ISSN : 1411-1624
e-ISSN: 2621-8089

KOMPARASI KECEPATAN *HADOOP MAPREDUCE* DAN *APACHE SPARK* DALAM MENGOLAH DATA TEKS

Condro Wibawa[1], Setia Wirawan[2] , Metty Mustikasari[3], Dessy Tri Anggraeni[4]
Dosen Universitas Gunadarma[1,2,3,4]
Jalan Margonda No 100, Kota Depok, Jawa Barat
Sur-el : condro_wibawa@staff.gunadarma.ac.id[1], setia@staff.gunadarma.ac.id[2], metty@staff.gunadarma.ac.id[3], dessytri@staff.gunadarma.ac.id[4]

Abstract : *Today, the term of Big Data is nothing new. One of the Big Data's components is the massive amount of data, which makes data unable to be processed traditionally. To solve this problem, the Map Reduce method was developed. Map Reduce is a data processing method by breaking data into small parts (mapping) and then the results are put together again (reducing). The most widely used of Map Reduce frameworks are Hadoop MapReduce and Apache Spark. The concept of these two frameworks is the same but differs in the management of data sources. Hadoop MapReduce uses an HDFS (disk) approach, while Apache Spark uses RDD (in-memory). The use of RDD in Apache Spark makes this framework performance faster than Hadoop MapReduce. This is proven in this study, where to process the same text data, Apache Spark's average speed is 4.99 times faster than Hadoop MapReduce.*

Keywords: *Big Data, Map Reduce, Hadoop MapReduce, Apache Spark*

Abstrak : *Istilah Big Data saat ini bukanlah hal yang baru lagi. Salah satu komponen Big Data adalah jumlah data yang masif, yang membuat data tidak bisa diproses dengan cara-cara tradicional. Untuk menyelesaikan masalah ini, dikembangkanlah metode Map Reduce. Map Reduce adalah metode pengolahan data dengan memecah data menjadi bagian-bagian kecil (mapping) dan kemudian hasilnya dijadikan satu kembali (reducing). Framework Map Reduce yang banyak digunakan adalah Hadoop MapReduce dan Apache Spark. Konsep kedua framework ini sama akan tetapi berbeda dalam pengelolaan sumber data. Hadoop MapReduce menggunakan pendekatan HDFS (disk), sedangkan Apache Spark menggunakan RDD (in-memory). Penggunaan RDD pada Apache Spark membuat kinerja framework ini lebih cepat dibandingkan Hadoop MapReduce. Hal ini dibutktikan dalam penelitian ini, dimana untuk mengolah data teks yang sama, kecepatan rata-rata Apache Spark adalah 4,99 kali lebih cepat dibandingkan Hadoop MapReduce.*

Kata kunci*: Big Data, Map Reduce, Hadoop MapReduce, Apache Spark*

1. PENDAHULUAN

Penggunaan internet membuat jumlah data yang beredar di dunia tumbuh dengan cepat, pertumbuhan data rata-rata adalah 60% setiap tahunnya [1]. Menurut International Data Corporation (IDC) dalam laporannya yang berjudul "The Digitization of the World From Edge to Core", di tahun 2018 jumlah data di dunia adalah sekitar 33 Zettabytes (1 Zettabytes = 1012 Gigabytes) dan diperkirakan mencapai 175 Zettabytes di tahun 2025 [2]. Jumlah data yang masif ini menunggu untuk diolah, karena ke depannya kita akan hidup pada masyarakat *Informational Society* menuju *Knowledge Based Society*, sehingga data dan informasi akan memegang peranan yang sangat penting di berbagai bidang [1]. Kondisi ini membuat praktisi komputer berlomba-lomba menemukan metode untuk mengolah jumlah data yang begitu besar. Perusahaan IT raksasa, Google, memperkenalkan model pemrograman Map

ISSN : 1411-1624
e-ISSN: 2621-8089

Reduce pada tahun 2004. Konsep dasar *MapReduce* adalah dengan memetakan suatu proses/masalah oleh komputer master ke dalam beberapa sub-proses *(Mapping)*. Tiap sub-proses kemudian didistribusikan ke dalam komputer pekerja *(worker/slave*) di dalam suatu cluster dan diproses secara paralel. Hasil pemrosessan kemudian dikirim kembali ke komputer master dan dilakukan proses *Reducing (summarizing*) untuk memperoleh hasil akhir [3].

Model pemrograman ini kemudian dikembangkan oleh berbagai perusahaan seperti *Google, Amazon, Apache,* dan lain-lain. *Apache Hadoop* adalah salah satu *framework* yang menerapkan model pemrograman *Map Reduce* yang perkembangannya paling signifikan dan paling banyak digunakan. Mengutip laman resmi Apache (2016), dijelaskan bahwa *Apache Hadoop* memiliki dua fitur utama yaitu *Hadoop MapReduce* yang merupakan *framework MapReduc*e dan *Hadoop Distributed File System* (HDFS) yang merupakan *framework* untuk *distributed file syste*m [4]. *Hadoop MapReduce* sendiri berjalan di atas *Yarn Cluster* yang merupakan *Resource Manager* yang digunakan *Hadoop* untuk menangani *computer clustering*.

Menurut Xue (2015), setiap proses dalam *Hadoop MapReduce* tidak bisa dipisahkan dengan HDFS, dikarenakan hasil pemrosesan di tiap komputer pekerja *(slave)* akan direplikasikan ke HDFS agar bisa diproses lebih lanjut oleh komputer master. Proses replikasi yang terjadi antar komputer yang berbeda membutuhkan waktu yang relatif lama, dikarenakan harus melalui media I/O setiap terjadi transaksi. Selain itu penggunaan disk yang merupakan dasar dari HDFS juga turut memperlambat performa *Hadoop MapReduce*[5].

Melihat kelemahan ini, Apache mengeluarkan *framework MapReduce* yang bernama Apache Spark. Mengutip dari laman resmi Apache (2016), *Apache Spark* menawarkan teknologi *Resilient Distributed Datasets* (RDDs) untuk mendukung proses *Map* dan *Reducing* secara lebih efektif dan cepat [6]. RDDs bekerja pada in-memory, sehingga proses read dan write data jauh lebih cepat daripada *Hadoop MapReduce*. Laman resmi *Apache Spark* (2016) bahkan mengklaim bahwa *Apache Spark* 10-100 kali lebih cepat dibandingkan dengan *Hadoop MapReduce* [6]. Selain itu, Apache Spark juga bisa dijalankan pada *Spark Cluster, Yarn Cluster,* dan *Mesos Cluster*.

2. METODOLOGI PENELITIAN

Pada bagian ini akan dijelaskan mengenai langkah-langkah penelitian yang dilakukan. Dimulai dari penjelasan mengenai konsep dasar yang meliputi konsep *Map Reduce*, *Hadoop MapReduce*, dan *Apache Sp*ark. Langkah selanjutnya adalah konfigurasi *computer node* baik untuk infrastruktur *Hadoop MapReduce* maupun *Apache Spark*. Setelah itu dilakukan persiapan data uji, pembuatan program uji, dan evaluasi program.

2.1. Konsep Dasar

2.1.1 Map Reduce

Map Reduce adalah model pemrograman dan implementasi untuk memproses data

ISSN : 1411-1624
e-ISSN: 2621-8089

berukuran besar [7]. Sejalan dengan [7], [8] menjelaskan bahwa *Map Reduce* adalah model pemrograman untuk *data processing.* Momtaz [9] menyebutkan bahwa pengolahan data pada *Map Reduce* dilakukan secara terdistribusi dan paralel dalam sebuah *cluster*. Satu *cluster* bisa terdiri dari satu komputer *master* dan beberapa komputer *slave* (pekerja) atau hanya satu komputer yang bertindak baik sebagai komputer *master* dan komputer *slave*. Jumlah komputer *slave* tidak terbatas dan *hardware* yang digunakanpun tidak harus sama.

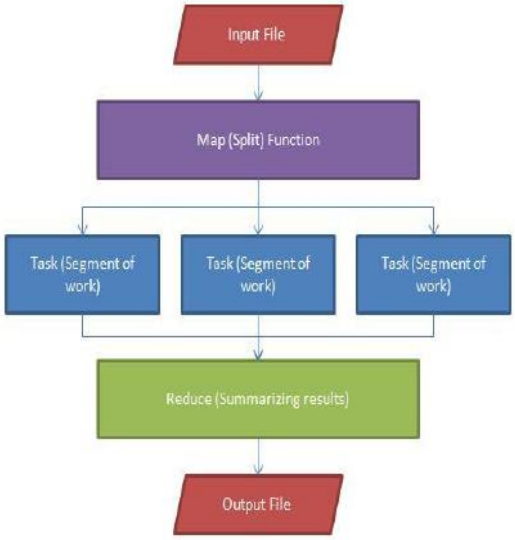
Google adalah yang pertama kali merilis model pemrograman ini untuk mengolah data berukuran raksasa (*big data*). Secara garis besar proses di dalam *Map Reduce* dibagi menjadi dua proses, yaitu proses *Map* dan proses *Reduce* [10]. Kedua proses ini didistribusikan ke semua komputer *slave* yang terhubung di dalam *cluster* dan berjalan secara paralel. Proses *Map* bertugas membagi masalah ke dalam sub-masalah yang lebih kecil dan mendistribusikannya ke komputer-komputer *slave*. Hasil proses di komputer *slave* akan dikumpulkan oleh komputer *master* melalui proses *Reduce*. Hasil dari proses *Reduce* inilah yang kemudian dikirimkan ke pengguna sebagai hasil akhir proses. Gambar 1 berikut adalah gambaran sederhana proses *Map Reduce*.

Fungsi *Map* bertugas untuk membaca input dalam bentuk pasangan *Key/Value* dan menghasilkan keluaran berupa pasangan *Key/Value* juga.

map: (key 1, value 1) → [(key 2, value 2)]

Fungsi *Reduce* akan membaca hasil keluaran dari fungsi *Map* dan menggabungkan atau mengelompokkan berdasar *Key* tersebut. Fungsi *Reduce* juga menghasilkan keluaran berupa pasangan *Key/Value*.

reduce: (key 2, [value 2]) → [(key 3, value 3)]



Gambar 1. Skema Proses Map Reduce [3]

2.1.2 *Hadoop Map Reduce*

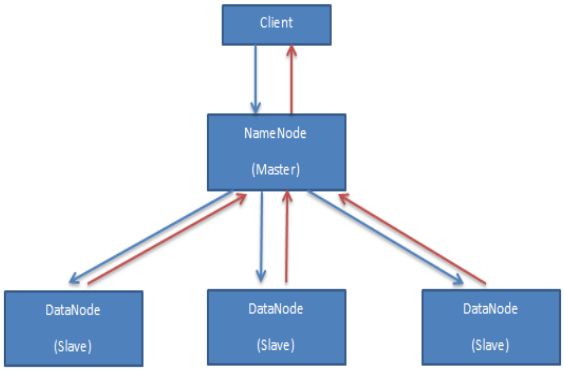
Hadoop MapReduce adalah modul yang menangani fungsi *Map Reduce*. Hadoop MapReduce dibangun di atas bahasa Java, sehingga hanya aplikasi berbasis Java saja yang bisa berjalan pada Hadoop *System*. Namun begitu, Hadoop menyediakan fungsi *Hadoop Streaming*, yaitu fungsi yang memungkinkan aplikasi lain yang dibangun dengan bahasa selain Java untuk tetap bisa berjalan di atas *Hadoop System*.

Hadoop MapReduce menggunakan teknologi HDFS (*Hadoop Distributed File System*) dalam melakukan pemrosesan data. HDFS berfungsi sebagai media penyimpanan dari file-file yang beroperasi di dalam *framework* Hadoop. File-file ini dapat

ISSN : 1411-1624
e-ISSN: 2621-8089

diakses/didistribusikan oleh komputer/operasi lain di dalam satu *cluster*. Posisi media penyimpanan ini bisa terdapat pada media/komputer yang sama (*single node*) ataupun pada lokasi yang tersebar (*multi node*).

Cara kerja HDFS adalah memecah data menjadi potongan-potongan data dengan ukuran tertentu. Potongan yang disebut *block* ini kemudian disebar ke komputer-komputer yang berada di dalam *cluster*. HDFS memiliki dua komponen utama, yaitu *namenode* dan *datanode*. *Namenode* adalah komponen yang bertindak sebagai *master*, sedangkan *datanode* adalah komponen yang bertindak sebagai *slave*. *Namenode* dan *datanode* biasanya berupa komputer terpisah. Namenode bertugas menyimpan dan mengelola metadata informasi dan merespon semua operasi file [11]. Pengelolaan juga meliputi penempatan blok data, mengorganisir, dan mengontrol blok-blok data yang disebar di dalam *cluster* tersebut. Sedangkan *datanode* bertugas menyimpan data-data yang dialamatkan kepadanya dan melaporkan kondisinya kepada *namenode* secara berkala. Ketika ada permintaan dari *client*, maka *client* akan berkomunikasi dengan *namenode*. Selanjutnya *namenode* (*master*) akan memecah proses menjadi blok-blok data. Blok data kemudian dikirimkan ke *datanode* (*slave*) untuk diproses. Setelah proses selesai dilakukan, *namenode* akan melaporkan hasilnya kepada *namenode*. *Namenode* akan melakukan proses *reduce* terakhir, dan kemudian menyampaikan hasilnya kepada *client*.



Gambar 2. Skema Proses *HDFS* [9]

2.1.3 *Apache Spark*

Laman resmi Apache (2016) menjelaskan bahwa Apache Spark pertama kali diperkenalkan oleh *Apache Software Foundation* untuk meningkatkan kecepatan proses komputasi dari Hadoop MapReduce [6]. Apache Spark mengusung konsep yang sama dengan Hadoop MapReduce yaitu model pemrosesan data *MapReduce* dan dilakukan secara paralel. Namun demikian, Spark bukanlah hasil modifikasi dari *Hadoop MapReduce* [12]. Hal ini dikarenakan Spark mengusung teknologi yang berbeda dengan yang digunakan pada Hadoop MapReduce. Apache Spark menggunakan teknologi yang disebut *Resilient Distributed Datasets (*RDDs*)* yang membuat Apache Spark dapat memproses data dengan lebih cepat. Laman resmi *Apache Spark* bahkan mengklaim bahwa Apache Spark 10-100 kali lebih cepat dibandingkan dengan Hadoop MapReduce [6]. Aspek lain yang membuat Spark benar-benar bukan bagian dari Hadoop adalah Apache Spark memiliki *cluster* tersendiri untuk mengolah data, sehingga tanpa adanya lingkungan Hadoop-pun Apache Spark tetap bisa berjalan.

ISSN : 1411-1624
e-ISSN: 2621-8089

RDDs adalah kumpulan *record* yang bersifat dinamis dan terdistribusi yang memungkinkan pengguna untuk mendapatkan hasil yang cepat. Konsep dari RDDs sebenarnya mirip dengan HDFS dimana terdapat *master node* dan *slave node* untuk mengerjakan pekerjaan secara paralel. Yang membedakan adalah, jika HDFS menggunakan media *disk* untuk *sharing* data, maka RDDs memanfaatkan *in-memory*. Konsep ini dibuat dikarenakan, proses dalam HDFS terlalu lama. Bahkan menurut [11] dalam mengolah data, Hadoop menghabiskan 90% waktunya untuk melakukan operasi *read-write* pada *disk*, selain juga proses *read-write* pada disk jauh lebih lama dibandingkan dengan proses *read-write* pada *in-memory*.

2.2. Konfigurasi Node

Dalam *parallel processing* dibutuhkan lebih dari satu komputer untuk memproses data. Komputer yang dimaksud bisa merupakan *dedicated computer* ataupun *virtual computer*. Pada penelitian ini digunakan empat buah *virtual computer* dimana satu komputer digunakan sebagai *master node* dan tiga komputer sebagai *slave node*. Masing-masing komputer memiliki *memory* sebesar 1GB, *Network Interface Card* (NIC), *hardisk* berkapasitas 8GB, dan menggunakan sistem operasi Ubuntu Server 12.04.3.

Adapun *software* yang dibutuhkan adalah *Java Development Kit*, Hadoop-1.0.3, Python-2.7.7, *Hadoop MapReduce* 2.7.2, Apache Spark 1.6.1, and mrjob (*library* python untuk model pemrograman *MapReduce).*

2.2.1 Konfigurasi *Computer Node*

Sebelum melangkah ke konfigurasi Hadoop MapReduce *framework* dan Apache Spark *framework*, ada beberapa hal yang perlu disiapkan di masing-masing *node* komputer. Di antaranya adalah instalasi Java Runtime Environment (JRE), konfigurasi Secure Shell (SSH), dan Konfigurasi *Internet Protocol* (IP) *Address*. Konfigurasi *IP Address* bisa dilihat pada tabel 1 berikut.

Tabel 1. Setting IP Address Pada *Computer Node*

Nama Komp.	***IP Address***	***IP Gateway***	***Keterangan***
master	192.168.0.1	192.168.0.10	master node
slave1	192.168.0.2	192.168.0.10	slave node
slave2	192.168.0.3	192.168.0.10	slave node
slave3	192.168.0.4	192.168.0.10	slave node

2.2.2 Konfigurasi Hadoop MapReduce dan Apache Spark

Instalasi *Hadoop MapReduce* relatif mudah, yaitu hanya dengan meng-*copy*-kan file hasil unduhan dari laman resmi Hadoop. Sedangkan untuk konfigurasi pada Hadoop MapReduce dilakukan dengan mengubah isi beberapa file antara lain : *hadoop-env.sh*, *core-site.xml*, *mapred-site.xml*, *hdfs-site.xml*, *yarn-site.xml*, dan *slave*. Instalasi dan konfigurasi dilakukan di semua komputer baik *master node* maupun *slave node.*

Proses instalasi dan konfigurasi pada *Apache Spark* sama dengan *Hadoop MapReduce*. Hanya saja file konfigurasi yang perlu lebih sedikit, *yaitu spark-env.sh, spark-defaults.sh,* dan *slaves*.

ISSN : 1411-1624
e-ISSN: 2621-8089

2.3 Persiapan Data Uji

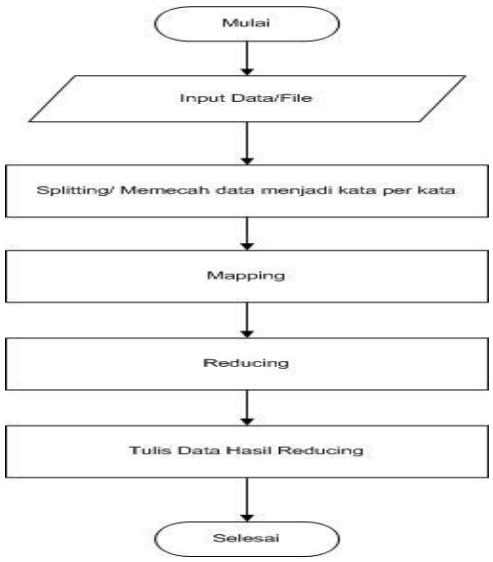
Data yang akan digunakan adalah data *dummy*. Data akan dikategorikan ke dalam tiga kelompok berdasarkan tipe datanya. Masing-masing tipe data akan memiliki ukuran yang berbeda-beda yaitu 4MB, 8MB, 16MB, 32MB, dan 64MB. Tabel 2 menjelaskan karakteristik dari masing-masing data.

2.4 Pembuatan Program Uji

Dalam eksperimen ini ada tiga jenis program yang akan diujikan, yaitu program *wordcount*, *date grouping*, dan *sum of the number*. Program dibuat dengan Bahasa Python. Masing-masing program dibuat dengan tujuan untuk mengolah data dengan tipe data yang berbeda, sehingga hasil eksperimen diharapkan dapat mewakili berbagai jenis tipe data. Program *word count* digunakan untuk mengolah data bertipe karakter (string), *date grouping* untuk data bertipe tanggal (*date*), dan *sum of the number data* bertipe bilangan (*numeric*). *Flowchart* untuk salah satu program bisa dilihat pada gambar 3.

Tabel 2. Klasifikasi Data Uji

Nama	*Deskripsi*	*Ukuran Data*
Data 1	- Data bertipe teks dengan ekstensi .txt - Data berupa artikel/kumpulan kata-kata. - Digunakan untuk menguji program *Wordcount*, yaitu program untuk menghitung jumlah kata yang muncul dalam sebuah file/data.	4, 8, 16, 32, dan 64 (MB).
Data 2	- Data bertipe teks dengan ekstensi .txt - Data berupa kumpulan tanggal. - Masing-masing tanggal dipisahkan dengan tanda titik koma (;) - Digunakan untuk menguji program *Date Grouping*, yaitu program untuk mengelompokkan data berdasar tanggal, bulan, ata tahun.	4, 8, 16, 32, dan 64 (MB).
Data 3	- Data bertipe teks dengan ekstensi .txt - Data berupa kumpulan angka. - Masing-masing akan dipisahkan dengan tanda titik koma (;). - Digunakan untuk menguji program *Sum of The Number*, yaitu program untuk menjumlahkan deret angka dalam suatu file/data.	4, 8, 16, 32, dan 64 (MB).



Gambar 3. *Flowchart* Program *Word Count*

3. HASIL DAN PEMBAHASAN

Total pengujian dalam eksperimen ini adalah 90 kali percobaan yang terdiri dari 45

ISSN : 1411-1624
e-ISSN: 2621-8089

percobaan menggunakan Hadoop MapReduce dan 45 percobaan menggunakan Apache Spark. Dari 45 percobaan tersebut terbagi menjadi tiga kelompok yaitu kelompok pengujian program wordcount, date grouping, dan sum of the number sehingga terdapat 15 kali percobaan di masing-masing kelompok. Dari 15 percobaan tersebut diuji menggunakan 1, 2, dan 3 slave node dengan varian data dari 4MB, 8 MB, 16 MB, 32 MB, dan 64 MB. Hasil pengujian program bisa dilihat pada Tabel 3. Waktu pemrosesan data dicatat dalam satuan detik. Waktu pemrosesan bertanda F berarti proses gagal karena beberapa alasan seperti *out of memory, time out*, dan lain-lain.

Data hasil eksperimen kemudian bisa dianalisa untuk membandingkan kecepatan antara *Hadoop MapReduce* dan *Apache Spark.* Dari 45 percobaan perbandingan yang dilakukan (45 percobaan dengan *Hadoop MapReduce* dan 45 percobaan dengan *Apache Spark*), didapatkan hasil 32 percobaan perbandingan berhasil dan 13 percobaan perbandingan gagal karena data yang tidak valid. Sementara dari 32 percobaan yang berhasil kesemuanya menunjukkan bahwa *Apache Spark* lebih cepat dibandingkan dengan *Hadoop MapReduce*.

Selanjutnya akan dicari nilai berapa kali lebih cepat *Apache Spark* dibandingkan *Hadoop MapReduce*. Nilai perbandingan bisa dilihat pada Tabel 4. Nilai yang didapatkan menunjukkan bahwa pada program Word count Apache Spark lebih cepat 5,83 kali dibanding *Hadoop MapReduc*e, pada program Date *Grouping Apache Sp*ark lebih cepat 3,07 kali, dan pada program *Sum of The Number Apache Spark* lebih cepat 6,05 kali. Sehingga rata-rata Apache Spark 4,99 kali lebih cepat dibandingkan dengan *Hadoop MapReduce*. Gambar 4 berikut menggambarkan perbandingan kecepatan *framework Hadoop MapReduce* dan *Apache Spark* dalam bentuk grafik, dimana terlihat bahwa *Apache Spark* lebih cepat dibanding *Hadoop MapReduce.*

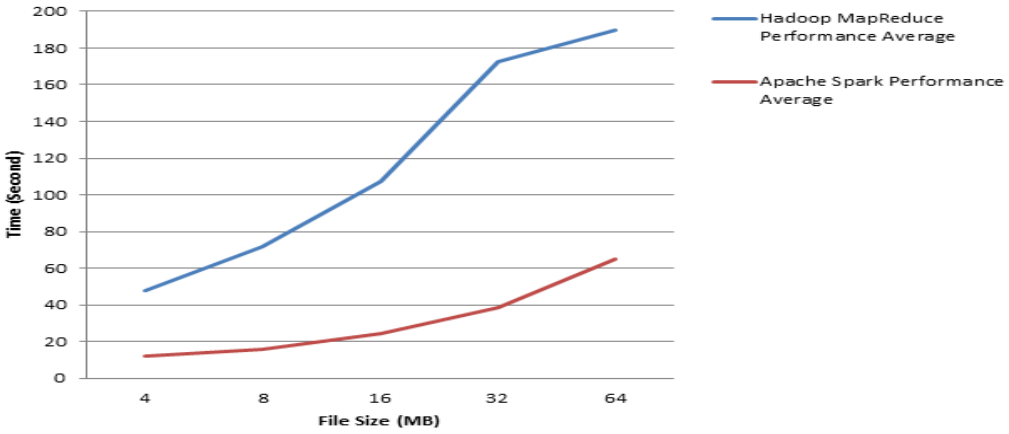
Tabel 3. Pencatatan Waktu Eksekusi Program

Nama Program	***Framework***	***Node***	***Ukuran Data (dalam MB)***				
			4	***8***	***16***	***32***	***64***
Wordcount	MapReduce	1	42	51	68	108	F
		2	42	50	70	108	188
		3	44	52	72	112	192
	Spark	1	7	12	14	16	23
		2	8	12	14	16	22
		3	10	12	14	16	20
Date Grouping	MapReduce	1	46	70	114	F	F
		2	42	61	105	F	F
		3	42	54	96	167	F
	Spark	1	16	20	31	61	114
		2	16	20	31	58	102
		3	16	20	32	56	98
Sum of The Number	MapReduce	1	60	104	F	F	F
		2	56	104	191	F	F
		3	54	102	144	368	F
	Spark	1	14	17	32	52	80
		2	12	16	26	38	64
		3	12	16	24	36	60

ISSN : 1411-1624
e-ISSN: 2621-8089

Tabel 4. . Kecepatan Apache Spark Dibandingkan Hadoop MapReduce

Nama Program	*Node*	*Ukuran Data (dalam MB)*					*Rata-rata*		
		4	*8*	*16*	*32*	*64*			
Word Count	1	6,0	4,2	4,8	6,7	-	5,4	**5,83**	**4,99**
	2	5,2	4,1	5,0	6,7	8,5	5,9		
	3	4,4	4,3	5,1	7,0	9,6	6,1		
Date Grouping	1	2,8	3,5	3,6	-		3,3	**3,07**	
	2	2,6	3,0	3,3	-	-	3,0		
	3	2,6	2,7	3,0	2,9	-	2,8		
Sum of The Number	1	4,3	6,1	-	-	-	5,2	**6,05**	
	2	4,7	6,5	7,3	-	-	6,1		
	3	4,5	6,3	6,0	10,2	-	6,7		

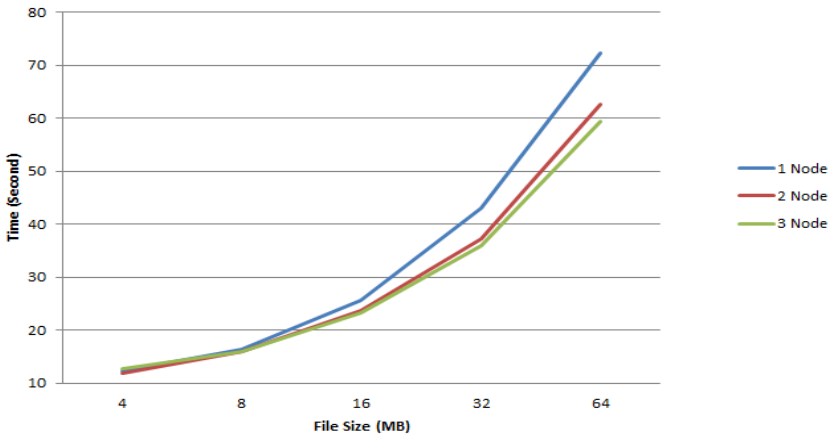


Gambar 5. Pengaruh Jumlah Slave Node Terhadap Kecepatan Pada Hadoop MapReduce

Tabel 5. Pengaruh Jumlah Slave Node Terhadap Kecepatan Pada Hadoop MapReduce

Nama Program	***Node***	**Ukuran Data (dalam MB)**					**Rata-rata**
		4	**8**	**16**	**32**	**64**	
WordCount	2:1	1,00	1,02	0,97	1,00	-	1,00
	3:1	0,95	0,98	0,94	0,96	-	0,96
	3:2	0,95	0,96	0,97	0,96	-	0,96
Date Grouping	2:1	1,10	1,15	1,09	-	-	1,11
	3:1	1,10	1,30	1,19	-	-	1,20
	3:2	1,00	1,13	1,09	-	-	1,07
Sum of The Number	2:1	1,07	1,00	-	-	-	1,04
	3:1	1,11	1,02	-	-	-	1,07
	3:2	1,04	1,02	1,33	-	-	1,13
	Rata-rata perbandingan kecepatan penggunaan 2:1 *slave node*						**1,05**
	Rata-rata perbandingan kecepatan penggunaan 3:1 *slave node*						**1,08**
	Rata-rata perbandingan kecepatan penggunaan 3:2 *slave node*						**1,05**

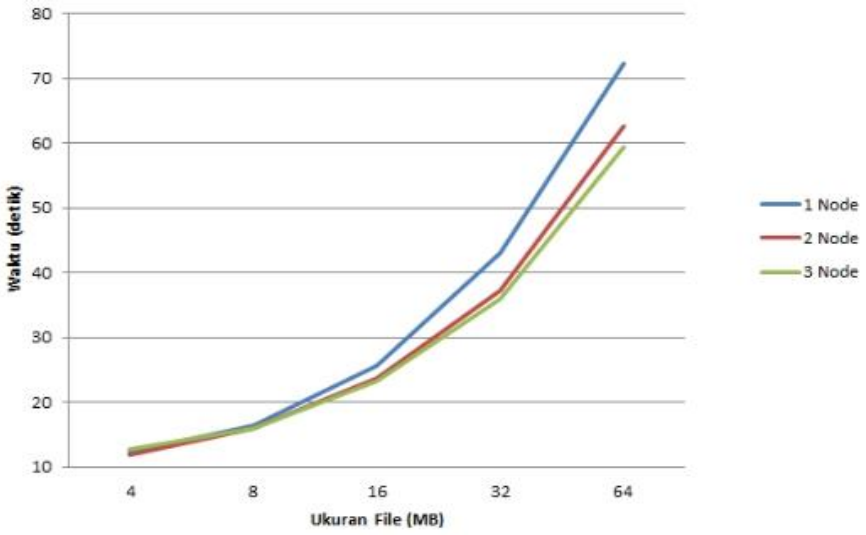
ISSN : 1411-1624
e-ISSN: 2621-8089



Gambar 5. Pengaruh Jumlah Slave Node Terhadap Kecepatan Pada Hadoop MapReduce

Tabel 6. Pencatatan Waktu Eksekusi Program

Nama Program	***Slave Node***	***Ukuran Data***					***Rata-rata***
		4	***8***	***16***	***32***	***64***	
WordCount	2:1	0,88	1,00	1,00	1,00	1,05	0,98
	3:1	0,70	1,00	1,00	1,00	1,15	0,97
	3:2	0,80	1,00	1,00	1,00	1,10	0,98
Date Grouping	2:1	1,00	1,00	1,00	1,05	1,12	1,03
	3:1	1,00	1,00	0,97	1,09	1,16	1,04
	3:2	1,00	1,00	0,97	1,04	1,04	1,01
Sum of The Number	2:1	1,17	1,06	1,23	1,37	1,25	1,22
	3:1	1,17	1,06	1,33	1,44	1,33	1,27
	3:2	1,00	1,00	1,08	1,06	1,07	1,04
	Rata-rata perbandingan kecepatan penggunaan 2:1 *slave node*						**1,07**
	Rata-rata perbandingan kecepatan penggunaan 3:1 *slave node*						**1,09**
	Rata-rata perbandingan kecepatan penggunaan 3:2 *slave node*						**1,04**



Gambar 6. Pengaruh Jumlah Slave Node Terhadap Kecepatan Pada Apache Spark

ISSN : 1411-1624
e-ISSN: 2621-8089

Selain membandingkan performa kedua framework, dari data yang diperoleh juga bisa dianalisa mengenai pengaruh jumlah slave node terhadap performa masing-masing framework. Data pada Tabel 5 menunjukkan bahwa pada *Hadoop MapReduce* yang dijalankan menggunakan 2 slave node sedikit lebih cepat dibandingkan hanya menggunakan 1 slave node, yaitu sebesar 1,05 kali lebih cepat. Sedangkan penggunaan 3 slave node sedikit lebih cepat 1,08 kali dibandingkan dengan menggunakan 1 slave node, dan penggunaan 3 slave node 1,05 lebih cepat dibandingkan 2 slave node. Gambar 10 menunjukkan perbedaan tersebut dalam bentuk grafik.

Pada *Apache Spark*, jumlah *slave node* juga berpengaruh terhadap kecepatan program. Data pada Tabel 6 menunjukkan bahwa Apache Spark yang dijalankan menggunakan 2 slave node sedikit lebih cepat jika dibandingkan hanya menggunakan 1 *slave node* yaitu sebesar 1,07 kali lebih cepat. Sedangkan penggunaan 3 slave node lebih cepat 1,09 kali dibandingkan dengan menggunakan 1 slave node, serta penggunaan 3 slave node 1,04 lebih cepat dibandingkan penggunaan 2 slave node. Gambar 6 berikut menunjukkan perbedaan tersebut dalam bentuk grafik.

4. KESIMPULAN

Penelitian ini membandingkan kinerja dua *framework Map Reduce* yaitu *Hadoop MapReduce* dan *Apache Spark* berdasarkan lama waktu yang digunakan untuk melakukan pemrosesan data. Hasil penelitian menunjukkan bahwa *Apache Sp*ark lebih cepat dibandingkan *Hadoop MapReduce* dalam memproses data bertipe karakter (*string*), bilangan (*numeri*c), dan tanggal (*date*) dengan rata-rata kinerja 4,99 kali lebih cepat.

Di sisi lain penggunaan jumlah slave node juga berpengaruh terhadap kinerja program baik pada *framewok Hadoop MapReduce* maupun *Apache Spark*. Pada *Hadoop MapReduce* program yang dijalankan menggunakan dua *slave node* memiliki kinerja 1,05 kali lebih cepat dibandingkan dengan menggunakan satu *slave node*, penggunaan tiga *slave node* lebih cepat 1,08 kali dibandingkan satu *slave node*, dan penggunaan tiga *slave node* lebih cepat 1,05 kali dibandingkan dua *slave node*. Sementara pada *Apache Spark*, program yang dijalankan menggunakan dua *slave node* lebih cepat 1,07 kali dibandingkan satu *slave node*, penggunaan tiga *slave node* lebih cepat 1,09 kali dibandingkan satu *slave node*, dan penggunaan tiga slave node lebih cepat 1,04 kali dibandingkan dengan dua *slave node*. Dari data tersebut dapat disimpulkan bahwa jumlah *slave node* berpengaruh pada kinerja kedua *framework*. Semakin banyak *slave node* yang digunakan, semakin cepat pula kinerja *framework*.

ISSN : 1411-1624
e-ISSN: 2621-8089

DAFTAR PUSTAKA

[1] R. D. Gantz, R. John, *"The Digitalization of the World From Edge to* Core". Framingham : IDC, 2018.
[2] E.G. Ularu, et al, "Perspectives on Big Data and Big Data Analytics", Database Systems *Journal, vol III, no 4, 2012*.
[3] J. Dean dan S. Ghemawat, "MapReduce : Simplified Data Processing on Large Cluster", San Fransisco : *Sixth Symposium on Operating System Design and Implementation*, 2014.
[4] The Apache Software Foundation, "*What Is Apache Hadoop?*" [Online], Available: http://hadoop.apache.org/#What+Is+Apache +Hadoop%3F. [Accessed : 15-Jun-2016].
[5] R. Xue, "*SQL Engines for Big Data Analytics : SQL on Hadoop*", Aalto University, Espoo, 2015.
[6] The Apache Software Foundation, "*Home - Speed",* [Online], Available : http://hadoop.apache.org/#What+Is+Apache +Hadoop%3F, [Accessed : 15-Jun-2016].
[7] T. White, "*Hadoop : The Definitive Guide, Second Edition*". Sebastopol : O'Reilly Media, Inc, 2011.
[8] P. C. Zikopolous, et al, "*Understanding Big Data : Analytics for Enterprise Class Hadoop and Streaming Data*". New York : The McGraw-Hill Companies, 2012.
[9] A. Momtaz, *"Detecting Document Similarity in Large Document Collection using MapReduce and the Hadoop Framework*". Dhaka : Brac University, 2012.
[10] C. Wibawa, et al, "Document Similarity Measurement Using Ferret Algorithm and Map Reduce Programming Model". *International Journal of Computer Trends and Technology, vol 19, no 2*, 2015.
[11] V. Pellakuri dan R. Rao, "Hadoop MapReduce Framework in Big Data Analytics". *International Journal of Computer Trends and Technology, vol 8, no 3*, 2014.
[12] V. S. Jonnalagadda, et al, "A Review Study of Apache Spark in Big Data Processing". *International Journal of Computer Trends and Technology, vol 4, Issue 3*, 2016.