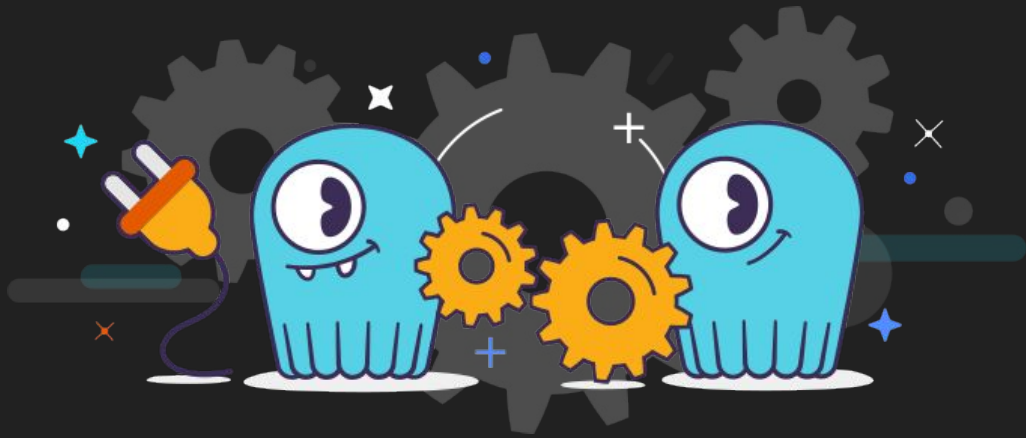
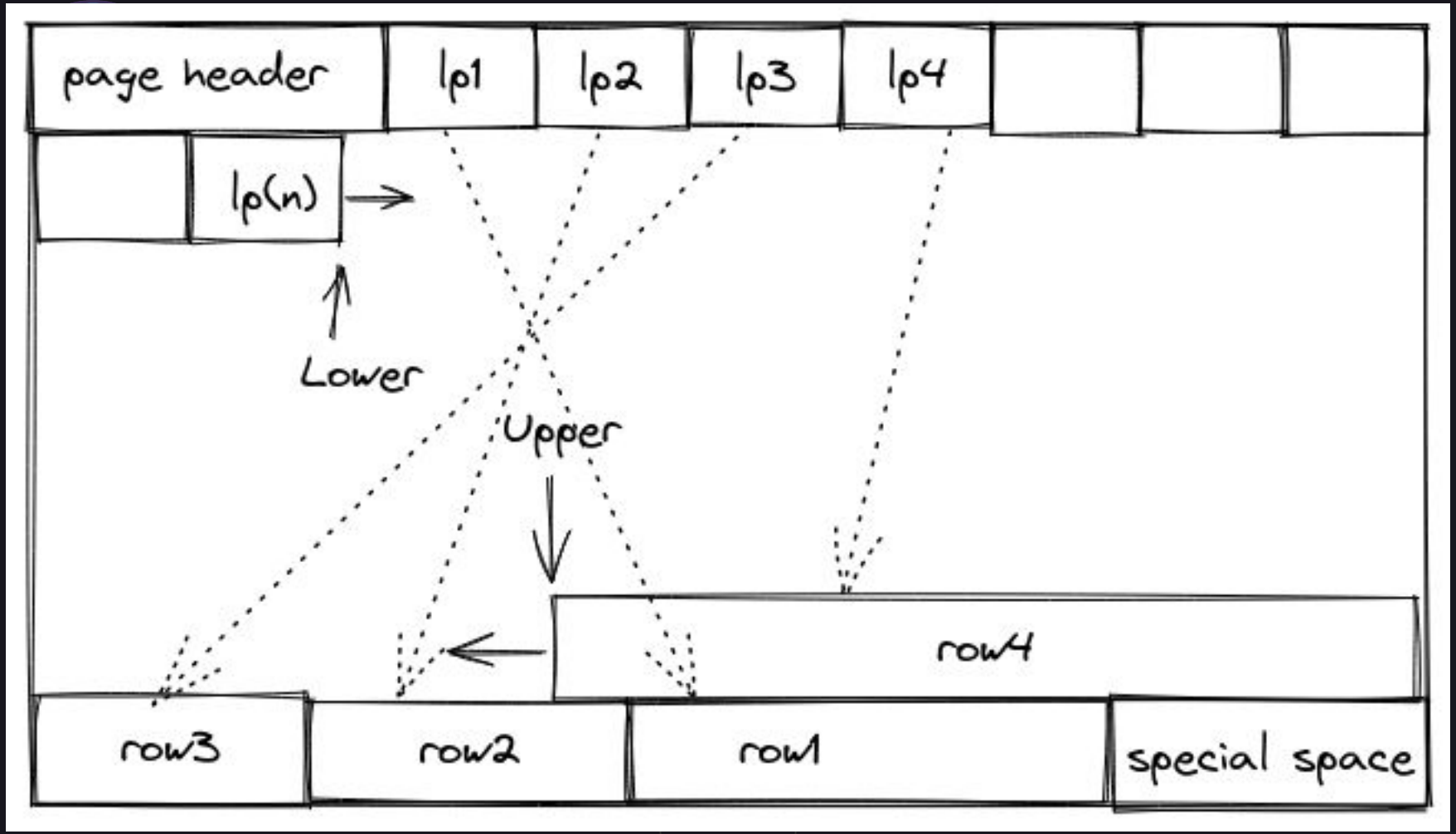
Objetivos

1. Características centrais do Scylla e como utilizá-lo
2. Replicação, consistência e queries
3. Adapter
4. Performance



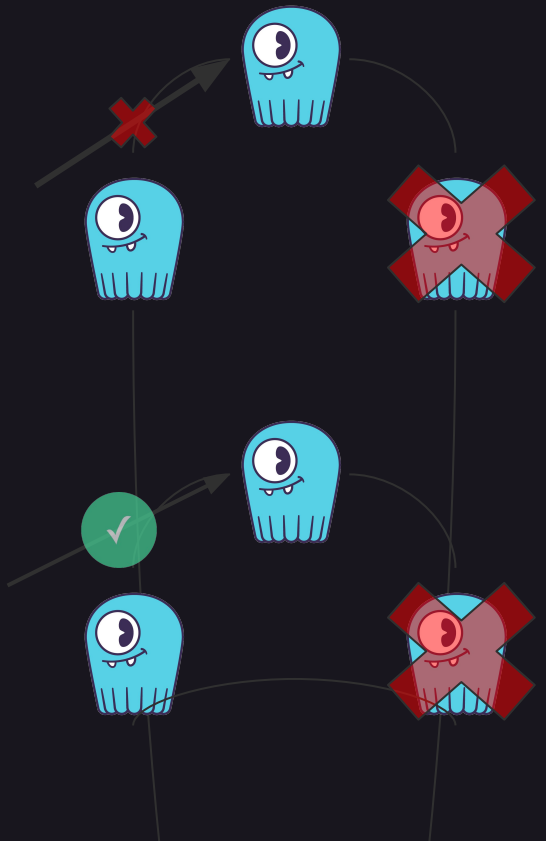

page header
lp1
lp2
lp3
lp4
lp(n)
Lower
Upper
row4
row3
row2
row1
special space

CAP Theorem

Partition
Falha de comunicação entre nós

Consistency
Nega requisição até que recupere comunicação

Availability
Aceita requisição mas gera inconsistência



```
version: "3"

services:
```


```
version: "3"

services:
```


Cluster Name

Enter cluster name

$100

Choose Your Cloud Provider

Where should we install your Cluster?

Run Under Which Account?

Under which account should we work?

Additional Services - Scylla Cloud clusters include additional instances: Scylla Manager (t3.medium) and Scylla Monitoring (m5.large)

Scylla Version

Scylla Enterprise 2022.2.6 - CQL Compatible API

Scylla Enterprise 2022.2.6 - CQL Compatible API

Scylla Alternator 2022.2.6 - DynamoDB™ compatible API

Regions

Choose your default region, you'll be able to add more regions later (VPC must be enabled for multi-region)

Region

US East (N. Virginia)

Scylla cloud prices are billed hourly. For special pricing on an annual contract, please contact Scylla Cloud Sales.

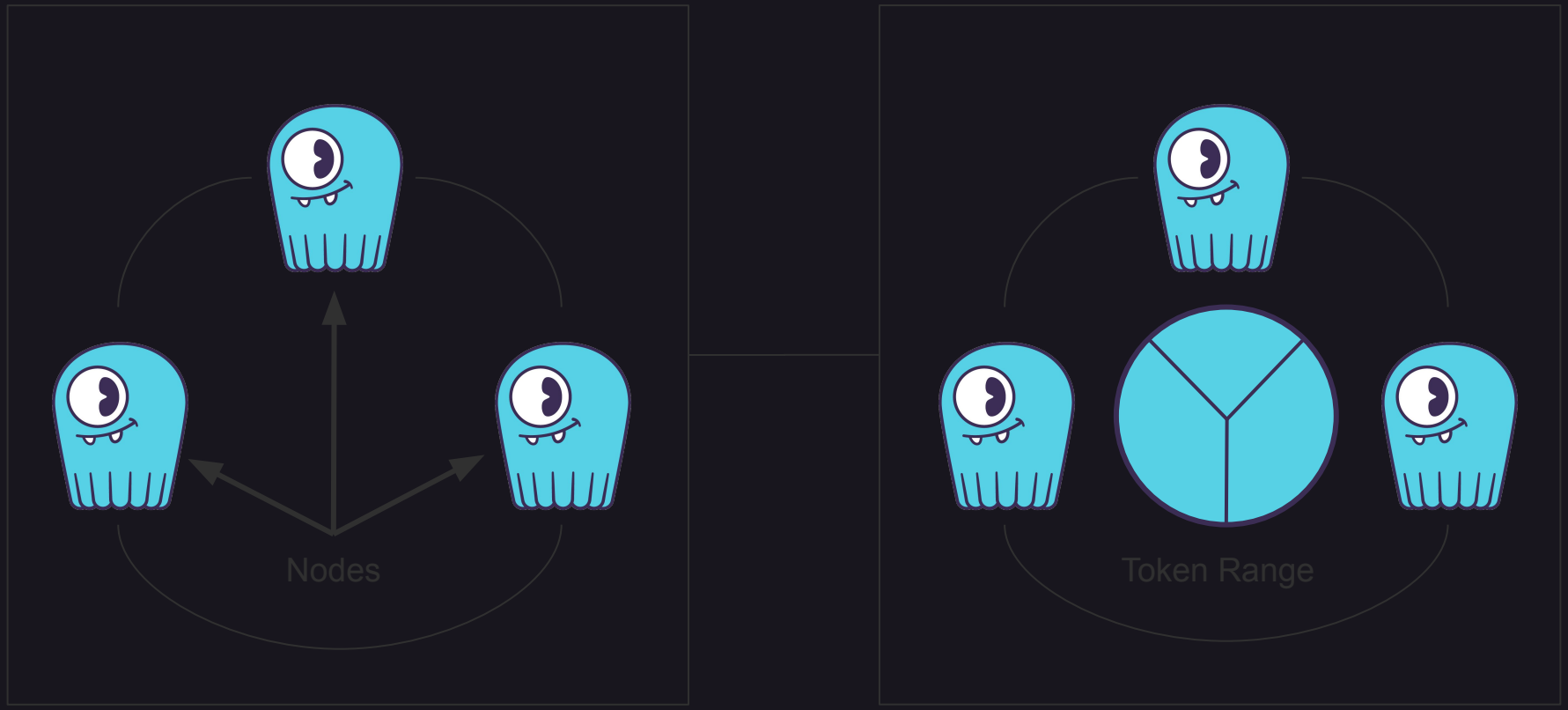

```
web:
  driver: bridge
```


```
web:
  driver: bridge
```

Cluster | Datacenters

DC1 - us-east-1

DC2 - sa-east-1



Cluster | Keys

PRIMARY KEY((rule_id, date), timestamp)

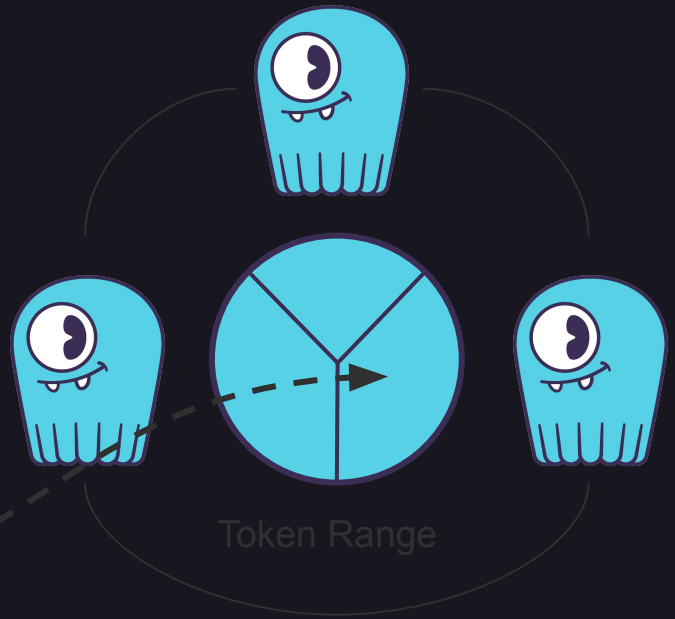
Partition key Cluster key

Ordenação

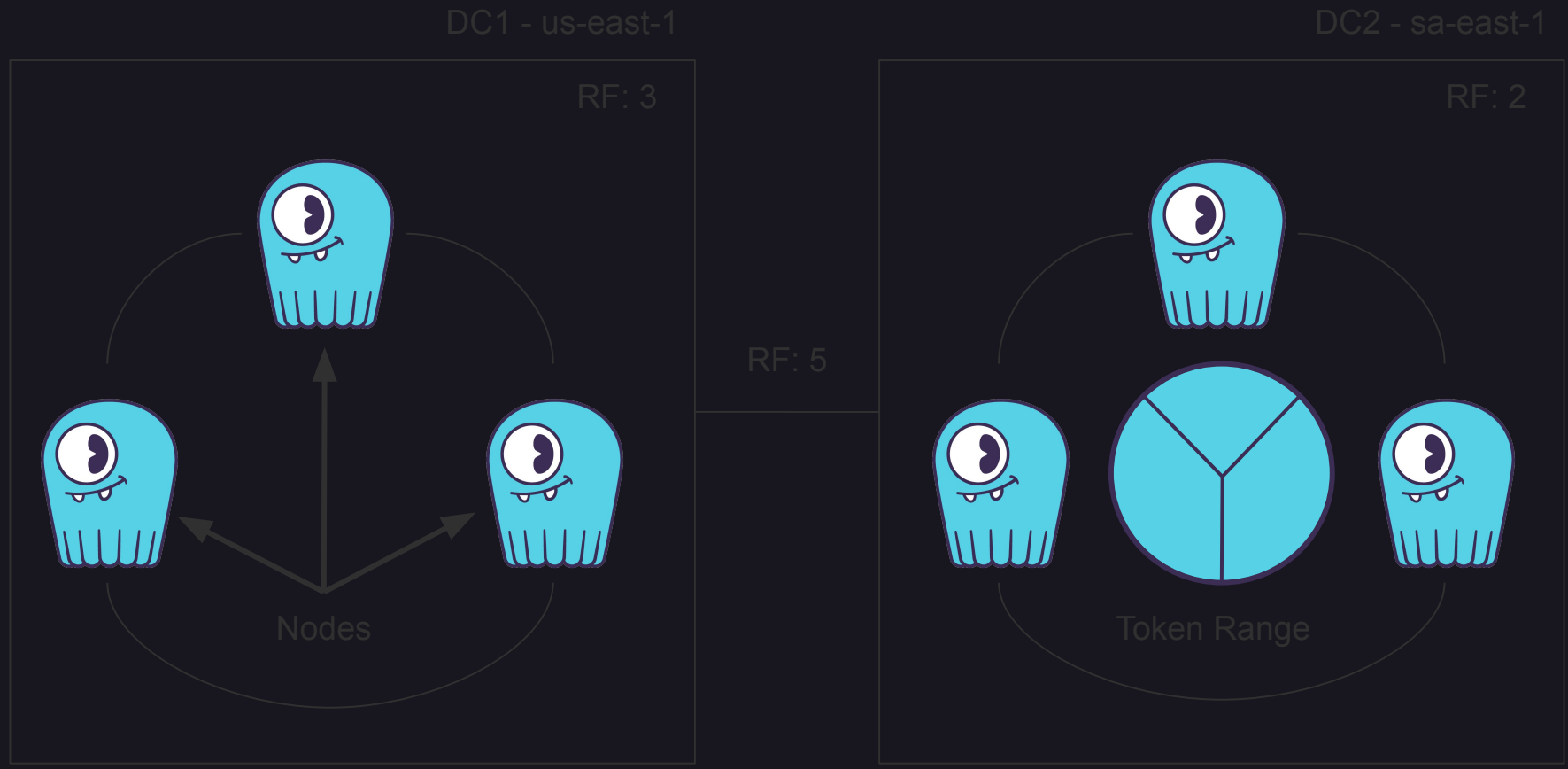
Partitioner

Token

9143256562457711404



Cluster | Replication Factor



Keyspace | Replication Factor

```
CREATE KEYSPACE swat
WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1': 3, 'DC2': 2}
```

Keyspace | Create Table

```
CREATE TYPE dependency(
        id uuid,
        status_map map<text,
text>
)
```

```
CREATE KEYSPACE swat
WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1': 3, 'DC2': 2}
```

```
CREATE TABLE feature (
id uuid
name text,
internal boolean,
tags set<text>,
dependencies list<frozen<dependency>>,
date date,
created_at timestamp,
PRIMARY KEY((id), date, created_at))
WITH CLUSTERING ORDER BY (date
DESC);
```

Operações | Insert

```
INSERT INTO feature
(id, name, internal, tags, dependencies, date, created_at)
VALUES (uuid(), "BOPE Logs", {'Bope', 'Logs'}, [{'1234': 'Insert Log'}, {'12345': 'Batch
Insert Log'}], '08:12:54', 1299038700000);
```

Operações | Consistency Level

```
INSERT INTO feature
(id, name, internal, tags, dependencies, date, created_at)
VALUES (uuid(), "BOPE Logs", {'Bope', 'Logs'}, [{'1234': 'Insert Log'}, {'12345': 'Batch
Insert Log'}], '08:12:54', 1299038700000);
```

```
CONSISTENCY ONE;
CONSISTENCY QUORUM;
CONSISTENCY ALL;
```

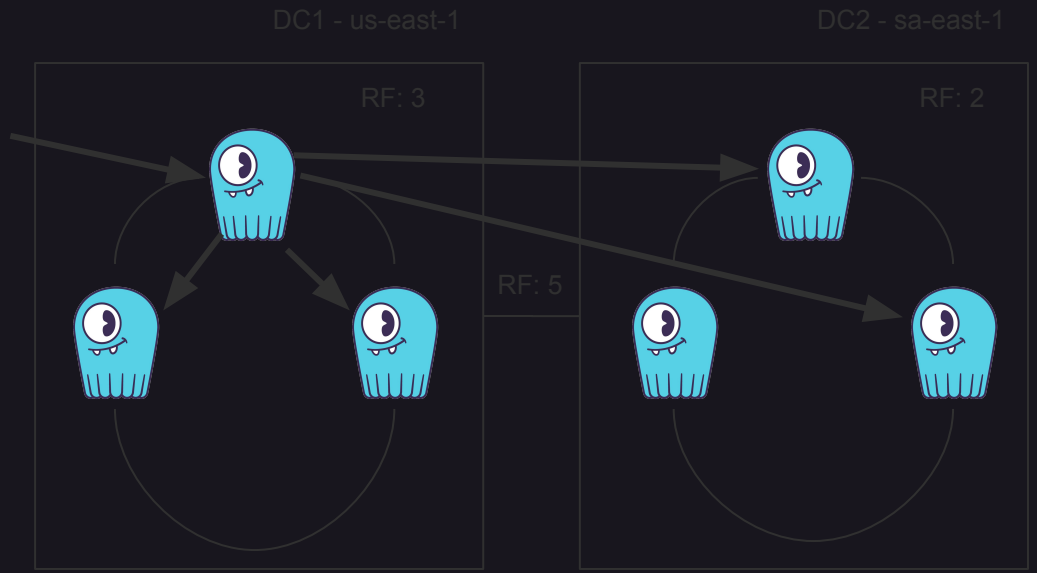
```
CONSITENCY LOCAL_XXX
```

Operações | Select

TODA QUERY DEVE POSSUIR TODAS AS COLUNAS DEFINIDAS NA PARTITION KEY

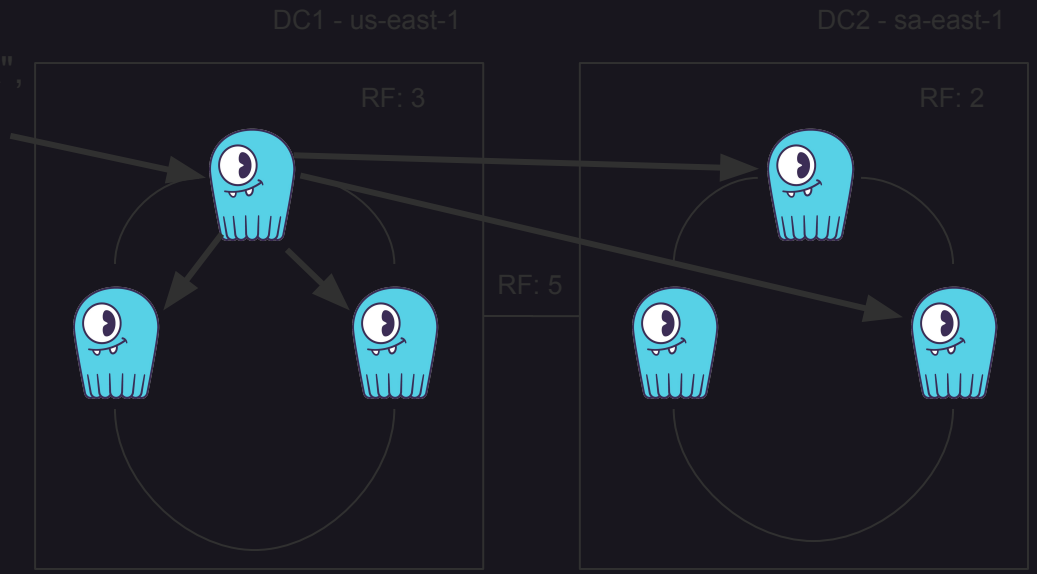
```
SELECT name, tags, dependencies
FROM feature
WHERE id =
'e94ef903-88ec-46d5-9739-219d02b5e8
d8' AND date <= '2023-01-23';
```

```
SELECT name, tags, dependencies
FROM feature
WHERE id =
'e94ef903-88ec-46d5-9739-219d02b5e8
d8' and name = 'Logs BOPE';
```



Operações | JSON

```
INSERT INTO feature
JSON '{
\"id\":
\"e94ef903-88ec-46d5-9739-219d02b5e8d8\",
\"name\": \"Logs BOPE\",
\"tags\": [\"Bope\",  \"Logs\"],
\"dependencies\": [{\"1234\": \"Insert Log\"},
{\"12345\": \"Batch Insert Log\"],
\"date\": \"08:12:54\",
\"created_at\": 1299038700000}'
```



MVs | Materialized Views

```
CREATE MATERIALIZED VIEW internal_feature AS
SELECT id, name, tags, dependencies
WHERE id IS NOT NULL AND name IS NOT NULL AND internal = true
PRIMARY KEY (name, id);
```

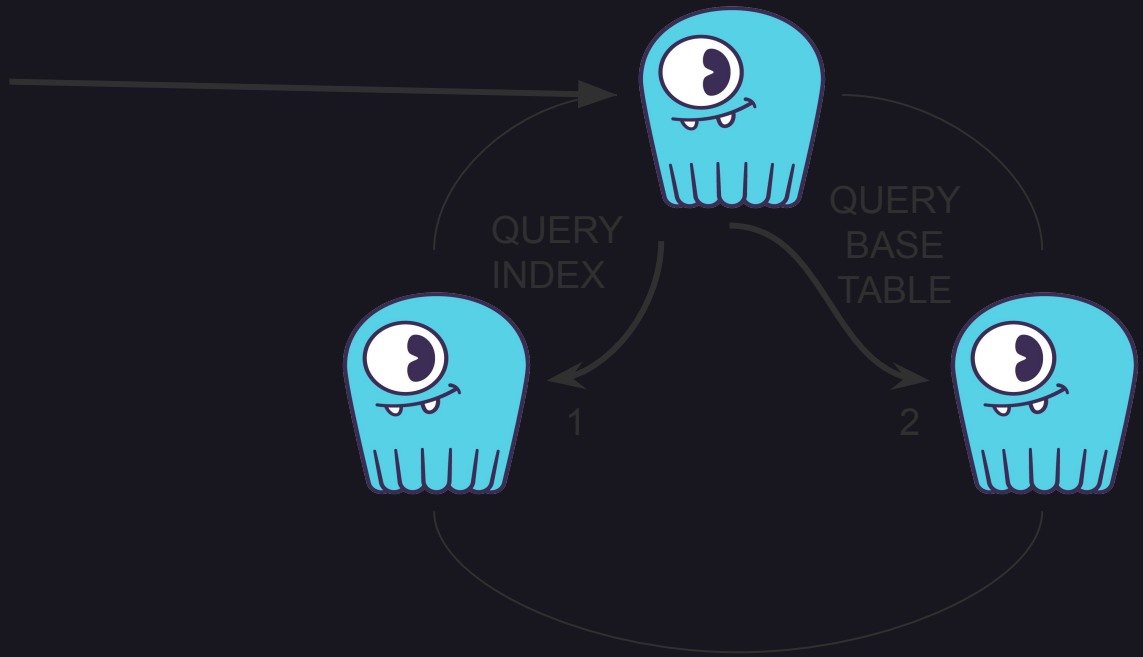
id	name	tags	dependencies
…	…	…	…
…	…	…	…

CRIA OUTRA TABELA
OUTRA PARTITION KEY

Índices | Global Secondary Index

```
CREATE INDEX feature_name ON feature(name);
```

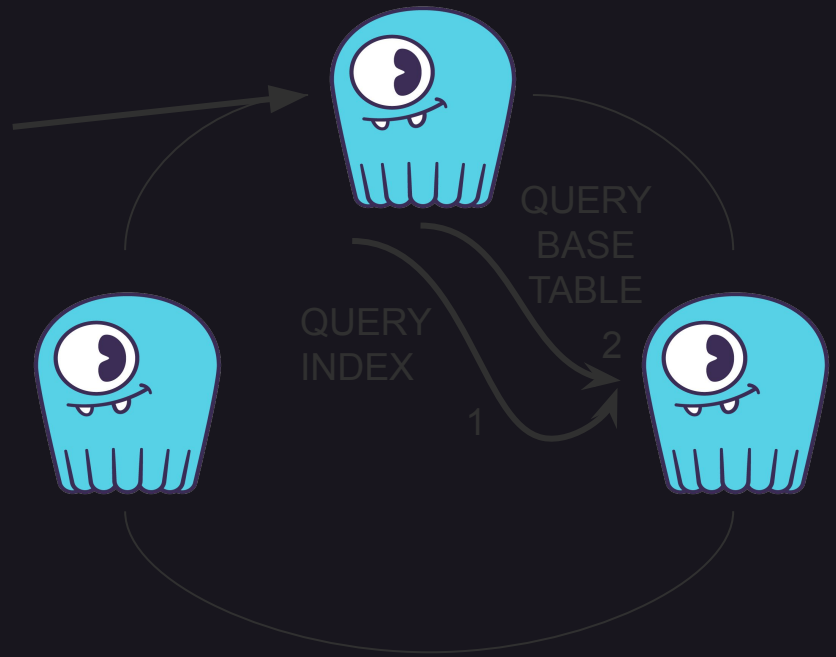
```
SELECT * FROM feature
WHERE name = 'Logs BOPE';
```



Índices | Local Secondary Index

```
CREATE INDEX ON feature((id), name);
```

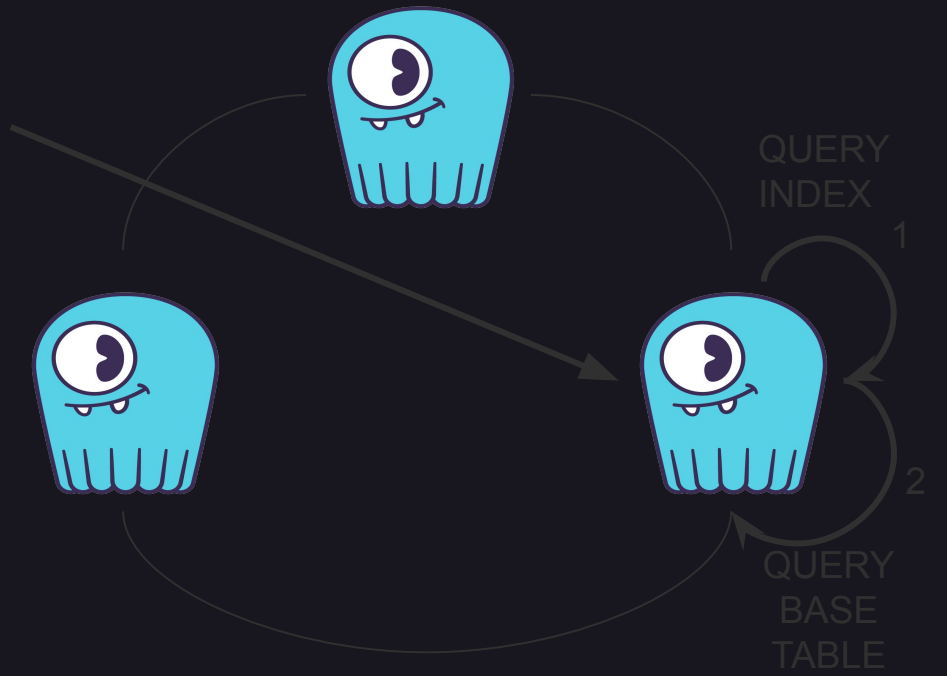
```
SELECT * FROM feature
WHERE
id = 'e94ef903-88ec-46d5-9739-219d02b5e8d8' AND
name = 'Logs BOPE';
```



Driver | Token Aware Policy

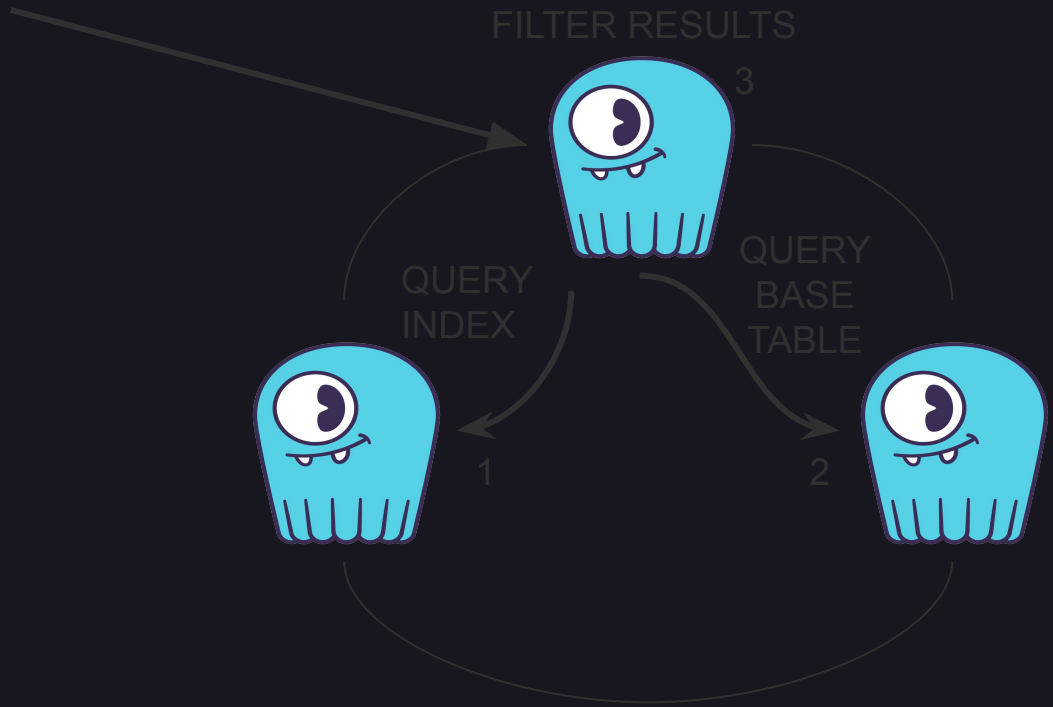
```
CREATE INDEX ON feature((id), name);
```

```
SELECT * FROM feature
WHERE
id = 'e94ef903-88ec-46d5-9739-219d02b5e8d8' AND
name = 'Logs BOPE';
```



Filter | Filtering

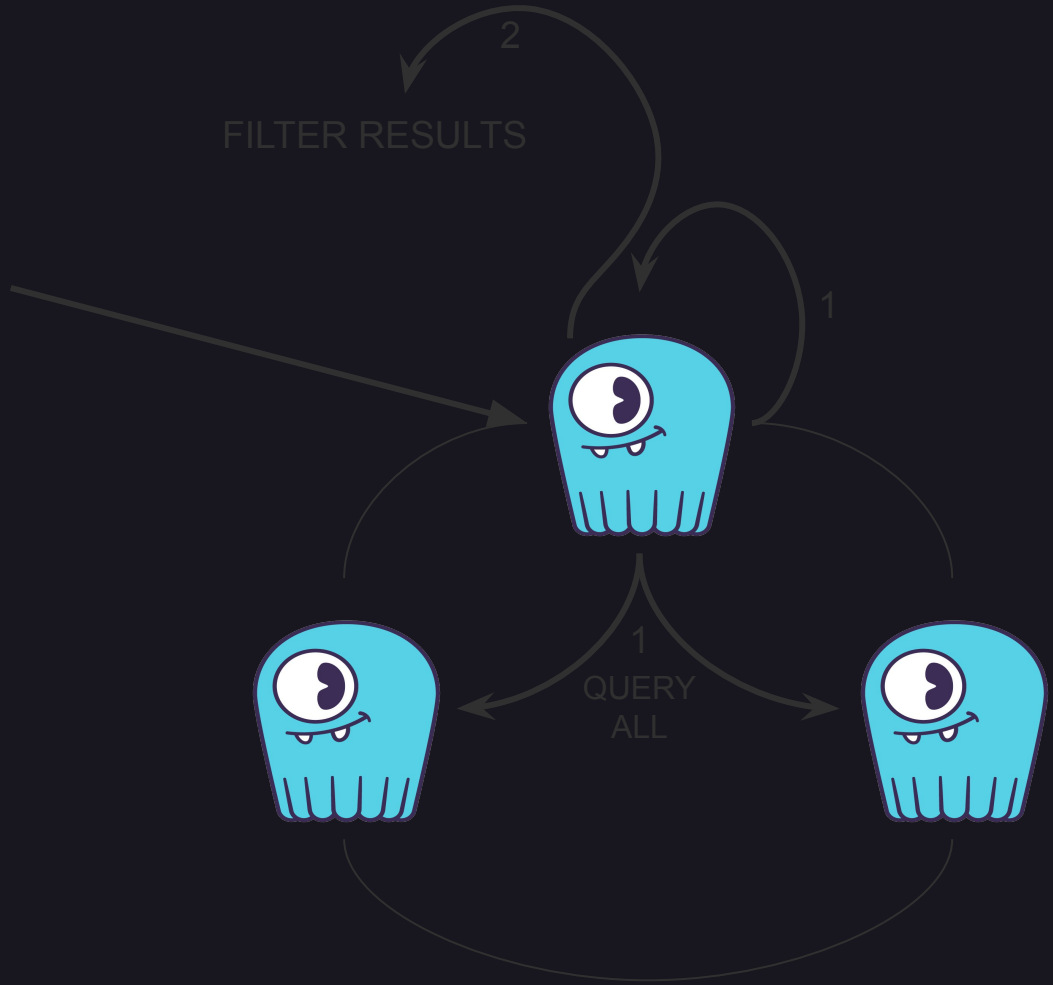
```
SELECT * FROM feature
WHERE name = 'Logs BOPE'
AND created_at >= 1299038700000
ALLOW FILTERING;
```



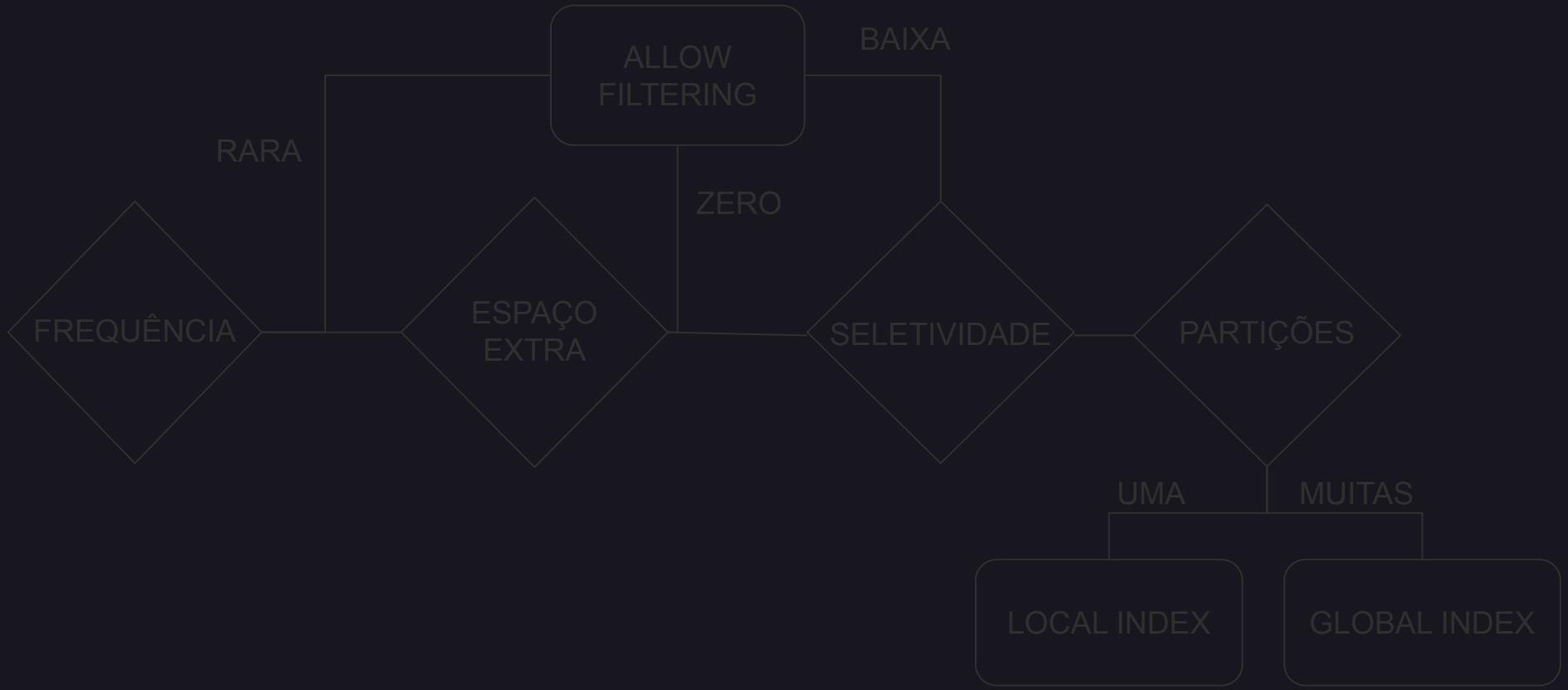
Filter | Filtering

```
SELECT * FROM feature
WHERE
created_at >= 1299038700000
ALLOW FILTERING;
```

UTILIZAR FILTERING
PODE SER MELHOR
QUANDO A SELETIVIDADE
É BAIXA



Melhor Decisão | INDEX VS FILTERING



Transactions | Lightweight Transactions (LWTs)

- Transactions são pré-condições (IFs) que devem ser satisfeitas para que uma operação seja executada.

```
INSERT INTO feature
(id, name, internal, tags, dependencies, date, created_at)
VALUES (uuid(), "BOPE Logs", {'Bope', 'Logs'}, [{'1234': 'Insert Log'}, {'12345': 'Batch
Insert Log'}], '08:12:54', 1299038700000)
IF NOT EXISTS;
```

```
UPDATE feature SET name = "Bope Logres"
WHERE id = 'e94ef903-88ec-46d5-9739-219d02b5e8d8'
IF EXISTS;
```

Transactions | Lightweight Transactions (LWTs)

- Transactions são pré-condições (IFs) que devem ser satisfeitas para que uma operação seja executada.

```
INSERT INTO feature
(id, name, internal, tags, dependencies, date, created_at)
VALUES (uuid(), "BOPE Logs", {'Bope', 'Logs'}, [{'1234': 'Insert Log'}, {'12345': 'Batch
Insert Log'}], '08:12:54', 1299038700000)
IF NOT EXISTS;
```

```
UPDATE feature SET name = "Bope Logres"
WHERE id = 'e94ef903-88ec-46d5-9739-219d02b5e8d8'
IF EXISTS;
```

Transactions | Lightweight Transactions (LWTs)

- Transactions são pré-condições (IFs) que devem ser satisfeitas para que uma operação seja executada.

```
BEGIN BATCH
        INSERT INTO feature
        (id, name, internal, tags, dependencies, date, created_at)
        VALUES (uuid(), "BOPE Logs", {'Bope', 'Logs'}, [{'1234': 'Insert Log'}, {'12345':
        'Batch Insert Log'}], '08:12:54', 1299038700000)
        IF NOT EXISTS;

        UPDATE status SET state = 'up'
        where id = '1234'
        IF status != 'down';
APPLY BATCH
```

Monitor


Overview
Cluster overview None
Nodes
Unreachable
Inactive
Manager
Online
Avg Write
686 µs
99% Write
1 ms
Average Read
950 µs
99% Read
9 ms
Requests/s
81.7 k
Load
39%
Timeouts
0
Active Alerts
Writes
Write Latencies
Reads
Read Latencies
Advisor
Report
Time
Category
dashboard
description
2021-03-22 15:57:09
cqlOptimization
Some queries are not token-aware
2021-03-22 15:28:09
Some SELECT queries are non-paged
Balance
An imbalance between shards or nodes may indicates a potential problem
Connections
CQL OPs
Node Latency
Shard Latency
Cache
SSTable
Information for Datacenter1
Disk Size by DC
Nodes
Running Compactions
Cache Hits/Misses
Read/Write Timeouts by DC
Your panels
Your Panels
Your Graph here
No data
Scylla Monitoring version - master
Report an issue on this page
Screenshot


SCYLLA.
Total Nodes
6
Unreachable
0
Joining
0
Leaving
0
Total Requests
Dashboards
Scylla CPU Per Server Metrics
Scylla CQL Optimization
Scylla Overview Metrics
Scylla Per Machine Metrics
Scylla Per Server Metrics
CQL None Prepared St...
1%
CQL None Prepared Statements
None Token Aware
0%
None Token Aware
Non-Paged CQL Reads
0.9%
Non paged CQL Reads
Reversed CQL Reads
0%
Reversed CQL Reads
ALLOW FILTERING CQ...
49%
ALLOW FILTERING CQL Reads
ALLOW FILTERING CQ...
1.0%
ALLOW FILTERING CQL Read Filtering
Cross DC read requests dc1
40%
Cross DC read requests dc2
40%

Integrações

Modelagem de grafos

Usando o Scylla como persistência
(Interessante para features do SWAT)

Streaming de mensagens

Pode funcionar como sink ou para emitir mensagens com mudanças do Scylla

elasticsearch

Pesquisas por texto

Facilidade para front-end

Engenharia de Dados

Agregador de queries, batch processing, data science e machine learning

Adapter

./.cql
- migration-00001.cql
- migration-00002.cql

Hosts
Keyspace
DefaultConsistency
MigrationDir

Adapter

Migrate()
NewTable()

TableBuilder

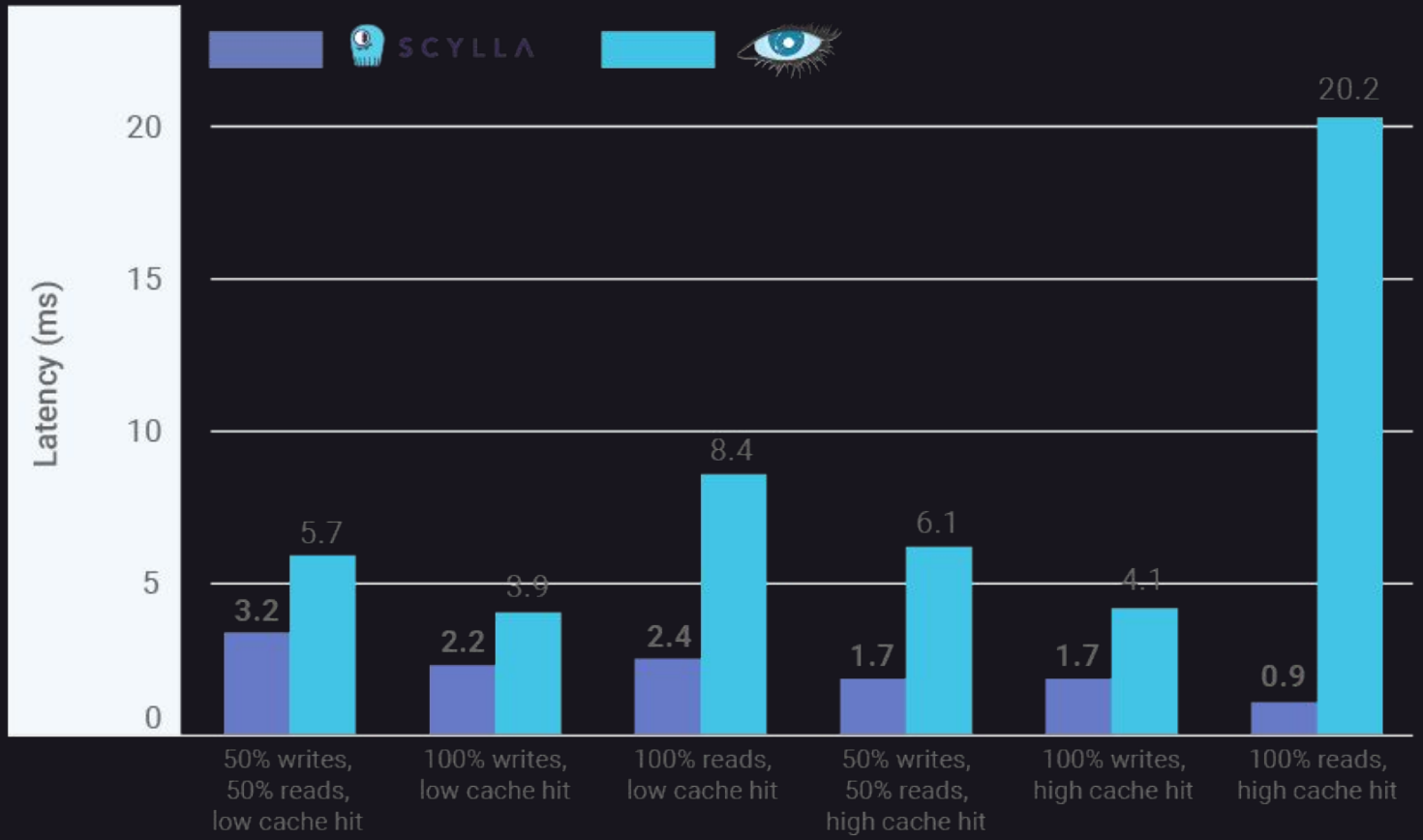
Name()
PartitionKeys()
ClusterKeys()
Columns()
Do()

Insert(obj)
Update(obj)
Delete(obj)
Select(columns…)
XXXBuilder(columns…) -> gocqlx/qb

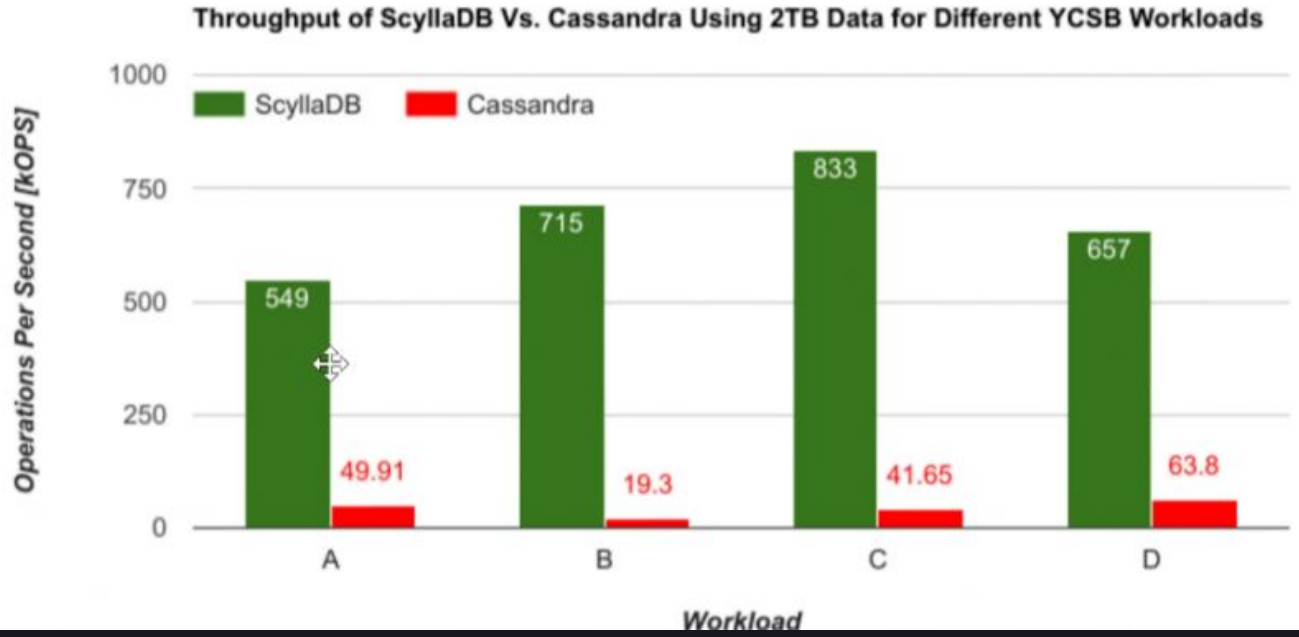
Table

SetupScylla
(resources)

Performance | Cassandra

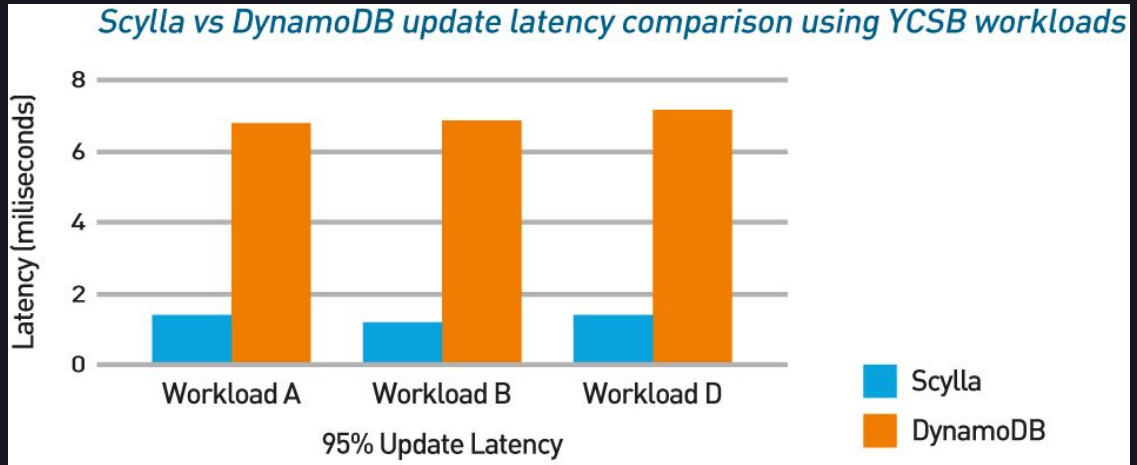


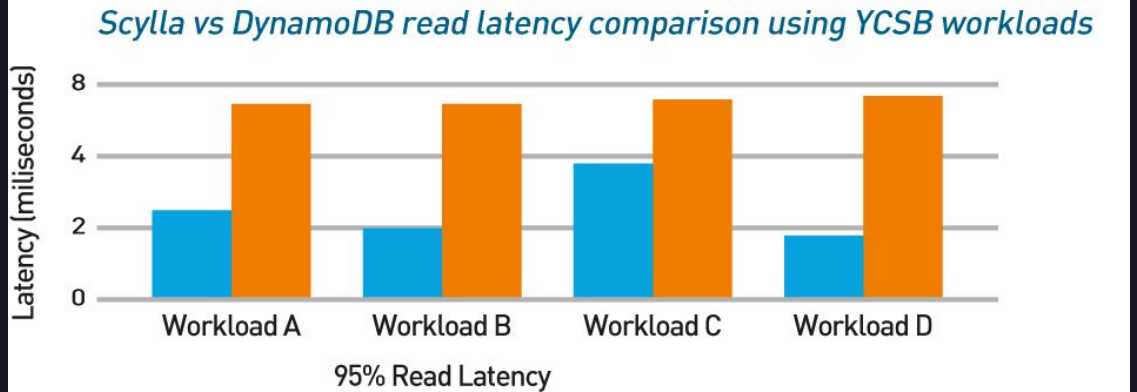
Scylla benchmark by Samsung



Performance | DynamoDB

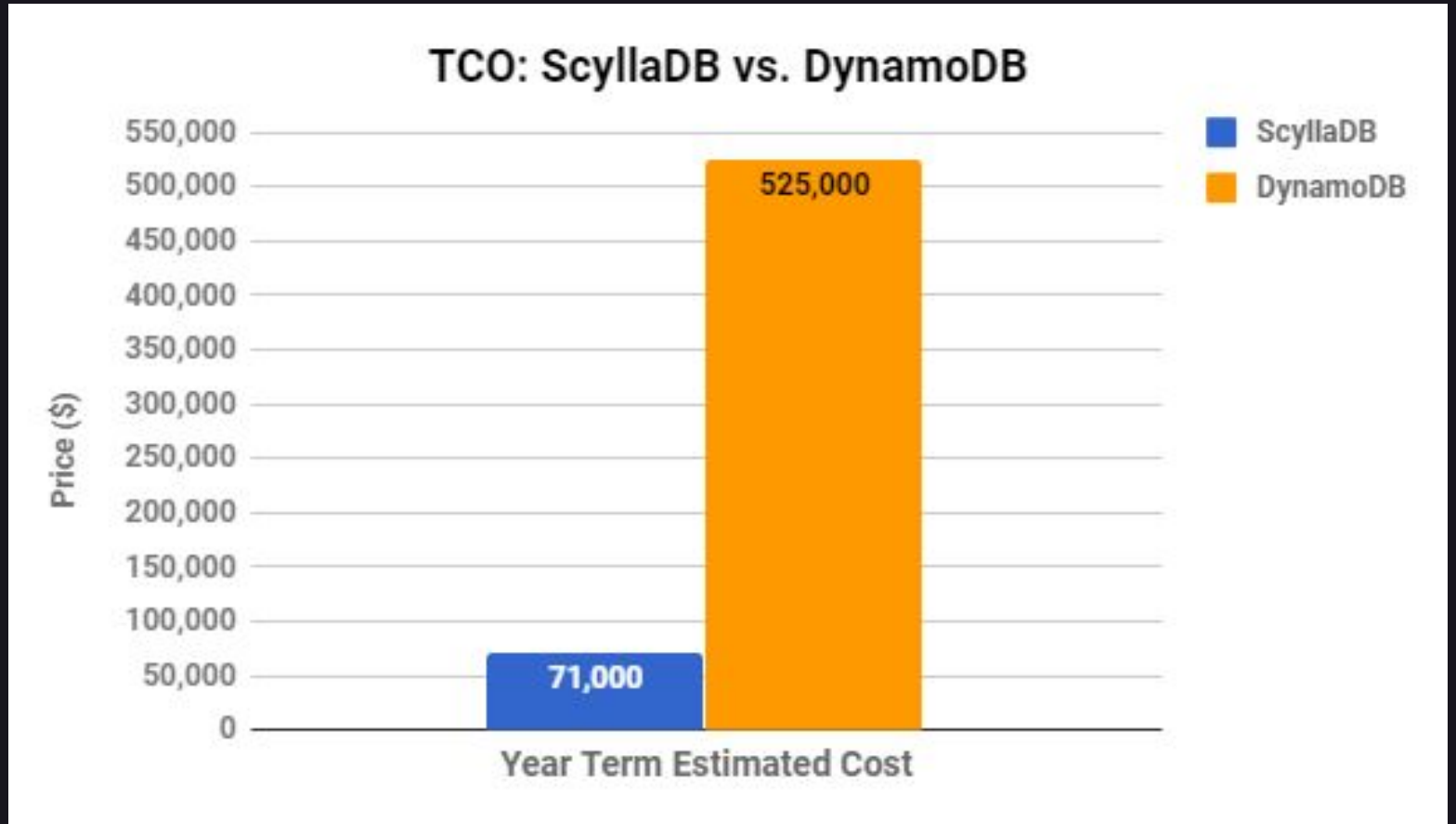
3 x i3.8xlarge vs 160K write | 80K Read



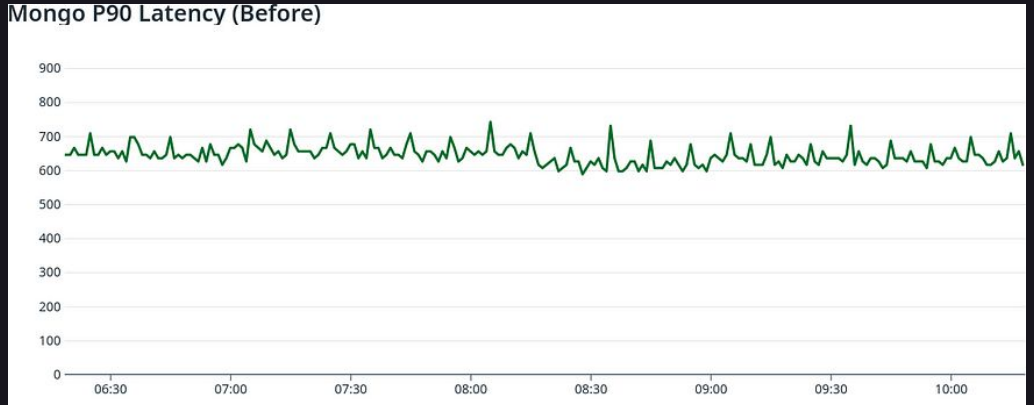


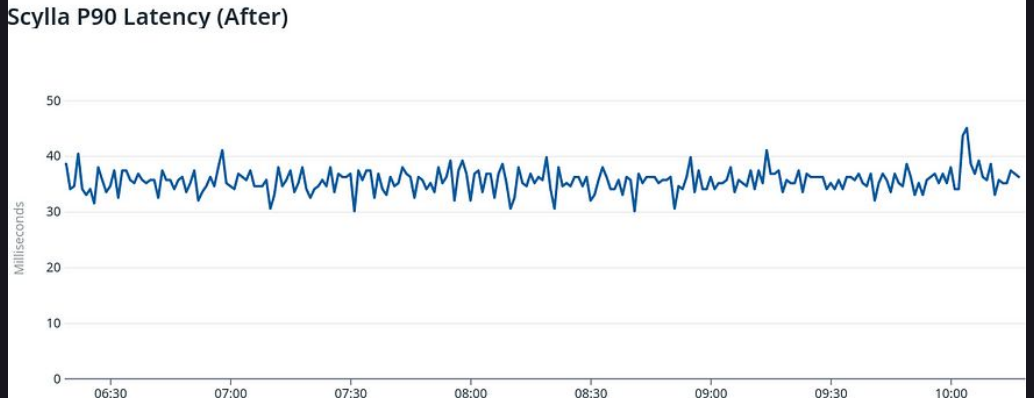
Performance | DynamoDB

3 x i3.8xlarge vs 160K write | 80K Read

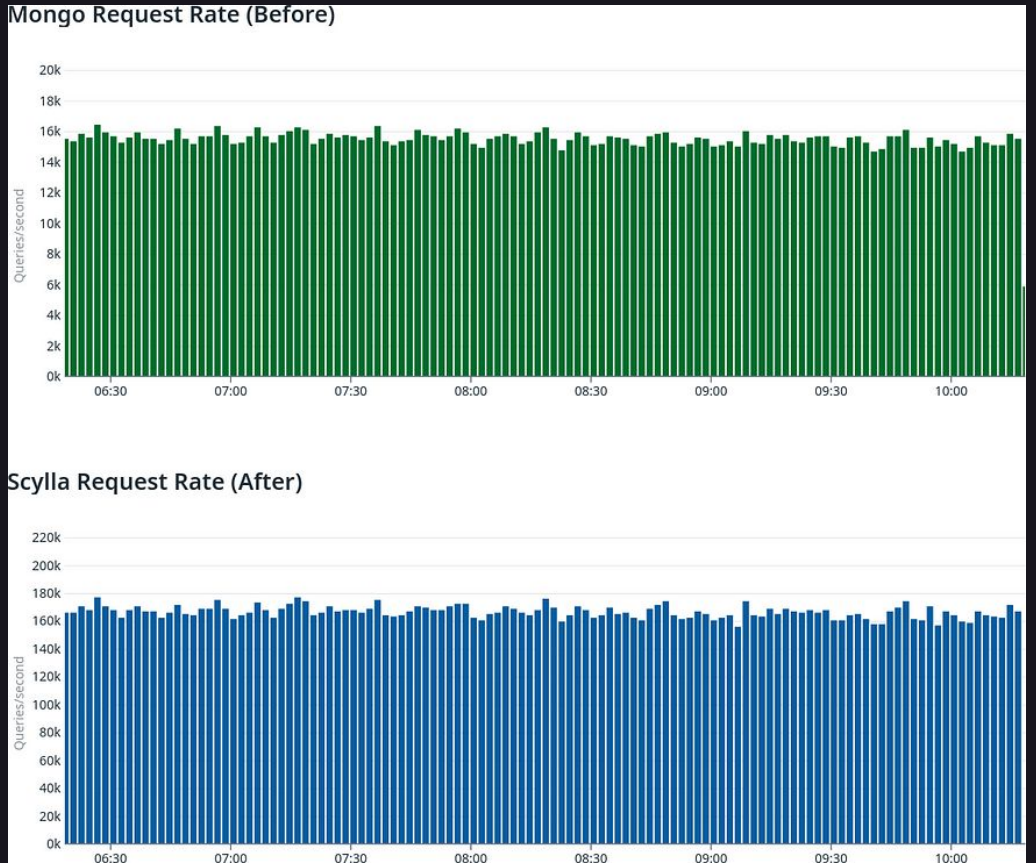


Performance | MongoDB





Performance | MongoDB




MAS BÁTIMA
DE ONDE VOCÊ TIROU ESSE ESCUDO?
made on imgur

YCSB

Traduzido do inglês - O Yahoo! O Cloud Serving Benchmark é uma especificação de código aberto e um conjunto de programas para avaliar os recursos de recuperação e manutenção de programas de computador. Geralmente é usado para comparar o desempenho relativo dos sistemas de gerenciamento de banco de dados NoSQL. Wikipedia (inglês)

Ver descrição original


VOCÊ TÁ MUITO ENGRAÇADINHO HEIN ROBIN
LÓGICO QUE FOI DO CU! DE ONDE MAIS?
made on imgur