MapReduce Example in Hadoop

Mata Kuliah Big Data

Semester Genap 2021/2022

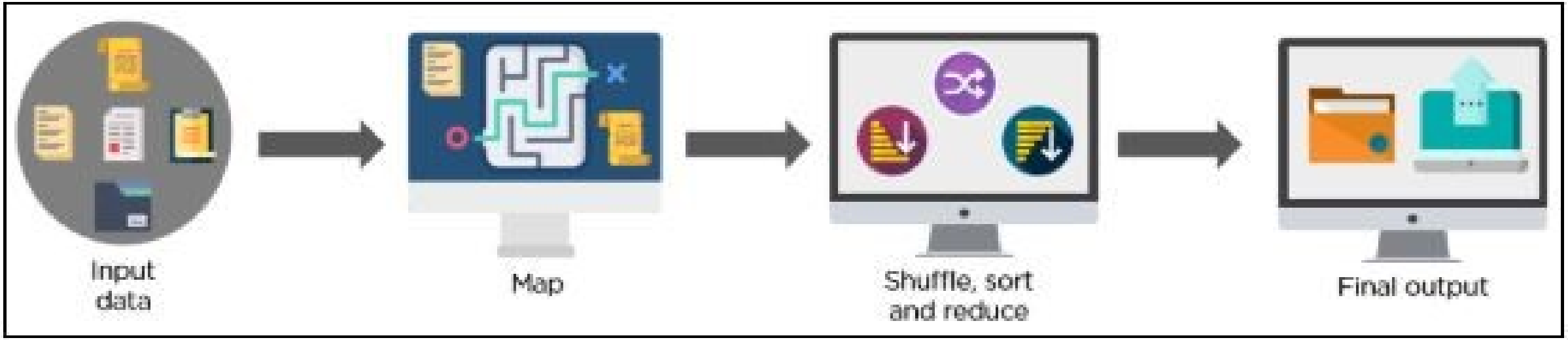
Outline

- Apa itu Apache MapReduce? (Overview)
- Fitur MapReduceBagaimana Algoritma Hadoop MapReduce Bekerja
- MapReduce Example Program
- Kesimpulan
- Praktikum (Jobsheet)

Apa itu Apache MapReduce? (overview)

- Hadoop adalah sebuah tool Big Data yang banyak digunakan untuk menyimpan dan memproses data dalam jumlah besar di beberapa cluster.
- Apache MapReduce adalah salah satu komponen utama pada Hadoop yang memungkinkan pemrosesan data lebih cepat.
- Apache MapReduce adalah mesin pemrosesan pada Hadoop yang memproses dan menghitung sebuah data dengan volume data yang sangat besar.
- Paradigma pemrograman MapReduce memungkinkan Anda untuk menskalakan data tidak terstruktur di ratusan atau ribuan server dalam cluster Apache Hadoop.

Tahapan di MapReduce



- Data input diumpankan ke fase mapper untuk memetakan data.
- Operasi shuffle, sort, dan reduce kemudian dilakukan untuk memberikan hasil akhir.

Fitur MapReduce

1. Algoritme MapReduce membantu dalam mengorganisasi dalam memproses sejumlah besar data yang disimpan secara paralel di Hadoop Distributed File System (HDFS).
2. MapReduce mengurangi waktu pemrosesan dan mendukung pemrosesan data yang lebih cepat. Ini karena semua node bekerja dengan setiap bagian datanya secara paralel.
3. Developer dapat menulis kode MapReduce dalam berbagai bahasa seperti Java, C++, dan Python.
4. MapReduce memiliki fault-tolerant karena mempertimbangkan salinan dari blok yang direplikasi di mesin/Node lain untuk diproses lebih lanjut, jika terjadi kegagalan.

Bagaimana Algoritma Hadoop MapReduce Bekerja

Mari kita pahami cara kerja algoritma MapReduce dengan memahami alur eksekusi pekerjaan secara detail.

- Data input diproses menggunakan MapReduce task yang disimpan dalam file input yang berada di HDFS.
- Format input-an mendefinisikan spesifikasi input dan bagaimana file input dibagi/split dan dibaca.
- Pemisahan input secara logic mewakili data yang akan diproses oleh masing-masing Mapper.
- Pembaca record berkomunikasi dengan pemisahan input dan mengubah data menjadi pasangan nilai kunci yang sesuai untuk dibaca oleh mapper (k, v).

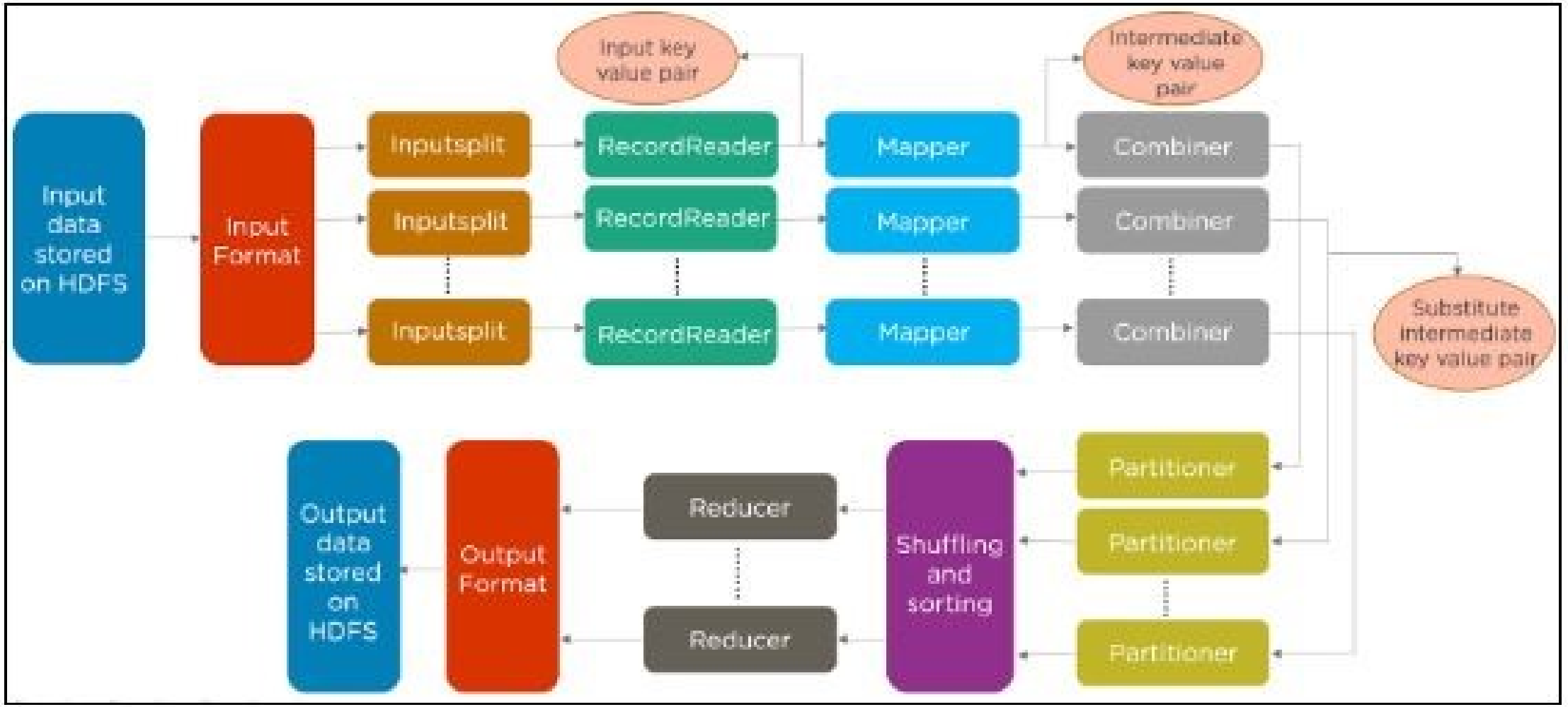
Bagaimana Algoritma Hadoop MapReduce Bekerja

- Kelas mapper memproses catatan masukan dari RecordReader dan menghasilkan pasangan nilai kunci perantara (k’, v’). Logika bersyarat diterapkan ke 'n' jumlah blok data yang ada di berbagai node data.
- Tugas penggabung adalah reducer mini. Untuk setiap combiner, ada satu mapper yang digunakan untuk mengoptimalkan kinerja pekerjaan MapReduce.
- Partisi memutuskan bagaimana output dari combiner dikirim ke reducer.

Bagaimana Algoritma Hadoop MapReduce Bekerja

- Output dari partisi diacak dan diurutkan. Semua nilai duplikat dihapus, dan nilai yang berbeda dikelompokkan berdasarkan kunci yang serupa.
- Output ini selanjutkan diumpankan sebagai input ke reducer. Semua nilai perantara untuk kunci perantara digabungkan ke dalam daftar oleh reducer yang disebut tupels.
- Penulisan record mencatat pasangan nilai kunci keluaran ini dari reducer ke file keluaran. Kemudian data keluaran disimpan pada HDFS.

MapReduce Workflow

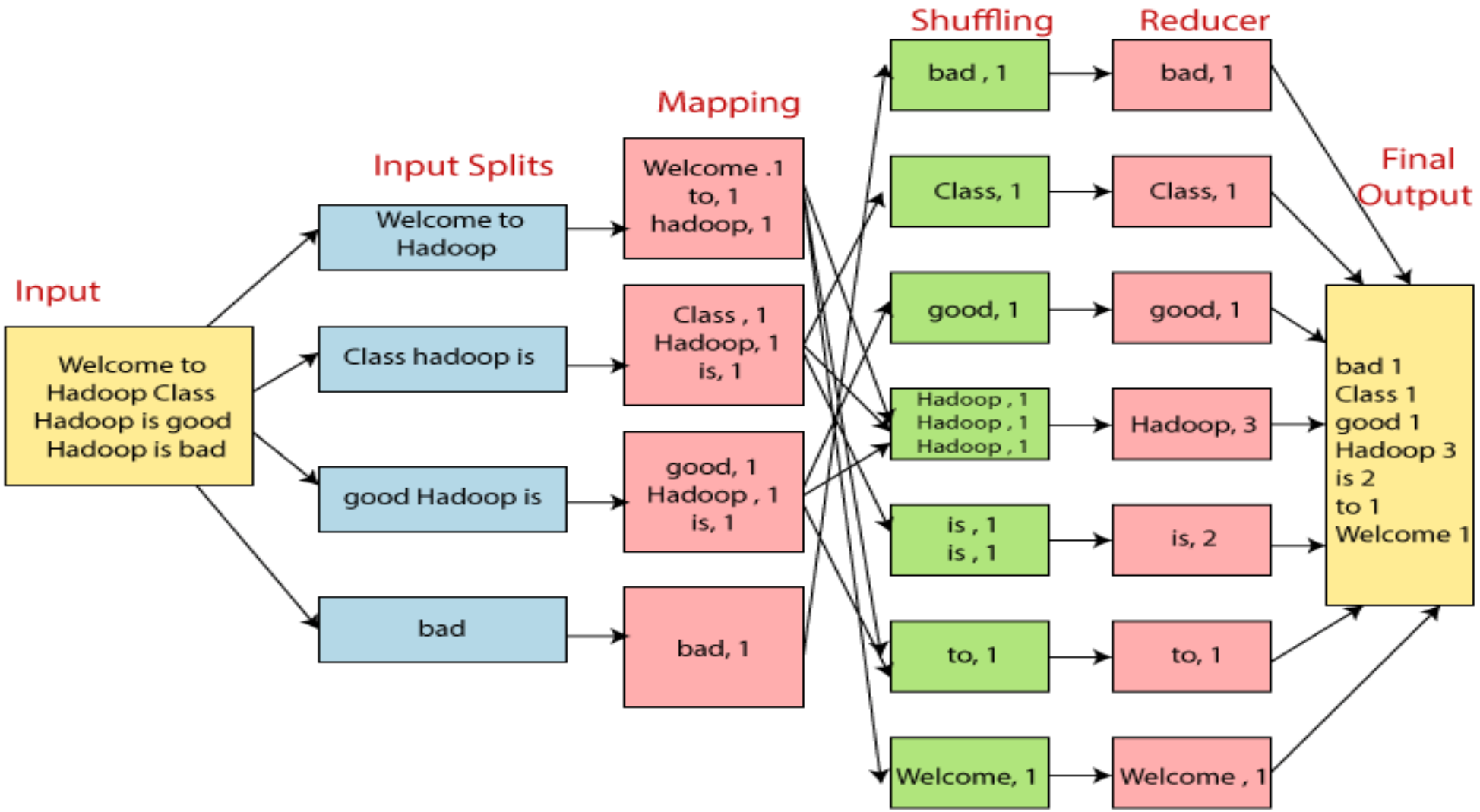


MapReduce Example WordCount

Shown in picture in the next slide is a MapReduce example to count the frequency of each word in a given input text. Our input text is,

*"**Welcome to Hadoop Class, Hadoop is good, Hadoop is bad**."*

MapReduce Example WordCount



MapReduce Example Program

Pada keseluruh implementasi program MapReduce secara fundamental dapat dibagi menjadi tiga bagian:

a. Mapper Phase Code
b. Reducer Phase Code
c. Driver Code

Mapper Code

```
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();
  private long numRecords = 0;
  private static final Pattern WORD_BOUNDARY = Pattern.compile("\\s*\\b\\s*");

  public void map(LongWritable offset, Text lineText, Context context)
      throws IOException, InterruptedException {
    String line = lineText.toString();
    Text currentWord = new Text();
    for (String word : WORD_BOUNDARY.split(line)) {
      if (word.isEmpty()) {
          continue;
      }
          currentWord = new Text(word);
          context.write(currentWord,one);
      }
  }
}
```

Mapper Code

- Class Map mengextend class Mapper yang sudah didefinisikan dalam Kerangka MapReduce.
- Input dan output dari Mapper adalah pasangan key/value
- Input:
  - *key* adalah offset dari masing-maing baris pada file teks: *LongWritable*
  - *value* adalah teks pada masing-masing baris: *Text*
- Output:
  - *key* adalah kata-kata yang sudah ditokenisasi/dipisah-pisah: *Text*
  - Kita punya *hardcoded value* pada kasus ini yaitu 1: *IntWritable*
- Sekarang kita sudah menuliskan sebuah kode dalam bahasa java untuk melakukan tokenisasi kata-kata dan memberikan nilai 1.

Reducer Code

```java
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
  @Override
  public void reduce(Text word, Iterable<IntWritable> counts, Context context)
      throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable count : counts) {
      sum += count.get();
    }
    context.write(word, new IntWritable(sum));
  }
}
```

Reducer Code

- Class Reduce mengextend class Reducer yang sudah didefinisikan dalam Kerangka MapReduce.
- Input dan output dari Reducer adalah pasangan key/value
- Input:
  - *key* adalah kata-kata unik yang telah digenerate setelah fase *sorting* dan *shuffling*: *Text*
  - *value* adalah daftar integer yang berkorespondensi ke masing-masing key: *IntWritable*
- Output:
  - *key* adalah kata-kata unik yang ada pada file: *Text*
  - *value* adalah jumlah kemunculan masing-maing kata unik: *IntWritable*
- Sekarang kita sudah menuliskan sebuah kode dalam bahasa java untuk melakukan agregasi nilai-nilai yang ada pada daftar integer yang berkorespondensi ke masing-masing key lalu menghasilkan jawaban akhir perhitungan.

Driver Code

```java
public static void main(String[] args) throws Exception {
  int res = ToolRunner.run(new WordCount(), args);
  System.exit(res);
}

public int run(String[] args) throws Exception {
  Job job = Job.getInstance(getConf(), "wordcount");
  job.setJarByClass(this.getClass());
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  job.setMapperClass(Map.class);
  job.setReducerClass(Reduce.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  return job.waitForCompletion(true) ? 0 : 1;
}
```

Driver Code

- Pada Class driver, kita lakukan pengaturan konfigurasi pekerjaan MapReduce untuk dijalankan di Hadoop.
- Kita tentukan nama pekerjaan (job), tipe data input / output mapper dan reducer.
- Kita juga tentukan nama-nama Class mapper dan reducer.
- Path folder input dan output juga ditentukan.
- Fungsi setInputFormatClass() digunakan untuk menentukan bagaimana Mapper akan membaca input data. Di sini, kita pilih TextInputFormat sehingga satu baris kata dari file teks input dibaca oleh mapper sekaligus dalam satu waktu .
- Fungsi main() adalah fungsi utamanya. Dalam metode ini, kita buat instance objek baru untuk pekerjaan tersebut.

Kesimpulan

MapReduce adalah kerangka kerja Hadoop yang membantu dalam memproses data dalam jumlah besar di banyak node. Setelah mengikuti kelas ini, Anda akan belajar tentang apa itu MapReduce, dan fitur-fitur penting dari MapReduce.

Selain into anda mampu memahami bagaimana algoritma MapReduce bekerja step by step menjalankan contoh program MapReduce di Hadoop untuk menghitung kata.

Q & A

Tugas

Kerjakan Jobsheet Praktikum *MapReduce di Hadoop*

Referensi

- https://www.udemy.com/course/the-ultimate-hands-on-hadoop-tame-your-big-data/learn/lecture/5951208
- Big Data Analytics with Hadoop 3 (2018)
- https://www.cloudera.com