Perbandingan Metode *Mapreduce Berbasis Single Node Hadoop* Pada Aplikasi *Word Count*

Elvira Oktaviani[1], Mastura Diana Marieska*[2,] Alvi Syahrini Utami[3],
[1,2,3]Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Sriwijaya
e-mail: [1]elviraviany@gmail.com, ***[2]mastura.diana@ilkom.unsri.ac.id**,
[3]alvisyahrini@ilkom.unsri.ac.id,

Abstrak

Dalam konteks pemrosesan Big Data, Hadoop MapReduce menjadi framework yang digunakan untuk mengembangkan perangkat lunak dan melakukan pemrosesan data yang besar secara paralel. Word Count merupakan jenis pekerjaan yang digunakan untuk menghitung kemunculan kata-kata unik dalam sebuah file teks. Kecepatan dalam pemrosesan data merupakan faktor penting yang harus dipertimbangkan dalam pemenuhan standar pemrosesan Big Data. Penelitian yang dilakukan melibatkan pemrosesan file teks menggunakan metode MapReduce pada Hadoop Distributed File System (HDFS) menggunakan single node dengan membandingkan hasil pemrosesan word count dengan dan tanpa menggunakan metode MapReduce. Penelitian ini memberikan hasil bahwa implementasi Word Count tanpa menggunakan metode MapReduce menawarkan kecepatan yang lebih baik dalam pemrosesan data teks bahasa Indonesia dibandingkan dengan menggunakan Hadoop single node. Selain itu, perbandingan waktu pemrosesan antara program Word Count dengan metode MapReduce berbasis Hadoop dan Word Count tanpa MapReduce menunjukkan bahwa program Word Count tanpa MapReduce memiliki waktu pemrosesan yang lebih cepat. Reduksi waktu yang tinggi, hingga 95% pada file berukuran 5 MB, dapat dicapai dengan penggunaan metode Word Count tanpa MapReduce, meskipun tingkat reduksi tersebut menurun seiring dengan peningkatan ukuran file.

Kata kunci*— Big Data, Word Count, MapReduce, HDFS, Hadoop Single Node*

Abstract

In the context of Big Data processing, Hadoop MapReduce serves as a framework used to develop software and process large-scale data in parallel. Word Count is a type of task used to count the occurrences of unique words in a text file. Considering processing time is crusial in adhering to standards of Big Data Processing. The conducted research involved the processing of text files using the MapReduce method on the Hadoop Distributed File System (HDFS) using a single node, comparing the results of Word Count processing with and without the use of MapReduce. The research findings indicate that the implementation of Word Count without using MapReduce offers better speed in processing Indonesian language text data on a Hadoop single node. Additionally, the comparison of processing time between the Word Count program with Hadoop MapReduce and the Word Count program without MapReduce shows that the latter has faster processing time. A significant reduction in processing time, up to 95% for a 5 MB file size, can be achieved by using the Word Count method without MapReduce. However, the level of reduction decreases with increasing file size.

Keywords*—Big Data, Word Count, MapReduce, HDFS, Hadoop Single Node*

1. PENDAHULUAN

Word Count [1] merupakan jenis *job* yang digunakan untuk menghitung kemunculan sebuah kata unik yang terdapat pada tipe data file teks dengan keluaran berupa jumlah berapa kata yang muncul pada pengujian tersebut. *Word Count* sendiri dapat mempengaruhi akurasi dalam analisis sentimen terkait pengelompokan kata positif, negatif dan netral pada *text mining* yang merupakan teknik mengekstraksi informasi dari data teks. Data dalam jumlah yang sangat besar, variatif dan sangat cepat pertumbuhannya disebut dengan *Big Data*. Faktor kecepatan pada komputasi tersebut berpengaruh pada komponen yang memenuhi standar pemrosesan terkait. Maka dari itu, untuk menunjang proses komputasi tersebut sangat penting untuk mengaplikasikan suatu algoritma tertentu yang membuat pemrosesan data tersebut menjadi efektif dan efisien [2].

Adapun definisi *Big Data* yakni perpaduan sistem teknologi yang memungkinkan kita untuk mengelola sejumlah besar data berbeda dengan kecepatan dan akurasi yang wajar dalam menganalisis. Terdapat 3 atribut pada *Big Data* yakni *variety*, *volume*, serta *velocity* [3]. Utilitas teknologi yang dimiliki dari proses *big data* meliputi melakukan koneksi data penyimpanan dengan memanfaatkan *key-value store (KVS)* juga melakukan komputasi secara paralel dengan mengaplikasikan *MapReduce* [4].

Volume data yang akan diproses oleh aplikasi *cloud* tumbuh jauh lebih cepat daripada daya komputasi. Pertumbuhan ini menuntut strategi baru untuk memproses dan menganalisis informasi. *Hadoop MapReduce* telah menjadi model komputasi yang kuat untuk mengatasi masalah ini. *Hadoop HDFS* menjadi lebih populer di antara semua alat *Big Data* karena *open source* dengan skalabilitas yang fleksibel, lebih sedikit mengeluarkan biaya & memungkinkan penyimpanan data dalam bentuk apa pun tanpa perlu memiliki tipe data atau skema yang ditentukan. [5]

Berdasarkan uraian tersebut, *Hadoop MapReduce* adalah suatu *framework* untuk mengembangkan suatu *software*, serta mampu melakukan pengolahan kumpulan data yang berukuran besar dengan pemrosesan paralel pada *node* komputasi [5]. *MapReduce* dan *HDFS* merupakan bagian inti dari teknologi *Hadoop* di mana *HDFS* menyediakan penyimpanan data yang banyak dan beragam, dan *MapReduce* menyediakan pemrosesan data yang cepat. *MapReduce* biasanya mendukung *batch-based* paralelisasi skala tinggi dan pemrosesan yang andal. Kerangka kerja *Hadoop MapReduce* menggunakan sistem file terdistribusi untuk membaca dan menulis datanya. Oleh karena itu, kinerja *I/O* dari *Hadoop MapReduce* sangat bergantung pada *HDFS* [6]. Kajian yang akan diteliti melibatkan pemrosesan *big data* menggunakan metode *MapReduce* pada *Hadoop Distributed File System (HDFS)*. Hasil pemrosesan akan dibandingkan dengan pemrosesan *word count* tanpa *map reduce*.

Berbagai studi telah dilakukan terkait metode MapReduce pada Hadoop Single Node. Pada penelitian [7], dilakukan pengujuan terhadapat pemrosesan *polling* dengan menggunakan metode *MapReduce* dan tanpa *MapReduce* serta membandingkan hasil pengujian terhadap *Hadoop* dengan ukuran *block default* dan *block* yang diperbesar. Dari penelitian tersebut, disimpulkan bahwa metode *MapReduce* akan bekerja secara efektif di data ukuran besar tetapi tidak di data yang berukuran kecil. Sedangkan pengujian terhadap *block* pada *Hadoop* didapatkan hasil bahwa *block* yang diperbesar tidak lebih cepat dibandingkan dengan *block* berukuran *default*.

Penelitian lain [2] melakukan implementasi metode *MapReduce* berbasis *HDFS*, yang mengungkapkan bahwa *MapReduce* mampu diterapkan dalam beragam cara. *Apache Flink* menyediakan fungsi transformasi untuk membuat metode *MapReduce* seperti *Hadoop* dengan cara yang sama. Arsitektur *Apache Flink* memungkinkan *MapReduce* mengirimkan data *mikro-batch* ke memori hampir secara *real-time*, sehingga proses komputasi menjadi lebih cepat. Kecepatan yang dihasilkan oleh metode *MapReduce* pada *Apache Flink* sebesar dua kali daripada kecepatan penggunaan *Hadoop MapReduce*, namun maksimalnya keluaran yang diharapkan dipengaruhi oleh spesifikasi komputer yang memumpuni.

Kemajuan teknologi *cloud* & peningkatan penggunaan *Internet* menciptakan kumpulan data baru yang sangat besar dengan peningkatan nilai bagi bisnis dan kekuatan pemrosesan untuk

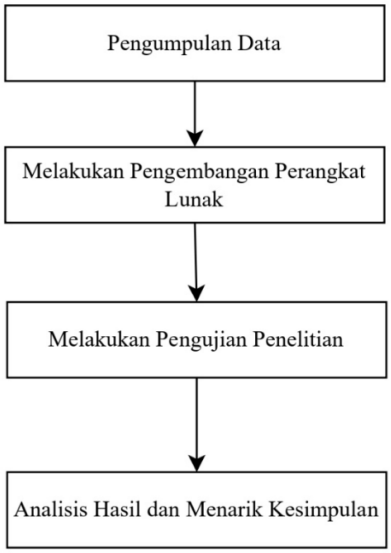
menganalisisnya dengan harga terjangkau. *Big Data* masih dalam masa pertumbuhan awal tetapi sudah memiliki efek mendalam pada Perusahaan teknologi dan cara kami melakukan bisnis. Ukuran kumpulan data ini menunjukkan bahwa eksploitasi mungkin memerlukan kategori baru penyimpanan data dan sistem analisis dengan arsitektur yang berbeda. Paradigma pemrograman *Hadoop-MapReduce* memiliki basis substansial dalam komunitas *Big Data* karena efektivitas biaya pada kluster *Linux*, dan di *cloud* melalui unggahan data ke *vendor cloud* yang telah mengimplementasikan *Hadoop/HBase*. Efektivitas dan kemudahan penggunaan metode *MapReduce* dalam paralelisasi melibatkan banyak algoritma analisis data. Adapun *HDFS* merupakan sistem file terdistibusi yang menjadi suatu wadah penampungan data yang terbilang besar [5].

Pada penelitian ini diimplementasikan aplikasi *Word Count* yang dapat menghitung jumlah kemunculan kata unik pada dokumen teks. Implementasi *Word Count* akan dilakukan dengan 2 cara, yaitu secara sekuensial tanpa *MapReduce* dan dengan *MapReduce* memakai Hadoop. Konfigurasi Hadoop yang digunakan adalah Hadoop Single Node, dimana HDFS dan *MapReduce* dijalankan pada satu komputer. Selanjutnya dilakukan pengukuran lamanya *processing time* untuk variasi ukuran file teks. Nilai *processing time* pada implementasi *Word Count* tanpa *MapReduce* akan dibandingkan dengan menggunakan *MapReduce*, untuk kemudian dianalisis perbedaannya.

2. METODE PENELITIAN

2.1 Tahapan Penelitian

Tahapan penelitian ini terbagi atas 4 tahapan, yaitu pengumpulan data, pengembangan perangkat lunak, pengujian, dan analisis hasil. Gambar 1 alur tahapan penelitian.

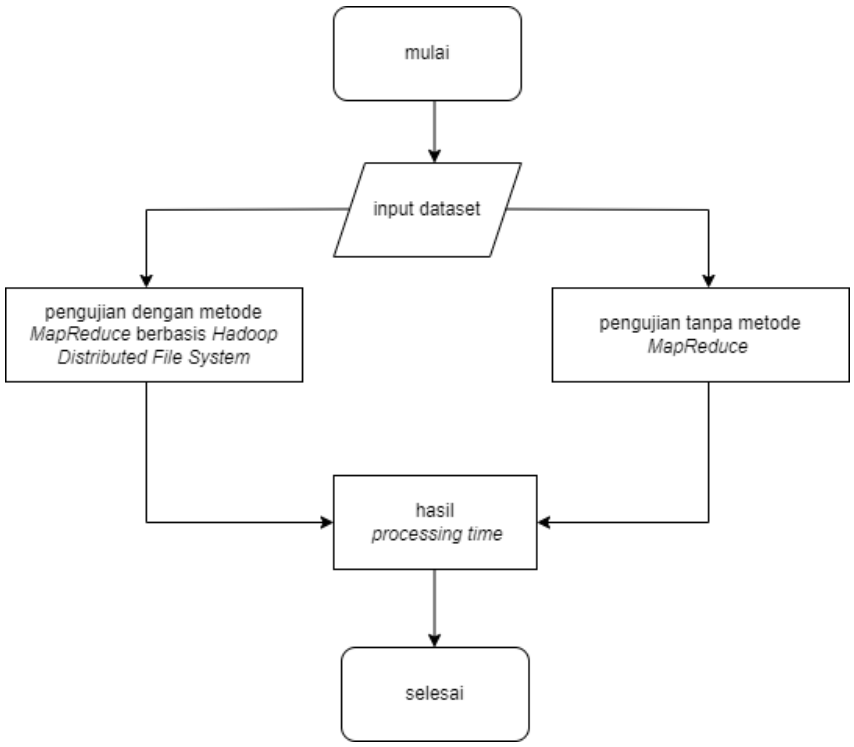


Gambar 1. Tahapan Penelitian

Pada tahapan pertama, dilakukan pengumpulan data sesuai dengan ukuran yang diinginkan. Tahap pengembangan perangkat lunak merupakan tahap dimana aplikasi *Word Count* dikembangkan dengan 2 cara, yaitu dengan Hadoop *MapReduce* yang dijalankan pada HDFS dan tanpa *MapReduce* berupa program bahasa Java yang dijalankan secara *sequential*. Tahap ketiga merupakan pengujian penelitian untuk mendapatkan nilai *processing time* pada tiap kasus uji. Tahap terakhir merupakan tahap membandingkan dan menarik kesimpulan. Pada tahap terakhir ini akan dihitung persentase reduksi nilai *processing time* pada implementasi dengan *MapReduce* dan tanpa *MapReduce.*

2.2 Pengujian Penelitian

Pengujian dilakukan berdasarkan kriteria yang telah ditentukan sebelumnya. Data yang diolah merupakan data *plain text* yang telah dimodifikasi agar menyesuaikan ukuran yang diinginkan. Pengujian dimulai dari menginputkan data dan sistem melakukan proses pengolahan data menggunakan program *MapReduce Word Count* dengan proses pengujian yang dijalankan pada *Hadoop*. Lalu pengujian dilanjutkan dengan mengolah data menggunakan program *Word Count* tanpa *MapReduce*. Pengujian dilakukan secara bertahap mulai dari ukuran data uji terkecil hingga data terbesar sesuai dengan distribusi ukuran data sebelumnya agar menghasilkan keluaran yang diharapkan. Hal ini diilustrasikan dalam Gambar 2 sebagai berikut:



Gambar 2. Kerangka Kerja Pengujian

Berikut penjelasan dari kerangka kerja pada Gambar 1 sebagai berikut:

a. Input Dataset
   Pada tahap ini dilakukan penginputan dataset berupa data *plain text* yang telah dimodifikasi sehingga menyesuaikan dengan ukuran yang akan digunakan.
b. Pengujian dengan metode *MapReduce* berbasis *Hadoop*
   Pada tahap ini akan dilakukan pengujian program *Word Count* dengan metode *MapReduce* berbasis *Hadoop Distributed File System* pada 1 komputer (*Single Node*)
c. Pengujian tanpa *Mapreduce*
   Pada tahap ini akan dilakukan pengujian program *Word Count* tanpa *Mapreduce* yang dijalankan secara *sequential*.
d. Hasil
   Hasil akhir pada proses pengujian ini berupa data teks dengan menampilkan hasil dari jumlah setiap kata yang muncul dan akumulasi waktu yang digunakan pada pemrosesan tersebut.

2.3 Pengumpulan Data

Data yang digunakan dalam penelitian ini berupa data sekunder berupa file teks dengan format *plain text* yang diperoleh dari sumber yang menyediakan dataset yang dapat diakses secara

publik pada *website Kaggle* dengan dibagi menjadi beberapa ukuran. Tabel 1 menampilkan pembagian data teks yang akan diproses. Dan ukuran data maksimal untuk pengujian adalah 200 MB. Pembatasan ukuran data uji dilakukan untuk menghindari terjadinya *lag*.

Tabel 1 Pembagian Ukuran Data Uji

Data ke-	Ukuran
1	5 MB
2	10 MB
3	20 MB
4	50 MB
5	100 MB
6	200 MB

2.4 Format Data Pengujian

Adapun format data pengujian dalam penelitian ini dapat dilihat pada Tabel 2 sebagai berikut:

Tabel 2. Format Data Pengujian *Word Count* dengan *MapReduce* dan tanpa *MapReduce*

Data ke-	Ukuran	Pengujian 1	Pengujian 2	Pengujian 3	Rata-rata
1	5 MB				
2	10 MB				
3	20 MB				
4	50 MB				
5	100 MB				
6	200 MB				

Untuk tiap data uji akan dihitung nilai *processing time* sebanyak 3 kali. Dari ketiga nilai *processing time* ini kemudian diambil nilai rata-ratanya. Hal ini dilakukan agar nilai *processing time* yang terukur dapat lebih valid. Setelah diambil nilai rata-rata untuk tiap data ujia pada program *Word Count* dengan dan tanpa *MapReduce*, selanjutnya dibandingkan dan diambil selisihnya seperti yang terlihat pada Tabel 3.

Tabel 3. Hasil Perbandingan Pengujian

Data ke-	Ukuran	Dengan *MapReduce*	Tanpa *MapReduce*	Selisih
1	5 MB			
2	10 MB			
3	20 MB			
4	50 MB			
5	100 MB			
6	200 MB			

2.5 Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak pada penelitian ini menerapkan model *Rational Unified Process (RUP),* yang terdiri dari empat fase pengembangan [8], yaitu:

1. Fase Insepsi

   Pada tahap awal ini dilakukan pengumpulan literatur yang digunakan sebagai

penyempurnaan sebuah rumusan masalah terhadap penelitian tentang implementasi *word count* dengan metode *MapReduce* berbasis *Hadoop*, mengumpulkan data, menentukan batasan masalah dan pada tahap ini juga telah melakukan perencanaan proyek pengembangan.

2. Fase Elaborasi
   Pada fase ini untuk menganalisa rumusan masalah dengan bertujuan menyempurnakan rumusan masalah penelitian dan batasan penelitian serta memperbaiki. Pada fase ini juga melakukan analisa terhadap fase sebelumnya yaitu insepsi dengan menggunakan teknik *Unified Modeling Language (UML)* dan juga fase ini melakukan pembuatan diagram aktivitas dari perangkat lunak yang akan dikembangkan sesuai kebutuhan fungsional dan non fungsional.
3. Fase Konstruksi
   Fase Konstruksi bertujuan untuk mengembangkan perangkat lunak berdasarkan diagram *UML*. Pada fase ini terdapat proses implementasi, desain dan pengujian. Dari proses tersebut memiliki hasil jumlah kata dari masukan yang telah diproses menggunakan *Hadoop MapReduce word count* yang akan diujikan pada penelitian ini.
4. Fase Transisi
   Pada Fase yang terakhir ini dilakukan pengujian dan analisa untuk penarikan kesimpulan terhadap implementasi *word count* dengan metode *MapReduce* berbasis *Hadoop*.

3. HASIL DAN PEMBAHASAN

3.1 Skenario Pengujian

Pengujian penelitian dilakukan dengan menggunakan perangkat lunak yang telah dikembangkan dan digunakan sebanyak 6 dokumen *plain text* berbahasa Indonesia dengan 6 dokumen tersebut digunakan pada pengujian *word count* dengan *MapReduce* dan *Word Count* tanpa *MapReduce*. Sesuai dengan kebutuhan penelitian yang telah di diuraikan sebelumnya, dalam pengujian penelitian, dokumen data uji akan diproses dan waktu pemrosesannya akan diakumulasi. Hasil akumulasi waktu ini akan dibandingkan dan dianalisis untuk mendapatkan pemahaman yang lebih mendalam.

Pengujian *Word Count* menggunakan metode *MapReduce* dan tanpa *MapReduce* ini bertujuan untuk membandingkan performa peningkatan *processing time* pada *Hadoop Distributed File System* dan pemrosesan serial. Dalam pengujian penelitian, setiap dokumen data uji diproses menggunakan perangkat lunak yang telah dikembangkan satu-persatu. Totalnya, dilakukan 18 kali pengujian.

Proses pengujian penelitian dilakukan dengan ketentuan 3 kali pengujian untuk setiap file data uji yang akan di ambil rata-rata *processing time* file tersebut dalam satuan waktu detik. Hasil pengujian kemudian dirangkum berupa reduksi waktu dari kedua metode tersebut untuk selanjutnya dianalisis dan dievaluasi.

3.2 Analisis Hasil

Data hasil pengujian ini adalah hasil dari pengujian *Word Count* dengan *MapReduce* dan tanpa *MapReduce*. Pada tabel 4 dan 5 dilakukan pengujian pada 6 file dan setiap file di jalankan sebanyak 3 kali. Hasil dari 3 kali pengujian tersebut akan di ambil rata-rata waktu pemrosesan nya dalam satuan waktu detik.

Tabel 4. Hasil Pengujian *Word Count* dengan *MapReduce*

Data ke-	Ukuran	Pengujian 1 (detik)	Pengujian 2 (detik)	Pengujian 3 (detik)	Rata-rata (detik)
1	5 MB	4.477	5.060	4.403	4.646

Data ke-	Ukuran	Pengujian 1 (detik)	Pengujian 2 (detik)	Pengujian 3 (detik)	Rata-rata (detik)
2	10 MB	6.238	5.936	6.650	6.274
3	20 MB	13.604	13.140	15.027	13.923
4	50 MB	17.239	13.934	15.155	15.442
5	100 MB	27.019	27.584	26.291	26.964
6	200 MB	118.451	117.258	121.250	118.986

Tabel 5. Hasil Pengujian *Word Count* tanpa *MapReduce*

Data ke-	Ukuran	Pengujian 1 (detik)	Pengujian 2 (detik)	Pengujian 3 (detik)	Rata-rata (detik)
1	5 MB	0.005	0.004	0.004	0.004
2	10 MB	1.640	1.293	1.350	1.407
3	20 MB	2.699	3.299	3.251	3.083
4	50 MB	6.157	6.350	6.131	6.212
5	100 MB	24.874	24.601	25.564	25.013
6	200 MB	57.458	55.938	54.820	56.072

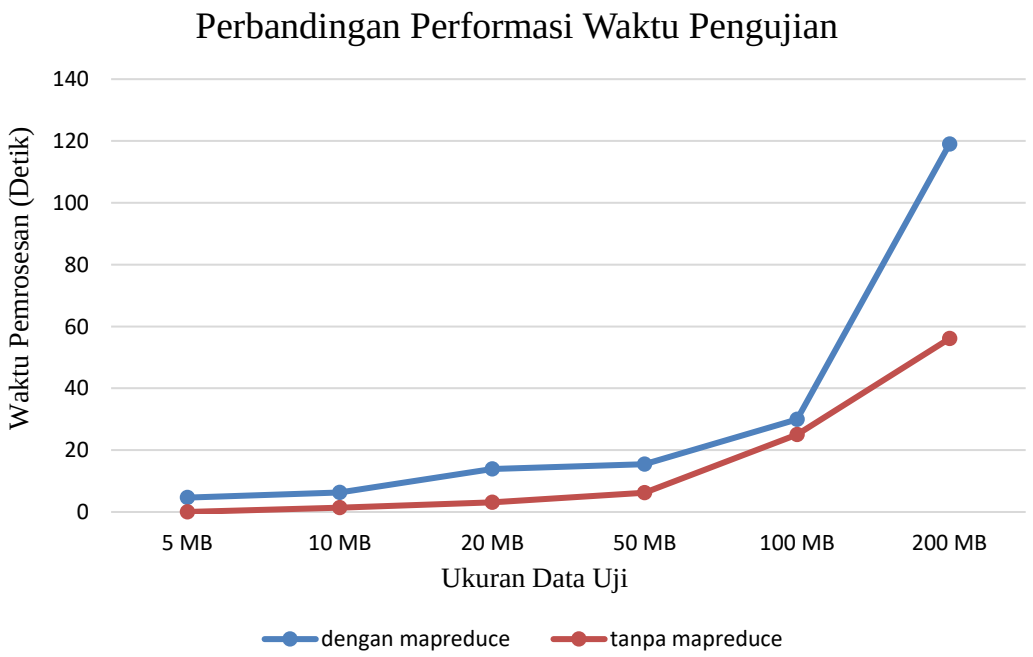
Data dari pengujian ini adalah hasil pengujian penghitungan kata dengan *MapReduce* dan tanpa *MapReduce*. Pada Tabel 4 dan 5, file diuji dan setiap file dijalankan 3 kali. Hasil dari 3 pengujian akan diambil rata-rata *processing time* dalam detik.

Tabel 6. Reduksi *Processing Time Word Count* dengan *MapReduce* dan tanpa *MapReduce*

Data ke-	Ukuran	Dengan *MapReduce* (detik)	Tanpa *MapReduce* (detik)	Reduksi Waktu (detik)	Reduksi Waktu (%)
1	5 MB	4.646	0.004	4.642	95%
2	10 MB	6.274	1.407	4.867	77.57%
3	20 MB	13.923	3.083	10.840	77.85%
4	50 MB	15.442	6.212	9.230	59.77%
5	100 MB	26.964	25.013	1.951	7.23%
6	200 MB	118.986	56.072	62.914	52.87%

Hasil akhir dari penelitian ini adalah perhitungan kinerja waktu dengan tujuan untuk mengetahui tingkat pengurangan yang dihasilkan dalam setiap uji coba. Tingkat pengurangan waktu proses diungkapkan dalam bentuk persentase. Semakin tinggi nilai persentasenya, semakin tinggi pula tingkat pengurangan waktu.

Berdasarkan Tabel 6, pengolahan dengan tingkat pengurangan yang menghasilkan ukuran berkas terkecil sebesar 5 MB memiliki pengurangan waktu hingga 95%. Secara bertahap, pengurangan waktu untuk ukuran berkas yang lebih besar menurun dan mencapai 7.23% pada berkas sebesar 100 MB, kemudian meningkat kembali menjadi 52.87% pada berkas sebesar 200 MB.



Gambar 3. Grafik Perbandingan Performasi Waktu Pengujian *Word Count* dengan *MapReduce* dan tanpa *MapReduce*

Berdasarkan grafik yang dihasilkan pada Gambar 3, terlihat bahwa waktu pemrosesan untuk penghitungan kata tanpa *MapReduce* lebih cepat dibandingkan dengan penghitungan kata menggunakan *MapReduce* berbasis *Hadoop*. Hal ini dapat terjadi karena pemrosesan pada sistem *Hadoop* dijalankan pada satu *node* sehingga fitur paralel pada *Hadoop* tidak berjalan secara optimal. *Hadoop* sendiri bekerja dengan fitur paralel yang membagi data yang akan diproses menjadi blok-blok dan kemudian didistribusikan ke beberapa *node* dalam sebuah *cluster*. Pada *Hadoop single node* ini, semua proses berjalan pada satu mesin sehingga sistem paralel tidak berfungsi secara optimal karena hanya menggunakan 1 *node*. Namun, *Hadoop single node* tentu masih dapat menjalankan sistem file terdistribusi pada *HDFS* dan *YARN* yang bertindak untuk mengalokasikan sumber daya pada sistem *Hadoop single node* tersebut.

Persentase reduksi *processing time* cenderung mengecil seiring dengan bertambahnya ukuran file. Persentase reduksi terbesar didapat pada ukuran file terkecil, yaitu 5 MB. Persentase reduksi terbesar didapat pada ukuran file 100MB. Hal ini dapat terjadi akibat implementasi sistem blok pada HDFS. Pada HDFS data disimpan, dibuat replikanya, dan diproses untuk tiap satu ukuran blok. Pada ukuran file 100MB didapat nilai *processing time* yang mendekati nilai *processing time* pada pemrosesan serial, menunjukkan bahwa pemrosesan blok pada HDFS paling optimal di ukuran file 100MB.

4. KESIMPULAN

Berdasarkan proses penelitian yang telah dilakukan dan analisis hasil pengujian pemrosesan *Word Count* secara serial atau tanpa menggunakan metode *MapReduce* berbasis *Hadoop Distributed File System* pada file teks berbahasa Indonesia berhasil dilakukan dengan mengukur tingkat reduksi *processing time*. Adapun kesimpulan yang diperoleh pada penelitian implementasi *Word Count* pada teks berbahasa Indonesia ini adalah sebagai berikut:

1. Didapatkan hasil bahwa implementasi *Word Count* tanpa *MapReduce* menawarkan kecepatan dan skalabilitas yang lebih baik dalam pemrosesan pengujian 6 file data teks berbahasa indonesia pada *Hadoop single node*.
2. Dapat disimpulkan bahwa hasil perbandingan waktu proses dari program *Word Count* dengan *MapReduce* berbasis *Hadoop Distributed File System* dan program *Word Count* tanpa *MapReduce* dengan waktu proses *word count* tanpa *map reduce* lebih cepat. Kedua metode menghasilkan hasil yang baik dalam pemrosesan teks berbahasa Indonesia. Namun, metode *word count* tanpa *MapReduce* memberikan waktu pemrosesan dengan tingkat reduksi tertinggi yaitu 95% pada file 5 MB. Tingkat reduksi menurun secara berkala untuk file yang lebih besar.

5. SARAN

Berdasarkan penelitian yang telah dilaksanakan, berikut saran yang dapat dijadikan acuan untuk penelitian berikutnya :

1. Untuk meningkatkan performa kinerja proses, diperlukan penggunaan perangkat dengan spesifikasi yang tinggi.
2. Mengevaluasi kembali program *Word Count* dengan menggunakan metode *MapReduce* berbasis *Hadoop Distributed File System* untuk pemrosesan paralel teks berbahasa Indonesia bisa menggunakan mode instalasi *Pseudo-Distributed* atau *Fully distributed* untuk kinerja yang lebih efisien.

DAFTAR PUSTAKA

[1] S. Prabowo, “Implementasi Algoritma Penjadwalan untuk pengelolaan Big Data dengan Hadoop,” Indones. J. Comput., vol. 2, no. 2, p. 119, 2017, doi: 10.21108/indojc.2017.2.2.189.

[2] P. A. T. Taqwin, A. B. Osmond, and R. Latuconsina, “Implementasi Metode Mapreduce Pada Big Data Berbasis Hadoop Distributed File System,” e-Proceeding Eng., vol. 5, no. 1, pp. 1013–1020, 2018, [Online]. Available: https://doi.org/10.1109/ACDT.2017.7886152.

[3] J. Hurwitz, A. Nugent, F. Halper, and M. Kaufman, Big Data for Dummies, no. 1. 2013.

[4] A. P. Sujana, “Memanfaatkan Big Data Untuk Mendeteksi Emosi,” Tek. Komput. Unikom, vol. 2, no. 2, pp. 1–4, 2013.

[5] Phady, R. P. (2012). Big Data Processing with Hadoop-MapReduce in Cloud Systems. Institute of Advanced Engineering and Science, International Journal of Cloud Computing and Services Science. Vol.2, No.1. https://doi.org/10.11591/closer.v2i1.1508.

[6] C. Engineering, M. Sahane, S. Sirsat, R. Khan, N. S. Mahal, and R. Baug, “Analysis of Research Data using MapReduce Word Count Algorithm,” vol. 4, no. 5, pp. 8–11, 2015, doi: 10.17148/IJARCCE.2015.4542.

[7] Yunadian, Y., Nuha, H. H., & Prabowo, S. (2020). Pengolahan Data Polling berbasis Media Sosial menggunakan MapReduce pada Framework Hadoop. 7(1), 2581–2591.

[8] Tia, T. K., & Kusuma, W. A. (2018). Model Simulasi Pengembangan Perangkat Lunak Menggunakan Rational Unified Process (Rup). *Teknika: Engineering and Sains Journal*, *2*(1), 33. https://doi.org/10.51804/tesj.v2i1.226.33-40