

---

# Hadoop

**Fernando Anselmo**

<http://fernandoanselmo.orgfree.com/wordpress/>

---

Versão 1.3 em 6 de julho de 2020

## Resumo

**H**adoop[1] é o principal framework usado para processar e gerenciar grandes quantidades de dados. Qualquer pessoa que trabalhe com programação ou ciência de dados deve se familiarizar com a plataforma. Hadoop é uma estrutura que permite o processamento distribuído de grandes conjuntos de dados em clusters de computadores usando modelos de programação simples. Projetado para escalar de servidores únicos para milhares de máquinas, cada uma oferecendo computação e armazenamento local. Em vez de confiar no hardware para fornecer alta disponibilidade, a biblioteca em si é projetada para detectar e lidar com falhas na camada do aplicativo, entregando um serviço altamente disponível em um cluster de computadores, cada um dos quais pode estar sujeito a falhas.

## 1 Como surgiu o Hadoop?

Nos últimos anos o termo Big Data vem se tornando um assunto cada vez mais discutido em reuniões de planejamento estratégico em empresas de todos os portes. Hadoop é uma plataforma de software de código aberto para o armazenamento distribuído e processamento distribuído de grandes conjuntos de dados em clusters de computadores construídos a partir de hardware a um custo acessível (*commodity*). serviços Hadoop fornecem para armazenamento de dados, processamento de dados, acesso a dados, governança de dados, segurança e operações.



Figura 1: Logo do Hadoop

A gênese do Hadoop veio do papel **Google File System**, que foi publicado em Outubro de 2003. Este trabalho deu origem a outro trabalho de pesquisa do Google – **MapReduce: simplificado Processamento de Dados em grandes aglomerados**. Desenvolvimento começou no projeto Apache Nutch, mas foi transferido para o novo subprojeto Hadoop em janeiro de 2006. A primeira

commiter adicionado ao projeto Hadoop foi Owen O'Malley<sup>1</sup> em março de 2006. Hadoop 0.1.0 foi lançado em abril de 2006 e continua a ser evoluído por muitos contribuintes para o projeto Apache Hadoop. Curiosidade: O nome Hadoop veio do nome do elefante de brinquedo do fundador.

Algumas das organizações razões usar Hadoop é a sua capacidade de armazenar, gerenciar e analisar grandes quantidades de dados estruturados ou não estruturados de forma rápida, confiável, flexível e de baixo custo:

- **Escalabilidade e desempenho** – tratamento de dados distribuídos em um local para cada nó em um cluster Hadoop permite armazenar, gerenciar, processar e analisar dados em escala petabyte.
- **Confiabilidade** – clusters de computação de grande porte são propensos a falhas de nós individuais no cluster. Hadoop é fundamentalmente resistente, quando um nó falha de processamento é redirecionado para os nós restantes no cluster e os dados são automaticamente re-replicado em preparação para falhas de nó futuras.
- **Flexibilidade** – ao contrário de sistemas de gerenciamento de banco de dados relacionais tradicionais, no Hadoop não existem esquemas estruturados criados antes de armazenar dados. Pode-se armazenar dados em qualquer formato, incluindo formatos semi-estruturados ou não estruturados, e em seguida, analisar e aplicar esquema para os dados quando ler.
- **Baixo custo** – ao contrário de software proprietário, o Hadoop é open source e é executado em hardware de baixo custo.

## 2 HDFS e MapReduce

”Hadoop é composto de dois componentes principais: um sistema distribuído de arquivos conhecido como HDFS e um framework distribuído de processamento chamado MapReduce” (Hadoop for Dummies). Na verdade no ecossistema do Hadoop são 3 os componentes principais:

- **Hadoop Common (core libraries)** - São as bibliotecas básicas do sistema.
- **HDFS** (Hadoop Distributed File Systems) - Sistema de Arquivos.
- **Hadoop MapReduce** - Modelo de Programação.

### 2.1 HDFS

*Hadoop Distributed File System* fornece armazenamento de arquivos escalável e tolerância a falhas, possui um custo eficiente para um grande conjunto de dados. Foi projetado para abranger clusters de servidores de baixo custo. Distribui o armazenamento através de muitos servidores permitindo que este recurso cresça linearmente.

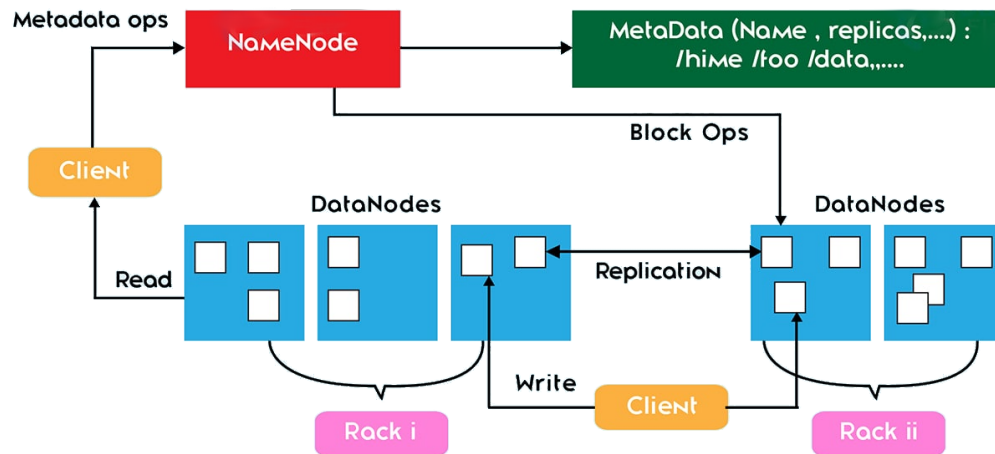
Este sistema distribuído de arquivos do Hadoop que nasceu a partir da ideia do GFS, e possui as seguintes características:

- Apenas lida com arquivos não sendo um banco de dados.

---

<sup>1</sup>Em 2011, Rob Bearden firmou parceria com a Yahoo! para fundar a Hortonworks com 24 engenheiros da equipe original Hadoop, dentre eles os fundadores Alan Gates, Arun Murthy, Devaraj Das, Mahadev Konar, Owen O'Malley, Sanjay Radia e Suresh Srinivas.

- Sistema Escalável (Volume, Velocidade, Variedade)
- Organização de arquivos hierárquica
- Leitura intensiva
- Altamente otimizado



**Figura 2:** *Arquitetura do HDFS*

O sistema tem por base os seguintes princípios:

1. Para uma escalabilidade eficiente não trata da coordenação e comunicação de outros componentes.
2. Um nó não sabe nada sobre outros nós, que dados possuem ou tarefas estão executando.
3. A tarefa de organização fica a cargo de um servidor master chamado de **Name Node**.
4. Para salvar o arquivo divide-o em blocos de tipicamente 64 ou 128 Mb
5. Os blocos são replicados em cada nó (normalmente 3 cópias)

## 2.2 MapReduce

”MapReduce é um modelo de programação para processamento de dados.”(Hadoop - The Definitive Guide). MapReduce é um framework para escrever aplicações paralelas que processam grandes quantidades de dados estruturados e não estruturados armazenados no HDFS. MapReduce tira vantagem da localidade de dados, ao processá-los perto do local onde é armazenado em cada nó no cluster, a fim de reduzir a distância do que deve ser transmitido.

É uma técnica criada para ajudar no processamento paralelo:

- Considerado um novo paradigma da programação.
- Utiliza o HDFS para entrada e saída de dados
- Usa a ideia de MAPA - Chave + Valor

Como funciona?

1. Um processo que é disparado é uma tarefa que o hadoop deve executar chamada de "Map Reduce Job".
2. Transforma dados maiores em menores, agrupando-os, sintetizando-os, somando-os e transformando-os em um segundo conjunto de dados.
3. O job executa programas MapReduce construídos com poucas linhas de código em Java, Python, C++ entre outras.

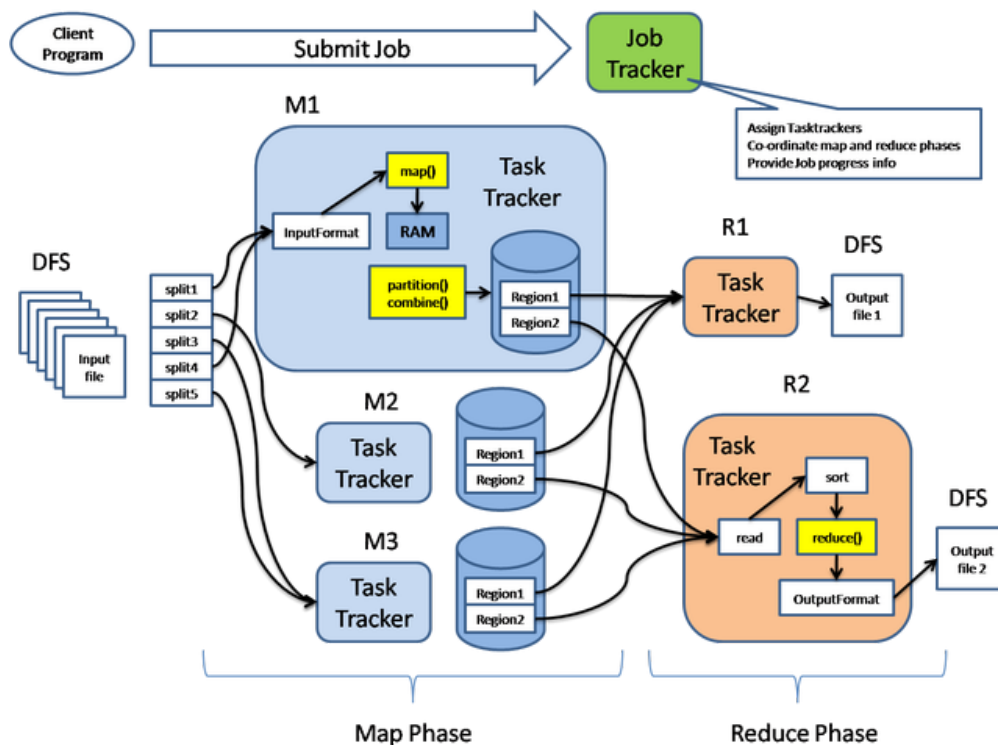


Figura 3: Arquitetura do MapReduce

### 3 Outros produtos do Ecosistema

O Hadoop conta ainda com os seguintes produtos no seu ecossistema para acrescentar funcionalidades complementares e obtermos uma camada de abstração a nível mais alto:

**Hive e Drill** Data warehouse para consultas SQL, que possui uma camada de abstração em linguagem Hive Query Language (HiveQL), é executado nos bastidores e nasceu nos laboratórios do Facebook.

**Mahout e Spark MLlib** Serviços de Machine Learning.

**Pig** Plataforma para análise de grande conjuntos de dados com linguagem de alto nível.

**HBase** Banco de dados padrão NoSQL.

**Spark** Processamento de dados em memória.

**Kafka e Storm** Processamento de streaming.

**Solr e Lucene** Utilizados para pesquisa e indexação.

**Oozie** Workflow para gerenciamento de jobs (Job Scheduling).

**Zookeeper** Gerenciamento de Cluster.

**Ambari** Provisão, monitoramento e manutenção do Cluster.

**Yarn** Node Manager, um negociador de recursos.

**Flume** Serviço de Ingestão de Dados.

**Sqoop** Realiza a importação e exportação para bancos estruturados.

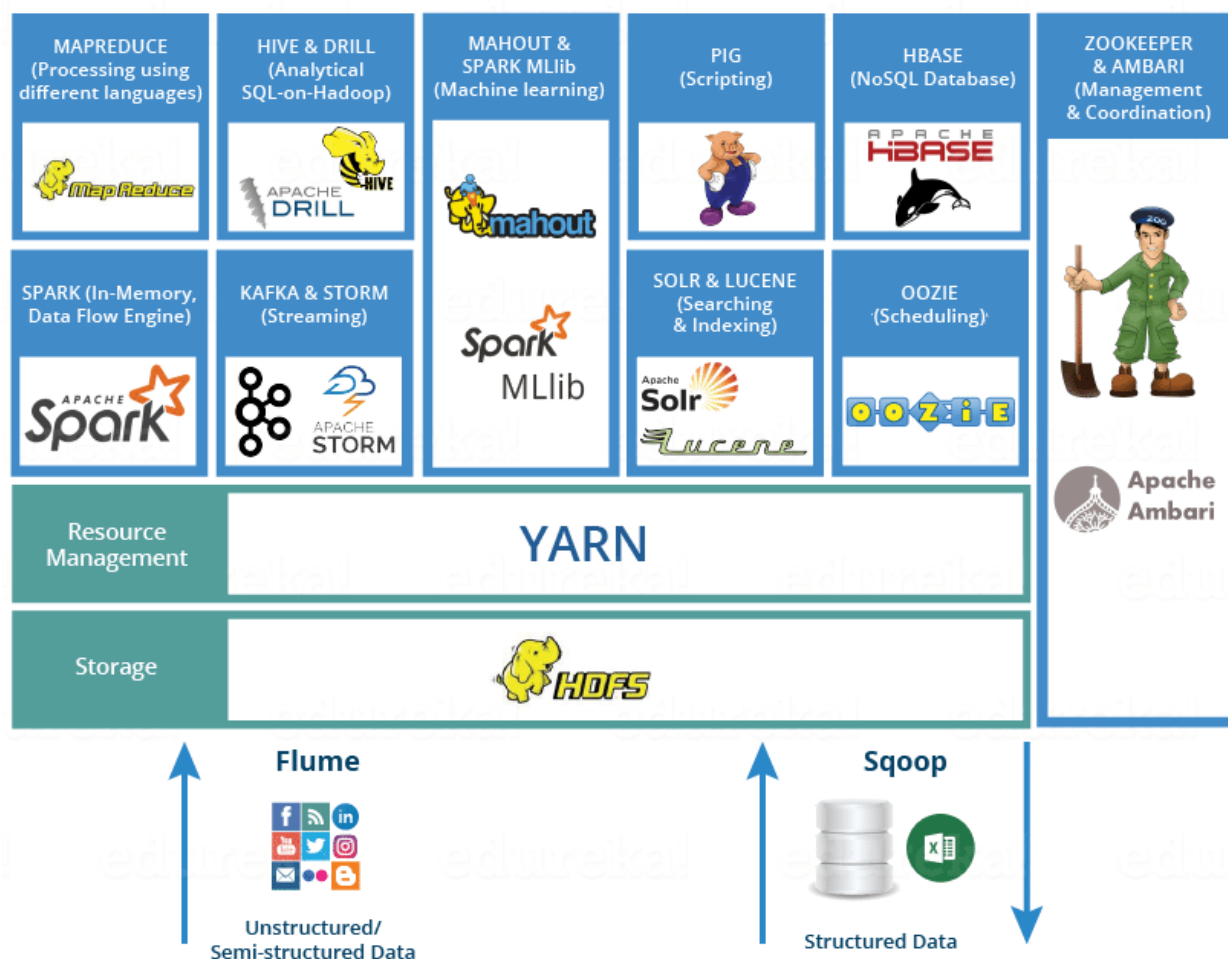


Figura 4: *Ecosistema completo do Hadoop*

## 4 Hadoop no Docker

O modo mais simples de se conseguir trabalhar com o Hadoop é utilizando o Docker, para baixar a imagem do Hadoop:

```
$ docker pull sequenceiq/hadoop-docker:2.7.1
```

E para criar e executar a primeira vez o contêiner (a pasta que este comando for executado será associada a uma pasta interna chamada **/home/hadoop**):

```
$ docker run -it --name hadoop -v $(pwd):/home/tsthadoop  
sequenceiq/hadoop-docker:2.7.1 /etc/bootstrap.sh -bash
```

Uma vez interrompido o contêiner:

```
$ docker stop hadoop
```

Podemos executá-lo novamente com os seguintes comandos:

```
$ docker start hadoop
$ docker exec -it hadoop /etc/bootstrap.sh -bash
```

## 4.1 No bash do Hadoop

A primeira vez é necessário configurar a variável de ambiente e um caminho para a pasta de saída, para tanto, usamos os seguintes comandos:

```
bash-4.1# cd
bash-4.1# vi .bashrc
```

E inserimos [i] as seguintes linhas:

```
1 export HADOOP_PREFIX=/usr/local/hadoop
2 export PATH=$PATH:$HADOOP_PREFIX/bin
3 cd /home/tsthadoop/
```

Gravar [esc] [:w] e sair [:q] do Editor VI. E podemos sair do bash com o comando:

```
bash-4.1# exit
```

## 4.2 Testando o ambiente

Primeiramente, devemos ter ciência do que o Hadoop consome verificando a memória:

```
$ free -m
```

Em seguida verificar seu endereço padrão:

```
$ docker inspect hadoop | grep IP
```

E veremos algo como "IPAddress": "172.17.0.2", sendo assim, no navegador podemos testar os seguintes endereços:

- HDFS: <http://172.17.0.2:50070>
- Cluster: <http://172.17.0.2:8088>
- Nodes: <http://172.17.0.2:8042>
- Status: <http://172.17.0.2:50090>

Além desses ainda temos o endereço de acesso ao HDFS: [hdfs://172.17.0.2:9000](http://172.17.0.2:9000).

## 4.3 Arquivos de Configuração

Estes são os arquivos de configuração do Hadoop:

- **hadoop-env.sh** - Variáveis de configuração que são usadas para executar os scripts.

- **core-site.xml** - Definições de configuração para o Hadoop Core, como configurações de E/S que são comuns ao HDFS e MapReduce.
- **hdfs-site.xml** - Definições de configuração para o HDFS daemons: nome do nó, nome do nó secundário e outros nós de dados.
- **mapred-site.xml** - Definições de configuração para o MapReduce daemons, *jobtracker* e *tasktrackers*.
- **masters** - Lista de máquinas (uma por linha) para cada execução secundária do *NameNode*.
- **slaves tasktracker** - Lista de máquinas (uma por linha) para cada execução dos nós de dados e *tasktracker*.
- **hadoop-metrics.properties** - Propriedades para controlar quais métricas são publicadas no Hadoop.
- **log4j.properties** - Propriedades para o registros (logfiles) do sistema, registro de auditoria do *NameNode*, e os registros de tarefas para os processos do *tasktracker*.

Nesta imagem os arquivos se encontram na pasta:  
 /usr/local/hadoop-2.7.0/etc/hadoop

## 4.4 Comandos básicos no bash

Se o Hadoop está rodando corretamente:  
 bash-4.1# jps

Informações do config:  
 bash-4.1# ifconfig

Relatório informativo do HDFS:  
 bash-4.1# hdfs dfsadmin -report

Se quiser ir para a pasta HOME do Hadoop:  
 bash-4.1# cd \$HADOOP\_PREFIX

Rodar o mapreduce  
 bash-4.1# hadoop jar  
 share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar grep input  
 output 'dfs[a-z.]+'

Verificar o diretório de saída:  
 bash-4.1# hdfs dfs -cat output/\*

Listar todos os arquivos do diretório de entrada:  
 bash-4.1# hdfs dfs -ls input/\*

Remover todos arquivos do diretório de saída:  
 bash-4.1# hdfs dfs -rm rf output/\*

Listagem dos arquivos  
 bash-4.1# hdfs dfs -ls /

E podemos sair do bash com o comando:

```
bash-4.1# exit
```

## 5 Exemplo Completo

Acessamos os dados disponibilizados pelo Portal da Transparência[2] sobre o Bolsa Família (acessar a opção "Benefícios ao Cidadão- "Bolsa Família - Pagamentos") baixar qualquer mês/ano desejado e temos um arquivo CSV para trabalhar (colocamos o arquivo em uma pasta a partir do \$PWD - usado na associação do Docker - /Aplicativo/hadoop-model/bolsa).

### 5.1 Fora do Hadoop

O primeiro tratamento que fazemos é convertê-lo para um formato UTF-8 com o comando:

```
$ iconv -f ISO-8859-1 -t UTF-8 [arqOriginal].csv > 2018_Pagamento.utf8.csv
```

Observamos que este arquivo é gigante para realizarmos um teste (demandará muito tempo de processamento), então usaremos o seguinte programa para gerar um arquivo AMOSTRA de 20.000 elementos (com base na técnica de Amostragem Sistemática). Listagem para o **amostra.py** (em linguagem Python):

```
1 #!/usr/bin/env python
2 import sys
3 import os
4 from random import randint
5
6 # Selecionar um numero randomico para saltar
7 salto = randint(1,1000)
8 print('Salto',salto)
9
10 saida = "201808_BF_AmostraA.csv"
11
12 lin = 0
13 passo = 0
14 with open(saida, 'w+') as file:
15     for line in sys.stdin:
16         passo += 1
17         # so grava de tantos em tantos registros
18         if passo == salto:
19             passo = 0
20             file.write(line)
21             lin += 1
22             # Quando passar de 20.000 registros gravados
23             if (lin > 20000):
24                 break
25
26 print(lin, 'gravadas')
27 file.close()
```

Executamos da seguinte forma:

```
$ cat 2018_Pagamento.utf8.csv | ./mapper.py
```



Utilizaremos este arquivo para o exemplo, mas se desejar pode utilizar o arquivo completo com a realização das devidas alterações no nome deste. Para processar o MapReduce precisamos criar dois arquivos. Listagem para o **mapper.py**:

```
1 #!/usr/bin/env python
2 import sys
3
4 for line in sys.stdin:
5     line = line.strip()
6     fields = line.split(";")
7     estado = fields[2]
8     estado = estado[1:-1]
9     valor = fields[7]
10    valor = valor[1:-1]
11    valor = str(valor.replace(",", "."))
12    print("%s\t%s" % (estado, valor))
```

Podemos testar este com a seguinte linha de código:

```
$ cat 201808_BF_Amostra.csv | ./mapper.py
```

Observe que o resultado desta primeira fase fornece como saída o par "chave + valor" (estado e o valor pago) para cada linha encontrada no arquivo. E a listagem para o **reducer.py**:

```
1 #!/usr/bin/env python
2 import sys
3
4 previous_value = ""
5 sum = 0.0
6 for line in sys.stdin:
7     line = line.strip()
8     value, count = line.split("\t")
9     count = float(count)
10    if value == previous_value:
11        sum += count
12    else:
13        print("%s\t%s" % (previous_value, sum))
14        previous_value = value
15        sum = count
16 print("%s\t%s" % (previous_value, sum))
```

Testamos este programa com a seguinte linha de código:

```
$ cat 201808_BF_Amostra.csv | ./mapper.py | sort | ./reducer.py
```

Como não estamos executando no Hadoop precisamos usar o comando "sort" para ordenar os valores, e a saída desta segunda fase será os valores pagos totais agrupados do estado.

## 5.2 No Hadoop

Iniciar o contêiner:

```
$ docker start hadoop
```

Executar o bash do Hadoop:

```
$ docker exec -it hadoop /etc/bootstrap.sh -bash
```

Verificar a pasta na qual estamos:

```
bash-4.1# pwd
```

E como resposta devemos obter: **/home/tsthadoop** que é a pasta que nos liga ao sistema externo. Seleccionar a pasta aonde se localizam os arquivos:

```
bash-4.1# cd Aplicativos/hadoop-model/bolsa
```

Verificar a existência dos arquivos com o comando:

```
bash-4.1# ls
```

Adicionar os arquivos no HDFS:

```
bash-4.1# hadoop fs -put mapper.py
```

```
bash-4.1# hadoop fs -put reducer.py
```

```
bash-4.1# hadoop fs -put 201808_BF_Amostra.csv
```

Verificar os arquivos no HDFS:

```
bash-4.1# hdfs dfs -ls
```

Executar o MapReduce:

```
bash-4.1# hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.0.jar  
-mapper "python mapper.py" -reducer "python reducer.py" -input 201808_BF_Amostra.csv  
-output OutputDir -file mapper.py -file reducer.py -file 201808_BF_Amostra.csv
```

Verificar o diretório de saída:

```
bash-4.1# hadoop fs -ls OutputDir
```

Verificar a informação de saída:

```
bash-4.1# hadoop fs -cat OutputDir/part-00000 | head
```

Baixar o arquivo para a pasta local:

```
bash-4.1# hadoop fs -getmerge OutputDir/ my-local-file.txt
```

Para remover os arquivos do HDFS:

```
bash-4.1# hdfs dfs -rm mapper1.py
```

```
bash-4.1# hdfs dfs -rm reducer1.py
```

```
bash-4.1# hdfs dfs -rm 2018.Pagamento.utf8.csv
```

```
bash-4.1# hdfs dfs -rm -r OutputDir
```

## 6 Conclusão

”Big Data” é um termo que ganha cada vez mais espaço no vocabulário das empresas de TI e entre administradores de data centers. Afinal de contas, o volume de dados gerado hoje, graças à facilidade de acesso à internet a partir de quase qualquer lugar, é maior do que se podia imaginar há alguns anos atrás.

O que o Hadoop faz é organizar melhor esse volume exaustivo de dados para encontrar informações específicas sobre eles de maneira mais rápida e eficiente. Trata-se de conjuntos de clusters que trabalham com um hardware barato para executar um grande número de tarefas simultâneas sem comprometer a infraestrutura de processamento da rede.

Sou um entusiasta do mundo **Open Source** e novas tecnologias. Qual a diferença entre Livre e Open Source? Livre significa que esta apostila é gratuita e pode ser compartilhada a vontade.

Open Source além de livre todos os arquivos que permitem a geração desta (chamados de arquivos fontes) devem ser disponibilizados para que qualquer pessoa possa modificar ao seu prazer, gerar novas, complementar ou fazer o que quiser. Os fontes da apostila (que foi produzida com o LaTeX) está disponibilizado no GitHub [5]. Veja ainda outros artigos que publico sobre tecnologia através do meu Blog Oficial [3].

## Referências

- [1] Página Oficial do Apache Hadoop  
<http://hadoop.apache.org/>
- [2] Dados do Portal da Transparência  
<http://www.portaldatransparencia.gov.br/download-de-dados>
- [3] Fernando Anselmo - Blog Oficial de Tecnologia  
<http://www.fernandoanselmo.blogspot.com.br/>
- [4] Encontre essa e outras publicações em  
<https://cetrex.academia.edu/FernandoAnselmo>
- [5] Repositório para os fontes da apostila  
<https://github.com/fernandoans/publicacoes>