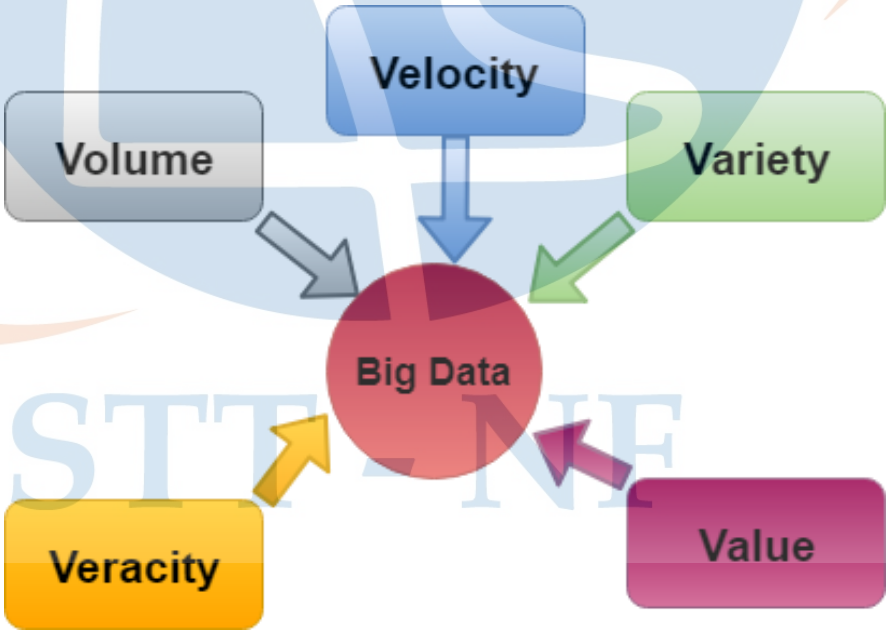
BAB II
KAJIAN LITERATUR

Pada bab ini, peneliti akan menguraikan teori-teori dan konsep yang mendukung dalam penelitian yang dilakukan.

2.1 *Big* Data

Big data adalah data yang sudah sangat sulit untuk dikoleksi, disimpan, dikelola maupun dianalisa dengan menggunakan sistem database biasa karena volumenya yang terus berlipat[6]. Dapat disimpulkan bahwa *big* data adalah data dengan ciri berukuran sangat besar, sangat variatif, sangat cepat pertumbuhannya dan mungkin tidak terstruktur yang perlu diolah khusus dengan teknologi inovatif sehingga mendapatkan informasi yang mendalam dan dapat membantu pengambilan keputusan yang lebih baik serta tidak dapat di proses menggunakan alat tradisional biasa dan harus menggunakan cara dan alat baru untuk mengolah data sehingga menjadi nilai.



Gambar 2.1. 1Karakteristik Big Data

Big data memiliki 5 karakteristik yaitu:

1. *Volume* yaitu sejumlah data besar yang dihasilkan perdetiknya. Data besar tersebut terdiri dari berbagai macam-macam data seperti data yang dihasilkan dari email, pesan twitter, foto, status, dan yang lainnya.
2. *Velocity* yaitu kecepatan data baru yang dihasilkan dan pertumbuhan serta perubahannya. Seperti pesan media sosial yang cepat menjadi viral dalam hitungan detik.
3. *Variety* yaitu mengacu pada berbagai jenis data yang sekarang bisa kita gunakan, karena dimasa sebebelumnya orang-orang hanya berfokus pada data yang sudah terstruktur. Faktanya data di dunia sekarang banyak bahkan hampir semua tidak terstruktur dan hal itu menyebabkan kita kesusahan untuk memasukkan data tersebut kedalam tabel.
4. *Veracity* yaitu kebenaran serta keakuratan dari suatu data. Karena data yang begitu cepat dan besar sehingga kualitas dan keakuratan data yang kurang terkontrol.
5. *Value* yaitu kemampuan kita mengubah data yang ada menjadi sebuah nilai.

Teknologi *Big* data diciptakan untuk menangani 5 karakteristik diatas, jadi jika data yang kita punya memiliki salah satu karakteristik diatas maka kita bisa menggunakan teknologi *big* data tersebut. Kehadiran teknologi *big* data ini seolah mampu menangani atau memberikan solusi terhadap perkembangan data yang beggitu cepat dan beragam pada saat ini, dimana *big* data dapat dengan cepat mengolah data yang terstruktur maupun tidak terstruktur dalam jumlah yang begitu banyak sehingga menghasilkan sebuah nilai yang dimana nilai tersebut bisa menjadi indikator pengambilan keputusan serta strategi bisnis yang lebih baik.

Big data juga bukan hanya masalah ukuran yang besar, lebih dari itu yang menjadi ciri khasnya adalah jenis datanya yang sangat beragam dan laju pertumbuhan maupun frekwensi pebuhannya yang tinggi[6].

2.2 Kinerja

Menurut kamus besar bahasa Indonesia, kinerja adalah kemampuan kerja. Kinerja atau kemampuan kerja dari suatu software untuk menangani tugas-tugasnya. Kinerja suatu software bisa menurun karena suatu hal, dan kinerja suatu software juga bisa dioptimalkan. Akanlebih baik kita mengoptimalkan kinerja suatu software agar software tersebut dapat berjalan atau bekerja

sesuai dengan fungsinya. Untuk mengoptimalkan kinerja suatu software kita harus mengetahui faktor-faktor yang dapat mengoptimalkannya.

2.3 Hadoop

Hadoop atau Apache Hadoop adalah software *open source* yang ditulis menggunakan bahasa Java untuk dijalankan secara terdistribusi dan *scalable*. Hadoop dibangun berdasarkan algoritma MapReduce dari Google Inc. Hadoop menyediakan penyimpanan besar-besaran untuk jenis data, serta kemampuan untuk menangani tugas-tugas atau pekerjaan secara bersamaan.

Hadoop muncul karena terinspirasi dari makalah tentang Google MapReduce dan Google File System (GFS) yang ditulis oleh ilmuwan dari Google, Jeffrey Dean dan Sanjay Ghemawat pada tahun 2003. Proses *development* hadoop dimulai pada saat proyek Apache Nutch, yang kemudian baru dipindahkan menjadi sub proyek hadoop pada tahun 2006. Penamaan hadoop sendiri diberikan oleh Doug Cutting yang terinspirasi dari mainan gajah milik anaknya.

Hadoop sendiri sejak tanggal 23 Januari 2008 telah menjadi proyek tingkat atas yang dimiliki lingkungan Apache Software Foundation dan dikembangkan secara terbuka oleh komunitas kontributor secara global. Owen O'Malley adalah orang yang diikut sertakan ke proyek Hadoop pada Maret 2006, dan hadoop 0.1.0 dirilis pada April 2006.

Hadoop membagi file menjadi *block-block* besar dan mendistribusikannya di seluruh *node* di dalam sebuah *cluster*, lalu men*transfer* kode yang dikemas ke dalam *node* untuk memproses data secara *parallel*. Pendekatan ini menggambil keuntungan dari data lokalitas dimana *node* memanipulasi data yang mereka akses dan memungkinkan data untuk di proses arsitektur super komputer yang lebih konvensional yang bergantung pada sistem file *parallel* dimana komputasi dan data di distribusikan melalui jaringan berkecepatan tinggi.

Kerangka dasar Apache Hadoop terdiri dari beberapa modul, yaitu:

- Hadoop Common – berisikan pustaka dan utilitas yang dibutuhkan oleh modul hadoop lainnya.
- Hadoop Distributed File System (HDFS) – sebuah sistem berkas terdistribusi dengan *high-availibility* yang dapat menyimpan data pada mesin komoditas, digunakan untuk menyediakan *bandwidth* sangat tinggi yang di agregasi ke semua *cluster (node)*. Berkas dibagi menjadi *block* data dengan panjang yang baku dan di distribusikan secara redundan (berlebihan) pada simpul (*node*) yang berpartisipasi.

HDFS bekerja menggunakan pendekatan master-slave, dimana sebuah *node* master, yang disebut NameNode, memproses permintaan yang masuk, mengorganisasi berkas di dalam node slave dan menyimpan metadata yang dihasilkannya. HDFS mendukung sistem berkas dengan beberapa ratusan juta file. Panjang *block* berkas maupun tingkat redundansi, keduanya bisa dikonfigurasi.

- Hadoop YARN – sebuah platform manajemen sumber daya yang bertanggung jawab atas pengelolaan sumber daya komputasi dalam sebuah *cluster* dan digunakan untuk penjadwalan aplikasi pengguna.
- Hadoop MapReduce – model pemrograman untuk pengolahan data dalam sekala besar.
- HBase – sebuah database sederhana dan skalabel untuk mengelola data dengan jumlah yang sangat besar dalam cluster hadoop. Database ini didasarkan pada implementasi bebas dari BigTable besutan Google. Struktur data ini cocok untuk data yang jarang berubah, tapi sangat sering ditambahkan. HBase ini bisa mengelola miliaran baris data secara efisien.
- Hive – mempunyai fungsi Data-Warehouse untuk melengkapi Hadoop Hive yaitu bahasa query HiveQL dan indeks. HiveQL sendiri adalah bahasa *query* berbasis SQL dan memungkinkan pengembang untuk menggunakan *sintaks* seperti SQL. Pada musim panas tahun 2008 Facebook yang merupakan pengembang asli dari Hive, menyerahkan Hive menjadi proyek komunitas *open source*.
- Pig – dapat digunakan sebagai bahasa pemrograman *high-level* (Pig Latin) untuk menulis program pada Hadoop MapReduce.
- Chukwa – memungkinkan pemantauan secara *real-time* dari sistem terdistribusi yang sangat besar.
- Zookeeper – digunakan untuk koordinasi dan konfigurasi pada sistem terdistribusi.

Kerangka hadoop sendiri sebagian besar ditulis dalam bahasa pemrograman Java, dengan beberapa kode asli dalam bahasa C dan utilitas baris perintah di tulis sebagai *shell scripts*. Hadoop membutuhkan Java Runtime Environment (JRE) 1.6 atau lebih tinggi.

Hadoop bekerja secara langsung dengan sistem file terdistribusi yang dapat dipasang oleh sistem operasi yang mendasarinya hanya dengan menggunakan file: // URL;. Untuk mengurangi

lalu lintas jaringan, hadoop perlu mengetahui server mana yang paling dekat dengan data, informasi yang dapat diberikan oleh sistem file khusus hadoop. Download Hadoop di https://www.apache.org/dist/hadoop/core/hadoop-2.6.0/hadoop-2.6.0.tar.gz

2.3.1 Hadoop File System (HDFS)

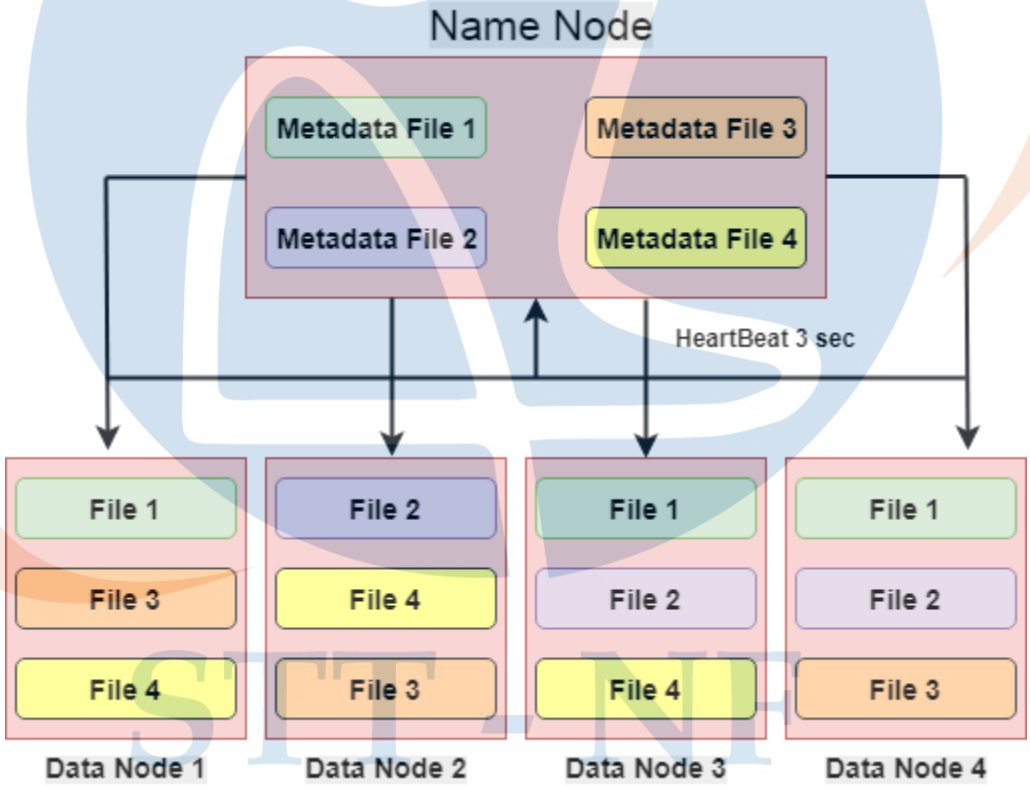
HDFS adalah sistem file yang dirancang untuk menyimpan file yang sangat besar dengan pola akses data streaming, berjalan di cluster pada perangkat keras komoditas [10]**.** Hadoop File System atau HDFS adalah sebuah file terdistribusi, skalabel, dan portable yang ditulis menggunakan bahasa pemrograman Java untuk kerangka Hadoop. HDFS menyimpan file besar (biasanya dalam kisaran gigabytes hingga terabytes) di dalam beberapa mesin. HDFS merupakan sistem penyimpanan terdistribusi dimana HDFS akan melakukan proses pemecahan file besar menjadi bagian-bagian lebih kecil yang kemudian akan didistribusikan ke *cluster-cluster* dari komputer. *Cluster* ini bisanya terdiri dari banyak *node* atau komputer, setiap *node* di dalam *cluster* ini harus ter*install* hadoop untuk bisa berfungsi.

Data yang di potong menjadi lebih kecil disebut *block* dan ukuran *block* dapat diatur sesuai kebutuhan. HDFS menyimpan setiap file sebagai kuran *block*, semua *block* dalam file memiliki ukuran yang sama terkecuali *block* terakhir. *Block-block* file direplikasi untuk toleransi kesalahan. Ukuran *block* dan faktor replikasi dapat dikonfigurasi per filenya. Aplikasi dapat menentukan jumlah replika file. Faktor replikasi dapat ditentukan pada waktu pembuatan file dan dapat diubah nanti. File HDFS ditulis sekali dan memiliki satu penulis secara ketat setiap saat. HDFS ini pada dasarnya adalah sebuah tempat atau direktori di komputer dimana data hadoop tersimpan. Meskipun namanya file *system*, HDFS ini tidak sejajar dengan jenis file *system* dari sistem operasi misalnya NTFS, FAT32. HDFS ini menumpang diatas file *system* milik sistem operasi linux atau windows.

HDFS memiliki komponen utama yaitu NameNode dan DataNode. NameNode adalah sebuah komputer yang bertindak sebagai master, lalu DataNode adalah komputer-komputer dalam hadoop *cluster* yang bertugas sebagai slaves atau anak buah. NameNode mempunyai tanggung jawab menyimpan informasi tentang penempatan block-block data dalam hadoop *cluster*. NameNode juga bertanggung jawab mengorganisir dan mengontrol block-block data yang disimpan tersebar dalam komputer-komputer yang menyusun hadoop *cluster*. Sedangkan

DataNode mempunyai tugas menyimpan *block-block* data yang dialamatkan kepadanya, dan secara berkala melaporkan kondisinya kepada NameNode.

Setiap data atau file yang kita simpan di HDFS akan selalu memiliki lebih dari satu *copy* data atau file tersebut. Hal tersebut di namakan Replication Factor (RF) dimana satu file disimpan di 3 data *node* sehingga jika ada satu DataNode yang rusak, maka DataNode yang lainnya bisa memberikan filenya. Setiap 3 detik sekali, DataNode mengirim sinyal (heartbeat) ke NameNode untuk menunjukkan bahwa DataNode masih aktif. Jika dalam 10 menit NameNode tidak menerima heartbeat dari DataNode, maka DataNode tersebut dianggap rusak atau tidak berfungsi sehingga setiap ada *request read/write* dialihkan ke *node* lain. Dengan heartbeat ini NameNode dapat mengetahui dan menguasai kondisi *cluster* secara keseluruhan. NameNode akan mengirimkan perintah kepada DataNode sebagai balasan atas heartbeat.



Gambar 2.4 1Penyimpanan Data pada HDFS

Untuk HDFS versi pertama (1.x) ada beberapa jenis *node* didalam *cluster*, yaitu:

1. Name node: Node utama yang mengatur penempatan data di *cluster*, menerima job dan program untuk melakukan pengolahan dan analisis data melalu MapReduce. Name node menyimpan metadata tempat data di cluster dan juga replikasi data tersebut.
2. Data Node: node tempat data ditempatkan. Satu *block* di HDFS / datanode adalah 64 MB, jadi sebaiknya data yang disimpan di HDFS ukurannya minimal 64 MB untuk memaksimalkan kapasitas penyimpanan di HDFS.
3. Secondary name node: *node* ini bertugas untuk menyimpan informasi penyimpanan data dan pengolahan data yang ada di name node. Fungsinya kalau ada name node mati dan diganti dengan name node baru maka name node baru bisa langsung bekerja dengan mengambil data dari secondary name node.
4. Checkpoint node dan Backup node: kedua *node* ini berfungsi kurang lebih sama dengan secondary node, yaitu sebagai *backup* dari operasi-operasi data dari name node. *Checkpoint* node melakukan pengecekan setiap *interval* waktu tertentu dan mengambil data dari name node. Dengan *checkpoint* node maka semua operasi perubahan pada data akan terekam. Bedanya dengan secondary name node adalah hanya menyimpan *checkpoint* terkahir. *Backup* node juga berfungsi sama, hanya bedanya data perubahan yang disimpan dilakukan di *memory* dan bukan difile seperti checkpoint dan secondary node.

Untuk HDFS versi terakhir (2.x) ada beberapa perubahan jenis node di cluster untuk meningkatkan performasinya, yaitu:

1. Lebih dari satu name node yang berfungsi sebagai implementasi dari *High Availibility*. Hanya aka nada satu name node yang berjalan di cluster (aktif) sedangkan yang lainnya dalam kondisi pasif. Jika name node yang aktif tiba-tiba rusak atau mati, maka name node yang pasif akan otomatis menjadi aktif dan mengambil alih tugasnya sebagai name node.
2. Secondary name node, *checkpoint node* dan *backup node* sudah tidak diperlukan lagi karena selain fungsi yang redundan, juga lebih baik mengalokasikan node untuk membuat tambahan name node sehingga tingkat *High Availibility* lebih tinggi.
3. Data node tidak ada perubahan yang signifikan di versi HDFS 2.x dari versi sebelumnya.

HDFS memiliki asumsi dan tujuan, yaitu:

1. Kegagalan Perangkat keras

   HDFS mungkin terdiri dari ratusan atau ribuan mesin server, masing-masing menyimpan bagian dari data sistem file. Fakta bahwa ada sejumlah besar komponen dan setiap komponen memiliki probabilitas kegalalan yang menyebabkan beberapa komponen HDFS tidak berfungsi baik, sehingga deteksi kesalahan dan pemulihan otomatis yang cepat dari HDFS merupakan tujuan arsitektur inti dari HDFS.

2. Akses Data Streaming

   Aplikasi yang berjalan di dalam GDFS membutuhkan akses streaming ke set datanya. HDFS dirancang lebih untuk batch processing daripada penggunaan interaktif untuk pengguna. Penekanannya adalah pada throughput data akses yang tinggi daripada latensi rendah dari akses data. POSIX membebankan banyak persyaratan yang tidak diperlukan untuk aplikasi yang ditargetkan untuk HDFS. Semantic POSIX di beberapa bidang utama telah diperdagangkan untuk meningkatkan laju *throughput* data.

3. Kumpulan data besar

   Aplikasi yang berjalan di HDFS memiliki set data besar. Karena khasnya HDFS menyimpan data dalam ukuran yang besar dari gigabyte sampai terabyte. Dengan demikian HDFS di setel untuk mendukung data yang besar.

4. Model koherensi sederhana

   HDFS membutuhkan model akses sekali baca untuk banyak file. File yang pernah dibuat, ditulis, dan ditutup tidak perlu diubah. Hal ini menyederhanakan masalah koherensi data dan memungkinkan akses data *throughput* yang tinggi.

5. Portabilitas di seluruh perangkat keras dan *Platform* perangkat keras Heterogen

   HDFS telah dirancang untuk mudah dibawa daru satu *platform* ke *platform* lainnya. Ini memfasilitasi adopsi HDFS secara luas sebagai *platform* pilihan untuk sejumlah besar aplikasi.

2.3.2 Block Size

Block size adalah potongan-potongan data yang tersimpan di dalam HDFS [9]. Ukuran block size pada HDFS umumnya adalah 128 MB. Dengan begitu, file HDFS dipotong menjadi potongan 128 MB, dan jika memungkinkan setiap potongan akan berada dalam DataNode yang berbeda.

2.3.3 MapReduce

Kerangka kerja perangkat lunak untuk menulis aplikasi dengan mudah yang memproses sejumlah besar data (kumpulan data multi-terabyte) secara paralel pada kelompok besar (ribuan node) perangkat keras komoditas dengan cara yang dapat diandalkan dan toleran kesalahan[7]. Pekerjaan MapReduce biasanya membagi set data input ke dalam potongan independen yang diproses oleh tugas peta dengan cara yang sepenuhnya paralel.

MapReduce memiliki 2 proses utama yaitu, *map* dan *reduce.* Proses *map* yaitu masternode menerima input, kemudian input tersebut dipecah menjadi beberapa subproblem yang kemudian didistribusikan ke *worker nodes. Worker nodes* ini akan memproses subproblem yang diterimanya untuk kemudian apabila problem tersebut diselesaikan, maka akan dikembalikan ke masternode. Proses *reduce* yaitu masternode menerima jawaban dari semua subproblem dari banyak data *nodes,* menggabungkan jawaban-jawaban tersebut menjadi satu jawaban besar untuk mendapatkan penyelesaian dari permasalahan utama.

Dua komponen penting untuk mengeksekusi program MapReduce, yaitu:

a. Job Tracker, komponen ini berada pada NameNode yang memiliki fungsi mengatur eksekusi dari MapReduce. Job Tracker ini akan mengkoordinasikan eksekusi dari MapReduce terhadap seluruh data yang ada dicluster, dan juga akan menjadwalkan eksekusi MapReduce lalu mengulang eksekusi jika eksekusi berikutnya gagal.
b. Task Tracker, komponen ini berada di DataNode yang sebenarnya mengeksekusi MapReduce terhadap data yang ada di cluster. Setelah eksekusi selesai, Task Tracker memberitahu Job Tracker hasil dari eksekusi tersebut, gagal atau berhasil.

2.3.4 YARN (Yet Another Resource Negotiator)

YARN adalah singkatan dari Yet Another Resource Negotiator. YARN adalah platform sumber daya generik untuk mengelola sumber daya dalam sebuah typical cluster. YARN

diperkenalkan pada Hadoop 2.0, yang merupakan kerangka kerja pemrosesan sumber terbuka yang terdistorsi dari Apache Software Foundation. Pada tahun 2012, YARN menjadi salah satu sub proyek dari proyek hadoop apache yang lebih besar. YARN juga diciptakan oleh nama MapReduce 2.0. ini sejak apache hadoop mapreduce telah dirancang ulang apache hadoop YARN.YARN sebagai bahan komputasi generik untuk mendukung MapReduce dan paradigma aplikasi lainnya dalam cluster hadoop yang sama; sebelumnya, ini dibatasi untuk pemrosesan batch menggunakan MapReduce. Ini benar-benar mengubah permainan untuk menyusun kembali hadoop apache sebagai sistem pemrosesan data yang jauh lebih kuat. dengan munculnya YARN, hadoop sekarang terlihat sangat berbeda dibandingkan dengan yang sebelumnya.

YARN memungkinkan banyak aplikasi untuk berjalan secara bersamaan pada cluster bersama yang sama dan memungkinkan aplikasi untuk sumber daya dinegosiasikan berdasarkan kebutuhan. Oleh karena itu, alokasi / manajemen sumber daya adalah pusat dari YARN. YARN telah benar-benar diuji di Yahoo sejak September 2012. Telah di produksi di 30.000 node dan 325 PB data sejak Januari 2013. Baru-baru ini, apache hadoop YARN memenangkan penghargaan kertas terbaik di ACM Sysmposium Cloud Computing (SoCC) pada 2013 [13]

2.4 Java Development Kit (JDK)

Java Development Kit (JDK) adalah implementasi salah satu *Platform* Java, Edisi Standar, *Platform* Java, Edisi *Enterprise*, atau *platform Micro Edition* yang dirilis oleh Oracle Corporation dalam bentuk produk biner yang ditujukan untuk Pengembang Java di Solaris, Linux, macOS atau Windows. JDK termasuk JVM pribadi dan beberapa sumber lain untuk menyelesaikan pengembangan Aplikasi Java. Download JDK yang dibutuhkan di http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

2.5 Secure Shell (SSH)

Secure Shell adalah sebuah protokol jaringan kriptografi untuk komunikasi data yang aman, login antarmuka baris perintah, perintah eksekusi jarak jauh, dan layanan jaringan lainnya antara dua jaringan komputer. Ini terkoneksi, melalui saluran aman atau melalui jaringan tidak aman, server dan klien menjalankan server SSH dan SSH program klien secara masing-masing.

SSH dirancang sebagai pengganti Telnet dan protokol remote shell lainnya yang tidak aman. semua komunikasi antara klien dan server dienkripsi dengan aman dan dilindungi dari modifikasi [11].

2.6 NameNode Benchmark (NNBench)

NNBench adalah aplikasi *benchmark* hadoop yang memiliki tujuan untuk memeriksa konfigurasi NameNode pada multinode cluster hadoop. Pengujian ini dilakukan dengan menggunakan pekerjaan MapReduce sebagai cara yang nyaman untuk membaca dan menulis file secara parallel. Tolok ukur nnbench berguna untuk menguji-beban namenode. Benchmark ini mensimulasikan volume tinggi permintaan manipulasi file terhadap HDFS untuk "stress-test" kemampuan namenode untuk mengelola HDFS [15].

2.7 TestDFSIO

TestDFSIO adalah aplikasi *benchmark* hadoop yang berfungsi menguji kinerja I/O pada HDFS dan juga ini sangat membantu untuk tugas-tugas seperti stress testing HDFS, untuk menemukan bottleneck kinerja di jaringan [8]. Tes ini dapat mengukur waktu yang dibutuhkan untuk membuat sejumlah file besar, dan kemudian menggunakan file yang sama sebagai *input* ke tes untuk mengukur kinerja baca yang dapat dipertahankan oleh HDFS. Setiap file yang dibaca atau ditulis dalam tugas yang terpisah. TestDFSIO terdiri dari dua pengujian, yang pertama adalah membuat data pada HDFS dan yang kedua adalah membaca data yang dibuat dan melakukan pengukuran.

TestDFSIO menguji kinerja I / O HDFS. Itu melakukan ini dengan menggunakan pekerjaan MapReduce sebagai cara mudah untuk membaca atau menulis file secara paralel. Setiap file dibaca atau ditulis dalam tugas peta yang terpisah, dan output peta digunakan untuk mengumpulkan statistik yang berkaitan dengan file yang baru saja diproses. Statistik diakumulasi dalam pengurangan, untuk menghasilkan ringkasan [10].

2.8 Throughput

Throughput adalah kemampuan sebenarnya suatu jaringan dalam melakukan pengiriman data. Biasanya throughput selalu dikaitkan dengan bandwidth, karena throughput bisa juga

disebut dengan bandwidth. Akan tetapi bandwidth bersifat *fix*, sedangkan throughput sifatnya adalah dinamis tergantung *traffic* yang sedang terjadi [12].

2.9 Penelitian Terkait

Dalam penelitian ini peneliti melakukan studi literatur penelitian yang terkait, sebagai komparasi dan keterkaitan dengan masalah yang peneliti ambil. Hal ini bertujuan untuk mengetahui posisi penelitian yang dilakukan peneliti. Pada tabel 1 dengan *field* yang diberi warna kuning atau penelitian dengan nomor (1) merupakan penelitian terkait menganalisis kinerja HDFS dan membahas beberapa masalah kinerja HDFS, hanya saja penulis menggunakan sistem operasi FreeBSD untuk penelitiannya. Pada tabel dengan *field* yang diberi warna hijau atau penelitian dengan nomor (2) merupakan penelitian tentang mencari faktor yang mempengaruhi efisiensi Hadoop Cluster dan melakukan eksperimental peritah membaca dan menulis pada HDFS. Pada tabel dengan *field* yang diberi warna biru atau penelitian dengan nomor (3) merupakan penelitian untuk mengevaluasi kinerja menulis pada HDFS menggunakan dua skema replikasi, yaitu default Pipelined Replication dan Parallel Replication.

No	Judul Penelitian	Tahun	Kesimpulan
1	The Hadoop Distributed Filesystem: Balancing Portability and Performance Oleh Jeffrey Shafer, Scott Rixner, dan Alan L. Cox (Rice University, Houston, TX), USA	2010	Penelitian ini menganalisis kinerja HDFS yaitu: Software Architectural Bottlenecks, Keterbatasan portabilitas dan Asumsi Portabilitas. Penelitian ini membuat sebuah cluster dengan 5 node yang masing-masing node menggunakan sistem operasi FreeBSD.
2	Performance Evaluation of HDFS in Big Data Management	2014	Penelitian ini mencari faktor yang mempengaruhi efisiensi Hadoop Cluster dan melakukan eksperimental HDFS menggunakan TestDFSIO dengan data besar untuk

	Oleh Dipayan Dev, Ripon Patgiri – Department of Computer Science & Engineering National Institute of Technology, Silchar – India		perintah membaca dan menulis file yang berguna untuk mengukur kinerja sistem. Penelitian ini juga menemukan bahwa ada kekurangan dalam hal throughput ketika jumlah file yang relative lebih besar dibandingkan dengan jumlah file yang jauh lebih kecil.
3	Analysis of HDFS RPC and Hadoop with RDMA by Evaluating Write Performance Oleh Somya Singh - Student, CSE Department Amity University Uttar Pradesh, Gaurav Raj - Asst. Prof., CSE Department Amity University, Uttar Pradesh, Gurneet Kaur - Verizon Data Services India, India	2016	Penelitian ini mengevaluasi kinerja menulis pada HDFS menggunakan dua skema replikasi, yaitu default Pipelined Replication dan Parallel Replication. Penerapan Pararel Replication terbukti lebih baik karena Throughtput yang diperoleh oleh Paralel Replication adalah 8% .
4	Implementasi dan Analisis Kinerja HDFS sebagai Infrastruktur Pembangunan *Big* Data Oleh Yunita Surahman, Teknik Informatika STT Terpadu Nurul Fikri	2018	Peneliti akan melakukan implementasi HDFS dan menganalisis kinerja HDFS menggunakan NNBench & TestDFSIO pada lingkungan linux Ubuntu dengan menggunakan 3 node dan kemudian penelitian ini akan melakukan dokumentasi

			langkah-langkah implementasi dari HDFS yang akan membantu dalam merancang serta mengetahui kinerja dari HDFS.

Tabel 2.7.1 Penelitian Terkait