Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.201961520
Vol. 6, No. 5, Oktober 2019, hlm. 519-526 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

PENGUKURAN PERFORMA APACHE SPARK DENGAN LIBRARY H2O MENGGUNAKAN BENCHMARK HIBENCH BERBASIS CLOUD COMPUTING

Aminudin[1], Eko Budi Cahyono[2*]

[1,2]Jurusan Teknik Informatika, Universitas Muhammadiyah Malang, Malang, Indonesia
Email: [1]aminudin2008@umm.ac.id, [2*]ekobudi@umm.ac.id

(Naskah masuk: 01 Januari 2019, diterima untuk diterbitkan: 06 Februari 2019)

Abstrak

Apache Spark merupakan platform yang dapat digunakan untuk memproses data dengan ukuran data yang relatif besar (*big data*) dengan kemampuan untuk membagi data tersebut ke masing-masing cluster yang telah ditentukan konsep ini disebut dengan parallel komputing. Apache Spark mempunyai kelebihan dibandingkan dengan framework lain yang serupa misalnya Apache Hadoop dll, di mana Apache Spark mampu memproses data secara streaming artinya data yang masuk ke dalam lingkungan Apache Spark dapat langsung diproses tanpa menunggu data lain terkumpul. Agar di dalam Apache Spark mampu melakukan proses machine learning, maka di dalam paper ini akan dilakukan eksperimen yaitu dengan mengintegrasikan Apache Spark yang bertindak sebagai lingkungan pemrosesan data yang besar dan konsep parallel komputing akan dikombinasikan dengan library H2O yang khusus untuk menangani pemrosesan data menggunakan algoritme machine learning. Berdasarkan hasil pengujian Apache Spark di dalam lingkungan cloud computing, Apache Spark mampu memproses data cuaca yang didapatkan dari arsip data cuaca terbesar yaitu yaitu data NCDC dengan ukuran data sampai dengan 6GB. Data tersebut diproses menggunakan salah satu model machine learning yaitu deep learning dengan membagi beberapa node yang telah terbentuk di lingkungan cloud computing dengan memanfaatkan library H2O. Keberhasilan tersebut dapat dilihat dari parameter pengujian yang telah diujikan meliputi nilai running time, throughput, Avarege Memory dan Average CPU yang didapatkan dari Benchmark Hibench. Semua nilai tersebut dipengaruhi oleh banyaknya data dan jumlah node.

Kata kunci: *Apache Spark, HPC, Parallel Komputing, H2O, Benchmark Hibench*

MEASURING PERFORMANCE APACHE SPARK WITH LIBRARY H2O USING BENCHMARK HIBENCH BASED CLOUD COMPUTING

Abstract

Apache Spark is a platform that can be used to process data with relatively large data sizes (big data) with the ability to divide the data into each cluster that has been determined. This concept is called parallel computing. Apache Spark has advantages compared to other similar frameworks such as Apache Hadoop, etc., where Apache Spark is able to process data in streaming, meaning that the data entered into the Apache Spark environment can be directly processed without waiting for other data to be collected. In order for Apache Spark to be able to do machine learning processes, in this paper an experiment will be conducted that integrates Apache Spark which acts as a large data processing environment and the concept of parallel computing will be combined with H2O libraries specifically for handling data processing using machine learning algorithms . Based on the results of testing Apache Spark in a cloud computing environment, Apache Spark is able to process weather data obtained from the largest weather data archive, namely NCDC data with data sizes up to 6GB. The data is processed using one of the machine learning models namely deep learning by dividing several nodes that have been formed in the cloud computing environment by utilizing the H2O library. The success can be seen from the test parameters that have been tested including the value of running time, throughput, Avarege Memory and CPU Average obtained from the Hibench Benchmark. All these values are influenced by the amount of data and number of nodes.

Keywords: *Apache Spark, HPC, Parallel Computing, H2O, Benchmark Hibench*

1. PENDAHULUAN

National Climatic Data Center (NCDC) merupakan arsip data cuaca terbesar di dunia dengan frekuensi 3-4 kali per-jam (Chouksey & Chauhan 2017). Data tersebut sudah ada sejak 1900 sampai dengan sekarang. Jumlah data yang besar ini telah terakumulasi sejak bertahun-tahun terakhir dan akan terus bertambah. Pengolahan data secara langsung

dengan ukuran data yang besar ini menggunakan metode dan alat konvensional sangat sulit dan tidak efisien. Hal ini merupakan tantangan di dalam penelitian terkait dengan keilmuan big data dan parallel komputing.

Data NCDC memiliki ukurannya yang semakin besar tentunya membutuhkan tempat penyimpanan yang sangat besar dan sistem pengelolaan yang tepat agar mudah dalam pengelolaanya. Ada beberapa *platform* atau framework yang dapat menjadi mengelola dan memproses data yang berukuran cukup besar (*big data*) antara lain 1010data, Actian, *Amazon Web Services* (AWS), Cloudera, IBM SmartCloud, Rackspace, dan lain-lain (Khusumanegara, 2014). Untuk dapat menyimpan dan mengolah data dengan ukuran yang besar (*big data*) secara baik dan cepat dibutuhkan teknologi komputer yang disebut *high performance computer* atau *super computer*, akan tetapi untuk membangun suatu sistem *super computer* membutuhkan biaya yang tidak murah. Untuk mengatasi masalah tersebut, teknik parallel computing adalah salah satu solusi tersebut. (Aminudin & Alwi 2018). *Parallel computing* adalah penggunaan beberapa komputer yang saling terhubung untuk mengolah data dalam ukuran yang besar. Salah satu *platform* yang sering digunakan ini untuk mengolah data yang berukuran besar (*big data*) secara parallel dan dapat berjalan diatas beberapa *cluster* adalah Apache Spark (Wang, dkk, 2016).

Apache Spark merupakan salah satu framework terdistribusi untuk pemrosesan data dengan skala besar yang fungsinya hampir sama dengan Map-Reduce kepunyaan Apache Hadoop terutama digunakan untuk pemrosesan secara parallel. Salah satu komponen yang terdapat Apache Spark yang tidak dimiliki oleh platform yang lain yaitu RDD (*Resilient Distributed Dataset*). RDD adalah memori terdistribusi yang dapat dibaca ketika kluster dioperasikan secara parallel (Pan 2016). Ketika operasi dijalankan pada setiap RDD, jumlah partisi di dalamnya akan menentukan tingkat paralelisme.

Apache Spark terdiri dari beberapa komponen yaitu (1) Spark SQL adalah evolusi terbaru sesudah dari Shark SQL yang lebih dulu digunakan di dalam Spark. Spark SQL dapat melakukan penyimpanan memori atau dalam Bahasa yang lain (*In-Memory Columnar Storage*) dari Shark, Spark SQL juga compatible dengan Hive salah satu Query yang terdapat pada Hadoop (Ivanov et al. 2014). Maka dari itu Spark SQL Ini memiliki perkembangan besar dalam hal kompatibilitas data dan optimalisasi kinerja (Han and Zhang 2016). (2) Spark *Streaming* adalah framework stream yang berjalan di Apache Spark yang terdapat API dan langsung terintegrasi atau terhubung dengan streaming secara interaktif. Adanya Spark Streaming ini mampu mengatasi kelemahan yang terdapat pada Apache Hadoop di dalam bidang analisa pada streaming data secara realtime. Salah satu kelemahan di dalam Apache Hadoop, di mana data yang dapat dianalisis di dalam Apache Hadoop tidak dapat langsung diproses dikarenakan Apache Hadoop cara pemrosesanya bersifat batch atau periodik (Wang, Fu, and Wang 2016) (Aminudin 2019), (3) GraphX adalah adalah sebuah parallel komputasi dengan menggunakan API chart dan graph pada pemrosesan data. GraphX memiliki perbaikan besar pada kinerja dan penurunan overhead memori. GraphX memperluas RDD Apache Spark dengan menampilkan secara grafik secara terdistribusi. Beberapa grafik dapat ditampilkan ke masing-masing atribut baik vertex maupun edge (Han & Zhang 2016).

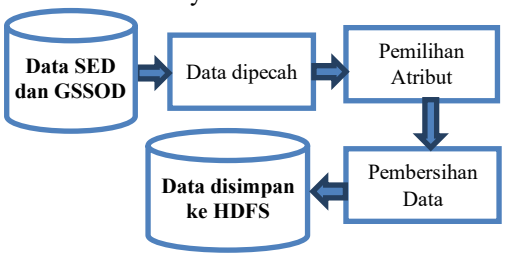
Untuk melakukan pemrosesan data baik itu prediksi, klasifikasi dan klasterisasi Apache Spark membutuhkan tambahan library untuk melakukan pemrosesan *machine learning* library yang dibutuhkan yaitu H2O (Ng, 2016). H2O merupakan library *distributed machine learning* yang bersifat *open source* biasanya digunakan untuk sebagai pemrosesan *machine learning*. H2O mendukung pemrosesan machine learning dan mendukung beberapa algoritma *machine learning* seperti *deep learning*, *Naive Bayes Classifier*, *k-means clustering* dsb. Untuk meningkatkan kecepatan pemrosesan *machine learning* pada H2O dibutuhkan Apache Spark agar pemrosesan machine learning dapat dilakukan dengan pemrosesan secara terdistribusi atau secara parallel (Chen, 2017); (Gupta, 2017). Maka di dalam paper ini akan dilakukan pemrosesan data prediksi memanfaatkan library H2O dengan diproses secara terdistribusi menggunakan Apache Spark dengan membagi beberapa node-node yang saling terhubung untuk memproses data secara terdistribusi(Mavridis & Karatza 2017). Untuk mengukur dan menganalisa performa Spark di dalam memproses data secara terdistribusi digunakan Banchmark Hibench untuk mengetahui running time, kecepatan pemrosesan, hasil troughput, rata-rata penggunaan memori dan CPU didalam memproses data secara terdistribusi (Liu, 2015) (Samadi, dkk, 2016). Penelitian ini menggunakan data dari NCDC (*National Centers for Environmental Information*) dengan jenis data SED (*Storm Event Database*) dan GSOD (*Global Surface Summary of Database*) yang didapatkan dari database NOA yang secara free dapat diunduh diwebsite tersebut.

Kontribusi yang diharapkan dari paper ini adalah pertama membahas secara komprehensif atau *review* tentang cara kerja Apache Spark dan komponen yang terlibat di dalamnya, kedua melihat hasil evaluasi Apache Spark dengan Benchmark Hibench untuk memproses data dengan skala yang besar untuk mengukur performa kemampuan Apache Spark di dalam lingkungan terdistribusi menggunakan Cloud Computing.

2. METODE PENELITIAN

2.1 Dataset Penelitian

Dataset yang digunakan di dalam paper ini menggunakan dua buah jenis dataset yang telah disediakan oleh database *National Climate Data Center* (NCDC) yaitu *Storm Event Database* (SED) dan *Global Surface Summary of the Day* (GSSOD). kedua jenis dataset tersebut memiliki ukuran dan cara pengambilan data berdasarkan dari sensor yang berbeda. Dataset SED hanya mencatat fenomena cuaca dengan jenis tertentu misalnya badai, hujan yang intensitas yang tinggi dll, sehingga data di dalam SED tidak begitu besar. Begitupula dataset GSSOD berisi data dengan parameter maksimum, temperatur dan kecepatan angin dengan input ke sensor 3-4 jam, sehingga dilihat dari besarnya data antara kedua dataset tersebut maka dataset GSSOD jauh lebih besar dibandingkan dengan SED, dikarenakan data SED hanya jika terjadi badai data tersebut akan masuk ke dalam database tetapi sebaliknya dataset GSSOD akan secara periodik input data ke dalam database berdasarkan parameter iklim dan cuaca. Implementasinya data dari SED akan dipecah menjadi 250 MB, 500MB, 1GB dan data dari GSSOD dipecah menjadi 2GB, 4 GB dan 6GB. Hal itu dilakukan agar pada pengujian dapat diketahui pola pemrosesan Apache Spark ketika diberikan data yang bervariasi ukurannya.

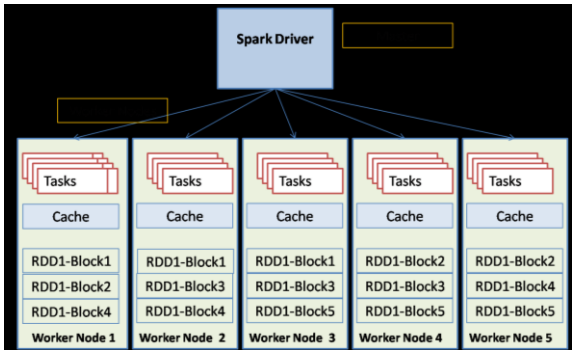


Gambar 1. Pemrosesan Dataset

2.2 Rancangan Berjalanya Apache Spark

Apache Spark bekerja dengan konsep parallel komputing dengan memanfaatkan salah satu komponenya yaitu *Resilient Distributed Datasets* (RDD) dengan maksud data akan didistribusikan dengan sekumpulan machine yang tersebar. Data yang telah tersebar ke masing-masing machine akan dapat dibangun kembali kalau didalam perjalanan pengiriman data terjadi kegagalan. Rancangan Spark ditunjukan pada Gambar 2.

Apache Spark didalam memproses data membagi sekumpulan data ke masing-masing machine yang tersebar dan data tersebut akan bagikan secara terdistribusi ke machine tersebut. Gambar 2 menunjukan pembagian pemrosesan data berdasarkan dari masing-masing slave yang telah dibagi oleh sisi master dengan membagi sumber daya yang telah disediakan di sisi master.



Gambar 2. Rancangan Pemecahan Data di Apache Spark

Di sisi slave data akan diproses dengan cara membagikan ke masing-masing block untuk distribusi data.

2.3 Kebutuhan Sumberdaya Cloud Computing

Cloud Computing merupakan salah satu sumberdaya komputer yang memanfaatkan agresivitas koneksi internet untuk pengolahan daya komputasi misalnya CPU, RAM, sistem operasi, storage dan software dilakukan melalui jaringan internet. Salah satu jenis cloud computing yang paling populer adalah *Infrastructure as a Service* (IaaS). IaaS merupakan layanan virtual yang menyediakan sumber daya komputer yang meliputi kebutuhan server, storage, network dll. Sehingga pengguna layanan tersebut bebas untuk mengatur dan menjalankan fasilitas tersebut.

Di dalam paper kali ini menggunakan layanan *cloud computing* yang disediakan oleh BiznetGioCloud. Di mana di dalam penelitian ini menggunakan lima buah node yang dibagi ke dalam satu node master dan empat node lainya sebagai slave. Secara konfigurasi master node memiliki spesifikasi kebutuhan yang lebih tinggi dibandingkan dengan empat node yang bertindak sebagai slave, hal ini dikarenakan di dalam master yang bertindak sebagai pengatur atas berjalanya slave node dan di dalam master node juga dikonfigurasi beberapa perangkat lunak untuk untuk mengendalikan slave node. Spesifikasi kebutuhan komputasi di dalam membangun lingkungan parallel komputing ditunjukan pada Tabel 1.

Tabel 1. Spesifikasi kebutuhan node pada cloud computing

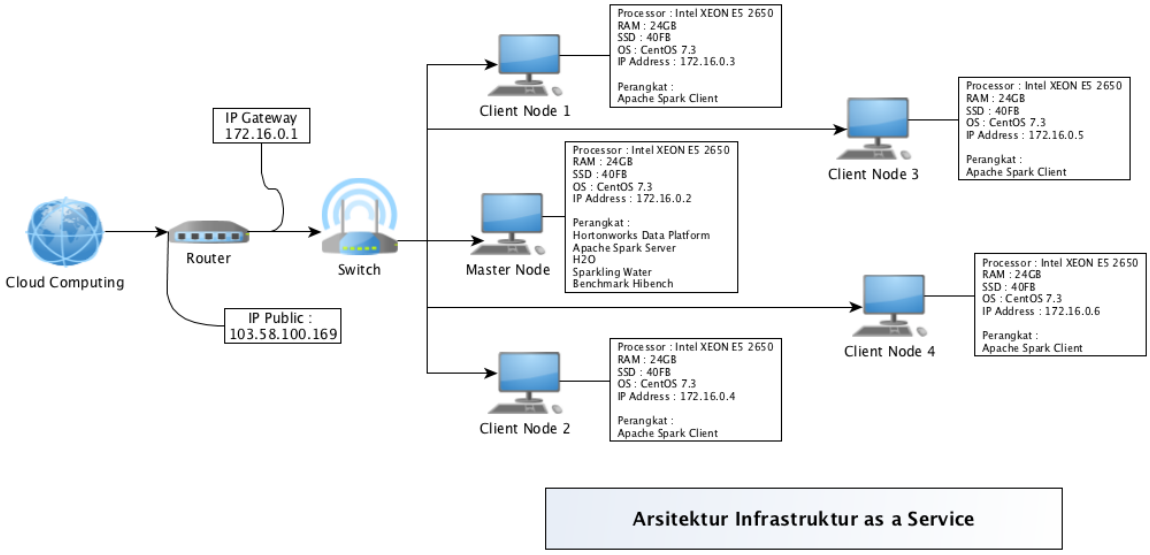
Nama	Spesifikasi Kebutuhan		
	Processor	RAM	SSD
Master Node	Intel Xeon @ 2.299 GHz	24 GB	80 GB
Slave Node 1	Intel Xeon @ 2.299 GHz	24 GB	40 GB
Slave Node 2	Intel Xeon @ 2.299 GHz	24 GB	40 GB
Slave Node 3	Intel Xeon @ 2.299 GHz	24 GB	40 GB
Slave Node 4	Intel Xeon @ 2.299 GHz	24 GB	40 GB

Pada komputer cluster dipasang dengan sistem operasi CentOS untuk memudahkan pemasangan perangkat lunak yang dibutuhkan oleh Apache Spark. Untuk menghubungkan masing-masing node baik master dan slave akan dipasang SSH *public key*, hal

ini dimaksudkan untuk membangun lingkungan komputasi terdistribusi.

2.4 Topologi Arsitektur Apache Spark Berbasis Cloud Computing

Setelah mengetahui kebutuhan di lingkungan terdistribusi, maka langkah selanjutnya adalah membangun topologi untuk mengetahui hubungan secara detail masing-masing node. Secara kebutuhan lingkungan terdistribusi dibagi menjadi dua node yang slave dan master. Dikarenakan perbedaan tugas yang mendasar maka resource, perangkat lunak di sisi master lebih banyak dibandingkan dengan di sisi slave. Di dalam master slave dipasang beberapa software yang mendukung untuk mengatur berjalanya beberapa slave node untuk dilakukan pengujian.



Gambar 3. Topologi Cloud Computing untuk Menjalankan Apache Spark

Gambar 3 ditunjukan secara detail hubungan yang terjadi pada lingkungan terdistribusi yang dibangun dibawah layanan cloud computing. Menunjukan pembagian cluster komputing dibagi menjadi dua yaitu master node dan slave node. Pada master node dipasang beberapa perangkat lunak untuk mendukung lingkungan komputasi terdistribusi diantaranya adalah :

(1) Hortonwork Data Platform (HDP) sebagai media atau pemasangan perangkat lunak yang dilingkungan komputasi terdistribusi.
(2) Apache Spark Server perangkat lunak utama yang berfungsi untuk proses komputasi parallel.
(3) Library H2O merupakan library yang menyediakan pemrosesan algoritme machine learning seperti Deep Learning, Random Forest, K-Means dll yang terdapat pada lingkungan komputasi terdistribusi yang dapat dimanfaatkan di dalam pemrosesan data dengan skala yang besar.
(4) Sparkling Water yang berfungsi untuk mengintegrasikan atau menghubungkan antara Apache Spark dan library H2O.
(5) Bencmark Hibench merupakan salah tools yang dapat merekam data performa Apache Spark yang didasarkan atas beberapa parameter yang telah ditentukan.

2.5 Skenario Pengujian

Pengujian yang akan dilakukan di dalam penelitian ini meliputi beberapa tahapan diantaranya (1) Menguji beberapa komputer cluster yang berjalan di lingkungan terdistribusi dengan membagi beberapa node untuk melakukan pemrosesan data menggunakan deep learning. Seperti yang sudah dijelaskan sebelumnya bahwa data yang diujikan merupakan data yang didapatkan dari data NDCD dengan jenis SED dan GSSOD untuk menguji performa Apache maka data tersebut dipecah dengan beberapa variasi ukuran (2) Pengujian performa Apache Spark dengan menggunakan beberapa parameter yang sudah ditentukan, di mana semua parameter tersebut didapatkan dari berkas tools Benchmark Hibench dengan memproses data menggunakan algoritme *machine learning* dengan memanfaatkan library H2O.

3. IMPLEMENTASI DAN HASIL ANALISA

3.1 Implementasi Sistem

Implementasi dilakukan untuk mewujudkan rancangan yang telah dibuat. Rancangan tersebut selanjutnya diimplementasikan dalam sebuah layanan *Infrastructure as a Service*, yaitu BiznetGioCloud. Untuk implementasi perangkat keras dan perangkat lunak, BiznetGioCloud telah menyediakan hal tersebut sehingga penulis hanya cukup menggunakan saja tanpa perlu melakukan konfigurasi yang rumit.

3.2 Pengujian Sistem

Pengujian sistem akan dilakukan dengan menggunakan dua skenario. Skenario tersebut yaitu penggunaan jumlah node dan ukuran dataset yang berbeda oleh Apache Spark saat melakukan komputasi terdistribusi dengan menggunakan algoritma Deep Learning. Pengujian sistem dengan kedua skenario tersebut dilakukan menggunakan alur pengujian yang telah ditentukan agar menghasilkan data yang akurat dan valid untuk keperluan pengambilan kesimpulan. Selain itu, ada 6 parameter pengujian yang digunakan sehingga dapat memberikan gambaran detail mengenai pengaruh jumlah *node* yang dipakai dan ukuran dataset yang diproses terhadap kinerja komputasi Apache Spark dengan menggunakan algoritma Deep Learning.

Skenario pertama akan dilakukan dengan cara mengukur performasi komputasi Framework Apache Spark di dalam memproses dataset yang ukuranya bervariasi dengan memanfaatkan libary H2O yang di dalamnya menggunakan algoritme Deep Learning dengan menggunakan jumlah node tertentu di dalam komputer cluster. Sedangkan skenario kedua dilakukan dengan cara mengukur kinerja komputasi Apache Spark dalam memproses ukuran dataset yang berbeda dengan rincian : pengujian dengan ukuran 250 MB, 500 MB dan 1 GB akan menggunakan Storm Event Database, sedangkan pengujian dengan ukuran 2 GB, 4 GB dan 6 GB akan menggunakan Global Surface Summary of The Day.

3.3 Menjalankan Beban Kerja

Pengujian sistem dilakukan dengan menjalankan beban kerja tertentu. Definisi beban kerja yaitu sebuah persoalan komputasi yang harus dilakukan dan diselesaikan oleh sistem. Dalam pengujian sistem, beberapa beban kerja harus dijalankan karena penelitian menggunakan banyak variabel antara lain *framework big data* yaitu Apache Spark, perangkat lunak tambahan untuk menyediakan algoritma Deep Learning (H2O dan Sparkling Water), jumlah node yang digunakan (dua node, tiga node dan empat node), dataset yang digunakan adalah SED dan GSSOD yang didapatkan dari data NCDC, adapun parameter yang dijadikan pengujian yaitu *running time*, *throughput*, *average CPU usage*, *max CPU usage*, *average memory usage* dan *max memory usage*),serta ukuran dataset yang diproses (250 MB, 500 MB, 1 GB, 2 GB, 4 GB dan 6 GB).

3.4 Mengumpulkan Hasil Pengujian

Pengujian sistem dilakukan dengan menggunakan tools Benchmark Hibench. Benchmark Hibench adalah program yang dapat merekam aktivitas komputer atau sebuah komputer kluster saat sebuah proses sistem atau perangkat lunak dijalankan, mulai dari awal dieksekusi hingga waktunya berhenti. Dalam pengujian ini, proses sistem atau perangkat lunak yang dijalankan tersebut adalah sebuah beban kerja. Banyaknya beban kerja yang perlu dijalankan dalam pengujian sistem, disesuaikan dengan banyaknya variabel penelitian yang telah ditentukan sebelumnya. Benchmark Hibench mulai bekerja saat sebuah beban kerja dijalankan. Setelah beban kerja tersebut berhenti, maka Benchmark Hibench secara otomatis akan mengumpulkan hasil rekaman aktivitas sistem mulai dari awal waktu hingga akhir waktu dan menyimpannya dalam bentuk teks. Teks tersebut kemudian diurai untuk mendapatkan nilai - nilai parameter pengujian yang selanjutnya dikumpulkan untuk keperluan analisis.

3.5 Parameter dan Hasil Pengujian

Pengujian sistem menggunakan 6 parameter pengujian yang telah ditentukan. Nilai parameter pengujian tersebut didapatkan melalui program tools Benchmark Hibench. Benchmark Hibench merekam seluruh aktivitas komputer atau sebuah komputer kluster saat sebuah proses sistem atau perangkat lunak dijalankan mulai dari awal waktu saat dieksekusi hingga akhir waktu saat berhenti. Dalam pengujian ini tentu sudah jelas, bahwa akan menjalankan Apache Spark untuk memproses dataset dalam komputer kluster dengan menggunakan algoritma Deep Learning.

a. Running Time

Running Time adalah waktu yang diperlukan Apache Spark dalam memproses data. Nilai parameter ini didapatkan dengan cara mencari selisih antara waktu awal dan waktu akhir saat Apache Hadoop dan Apache Spark dijalankan atau dihentikan untuk memproses dataset dalam komputer kluster dengan menggunakan algoritma Deep Learning. Satuan pengukuran untuk parameter *running time* adalah detik / *seconds*.

Tabel 2. Hasil Pengujian running time

Dataset	Running Time (s)		
	2 Node	3 Node	4 Node
SED 250 MB	93	86	85
SED 500 MB	122	106	102
SED 1 GB	184	149	134
GSOD 2 GB	225	210	188
GSOD 4 GB	334	272	266
GSOD 6 GB	431	362	288

Berdasarkan dari Table 2 didapatkan hasil bahwa Apache Spark pemrosesan di dalamnya bersifat linier artinya nilai running time tergantung dari besarnya data dan banyaknya node, semakin banyak node yang digunakan maka waktu pemrosesan running time semakin kecil, begitu pula semakin besar data maka running time yang dibutuhkan juga semakin besar.

b. Throughput

Throughput adalah kecepatan dalam bertukar data dengan ukuran tertentu. Kegiatan bertukar data tersebut terjadi pada node yang dipakai dalam komputer *cluster*, saat Apache Spark memproses dataset. Oleh karena itu, semakin tinggi nilai *throughput* yang dicapai, maka semakin sedikit waktu

yang dibutuhkan Apache Spark untuk menyelesaikan komputasi. Adapun hasil yang diperoleh dari pengujian Apache Spark untuk parameter throughput ditunjukan pada Tabel 3.

Tabel 3. Hasil Pengujian Troughput

Dataset	Throughput (b/s)		
	2 Node	3 Node	4 Node
SED 250 MB	2818751,57	3048184,837	3084045,835
SED 500 MB	4297499,402	4946178,557	5140146,343
SED 1 GB	5836095,19	7206990,034	8013742,649
GSOD 2 GB	9544371,622	10226112,45	11422785,19
GSOD 4 GB	12859183,03	15790320,34	16146492,98
GSOD 6 GB	14947683,93	17796828,1	22369624,21

Hasil dari Pengujian throughput tidak jauh berbeda dengan hasil pada running time, di mana hasilnya dipengaruhi oleh besarnya data dan banyaknya node. Berdasarkan Tabel 3 didapatkan data semakin besar data diproses dan banyaknya node yang digunakan untuk memproses data maka jumlah throughput juga meningkat, artinya bahwa Apache Spark mampu memproses data dengan ukuran yang besar dan mampu berjalan juga di lingkungan komputasi terdistribusi.

c. Average Memory Usage

Average memory usage adalah persentase rata - rata penggunaan *memory* pada *node - node* yang digunakan pada saat Apache Spark memproses dataset. Nilai parameter ini didapatkan dari berkas hasil rekaman aktivitas sistem yang dihasilkan oleh Benchmark Hibench. Satuan pengukuran yang dipakai adalah persentase (%).

Tabel 4. Hasil Pengujian Average Memory Usage

Dataset	Average Memory Usage (%)		
	2 Node	3 Node	4 Node
SED 250 MB	44,262	27,409	18,625
SED 500 MB	48,091	27,993	20,586
SED 1 GB	45,614	30,720	25,291
GSOD 2 GB	60,984	37,812	27,799
GSOD 4 GB	75,565	44,914	39,855
GSOD 6 GB	71,518	55,461	44,819

Berdasarkan Tabel 4 ditunjukan bahwa kebutuhan memory Apache Spark saat memproses data sangat dipengaruhi oleh intensitas besarnya data yang diproses. Pergerakan kapasitas penggunaan memory naik seiring dengan peningkatan data. Tetapi nilai *average memory usage* pada Apache Spark meningkat saat hanya menggunakan 2 *node*. Tetapi, nilai tersebut turun secara signifikan saat menambah *node*. Hal tersebut membuktikan bahwa dengan menambah *node* akan mengurangi beban kerja yang dilakukan sehingga berujung pada penggunaan memori yang lebih sedikit. Pada data 500 MB dan 1 GB dengan menggunakan 2 node ada perbedaan di mana nilai data dengan besar 1 GB waktu yang dibutuhkan lebih sedikit dibandingkan dengan 500 MB. Hal ini dikarenakan ketika dilakukan pemrosesan data dengan Deep Learning untuk mengirimkan data dari Library H2O ke Apache Spark melewati tools integrasi yaitu Sparkling water. Hal inilah yang menjadikan nilai average memory usage tidak ditentukan oleh banyaknya data dan node.

d. Max Memory Usage

Max Memory Usage adalah persentase maksimum penggunaan memory pada node - node yang digunakan saat Apache Spark ketika memproses dataset. Nilai parameter ini didapatkan dari berkas hasil rekaman aktivitas sistem yang dihasilkan oleh Benchmark Hibench. Satuan pengukuran yang dipakai adalah persentase (%).

Tabel 5. Hasil Pengujian Max Memory Usage

Dataset	Max Memory Usage (%)		
	2 Node	3 Node	4 Node
SED 250 MB	49,920	41,100	23,247
SED 500 MB	56,210	68,470	26,277
SED 1 GB	62,680	78,700	34,140
GSOD 2 GB	67,220	85,420	31,727
GSOD 4 GB	98,920	64,535	82,060
GSOD 6 GB	98,820	74,240	52,625

Berdasarkan Tabel 5 ditunjukan bahwa nilai *max memory usage* dalam pengujian Apache Spark sangat tidak stabil. Kadang nilai tersebut naik pada beberapa pengujian, tetapi menurun pada pengujian berikutnya. Hal ini dikarenakan proses transfer data antar node yang terjadi di dalam pada komponen Apache Spark yaitu Resilient Distributed Datasets (RDD) sehingga mengakibatkan penggunaan Max Memory Usage tidak beraturan. Ketidakberaturan itu juga terjadi pada salah satu tools integrasi machine learning yang digunakan untuk menghubungkan antara lingkungan terdistribusi dengan pemrosesan machine learning yaitu tools Sparkling Water. Sehingga akibat dari kedua tools tersebut sehingga kebutuhan akan maksimal memory yang dibutuhkan tidak bergantung pada jumlah data dan jumlah node.

e. Average CPU Usage

Average CPU usage adalah persentase rata - rata penggunaan CPU pada *node - node* yang digunakan saat Apache Spark memproses dataset. Nilai parameter ini didapatkan dari berkas hasil rekaman aktivitas sistem yang dihasilkan oleh Benchmark Hibench. Satuan pengukuran yang dipakai adalah persentase (%). Berikut Tabel 6 mengunjukan pengujian sistem dengan parameter *Average CPU Usage*.

Tabel 6. Hasil Pengujian Avarege CPU Usage

Dataset	Average CPU Usage (%)		
	2 Node	3 Node	4 Node
SED 250 MB	49,117	37,761	27,713
SED 500 MB	45,904	37,152	27,309
SED 1 GB	47,000	38,069	27,817
GSOD 2 GB	64,865	55,295	50,814
GSOD 4 GB	66,845	59,711	54,068
GSOD 6 GB	70,504	59,029	52,429

Berdasarkan Tabel 6 ditunjukan bahwa rata-rata penggunaan CPU yang digunakan untuk memproses data menggunakan algoritme deep learning bersifat linear artinya masih ditentukan oleh jumlah data dan jumlah node. Di dalam Tabel tersebut juga ditunjukan bahwa nilai *average CPU usage* oleh Apache Spark meningkat jika hanya menggunakan dua buah *node* dan menurun jika menggunakan lebih banyak *node*. Selain itu terlihat bahwa ukuran dataset yang berbeda secara tidak langsung mempengaruhi nilai *average CPU usage*. Berikut Gambar 8 menunjukan hasil konsumsi penggunaan di dalam titik maksimum memory yang dibutuhkan oleh Apache Spark ketika memproses dataset yang bervariasi.

f. Max CPU Usage

Max CPU usage adalah presentase maksimum penggunaan *CPU* saat Apache Spark memproses dataset mulai dari awal dieksekusi hingga sampai berakhir. Nilai *Max CPU Usage* juga didapatkan melalui berkas rekaman aktivitas sistem yang dihasilkan oleh Benchmark Hibench dengan mengintegrasikan antara beberapa tools yang sudah disetting di dalam lingkungan terdistribusi diantaranya Spark, H2O dan Sparkling Water. Berikut Tabel 7 menunjukan pengujian hasil Apache Spark dengan menggunakan parameter *Max CPU Usage*.

Tabel 7. Hasil Pengujian Paraneter Max CPU Usage

Dataset	Max Memory Usage (%)		
	2 Node	3 Node	4 Node
SED 250 MB	91,902	88,156	81,952
SED 500 MB	96,667	92,701	70,069
SED 1 GB	92,725	94,170	98,150
GSOD 2 GB	97,665	99,364	99,503
GSOD 4 GB	98,224	99,435	99,590
GSOD 6 GB	99,825	99,503	99,652

Berdasarkan Tabel 7 menunjukan bahwa nilai Max CPU Usage dalam penggunaan maksimal CPU menurun pada pemrosesan dengan SED 250 MB dan SED 500 MB lalu meningkat saat menggunakan dataset lebih dari 1 GB. Apache Spark memiliki nilai *max CPU usage* yang tidak stabil untuk pengujian dengan dataset SED 250 MB dan SED 500 MB. Kemudian, nilai tersebut cukup stabil untuk pengujian dengan dataset lebih dari 1 GB walaupun nilai antara penggunaan dua, tiga dan empat node memiliki selisih yang kecil.

4. KESIMPULAN DAN SARAN

Berdasarkan dari pengujian yang telah dilakukan di beberapa sub bab sebelumnya, maka dapat diambil kesimpulan sebagai berikut :

a. Berdasarkan dari pengujian nilai running dan nilai throughput bahwa Apache Spark mampu memproses data dengan skala yang besar di dalam lingkungan komputasi parallel dengan membagi beberapa node yang berbeda.
b. Berdasarkan dari pengujian parameter yang telah diujikan dengan menggunakan parameter yang didapatkan dari tools Benchmark Hibench nilai pengujian parameter bersifat linear artinya hasil nilai parameter yang diujikan didasarkan atas jumlah node dan jumlah besarnya data.
c. Ada dua parameter yang ketika diujikan tidak bersifat linear yaitu Max Memory Usage dan Max CPU Usage hal ini dikarenakan pengiriman data antar node di dalam RDD yang tidak beraturan serta penggunaan salah satu tools Sparkling Water yang digunakan untuk integrasi di dalam lingkungan terdistribusi dan tools machine learning H2O.
d. Beberapa masukan dan saran sebagai bahan penelitian selanjutnya yaitu muncul beberapa teknologi yang dapat digunakan sebagai pengganti H2O misalnya saja TensorFlow, MXNET dll hal ini dapat digunakan sebagai alternative pengujan performa Spark dengan H2O ataupun Spark dengan library machine learning yang lain.

DAFTAR PUSTAKA

AMINUDIN, AMINUDIN. 2019. "Analisa Performa Apache Hadoop Dengan H2o Menggunakan Benchmark Hibench Via Cloud Computing." In *Prosiding SENTRA (Seminar Teknologi Dan Rekayasa)*.

AMINUDIN, AMINUDIN, AND MUHAMMAD ALWI. 2018. "Analisa Multithreading Pada Sistem Rekomendasi Menggunakan Metode Collaborative Filtering Dengan." *Techno. Com* 17 (1): 1–11.

CHEN, JIANGUO, KENLI LI, ZHUO TANG, KASHIF BILAL, SHUI YU, CHULIANG WENG, AND KEQIN LI. 2017. "A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment." *IEEE Transactions on Parallel and Distributed Systems* 28 (4): 919–33. https://doi.org/10.1109/TPDS.2016.2603511.

CHOUKSEY, PRIYANKA, AND ABHISHEK SINGH CHAUHAN. 2017. "Weather Data Analytics Using MapReduce and Spark" 6 (2): 42–47. https://doi.org/10.17148/IJARCCE.2017.6210.

GUPTA, ANAND, HARDEO THAKUR, RITVIK SHRIVASTAVA, PULKIT KUMAR, AND SREYASHI NAG. 2017. "A Big Data Analysis Framework Using Apache Spark and Deep Learning," no. 1. https://doi.org/10.1109/ICDMW.2017.9.

HAN, ZHIJIE, AND YUJIE ZHANG. 2016. "Spark: A Big Data Processing Platform Based on Memory Computing." *Proceedings - International Symposium on Parallel Architectures, Algorithms and Programming, PAAP* 2016–Janua: 172–76.

https://doi.org/10.1109/PAAP.2015.41.

IVANOV, TODOR, RAIK NIEMANN, SEAD IZBEROVIC, MARTEN ROSSELLI, KARSTEN TOLLE, AND ROBERTO V ZICARI. 2014. "Benchmarking DataStax Enterprise/Cassandra with HiBench." *ArXiv Preprint ArXiv:1411.4044*.

JONNALAGADDA, V SRINIVAS, P SRIKANTH, KRISHNAMACHARI THUMATI, SRI HARI NALLAMALA, AND KRISHNA DIST. 2016. "A Review Study of Apache Spark in Big Data Processing" 4 (3): 93–98.

KHUSUMANEGARA, PRIAGUNG. 2014. "Analisis Performa Kecepatan Mapreduce Pada Hadoop Menggunakan Tcp Packet Flow," 72.

LIU, LU. 2015. "Performance Comparison by Running Benchmarks on Hadoop, Spark, and Hamr."

MAVRIDIS, ILIAS, AND HELEN KARATZA. 2017. "Performance Evaluation of Cloud-Based Log File Analysis with Apache Hadoop and Apache Spark." *Journal of Systems and Software* 125: 133–51. https://doi.org/10.1016/j.jss.2016.11.037.

NG, S. S. Y., W. ZHU, W. W. S. TANG, L. C. H. WAN, AND A. Y. W. WAT. 2016. "An Independent Study of Two Deep Learning Platforms - H2O and SINGA." *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 1279–83. https://doi.org/10.1109/IEEM.2016.7798084.

PAN, SHENGTI. 2016. "The Performance Comparison of Hadoop and Spark."

SAMADI, YASSIR, MOSTAPHA ZBAKH, AND CLAUDE TADONKI. 2016. "Comparative Study between Hadoop and Spark Based on Hibench Benchmarks." https://doi.org/10.1109/CloudTech.2016.7847709.

WANG, KAIYUAN, JIAN FU, AND KAIYUAN WANG. 2016. "SPARK – A Big Data Processing Platform for Machine Learning." *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration*, 48–51. https://doi.org/10.1109/ICIICII.2016.27.