# Hadoop Performance Analysis

Kelompok 7 Sistem
Basis Data-02 22/23

# Kelompok 7 SBD-02 (Pagi)


**M. Suhaili**

2106731535

**Satya Ananda S.**

2106705524

**Althaf Nafi A.**

2106634881

**Sulthan S. Y. D.**

2106731560

01

# Experiment

Comparing execution time of Word Count program using Hadoop MapReduce and without any Framework.

| Komponen | Spesifikasi |
|---|---|
| CPU | AMD Ryzen 9 4900HS 3.00 GHz 8 Core |
| RAM | 2x8 GB DDR4 3200 MHz |
| Storage | Intel SSD 660P Series 1 TB M.2 PCIE 3.0 |

# Sources

## 1 MB

http://textfiles.com/etext/FICTION/alcott-little-261.txt

## 10 MB

http://textfiles.com/etext/FICTION/

## 100 MB

https://www.kaggle.com/datasets/bittlingmayer/amazonreviews

## 1 GB

https://www.i3s.unice.fr/~jplozi/hadooplab_lsds_2015/datasets/

## 10 GB

https://www.kaggle.com/datasets/toastedalmonds/wikipedia-dump-20200820

# FICTION

| Text | Size | Description |
|---|---|---|
| 2000010.txt | 593683 | Project Gutenberg: 20,000 Leagues Under the Sea by Jules Verne |
| 2city10.txt | 776674 | A Tale of Two Cities, by Charles Dickens |
| 2city11.txt | 787707 | Project Gutenberg: A Tale of Two Cities, by Dickens |
| 7gabl10.txt | 637842 | The House of the Seven Gables, by Nathaniel Hawthorne |
| 80day10.txt | 375033 | Around the World in 80 Days, by Jules Verne |
| abbott-flatland-361.txt | 201401 | INTERNET WIRETAP: Flatland: A Romance of Many Dimensions, With Illustrations by the Author, a Square (Edwin A. Abbott) |
| aeneid.txt | 643412 | INTERNET WIRETAP: Virgil's Aeneid (1909) |
| aesop11.txt | 229320 | Aesop's Fables Translated by George Fyler Townsend |
| aesopa10.txt | 77284 | Project Gutenberg: Aesop's Fables, Second Version |
| alad10.txt | 38818 | Project Gutenberg: Aladdin and the Lamp |
| alcott-flower-619.txt | 199538 | PROJECT GUTENBERG: Flower Fables, by Louisa May Alcott's |
| alcott-little-261.txt | 1044021 | Little Women, by Louisa May Alcott |

ADAM BITTLINGMAYER · UPDATED 4 YEARS AGO

863 New Notebook Download (517 MB)

## Amazon Reviews for Sentiment Analysis

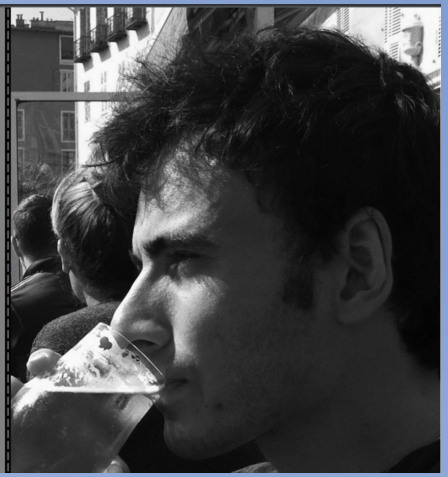
A few million Amazon reviews in fastText format

Data Card Code (109) Discussion (5)

### About Dataset

Usability
6.88

```
/*
 * Research Scientist (Chargé de Recherche)
 * @[Inria] Paris, [Whisper] team
 */

Office address:
Inria, bureau 223
2, rue Simone Iff
75012 Paris, France

E-mail address:
jean-pierre.lozi _at_ inria.fr


I am Jean-Pierre Lozi, a Research Scientist (Chargé de Recherche) in the [Whisper] team at
[Inria] Paris. I hold Master's degrees from both [Télécom Paris] and [Sorbonne Université], as
well as a PhD from [Sorbonne Université]. My research interests include multicore architectures,
synchronization primitives, operating systems, task scheduling, and graph processing.


/*
 * =============
 * Main Contents:
 * =============
 */

    [Program Committees]
    [PhD Juries]
    [Notable Talks]
    [Selected Publications]
    [Other Publications]
    [Theses]
    [Photography] * External link *
```



## Wikipedia Dump 20200820

Data Card Code (2) Discussion (0)

5 New Notebook

**wikipedia-dump.txt** (17.23 GB)

### About this file

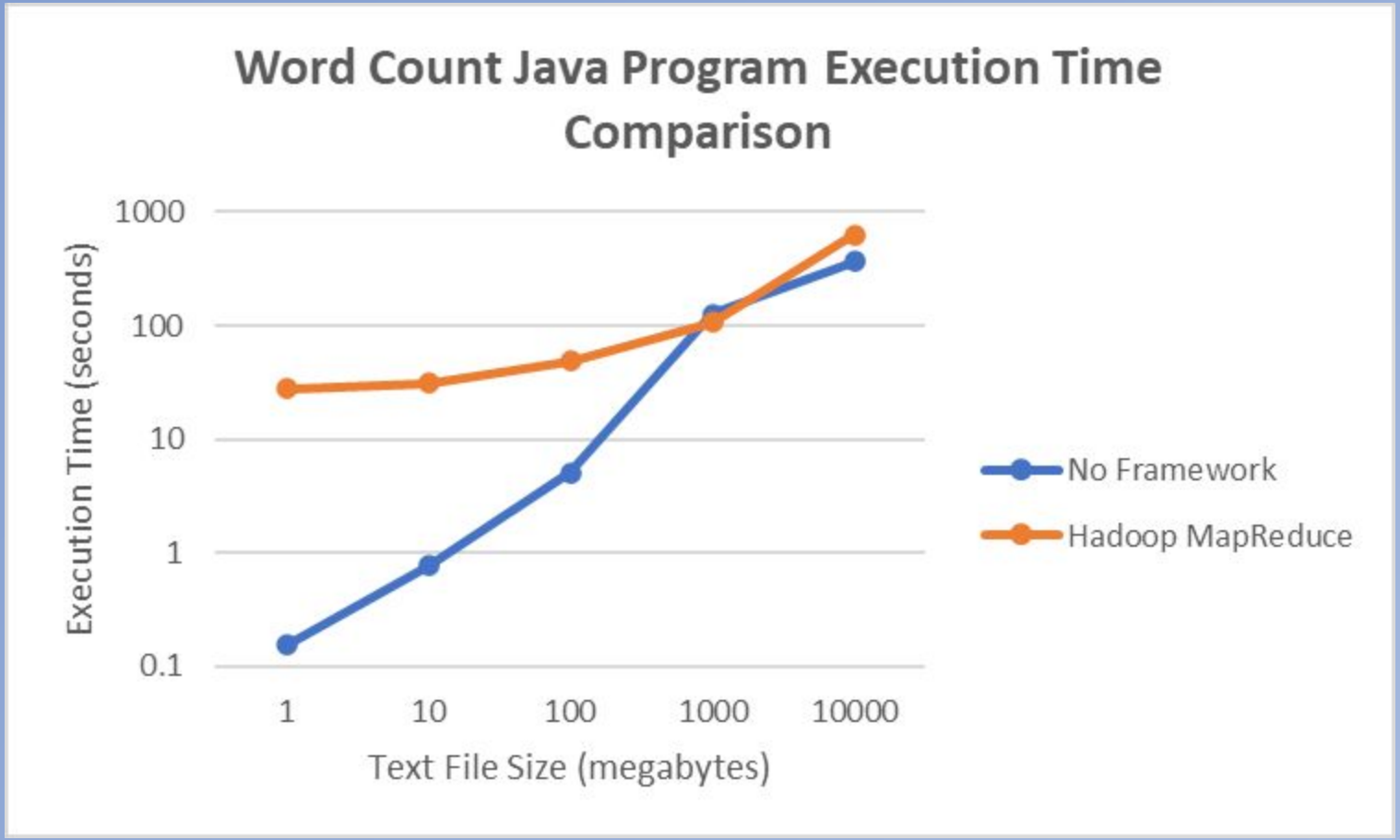
Each line in the file is a separate Wikipedia article. All punctuation has been stripped. All words are lower-cased.

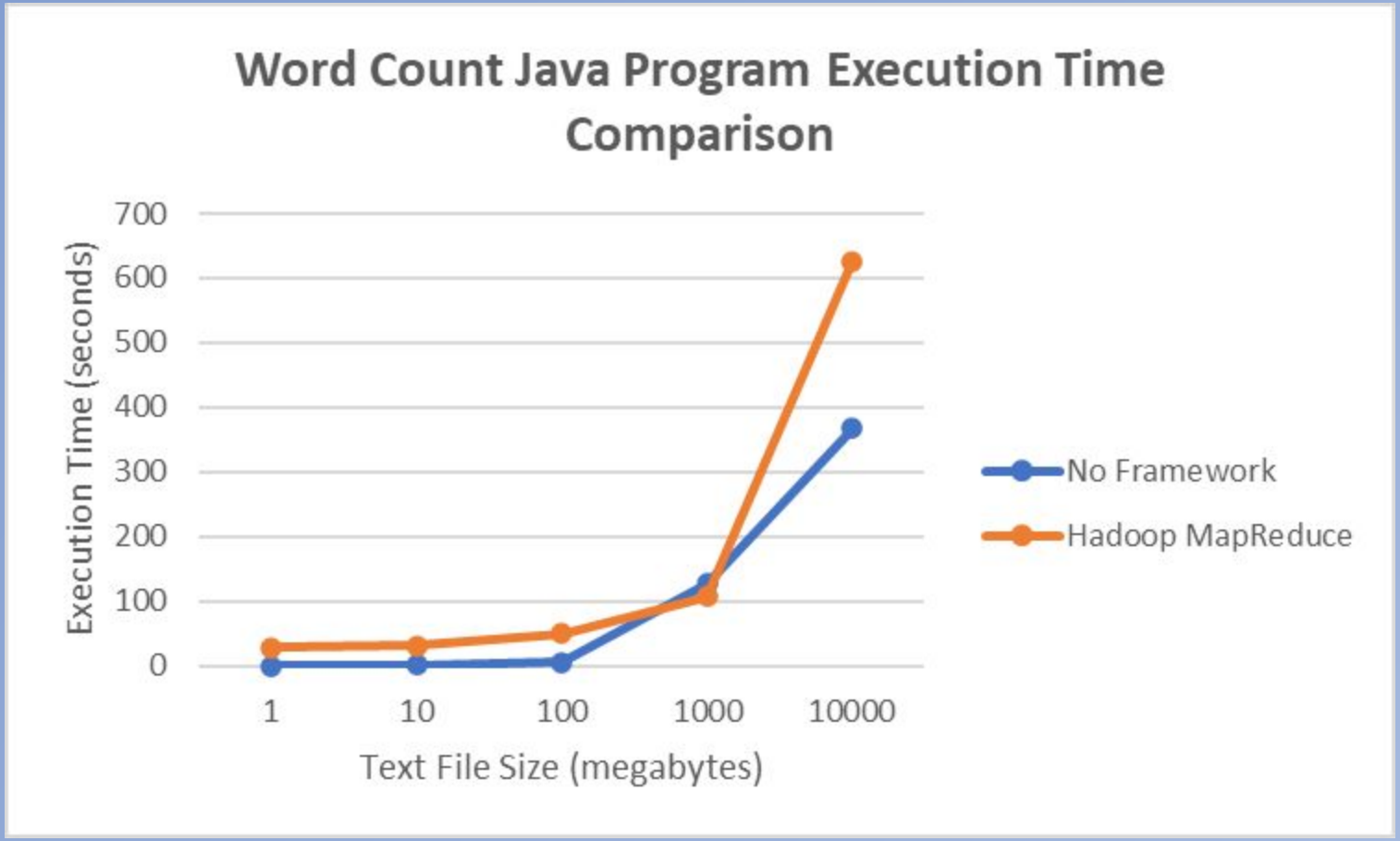
```java
import java.io.*;
import java.nio.file.*;
import java.util.*;

public class CountDriver {
    private static final HashMap<String, Integer> counts = new HashMap<>();

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

      long startTime = System.currentTimeMillis();

      try (BufferedReader reader = Files.newBufferedReader(Paths.get(args[0]))) {
          String line;
          while ((line = reader.readLine()) != null) {
              StringTokenizer tokenizer = new StringTokenizer(line);
              while (tokenizer.hasMoreTokens()) {
                  String word = tokenizer.nextToken();
                  counts.put(word, counts.getOrDefault(word, 0) + 1);
              }
          }
      }

      try (PrintWriter writer = new PrintWriter(new FileWriter(args[1]))) {
          for (Map.Entry<String, Integer> entry : counts.entrySet()) {
              writer.println(entry.getKey() + "\t" + entry.getValue());
          }
      }

      long endTime = System.currentTimeMillis();

      System.out.println("Execution time in milliseconds: " + (endTime - startTime));
    }
}
```

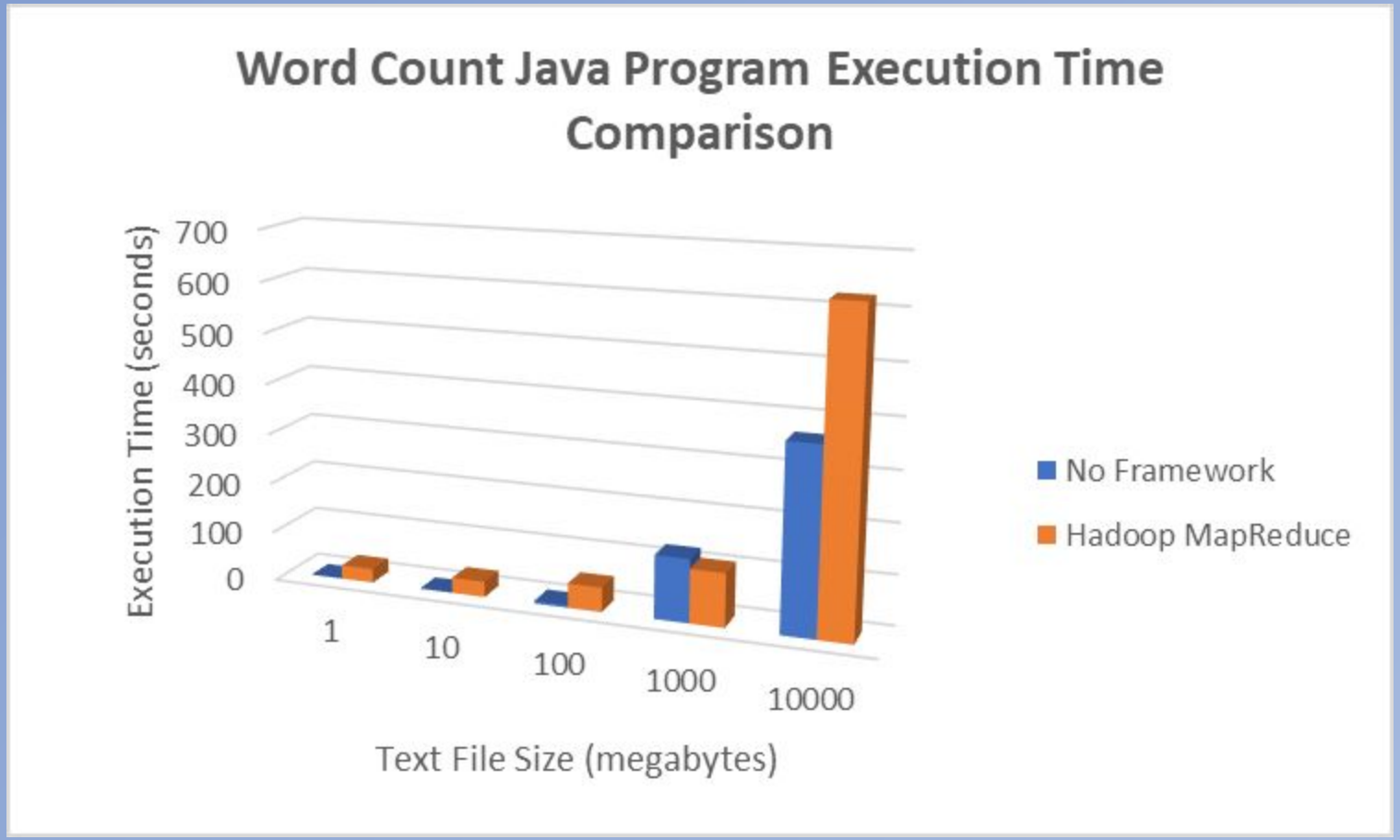
References:


1. https://stackoverflow.com/questions/18448211/word-count-from-a-file
2. https://stackoverflow.com/questions/33927476/counting-words-from-a-file
3. https://youtu.be/dOYieTlItMM
4. https://codereview.stackexchange.com/questions/188399/word-counter-using-a-word-list-and-some-text-files

| File Size | Time with Framework (seconds) | | Ratio |
|---|---|---|---|
| | None | Hadoop MapReduce | |
| 1 MB / 0.001 GB | 0.153 | 28 | 183.01 |
| 10 MB / 0.01 GB | 0.774 | 31 | 40.05 |
| 100 MB / 0.1 GB | 5.061 | 49 | 9.68 |
| 1000 MB / 1 GB | 126.28 | 107 | 0.85 |
| 10000 MB / 10 GB | 367.523 | 626 | 1.70 |


Word Count Java Program Execution Time Comparison
Execution Time (seconds)
1000
100
10
1
0.1
1
10
100
1000
10000
Text File Size (megabytes)
No Framework
Hadoop MapReduce


Word Count Java Program Execution Time Comparison
Execution Time (seconds)
700
600
500
400
300
200
100
0
1
10
100
1000
10000
Text File Size (megabytes)
No Framework
Hadoop MapReduce


Word Count Java Program Execution Time Comparison
Execution Time (seconds)
700
600
500
400
300
200
100
0
1
10
100
1000
10000
Text File Size (megabytes)
No Framework
Hadoop MapReduce

02

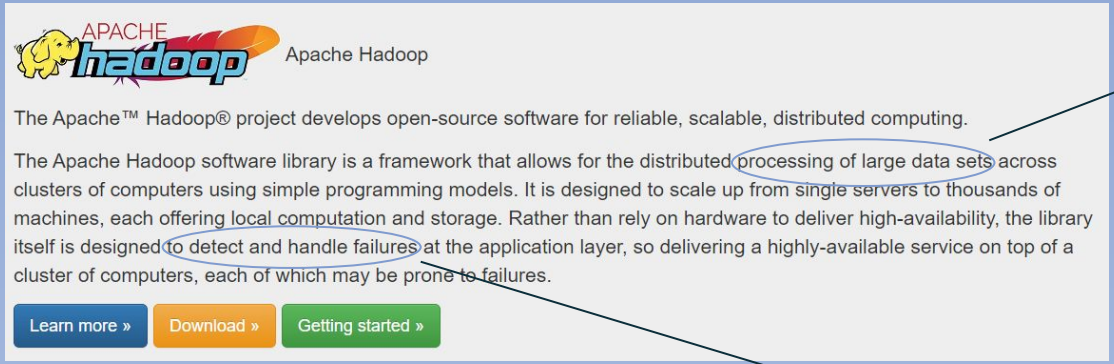
# Analysis

Why is Hadoop MapReduce slower?

# Analysis

Bila terlihat dari tabel perbandingan execution time dengan menggunakan framework hadoop mapreduce dengan tidak menggunakan hadoop, secara garis besar memang terlihat di lapangan bahwa penggunaan hadoop mapreduce **lebih lama** dibandingkan dengan tidak menggunakan mapreduce.

Namun hal tersebut **tidak berarti** bahwa hadoop mapreduce tidak efektif sama sekali dalam menyelesaikan masalah word count.




Sumber: Apache Hadoop

Bila dikutip dari laman resmi hadoop, dapat disimpulkan bahwa hadoop memang didesain ***untuk menangani data sets yang sangat besar*** dan sifatnya yakni scalable ke beberapa komputer.

Dengan demikian, hadoop juga didesain untuk mendeteksi failure dan kesalahan dengan semakin besar nya cluster komputer

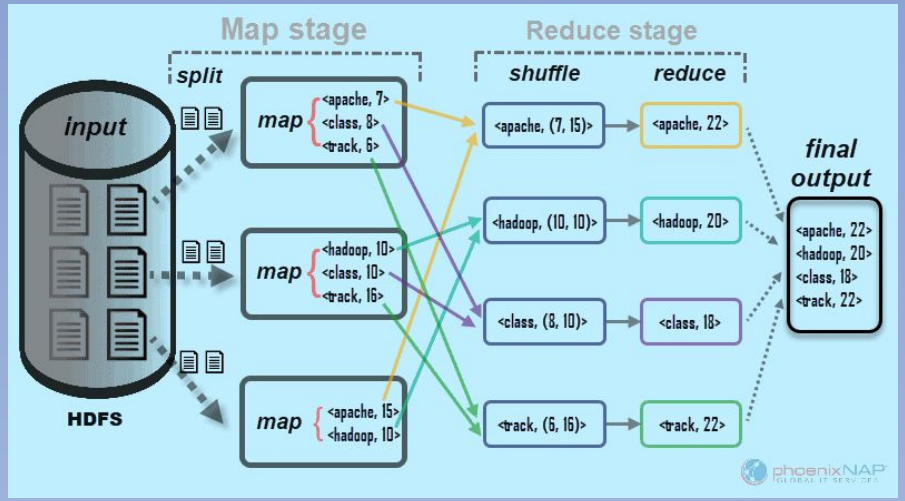
| File Size | Time with Framework (seconds) | | Ratio |
|---|---|---|---|
| | None | Hadoop MapReduce | |
| 1 MB / 0.001 GB | 0.153 | 28 | 183.01 |
| 10 MB / 0.01 GB | 0.774 | 31 | 40.05 |
| 100 MB / 0.1 GB | 5.061 | 49 | 9.68 |
| 1000 MB / 1 GB | 126.28 | 107 | 0.85 |
| 10000 MB / 10 GB | 367.523 | 626 | 1.70 |

Rasio nya terus berkurang secara signifikan seiring dengan bertambahnya data sets/file size

Sehingga hadoop memang kurang cocok digunakan untuk data sets yang size nya kecil, seperti pada eksperimen itu yaitu dengan menggunakan file .txt.

## Mengapa hal ini bisa terjadi?

## Hal ini dapat terjadi karena dari bagaimana hadoop mapreduce itu bekerja




Sumber: [What is Hadoop Mapreduce and How Does it Work (phoenixnap.com)]

1. Mapping : data di split dan map dalam ke seluruh cluster.
2. Shuffling : memastikan data sudah dalam keadaan di sorted sebelum masuk ke reducing.
3. Reducing : Hasil dari mapreduce yang akan di group kepada final output.

Sehingga, apabila data sets nya hanya memiliki size yang kecil, jika menggunakan mapreduce, maka harus tetap melewati seluruh stage di atas sehingga ***sangat membuang waktu*** !

# Hadoop MapReduce – 1 MB

```
PS D:\Java Codes\WC\WordCount> hadoop jar target/WordCount-1.0-SNAPSHOT.jar org.satyaananda.WordCount /tugas_in/1-MB.txt /tugas_out/1-MB
2023-06-06 23:43:07,865 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2023-06-06 23:43:08,927 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute
your application with ToolRunner to remedy this.
2023-06-06 23:43:08,959 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Asus/.staging/job_1686069479651
_0001
2023-06-06 23:43:09,220 INFO input.FileInputFormat: Total input files to process : 1
2023-06-06 23:43:09,309 INFO mapreduce.JobSubmitter: number of splits:1
2023-06-06 23:43:09,909 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1686069479651_0001
2023-06-06 23:43:09,911 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-06-06 23:43:10,120 INFO conf.Configuration: resource-types.xml not found
2023-06-06 23:43:10,120 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-06-06 23:43:10,570 INFO impl.YarnClientImpl: Submitted application application_1686069479651_0001
2023-06-06 23:43:10,623 INFO mapreduce.Job: The url to track the job: http://LAPTOP-K086D7F9:8088/proxy/application_1686069479651_0001/
2023-06-06 23:43:10,624 INFO mapreduce.Job: Running job: job_1686069479651_0001
2023-06-06 23:43:26,905 INFO mapreduce.Job: Job job_1686069479651_0001 running in uber mode : false
2023-06-06 23:43:26,908 INFO mapreduce.Job:  map 0% reduce 0%
2023-06-06 23:43:33,054 INFO mapreduce.Job:  map 100% reduce 0%
2023-06-06 23:43:40,163 INFO mapreduce.Job:  map 100% reduce 100%
2023-06-06 23:43:40,184 INFO mapreduce.Job: Job job_1686069479651_0001 completed successfully
2023-06-06 23:43:40,292 INFO mapreduce.Job: Counters: 54
```

Submit job memakan +- 3 detik

Alokasi resource sebelum berjalan memakan +- 16 detik

Eksekusi MapReduce sendiri memakan +- 14 detik

# Tanpa Hadoop MapReduce – 1 MB

```
PS D:\Books\Various Text Files> java WordCount 1-MB.txt ./OutputNon/1-MB-out.txt
Execution time in milliseconds: 153
PS D:\Books\Various Text Files> java WordCount 10-MB.txt ./OutputNon/10-MB-out.txt
Execution time in milliseconds: 774
PS D:\Books\Various Text Files> java WordCount 100-MB.txt ./OutputNon/100-MB-out.txt
Execution time in milliseconds: 5061
```

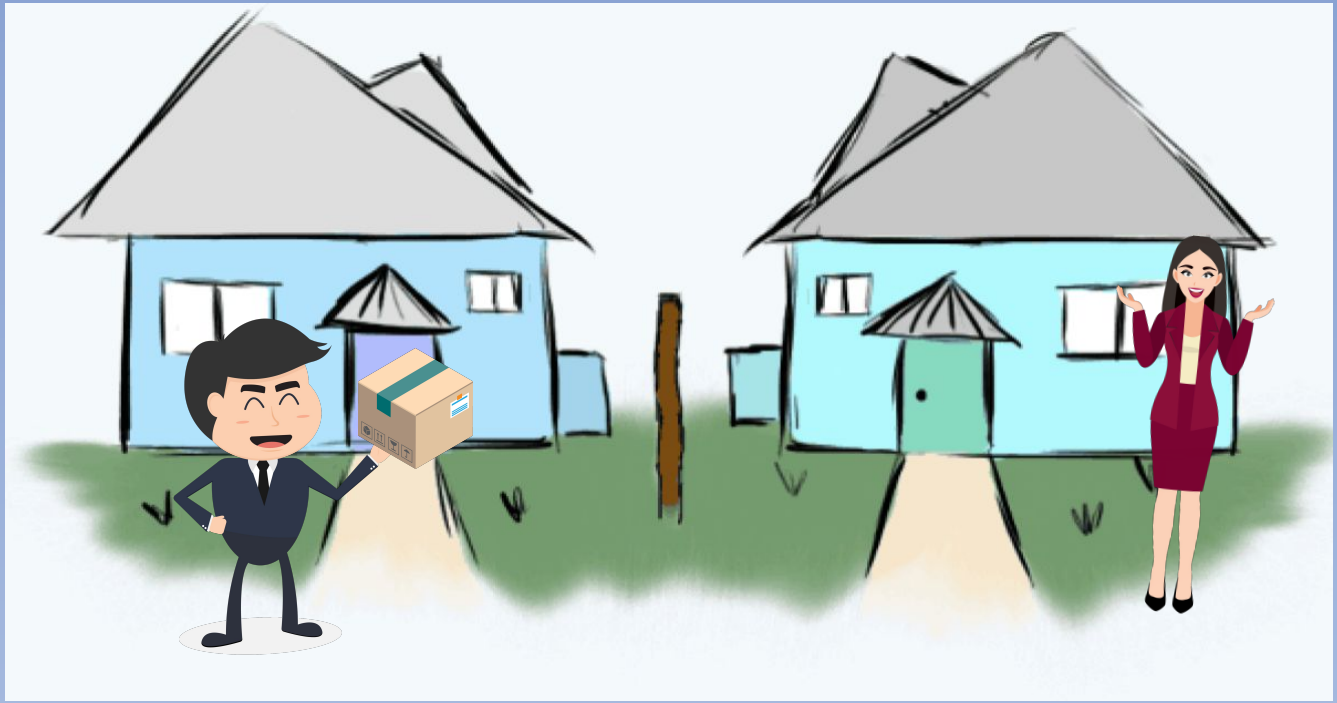
Hanya memakan 153 milidetik / 0.153 detik

```
1   frowning  39
2   undermining 3
3   spilling  3
4   Rhadamanthus, 1
5   DUKE, 1
```

Performa Word Count konvensional juga lebih cepat karena sifatnya read & write secara langsung (tidak memperhatikan urutan), sedangkan pada MapReduce, setiap kata diurutkan secara alfabet.

```
□     1
!     4
!"    3
"     83
"""You     1
```

# Analogi Word Count Tanpa Hadoop pada File Kecil



# Analogi

# Analogi Hadoop MapReduce Word Count Single Node pada File Kecil

# Analogi

# Analogi

Hadoop = Perusahaan Delivery, MapReduce = Jasa Pengiriman, Word Count = Parsel/Paket

# Kesimpulan *Eksperimen*

Pada input file yang kecil, Hadoop MapReduce tidak memberikan benefit karena adanya overhead. Hadoop MapReduce memiliki kemampuan alokasi resource dan fault tolerance yang bisa memakan waktu lama untuk file kecil. Terlihat bahwa semakin besar file input, perbedaan waktu semakin kecil.

# Thanks!

Do you have any questions?

slidesgo