



# Fundamentos de Big Data

UNIDADE 07

## Trabalhando com o MapReduce

O uso de tecnologias escaláveis é muito importante quando trabalhamos com *big data* – afinal, queremos que o nosso código funcione bem não importando se estaríamos manipulando dados de 1, 10, 100, ou 100 milhões de usuários. Isso também vale para as atividades de *big data analytics* – no caso, problemas de análise de dados e de ciência de dados para um grande volume de informações. Dentro deste conjunto de soluções temos o *MapReduce*.

## | Trabalhando com MapReduce

Neste momento trabalhamos com o Hadoop, HDFS, PySpark, DataFrames e o Jupyter. Também vimos alguns outros temas correlatos, como a manipulação de arquivos CSV. Agora, é a hora de nos aprofundarmos no MapReduce. Vamos lá?

## | MapReduce

### MapReduce na análise de dados

Segundo Grus (2021), MapReduce é um modelo de programação utilizado para o **processamento paralelo de grandes conjuntos de dados**, como *logs* de sites, textos de livros, arquivos de imagem, dentre outras coleções de itens. Por isso, ele é usado em empresas para finalidades de análise de dados em aplicações como, por exemplo (Schneider, 2012):

#### Detectção de fraudes em serviços financeiros



As empresas sempre buscam por maneiras de prever e prevenir fraudes. Uma das formas de fazer isso é analisando grandes volumes de dados. Por exemplo, o MapReduce é muito útil para **reconhecer padrões**, o que é bem importante na detecção de fraudes. Com ele, uma instituição financeira pode identificar transações suspeitas e sinalizá-las para uma verificação adicional de segurança.



Você sabia que um site de compras online pode gerar uma enorme quantidade de *logs*? O framework MapReduce é bem interessante para analisar esses dados, especialmente os relacionados às compras dos consumidores. Com ele, é possível identificar **padrões de comportamento** de compra, o que ajuda a criar programas de marketing direcionados e mais eficazes.

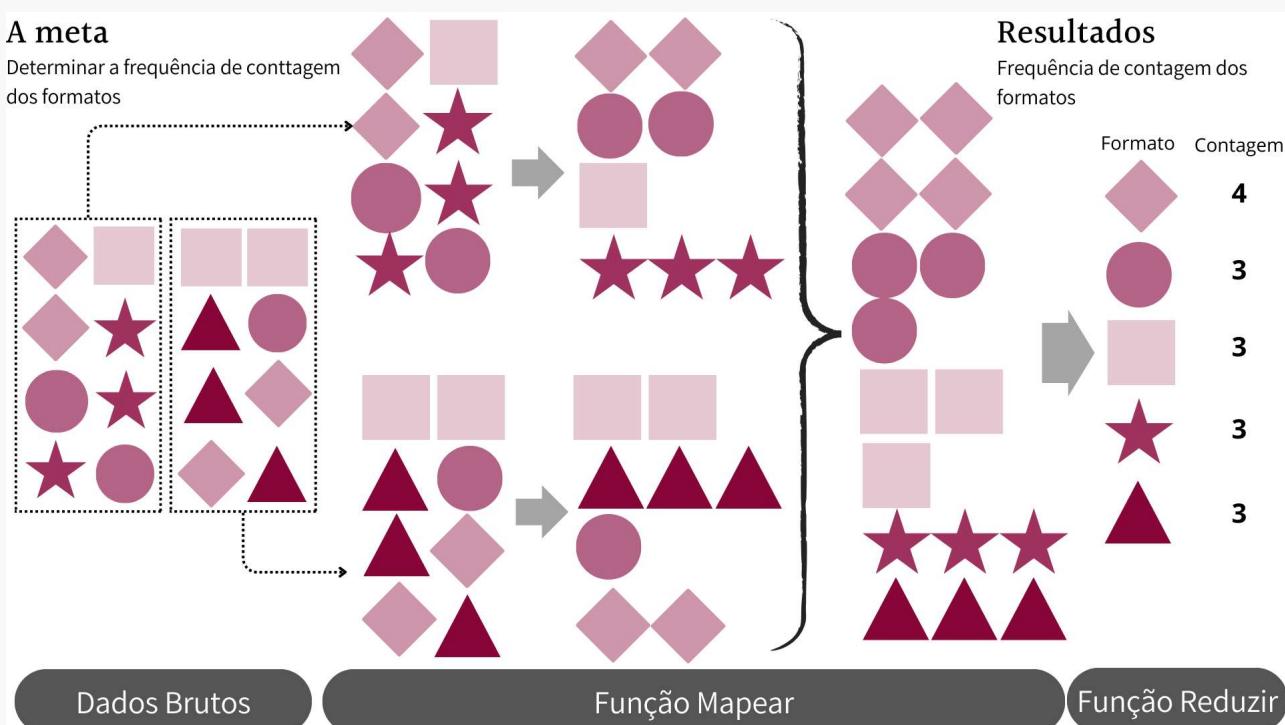
## Otimização da cadeia de suprimentos



O MapReduce é uma ferramenta muito importante para **analisar dados** da cadeia de suprimentos. Com ele, é possível simplificar o processo de aquisição de estoque, determinando de forma mais eficiente como, onde e quando comprar. Isso, na prática, significa uma cadeia de suprimentos com gastos menores e uma efetividade maior.

Agora, como ele funciona? Vamos usar a imagem abaixo para explicar:

Figura 1. Etapas de mapeamento e redução do MapReduce



1. Começamos com os dados brutos. Isto significa, na prática, em um ou mais arquivos que leríamos com o Spark.
2. Depois, vem o **MapReduce** de fato. Começamos com a função de mapeamento (**map**). Aqui, os “*dados brutos*”, que são uma coleção de formas geométricas, são divididos e classificados. Cada forma é mapeada para um par chave-valor, onde a chave é o tipo da forma (círculo, estrela, triângulo) e o valor para a chave é “1”, indicando a ocorrência de uma forma.
3. Depois, temos a função de redução (**reduce**). Aqui, a contagem total de cada tipo de forma é calculada somando os valores de cada chave. O resultado é uma lista de pares chave-valor, onde a chave é o tipo da forma e o valor é a contagem total daquela forma em específico. Por exemplo, três triângulos.

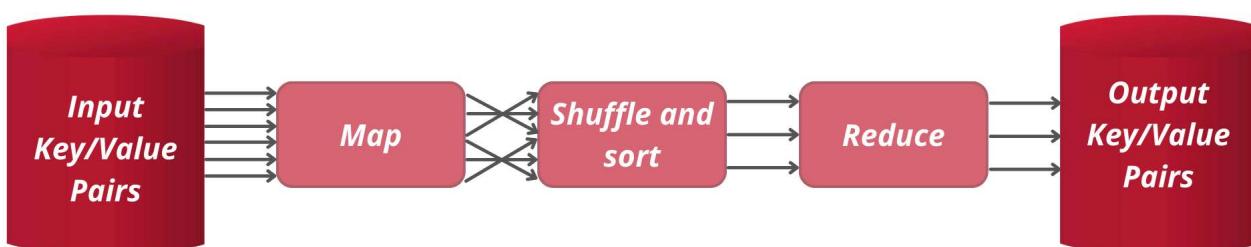


### DICA

A ideia do MapReduce é dividir um problema (como a contagem de formas geométricas do exemplo acima) em problemas menores (*map*). No final, combinamos estes resultados dos problemas menores em um resultado final (*reduce*).

Entre o *map* e o *reduce* também temos alguns passos intermediários que podem ficar ocultos a maioria das vezes, como as funções de embaralhamento (*shuffle*) e classificação (*sort*), conforme você pode ver na figura a seguir.

Figura 2. Fluxo de dados do MapReduce



As etapas de embaralhamento (*shuffle*) e classificação (*sort*) agrupam e organizam os pares chave-valor intermediários gerados pelo mapeamento, preparando-os para a redução. Em um cluster de computadores, isso significa que os dados são transferidos de forma que as máquinas responsáveis pelo *reduce* recebam apenas os dados relevantes para as chaves que estão processando. Na prática, isso significa em uma transmissão de dados eficiente.



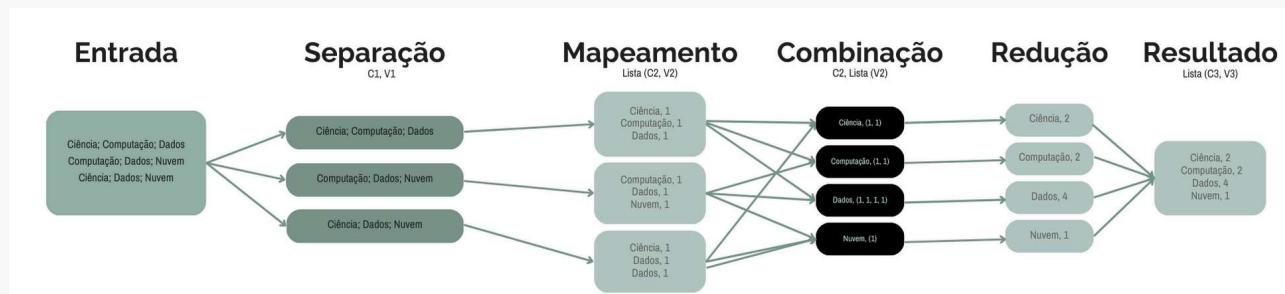
### SAIBA MAIS

No caso do Hadoop, o tratamento das operações de embaralhamento e classificação é feito automaticamente, otimizando a transferência de dados. Para que você tenha mais clareza de como é esse processo, há uma [animação](#) dele no site **Systems Deployment** que vale a pena conferir!

## Expressões Regulares (RegEx)

Um dos jeitos de entendermos como o MapReduce pode ser utilizado em análise e processamento de dados é o da **contagem de palavras**. A ideia consiste em encontrar as palavras mais comuns em algum texto. Veja um exemplo disso na imagem a seguir. Veja como um conteúdo de texto na primeira parte (*entrada*) é separado, dividido e depois concatenado pelo processo do MapReduce.

Figura 3. MapReduce para contagem de palavras



Vamos testar um exemplo prático deste algoritmo de contagem de palavras, mas para isso precisaremos dividir as palavras. Na figura anterior, dividimos um texto em palavras na fase de *separação*. Esta divisão pode ocorrer ao encontrarmos caracteres como espaços, hífens, vírgulas e pontos. Na realidade, existe uma forma mais inteligente de resolvemos isso: com *regex*, ou expressões regulares.

O regex funciona como uma fórmula mágica para encontrar padrões em textos. Imagine que você tem um texto enorme e precisa encontrar todas as placas de veículos dentro do texto. São várias as combinações, certo? O regex permite que façamos isso de um jeito bem rápido.

## Preparando os testes: tokenização e expressões regulares

Nada mais justo do que aprendermos regex em uma videoaula voltada ao MapReduce. Agora, vamos criar um algoritmo para demonstrar um problema de contagem de palavras usando Python e regex. Vamos lá?

## Big data analytics e escalabilidade

*Big data analytics* é um termo relacionado ao processo usado para extrair *insights* importantes, como padrões ocultos, correlações desconhecidas, tendências de mercado e preferências do cliente. Esse processo funciona a partir da análise de grandes volumes de dados, coletados de diversas fontes (Padilha *et al.*, 2022). Quer dizer, nem tudo precisa de uma inteligência artificial, e nem tudo precisa de big data.

Para a escolha de tecnologias, algoritmos e estruturas a ser utilizados para a análise, precisamos entender qual é o nosso desafio. Alguns exemplos válidos para usarmos *big data analytics* incluem (Padilha *et al.*, 2022):

---

### **Prever algo**

Se, por exemplo, uma transação é fraude ou não, se vai chover em um dia específico, se um tumor é benigno ou maligno, etc.

---

### **Encontrar padrões nos dados**

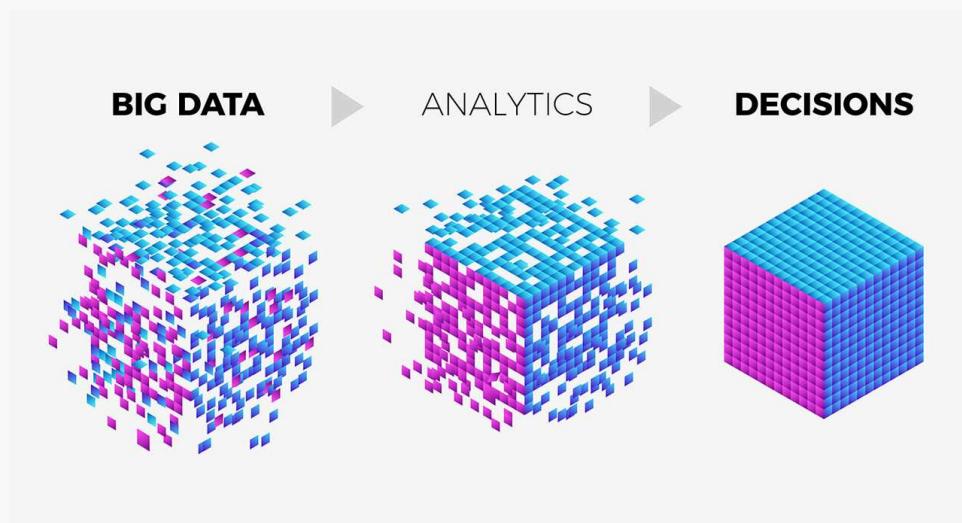
Descobrir as páginas mais visitadas em determinado *site* ou as pessoas mais pesquisadas de uma revista, por exemplo.

---

### **Encontrar relações entre dados**

Pesquisar artigos e

notícias semelhantes, pacientes compatíveis em um sistema de registro eletrônico, correlação entre itens de notícias e preços de ações, dentre outros.



©Alex/Adobe Stock

As aplicações de *big data analytics* não só usam os dados dos sistemas internos das empresas, mas também investigam fontes externas para encontrar informações que sejam relevantes para a empresa. Esse processo é bem sistemático e ajuda muito na hora de tomar decisões importantes.

Para desenvolver essas aplicações, é fundamental entender bem o negócio, especialmente o que se quer descobrir com a análise dos dados. Também é essencial compreender os próprios dados: como estão organizados, quais informações contêm e em que contexto foram gerados. Como o processamento de *big data* é um desafio gigantesco, surgiram ambientes e tecnologias específicas, como o Hadoop, o MapReduce e os bancos de dados NoSQL.

## Comparando a performance com PySpark e Pandas

Será que somos realmente obrigados a usar o PySpark? Não seria mais fácil usar uma biblioteca mais popular, como o Pandas? Bom, o que acha de testarmos uma implementação do PySpark e comparar a sua performance com o Pandas?

## Conclusão

Nesta unidade nos aprofundamos com o MapReduce. Ele é relevante para as aplicações de *big data analytics* para a análise e processamento de massa de dados. O MapReduce realiza o mapeamento dos dados em pares chave-valor e a entrega para uma redução que sumariza os dados mapeados. Essa ferramenta permite que o processamento seja feito de forma distribuída e paralela, o que traz ganhos de velocidade necessários para o *big data*. Por meio de um exemplo de código em Python, observamos como o MapReduce pode auxiliar na análise de dados, sendo um apoio para o processo de tomada de decisão.

## Referências

BENGFORT, B.; KIM, J. **Data analytics with Hadoop**: an introduction for data scientists. Sebastopol: O'Reilly Media, 2016.

GRUS, J. **Data science do zero**: noções fundamentais com Python. 2. ed. Rio de Janeiro: Alta Books, 2021.

MRJOB. **Mrjob v0.7.4 documentation.** Disponível em:

<https://mrjob.readthedocs.io/en/latest/guides/why-mrjob.html#why-use-mrjob-instead-of-x>. Acesso em: 1 ago. 2023.

PADILHA, J. *et al.* **Analytics para big data.** Porto Alegre: SAGAH, 2022.

PEREIRA, M. A. *et al.* **Framework de big data.** Porto Alegre: SAGAH, 2019.

RADTKA, Z.; MINER, D. **Hadoop with Python.** Sebastopol: O'Reilly Media, 2016.

SCHNEIDER, R.D. **Hadoop for dummies.** Mississauga: John Wiley & Sons, 2012.



© PUCPR - Todos os direitos reservados.