IMPLEMENTASI DAN ANALISIS *HADOOP ELEMENT AVAILABILITY* BERDASARKAN *DAEMON LOG MONITORING*

HADOOP ELEMENT AVAILABILITY IMPLEMENTATION AND ANALYSIS BASED ON DAEMON LOG MONITORING

Nurahdiatna Subagya[1], **Adityas Wijajarto** [2], **Ahmad Almaarif**[3]

[1,2,3] Universitas Telkom, Bandung

[1]**yatnasubagja@student.telkomuniversity.ac.id,** [2]**adtwjrt@telkomuniversity.ac.id,**
[3]**ahmadalmaarif@telkomuniversity.ac.id**

Abstrak

***Big Data* pada saat ini adalah isu yang sering dibahas dalam industri, hal ini berkaitan dengan perkembangan teknologi informasi yang *menggunakan Big Data* untuk mendukung industri khususnya dalam *Business Intelligent, Data Analytic,* dan *Machine Learning*. Hadoop adalah sebuah *software framework* untuk komputasi yang *reliable, scalable, parallel,* dan komputasi terdistribusi. Hadoop digunakan untuk memproses, mengatur serta menganalisis dari berbagai macam tipe data misalnya structured, unstructured dan semi-structured. Penggunaan Hadoop yang semakin meningkat juga berbanding lurus dengan ketersediaan atau *availability* dari core services Hadoop itu sendiri. Untuk itu kita perlu melakukan monitoring log dari *daemon* Hadoop. Dengan memonitoring Hadoop *core system daemon* kita dapat melihat *availability* dari Hadoop *element* tersebut ketika proses sedag berjalan.**

Kata kunci : ***Hadoop, core services, element, log, availability***

Abstract

Big Data is currently an issue that is often discussed in the industry, this is related to the development of information technology that uses Big Data to support the industry, especially in Business Intelligent, Data Analytic, and Machine Learning. Hadoop is a software framework for reliable, scalable, parallel, and distributed computing. Hadoop is used to process, organize and analyze various types of data such as structured, unstructured and semi-structured. The increasing use of Hadoop is also directly proportional to the availability of the Hadoop core services itself. For that we need to monitor logs from the Hadoop daemon. By monitoring the Hadoop core system daemon we can see the availability of the Hadoop element when the process is running

Keywords: ***Hadoop, core services, element, log, availability***

Pendahuluan

Big Data pada saat ini adalah isu yang sering dibahas dalam industri, hal ini berkaitan dengan perkembangan teknologi informasi yang menggunakan Big Data untuk mendukung industri khususnya dalam Business Intelligent, Data Analytic, dan Machine Learning. Big Data merupakan sejumlah besar data-data yang penting dan memiliki sejarah yang dikumpulkan, dan merupakan aset paling berharga dari setiap organisasi dan individu yang dimanfaatkan secara cerdas untuk keperluan bisnis agar dapat mendukung sebuah keputusan berdasarkan fakta yang ada daripada persepsi (Gurjit Singh Bhathal, 2019)[1]. Berbagai Industri saat ini memanfaatkan Big Data untuk mendukung bisnisnya. Data-data dapat berasal dari berbagai macam sumber seperti data dari sosial media, data dari berbagai sensor remote, data transaksi, serta log sistem yang dikumpulkan untuk mendukung aktivitas bisnis (Gurjit Singh Bhathal, 2019)[1]. Untuk mengumpulkan Big Data dan melakukan pengolahan terhadap Big Data, diperlukan sebuah infrastruktur mumpuni yang dapat digunakan untuk menjalankan sistemnya. Salah satu framework pengolahan Big Data yang saat ini sering digunakan oleh industri yaitu Hadoop.

Keamanan sistem merupakan hal yang perlu dipikirkan dalam komunikasi antar mesin, baik secara offline maupun online. Hal mendasar yang dapat dijadikan acuan dalam keamanan sistem yaitu Thriad of CIA (Confidentiality, Integrity, dan Availablity). Jika dijelaskan secara singkat, Confidentiality atau kerahasiaan mengacu kepada upaya agar informasi tidak dapat diakses oleh pihak yang tidak berwenang. Integrity mengacu pada sebuah metode atau cara untuk menjaga agar data atau informasi tidak dapat dimanipulasi atau diubah oleh pihak yang tidak berwenang. Dan konsep terakhir yaitu Availability, menjelaskan mengenai sebuah sistem harus terjaga ketersediaannya.

Untuk itu penilitian ini bertujuan untuk melakukan implementasi dan analisis hadoop element availability berdasarkan daemon log monitoring, untuk melakukan langkah teknis mengenai bagaimana sebuah log monitoring system dapat menjaga ketersediaan atau availability dari Hadoop core services.

1. Dasar Teori

2.1 *Big Data*

Big Data merupakan sejumlah besar data-data yang penting dan memiliki sejarah yang dikumpulkan, dan merupakan aset paling berharga dari setiap organisasi dan individu yang dimanfaatkan secara cerdas untuk keperluan bisnis agar dapat mendukung sebuah keputusan yang berdasarkan fakta yang ada daripada persepsi (Gurjit Singh Bhathal, 2019). Berdasarkan definisi, Big Data memiliki 3 karakteristik yang dapat menggambarkan sebuah Big Data, yaitu Volume, Velocity, dan Variety. Volume maksudnya disini adalah suatu Big Data dapat dikenali berdasarkan ukuran datanya, Big Data memiliki ukuran data yang besar dari rata-rata ukuran data. Velocity adalah ukuran kecepatan dari pertukaran data yang dialami oleh kumpulan data-data tersebut. Jadi dapat dikatakan Big Data selain memiliki Volume yang besar tapi kecepatan dari pertukaran ataupun peningkatan datanya itu memiliki kecepatan diatas rata-rata. Kemudian yang ketiga yaitu Variety, yaitu Big Data memiliki berbagai macam jenis mulai dari file berupa audio, video, text, log, data transaksi, data dari sensor, dan sebagainya.

2.2 Hadoop

Hadoop adalah software framework open-source yang digunakan untutk menyimpan, mengolah, memproses, dan menganalisis sejumlah besar data dari berbagai tipe data, yaitu data terstruktur, tidak terstruktur, dan semi-terstruktur (Dharminder Yadav, 2019). Hadoop framework juga digunakan oleh berbagai macam perusahaan terkenal seperti Google, Yahoo, Amazon dan IBM untuk mendukung aplikasinya dalam menghadapi sejumlah besar data (Refik Samet, 2019). Hadoop menyediakan sebuah file sistem terdistribusi yang disebut Hadoop Distributed File System (HDFS) yang menyimpan data di tiap node pada cluster. Selain itu Hadoop mengimplementasikan model pola komputasi yang disebut MapReduce, di mana aplikasi dibagi menjadi banyak pecahan di mana tiap pecahan dapat diproses di tiap node pada sebuah cluster. Framework Hadoop memberi ketersediaan dan pergerakan data dalam aplikasi serta bandwidth tinggi pada cluster. MapReduce dan HDFS dirancang agar kerusakan pada node ditangani secara otomatis oleh framework (Agarwal, 2016).

2.3 *Triad Of CIA*

YARN Ada tiga aspek utama dalam keamanan informasi, hal itu adalah CIA (Confidentiality, Integrity, dan Availability) (Rahardjo, 2017). CIA atau Triad of CIA, merupakan 3 aspek dasar yang digunakan untuk menjelaskan bagaimana membangun sebuah sistem yang aman.

2.3.1 *Confidentiality*

Confidentiality menjelaskan mengenai kerahasiaan data menjadi aspek yang penting dalam keamanan informasi. Confidentiality berarti data atau informasi penting lainnya hanya boleh diakses oleh orang yang berhak atau memiliki privilege.

2.3.2 *Integrity*

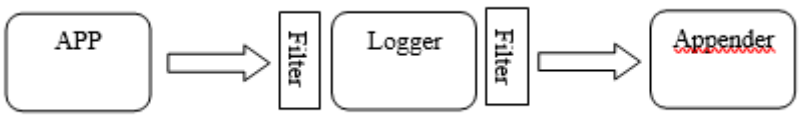
Integrity menjelaskan mengenai sebuah sistem yang aman seharusnya terlindungi dari modifikasi oleh orang yang tidak mempunyai hak akses. Integrity menyatakan data tidak boleh berubah tanpa izin dari yang berhak

2.3.3 *Availability*

Availability adalah situasi dimana informasi tersedia ketika informasi tersebut dibutuhkan. Availability mencakup mengenai bagaimana sistem yang sedang digunakan dapat tetap bekerja dengan baik. Ancaman yang didapatkan dari sisi availability yaitu ancaman berupa data dan informasi yang berada didalam komputer atau server dihapus atau dihancurkan oleh orang yang tidak dikenali.

2.6 Log4J Monitoring

Log4j adalah sebuah library yang dibuat oleh apache, yang berfungsi untuk mencetak suatu log. Log4j memiliki beberapa komponen utama yaitu Logger, Appender, Filter,dan Layout. Untuk menjelaskan komponen Log4j dapat dilihat pada Gambar xxx:



Gambar 2 Alur Dari Log4J Monitoring

3. Metodologi Penelitian

3.1 Model Konseptual

Model konseptual merupakan suatu gambaran logis dari suatu masalah yang digambarkan dalam rangkaian konsep berdasarkan aspek hipotesis dan teoritis. Model konseptual dapat digunakan untuk membantu peneliti dalam merumuskan pemecahan masalah, perumusan solusi, serta evaluasi dari permasalahan yang ada. Model konseptual juga dapat membantu memperjelas masalah yang ada dan dapat mengidentifikasikan faktor-faktor masalah yang relevan, serta memberikan penjelasan yang saling terkait sehingga pemecahan masalah dapat dilakukan dengan lebih mudah.

3.2 Sistematika Penyelesaian Masalah

Dalam menyelesaikan permasalahan pada penelitian ilmiah ini, dilakukan sebuah langkah kerja ataupun tahapan yang terstruktur dan sistematis untuk memecahkan permasalahan tersebut. Dengan tahapan tersebut dapat mempermudah dalam melakukan penelitian untuk memecahkan masalah yang telah ditemukan. Berikut tahapan yang dilakukan pada pelaksanaan penelitian ini dimulai dengan tahapan identifikasi masalah, kemudian melakukan review literatur, tahap perancangan, kemudian pengujian, lalu melakukan tahap analisis dan tahap akhir penelitian. Untuk gambaran sistematikanya dapat dilihat pada gambar xxx:

3.2.1 Tahap Identifikasi Masalah

Pada tahap ini dilakukan pencarian masalah untuk membuat sebuah topik penelitian berdasarkan informasi-informasi yang telah dikumpulkan. Kemudian ketika masalahnya telah diemukan mencoba mencari latar belakang dari permasalahan yang telah ditemukan, kemudian merumuskan poin-poin permasalahan yang ingin selesaikan dan menentukan batasan-batasan masalah yang akan dibahas.

3.2.2 Tahap Penyusunan Hipotesa

Pada tahapan ini dilakukan pencarian-pencarian jurnal ataupun sumber literatur lainnya mengenai topik terkait, untuk mendapatkan informasi mengenai penelitian-penenlitian terdahulu dan mengumpulkan dasar-dasar teori keilmuan yang dibutuhkan untuk menyusun hipotesa.

3.2.3 Tahap Eksperimen

Pada tahapan ini sudah mulai mempelajari sistem yang akan dibangun dan juga melakukan pemasangan sistem sesuai dengan yang dibutuhkan. Dalam tahap perancangan ini terdapat iterasi aktivitas jika sistem yang dalam tahap pemasangan masih gagal atau sumber referensi yang digunakan untuk penerapan sistem masih belum cukup. Tahapan perancangan baru dapat dikatakan selesai ketika sistem yang kita pasang sudah berhasil fungsionalitasnya sesuai dengan yang diinginkan. selanjutnya melakukan pengujian untuk medapatkan hasil dari penelitian. Pada tahap ini baru pengujian dapat dikatakan selesai, ketika kita mendapatkan informasi yang dibutuhkan untuk dilakukan sebuah analisa. Tahapan ini akan dilakukan berulang-ulang sehingga tidak mendapatkan hasil baru lagi dari pengujian Vulnerability Assessment yang dilakukan.

3.2.1 Tahap Analisis

Pada tahapan ini setelah mendapatkan data-data atau informasi dari hasil pengujian dan melakukan sebuah analisis yang sekiranya dengan hasil analisis tersebut dapat menyelesaikan permasalahan yang sebelumnya ditemukan pada tahap identifikasi masalah, dan tentunya sesuai dengan batasan masalah yang sebelumnya telah ditentukan.

3.2.1 Tahap Akhir

Ditahap ini telah melakukan analisa kemudian analisa tersebut ditulis didalam laporan hasil penelitian dan membuat sebuah kesimpulan dan saran terkait masalah yang ada dan hasil temuan pada penelitian yang ditemukan.

4. Analisa Perancangan Implementasi Dan Pengujian

4.1 Rancangan Sistem

Perancangan sistem dan analisis yang akan dilakukan pada penelitian ini membutuhkan environment dan scenario pengujian dalam melakukan simulasi. Environment pendukung yang digunakan pada tahap perancangan dan analisis ini terdiri dari hardware dan software. Rincian hardware dan software yang akan digunakan lebih jelasnya dapat dilihat sebagai berikut:

4.1.1 *Hardware*

Perangkat keras yang digunakan dikategorikan menjadi dua komponen yaitu perangkat keras fisik dan perangkat keras mesin virtual. Untuk rincian spesifikasi perangkat keras dapat dilihat pada tabel dibawah:

Komponen	Informasi	
Spesifikasi Perangkat *Host*	*Processor*	Intel(R) Core™ i7-7700HQ CPU @ 2.80GHz (8 CPUs), ~2.8GHz.

	Memory	16384MB DDR4 2400MHz
	Hard Disk	TOSHIBA MQ01ABD100 1TB
	Solid State Disk	SK Hynix SC311 SATA 128GiB
	System Type	64-bit *Operating System*
	System Model	Inspiron 15 7000 *Gaming*
	Operating System	Windows 10 *Education* 64-Bit (10.0, Build 19043)
Spesifikasi Perangkat *Guest*	*Processor*	Intel(R) Core™ i7-7700HQ CPU @ 2.80GHz, ~2.8GHz.
	Memory	3 GiB
	Hard Disk	30 GiB
	System Type	X86_64 GNU/Linux
	Operating System	Debian 10.10

4.1.2 *Software* yang digunakan pada penelitian ini dapat dilihat pada tabel berikut:

No.	*Software*	Versi	Keterangan
1.	Hadoop	3.3.1	Data Analytic *Framework* meliputi HDFS,MapReduce, dan YARN
2.	Java Development Kit	8	Java Development Kit, untuk membangun aplikasi dan komponen menggunakan bahasa pemrograman Java
3.	VMWare	15.5.1	Perangkat lunak *virtualisasi*.
4.	Log4j		Java *Based Log Utility.*
5.	Debian	10.10	Sistem Operasi yang digunakan.

4.2 Komponen Eksperimen

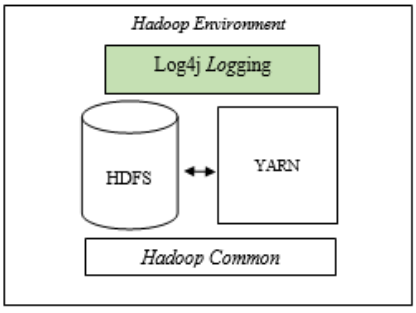
Komponen Eksperimen digunakan untuk menjelaskan arsitektur sistem dari Hadoop dan log monitoring yang akan di implementasikan. Berikut komponen ekperimen yang akan digunakan:

4.2.1 Pemetaan Alamat IP

No.	Perangkat	*Network Adapter*	Keterangan Alamat IP
1.	Hadoop (*Guest1*)	Bridge Adapter Vmnet0	*Automatic DHCP*
2.	*Host*	Wi-Fi Adapter	*Automatic DHCP*

Pada Tabel xxxx dijelaskan pemetaan alamat IP dari perangkat yang digunakan. Pada penelitian ini digunakan mesin virtual sebagai Hadoop SingleNode dengan pengaturan adapter “Bridge” dan IP address Automatic DHCP. Dan sebagai client untuk mengakses WebUI digunakan Main OS atau Host dengan pengaturan IP address DHCP pada adapter WIFI-nya.

4.2.2 Arsitektur Implementasi Pencatatan *log* dan *monitoring*

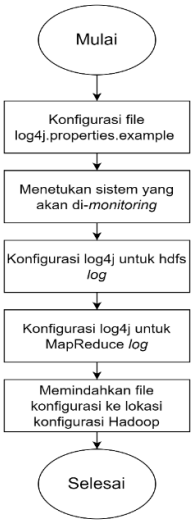


Implementasi yang dilakukan pada penelitian ini akan menggunakan komponen tambahan yang di konfigurasi untuk melakukan monitoring log. Monitoring log dilakukan dengan menggunakan Log4j sebagai alat untuk mengambil data log dari sistem yang sedang berjalan. Log akan dikumpulkan sesuai dengan konfigurasi yang diberikan pada direktori “conf” pada Hadoop yang telah disesuaikan variabel-variabel konfigurasinya pada file Log4j.properties tersebut, kemudian catatan log akan disimpan pada direktori ‘logs” pada hadoop.

4.4 Skenario Pengujian

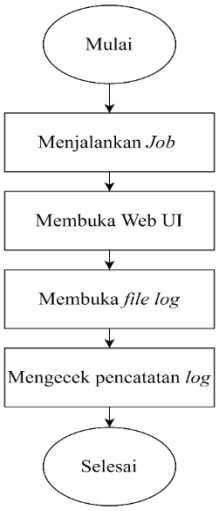
4.4.1 Skenario pemasangan system Log4j

Pemasangan dilakukan dengan melakukan kofigurasi pada file "Log4j.properties.example" yang bertujuan untuk menghubungkan pencatatan log dan sistem, kemudian dipindahkan kedalam file konfigurasi Hadoop. Langkah yang dilakukan untuk melakukan implementasi dapat dilihat pada gambar xxx :



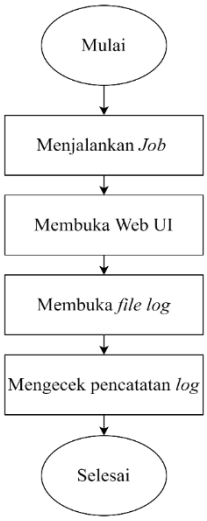
Penjelasan tahapan:
a.Pertama-tama lakukan konfigurasi pada file Log4j.properties.example.
b.Kemudian tentukan sistem yang akan dikonfigurasi, pada eksperimen penelitian ini yaitu HDFS dan MapReduce.
c.Selanjutnya ditahap ketiga, yaitu memasukkan konfigurasi yang diperlukan untuk monitoring log HDFS.
d.Ditahap ketiga, yaitu memasukkan konfigurasi yang diperlukan untuk monitoring log MapReduce.
e.Lalu memindahkan file konfigurasi dengan mengubah namanya file menjadi Log4j.properties ke direktori file konfigurasi Hadoop.

4.4.2 Skenario pemasangan system HDFS



Pada gambar hhhh menjelaskan Langkah-langkah pengujian log monitoring pada HDFS untuk rinciannya, sebagai berikut:
a.Pertama-tama, dilakukan percobaan running job pada Hadoop.
b.Kemudian, jika job telah running, kita perlu mengakses resource manager Web UI pada <ip address>:8088.
c.Selanjutnya, buka tab log pada WebUI tersebut.
d.Dan terakhir kita perlu mengecek apakah log yang telah dikonfigurasi tersebut benar mencatat proses daemon yang bekerja ketika Hadoop yang sedang melakukan proses berjalan atau tidak.

4.4.3 Skenario pencatatan Log YARN



Pada gambar hhhh menjelaskan Langkah-langkah pengujian log monitoring pada proses YARN untuk rinciannya, sebagai berikut:
a.Pertama-tama, dilakukan percobaan running job pada Hadoop.
b.Kemudian, jika job telah running, kita perlu mengakses resource manager Web UI pada <ipaddress>:8088.
c.Selanjutnya, buka tab log pada WebUI tersebut.
d.Dan terakhir kita perlu mengecek apakah log yang telah dikonfigurasi tersebut benar mencatat proses daemon yang bekerja ketika Hadoop yang sedang melakukan proses berjalan atau tidak.

5. Implementasi Dan Pengujian

5.1 Hasil Implemtasi

5.1.1 Konfigurasi Log4j monitoring HDFS

Untuk menjalankan pencatatan log, kita perlu memasukkan beberapa syntax konfigurasi pada file. Untuk penjelasan lebih detailnya dapat dilihat pada Tabel V-1:

Tabel V-1 Konfigurasi HDFS Log Audit

No.	*Syntax*	Keterangan
1.	hdfs.audit.logger=INFO, RFAAUDIT	Mengatur Level Audit dan Logger
2.	hdfs.audit.log.max*file*size=256MB	Mengatur ukuran maksimal *file log*
3.	hdfs.audit.log.maxbackupindex=20	Mengatur max backup index
4.	log4j.logger.org.apache.hadoop.hdfs.*server*.*NameNode* .FSName*system*.audit=${hdfs.audit.logger}	Memanggil object yang akan di audit audit
5.	log4j.additivity.org.apache.hadoop.hdfs .*server*.*NameNode*.FSName*system*.audit=false	Mengatur Log4j *additivity*
6.	log4j.appender.RFAAUDIT=org.apache. log4j.Rolling*File*Appender	Mengatur *appender*
7.	log4j.appender.RFAAUDIT .*File*=${hadoop.log.dir}/hdfs-audit.log	Memetakan nama *file log*
8.	log4j.appender.RFAAUDIT.layout= org.apache.log4j.PatternLayout	Mengatur *layout log*
9.	log4j.appender.RFAAUDIT.layout. ConversionPattern=%d{ISO8601} %p %c{2}: %m%n	Mengatur *convertion pattern layout*
10.	log4j.appender.RFAAUDIT.Max*File*Size= ${hdfs.audit.log.max*file*size}	Mengatur ukuran maksimal *file appender*
11.	log4j.appender.RFAAUDIT.MaxBackupIndex= ${hdfs.audit.log.maxbackupindex}	Mengatur *max backup index appender*

5.1.2 Konfigurasi Log4j Monitoring YARN

Konfigurasi juga perlu dilakukan untuk mencatat log dari YARN. Ada 2 file yang perlu kita konfigurasi untuk mencatat YARN yaitu sebagai berikut:

a. Resource Manager

Berikut adalah konfigurasi yang dilaakukan untuk memonitoring core services YARN Resource Manager:

Tabel V-2 Konfigurasi YARN Log Audit 1

No.	*Syntax*	Keterangan
1.	rm.audit.logger=INFO,NullAppender	Mengatur Level Audit dan Logger
2.	rm.audit.log.max*file*size=256MB	Mengatur ukuran maksimal *file log*
3.	rm.audit.log.maxbackupindex=20	Mengatur max backup index
4.	log4j.logger.org.apache.hadoop.yarn.*server* .resourcemanager.RMAuditLogger=${rm.audit.logger}	Memanggil object yang akan di audit audit
5.	log4j.additivity.org.apache.hadoop.yarn .*server*.resourcemanager.RMAuditLogger=false	Mengatur Log4j *additivity*
6.	log4j.appender.RMAUDIT= org.apache.log4j.Rolling*File*Appender	Mengatur *appender*
7.	log4j.appender.RMAUDIT.*File*=${hadoop.log.dir}/rm-audit.log	Memetakan nama *file log*
8.	log4j.appender.RMAUDIT.layout =org.apache.log4j.PatternLayout	Mengatur *layout log*
9.	log4j.appender.RMAUDIT.layout .ConversionPattern=%d{ISO8601} %p %c{2}: %m%n	Mengatur *convertion pattern layout*
10.	log4j.appender.RMAUDIT.Max*File*Size= ${rm.audit.log.max*file*size}	Mengatur ukuran maksimal *file appender*
11.	log4j.appender.RMAUDIT.MaxBackupIndex= ${rm.audit.log.maxbackupindex}	Mengatur *max backup index appender*

b. Node Manager

Tabel V-3 Konfigurasi YARN Log Audit 2

No.	*Syntax*	Keterangan
1.	nm.audit.logger=INFO,NullAppender	Mengatur Level Audit dan Logger
2.	nm.audit.log.max*file*size=256MB	Mengatur ukuran maksimal *file log*
3.	nm.audit.log.maxbackupindex=20	Mengatur max backup index
4.	log4j.logger.org.apache.hadoop.yarn.*server* .nodemanager.NMAuditLogger=${rm.audit.logger}	Memanggil object yang akan di audit audit

5.	log4j.additivity.org.apache.hadoop.yarn .*server*.nodemanager.NMAuditLogger=false	Mengatur Log4j *additivity*
6.	log4j.appender.NMAUDIT= org.apache.log4j.Rolling*File*Appender	Mengatur *appender*
7.	log4j.appender.NMAUDIT.*File*=${hadoop.log.dir}/nm-audit.log	Memetakan nama *file log*
8.	log4j.appender.NMAUDIT.layout =org.apache.log4j.PatternLayout	Mengatur *layout log*
9.	log4j.appender.NMAUDIT.layout .ConversionPattern=%d{ISO8601} %p %c{2}: %m%n	Mengatur *convertion pattern layout*
10.	log4j.appender.NMAUDIT.Max*File*Size= ${nm.audit.log.max*file*size}	Mengatur ukuran maksimal *file appender*
11.	log4j.appender.NMAUDIT.MaxBackupIndex= ${nm.audit.log.maxbackupindex}	Mengatur *max backup index appender*

5.1.3 Tampilan hasil monitoring log HDFS

a. NameNode Log

```
2021-07-30 03:05:11,908 INFO org.apache.hadoop.hdfs.StateChange: DIR* completeFile:
/user/hadoop/QuasiMonteCarlo_1627614251199_118044768/out/_SUCCESS is closed by
DFSClient_NONMAPREDUCE_1920248746_1
2021-07-30 03:05:11,925 INFO org.apache.hadoop.hdfs.StateChange: DIR* completeFile: /tmp/hadoop-
yarn/staging/hadoop/.staging/job_1627614037461_0001/COMMIT_SUCCESS is closed by
DFSClient_NONMAPREDUCE_1920248746_1
2021-07-30 03:05:11,949 INFO org.apache.hadoop.hdfs.StateChange: DIR* completeFile: /tmp/hadoop-
yarn/staging/hadoop/.staging/job_1627614037461_0001/job_1627614037461_0001_1.jhist is closed by
DFSClient_NONMAPREDUCE_1920248746_1
2021-07-30 03:05:11,970 INFO org.apache.hadoop.hdfs.StateChange: BLOCK* allocate
blk_1073741958_1134, replicas=127.0.0.1:9866 for /tmp/hadoop-
yarn/staging/history/done_intermediate/hadoop/job_1627614037461_0001.summary_tmp
2021-07-30 03:05:11,996 INFO org.apache.hadoop.hdfs.StateChange: DIR* completeFile: /tmp/hadoop-
yarn/staging/history/done_intermediate/hadoop/job_1627614037461_0001.summary_tmp is closed by
DFSClient_NONMAPREDUCE_1920248746_1
```

Gambar V-1 Hasil Log HDFS 1

Pada Gambar V-1 diatas ditampilkan mengenai informasi pencatatan log yang telah terimplementasi pada namenode.

b. DataNode Log

```
2021-07-30 03:05:15,941 INFO
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Deleted BP-
784522513-127.0.0.2-1624896655873 blk_1073741951_1127 URI
file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-784522513-127.0.0.2-
1624896655873/current/finalized/subdir0/subdir0/blk_1073741951
2021-07-30 03:06:36,955 INFO
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Scheduling
blk_1073741958_1134 replica FinalizedReplica, blk_1073741958_1134, FINALIZED
  getNumBytes()     = 451
  getBytesOnDisk()  = 451
  getVisibleLength()= 451
  getVolume()       = /opt/bitnami/hadoop/var/lib/dfs/data
  getBlockURI()     = file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-784522513-127.0.0.2-
1624896655873/current/finalized/subdir0/subdir0/blk_1073741958 for deletion
2021-07-30 03:06:36,956 INFO
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Deleted BP-
784522513-127.0.0.2-1624896655873 blk_1073741958_1134 URI
file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-784522513-127.0.0.2-
1624896655873/current/finalized/subdir0/subdir0/blk_1073741958
```

Gambar V- 2 Hasil Log HDFS 2

5.1.4 Tampilan Hasil monitoring Log YARN

a. Resource Manager Log

```
2021-07-30 03:05:18,771 INFO
org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.ParentQueue: Application removed
- appId: application_1627614037461_0001 user: hadoop leaf-queue of parent: root #applications: 0
2021-07-30 03:05:18,793 INFO
org.apache.hadoop.yarn.server.resourcemanager.RMAppManager$ApplicationSummary:
appId=application_1627614037461_0001,name=QuasiMonteCarlo,user=hadoop,queue=default,state=FINISHE
D,trackingUrl=http://localhost:8088/proxy/application_1627614037461_0001/,appMasterHost=localhost
,submitTime=1627614255020,startTime=1627614255082,launchTime=1627614256273,finishTime=16276143122
49,finalStatus=SUCCEEDED,memorySeconds=319019,vcoreSeconds=151,preemptedMemorySeconds=0,preempted
VcoreSeconds=0,preemptedAMContainers=0,preemptedNonAMContainers=0,preemptedResources=<memory:0\,
vCores:0>,applicationType=MAPREDUCE,resourceSeconds=319019 MB-seconds\, 151 vcore-
seconds,preemptedResourceSeconds=0 MB-seconds\, 0 vcore-
seconds,applicationTags=,applicationNodeLabel=,diagnostics=,totalAllocatedContainers=12
2021-07-30 03:05:18,791 INFO org.apache.hadoop.yarn.server.resourcemanager.amlauncher.AMLauncher:
Cleaning master appattempt_1627614037461_0001_000001
2021-07-30 03:10:38,932 INFO
org.apache.hadoop.yarn.server.resourcemanager.scheduler.AbstractYarnScheduler: Release request
cache is cleaned up
```

Gambar V-3 Hasil Log YARN 1

Pada Gambar V-3 diatas ditampilkan mengenai informasi pencatatan log yang telah terimplementasi. Informasi log yang didapatkan telah sesuai dengan konfigurasi yang diberikan.b. Node Manager Log

```
2021-07-30 03:05:19,766 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.loghandler.NonAggregatingLogHandler:
Scheduling Log Deletion for application: application_1627614037461_0001, with delay of 10800
seconds
2021-07-30 03:05:19,767 INFO org.apache.hadoop.yarn.server.nodemanager.DefaultContainerExecutor:
Deleting absolute path : /opt/bitnami/hadoop/var/lib/nm-local-
dir/usercache/hadoop/appcache/application_1627614037461_0001
2021-07-30 03:10:19,307 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService:
Cache Size Before Clean: 0, Total Deleted: 0, Public Deleted: 0, Private Deleted: 0
2021-07-30 03:20:19,306 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService:
Cache Size Before Clean: 0, Total Deleted: 0, Public Deleted: 0, Private Deleted: 0
2021-07-30 03:30:19,309 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService:
Cache Size Before Clean: 0, Total Deleted: 0, Public Deleted: 0, Private Deleted: 0
```

Gambar V-4 Hasil Log YARN 2

5.2 Proses Dan Hasil Pengujian

5.2.1 Menjalankan Example job mencari nilai phi

Pada sub bab ini menjelaskan bagaimana melakukan sebuah melakukan job test pada Hadoop. Berikut adalah perintah untuk menjalankan sebuah example job mencari nilai phi

```
$hadoop jar /opt/bitnami/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-*.jar pi 10 100
```

```
bitnami@debian:~$ hadoop jar /opt/bitnami/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.
jar pi 10 100
Number of Maps  = 10
Samples per Map = 100
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Wrote input for Map #5
Wrote input for Map #6
Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
Starting Job
2021-07-30 02:46:42,210 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManage
r at /0.0.0.0:8032
2021-07-30 02:46:42,684 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/
hadoop-yarn/staging/hadoop/.staging/job_1627472950857_0004
2021-07-30 02:46:42,879 INFO input.FileInputFormat: Total input files to process : 10
2021-07-30 02:46:42,941 INFO mapreduce.JobSubmitter: number of splits:10
2021-07-30 02:46:43,167 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1627472950857_00
04
```

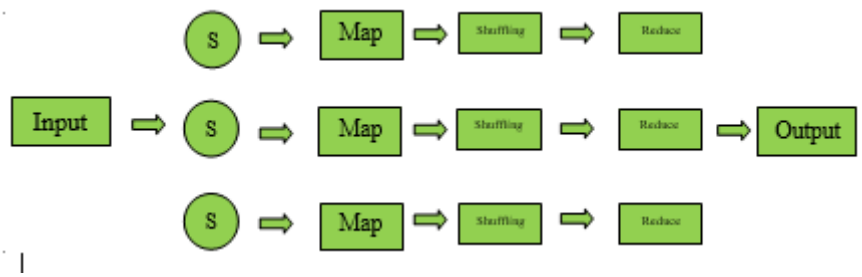
Gambar V-5 Proses Phi example program MapReduce

Langkah yang dilakukan dalam proses tersebut dapat dilihat pada Gambar V-5 berikut:
Ketika proses berjalan yang pertama kali dilakukan adalah sistem Hadoop menerima input, kemudian inputan tersebut di mapping sesuai dengan banyaknya mapping dibutuhkan. Kemudian shuffling di lakukan untuk mencocokkan data-data dengan proses map lainnya, lalu setelah selesai dilakukan reduce yaitu menyatukan Kembali data-data yang telah di proses dan mengurangi data yang redundant lalu memberikan output dari proses.

5.2.2 Menjalankan example job wordcout

Model Pada sub bab ini menjelaskan bagaimana melakukan sebuah melakukan *job test* pada Hadoop. Berikut adalah perintah untuk menjalankan sebuah *example job* menghitung kata:

```
echo "can you can a can as a canner can can a can can can can water melon apple
melon water apple you " | hadoop fs -put - /tmp/hdfs-example-inputhadoop jar
/opt/bitnami/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar grep
/tmp/hdfs-example-input /tmp/hdfs-example-output 'c[a-z]+'hadoop fs -cat /tmp/hdfs-
example-output/part-r-00000
```



Gambar V- 6 Alur Kerja Program WordCount

5.2.3 Hasil Daemon Log Monitoring pada HDFS NameNode

a. Sebelum Proses

```
2021-07-30 11:27:03,201 INFO org.apache.hadoop.hdfs.StateChange: BLOCK* allocate blk_1073741981_1157,
replicas=127.0.0.1:9866 for /tmp/hadoop-
yarn/staging/history/done_intermediate/hadoop/job_1627614037461_0002_conf.xml_tmp
2021-07-30 11:27:03,225 INFO org.apache.hadoop.hdfs.StateChange: DIR* completeFile: /tmp/hadoop-
yarn/staging/history/done_intermediate/hadoop/job_1627614037461_0002_conf.xml_tmp is closed by
DFSClient_NONMAPREDUCE_739385213_1
2021-07-30 11:27:05,475 INFO org.apache.hadoop.hdfs.server.namenode.FSEditLog: Number of transactions:
135 Total time for transactions(ms): 6 Number of transactions batched in Syncs: 47 Number of syncs: 88
SyncTimes(ms): 171
```

Gambar V-7 Initial Namenode Log

Pada Gambar V-7 menampilkan keterangan log sebelum proses dijalankan. Log tersebut masih berisi informasi log yang sebelumnya.

b. Sesudah proses

```
2021-07-30 11:53:13,218 INFO org.apache.hadoop.hdfs.StateChange: BLOCK* allocate blk_1073742001_1177,
replicas=127.0.0.1:9866 for /tmp/hadoop-
yarn/staging/history/done_intermediate/hadoop/job_1627614037461_0003-1627645937629-hadoop-
QuasiMonteCarlo-1627645993123-10-1-SUCCEEDED-default-1627645944945.jhist_tmp
2021-07-30 11:53:13,237 INFO org.apache.hadoop.hdfs.StateChange: DIR* completeFile: /tmp/hadoop-
yarn/staging/history/done_intermediate/hadoop/job_1627614037461_0003-1627645937629-hadoop-
QuasiMonteCarlo-1627645993123-10-1-SUCCEEDED-default-1627645944945.jhist_tmp is closed by
DFSClient_NONMAPREDUCE_-2137785158_1
2021-07-30 11:53:13,260 INFO org.apache.hadoop.hdfs.StateChange: BLOCK* allocate blk_1073742002_1178,
replicas=127.0.0.1:9866 for /tmp/hadoop-
yarn/staging/history/done_intermediate/hadoop/job_1627614037461_0003_conf.xml_tmp
2021-07-30 11:53:13,281 INFO org.apache.hadoop.hdfs.StateChange: DIR* completeFile: /tmp/hadoop-
yarn/staging/history/done_intermediate/hadoop/job_1627614037461_0003_conf.xml_tmp is closed by
DFSClient_NONMAPREDUCE_-2137785158_1
```

Gambar V- 8 NameNode Log Setelah Proses

Pada Gambar V-8 menunjukkan log informasi Namenode setelah melakukan proses. Dapat dilihat setelah melakukan proses informasi yang didapatkan menuliskan bagaimana daemon yang bekerja ketika melakukan proses pengolahan data bekerja, mulai dari memindahkan lokasi blok, menunjukkan informasi "SUCCEDED" ketika proses berhasil dijalankan, dan menutup job setelah job dinyatakan selesai. Berdasarkan hasil informasi diatas dapat dikatakan availability dari sistem NameNode HDFS dapat termonitor.

5.2.4 Hasil Daemon Log Monitoring pada HDFS NameNode

a. Sebelum di proses

```
2021-07-30 03:06:36,955 INFO
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Scheduling
blk_1073741958_1134 replica FinalizedReplica, blk_1073741958_1134, FINALIZED
  getNumBytes()     = 451
  getBytesOnDisk()  = 451
  getVisibleLength()= 451
  getVolume()       = /opt/bitnami/hadoop/var/lib/dfs/data
  getBlockURI()     = file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-784522513-127.0.0.2-
1624896655873/current/finalized/subdir0/subdir0/blk_1073741958 for deletion
2021-07-30 03:06:36,956 INFO
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Deleted BP-784522513-
127.0.0.2-1624896655873 blk_1073741958_1134 URI file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-
784522513-127.0.0.2-1624896655873/current/finalized/subdir0/subdir0/blk_1073741958
2021-07-30 06:08:08,185 INFO org.apache.hadoop.hdfs.server.datanode.DirectoryScanner: Scan Results:
BlockPool BP-784522513-127.0.0.2-1624896655873 Total blocks: 18, missing metadata files: 0, missing
block files: 0, missing blocks in memory: 0, mismatched blocks: 0
2021-07-30 07:06:39,451 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Successfully sent block
report 0x25d29b92c8acf424 to namenode: localhost/127.0.0.1:8020,  containing 1 storage report(s), of
which we sent 1. The reports had 18 total blocks and used 1 RPC(s). This took 0 msecs to generate and
3 msecs for RPC and NN processing. Got back one command: FinalizeCommand/5.
2021-07-30 07:06:39,451 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Got finalize command for
block pool BP-784522513-127.0.0.2-1624896655873
```

Gambar V-9 Initial Datanode Log

Pada Gambar V-9 menampilkan keterangan log sebelum proses dijalankan. Log tersebut masih berisi informasi log yang sebelumnya.

b. Sesudah di proses

```
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Deleted BP-784522513-
127.0.0.2-1624896655873 blk_1073741975_1151 URI file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-
784522513-127.0.0.2-1624896655873/current/finalized/subdir0/subdir0/blk_1073741975
2021-07-30 11:27:07,351 INFO
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Deleted BP-784522513-
127.0.0.2-1624896655873 blk_1073741976_1152 URI file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-
784522513-127.0.0.2-1624896655873/current/finalized/subdir0/subdir0/blk_1073741976
2021-07-30 11:27:07,354 INFO
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Deleted BP-784522513-
127.0.0.2-1624896655873 blk_1073741977_1153 URI file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-
784522513-127.0.0.2-1624896655873/current/finalized/subdir0/subdir0/blk_1073741977
2021-07-30 11:27:07,354 INFO
org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetAsyncDiskService: Deleted BP-784522513-
127.0.0.2-1624896655873 blk_1073741978_1154 URI file:/opt/bitnami/hadoop/var/lib/dfs/data/current/BP-
784522513-127.0.0.2-1624896655873/current/finalized/subdir0/subdir0/blk_1073741978
```

Gambar V-10 DataNode Log Setelah Proses

Pada Gambar V-10 menunjukkan log informasi kinerja datanode setelah melakukan proses. Dapat dilihat setelah melakukan proses apabila proses pengolahan data sudah selesai datanode menghapus lagi blok-blok yang sudah tidak terpakai. Dari informasi ini dapat kita simpulkan availability dari datanode ketika mengolah suatu proses apabila dilihat dari informasi yang didapatkan masih terjaga dengan baik.

5.2.5 Hasil Daemon Log Monitoring pada HDFS Resource Manager

a. Sebelum di proses

```
2021-07-30 11:53:19,813 INFO
org.apache.hadoop.yarn.server.resourcemanager.RMAppManager$ApplicationSummary:
appId=application_1627614037461_0003,name=QuasiMonteCarlo,user=hadoop,queue=default,state=FINISHED,tra
ckingUrl=http://localhost:8088/proxy/application_1627614037461_0003/,appMasterHost=localhost,submitTim
e=1627645937629,startTime=1627645937630,launchTime=1627645938683,finishTime=1627645993371,finalStatus=
SUCCEEDED,memorySeconds=315643,vcoreSeconds=148,preemptedMemorySeconds=0,preemptedVcoreSeconds=0,preem
ptedAMContainers=0,preemptedNonAMContainers=0,preemptedResources=<memory:0\,
vCores:0>,applicationType=MAPREDUCE,resourceSeconds=315643 MB-seconds\, 148 vcore-
seconds,preemptedResourceSeconds=0 MB-seconds\, 0 vcore-
seconds,applicationTags=,applicationNodeLabel=,diagnostics=,totalAllocatedContainers=12
2021-07-30 11:53:19,813 INFO org.apache.hadoop.yarn.server.resourcemanager.amlauncher.AMLauncher:
Cleaning master appattempt_1627614037461_0003_000001
```

Gambar V-11 Initial Log YARN Resource Manager

Pada Gambar V-11 memuat informasi log sebelum dilakukan pemrosesan data salah satu contohnya resource manager membersihkan temporary dari job sebelumnya yang telah dijalankan pada keterangan waktu 2021-07-30 11:53:19.

b. Sesudah di proses

```
2021-07-30 12:11:17,255 INFO
org.apache.hadoop.yarn.server.resourcemanager.rmcontainer.RMContainerImpl:
container_1627614037461_0004_01_000002 Container Transitioned from ALLOCATED to ACQUIRED
2021-07-30 12:11:18,077 INFO
org.apache.hadoop.yarn.server.resourcemanager.rmcontainer.RMContainerImpl:
container_1627614037461_0004_01_000002 Container Transitioned from ACQUIRED to RUNNING
2021-07-30 12:11:18,288 INFO
org.apache.hadoop.yarn.server.resourcemanager.scheduler.AppSchedulingInfo: checking for deactivate of
application :application_1627614037461_0004
2021-07-30 12:11:21,435 INFO
org.apache.hadoop.yarn.server.resourcemanager.rmcontainer.RMContainerImpl:
container_1627614037461_0004_01_000002 Container Transitioned from RUNNING to COMPLETED
```

Gambar V- 12 Resource Manager Log Setelah melakukan Proses

Pada Gambar V-12 bisa dilihat bagaimana resource manager mengatur container yang bekerja untuk melakukan proses mulai dari ALLOCATED ke ACQUIRED kemudian RUNNING dan ketika proses telah selesai informasinyaa berubah menjadi COMPLETED. Dari sini dapat disimpulkan juga bahwa pengimplementasian log monitoring pada daemon Hadoop core service berhasil dilakukan dan juga menemukan bahwa availability dari resource manager ketika proses sedang dijalakan dapat dimonitor dengan sistem log monitoring yang telah dipasang.

5.2.6 Hasil Daemon Log Monitoring pada HDFS NameNode

a. Sebelum di proses

```
2021-07-30 12:20:19,480 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService:
Cache Size Before Clean: 0, Total Deleted: 0, Public Deleted: 0, Private Deleted: 0
2021-07-30 12:30:19,481 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService:
Cache Size Before Clean: 0, Total Deleted: 0, Public Deleted: 0, Private Deleted: 0
```

Gambar V-13 Initial Log YARN Node Manager

Gambar V-13 menunjukkan log node manager sebelum proses dijalankan. Pada log tersebut memberikan informasi mengenai container manager yang hanya brada pada kondisi idle.

b. Sesudah di proses

```
2021-07-30 12:33:51,448 INFO org.apache.hadoop.yarn.server.nodemanager.NodeStatusUpdaterImpl: Removed
completed containers from NM context: [container_1627614037461_0007_01_000001,
container_1627614037461_0007_01_000003]
2021-07-30 12:33:51,448 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.application.ApplicationImpl: Application
application_1627614037461_0007 transitioned from RUNNING to APPLICATION_RESOURCES_CLEANINGUP
2021-07-30 12:33:51,448 INFO org.apache.hadoop.yarn.server.nodemanager.containermanager.AuxServices:
Got event APPLICATION_STOP for appId application_1627614037461_0007
2021-07-30 12:33:51,448 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.application.ApplicationImpl: Application
application_1627614037461_0007 transitioned from APPLICATION_RESOURCES_CLEANINGUP to FINISHED
2021-07-30 12:33:51,448 INFO
org.apache.hadoop.yarn.server.nodemanager.containermanager.loghandler.NonAggregatingLogHandler:
Scheduling Log Deletion for application: application_1627614037461_0007, with delay of 10800 seconds
2021-07-30 12:33:51,448 INFO org.apache.hadoop.yarn.server.nodemanager.DefaultContainerExecutor:
Deleting absolute path : /opt/bitnami/hadoop/var/lib/nm-local-
dir/usercache/hadoop/appcache/application_1627614037461_0007
```

Gambar V-14 Log YARN Resource Manager Setelah Proses

Pada Gambar V-14 memberikan inforrmasi mengenai kondisi container ketika sebuah proses berjalan. Jika dilihat log tersebut menjelaskan detail bagaimana YARN nodemanager melakukan manajemen container dalam memproses sebuah aplikasi. Dari Gambar V-14 dapat dilihat bahwa availability dari node manager ketika menjalankan proses juga dapat dilihat melalui daemon log. Untuk itu dapat dikatakan implementasi log monitoring yang dilakukan bisa digunakan untuk melihat Hadoop element availability berdasarkan implementasi yang telah dilakukan.

5.3 Hasil Penelitian

Hasil penelitian yang didapatkan menunjukkan bahwa implementasi monitoring log Hadoop core services untuk melihat availability dari Hadoop element dengan berdasarkan daemon log dapat dilakukan. Untuk keterangan dari simpulan hasil pengujian yang dilakukan dapat dilihat pada Tabel V-4 berikut:

Tabel V- 4 Hasil Penelitian

Hadoop Core Service	***Hadoop Element***	**Pengujian**	**Simpulan Percobaan**
HDFS	*Namenode*	Monitoring log saat menjalankan proses *MapReduce Job example* mencari *phi*	Berhasil
	DataNode	Monitoring log saat menjalankan proses *MapReduce Job example* mencari *phi*	Berhasil

YARN	*YARN Resource Manager*	Monitoring log saat menjalankan proses *Job example WordCount*	Berhasil
	YARN Node Manager	Monitoring log saat menjalankan proses *Job example WordCount*	Berhasil

6. Kesimpulan Dan Saran

6.1 Kesimpulan

1.Dengan implementasi monitoring log menggunakan log4j dapat memantau bagaimana Hadoop element bekerja ketika sistem Hadoop sedang melakukan proses.
2.Availability dari Hadoop Core system dapat dipantau dari daemon log saat menjalankan sebuah proses.

6.2 Saran

Berdasarkan hasil implementasi dan penelitian yang sudah dilakukan, berikut beberapa saran yang dapat berguna untuk kedepannya:
1. Melakukan penelitian lebih lanjut menggunakan alat monitoring log yang berbeda untuk menganalisa log dari Hadoop ecosystem.
2. Menganalisis log sistem terkait Hadoop dan mencoba membuat sistem monitoring pada ecosystem Hadoop di Cloud.

Referensi:

[1] Dharminder Yadav, D. H. (2019). Big Data Hadoop: Security and Privacy. INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING AND SOFTWARE ENGINEERING.
[2] Gurjit Singh Bhathal, A. S. (2019). Big Data: Hadoop Framework Vurnerability, Security Issues and Attacks. 1.
[3] Karim Abouelmehdi, A. B.-H. (2016). Big Data Emerging Issues: Hadoop Security.
[4] Mohd Rehan Ghazi, D. G. (2015). Hadoop, MapReduce and HDFS: A Developers Perspective. International Conference on Intelligent Computing, Communication & Convergence.
[5] Refik Samet, A. A. (2019). Big Data Security Problem Based on Hadoop. 4rd International Conference on Computer Science and Engineering, 2.
[6] Goel, J. N., & Mehtre, B. M. (2015). Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology. Procedia Computer Science, 57, 710–715. https://doi.org/10.1016/j.procs.2015.07.458
[7] Jain, J., & Ram Pal, P. (2017). A Recent Study over Cyber Security and its Elements. International Journal of Advanced Research in Computer Science, 8(3), 791–793. https://search.proquest.com/docview/2082950830
[8] Kumar, S. (2014). Eco-Friendly Hadoop. 16(4), 52–56.
[9] Mehta, T., & Mangla, N. (2016). A Survey Paper on Big Data Analytics using Map Reduce and Hive on Hadoop Framework. National Conference on Recent Innovations in Science, Technology & Management (NCRISTM), February, 112–118.
[10] Mishra, B. (2020). Big Data Analysis Using Hadoop Map Reduce. International Research Journal of Computer Science, 07(05), 114–122. https://doi.org/10.26562/irjcs.2020.v0705.005
[11] Shinde, P. S., & Ardhapurkar, S. B. (2016). Cyber security analysis using vulnerability assessment and penetration testing. IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare. https://doi.org/10.1109/STARTUP.2016.7583912
[12] Tandel, M. K. B. ; M. A. G. M. H. (2019). A Review Paper on Big Data and Hadoop for Data Science. International Journal of Trend in Scientific Research and Development, 4(1), 1216–1221. https://www.ijtsrd.com/papers/ijtsrd29816.pdf%0Ahttps://www.ijtsrd.com/computer-science/data-miining/29816/a-review-paper-on-big-data-and-hadoop-for-data-science/mr-ketan-bagade
[13] Umar, M. A., & Zhanfang, C. (2019). A Study of Automated Software Testing : Automation Tools and Frameworks. International Journal of Computer Science Engineering (IJCSE), 8(06), 217–225.