Laporan Demonstrasi WordCount Hadoop dan Perbandingannya Terhadap Program Luar Hadoop Kelas 01 Grup 6

Anggota:

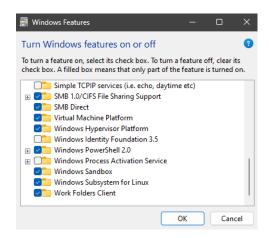
- Brian Yudha Sandi
- Muhammad Irfan Fakhrianto
- Muhammad Rizky Utomo
- Muhammad Salman Sadad

A. Instalasi dan Setup

Hadoop yang diinstal adalah Hadoop versi 3.3.2 di platform Ubuntu 20.04.2 yang berjalan sebagai sistem operasi Windows Subsystem for Linux (WSL) 2. Instalasi Ubuntu berjalan dengan cara seperti yang dijelaskan pada link berikut:

https://learn.microsoft.com/en-us/windows/wsl/insta

Sebelum menginstal Ubuntu, pengguna menginstal WSL dengan menceklis opsi penginstalannya pada daftar fitur opsional di Windows:



User melakukan restart setelah melakukan aksi di atas. Selanjutnya, user memasukkan perintah berikut di Windows Terminal untuk menginstal Ubuntu:

Penginstalan berjalan dan Ubuntu memerintahkan pengguna untuk memasukkan username serta password lokal.

Instalasi Hadoop

Pengguna mengikuti tutorial di laman berikut untuk menginstal Hadoop 3.3.2 di WSL:

https://kontext.tech/article/978/install-hadoop-332-in-wsl-on-windows

Bagian pertama ada penginstalan Java Development Kit (JDK) versi 8 di Ubuntu. Untuk JDK 11, kami mendapatkan masalah dalam mencoba untuk menjalankan Hadoop pada JDK tersebut. Padahal, JDK 11 didukung seperti yang dinyatakan pada link berikut:

https://cwiki.apache.org/confluence/display/HADO OP/Hadoop+Java+Versions

Maka, JDK 8 adalah versi JDK yang digunakan:

```
riztomo@RIZKY-UBUNTU:~$ java -version
openjdk version "1.8.0_362"
OpenJDK Runtime Environment (build 1.8.0_362-8u372-ga~us1-0ubuntu1~22.04-b09)
OpenJDK 64-Bit Server VM (build 25.362-b09, mixed mode)
riztomo@RIZKY-UBUNTU:~$
```

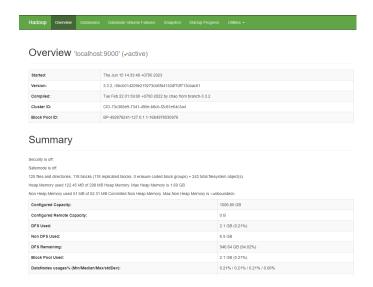
Setup selanjutnya adalah men-download Hadoop, melakukan Unzip terhadap package-nya, mencoba SSH ke Hadoop, dan melakukan konfigurasi terhadap file *environment variable* dan file konfigurasi Hadoop. Jika sudah, namenode akan diformat untuk mode single-node yang dijelaskan dalam tutorial. Ketika selesai, daemon untuk file system Hadoop dan YARN cluster dijalankan dengan script berikut di dalam Hadoop:

hadoop-*/sbin/start-dfs.sh hadoop-*/sbin/start-yarn.sh

(*versi Hadoop)

Dua laman berikut dapat diakses setelah semua service yang diperlukan di-start.

http://localhost:9870/dfshealth.html#tab-overview



http://localhost:8088/cluster



Untuk menghentikan service, dua script berikut dijalankan:

hadoop-*/sbin/start-dfs.sh hadoop-*/sbin/start-yarn.sh

B. Pengujian

Penulis menguji kemampuan Hadoop dengan memberikan file ukuran 10 MB, 50 MB, 100 MB, 500 MB dan 1 GB berisi kata-kata dari lagu "Never Gonna Give You Up" yang di-generate secara acak dengan sebuah program. Kelima file ini dijalankan pada program MapReduce di Hadoop untuk dicari jumlah setiap kata yang ada di dalamnya.

Selain menjalankannya di Hadoop, sebuah program Java lain yang tidak menggunakan Hadoop dibuat sebagai perbandingan dengan MapReduce Hadoop.

```
import java.io.IOException;
import java.util.Map;
import java.util.TreeMap;
import java.nio.file.Files;
import java.nio.file.Paths;
public class WordCount {
```

```
// words in the given string
  static void count freq(String[] arr)
     Map<String,Integer> mp=new TreeMap<>();
     // Splitting to find the word
     // Loop to iterate over the words
     for(int i=0;i<arr.length;i++)
       // Condition to check if the
       // array element is present
       // the hash-map
       if(mp.containsKey(arr[i]))
          mp.put(arr[i], mp.get(arr[i])+1);
       else
          mp.put(arr[i],1);
     }
     // Loop to iterate over the
     // elements of the map
     for(Map.Entry<String,Integer> entry:
            mp.entrySet())
       System.out.println(entry.getKey()+
            " - "+entry.getValue());
  }
  // Driver Code
  public static void main(String[] args) {
                              String
                                        filePath
"/home/riztomo/hadoop/input/500MB.txt";
       String contents = null;
       try {
               contents
                                                 new
String(Files.readAllBytes(Paths.get(filePath)));
       } catch (IOException e) {
               e.printStackTrace();
       }
```

// Function to count frequency of

```
System.out.println("##### Split string by
multiple spaces");
    String[] str = contents.split("\\s+");

// Function Call
    System.out.println("##### Counting words");
    count_freq(str);
}
```

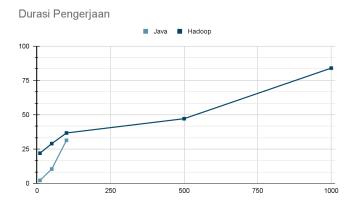
Program diambil dari referensi berikut:

- https://www.geeksforgeeks.org/calculate-the -frequency-of-each-word-in-the-given-string
- https://www.jackrutorial.com/2018/06/java-s plit-string-by-space-and-newline.html

C. Analisis

Setelah dilakukan beberapa kali pengujian dengan ukuran file di atas, didapat hasil perhitungan dengan durasi waktu sebagai berikut:

Ukuran File (MB)	Java	Hadoop
10	2,1244s	21,964s
50	10,4306s	28,9392s
100	31,3466s	36,694s
500	-	47,1064s
1000	-	83,9734s



Melihat dari bentuk grafik yang dihasilkan, java dapat mengolah data dengan lebih cepat dari hadoop

pada file yang lebih kecil. Namun setelah 100 MB, program java tidak mampu melakukan runtime karena keterbatasan komputer dan kapasitas head space java.

D. Kesimpulan

Karena keterbatasan komputer yang digunakan, jadi hanya bisa melakukan runtime pada file sebesar 100 MB saja. Namun pada real case nya, apabila dilakukan wordcount pada file yang jauh lebih besar, hadoop dapat melakukan runtime dengan durasi waktu yang lebih singkat karena hadoop memang didesain untuk menghandle big data. Mengapa hadoop cenderung kurang baik ketika meng-handle file yang lebih kecil? Karena hadoop melakukan pemecahan data terlebih dahulu dengan modul MapReduce yang dimiliki, bahkan untuk file yang kecil. Sehingga hadoop akan kurang efisien ketika mengerjakan file yang kecil