IMPLEMENTASI METODE MAPREDUCE PADA BIG DATA BERBASIS HADOOP DISTRIBUTED FILE SYSTEM

IMPLEMENT OF MAPREDUCE METHOD ON BIG DATA BASED ON HADOOP DISTRIBUTED FILE SYSTEM

[1]Pandu Akas Tachli Taqwin, [2]Andrew Brian Osmond, [3]Roswan Latuconsina

[123]Program Studi S1 Sistem Komputer, Fakultas Teknik Elektro, Universitas Telkom
[1]oliviandisevian@gmail.com, [2]abosmond@telkomuniversity.ac.id, [3]roswan78@gmail.com

Abstrak

Teknologi *Big data* merupakan kumpulan data dalam skala besar, yang mempunyai karakteristik data yang variatif, sangat cepat pertumbuhannya dan kompleks datanya. Data yang kompleks merupakan data yang tidak terstruktur yang perlu diolah khusus dengan suatu infrastruktur yang dapat mengelola data dalam volume besar berjalan secara realtime. Untuk itu diperlukan suatu metode yaitu Mapreduce, guna memudahkan komputasi yang akan dilakukan pada *big data*. Mapreduce digunakan untuk melakukan komputasi kumpulan data yang terdapat pada *Hadoop Distributed File System* (HDFS). Metode Mapreduce dapat ditransformasi dengan berbagai bentuk. Dengan Apache Flink metode Mapreduce dapat dikaji kembali pada arsitektur yang berbeda. Pada tugas akhir ini pengelolaan data berupa data yang tidak terstruktur dalam bentuk teks. Merancang Aplikasi HDFS pada sistem operasi linux dan mengimplementasikan metode Mapreduce. Program mapreduce yaitu berupa program penghitung jumlah kata menggunakan fungsi yang terdapat pada Apache Flink. Pada penelitian ini, Flink Mapreduce dapat melakukan komputasi lebih cepat sekitar 37,18% dari Hadoop Mapreduce.

Keyword : Big Data, Hadoop Apache, MapReduce, Apache Flink

Abstract

Big data technology is a collection of data on a large scale, which has the characteristics of data varied, very fast growth and complex data. Complex data is unstructured data that needs to be specially processed with an infrastructure that can manage large volumes of data running in realtime. For that required a method that is Mapreduce, in order to facilitate the computation to be performed on the big data. Mapreduce is used to compute data sets contained in the Hadoop Distributed File System (HDFS). The Mapreduce method can be transformed in various forms. With Apache Flink the Mapreduce method can be reviewed on different architectures. In this final project data management is unstructured data in the form of text. Designing HDFS Applications on Linux operating systems and implementing Mapreduce methods. The mapreduce program is a number of word counting program using the function found in Apache Flink. In this study, Flink Mapreduce can perform faster computation of about 37.18% of Hadoop Mapreduce.

Keyword: Big Data, Hadoop Apache, MapReduce, Apache Flink

1. Pendahuluan

1.1 Latar Belakang

Perkembangan teknologi semakin pesat dan layanan yang disediakan semakin banyak. Terdapat layanan yang berkembang sangat cepat yaitu teknologi *internet*. Dimana dengan *internet* semua perangkat elektronik yang mempunyai alamat *Internet Protocol* maka dapat saling terkoneksi. Layanan *software* dapat digunakan oleh setiap pengguna secara berbayar ataupun gratis. Semakin berkembangnya teknologi pada *software* yang kompleks menyebabkan konsumsi data penyimpanan semakin besar. Oleh karena itu, Segala data yang terdapat pada suatu aplikasi pada *internet* akan berkembang dari data yang kecil kemudian menjadi data yang besar.

Data dalam skala besar yang disebut dengan *Big Data*. *Big Data* adalah kumpulan data yang sangat besar, sangat variatif, sangat cepat pertumbuhannya dan mungkin tidak terstruktur. Proses komputasi yang terjadi pada *big data* dapat berjalan lambat apabila komputer yang digunakan untuk memproses data tersebut tidak memenuhi standar yang dibutuhkan oleh suatu *big data*. Maka, diperlukan suatu algoritma khusus sehingga informasi yang mendalam mudah didapatkan dan dapat membantu pengambilan keputusan yang lebih baik [1]. Solusi untuk *Big Data* yaitu Hadoop. Hadoop adalah *framework open source* di bawah lisensi Apache untuk mensupport aplikasi yang jalan pada *Big Data*. Asal mula Hadoop muncul karena terinspirasi dari makalah tentang *Google MapReduce* dan *Google File System (GFS)* yang ditulis oleh ilmuwan dari Google, *Jeffrey Dean* dan *Sanjay Ghemawat* pada tahun 2003. Penamaan menjadi Hadoop adalah diberikan oleh *Doug Cutting*, yaitu berdasarkan nama dari mainan gajah anaknya [2].

Hadoop dijalankan pada lingkungan yang menyediakan *storage* dan komputasi secara terdistribusi atau bisa disebut sebagai *Hadoop Distributed File System* (HDFS). *Hadoop* mendistribusikan klaster-klaster dari komputer/*node* menggunakan suatu model pemrograman. Mapreduce adalah paradigma pemrograman yang berjalan di latar belakang Hadoop untuk menyediakan skalabilitas dan mudah solusi pengolahan data [3]. Pengolahan data dapat pada sebuah data yang terstruktur, semi-terstruktur, dan tidak terstruktur [4].

Dengan mengganti disk yang tidak efisien dengan cache memori *low latency*, *high-throughput* yang terdistribusi untuk mengoptimalkan proses pengiriman data dari tugas *map* ke tugas *reduce* [5]. Untuk memenuhi kebutuhan tersebut proses komputasi menggunakan fungsi yang terdapat pada Apache Flink.

Apache Flink yaitu salah satu *platform open source* yang dapat digunakan untuk merancang program Mapreduce dengan *in-memory batch processing* dan *stream processing* [6]. Ada dua API inti di Flink: API DataSet untuk memproses kumpulan data yang terbatas (sering disebut sebagai pemrosesan *batch*), dan API DataStream untuk memproses data *stream* yang tak terbatas (sering disebut sebagai pemrosesan *stream*) [7]. Pada penelitian ini, akan dilakukan pengolahan data pada sebuah data yang tidak terstruktur dalam bentuk teks. Mengimplementasikan metode Mapreduce *batch processing* berbasiskan HDFS yang dijalankan pada sistem operasi linux.

1.2 Tujuan

Tujuan pembuatan Tugas Akhir ini adalah:

a) Merancang aplikasi HDFS untuk mengolah kumpulan data pada *Big Data*.
b) Implementasi metode Mapreduce berbasis HDFS.
c) Menganalisa response time dan penggunaan resource dari sistem berupa *memory*, *processor*, serta *disk* pada metode Mapreduce.
d) Mengkaji efisiensi metode Hadoop Mapreduce dengan metode Mapreduce *in-memory batch processing* Apache Flink.

1.3 Identifikasi Masalah

Rumusan masalah dalam pembuatan Proposal Tugas Akhir ini adalah rendahnya efisiensi dalam pengaksesan suatu data pada *big data* dan proses komputasi metode Hadoop Mapreduce memerlukan waktu yang lama. Terdapat kumpulan data dalam skala besar dan beragam variasinya untuk dikelola. Data yang bertumbuh secara cepat (*up to date*).

2. Dasar Teori

2.1 Big Data

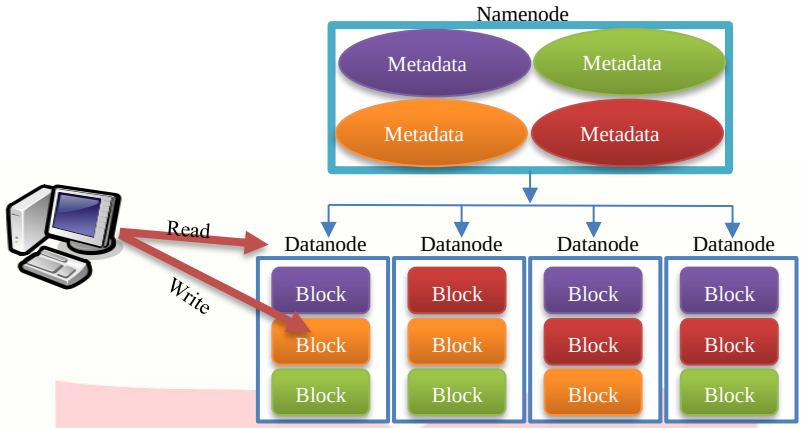
Big data adalah kumpulan data dalam volume besar yang terdiri dari data terstruktur dan data tidak terstruktur. *Big Data* dapat dianalisis untuk wawasan yang mengarah pada pengambilan keputusan dan strategi bisnis yang lebih baik. Dimulai pada tahun 2000-an ketika seorang analis industri Doug Laney menyampaikan konsep Big Data yang terdiri dari tiga bagian penting, diantaranya: *Volume*, yaitu kumpulan data dengan ukuran yang besar dan tidak dapat diolah dengan satu computer. Data ini tidak dapat diproses dengan cara tradisional karena membutuhkan waktu yang sangat lama. *Velocity* (kecepatan), dengan adanya data yang besar maka kumpulan data akan mengalir dengan cepat dan belum pernah terjadi pada teknologi sebelumnya yang harus ditangani dengan tepat waktu. *Variety* (variasi), terdapat data dalam berbagai tiper format yaitu: data terstruktur, *numeric* data pada basis data tradisional dan data yang tidak terstruktur.

2.2 Apache Hadoop

Apache Hadoop adalah suatu *software platform* yang dibangun dengan Bahasa pemrograman Java untuk menghubungkan beberapa komputer sehingga dapat saling bekerja sama dan sinkron dalam menyimpan dan mengolah data sebagai satu kesatuan. Apache Hadoop didukung dari empat komponen yaitu: Hadoop Distributed File-System (HDFS), Mapreduce, Hadoop Common, *framework* Yet Another Resource Negotiator (YARN). YARN merupakan sebuah *framework* yang mengatur jadwal pekerjaan (job scheduling) yaitu program mapreduce serta *resource management* pada klaster [9].

2.3 Arsitektur HDFS

Berikut adalah arsitektur dari Hadoop Distributed File System:



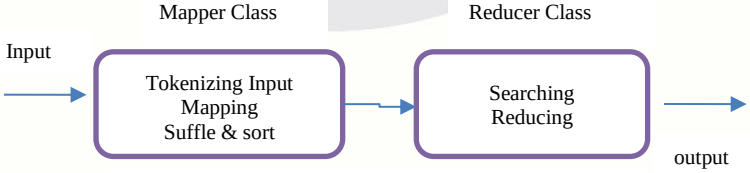
Gambar 1 Arsitektur HDFS [10].

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 [10]: Mengelola sistem *file* namespace, mengatur akses klien ke *file*, mengeksekusi operasi sistem *file* seperti mengubah nama, menutup, dan membuka file dan direktori [10]. *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 pada file sistem, sesuai permintaan *client*. Melakukan operasi seperti pembuatan blok, penghapusan, dan replikasi sesuai dengan petunjuk dari *Name Node* tersebut[10]. Block merupakan data pengguna yang disimpan dalam *file* dari HDFS. *File* dalam sistem *file* akan dibagi menjadi satu atau lebih segmen atau disimpan di *node data* individual. Segmen file ini disebut sebagai Blok. Dengan kata lain, jumlah minimal data yang HDFS dapat baca atau menulis [10]. Ukuran Blok *default* adalah 64MB [11], tetapi dapat ditingkatkan sesuai kebutuhan untuk mengubah konfigurasi HDFS.

2.4 Metode Mapreduce

Model pemrograman MapReduce digunakan untuk memproses secara efisien kumpulan data yang besar secara paralel *MapReduce* terdiri atas tiga tahap, yaitu tahap *map*, tahap *shuffle*, dan terakhir tahap *reduce*. Untuk tahapan *shuffle* dan *reduce* digabungkan kedalam satu tahap besaran-nya yaitu tahap *reduce*. Pemrogram dapat menyelesaikan aplikasi MapReduce dengan menerapkan dua fungsi: fungsi Map dan fungsi Reduce. Selanjutnya, kedua dari dua fungsi memiliki <key, value> pair sebagai input dan outputnya [11].

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. Berikut ini ilustrasi untuk mendapatkan gambaran tentang proses *map* dan *reduce*.



Gambar 2 Mapper & Reducer [10]

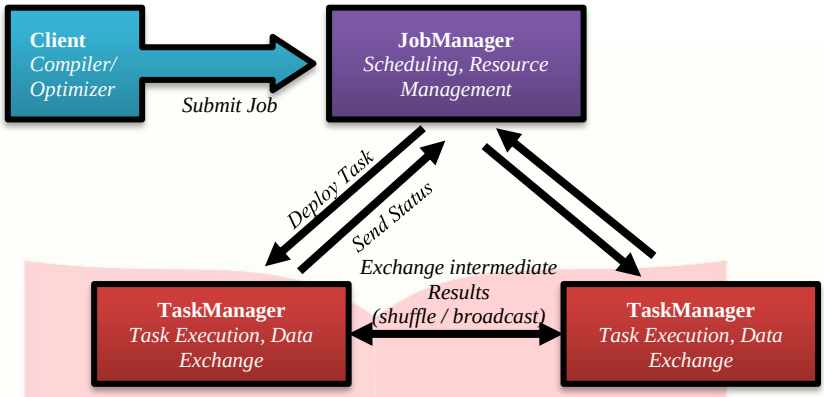
Mapper Class mengambil input, *tokenizes* menjadi pasangan kunci nilai, memetakan dan melakukan penyortiran. *Output* dari *Mapper Class* digunakan sebagai input oleh *Reducer Class*, yang kemudian dikelompokkan sesuai kunci nilai yang sama.

2.5 Apache Flink

Apache flink adalah *platform* yang dapat melakukan suatu komputasi pada *big data*. Flink dapat melakukan *read* atau *write data* dari berbagai macam sistem penyimpanan. Inti dari Flink adalah mesin *streaming dataflow* yang menyediakan distribusi *data*, komunikasi *data*, dan toleransi kesalahan untuk perhitungan terdistribusi melalui aliran *data*. Flink membangun pemrosesan *batch* diatas mesin *streaming*, tersedia untuk iterasi *overlay native*, pengelolaan *memory*, dan pengoptimalan program [12].

Apache flink dapat melakukan proses Mapreduce seperti yang digunakan pada Hadoop. Pengimplementasian metode Mapreduce pada Apache Flink, Hadoop Mappers ekuivalen dengan fungsi FlatMap. Dan Hadoop Reducer ekuivalen dengan fungsi GroupReduce. Gambar diatas menunjukkan arsitektur metode Mapreduce yang digunakan menggunakan fungsi yang ada pada program Flink. Apache flink dapat dijalankan diatas beberapa macam *file system*, misalnya pada HDFS [13]. Berikut adalah arsitektur dari Apache Flink:

Gambar 3 Arsitektur Apache Flink[12].



JobTracker digunakan untuk memasukkan tugas MapReduce ke cluster, ia akan mengalokasikan tugas ke cluster yang memiliki data bahwa tugas tersebut memerlukan prioritas tertinggi untuk memaksimalkan kinerjanya. Ketika klien mengirimkan pekerjaan ke JobTracker, ia akan berbicara ke NameNode untuk menemukan lokasi data, maka ia dapat menentukan node mana yang paling dekat dengan data dan mengirimkan tugas ke node TaskTracker yang dipilih. Ketika TaskTracker menyelesaikan tugas, JobTracker akan mendapatkan sinyal dan memberikan tugas lain untuknya. Ini berjalan di mesin NameNode secara default [14]. Yang kedua adalah TaskManager atau disebut *worker* yang menjalankan tugas dari aliran data, dan menyangga dan menukar *data stream*.

Setiap TaskTracker memiliki satu set slot, masing-masing slot menerima sebuah tugas. Ketika JobTracker mencoba menemukan TaskTacker untuk menjalankan tugas MapReduce, pertama-tama mencari slot kosong pada server yang sama yang menampung DataNode yang berisi data, dan jika tidak, ia mencari slot kosong pada mesin di rak yang sama. TaskTracker menjalankan pekerjaan sebenarnya dalam proses JVM yang terpisah, jika tugas mogok, tidak akan menurunkan TaskTracker. Saat tugas berjalan, TaskTracker dapat memantau proses ini dan menyimpan output dan nilai pengembaliannya. Setelah menyelesaikan tugas, ia mengirimkan sinyal untuk memberi tahu JobTracker [14].

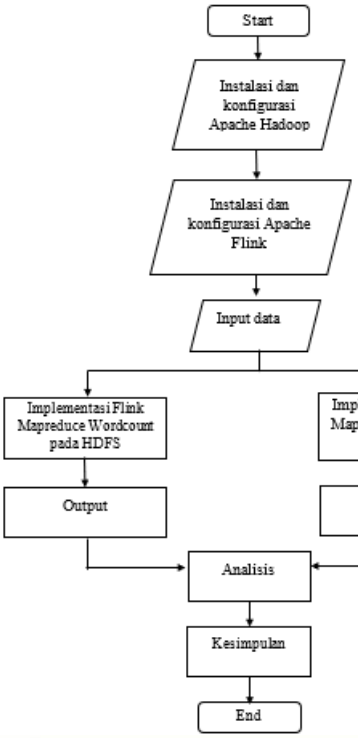
3. Perancangan dan Implementasi Sistem

3.1 Gambaran Umum

Pada penelitian ini perancangan sistem yang akan dibangun dengan memulai proses instalasi dan konfigurasi HDFS. Dilanjutkan instalasi dan konfigurasi Apache Flink. Kemudian setelah proses konfigurasi dan instalasi selesai maka diperlukan untuk mengatur environment Hadoop pada Apache Flink agar metode Mapreduce dapat dijalankan pada HDFS.

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. Pengujian dilakukan dimulai dari *data* yang berukuran kecil kemudian bertahap ke *data* yang berukuran besar. *Data* tersebut akan diproses menggunakan program Hadoop Mapreduce pada HDFS.

Pengujian akan dilakukan dengan dua tahap. Yaitu dengan metode Mapreduce pada Hadoop menggunakan paradigma *disk-based* dan metode Mapreduce pada *flink in-memory batch processing* berbasiskan HDFS. Aplikasi ini berjalan secara lokal atau *standalone*.



Gambar 4 Gambaran Umum Sistem

3.2 Implementasi Metode Mapreduce

Pada tahap ini akan menjelaskan tentang pengimplementasian program Mapreduce. Program Mapreduce akan dijalankan pada dua *environment* secara terpisah. Analisis akan dilakukan setelah program mapreduce mendapat output dari masing-masing *environment* yang digunakan. Berikut skema implementasi program Mapreduce. Pada Hadoop *environment* program Mapreduce dijalankan pada HDFS. Yang pertama *client* harus lakukan yaitu mendistribusikan data pada HDFS untuk memulai program Mapreduce. Program Mapreduce menggunakan Bahasa java yang berisikan implementasi dari fungsi Mapper dan Reducer. Program Mapreduce yang akan dieksekusi diterima oleh Jobtracker. Tugas Jobtracker yaitu mendistribusikan perangkat lunak atau konfigurasi pada pekerja, *job scheduling*, dan memonitor kinerja Mapreduce yang kemudian memberikan informasi kepada *client*. Pada Tasktracker bertugas untuk implementasi Mapreduce dan menghasilkan *output* yang akan disimpan pada *file system*.

3. Pengujian dan Analisis

Pada penelitian ini akan ada beberapa skenaraio pengujian yang dilakukan seperti dibawah ini:

1) *Input data* berupa file berisi teks yang mempunyai ukuran berbeda, yaitu 1,6 GB dan 2,5GB
2) Pengujian dengan *single node cluster*.
3) Pengujian waktu respon metode Mapreduce Wordcount *disk-based* pada HDFS.
4) Pengujian waktu respon metode Mapreduce Wordcount Apache Flink berbasis HDFS.

Skenario yang diujikan yaitu waktu respon (*response time*) dan sumber daya (*resource*) yang digunakan pada saat melakukan komputasi dan performa pada komputer.

4.1 Pengujian

Pengujian ini dilakukan untuk mengetahui kecepatan dari pemrosesan data, dan sumber daya yang digunakan. Pada bagian ini yang di uji ada dua *file* dengan *size* yang berbeda-beda dimulai dari *size* yang kecil, sedang, hingga besar dengan file ekstensi .txt dan .csv seperti pada daftar di bawah ini:

1. Facebook-names-unique size 1,6 GB
2. Facebook-names-original size 2,5 GB

4.1.1 Hasil Pengujian Waktu Respon Mapreduce pada HDFS

Berikut merupakan hasil data pengujian untuk waktu respon kedua *file* diatas, yaitu: file berukuran 1,6 GB dan 2,5 GB dengan menggunakan Hadoop Mapreduce *disk-based* pada HDFS. Pengujian dilakukan sebanyak lima kali dan diambil hasil rata-rata.

Tabel 1 Pengujian Pengujian waktu respon Hadoop Mapreduce

Pengujian Hadoop Mapreduce		
Pengujian ke-	Waktu pengerjaan (satuan detik)	
	File 1,6 GB	File 2,5 GB
1	290.751	376.930
2	289.804	384.687
3	284.844	386.865
4	276.440	393.851
5	284.525	387.548
Rata-rata	285.273	385.976

Dari hasil pengujian pada Tabel 1 diatas maka dapat disimpulkan bahwa pengujian pada *file* yang berukuran lebih kecil, untuk melakukan komputasi pada Hadoop Mapreduce mendapatkan waktu yang cepat. Semakin besar ukuran *file* maka proses komputasi akan semakin lama. Kemudian dilanjutkan pengujian Flink Mapreduce pada HDFS.

Tabel 2 Pengujian *Respon Time* Flink Mapreduce

Pengujian Flink Mapreduce		
Pengujian ke-	Waktu pengerjaan (satuan detik)	
	File 1,6 GB	File 2,5 GB
1	178.399	264.775
2	179.361	264.225
3	177.802	265.043
4	179.824	266.769
5	180.613	267.754
Rata-rata	179.200	265.713

Dari hasil pengujian pada Tabel 2 diatas dapat disimpulkan pengujian pada *file* berukuran 1,6 GB membutuhkan waktu sekitar 179,2 detik. Sedangkan untuk *file* berukuran 2,5 GB membutuhkan waktu sekitar 265.713 lebih lama. Dari tabel 4.3diatas membuktikan bahwa *in-memory based processing* Flink Mapreduce pada HDFS dapat berlangsung secara sangat cepat apabila dibandingkan dengan hasil pengujian tabel 4.1. Serta output yang dihasilkan oleh Flink Mapreduce lebih stabil. Hal ini dibuktikan dengan hasil standar deviasi Flink Mapreduce lebih kecil dari pada Hadoop Mapreduce.

4.1.2 Penggunaan Sumber Daya

Pengujian ini dilakukan untuk mengetahui seberapa besar kebutuhan *hardware* khususnya *memory* dalam melakukan pemrosesan *data* untuk mengetahui manakah yang lebih banyak menggunakan sumber daya. Terdapat beberapa komponen yang akan diamati penggunaan sumber daya. Pada Hadoop Mapreduce menggungunakan HDFS dan Yarn. Pada saat menjalankan HDFS terdapat komponen yang aktif yaitu: Datanode, Namenode, dan Secondary Namenode. Kemudian Yarn sebagai mesin dari Hadoop Mapreduce maka akan terdapat komponen yang aktif yaitu: Resourcemanager dan Nodemanager. Pada Flink mapreduce hanya membutuhkan HDFS dan komponen pada Flink yaitu Taskmanager dan Jobmanager.

Tabel 3 Pengujian Performa Hadoop Mapreduce

Penggunaan sumber daya Flink Mapreduce								
	File 1,6 GB				File 2,4 GB			
		Disk kB/s				Disk kB/s		
No.	CPU	Read	Write	Memory	CPU	Read	Write	Memory
1	91.38 %	10343.25	836.80	56.60 %	89.91 %	11088.34	748.51	63.20 %
2	90.93 %	10292.18	362.17	55.60 %	89.76 %	11324.37	694.34	63.50 %
3	90.72 %	10532.05	818.51	62.01 %	89.67 %	11449.95	663.15	64.40 %
4	90.61 %	10526.34	1174.51	62.60 %	89.59 %	11090.00	1009.80	64.90 %
5	90.63 %	10465.14	1217.20	63.20 %	89.51 %	11188.95	897.40	64.70 %
Avg	90.85 %	10431.79	881.84	60.00 %	89.69 5	11228.32	802.64	64.14 %

Dari Tabel 3, adalah hasil pengujian performa Hadoop Mapreduce pada masing-masing *file* yang berbeda ukuran. Pada *file* pertama penggunaan sumber daya *memory* relatif kecil sekitar 29,66 % dan sumber daya oleh *processor* dikonsumsi dengan angka yang tinggi. Dari hasil pengujian tersebut *processor* bekerja lebih daripada percobaan *file* pertama dan penggunaan sumber daya *memory* lebih kecil. Pada percobaan kedua, peningkatan terjadi pada penggunaan memory. Yaitu mengalami peningkatan sebesar 0.66%. Kemudian pengujian dilanjutkan *pada in-memory batch* Flink Mapreduce. Pada saat Apache Flink dijalankan maka kondisi *memory* akan mengalami perubahan. Penggunaan sumber daya lebih ditekankan pada *memory* dan *processor*. Setiap proses yang berjalan, grafik *processor* dan *memory* tidak terlalu mengalami banyak perubahan. Tetapi jika dilihat dari proses baca dan tulis pada *disk* mengalami penurunan dibandingkan dengan proses yang dilakukan oleh Hadoop Mapreduce. Penggunaan *memory* dan *processor* akan menyentuh titik maksimal pada saat pemrosesan *file* berukuran yang sangat besar. Apabila *file* berukuran kecil dan *memory heap* masih sanggup untuk menyediakan sumber daya maka, penggunaan *memory* dan *processor* tidak terlalu besar. Berikut data hasil pengujian dalam tabel.

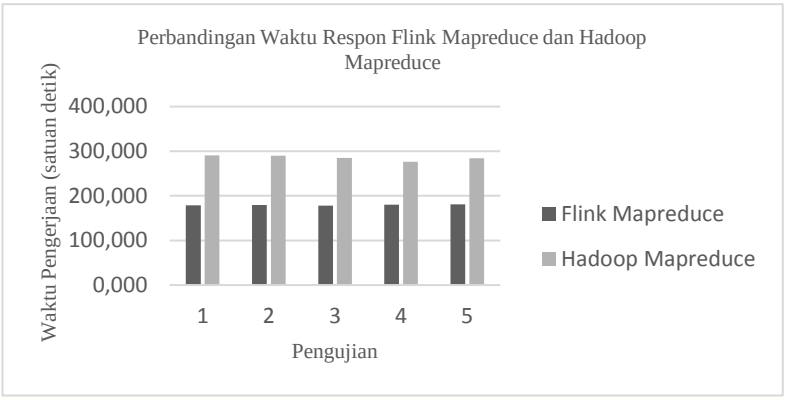
Tabel 4 Pengujian Performa Flink Mapreduce

Penggunaan sumber daya Hadoop Mapreduce								
	File 1,6 GB				File 2,4 GB			
		Disk kB/s				Disk kB/s		
No.	CPU	Read	Write	Memory	CPU	Read	Write	Memory
1	88.43 %	12764.19	5074.16	27.70 %	88.56 %	10966.68	3565.44	30.50 %
2	92.14 %	10715.80	4198.36	30.60 %	83.18 %	10861.00	3646.78	30.00 %
3	91.93 %	10585.68	4002.51	30.40 %	92.49 %	11837.47	3506.04	30.70 %
4	92.18 %	9595.20	3511.28	29.70 %	90.67 %	11025.31	3495.42	30.40 %
5	91.72 %	10857.88	3723.16	29.90 %	90.16 %	11241.53	3518.01	30.00 %
Avg	91.28 %	10903.75	4101.89	29.66 %	89.01	11186.40	3546.34	30.32 %

Dari Tabel 4 diatas menampilkan hasil pengujian performa *in-memory batch* Flink mapreduce berbasis HDFS pada *file* yang telah diuji. Pada Flink Mapreduce untuk file berukuran kecil yang memenuhi kapasitas dari *heap memory* yang telah dikonfigurasi maka tidak memerlukan sumber daya *processor* yang tinggi. Jika *file* yang diproses melebihi dari *heap memory* yang dikonfigurasi maka persentase sumber daya *processor* akan meningkat.

4.2 Analisis

Analisis yang pertama adalah analisis hasil pengujian respon waktu terhadap proses yang dilakukan oleh metode Mapreduce berbasis HDFS. Metode Mapreduce tersebut diterapkan pada dua platform. Yang pertama pada Hadoop, dan kemudian diterapkan pada Apache Flink dengan beberapa tambahan fungsi transformasi yang terdapat pada Apache Flink. Metode mapreduce yang digunakan diimplementasikan dalam bentuk program *wordcount*. Dilakukan pengujian secara *standalone* atau *local node*. Setelah pengujian dilakukan terbukti Hadoop Mapreduce dapat melakukan komputasi secara *disk-based*. Dan Apache Flink menyediakan komputasi secara *in-memory batch*. Flink mapreduce dapat melakukan komputasi lebih cepat daripada Hadoop Mapreduce. Pada pengujian Hadoop Mapreduce performa dan sumber daya yang dibutuhkan tidak bisa ditentukan secara pasti. Karena grafik yang didapat selalu mengalami fluktuasi. Untuk pengujian pada Flink Mapreduce, grafik lebih stabil sehingga hasil yang didapat memiliki akurasi yang bagus. Perbedaan kecepatan dapat terlihat secara signifikan pada saat pengujian *file* dalam ukuran yang sangat besar, yaitu diatas 1GB. Berikut grafik perbandingannya.



Gambar 5 Grafik analisis waktu respon

Dari Gambar 5 dapat disimpulkan bahwa pengimplementasian metode Mapreduce pada Flink dapat mempersingkat waktu komputasi. Waktu komputasi metode Mapreduce meningkat sekitar 37,18%. Hal ini karena Mapreduce pada Flink mempunyai metode yang berbeda dalam pengolahan file. Pada Tabel 3 dan 4 terbukti proses baca tulis yang terjadi pada disk pada saat melakukan Mapreduce pada Hadoop. Proses baca tulis pada Hadoop Mapreduce lebih besar dibandingkan Flink Mapreduce. Hal ini yang menyebabkan Hadoop Mapreduce lebih lambat. Untuk proses baca tulis pada Flink Mapreduce, yaitu dengan cara mengirim mikro batch data kedalam memori secara bertahap dan terus-menerus. Mikro batch data tersebut dikirim seperti streaming data. Oleh karena itu proses baca tulis pada Flink Mapreduce dapat dilakukan secara bersamaan dan dapat menghemat waktu pemrosesan.

5. Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan dapat disimpulkan bahwa metode Mapreduce dapat diimplementasikan dalam berbagai macam bentuk. Apache Flink menyediakan fungsi transformasi yang digunakan untuk membuat metode Mapreduce seperti Hadoop dengan sistem kerja yang sama. Arsitektur pada Apache Flink memungkinkan Mapreduce untuk melakukan pengiriman Near-realtime micro-batch data ke memori sehingga proses yang dilakukan untuk komputasi dapat berjalan dengan cepat. Kecepatan yang dapat diraih oleh metode Mapreduce pada Apache Flink lebih cepat dua kali dari Hadoop Mapreduce. Untuk mendapatkan hasil yang maksimal diperlukan spesifikasi computer server dengan memori yang besar dan hardisk yang banyak.

Daftar Pustaka:

[1] D. Prasetyo, "idioticus!," MEMAHAMI BIG DATA, 5 May 2017. [Online]. Available: http://www.idioticus.com. [Diakses 15 September 2017].
[2] P. Gohil, D. Garg dan P. B. Panchal, "A Performance Analysis of MapReduce Applications on Big Data in Cloud based Hadoop," *ICICES2014 - S.A.Engineering College, Chennai, Tamil Nadu, India,* 2014.
[3] R. S. K. V. dan D. N. P. Kavya, "Big Data Processing with harnessing Hadoop - MapReduce for Optimizing Analytical Workloads," *IEEE,* 2014.
[4] A. Shinnar, D. Cunningham dan B. Herta, "M3R: Increased Performance for InMemory," *Proceedings of the VLDB Endowment,* 2012.
[5] S. Zhang, J. Han, Z. Liu, K. Wang dan S. Feng, "Accelerating MapReduce with Distributed Memory Cache," *15th International Conference on Parallel and Distributed Systems,* 2009.
[6] M. Junghanns, A. Petermann, N. Teichmann, K. Gómez dan E. Rahm, "Analyzing Extended Property Graphs with Apache Flink," *ISBN,* 2016.
[7] P. Carbone, S. Ewen, S. Haridi, A. Katsifodimos, V. Markl dan K. Tzoumas, "Apache Flink: Stream and Batch Processing in a Single Engine," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering,* 2015.
[8] "Big Data: What it is and why it matters," SAS, [Online]. Available: https://www.sas.com/en_us/insights/big-data/what-is-big-data.html. [Diakses 27 11 2017].
[9] "Apache Hadoop," [Online]. Available: hadoop.apache.org. [Diakses 20 October 2017].
[10] "w3ii.com," Hadoop Panduan Singkat, [Online]. Available: http://www.w3ii.com. [Diakses 15 September 2017].
[11] X. Wu, "A MapReduce Optimization Method on Hadoop Cluster," *International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information,* 2015.
[12] S. Baltagi, Overview of Apache Flink: Next-Gen Big Data, Chicago: Chicago Apache Flink Meetup, 2015.
[13] V. Markl, "Breaking the Chains: On Declarative Data Analysis and Data Independence in the Big Data Era," *Proceedings of the VLDB Endowment,* 2014.
[14] P. M. R. Lyu, Cloud computing technologies and applications, Hong Kong: Department of Computer Science and Engineering, CUHK, 2011.
[15] R. Ho, "How Hadoop Map/Reduce works," Dzone / Big Data Zone, 8 December 2016. [Online]. Available: https://dzone.com/articles/how-hadoop-mapreduce-works. [Diakses 7 December 2017].
[16] K. Tzoumas, "Apache Flink," 2015. [Online]. Available: https://www.slideshare.net/KostasTzoumas. [Diakses 4 December 2017].