Spark

Luis Ramirez Monterosa

Big Data

Big Data permite procesar data

Variable

Veloz

Volumen

Big Data

Big Data permite procesar data

Variable

Veloz

Volumen

BD relacionales, BD nosql

Big Data

Big Data permite procesar data

Variable	BD relacionales, BD nosql
Veloz	cargas ETL son ahora en tiempo real
Volumen	

Big Data

Big Data permite procesar data

Variable	BD relacionales, BD nosql
Veloz	cargas ETL son ahora en tiempo real
Volumen	Terabytes al día, incluso Petabytes

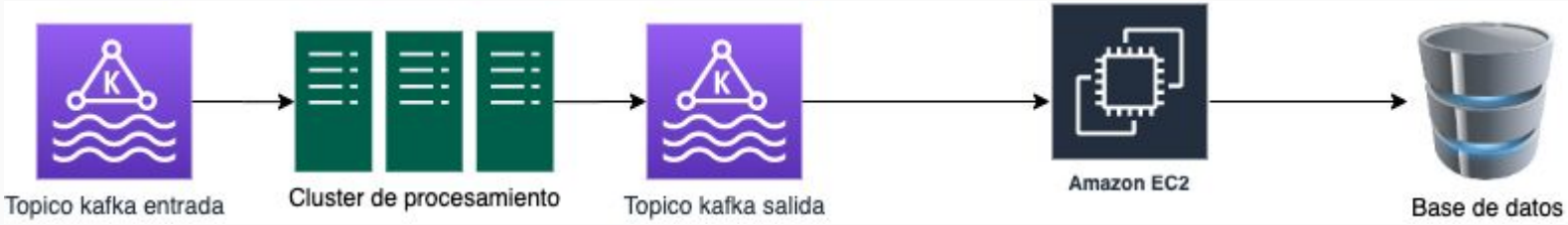
Aplicación típica de Big Data usando Kafka

Kafka no es una base de datos

Tópico de salida produce 1M de trabajos diarios

Se ingresan a base de datos 10%

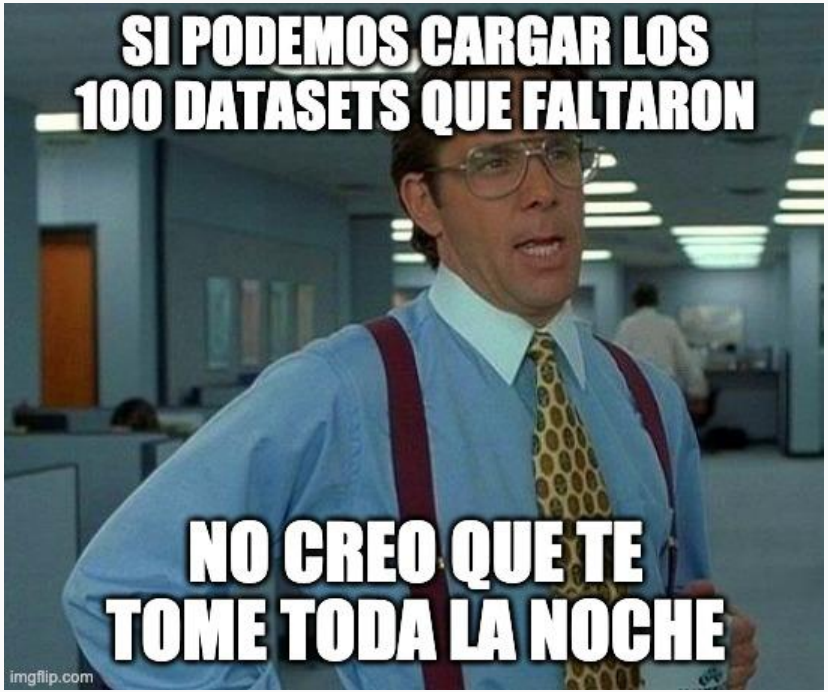
Kafka es un log con expiración



Valor expiración por default: 7 dias

Cargar la data es costoso en bases altamente relacionales

Tu jefe a las 4:00 pm



La vida del desarrollador a las 4:00 pm

- El tópico de resultados fue respaldado
- Un archivo de texto de 30GB
- Cada línea es un JSON
- 2M de records asociados a un dataset
- La llave “datasetId” esta en el JSON
- Hay 2000 datasets
- Solo es necesario cargar 100 datasets

La vida del desarrollador a las 4:00 pm

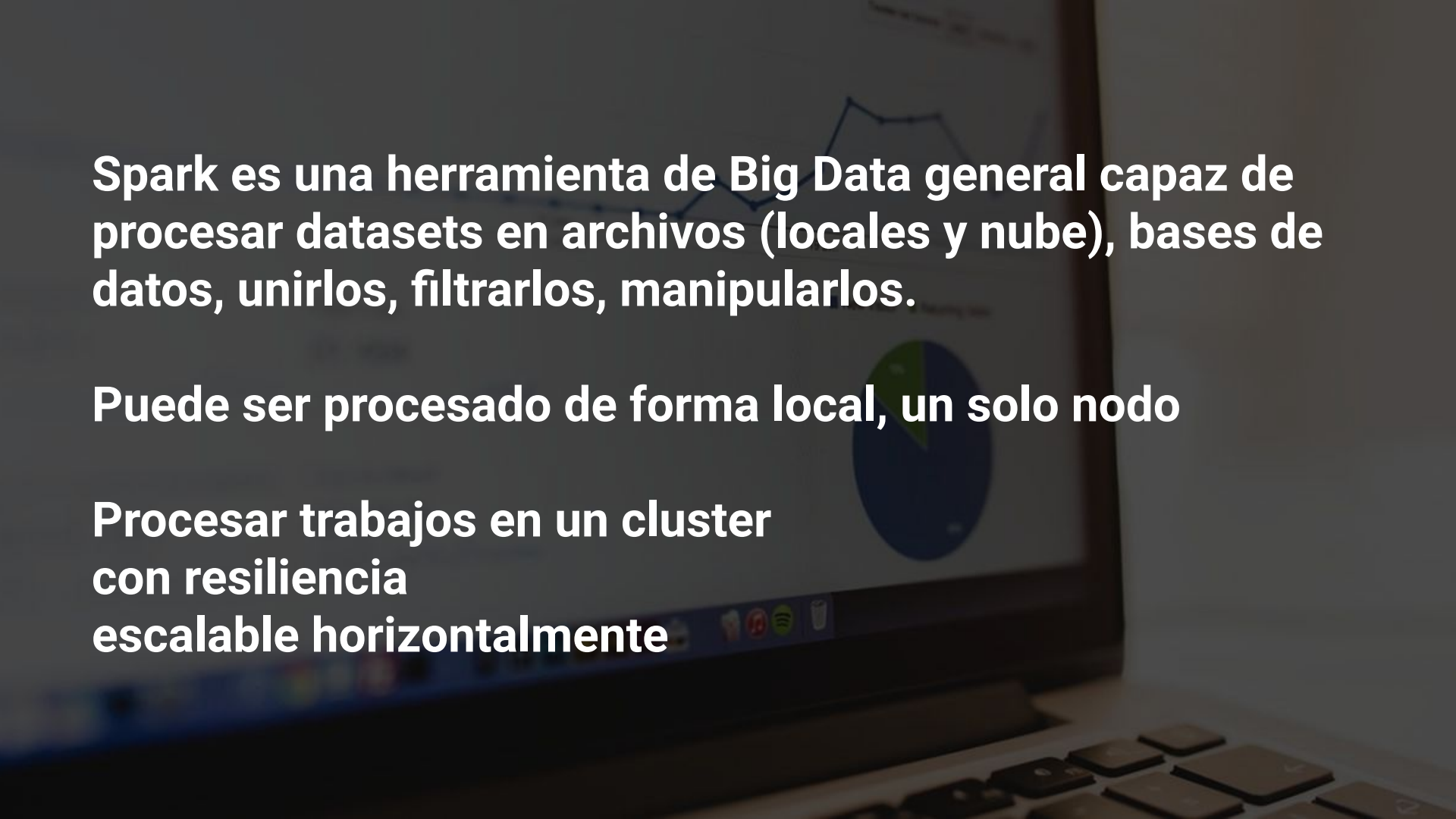
- El tópico de resultados fue respaldado
- Un archivo de texto de 30GB
- Cada línea es un JSON
- 2M de records asociados a un dataset
- La llave “datasetId” esta en el JSON
- Hay 2000 datasets
- Solo es necesario cargar 100 datasets

La solución

Spark requiere

10 líneas de código

Qué es Spark?



Spark es una herramienta de Big Data general capaz de procesar datasets en archivos (locales y nube), bases de datos, unirlos, filtrarlos, manipularlos.

Puede ser procesado de forma local, un solo nodo

Procesar trabajos en un cluster
con resiliencia
escalable horizontalmente

Cuantos commits de mis empleados existen en Github

Los eventos en github tienen actores cuyo id es único por persona.

Filtrar por actor.id

Hay 8,440 empleados de mi empresa

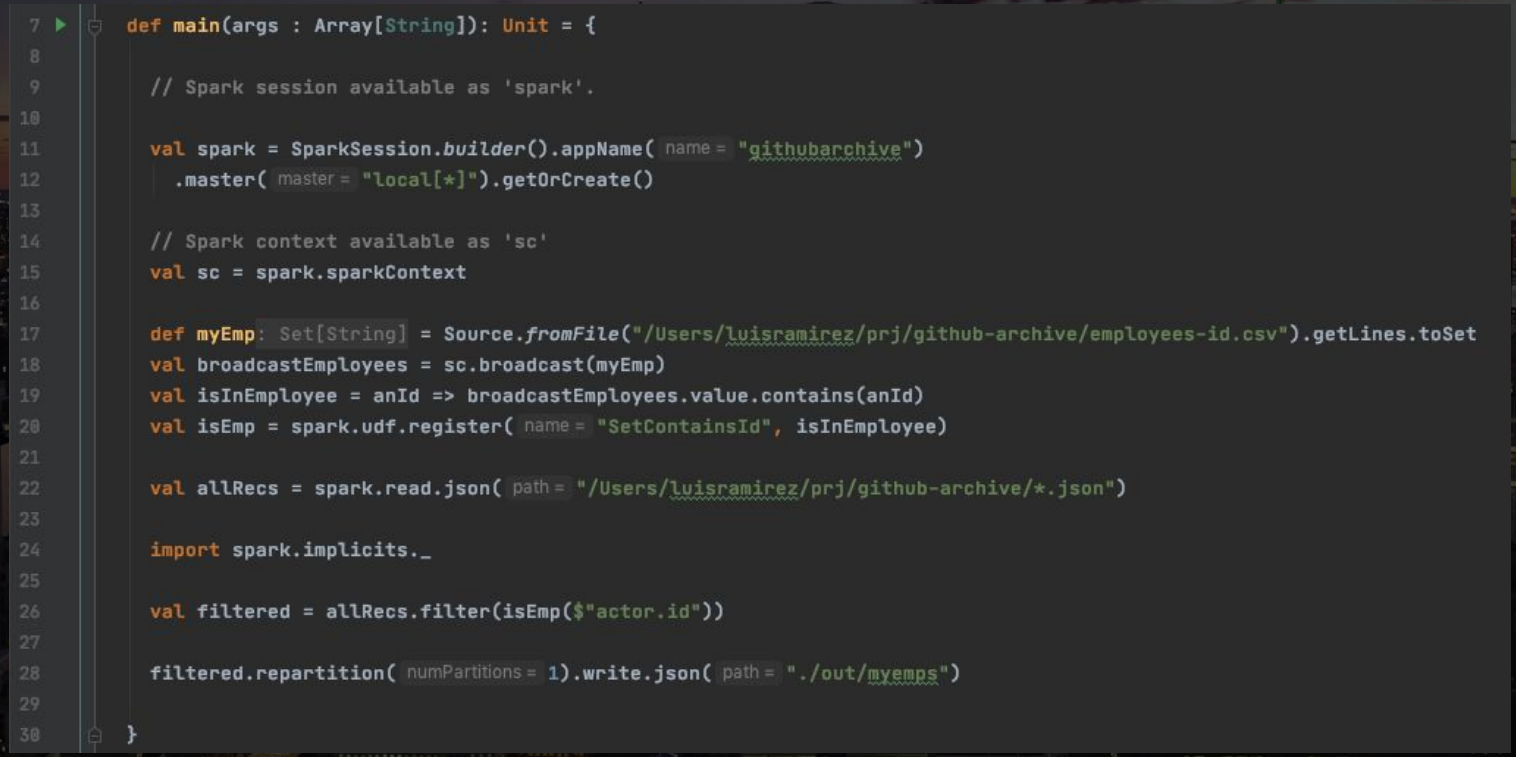
Hay 94,950 committers

```
root
 |-- actor: struct (nullable = true)
 |    |-- avatar_url: string (nullable = true)
 |    |-- gravatar_id: string (nullable = true)
 |    |-- id: long (nullable = true)
 |    |-- login: string (nullable = true)
 |    |-- url: string (nullable = true)
 |-- created_at: timestamp (nullable = true)
 |-- id: string (nullable = true)
 |-- org: struct (nullable = true)
 |    |-- avatar_url: string (nullable = true)
 |    |-- gravatar_id: string (nullable = true)
 |    |-- id: long (nullable = true)
 |    |-- login: string (nullable = true)
 |    |-- url: string (nullable = true)
```

Commits en

- wget http://data.githubarchive.org/2015-03-01-{0..23}.json.gz
- 24 archivos
- Formato JSON
- Cada linea es un record
- Hay 438,018 records

Demo



```scala
def main(args : Array[String]): Unit = {

  // Spark session available as 'spark'.

  val spark = SparkSession.builder().appName( name = "githubarchive")
    .master( master = "local[*]").getOrCreate()

  // Spark context available as 'sc'
  val sc = spark.sparkContext

  def myEmp: Set[String] = Source.fromFile("/Users/luisramirez/prj/github-archive/employees-id.csv").getLines.toSet
  val broadcastEmployees = sc.broadcast(myEmp)
  val isInEmployee = anId => broadcastEmployees.value.contains(anId)
  val isEmp = spark.udf.register( name = "SetContainsId", isInEmployee)

  val allRecs = spark.read.json( path = "/Users/luisramirez/prj/github-archive/*.json")

  import spark.implicits._

  val filtered = allRecs.filter(isEmp($"actor.id"))

  filtered.repartition( numPartitions = 1).write.json( path = "./out/myemps")

}
```

Conclusión


Este es José
El aprendió Spark
No como Luis
que uso Grep
y no durmió toda una noche
Se como José
imgflip.com