DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Introdução à Ciência de Dados LNCC - IBGE

Introdução a Big Data e Spark Apache – Aula 2

Prof. Fábio Porto
Abril, 2022.

Laboratório Nacional de Computação Científica
UNIDADE DE PESQUISA DO MCTI

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

APACHE SPARK

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

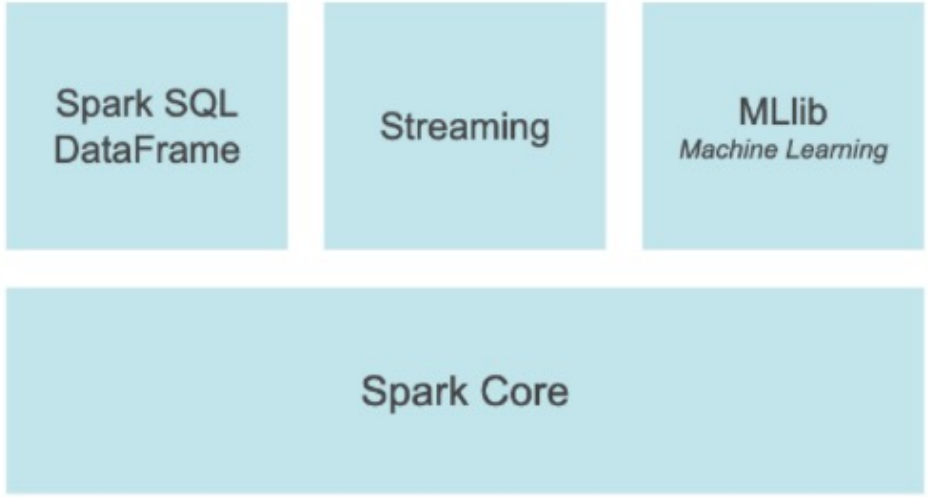
Apache Spark

- Sistema aberto projeto Apache, desenvolvido originariamente em UC Berkeley, AmpLab,
  - Compatível com os dados em diversos formatos: parquet, CSV, Avro JSON e armazenado em sistemas de arquivos, incluindo o Hadoop HDFS
- Extensão do modelo MapReduce para duas classes de aplicações
  - Algoritmos Iterativos (machine Learning, grafos)
  - Mineração de dados (R, Python)
- Melhoria de desempenho (fator 10 a 20x Hadoop MR) /* para casos especiais */
  - processamento in-memory
  - Grafo de tarefas
- Interface de programação
  - APIs para Java, Scala, Python, R

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

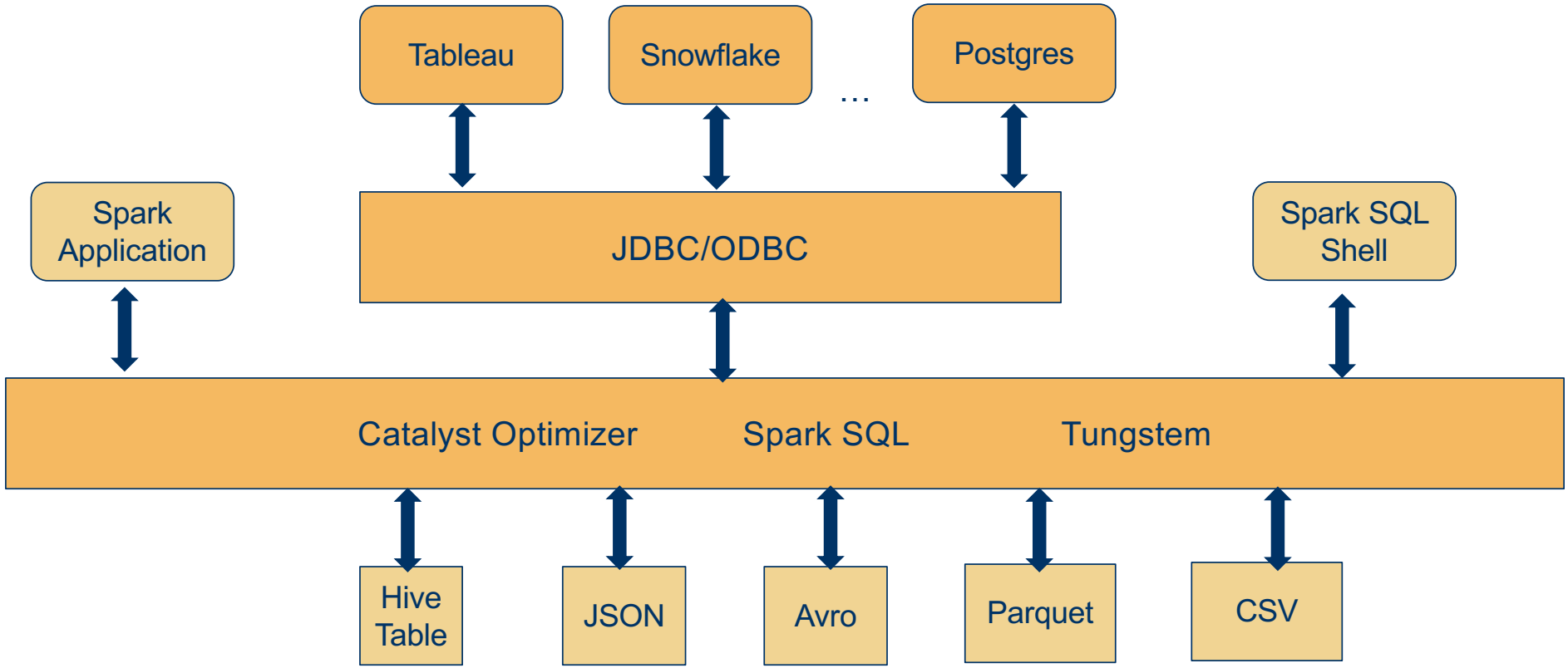
Componentes da Arquitetura Spark



DEXL
DATA EXTREME LAB

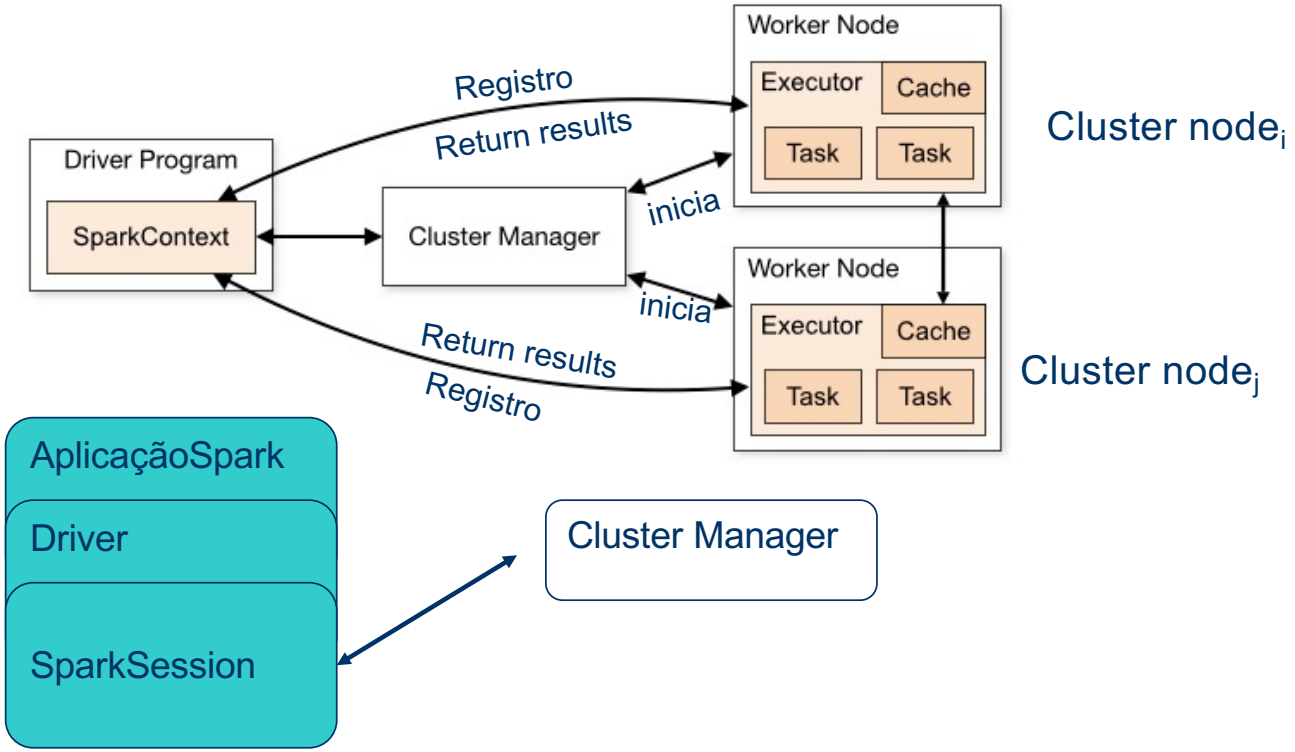
IBGE
Instituto Brasileiro de Geografia e Estatística

Spark SQL



Spark: ambiente de execução

SparkApplication = (MASTER (Driver) -->SLAVES MODEL (Workers))



DEXL
DATA EXTREME LAB

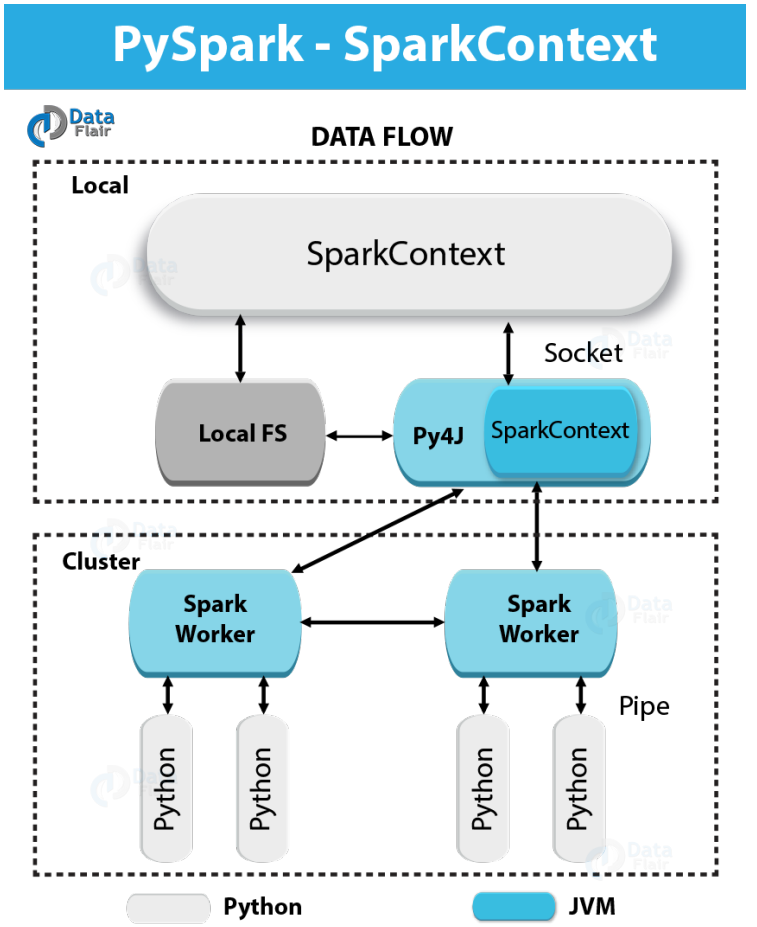
IBGE
Instituto Brasileiro de Geografia e Estatística

Componentes

- Cluster Manager
  - Responsável por gerenciar e alocar os recursos para o cluster de nós de computação aonde as tarefas Spark executarão
  - Aloca executires conforme os recursos configurados: #cores, qtd de memoria, etc..
  - No modo “cluster”, lança o Driver tb.
  - Atualmente suporta os seguintes gerenciadores de cluster:
    - Apache Hadoop YARN
    - Apache Mesos
    - Kubernets
    - Spark Standalone
- Driver
  - Contém o método “main” da aplicação
  - Determina as tarefas a serem executada, a partir da análise do gtafo de execução
    - Cria o plano de execução (com a ajuda do Catalyst)
  - Usa o Cluster Manager para escalonar as tarefas
  - Gerencia os dados “cached” nos executores
- Spark Executores
  - Executa em cada um dos nós do cluster
  - Comunica com o programa driver e é responsável por executar tarefas nos nós de trabalho.

SparkContext

- Primeira estrutura criada para conexão entre aplicações e Spark
- Permite a crição de”
  - RDDs, acumuladores
  - “broadcast variables”
    - Varíavels que são enviadas para os nós trabalhadores
  - Usar o método *parallelize*
- Permite à Aplicação *Spark* acessar o *Spark Cluster* através dos gerentes de recursos: (YARN/Mesos)
- *Criação:*
  - *SparkContext.getOrCreate(sparkConf)*
- Apenas uma variavel (i.e. estrutura) disponível por aplicação



DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

SparkSession

- Criado pelo Spark Driver
- Um conduite unificado para todas as operações e dados em Spark
  - Generaliza acesso a: SparkContext, SQLContext, HiveContext, StreamingContext
    - Todos são mantidos por compatibilidade de código
- Através de um SparkSession pode-se criar:
  - Dataframes, Datasets, enviar consultas SQL etc…

Ex. em Python:

```
from pyspark.sql import SparkSession

spark = (SparkSession.builder \
    .master("local")    \
    .appName("IBGE")  \
    .getOrCreate() )
```

- Reusando uma SparkSession
  - SparkSesssion.getActiveSession()

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

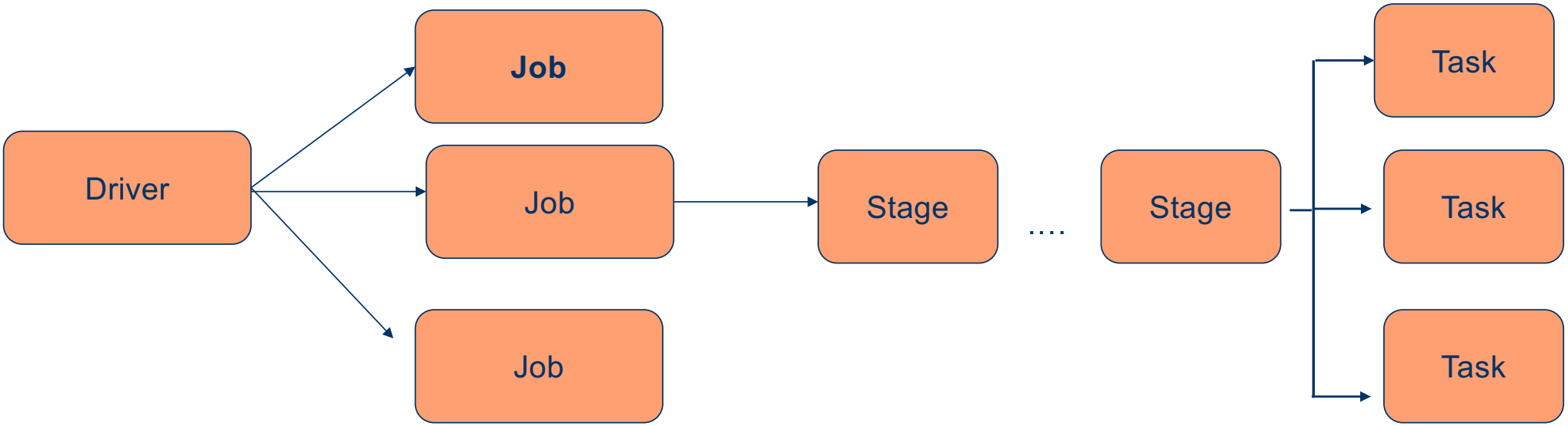
Modelos de disponibilização

Modo	SparkDriver	Spark Executor	Cluster Manager
Local	Executa em uma única JVM, em um notebook ou nó único	Executa na mesma JVM que o Driver	Executa na mesma máquina
Standalone	Pode rodar em qualquer nó do cluster	Cada nó do cluster lança seu próprio JVM do Executor	Pode ser alocado a qualquer nó do cluster
YARN(client)	Roda em um cliente fora do cluster	Container alocado pelo NodeManager	YARN Resource Manager e YARN appication Manager
YARN(cluster)	Roda com o YARN Application Master	O mesmo que o modo cliente	O Mesmo que modo cliente
Kubernets	….	….	…..

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Anatomia da Execução de uma Aplicação Spark

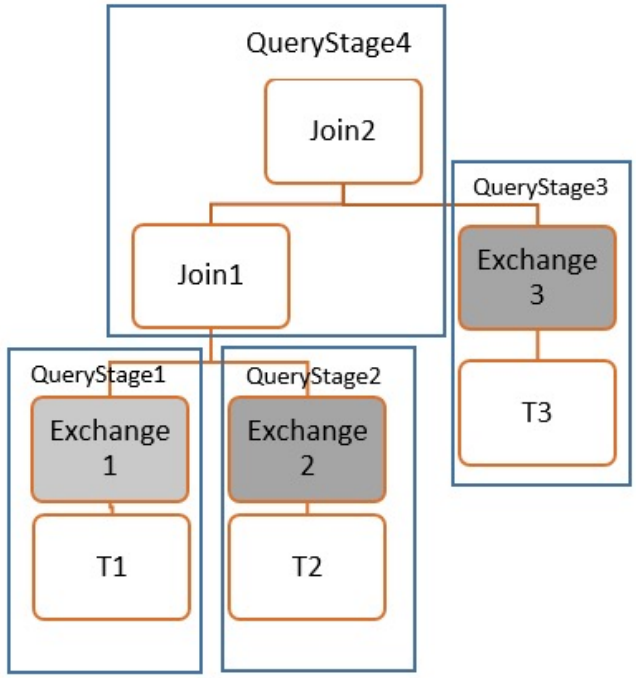


DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Stages in Spark SQL Job

Estágios definidos
Em pontos em que
“shuffle” é necessário.

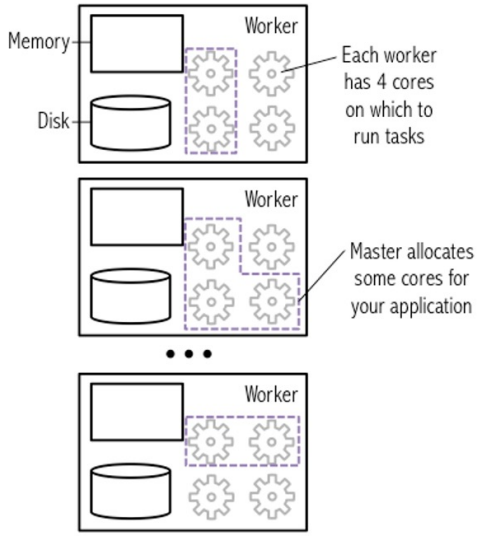


DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Spark Alocação de aplicação

- Um job Spark é iniciado pelo nó “driver”
- Driver distribui tarefas pelo ”workers”
- Os resultados das tarefas dos workers são enviados de volta ao “Driver”



- Pode verificar o ambiente de execução na WebUI:
- Executors -> JVM

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Resilient Distributed Datasets(RDDs)

Coleções de dados em memória

Construídos a partir de transformações paralelas (map, filter, etc..) ou a partir de leitura de arquivos.

São imutáveis

Podem residir em memória para re-utilização eficiente em pipelines

Preserva e garante as propriedade MapReduce

- Tolerância à falhas, localidade de dados, escalabilidade

Cada RDD possui sua informações de proveniência para reconstrução (lineage)

O usuário pode determinar como armazenar os dados

- Disco ou memória RAM
- Cache de dados em memória

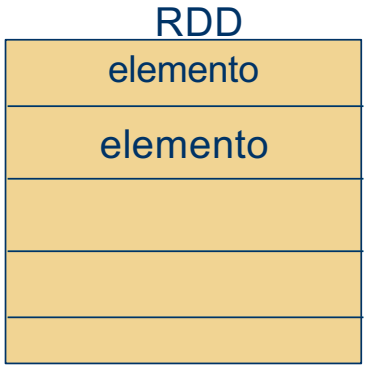
Operações: transformações e ações

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

RDD

- RDD é uma interface para referência à coleção de dados
- Todos os itens de um RDD são de um tipo:
  - int, String, [long], [int, int, float],…
- A coleção representada por um RDD é distribuída pelos nós do cluster
  - conjunto de inteiros { 10, 20,.., 30, 40,....,99,45, ..., 2}:
  - no1: {10, 20,...,30}
  - no2: {40,...,99}
  - no3: {45,…,2}



Cada elemento é acessado como um item de tipo Python, Simples ou complexo, ou um par [Key,Value]

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Tipos de dados Simples Python em Spark

Data Types	Valor associado em Python	API
ByteType	int	DataTypes.ByteType
ShortType	int	DataTypes.ShortType
IntegerType	int	DataTypes.IntegerType
LongType	int	DataTypes.LongType
FloatType	float	DataTypes.FloatType
DoubleType	float	DataTypes.DoubleType
String	str	DataTypes. StringType
BooleanType	bool	DataTypes. BooleanType
DecimalType	decimal.Decimal	DecimalType

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

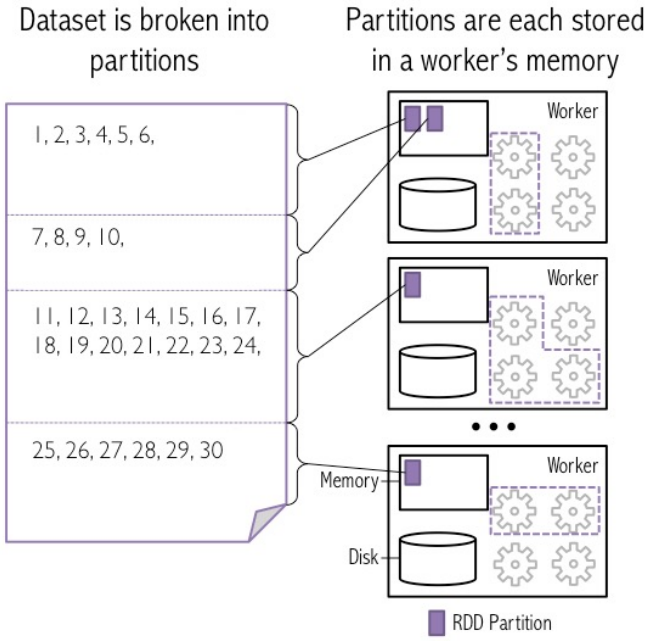
Tipos de dados estruturados em Spark e seus correspondentes em Python

Data Types	Valor associado em Python	API
BinaryType	bytearray	BinaryType()
TimestampType	datetime.datetime	TimestampTime()
DateType	datetime.date	DateType()
ArrayType	list, tuple, or array	ArrayType(dataType, [nullable])
MapType	dict	MapType(keyType, valueType,[nulable])
StructType	list or tuple	StructType([fields])
StructField	um valor de tipo simples	StructField(name, datatype,[nullable])

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Dataset distribuído em partições

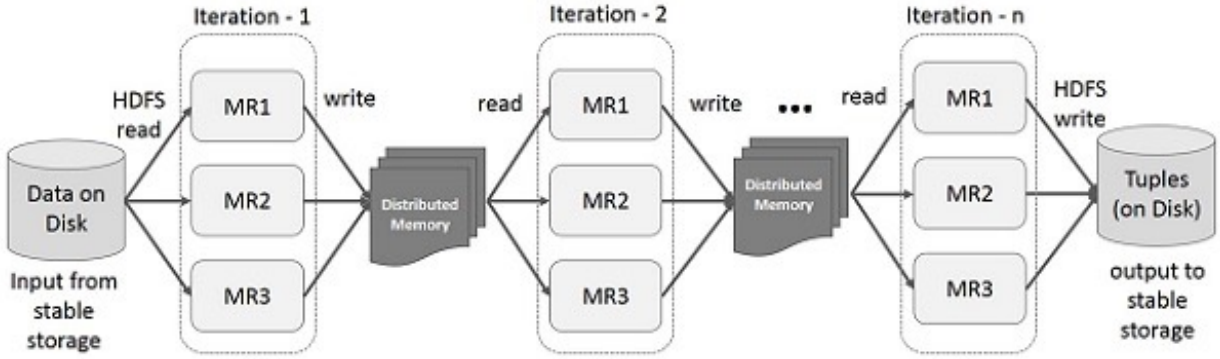


DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Pipeline em memória

- RDD criado em memória repetindo o particionamento em disco
- Cada partição em disco gera, com prioridade, uma partição em memoria do mesmo nó;
- Transformações geram outras novos RDD com o resultado da operação
- Ao final do pipeline, garante-se a persistência do RDD com garavação em disco



DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

RDD - Propriedades

- "Vive" em um SparkContext (sc)
  - não pode ser compartilhados entre sc
- Pode ser "fixado" em cache
  - em disco ou em memória
- Exemplo:
  - text_rdd = sc.textFile("input_file_path")
  - rddQT = text_rdd.mappartition(quadtree)
  - rddQT.cache()
  - rddQT.count()
- Quando criar RDDs
  - Dados não estruturados, como texto, por exemplo
  - Preferencia por definir um processo eficiente, independente do otimizador "Catalyst"
- Pode-se transformar RDD and Dataframe
  - df.rdd

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

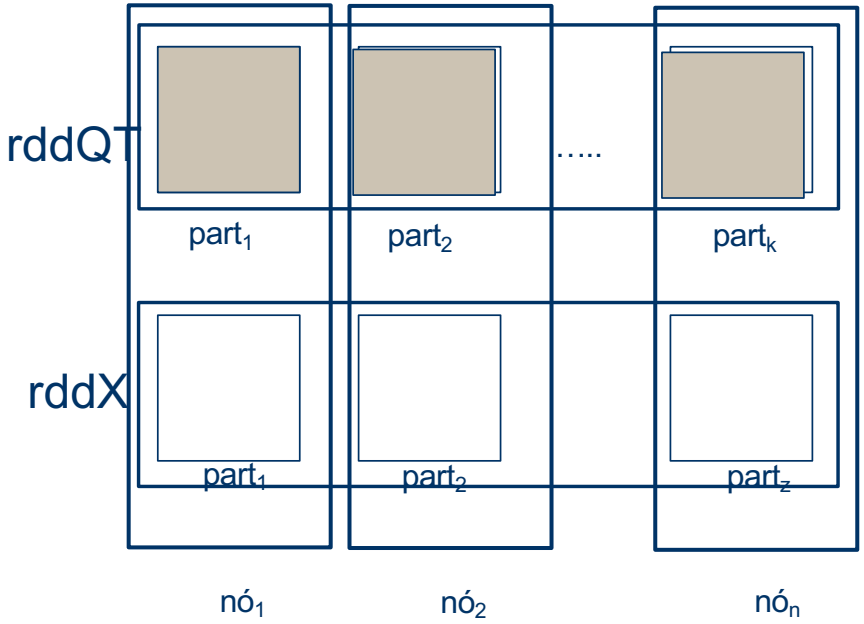
Cache ou persist

- Aloca as partições de RDDs ou Dataframes no “storage level” especificado
  - Para *storage level=MEMORY_ONLY, MEMORY_ONLY_SER e MEMORY_AND_DISK,* apenas as partições que couberem em memória são mantidas
  - Spark vai priorizar a manutenção em memória
  - Se uma partição não estiver em memória, ao ser referenciada, o processo de sua geração é recomputado
- Quando usar
  - RDDs ou DataFrames usados iterativamente em treinamento de modelos
  - RDDs ou DatFrames referenciados várias vezes na aplicação
- Quando não usar
  - RDDs ou Dataframes ocupando muita memória
  - RDDs ou DataFrames cujo processo de geração seja muito barato

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Cache de RDDs – Memória Distribuída



rddQT.cache()

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Processamento RDD

- Uma função Spark itera implicitamente sobre os itens de um RDD
- Cada invocação da função ocorre em um estado próprio não mantendo estado entre as invocações

```
transformRDD = listTrip.map(lambda a: (a,1))
```

for each trip in listTrip:
    return (trip,1)

trip –>map-> (trip,1)

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

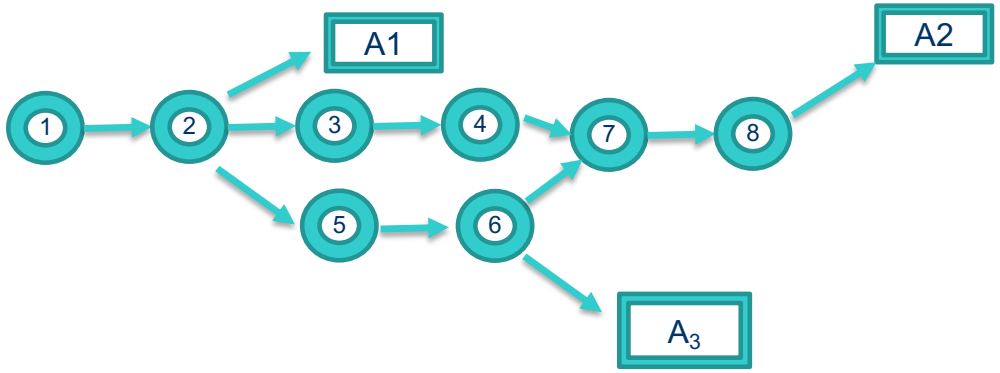
Modelo de Execução: Lazy

- Um Dataflow Spark é um grafo composto de ***transformações*** e ***ações***
- ***Transformações*** são executadas apenas quando uma ***ação*** que precisa de seu resultado é acionada
- O analisador Driver)de dataflows varre o grafo de dependências definido na função *main* e empilha transformações até que uma ação seja encontrada

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Exemplo Ilustrativo



Transformações

1,2,3,4,5,6,7,8,9,10

Ações

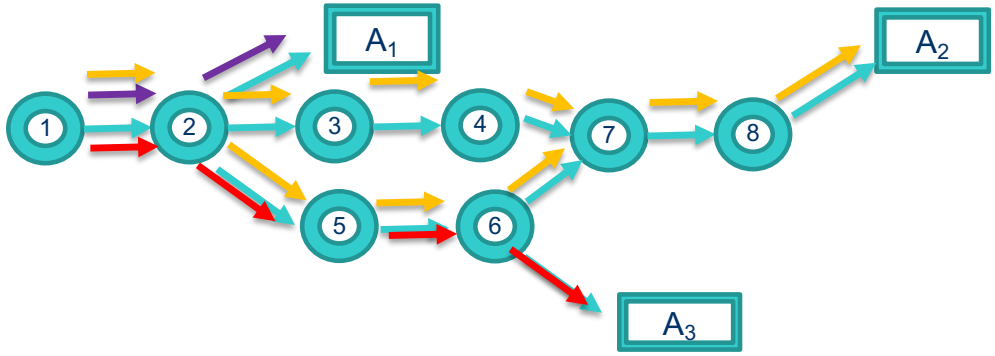
A1, A2, A3

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Possível escalonamento

- <1,2,A1,1,2,3,4,5,6,7,8,A2,1,2,5,6,A3>



DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

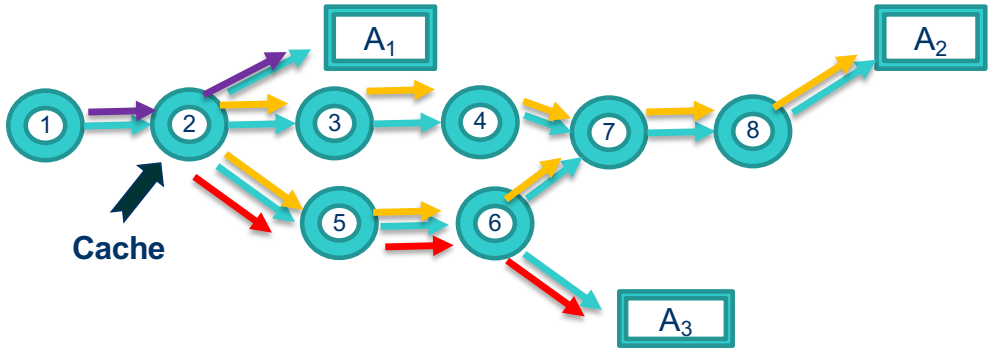
Modelo de Execução: Tardio (*Lazy*)

- Observe que o modelo de execução é influenciado pela escolha em se manter arquivos intermediários em memória
- Fica aparente o custo de se re-executar os trechos do dataflow para reconstrução do RDD necessário para a ação
- O método *cache* pode mitigar esse problema

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

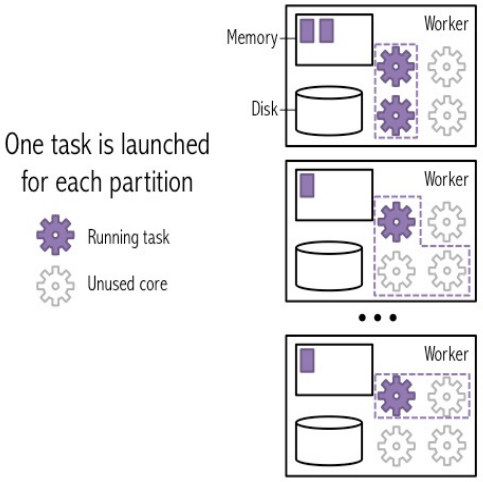
Execução com Cache

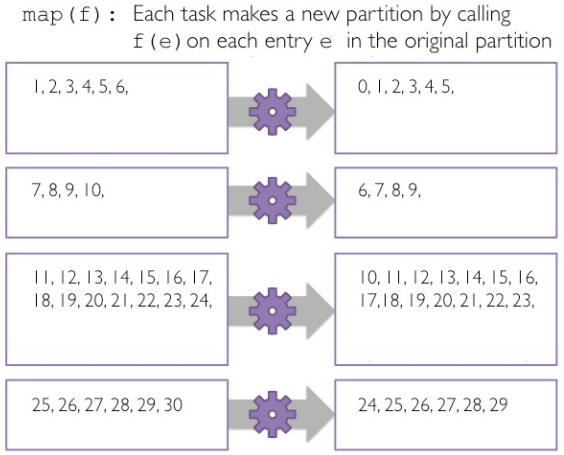


DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Execução Spark x Partições

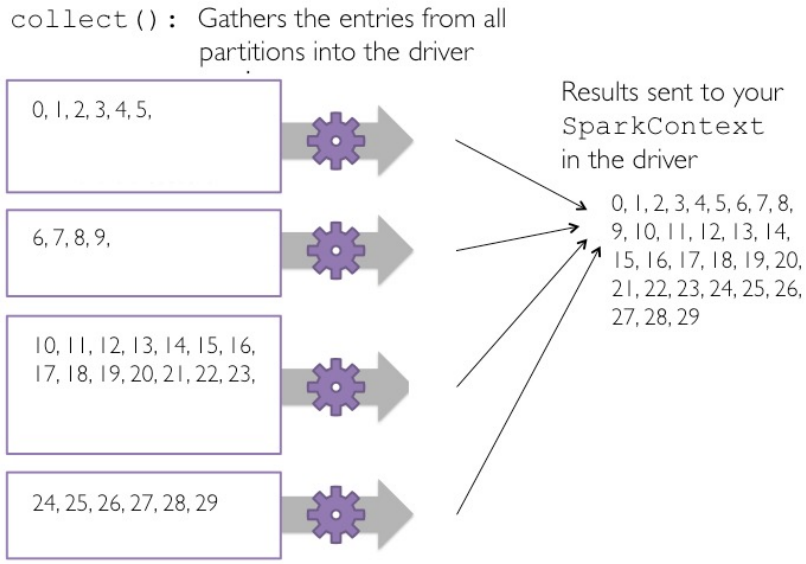




DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Coletando o resultado final



DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Exemplo 1 (Criando RDDs e DFs)

Vamos Praticar!

Exemplo 2 (Conta M&M)

Vamos Praticar!

Exemplo 3 (Criando Dataframe com Esquema)

Vamos Praticar!

FIM da parte 2.1!

Perguntas

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

SPARK API

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Leitura dos Dados

- Já vimos vários exemplos de transformação de arquivos de entrada em RDDs
  - parallelize
  - textFile
  - newAPIHadoopFile

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Operação do Contexto Spark:Parallelize

- Processa coleções no programa “driver”
- Os dados de coleções são enviados aos executores para criação das partições dos RDDs
- O número de partições pode ser definido como parâmetro da função;

Parâmetros:

- Entrada: Uma coleção de dados, por exemplo: listas, tuplas
- Saída: Um *RDD* com os dados representando cada elemento da coleção

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Exemplo de aplicação com o paralelize (collection[,#Partitions]) :

- Cria um RDD com 1000 valores pseudo-aleatórios entre 1 e 5000
- Dados são distribuidos por duas partições

```
pointsRDD=sc.parallelize(random.sample(range(1,5000),1000),2)
```

- RDD criado a partir de uma lista

```
particionadoRDD = sc.parallelize([1,4,6,45,98],2)
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

textFile (path [, #Partitions])

- Pode-se ler: um arquivo, uma lista de arquivos em um diretório, todos os arquivos em um diretóri
  na criação de um RDD;
- Cada linha do arquivo é lida como uma “Row” do tipo “String” no RDD

- spark.textFile()
  - Lê de HDFS, S3 ou qq Sistema de arquivo suportado pelo Hadoop.
  - Parâmetros: path, [#partições]
  - Retorna um RDD

Ex:

```
from pyspark.context import SparkContext
spark = SparkContext(master=“local[*]”)
taxi_file_header =
spark.textFile("/content/gdrive/MyDrive/BigData/data/yellow_tripdata_2021-01.csv")
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

newAPIHadoopFile

API de leitura e gravação de Hadoop files no formato [Key,value]

Parâmetros:

- Entrada:
  - Lista de caminhos para os arquivos a serem lidos
  - Classe indicando o formato de armazenamento do arquivo sendo lido
  - Classe do parâmetro *key*
  - Classe do parâmetro *value*
- Saída: Um *RDD* com a *key* e correspondente valores

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Exemplo de aplicação com o *newAPIHadoopFile*:

```
taxi_sem_headerRDD=taxi_file_header.map(lambda a:a.split(",")).filter(lambda a: a[0] != "VendorID" and a[0]!="")

taxi_filteredRDD = taxi_sem_headerRDD.filter(lambda taxi: True if int(taxi[0])==1 else False)

taxi_filteredRDD.saveAsNewAPIHadoopFile("/content/gdrive/MyDrive/BigData/data/2018_trip_treated.csv", outputFormatClass="org.apache.hadoop.mapreduce.lib.output.TextOutputFormat")
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Transformações

- Produzem um RDD output a partir de um RDD input
- map (f)
  - Produz um RDD aplicando a função ***f*** a cada elemento do RDD fonte
- MapPartition(f) – uma partição é consumida de forma integral
- flatMap (f)
  - Produz um RDD em que uma chave de entrada pode gerar *0* ou *n* na saída.
- union (RDD)
  - Produz um RDD que contém a união dos RDDs de entrada com o informado como parâmetro

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Transformação: map

- *map (f): rdd_in.map(f)*
  - Aplica *f a* cada entrada do *rdd_in*;
- Resulta em *RDD* com uma entrada para cada resultado de f(rdd_in);

Parâmetros:

- Entrada: Uma função *f (definida ou lambda)* com parâmetros correspondentes aos tipos dos dados nos elementos de *rdd_in*
- Saída: Um novo *RDD* com elementos do tipo da saída da função *f*.

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Exemplo de aplicação com o *map*:

```
def find_k_nearest_neighbors(k,p, points):
    distances = [abs(p-point) for point in points]
    neighbors = distances[0:k]
    return [p,neighbors]

number_rdd = sc.parallelize(range(1,1000))
points = random.sample(range(1,5000),1000)

neighborsRDD = number_rdd.map (lambda p: find_k_nearest_neighbors(3,p, points))
neighborsRDD.take(10)
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Transformação: flatMap

- Produz entre [0,n] elementos no *RDD de saída* a partir de um elemento do *RDD* de entrada.

Parâmetros:

- Entrada: Um *RDD* de entrada; uma função de transformação 1->n
- Saída: Um novo *RDD* de saída; cada elemento de saída da função é um elemento do RDD

```
readTextDirectory=sc.textFile(“word-count.txt",2)

rdd_map =rddTextDirectory.map(lambda a: a,split(“ “))

rdd_map.count()
38

rdd_flat=readTextDirectory.flatMap(lambda a: a.split(" "))
rdd_flat.count()
604
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Transformação *flatMapValues*:

```
x = sc.parallelize( [(“a”,[“x”,”y”,”z”]), (“b”, [“p”,”r”])])

def  mirror(x): return x

x.flatMapValues(lmirror).collect ()
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Transformação: mapPartitions

- A operação mapPartitions recebe uma partição de RDD de entrada e produz um RDD de saida. Torna disponível para a função de primeira ordem todos os elementos da partição através de um iterador.

Parâmetros:

- Entrada: função que itera sobre o conjunto de elementos de cada partição.
  - O parâmetro da função é o iterador
- Saída: Um novo *RDD*

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

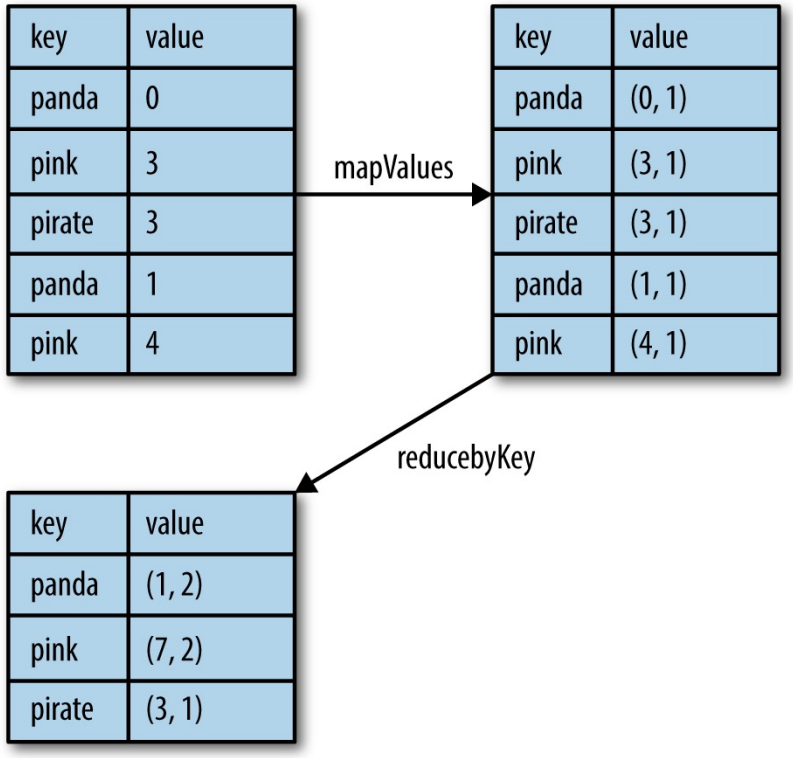
Exemplo : mapPartition

```
def processPartition(part):
        for element in part:
        yield element
sampleRDD = spark.parallelize([1,2,3,4,5,6,7],2)
resultRDD =
sampleRDD.mapPartitions(processPartition)
resultRDD.take(7)
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

PairRDD



DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

PairRDD

- Um tipo variante de RDD construido para representar pares (key, value);
- Funções aplicadas em pairRDD receben uma tuple de chave, valor.

data = [(**'Project'**, 1), (**'Gutenberg's'**, 1), (**'Alice's'**, 1),(**'Adventures'**, 1),(**'in'**, 1),
(**'Wonderland'**, 1),(**'Project'**, 1),(**'Gutenberg's'**, 1),(**'Adventures'**, 1),(**'in'**, 1),
(**'Wonderland'**, 1),(**'Project'**, 1),(**'Gutenberg's'**, 1)]

```
pairrdd=sc.parallelize(data)
pairrdd.coalesce(1).saveAsTextFile("path")
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Transformação: reduceByKey

Também é considerada uma transformação similar a função *reduce*, porém agrega os itens do RDD com base no valor de um atributo considerado chave. O RDD deve estar estruturados em um par (chave,valor), pair RDD.

Parâmetros:

- Entrada: Uma função com dois parâmetros, o primeiro corresponde à chave e o segundo são todos os valores do *RDD*
- Saída: RDD com <chave,valor-agregado>.

DEXL
DATA EXTREME LAB

Exemplo Transformação *reduceByKey:*

IBGE
Instituto Brasileiro de Geografia e Estatística

```
data = [('Project', 1), ('Gutenberg's', 1), ('Alice's',
   1),('Adventures', 1),('in', 1),
   ('Wonderland', 1),('Project', 1),('Gutenberg's',
   1),('Adventures', 1),('in', 1),
   ('Wonderland', 1),('Project', 1),('Gutenberg's', 1)]

   rdd=sc.parallelize(data)
   x=rdd.reduceByKey(lambda x,y:x+y)
   x.collect()

>>[('Project', 3), ('Gutenberg's', 3), ('Alice's', 1), ('Adventures', 2),
   ('in', 2), ('Wonderland', 2)]
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Outras transformações key/value

- groupbykey()
- combinebykey()
- mapvalues(func)
- flatMapValues(func)
- keys()
- values()
- sortbykey()

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

AÇÕES

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Ações

- Disparam todas as transformações necessárias para produzir o RDD de sua entrada.
- Produz um valor (não gera RDDs) e retorna para o programa driver

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Ação: *reduce*

Agrega todos os elementos de um RDD por uma função de agregação **comutativa** e **associativa**. Inicia um único processo.

Parâmetros:

- Entrada: Uma função com dois parâmetros, o primeiro mantém o estado da redução. O segundo assume os valores contidos no *RDD*
- Saída: O resultado da agregação.

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Exemplo de aplicação com a função *reduce*:

```
data_rdd = sc.parallelize([1,2,3,4,5,6,7,8,9,10], 2)
cSum = data_rdd.reduce(lambda accum, n: accum + n)
print(cSum)
>> 55
cMin = data_rdd.reduce(lambda a, b: min(a,b))
print(cMin)
>>1
cMax = data_rdd.reduce(lambda a, b: max(a,b))
print(cMax)
>>10
```

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Ação: count()

Retorna o número de elementos em um RDD.

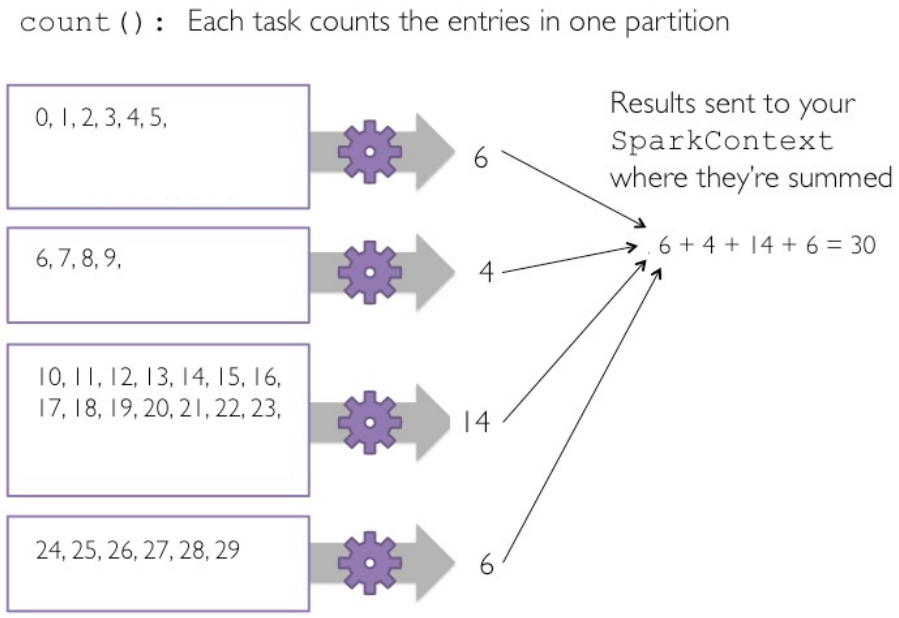
Parâmetros:

1. Entrada: RDD
2. Saída: Um inteiro representado o total de objetos no RDD de entrada

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Operação Reduce distribuida



DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Outras ações

- first()
- take(n) – *n* primeiros
- top(m) – *m* maiores valores
- takeSample(withReplacement=true,num=k) – *k* amostras do RDD

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

HDFS – SISTEMA DE ARQUIVOS DISTRIBUÍDO

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Armazenamento de Dados

- O processamento de dados Big Data privilegia localidade de dados
  - Funções são enviadas aos locais onde se encontram os dados
  - Arquitetura sem compartilhamento
    - máquinas com disco, processador e memória
- Arcabouços usam sistemas de arquivos
  - dados são particionados pelos discos do sistema

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

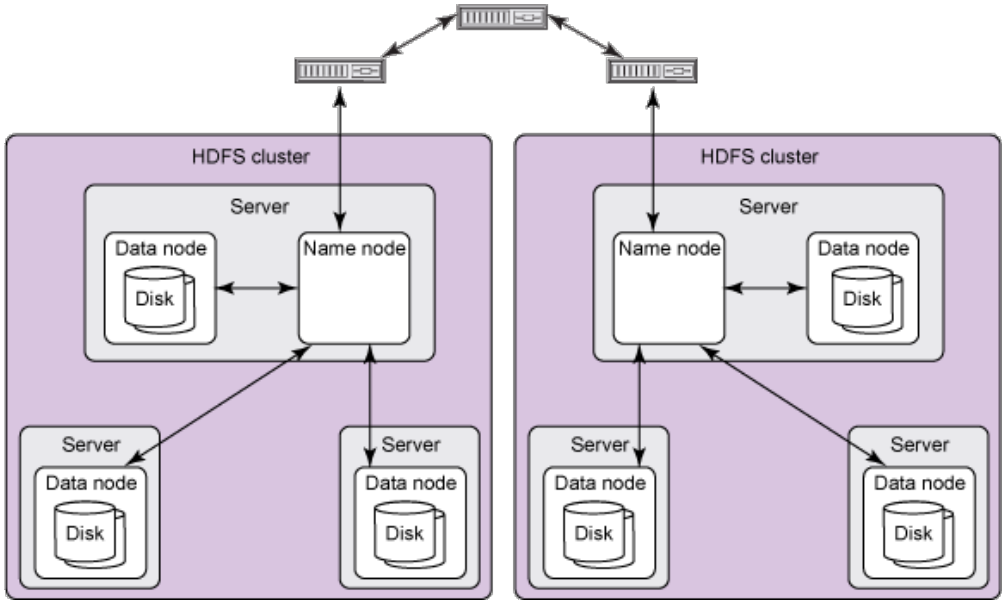
HDFS – Hadoop File System

- Baseado no GFS da Google
- Dados são replicados através do cluster
  - defaut 3 vezes;
- suporta datasets até PBs
- gravação no final do arquivo
- arquivos (grava-uma-vez), majoritariamente leitura
- Leitura sequencial
- Suporta clusters não confiáveis

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Arquitetura



DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Características

- arquivos são armazenados em blocos (default 128MB) maiores que a maioria dos FSs
- Cada bloco replicado em, pelo menos, 3 nós de dados;
- Um ponto de falha (NameNode), Mestre único coordena todos os nós de dados
- Não realiza caching de dados

DEXL
DATA EXTREME LAB

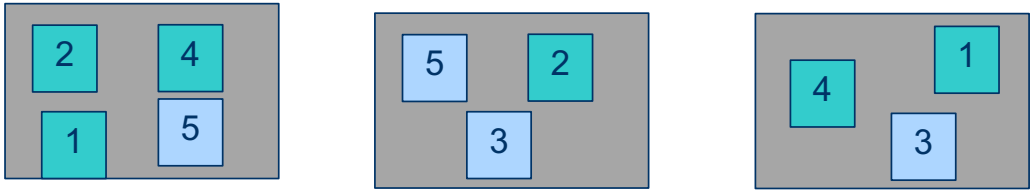
IBGE
Instituto Brasileiro de Geografia e Estatística

NameNode e dataNode

NameNode:
apenas metadados

Metadados:
/user/aaron/foo -> 1,2,4
/user/aaron/bar -> 3,5

DataNodes: armazena blocos de arquivos



DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Metadados

- Um nó armazena todas as informações de metadados:
  - nomes de arquivos, localizações dos blocos nos DataNodes
- Mantido inteiramente na memória
  - bastante memória na máquina abrigando o NameNode
- Blocos mantidos no sistema de arquivos local

DEXL
DATA EXTREME LAB

IBGE
Instituto Brasileiro de Geografia e Estatística

Interface de Manipulação de arquivos

- hadoop [comando] [opções-genéricas] [opções-comando]
  - hadoop fs –ls /home/curso/file1
  - hadoop fs –mkdir / home/curso /dir1
  - hadoop fs –cp / home/curso /file1 / home/curso /file2
  - hadoop fs –copyFromLocal –f] *path_local* /data/proj/file1

Exemplo *5* (HDFS)

Vamos Praticar!

FIM da parte 2.2.1

Perguntas