Big Data

MapReduce

Prof. Leandro Batista de Almeida

leandro@utfpr.edu.br

2016

MapReduce

- Modelo de programação para processamento de dados
  - Simples
- Hadoop pode executar programas MapReduce escritos em varias linguagens
  - Java, Ruby, Python, C++, etc
- Inerentemente paralelo
  - Analise de dados em larga escala disponível a todos

Aplicação de exemplo

- Weather dataset
  - Sensores de tempo coletam dados a cada hora em muitas localidades
    - Por todo o globo
    - Grande volume de dados
    - Dados em formatos variados e possivelmente não estruturados
  - Bom candidato para analise via MapReduce
    - Semi-estruturado
    - Orientado a registros

Aplicação de exemplo

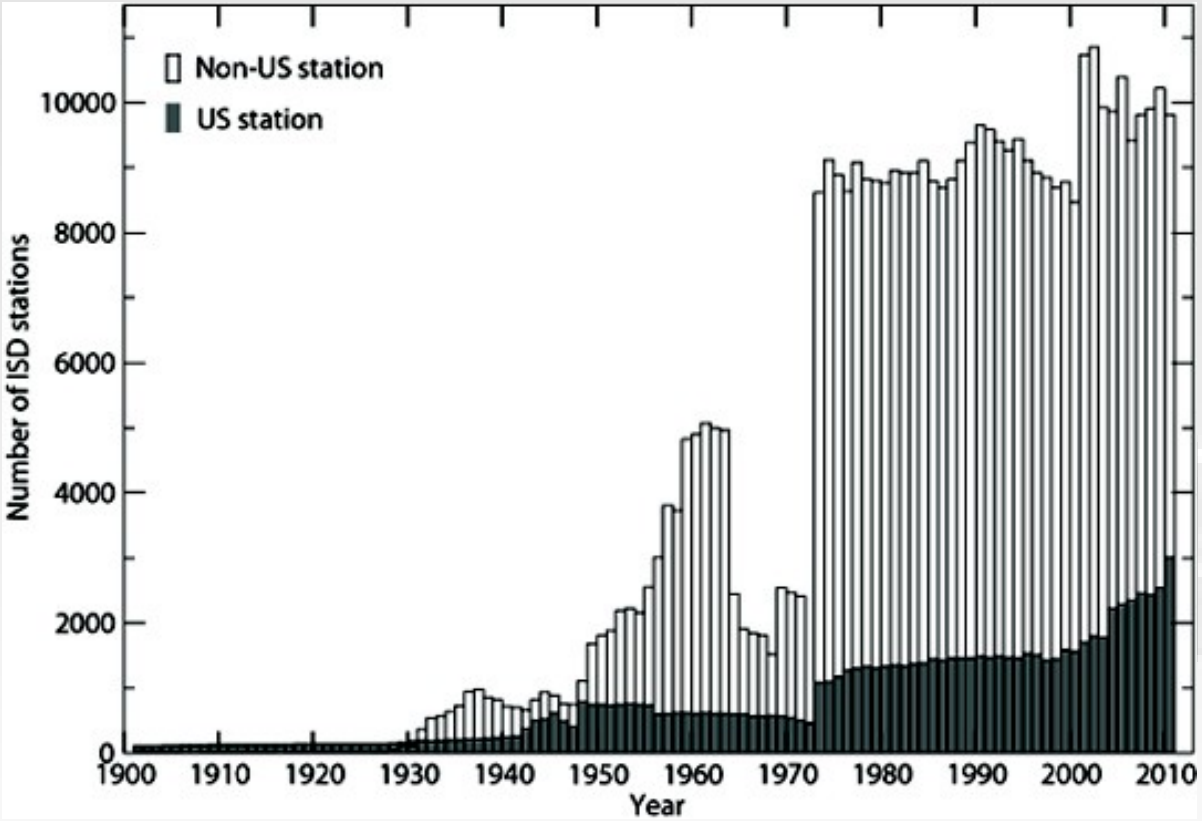
- NOAA
  - National Oceanic and Atmospheric Administration
- NCDC
  - National Climatic Data Center
- Alguns locais
  - Federal Climate Complex, Integrated Surface Data
    - ftp://ftp.ncdc.noaa.gov/pub/data/noaa/
  - Quality Controlled Local Climatological Data
    - http://cdo.ncdc.noaa.gov/qclcd_ascii/

Aplicação de exemplo

- ISH
  - Arquivos para cada estação de rastreamento
    - Formato em ish-format-document.pdf
  - Dados desde 1901
  - Estrutura baseada em posicionamento de caracteres
    - Campos fixos
  - Problemas por dados estarem espalhados em diversos pequenos arquivos
    - Muitas estações de rastreamento por ano
    - Arquivos pequenos diminuem eficiência do HDFS
    - Concatenação de dados pode resolver
      - Entretanto, é operação lenta

Aplicação de exemplo

- ISH



Aplicação de exemplo

- QCLCD
  - Dados compactados a partir de 1996
    - Temperatura, ventos, etc
  - Formato baseado em CSV
    - Fácil recuperação
  - Arquivos de maior tamanho
    - Dispensa concatenação
  - Simples para provas de conceito

Aplicação de exemplo

- Como comparar desempenho?
  - Em geral contra bases de dados conhecidas
    - Bancos relacionais
  - Scripts de sistema operacional podem ser usados também
    - Mais rápidos quando todos os dados são varridos
    - Utilizam ao máximo as otimizações de sistemas de arquivos locais
    - Não possuem retardos devido a estruturas de controle de acesso a dados
    - Entretanto, podem ser mais complexos para desenvolvimento

Aplicação de exemplo

```
#!/usr/bin/env bash

for ano in geral/*

do

  echo -ne `basename $ano .zip`"\t"
  gunzip -c $ano | \
    awk '{ temp = substr($0, 88, 5) + 0;
           q = substr($0, 93, 1);
           if (temp !=9999 && q ~ /[01459]/ && temp >
  max) max = temp }
      END { print max }'

done
```

Aplicação de exemplo

Analisando dados com Hadoop

- Hadoop proporciona acesso simples a paralelismo
  - Desde que o problema seja expresso no paradigma MapReduce
- Duas etapas
  - Map
  - Reduce
- Cada etapa possui estruturas de dados baseadas em chave-valor como entrada e saída de dados
  - Coleções, na API
- Programador fornece duas “funções”, mapper e reducer

Analisando dados com Hadoop

- Entrada de dados são os arquivos do NOAA
  - Cada linha do arquivo é uma entrada
    - Chave pode ser o número da linha
- Mapper
  - Ler todas as linhas e buscar na linha a informação que interessa
    - Fase de preparação de dados
  - Armazenar os dados lidos (ano e temperatura) em uma coleção intermediária
- Reducer
  - Buscar, na coleção intermediária, um ano
  - Ler todas as temperaturas e selecionar a maior
  - Gravar resultados em coleção de resposta

Analisando dados com Hadoop

- Saída de um mapper
  - (1901, 232)
  - (1901, 317)
  - (1901, 223)
  - (1901, 294)
  - (1902, 200)
  - (1902, 225)
  - Etc

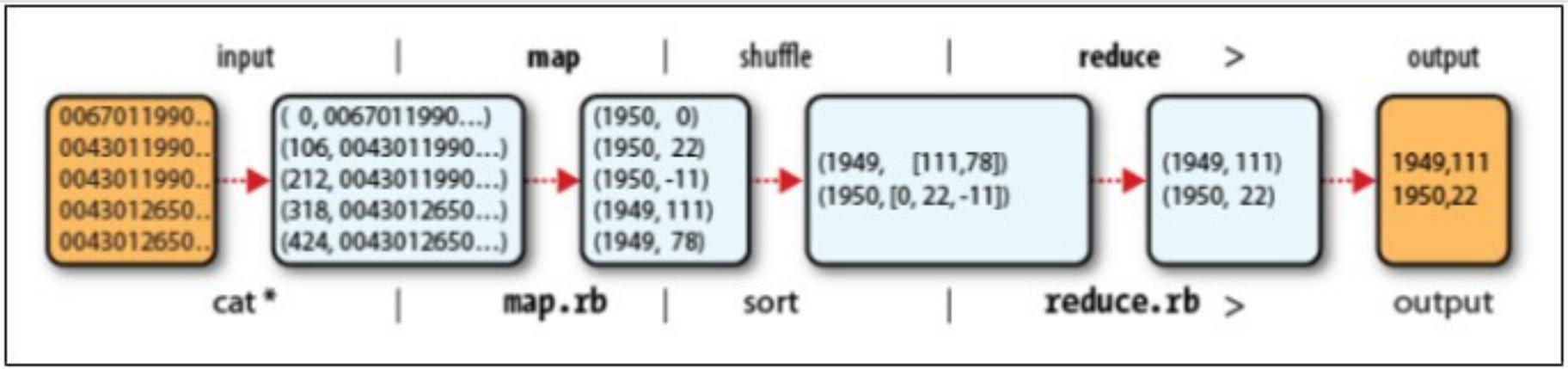
Analisando dados com Hadoop

- Sorting e combiner
  - (1901, [232, 317, 223, 294])
  - (1902, [200, 225, 244])
  - Etc

Analisando dados com Hadoop

- Saída do Reducer
  - (1901, 317)
  - (1902, 244)
  - etc

Analisando dados com Hadoop



Mapper

```
public class MaxTemperatureMapper
extends Mapper<LongWritable, Text, Text, IntWritable> {

private static final int MISSING = 9999;

@Override
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
  String line = value.toString();
  String year = line.substring(15, 19);
  int airTemperature;
  if (line.charAt(87) == '+') { // retirar sinal +
    airTemperature = Integer.parseInt(line.substring(88, 92));
  } else {
    airTemperature = Integer.parseInt(line.substring(87, 92));
  }
  String quality = line.substring(92, 93);
  if (airTemperature != MISSING && quality.matches("[01459]")) {
    context.write(new Text(year), new IntWritable(airTemperature));
  }

}
}
```

Reducer

```
public class MaxTemperatureReducer extends
   Reducer<Text, IntWritable, Text, IntWritable> {

@Override
public void reduce(Text key, Iterable<IntWritable>
   values, Context context) throws IOException,
   InterruptedException {

   int maxValue = Integer.MIN_VALUE;
   for (IntWritable value : values) {
     maxValue = Math.max(maxValue, value.get());
   }
   context.write(key, new IntWritable(maxValue));
}
}
```

JobControl

```
public class MaxTemperature {

public static void main(String[] args) throws Exception {
  if (args.length != 2) {
    System.err.println("Sintaxe: MaxTemperature <input path> <output path>");
    System.exit(-1);
  }

  Job job = new Job();
  job.setJarByClass(MaxTemperature.class);
  job.setJobName("Max temperature");

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

  job.setMapperClass(MaxTemperatureMapper.class);
  job.setReducerClass(MaxTemperatureReducer.class);

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

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

}
}
```

Execução dos programas

- Projeto em NetBeans ou Eclipse
- Acrescentar JARs do Hadoop no classpath
- No caso deste projeto, incluir os caminhos para os arquivos
- Execução local
  - Dentro da IDE
- Execução single node
  - Com parâmetros para HDFS e JAR executado através da ferramenta hadoop

Execução em cluster

- Criando diretórios

```
hdfs dfs -mkdir /data
hdfs dfs -mkdir /data/noaa1
```

- Carregando arquivos

```
hdfs dfs -put 201501hourly.txt /data/noaa1
```

- Executando o job

```
hadoop jar HadoopNoaa.jar /data/noaa1m /data/noaa1m-out1
```

Questões

Prof. Leandro Batista de Almeida
leandro@utfpr.edu.br