Introdução ao Big Data

Big Data - Histórico

Os 5 V’s do Big Data

- Volume
- Variedade
- Velocidade
- Veracidade
- Valor

Big Data - Histórico

2003 - GoogleFS (GFS) – Sistema de arquivos distribuído

- Cluster, master, chunkserver
- Tolerante a falhas
- Composto por maquinas comuns em geral

2006 – Projeto Hadoop -

- Inspirado no GoogleFS
- Idealizadores Doug Cutting e Mike Cafarella
- Iniciou como Apache Nutch, migrando para Hadoop em 2006

- https://research.google/pubs/pub51/
-

Big Data - Histórico

Cluster Hadoop

- NameNode –
  - metadados dos arquivos, arvore de diretorios do file system
  - Acesso aos dados passa por requisição ao NameNode
- Secundary NameNode
  - Tarefas de manutenção. (housekeeping),
  - Pontos de verificação (checkpointing)
- DataNode
  - Administração blocos HDFS
  - Informa Saude/Status individual ao NameNode

Big Data - Histórico

YARN – Gerenciador de recursos do cluster

- Resource Manager – master
  - Aloca/monitora recurso do cluster
  - Escalonamento dos jobs do cluster
- ApplicationMaster - master
  - Coordena aplicação executada no cluster, escalonada pelo resource manager
- Node manager - worker
  - Executa tarefas de processamento em um nó individual
  - Informa Saude/Status individual
  -

Big Data - Histórico

HDFS – Sistema de arquivos distribuido

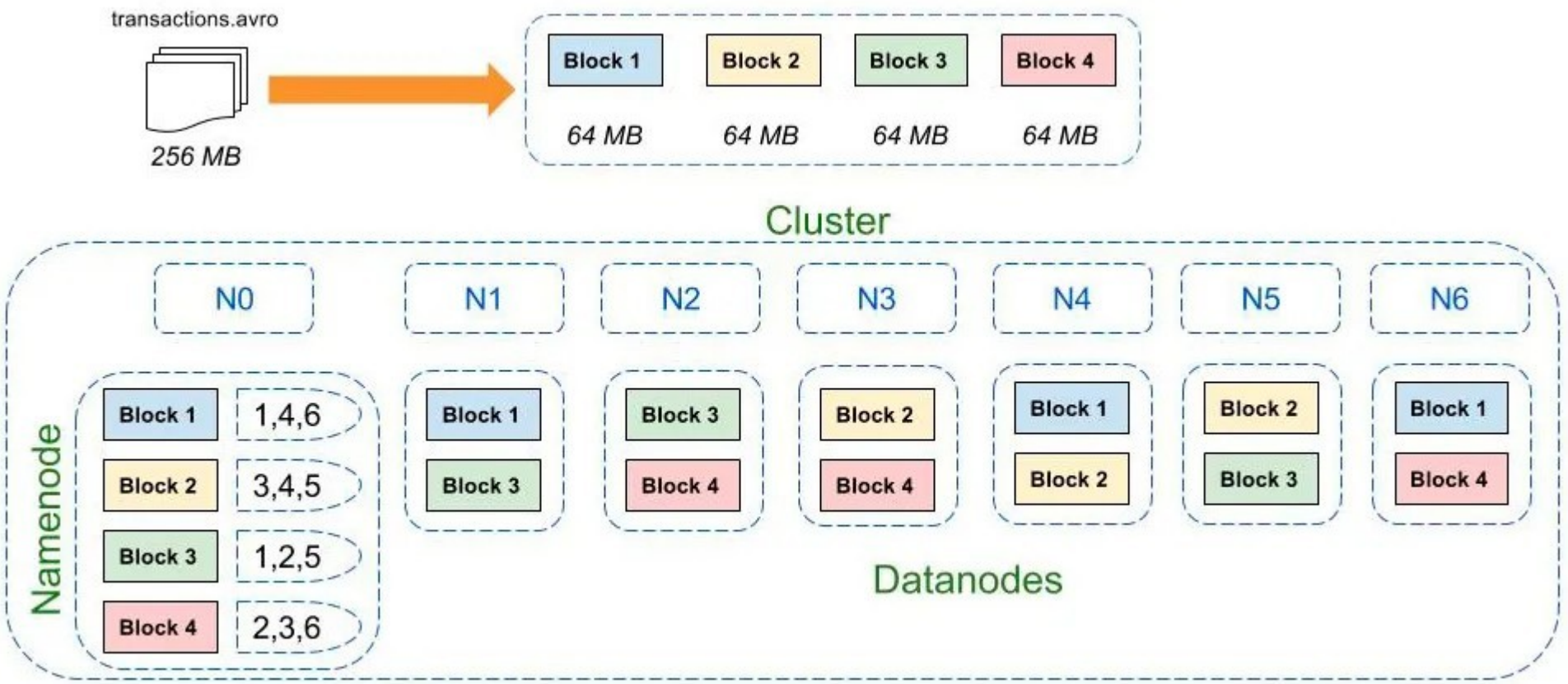
- Armazenamento redundante, computadores baratos, não confiaveis
- Camada de software sobre sistema de arquivos linux nativo
- Otimizado para leitura de grandes arquivos
- Arquivos divididos em blocos de 64MB, 128MB, 256MB
- Blocos replicados 3 vezes por padrão
- Interação com arquivos feita via linha de comando

Map Reduce – modelo de programação distribuida

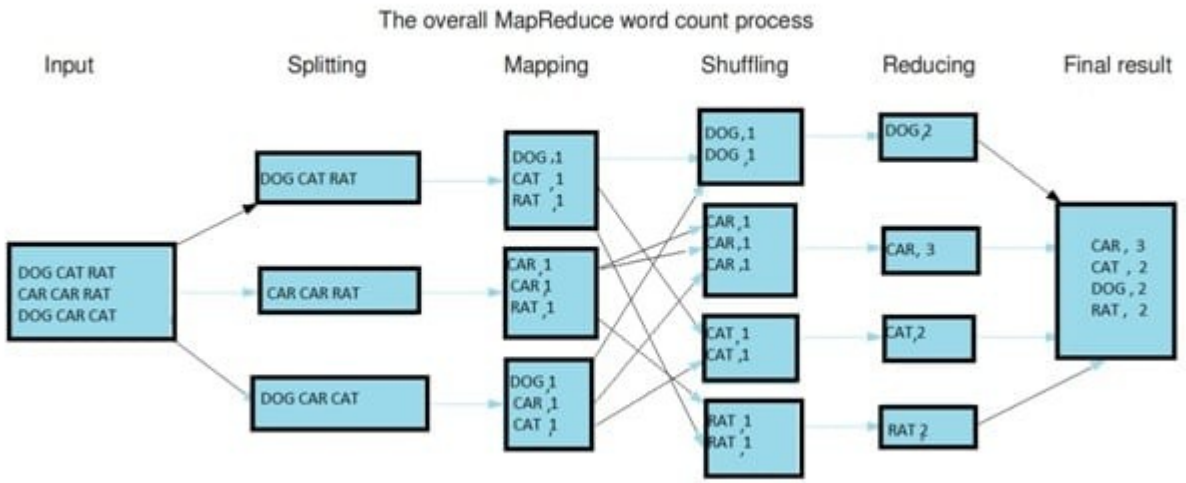
- Permite computação distribuida, tolerante a falhas em cluster
- Programação paralelizada, tarefas independentes executam porçoes de dados locais e agregam o resultado após o processamento
- Saida de uma função é a entrada da próxima função

Big Data - Histórico

HDFS – Sistema de arquivos distribuido



Big Data - Histórico



Map Reduce – exemplo

- Fase 1 - mapeamento como chaves e valores
- Fase 2 - mapeadores geram zero ou mais chaves valor, mapeando valor para cada chave
- Fase 3 - chaves/valores ordenado e embaralhados com base na chave e enviados para o redutor
- Fase 4 - Redutores criam zero ou mais chaves valores finais, a saida

Big Data - Histórico

Desvantagens do Hadoop

- Map reduce usa escrita de dados locais
- map reduce produz muitos dados intermediarios
- Programação do Map Reduce pode ser complexa

Apache Spark - Teoria

O que é o Apache Spark

- Framework 100% open .
- Processamento de dados distribuido.
- Projetado para grande volumetria de dados.
- Suporte para dados estruturados
- Streaming
- Grafos
- Machine Learning

Apache Spark - Teoria

Algumas caracteristcas do Apache Spark

- Veloz
- Simples
- Modular.
- Extensível

Apache Spark - Teoria

Pq o Spark é Veloz ?

- Armazenamento em Memória .
- Catalyst Optimizer.
- Lazy Evaluation
- DAGScheduler

Apache Spark - Teoria

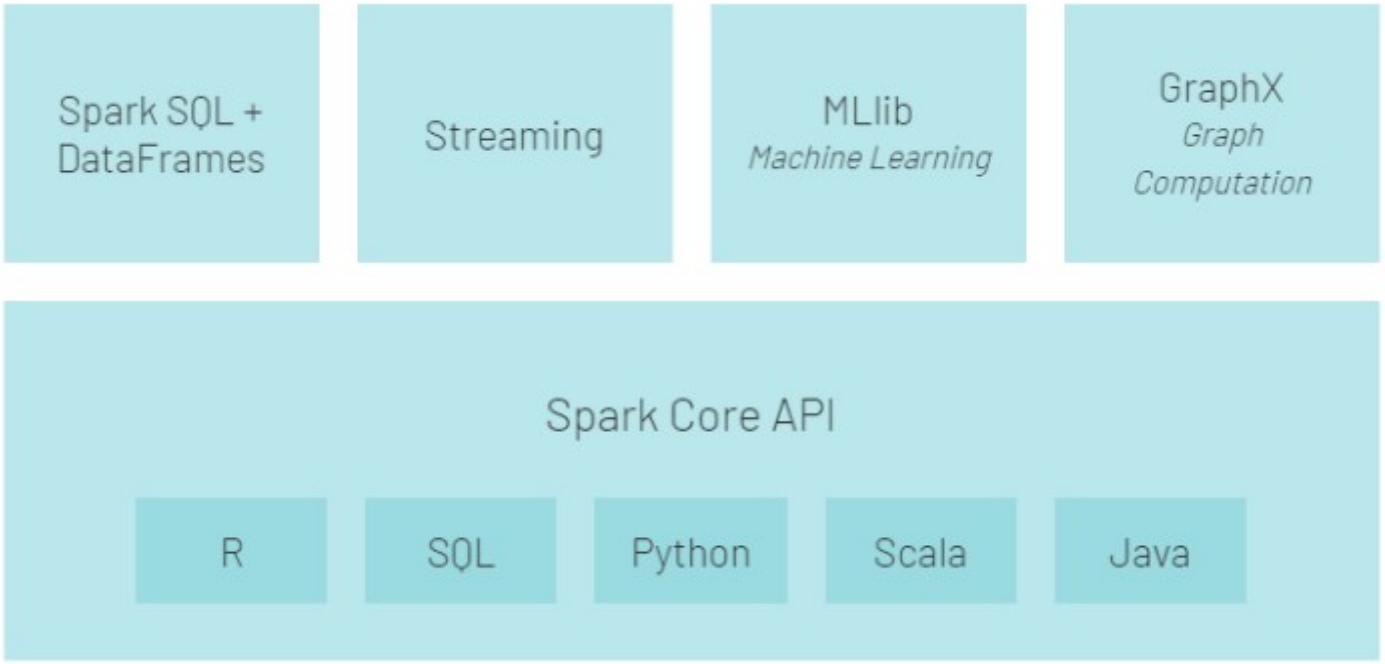
Pq o Spark é Simples ?

- Dataframes .
- Dataframe são fortemente baseados em SQL

Apache Spark - Teoria

Pq o Spark é Modular ?

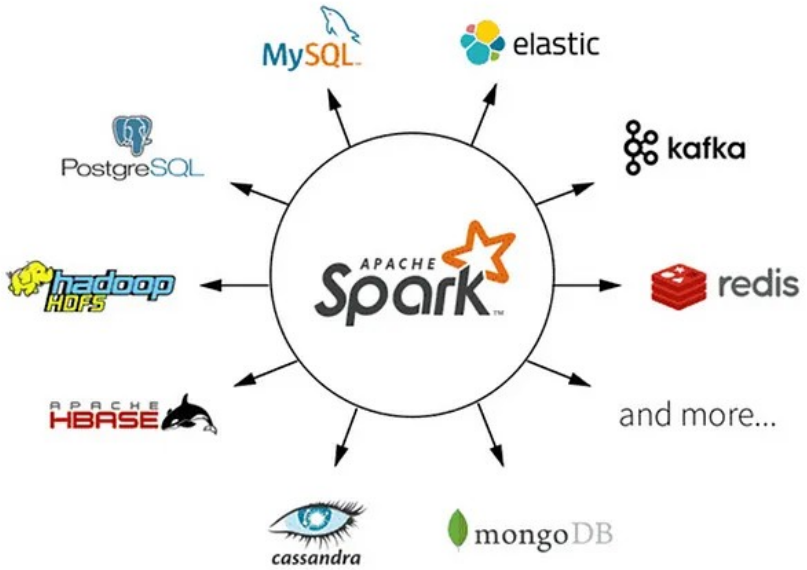
- Spark SQL – Modulo de processamento de dados estruturados
- Structured Streaming – Modulo de processamento de Streaming
- Spark ML – Machine learning
- GraphX – Computação de Grafos



Apache Spark - Teoria

Pq o Spark é Extensível ?

- Spark é focado somente no processamento de dados
- Tem conexão com várias outras ferramentas

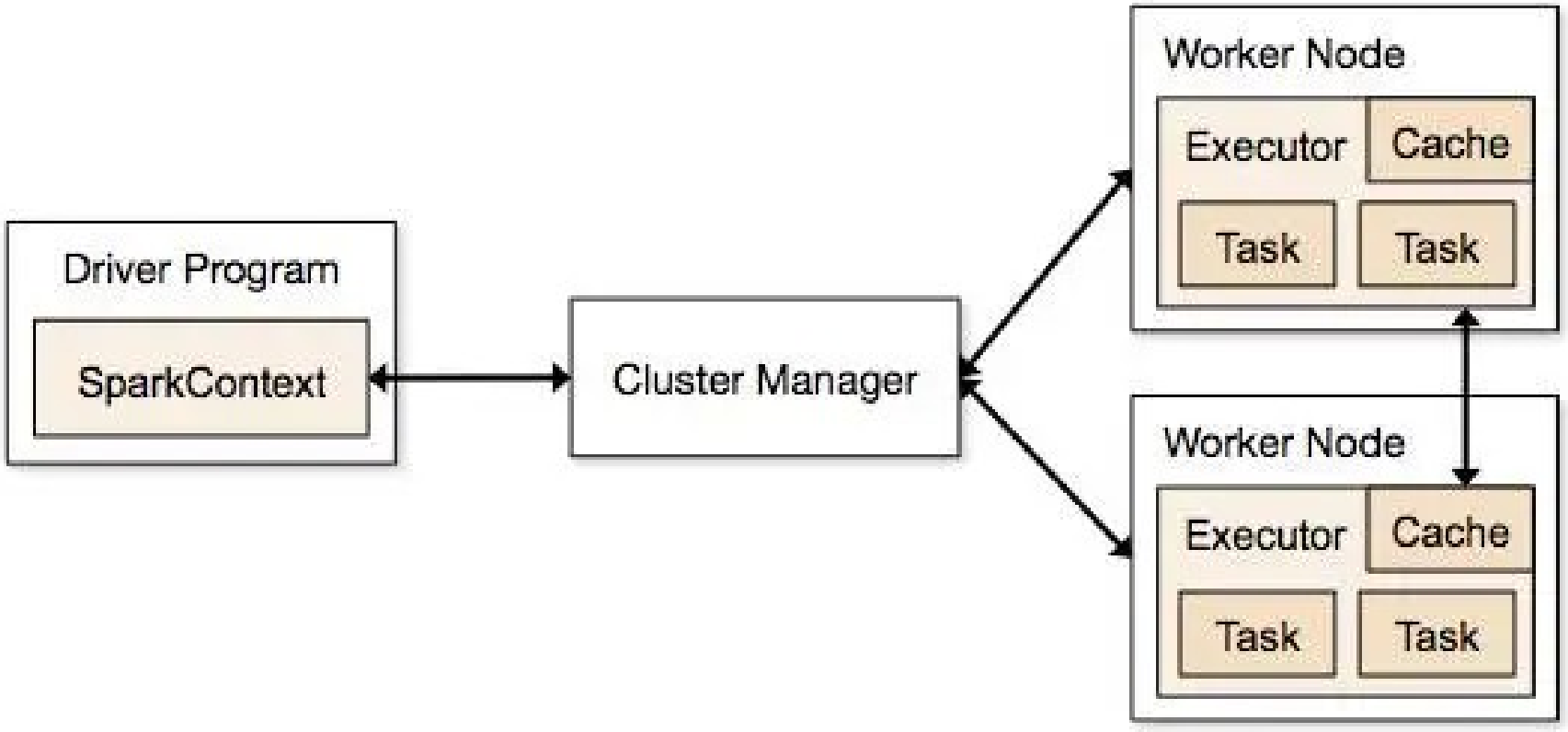


Apache Spark - Teoria

Arquitetura Spark

- Driver process :
  - Executa a Spark Application
  - Mantem as informações do Spark Application
  - Responde consulta do usuário
  - Agenda e distribui tarefas aos executores
- Cluster manager :
  - Executa o codigo em vários nós
  - Gerencia o estado e saude dos processos
  - YARN / Mesos
- Executors process:
  - Armazenamento dos dados/execução do código nos dados distribuidos
  - Reporta os estados das tarefas e resultados ao driver

Apache Spark - Teoria



Apache Spark - Teoria

Modos de execução do Spark

- Modo Cluster
  - Codigo spark submetido ao cluster manager
  - Cluster manager inicializa processo driver em um nó do cluster
  - Cluster manager inicializa processos executors em outros nós do cluster (workers)

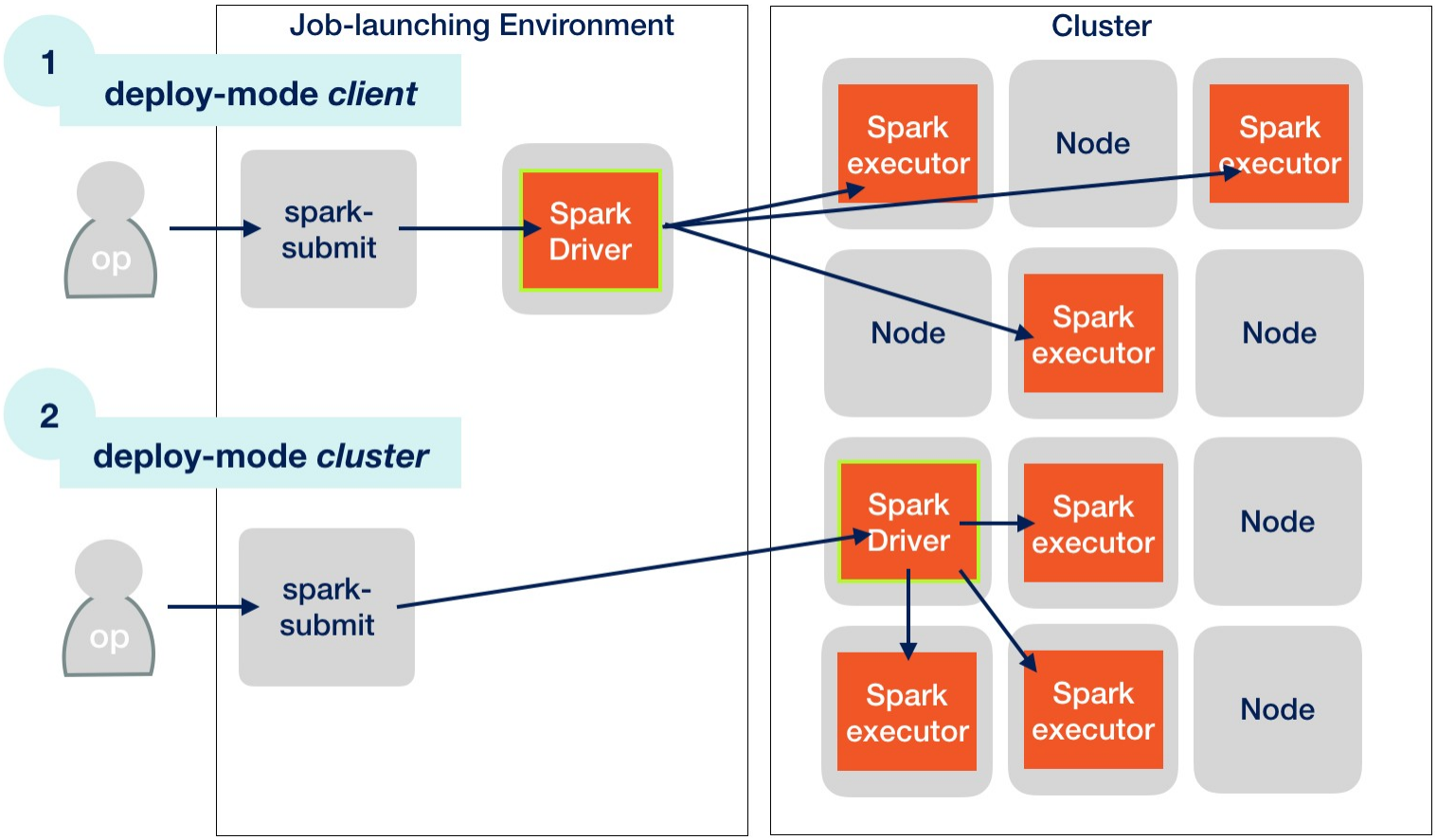
Apache Spark - Teoria

Modos de execução do Spark

- Modo Client
  - Processo driver se encontra numa maquina fora do cluster
  - Maquina client é responsável por gerenciar processo driver
  - Cluster manager atua no gerenciamento dos processos executors dentro dos nós do cluster (workers)

Apache Spark - Teoria

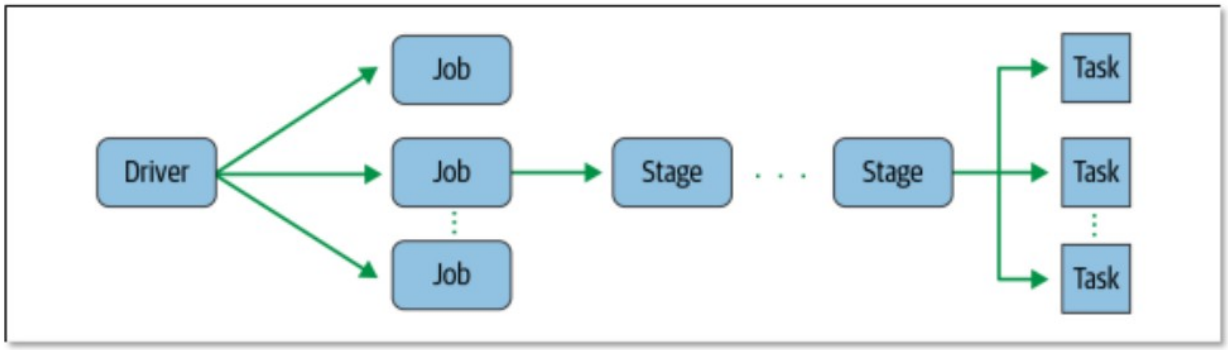
Modos de execução do Spark



Apache Spark - Teoria

Etapas de uma execução Spark

- Jobs
  - Um job para cada ação, ações engatilham transformações para retornar um resultado
- Stages
  - Grupos de task identicas que podem ser executas em conjunto
  - Cada novo processo de shuffe representa um novo Stage
- Tasks
  - Unidade de processamento aplicada a uma partição de dado



Fonte: Learning Spark (Cap. 2, p. 28)

Apache Spark - Teoria

SparkContext e SparkSession

- SparkContext
  - Programa do driver para interagir com o cluster
  - Cria os RDD – Resilient Distributed Datasets
  - Versões 1.x
  - Qualquer fonte de dados
- SparkSession
  - Programa do driver para interagir com o cluster.
  - Encapsula o SparkContext
  - Cria RDD, Dataset e Dataframes
  - Versão 2.0 em diante

Apache Spark - Teoria

Interfaces do Apache Spark

- RDD – Resilient Distributed DataSet
  - Conjunto de dados distribuidos e resilitentes
  - Dados primitivos e outros objetos , imutavel
  - Dados Estruturados, Não estruturados, sem esquema
  - Qualquer fonte de dados
- Dataframe
  - Coleção de dados distribuidos, resilientes
  - Organizado em colunas nomeadas, conteito BD relacional
  - Dados estruturados, semi estruturados , como esquema
  - AVRO,CSV, JSON, HDFS, PARQUET, DELTA, SQL, etc

  DataSet
  - Coleção de dados distribuidos, resilientes
  - Extensão do Dataframe, Dataframe + RDD

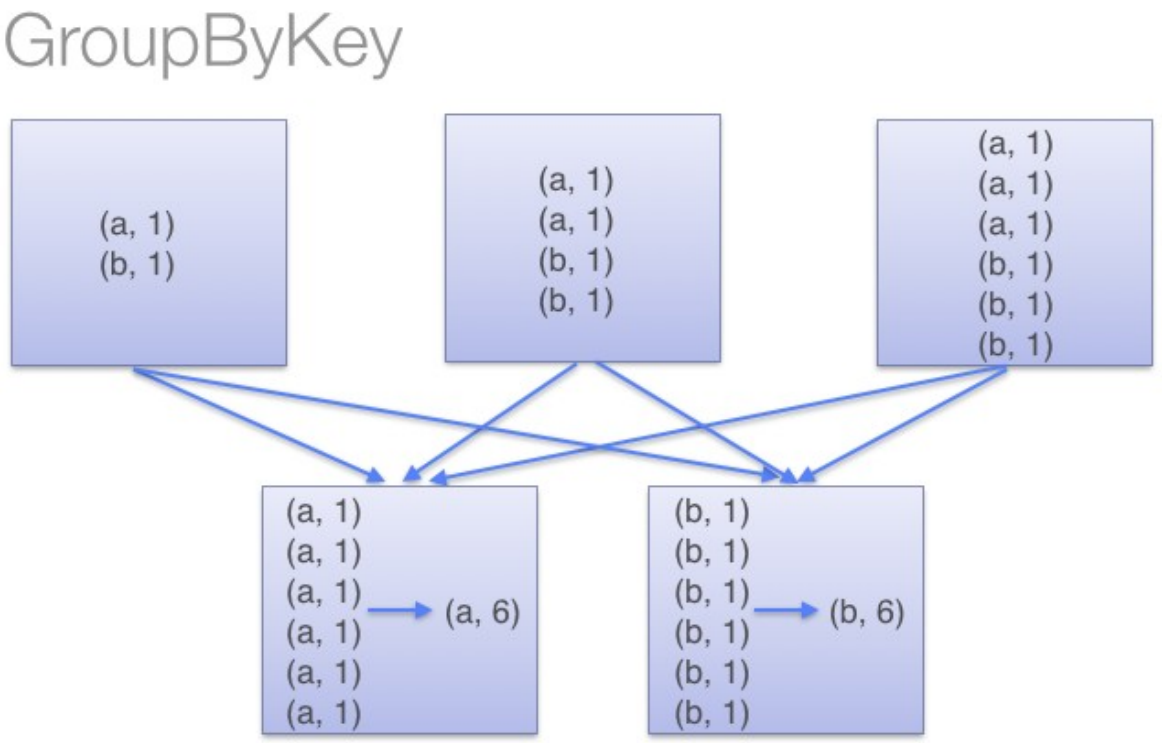
Apache Spark - Teoria

Operações do Apache Spark - Transformações

- Transformações operações sobre um dataframe spark
- O resultado de uma transformação é um novo dataframe
- Não são executados até que uma ação seja executada
- Algumas transformações no Apache Spark
  - Select(), filter(), withColumn()
- Transformações Narrow
  - Dados necessários residem no máximo em uma partição do RDD
- Transformações Wide
  - Shuffle – agrupamento de dados e redistribuição entre partições
  - Transferencia de dados pela rede,serialização e deserialização
  - Join , order by, group by acionam o shuffle

Apache Spark - Teoria

Exemplo de Shuffle em uma transformação Wide



Apache Spark - Teoria

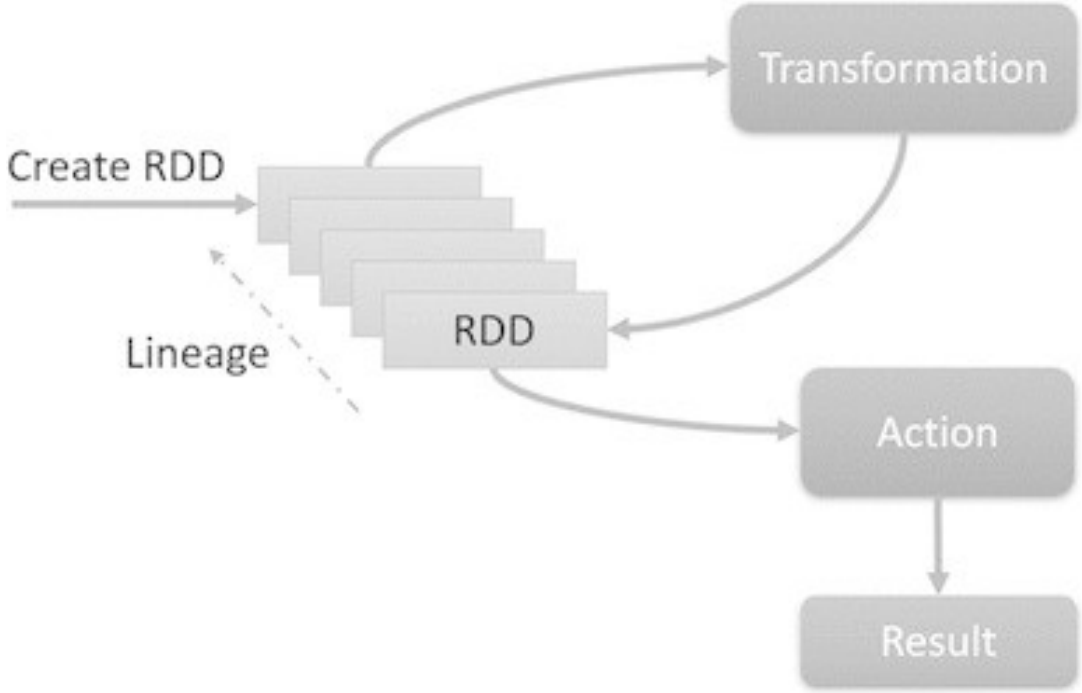
Operações do Apache Spark - Ações

- Ativam o histórico de transformações
- Executam todas as transformações anteriores (EAGER)
- Retornam resultado ou gravam os dados no disco.
- Algumas transformações no Apache Spark
  - Count(), show(),toPandas(), collect(), save()

Apache Spark - Teoria

Apache Spark – Lazy Evaluation

- Operações não são executadas até que sejam necessárias
- Organiza um Histórico de operações
- Otimiza o plano de execução da consulta
- Economiza memória

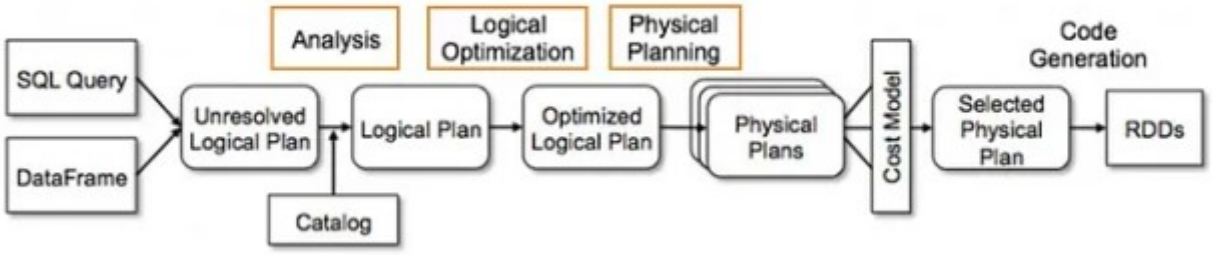


Apache Spark - Teoria

Apache Spark – Catalyst Optmizer

Transforma qualquer operação de API de Dataframes,SQL em plano de execução

- Fases do Catalys Optimizer
  - Analise
  - Planejamento Lógico
  - Planejamento Fisico
  - Geração de código



Apache Spark - Teoria

Apache Spark – Catalyst Optmizer

Planos de execução

- Planejamento Lógico: Nesta fase, o Catalyst recebe a query do usuário e identifica formas de otimizar o processo, principalmente movendo a ordem das operações, ainda abstraindo as transformações a serem aplicadas. O plano otimizado é, então, input do planejamento físico.

- Planejamento Físico: Essa é a etapa em que o plano lógico escolhido é transformado em diversas opções de planos físicos que dizem respeito a como o plano lógico será realmente executado. Então, essas opções são comparadas utilizando um modelo de custo para escolher aquela mais eficiente.

Apache Spark - Teoria

Apache Spark – Analise de plano de execução

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import desc
## Cria a sessão do Spark
spark = SparkSession.builder.getOrCreate()
## Lê os dados de notas e informações básicas dos títulos
df_ratings = spark.read.format('csv').load('title_ratings.tsv', sep='\t', 'header'=True)
df_basics = spark.read.format('csv').load('title_basics.tsv', sep='\t', 'header'=True)

## Seleciona os 15 títulos com a melhor nota no IMDB
df_top15 = (
    df_basics
    .select("tconst", 'primaryTitle')
    .join(df_ratings, ['tconst'])
    .orderBy(desc('averageRating'))
    .select('primaryTitle', 'averageRating')
    .limit(15)
)

## Analisa os planos de execução
df_top15.explain(mode='formatted')
```

Apache Spark - Teoria

Apache Spark – Analise de plano de execução

```
== Physical Plan ==
TakeOrderedAndProject (11)
+- * Project (10)
   +- * SortMergeJoin Inner (9)
      :- * Sort (4)
      :  +- Exchange (3)
      :     +- * Filter (2)
      :        +- Scan csv  (1)
      +- * Sort (8)
         +- Exchange (7)
            +- * Filter (6)
               +- Scan csv  (5)
```

```
(1) Scan csv
Output [2]: [tconst#38, primaryTitle#40]
Batched: false
Location: InMemoryFileIndex [file:/D:/projects/imdb/title_basics.tsv]
PushedFilters: [IsNotNull(tconst)]
ReadSchema: struct<tconst:string,primaryTitle:string>

(2) Filter [codegen id : 1]
Input [2]: [tconst#38, primaryTitle#40]
Condition : isnotnull(tconst#38)

(3) Exchange
Input [2]: [tconst#38, primaryTitle#40]
Arguments: hashpartitioning(tconst#38, 200), ENSURE_REQUIREMENTS, [id=#78]

(4) Sort [codegen id : 2]
Input [2]: [tconst#38, primaryTitle#40]
Arguments: [tconst#38 ASC NULLS FIRST], false, 0

(5) Scan csv
Output [3]: [tconst#16, averageRating#17, numVotes#18]
Batched: false
Location: InMemoryFileIndex [file:/D:/projects/imdb/title_ratings.tsv]
PushedFilters: [IsNotNull(tconst)]
ReadSchema: struct<tconst:string,averageRating:string,numVotes:string>
```

```
(6) Filter [codegen id : 3]
Input [3]: [tconst#16, averageRating#17, numVotes#18]
Condition : isnotnull(tconst#16)

(7) Exchange
Input [3]: [tconst#16, averageRating#17, numVotes#18]
Arguments: hashpartitioning(tconst#16, 200), ENSURE_REQUIREMENTS, [id=#86]

(8) Sort [codegen id : 4]
Input [3]: [tconst#16, averageRating#17, numVotes#18]
Arguments: [tconst#16 ASC NULLS FIRST], false, 0

(9) SortMergeJoin [codegen id : 5]
Left keys [1]: [tconst#38]
Right keys [1]: [tconst#16]
Join condition: None

(10) Project [codegen id : 5]
Output [4]: [tconst#38, primaryTitle#40, averageRating#17, numVotes#18]
Input [5]: [tconst#38, primaryTitle#40, tconst#16, averageRating#17, numVotes#18]

(11) TakeOrderedAndProject
Input [4]: [tconst#38, primaryTitle#40, averageRating#17, numVotes#18]
Arguments: 15, [averageRating#17 DESC NULLS LAST], [tconst#38, primaryTitle#40, averageRating#17, numVotes#18]
```

Apache Spark - Prática

Repositório com codigos

- https://github.com/jader-lima/pyspark_introducao

Apache Spark - Prática

Apache Spark – Spark Session e SparkContext

- Utilizando o ambiente
- Spark Context.
- Spark Session.
- Criando um RDD
- Criando um Dataframe

Apache Spark - Prática

Apache Spark – Leitura e escrita

- Ler dados com Spark
- Tipos de dados
- Criando um Schema
- Ler/Criar dataframe com Schema
- Salvar os dados

Apache Spark - Prática

Apache Spark – Seleção de dados

- Seleção de colunas
- Filtrando linhas
- Ordenando as linhas
- Renomeando colunas
- Criando novas colunas
- Expressões Spark

Apache Spark - Prática

Apache Spark – Tipos de dados e funções

- String, valores numericos, Datas, Arrays, Nulos
- Funções Spark
- Exemplo de operações utilizadas na prática

Apache Spark - Prática

Apache Spark – Agrupamento, junções

- Agregações e Agrupamentos
- Joins
- Union
- Pivot e UnPivot

Apache Spark - Prática

Apache Spark – Windows functions e funções UDF

- Windows Functions
- User Defined Functions (UDF)

Apache Spark - Prática

Apache Spark

- Particionamento
- Reparticionamento

Apache Spark - Prática

Apache Spark – Spark SQL

- Spark SQL
- Criando Tabelas e Views
- Fazendo Queries no Spark SQL
- Databases e Catalog