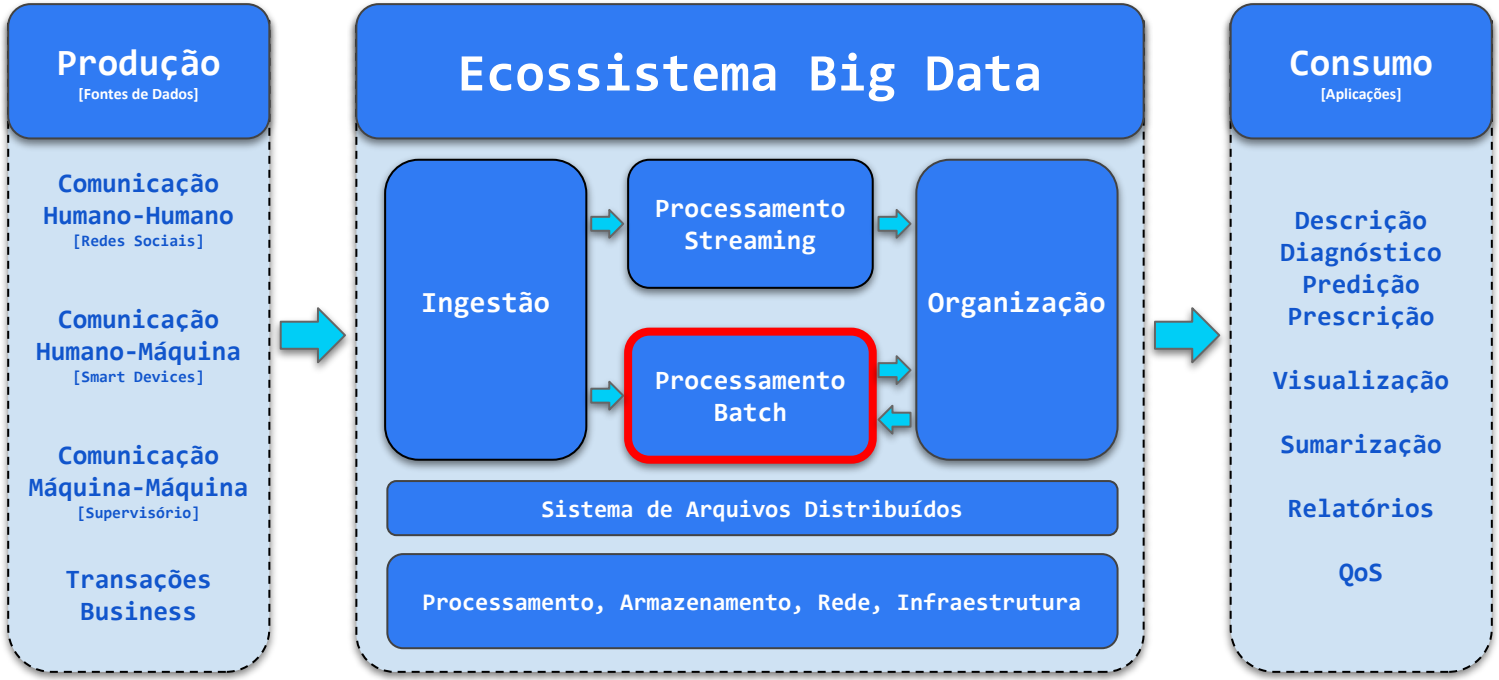

Facens
POS
Facens

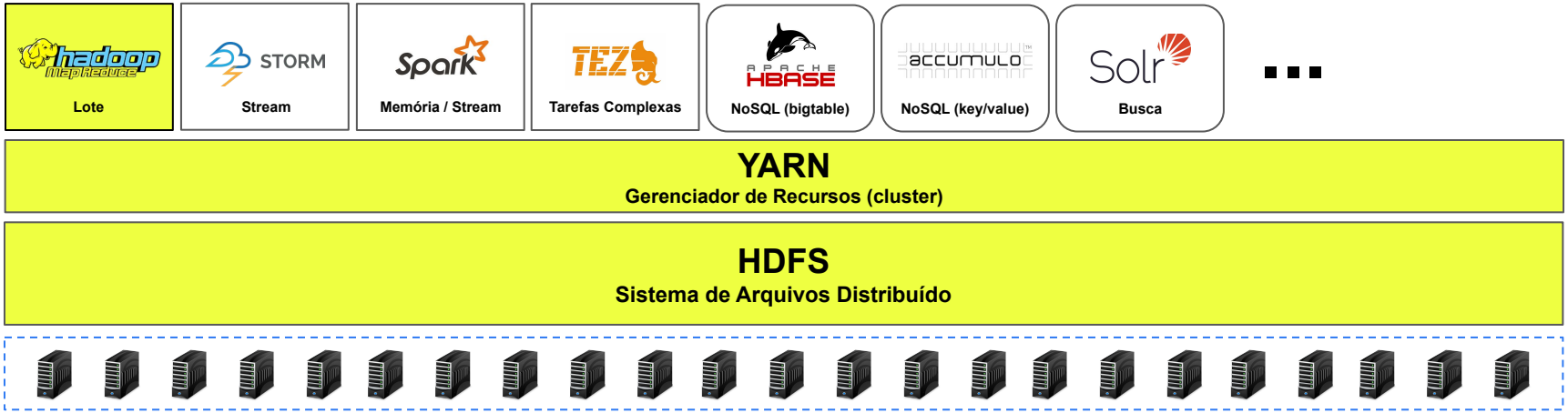
YARN Execution Engines 1

Apache Hadoop MapReduce

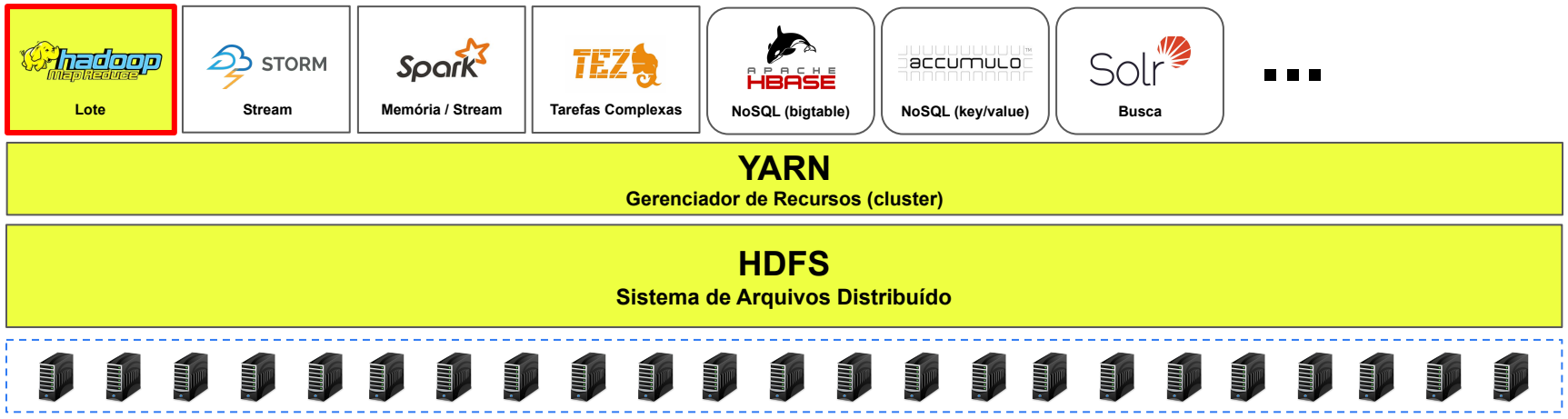
Big Data - Meta-Arquitetura



Big Data - Ecossistema



Big Data - Ecossistema



MapReduce

Vamos separar as coisas

- MapReduce - Técnica de Programação
  - Forma de pensar
  - Não resolve todos os problemas, mas pode ser adaptado para a maioria deles

- MapReduce - Componente do Hadoop
  - APIs
  - Java Interfaces
  - Java Classes

MapReduce

Vamos separar as coisas

- MapReduce - Técnica de Programação
  - **Forma de pensar**
  - Não resolve todos os problemas, mas pode ser adaptado para a maioria deles

- MapReduce - Componente do Hadoop
  - APIs
  - Java Interfaces
  - Java Classes

MapReduce - Como organizar cartas de um baralho?

MapReduce - Como organizar cartas de um baralho?

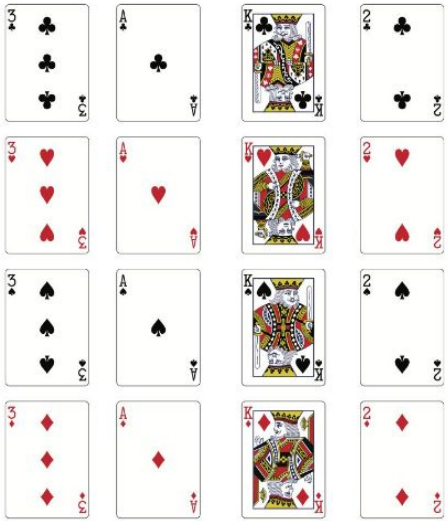
Alternativas

- Carta por carta
- Separar em pilhas por tipo (13 pilhas)
- Separar em pilhas por naipe (4 pilhas)

MapReduce - Como organizar cartas de um baralho?

Alternativas

- Carta por carta
- **Separar em pilhas por tipo (13 pilhas)**
- Separar em pilhas por naipe (4 pilhas)

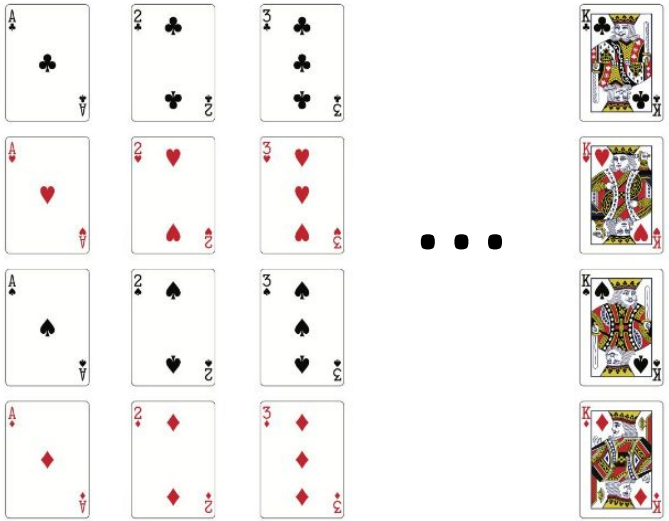


...

MapReduce - Como organizar cartas de um baralho?

Alternativas

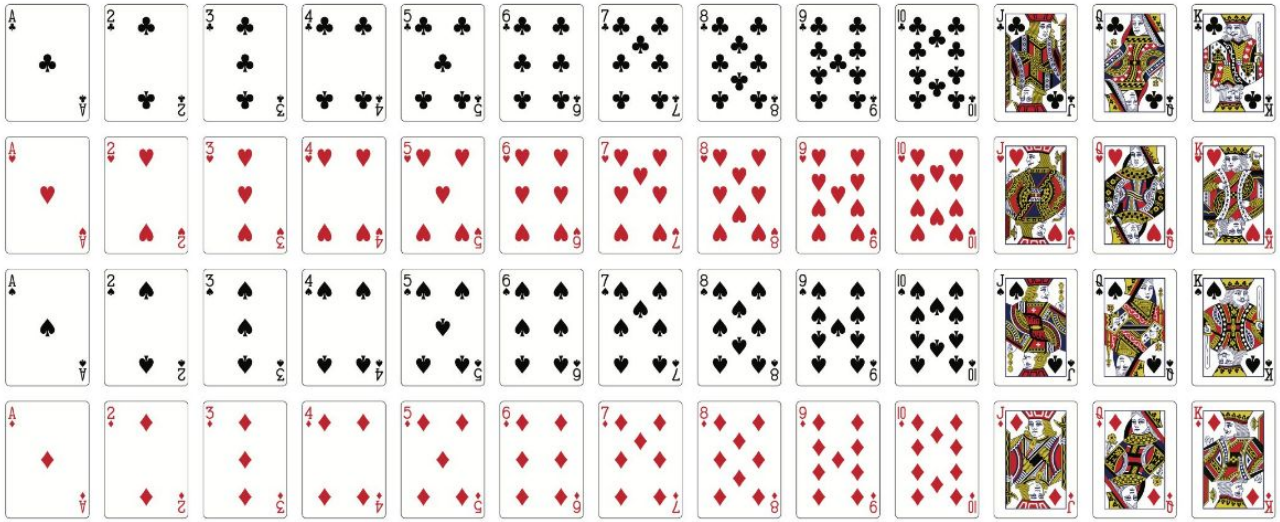
- Carta por carta
- **Separar em pilhas por tipo (13 pilhas)**
- Separar em pilhas por naipe (4 pilhas)



MapReduce - Como organizar cartas de um baralho?

Alternativas

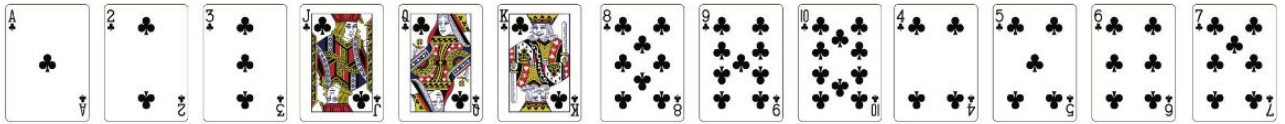
- Carta por carta
- **Separar em pilhas por tipo (13 pilhas)**
- Separar em pilhas por naipe (4 pilhas)



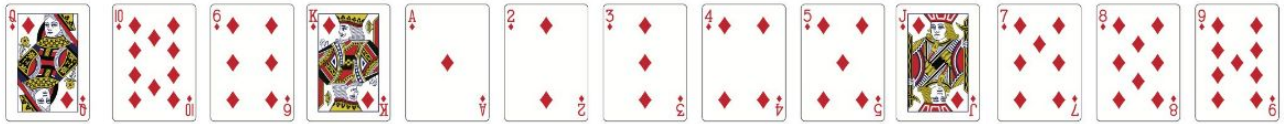
MapReduce - Como organizar cartas de um baralho?

Alternativas

- Carta por carta
- Separar em pilhas por tipo (13 pilhas)
- **Separar em pilhas por naipe (4 pilhas)**



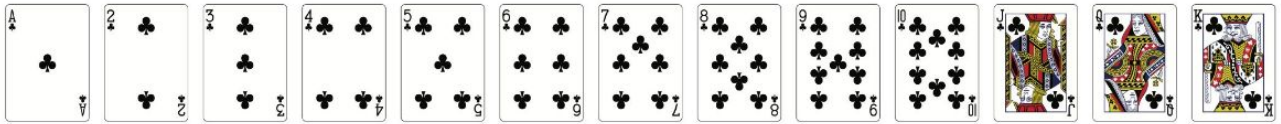
...

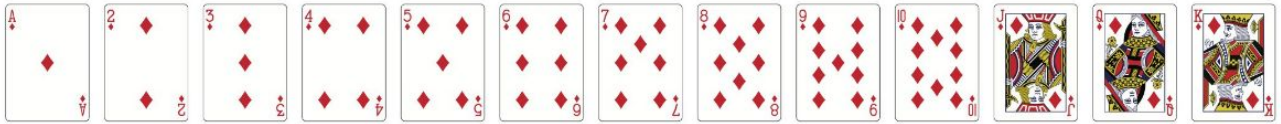


MapReduce - Como organizar cartas de um baralho?

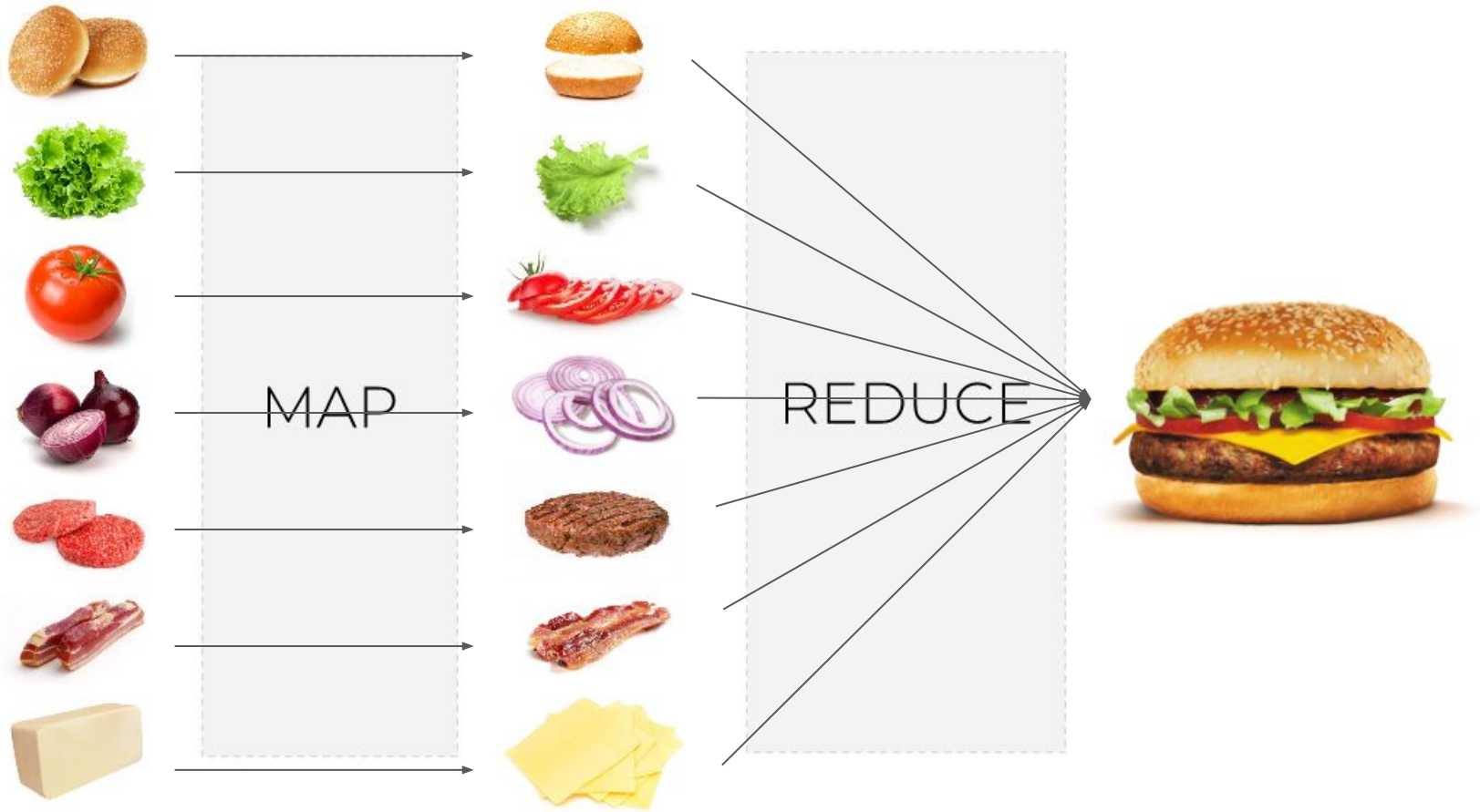
Alternativas

- Carta por carta
- Separar em pilhas por tipo (13 pilhas)
- **Separar em pilhas por naipe (4 pilhas)**





MapReduce - Analogia 2



MapReduce

Map: Transformar

Reduce: Agregar

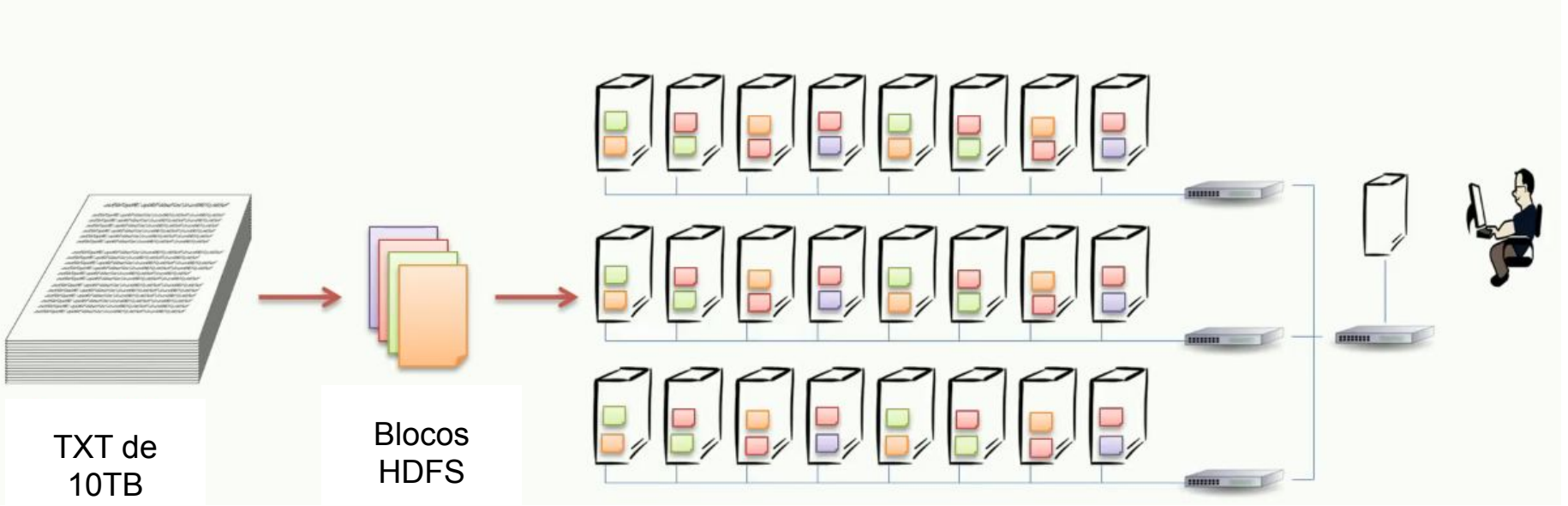
MapReduce - Hadoop

Map: Transformar blocos ou pedaços dos blocos, gerar resultado intermediário

Reduce: Usar resultado do Map para agregar e produzir resultado final

Framework: Monitora e abstrai todos os desafios de processamento distribuído

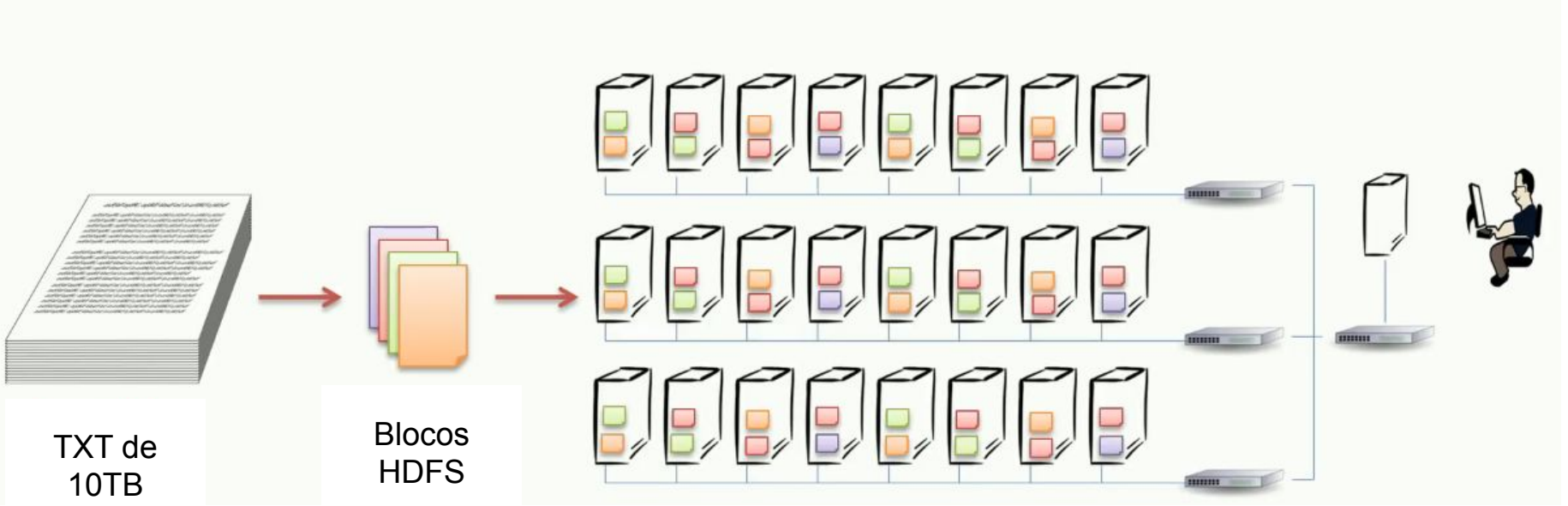
Problema



Problema: Como contar quantas x a mesma palavra apareceu

Ler o arquivo linha por linha?

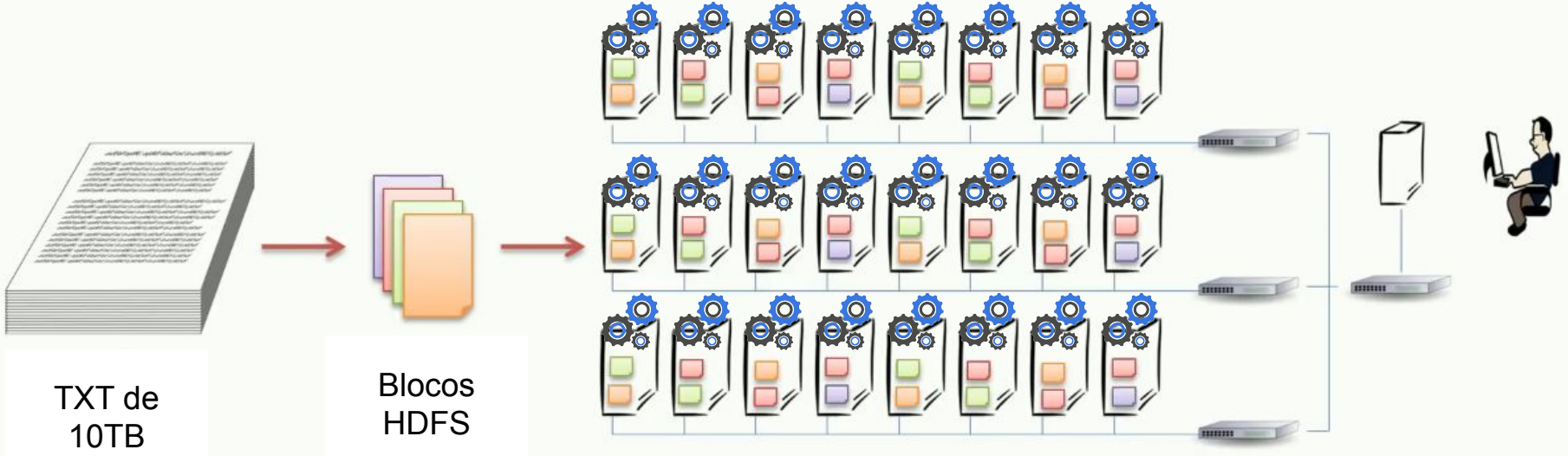
Manter um contador por palavra distinta?



Problema: Como contar quantas x a mesma palavra apareceu

Cada datanode pode contar as palavras dos blocos

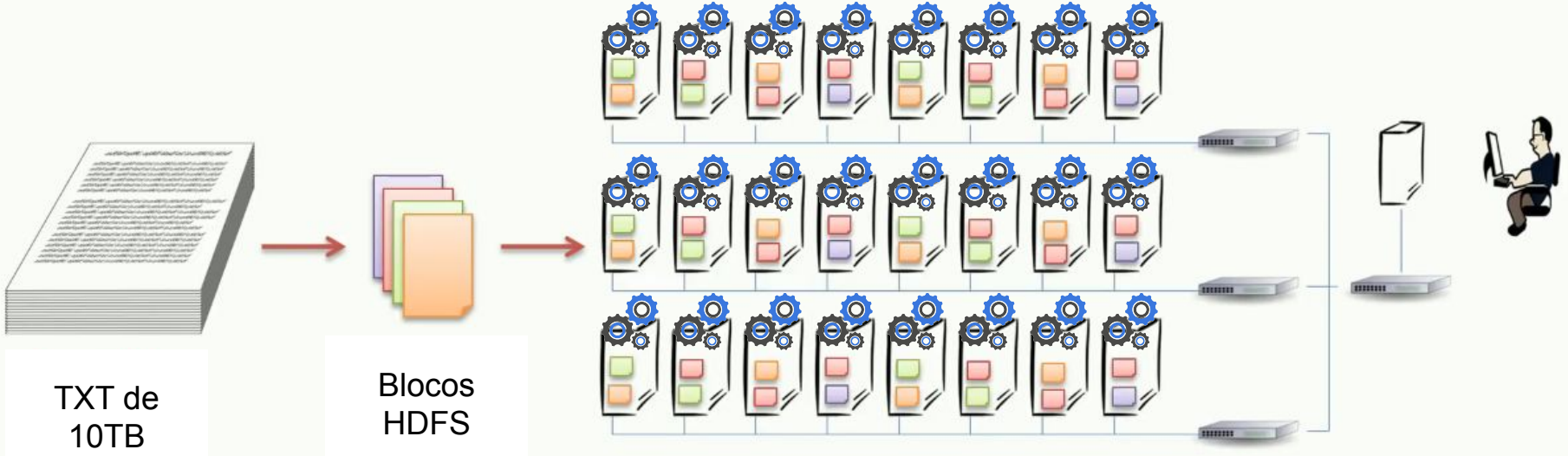
Depois, apenas os contadores das mesmas palavras são somados



Problema: Como contar quantas x a mesma palavra apareceu

Map: Cada datanode pode contar as palavras dos blocos

Reduce: Depois, apenas os contadores das mesmas palavras são somados



Perguntas

Como a função de map vai “ler” as linhas de um bloco?

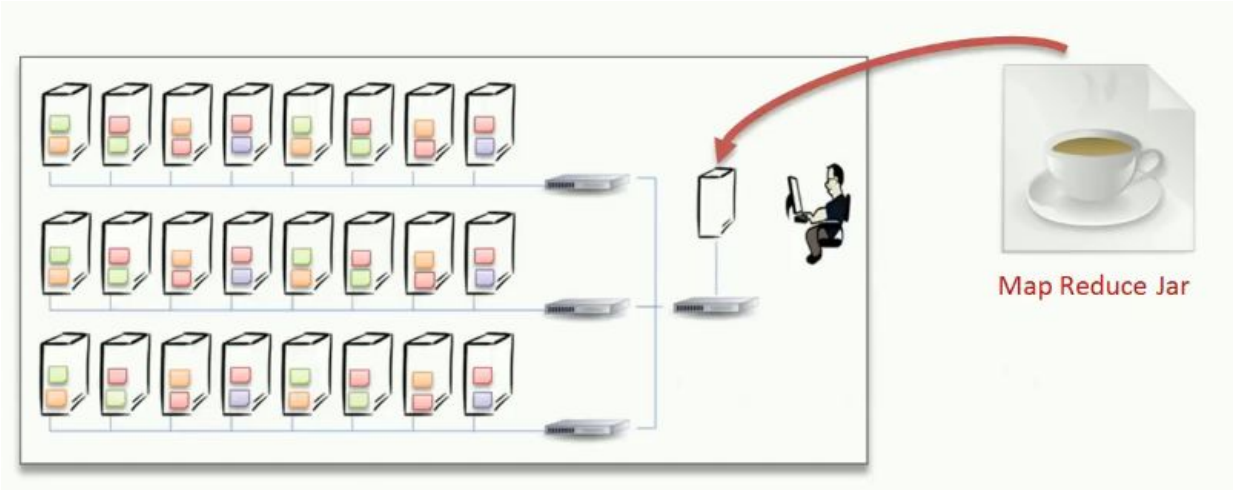
Como cada um dos maps, de nodes distintos, serão enviados para a função final de reduce?

Perguntas

Como a função de map vai “ler” as linhas de um bloco?

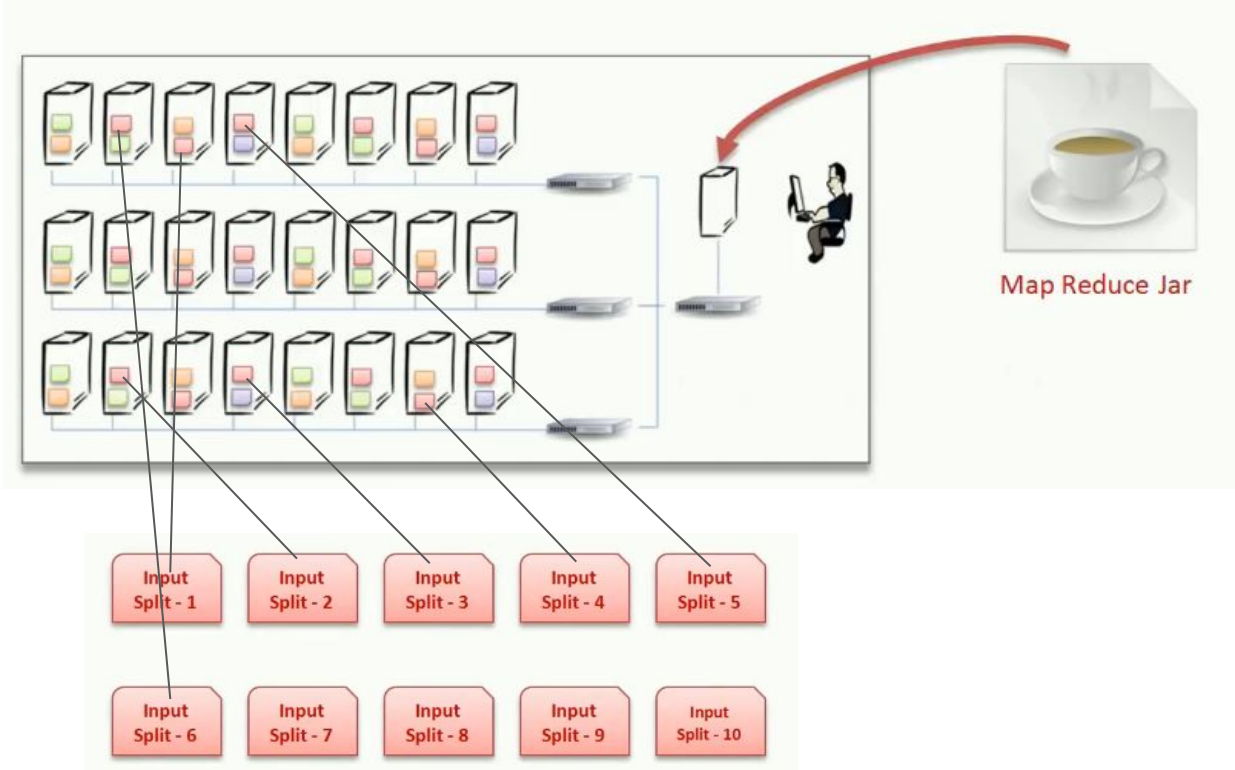
Como cada um dos maps, de nodes distintos, serão enviados para a função final de reduce?

Passo a passo



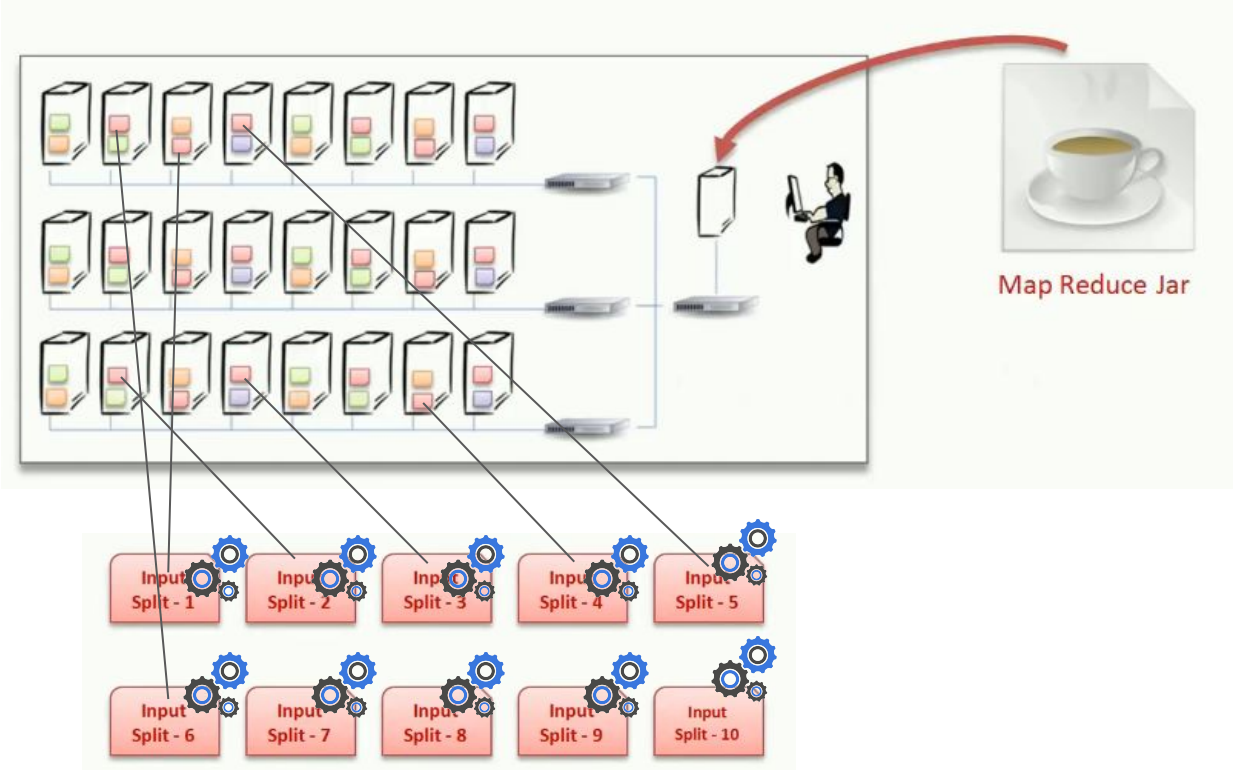
Passo 1: Dividir o input

Depende do número de blocos, parâmetros etc.



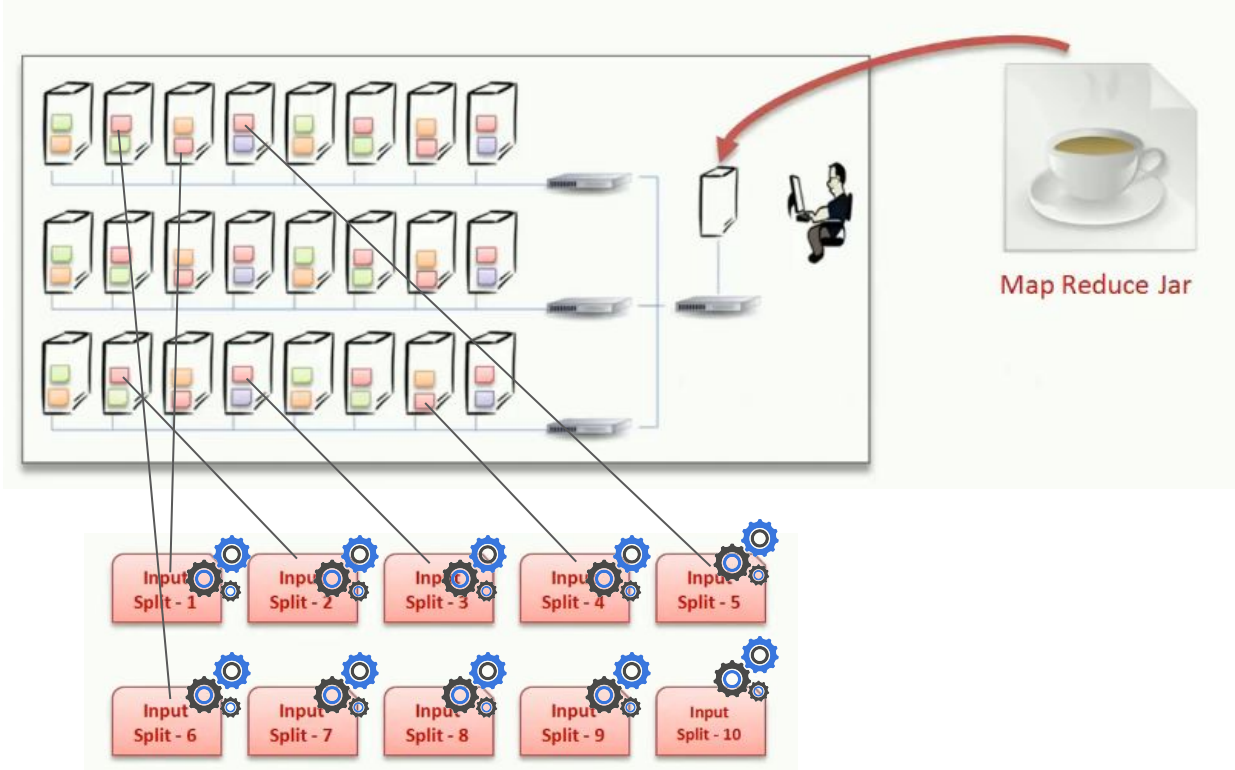
Passo 2: Disparar map() em cada split

Cada bloco, por exemplo



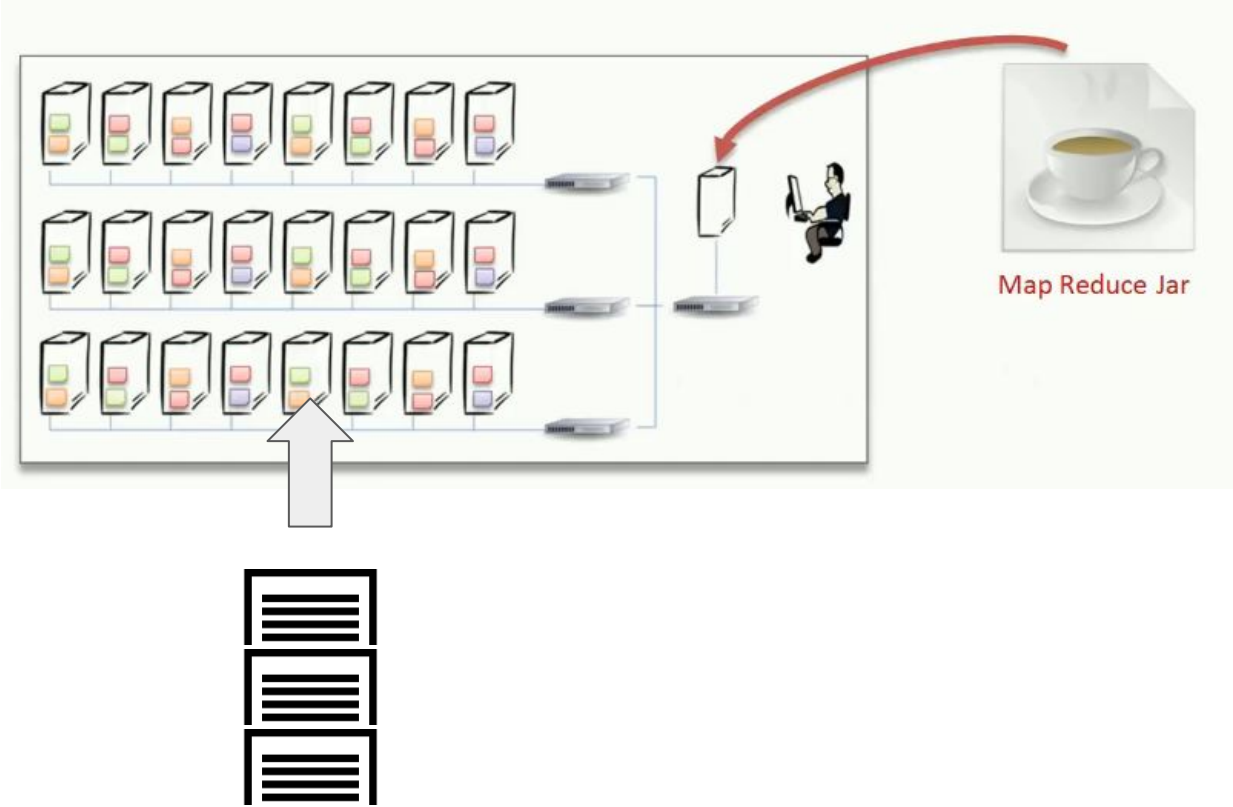
Passo 3: Monitorar todos os maps

Erros, interrupções etc.



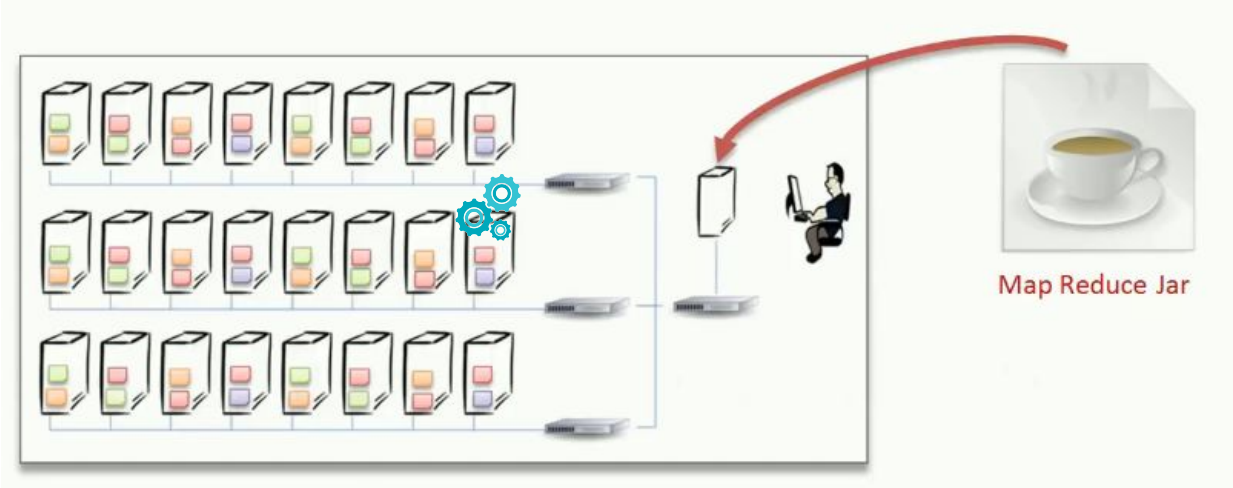
Passo 4: Gerar resultado intermediário que será usado

Rearmazenar no próprio cluster (append de resultados)



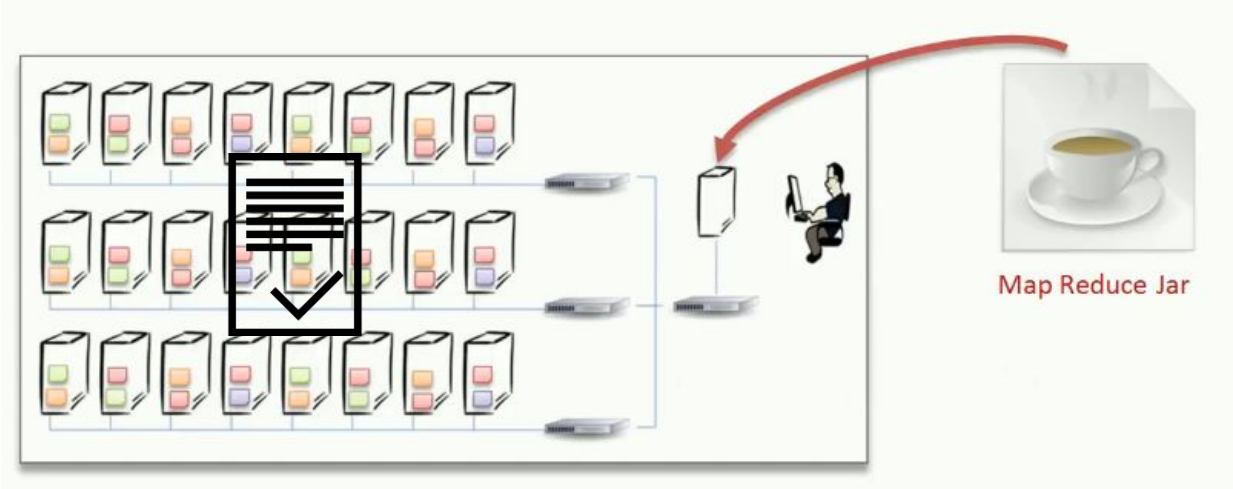
Passo 5: Disparar o reduce() em uma ou mais instâncias

Tamanho do resultado intermediário (blocos) também pode determinar



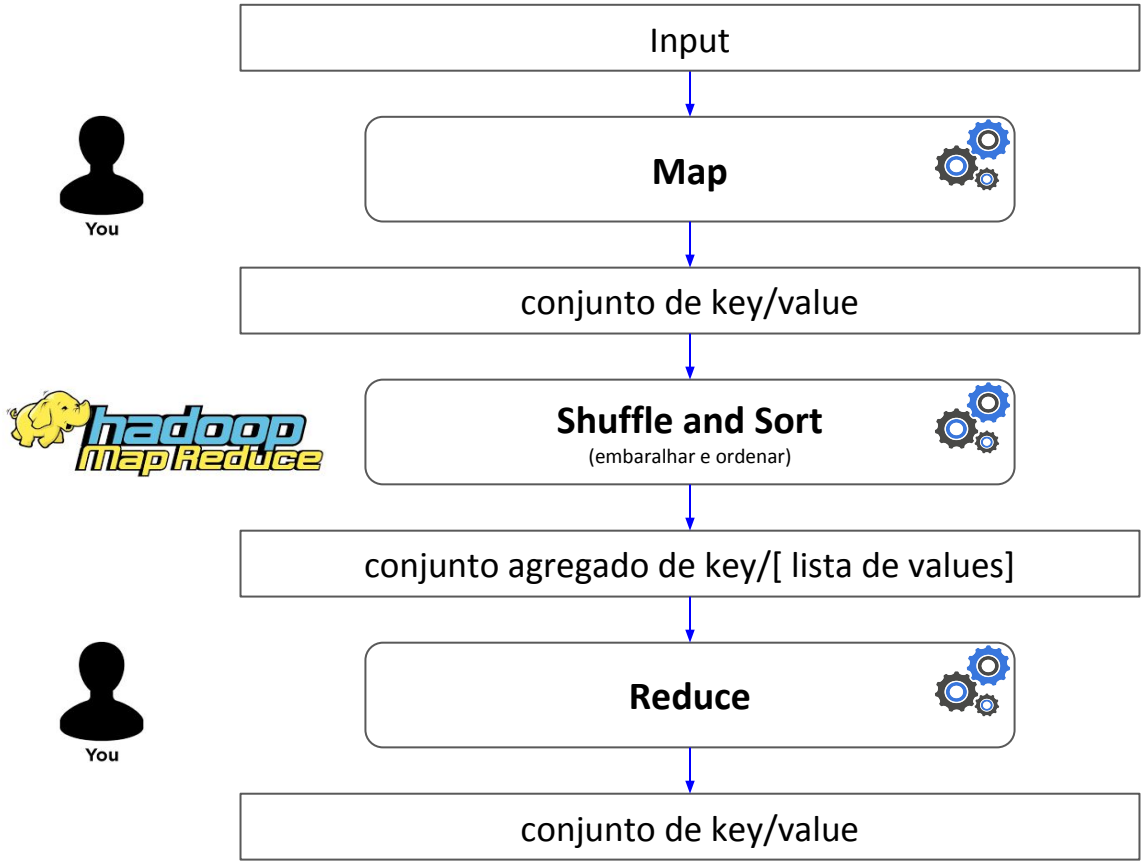
Passo 6: Gerar resultado final

Armazenar o resultado final também no cluster



MapReduce

Sequência completa



Cada node emite pares de key-value com base no registro de entrada, um registro por vez

Gerenciada pelo Hadoop. Transfere resultado dos mappers para reducers agregando por key

Cada reducer recebe uma key e uma lista de values associados a essa key. Reducers emitem então zero ou mais pares key/value

Exemplo MapReduce

Input:

id_usuario	nota	id_restaurante	visitou_em
43	3	b83d	1532046887
27	5	4g45	1516408487
19	1	b83d	1488414887
43	5	wqi4	1488674087
43	3	43gh	1527899687
27	3	xc19	1524184487

Exemplo MapReduce

Input:

id_usuario	nota	id_restaurante	visitou_em
43	3	b83d	1532046887
27	5	4g45	1516408487
19	1	b83d	1488414887
43	5	wqi4	1488674087
43	3	43gh	1527899687
27	3	xc19	1524184487

Exemplo MapReduce

Input:

id_usuario	nota	**id_restaurante**	visitou_em
43	3	b83d	1532046887
27	5	4g45	1516408487
19	1	b83d	1488414887
43	5	wqi4	1488674087
43	3	43gh	1527899687
27	3	xc19	1524184487

Exemplo MapReduce - Quantos restaurantes cada user visitou?

id_usuario	nota	id_restaurante	visitou_em
43	3	b83d	1532046887
27	5	4g45	1516408487
19	1	b83d	1488414887
43	5	wqi4	1488674087
43	3	43gh	1527899687
27	3	xc19	1524184487

Mapper
id_usuario:id_restaurante
“restaurante visitados pelos usuários”

43:b83d | 27:4g45 | 19:b83d | 43:wqi4| 43:43gh | 27:xc19

Shuffle and Sort
(embaralhar e ordenar)

19:b83d | 27:4g45,xc19 | 43:43gh,b83d,wqi4

Reducer
função: tamanho(values)
“quantidade de restaurante visitados”

19:1 | 27:2 | 43:3

MapReduce - Exemplo 2

"A bela jovem saiu com um belo rapaz para um belo passeio no belo parque desta bela cidade"

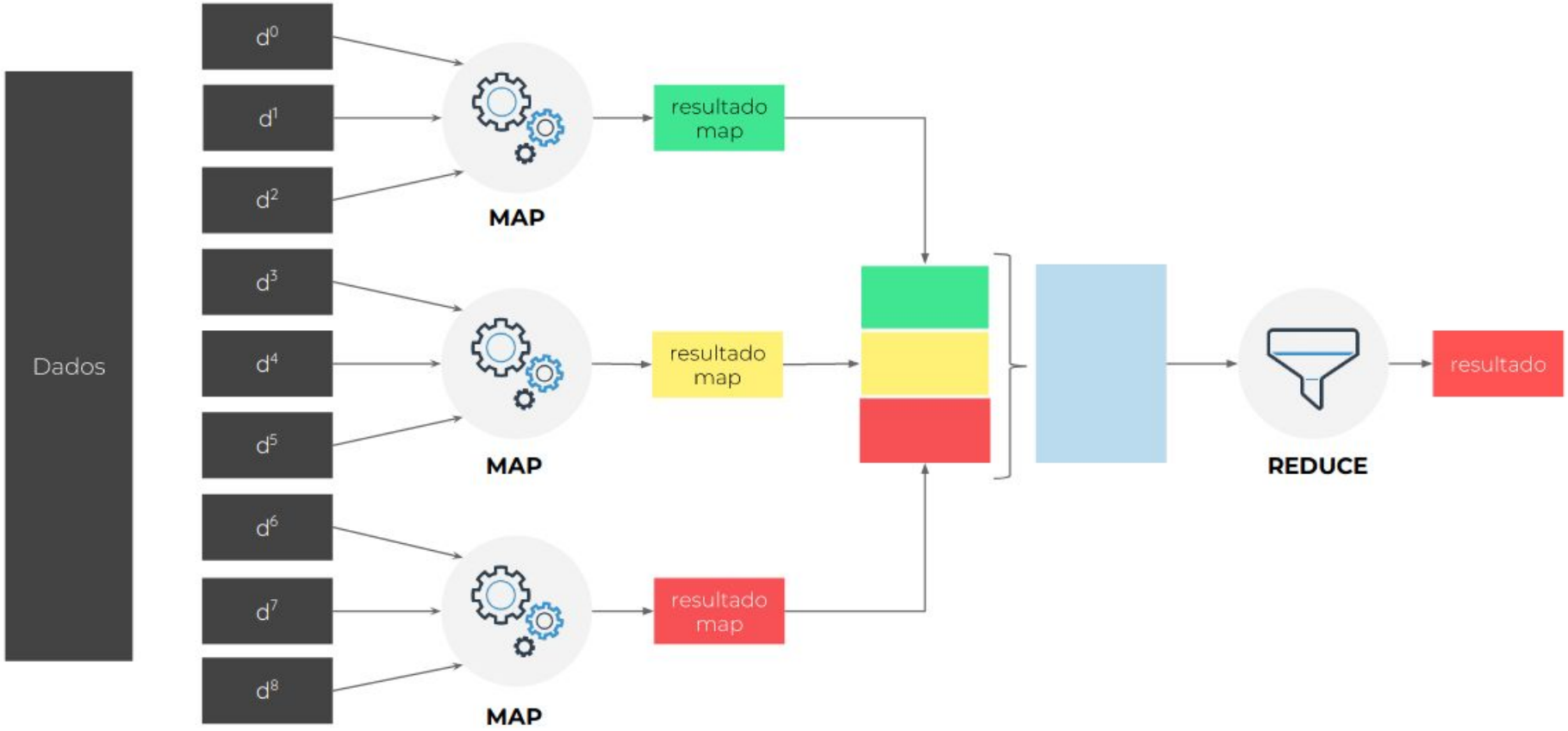
Chave	Valor
A	1
bela	1
jovem	1
saiu	1
com	1
um	1
belo	1
rapaz	1
para	1
um	1
belo	1
passeio	1
no	1
belo	1
parque	1
desta	1
bela	1
cidade	1

Chave	Valor
A	1
bela	1
bela	1
belo	1
belo	1
belo	1
cidade	1
com	1
desta	1
jovem	1
no	1
para	1
parque	1
passeio	1
rapaz	1
saiu	1
um	1
um	1

Chave	Valor
A	1
bela	1, 1
belo	1, 1, 1
cidade	1
com	1
desta	1
jovem	1
no	1
para	1
parque	1
passeio	1
rapaz	1
saiu	1
um	1, 1

Chave	Valor
A	1
bela	2
belo	3
cidade	1
com	1
desta	1
jovem	1
no	1
para	1
parque	1
passeio	1
rapaz	1
saiu	1
um	2

Resumindo


Dados
d0
d1
d2
d3
d4
d5
d6
d7
d8
MAP
MAP
MAP
resultado map
resultado map
resultado map
REDUCE
resultado

Resumindo

Map

Cada nó na fase **map** emite pares chave-valor com base no registro de entrada, um registro por vez.

Shuffle

A fase **shuffle** é tratada pela estrutura Hadoop. Ela transfere os resultados dos **mappers** para os **reducers** juntando os resultados por meio **chave**.

Reduce

A saída dos **mappers** é enviado para os **reducers**. Os dados na fase **reduce** são divididos em partições onde cada **reducer** lê uma **chave e uma lista de valores** associados a essa chave. Os reducers emitem **zero ou mais pares chave-valor** com base na lógica utilizada.

Prática MapReduce

The Data Society