SNESTIK

Seminar Nasional Teknik Elektro, Sistem Informasi, dan Teknik Informatika

https://ejurnal.itats.ac.id/snestik dan https://snestik.itats.ac.id

Informasi Pelaksanaan :
SNESTIK I - Surabaya, 26 Juni 2021
Ruang Seminar Gedung A, Kampus Institut Teknologi Adhi Tama Surabaya

Informasi Artikel:
DOI : 10.31284/p.snestik.2021.1774

Prosiding ISSN 2775-5126

Fakultas Teknik Elektro dan Teknologi Informasi-Institut Teknologi Adhi Tama Surabaya
Gedung A-ITATS, Jl. Arief Rachman Hakim 100 Surabaya 60117 Telp. (031) 5945043
Email : snestik@itats.ac.id

Pemanfaatan Raspberry Pi 3 Dan Hadoop Sebagai Pembatas Penyimpanan Online Berbasiskan Website

Shah Khadafi[1], Roeddy Oktodijanto[2]

Institut Teknologi Adhi Tama Surabaya[1,2]

khadafi@itats.ac.id

ABSTRACT

Today's computer network is a means to access data at any time through a computer or mobile device. Data storage connected to the internet network is expected to provide optimal service when there is data transfer, upload process or data download process to data storage through the device used by the user. When there are more users, this can cause a flood of data to be stored on the storage device. In addition, the more users with big data they have, this can lead to a full capacity digital main storage device or hard disk. Thus, data storage is needed in the form of a storage server that is used as a backup for data storage on the network computer. This research designed a storage server technology using Apache Hadoop Distributed File System (HDFS) and Java JDK software which is implemented on a Raspberry Pi computer as a server computer, by using storage capacity quota restrictions. In the HDFS system, a user directory will be created that can be integrated with the user on the Raspberry Pi, where each user is imposed with a storage quota capacity limitation of 10GB for each user directory. The results of this study, the storage server can be used jointly by the user, where each user can store 10GB of data in its directory.

Keywords: *storage server; raspberry pi; apache hadoop; disc quota; snestik*

ABSTRAK

Jaringan komputer saat ini, merupakan sarana untuk mengakses data di setiap ruang dan waktu melalui perangkat komputer ataupun perangkat mobile. Penyimpanan data yang terhubung ke jaringan internet diharapkan dapat memberikan layanan yang optimal ketika terjadinya perpindahan data, proses upload ataupun proses download data ke penyimpanan data melaui perangkat yang digunakan oleh *user*. Ketika *user* semakin banyak, hal ini dapat menimbulkan kebanjiran data yang akan disimpan dalam perangkat penyimpanan. Selain itu, semakin banyak user dengan data-data besar yang dimilikinya, hal ini dapat menimbulkan perangkat penyimpanan utama digital atau *harddisc* kapasitasnya penuh. Dengan demikian,

dibutuhkan penyimpanan data berupa *storage server* yang digunakan sebagai cadangan penyimpanan data pada jaringan komputer. Penelitian ini merancang teknologi *storage server* dengan menggunakan perangkat lunak Apache Hadoop Distributed File System (HDFS) dan Java JDK yang diimplementasikan pada komputer Raspberry Pi sebagai komputer *server*, dengan menggunakan pembatasan kuota kapasitas penyimpanan. Rancangan storage server sebagai penyimpanan cadangan ini dapat digunakan bersama-sama namun dengan pembatan jumlah kapasitas dan *directory*. Pembatasan kuota kapasitas yang diberikan sebesar 10GB untuk masing-masing *directory user*. Hasil dari penelitian ini, *storage server* dapat digunakan secara bersama-sama oleh user, dimana masing-masing *user* dapat menyimpan data sebesar 10GB pada directory yang dimilikinya.

Kata kunci: storage server; raspberry pi; apache hadoop; kuota disk, snestik

PENDAHULUAN

Ketika perangkat penyimpanan utama digital atau *harddisc* kapasitasnya sudah penuh atau bahkan hilang, akan timbul permasalahan terhadap tata cara penyimpanan data, yang bisa mengakibatkan data tersebut hilang atau bahkan tidak dapat bisa disimpan di dalam *harddisc*. Apabila dipaksakan untuk disimpan, akibatnya data yang sebelumnya sudah tersimpan harus dihapus terlebih dahulu, kemudian diganti dengan data yang baru. Penelitian yang sebelumnya yang membahas juga tentang *storage server*, tentang *Network Attached Storage* (NAS) adalah sebuah *server* dengan sistem operasi yang dikhususkan untuk melayani akses kecepatan data saat *upload* dan *download* pada *storage server*, tanpa memperhatikan kapasitas kuota penyimpanan [1].

Ada bermacam-macam teknik yang digunakan untuk melakukan *transfer* data dan juga *monitoring packet* pada mekanisme pengiriman data yang terjadi. Penelitian sebelumnya yang membahas tentang monitoring paket data pada *network traffic* melalui layanan file transfer protocol (FTP) menggunakan *open source tools* berbasiskan Linux Ubuntu Snort [2]. Hasilnya dapat mendeteksi adanya serangan terhadap transfer data. Berbagai macam layanan *cloud storage* yang tersedia yang dapat dimanfaatkan oleh pengguna secara *online*, contohnya Google Drive, One Drive, iCloud, pCloud, Dropbox, MediaFire, dll. Pada penelitian sebelumnya yang membahas implementasi layanan *private cloud* berbasiskan open source menggunakan OwnCloud untuk menyimpan data E-Learning[3]. Sebuah perusahaan IT di Indonesia selaku distributor produk IT merancang sebuah sistem penyimpanan berbasiskan *cloud computing* untuk semua karyawannya terkait dengan status *work from home* (WFH) akibat masa pandemic Covid-19. Pada penelitian ini membahas mengenai penyimpanan *big data* berbasiskan sistem operasi Ubuntu 20.04 dan OwnCloud [4].

Trend dari *cloud computing* saat ini menjadi isu terkini untuk menampung beban kerja dengan data yang besar. *Cloud computing* didasarkan pada konsep konsolidasi dan penggabungan sumber daya, tetapi sistem data besar, (seperti Hadoop *file system*) dibangun dengan prinsip tidak ada yang dibagikan, di mana setiap *node* independen dan mandiri [5]. Untuk mengatasi permasalahan *big data*, maka dibutuhkan penyimpanan data utama berupa *private storage server*. *Storage server* ini dirancang khusus layaknya komputer *server* khusus yang digunakan sebagai *back up* penyimpanan data utama menggunakan Raspberry Pi dan *harddisc external*. *Private storage server* memungkinkan pengaturan kapasitas penyimpanan masing-masing user yang mudah diatur dan mengatur jumlah kapasitas data yang dapat disimpan. Rancangan penelitian *storage server* dengan pembatasan kuota kapasitas penyimpanan yang dilakukan ini mengkombinasikan berbagai macam *software* dan *hardware* antara lain yaitu : Apache Hadoop sebagai *software* sistem *storage server* yang berbasiskan website, Raspberry Pi sebagai komputer *server*, Sistem operasi Raspbian OS, *disc quota* sebagai sistem yang membatasi kapasitas alokasi *harddisc* secara *logic* untuk *user* atau *group* tertentu.

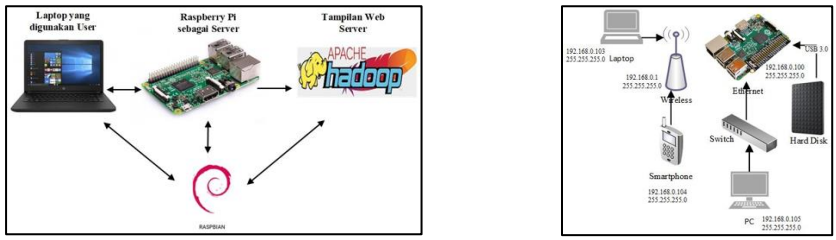
METODE

Perancangan *hardware* dan *software*

Rancangan sistem *storage server* pembatasan kuota kapasitas penyimpanan menggunakan *hardware* dengan spesifikasi khusus, antara lain : Raspberry Pi 3 Model B, Micro SD card Sandisk 32 GB, Harddisc External kapasitas 1 Tb, dan Access Point TP-Link. Sedangkan untuk kebutuhan *software* yang digunakan yang digunakan, antara lain : Sistem operasi Raspian OS., Apache hadoop versi hadoop-3.1.1.tar.gz, dan Java JDK versi jdk1.8.0_192.

Topologi Storage Server

Pada penelitian ini terdapat beberapa bagian yang nampak pada gambar 1.a, bagian pertama yaitu *Hadoop Distributed File System* (HDFS) adalah sistem file yang dirancang untuk menyimpan file besar dengan pola akses data *streaming* yang berjalan pada kelompok perangkat keras komoditas [6], berfungsi sebagai *file system* untuk *storage cluster server*. Bagian kedua perangkat Raspberry Pi sebagai *storage server* dengan sistem operasi Raspbian, terkoneksi dengan *external harddisc* dengan kapasitas 1 TB. Bagian ke tiga, yaitu perangkat user (*client*), dimana *user* pada Hadoop *file system* dapat melakukan penyimpanan data pada *storage cluster server*.



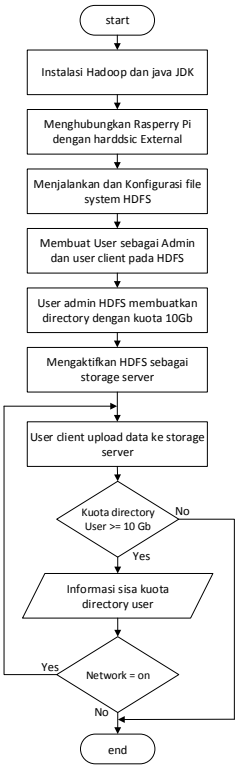
(a) (b)

Gambar 1. a) Blok Diagram Sistem, b) Topologi Jaringan Sistem

Flowchart sistem *storage server* pembatasan kuota kapasitas penyimpanan

Rancangan *flowchart* sistem *storage server* pembatasan kuota kapasitas penyimpanan nampak pada gambar 2. Penjelasan *flowchart* dapat dijelaskan sebagai berikut :

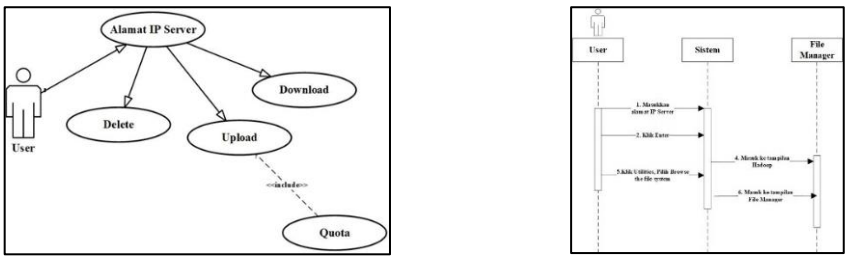
1. Instalasi Hadoop file system dan Java JDK *Java Development Kit* sistem operasi Raspbian. Konfigurasi dimulai dengan mengubah *file* sshd_config pada *directory* /etc/ssh/sshd_config.
2. Menghubungkan Raspberry Pi dengan *harddisc external* melaui port USB versi 3.0
3. Konfigurasi file ~/.bashrc, yang merupakan bash *shell linux* untuk menjalankan *file system* Hadoop dan Java. Perintah export PATH=$PATH:$HADOOP_HOME/bin:$PATH:$JAVA_ HOME/bin:$HADOOP_HOME/sbin, untuk mensingkronkan *directory* Java dengan Hadoop.
4. Untuk mengatur file system Hadoop, dilakukan pembuatan beberapa *user*, antara lain
   i. user Pi sebagai administrator system operasi Raspbian dan membuatkan *directory* dengan merubah hak aksesnya, karena file system Hadoop dibutuhkan kepemilikan *user* pi sebagai *administrator*, dengan perintah hadoop fs -chmod 755 /user/pi.
   ii. *user* Rudy sebagai *user client*, dengan perintah hadoop fs -chmod 777 /user/rudy
   iii. Membuat *public key* untuk *user* pi, perintah ssh-keygen -t rsa -P " -f ~/.ssh/id_rsa. *Public key* disimpan menjadi kunci yang digunakan untuk mengizinkan *user* pi mengakses ssh.
5. Membatasi kuota kapasitas *directory* sebesar pada *user admin* dan *user client* Rudy sebesar 10GB pada masing-masing *directory user*.
   i. *directory user admin* pi dengan perintah, hdfs dfsadmin –setSpaceQuota 10G /user/pi
   ii. *directory user clien* rudy dengan perintah, hdfs dfsadmin –setSpaceQuota 10G /user/rudy
6. Menjalankan sistem dengan perintah start-all.sh (NameNode) atau dfs-all.sh (DataNode) dan yarn-all.sh (*resource manager*), dan juga menjalankan dan *jobhistory server* pada terminal menggunakan perintah mr-jobhistory-daemon.sh start historyserver.



Gambar 2. *Flowchart system storage server* pembatasan kuota kapasitas penyimpanan

Saat membuat unit *cluster* Hadoop, sebagian besar dikonfigurasi dilakukan di *master*, *slave* hanya mengkonfigurasi beberapa pengaturan termasuk core-site.xml, HDFS-site.xml, YARN-site.xml dan mapred-site.xml, pada penelitian yang membahas tentang pengembangan Hadoop *cluster* sebagai *data center* yang menyediakan solusi hemat biaya, daya rendah, kecepatan tinggi bersama dengan dukungan pusat data mikro *big data* [7].

Use case diagram



(a) (b)

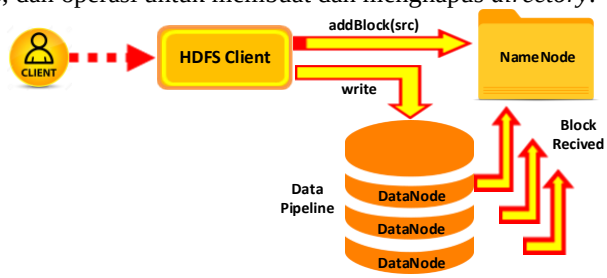
Gambar 3.a) *Use case* Diagram, b) *Sequence diagram* Sistem

Use case diagram merupakan representasi interaksi *user* dengan *website* [8]. Desain perancangan web sistem nampak pada gambar 3, menggunakan diagram UML *use case* karena *user* berinteraksi menggunakan *web*, *activity diagram* (sistem prosedural dan paralel) nampak pada gambar 3.a, dan desain s*equence diagram* (proses interaksi *user* dan sistem) nampak pada gambar 3.b.

Hadoop Distributed File System (HDFS)

Konsep dari Hadoop *file system* berdasarkan *block size* pada *disc* penyimpanan. *File system* untuk *single disc* dibuat untuk menangani data-data (diilustrasikan bentuk blok). Blok *disc* secara *default* berukuran 128 Mb, *file* yang tersimpan dalam HDFS kemudian dipecah menjadi beberapa potongan yang direpresentasikan dalam blok, disimpan dalam *disc* penyimpanan sebagai unit independen. Ilustrasi dari sebuah blok *disc* terhadap penyimpanan dijelaskan sebagai berikut, bila terdapat *file* dengan ukuran 1 MB disimpan dengan ukuran blok 128 MB, file tersebut menggunakan ruang disk 1 MB, bukan 128 MB dalam *disc* penyimpanan [9]. Sistem MapReduce pada Hadoop digunakan dalam penerapan skala besar, yang dapat menghilangkan hambatan pada HDFS untuk meningkatkan kinerja aplikasi dan efisiensi *cluster storage*. Hadoop menyediakan fungsionalitas *built-in* untuk membuat *Map and Reduce* dalam eksekusi tugas [10]. Di dalam Hadoop *framework*, *user* yang sah dapat menjalankan pekerjaan MapReduce apa pun dengan menggunakan perintah jar Hadoop dan *daemon* untuk pelacak pekerjaan yang ada di *namenode* kemudian meneruskan hal yang sama ke node data masing-masing, selanjutnya *task tracer* Hadoop memanggil *instance* baru JVM, setiap *block file* untuk mengeksekusi pekerjaan MapReduce dilakungan dengan cara pemrosesan paralel terdistribusi [11].

Pada gambar 4 nampak cara kerja Hadoop *file system*, terdapat 3 komponen, yaitu, NameNode, DataNode, dan HDFS Client [12]. NameNode mewakili *file* dan *directory*, yang dapat merekam atribut *file* dan *directory* seperti izin, modifikasi dan waktu akses, *namespace* dan juga *quota* ruang disk. NameNode melakukan *maintain* terhadap *namespace tree* dan *mapping* blok *file* ke DataNodes (lokasi fisik data *file*). DataNode mengidentifikasi dan mengenali *mapping* blok yang dimilikinya dari NameNode. Setiap *mapping* blok pada DataNode diwakili oleh dua *file* dalam *file system* asli *local host*. *File* pertama berisi data itu sendiri dan *file* kedua adalah *metadata* blok termasuk *checksum* untuk data blok dan *stamp generation* blok. Ketika NameNode dan DataNode dijalankan, setiap DataNode terhubung ke NameNode dan melakukan *handshake*, tujuannya melakukan verifikasi ID *namespace* dan versi perangkat lunak dari DataNode, dimana *namespace* digunakan oleh *user* mereferensikan *file* dan *directory*. Ketika *user* ingin mengakses *file* atau *directory* pada HDFS, maka aplikasi *user* mengakses menggunakan *HDFS Client*, yang sebelumnya terlebih dahulu menghubungi NameNode untuk melihat daftar DataNodes terkait dengan lokasi blok data dan kemudian membaca isi dari *block* tersebut. HDFS sama dengan kebanyakan sistem file konvensional lainnya yang mendukung operasi untuk membaca, menulis dan menghapus *file*, dan operasi untuk membuat dan menghapus *directory*.



Gambar 4. Interaksi antara *Client* dengan NameNode dan DataNode

Hadoop MapReduce

MapReduce adalah *software framework* yang dapat melakukan pemrosesan terhadap data yang tersimpan dalam Hadoop file system. MapReduce ditulis menggunakan pemrograman Java, yang memiliki kemampuan untuk membagi data masukan menjadi tugas-tugas yang lebih kecil yang dapat dijalankan dalam proses paralel. Hasil dari data yang diproses oleh MapReduce kemudian dikurangi dan disimpan ke HDF *file system*. Di dalam MapReduce terdapat istilah InputSplits, yang merupakan representasi logis dari sebuah data. InputSplits ini yang menyediakan jalur input untuk *single Mapper Job*, dimana satu InputSplits dapat tersebar ke beberapa *block* data.

Tugas khusus dari InputSplits yaitu menyediakan *Mapper* dengan lokasi yang akurat dari data logis, sehingga setiap *Mapper* dapat memproses sekumpulan data yang lengkap yang tersebar di lebih dari satu *block* data [13].

YARN (*Yet Another Resourve Negoitator*)

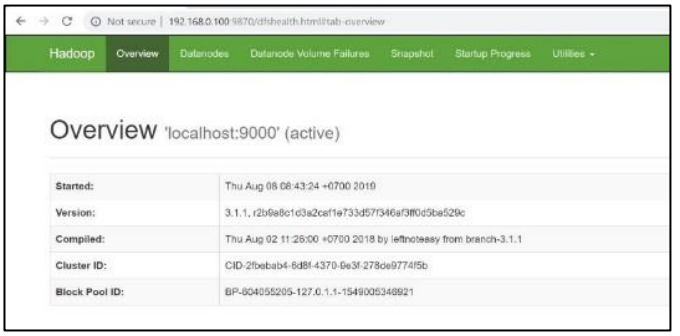
YARN menyajikan *advance scheduller*, yang melakukan *schedulling* yang adil dan mengatur kapasitas pada sumber daya, berbagi skalabilitas [14]. YARN memungkinkan berbagai *framework* perangkat lunak untuk bekerja dengan *hardware* tunggal ketika Hadoop diimplementasikan. Pada *framework* lingkungan YARN berfungsi sebagai *master slave* dimana *resource manager* bertindak seperti *master* dan *application master* berfungsi seperti *slave*. YARB, *resource manager* menangani dan menetapkan sumber daya ke aktivitas aplikasi *cluster*.

Raspberry Pi

Raspberry Pi merupakan sebuah komputer mini dimensi sebesar bungkus rokok yang diciptakan mirip sebuah komputer umumnya, dapat terhubung dengan monitor LED, *mouse* dan *keyboard* standard. Energi listrik yang dibutuhkan sedikit untuk beroperasi, yaitu 5v/2.5A arus DC. Komputer Raspberry Pi memungkinkan untuk dijadikan komputer *server*. Pengukuran pengujian Raspberry Pi yang digunakan sebagai sebuah komputer *server web*, dimana *performance* yang dihasilkan mampu menangani layanan HTTP *request* dari komputer *client* menujukkan *performance* cukup bagus dan tanpa kesalahan [15]. Sebagai perangkat keras dengan komoditas berbiaya yang cukup hemat, Raspberry Pi juga dapat dimanfaatkan sebagai alternatif komputer untuk mengimplementasikan *cluster* Hadoop, yang hasilnya dapat berjalan dengan baik pada Raspberry Pi, dengan distribusi *block file* data Hadoop dapat mengurangi penggunaan *processor* sebanyak 24,14%, pemakaian kapasitas memori menghemat sekitat 8,49%, dan mengakibatkan peningkatan *time completion* sebanyak 31,53% [16].

HASIL DAN PEMBAHASAN

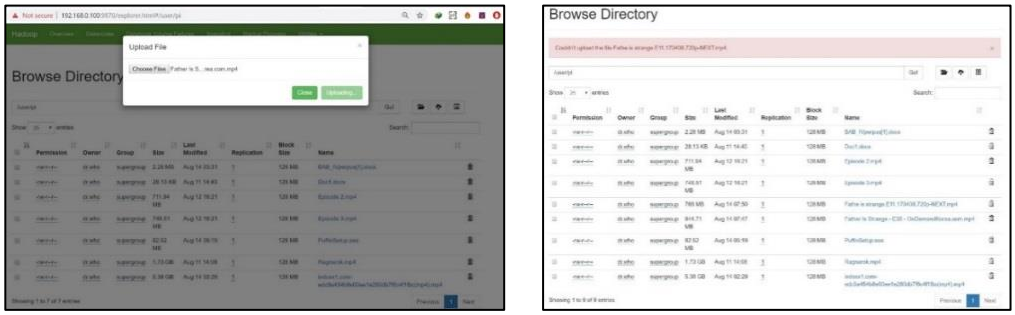
Penngujian *storage server* pembatasan kuota kapasitas penyimpanan



Gambar 5. Tampilan home *interface* web Hadoop

Untuk menjalankan layanan HDFS pada storage *server*, Raspbian menjalankan perintah start-all.sh atau dfs-all.sh (NameNode, DataNode) dan yarn-all.sh (*resource manager*) pada terminal dan *job history server* menggunakan perintah mr-jobhistory-daemon.sh start historyserver. Pada komputer *client* dengan menggunakan web *browser* mengakses IP address *storage server*: 192.168.0.100, seperti yang nampak pada gambar 5.

Selanjutnya dilakukan pengujian melakukan *upload* data oleh *client* user rudy, seperti gambar 6 *file* yang akan diungah *(upload)*. Bila proses upload telah berhasil nampak pada gambar 6.a, sedangkan proses upload data gagal dikarenakan kuota *user directory* telah melebihi 10 Gb, selanjutnya sistem akan menampilkan pesan "*couldn't upload the file*" nampak pada gambar 6.b.



(a) (b)

Gambar 6.a) Tampilan proses upload data sukses, b) Tampilan proses upload data gagal

Pembahasan Data

Dari hasil pengujian yang dilakukan terhadap *user* pi dan *user* rudi, maka analisa data yang dapat disajikan dari penelitian sistem nampak pada tabel 1 di bawah ini.

Tabel 1. Informasi penyimpanan user pada Hadoo file system

Atribut ke-	Nama Atribut	Status User	
		Admin : pi	Client : rudy
1	QUOTA	None	None
2	REM_QUOTA	Inf	Inf
3	SPACE_QUOTA	10.0 Gb	10.0 Gb
4	DIR_COUNT	1	1
5	FILE_COUNT	6	9
6	CONTENT_SIZE	9.9	10
7	PATH_NAME	/user/pi	/user/rudy

KESIMPULAN

Setelah dilakukan pengujian pada sistem server *cloud storage* Hadoop, dapat ditarik beberapa kesimpulan, antara lain :

1. Pembatasan kuota menggunakan *disk quota* pada *directoy user* dapat dilakukan dengan menggunakan Hadoop, sesuai kapasitas yang ditentukan sebesar yaitu 10 GB.
2. Sinkronisasi antara HDFS dengan sistem operasi Raspbian berjalan dengan baik, terutama untuk pembatasan *quota directory* untuk *user client*.
3. Pengaturan Hak akses untuk *directory* masing-masing *user* yang tersimpan di dalam HDFS dilakukan pada sistem Raspbian.
4. Pembuatan *file* baru pada sistem hadoop berlaku kelipatan dari kapasitas *block size* 128 MB, bila melebihi, maka dibuat *block size* baru, sehigga dapat mengurangi pemakaian kapasitas *harddisk* secara langsung, disebut dengan MapReduce.
5. Data yang dikirimkan *host*, NameNode (nama *host*), DataNode (*host*), dan journal node (penjadwalan lalu lintas *data*) diatur oleh HDFS.

DAFTAR PUSTAKA

[1] K. I. Santoso and M. A. Muin, "Implementasi Network Attached Storage (NAS) Menggunakan NAS4Free untuk Media Backup File," *Sci. J. Inform.*, vol. 2, no. 2, p. 123, Feb. 2016, doi: 10.15294/sji.v2i2.5078.

[2] S. Khadafi, Y. D. Pratiwi, and E. Alfianto, "Keamanan FTP Server Berbasiskan IDS Dan IPS Menggunakan Sistem Operasi Linux Ubuntu," *Netw. Eng. Res. Oper.*, vol. 6, no. 1, p. 11, Apr. 2021, doi: 10.21107/nero.v6i1.190.

[3] I. Santiko and R. Rosidi, "Pemanfaatan Private Cloud Storage Sebagai Media Penyimpanan Data E-Learning Pada Lembaga Pendidikan," *J. Tek. Inform.*, vol. 10, no. 2, pp. 137–146, Jan. 2018, doi: 10.15408/jti.v10i2.6992.

[4] Ali Idrus, "Perancangan Owncloud Storage Server Berbasis Ubuntu 20.04 Pada PT. Harrisma Globaltechnologies Jakarta," *PINTER J. Pendidik. Tek. Inform. Dan Komput.*, vol. 4, no. 2, pp. 45–48, Dec. 2020, doi: 10.21009/pinter.4.2.9.

[5] S. A. El-Seoud, H. F. El-Sofany, M. A. F. Abdelfattah, and R. Mohamed, "Big Data and Cloud Computing: Trends and Challenges," *Int. J. Interact. Mob. Technol. IJIM*, vol. 11, no. 2, p. 34, Apr. 2017, doi: 10.3991/ijim.v11i2.6561.

[6] Amol Mahadev Kadam, Pradip K. Deshmukh, Prakash B. Dhainje, and Shriram Institute of Engineering and Technology paniv, "A Review on Distributed File System in Hadoop," *Int. J. Eng. Res.*, vol. V4, no. 05, p. IJERTV4IS050104, May 2015, doi: 10.17577/IJERTV4IS050104.

[7] K. Srinivasan, C.-Y. Chang, C.-H. Huang, M.-H. Chang, A. Sharma, and A. Ankur, "An Efficient Implementation of Mobile Raspberry Pi Hadoop Clusters for Robust and Augmented Computing Performance," *J. Inf. Process. Syst.*, vol. 14, no. 4, pp. 989–1009, Aug. 2018, doi: 10.3745/JIPS.01.0031.

[8] S. Khadafi, A. Salim, Nopendri, R. Prabowo, and C. Anam, "Seminar Nasional Sains dan Teknologi Terapan (SNTEKPAN) VII," in *Menuju Penerapan Teknologi Terbarukan pada Industri 4.0 : Perubahan Industri dan Transformasi Pertumbuhan Digital*, Sep. 2019, pp. 705–710.

[9] T. White, *Hadoop: the definitive guide*, Fourth edition. Beijing: O'Reilly, 2015.

[10] J. Shafer, S. Rixner, and A. L. Cox, "The Hadoop distributed filesystem: Balancing portability and performance," in *2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, White Plains, NY, Mar. 2010, pp. 122–133, doi: 10.1109/ISPASS.2010.5452045.

[11] Gousiya Begum, S. Z. U. Huq, and A. P. S. Kumar, "Sandbox security model for Hadoop file system," *J. Big Data*, vol. 7, no. 1, p. 82, Dec. 2020, doi: 10.1186/s40537-020-00356-z.

[12] 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)*, Incline Village, NV, USA, May 2010, pp. 1–10, doi: 10.1109/MSST.2010.5496972.

[13] D. Papakyriakou, "Benchmarking Raspberry Pi 2 Hadoop Cluster," *Int. J. Comput. Appl.*, vol. 178, no. 42, pp. 37–47, Aug. 2019, doi: 10.5120/ijca2019919328.

[14] N. Deshai, B. V. D. S. Shekhar, V. V. S. S. S. Chakravharthy, and P. S. R. Chowdary, "A cross study on apache hadoop and yarn schedulers," *Int. J. Eng.*, vol. 7, no. 4, pp. 4850–4855, doi: 10.14419/ijet.v7i4.27946.

[15] S. Khadafi, B. D. Meilani, and S. A. Hidayat, "Pengukuran Kompatibilitas Performa Komputer Server Menggunakan Jmeter Pada Raspberry Pi Dan PC Sebagai Layanan Web Server," in *Seminar Nasional Sains dan Teknologi Terapan V 2017 - ITATS*, 2017, vol. 2017, pp. C157–C162.

[16] J. S. Turana, H. Sukoco, and W. A. Kusuma, "Hadoop Performance Analysis on Raspberry Pi for DNA Sequence Alignment," *TELKOMNIKA Telecommun. Comput. Electron. Control*, vol. 14, no. 3, p. 1059, Sep. 2016, doi: 10.12928/telkomnika.v14i3.1886.