Perbandingan Menggunakan Metode Hadoop Archive dengan Combine File Input Format untuk Mengatasi File Berukuran Kecil di HDFS

Arif Sumanggara Nainggolan[1], Setyorini[2], Erwid Mustofa Jadied[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]arifsumanggara@students.telkomuniversity.ac.id, [2]setyorini@telkomuniversity.ac.id,
[3]jadied@telkomuniversity.ac.id

Abstrak

Hadoop adalah *platform open source* berbasis Java yang berada di bawah lisensi Apache dan digunakan untuk mendukung aplikasi yang berjalan pada *big data.* HDFS merupakan komponen dari Hadoop yang dapat menyimpan *file* yang sangat besar dengan akses data *streaming* dan berjalan pada kelompok komoditas perangkat keras. HDFS dirancang untuk menangani *file* besar dalam jumlah banyak yang ukuran *file* nya hingga petabyte, exabyte. Di era sekarang ini, *Big Data* menjadi topik yang menarik untuk dibahas dengan bentuk data yang berbagai *variety,velocity* dan *volume* data. Di dalam *Big Data* terdapat *file-file* yang berukuran besar dan kecil yang akan di proses HDFS. Akan tetapi terdapat masalah yang di temukan ketika HDFS menangani *file* kecil dalam jumlah banyak, sehingga terdapat beberapa solusi yang ditawarkan untuk menangani *file* kecil dalam jumlah banyak di HDFS yaitu dengan menggunakan metode HAR (*Hadoop Archive*) dan *Combine File Input Format*. Dengan kedua metode ini maka file kecil di HDFS teratasi. kemudian di lakukan perbandingan yang mana metode yang digunkan menghasilkan penggunaan block dan waktu pemrosesan yang di gunakan sedikit.

Kata kunci : Hadoop, HDFS, HAR, combine file input format, file kecil, block.

Abstract

Hadoop is an open source platform based on Java that is under the Apache license and is used to support applications running on big data. HDFS is a component of Hadoop that can store very large files with streaming data access and run on hardware commodity groups. HDFS is designed for large files in the number of files up to petabytes, exabytes. In today's era, Big Data has become an interesting topic to discuss with various forms of data, speed and volume data. In Big Data, there are large and small files that will be processed in HDFS. However, there are problems when HDFS files are small in large numbers, so that there are several solutions offered for large numbers of small files in HDFS using HAR (Hadoop Archive) and Combine File Input Format methods. With these two methods the small files in HDFS are resolved. then do a comparison which method is used to produce blocks and the right time is used a little.

Keywords: Hadoop, HDFS, HAR, combine file format input, small file, block.

1. Pendahuluan

Latar Belakang

Hadoop adalah Suatu *software framework* (kerangka kerja perangkat lunak) open source berbasis *Java* di bawah lisensi *Apache* untuk aplikasi komputasi data besar secara intensif [1]. Hadoop mampu menyelesaikan masalah yang berhubungan dengan data yang sangat besar. Dengan banyaknya perkembangan data saat ini Hadoop menjadi solusi y/ang tepat untuk menangani, menyimpan dan mengolah data yang sangat besar. Hadoop juga sudah sangat banyak digunakan di berbagai prusahaan seperti yahoo, facebook dll [2].

HDFS adalah *system file* yang dirancang untuk menyimpan *file* yang sangat besar dengan akses data *streaming* dan berjalan pada kelompok perangkat keras komoditas [3]. HDFS adalah tempat atau direktori di komputer dimana data Hadoop disimpan. HDFS akan melakuakan proses pemecahan *file* yang besar menjadi bagian yang lebih kecil. *Hadoop distributed file system* dirancang untuk menangani data dalam jumlah banyak yang ukuran *file* nya hingga petabyte, exabyte.

Namun terdapat masalah yang di temukan ketika HDFS menangani *file* kecil dalam jumlah banyak [4], sehingga terdapat beberapa solusi yang ditawarkan yaitu dengan menggunakan metode HAR(*Hadoop Archive*) dan *Combine File Input*. Dengan kedua metode ini maka file kecil di HDFS teratasi. Kemudian akan di lakukan

perbandingan, yang mana metode yang digunkan menghasilkan penggunaan block dan waktu pemrosesan yang di gunakan pun sedikit [5].

Topik dan Batasannya

Terdapat masalah yang di temukan ketika HDFS menangani file kecil dalam jumlah banyak sehingga tidak bekerja dengan baik. File kecil dapat didefinisikan sebagai file apa pun yang secara signifikan lebih kecil dari ukuran blok HDFS [6]. Ukuran blok Hadoop di *set* ke 128 dan 512 MB. Sehingga untuk mengatasi masalah file kecil di HDFS maka digunakan metode Combine File Input Format dan Hadoop Archive. Metode ini memproses setiap file kecil di hdfs hingga menghasilkan *output* file yang sudah digabungkan.

Batasan pekerjaan yang dilakukan di dalam TA ini berupa Inputan file text yang berisikan 2000 dan 4000 file karena keterbatasa Komputer yang tersedia Program yang dibangun hanya dapat memproses small file yang akan di merge, tidak untuk mengeluarkan file atau data tertentu.

Tujuan

Didalam masalah yang terjadi di hdfs maka tujuan daripada Tugas Akhir ini adalah dapat mengimplementasikan HDFS yang mampu menangani *file* berukuran kecil dalam jumlah banyak. Maka untuk mengatasi file berukuran kecil dalam jumlah banyak yaitu Dengan menggunakan metode *combine file input format* dan HAR (*Hadop Archive*) maka masalah *file* kecil yang di proses di HDFS bisa teratasi. Kemudia akan di hitung waktu pemrosesan *file* dari setiap metode dan block yang digunakan.

Organisasi Tulisan

Di dalam studi terkait terdapat penjelasan mengenai teori-teori yang terkait atau berhubungan dengan tugas akhir yang dibuat. Yang didalamnya terdapat teori hadoop, hadoop distributed file system, mapreduce, combine file input format, dan hadoop archive. Sistem yang dibangun mejelaskan tentang alur alur atau cara kerja sistem yang dibuat. Yang didalamnya terdapat alur kerja hadoop archive, combine file input format, dataset, skenario pengujian, skenario pengujian combine file input format, skenario pengujian hadoop Archive, dan spesifikasi sistem. Evaluasi menjelaskan tentang penilaian untuk menentukan, apakah pengujian yang dilakukan sesuai dengan tujuan yang ingin dicapai. yang didalamnya terdapat hasil pengujian dan hasil analisis pengujian. Kemudian terdapat kesimpulan yang menjelaskan kesimpulan daripada pengujian yang telah dilakukan. Dan terdapat daftar pustaka sebagai acuan paper yang berkaitan dengan tugas akhir yang telah dibuat. Kemudian yang terakhir terdapat lampiran, lampiran berupa detail data, detail hasil pengujian, analisis hasil pengujian dan screenshot tampilan system.

2. Studi Terkait

Hadoop

Hadoop adalah Suatu *software framework* (kerangka kerja perangkat lunak) open source berbasis *Java* di bawah lisensi *Apache* untuk aplikasi komputasi data besar secara intensif [1]. Hadoop dapat mengolah data dalam jumlah yang sangat besar hingga petabyte dan dijalankan diatas ribuan komputer. Hadoop bersifat open source dan terdapat empat komponen didalam hadoop yaitu terdiri atas [7]:

1. HDFS (*Hadoop Distributed File System*) yaitu Data yang terdistribusi
2. MapReduce yaitu *Framework* dari aplikasi yang terdistribusi
3. Hadoop Common
4. *framework* Yet Another Resource Negotiator (YARN).

Hadoop Distributed File System

HDFS adalah system file yang dirancang untuk menyimpan file yang sangat besar dengan akses data streaming dan berjalan pada kelompok perangkat keras komoditas [3]. System file ini menyimpan file berukuran

besar yang ukurannya mencapai gigabyte, terabyte hingga petabyte. HDFS dibangun berdasarkan pola pemrosesan data yang paling effisien yaitu pola tulis sekali dan baca berkali kali.

Dalam HDFS terdapat *Name Node* dan *Data Node. Name Node* adalah perangkat keras yang berisi sistem operasi GNU / Linux dan *software Name Node*. Sistem HDFS memiliki Name Node yang bertindak sebagai *server master* berisikan metadata dan melakukan tugas-tugas.

: Mengelola sistem *file* namespace, mengatur akses klien ke *file*, mengeksekusi operasi system *file* seperti mengubah nama, menutup, dan membuka file dan direktori. *Data Node* adalah komoditas perangkat yang memiliki sistem operasi GNU/ Linux dan *software Data Node*. Untuk setiap dalam sebuah *cluster*, akan ada beberapa *Data Node*. *Node* ini merupakan penyimpanan data pada sistem HDFS. *Data Nodes* melakukan operasi baca-tulis

Mapreduce

Model pemrograman *MapReduce* membagi proses menjadi tiga tahapan, yaitu tahapan *Map*, tahapan *shuffle* dan tahapan *Reduce* [7]. Untuk tahapan *shuffle* dan *reduce* digabungkan dalam satu tahap yaitu tahap *reduce*. *Map* merupakan proses yang berjalan secara parallel, sedangkan Reduce merupakan proses penggabungan hasil dari proses *map* [8]. MapReduce menerapkan dua fungsi: fungsi Map dan fungsi Reduce. Selanjutnya, kedua dari dua fungsi memiliki <key, value> pair sebagai input dan outputnya [8]:

1. Tahap map, memproses data inputan yang umumnya berupa file yang tersimpan dalan HDFS, inputan tersebut kemudian diubah menjadi tupel yaitu pasangan antara *key* dan *value*-nya.

2. Tahap reduce, memproses data inputan dari hasil proses map, yang kemudian dilakukan tahap *shuffle* dan *reduce* yang hasil data set baru-nya disimpan di HDFS kembali.

Hadoop Archive

Hadoop Archive adalah sebuah teknik yang digunakan untuk mengkemas *file-file* yang lebih kecil di banyak nomor pada blok HDFS dengan cara yang efisien. Keuntungan dari HAR adalah untuk mengurangi *overhead* penyimpanan data pada *Name Node* dan mengurangi operasi pemetaan [5]. Dan juga ada beberapa kelemahan dari sistem HDFS yaitu [9]:

1. Membaca HAR kurang efisien dan lebih lambat dari membaca *file* menggunakan HDFS.
2. Sulit untuk mengoperasikan proses peta di HAR.
3. Untuk meng-upgrade HAR harus ada perubahan yang dilakukan dalam *system* HDFS.

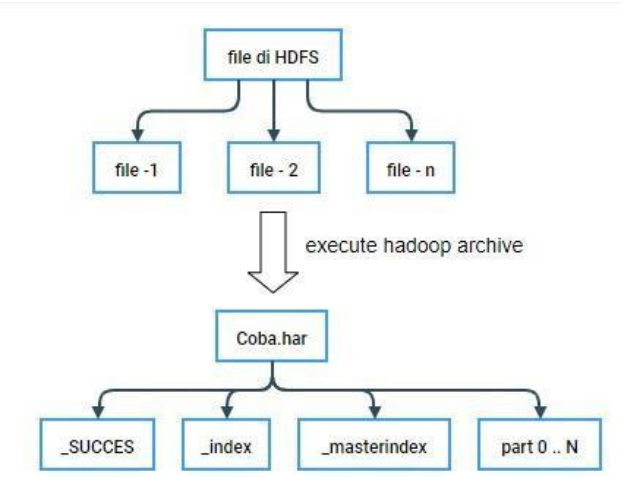
Combine File Input Format

Ini adalah jenis kelas abstrak yang secara eksplisit dapat digunakan untuk menggabungkan beberapa "*file* kecil "menjadi satu" *file* besar ". Ini adalah kelas bawaan di Hadoop. Ia bekerja dengan menggabungkan beberapa *file* kecil di setiap bagian. Di sini kita dapat memproses banyak *file* kecil dalam *format* satu tugas *Map* [10].

Combine File Input Format ini menggunakan *Map* & *Reduce*. Tugas *InputFormat* bertanggung jawab untuk memecah *file* masukan. Metode ini akan meningkatkan kinerja sistem dengan memodifikasi kelas *InputFormat*. Dengan menggunakan *Map* akan mendapatkan banyak *inputan* yang akan di proses. Sehingga mencapai paralelisem [11].

3. Sistem yang Dibangun

Hadoop Archive



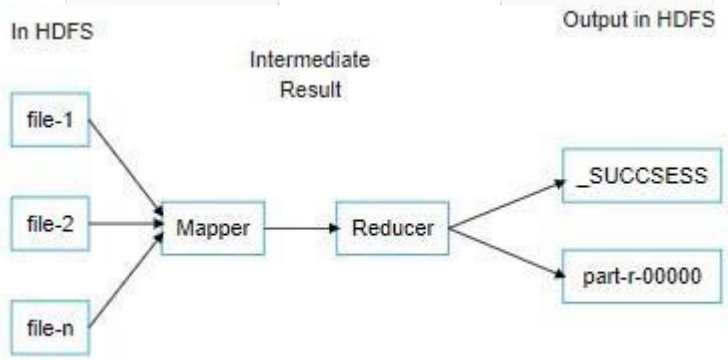
Gambar 1. *Hadoop Archive.*

Untuk menangani file kecil dalam jumlah banyak, rancangan *system* yang di buat ialah Hadoop Archive. Hadoop Archive ini akan mengkemas sejumlah *file* kecil. Kemudian *file* kecil akan di archivekan sehingga menghasilakan Coba.har. Coba.har terdiri dari 4 file yaitu file sementara, masterindex, index dan part-0 file.

I) masterindex: berisi hash dan offset (lokasi) informasi. II)
index: berisi struktur direktori dan status file.
III) Bagian-0: Penyimpanan file data aktual.

untuk mengakses file dari bagian-0, kita harus mengakses dua *file indeks*. Untuk membaca file, program harus melewati *masterindeks*, *indeks* dan kemudian mencari di *file* (bagian-0). Jenis pengindeksan ini merupakan tambahan *overhead* ke *NameNode.*

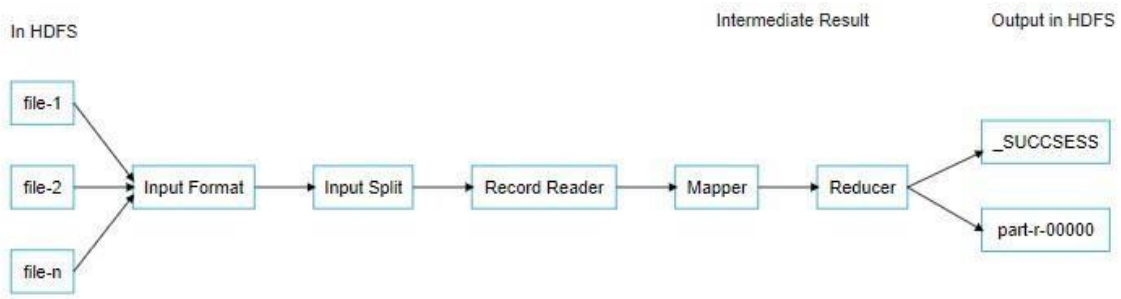
Setelah proses *HAR* di jalankan maka akan dilakukan pemrosesan *WordCount* yang mana file HAR akan dijadikan input pada *WordCount.* Berikut prosesnya:



Gambar 2. *WordCount.*

Maper memproses setiap *Input file* dari Hdfs dan menghasilkan pasangan *key-value pair*. *Output mapper* juga dikenal sebagai *Intermediate Result* yang ditulis ke disk lokal. Kemudian *Output mapper* yang terletak di disk lokal akan di proses *reducer*, yang mana *Reducer* melakukan pemrosesan ringan seperti *agregasi* / penjumlahan dan akan mendapatkan *output* dari proses penjumlahan. *Output* dari *reducer* adalah *output* akhir, yang disimpan dalam HDFS.

Combine File Input Format

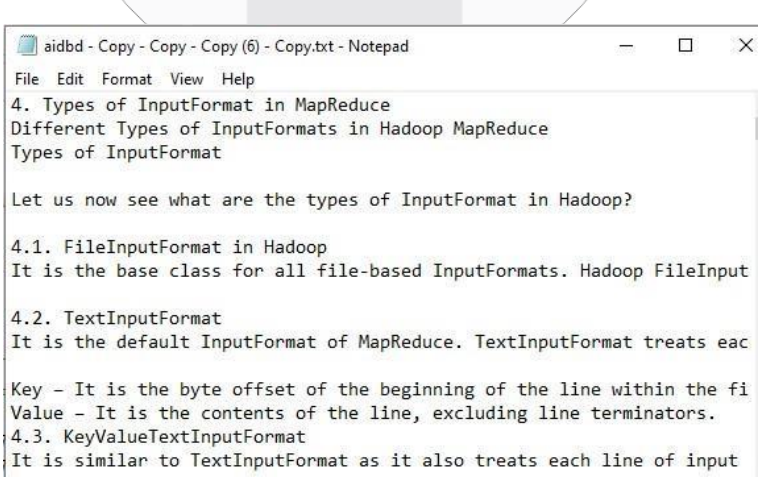


Gambar 3. *Combine File Input Format.*

Untuk menangani *file* kecil dalam jumlah banyak metode rancangan *system* yang dibangun ialah *Combine File Input Format. Combine file input format* menggunakan mapreduce, dalam sistem yang di bangun Input Format bertanggung jawab membaca file dan membuat *InputSplit*, kemudian *InputSplit* bertangung jawab melakukan pemisahan satu atau lebih input split. Satu *MapTask* dibuat untuk setiap *spliting*/pemisahan; dengan demikian jumlah *MapTask* akan sama dengan jumlah *InputSplits*. Kemudian pemisahan dibagi menjadi beberapa *record* dan setiap *record* akan diproses oleh *Maper*. *Record Reader* berkomunikasi dengan *InputSplit* di *Hadoop MapReduce* dan mengubah data menjadi pasangan *key-value* yang cocok untuk dibaca oleh *Maper*. *Record Reader* menggunakan *FileInputFormat* untuk mengonversi data menjadi pasangan *key-value.* Selanjutnya, pasangan *keyvalue* ini dikirim ke *Maper* untuk diproses lebih lanjut. *Maper* memproses setiap *Input record* dari *Record Reader* dan menghasilkan pasangan *key-value* baru, dan pasangan *key-value* yang dihasilkan oleh *Mapper* ini sama sekali berbeda dari pasangan *Input*. *Output mapper* juga dikenal sebagai *Intermediate Result* yang ditulis ke disk lokal. Kemudian *Output mapper* yang terletak di disk lokal akan di proses *reducer*, yang mana *Reducer* melakukan pemrosesan ringan seperti *agregasi* / penjumlahan dan akan mendapatkan *output* dari proses penjumlahan. *Output* dari *reducer* adalah *output* akhir, yang disimpan dalam HDFS.

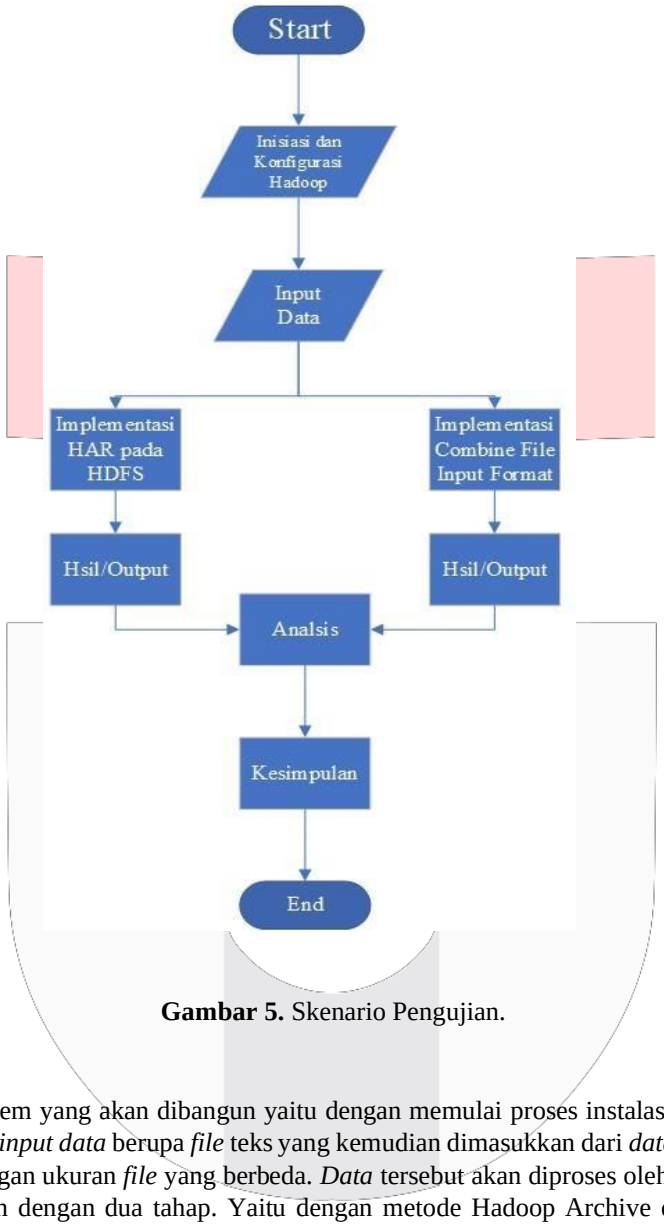
DataSet

Pertama, mulai menyiapkan *file* yang berisikan *dataset* yang ukuran *file* nya kecil namun dalam jumlah besar yang tidak lebih dari ukuran block HDFS. Dimana setiap *file diperoleh* dari internet dengan mengcopy file text. Berikut contoh file nya:



Gambar 4. *DataSet.*

Skenario pengujian



Gambar 5. Skenario Pengujian.

Perancangan sistem yang akan dibangun yaitu dengan memulai proses instalasi dan konfigurasi Hadoop. Setelah Hadoop terinstal, *input data* berupa *file* teks yang kemudian dimasukkan dari *data local* ke HDFS. Terdapat berbagai macam data dengan ukuran *file* yang berbeda. *Data* tersebut akan diproses oleh HDFS.
Pengujian akan dilakukan dengan dua tahap. Yaitu dengan metode Hadoop Archive dan metode Combine File Input Format berbasiskan HDFS. Kemudian dari metode yang digunakan akan menampilkan hasil yang nantinya setiap hasil akan di analisis dan dibandingkan. Setelah itu akan disimpulkan, metode manakah yang lebih baik digunakan untuk mengatasi file kecil dalam jumlah banyak.

Skenario pengujian Hadoop Archive

Beberapa langkah skenario pengujian yang harus dilakukan sebagai berikut ini: Yang pertama, kita harus menyiapkan *file-file* yang berukuran besar yang akan di uji dengan ukuran yang berbeda yakni 1,2 GB dan 2,3 GB dengan jumlah *file* sebanyak 2000 dan 4000 *file.* Setelah *file* sudah tersedia maka yang akan dilakukan ialah dengan *Start Hadoop* nya. Ketika Hadoop sudah berjalan maka *file* yang sudah disiapkan akan diinputkan ke dalam *hdfs*, yang mana *hdfs* berfungsi sebagai penyimpanan data. Kemudian file yang tersimpan di hdfs akan di proses dengan

mengeksekusi command line yang sudah disediakan oleh Hadoop, dengan perintah Hadoop Archive. Setelah itu file Hadoop archive akan dijadikan input , untuk diproses oleh *wordcount* dengan menggunakan perintah *Hadoop jar*.

Skenario pengujian Combine File Input Format

Beberapa langkah skenario pengujian Combine File Input Format yang harus dilakukan sebagai berikut ini: Yang pertama, kita harus menyiapkan *file-file* kecil berjumlah banyak dengan ukuran yang berbeda yakni 1,2 GB dan 2,3 GB dengan jumlah *file* sebanyak 2000 dan 4000 *file*. Setelah *file* sudah tersedia maka yang akan dilakukan ialah dengan *Start Hadoop* nya. Ketika Hadoop sudah berjalan maka *file* yang sudah disiapkan akan diinputkan ke dalam *hdfs*, yang mana *hdfs* berfungsi sebagai penyimpanan data. Kemudian file yang tersimpan di hdfs akan di proses dengan mengeksekusi command line yang sudah disediakan oleh Hadoop, dengan perintah Hadoop Jar. Kemudian akan didapatkan output, yang mana output ini akan di analisis.

Spesifikasi Sistem

Spesifikasi Sistem			
1	Laptop	:	1 Buah
2	Prosesor	:	Intel (R) Core (TM) i5-2500 CPU @ 3.30GHz
3	RAM	:	3 GB In Virtual Box Master, 2 GB Virtual Box Slave 1 dan 2 GB Virtual Box Slave 2.
4	OS	:	Ubuntu-18.04.6-Dekstop-amd64
5	Apache Hadoop	:	3.2.1 Multi Node

Table 1. Spesifikasi System.

4. Evaluasi

4.1 Hasil Pengujian

Pengujian Single Node

Perbandingan waktu pemrosesan *file* Combine File Input Format dan HadoopArchive + WordCount				
Pengujian Ke-	HAR+WordCount File Input 1,1 GB Mapper = 1, Reducer= 1.	CFIF File Input 1,1 GB Mapper = 1, Reducer= 1.	HAR+ WordCount File Input 2,3 GB Mapper = 1, Reducer= 1.	CFIF File Input 2,3 GB Mapper = 1, Reducer = 1.
1	96s+990 = 1086	1000s	225s+1801 = 2026	1931s
2	97s+980 = 1077	931s	210s+1820 = 2030	2003s
3	104s+971 = 1075	912s	202s+1810 = 2012	1964s
4	100s+970 = 1070	945	199s+1807 = 2006	1866s
5	89s+975 = 1064	1145s	228s+1830 = 2058	1993s
Rata-Rata	1074.4s	986.6s	2026.4s	1951.4s

Table 2. Pengujian *SingleNode.*

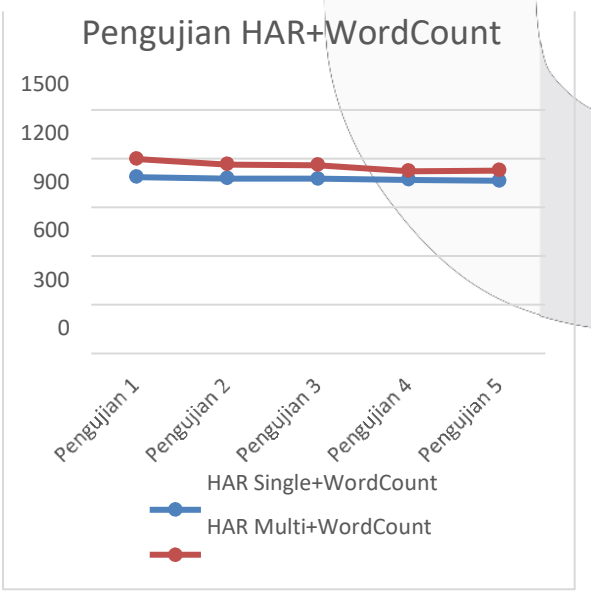
Pengujian Multi Node

Perbandingan waktu pemrosesan file Combine File Input Format dan HadoopArchive+ WordCount				
Pengujian Ke-	HAR+WordCount File Input 1,1 GB Mapper = 1, Reducer= 1.	CFIF File Input 1,1 GB Mapper = 1, Reducer= 1.	HAR+ WordCount File Input 2,3 GB Mapper = 1, Reducer= 1.	CFIF File Input 2,3 GB Mapper = 1, Reducer= 1.
1	118s+1080 = 1198	1040s	223s+1910 = 2133	1914s
2	114s+1050 = 1164	1038s	221s+1970 = 2191	2001s
3	100s+1060 = 1160	1022s	207s+2020 = 2227	2252s
4	103s+1019 = 1122	973s	217s+1900 = 2117	2198s
5	117s+1011 = 1128	917s	254s+2010 = 2264	2126s
Rata-Rata	1154.4s	998s	2186.4s	2098.2s

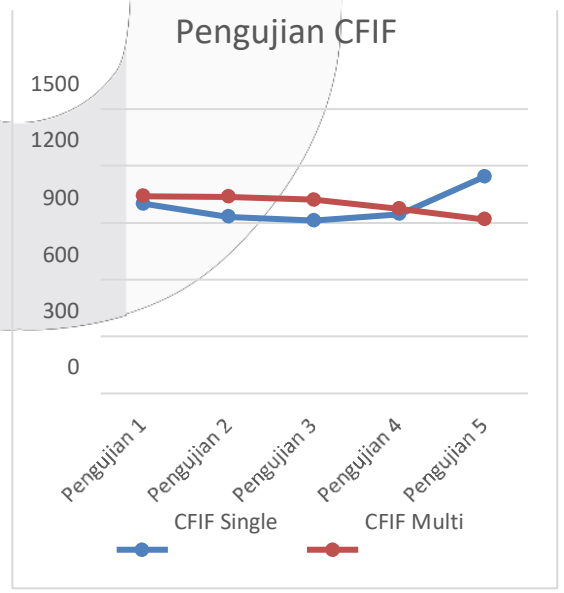
Table 3. Pengujian *MultiNode*.

Dapat dilihat pada table 3. hasil pengujian waktu pemrosesan *file* Hadoop MapReduce antara metode Hadoop Archive+WordCount dengan *Combine File Input Format* menggunakan *singlenode* dan *multinode*. Masing masing menggunakan jumlah *mapper* 1 dan *reducer* 1. Pengujian dilakukan 5 kali dan diambil rata rata *respone time* dari setiap metode.

4.2 Analisis Hasil Pengujian

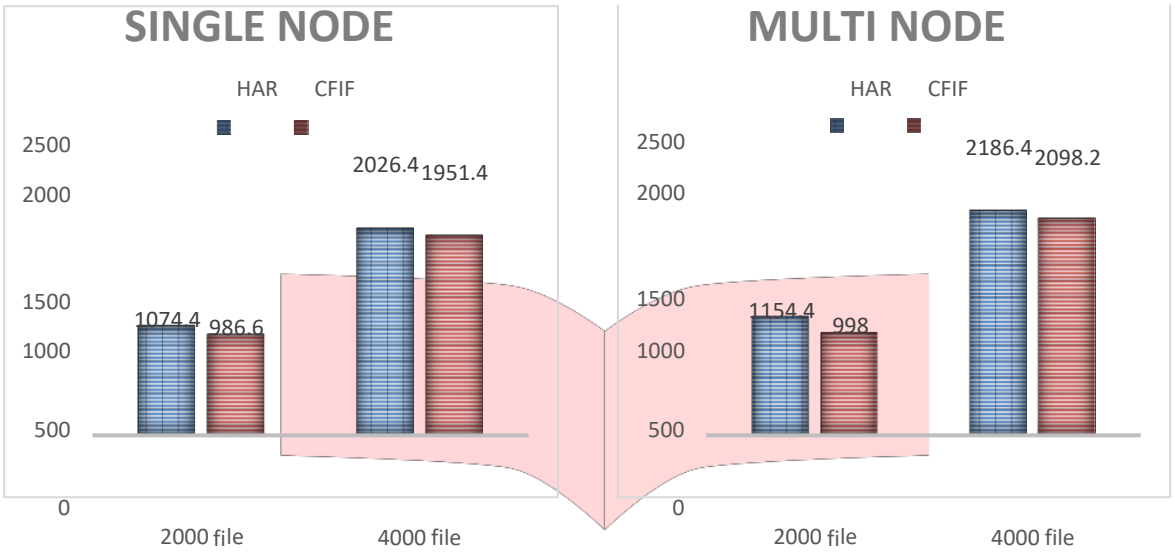


Gambar 6. Hasil Pengujian HAR.



Gambar 7. Hasil Pengujian CFIF.

Dapat di lihat dari pengujian gambar 6. diamana waktu yang dibutuhkan dalam memproses *file* untuk *single node* dan *multi node* tidak jauh berbeda, namun lebih tinggi penggunaan waktu *multi node*, itu disebabkan karena penggunaan *mapper* hanya 1. Kemudian untuk pengujian pada gambar 7. diamana waktu yang dibutuhkan dalam memproses *file* untuk *single node* dan *multi node* tidak jauh berbeda, namun lebih tinggi penggunaan waktu *multi node* itu disebabkan karena penggunaan *mapper* hanya 1 dan didalam grafik terdapat waktu yang tidak stabil itu disebabkan karena disk dan cpu.



Gambar 8. Hasil *SingleNode.*

Gambar 9. Hasil *MultiNode*.

Berdasarkan Gambar 8. dan Gambar 9. Waktu pemrosesan *file* yang digunakan *Hadoop Archive+WordCount* lebih banyak karena *Hadoop Archive* dijadikan *input* untuk diproses oleh *WordCount* sehingga terjadi tambahan waktu*/Overhead.*. *WordCount* melakukan penghitungan jumlah kata yang sama sama dengan *Combine File Input Format* menghitung jumlah kata seingga pemrosesan waktu yang dibutuhkan cukup lama. Untuk *Storage Combine File input Format* dan *WordCount* sama sama sedikit karena kata yang sama akan dihitung jumlahnya tanpa menulis ulang kata yang sama. *Mapper* yang digunakan masing masing metode sama sama menggunkan 1 *mapper* dan 1 *reducer*. Untuk 2000 *file* dengan metode *Hadoop Archive+WordCount* penggunaan blok sebanyak 1 blok dengan ukuran 128MB *blok size*, untuk 4000 *file* sebanyak 1 *blok* dengan ukuran 128MB *blok size*. Sedangkan Untuk 2000 *file* dengan metode *Combine File Input Format* penggunaan *blok* sebanyak 1 *blok* dengan ukuran 128 *blok size*, untuk 4000 *file* sebanyak 1 *blok* dengan ukuran 128 *blok size*. Untuk *hadoop archive* dan *combine file input format* memiliki *output* dari *file input* yang sudah di proses, berikut contoh output dari *Hadoop archive+WordCount* dan *combine file input format*:

```
"   10
"274    8
"3  6
"AS 16
"Battle 2
"Bila   2
"Crash"?    2
"Cuties"    12
"Edukasi    2
"Fall   2
"Food   4
"Fortnite"  2
"Hello  16
"Hidup  8
"Ingat  2
"Ini    2
"Jadi   2
"KPPU   2
"Kami   12
"Kecewa 2
"Langit 8
```

```
"Left   4
"License"); 32
"Lockdown"  12
"Microsoft  4
"Miss   16
"Mulan" 48
"NASA   16
"Ninja" 4
"Pas-pasan" 4
"Pemakan    16
"Saat   16
"Saatnya    16
"Satgas:    4
"Saya   4
"Sayangilah 16
"Sign   4
"Super  4
```

Gambar10. *Output* CFIF.

Gambar 11. *Output* HAR+*WordCount*.

5. Kesimpulan

Dari hasil pengujian dapat disimpulkan bahwa penggunaan *Hadoop Archive+WordCount* dan *Combine File Input Format* sangatlah tepat untuk memproses *file kecil* karena kedua *metode* ini dapat mengatasi *file* kecil di hdfs. Tetapi untuk waktu pemrosesan *file Hadoop Archive+WordCount* lebih lama dibandingkan dengan *Combine File Input Format*, yang menyebabkan *Hadoop archive+WordCount* lebih lama adalah karena *Hadoop archive+WordCount* memiliki tambahan waktu pada HAR yang dijadikan input yang akan diproses oleh *WordCount. WordCount* dan *Combine File Input Format* sama sama menghitung jumlah kemunculan kata. Untuk penggunaan *blok* sama sama menggunakan 1 *blok* karena *Hadoop archive+WordCount* dan *Combine File Input Format* melakukan penjumlahan kata yang sama sehingga *blok* yang digunakan sedikit.

Referensi

[1] I. Cholissodin and E. Riyandani, ANALISIS BIG DATA (Teori & Aplikasi) “Big Data vs Big Information vs Big Knowledge” Versi 1.01, Malang: FILKOM UB, 2016.

[2] D. Dev and R. Patgiri, "HAR+: Archive and Metadata Distribution! Why Not," *International Conference on Computer Communication and Informatics (ICCCI)*, p. 1, 2015.

[3] T. White, FOURTH EDITION Hadoop: The Definitive Guide, United States of America: O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, April 2015.

[4] P. Phakade and S. Raut, "An Innovative Strategy for Improved Processing of Small Files in Hadoop," *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, vol. 3, no. 7, p. 279, July 2014.

[5] C. Vorapongkitipun and N. Nupairoj, "Improving Performance of Small-File Accessing," *International Joint Conference on Computer Science and Software Engineering (JCSSE)*, p. 203, 2014.

[6] "The Small Files Problem. https://blog.cloudera.com/the-small-files-problem/".

[7] P. A. T. Taqwin, A. B. Osmond and R. Latuconsina, "IMPLEMENT OF MAPREDUCE METHOD ON BIG DATA BASED ON HADOOP," *ISSN*, vol. 5, no. 1, p. 1014, 2018.

[8] D. Permatasari, M. Dani and M. Rosmiati, "Analisis Performansi Hadoop Cluster Multi Node pada Komputasi Dokumen".

[9] A. Mohanty, P. Ranjana and D. V. Subramanian, "Small Files Consolidation Technique in Hadoop Cluster," *International Journal of Simulation: Systems*, p. 1474, December 2018.

[10] M. A. Ahad and R. Biswas, "Handling Small Size Files in Hadoop: Challenges, Opportunities, and Review," *Springer*, p. 659, 2019.

[11] S. Bende and R. Shedge, "Dealing with Small Files Problem in Hadoop Distributed File," *ScienceDirect*, pp. 1006-1010, 2016.