TOC

06 September 2024 12:23

Spark [29 Aug24]

29 August 2024 10:02

Types of big data

unstructured	- images - videos
structured	- rdbms - excel - spreadsheet
quasi structured	- web - clickstream
semi structured	- xml data with scheme - json, csv - *ORC, Parquet, AVRO (these are the ones we're gonna use)*

ORC (Optimized Row Columnar)

- **Type**: Columnar
- **Best For**: High-performance querying and storage efficiency in Hadoop environments.
- **Features**: Efficient compression, column pruning, and built-in indexing.
  takes 256MB by default

Parquet

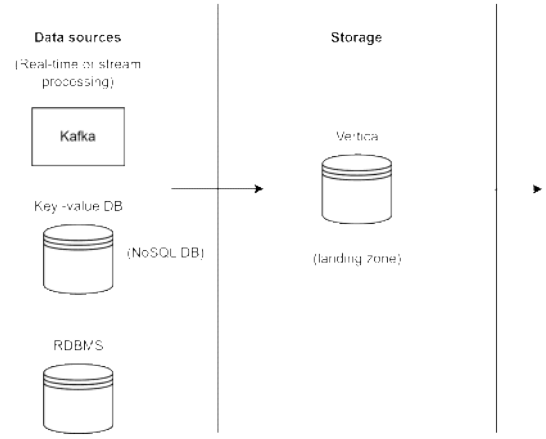
- **Type**: Columnar
- **Best For**: Versatile analytics and data storage across big data tools.
- **Features**: Flexible schema evolution, efficient compression, and broad support.

Avro

- **Type**: Row-based
- **Best For**: Data serialization and schema evolution, often used in data streaming (e.g., Kafka).
- **Features**: Compact format, efficient serialization, and schema evolution support.

Case study: Uber's infrastructure

Previous uber infrastructure



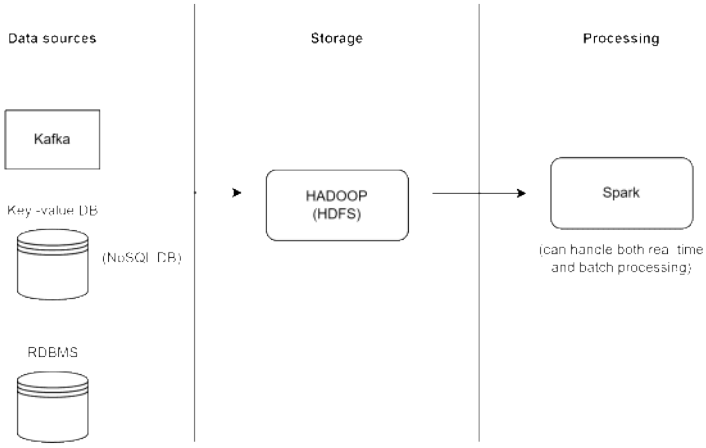
Features

- analytical data warehouse with goal of centralizing al data and facilitating data access
- vertica was chosen cos it was fast, reliable and column oriented.
- ETL - extract, transform and load

Problems:

- large number of files put strain on NameNodes
- data was accessible to users only once every 24 hours (slow realtime decisions)
- scalability limitations and high latency cos of hadoop architecture

New uber infrastructure



Features

- more data warehouse options
- hadoop upsets and Incremental (Hudi) introduced
- Parquet : faster output (since its a columnar storage)
- Hudi allowed users to take modified data incrementally (boosting efficiency, enabling incremental changes)

New term: Incremental load

- process of updating a dataset with only new/changed data rather than reprocessing entire dataset
- boosts efficiency

Data processing

- technique of manipulating info
- transformation of unstructured data into meaningful and readable info

- Types
  - batch processing
    - applying processing to a bulk amount of data as a single batch
    - data collected over a certain period
  - real time processing
    - applying processing to data that is captured at regular intervals
    - so data is processed as soon as it's generated

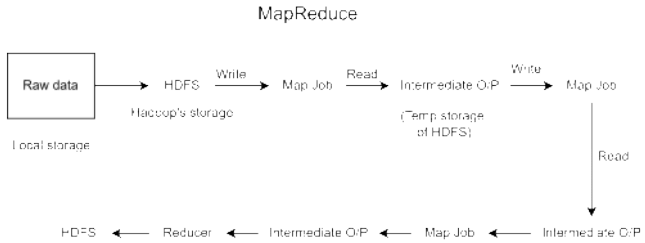
- [ ] checkout batch processing vs realtime processing in sirs notes

Limitations of MapReduce in Hadoop

- unsuitable for real time processing
- unsuitable for trivial operations
  - since they'd have to rewrite the data into k-v pairs
- unsuitable for large data on the network
  - since it works on in-disk operations
- unsuitable with OLTP
- unsuitable for processing graphs (property graphs)
- unsuitable for iterative execution

Why spark?

- This is what MapReduce process looked like



  - spark does not have intermediate outputs
  - it performs in-memory operations (in disk if too much memory required)
- Its a general purpose solution.

Why spark over Mapreduce?

interactive services	has its own Spark Shell

programming languages	hadoop uses Java while spark uses scala

batch processing	spark batch can be used instead
structured data analysis	its data frames are simple and can be used for quick analysis
ML analysis	can be used for clustering, recommendations and classification
Interactive SQL analysis	SQL over Impala
Real time streaming data analysis	Spark streaming is enough for this instead of specialized libraries like Storm

Why spark over Apache Storm? (refer to storm here)

- Storm can be complicated for developers
- spark can help with multiple processing problems (batch, real time)
- spark provides uniform model ML library that runs on top of its core engine

Spark components

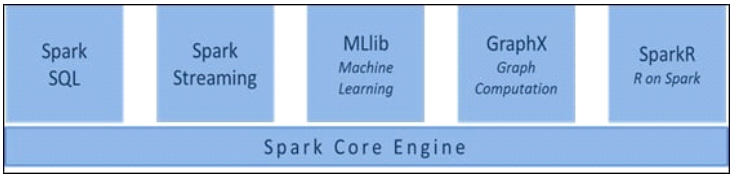
- python
- java
- scala
- ML lib
- R
- Graphx
- streaming
- SQL

Spark features

- suitable for real time applications
- processes larger data on a network
- opensource cluster computing framework
- in memory computation
- up to 100 times faster performance to other in memory computations (and upto 10x for in-disk computations compared to MapReduce)
- suitable for ML algorithms

Components of a spark project

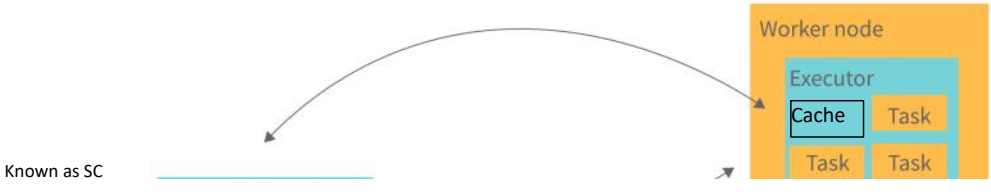
- spark SQL
- spark Streaming
- MLLib
- GraphX
- SparkR
- Spark Core

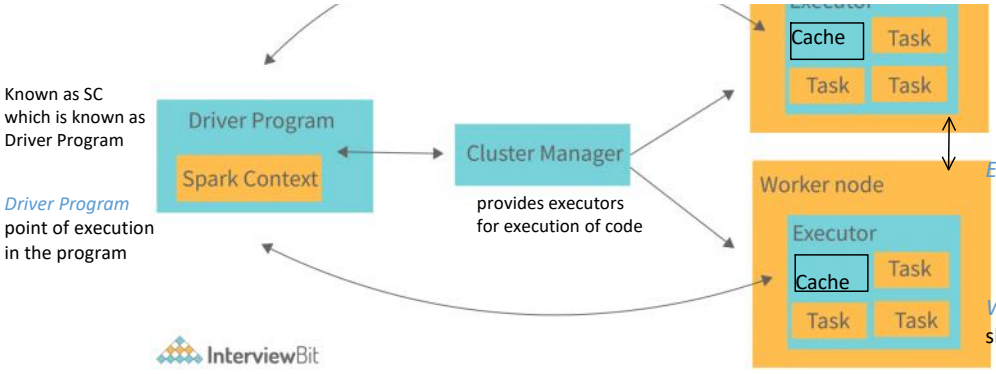


Spark Architecture

Types of cluster manager

- spark standalone
  - launched by manually launching scripts or starting a master and workers
- YARN
- Mesos (3rd party)
  - advantage - scalable partitioning among different Spark instances





- Mesos (3rd party)
  - advantage - scalable partitioning among different Spark instances

Executors
- uses cache slots to keep data in memory
- run tasks scheduled by driver

Worker node
slave nodes

Driver Program
point of execution in the program

Note
Everything that you write in Spark (Python, Scala or Java) gets converted to Java Bytecode

spark context

- it's the crucial component which represents connection to spark cluster
- entry point for using spark APIs, spark methods
- responsible for managing lifecycle of spark application
- | hadoop/ mapReduce jobs | spark application |
- resource management
  - manages allocation of resources which are required by executors (worker nodes)
- this manages configuration for spark application. for example, memory, number of executors, CPU
- RDD
  - resilient distributed datasets
  - fundamental data structure of Spark

Cluster Manager

- it schedules and dispatches tasks to the worker node. it also ensures that the tasks are distributed among the worker nodes and managed efficiently
- scaling
  - its responsible for scaling the resources up or down depending on workload or demand
- fault tolerance
  - handles node failures and resource allocation to maintain stability and reliability of the cluster
- if the job fails, this can reschedule the tasks
- resource management
  - monitors and manages clusters resources efficiently

RDDs

- immutable collection of objects which defines data structure of Spark

rdd.repartition()	to increase number of partitions
rdd.coalesce()	to reduce number of partitions

Features

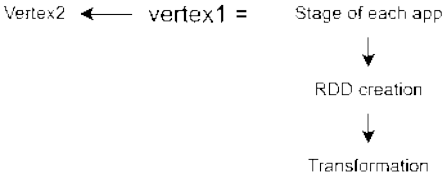
- lazy evaluation
  - parallelized collections
    sc.parallelize()

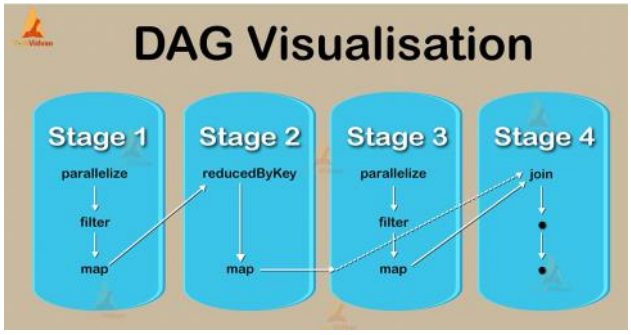
- existing RDDs
- external Data
  `sc.textFile()`

- coarse grained evaluation
- in memory computation
- fault tolerance
- partitioning
- persistent
- immutable
- location - stickiness

DAG

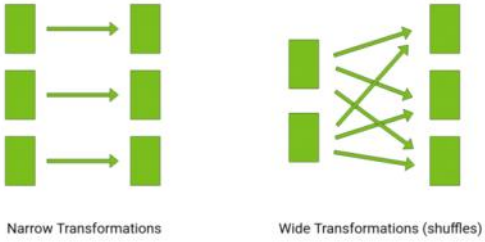
- Directed Acyclic Graph
- graph where RDDSs and operations to be performed on them are represented in the form of vertices and edges

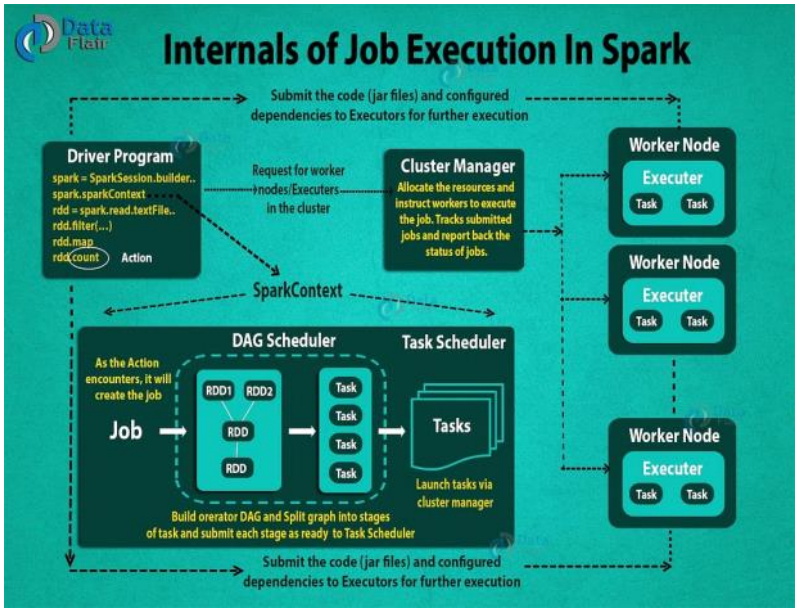


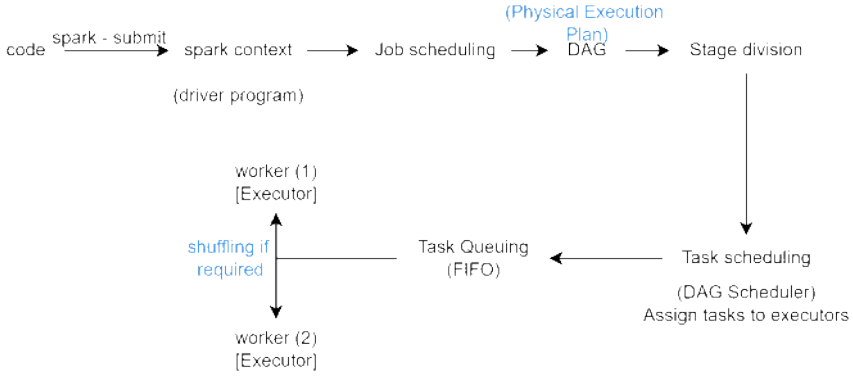


Types of transformation

narrow	wide
self sufficient	not self sufficient
result of map() and filter() (non-aggregate transformation)	result of GroupByKey() and ReduceByKey() (aggregate transformation)
data is from single partition	data is from multiple partitions






code
spark - submit
spark context
(driver program)
Job scheduling
(Physical Execution Plan)
DAG
Stage division
Task scheduling
(DAG Scheduler)
Assign tasks to executors
Task Queuing
(FIFO)
shuffling if required
worker (1)
[Executor]
worker (2)
[Executor]

Spark Practical [30 Aug 24]

30 August 2024 14:21

note:
u cant run more than one spark session in sir's vm

when you start spark shell (or pyspark), it starts 2 sessions by default

- spark session (sql)
- spark context
    - compulsory
    - needed for running even a small program

```
hadoop@hadoop-VirtualBox:~$ spark-shell
24/08/30 14:24:44 WARN util.Utils: Your hostname, hadoop-VirtualBox resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instea
d (on interface enp0s3)
24/08/30 14:24:44 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
24/08/30 14:24:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
ere applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1725008106762).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.8
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_91)
Type in expressions to have them evaluated.
Type :help for more information.

scala> println(sc)
org.apache.spark.SparkContext@319ea996

scala> println(spark)
org.apache.spark.sql.SparkSession@1bcdd302

scala>
```

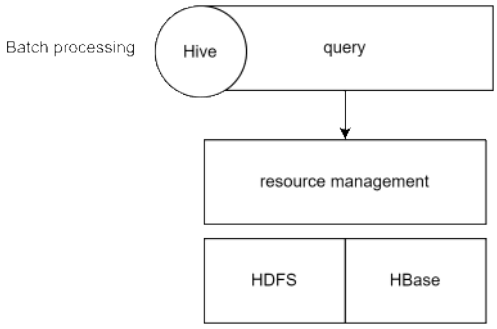

StorageLevel	Description	Space used	CPU Time
MEMORY_ONLY	Data stored directly as objects and stored only in memory	High	Low
MEMORY_ONLY_SER	Data is serialized, reducing its size, and stored only in memory. However, deserializing the data has a cost.	Low	High
MEMORY_AND_DISK	Data is stored directly to memory, but serialized and stored on disk if memory is insufficient	High	Medium
DISK_ONLY	Data is serialized and stored on disk	Low	High
MEMORY_AND_DISK_SER	Same as MEMORY_AND_DISK, however data is also serialized to be stored in memory	Low	High

Spark and Hive [4 Sept 24]

04 September 2024 09:18

Hive

SQL over Hadoop MapReduce



Hive query -> runs on top of Hadoop Services -> execute each query as mapreduce job

definition

- SQL-like
- open source
- data warehousing app
- extracts data from Hadoop and related systems

features

- open source
- provides high level abstraction layer on top of MapReduce and Spark
- uses HiveQL
- suitable for structured data
- can be used on unstructured data too

datatypes supported by Hive

- int
- tiny/small/big int
- boolean
- float
- double
- string
- array, map, struct, union
- decimal
- char
- varachar
- date
- binary
- timestamp

SQL semantics supported by Hive

- select, load, insert
- where, having
- group/order
- cluster by (for buckets), distribute by
- sub queries
- join op
- window functions
- exists/not
- rollup
- union

load query in Hive

used to insert data into Hive table

```
load data inpath 'hdfs path'
```

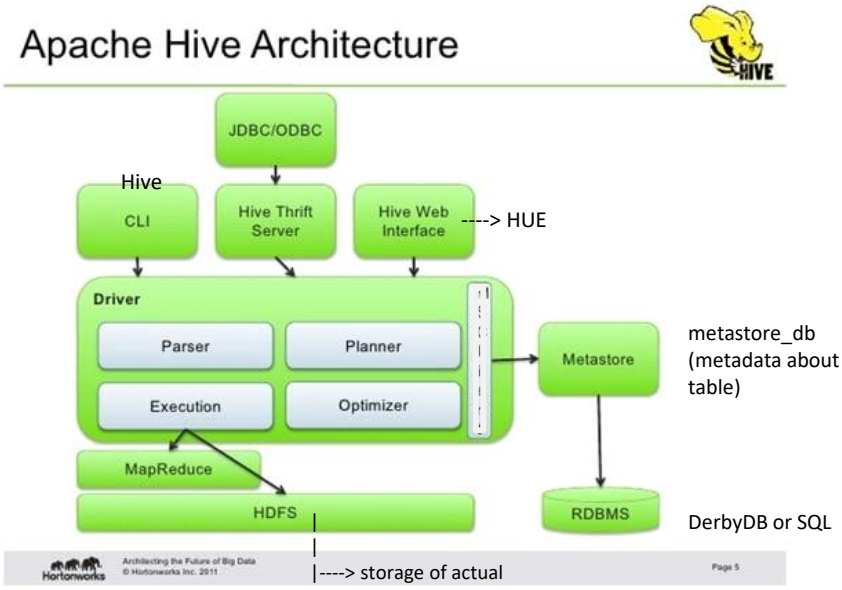
2 things are created when u use this:

- metadata
- warehouse in HDFS

insert query in hive

used when we want to load data from existing table into new table

Hive architecture



Ways to connect to Hive

- .hql fileformat
- hive CLI
- beeline (multi user CLI)
- HUE (web based editor -Hive's web interface)

Hive thrift server

- it uses beeline, which has its own CLI that lets multiple users use server at the same time

(Hive CLI lets only one user use it at a time)

Planner

planning the physical execution plan to how it should be executed

RDBMS

metastore is used to store metadata explains how it is stored

Whats JDBC & ODBC?

- database connectors
- JDBC (java database connector) connects Derby DB
- ODBC (open database connecitvity) connects OracleDB

- external resources like Java and Spark also connect using JDBC

(in pyspark jupyter nb, we used it, refer to it)

Job execution flow

Drivers

- parsers
  Hive query ---> parser ---> checks syntax, error, schema and its datatype
  if error is not found, it will convert into DAG

- optimizers
  - optimizes the query
  - catalyst can be added too for faster execution

- executor
  - it splits job into multiple tasks and executes them
  - responsible for fetching the DAG (Physical execution plan) and converts it into a MapReduce job.

Steps

1. Parse HiveQL
2. make optimizations
3. plan execution
4. submit jobs to cluster
5. monitor progress
6. process data in MapReduce or spark
   a. if run in spark, it runs as a spark job
   b. if run in MapReduce,
7. display it or store data in HDFS

steps 5-7 are YARN procedures

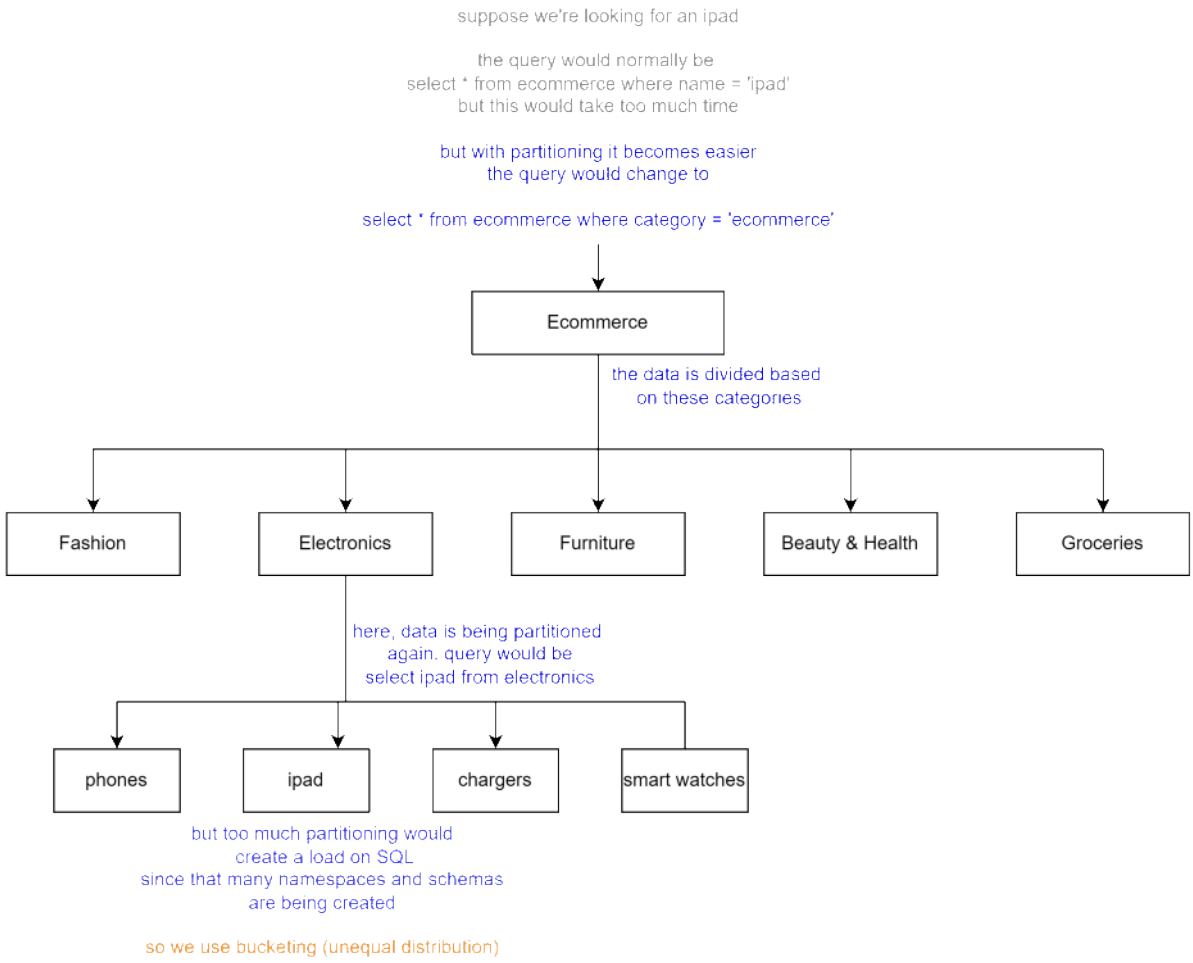
Partitioning and bucketing

one of the main reasons why Hive is still prominent

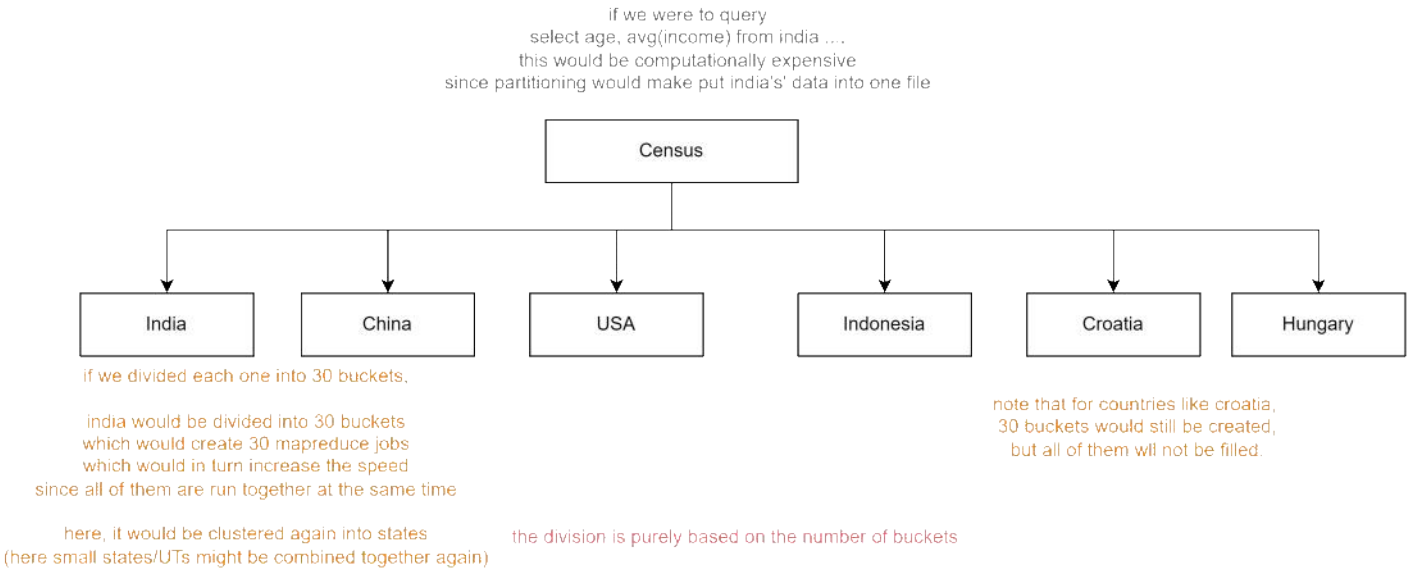
the main goal here is to query faster (in mongodb its done via indexing)

technically, partitioning is mapping and bucketing is reducer

Partitioning use



Bucketing



here, we're not sure how the data is divided in Buckets

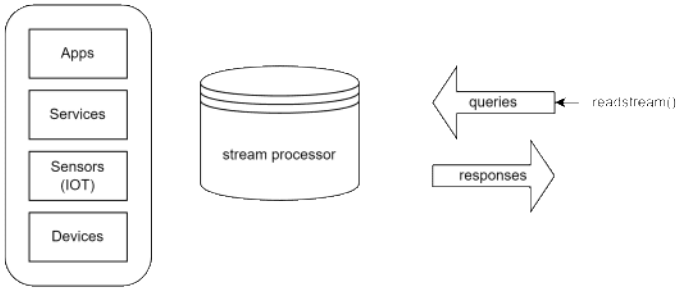
Spark Streaming [4 Sept24]

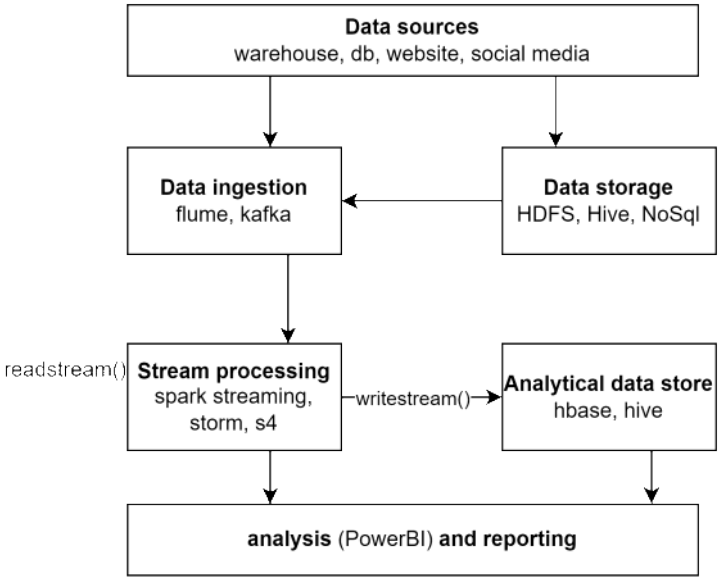
04 September 2024 12:52

what's streaming?

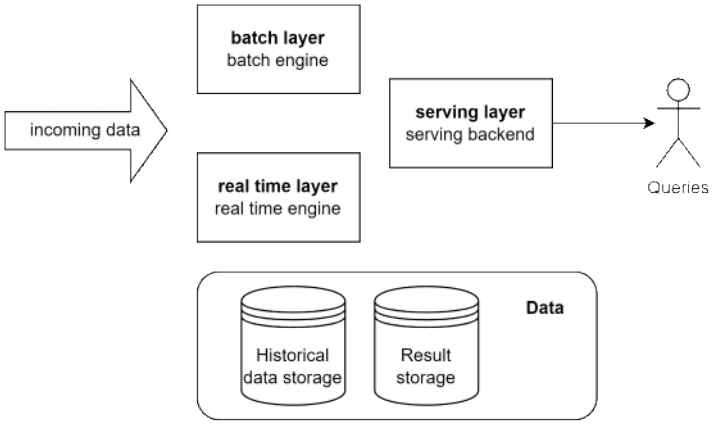
continuous flow of data at high speed rate

volume	batch processing
velocity	stream processing





readstream() and writestream() is for structured streaming



Spark Streaming architecture

Kafka

- acts as broker

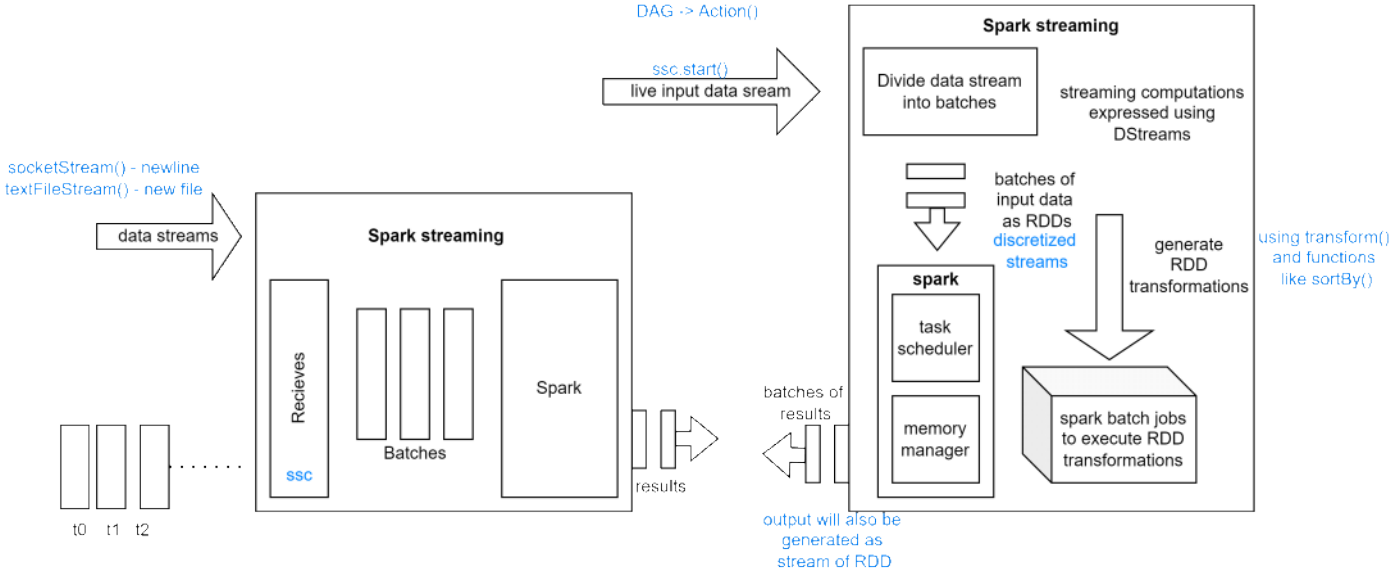
- refer to its architecture here

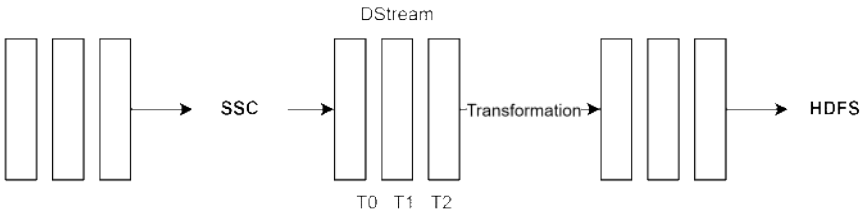
SSC
Spark Streaming Context
entry point to spark streaming

Features

scaling	can scale to 100s of nodes
speed	low latency
fault tolerance	efficiently recovers from failures
integration	works with both batch and real time data
business analysis	can track customer support

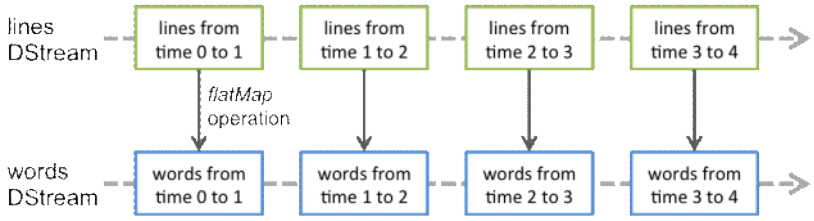
Incomplete spark streaming workflow





Dstreams

- fundamental abstraction in spark streaming
- a series of RDDs that characterizes it



transformation functions on DStreams

- map
- flatMap
- filter
- repartition
- union
- count
- reduce
- countByValue
- reduceByKey
- join
  - returns DStream of (k, (v,w)) with all elements for each key when being called on 2 DStreams of (K,V) and (K,W) pairs.
- cogroup (refer pyspark RDD notebook for example)
- transform
  - returns new DStream by applying an RDD-to-RDD function to every RDD of the source DStream
  - an alternative would be foreachRDD() but its an alternative function

for all these functions except transform, the input and output are DStreams

output functions

- print()
- saveAsTextFiles()
- saveAsObjectFiles()
- saveAsHadoopFiles()

MLlib Spark [5 Sept 24]

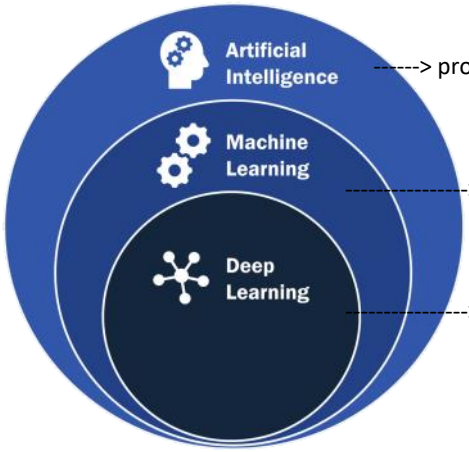
05 September 2024 10:18

Types of analytics

- descriptive
  - describes the past and answers 'what happened?'
  - data aggregation(EDA, statistical models) + data mining

- predictive
  - can predict the future and answers 'what might happen?'
  - statistical model (all ML algos) + forecast technique

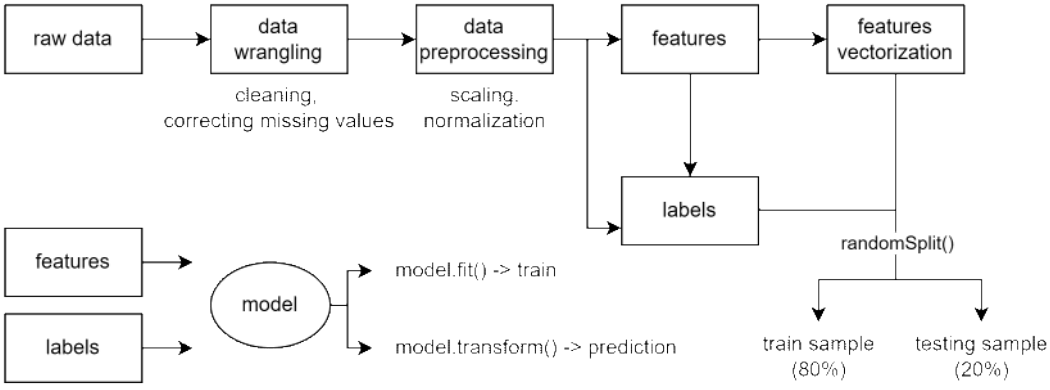
- perspective
  - optimization + simulation algorithms
  - advices users on possible outcomes and answers 'what should be done'



-----> program that mimics human/animal decision making ability

----------> algos improves performance as they're exposed to more data over time

----------->inspired by functionality of neurons, which led to the concept of ANN



why MLlib even when there's other good packages ?

- scalable ML library
- consists of common learning algos, tools and optimization techniques

Tools

- ml algos
- featurization
- *pipelines*
- persistence
- utilities

Algorithms

- classification
- regression
- clustering
- basic statistics
- optimization
- feature extraction
- recommendation (association rule)
- reduction

StringIndexer()

maps variables to numbers

gender	
male	1
female	0

contract	
month to month	0
one year	1
two years	2

One hot encoder

contract			
	month to month	one year	two years
month to month	1	0	0
one year	0	1	0
two years	0	0	1

for each row, only one value is 1 and rest are 0.

Spark optimization [6 Sept24]

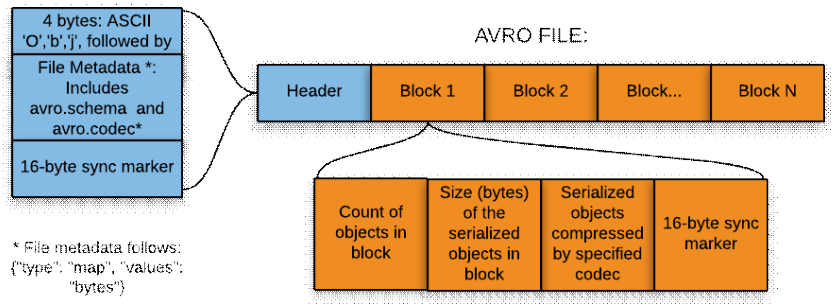
06 September 2024 09:31

Types of bigdata fileformats

refer to this for some basic info

AVRO

- row based storage format used for Hadoop
- stores schema in JSON format, making it easy to read and interpret
- data is stored in a binary format making it compact and efficient
- no compression in this format, but ideal for landing zone (for faster reads and transforming the data)



Parquet

- open source file format for Hadoop
- stores nested data structures in flat columnar format

- compared to traditional approach (where data is stored row wise), this is more efficient in terms of storage and performance
  - for example, lets say we are trying to get a list of names. in a row wise structure like sql, first we take the entire dataset then filter out the rest and take the names column. parquet would just take that single column

-

ORC - optimized row columnar

- the most efficient one
- it stores collections of rows in one file
- within the collection, the row data is stored in a columnar format

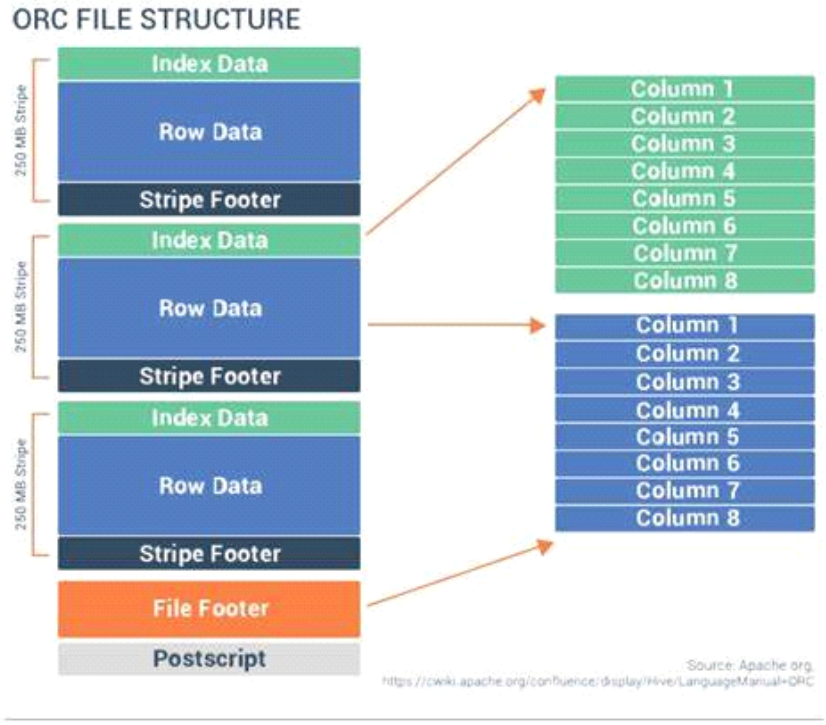
- each one is 250 MB in size

- File footer contains
  - list of stripes in the file
  - number of rows per stripe
  - each columns' data type (schema)
- row data
  - used in table scans

landing zone

- intermediate storage area used during data processing, particularly in the ETL process.
- acts as a temporary holding area for data before it is moved to its final destination, such as a data warehouse.

- stripe footer
    - contains directory of stream locations
    - also contains column-level aggregate values like count, min, max.

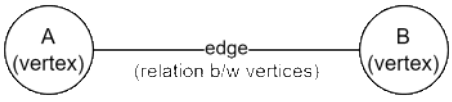


Spark GraphX [6 Sept24]

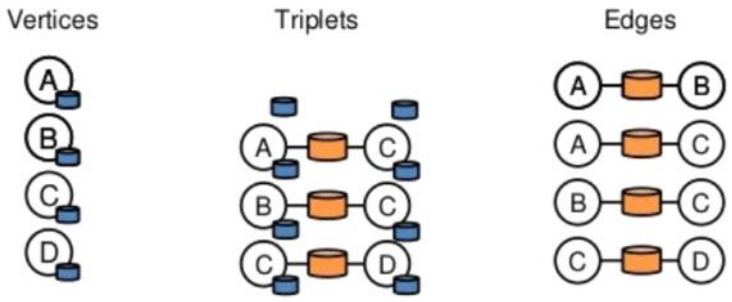
06 September 2024 12:22

Graph

- a set of points that are interconnected by lines



Components of Graph



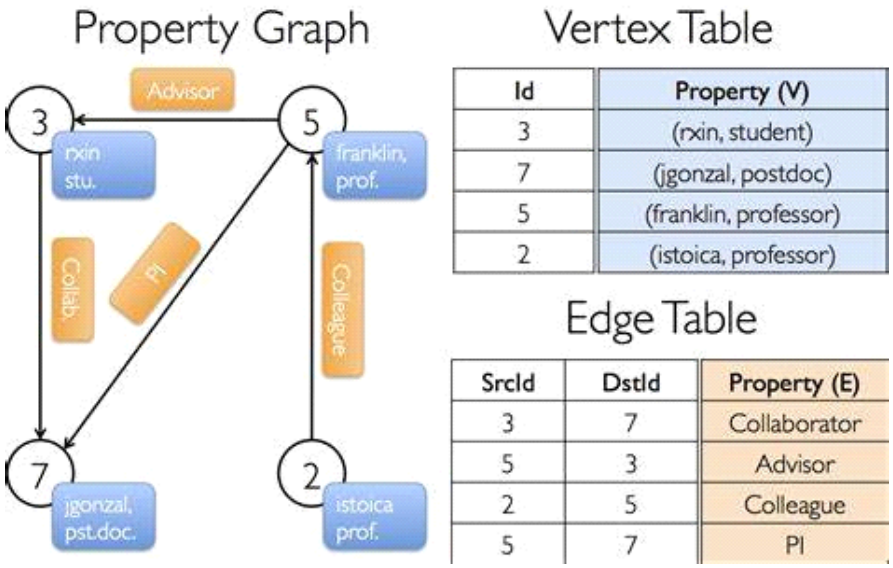
Use cases of GraphX

- fraud detection
- page rank
- disaster detection system
- business analysis
- geographic information system
- google Pregel

widely used in scala programming

Types of graphs

- undirected graph
- directed graph
- vertex labeled graph
- edge labelled graph
- cyclic graph
- weighted graph
- directed acyclic graph disconnected graph



Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

Features of GraphX

- Realtime processing framework
- extends RDD abstraction to RDG

References

06 September 2024 09:42

Kafka architecture