Banco de dados NoSQL - Cassandra (Introdução)

Prof. Gustavo Leitão

AGENDA

- Introdução ao Cassandra
- Instalação com Docker
- Comandos básicos
- Operações com CQL

- Cassandra é um banco de dados open-source distribuído e decentralizado altamente escalável projetado para lidar com volume extremamente grande de dados

- Cassandra possui alta disponibilidade e não possui um ponto único de falha

Características

- Horizontalmente escalável
- Orientado a coluna
- Arquitetura baseada no Amazon Dynamo e estrutura de dados baseada no Google Big Table
- Escrito em Java

.com Houghton Mifflin Harcourt Morningstar platform46 accenture adform adku Adobe ADXPOSE AGENTIS aiMatch APIXIO

applifier AppScale appsavvy Apptegic ARKIVUM Assistly backupify BRIGHT EDGE Bronto Burt. CallFire Catalyst SGC

cisco citypath clearspring cloudkick CLOUDTALK clover Constant Contact Dachis Group DATADOG D Datagraph, Inc. Demandforce despegar.com

digg Digital Reasoning Digital River dotomi druva ebay buddy egnyte embed.ly Engine Yard enkata ERICSSON

Expedia FanBridge Fight My Monster FINN formspring Frugal Mechanic GEMINI GIGASPACES GOTiME Gowalla GRIDPOINT

Healthline Health Market Science Healthx hobsons hp HubSpot IBM impetus inkling iovation ISIDOREY jetpac

Jumptap junction networks justSignal librato LiveOffice LIVEPERSON Mahalo martini media media6degrees MEDIALETS MedProcure MolloM

monitis movile napshare NAVTEQ NETFLIX Nextag NEXT BIG SOUND NextSlide ngmoco:) nicira ONEsite one spot OOYALA

OpenAdExchange OpenFeint opentracker OPENWAVE Other Inbox outbrain page2rss Palantir PANTHEON Pitney Bowes PBS KIDS PCPhase PeopleBrowsr

plaxo POLYVORE PostRank PRESS+ proofpoint quantiFind QUOVA rackspace Rakuten RALLY ReachLocal real

reddit referentia re tickr Rhapsody RockMelt rockyou Ruckus Scandit SEOmoz SEVEN SHAZAM shopsavvy

SimpleGeo skillpages smeet SOCIALFLOW SPI sonico SOUNDCLOUD spidertracks Spotify Spredfast springpad SQUIDOO Symantec

Tapulous TEADS Tellybug TENDRIL thefind trap!t TWIMPACT twitter Urban Airship usergrid VigLink Visual DNA

VTEX wajam we R where WSO2 Wupa xobni Yakaz YCHARTS ZipZapPlay zoominfo.

Quem usa?

Apple atualmente possui

75.000 nós Cassandra

armazenando **10 petabytes**

Fonte: cassandra.apache.org

Quem usa?

Netfix utiliza Apache Cassandra para o stream de videos.
Atualmente são **2.500** nós armazenando **420 terabytes** de dados e respondendo **1 trilhão** de requisições por dia

Fonte: cassandra.apache.org

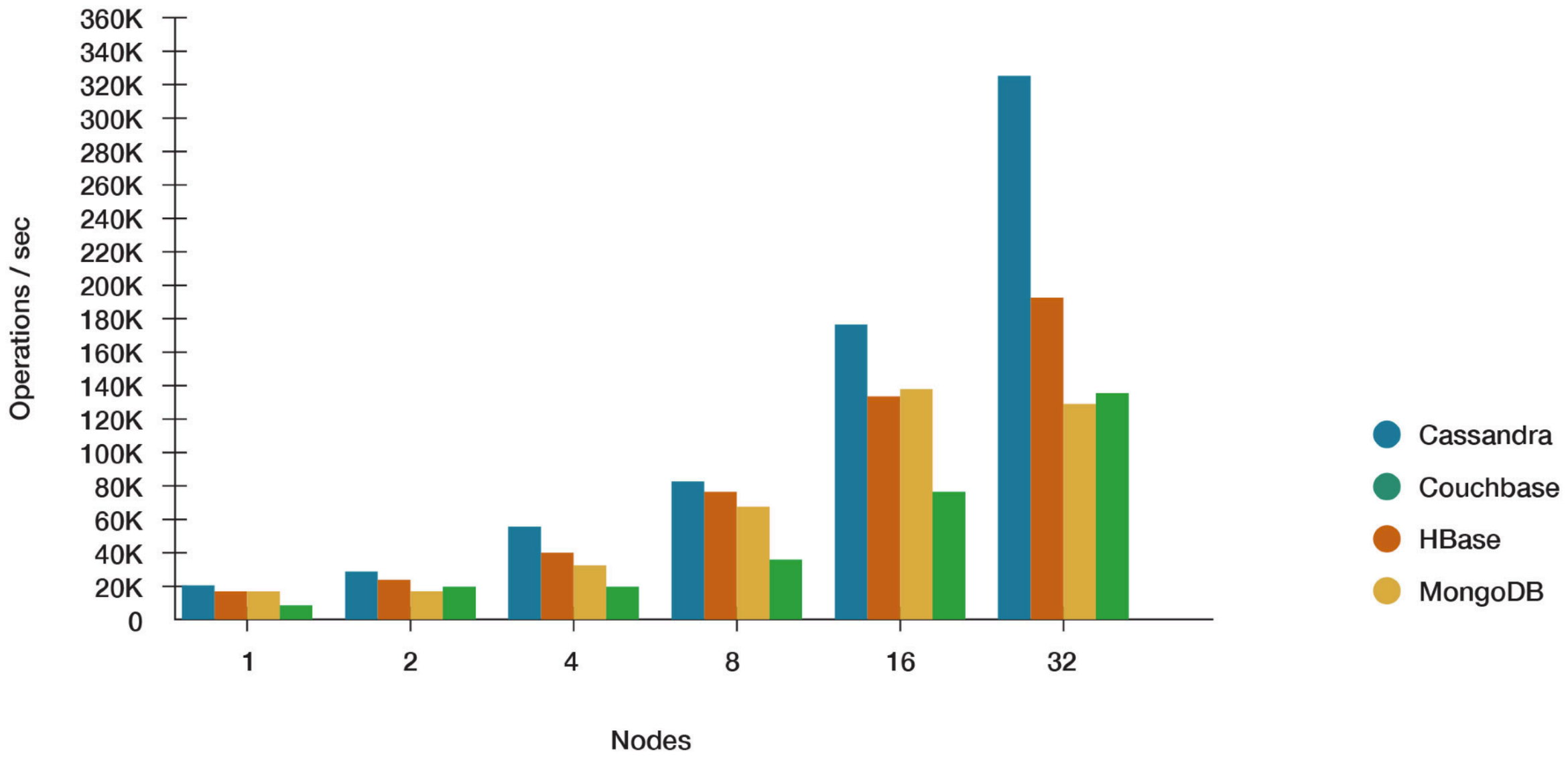
Quem usa?

Ebay atualmente possui 100 nós Apache Cassandra armazenando 250Tb de dados

Fonte: cassandra.apache.org

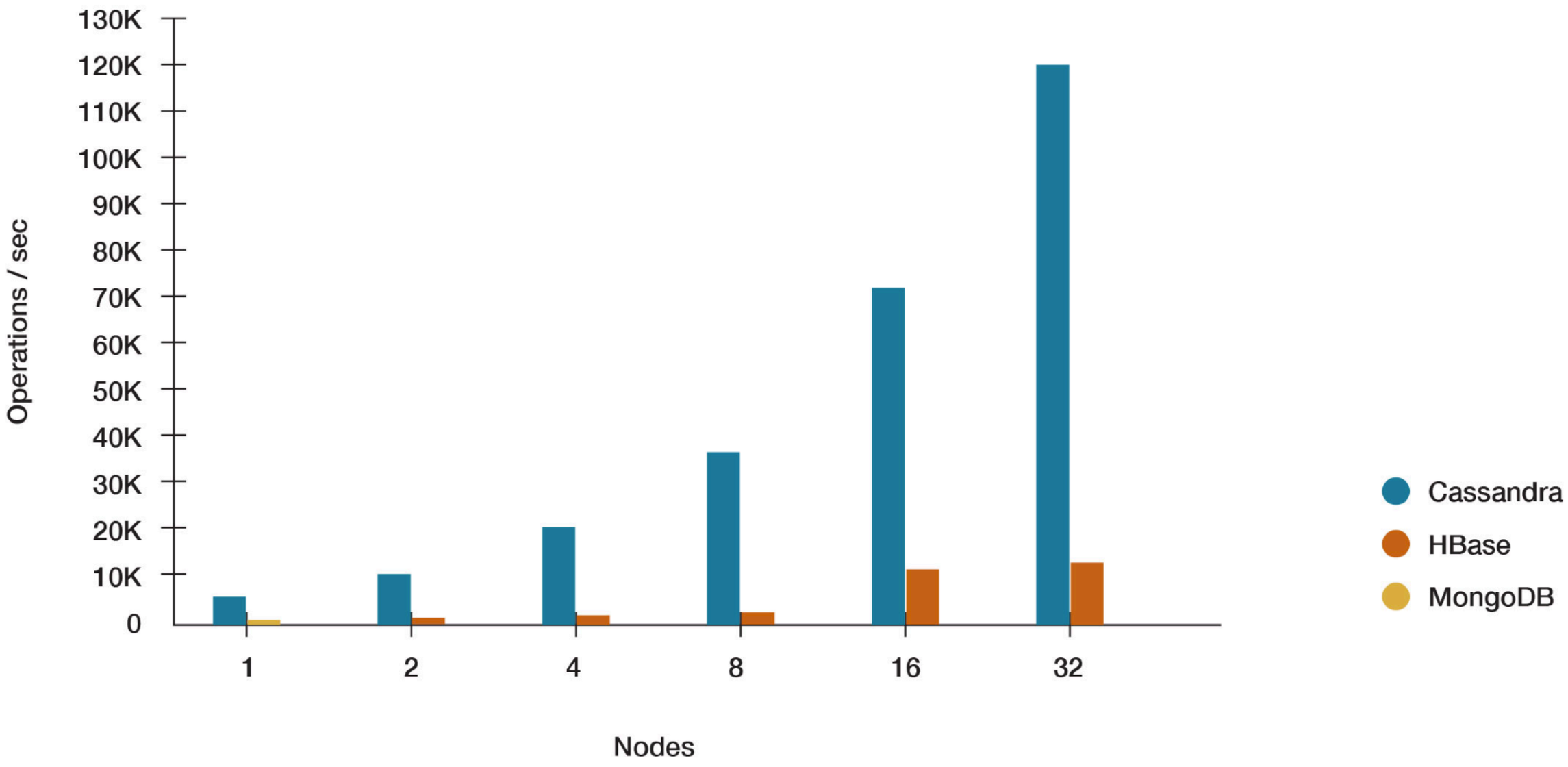
Benchmarking

Inserting…

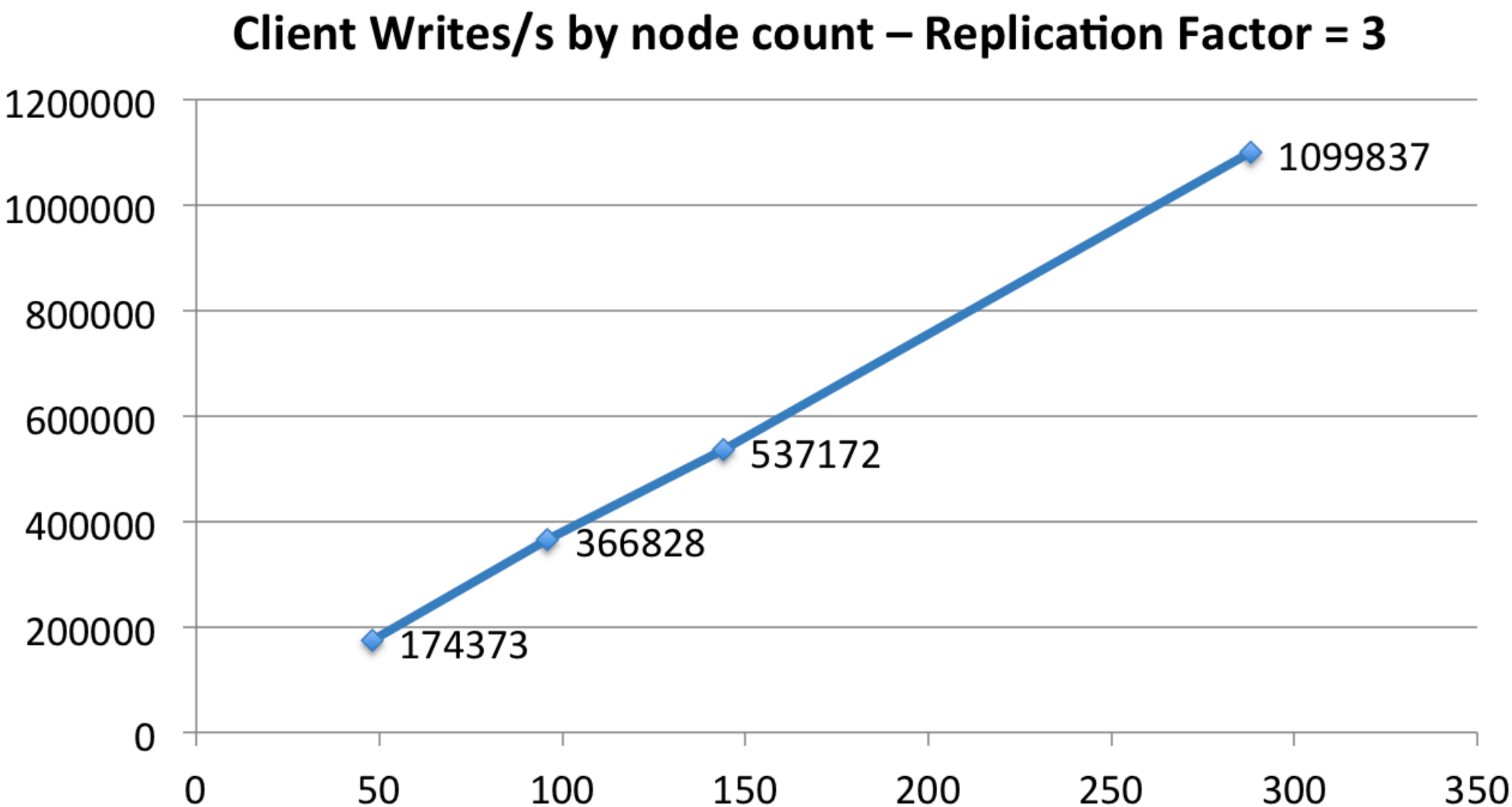


Benchmarking

Operações diversas



Scale-Up Linearity



https://medium.com/netflix-techblog/benchmarking-cassandra-scalability-on-aws-over-a-million-writes-per-second-39f45f066c9e

NETFLIX

Time is Money

	48 nodes	96 nodes	144 nodes	288 nodes
Writes Capacity	174373 w/s	366828 w/s	537172 w/s	1,099,837 w/s
Storage Capacity	12.8 TB	25.6 TB	38.4 TB	76.8 TB
Nodes Cost/hr	$32.64	$65.28	$97.92	$195.84
Test Driver Instances	10	20	30	60
Test Driver Cost/hr	$20.00	$40.00	$60.00	$120.00
Cross AZ Traffic	5 TB/hr	10 TB/hr	15 TB/hr	30[1] TB/hr
Traffic Cost/10min	$8.33	$16.66	$25.00	$50.00
Setup Duration	15 minutes	22 minutes	31 minutes	66[2] minutes
AWS Billed Duration	1hr	1hr	1 hr	2 hr
Total Test Cost	$60.97	$121.94	$182.92	$561.68

[1] Estimate two thirds of total network traffic
[2] Workaround for a tooling bug slowed setup

https://medium.com/netflix-techblog/benchmarking-cassandra-scalability-on-aws-over-a-million-writes-per-second-39f45f066c9e

NETFLIX

Funcionalidades

- **Escalabilidade:** Cassandra é altamente escalável. Permite adicionar hardware para acomodar mais dados

- **Arquitetura decentralizada:** Cassandra não possui um ponto único de falha. Todos os nós tem as mesmas responsabilidades

- **Escalabilidade Linear:** Cassandra é linearmente escalável. Ou seja, aumenta a vazão de operações linearmente com a quantidade de nós na rede.

- **Estrutura flexível:** Salva diversos tipos de dados sejam eles estruturados ou não. Permite mudanças no modelo dinamicamente

Funcionalidades

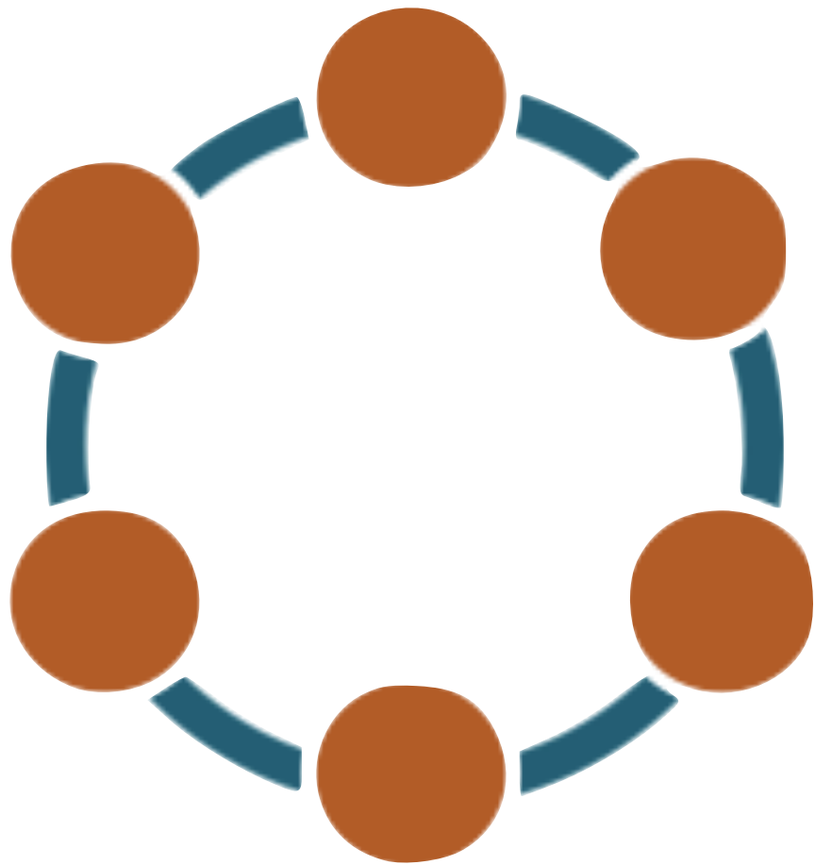
- **Distribuição dos dados:** Os dados são replicados entre os nós do cluster através da definição do fator de replicação.

- **Suporte a transação:** Cassandra suporta a execução de transações (segue princípios ACID)

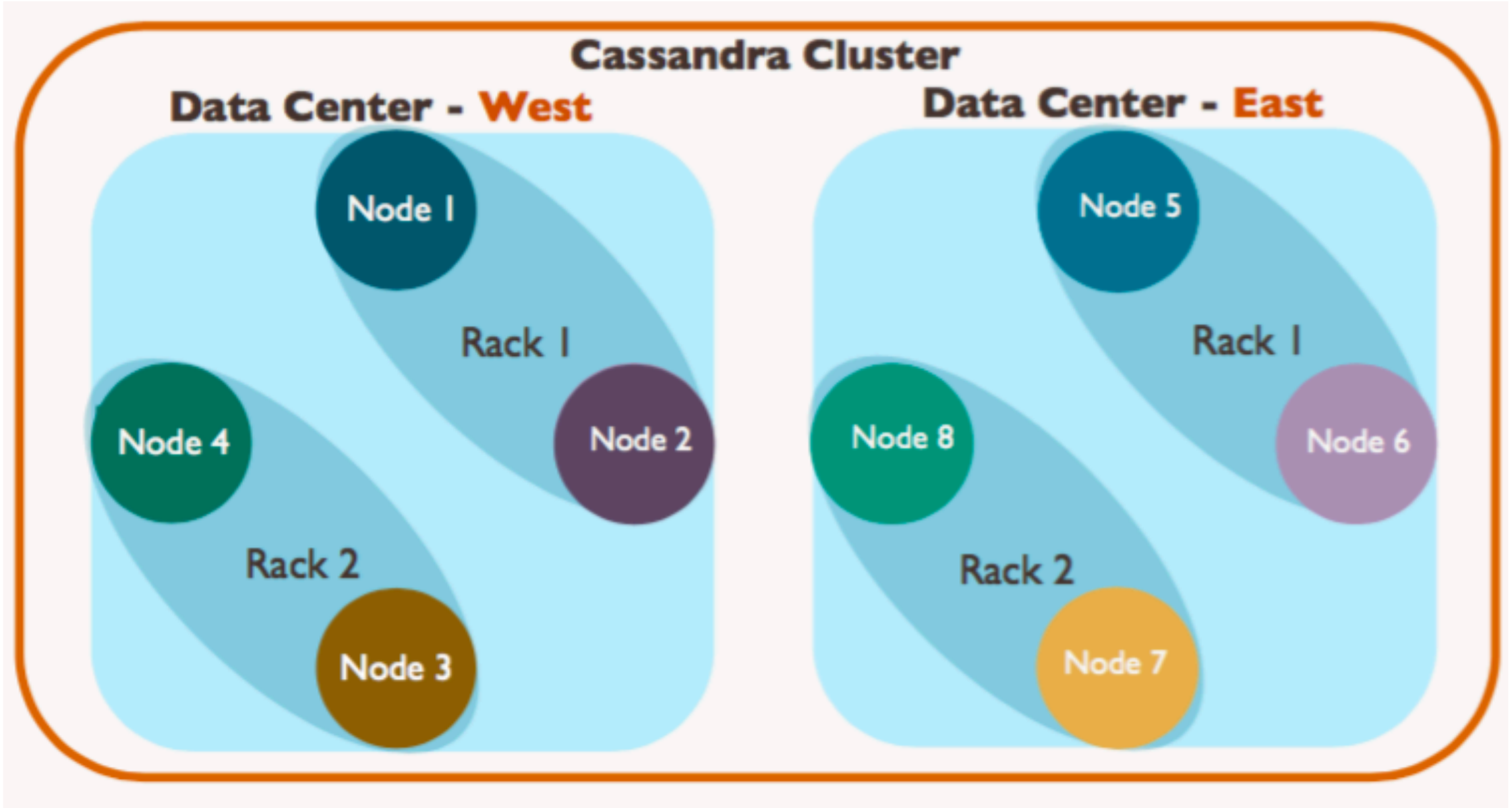
- **Escrita Rápida:** Cassandra possui escritas rápidas mesmo com hardwares simples sob milhares de dados.

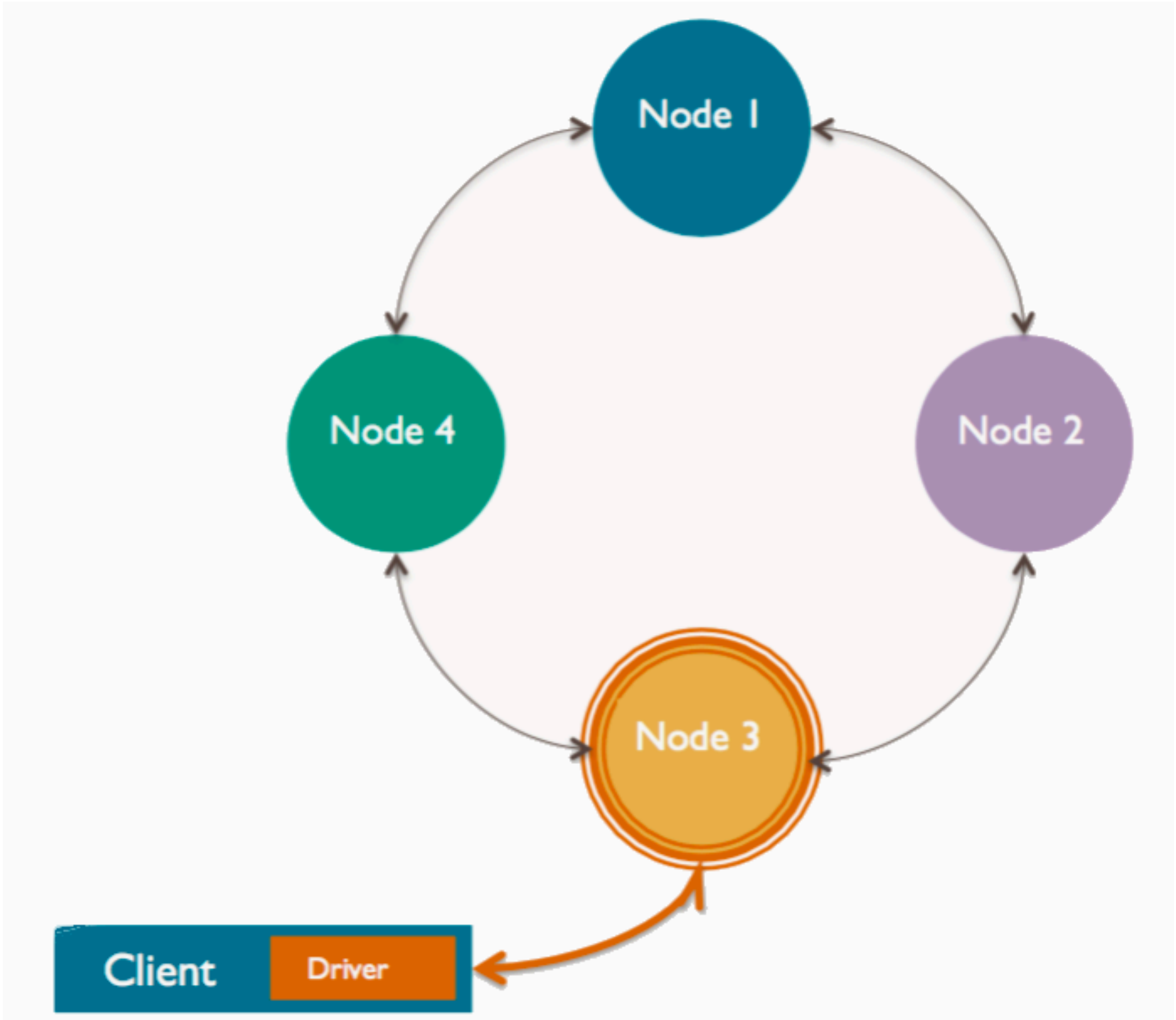
Arquitetura

- Sistema distribuído p2p
- Todos os nós são iguais
- Os dados são particionados pelos nós
- Replicação de dados customizável para melhorar disponibilidade
- Leia/Escreva em qualquer lugar

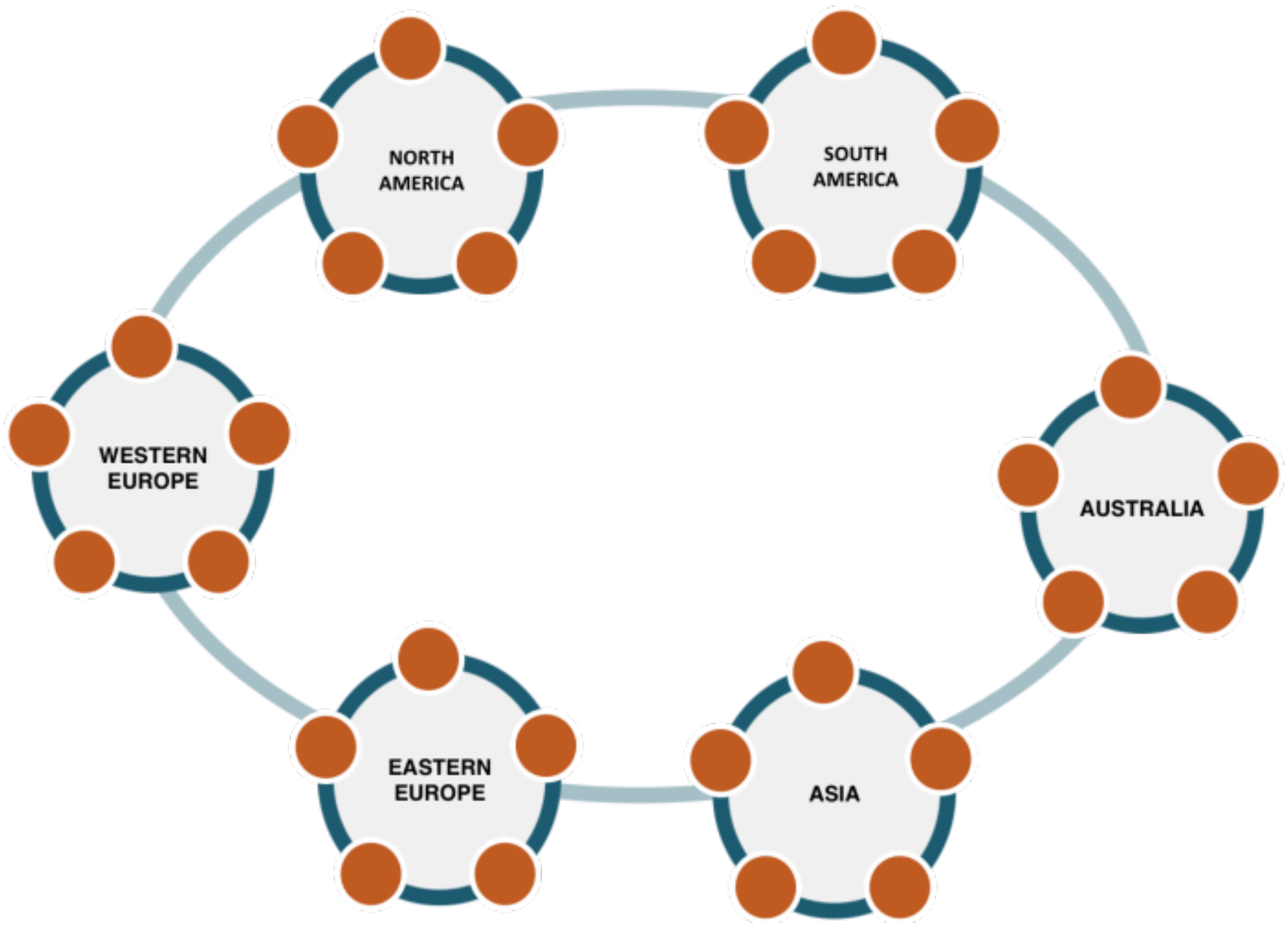


Arquitetura

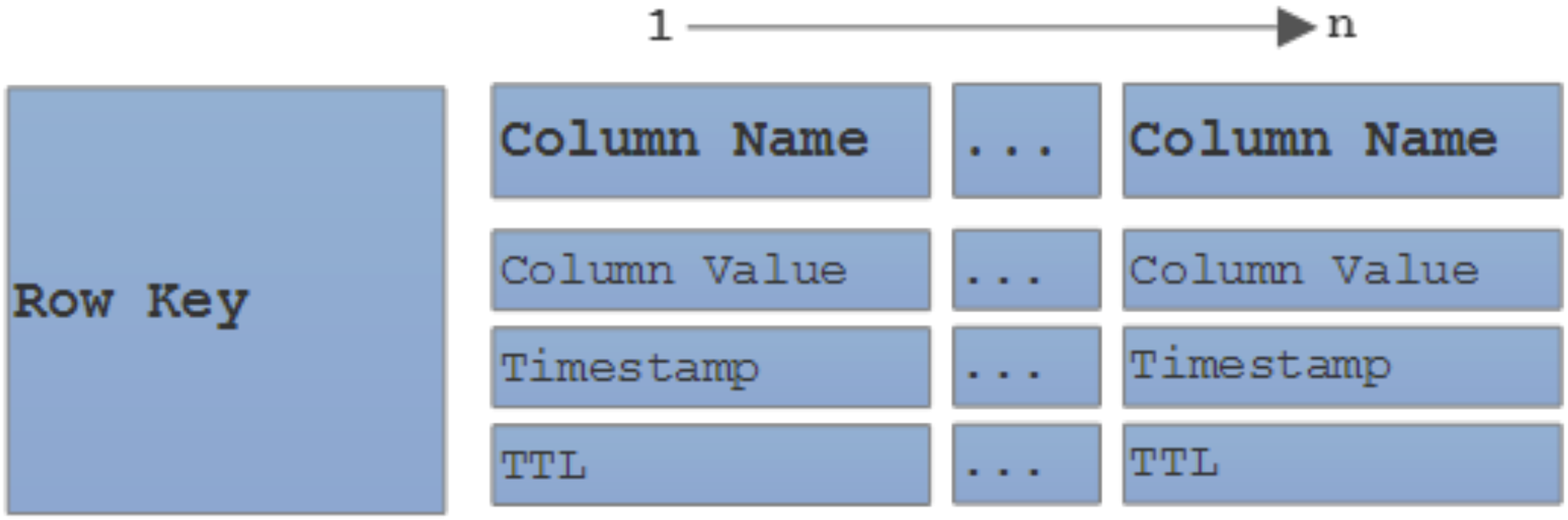



Node 1
Node 4
Node 2
Node 3
Client
Driver

Arquitetura

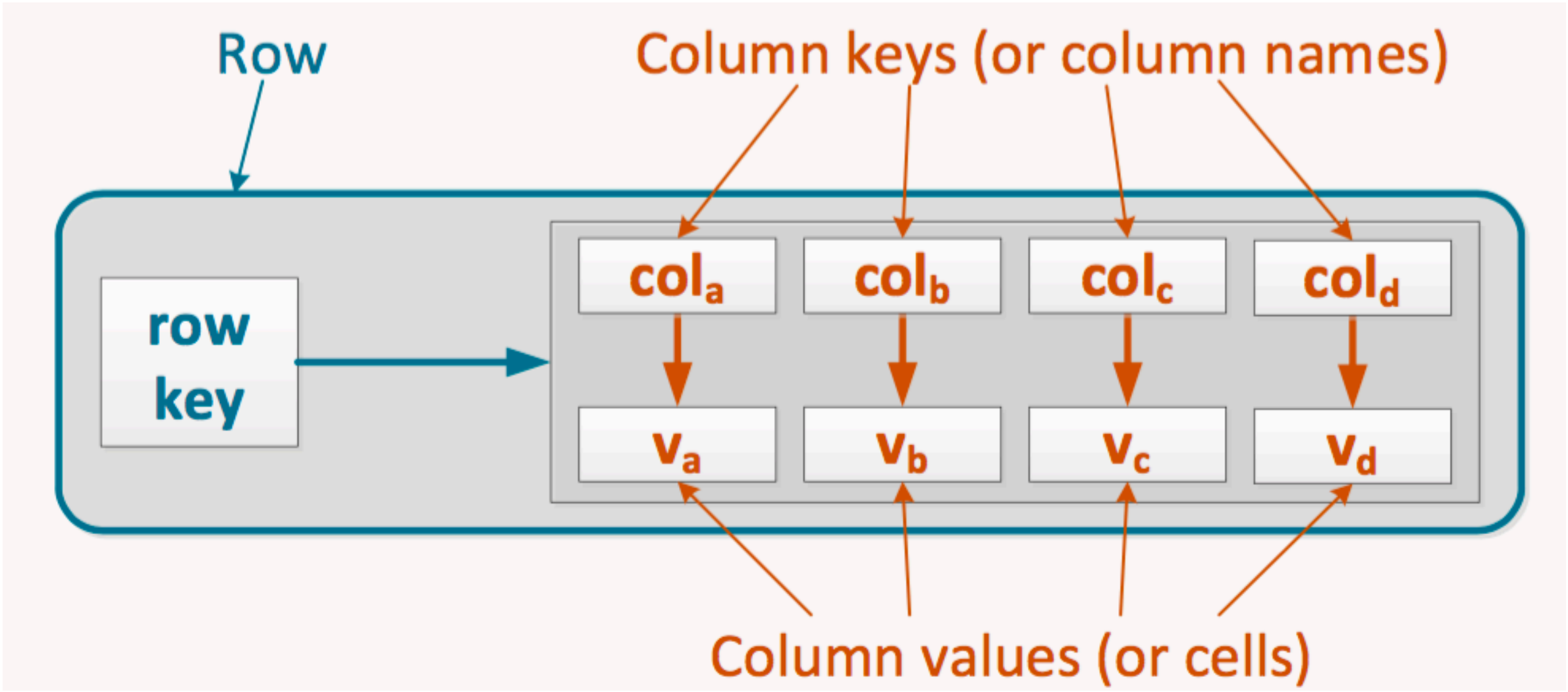


Modelo de Dados



TTL (Time To Live) - Tempo de vida da coluna

Modelo de Dados



COLUMNS

ROWS

row	col_a	col_b	col_c	col_d
key_3	$v_{3.a}$	$v_{3.b}$	$v_{3.c}$	$v_{3.d}$
key_1	$v_{1.a}$	$v_{1.b}$	$v_{1.c}$	$v_{1.d}$
key_2	$v_{2.a}$	$v_{2.b}$	$v_{2.c}$	$v_{2.d}$

CELLS

SGBD	Cassandra
Database	Keyspace
Table	Column Family
Linha	Linha
Primary Key	Row Key

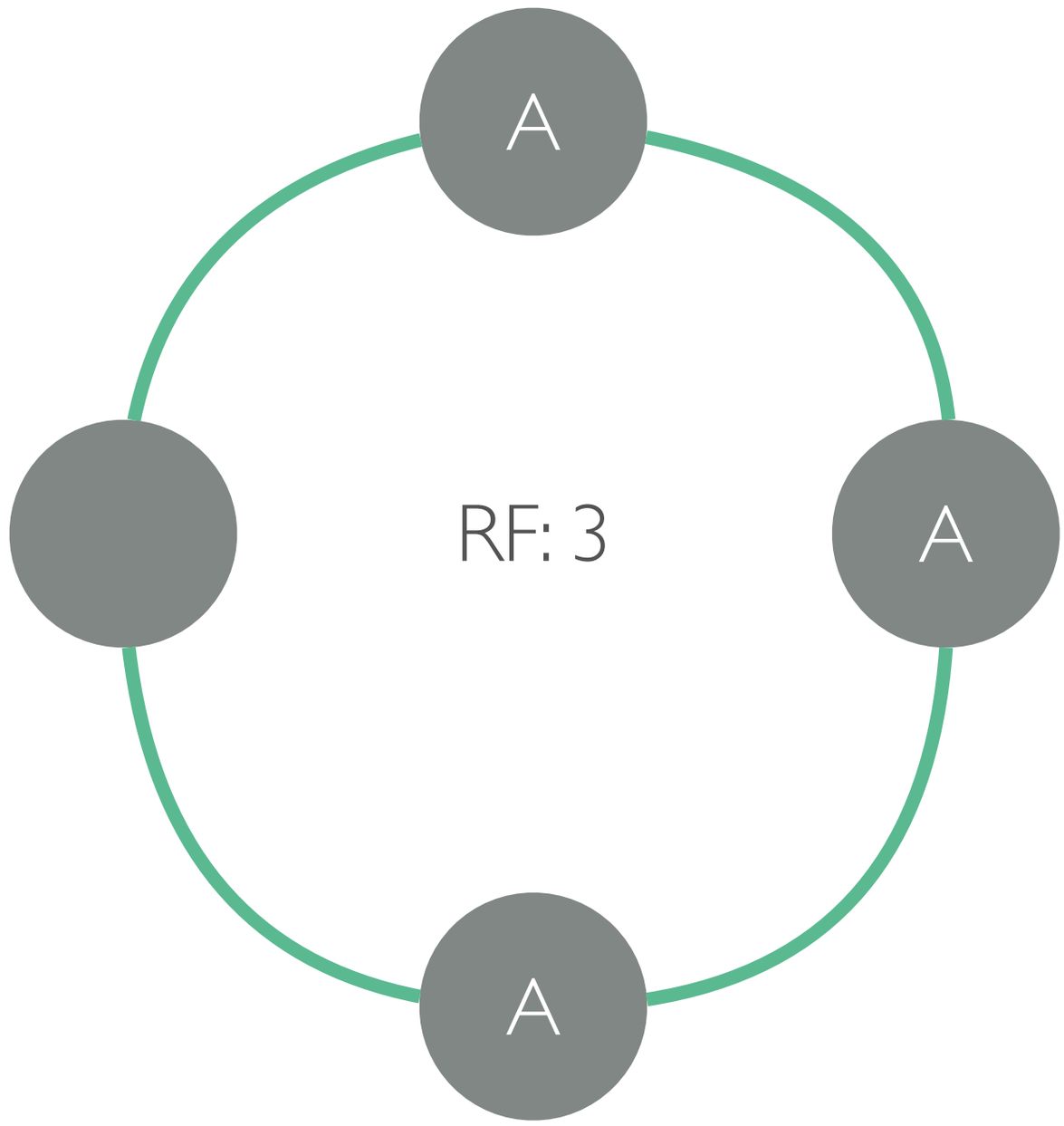
Por que cassandra?

- Altamente escalável
- Ganhos de desempenho linear através da adição de nós
- Nenhum ponto único de falha
- Fácil replicação / distribuição de dados
- Multi-data center e Cloud
- Não há necessidade de camada de cache separada
- Consistência de dados ajustáveis
- Esquema flexível do esquema
- Compressão de dados
- Linguagem CQL (como SQL)
- Não há necessidade de hardware ou software especial

Replicação

- Quantas cópias de cada parte do dado serão feitas?

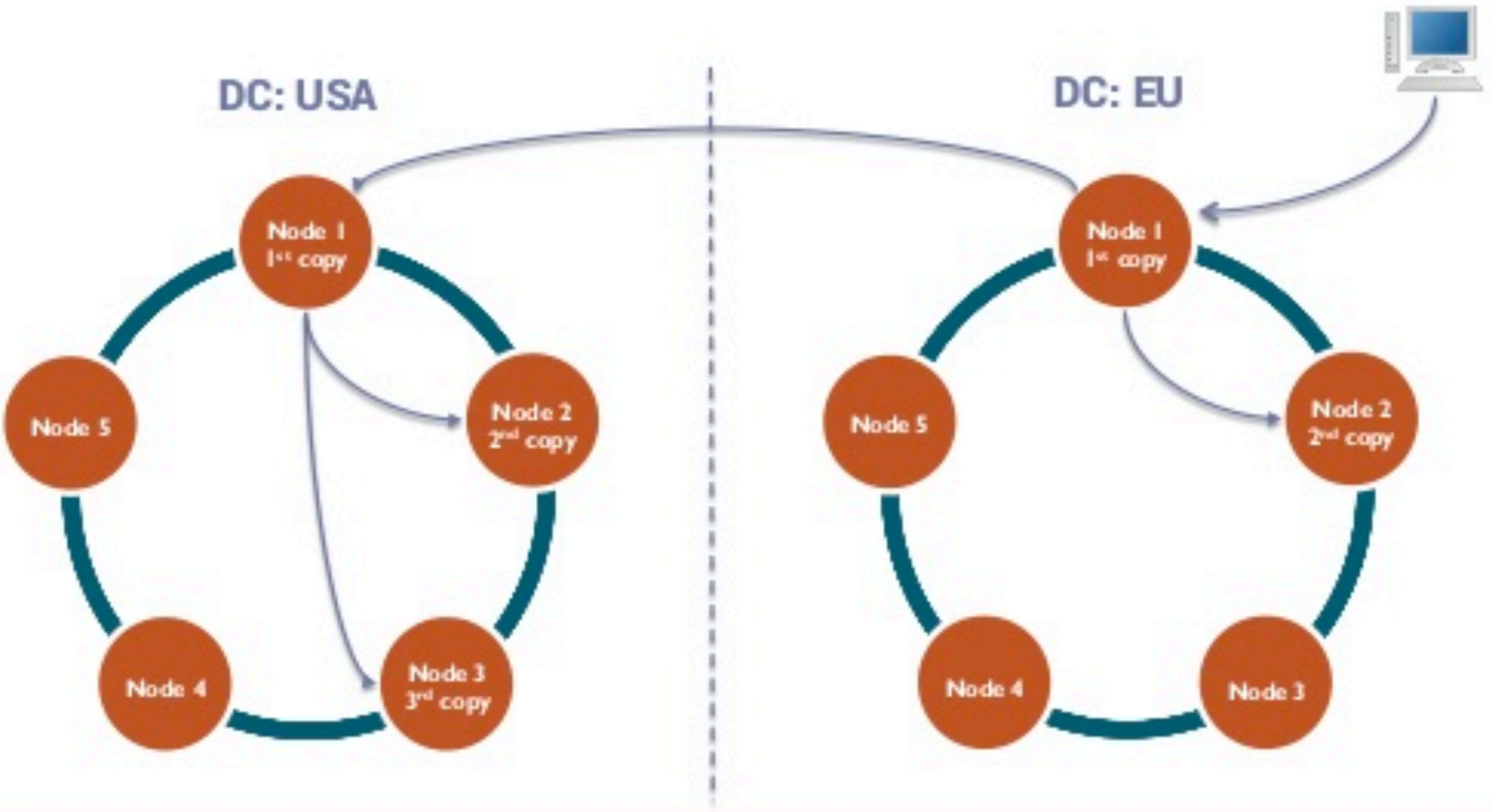
- O fator de replicado pode ser definido por data center



Creating A Keyspace

DATASTAX

```
CREATE KEYSPACE johnny WITH REPLICATION =
{'class':'NetworkTopologyStrategy', 'USA':3, 'EU': 2};
```



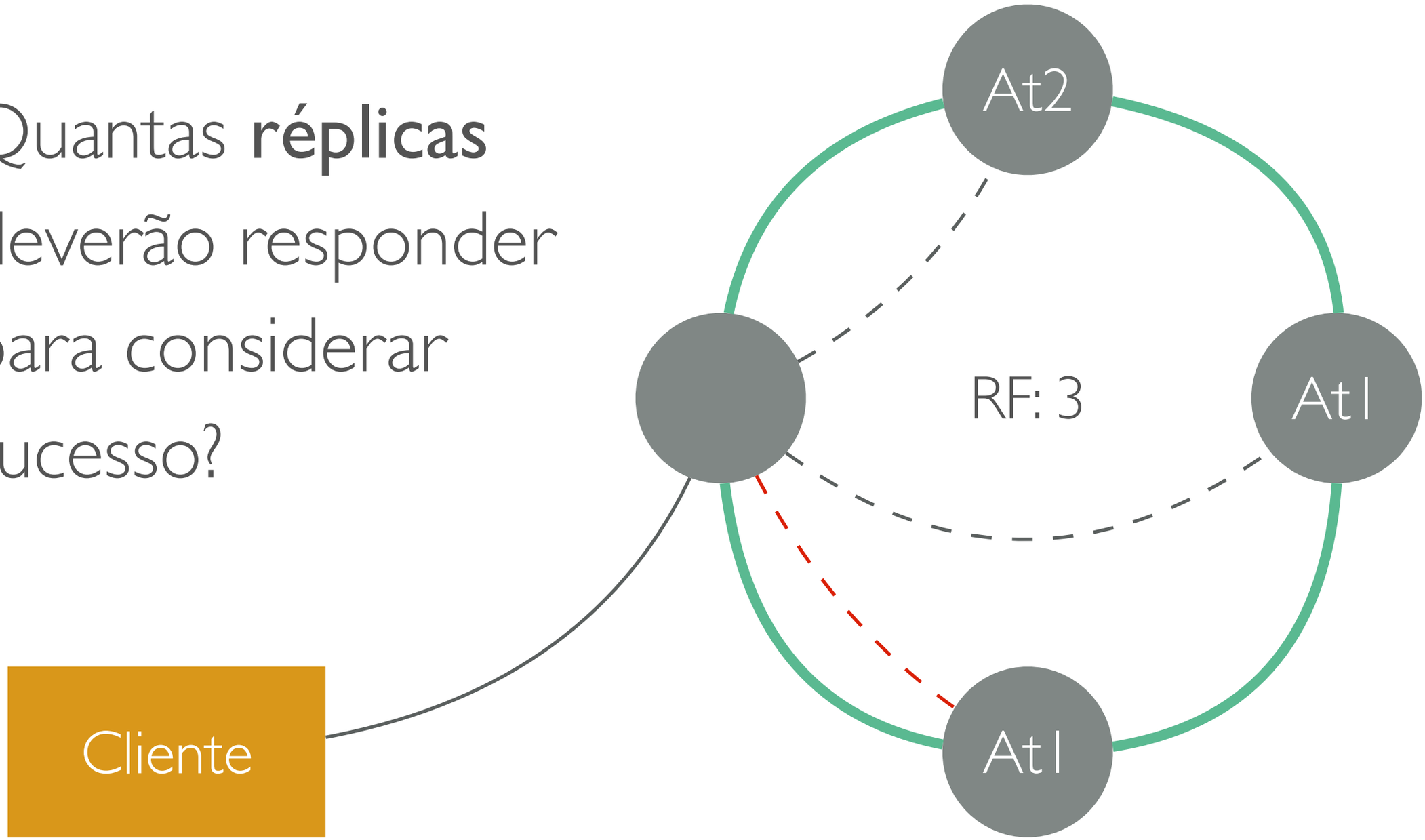
Estratégias de Replicação

- **SimpleStrategy:** use apenas para um único datacenter e um rack. Se você planeja mais de um datacenter, use o NetworkTopologyStrategy.

- **NetworkTopologyStrategy:** recomendado para a maioria das implantações, porque é muito mais fácil expandir para vários datacenters quando exigido por futuras expansões.

Consistência

- Quantas **réplicas** deverão responder para considerar sucesso?



Consistência Leitura

Consistência	Descrição
ONE	Retorna a resposta da réplica mais próxima (em seguida read-repair)
LOCAL_ONE	Retorna a resposta da réplica mais próxima no mesmo DC
TWO	Retorna o dado mais recente de duas das réplicas
THREE	Retorna o dado mais recente de três das réplicas
QUORUM	Retorna depois de obter resposta de um quorum de qualquer DC
LOCAL_QUORUM	Retorna depois de obter resposta de um quorum no DC
ALL	Retorna após todas as replicas responderem

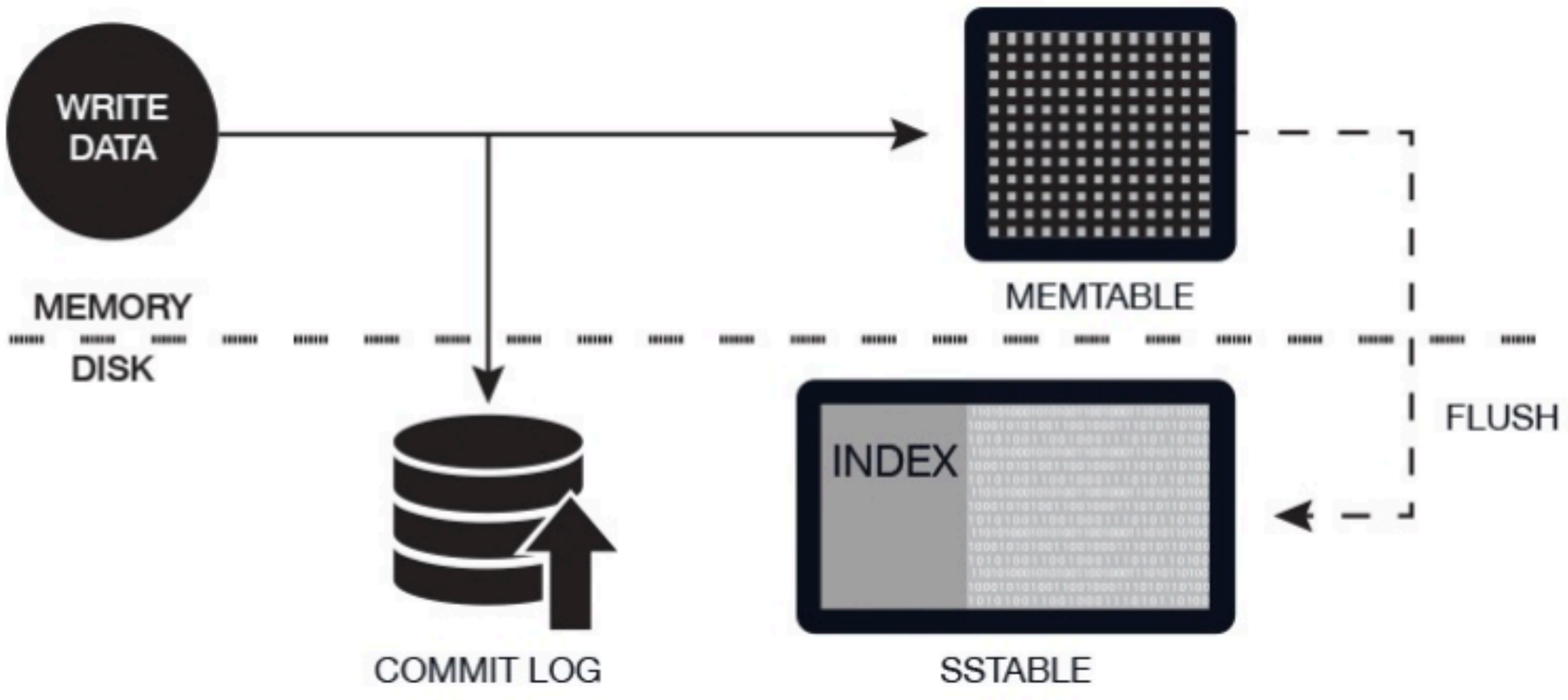
Consistência Escrita

Consistência	Descrição
ANY	A escrita deve retornar sucesso se qualquer nó reportar sucesso
ONE	Pelo menos uma réplica deve retornar sucesso
LOCAL_ONE	Pelo menos uma réplica no mesmo DC deve retornar sucesso
TWO	Pelo menos duas réplicas devem retornar sucesso
THREE	Pelo menos três réplicas devem retornar sucesso
QUORUM	Deve retornar sucesso por um quorum de réplicas ((RF/2)+1)
LOCAL_QUORUM	Deve retornar sucesso por um quorum de réplicas no mesmo DC
EACH_QUORUM	Deve retornar sucesso por um quorum de réplicas em cada DC
ALL	Todas as replicas devem retornar sucesso

Componentes

- **Nó:** onde os dados são armazenados. Case da arquitetura.
- **Datacenter:** Coleção de nós relacionados
- **Cluster:** Contém um ou mais datacenters. Os datacenters podem estar fisicamente separados
- **Commit log:** Todos os dados são salvos inicialmente no commit log para garantir durabilidade. Depois os dados são salvos no SSTable para arquivamento.
- **SSTable:** Sorted String Table é um arquivo imutável onde o Cassandra armazena os dados periodicamente.

Escrita


WRITE DATA
MEMORY
DISK
MEMTABLE
FLUSH
INDEX
COMMIT LOG
SSTABLE

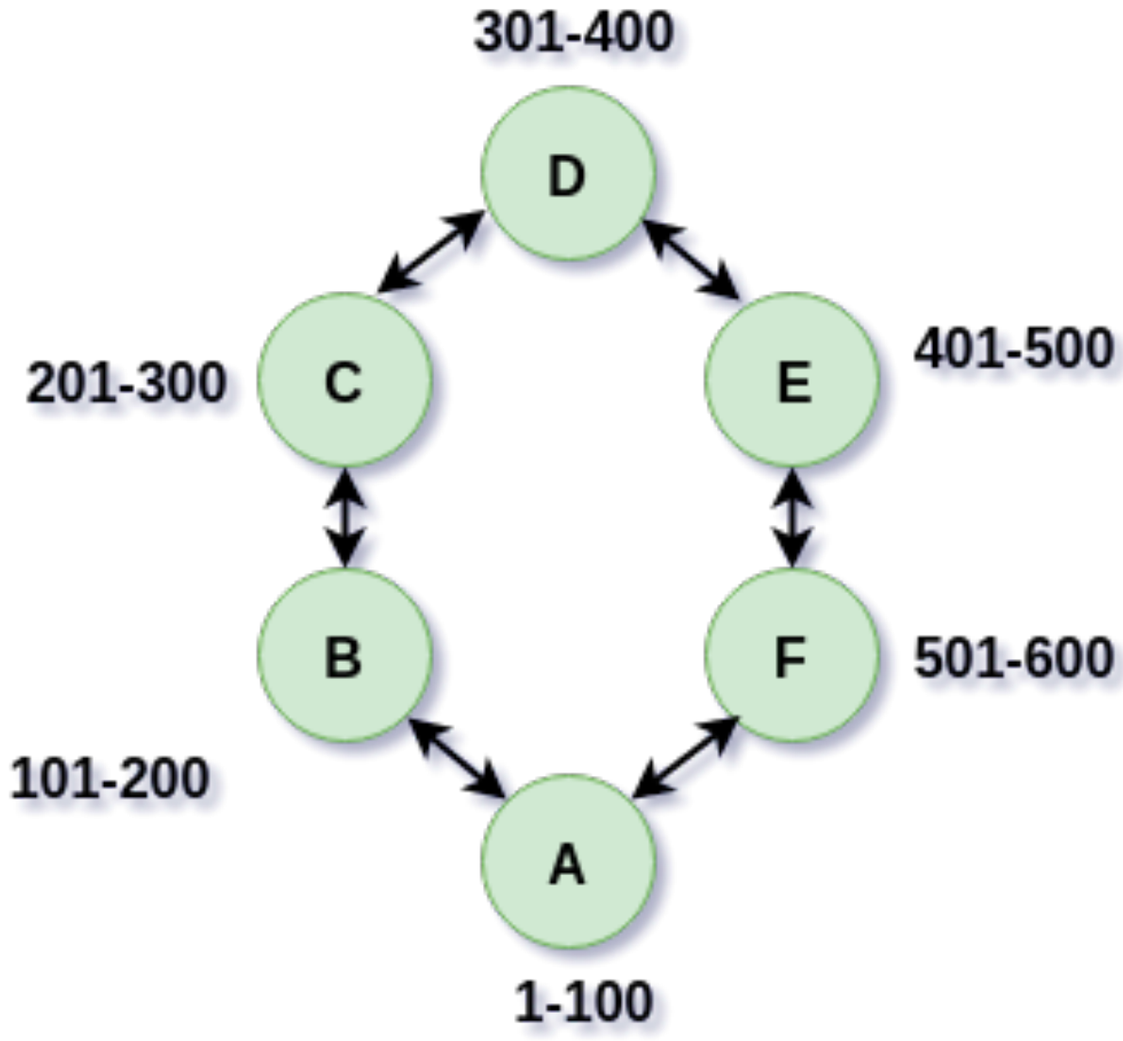
CONCEITOS

- Partition Key —> Como os dados vão ser distribuídos. Cada partição residirá em um nó. Limite de 2 bilhões de colunas por partição.

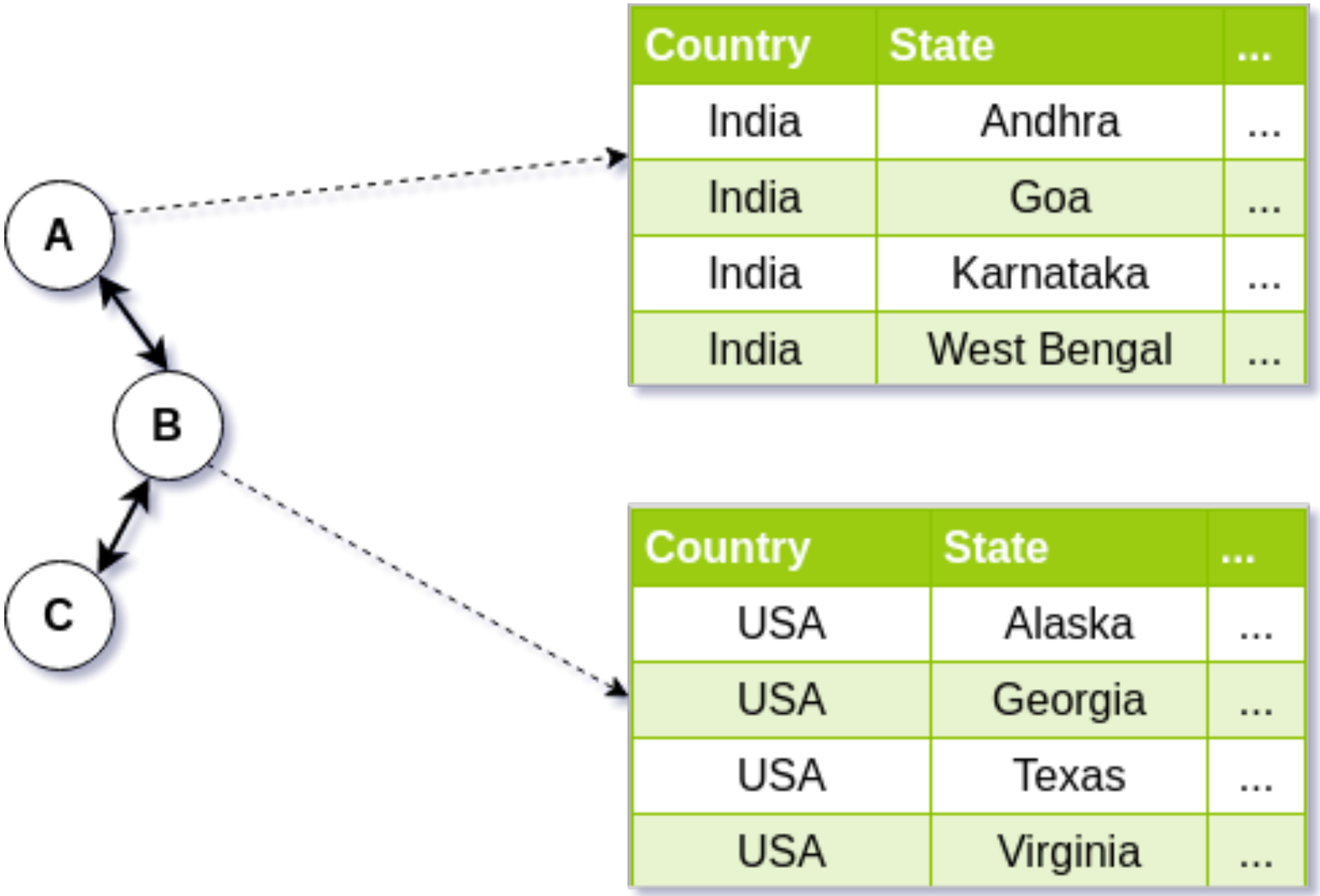
- Cluster order —> Controla a ordem (ASC ou DESC) de como os dados dentro de uma partição serão salvos no disco.

PARTITION KEY

- Quando um dado é inserido, uma função hash da patition key é feita para decidir em qual nó o dado residirá.



```
CREATE TABLE store_by_location (
    country text,
    state text,
    city text,
    store_name text,
    PRIMARY KEY (country, state)
);
```



Instalação

https://www.docker.com/

INSTALAÇÃO

Crie uma rede

```
docker network create cassandra-net
```

INSTALAÇÃO

Instalando o Cassandra (Instância simples)

```
docker run --name cassandra1 -p 9042:9042  --network
cassandra-net -d cassandra
```

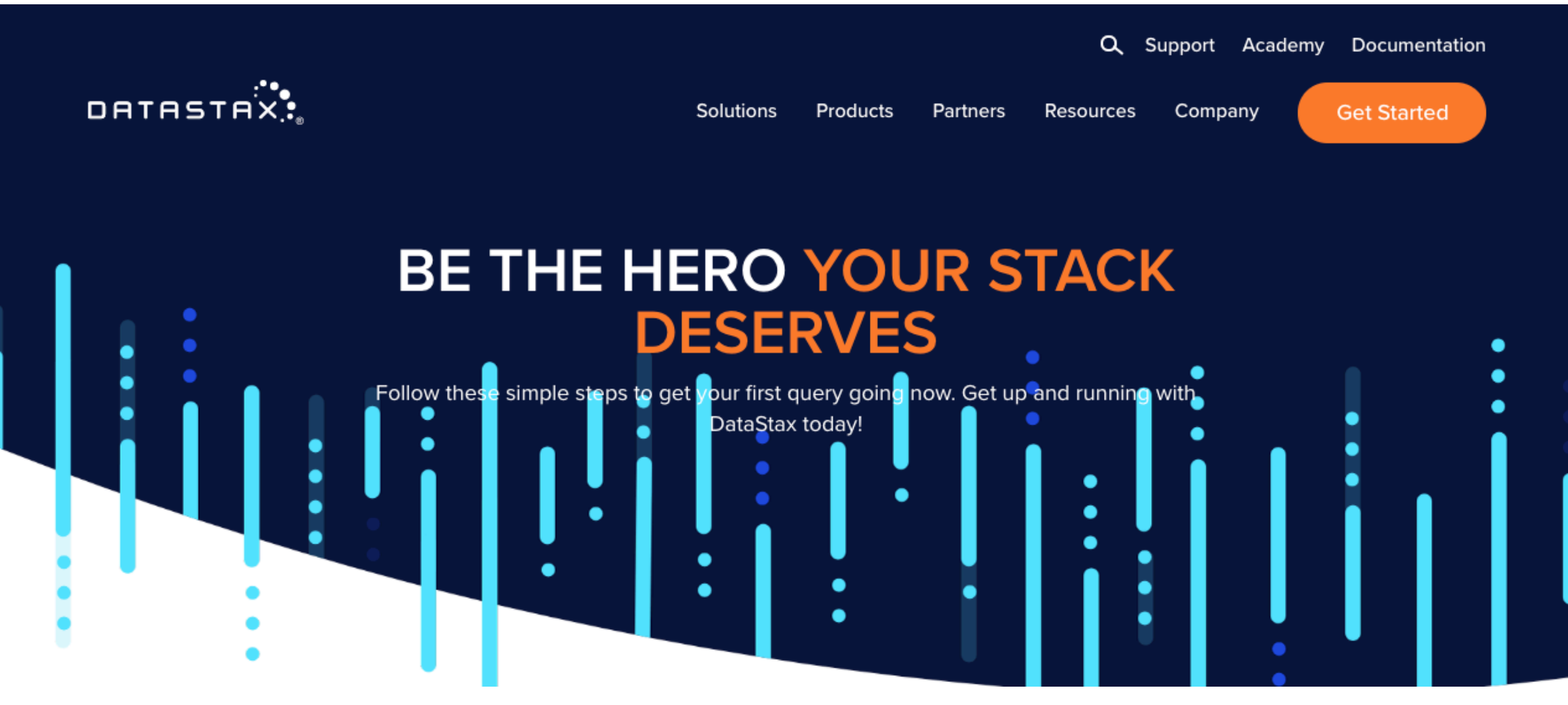
INSTALAÇÃO

Iniciando o prompt do cassandra CQLSH

```
$ docker run -it --network cassandra-net --rm
cassandra cqlsh cassandra1
```

INSTALAÇÃO

https://www.datastax.com/get-started#two



BE THE HERO YOUR STACK DESERVES

Follow these simple steps to get your first query going now. Get up and running with DataStax today!

```
$ docker run -e DS_LICENSE=accept --name my-dse -d
datastax/dse-server:6.8.10 -g -s -k
```

```
$ docker run -e DS_LICENSE=accept --link my-dse -p
9091:9091 --memory 1g --name my-studio -d datastax/
dse-studio
```

EDIT CONNECTION

Name *

default localhost

Username

Host/IP (comma delimited) *

my-dse

Password

Port *

9042

Use SSL

Save

Cancel

Test

Primeiros Comandos (CQL)

CQL

- CQL (Cassandra Query Language) é uma linguagem baseada no SQL para operações no Apache Cassandra

CQL

Criando um keyspace

```
CREATE  KEYSPACE | SCHEMA  IF NOT EXISTS keyspace_name
WITH REPLICATION = map
AND DURABLE_WRITES =  true | false
```

Exemplo para estratégia simples

```
CREATE KEYSPACE cycling WITH REPLICATION = { 'class':
'SimpleStrategy', 'replication_factor': 3}
```

CQL

Criando um keyspace

Exemplo para estratégia de data center

```
CREATE KEYSPACE NTSkeyspace WITH REPLICATION = { 'class':
'NetworkTopology', 'datacenter1': 1}
```

Exemplo para estratégia de data center

```
CREATE KEYSPACE Excalibur WITH REPLICATION = { 'class':
'NetworkTopology', 'dc1': 3, 'dc2': 2}
```

CQL

Alterand um keyspace

```
ALTER  KEYSPACE | SCHEMA  keyspace_name
WITH REPLICATION = map
|   WITH DURABLE_WRITES =  true | false
AND  DURABLE_WRITES =  true | false
```

Exemplo de alteração

```
ALTER KEYSPACE "Excalibur" WITH REPLICATION =
  { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 3 };
```

```
ALTER KEYSPACE Excelsior WITH REPLICATION = { 'class':
'SimpleStrategy', 'replication_factor': 3}
```

CQL

Criando uma tabela

```
CREATE TABLE IF NOT EXISTS keyspace_name.table_name
( column_definition, column_definition, ...)
WITH property AND property ...
```

Exemplo de criação de tabela

```
CREATE TABLE cycling.cyclist_name (
   id UUID PRIMARY KEY,
   lastname text,
   firstname text );
```

CQL

Criando uma tabela

Criando tabela com chave primária composta

```
CREATE TABLE cycling.cyclist_category (
   category text,
   points int,
   id UUID,
   lastname text,
   PRIMARY KEY (category, points))
WITH CLUSTERING ORDER BY (points DESC);
```

CQL

Criando uma tabela

Criando tabela com chave partition key composta

```
CREATE TABLE cycling.rank_by_year_and_name (
    race_year int,
    race_name text,
    cyclist_name text,
    rank int,
    PRIMARY KEY ((race_year, race_name), rank) );
```

Partition key

CQL

Inserindo Dados

```
INSERT INTO keyspace_name.table_name
  ( identifier, column_name...)
  VALUES ( value, value ... ) IF NOT EXISTS
  USING option AND option
```

```
INSERT INTO cycling.cyclist_name (id, lastname, firstname)
  VALUES (6ab09bec-e68e-48d9-a5f8-97e6fb4c9b47,
‘KRUIKSWIJK','Steven');
```

CQL

Inserindo Dados

```
INSERT INTO keyspace_name.table_name
  ( identifier, column_name...)
  VALUES ( value, value ... ) IF NOT EXISTS
  USING option AND option
```

```
INSERT INTO cycling.cyclist_name (id, lastname, firstname)
  VALUES (6ab09bec-e68e-48d9-a5f8-97e6fb4c9b48, 'SILVA','JOÃO')
  USING TTL 10;
```

CQL

Inserindo Dados

Insere apenas se não existe

```
INSERT INTO cycling.cyclist_name (id, lastname, firstname)
   VALUES (c4b65263-fe58-4846-83e8-f0e1c13d518f, 'RATTO', 'Rissella')
IF NOT EXISTS;
```

CQL

Atualizando Dados

Atualizando dados

```
UPDATE cycling.cyclist_name
SET lastname = 'novo lastane'
WHERE id = c4b65263-fe58-4846-83e8-f0e1c13d518f IF EXISTS;
```

CQL

Removendo Dados

Deletando apenas algumas colunas

```
DELETE firstname, lastname FROM cycling.cyclist_name
WHERE id = 6ab09bec-e68e-48d9-a5f8-97e6fb4c9b47;
```

Deletando toda a linha

```
DELETE FROM cycling.cyclist_name
WHERE id = 6ab09bec-e68e-48d9-a5f8-97e6fb4c9b47;
```

CQL

Selecionando Dados

Selecionando dados

```
SELECT select_expression
FROM keyspace_name.table_name
WHERE relation AND relation ...
ORDER BY ( clustering_column  ASC | DESC ...)
LIMIT n
ALLOW FILTERING
```

CQL

Selecionando Dados

Selecionando registros:

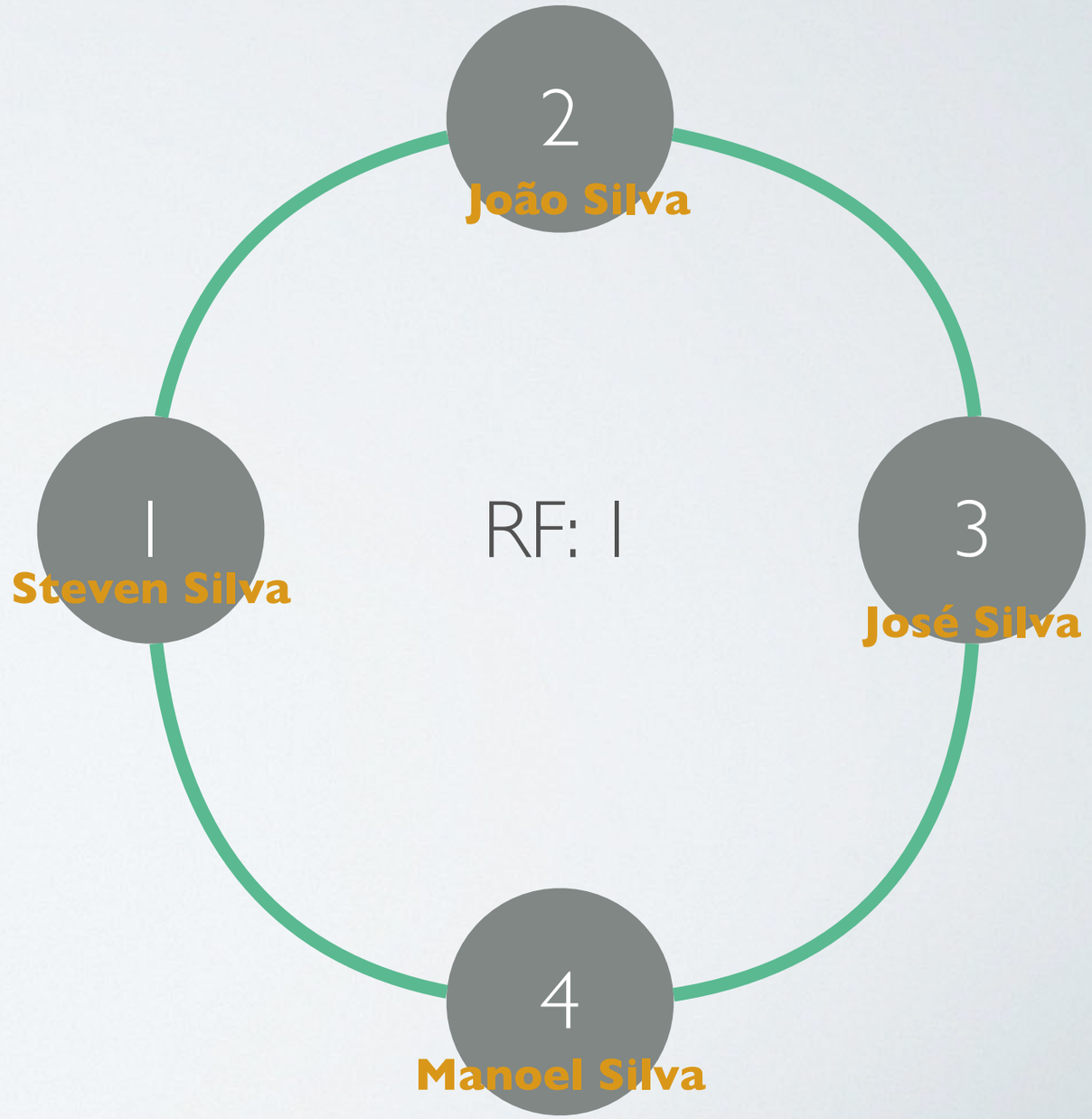
```
SELECT * FROM cycling.cyclist_name;
```

```
SELECT lastname FROM cycling.cyclist_name;
```

```
SELECT lastname FROM cycling.cyclist_name LIMIT 2;
```

```cql
CREATE TABLE cycling.cyclist_name (
    id UUID PRIMARY KEY,
    lastname text,
    firstname text );
```

```cql
INSERT INTO cycling.cyclist_name (id, lastname, firstname) VALUES (1, 'Silva','Steven')
INSERT INTO cycling.cyclist_name (id, lastname, firstname) VALUES (2, 'Silva','Joao')
INSERT INTO cycling.cyclist_name (id, lastname, firstname) VALUES (3, 'Silva','José')
INSERT INTO cycling.cyclist_name (id, lastname, firstname) VALUES (4, 'Silva','Manoel')
```



Front

Mobile

GET http://localhost/usuarios?nome="joao"

API

Backend

DB

Banco de dados NoSQL - Cassandra (Introdução)

Prof. Gustavo Leitão