

Jornada em Ciência de Dados 2021

2ª Parte

Algoritmos e Modelo de Programação em Big Data

Fabio Porto (fporto@lncc.br)
LNCC – DEXL Lab
<http://dexl.lncc.br>



Laboratório
Nacional de
Computação
Científica

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA,
INOVAÇÕES E COMUNICAÇÕES



1

Agenda – 1a Parte: Arcabouço MR

Fábio Porto




- Modelo Computacional Big Data
- Modelo Map-Reduce
- Hadoop – Map Reduce
- SPARK

AMPBD

DEXL
DATA EXTREME LAB

2

Fábio Porto



O MODELO COMPUTACIONAL BIG DATA


AMPBD

DEXL
DATA EXTREME LAB

3

Processamento de Grandes Volumes de Dados

Fábio Porto



- Processar grandes volumes de dados para tomada de decisão requer eficiência
 - Reduzir o tempo de processamento
 - Paralelismo de tarefas aparece como uma estratégia intuitiva
 - Dados podem ou não estar distribuídos
- Processos sequencias precisam ser modelados de forma paralela segundo o modelo de paralelismo a ser adotado

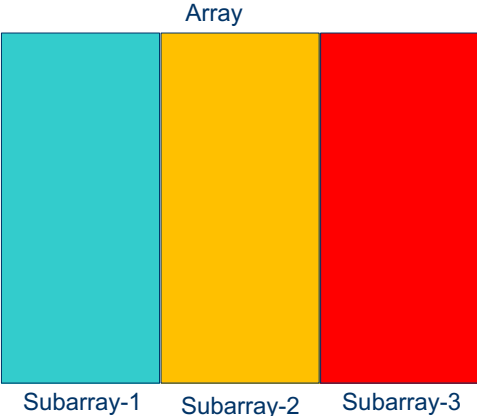
AMPBD

DEXL
DATA EXTREME LAB

4

Modelos baseados no particionamento de dados

Fábio Porto



Array

Subarray-1 Subarray-2 Subarray-3

AMPBD

DEXL
DATA EXTREME LAB

5

Localidade de Dados

Fábio Porto

- No processamento de grandes volumes de dados o custo de transferência pela rede deve ser minimizado
 - Chamamos de *Localidade de dados* ao processo de escalonamento de processos sobre os dados que favorece a execução nos locais em que os dados se encontram, i.e. minimize a movimentação

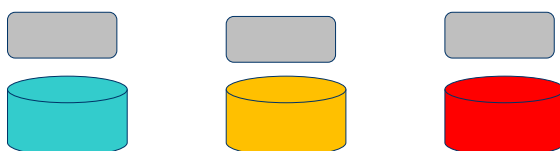
AMPBD

DEXL
DATA EXTREME LAB

6

Uma Tarefa vários fragmentos de dados

Fábio Porto

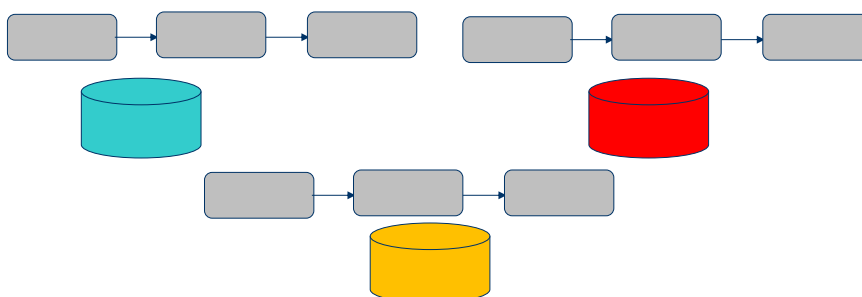
AMPBD

DEXL
DATA EXTREME LAB

7

Um fluxo de execução(Pipeline) vários fragmentos de dados

Fábio Porto

AMPBD

DEXL
DATA EXTREME LAB

8

Como tirar proveito de dados particionados?

Fábio Porto



- Modelo de paralelismo define restrições para modelagem de problemas
 - Identificação de fragmentos do pipeline que podem ser executados independentemente;
 - Identificação de ponto de convergência global de dados
- Processamento dividido em
 - Fragmentos locais
 - Fragmentos globais

AMPBD

DEXL
DATA EXTREME LAB

9

Limites de Programas M/R

Fábio Porto



- Algoritmos que requeiram a manutenção de um estado global atualizado a todo tempo
 - Pode ser mapeado para uma série de M/Rs
- Critério de particionamento de dados deve considerar necessidade de localização de dados pela aplicação
 - Estabelecimento de correlação de vizinhança
- Em essência, M/R deve limitar a troca de mensagens entre os nós de computação

AMPBD

DEXL
DATA EXTREME LAB

10

Linguagem de programação p frameworks Big Data

Fábio Porto



- Problemas implementados na linguagem do framework
 - K-means
 - Regressão Linear
 - Operadores Relacionais: ex:Junção, agregação,...
- Descritos como um fluxo de dados
 - Na sua forma mais simples: Maps-Reduces
- Critério de particionamento influencia no algoritmo
 - Pontos de sincronismo entre os nós de processamento

AMPBD

DEXL
 DATA EXTREME LAB

11

Decomposição k-means

Fábio Porto



Map [key_i,[centroid]-> [key_i,centroid_j]




New_centroid_j=Reduce[centroid_j,[k_i,k_i+1,...,k_n]] -> avg[k_i,k_i+1,...,k_n]

AMPBD

DEXL
 DATA EXTREME LAB

12

Fábio Porto



Decomposição k-means

While not converge centroid_i <-> new_centroid_i
Map [key_i,[centroid]-> [key_i,centroid_j]

↓


new_centroid_j=Reduce[centroid_j,[k_i,k_i+1,...,k_n]] -> avg[k_i,k_i+1,...,k_n]

AMPBD

DEXL
DATA EXTREME LAB

13

Fábio Porto



DATAFLOWS

AMPBD

DEXL
DATA EXTREME LAB

14

Modelos Computacionais para processamento de grandes volumes de dados

Fábio Porto



- Workflows científicos – processamento de dados realizado como parte de um experimento científico
 - Nesta apresentação estaremos considerando processos automáticos, i.e., sem intervenção do usuário;
 - Topologia de um grafo direcionado, com ou sem ciclos, $G=(N,E)$, onde N representa um conjunto de atividades e E a dependência entre as atividades espelhada na relação produtor/consumidor de dados;
 - Programas desenvolvidos de forma independentes.
 - Processamento de arquivos, sem interface comum
- Dataflows – generalização de workflows científicos para outros domínios de aplicação
 - Uso de linguagem (API) comum
 - Compartilhamento de um Modelo de Dados
 - Pode incluir um pouco mais de conhecimento dos programas e dados, se projetado com componentes internos à instituição;
- Consultas a BDs – processamento dados em função de uma requisição externa (consulta ou atualização);
 - Processamento inteiramente automático
 - Expresso em linguagem de alto nível (SQL – ou variante)
 - Topologia – árvore – profunda ou equilibrada
 - Conhecimento dos dados (modelo e estatísticas) e da semântica das operações (álgebra)
- Event Processing Systems
 - Modelo de Processamento baseado em eventos:
 - Reação a dados de sensores (IoT)
 - streaming
 - processos de negócio
- Frameworks Distribuídos
 - Intensivo de CPU
 - Derivados de implementações em Ciência de Dados
 - Com crescimento do volume de dados analisados evoluíram para framework complexos

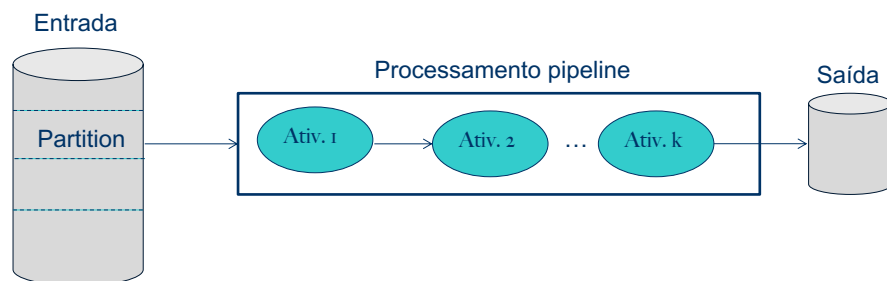
AMPBD

DEXL
 DATA EXTREME LAB

15

Modelo de Base – Dataflows Big Data

Fábio Porto



AMPBD

DEXL
 DATA EXTREME LAB

16

Fábio Porto



Modelo Abstrato

- Entrada: conjunto de itens a serem processados
- Atividades: ações de processamento de dados: programa; função etc
- Ordem Parcial: Atividades são ordenadas segundo uma relação produtor-consumidor
- Saída: conjunto resultado

AMPBD

DEXL
 DATA EXTREME LAB

17

Fábio Porto

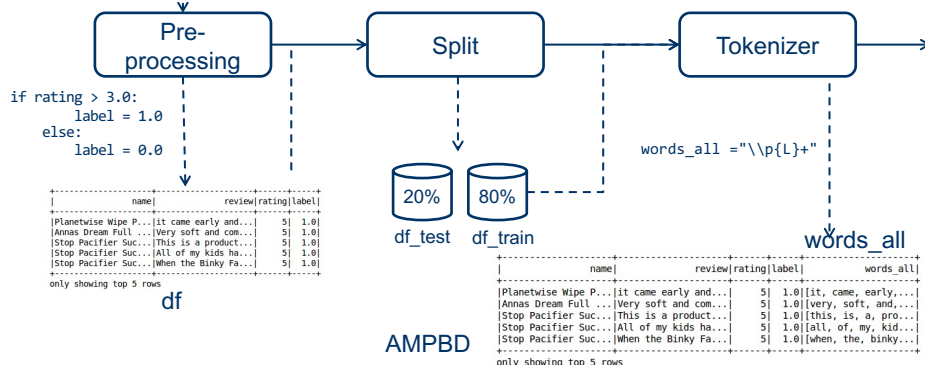


Revisão de filmes - Amazon

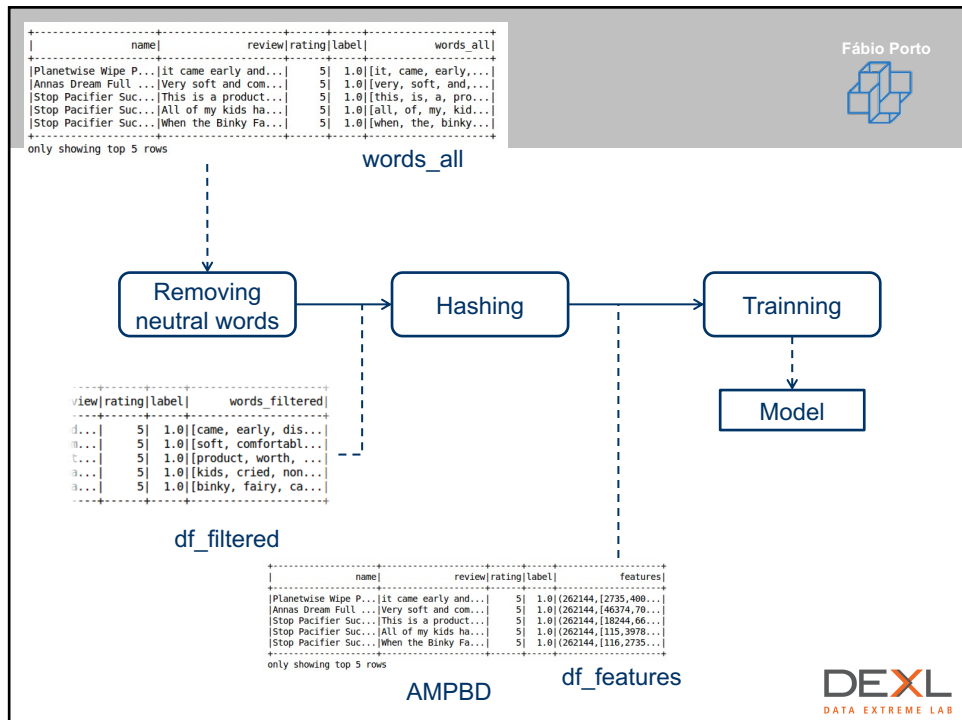
- Workflow: Ranqueamento de Filmes;
- Entrada: Conjunto de Filmes e revisões
- Saída: Filme e ranking

name	review	rating
Planetwise Flanne...	These flannel wip...	3
Planetwise Wipe P...	It came early and...	5
Annas Dream Full ...	Very soft and com...	5
Stop Pacifier Suc...	This is a product...	5
Stop Pacifier Suc...	All of my kids ha...	5

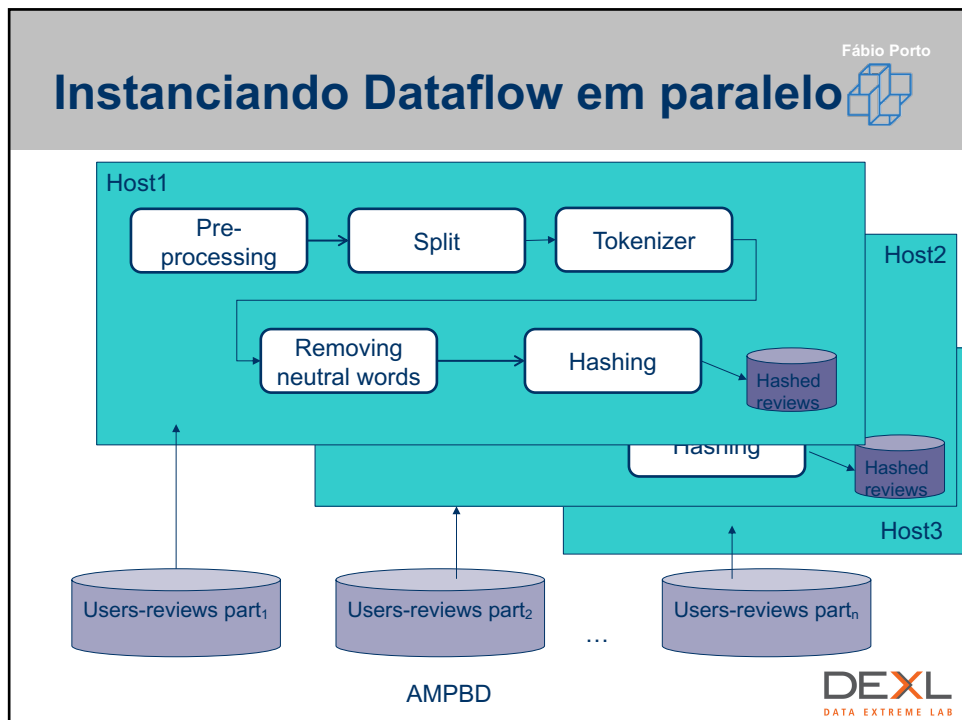
only showing top 5 rows



20



21



22

Fábio Porto

Notebook:yellow_tripdata_2021-01.csv

AMPBD

DEXL
DATA EXTREME LAB

23

Fábio Porto


MODELO MAP/REDUCE

AMPBD

DEXL
DATA EXTREME LAB

24


Fábio Porto



Princípios

- MapReduce é:
 - Um modelo de programação
 - Um ambiente de execução para aplicações desenvolvidas sob o modelo
- Map Reduce considera:
 - Uma arquitetura de clusters de máquinas sem compartilhamento
 - Um sistema distribuído executando o modelo (MR) sobre a arquitetura de clusters
 - Um mecanismo de tolerância à falhas
 - Um sistema de arquivo distribuído com particionamento de arquivos


AMPBD



DATA EXTREME LAB

25


Fábio Porto



Quanto à linguagem: Problema

- Como integrar novas funcionalidades de um domínio à linguagens de programação?
 - DSL – Domain Specific language
 - Integração à linguagens de proposito geral
 - Externas
 - SQL – sintaxe e semântica próprias
 - Embedded
 - DSL integrada `a linguagem de propósito geral
 - Spark, Flink

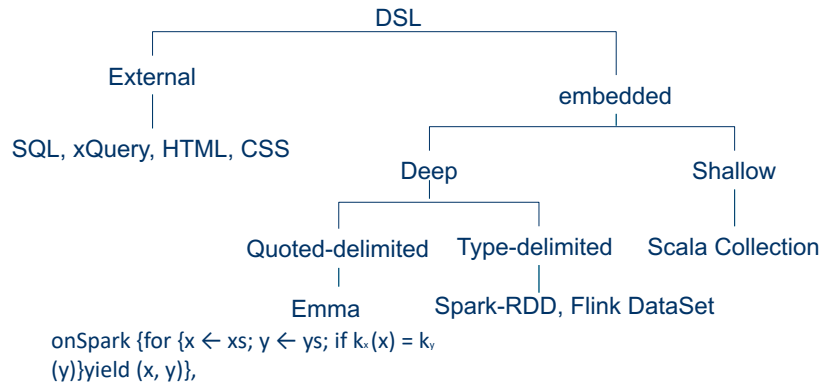
AMPBD



DATA EXTREME LAB

26

Introdução de Linguagens Específicas de Domínio



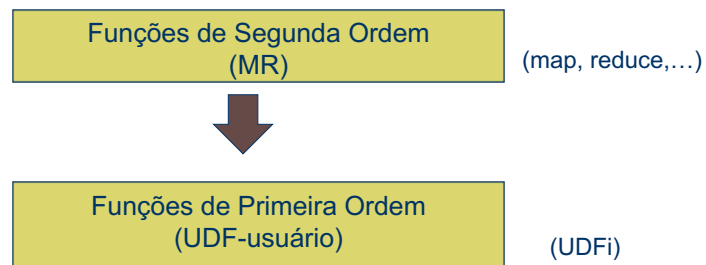
Alexandrov, A, et al., Representations and Optimizations for Embedded Parallel Dataflow Languages, TODS, 2019

AMPBD



27

Modelo de Programação



AMPBD

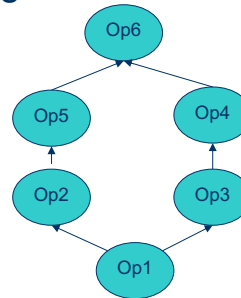


28



Modelo de Programação

- Transformação da expressão em uma *DSL* em uma *Representação Intermediária*
 - Grafo de dependências
- Aplicação modelada como uma ordem parcial de operações de segunda ordem:
 - $op1 \rightarrow op2, op3$
 - $op3 \rightarrow op4$
 - $op2 \rightarrow op5$
 - $op4, op5 \rightarrow op6$
 - Onde $op_i \in DSL$



AMPBD

DEXL
 DATA EXTREME LAB

29



Desafio quanto à aplicação

- Modelar como uma combinação das primitivas do modelo (API).
- No caso MapReduce, essencialmente:
 - `map()`
 - `reduce()`
 - `combine()`

AMPBD

DEXL
 DATA EXTREME LAB

30

Modelo de Programação

Fábio Porto



- Modelo **funcional** de programação
 - Semântica de transformações implementada como funções
- Funções de primeira ordem implementadas pelos usuários:
 - **map** ((in_key, in_value), *F*) -> (inter_key, inter_value list)
 - **reduce** ((inter_key, inter_value list), *F*) -> (out_key, out_value list)
 - **combine** ((inter_key, inter_value list), *F*) -> (out_key, out_value list)

AMPBD

DEXL
 DATA EXTREME LAB

31

Funções Primeira ordem

Fábio Porto



```
function writeWords(key, value){
    StringTokenizer itr = new StringTokenizer(value.toString());

    while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one )
    }
}
```

AMPBD

DEXL
 DATA EXTREME LAB

32

função *writeWord* com segunda ordem Map

Fábio Porto



```

Public static class LineToWordMapper
    extends Mapper<LongWritable, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    Public void map(LongWritable key, Text value, Context context) throws IOException,
    InterruptedException {

        StringTokenizer itr = new StringTokenizer(value.toString());

        while (itr.hasMoreTokens()) {

            word.set(itr.nextToken());
            context.write(word, one);    /* gera um par (chave: "word", valor:one) */

        }

    }
}

```

AMPBD

DEXL
 DATA EXTREME LAB

33

A semântica da função map (k,v) corresponde a

Fábio Porto




- **for each** line in fileIn
 for each key in line
 write(key, 1)
- Veja que o primeiro “for” que varre as linhas do dataset de entrada é implícito no Framework
 - As funções map/reduce operam sobre uma coleção cuja iteração é realizada pela implementação da função de segunda ordem

AMPBD

DEXL
 DATA EXTREME LAB

34

Fábio Porto



Função primeira ordem conta-palavras


```
function conta-palavra(key, values) {
    int sum = 0;
    for (IntWritable val : values) {
        sum += val.get();
    }
    write(key, sum);
}
```

DEXL
DATA EXTREME LAB

AMPBD

35

Fábio Porto



Uma função *conta-palavra* em primeira ordem do tipo Reduce

```
public static class FrequencyReducer extends
    Reducer<Text,IntWritable,Text,IntWritable> {

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
        IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

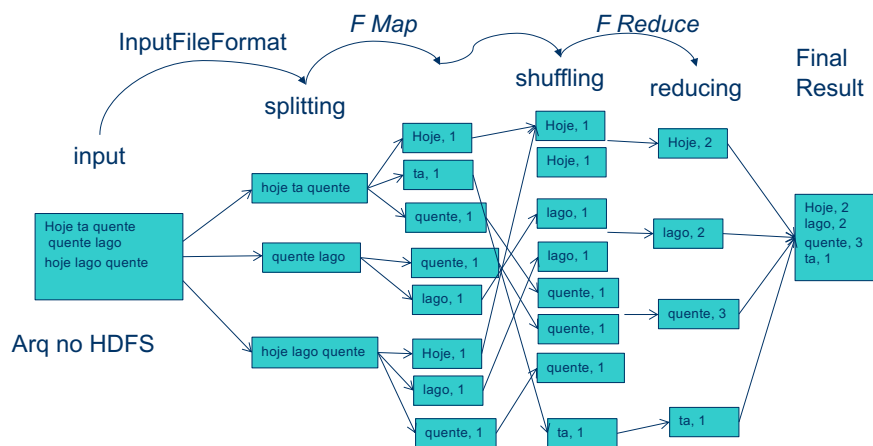
DEXL
DATA EXTREME LAB

AMPBD

36

Exemplo: Conta palavras - dataflow

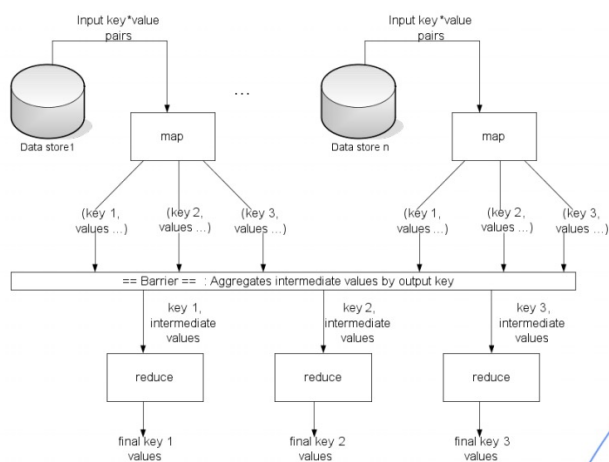
Fábio Porto



DEXL
DATA EXTREME LAB

37

Fábio Porto



AMPBD

DEXL
DATA EXTREME LAB

38

Restrições do modelo MapReduce para Dataflows

Fábio Porto



- Poucas funções
 - semântica sobrecarregada:
 - map pode : 1 -1; 1-n;
- Dataflow modelado como vários “jobs” MR
- Dataflow requer gravação de arquivos entre jobs (i.e. entre atividades)
 - ineficiência

AMPBD

DEXL
DATA EXTREME LAB

39

Fábio Porto




- Exemplo wordcount()
 - notebook

AMPBD

DEXL
DATA EXTREME LAB

40

Fábio Porto




APACHE SPARK

AMPBD

DEXL
DATA EXTREME LAB

41

Fábio Porto



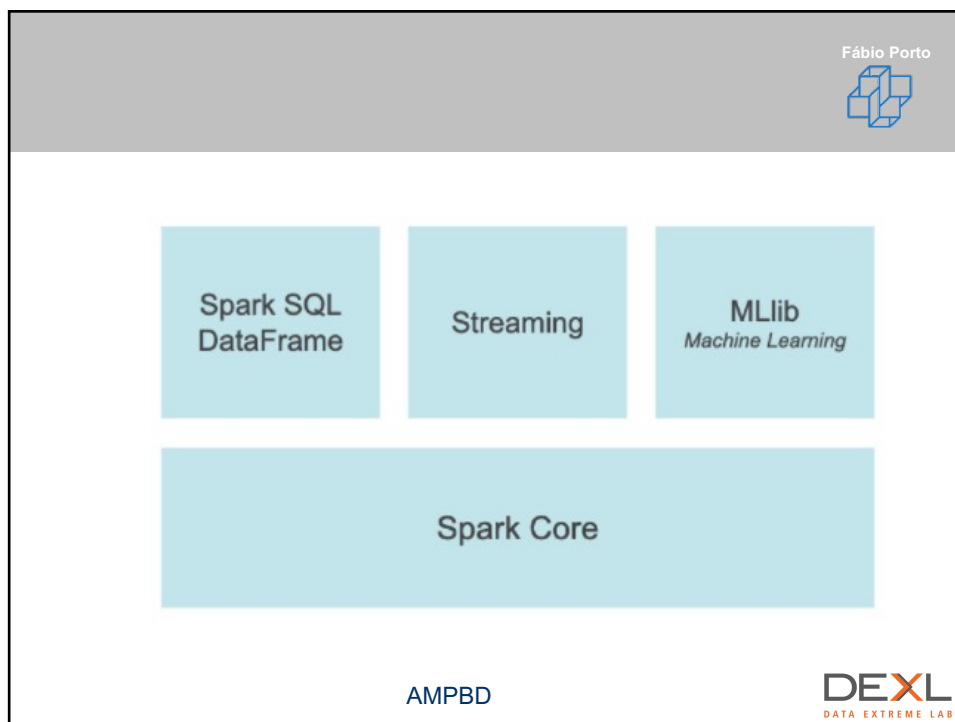
Apache Spark

- Sistema aberto projeto Apache, desenvolvido originariamente em UC Berkeley, AmpLab,
 - Compatível com os dados em HDFS
- Extensão do modelo MapReduce para duas classes de aplicações
 - Algoritmos Iterativos (machine Learning, grafos)
 - Mineração de dados (R, Python)
- Melhoria de desempenho (fator 100) /* para casos especiais */
 - processamento in-memory
 - Grafo de tarefas
- Interface de programação
 - APIs para Java, Scala, Python, R

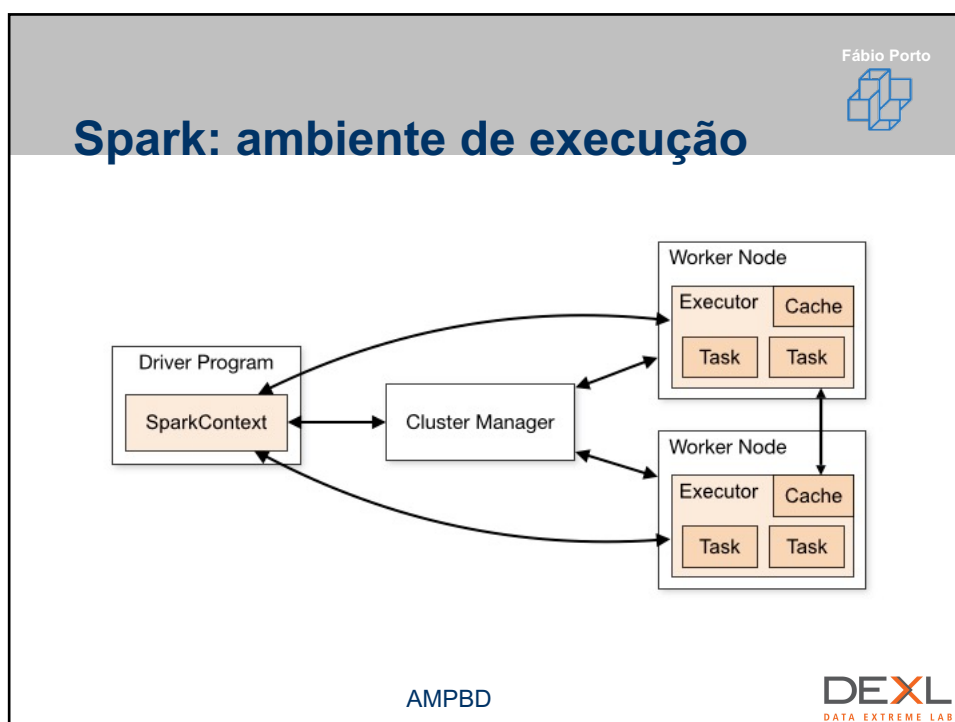
AMPBD

DEXL
DATA EXTREME LAB

42



43



44

Fábio Porto

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”

Each worker has 4 cores on which to run tasks

Master allocates some cores for your application

...

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

AMPBD

45

Fábio Porto

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 no so.
- 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
- O usuário pode determinar como armazenar os dados
 - Disco ou memória RAM
 - Cache de dados em memória
- Operações: transformações e ações

AMPBD

46

RDD

Fábio Porto



- RDD é uma interface para referência à coleção de dados
- Cada item do RDD é 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}

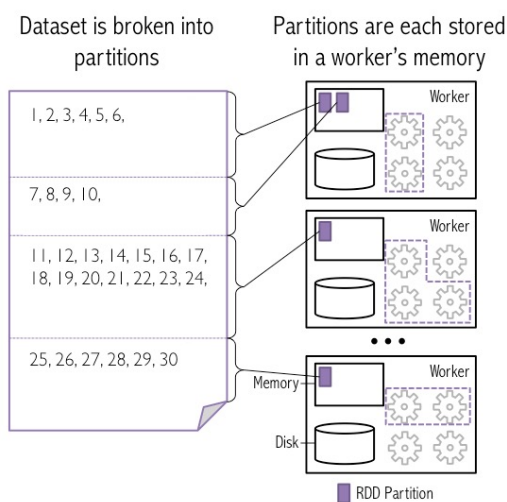
AMPBD

DEXL
 DATA EXTREME LAB

47

Dataset distribuido em partições

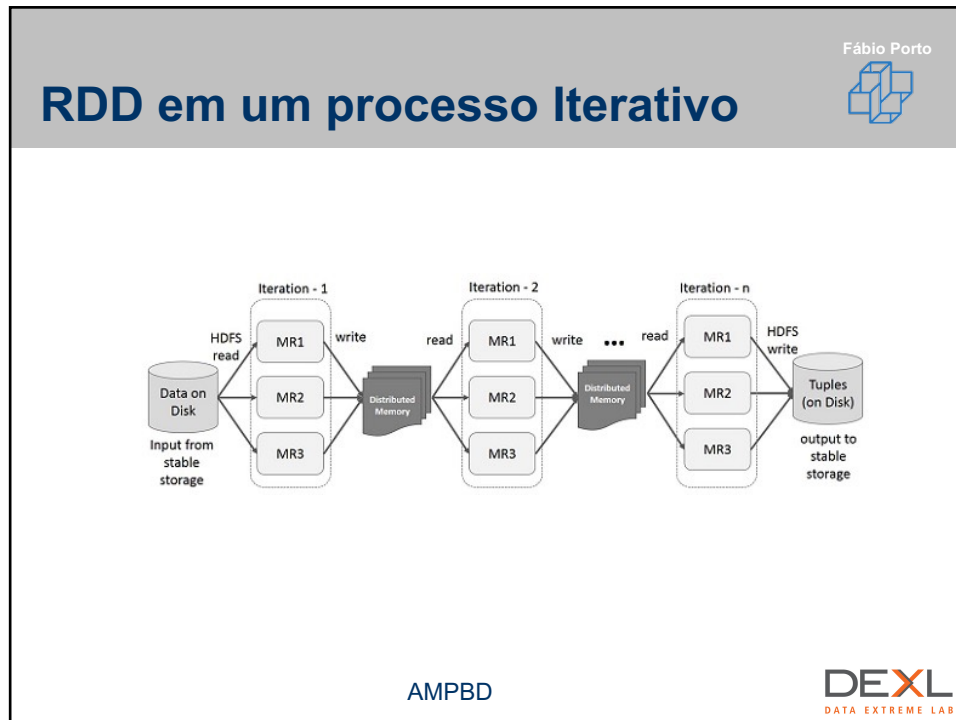
Fábio Porto



AMPBD

DEXL
 DATA EXTREME LAB

48



49

Fábio Porto

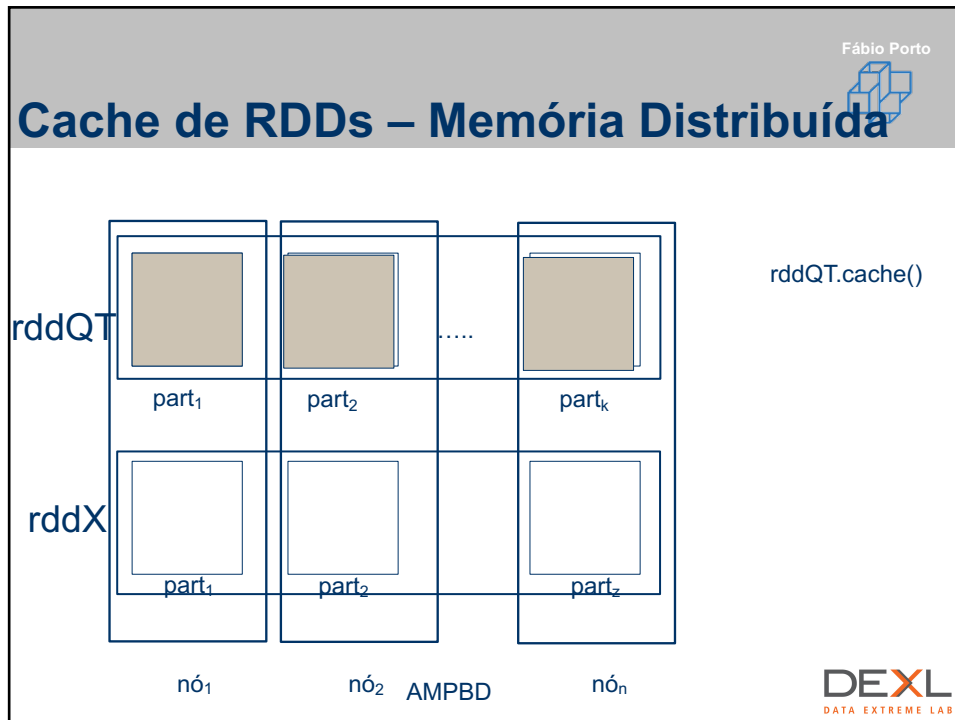
RDD - Propriedades

- “Vive” em um SparkContext (SC)
 - não podem ser compartilhados entre SCs
- Pode ser “fixado” em cache
 - em disco ou em memória
- Exemplo:
 - `text_rdd = sc.textFile(input_file)`
 - `rddQT = text_rdd.mappartition(quadtree)`
 - `rddQT.cache()`
 - `rddQT.count()`

AMPBD

DEXL
DATA EXTREME LAB

50



51

Fábio Porto

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
 - `rdd1.join (rdd2)`
 - `groupby(key)`
 - `filter(lambda line: "DEXL" in line).count()`

AMPBD

DEXL
DATA EXTREME LAB

52

Fábio Porto



Modelo de Execução: Lazy

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

AMPBD

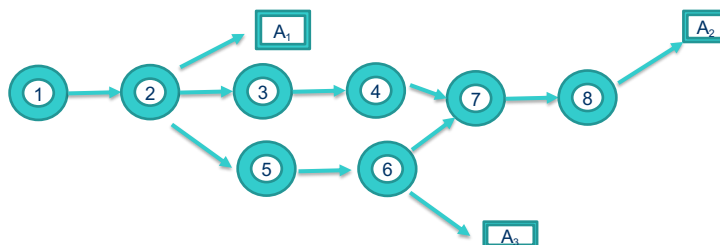
DEXL
DATA EXTREME LAB

53

Fábio Porto



Exemplo Ilustrativo



AMPBD

DEXL
DATA EXTREME LAB

54

Transformações e Ações

Fábio Porto



- Transformações
 - 1,2,3,4,5,6,7,8,9,10
- Ações
 - A1, A2, A3

AMPBD

DEXL
 DATA EXTREME LAB

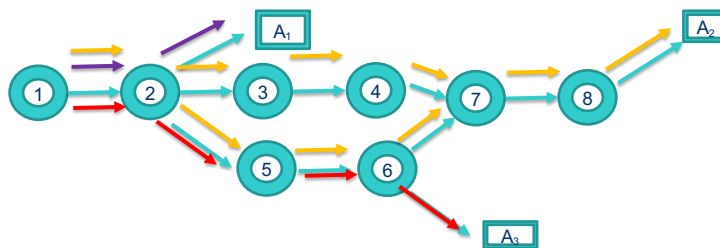
55

Possível escalonamento

Fábio Porto



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



AMPBD

DEXL
 DATA EXTREME LAB

56

Fábio Porto



Modelo de Execução: 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

AMPBD

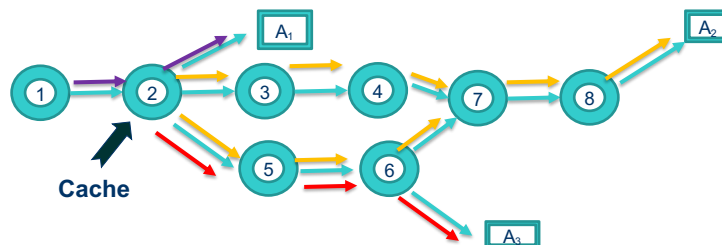
DEXL
 DATA EXTREME LAB

57

Fábio Porto



Execução com Cache



AMPBD

DEXL
 DATA EXTREME LAB

58

Fábio Porto

Execução Spark x Partições

One task is launched for each partition

Running task

Unused core

map (f) : Each task makes a new partition by calling f (e) on each entry e in the original partition

1, 2, 3, 4, 5, 6,	→	0, 1, 2, 3, 4, 5,
7, 8, 9, 10,	→	6, 7, 8, 9,
11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,	→	10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
25, 26, 27, 28, 29, 30	→	24, 25, 26, 27, 28, 29

AMPBD

DEXL
DATA EXTREME LAB

59

Fábio Porto

Coletando o resultado final

collect () : Gathers the entries from all partitions into the driver

0, 1, 2, 3, 4, 5,	→
6, 7, 8, 9,	→
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,	→
24, 25, 26, 27, 28, 29	→

Results sent to your SparkContext in the driver


0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29

AMPBD

DEXL
DATA EXTREME LAB

60

Fábio Porto




SPARK API

AMPBD

DEXL
DATA EXTREME LAB

61

Fábio Porto



Leitura dos Dados

- Transforma a entrada em RDDs
 - parallelize
 - textFile
 - newAPIHadoopFile

AMPBD

DEXL
DATA EXTREME LAB

62

Operação do Contexto Spark:Parallelize

Fábio Porto



- Processa coleções no programa “driver”
- Dados de coleções são copiados para criar RDDs
- O número de partições pode ser definido como parâmetro da função;

Parâmetros:

- 1.Entrada: Uma coleção de dados, por exemplo listas, tuplas e entre outros
- 2.Saída: Um *RDD* com os dados representando cada elemento da coleção

AMPBD

DEXL
DATA EXTREME LAB

63

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

Fábio Porto



```
number_rdd = sc.parallelize([1, 2, 3, 4])
aggregate= number_rdd.reduce(lambda a,b:a+b)
print(aggregate)
>>> 10
```

```
reducelist= number_rdd.reduce(lambda a,b:[a,b])
print( reducelist)=[[1, 2], 3], 4]
```

AMPBD

DEXL
DATA EXTREME LAB

64

textFile (path [, #Partitions])

Fábio Porto



- Pode-se ler: um arquivo, uma lista de arquivos em um diretório, todos os arquivos em um diretório na criação de um RDD;
- `sparkcontext.textFile()`
 - Lê de HDFS, S3 ou qq Sistema de arquivo suportado pelo Hadoop.
 - Parâmetros: `path`, `[#partições]`

Parâmetros:

1. **Entrada:** arquivo informado pelo usuário
2. **Saída:** *RDD* com uma ou mais partições representando as linhas de um arquivo

AMPBD

DEXL
DATA EXTREME LAB

65

Exemplo de aplicação com o uso do *textFile*:

Fábio Porto



```

text_rdd = sc.textFile(input_file) # leitura dos dados e
                                   # criação do RDD
rows = text_rdd.collect() # retorna os valores contidos no RDD
                           # para o programa driver.

for line in rows:             # Lista todas as linhas do RDD
    print(line)
  
```

Obs: `Collect()` retorna um array para o “driver” e não um RDD ou Dataframe. Deve-se usar com cautela pois requer qtd suficiente de memória no driver.

AMPBD

DEXL
DATA EXTREME LAB

66

newAPIHadoopFile

Fábio Porto



API de leitura de Hadoop files com indicação de parâmetros de leitura

Parâmetros:

1. Entrada:

1. Lista de caminhos para os arquivos a serem lidos
2. Classe indicando o formato de armazenamento do arquivo sendo lido
3. Classe do parâmetro *key*
4. Classe do parâmetro *value*

2. Saída: Um *RDD* com a *key* e correspondente valores

AMPBD

DEXL
DATA EXTREME LAB

67

Exemplo de aplicação com o *newAPIHadoopFile*:

Fábio Porto



```
text_rdd = sc.newAPIHadoopFile(
    input_file_path,
    'org.apache.hadoop.mapreduce.lib.input.TextInputFormat',
    'org.apache.hadoop.io.LongWritable',
    'org.apache.hadoop.io.Text',
    conf={'textinputformat.record.delimiter': '\n\n'})
numEntries=text_rdd.count()
result = text_rdd.collect() /* traz o conteúdo p driver */
print(result)
```

AMPBD

DEXL
DATA EXTREME LAB

68

Transformações

Fábio Porto



- Produzem um RDD a partir de um RDD
- 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 n informados como parâmetro

69

Funções de *map*

Fábio Porto



- map
- flatMap
- mapPartitions

AMPBD

70

Transformação: map

Fábio Porto



- `map (f): rdd_in.map(f)`
 - Aplica f a cada entrada do `rdd_in`;
- Resulta em *RDD* com uma entrada para cada resultado de $f(\text{rdd_in})$;

Parâmetros:

1. Entrada: Uma função f com parâmetros correspondentes aos tipos dos dados nos elementos de `rdd_in`
2. Saída: Um novo *RDD* com elementos do tipo da saída da função f .

AMPBD

DEXL
DATA EXTREME LAB

71

Exemplo de aplicação com o map:

Fábio Porto



```
number_rdd = sc.parallelize(range(1,1000))
squared_rdd = number_rdd.map(lambda x: x * x)
print(number_rdd.reduce(lambda x, y:x+y)
>>>499500
Print(squared_rdd.reduce(lambda x, y:x+y)
>>>.....
filtered_rdd=squared_rdd.filter(lambda x: x<1000)
filtered_rdd.count()
6
```

AMPBD

DEXL
DATA EXTREME LAB

72

Transformação: flatMap

Fábio Porto



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

Parâmetros:

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

AMPBD

DEXL
 DATA EXTREME LAB

73

Exemplo de aplicação com o flatMap

Fábio Porto



- readTextDirectory=sc.textFile("/Users/fabioporto/Documents/CSV/2018-census-totals-by-topic-national-highlights-csv",2)
- rdd_map =rddTextDirectory.map(lambda a: a.split(" "))
- rdd_map.count()
- >>>4388
- rdd_flat=readTextDirectory.flatMap(lambda a: a.split(" "))
- >>> rdd_flat.count()
- 12315

AMPBD

DEXL
 DATA EXTREME LAB

74

Transformação: mapPartitions

Fábio Porto



- A operação mapPartitions recebe uma partição de RDD de entrada e produz um RDD de saída. 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*

AMPBD

DEXL
 DATA EXTREME LAB

75

Exemplo : mapPartition

Fábio Porto



```
def processPartition (inline):
    resultList = [ ]
    for l in inline: /* processa partição */
        for x in l: /*itera sobre os elementos */
            resultList.append(x)
    return resultList
rddMap = rddin.mapPartitions(processPartition)
x=rdd.Map.collect()
Len(x)
>>>155989
```

AMPBD

DEXL
 DATA EXTREME LAB

76

PairRDD



- Um tipo variante de RDD construído para representar key-value pairs;
 - 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(Users/fabioporto/Documents/Data/pair.txt)

AMPBD



77

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:

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

AMPBD



78

Exemplo Transformação *reduceByKey*:

Fábio Porto



```
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)
y=x.collect()
print(y)
>>[('Project', 3), ('Gutenberg's', 3), ('Alice's', 1), ('Adventures', 2), ('in', 2),
    ('Wonderland', 2)]
```

AMPBD

DEXL
DATA EXTREME LAB

79

Outras transformações key/value

Fábio Porto




- groupByKey()
- combineByKey()
- mapValues(func)
- flatMapValues(func)
- keys()
- values()
- sortBykey()

AMPBD

DEXL
DATA EXTREME LAB

80

Fábio Porto




Ações

AMPBD

DEXL
DATA EXTREME LAB

81

Fábio Porto



Ações

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

AMPBD

DEXL
DATA EXTREME LAB

82

Ação: *reduce*

Fábio Porto



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.

AMPBD

DEXL
DATA EXTREME LAB

83

Exemplo de aplicação com a função *reduce*:

Fábio Porto



```
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
```

AMPBD

DEXL
DATA EXTREME LAB

84

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

AMPBD

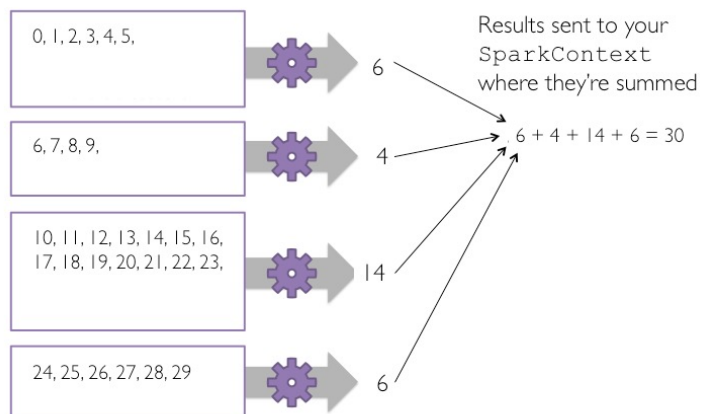


85


Operação Reduce distribuída



count () : Each task counts the entries in one partition




86


Fábio Porto

Outras ações

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


AMPBD

87

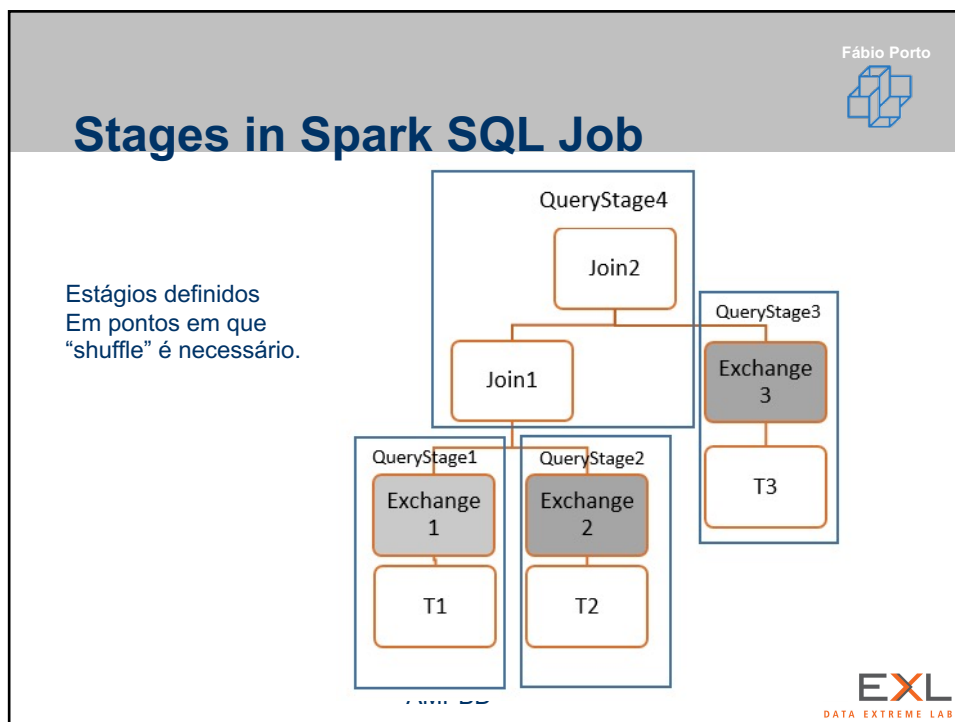
Fábio Porto

Estágios

- Um dataflow em Spark é dividido em estágios
- Cada estágio delimita um fluxo que pode ser executado inteiramente sem troca de dados entre estágios

AMPBD

88



89

Fábio Porto


HDFS – SISTEMA DE ARQUIVOS DISTRIBUÍDO

AMPBD

DEXL
DATA EXTREME LAB

90


Fábio Porto



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

AMPBD



91


Fábio Porto



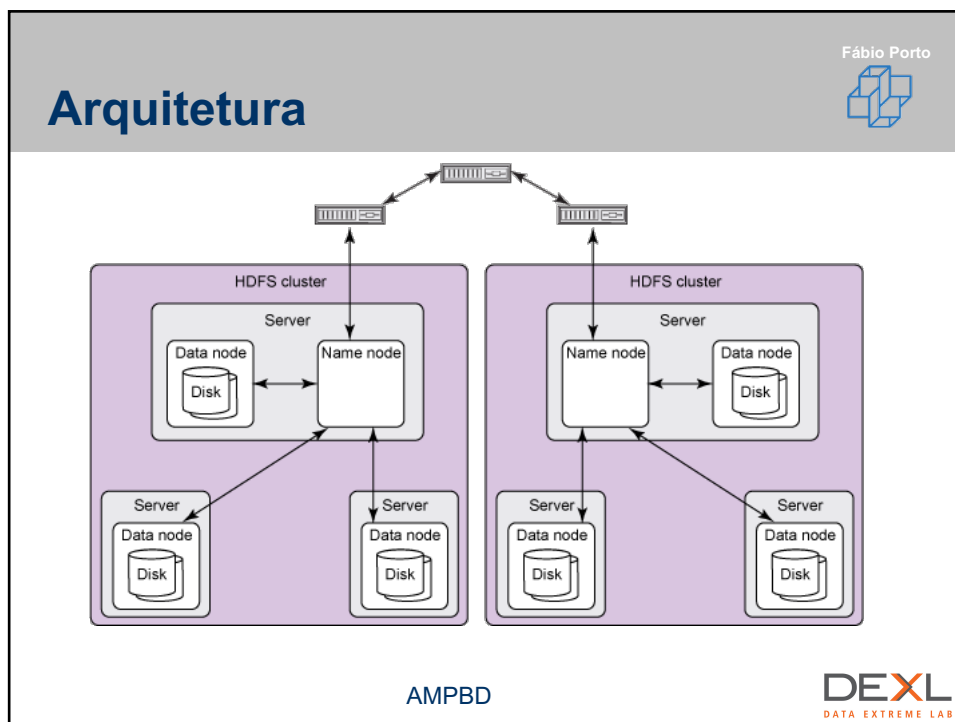
HDFS – Hadoop File System

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

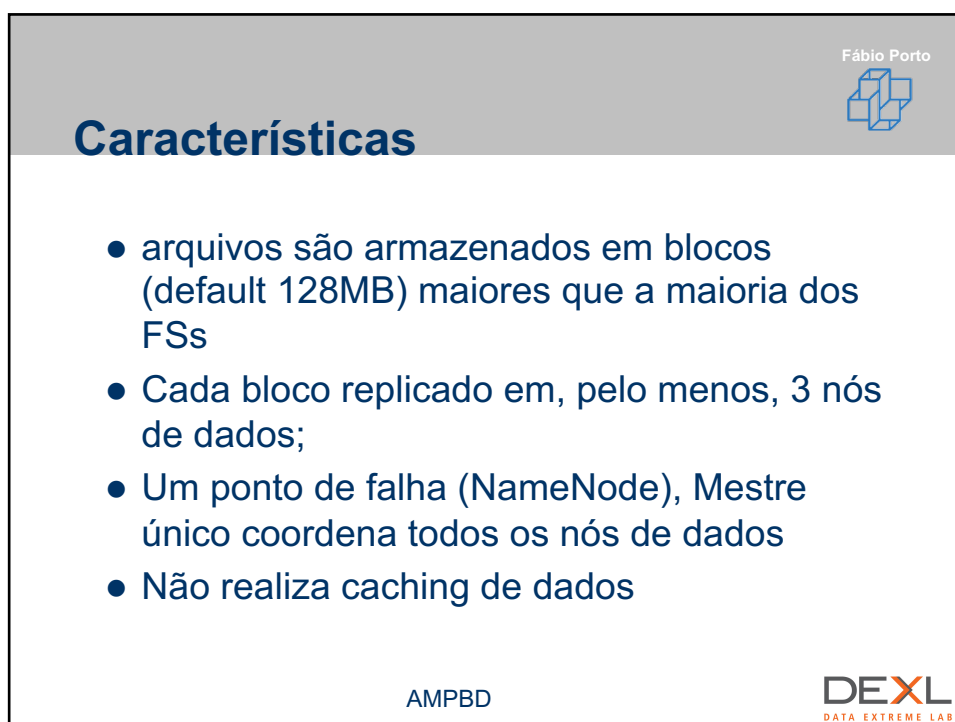
AMPBD



92



93



94

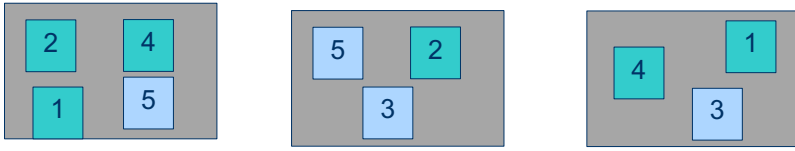
Fábio Porto

NameNode e dataNode

NameNode:
apenas metadados

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

DataNodes: armazena blocos de arquivos



AMPBD

DEXL
DATA EXTREME LAB

95

Fábio Porto

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

AMPBD

DEXL
DATA EXTREME LAB

96

Interface de Manipulação de arquivos

Fábio Porto



- `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`

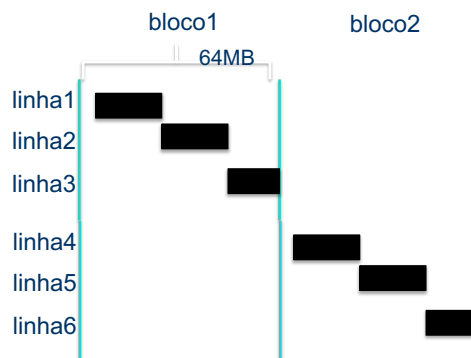
AMPBD

DEXL
DATA EXTREME LAB

97

Distribuição do arquivo em blocos e splits

Fábio Porto

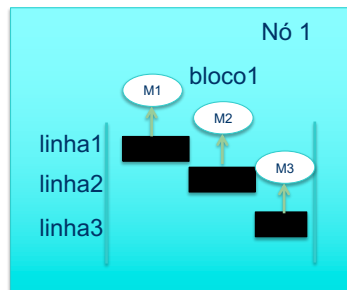


AMPBD

DEXL
DATA EXTREME LAB

98

Splits associados à Funções Map

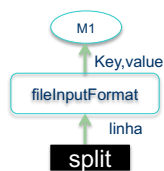


AMPBD



99

Zoon In



Classe Mapper

```
1 .../...
2 FileInputFormat.addInputPath(job, inputPath);
3 job.setInputFormatClass(CustomFileInputFormat.class);
4 .../...
```


Classe InputFileFormat

```
1 public class CustomFileInputFormat extends FileInputFormat<Long>
2
3 @Override
4 public RecordReader<LongWritable, Text> createRecordReader(
5     InputSplit split, TaskAttemptContext context) throws
6     InterruptedException {
7     return new CustomLineRecordReader();
8 }
9 }
```

AMPBD




100


Fábio Porto


Spark SQL

- Facilita o desenvolvimento de aplicações sobre dados estruturados: *dataset*
- *Spark SQL pode ser visto como um Sistema BDR*
- *Possui um otimizador Catalyst*
- *DataFrame*
 - *Armazena dados de forma mais eficiente do que RDDs*
 - *Organizado em colunas como em Tabelas*

AMPBD



101

Fábio Porto


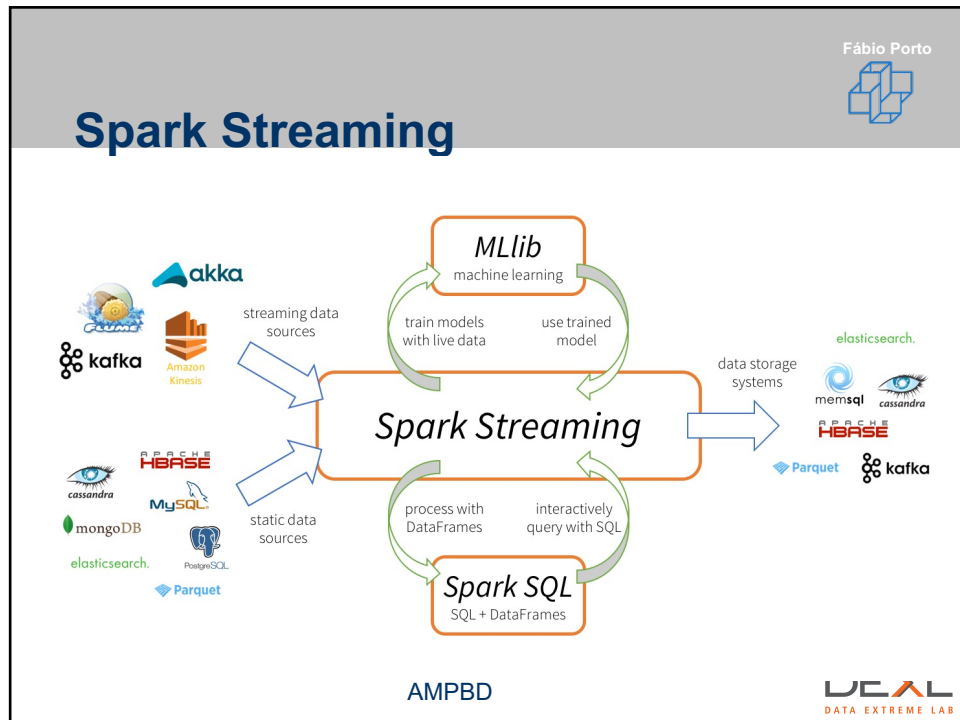
DataFrame API

Tipos de Join :

- inner
- outer
- left outer
- right outer
- left semi

AMPBD


102



103

Fábio Porto

Fim da 2a parte Questões ?

AMPBD

DEXL
DATA EXTREME LAB

104



MODELOS ARQUITETURAIS PARALELOS

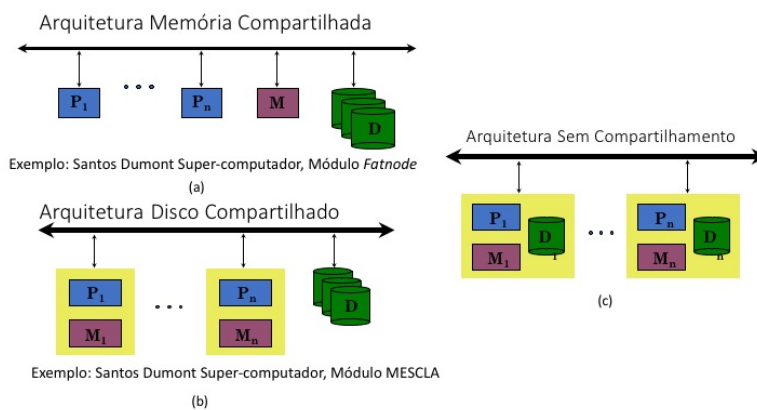
AMPBD

DEXL
 DATA EXTREME LAB

105



Arquiteturas Paralelas



AMPBD

DEXL
 DATA EXTREME LAB

106

Sistemas Memória Compartilhada



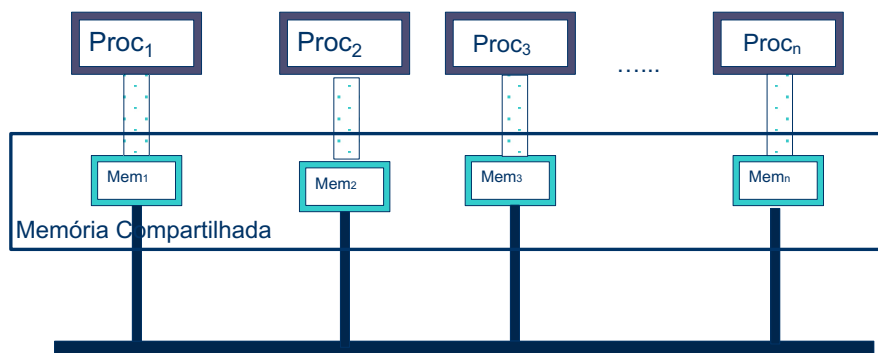
- Composto por vários nós de computação interligados por uma rede de comunicação de dados;
- Os nós de computação têm acesso a uma região de memória compartilhada entre os nós
- O sistema de arquivo pode ser compartilhado distribuído (ex: Lustre ou HDFS)
- Exemplos: Apache Ignite, Alluxio

AMPBD



107

Sistemas de Memória Compartilhada Unified Memory Access (UMA)



AMPBD



108

Arquitetura NUMA: Non-uniform Memory Access

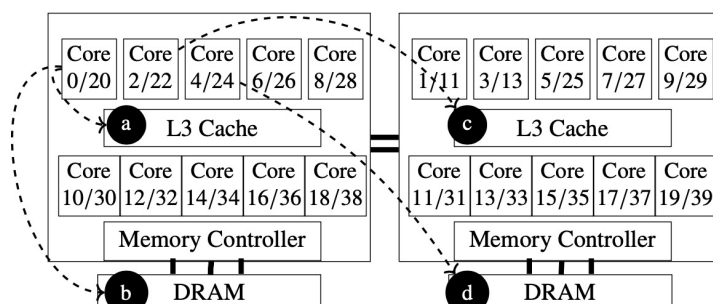


Figura 1.1: Exemplo de uma arquitetura NUMA com 2-nós baseada no Intel Xeon Silver 4114.

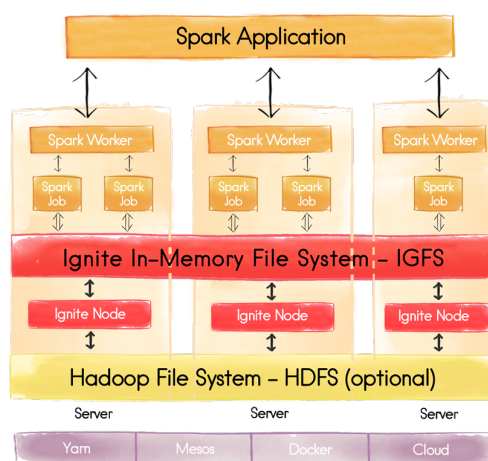
[Dominico, Simone, Tese de Doutorado, UFPR, 2022]

AMPBD



109

Apache Ignite com Apache Spark



IgniteContext ->
IgniteRDD
comunicação Ignite X
HDFS

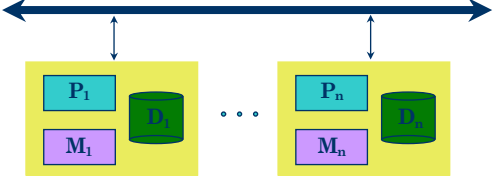
AMPBD



110


Fábio Porto

Sem Compartilhamento



Exemplos : Hadoop, Spark, Flink,
Myria, AsterixDB

AMPBD



 DATA EXTREME LAB


111

Fábio Porto

Desafio

- Um ambiente de execução escalável para processamento de Tera/Petabytes de dados
- uma linguagem integrada sem a complexidade dos controles associados ao paralelismo
- garantias do ambiente

AMPBD



 DATA EXTREME LAB

112