IMPLEMENTASI APACHE SPARK PADA BIG DATA BERBASIS HADOOP DISTRIBUTED FILE SYSTEM

IMPLEMENTATION APACHE SPARK ON BIG DATA BASED HADOOP DISTRIBUTED FILE SYSTEM

[1]Sevian Oliviandi, [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

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.

Pada tugas akhir ini digunakan metode MapReduce untuk memudahkan komputasi yang akan dilakukan pada suatu big data. MapReduce merupakan suatu model pemrograman untuk menulis aplikasi yang dapat memproses suatu big data secara paralel pada beberapa node. MapReduce memberikan kemampuan analitis untuk menganalisis volume besar data yang kompleks. Platform yang digunakan adalah Hadoop, Hadoop mempunyai algoritma MapReduce sendiri.

Tugas akhir ini akan menganalisis performa dari Hadoop MapReduce dan membandingkannya dengan Apache Spark yaitu platform yang dibuat untuk memproses suatu big data yang dikembangkan berdasarkan Hadoop MapReduce dengan peningkatan performa pemrosesan. Skenario yang digunakan adalah memproses wordcount suatu data dengan besar data yang berbeda yang bertujuan untuk menganalisis response time dan penggunaan hardware dari kedua platform tersebut.

Kata kunci : Big Data, Apache Spark, MapReduce, Hadoop

Abstract

Big data 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.

In this final project used MapReduce method to facilitate computation to be performed on a big data. MapReduce is a programming model for writing applications that can process a big data in parallel on multiple nodes. MapReduce provides analytical capabilities for analyzing large volumes of complex data. The platform used is Hadoop, Hadoop has its own MapReduce algorithm.

This final project will analyze the performance of Hadoop MapReduce and compare it with Apache Spark is a platform created to process a big data developed based on Hadoop MapReduce with improved processing performance. The scenario used is to process wordcount of a data with different data that aims to analyze the response time and hardware usage of both platforms.

Keywords: Big Data, Apache Spark, MapReduce, Hadoop

1. Pendahuluan

1.1 Latar Belakang

Seiring dengan berkembangnya zaman big data merupakan suatu yang menjadi trend dalam dunia informasi. Bisa dibilang big data merupakan kumpulan data yang sangat besar yang di dalamnya mencakup berbagai jenis data. Big Data menjadi kata yang popular seiring dengan bagaimana dapat menyimpan data dalam jumlah yang besar, melakukan proses serta analisa. Sesuatu yang tidak dapat dihindari bagaimana impact dari big data ini dalam kehidupan sehari-hari. Big Data telah memberikan kesempatan atau peluang bisnis bagi banyak perusahaan. Hampir semua industri telah memanfaatkan atau baru melakukan identifikasi tentang pentingnya big data dalam menumbuhkan bisnisnya atau tetap dapat bersaing bahkan menjadi keunggulan dalam berkompetisi [1].

Dari sekian banyak manfaat dan peluang, big data dapat meninggalkan beberapa tantangan diantaranya adalah tantangan teknologi yang dapat menghandle big data ini, tantangan skill atau keahlian orang yang akan mengolah data sehingga data yang tersedia dapat menjadi informasi, insight yang bermanfaat. Dalam dunia akademik, istilah big data mengacu pada aplikasi teknologi informasi untuk menangani masalah data yang sifatnya besar [1].

Guna mengatasi masalah data yang terus bertambah besar pada tugas akhir ini dibuatlah sebuah sistem yang dapat memproses big data. Metode yang digunakan untuk memproses big data tersebut yaitu MapReduce, MapReduce adalah model pemrograman untuk menulis aplikasi yang dapat mengolah data besar. MapReduce memberikan kemampuan analitis untuk menganalisis volume besar data yang kompleks [2]. Algoritma MapReduce berisi dua tugas penting yaitu Mapper dan Reducer. Pada proses Mapper data yang masuk diurai berdasarkan jenisnya sehingga dapat dengan mudah dipilih, kemudian masuk proses Reducer, pada proses ini data dikelompokkan berdasarkan tipe data yang sama kemudian keluarlah output hasil data yang sudah diproses. MapReduce sendiri mempunyai banyak platform misalnya yang penulis pakai di sini adalah Apache Spark, Apache Spark adalah teknologi komputasi klaster kilat-cepat, dirancang untuk perhitungan cepat. Hal ini didasarkan pada Hadoop MapReduce dan memperluas model MapReduce menggunakannya secara efisien untuk lebih banyak jenis perhitungan, yang mencakup query interaktif dan pengolahan aliran. Fitur utama dari Apache Spark adalah di memori klaster komputasi yang meningkatkan kecepatan pemrosesan aplikasi [3].

1.2 Tujuan

Tujuan pembuatan jurnal ini adalah:

a. Mengimplementasikan platform Apache Spark yang berjalan pada Hadoop Distributed File System secara standalone sebagai alternatif dari Hadoop MapReduce dalam memproses suatu big data.
b. Menganalisa pemrosesan big data menggunakan platform Apache Spark dan membandingkannya dengan Hadoop MapReduce dalam hal kecepatan atau response time.
c. Menganalisa penggunaan resource dari sistem memory, processor, serta disk dalam menjalankan Apache Spark dan Hadoop MapReduce untuk memproses big data.

1.3 Identifikasi Masalah

Rumusan masalah dalam jurnal ini adalah sebagai berikut. Pertama adalah lamanya proses dalam memproses data pada suatu big data, hal ini dikarenakan besarnya data itu sendiri. Kemudian keanekaragaman jenis data dalam suatu big data, data dalam suatu big data sangat bervariasi misalnya data media sosial, data transaksi keuangan, dan lain sebagainya. Oleh sebab itu diperlukannya suatu aplikasi yang dapat mengolah big data tersebut.

2. Dasar Teori

2.1 Big Data

Big Data merupakan istilah yang menggambarkan suatu *volume* data yang besar, baik yang struktur maupun tidak terstruktur. *Big Data* telah digunakan dalam banyak bisnis. Tidak hanya besar data yang menjadi poin utamanya, tetapi apa yang harus dilakukan organisasi dengan besar data tersebut. *Big Data* dapat dianalisis untuk wawasan yang mengarah pada pengambilan keputusan dan strategi bisnis yang lebih baik [4].

Istilah big data masih terbilang baru dan sering disebut sebagai tindakan pengumpulan dan penyimpanan informasi yang besar untuk analisis. Fenomena big data dimulai pada tahun 2000-an ketika seorang analisis industri Doug Laney menyampaikan konsep big data yang terdiri dari tiga bagian penting , diantaranya:

Volume: Organisasi mengumpulkan dari berbagai sumber, termaksud transaksi bisnis, media sosial dan informasi dari sensor atau mesin. Di masa lalu, aktivitas semacam ini menjadi masalah, namun dengan adanya teknologi baru (seperti Hadoop) bisa meredakan masalah ini [4].

Kecepatan: Aliran data harus ditangani dengan cepat dan tepat bisa melalui hardware maupun software. Teknologi hardware seperti tag RFID, sensor pintar lainnya juga dibutuhkan untuk menangani data yang real-time.

Variasi: Data yang dikumpulkan mempunyai format yang berbeda-beda. Mulai dari yang terstruktur, data numerik dalam database tradisional, data dokumen terstruktur teks, email, video, audio, transaksi keuangan dan lain-lain.

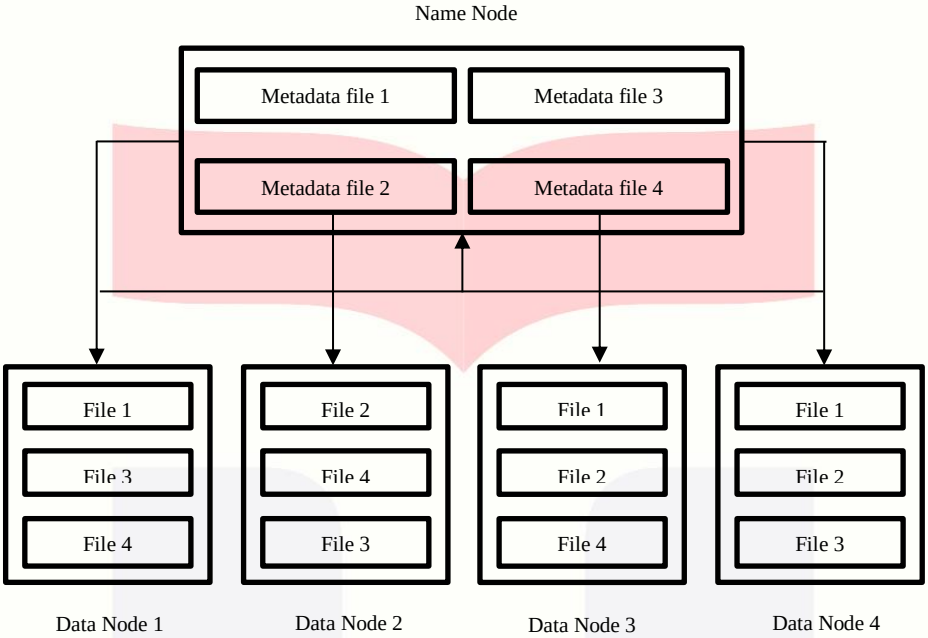
2.2 Hadoop

Big Data mulai jadi trend teknologi saat ini. Salah satu software platform yang bisa digunakan untuk mengelola big data adalah Hadoop. Secara ringkas Hadoop adalah software yang mampu menghubungkan banyak komputer untuk dapat bekerja sama dan saling terhubung untuk menyimpan dan mengelola data dalam satu kesatuan. Hadoop menyimpan dan mengolah big data menggunakan model pemrograman MapReduce. MapReduce adalah model pemrograman rilisan Google yang bisa digunakan untuk memproses data dalam ukuran besar secara terdistribusi dan paralel dalam cluster yang terdiri dari komputer berjumlah ribuan [5].

2.3 Hadoop Distributed File System

Hadoop Distributed File System (HDFS) merupakan sistem penyimpanan terdistribusi, yang melakukan proses pemecahan file besar menjadi bagian-bagian lebih kecil kemudian didistribusikan ke cluster-cluster dari komputer. Cluster ini biasanya terdiri dari banyak node atau komputer atau server. Setiap node di dalam cluster ini harus terinstal Hadoop untuk bisa berfungsi. Sebagai distributed file system, HDFS berguna untuk menangani data berukuran raksasa yang disimpan tersebar dalam clusternya [6].

Sebagai distributed file system, HDFS menyimpan suatu data dengan cara membaginya menjadi potongan-potongan data yang misalnya berukuran 64 MB, dan potongan-potongan data ini kemudian disimpan tersebar dalam komputer-komputer yang membentuk clusternya. Potongan-potongan data tersebut dalam HDFS disebut block, dan ukurannya tidak terpaku harus 64 MB misalnya. Ukuran block dapat diatur sesuai kebutuhan [6]



Gambar 1 Penyimpanan data pada HDFS [6]

HDFS memiliki komponen-komponen utama berupa NameNode dan DataNode. NameNode adalah sebuah komputer yang bertindak sebagai master, sedangkan DataNode adalah komputer-komputer dalam Hadoop Cluster yang bertugas sebagai slaves atau anak buah. NameNode bertanggung jawab menyimpan informasi tentang penempatan block-block data dalam Hadoop Cluster. Ia bertanggung jawab mengorganisir dan mengontrol block-block data yang disimpan tersebar dalam komputer-komputer yang menyusun Hadoop Cluster. Sedangkan DataNode bertugas menyimpan block-block data yang dialamatkan kepadanya, dan secara berkala melaporkan kondisinya kepada NameNode [6]

2.4 MapReduce

MapReduce merupakan model pemrograman untuk menulis suatu aplikasi yang dapat memproses big data secara paralel pada beberapa node. MapReduce mempunyai kemampuan untuk menganalisis volume data yang kompleks dan besar [1].

Algoritma MapReduce berisi dua tugas penting, yaitu Map dan Reduce [1].

1. Tugas dari Map adalah mengambil sekumpulan data dan mengubahnya menjadi kumpulan data lainnya, dimana masing-masing elemen dipecah menjadi tupel (key-value).

2. Tugas Reduce adalah mengambil output dari Map sebagai masukan dan menggabungkan data tupel tersebut (key-value) ke dalam kumpulan tupel yang lebih kecil.

Reduce selalu dilakukan setelah tugas dari Map selesai, Di bawah ini merupakan fase dari MapReduce [1].

1. Input Phase: Di sini ada Record Reader yang menerjemahkan setiap record dalam file input dan mengirimkan data yang diurai ke mapper dalam bentuk pasangan key-value.

2. Map: Fungsi yang ditentukan pengguna, yang mengambil serangkaian pasangan key-value dan memproses masing-masing untuk menghasilkan pasangan key-value bernilai nol atau lebih.

3. Intermediate Keys: Pasangan key-value yang dihasilkan oleh mapper dikenal sebagai interdiate key.

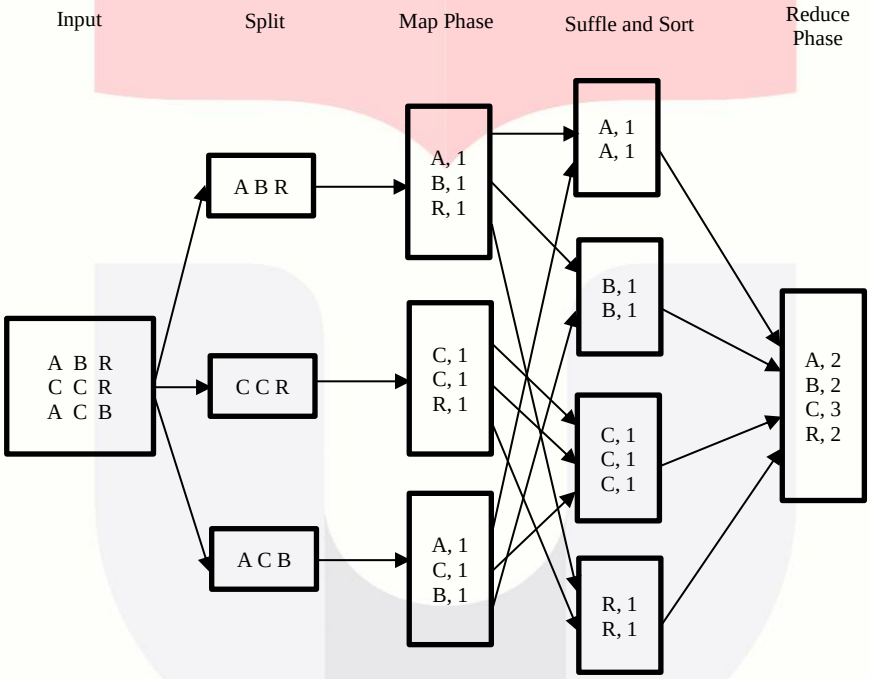
4. Combiner: Sejenis local Reducer yang mengelompokkan data serupa dari fase map ke dalam kumpulan data yang dapat diidentifikasi. Dibutuhkan Intermediate Key dari mapper sebagai masukan dan menerapkan kode yang ditentukan pengguna untuk mengumpulkan value dalam lingkup kecil satu mapper. Ini bukan bagian dari algoritma utama MapReduce atau opsional.

5. Suffle and Sort: Tugas Reducer dimulai dengan langkah Shuffle dan Sort. Ini akan mengambil pasangan key-value ke local machine, tempat dimana reducer sedang berjalan. Pasangan individual key-value diurutkan berdasarkan kuncinya ke daftar data yang lebih besar. Daftar data mengelompokkan equivalent key bersama sehingga nilainya dapat diulangi dengan mudah dalam tugas Reducer.

6. Reducer: mengambil data pasangan key-value berkelompok sebagai masukan dan menjalankan fungsi Reducer pada masing-masingnya. Di sini data dapat digabungkan, disaring, dan digabungkan dalam beberapa cara, dan ini memerlukan banyak pemrosesan. Setelah eksekusi selesai, ia memberikan nilai nol atau lebih pasangan key-value ke langkah terakhir.

7. Output Phase: Pada tahap output, kita memiliki formatter keluaran yang menerjemahkan pasangan key-value akhir dari fungsi Reducer dan menuliskannya ke file menggunakan record writer.

Pada gambar 2 merupakan contoh diagram bagaimana cara kerja MapReduce.



Gambar 2 Diagram kerja MapReduce [1]

2.5 Apache Spark

Apache Spark merupakan teknologi komputasi cluster yang cepat, yang dirancang untuk perhitungan cepat. Hal ini didasarkan pada Hadoop MapReduce dan memperluas model dari MapReduce untuk efisiensi lebih banyak jenis perhitungan, yang mencakup query interaktif dan stream processing. Fitur utama Apache Spark adalah komputasi cluster di memory yang meningkatkan kecepatan pemrosesan aplikasi. Apache Spark dirancang untuk mencakup berbagai macam beban kerja seperti batch aplication, iterative algorithms, interactive queres dan streaming [2].

Di bawah ini merupakan penjelasan fitur dari Apache Spark [2].

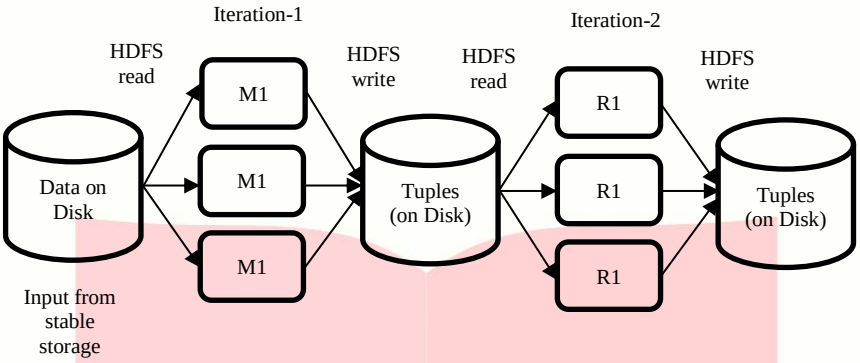
1. Speed: Apache Spark membantu menjalankan aplikasi di cluster Hadoop hingga 100 kali lebih cepat dalam memory, dan 10 kali lebih cepat saat menjalankan disk. Hal ini memungkinkan karena dengan mengurangi jumlah operasi baca atau tulis ke disk dengan menyimpan data pemrosesan di memory.

2. Supports Multiple Languages: Apache Spark menyediakan built-in API di Java, Scala, atau Python. Karena itu aplikasi dapat ditulis dalam baasa yang berbeda.

3. Advanced Analytics: Apache Spark tidak hanya mendukung Map dan Reduce. Apache Spark juga mendukung SQL queries, Streaming Data, Machine learning (ML), dan Graph algorithms.

Apache Spark bekerja dengan cara menyimpan semua proses iterasi ke dalam memory, bukan ke dalam disk seperti halnya MapReduce. Apache Spark mempunyai kecepatan 100 kali lebih cepat di dalam memory di bandingkan MapReduce. Di bawah ini merupakan ilustrasi perbandingan proses iterasi antara Apache Spark dengan Mapreduce ketika mengolah data.

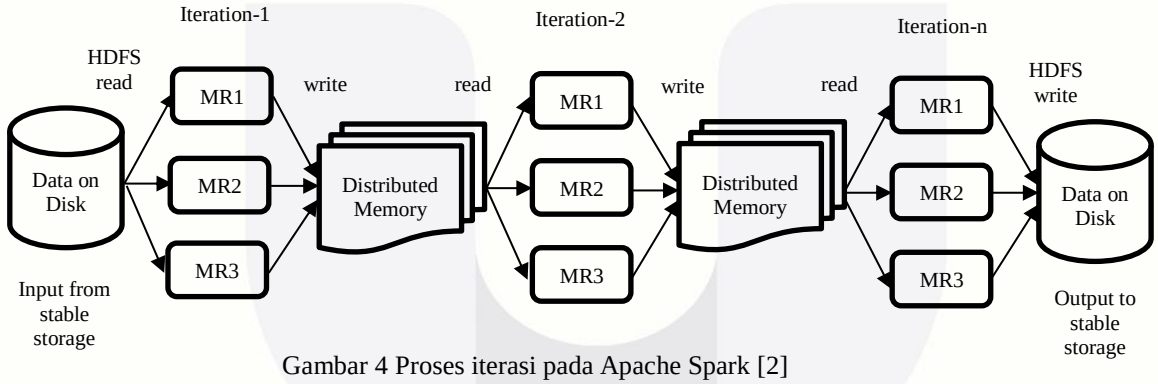
1. Iterasi pada Mapreduce



Gambar 3 Proses iterasi pada MapReduce [2]

Dapat dilihat pada ilustrasi gambar 3 bahwa setelah melakukan baca dan tulis data maka hasilnya akan dikembalikan ke disk, hal inilah yang menyebabkan menjadi lambat karena beban dari disk bertambah.

2. Iterasi pada Apache Spark



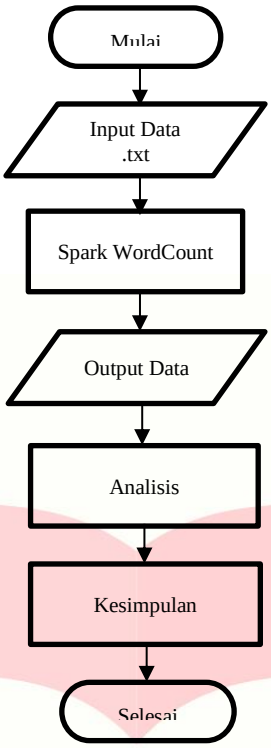
Gambar 4 Proses iterasi pada Apache Spark [2]

3. Perancangan dan Implementasi Sistem

Perancangan sistem dibagi menjadi 4 bagian. Bagian yang pertama yaitu menyiapkan spesifikasi hardware yang akan digunakan, yang kedua adalah instalisasi dan konfigurasi Apache Hadoop, yang ketiga yaitu instalisasi dan konfigurasi Apache Spark, serta yang terakhir yaitu pengolahan data pada cluster dan analisisnya.

Perancangan sistem dimulai dengan menyiapkan spesifikasi hardware yang akan digunakan, hardware yang akan digunakan harus mempunyai spesifikasi yang mumpuni supaya cluster dapat berjalan dengan baik. Perancangan berikutnya adalah instalisasi Apache Hadoop, Apache Hadoop digunakan sebagai salah satu basis dari penyimpanan data dengan memanfaatkan HDFSnya selain dari penyimpanan local.

Berikutnya yaitu instalisai Apache Spark, Apache Spark digunakan sebagai pemroses dari cluster. Apache Spark bertugas untuk memproses wordcount yang akan digunakan untuk pengujian clusternya. Perancangan yang terakhir yaitu pengolahan data, data yang digunakan yaitu dokumen yang berformat .txt yang isinya kumpulan dari kata-kata yang nantinya akan di hitung perkata yang disebut wordcount dan output dari wordcount tersebut selanjutnya dianalisis. Ilustrasi dari gambaran umum sistem dapat dilihat pada gambar 5.



Gambar 5 Gambaran Umum Sistem

Setelah mendapatkan gambaran umum dari sistem maka dilakukan analisis untuk memenuhi kebutuhan sistem yang akan digunakan mulai dari spesifikasi hardware yang akan digunakan serta perangkat lunaknya, daftar dari spesifikasi sitem diantaranya adalah sebagai berikut:

Tabel 1 Spesifikasi Sistem

Spesifikasi Sistem			
1	Notebook		1 Buah
	Processor	:	Core i5 2,4 GHz Max Turbo Frequency 3,0 GHz
	RAM	:	4 GB
	OS	:	Ubuntu 16.04 LTS
2	Apache Hadoop	:	Versi 2.9.0
3	Apache Spark	:	Versi 2.2.1 Pre-built for Apache Hadoop 2.7 and Later

4. Pengujian dan Analisis

Untuk mengetahui performa dan response time sistem berbasis Apache Spark dan Hadoop MapReduce digunakan beberapa skenario pengujian seperti berikut ini:

1. Ukuran file input yang berbeda ukurannya, yakni 1,5 GB dan 2,5 GB.
2. Pengujian dengan single node cluster.
3. Pengujian wordcount dengan menggunakan Apache Spark dan Hadoop MapReduce.

Pengujian ini dilakukan untuk mengetahui kecepatan dari pemrosesan data serta manajemen performa dengan menggunakan Apache Spark dan Hadoop MapReduce dalam memproses wordcount.

Tabel 2 Hasil Pengujian Response Time

Perbandingan Response Time Hadoop MapReduce & Apache Spark					
Pengujian Ke	Hadoop MapReduce File Input 1,5 GB	Apache Spark File Input 1,5 GB	Pengujian Ke	Hadoop MapReduce File Input 2,5 GB	Apache Spark File Input 2,5 GB
1	811	198	1	1235	240
2	530	210	2	741	204
3	533	198	3	766	204
4	516	210	4	758	228
5	533	216	5	742	228
Rata-rata	584,6	206,4	Rata-rata	848,4	220,8

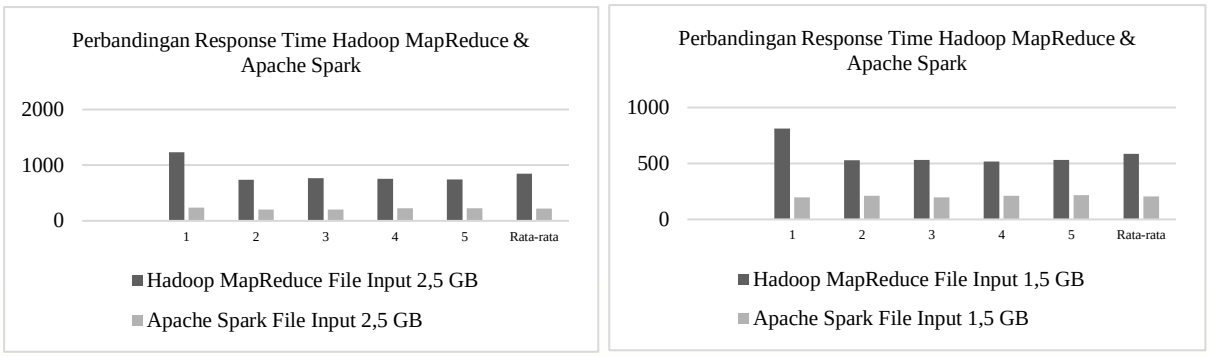
Tabel 3 Hasil Pengujian Performa Hadoop MapReduce

Pengujian Performa Hadoop MapReduce								
Pengujian Ke	File 1,5 GB				File 2,5 GB			
	Memory	Processor	Disk kB/s		Memory	Processor	Disk kB/s	
			Read	Write			Read	Write
1	12,90%	19,00%	2960	3076	11,10%	13,10%	8732	12288
2	11,80%	33,90%	9016	2899	11,30%	17,50%	4296	10556
3	9,60%	14,10%	7280	4589	8,40%	16,00%	11380	1972
4	8,70%	23,30%	8644	7988	8,00%	25,30%	3232	2720
5	8,70%	12,50%	7760	9257	7,60%	19,00%	7521	0
Rata-rata	10,34%	20,56%	7132	5561,8	9,28%	18,18%	7032,2	5507,2

Tabel 4 Hasil Pengujian Performa Apache Spark

Pengujian Performa Apache Spark								
Pengujian Ke	File 1,5 GB				File 2,5 GB			
	Memory	Processor	Disk kB/s		Memory	Processor	Disk kB/s	
			Read	Write			Read	Write
1	29,30%	92,50%	10588	0	34,00%	85,30%	15984	0
2	34,70%	77,25%	13920	0	33,90%	96,25%	8380	56
3	33,60%	97,75%	9556	0	34,20%	92,50%	12440	0
4	34,40%	96,25%	11848	0	34,40%	88,50%	8328	0
5	34,50%	55,50%	9376	8252	34,30%	78,00%	8712	0
Rata-rata	33,30%	83,85%	11057,6	1650,4	34,16%	88,11%	10768,8	11,2

Dapat dilihat pada tabel 2 hasil pengujian response time Hadoop MapReduce dan Apache Spark, dapat dilihat perbedaan selisih waktu Apache Spark lebih cepat rata-rata sekitar 300 sampai 600 seconds dari Hadoop MapReduce. Untuk lebih jelasnya perbedaan selisih waktu tersebut dapat dilihat pada gambar 6.



Gambar 6 Perbandingan Response Time Hadoop dan Apache Spark File 1,5 dan 2,5 GB

Dapat dilihat pada gambar 6 bahwa performa dari Apache Spark jauh lebih baik dalam mengimplementasikan pemrosesan suatu big data, hasil seperti di atas dapat tercapai karena pemrosesan pada Apache Spark berbasis memory. Data dari penyimpanan disk HDFS dibaca dan semua pemrosesan datanya di lakukan di memory hingga pemrosesan data tersebut selesai baru kemudian output ditulis ke dalam disk. Berbeda halnya dengan Hadoop MapReduce yang sebagian besar pemrosesannya di lakukan di dalam disk, hal ini lah yang menyebabkan pemrosesan dari Hadoop MapReduce menjadi lambat karena beban dari disk bertambah.

5. Kesimpulan

Dari hasil pengujian dapat disimpulkan bahwa penggunaan Apache Spark untuk memproses big data sangatlah tepat karena dapat menurunkan response time rata-rata 50% sampai 70% dari Hadoop MapReduce. Tetapi untuk mengoptimalkan kemampuannya dibutuhkan juga spesifikasi hardware yang mumpuni khususnya spesifikasi memory yang besar jika ingin mengoptimalkan kinerja dari Apache Spark.

Daftar Pustaka:

[1] “tutorialspoint,” MapReduce Turorial, [Online]. Available: https://www.tutorialspoint.com. [Diakses 16 November 2017].

[2] “tutorialspoint,” Apache Spark Tutorial, [Online]. Available: https://www.tutorialspoint.com. [Diakses 16 November 2017].

[3] K. B. Aryasa, “Komang B Aryasa Big Data Expert, Strategist & Practitioner,” Latar Belakang Big Data, 19 Januari 2015. [Online]. Available: http://komangaryasa.com. [Diakses 16 November 2017].

[4] Y. Permana, “CODEPOLITAN,” Mengenal Big Data, 29 Mei 2016. [Online]. Available: https://www.codepolitan.com. [Diakses 16 November 2017].

[5] g. “GamatechnoBlog,” Mengulas Lengkap Tentang Hadoop: Software Pengelolaan Big Data, 15 Juni 2017. [Online]. Available: https://blog.gamatechno.com. [Diakses 16 November 2017].

[6] S. Widy, “skyshi,” Hadoop Distributed File System, [Online]. Available: https://medium.com. [Diakses 16 November 2017].

[7] R. Athapathu, “Medium,” Installing Oracle Java 8 in Ubuntu, Maret 2015. [Online]. Available: https://medium.com. [Diakses 4 Desember 2017].

[8] S. “Data Flair,” Apache Spark Instalation in Standalone Mode, 2 Agustus 2016. [Online]. Available: https://data-flair.training. [Diakses 4 Desember 2017].

[9] J. Aven, Sams Teach Yourself Apache Spark in 24 Hours, USA, 2016.

[10] S. Penchikala, “InfoQ,” Big Data Processing with Apache Spark – Part 1: Introduction, 30 Januari 2015. [Online]. Available: https://www.infoq.com. [Diakses 24 November 2017].

[11] D. Team, “Chennai Hadoop User Group,” Spark Master/Slave installation in Multi Node, [Online]. Available: http://chennaihug.org. [Diakses 24 November 2017].