Hadoop - MapReduce

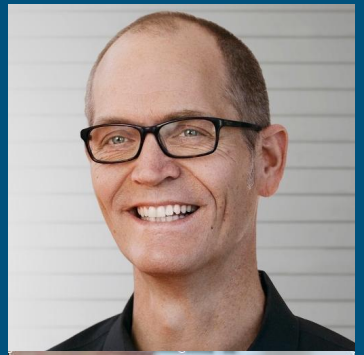
João Paulo Siqueira Lins

Roteiro

- História
  - Motivação
  - Objetivo
- Módulos do Hadoop
  - Principais
  - Adicionais
- Exemplos

História - A motivação

- Doug Cutting e Mike Cafarella - Criadores do Hadoop
- Lucene
  - Biblioteca de Recuperação de Informação
  - Integrado ao Apache Software Foundation em 2001
- Nutch
  - Web crawler, usava o Lucene para obter informação das páginas
  - Em 2003, conseguia indexar 100 páginas por segundo, limitado por rodar em 1 máquina.
  - Tentaram aumentar o número de máquinas para 4, sem uma plataforma de gerenciamento de clusters (gerenciamento manual de espaço em disco e de informação entre os nós)



História - NDFS

- Doug e Mike viram que precisavam melhorar o Nutch com uma camada de armazenamento distribuído :
  - Ausência de esquema : estruturas não-predefinidas
  - Uma vez que os dados sejam escritos, eles não podem ser perdidos
  - Lidar com falha de hardware automaticamente
  - Balancear carga entre os nós do cluster
- Depois de meses tentando resolver o problema, eles se surpreendem com a publicação de um artigo da Google sobre o GFS - Google File System.
  - Em 2004, o NDFS - Nutch Distributed File System foi desenvolvido, de acordo com o GFS.
  - O NDFS possibilitaria manipulação concorrente de arquivos e poderia ser implementado com *commodity hardware*.

História - MapReduce

- Após o problema de infraestrutura ser resolvido, Doug e Mike começaram a pensar em algoritmos que fariam bom uso do NDFS.
- No fim de 2004, a Google surpreende com mais um artigo : “MapReduce: Simplified Data Processing on Large Clusters”.
- MapReduce resolvia 3 problemas principais:
    - Paralelização;
    - Distribuição;
    - Falha de componentes;
- No MapReduce, o programa vai até os dados, e não o contrário.
- Em 2005, o MapReduce foi adotado no projeto Nutch.

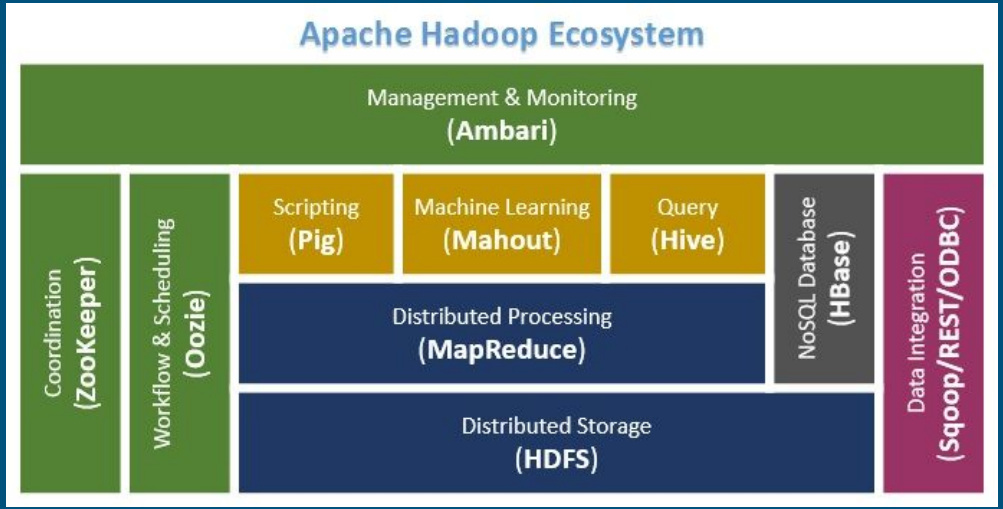
História - Hadoop

- Em fevereiro de 2006, Doug tirou o NDFS do projeto Nutch e criou um novo sub-projeto do Lucene chamado Hadoop, que era composto de :
    - Hadoop Common
    - HDFS - Antigo NDFS
    - MapReduce
- Em 2008, o Hadoop foi promovido a um projeto top-level na Apache Software Foundation, devido à dedicação de sua comunidade.

Hadoop - Módulos Principais

- **Hadoop Common** : Bibliotecas e utilidades que são utilizadas por outros módulos.
- **Hadoop Distributed File System (HDFS)** : Sistema de arquivos que possui como características ser distribuído e escalável.
- **YARN** – (Yet Another Resource Negotiator) : Gerenciador de recursos para os processos que estão rodando no Hadoop.
- **MapReduce** – Framework de processamento paralelo para clusters.

- Apache Hadoop foi desenvolvido em Java.
- Implementação open source baseada nos artigos do Google.

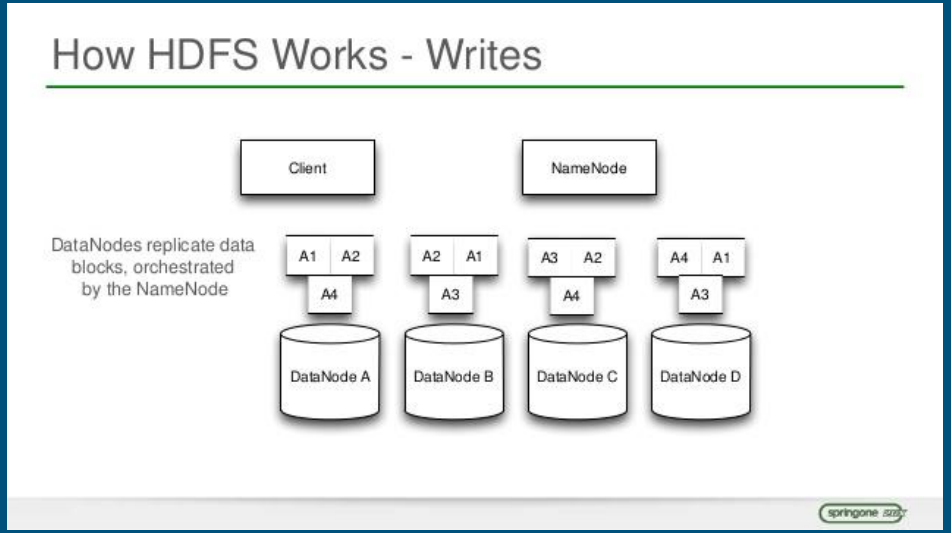


Hadoop - Hadoop Common

- Também chamado de Hadoop Core
- Contém serviços essenciais e processos básicos para a plataforma Hadoop e seus módulos.
- Camada de abstração para os usuários.

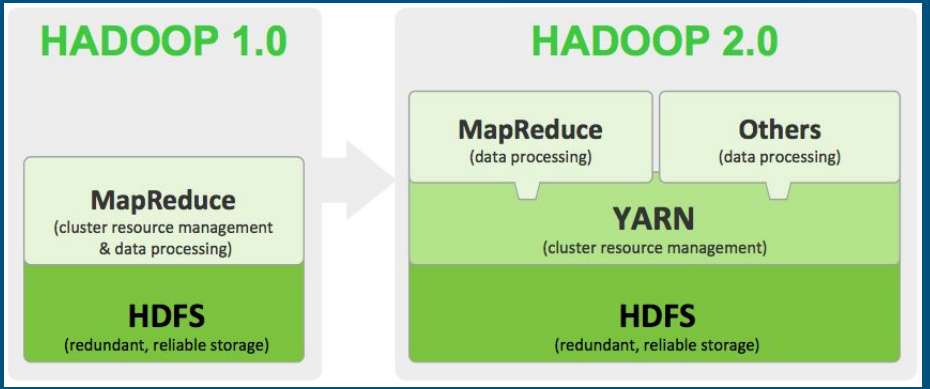
Hadoop - HDFS

- Guarda arquivos em diferentes máquinas, utilizando redundância.
- Valor de replicação padrão : 3
- As máquinas são livres para se comunicarem entre si, caso haja necessidade (falha, balanceamento de carga)
- O cluster do HDFS contém 3 tipos de componentes:
  - Client
  - NameNode
  - DataNode



Hadoop - YARN

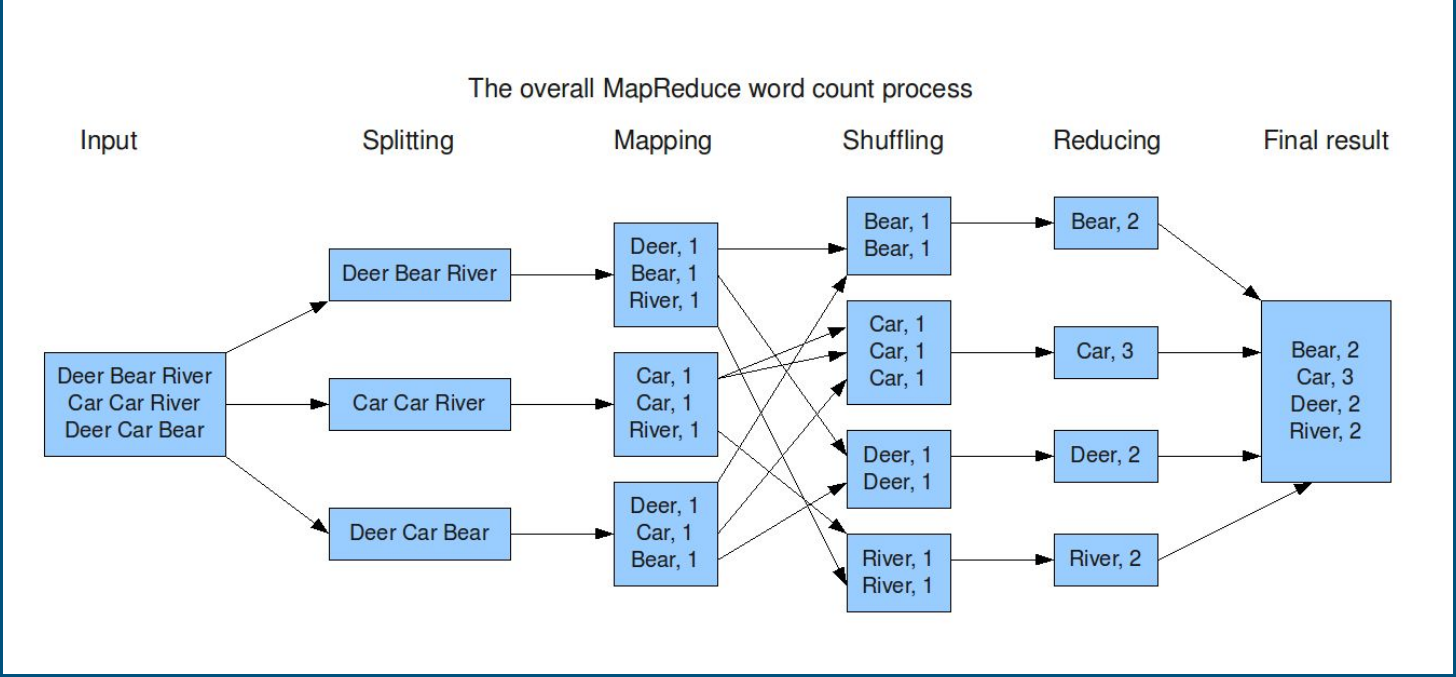
- Com o tempo, foi visto que existia muitas responsabilidades delegadas para o MapReduce, como por exemplo gerenciar execução de *jobs*, processamento de dados, e fazer a interface com os clientes.
- Para corrigir esse problema, o YARN foi desenvolvido.
- Os frameworks de mais alto nível agora não estavam mais limitados pelo MapReduce, e poderiam se integrar ao ecossistema Hadoop de forma mais livre.
- O MapReduce pode focar no que é realmente pra ele ser feito : Processamento em Lotes



Hadoop - MapReduce

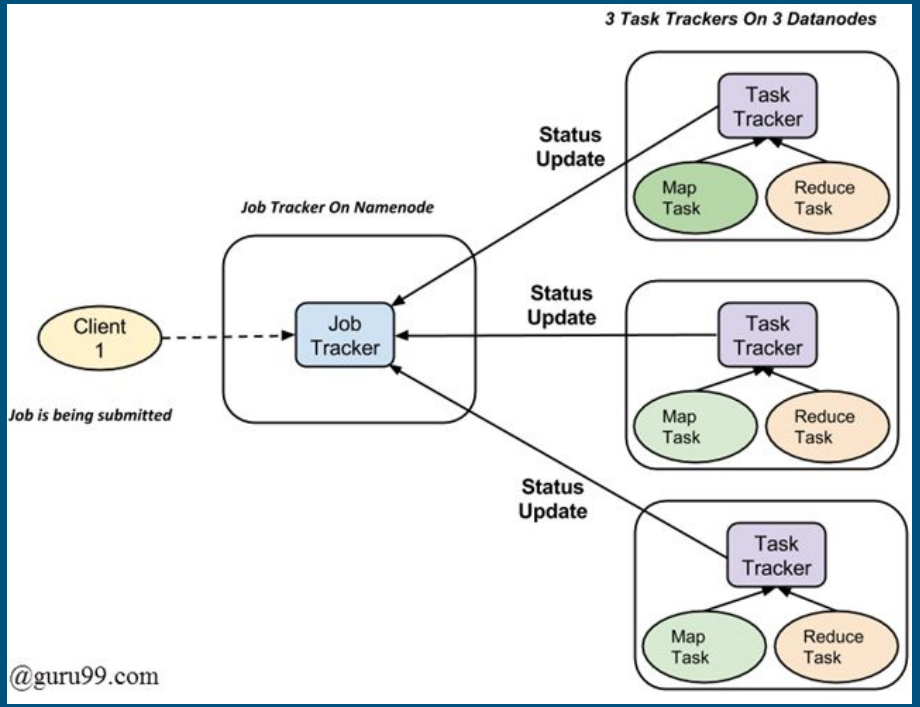
- MapReduce é um framework de processamento paralelo inspirado em programação funcional.
- A ideia principal do MapReduce é dividir e processar tarefas e depois juntar as informações.
- O funcionamento é dividido em vários passos:
  - Input split : A entrada é dividida em várias partes, onde cada parte será consumida por um Map.
  - Map : Age sobre a entrada, criando uma lista de pares chave-valor.
  - Shuffling : Classifica e agrupa a saída dos Maps para servir de entrada para o Reduce
  - Reduce : Processa o que foi dado pela etapa de shuffling e agrega as informações para serem retornadas.

Hadoop - MapReduce



Hadoop - MapReduce

- Hadoop divide o *job* em *tasks*.
  - Map tasks (Splits e Mapping)
  - Reduce tasks (Shuffling e Reducing)
- A execução dessas *tasks* é controlada por dois tipos de entidades.
  - Job Tracker
    - Age como Mestre
    - Envia as tarefas para os Escravos
    - Realoca tarefas em caso de falha
  - Task Tracker
    - Age como Escravo
    - Reporta status para o Mestre
- Cada *job* tem um Job Tracker e vários Task Trackers



Hadoop - MapReduce

- As *tasks* são executadas nos nós onde a informação está carregada, o que torna o processamento mais rápido - quem move são os programas, não os dados.
    - Os Job Trackers são responsáveis por checar se o nó que irá processar a informação contém os dados carregados
    - A redundância é importante nesse caso : A chance de haver algum nó livre com a informação é maior

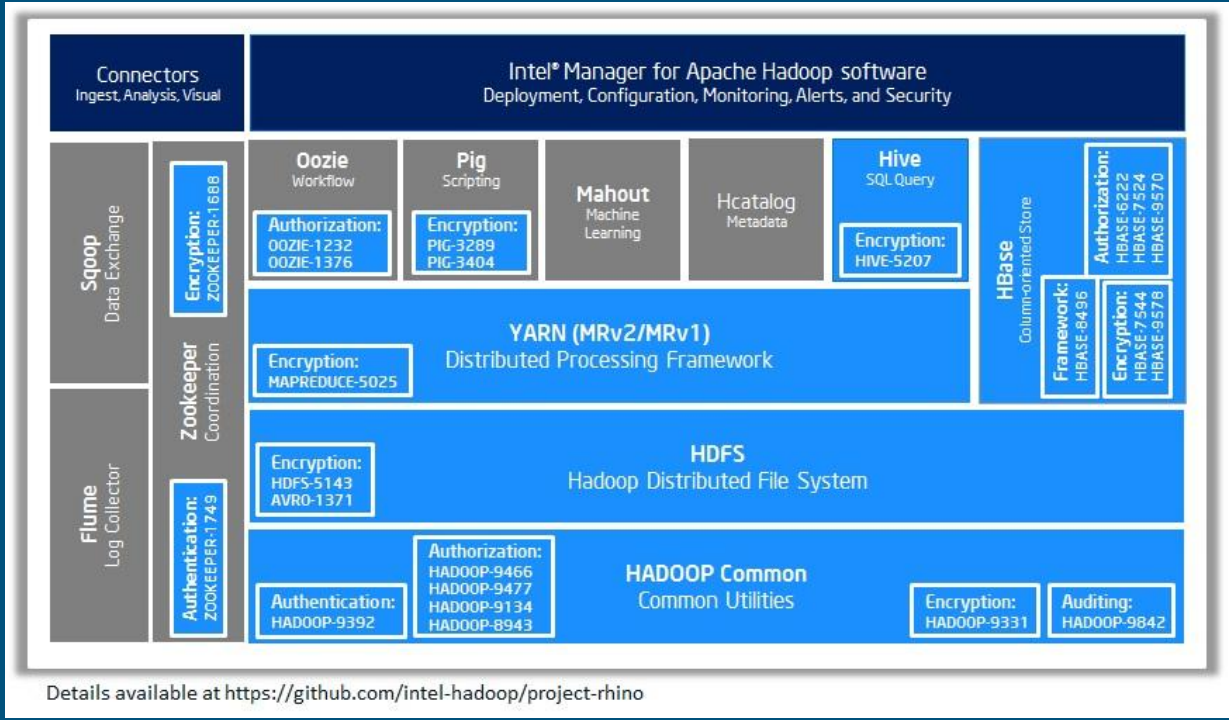
Hadoop - Módulos adicionais

- HBase
  - Oferece funcionalidades de bancos de dados NoSQL para o Hadoop. Alternativa Estruturada
- Hive
  - Permite se fazer operações em data warehouse usando uma linguagem chamada HiveQL, em vez de usar Java(MapReduce). Serve como uma camada de abstração para bancos de dados que integram com o Hadoop.
- Pig
  - Plataforma de alto nível para se programar para o Hadoop, assim como o Hive. A linguagem é a Pig Latin. Usada para realizar ETL por exemplo.
- Mahout
  - Oferece implementações de algoritmos de aprendizagem de máquina para o Hadoop.

Hadoop - Módulos adicionais

- Sqoop
  - Transferir dados de bancos de dados relacionais para o Hadoop
- Zookeeper
  - Gerenciamento de informação e configuração dos clusters.
- Flume
  - Coleta, agrega, e move grandes volumes de dados em stream, como logs, para o HDFS com seguridade a falhas.

Hadoop - Módulos adicionais



Details available at https://github.com/intel-hadoop/project-rhino

MapReduce - Exemplos - Amigos de amigos

- Amigos de Amigos:

  A -> B C D
  B -> A C D E
  C -> A B D E
  D -> A B C E
  E -> B C D

Mapear:

Map(A -> B C D) :
(A B) -> B C D
(A C) -> B C D
(A D) -> B C D

Map(B -> A C D E) :
(A B) -> A C D E
(B C) -> A C D E
(B D) -> A C D E
(B E) -> A C D E

[...]

Agrupar:

(A B) -> (A C D E) (B C D)
(A C) -> (A B D E) (B C D)
(A D) -> (A B C E) (B C D)
(B C) -> (A B D E) (A C D E)
(B D) -> (A B C E) (A C D E)
(B E) -> (A C D E) (B C D)
(C D) -> (A B C E) (A B D E)
(C E) -> (A B D E) (B C D)
(D E) -> (A B C E) (B C D)

MapReduce - Exemplos - Amigos de amigos

Agrupar:

(A B) -> (A C D E) (B C D)
(A C) -> (A B D E) (B C D)
(A D) -> (A B C E) (B C D)
(B C) -> (A B D E) (A C D E)
(B D) -> (A B C E) (A C D E)
(B E) -> (A C D E) (B C D)
(C D) -> (A B C E) (A B D E)
(C E) -> (A B D E) (B C D)
(D E) -> (A B C E) (B C D)

Reduzir:

(A B) -> (C D)
(A C) -> (B D)
(A D) -> (B C)
(B C) -> (A D E)
(B D) -> (A C E)
(B E) -> (C D)
(C D) -> (A B E)
(C E) -> (B D)
(D E) -> (B C)

MapReduce - Exemplos - Word Count

```java
public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

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

    @Override
    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String cleanLine = value.toString().toLowerCase().replaceAll("[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']", " ");
        StringTokenizer itr = new StringTokenizer(cleanLine);
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken().trim());
            context.write(word, one);
        }
    }
}

public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();

    @Override
    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);
    }
}
```

MapReduce - Exemplos - Word Count

```
public static void main(String[] args) throws Exception {

    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: wordcount <in> <out>");
        System.exit(2);
    }

    Job job = Job.getInstance(conf);
    job.setJobName("WordCount");

    job.setJarByClass(WordCount.class);

    job.setMapperClass(TokenizerMapper.class);
    job.setReducerClass(IntSumReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Referências

- https://medium.com/@markobonaci/the-history-of-hadoop-68984a11704#.csldj6x0l
- http://hadooptraininginhyderabad.co.in/motivation-behind-creation-of-hadoop/
- https://en.wikipedia.org/wiki/Apache_Hadoop
- http://saphanatutorial.com/hadoop-cluster-architecture-and-core-components/
- http://www.plottingsuccess.com/hadoop-101-important-terms-explained-0314/
- http://www.guru99.com/introduction-to-mapreduce.html