Pengolahan Data Sensor IOT Dengan Apache Spark Menggunakan Metode Batch Processing

Muhammad Dzaky Khairy*, Anya Nur Defitri, Al Hadi Akmal Nuzuli, Anugra Lulu Dyakiyyah, Ahmad Heryanto
Jurusan Sistem Komputer, Universitas Sriwijaya, Indonesia

*Korespondensi: 09011182126001@student.unsri.ac.id

ARTICLE INFO

Article History:
- Received 01 July 2023
- Received in revised form 25 August 2023
- Accepted 19 September 2023
- Available online 30 October 2023

ABSTRAK

Penelitian ini mengeksplorasi pengolahan data sensor IoT menggunakan Apache Spark melalui batch processing. Dalam era IoT, di mana data sensor dari berbagai sumber online melimpah, pengolahan data sangat strategis untuk meraih wawasan berharga. Pemilihan Apache Spark sebagai alat pengolahan didasarkan pada kemampuannya mengatasi volume data besar dengan cepat dan efisien. Melalui metode batch processing, data sensor dikumpulkan pada interval tertentu untuk analisis serentak, mengungkap pola, tren, dan anomali. Hasil analisis ini mendukung pengambilan keputusan yang lebih baik terkait sistem IoT. Penelitian ini melibatkan langkah-langkah penting, mulai dari pengumpulan, pembersihan, transformasi, hingga analisis data sensor. Evaluasi kinerja Apache Spark dalam batch processing mempertimbangkan waktu pemrosesan dan penggunaan sumber daya. Harapannya, penelitian ini memberikan kontribusi signifikan untuk pengembangan sistem IoT yang lebih efisien dan cerdas, serta mendukung pengambilan keputusan berbasis data sensor.

Kata Kunci: Apache Spark, Internet of Things, Batch Processing, Spark, Streaming.

ABSTRACT

This research explores the processing of Internet of Things (IoT) sensor data using Apache Spark through batch processing. In the era of IoT, where sensor data from various online sources is abundant, data processing is crucial for gaining valuable insights. The selection of Apache Spark as a processing tool is based on its ability to handle large volumes of data quickly and efficiently. Through batch processing, sensor data is collected at specific intervals for simultaneous analysis, revealing patterns, trends, and anomalies. The results of this analysis support better decision-making regarding IoT systems. This research involves crucial steps, ranging from data collection, cleaning, transformation to sensor data analysis. The evaluation of Apache Spark's performance in batch processing considers processing time and resource usage. The hope is that this research makes a significant contribution to the development of more efficient and intelligent IoT systems, as well as supporting data-driven decision-making based on sensor data.

Keywords: Apache Spark, Internet of Things, Batch Processing, Spark, Streaming.

1. PENDAHULUAN

Internet of Things (IoT) telah menjadi salah satu paradigma terkemuka dalam dunia teknologi, mengubah cara kita mengumpulkan, mengirimkan, dan menganalisis data dari berbagai perangkat dan sensor yang terhubung secara online. Perkembangan pesat dalam

teknologi sensor dan konektivitas telah memungkinkan pengumpulan data secara real-time dari berbagai lingkungan dan aplikasi, seperti kendaraan otonom, rumah pintar, kota pintar, dan industri otomatis[1]. Oleh karena itu, pengolahan data sensor IoT telah menjadi aspek kunci dalam menggali wawasan berharga, mendukung pengambilan keputusan yang lebih cerdas, dan meningkatkan efisiensi operasional.Konsep IoT dimulai pada tahun 1982 ketika mesin kokas yang dimodifikasi terhubung ke Internet yang mampu melaporkan minuman yang terkandung dan apakah minuman itu dingin [18]. Pada tahun 1999, Bill Joy memberikan petunjuk tentang komunikasi Perangkat ke Perangkat dalam karyanya taksonomi internet [19]. Pada tahun yang sama, Kevin Ashton mengusulkan istilah "Internet of Things" untuk menggambarkan sistem perangkat yang saling terhubung [20].

Salah satu tantangan utama dalam pengolahan data sensor IoT adalah jumlah data yang masif dan keragamannya yang terus berkembang. Data yang dihasilkan oleh sensor dapat mencapai volume yang sangat besar dalam waktu singkat, Sensor tersebut menghasilkan data secara terus menerus dengan kecepatan yang sangat tinggi pula[2]. Dengan adanya pertumbuhan jumlah sensor,peningkatan jumlah data *streaming* akan semakin tinggi dan tidak terbatas. Data *streaming* IoT memberikan manfaat yang besar jika dilakukan pengolahan data. Beberapa pengolahan data lebih bernilai jika data tersebut segera diproses setelah data dihasilkan. Oleh karena itu *stream processing* menjadi hal yang penting dalam aplikasi IoT.

Karakteristik pengolahan data *streaming* yang harus dipenuhi adalah *throughput* yang tinggi, *latency* yang rendah, skalabilitas dan*fault tolerance*[2][3]. Seiring bertambahnya data, hal tersebut memungkinkan penambahan *resource* atau mesin pada *cluster* untuk menjaga kinerja pengolahan[16]. Sehingga sistem harus didukung dengan arsitektur pengolahan paralel secara terdistribusi[17]. Arsitektur yang berbasis *centralized* akan menyebabkan *delay* yang tinggi dalam penyediaan layanan tetapi pada sistem terdistribusi dan jaringan, seringkali adanya celah gangguan seperti matinya server. Sistem harus mampu menopang dan memulihkan dari kesalahan tersebut. Sehingga dibutuhkan sistem yang mampu menangani pengolahan data *stream* IoT dengan *scalability* yang tinggi dan *fault tolerance*.

Apache Spark menjadi salah satu *engine* pengolahan *big data* terbaik karena kapabilitas kinerja yang tinggi, infrastruktur yang kaya dan komunitas yang luas. Pengolahan data *streaming* menggunakan salah satu *Application Programming Interface* (API) *core* Apache Spark yang disebut Spark Streaming[4][5] Spark Streaming membutuhkan komponen lain untuk memenuhi kebutuhan sistem dan tantangan yang dihadapi. Dalam arsitektur IoT, pengolahan data tidak dapat dilakukan oleh Spark Streaming sendiri. Pemilihan komponen tersebut berdasarkan ketersediaan dan menunjang sumber data serta *data sink* pada Spark Streaming. Hal tersebut menentukan perbedaan pada prinsip desain dan optimasi ketika mendesain solusi pengolahan data *streaming* untuk aplikasi IoT[15]. Desain arsitektur yang dibangunberdasarkan penelitian pengolahan terdistribusi, tetap mempertahankan karakteristik seperti *throughput* yang tinggi.

Untuk mengatasi tantangan tersebut maka diperlukannya pendekatan pengolahan yang canggih dan efisien seperti teknologi pengolahan data Big Data seperti Apache Spark telah menjadi sangat relevan. Apache Spark adalah kerangka kerja yang dirancang khusus untuk mengelola data dalam skala besar dan memungkinkan analisis data yang cepat dan distribusi[5]. Metode batch processing yang disediakan oleh Apache Spark dapat digunakan untuk mengolah data sensor IoT dengan efisien. Penelitian ini juga mengembangkan platform pengolahan data sensor IoT berjenis streaming secara terdistribusi menggunakan Spark Streaming[2]. Platform ini sebagai alternatif dalam pengolahan data IoT menggunakan Spark Streaming sebagai framework pengolahan data streaming. Sistem terdiri dari 4 komponen, yaitu pengolahan data, integrasi data, penyimpanan data dan interface.

Penelitian ini bertujuan untuk menyelidiki dan mengembangkan solusi pengolahan data sensor IoT menggunakan Apache Spark dengan metode batch processing. Dalam konteks ini, metode batch processing memungkinkan data sensor dikumpulkan dalam interval waktu tertentu, kemudian dianalisis secara bersamaan, dan hasil analisisnya digunakan untuk mendapatkan pemahaman yang lebih dalam tentang sistem yang dipantau. Penelitian ini juga akan mempertimbangkan aspek kinerja Apache Spark, seperti waktu pemrosesan dan penggunaan sumber daya, untuk memastikan pengolahan data sensor yang efisien.

Dengan mengintegrasikan teknologi Apache Spark dalam pengolahan data sensor IoT, diharapkan bahwa penelitian ini dapat memberikan kontribusi penting dalam pengembangan sistem berbasis IoT yang lebih cerdas, responsif, dan efisien. Hasil penelitian ini dapat bermanfaat bagi berbagai industri dan aplikasi, mulai dari pemantauan lingkungan, manajemen energi, hingga perawatan kesehatan. Selain itu, penelitian ini juga dapat menjadi panduan untuk pengembang dan peneliti lain yang tertarik dalam mengoptimalkan pengolahan data sensor IoT dengan menggunakan Apache Spark.

2. TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Pertama, penelitian ini menganalisis tentang pengolahan data sensor IoT menggunakan Apache Spark dengan metode batch processing. Ini berarti penelitian ini memfokuskan pada cara mengumpulkan, membersihkan, mentransformasi, dan menganalisis data yang dihasilkan oleh sensor IoT dengan bantuan teknologi Apache Spark dalam konteks pengolahan batch.

Kedua, penelitian ini bertujuan untuk mengoptimalkan pengolahan data sensor IoT agar lebih efisien dan cepat menggunakan Apache Spark, sehingga memungkinkan pemantauan yang lebih responsif dan pengambilan keputusan yang lebih baik dalam aplikasi berbasis IoT. Kedua, untuk mengidentifikasi pola, tren, atau anomali dalam data sensor IoT yang dapat digunakan untuk meningkatkan efisiensi operasional dan keamanan sistem IoT. Ketiga, untuk menyediakan panduan dan wawasan bagi pengembang dan peneliti lain yang tertarik dalam pengolahan data sensor IoT menggunakan teknologi Apache Spark.

Ketiga, penelitian ini membahas tentang pengolahan data sensor IoT, yaitu proses pengumpulan, pembersihan, transformasi, dan analisis data yang dihasilkan oleh berbagai sensor yang terhubung secara online dalam lingkungan IoT. Mengapa ini penting? Karena pengolahan data sensor IoT adalah langkah kunci dalam mendapatkan wawasan berharga dari data tersebut. Dengan pengolahan yang tepat, kita dapat mengidentifikasi masalah, mengoptimalkan kinerja, dan mengambil tindakan yang sesuai dalam berbagai aplikasi IoT, seperti pemantauan lingkungan, kendaraan otonom, dan manajemen energi.

Keempat, penelitian ini mencakup berbagai algoritma tergantung pada fokusnya. Beberapa algoritma yang mungkin terlibat dalam pengolahan data sensor IoT dengan Apache Spark meliputi algoritma pemrosesan batch, algoritma pemrosesan data streaming, algoritma deteksi anomali, dan algoritma analisis statistik. Setiap algoritma ini digunakan sesuai dengan kebutuhan analisis data yang dilakukan dalam penelitian ini.

2.2 Teori Penunjang

2.2.1 Internet of Things

Internet of Things (IoT) adalah konsep yang mengacu pada jaringan perangkat fisik yang terhubung secara online dan dapat berkomunikasi satu sama lain serta dengan sistem atau aplikasi. IoT mengintegrasikan teknologi, perangkat keras (hardware), perangkat lunak

(software), serta konektivitas untuk memungkinkan berbagai perangkat, sensor, dan objek sehari-hari untuk mengumpulkan, mentransmisikan, dan bertukar data secara otomatis. Berikut adalah penjelasan lebih rinci tentang elemen-elemennya [1]:

- Perangkat IoT adalah perangkat fisik yang dilengkapi dengan sensor, perangkat keras komunikasi, dan perangkat lunak untuk mengumpulkan dan mengirimkan data. Contohnya termasuk sensor suhu, kamera pengawas, kendaraan otonom, perangkat medis pintar, dan banyak lagi.
- Sensor adalah komponen utama dalam perangkat IoT. Mereka dapat mendeteksi berbagai parameter fisik seperti suhu, kelembaban, tekanan, cahaya, getaran, dan lainnya. Sensor ini mengonversi data fisik menjadi sinyal digital yang dapat diproses oleh perangkat komputer.
- Konektivitas adalah kemampuan perangkat IoT untuk terhubung ke jaringan, baik melalui WiFi, jaringan seluler, Bluetooth, atau protokol khusus seperti LoRaWAN untuk jangkauan jarak jauh. Konektivitas memungkinkan perangkat IoT untuk berkomunikasi dengan perangkat lain dan server pusat.
- Perangkat lunak IoT dilengkapi dengan perangkat lunak yang memungkinkan mereka mengumpulkan data, mengelola data, mengirim data, dan menerima instruksi dari pengguna atau sistem. Ini mencakup sistem operasi khusus IoT dan aplikasi yang dikembangkan untuk tujuan tertentu.
- Jaringan: IoT memanfaatkan infrastruktur jaringan yang ada, termasuk internet, untuk mengirim data antar perangkat dan ke server pusat. Data ini dapat dikirim melalui jaringan kabel, WiFi, jaringan seluler, dll.
- Data yang dikumpulkan oleh perangkat IoT sering kali disimpan, dikelola, dan dianalisis di cloud. Layanan cloud memungkinkan penyimpanan data yang skalabel dan penggunaan kapasitas komputasi yang fleksibel untuk analisis data yang kompleks.

Dengan pertumbuhan cepatnya teknologi IoT, kita melihat peningkatan penggunaan dalam aplikasi-aplikasi yang beragam, yang melibatkan perangkat IoT untuk meningkatkan efisiensi, produktivitas, dan kualitas hidup. Dari rumah pintar yang dapat mengendalikan pencahayaan, suhu, hingga pengukuran kesehatan, hingga transportasi otonom yang mengutamakan keselamatan, IoT telah mengubah cara kita berinteraksi dengan dunia di sekitar kita dan memberikan peluang yang tak terbatas untuk inovasi di masa depan. Namun, penting juga untuk memahami bahwa dengan pertumbuhan IoT, juga ada tantangan yang terkait dengan privasi, keamanan, dan pengelolaan data yang perlu diatasi secara cermat.

2.2.2 Big Data Analytic

Big Data Analytics adalah proses menganalisis, memproses, dan menginterpretasikan data besar (big data) untuk menghasilkan wawasan dan informasi[6]. Data yang dianalisis dapat berupa data terstruktur, semi-terstruktur, atau tidak terstruktur, dan dapat berasal dari berbagai sumber, seperti log, transaksi, sensor, media sosial, dan perangkat mobile.Analisa big data melibatkan pengolahan data besar dengan mempertimbangkan beberapa komponen utama :

1. Volume Big Data (Volume): Big Data Volume merujuk pada kuantitas atau jumlah data yang sangat besar. Dalam konteks Big Data, "big" sendiri menggambarkan volume data yang sangat besar. Saat ini, data yang berjumlah dalam petabyte di perkirakan akan terus meningkat hingga mencapai zettabyte di masa depan. Jumlah data yang sangat besar sehingga mencapai ukuran ratusan terabytes hingga petabytes. Merupakan fokus utama; mengelola dan menganalisis jumlah data yang sangat besar ini adalah tantangan utama karena melebihi kapasitas metode tradisional untuk menangani jumlah data yang sangat besar[7].
2. Kecepatan Pemrosesan (Velocity): Big Data Velocity adalah salah satu karakteristik kunci

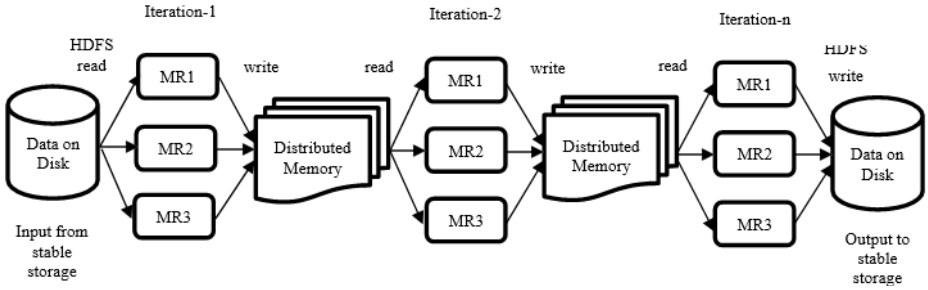
dari Big Data yang mengacu pada kecepatan atau laju penerimaan data serta kemampuan untuk segera bertindak berdasarkan data tersebut. Untuk mendapatkan wawasan secara real-time atau mendekati real-time, sistem harus memproses data dengan kecepatan tinggi, seperti transaksi perbankan dalam waktu nyata atau aliran tweet di media sosial[8].
3. Ragam Data (Varietas): Variety data adalah ukuran sejauh mana data direpresentasikan dalam berbagai bentuk seperti teks, gambar, video, audio, dan lain sebagainya. Format data yang tidak kompatibel, struktur data yang tidak selaras, dan semantik data yang inkonsisten adalah tantangan yang signifikan dan dapat mengakibatkan penyebaran analitik yang kurang efektif. Data dapat berasal dari berbagai sumber dalam berbagai format, seperti teks, video, gambar, suara, dan data terstruktur dan semi-terstruktur. Salah satu masalah terbesar adalah mengolah berbagai jenis data secara efektif[9].

2.2.3 Apache Spark

Apache Spark adalah platform terdistribusi yang digunakan dalam analisis data berskala besar dan Big Data Analytics[5][10]. Ini menawarkan antarmuka pemrograman pada cluster yang berfokus pada paralelisme data. Berbagai fitur unggulan Apache Spark mencakup:

1. RDD (Resilient Distributed Datasets), yang menyediakan API untuk melakukan transformasi data seperti map, filter, dan reduce, serta tindakan seperti count, collect, dan save[5].
2. DataFrame, yang menyediakan antarmuka pemrosesan data berbasis kolom yang terstruktur.
3. Spark Core, inti Apache Spark yang digunakan dalam pemrosesan data paralel dan terdistribusi[5].
4. Spark SQL, modul yang memproses data secara paralel dan terdistribusi[5].
5. MLlib Apache Spark, sebuah lembaga yang menyediakan berbagai algoritma pengajaran mesin.
6. GraphX, yang memanfaatkan API pemrosesan data untuk komputasi paralel.
7. Spark Streaming, kerangka stream yang berjalan di Apache Spark untuk memproses data secara real-time.
8. Spark Cluster Managers, komponen Spark yang membantu manajemen cluster dan sumber daya.

Tidak seperti MapReduce, Apache Spark menyimpan semua proses iterasi ke dalam memori, bukan ke disk, sehingga 100 kali lebih cepat di dalam memori daripada MapReduce. Perbandingan proses iterasi antara Apache Spark dan MapReduce digambarkan di bawah ini saat mengolah data.



Gambar 1. Metode iterasi Apache Spark

Pada Apache Spark, proses iterasi berarti pengulangan atau iterasi berulang atas operasi atau pemrosesan data yang dilakukan berulang kali.

2.2.4 Hadoop

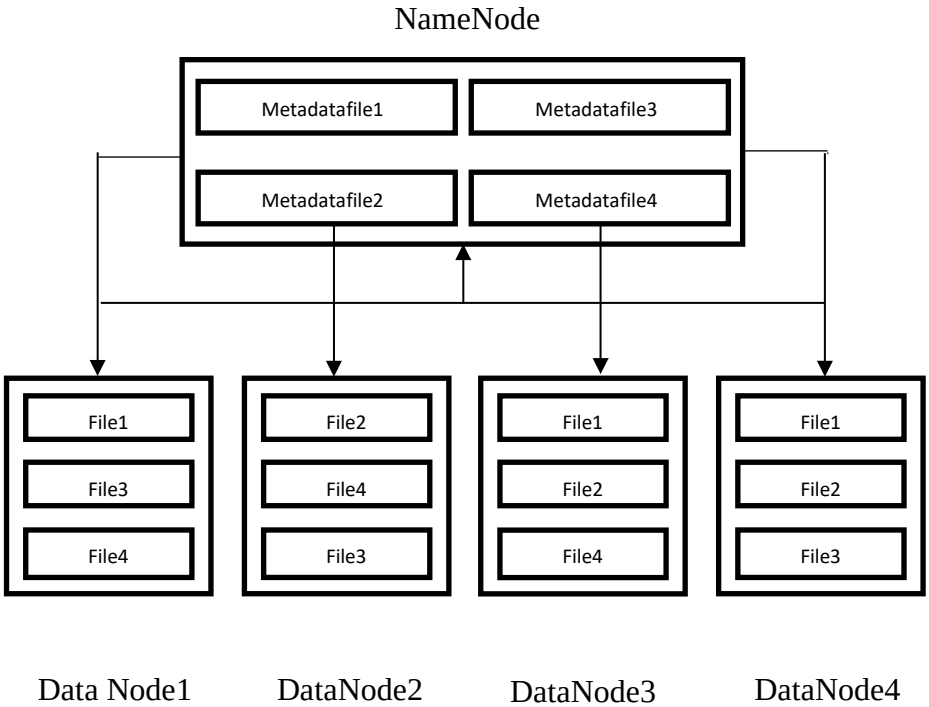
Hadoop adalah kerangka kerja open-source yang digunakan untuk mengelola dan memproses data besar yang tersebar di cluster komputer. Kerangka kerja ini dibangun untuk mengatasi kebutuhan pemrosesan data yang sangat besar dengan menggunakan sekelompok komputer yang bekerja sama satu sama lain.

Dua komponen utama kerangka kerja Hadoop adalah Hadoop Distributed File System (HDFS) dan Apache MapReduce[6] :

1. HDFS adalah sistem penyimpanan data terdistribusi yang memungkinkan penanganan data besar, akses data yang cepat, dan ketahanan terhadap kegagalan node. Ini memungkinkan penyimpanan file dalam potongan-potongan kecil di seluruh cluster dengan replikasi untuk keamanan dan keandalan data[11].
2. Apache MapReduce adalah model pemrograman dan prosedur yang digunakan untuk melakukan pemrosesan data terdistribusi di kluster Hadoop.

2.2.5 Hadoop Distributed File System (HDFS)

Sistem penyimpanan data terdistribusi Hadoop (HDFS) adalah bagian penting dari ekosistem Hadoop dan dirancang untuk menyimpan data di cluster komputer terdistribusi[6]. Ini memungkinkan akses cepat dan efisien ke data dari berbagai node dalam cluster.



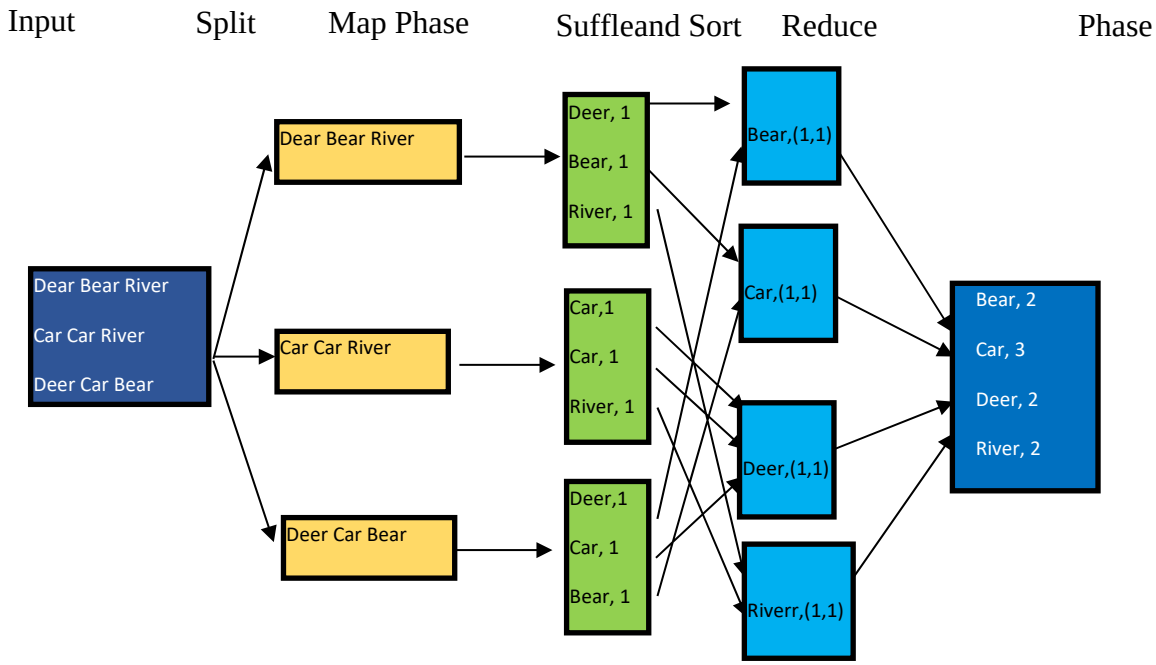
Gambar 2.Pemindahan data ke HDFS

Dua komponen utama HDFS yaitu NameNode dan DataNode. NameNode berfungsi sebagai pusat kontrol dan bertanggung jawab untuk mempertahankan informasi tentang lokasi penyimpanan block data di kluster Hadoop, mengelola dan mengatur block data yang

tersebar di semua komputer di kluster Hadoop, dan menyimpan block data yang diarahkan padanya[10].

2.2.6 MapReduce

MapReduce memiliki keunggulan dalam menangani data besar dan tugas-tugas yang dapat diparalelkan dengan baik karena memungkinkan pemrosesan data paralel yang efisien karena setiap langkah dapat dilakukan secara independen dan setiap elemen data dapat diolah secara terpisah[10]. MapReduce adalah inti dari kerangka kerja pemrosesan data terdistribusi sistem Apache Hadoop. HDFS (Hadoop Distributed File System) digunakan untuk penyimpanan data dan MapReduce digunakan untuk pemrosesan data[3].



Gambar 3. Diagram MapReduce yang bekerja

2.2.7 Batch Processing

Batch Processing adalah metode untuk melakukan transformasi pada sejumlah besar data. Setiap blok data diproses secara individu, dengan interval waktu tertentu yang memisahkan prosesnya. Jenis pemrosesan ini umumnya digunakan ketika data telah terakumulasi selama periode waktu tertentu[2].

3. METODELOGI PENELITIAN

3.1 Alat dan Bahan

Perlengkapan yang diperlukan untuk penelitian ini mencakup perangkat lunak, perangkat keras, dan data serta informasi yang mendukung selama penelitian berlangsung.

3.1.1. Alat

Alat yang digunakan pada penelitian berupa:

1. Perangkat Lunak (Software)
    a. Windows 10/11
    b. Virtual Box/Vmware
    c. Linux Mint OS /Ubuntu
    d. Jupyter Notebook
    e. Spark Framework digunakan untuk pemrosesan big data
2. Perangkat Keras (Hardware)
    a. Processor 13/17
    b. Memory DDR3 8GB / DDR4 32GB
    c. Local Drive 35GB

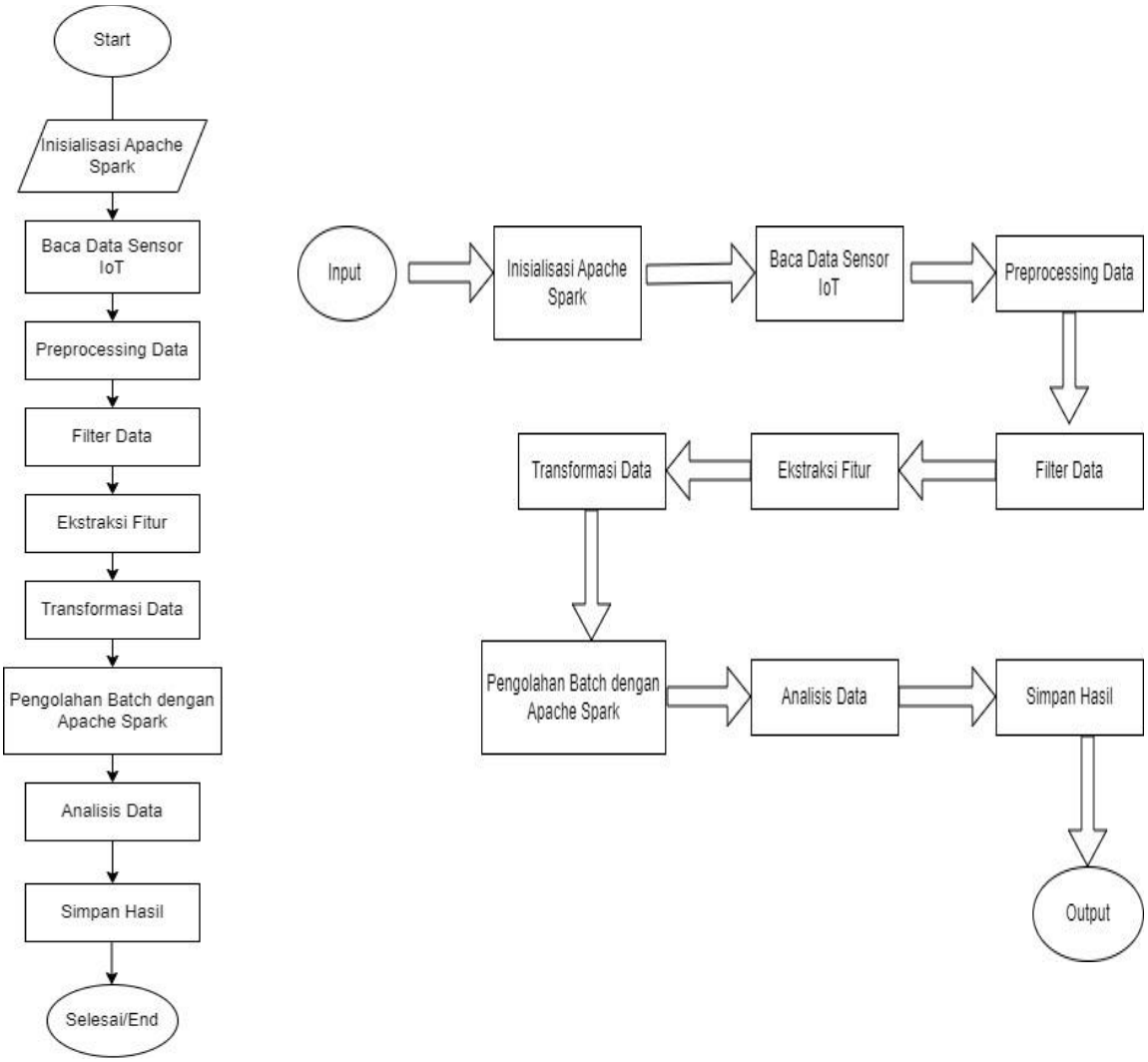
3.1.2. Bahan

Bahan yang digunakan dalam penelitian ini yaitu iot_sensordata yang berasal dari Kaggle

3.2 Alur Penelitian

Penelitian ini mengikuti serangkaian tahapan, termasuk studi literatur dan persiapan alat serta bahan, perancangan sistem, implementasi sistem, pengujian sistem, dan proses dokumentasi. Bisa dilihat pada Gambar 4. menunjukkan alur penelitian (Flowchart) yang dilakukan.

1. Langkah pertama adalah mengumpulkan data sensor IoT dari berbagai sumber. Data ini dapat berasal dari perangkat sensor yang terhubung ke jaringan IoT, sistem penyimpanan data, atau sumber eksternal lainnya. Setelah data dikumpulkan, langkah berikutnya adalah memproses data.
2. Proses ini dapat mencakup langkah-langkah, Mengambil data yang relevan dari sumber data, elakukan transformasi pada data, seperti mengubah format data, menggabungkan data dari beberapa sumber, membersihkan data dari nilai-nilai outlier.
3. Data yang telah diproses kemudian disimpan dalam penyimpanan yang sesuai, seperti Hadoop Distributed File System (HDFS) atau penyimpanan data lainnya yang dapat diakses oleh Apache Spark.
4. Menggunakan Apache Spark Batch Processing, membuat Spark Context untuk memulai proses batch. Spark Context adalah antarmuka utama yang digunakan untuk berinteraksi dengan Apache Spark, mengunakan Spark untuk membaca data ke dalam RDD (Resilient Distributed Dataset) atau DataFrame, menerapkan operasi-transformasi dan analisis yang diperlukan pada data[5].
5. Setelah analisis selesai, hasilnya disimpan dalam format Parquet atau database. Setelah data diolah, hasil analisis dapat digunakan untuk membuat laporan, grafik, atau visualisasi yang akan membantu untuk memahami data sensor IoT dan membuat keputusan berdasarkan informasi yang diberikan.



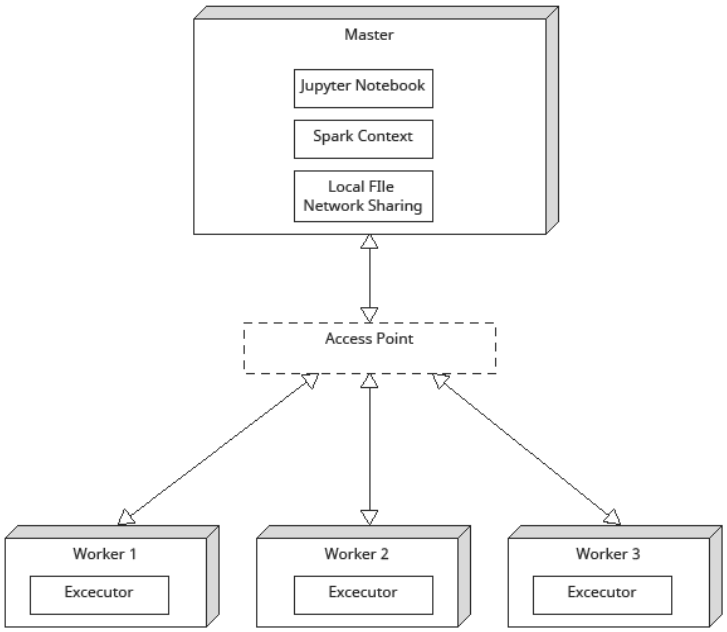
Gambar 4. Flowcart dan Blok Diagram

3.3 Perancangan Sistem

Perancangan sistem pemrosesan paralel ini mencakup topologi sistem yang dari berbagai komponen untuk efisiensi sumber daya, dan program MapReduce untuk mengoptimalkan pemrosesan data.

3.3.1 Topologi

Sebuah topologi menggambarkan arsitektur dan interaksi komponen dalam sebuah sistem yang menggunakan Apache Spark untuk komputasi terdistribusi. Pada Gambar 5. ini adalah contoh dari topologi "Cluster" dalam konteks komputasi terdistribusi.



Gambar 5. Topology system

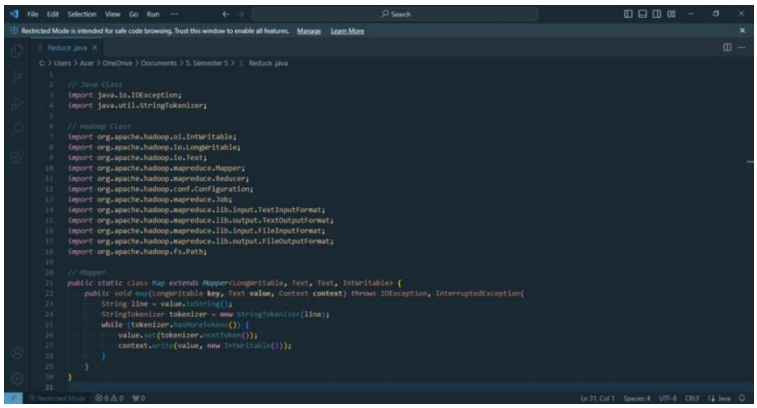
1. Pada Master: Jupyter Notebook: Ini adalah lingkungan pengembangan interaktif yang memungkinkan untuk membuat, menjalankan, dan berinteraksi dengan kode Python secara interaktif. Jupyter Notebook sering digunakan untuk mengembangkan dan menjalankan kode yang akan dijalankan di lingkungan Apache Spark.
2. Spark Context: Spark Context adalah komponen inti dari Apache Spark yang digunakan untuk mengontrol akses ke cluster Spark. Ini memungkinkan untuk mengirimkan tugas komputasi ke cluster, mengelola data terdistribusi, dan melakukan berbagai operasi dengan Spark[5][12].
3. Local File Network Sharing: Ini merujuk pada berbagi berkas lokal di antara berbagai komponen dalam lingkungan Spark. File-file ini bisa menjadi data yang akan diproses oleh Spark[13].
4. Pada tahap Access Point adalah titik akses atau pintu masuk ke cluster Spark. Ini adalah tempat di mana klien atau pengguna dapat berkomunikasi dengan cluster Spark. Access Point ini akan mengelola permintaan dan tugas yang dikirimkan oleh klien ke cluster Spark[14].
5. Dan pada tahap Worker1, Worker2, dan Worker3: Executor: Executor adalah proses yang berjalan di node-node yang ada dalam cluster Spark. Setiap worker biasanya memiliki beberapa executor yang berjalan secara paralel. Executor ini bertugas untuk menjalankan tugas komputasi yang diberikan oleh Spark Context. Mereka menerima tugas dan data yang diperlukan untuk menjalankan tugas tersebut. Ketiga worker ini memiliki peran yang sama, yaitu menjalankan tugas Spark. Mereka adalah bagian dari cluster yang berkontribusi pada pemrosesan data secara terdistribusi.

Secara keseluruhan, dalam lingkungan ini memiliki satu "master" yang mengelola cluster Spark dan satu atau lebih "worker" yang menjalankan tugas komputasi secara paralel[3]. "Access Point" adalah antarmuka melalui mana pengguna atau klien dapat berkomunikasi dengan cluster Spark, dan "Jupyter Notebook" mungkin digunakan untuk mengembangkan kode Spark dan mengirimkannya ke cluster untuk dieksekusi. "Local

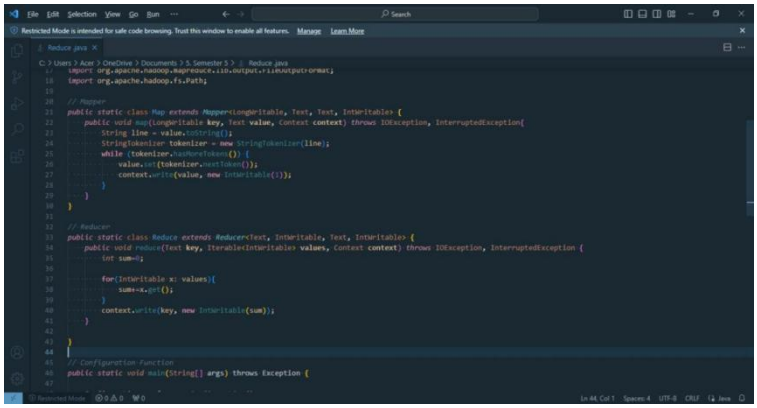
File Network Sharing" mungkin digunakan untuk berbagi data antara komponen dalam lingkungan Spark.

3.3.2 Program MapReduce

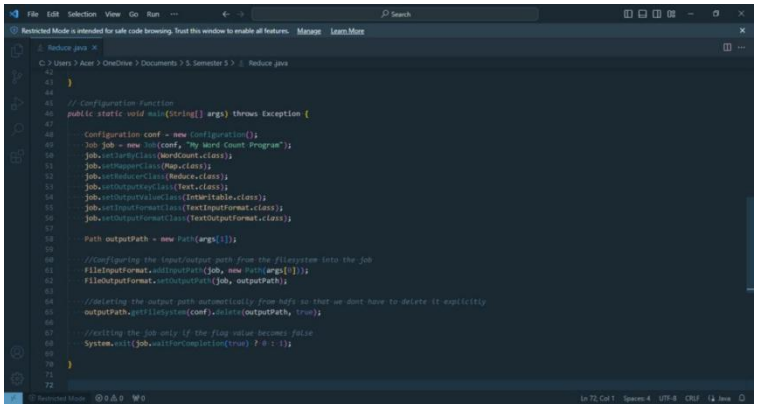
Dalam konteks Hadoop MapReduce, maka dengan mengatur konfigurasi kluster Hadoop, mengimplementasikan Mapper untuk mengolah data masukan, dan mengimplementasikan Reducer untuk menghasilkan hasil akhir. Progrsm bisa dilihat pada gambar dibawah ini :



Gambar 6. Hadoop Class



Gambar 7. Mapper dan Reducer



Gambar 8. Fungsi Konfigurasi

Di atas adalah contoh program menggunakan paradigma MapReduce yang ditulis dalam bahasa pemrograman Java untuk menghitung jumlah kata dalam sebuah file teks dengan menggunakan framework Apache Hadoop. Program ini terdiri dari tiga kelas utama: MyMapReduceProgram, MapClass, dan ReduceClass.

Kelas MapClass berperan sebagai Mapper yang membaca baris-baris teks dari file input, memecahnya menjadi kata-kata, dan mengirimkan setiap kata ke konteks dengan nilai awal 1, yang merepresentasikan bahwa kata tersebut ditemukan sekali. Sementara itu, kelas ReduceClass berfungsi sebagai Reducer yang menggabungkan kata-kata yang sama (dengan kunci yang sama) dan menghitung jumlah kemunculannya. Hasil akhirnya adalah jumlah kata yang sama yang dicetak ke output.

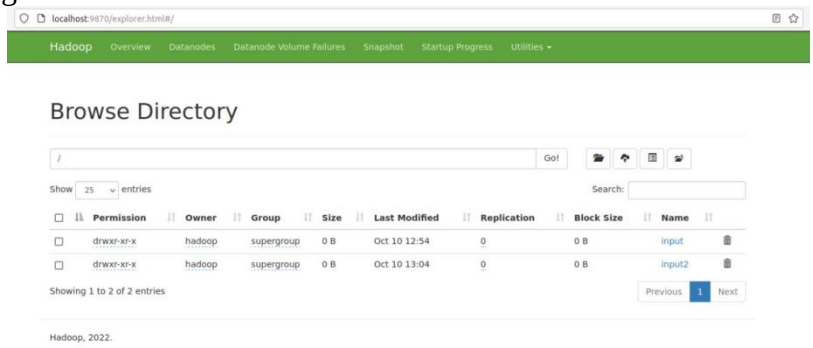
Metode main berperan sebagai titik masuk utama program, di mana konfigurasi pekerjaan MapReduce ditentukan, termasuk pengaturan kelas Mapper, Reducer, jenis kunci dan nilai, serta format input dan output. Path input dan output juga didefinisikan di sini. Pekerjaan MapReduce dijalankan menggunakan job.waitForCompletion(true). Jika pekerjaan selesai tanpa kesalahan, program akan mengembalikan status 0; jika terjadi kesalahan, program akan mengembalikan status 1. Program diakhiri dengan menjalankan pekerjaan MapReduce dan menghapus output jika sudah ada.

4. HASIL PENELITIAN

Setelah melakukan perancangan dan penerapan sistem, langkah berikutnya ialah mengevaluasi dan menganalisis proses pengolahan dan pemrosesan data secara paralel yang diterapkan. Berikut penjelasan nya :

1. Direktori

Hadoop menggunakan sistem file distribusi yang disebut Hadoop Distributed File System (HDFS). Tampilan nya bisa dilihat pada gambar 9. HDFS memiliki struktur direktori yang mirip dengan sistem file biasa.

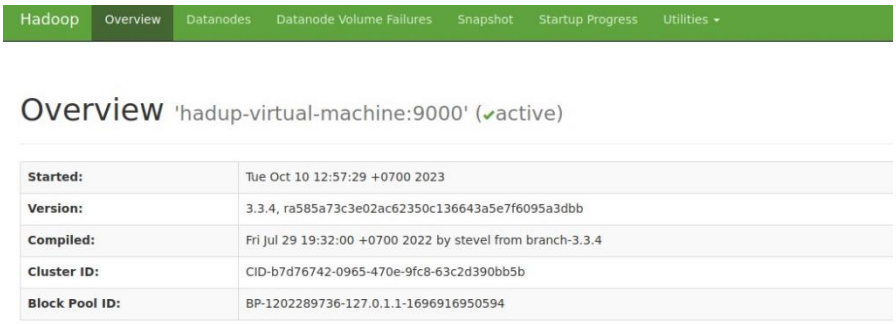


Gambar 9. Browse directory

Direktori ini memungkinkan penyimpanan dan pengelompokan file yang akan diakses dan diproses oleh sistem Hadoop.

2. Overview

Gambaran umum untuk membantu dalam perencanaan dan implementasi solusi yang sesuai dengan kebutuhan.

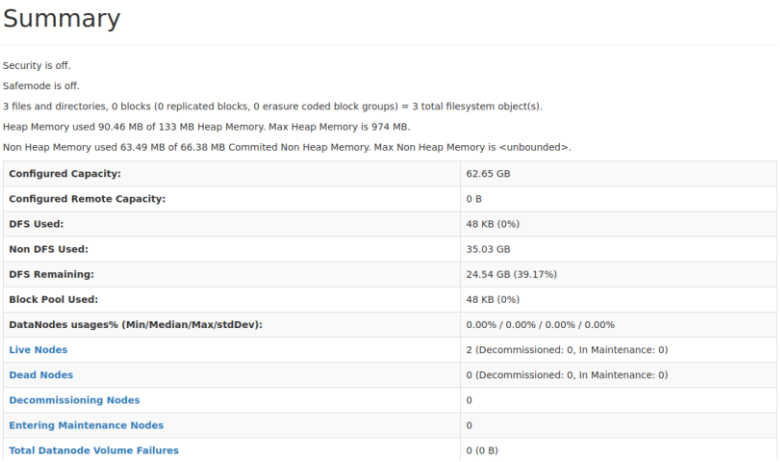


Gambar 10. Overview

Ini termasuk HDFS untuk penyimpanan data, MapReduce untuk pemrosesan data, dan komponen-komponen lain seperti YARN (Yet Another Resource Negotiator) untuk manajemen sumber daya dan banyak lagi.

3. Summary

Untuk menghitung ringkasan statistik dari data yang disimpan dalam Hadoop Distributed File System (HDFS). Ringkasan atau gambaran singkat bisa di lihat pada gambar 11 yang menyajikan informasi kunci tentang hasil dari suatu operasi yang dijalankan di atas data.

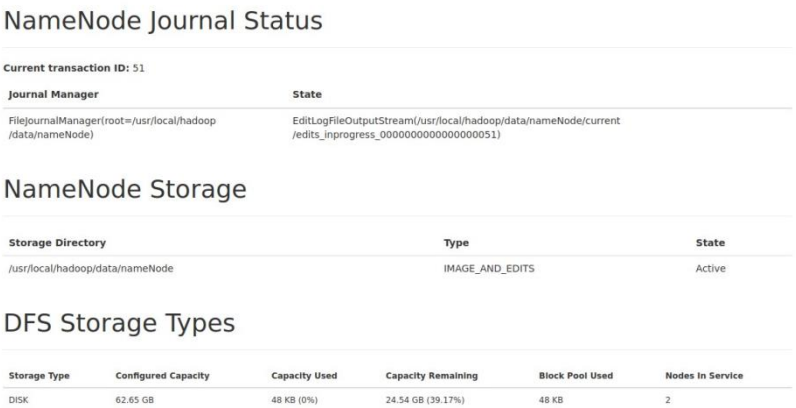


Gambar 11. Summary

Fungsi Summary ini umumnya digunakan dalam konteks pemrosesan data besar (big data) untuk mendapatkan wawasan cepat tentang data tanpa harus memproses seluruh dataset secara detail.

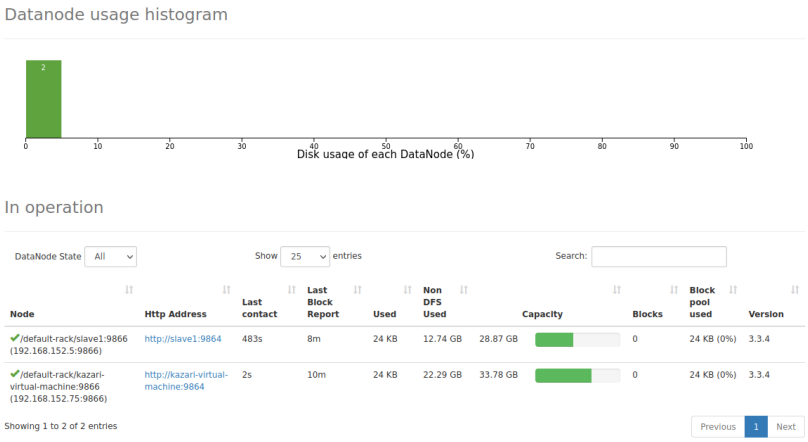
4. NameNode dan DataNode

NameNode dan DataNode adalah dua komponen utama dari Hadoop Distributed File System (HDFS) yang merupakan sistem penyimpanan terdistribusi di lingkungan Hadoop. Ini adalah bagian inti dari infrastruktur Hadoop yang memungkinkan penyimpanan dan pengelolaan data yang besar dan terdistribusi.



Gambar 12. NameNode

NameNode merupakan salah satu komponen utama dalam HDFS. Ini bertanggung jawab untuk menyimpan metadata HDFS, seperti informasi tentang file dan direktori, tetapi tidak data sebenarnya. NameNode berfungsi sebagai master node yang mengelola seluruh sistem file dan menjaga metadata agar tetap konsisten. Jika NameNode gagal, maka seluruh sistem file HDFS dapat menjadi tidak dapat diakses.



Gambar 13. Data Node

DataNode adalah komponen lain dalam HDFS yang menyimpan data sebenarnya. Setiap node dalam kluster Hadoop memiliki DataNode yang bertanggung jawab untuk menyimpan data fisik pada disk lokal. DataNode mentransfer data antara disk lokalnya dan klien yang mengakses data. DataNode juga berkomunikasi secara teratur dengan NameNode untuk melaporkan status dan replikasi data.

5. Menjalankan Spark

Dari gambar 13 dianjutkan dengan mengcoding dengan bahasa pemrograman python, berikut beberapa gambar dan penjelasannya.

```
import warnings
warnings.filterwarnings("ignore")

from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler, StringIndexer
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.functions import col
```

```
spark = SparkSession.builder \
    .appName("Pemerosesan Pararel Kelompok 1") \
    .master("spark://hadup-virtual-machine:7077") \
    .config("spark.some.config.option", "config-value") \
    .getOrCreate()

spark
```

```
<pyspark.sql.session.SparkSession at 0x7f42d954c310>
```

Gambar 14. Import library untuk menjalankan spark

Penelitian ini menggunakan bahasa pemerograman Python yang digunakan untuk menganalisis data telemetri IoT dari berbagai perangkat. Dalam penelitian ini, kami menggunakan berbagai modul Python, termasuk PySpark untuk pemrosesan data besar, Pandas untuk manipulasi data, dan Matplotlib untuk visualisasi data. Awalnya, kami menginisialisasi sesi Spark untuk pemrosesan data besar dengan mengatur konfigurasi yang sesuai. Selanjutnya, kami melakukan enumerasi terhadap file dalam direktori yang mengandung file data, memastikan ketersediaan file yang akan digunakan dalam analisis.

Kami juga mendefinisikan fungsi untuk mengonversi timestamp dalam format UNIX menjadi format datetime yang lebih mudah dipahami oleh Pandas. Data dari file CSV dibaca menggunakan Pandas, dan selama proses ini, kami mengonversi suhu dari satuan Celsius ke Fahrenheit untuk konsistensi.

```
import os
import sys
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter

for dirname, _, filenames in os.walk('iot_telemetry_data.csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

def parse(x):
    return pd.to_datetime(x, infer_datetime_format=True,
unit='s',utc=True)

nrows_read = 100000 # specify 'None' if want to read whole file
(405,184 rows)
data path = 'iot telemetry data.csv'

df = pd.read_csv(data_path,
                 delimiter=',',
                 nrows = nrows_read,
                 header=0,
                 infer_datetime_format=True,
                 date_parser=parse,
                 index_col=['ts'])

df = df.sort_values(by='ts', ascending=True)
```

```
df['temp'] = (df['temp'] * 1.8) + 32

df.head(5)
```

ts	device	co	humidity	light
2020-07-12 00:02:08+00:00	b8:27:eb:bf:9d:51	0.004956	51.000000	False
2020-07-12 00:02:08+00:00	b8:27:eb:bf:9d:51	0.004960	50.900000	False
2020-07-12 00:02:08+00:00	1c:bf:ce:15:ec:4d	0.004391	78.599998	True
2020-07-12 00:02:08+00:00	00:0f:00:70:91:0a	0.002840	76.000000	False
2020-07-12 00:02:08+00:00	b8:27:eb:bf:9d:51	0.004914	50.900000	False

ts	lpg	motion	smoke	temp
2020-07-12 00:02:08+00:00	0.007651	False	0.020411	72.860000
2020-07-12 00:02:08+00:00	0.007656	False	0.020425	72.680000
2020-07-12 00:02:08+00:00	0.007009	False	0.018589	80.600000
2020-07-12 00:02:08+00:00	0.005114	False	0.013275	67.639999
2020-07-12 00:02:08+00:00	0.007604	False	0.020276	72.680000

```
# filter temp/humidity, by device, for outliers (>1% & <99%)
df = df.loc[df['temp'] >df.groupby('device').temp.transform(lambda x:
x.quantile(.01))]
df = df.loc[df['temp'] < df.groupby('device').temp.transform(lambda x:
x.quantile(.99))]

df = df.loc[df['humidity'] >
df.groupby('device').humidity.transform(lambda x: x.quantile(.01))]
df = df.loc[df['humidity'] <
df.groupby('device').humidity.transform(lambda x: x.quantile(.99))]

# group data by iot device
groups = df.groupby('device')

print('DataFrame Stats')
print('.....')
print('Record count: {:,}'.format(df['temp'].count()))
print('DataFrame size (MB):
{:,.2f}'.format(sys.getsizeof(df)/1024/1024))
print('.....')
print('Time range (min): {:%Y-%m-%d %H:%M:%S %Z}'.format(df.index[1]))
print('Time range (max): {:%Y-%m-%d %H:%M:%S %Z}'.format(df.index[-
1]))
print('Temperature (min): {:.2f}'.format(df['temp'].min()))
```

Gambar 15. Menampilkan informasi spark session

Data ini dihasilkan dari tiga rangkaian sensor yang dibangun khusus yang terhubung ke perangkat Raspberry Pi. Setiap perangkat IoT ditempatkan di lokasi fisik dengan kondisi lingkungan yang berbeda. Ada tiga perangkat IoT dengan kondisi lingkungan yang berbeda: stabil dengan suhu lebih dingin dan lebih lembab, suhu dan kelembapan yang sangat variabel,

serta stabil dengan suhu lebih hangat dan lebih kering. Setiap perangkat IoT mengumpulkan tujuh jenis pengukuran dari empat sensor dalam interval waktu yang teratur, termasuk suhu, kelembapan, karbon monoksida (CO), gas petroleum cair (LPG), asap, cahaya, dan gerakan.

Data ini mencakup periode dari 07/12/2020 00:00:00 UTC hingga 07/19/2020 3:59:59 UTC, dengan total 405,184 baris data. Data ini diatur dalam format pesan MQTT (Message Queuing Telemetry Transport) dengan struktur seperti JSON. Setiap pesan berisi bacaan sensor bersama dengan ID perangkat yang unik dan timestamp.Ada sembilan kolom dalam dataset, termasuk timestamp (ts), nama perangkat (device), tingkat karbon monoksida (co), kelembapan (humidity), deteksi cahaya (light), tingkat gas petroleum cair (lpg), deteksi gerakan (motion), tingkat asap (smoke), dan suhu (temp).

```
print('Temperature (max): {:.2f}'.format(df['temp'].max()))
print('Humidity (min): {:.2f}{}'.format(df['humidity'].min(), '%'))
print('Humidity (max): {:.2f}{}'.format(df['humidity'].max(), '%'))
print('.....')
print('Record count:\n{}'.format(groups.size()))

DataFrame Stats
-------------
Record count: 93,800
DataFrame size (MB): 11.09
-------------
Time range (min): 2020-07-12 00:02:08 UTC
Time range (max): 2020-07-13 23:42:56 UTC
Temperature (min): 65.66
Temperature (max): 83.12
Humidity (min): 47.90%
Humidity (max): 77.80%
-------------
Record count:
device
00:0f:00:70:91:0a    25221
1c:bf:ce:15:ec:4d    24633
b8:27:eb:bf:9d:51    43946
dtype: int64
```

Gambar 16. Informasi dataset

Data dimuat ke dalam Pandas DataFrame dan diurutkan berdasarkan timestamp. Suhu awalnya dalam Celsius diubah menjadi Fahrenheit. Outlier dalam suhu dan kelembapan diidentifikasi dan dihapus dengan membandingkan nilai suhu dan kelembapan perangkat dengan persentil tertentu Data dikelompokkan berdasarkan ID perangkat IoT yang berbeda, yang memungkinkan analisis yang lebih terperinci untuk setiap perangkat. Statistik deskriptif dicetak, termasuk jumlah rekaman, ukuran DataFrame, rentang waktu, serta suhu dan kelembapan minimum dan maksimum. Kode dimulai dengan impor modul-modul yang diperlukan, seperti Pandas untuk manipulasi data, Matplotlib untuk visualisasi, dan beberapa modul lainnya. Selanjutnya, kode enumerasi direktori di dalam konteks Kaggle untuk mencari file data yang disebut 'iot_telemetry_data.csv'.

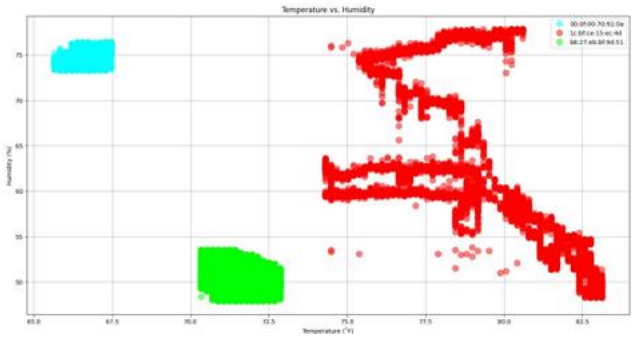
Fungsi parse didefinisikan untuk mengonversi timestamp dalam format UNIX menjadi format datetime yang lebih mudah dipahami oleh Pandas. Kemudian, data dari file CSV tersebut dimuat ke dalam Pandas DataFrame. Data ini diurutkan berdasarkan timestamp, dan suhu yang awalnya dalam satuan Celsius diubah ke Fahrenheit untuk konsistensi. Data kemudian diproses lebih lanjut dengan menghapus outlier suhu dan kelembapan menggunakan persentil ke-1 dan ke-99 untuk masing-masing perangkat IoT. Data yang sudah diproses kemudian dikelompokkan berdasarkan perangkat IoT yang berbeda.

```
colors = ['cyan', 'red', 'lime']

_, ax = plt.subplots(1, 1, figsize=(18, 9))

for i, (device, group) in enumerate(groups):
    ax.plot(group.temp,
            group.humidity,
            marker='o',
            linestyle='',
            alpha=.5,
            ms=10,
            label=device,
            color=colors[i])  # Assign a color to each device

ax.grid()
ax.margins(0.05)
ax.legend()
plt.title('Temperature vs. Humidity')
plt.xlabel('Temperature (°F)')
plt.ylabel('Humidity (%)')
plt.show()
```

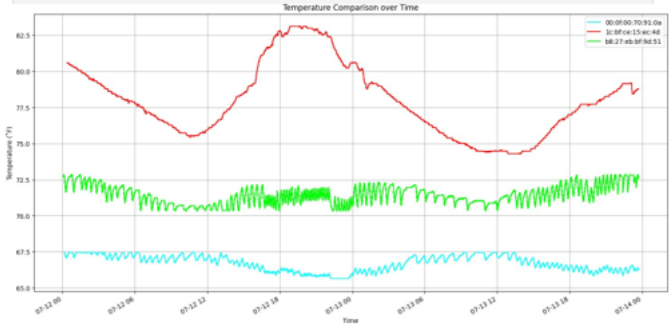


Gambar 17. Grafik Sensor Temperatur vs. Humidty

```
fig, ax = plt.subplots(1, 1, figsize=(18, 9))

for i, (device, group) in enumerate(groups):
    group['mean'] = group['temp'].rolling(window=20).mean()
    ax.plot(group['mean'],
            label=device,
            color=colors[i])  # Assign a color to each device

fig.autofmt_xdate()
ax.grid()
ax.margins(0.05)
ax.legend()
plt.title('Temperature Comparison over Time')
plt.ylabel('Temperature (°F)')
plt.xlabel('Time')
plt.show()
```

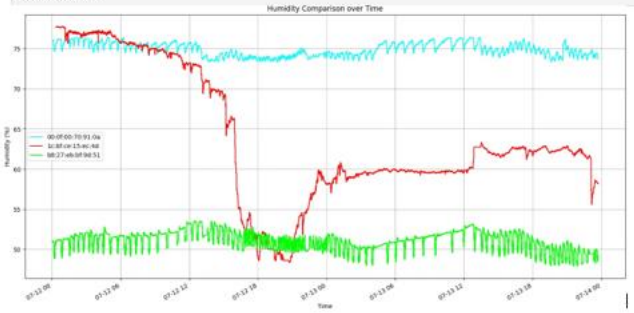


Gambar 18. Grafik Sensor Temperatur Comparison over Time

```
fig, ax = plt.subplots(1, 1, figsize=(18, 9))

for i, (device, group) in enumerate(groups):
    group['mean'] = group['humidity'].rolling(window=20).mean()
    ax.plot(group['mean'],
            label=device,
            color=colors[i])  # Assign a color to each device

fig.autofmt_xdate()
ax.grid()
ax.margins(0.05)
ax.legend()
plt.title('Humidity Comparison over Time')
plt.ylabel('Humidity (%)')
plt.xlabel('Time')
plt.show()
```



Gambar 19. Grafik Sensor Humidty Comparison over Time

Statistik ringkasan dari dataset ini dicetak, termasuk jumlah rekaman, ukuran DataFrame, dan rentang waktu, suhu, serta kelembapan. Selanjutnya, kode menghasilkan visualisasi data dengan tiga jenis plot:

- Scatter Plot: Plot ini memvisualisasikan hubungan antara suhu dan kelembapan untuk setiap perangkat IoT. Setiap perangkat IoT ditampilkan dalam plot dengan penanda data yang berbeda.
- Temperature Graph (Moving Average): Plot garis ini menunjukkan rata-rata bergerak suhu dari waktu ke waktu untuk setiap perangkat IoT. Rata-rata bergerak dihitung dengan jendela geser berukuran 20.
- Humidity Graph (Moving Average): Sama seperti sebelumnya, namun kali ini plot menunjukkan rata-rata bergerak kelembapan dari waktu ke waktu untuk setiap perangkat IoT.

Kode ini bertujuan untuk mengimpor, membersihkan, menganalisis, dan memvisualisasikan data telemetri IoT dari berbagai perangkat, serta membantu pemahaman tren suhu dan kelembapan seiring waktu. Itu juga mengidentifikasi dan mengatasi outlier dalam dataset tersebut untuk analisis yang lebih akurat.

6. KESIMPULAN

Dalam penelitian ini, metode batch processing digunakan untuk mengelola data sensor IoT secara efisien. Pendekatan ini melibatkan Apache Spark Batch Processing, di mana data dikumpulkan dalam batch besar, kemudian diproses dan dianalisis menggunakan Spark Context. Data yang telah diolah disimpan dalam format Parquet atau database, yang nantinya digunakan untuk membuat laporan, grafik, atau visualisasi guna memahami data sensor IoT dan mendukung pengambilan keputusan yang berdasarkan informasi yang diberikan. Pendekatan ini memungkinkan penelitian untuk mengatasi data besar dengan efisien, menjaga ketersediaan dan skalabilitas, serta menghasilkan wawasan berharga dari data sensor IoT. Program yang dipakai ke dalam hadoop ialah program map reduce menggunakan framework Apache Hadoop. Ini terdapat 3 kelas utama dan menggunakan metode main lalu pekerjaan

tersebut akan dijalankan menggunakan MapReduce dimana dijalankan menggunakan **job.waitForCompletion(true)**. Jika pekerjaan selesai tanpa kesalahan, program keluar dengan status 0; jika ada kesalahan, program keluar dengan status 1. Dimana Program diakhiri dengan menjalankan pekerjaan MapReduce dan menghapus output jika sudah ada.

DAFTAR PUSTAKA

[1] R. P. N. Budiarti, "Klasifikasi Air Sungai Berbasis Kombinasi Teknologi Iot-Big Data Menggunakan SVM," 2016.

[2] S. Benjelloun *et al.*, "Big Data Processing: Batch-based processing and stream-based processing," in *4th International Conference on Intelligent Computing in Data Sciences, ICDS 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020. doi: 10.1109/ICDS50568.2020.9268684.

[3] S. Shahrivari, "Beyond batch processing: Towards real-time and streaming big data," *Computers*, vol. 3, no. 4. MDPI AG, pp. 117–129, Dec. 01, 2014. doi: 10.3390/computers3040117.

[4] K. Husnul, "Analisis Kinerja Algoritma Distributed Support Vector Machine Menggunakan Spark Untuk Memprediksi Penundaan Penerbangan," 2023.

[5] M. Guller, *Big data analytics with Spark : a practitioner's guide to using Spark for large-scale data processing, machine learning, and graph analytics, and high-velocity data stream processing*.

[6] M. Khairul, R. Muhammad, S. A. Ismail, and M. N. Kama, "Data Analysis Using MapReduce in Hadoop Environment," *Open International Journal of Informatics (OIJI)*, vol. 5, no. 1, pp. 12–22, 2017.

[7] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money, "Big data: Issues and challenges moving forward," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2013, pp. 995–1004. doi: 10.1109/HICSS.2013.645.

[8] H. Barham, "Achieving Competitive Advantage Through Big Data: A Literature Review," 2017.

[9] M. Parashar, Jaypee Institute of Information Technology University, University of Florida. College of Engineering, Institute of Electrical and Electronics Engineers. Delhi Section, and Institute of Electrical and Electronics Engineers, "2013 sixth International Conference on Contemporary Computing (IC3-2013) : 8-10 August 2013, Jaypee Institute of Information Technology, Noida, India," 2013.

[10] S. Oliviandi, A. B. Osmond, and R. Latuconsina, "Implementasi Apache Spark Pada Big Data Berbasis Hadoop Distributed File System," *e-Proceeding of Engineering*, vol. 5, no. 1 Maret, pp. 1005–1012, 2018.

[11] M. B. Masadeh, M. S. Azmi, and S. S. S. Ahmad, "Available techniques in hadoop small file issue," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 2, pp. 2097–2101, 2020, doi: 10.11591/ijece.v10i2.pp2097-2101.

[12] IEEE Staff, "2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)," 2019.

[13] J. Zou, A. Iyengar, and C. Jermaine, "Architecture of a distributed storage that combines file system, memory and computation in a single layer," *VLDB Journal*, vol. 29, no. 5, pp. 1049–1073, Sep. 2020, doi: 10.1007/s00778-020-00605-w.

[14] G. Fortino, M. IEEE Systems, modellistica Università degli studi della Calabria. Dipartimento di ingegneria informatica, Université de technologie de Compiègne, and Institute of Electrical and Electronics Engineers, "Proceedings of the 2015 IEEE 19th

International Conference on Computer Supported Cooperative Work in Design (CSCWD) : May 6-8, 2015, Calabria, Italy".

[15] Ge, Y. et al. 2016. Adaptive Analytics Service for Real-Time Internet of Things Applications. s.l., IEEE.

[16] Maas, G. & Garillot, F. 2019. Stream Processing with Apache Spark. Sebastopol: O'Reilly Media Inc.

[17] Yasumoto, K., Yamaguchi, H. & Shigeno, H. 2016. Survey of Real-time Processing Technologies of IoT Data Streams. Journal of Information Processing, Volume 24, pp. 195-202

[18] The "Only" Coke Machine on the Internet," Carnegie Mellon University, School of Computer Science

[19] Jason Pontin, "Bill Joy's Six Webs," MIT Technology Review, 29 September 2005

[20] Kevin Ashton, "That 'Internet of Things' Thing", RFID Journal, 22 June 2009