JURNAL RESISTOR | ISSN 2598-7542 | E-ISSN 2598-9650
Vol. 3 No 1 – April 2020 | https://s.id/jurnalresistor
DOI : https://doi.org/10.31598
Publishing : LPPM STMIK STIKOM Indonesia

ANALISIS UTILISASI *RESOURCE CLUSTERS* PADA HADOOP MENGGUNAKAN *VIRTUALIZATION*

I Kadek Susila Satwika[1], I Putu Susila Handika[2], Made Hanindia Prami Swari[3]

[1,2]Sistem Komputer, Fakultas Teknologi dan Informatika, Institut Bisnis dan Teknologi Indonesia
Jl. Tukad Pakerisan No. 97, Denpasar, Indonesia
[3]Teknik Informatika, UPN "Veteran" Jawa Timur
Jl.Raya Rungkut Madya, Surabaya, Indonesia

Received : April, 2022	Accepted : April, 2022	Published : April, 2022

Abstract

Large amounts of data processing necessitate the use of a dependable infrastructure. Using clustering technology in Big Data processing is a solution for faster processing times. This study evaluated the performance of the Hadoop server using virtualization technology. A varying number of query requests are sent to Hadoop server clusters at the same time. Then, the CPU and memory (RAM) utilization was calculated. According to the test results, the CPU usage on the namenode reached 100% at the start of the process, followed by an increase in CPU usage on the datanode the next time. Meanwhile, the namenode uses the most memory when it receives 25 requests at once. This demonstrates that the namenode can only serve a maximum of 25 requests at the same time.

Keywords: *Hadoop, virtualization, Cluster*

Abstrak

Untuk melakukan pemrosesan data dengan jumlah yang banyak tentu membutuhkan infrastruktur yang handal. Menggunakan teknologi clustering dalam melakukan pemrosesan Big Data menjadi solusi untuk mendapatkan waktu pemrosesan yang lebih cepat. Penelitian ini melakukan pengujian kinerja server hadoop menggunakan teknologi virtualisasi. Hadoop server cluster diberikan request query dengan jumlah bervariasi secara bersamaan. Kemudian dilakukan pengukuran utilisasi CPU dan memori (RAM). Dari hasil pengujian didapatkan penggunaan CPU pada namenode mencapai 100% di awal proses dan di waktu berikutnya diikuti oleh kenaikan penggunaan CPU pada datanode. Sedangkan untuk penggunaan memori paling tinggi terjadi pada namenode saat diberikan 25 request sekaligus. Hal ini menunjukkan namenode hanya mampu melayani maksimal 25 request secara bersamaan.

Kata Kunci: *Hadoop, Virtualisasi, Cluster*

1. PENDAHULUAN

Peningkatan penggunaan media sosial serta peningkatan populasi penggunaan sensor yang berasal dari perangkat IoT, menyebabkan meningkatnya pertumbunhan data [1]. Keberagaman data tersebut dilihat dari jenis data, struktur data, organisasi data, dan juga aksesibilitas dari data itu sendiri. Representasi data merupakan bagian penting dalam *Big Data*, hal ini bertujuan agar data lebih bermakna sehingga dapat dipahami oleh pengguna dan ketika dianalisis oleh computer [2]. *Big Data* menjadi perhatian akademisi dan industri IT. Dalam dunia digital dan komputasi, informasi dapat dikumpulkan dan diproses dengan sangat cepat menggunakan teknologi ini [3].

Sejak beberapa dekade lalu, pertumbuhan data yang dihasilkan dan disimpan untuk memenuhi kebutuhan bisnis di perusahaan telah berkembang pesat. Melalui kemunculan IoT,

komputasi awan, dan kecerdasan buatan, perusahaan menghasilkan lebih banyak data daripada sebelumnya [4]. Keberadaan media sosial dan internet berkontribusi signifikan dalam peningkatan pertumbuhah data sehingga membuat situasi semakin meresahkan. Setiap detik, rata-rata sekitar 6.000 tweet di-tweet di Twitter. Ini berarti lebih dari 350.000 tweet dikirim per menit, 500 juta tweet per hari dan sekitar 200 miliar tweet per tahun [5]. Di hampir semua organisasi, penggunaan aplikasi dan server web menghasilkan banyak sekali data dari log. Ada berbagai sistem lain yang berkontribusi pada pertumbuhan data perusahaan [4]. Facebook, Whatsapp, dan media sosial lainnya mengikuti kecepatan yang sama. Dengan tsunami informasi ini, perusahaan menyadari kebutuhan akan sistem yang dapat memproses data dalam jumlah besar dan mampu menghasilkan sebuah informasi yang bernilai [6].

Untuk menangani jumlah data yang besar dan tidak terstruktur ini, konsep *MapReduce* telah terbukti sebagai teknik yang efisien. MapReduce pertama kali diusulkan oleh Google pada tahun 2004, yang merupakan sebuah framework yang mampu menangani data dalam jumlah yang sangat besar secara paralel dan terdistribusi. Ada banyak framework untuk melakukan operasi pada *Big Data* menggunakan model MapReduce, salah satunya adalah framework Apache Hadoop [7], [8]. Hadoop pertama-tama akan mengumpulkan data dan kemudian menyimpannya dalam sistem penyimpanan yang terhubung ke sistem cluster terdistribusi. Hadoop menggunakan HDFS (*Hadoop Distributed File System*) untuk memproses data dalam jumlah besar pada hard disk yang merupakan node penyimpanan dalam cluster [9], [10].

Menggunakan Hadoop cukup sulit bagi end-users, terutama bagi mereka yang tidak ahli dalam konsep map-reduce. Ini tidak mudah bahkan hanya melakukan tugas sederhana, seperti menghitung atau menjumlahkan, pengguna harus menulis program map-reduce [11]. Apache Hive, yang merupakan tools data warehouse untuk memproses data terstruktur di Hadoop membantu pengguna dengan mudah melakukan *query*, meringkas, dan menganalisis *Big Data* dengan ekspresi seperti SQL yang disebut HiveQL. Hive adalah solusi data warehouse yang dibangun di atas lingkungan Hadoop [12].

Untuk melakukan pemrosesan data dengan jumlah yang banyak tentu membutuhkan infrastruktur yang handal. Menggunakan teknologi clustering dalam melakukan pemrosesan Big Data menjadi solusi untuk mendapatkan waktu pemrosesan yang lebih cepat. Teknologi cluster dirancang untuk menyelesaikan permasalahan pada teknologi klasik (*single computer*), dimana teknologi cluster menggunakan pemrosesan paralel untuk membaca dan memproses kumpulan data yang besar pada banyak disk dan CPU. Hal yang membuat sistem ini bekerja dengan baik dibandingkan teknologi klasik adalah pengorganisasian resource pada computer (CPU, RAM, and *Hard Disk*) dalam melakukan komputasi pada masing-masing Node di dalam cluster [13]. Permasalahan lain muncul ketika ingin merancang sebuah sistem cluster diperlukan resource yang besar, dan tentunya membutuhkan biaya yang sangat besar juga.

Peneliti telah melakukan berbagai penelitian untuk menemukan solusi atas masalah ini dari berbagai sudut pandang. Dari segi infrastruktur, teknologi virtualisasi merupakan alternatif yang dapat diandalkan. Virtualisasi mampu memberikan, skalabilitas, kehandalan, dan kinerja yang lebih baik berkat pendekatan virtualisasi pada beberapa layer [14]. Teknologi virtualisasi mampu untuk membagi serta mengalokasikan sumber daya dari sebuah komputer fisik ke dalam beberapa environment.. Pada konteks ini, virtualisasi adalah menciptakan *virtual machine* (VM) beserta dengan sistem operasi dan aplikasi-aplikasinya [15]. Dengan kata lain, virtualisasi memberikan fleksibilitas dalam membangun cluster komputer.

Beberapa penelitian telah dilakukan sebelumnya terkait dengan pengujian kinerja Hadoop cluster, namun sebagian besar penelitian hanya menguji performansi dari Hadoop itu sendiri. Belum ada yang melakukan penelitian yang berfokus pada sisi server dari cluster node nya. Pada penelitian ini berfokus pada pengukuran utilisasi CPU dan memory server pada masing-masing node. Penelitian ini merancang sebuah sistem Hadoop cluster menggunakan *virtual machine* yang selanjutnya

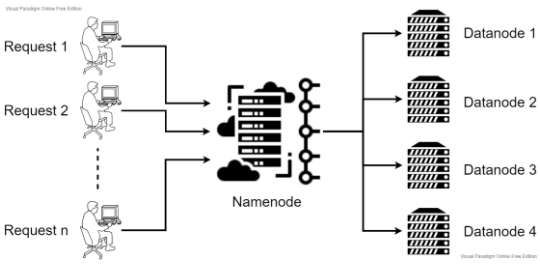
diberi *request* secara bersamaan dalam bentuk *query* script.

2. METODE PENELITIAN

2.1 Skema Infrastruktur

Penelitian ini menggunakan mesin virtual yang dipasang oleh Hadoop dan Hive. 1 dari 5 VM tersebut akan berfungsi sebagai *Namenode* atau sistem yang akan menerima *request* dan meneruskannya ke datanode tersebut. *Datanode* adalah server yang bertindak sebagai pekerja. Penelitian ini mengukur penggunaan memori dan CPU pada masing-masing server dengan variasi jumlah *request* yang dilakukan secara bersamaan. Untuk variasi jumlah *request* yaitu 1 *request*, 5 *request*, 10 *request*, 15 *request*, 20 *request*, dan 25 *request*. Kemudian dari hasil pengujian akan dibandingkan pengaruh peningkatan jumlah *request* tersebut terhadap kinerja memory dan juga CPU.

Gambar 1 menunjukkan topologi infrastruktur Hadoop. Namenode tersebut diberi *request* secara bersamaan dalam bentuk *query script*. Salah satu contoh kueri yang dapat dieksekusi adalah *query* untuk menampilkan jumlah penjualan di setiap departemen. Kemudian akan di variasikan jumlah *request* (n) sesuai dengan yang sudah ditentukan dalam bentuk *script*. Gambar 2 merupakan contoh *script* yang digunakan untuk melakukan *query*.



Gambar 1 Topologi Infrastruktur Hadoop

```
#!/bin/sh

count=25
for i in $(seq $count); do
   hive      -e      "select      dept,
sum(total_retail) from sales group by
dept order by dept" &
done
```

Gambar 2. Script untuk Memberikan *Request* ke Server

Setiap VM memiliki spesifikasi berikut:

Tabel 1. Spesifikasi Server Namenode

Nama	Spesifikasi
CPU	8 inti
RAM	16 GB
Ukuran Disk	100 GB

Tabel 2. Spesifikasi Server Datanode

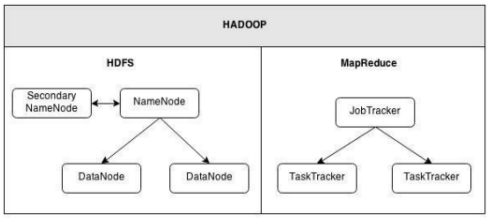
Nama	Spesifikasi
CPU	4 inti
RAM	8 GB
Ukuran Disk	100 GB

2.2 Apache Hadoop

Hadoop merupakan aplikasi framework yang dapat mengelola dan memproses banyak data yang didistribusikan sekaligus dengan melibatkan komputer cluster. Hadoop mengimplementasikan beberapa modul khusus dan berbeda[16]:

- Penyimpanan, pada prinsipnya menggunakan *Hadoop File System* (HDFS) meskipun alternatif lain yang lebih kuat juga tersedia
- Manajemen sumber daya dan penjadwalan untuk tugas komputasi
- Model pemrograman pemrosesan terdistribusi berdasarkan MapReduce
- Utilitas umum dan pustaka perangkat lunak yang diperlukan untuk seluruh platform Hadoop

Secara spesifik, Hadoop terdiri dari dua komponen utama yaitu HDFS dan MapReduce [17] seperti yang ditunjukkan pada Gambar 3.



Gambar 3. Komponen Inti Hadoop

Sebuah cluster kecil di Hadoop dapat terdiri dari satu atau lebih node master dan beberapa node slave. Node master ini terdiri dari Namenode dan JobTracker, sedangkan node slave terdiri dari Datanode dan TaskTracker.

- NameNode adalah beberapa komputer yang berperan sebagai master dan menjadi pusat sistem file pada HDFS. Ini akan mengoordinasikan DataNode untuk melakukan pekerjaan yang perlu dilakukan [18].

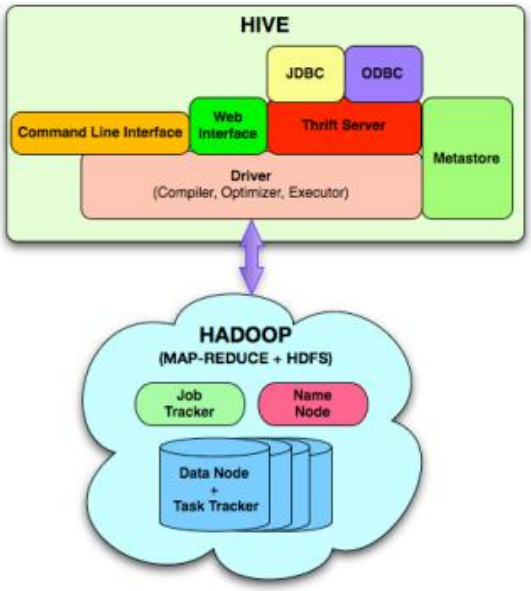
- JobTracker adalah komponen MapReduce yang berfungsi untuk membagi pekerjaan yang ditetapkan ke HDFS menjadi tugas-tugas yang lebih kecil berdasarkan jumlah budak yang tersedia.
- DataNode berada di setiap node slave dan berfungsi untuk mengambil dan menyimpan data pada node slave sesuai dengan instruksi dari NameNode.
- TaskTracker merupakan daemon yang berfungsi untuk menerima tugas yang diberikan oleh JobTracker kemudian mengerjakan tugas tersebut ke dalam Java Virtual Machine (JVM) tersendiri. Dengan menjalankan tugas ke dalam Java Virtual Machine (JVM) terpisah, ini akan mengurangi beban kerja yang dilakukan secara paralel yang diberikan oleh JobTracker.

2.3 Apache Hive

Hive diperkenalkan lebih dari 10 tahun yang lalu, misi pembuatnya adalah untuk mengekspresikan antarmuka seperti SQL di atas Hadoop MapReduce untuk memberikan gambaran umum bagi pengguna dari berurusan dengan detail implementasi tingkat rendah untuk pekerjaan pemrosesan batch paralel mereka [19]. Hive secara tradisional menggunakan MapReduce untuk bekerja secara paralel, dan melakukan langkah-langkah tingkat rendah dalam memproses kueri SQL seperti pemilihan[20]. Gambar 4 adalah Arsitektur sistem dan komponen Apache Hive melalui Hadoop.

Ada 7 blok komponen utama di Hive sebagai berikut:

- Metastore - Fungsi komponen ini adalah menyimpan katalog dan metadata sistem seperti tabel, kolom, partisi dll.
- Driver - Komponen yang mengelola siklus hidup pernyataan HiveQL saat bergerak melalui Hive. Pengemudi juga mempertahankan pegangan sesi dan statistik sesi apa pun.
- *Query* Compiler - Komponen ini bekerja untuk mengkompilasi HiveQL menjadi grafik asiklik yang diarahkan pada tugas pengurangan peta.
- Mesin Eksekusi - Komponen yang menjalankan tugas-tugas yang dihasilkan oleh kompilator dalam urutan ketergantungan yang tepat.
- HiveServer - Fungsi komponen tersebut adalah untuk menyediakan antarmuka hemat dan server JDBC / ODBC dan menyediakan cara untuk mengintegrasikan Hive dengan aplikasi lain.
- Komponen klien seperti Command Line Interface (CLI), UI web, dan driver JDBC / ODBC.

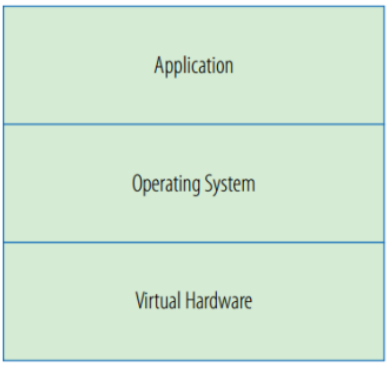


Gambar 4. Arsitektur Apache Hive

Extensibility Interfaces yang mencakup antarmuka SerDe dan Object Inspector yang sudah dijelaskan sebelumnya serta antarmuka UDF (User Defined Function) dan UDAF (User Defined Aggregate Function) yang memungkinkan pengguna untuk menentukan fungsi kustom mereka sendiri.

2.4 Teknologi Virtualisasi

Virtualisasi merupakan metode pembagian sumber daya (*resource*) pada komputer menjadi beberapa *environment* lain, dengan konsep atau teknologi-teknologi seperti emulator, QoS, partisi perangkat keras maupun lunak, dan lainnya. Server yang digunakan untuk menjalankan *software* virtualisasi (*hypervisor*) disebut host. Dimana *hypervisor* akan menciptakan *virtual machine* dengan menggunakan *resource* dari *host* yang disebut guest[21]. *Virtual Machine* atau yang biasa disebut dengan VM adalah sebuah *container* yang menjalankan aplikasi dan sistem operasi guest menggunakan sumber daya dari server fisik dibawahnya (*host*).



Gambar 5. Ilustrasi Virtualisasi

VM pada dasarnya terdiri dari configuration *file* dan *virtual disk file*. *Configuration file* menjelaskan apa saja sumber daya yang akan digunakan oleh VM. Asumsikan VM adalah sebuah casis server yang kosong, configuration *file* berisikan perangkat keras apa saja yang akan berada dalam casis tersebut seperti CPU, memori, penyimpanan, jaringan, CD *drive*, dan lain-lain. Gambar 5 menunjukkan gambar ilustrasi virtualisasi.

3. HASIL DAN PEMBAHASAN

Pengujian akan dilakukan dengan mengukur kinerja memory dan CPU dengan masing-masing akan diberkan *request* dengan jumlah yang berbeda-beda. Setelah itu dianalisis hasil pengujian berdasarkan hasil pengujian yang telah dilakukan.

3.1 Pengujian Kinerja CPU

Pertama dilakukan pengujian terhadap kinerja CPU dengan masing-masing pengujian diberikan jumlah *request* sebagai berikut: 1 *request*, 5 *request*, 10 *request*, 15 *request*, 20 *request*, dan 25 *request*.

Tabel 3 menunjukkan rata-rata penggunaan CPU saat proses *query* terjadi. Pada table terlihat rata-rata penggunaan CPU pada namenode lebih rendah dibandingkan dengan rata-rata penggunaan CPU pada datanode. Hal ini dikarenakan pada namenode hanya berfungsi menyimpan namespace dari filesystem. Semua data akan diteruskan ke datanode untuk dilakukan proses MapReduce [study of apache hadoop]. Sehingga proses yang terjadi pada namenode lebih rendah dibandingkan dengan datanode.

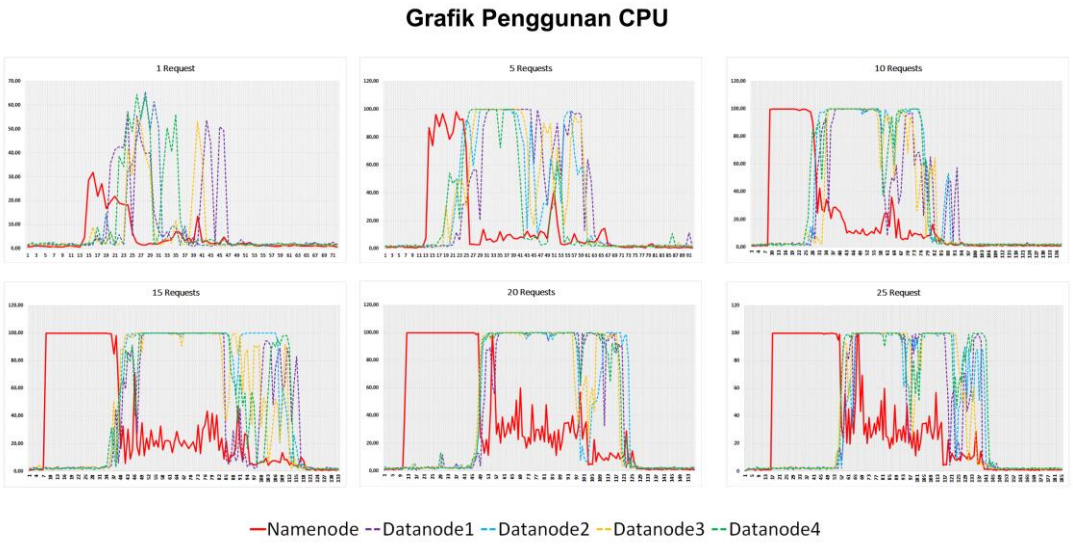
Tabel 3. Rata-rata Penggunaan CPU

Jumlah *Request*	Rata - rata Penggunaan CPU (%)				
	Name Node	Data Node1	Data Node2	Data Node3	Data Node4
1	22,41	38,23	49,59	41,76	40,60
5	28,39	57,60	52,72	53,81	59,54
10	39,32	66,90	79,19	67,39	76,77
15	54,94	71,01	89,15	83,73	81,22
20	55,35	92,91	87,97	88,03	93,90
25	57,18	92,39	95,98	94,33	91,88

Gambar 6 menunjukkan grafik pengujian kinerja CPU dengan masing-masing pengujian diberikan jumlah *request* yang berbeda. Berdasarkan hasil pengujian tersebut didapatkan bahwa penggunaan CPU 100% pada Namenode terjadi ketika diberikan *request* sebanyak 10 *request* secara bersamaan. Dan durasi waktu penggunaan cpu 100% meningkat sering dengan penambahan jumlah *request* yang diberikan. Hal menunjukan semakin banyak diberikan *request* secara bersamaan maka beban kerja dari Namenode juga meningkat. Pada Datanode untuk penggunaan CPU sudah mencapai 100% ketika diberikan *request* sejumlah 5 *request*. Kemudian untuk waktu CPU bekerja pada Namenode dan Datanode terlihat berbeda. Terlihat Namenode memiliki proses lebih awal jika dibandingkan dengan Datanode. Hal ini ditunjukkan oleh grafik berwarna merah yang tinggi di awal waktu pengukuaran, yang kemudian menurun sedangkan grafik yang lainnya (datanode) meningkat. Hal ini disebabkan pada proses Hadoop, Namenode menerima semua data di awal proses. Lalu akan diteruskan ke datanode untuk dilakukan proses MapReduce. Sehingga

penggunaan CPU pada datanode naik sedangkan penggunaan CPU pada namenode turun.

Jika dianalisa berdasarkan waktu proses dari awal (mulai grafik naik) sampai proses selesai (Grafik normal), terlihat bahwa terjadi peningkatan waktu proses yang terjadi. Hal ini menunjukkan semakin besar jumlah *request* yang diberikan maka semakin lama waktu yang proses yang dibutuhkan[22]. Walaupun penggunaan CPU sudah mencapai 100%, namun proses MapReduce masih tetap berjalan. Namun waktu prosesnya yang semakin lama.



Gambar 6. Grafik Peningkatan Penggunaan RAM

3.2 Pengujian Kinerja Memori (RAM)

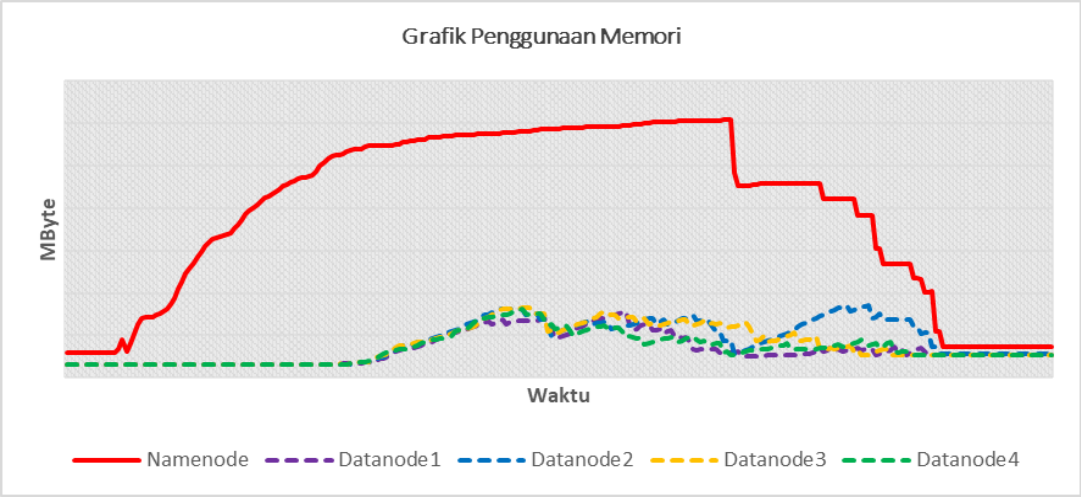
Tabel 4 menunjukkan penggunaan maksimal RAM pada masing-masing node selama proses *query* data. Dari tabel tersebut terlihat penggunaan RAM pada namenode lebih besar dibandingkan dengan datanode. Hal ini dikarenakan seluruh data yang akan diproses disimpan sementara pada namenode yang kemudian diteruskan ke datanode. Pada datanode penggunaan memori tidak terlalu tinggi dikarenakan data dibagi ke dalam node cluster yang terdiri dari 4 server. Pada datanode ini akan dilakukan proses MapReduce.

Tabel 4: Tabel Penggunaan Memori

Jumlah *Request*	Maksimal Penggunaan Memori (MB)				
	Name Node	Data Node1	Data Node2	Data Node3	Data Node4
1	1946	1342	1051	1015	1104
5	4951	2853	2722	2788	2788
10	7634	3169	3358	3068	3144
15	12175	3148	3438	3312	3266
20	14927	3355	3619	3720	3679
25	15837	3850	3813	3578	3829

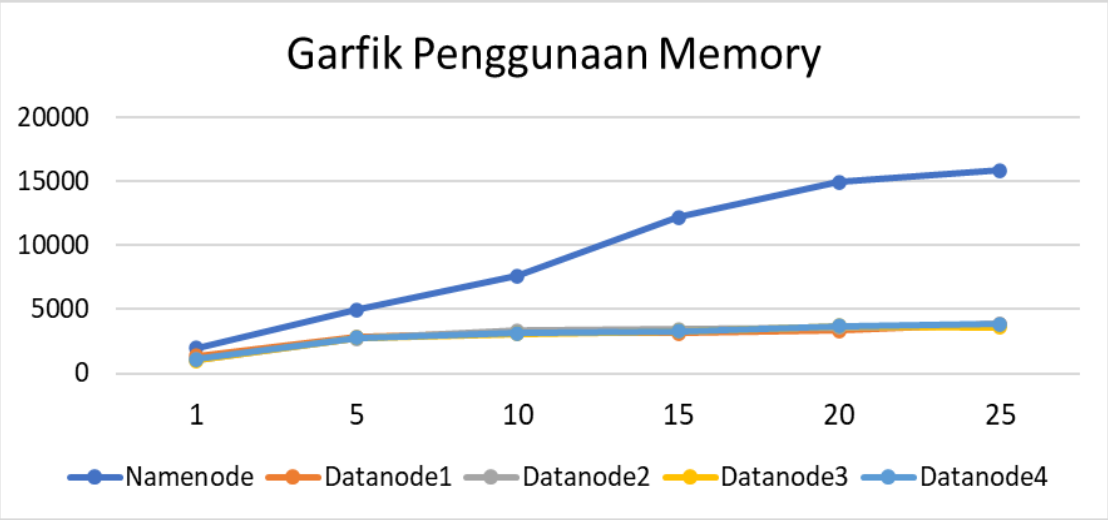
Dalam proses *query* menggunakan Hadoop data awal akan diterima oleh namenode. Semua data akan dikalkulasi dan selanjutnya akan diteruskan ke masing-masing datanode. Gambar 7 menunjukkan pada awal proses peningkatan penggunaan memori sangat meningkata tajam pada namenode, sedangkan pada datanode masig belum ada peningkatan. Kemudian pada pertengahan proses terjadi peningkatan penggunaan memori pada masing-

masing datanode. Hal ini menunjukkan data sudah didistribusikan dari namenode ke datanode. Perbedaan penggunaan memori antara datanode dan namenode sangat terlihat berbeda jauh. Dari hasil percobaan ini didapatkan referensi bahwa untuk membuat sistem hadoop cluster, resource memori untuk namenode harus lebih besar dari resource memori datanode. Ini bergantung juga dari jumlah datanode yang digunakan.



Gambar 7. Grafik Peningkatan Penggunaan Memori

Jumlah *request* yang diberikan ke server sangat berpengaruh pada total penggunaan memori pada server. Gambar 8 menunjukkan semakin banyak *request* yang diberikan akan meningkatkan penggunaan memori pada sever. Khususnya pada namenode. Pada grafik terlihat penggunaan maksimal RAM pada namenode terjadi ketika server diberikan 25 *request* secara bersamaan. Jika server diberikan *request* lebih dari 25 *request*, maka server tidak sanggup untuk memproses datanya dikarenakan resource RAM nya tidak cukup. Namun pada datanode untuk jumlah server sebanyak 4, masih dapat memproses jika diberikan 25 *request*. Hal ini dikarenakan namenode menyimpan semua informasi mengenai data yang di proses pada setiap datanode. Sehingga menyebabkan penggunaan memorinya lebih besar jika dibandingkan dengan penggunaan memori pada datanode.



4. KESIMPULAN

Infrasruktur *Big Data* telah berhasil dibangun yang digunakan untuk melakukan analisis utilisasi resource cluster hadoop menggunakan virtualisasi. Berdasarkan hasil pengujian didapatkan bahwa rata-rata penggunaan CPU pada namenode lebih rendah dibandingkan dengan rata-rata penggunaan CPU pada datanode. Kemudian namenode memiliki proses lebih awal jika dibandingkan dengan datanode. Untuk penggunaan memori, berdasarkan hasil pengujian didapat penggunaan memori pada namenode lebih besar dibandingkan dengan datanode. Hal ini disebabkan karena pada awal proses *query*, namenode akan memproses data terlebih dahulu sehingga penggunaan CPU dan memori sangat tinggi di awal, yang selanjutkan diteruskan ke masing-masing datanode untuk dilakukan proses MapReduce. Pada proses ini menyebabkan penggunaan CPU dan memori pada namenode menjadi meningkat. Selanjutnya penggunaan memori paling tinggi terjadi pada namenode saat diberikan 25 *request* sekaligus. Hal ini menunjukkan namenode hanya mampu melayani maksimal 25 *request* secara bersamaan.

DAFTAR PUSTAKA

[1] A. Wakde, P. Shende, S. Waydande, S. Uttarwar, and G. Deshmukh, "Comparative Analysis of Hadoop Tools and Spark Technology," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Aug. 2018, pp. 1–4, doi: 10.1109/ICCUBEA.2018.8697577.

[2] M. Chen, S. Mao, Y. Zhang, and V. C. M. Leung, *Big Data challenges and technologies, related future prospects*, vol. 6, no. 12. 2010.

[3] A. K. Bhadani and D. Jothimani, "Big data: Challenges, opportunities, and realities," in *Effective Big Data Management and Opportunities for Implementation*, 2016.

[4] J. Patel, "An Effective and Scalable Data Modeling for Enterprise Big Data Platform," in *2019 IEEE International Conference on Big Data (Big Data)*, Dec. 2019, pp. 2691–2697, doi: 10.1109/BigData47090.2019.9005614.

[5] D. Grevenbroich, "Inhalt." [Online]. Available: https://www.blog2social.com/docs/15-twitter-killer-tactics.pdf.

[6] V. Jovanovic, D. Subotic, and S. Mrdalj, "Data modeling styles in data warehousing," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2014, pp. 1458–1463, doi: 10.1109/MIPRO.2014.6859796.

[7] S. Sharma, "An Extended Classification and Comparison of NoSQL Big Data Models," Accessed: Apr. 14, 2022. [Online]. Available: https://arxiv.org/abs/1509.08035.

[8] M. Chen, S. Mao, and Y. Liu, "Big Data: A Survey," *Mob. Networks Appl.*, vol. 19, no. 2, pp. 171–209, Apr. 2014, doi: 10.1007/s11036-013-0489-0.

[9] K. Aziz, D. Zaidouni, and M. Bellafkih, "Real-time data analysis using Spark and Hadoop," in *2018 4th International Conference on Optimization and Applications (ICOA)*, Apr. 2018, pp. 1–6, doi: 10.1109/ICOA.2018.8370593.

[10] H. Dai, S. Zhang, L. Wang, and Y. Ding, "Research and implementation of big data preprocessing system based on Hadoop," in *2016 IEEE International Conference on Big Data Analysis (ICBDA)*, Mar. 2016, pp. 1–5, doi: 10.1109/ICBDA.2016.7509802.

[11] M. Gunay, M. N. Ince, and A. Cetinkaya, "Apache Hive Performance Improvement Techniques for Relational Data," in *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, Sep. 2019, pp. 1–6, doi: 10.1109/IDAP.2019.8875898.

[12] A. Thusoo *et al.*, "Hive - a petabyte scale data warehouse using Hadoop," in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, 2010, pp. 996–1005, doi: 10.1109/ICDE.2010.5447738.

[13] G. DeCandia *et al.*, "Dynamo: Amazon's highly available key-value store," 2007.

[14] D. Kusnetzky, *Virtualization : A Manager ' s Guide*. O'Reilly Media, Inc., 2011.

[15] R. Goldman, *Learning Proxmox VE*. Packt Publishing, 2016.

[16] Robert D. Schneider, *The Executive's Guide To Big Data & Apache Hadoop*. 2012.

[17] A. B. Patel, M. Birla, and U. Nair, "Addressing big data problem using

Hadoop and Map Reduce," in *2012 Nirma University International Conference on Engineering (NUiCONE)*, Dec. 2012, pp. 1–5, doi: 10.1109/NUICONE.2012.6493198.
[18] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, May 2010, pp. 1–10, doi: 10.1109/MSST.2010.5496972.
[19] J. Camacho-Rodríguez *et al.*, "Apache hive: From mapreduce to enterprise-grade big data warehousing," *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 1773–1786, 2019, doi: 10.1145/3299869.3314045.
[20] Cloudera, *Apache Hive Guide*. Cloudera, Inc., 2020.
[21] I. N. B. Hartawan and I. K. S. Satwika, "Rancang Bangun Laboratorium Virtual Berbasis Cloud Computing Di Stmik Stikom Indonesia," *S@CIES*, vol. 7, no. 1, pp. 54–60, Oct. 2016, doi: 10.31598/sacies.v7i1.117.
[22] M. H. P. Swari, I. K. S. Satwika, and I. P. S. Handika, "Performance Analysis of Sales Big Data Processing using Hadoop and Hive in Cloud Environment," in *2020 6th Information Technology International Seminar (ITIS)*, Oct. 2020, pp. 162–166, doi: 10.1109/ITIS50118.2020.9320964.