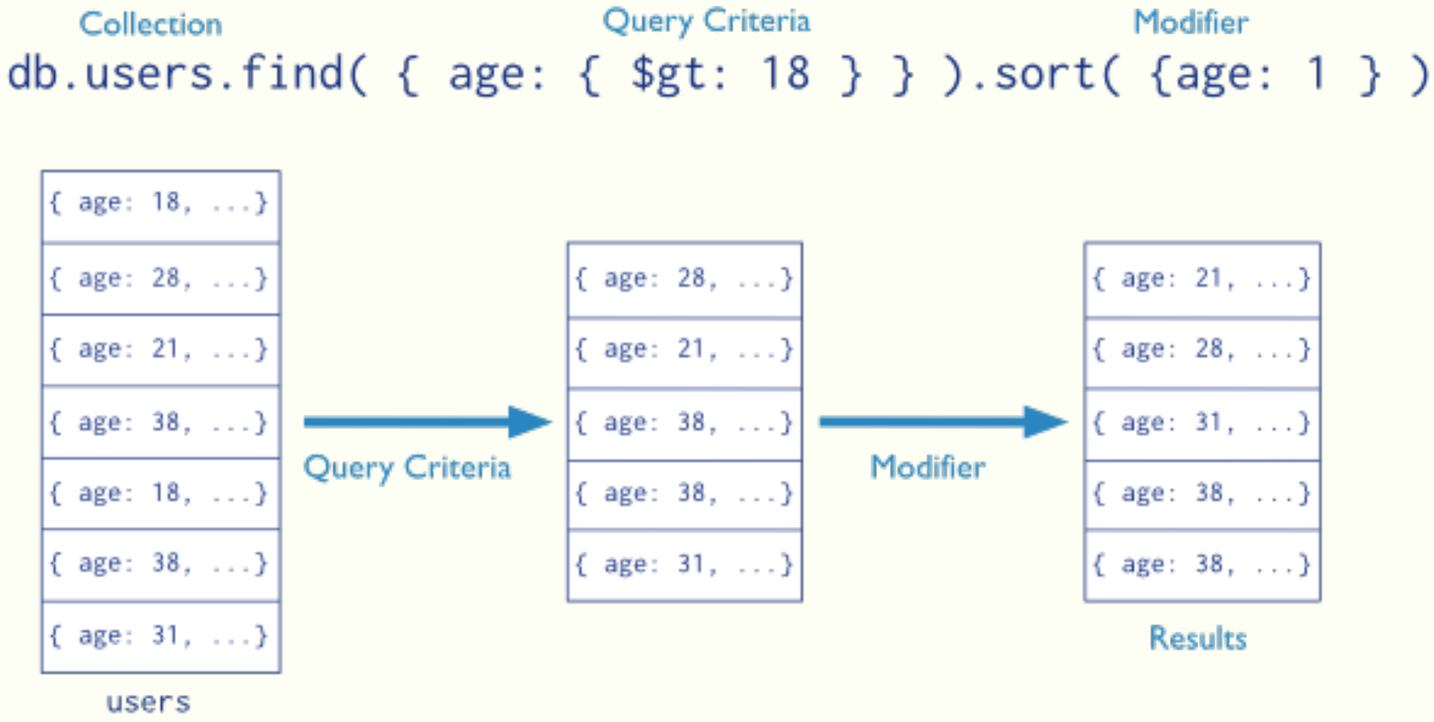
WRITING

- ATOMICITY ON SINGLE DOC
- NO MULTI-DOC TX
  - $ISOLATED



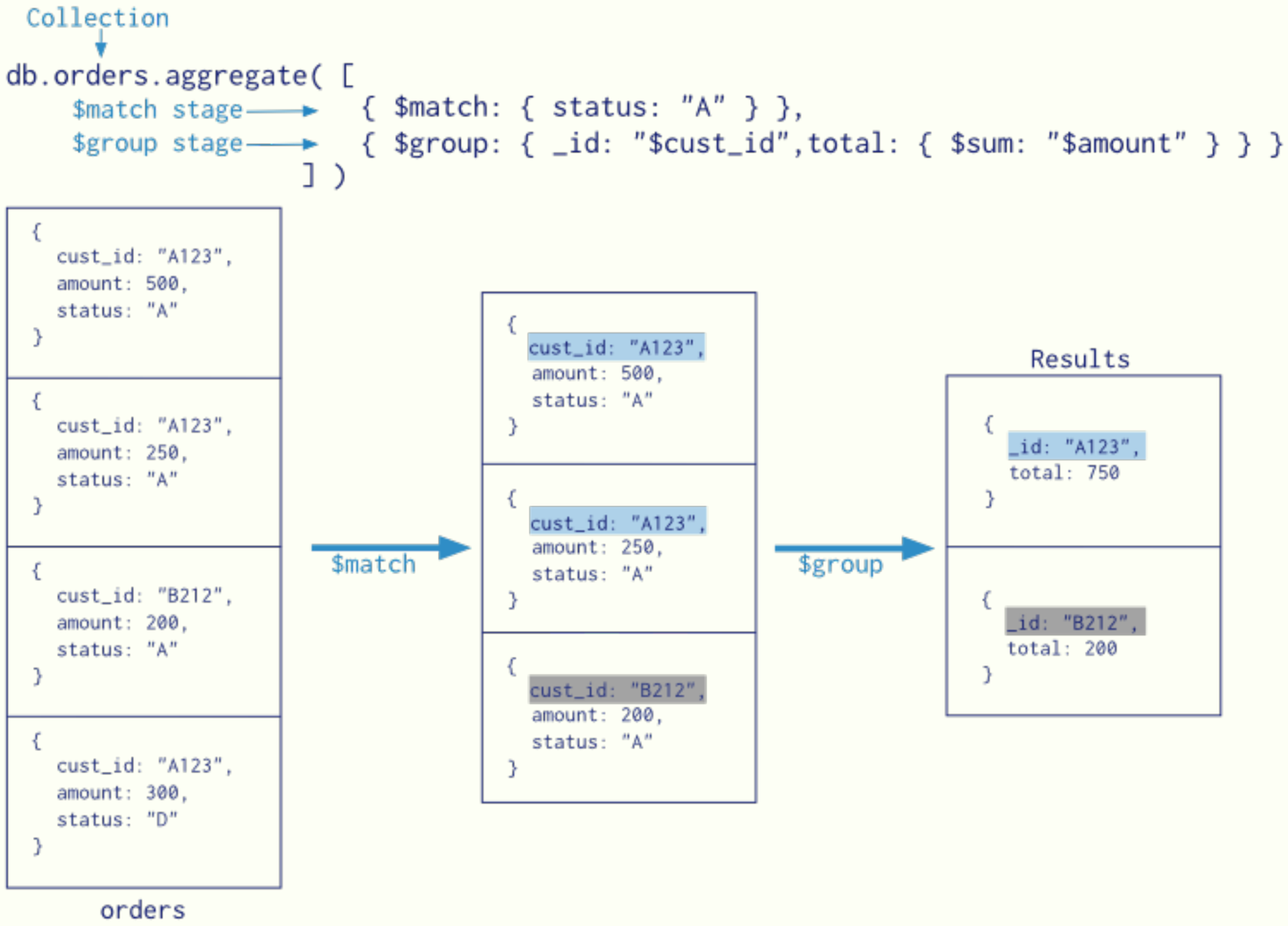
QUERYING

- NO SQL
- NO JOINS
  - $LOOKUP
- SINGLE COLLECTION
- PROJECTIONS
- AGGREGATIONS



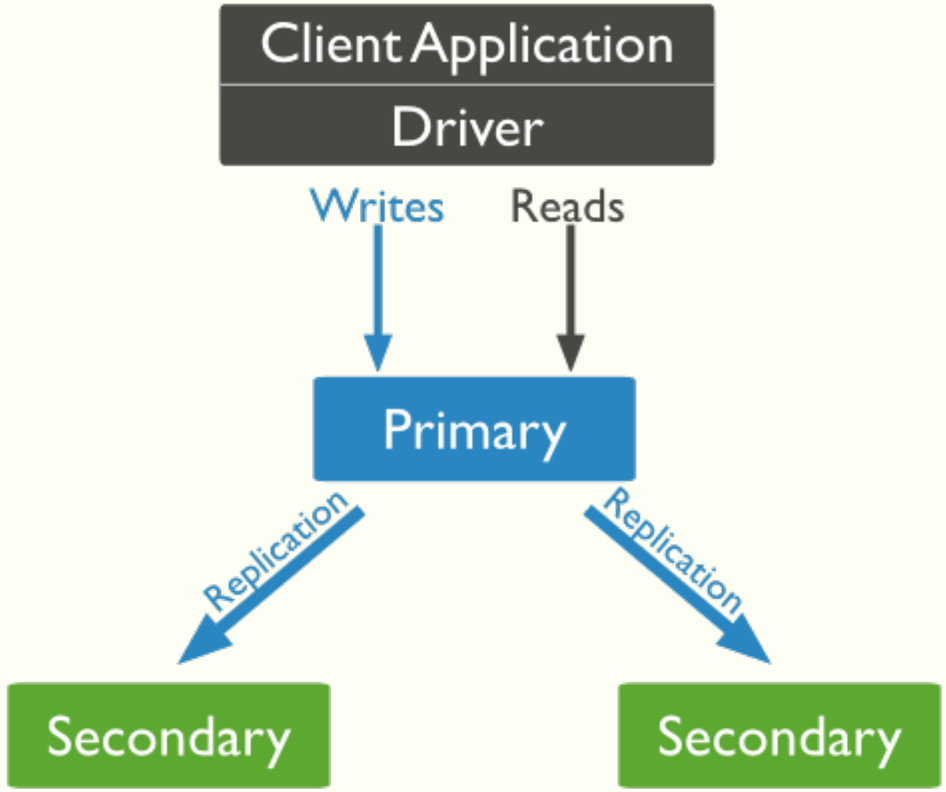
AGGREGATIONS

- MULTIPLE STAGES
  - $LOOKUP



REPLICATION

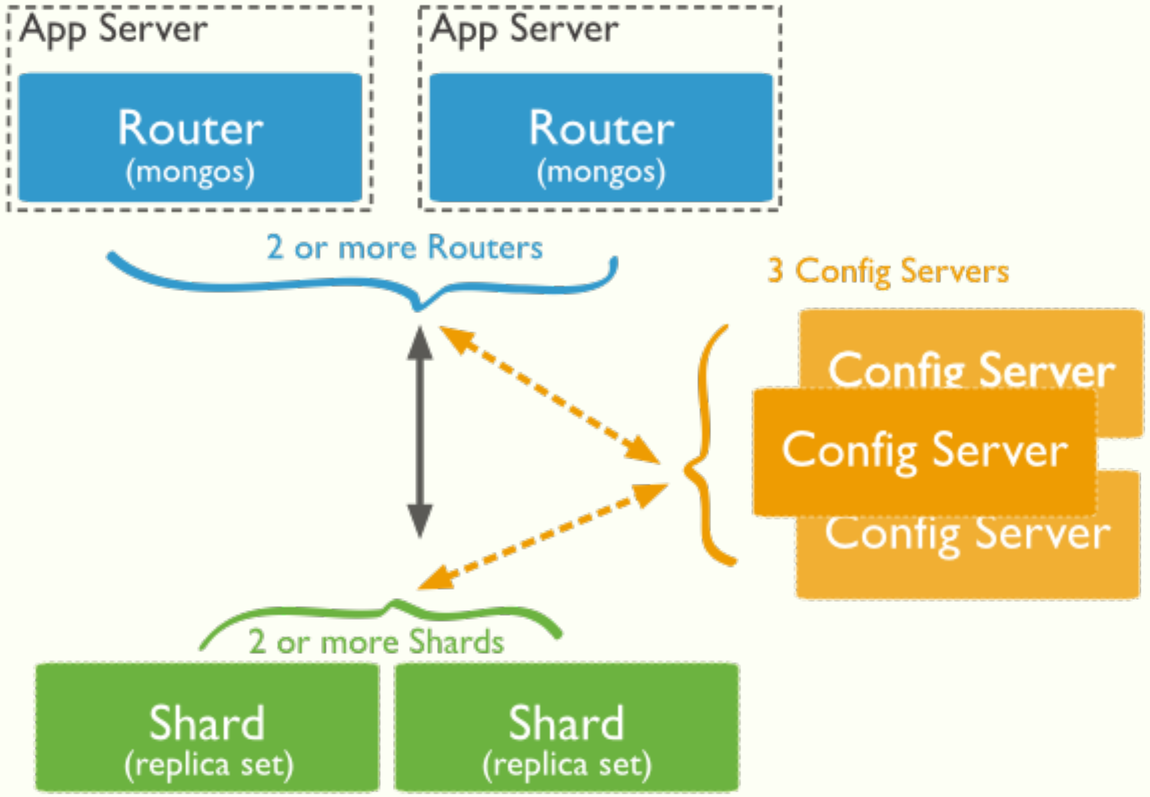
- ONLY ONE PRIMARY
- ASYNC REPLICATION
- WRITE TO PRIMARY ONLY
- READ FROM PRIMARY -> STRICT CONSIST.
- READ FROM SECONDARY -> EVENTUAL CONSIST.
- AUTOMATIC FAILOVER -> REELECTION



http://docs.mongodb.org/manual/core/replication-introduction/#asynchronous-replication

SHARDING

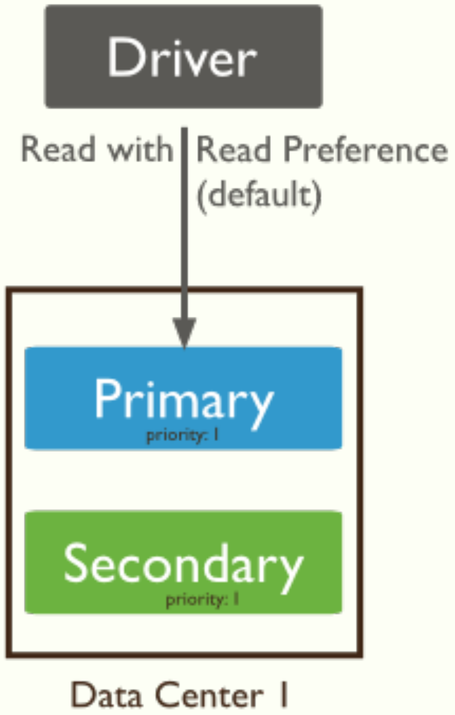
- SHARDS
- CONFIG SERVERS
- ROUTERS (MONGOS)
- SHARDS

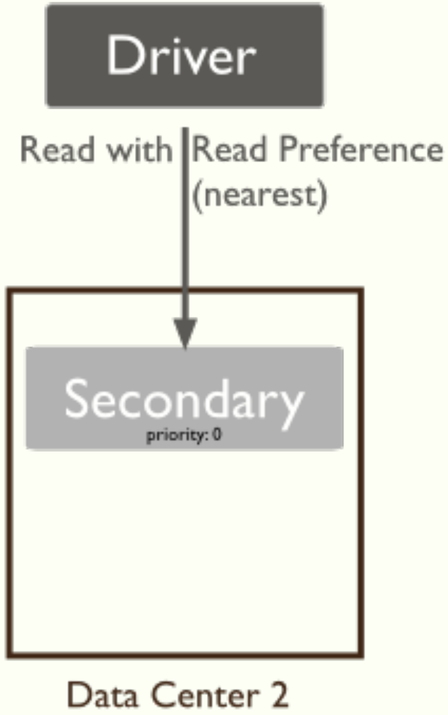


http://docs.mongodb.org/manual/core/sharded-cluster-components/

READ PREFERENCE

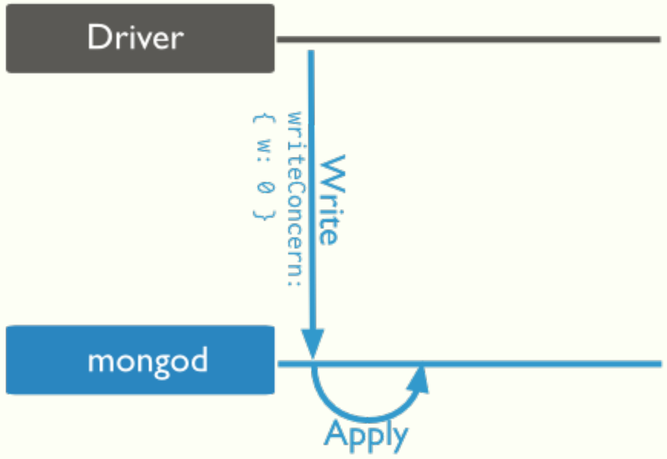
- PRIMARY (DEFAULT)
- PRIMARY_PREFFERED
- SECONDARY
- SECONDARY_PREF.
- NEAREST

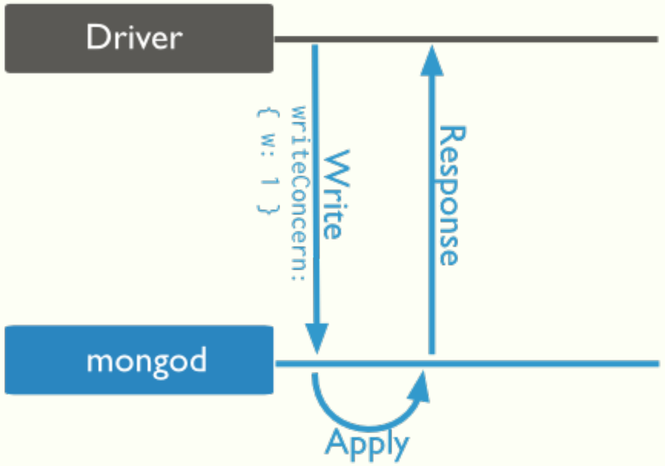


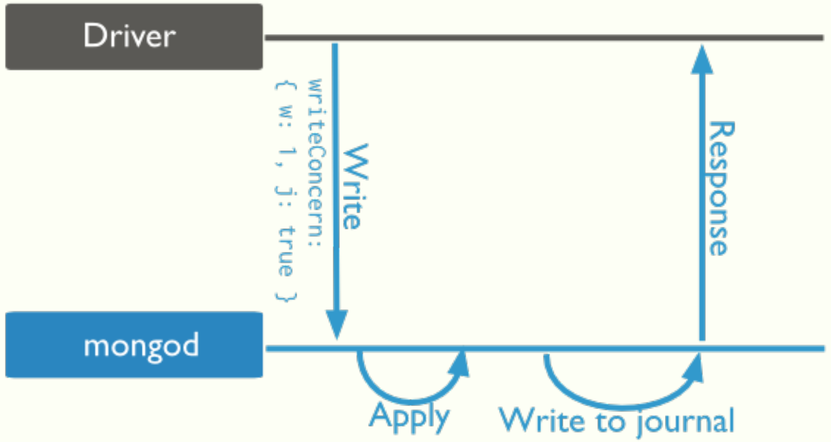


http://docs.mongodb.org/manual/core/read-preference/

WRITE CONCERN







- UNACKNOWLEDGED
  W == 0
- ACKNOWLEDGED
  W ==1
- JOURNALED
  W ==1 ; J=TRUE

USE CASES

- WEB-BASED APPS
- EBAY
  - METADATA STORAGE
  - CLOUD MANAGEMENT
  - MERCHANDISING CATEGORIZATION
- THE WEATHER CHANNEL
  - NEAR REAL TIME WEATHER ALERTS

http://www.mongodb.com/who-uses-mongodb/

http://www.mongodb.com/presentations/real-world-mongodb-use-cases-financial-services

APACHE HBASE

- BIG TABLE STORE
- COLUMN-ORIENTED
- STRONGLY CONSISTENT RW
- AUTOMATIC SHARDING

Row Key	Customer		Sales	
Customer Id	Name	City	Product	Amount
101	John White	Los Angeles, CA	Chairs	$400.00
102	Jane Brown	Atlanta, GA	Lamps	$200.00
103	Bill Green	Pittsburgh, PA	Desk	$500.00
104	Jack Black	St. Louis, MO	Bed	$1600.00

Column Families

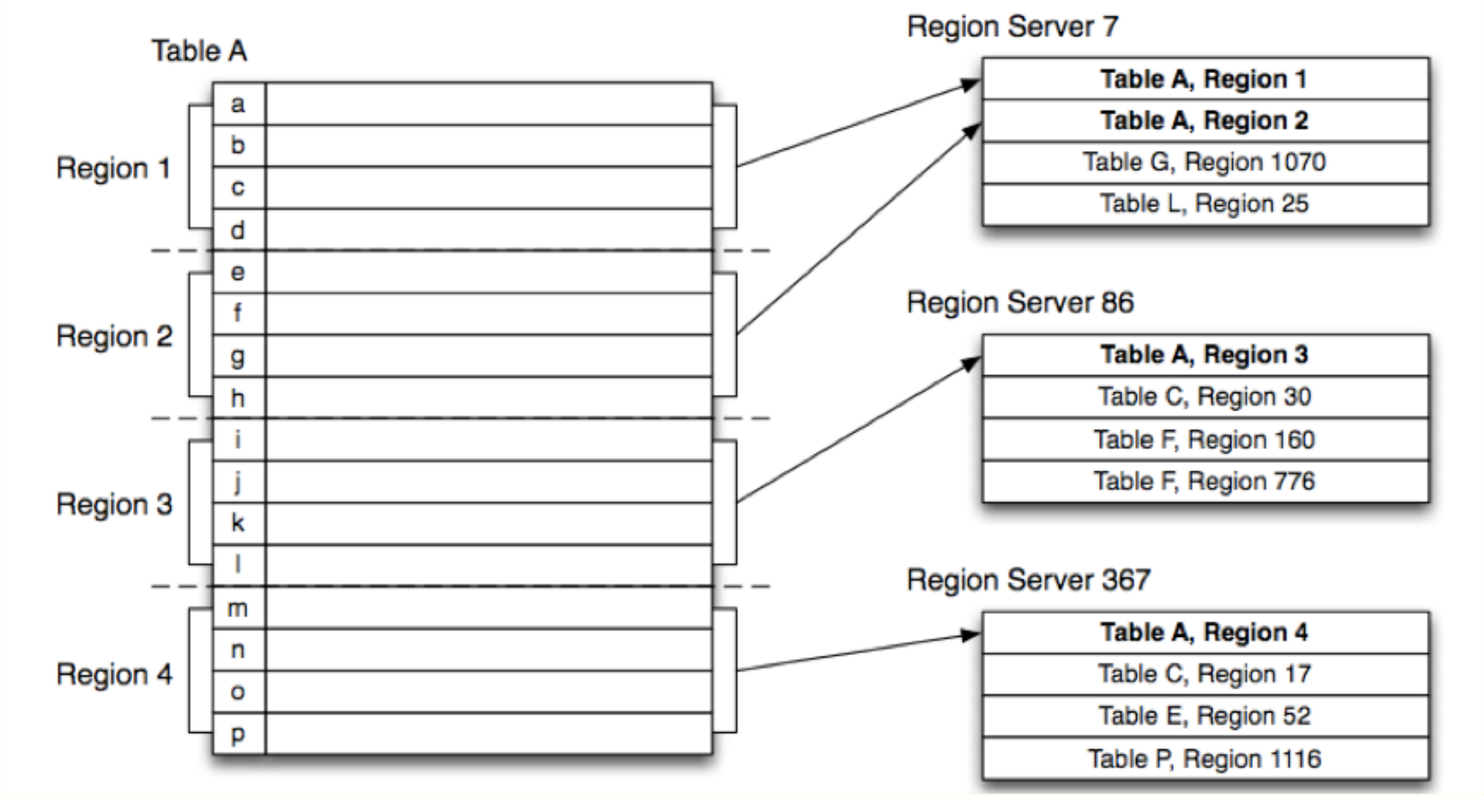
http://netwovenblogs.azurewebsites.net/wp-content/uploads/2013/10/hbase-2.png

STORAGE

- SCHEMA-LESS
- COLUMN-ORIENTED
- WIDE AND SPARSE TABLES

Row ID	Customer	Product	Amount
101	John White	Chairs	$400.00
102	Jane Brown	Lamps	$500.00
103	Bill Green	Lamps	$150.00
104	Jack Black	Desk	$700.00
105	Jane Brown	Desk	$650.00
106	Bill Green	Desk	$900.00

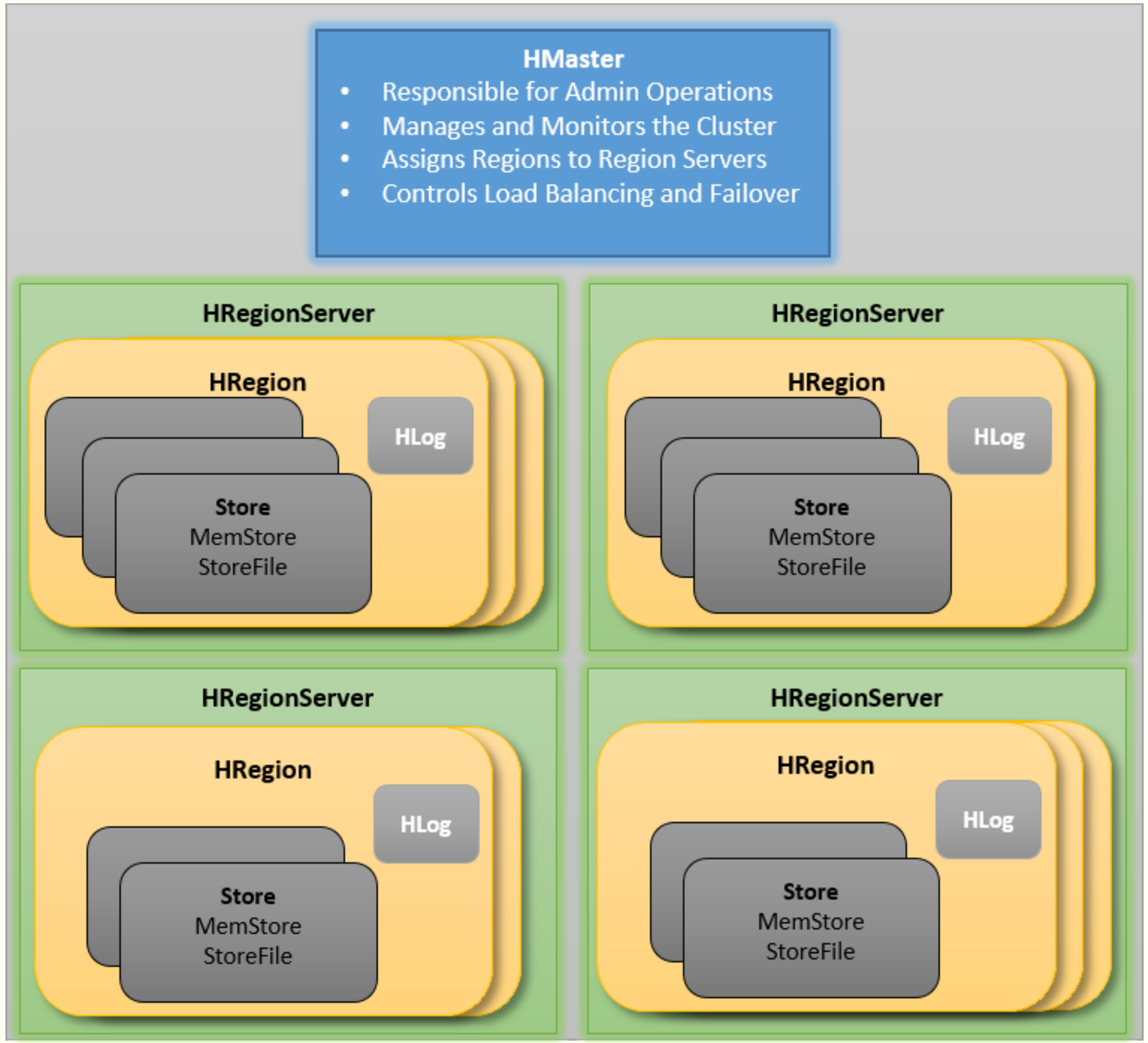
HBASE SHARDING



http://i62.tinypic.com/16h0rdk.jpg

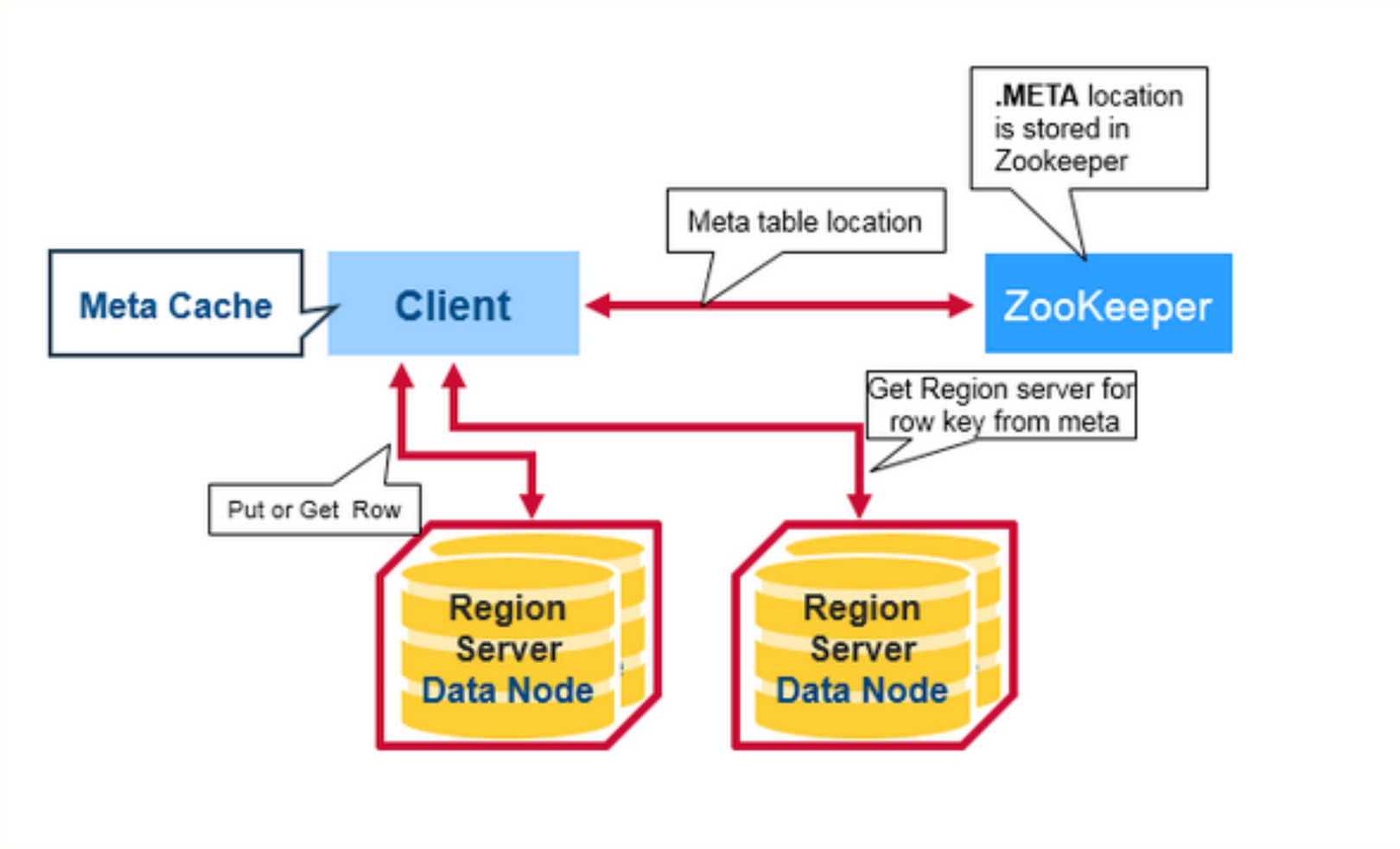
HBASE ARCHITECTURE

- MASTER
  - ASSIGNS THE REGIONS
  - LOAD-BALANCING
    - FAIL-OVER
- REGION SERVER
  - MANAGE/SPLIT REGIONS
  - READ/WRITE OPERATIONS



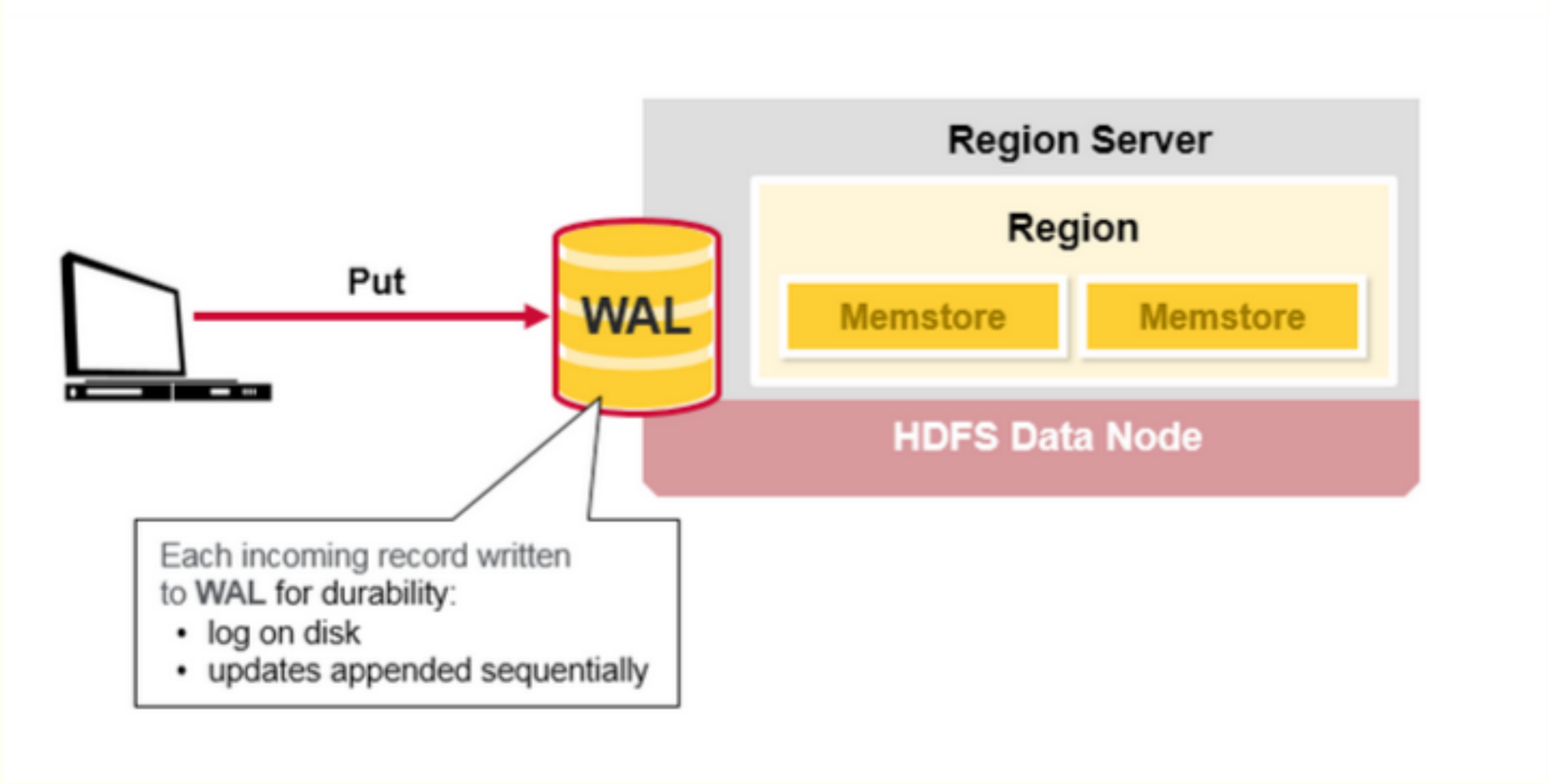
http://netwovenblogs.azurewebsites.net/wp-content/uploads/2013/10/hbase-12.png http://blog.pandorafms.org/wp-content/uploads/2014/07/hbase_logo.png

HBASE READ / WRITE



https://www.mapr.com/sites/default/files/blogimages/HBaseArchitecture-Blog-Fig6.png

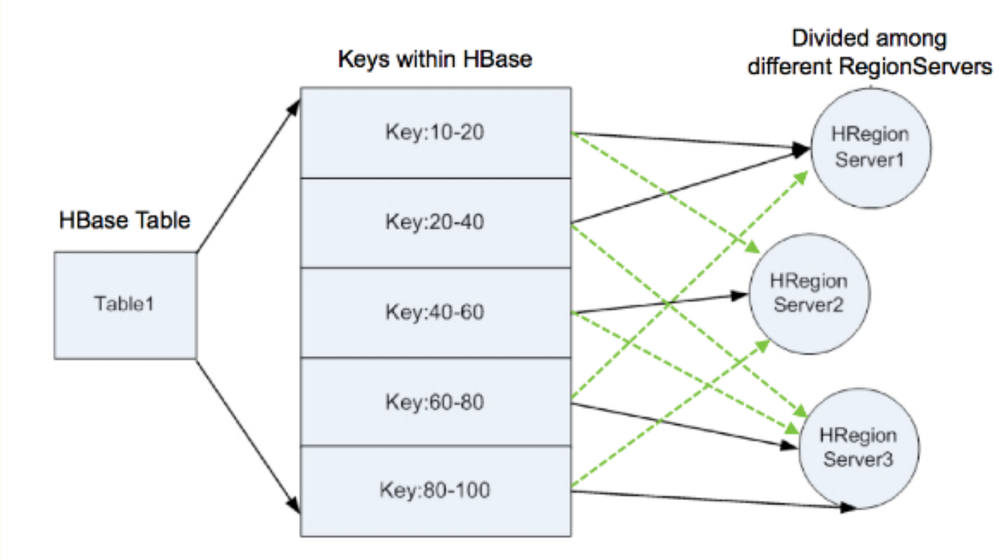
HBASE READ / WRITE



https://www.mapr.com/sites/default/files/blogimages/HBaseArchitecture-Blog-Fig9.png

HBASE REPLICATION & SHARDING

- ASYNC REPLICATION
  - HDFS REPLICATION FACTOR
- PRIMARY REGION -> READ-WRITE
- SECONDARY REGION -> READ ONLY
- SECONDARY -> MAY BE STALE

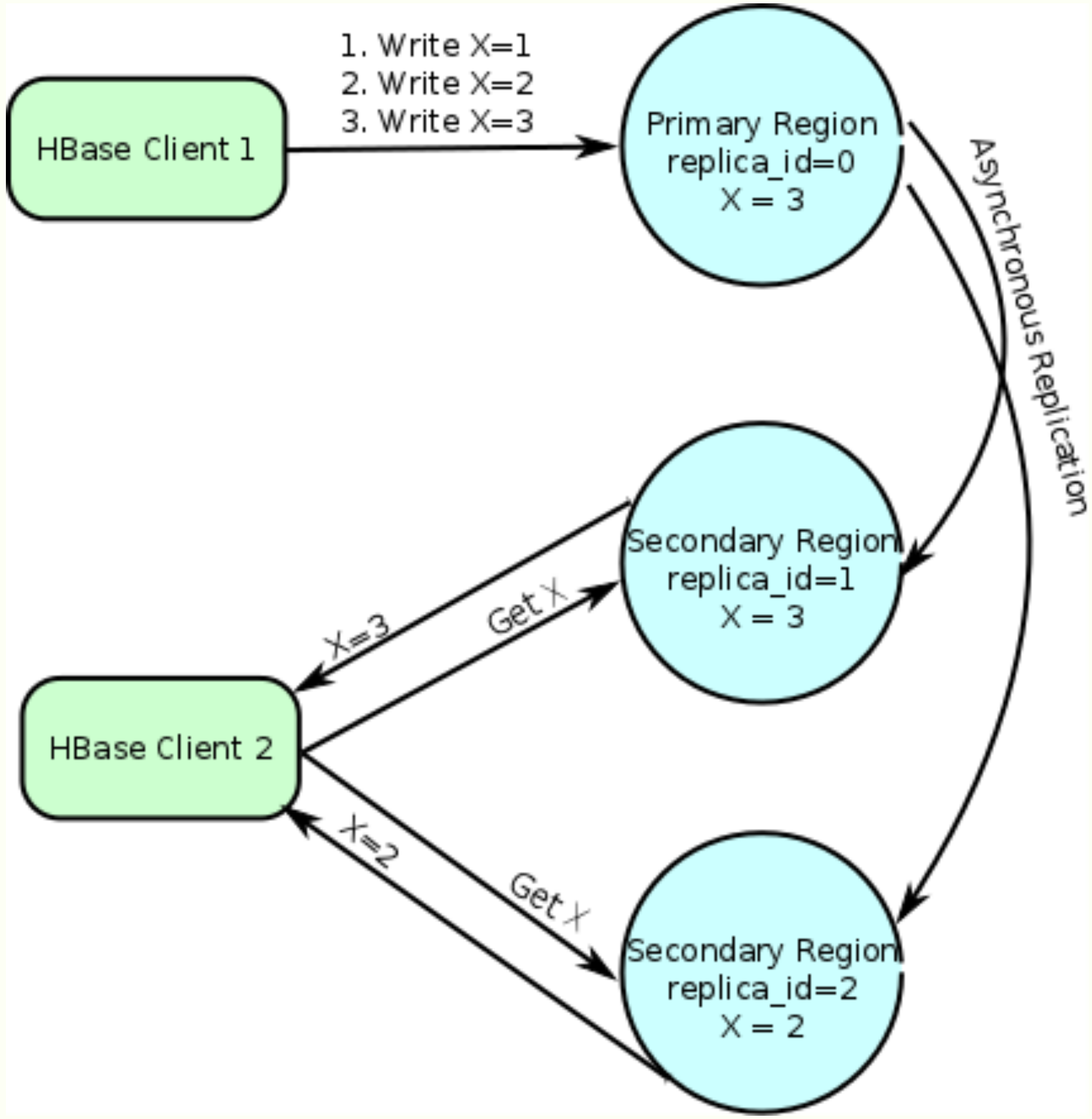


http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1.2/bk_system-admin-guide/content/figures/4/figures/primary-region-server.png

http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1.2/bk_system-admin-guide/content/figures/4/figures/secondary-region-servers.pr

HBASE CONSISTENCY

- STRONG
- TIMELINE (FROM 01.2015)



http://hortonworks.com/blog/apache-hbase-high-availability-next-level/

http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1.2/bk_system-admin-guide/content/figures/4/figures/timeline-data-consistency.p

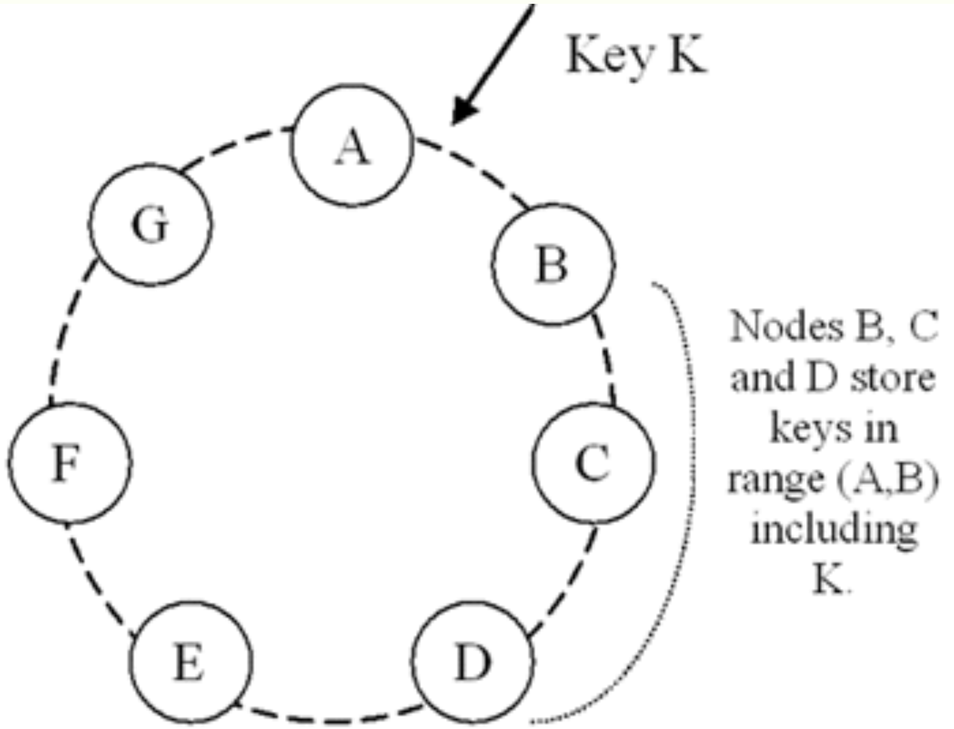
HBASE USE CASES

- FACEBOOK MESSAGING
- HIGH WRITE THROUGHPUT
- LOW LATENCY RANDOM READS
- STRONG CONSISTENCY
- HA VIA AUTOMATIC FAILOVER

http://de.slideshare.net/sawjd/h-base-20140613

DYNAMO STORAGE

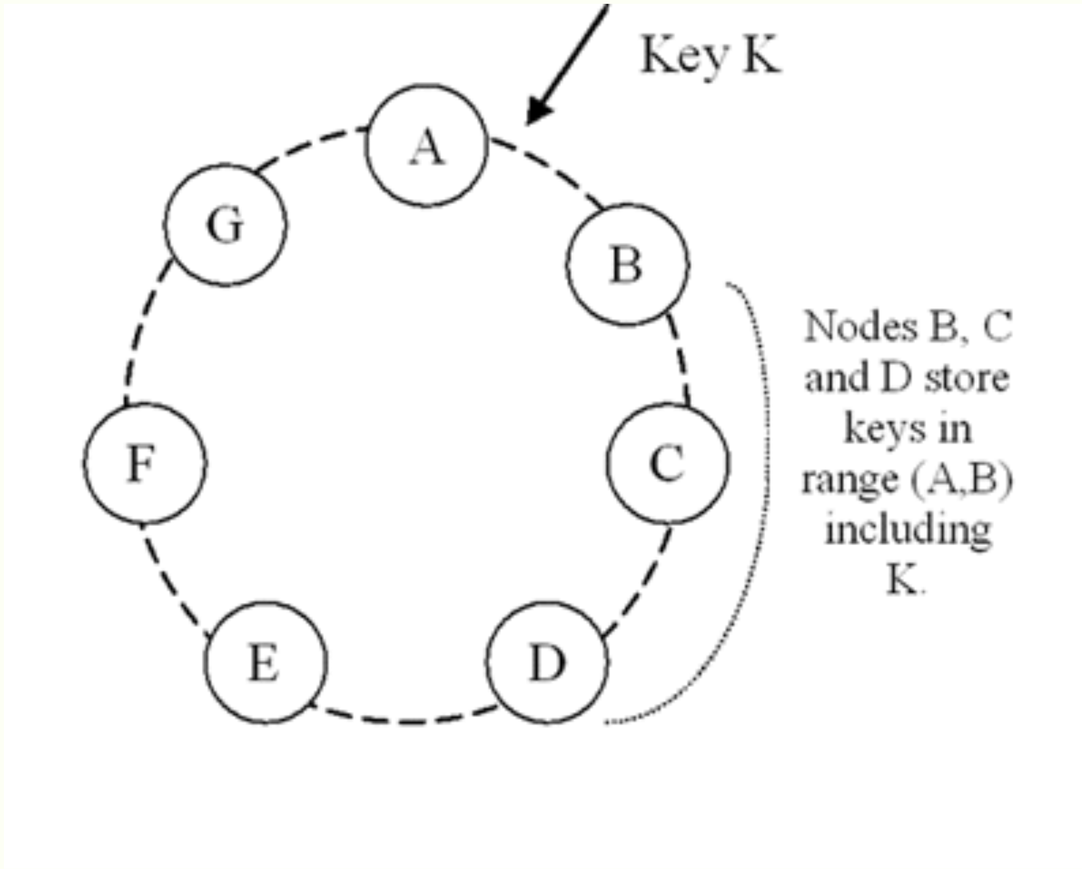
- HA KEY-VALUE STRUCTURED STORAGE
- DATA PARTITIONED USING CONSISTENT HASHING
- LINEAR SCALABILITY
- MASTER-MASTER
- CONSISTENCY FACILITATED BY OBJECT VERSIONING
- HINTED-HANDOFF
- ANTI-ENTROPY USING MERKLE-TREE



http://s3.amazonaws.com/wernervogels/public/sosp/sosp-figure2-small.png

APACHE CASSANDRA

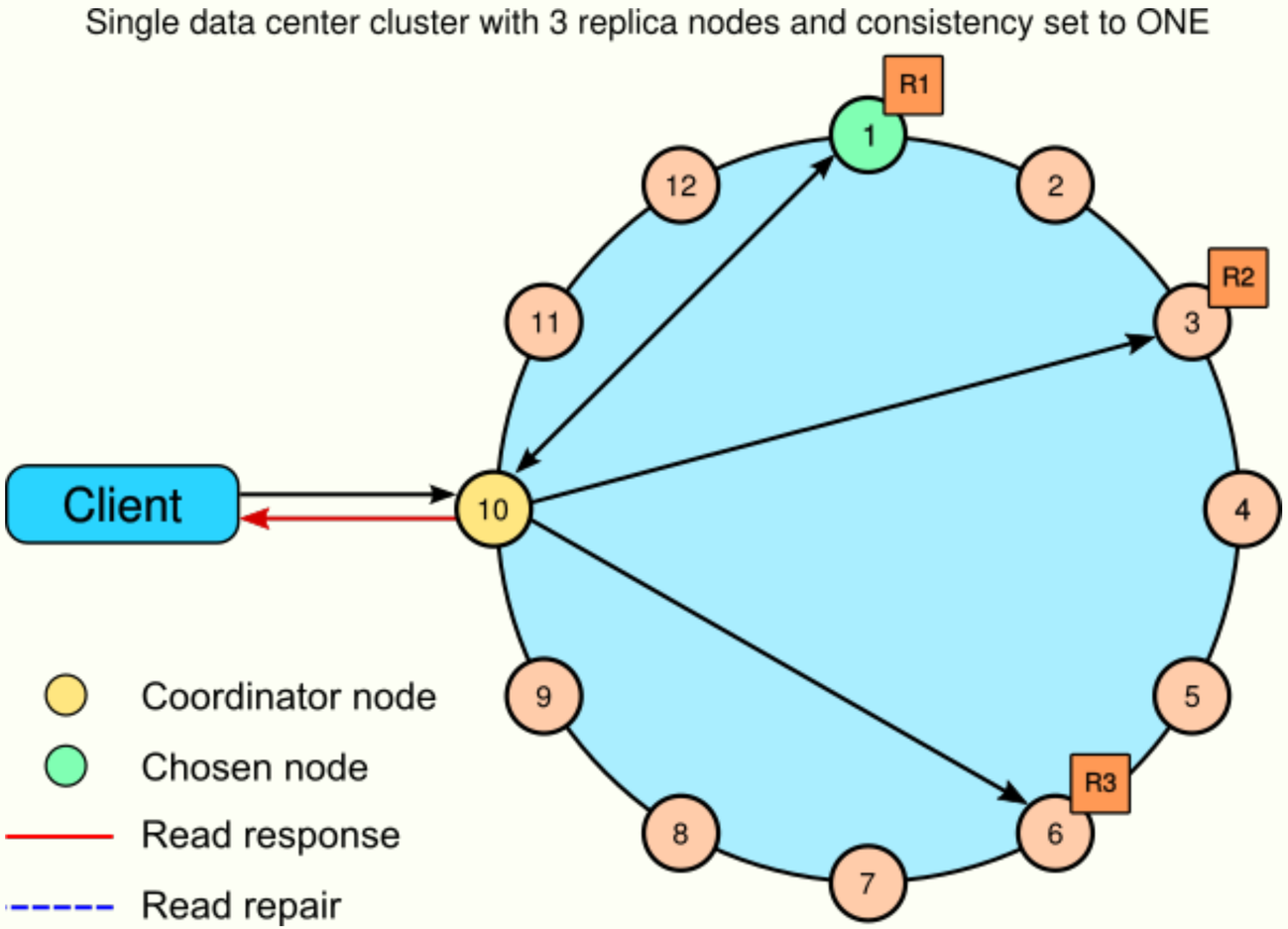
- BIG-TABLE + DYNAMO
- HIGH AVAILABILITY
- INCREMENTAL SCALABILITY
- EVENTUALLY CONSISTENT
- TUNNABLE CONSISTENCY AND LATENCY
- NO SINGLE POINT OF FAILURE



http://mbaron.developpez.com/tutoriels/nosql/cassandra/installation-outils-administration/images/cassandralogo.pn

KEY CONCEPTS

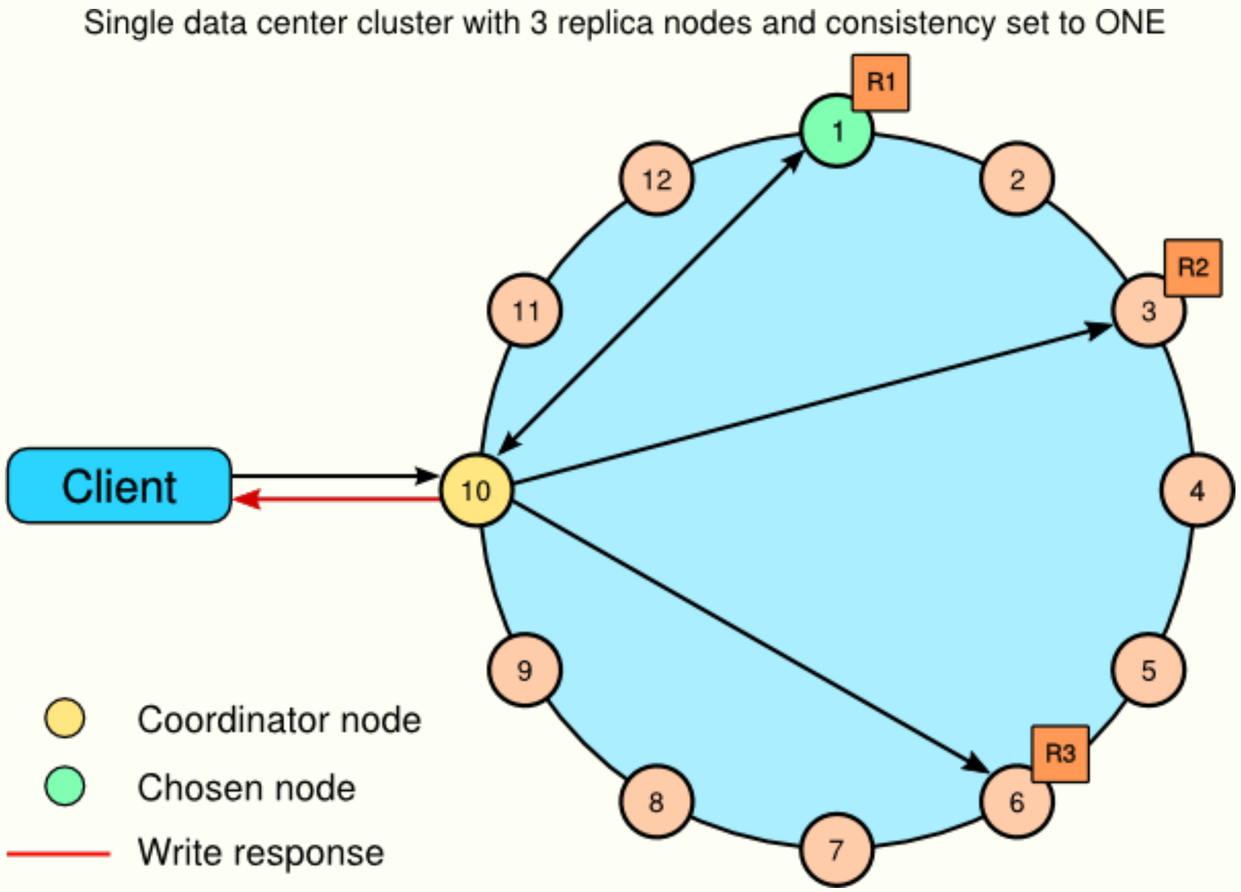
- MASTER-MASTER
- COORDINATOR
- QUERY “PROXYING”



http://docs.datastax.com/en/cassandra/2.0/cassandra/images/arc_write-singleDCConOne.sv

REPLICATION FACTOR

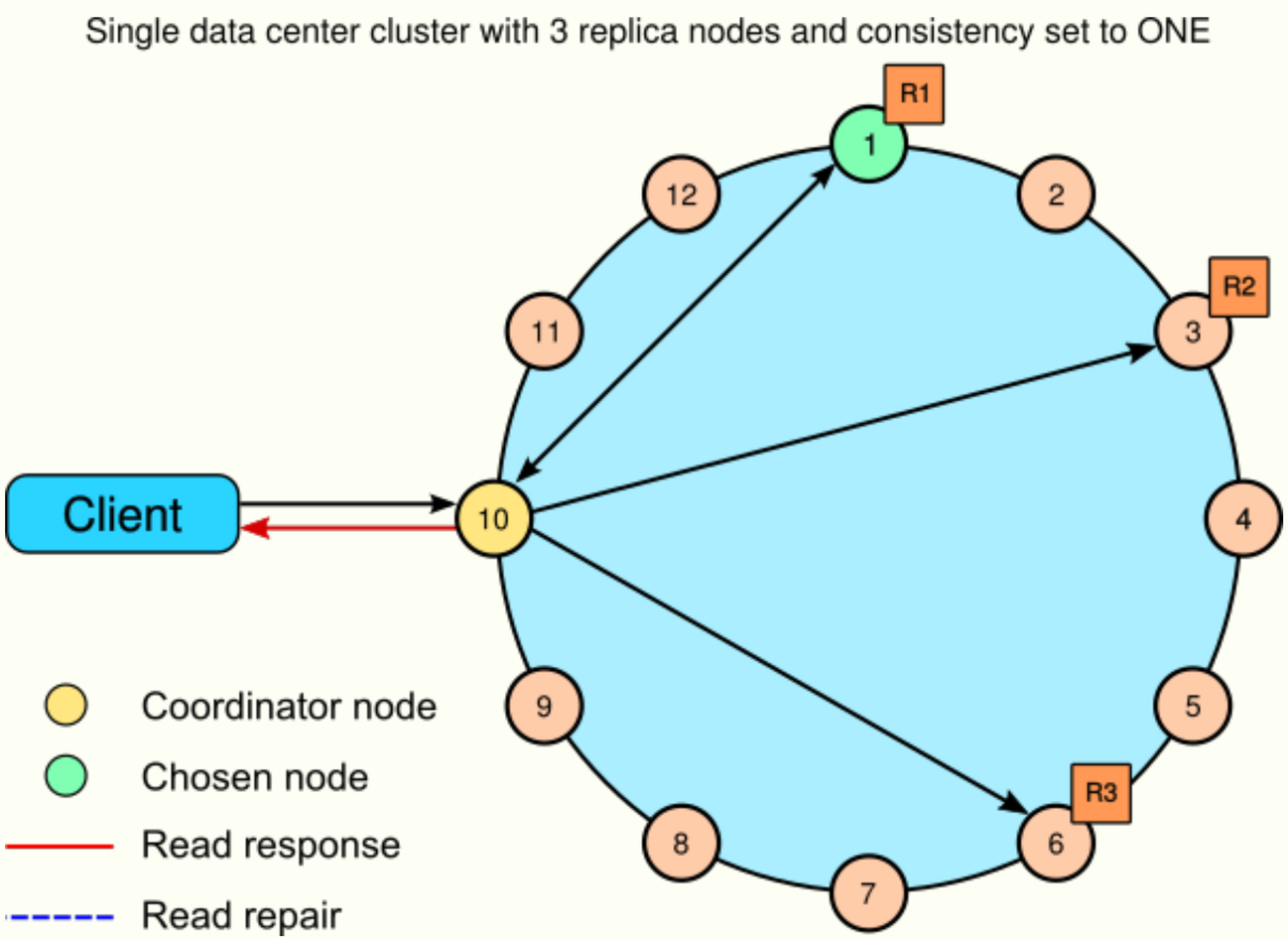
- RF =TOTAL NUMBER OF REPLICAS
  - REPLICATION STRATEGY



http://docs.datastax.com/en/cassandra/2.0/cassandra/images/arc_write-singleDCConOne.sv

CONSISTENCY LEVEL

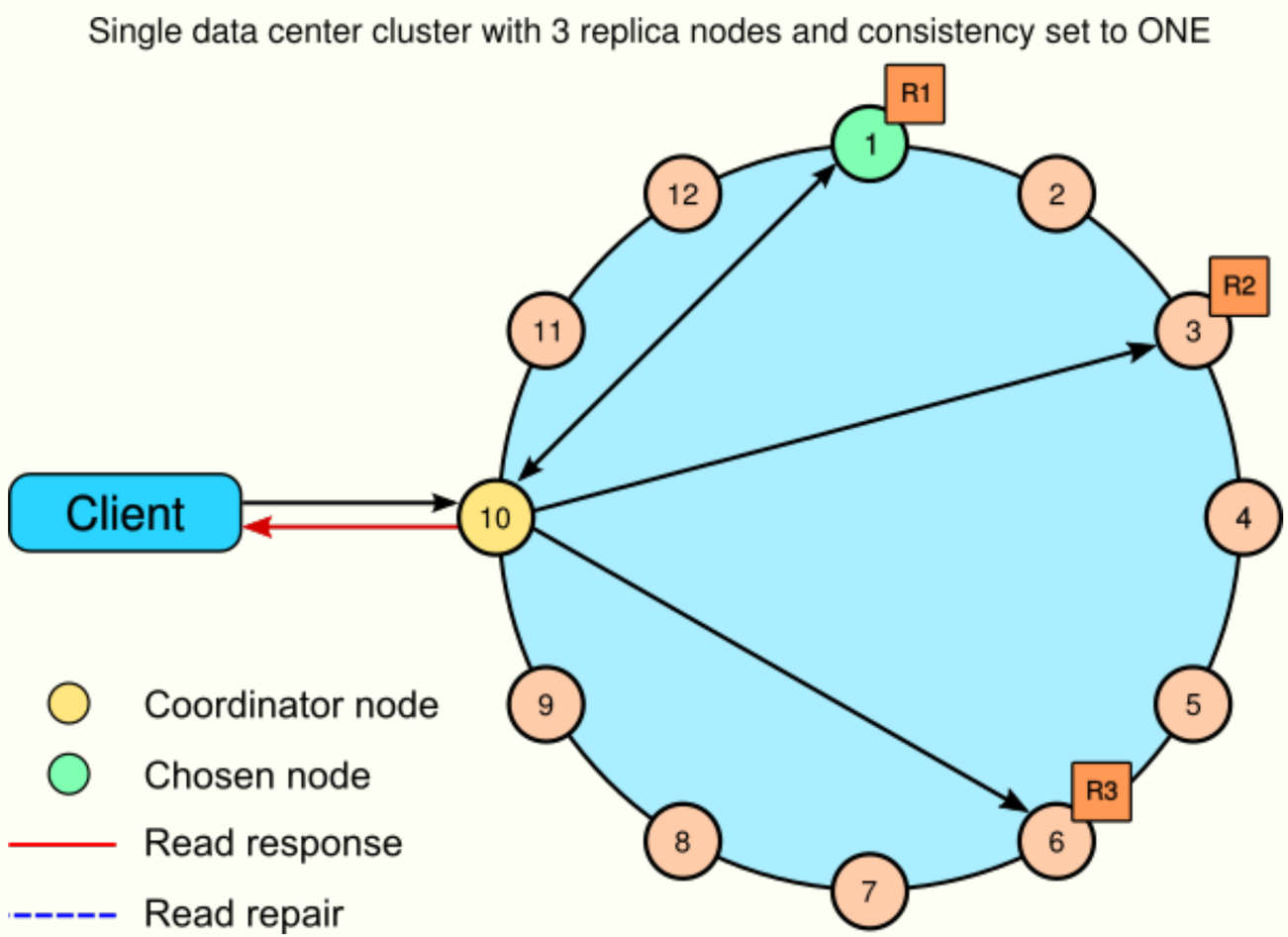
- TUNABLE CONSISTENCY
  - READ
  - WRITE



http://docs.datastax.com/en/cassandra/2.0/cassandra/images/arc_write-singleDCConOne.sv

CONSISTENCY LEVEL

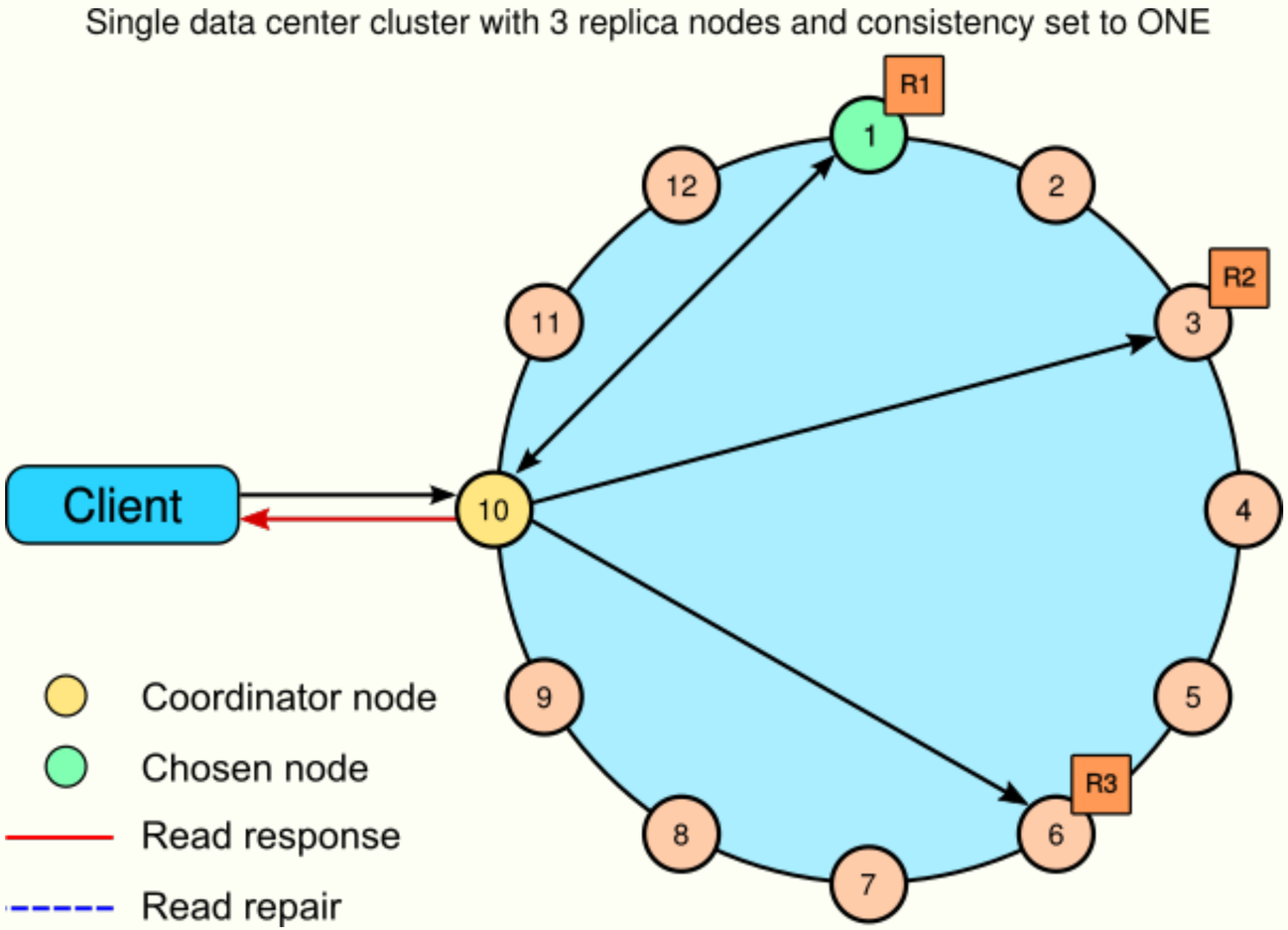
- READ REQUEST
  - DIGEST QUERY
  - REPAIRED READ
- CL=1



http://docs.datastax.com/en/cassandra/2.0/cassandra/images/arc_write-singleDCConOne.sv

READ

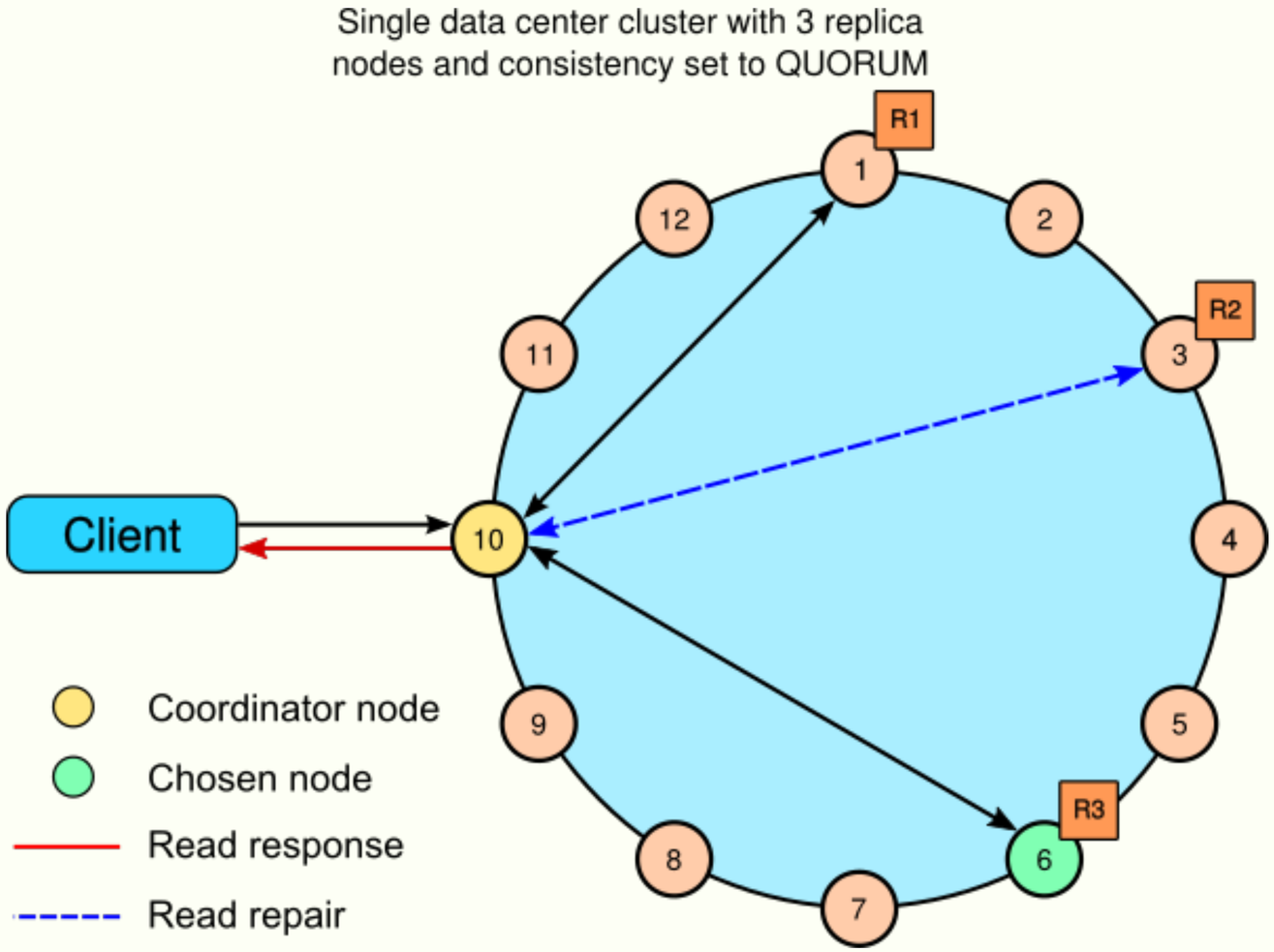
- RF = 3
- CL=1



http://docs.datastax.com/en/cassandra/2.0/cassandra/images/arc_write-singleDCConOne.sv

READ

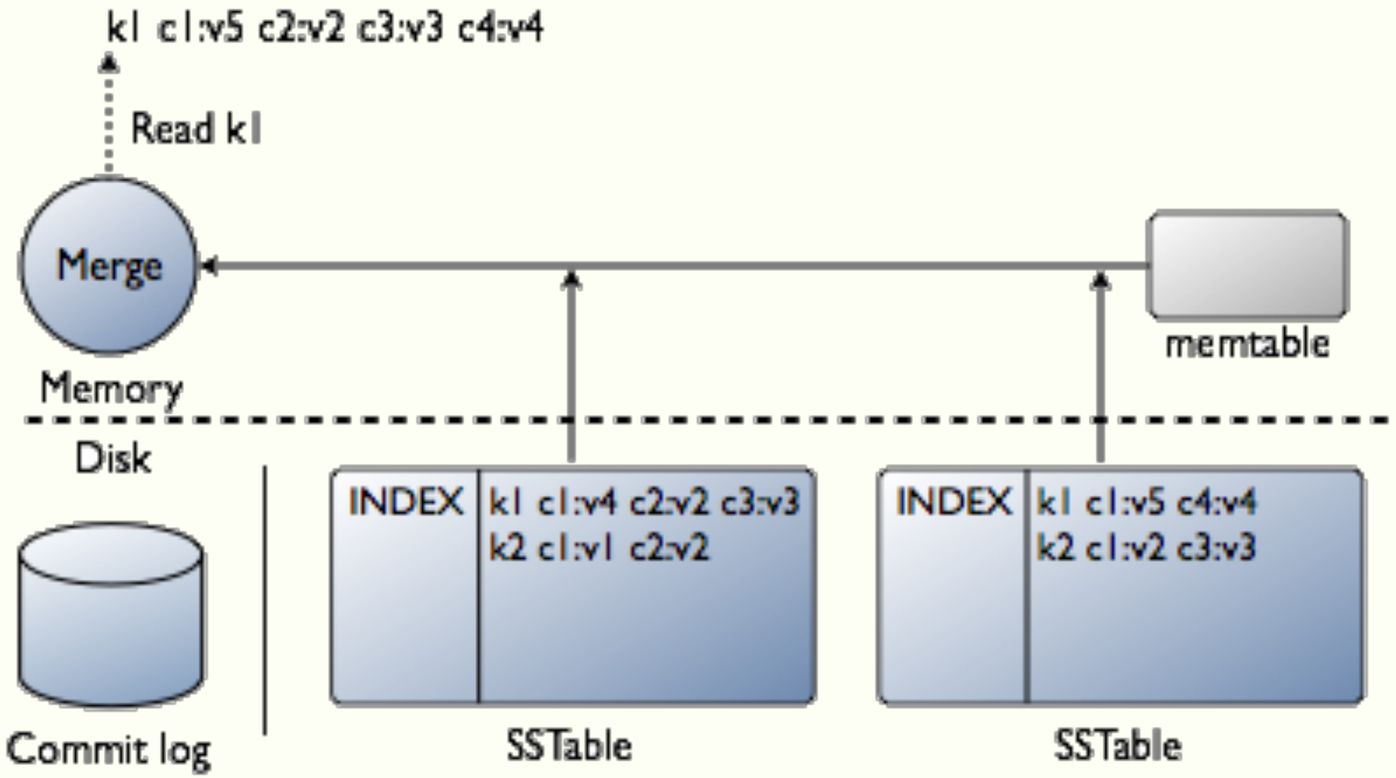
- RF = 3
- CL= QUORUM



http://docs.datastax.com/en/cassandra/2.0/cassandra/images/arc_write-singleDCConOne.sv

STORAGE CONCEPTS

- READS



http://docs.datastax.com/en/cassandra/1.2/cassandra/dml/images/read_ks_12.png

ANTI ENTROPY

- HINTED HANDOFF
- READ REPAIR
- FULL VS. INCREMENTAL REPAIR

USE CASES - NETFLIX

- 90 CLUSTERS, 2700 NODES, 4 DC
- NEARLY ALL DATA
  - FILM METADATA
  - USER RATINGS
  - RECOMMENDATIONS

GRAPH DATABASES

NOT A SWISS-ARMY KNIFE


ANY QUESTIONS?

THANK YOU

@TOM BUJOK