Jurnal Elektronik Ilmu Komputer Udayana
Volume 14, No 2. November 2025

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Sistem Kompresi *Data Logging* pada *Gateway* Orange Pi 3 Menggunakan Metode GZIP

I Putu Gede Mahardika Adi Putra[a1], I Ketut Gede Suhartana[a2], I Gede Arta Wibawa[a3]
I Dewa Made Bayu Atmaja Darmawan[a4]

[a]Informatics Department, Faculty of Math and Natural Science, Udayana University
Jalan Raya Kampus Unud, Jimbaran, Kuta Badung, Bali, Indonesia
[1]gedemahardika@student.unud.ac.id
[2]ikg.suhartana@unud.ac.id
[3]gede.arta@unud.ac.id
[4]dewabayu@unud.ac.id

Abstract

Smart farming is a breakthrough developed to increase rice production. Smart farming refers to the use of IoT technology to log environmental condition in agricultural area. Through smart farming, data can be stored in the Orange Pi 3 gateway, then sent directly to the central government. However, smart farming produces increasingly large data over time. Due to time-series nature, 302,4 MB of data from 25 nodes can be generated over 1 week. The increasing data size causes wastage of bandwidth and increases of latency in sending the data logging to the warehouse. System implements GZIP before the logging data dump is sent to the warehouse. The GZIP method is compared with BZIP2 to determine its effectiveness. As the result, GZIP has an average compression ratio of 5.164, lower than BZIP2's 7.426. The average percentage of storage saving of GZIP method is 86,5%, lower than BZIP2's 86.5%. BZIP2 excels in the transfer time parameter. BZIP2 reduces the transfer time to 39.53 seconds. 30.6% lower than the GZIP treatment and 82.8% lower than the uncompressed treatment. Finally, system testing using load testing shows that the built system can handle up to 100 virtual users with HTTP request durations below 1 second. CPU usage is successfully maintained below 60%.

Keywords: *compression, logging data, GZIP, gateway, Orange Pi 3.*

1. Pendahuluan

Smart farming merupakan terobosan yang terus dikembangkan untuk meningkatkan produksi padi. *Smart farming* merujuk pada penggunaan teknologi *internet of things* (IoT) untuk mengumpulkan data cuaca, memonitor pertumbuhan tanaman, pendeteksian awal penyakit tanaman, dan peningkatan produksi tanaman [1]. Kemajuan subak mendorong penerapan teknologi *smart farming*. Tiap subak dapat memantau kondisi sawah anggota-anggotanya lalu mengambil tindakan tertentu [2]. Subak juga dapat mengirim data ke pemerintah pusat, seperti menuju Kementerian Pertanian. Pemerintah pusat pun mengetahui kondisi lapangan secara langsung sehingga bantuan kepada petani lebih tepat sasaran dan tepat guna. Ilustrasi singkat ini menunjukkan *smart farming* dapat menjadi salah satu opsi untuk menentukan kelayakan tanam padi, terutama saat masa cocok tanam dimulai. Namun, *smart farming* dapat menghasilkan data berukuran semakin besar seiring berjalannya waktu. *Smart farming* melalui sensor *node* mendeteksi parameter lingkungan di persawahan secara *real-time* [3]. *Nodes* yang tersebar di petak-petak sawah mengirimkan data ke *gateway* yang sama. *Gateway* yang memiliki penyimpanan terbatas perlu mengirim *data logging* ke *warehouse* untuk *backup* dan pemrosesan lebih lanjut.

Perhitungan sederhana dapat dilakukan untuk mengetahui estimasi ruang yang diperlukan untuk menyimpan *data logging*. *Node* dapat menghasilkan data teks dengan ukuran 100 byte dalam sekali *logging*. Jika *logging* dilakukan setiap 5 detik, maka dalam 1 menit didapatkan *data logging* berukuran 1200 byte (1,2 KB). Ukuran *data logging* meningkat menjadi 72 KB dalam 1 jam. Ukuran ini akan bertambah dalam 1 hari menjadi 1,728 MB. Terus meningkat menjadi 12,096 MB dalam 1 minggu. Ukuran ini masih terlihat kecil jika *data logging* hanya berasal dari satu *node* pemantauan. Berdasarkan wawancara dengan Bapak I Nyoman Ribek, salah satu petani di Desa Nyitdah, Tabanan, beliau mengatakan bahwa 1 subak dapat memiliki hingga 25 petak sawah. Pernyataan ini menunjukkan bahwa 1 *gateway* yang diletakkan di subak akan menangani data hingga 302,4 MB *data logging* selama 1 minggu. Ukuran ini akan terus bertambah seiring berjalannya waktu dan bertambahnya jumlah *nodes* yang ditangani oleh *gateway*.

Metode kompresi data diperlukan untuk menghemat *bandwidth* transfer saat mengirim *data logging* menuju *warehouse*. Kompresi data juga dapat mengurangi latensi dan meningkatkan efisiensi perangkat *gateway* [4]. Metode kompresi *lossless* dipertimbangkan untuk dipakai karena dapat mempertahankan data tanpa terjadi kehilangan data [5]. GZIP merupakan salah satu metode kompresi lossless yang dapat diterapkan. GZIP mengimplementasikan algoritma deflate. Algoritma ini menggabungkan algoritma kompresi *lossless* Lempel-Ziv 77 (LZ77) dan Huffman coding [6]. GZIP merupakan format yang dikembangkan atas GNU dan dibuat khusus untuk sistem operasi Unix-like. Oleh karena itu, GZIP dapat bekerja optimal untuk banyak server yang menjalankan sistem operasi berbasis GNU/Linux [7]. Sifat kompresi GZIP yang hanya mengompresi satu *file* dalam sekali kompresi merupakan keunggulan yang dapat diterapkan untuk mengompresi *data logging* yang telah di-*dump*. Salah satu penelitian yang mengimplementasikan kompresi GZIP untuk transfer file pada web server. Penelitian menunjukkan GZIP dapat melakukan penghematan 30,87% untuk file ukuran kecil (78,11 KB). File dengan ukuran yang lebih besar (18 MB) dapat dihemat sebesar 14,26% oleh kompresi GZIP [8].

Pada penelitian ini, metode kompresi GZIP diimplementasikan pada sistem pengiriman *data logging* dari *gateway* Orange Pi 3 menuju *warehouse*. Sistem yang dibangun tidak hanya melakukan kompresi pada saat pengiriman *dump data logging*, tetapi juga menjalankan mekanisme melegakan ruangan penyimpanan di *gateway*. Mekanisme ini dijalankan dengan menghapus *data logging* yang telah ditransfer ke *warehouse*. Tujuan mekanisme ini adalah agar ruang penyimpanan pada *gateway* dapat menyimpan *data logging* baru hasil pemantauan. Pengembangan sistem yang mendukung kompresi GZIP pada *gateway* Orange Pi 3 dan latensi pengiriman *dump data logging* menjadi fokus utama pada sistem dalam penelitian ini. Agar penelitian berfokus, didefinisikan tujuan penelitian: 1) mengimplementasikan kompresi *dump data logging* pendeteksi kelayakan tanam padi menggunakan metode GZIP; 2) mengidentifikasi efektivitas kompresi GZIP jika dibandingkan dengan metode kmopresi lain seperti BZIP2 untuk mengompresi *dump data logging*; dan 3) mengetahui performa sistem kompresi data *logging pada gateway* Orange Pi 3 dalam pengujian *load testing*.

2. Metode Penelitian

2.1 Data Penelitian

a. Deskripsi Data

Data yang digunakan adalah primer. Tabel 1 menjelaskan detail dari masing-masing atribut. Data ini terdiri atas 10 atribut, yaitu ID, *timestamp*, *gateway_id, node_id, temperature, humidity, gas, soil_ph, soil_moisture*, dan GPS. Data dikumpulkan di salah satu sawah di Subak Nyitdah III, Desa Nyitdah, Kecamatan Kediri, Kabupaten Tabanan, Bali. Periode pengumpulan data dimulai dari tanggal 10 Mei 2025 hingga 17 Mei 2025. *Node* pengumpul *data logging* mengirim data menuju *gateway* untuk disimpan dan digunakan sebagai data pengujian sistem. Total data yang didapatkan adalah sebanyak 49.785 baris dengan ukuran 9,4 MB.

Tabel 1. Detail Atribut *Data Logging*

Nama Atribut	Deskripsi	Satuan	Tipe Data
ID	ID *data logging*	-	*string*
timestamp	*Timestamp data logging*	-	*string*
gateway_id	ID *gateway*	-	*string*
node_id	ID *node*	-	*string*
temperature	Suhu udara	^{o}C	*float*
humidity	Kelembapan udara	%	*float*
gas	Kandungan gas di udara	ppm (%)	*float*
soil_ph	pH tanah	-	*float*
soil_moisture	Kelembapan tanah	%	*float*
GPS	Koordinat *node*	-	*string*

b. Metode Pengumpulan Data

Data dikumpulkan menggunakan *node* IoT pengumpul *data logging* kelayakan tanam padi. *Node* ini berbasis mikrokontroler Arduino Nano yang terhubung ke beberapa sensor pengumpul. Tabel 2 menjelaskan sensor-sensor yang digunakan untuk mengumpulkan data. Sensor-sensor ini adalah DHT11, MQ2, *capacitive soil moisture, soil pH sensor,* dan GY-NEO6MV2. *Data logging* di-*log* setiap 5 detik, kemudian dikirimkan ke *gateway* menggunakan protokol HTTP dengan koneksi yang berasal dari *provider* GSM. *Data cleaning* diterapkan pada *data logging*. Nilai pembacaan *temperature, humidity, gas, soil_ph,* dan *soil_moisture* yang memiliki nilai 0 tidak dimasukkan ke dalam *dataset*. Transformasi juga dilakukan pada *data*

p-ISSN: 2301-5373
e-ISSN: 2654-5101

logging. Transformasi dilakukan dengan menambahkan ID pada masing-masing data. ID digunakan sebagai kunci primer pada data saat dimasukkan ke MySQL. ID yang digunakan berformat *Universally Unique Identifier* (UUID). Format ini digunakan agar kemungkinan bentrok pada saat *restore* dapat dikurangi. Format UUID memastikan tiap *record* memiliki *primary key* yang unik. Selain ID *data logging*, ID *gateway* juga ditambahkan saat data disimpan pada basis data. ID ini ditambahkan agar *log* data mudah dikenali *gateway* sumbernya. *Data cleaning* dan transformasi diterapkan saat *log* diterima oleh *gateway* dan sebelum disimpan ke dalam basis data.

Tabel 2. Detail Sensor Pengumpul *Data Logging*

Sensor	Parameter yang Dikumpulkan
DHT11	suhu dan kelembapan udara
MQ2	kandungan gas di udara
capacitive soil mosture	kelembapan tanah
soil pH sensor	kandungan ph tanah
GY-NEO6MV2	koordinat *nodes*

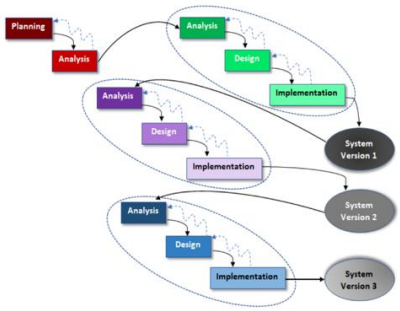
Langkah terakhir adalah menduplikat *dataset* sebanyak 11 kali. Proses ini bertujuan untuk menambah ukuran data yang akan digunakan saat pengujian. Didapatkan 12 *dataset* dengan *id* dan *node_id* yang berbeda. Data ini menyimulasikan 12 *nodes* IoT. Hasil akhir duplikasi didapatkan *dataset* dengan jumlah 597.420 baris. Tabel 3 menunjukkan contoh data yang ada pada *dataset*. Terdapat contoh data dengan atribut ID, *timestamp*, *gateway_id, node_id, temperature, humidity, gas, soil_ph, soil_moisture,* dan GPS.

Tabel 3. Contoh *Dataset* setelah *Cleaning* dan Transformasi

Nama Atribut	Contoh 1	Contoh 2
ID	78f06743-4197-4f07-bb76-0846ba945a04	20b52b06-31d0-4085-aa1e-a06334ac6f4c
timestamp	2025-05-10 06:10:10	2025-05-10 06:10:17
gateway_id	2a7902f1-dfd2-448e-8802-03055f4584ff	2a7902f1-dfd2-448e-8802-03055f4584ff
node_id	5cbb3f1b-4558-4e66-99a9-c92979d3326b	5cbb3f1b-4558-4e66-99a9-c92979d3326b
temperature	34.7	29.5
humidity	95	95
gas	4.82	3.99
soil_ph	31.77	31.28
soil_moisture	11.09	9.11
GPS	-8.588197, 115.102090	-8.588182, 115.102070

2.2 *Software Development Livecycle* (SDLC)

Sistem dikembangkan menggunakan model *rapid application development* (RAD), yaitu *iterative developement*. Grafik pada gambar 1 menunjukkan alur metode *iterative* development. Metode ini memecah sistem menjadi beberapa sub untuk dikembangkan secara berurutan. Kebutuhan mendasar dikerjakan dalam versi pertama. Kebutuhan lain dikerjakan dalam iterasi pengembangan lanjutan. Umpan balik dapat diberikan saat sistem dibangun dengan tujuan menyempurnakan sistem. *Iterative development* menyiratkan bahwa *requirements*, *plan*, *estimates*, dan *solution* dapat disempurnakan selama iterasi. Berbeda dengan *waterfall* yang sepenuhnya ditentukan dan dibekukan sebagai spesifikasi awal sebelum pengembangan [9], [10].



Gambar 1. *Iterative Development* SDLC

2.3 Analisis Kebutuhan Sistem

a. Kebutuhan Fungsional

1. Sistem pada Orange Pi 3 dapat menerima *data logging* dari *nodes* pendeteksi.
2. Sistem pada Orange Pi 3 dapat menyimpan *data logging* sementara dalam waktu yang lebih lama jika koneksi internet terputus.

3. Sistem pada Orange Pi 3 dapat mengompresi *dump data logging* yang telah sebelum dikirim ke *warehouse*.
4. Sistem pada Orange Pi 3 dapat mengirim *dump data logging* yang telah dikompresi menuju *warehouse*.
5. Sistem dapat melakukan mekanisme melegakan ruang penyimpanan *data logging* di *gateway* Orange Pi 3 setelah *dump data logging* terkirim ke *warehouse*.
6. Sistem pada *warehouse* dapat menerima *data logging* yang telah dikompresi dari *gateway*.
7. Sistem pada *warehouse* dapat mendekompresi *data logging* yang dikirim dari *gateway*.
8. Sistem dapat mengirimkan *data logging* antara perangkat yang memiliki perbedaan arsitektur, baik *software* maupun *hardware*.

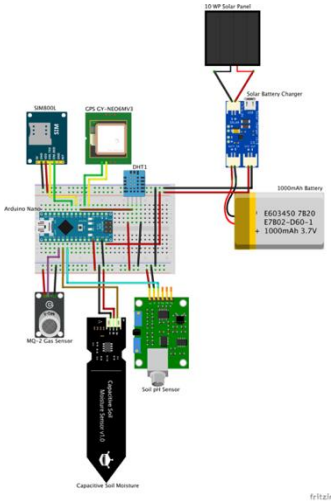
b. Kebutuhan Nonfungsional

1. Waktu respons sistem
   - Sistem pada *gateway* dapat merespons *request* dalam jangka waktu kurang dari 3 detik.
   - Sistem pada *warehouse* dapat merespons *request* dalam jangka waktu kurang dari 3 menit.
2. *Hardware* utama yang digunakan
   - Gateway : Orange Pi 3 LTS
   - Warehouse : *virtual private server* (VPS) dari *cloud provider*
3. Operasional
   - *Gateway* dan *warehouse* berkomunikasi menggunakan media internet yang didapatkan penyedia layanan internet.
   - Status *gateway* dapat dipantau melalui *dashboard* yang disediakan dalam bentuk web.

2.4 Desain Sistem

a. Perancangan Perangkat Keras

Perancangan perangkat keras merujuk pada perancangan *nodes*. Perangkat ini berfungsi untuk melakukan *log data* dari sensor-sensor pendeteksi lingkungan pertanian padi. Gambar 2 menunjukkan rangkaian modul dan sensor yang digunakan untuk membangun *nodes*. Arduino Nano bertindak sebagai mikrokontroler utama untuk *nodes*. Mikrokontroler ini terhubung menuju beberapa sensor, yaitu: 1) DHT11 untuk me-*log* suhu dan kelembapan udara; 2) *capacitive soil moisture* untuk me-*log* kelembapan tanah; 3) *soil* pH sensor untuk me-*log* pH tanah; 4) MQ2 untuk me-*log* kandungan gas di udara; dan 5) GY-NEO6MV2 untuk me-*log* koordinat *nodes*. Mikrokontroler juga terhubung menuju modul SIM800L yang memungkinkan Arduino Nano mendapatkan koneksi internet melalui General Packet Radio Service (GPRS). Modul SIM800L menerima data logging dari Arduino Nano untuk dikirim menuju *gateway*. Terakhir, mikrokontroler mendapatkan *power* dari baterai *rechargeable* sehingga dapat ditempatkan pada persawahan. Panel surya digunakan untuk mengisi ulang baterai agar dapat terus digunakan.

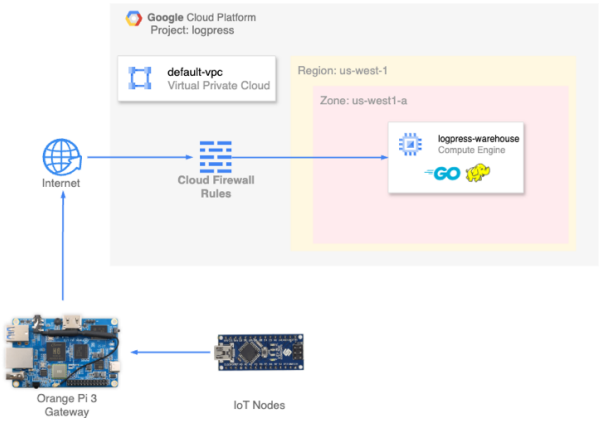


Gambar 2. Perancangan Perangkat Keras *Nodes*

b. Perancangan Arsitektur Sistem

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Gambar 3 menunjukkan arsitektur sistem yang dibangun. *Nodes* menggunakan Arduino Nano sebagai mikrokontroler utama. Beberapa *nodes* dapat mengirim *data logging* sesuai dengan selang waktu yang ditentukan, yaitu setiap 5 detik. Gateway Orange Pi 3 LTS menerima *data logging* dan dikirim ke *warehouse* setiap 10 menit. Namun, jika koneksi internet tidak tersedia, *data logging* akan disimpan dalam waktu yang lebih lama di *gateway*. Sebelum dikirim ke *warehouse*, sistem pada *gateway* melakukan *dump* pada basis data penyimpan *data logging*, lalu dikenakan kompresi GZIP. Pada penelitian ini, Orange Pi 3 LTS memiliki 4 fungsi utama yang saling berkaitan. Fungsi-fungsi tersebut adalah menerima *data logging*, menyimpan *data logging* sementara, mengompresi *data logging*, dan mengirimkan *data logging* yang telah dikompresi menuju *warehouse*.



Gambar 3. Perancangan Arsitektur Sistem

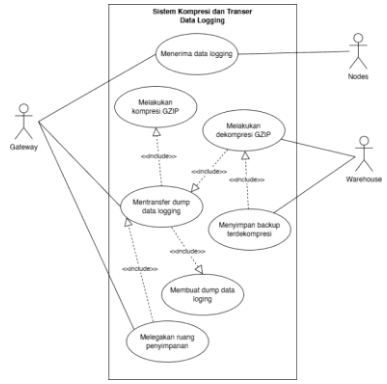
Warehouse dibangun pada lingkungan Google Cloud Platform (GCP). Apache Hadoop dipasang pada *virtual private server* (VPS) yang berperan sebagai *warehouse*. Sistem kompresi *data logging* juga dipasang pada VPS agar dapat menerima *dump data logging* yang dikirim oleh *gateway*. Sistem pada warehouse dapat mendekompresi *dump data logging* terkompresi dan melakukan *merge data logging* pada HDFS. Sistem pada *warehouse* memberikan respons "sukses" untuk *gateway* agar mekanisme melegakan ruangan dapat dijalankan oleh sistem pada *gateway*.

c. Desain Perangkat Lunak

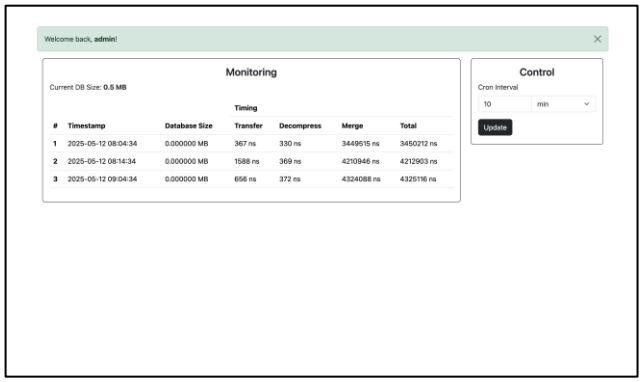
Diagram *Unified Modeling Language* (UML) digunakan untuk memodelkan sistem yang dibangun secara visual. UML dipilih karena sistem yang dibuat adalah sistem berorientasi objek. UML diagram merupakan standar yang digunakan untuk visualisasi, perancangan, dan dokumentasi sistem. Diagram *use case* pada Gambar 4 menggambarkan interaksi antara aktor dengan sistem yang dibuat. Terdapat tiga aktor, yaitu *nodes*, *gateway*, dan *warehouse*. Aktor-aktor ini berinteraksi dengan *use case* yang telah didefinisikan. *Nodes* memiliki satu *use case*, yaitu mengirim data menuju *gateway*. Selanjutnya, *gateway* berinteraksi dengan tiga *use case* secara langsung. *Gateway* dapat menerima *data logging* dari *nodes*, mentransfer *dump data logging*, dan melegakan ruang penyimpanan. Namun, *gateway* harus membuat dan mengompresi GZIP *dump data logging* terlebih dahulu sebelum mentransfer *data logging*. Terakhir, mekanisme melegakan ruang penyimpanan dilakukan saat *dump data logging* telah ditransfer. *Warehouse* berinteraksi dengan satu dua *case*, yaitu melakukan dekompresi GZIP dan menyimpan *data logging* terdekompresi. Namun, sebelum menyimpan *dump data logging*, *dump* yang diterima harus didekompresi terlebih dahulu dan telah dipastikan telah selesai ditransfer dari *gateway*.

d. Rancangan Antarmuka Sistem

Gambar 5 menunjukkan antarmuka berbasis web yang disediakan oleh sistem pada *gateway*. Antarmuka ini dapat digunakan oleh admin untuk mengetahui *log* pengiriman *dump data logging* dari *gateway* menuju *warehouse*. Disediakan juga detail waktu yang diperlukan untuk mentransfer *dump data logging*. Antarmuka ini juga menyediakan fitur kontrol untuk mengubah rentang waktu pengiriman. Admin hanya perlu mengeset rentang waktu dan satuan waktu pengiriman *data logging* dilakukan. Secara *default*, pengiriman dilakukan setiap 10 menit.



Gambar 4. *Use Case Diagram*



Gambar 5. Rancangan Antarmuka Sistem

2.5 Rancangan Implementasi

Tahap implementasi merupakan tahap membangun sistem sesuai dengan perancangan yang telah dilakukan sebelumnya. Kode ditulis menggunakan *text editor* Visual Studio Code. Kode dikelola menggunakan *version control system* (VCS) Git. Pengelolaan kode secara *remote* menggunakan GitHub sebagai *remote repository* untuk Git. *Back end* dibangun dengan bahasa pemrograman Go. Bahasa ini dipilih karena bisa mengimplementasikan fungsi-fungsi tertentu menggunakan b*uilt-in library*. Tipe *library* ini mempermudah proses pengembangan tanpa menggunakan *library* pihak ketiga. Go juga mendukung konsep pemrograman berbasis objek dengan menerapkan *struct*, *interface*, dan *method*. Konsep ini dibangun menggunakan bahasa yang lebih sederhana dibandingkan dengan Java. Go didukung dengan *driver* basis data sehingga koneksi dengan basis data lebih mudah untuk diterapkan.

Pemrograman pada mikrokontroller Arduino Nano menggunakan bahasa pemrograman C++. Bahasa ini merupakan bahasa *default* untuk Arduino. Pemrograman pada Arduino bertujuan agar masing-masing sensor dan modul yang terhubung dapat tersinkronisasi. Komunikasi serial antara Arduino dengan modul SIM800L juga ditangani dengan menggunakan pemrograman C++. Pada tingkat *gateway*, sistem diimplementasikan *back end* menggunakan bahasa pemrograman Go. Sistem berfungsi untuk menerima *data logging* dari *nodes*, memulai *web server*, menangani kompresi GZIP, mengirim ke *warehouse*, dan berkomunikasi dengan basis data MySQL. Terakhir, pada tingkat *warehouse*, sistem diimplementasikan menggunakan bahasa pemrograman Go. Sistem berfungsi untuk menerima *dump data logging* terkompresi GZIP dari *gateway*, menangani dekompresi, dan *merge* data ke klaster Hadoop.

2.6 Desain Evaluasi Sistem

a. Evaluasi Kompresi

Pengujian metode kompresi bertujuan untuk efektivitas metode kompresi GZIP yang diterapkan pada Orange Pi 3. Pengujian ini berdasarkan dua formula, yaitu rasio kompresi dan penghematan ruang. Rasio kompresi adalah perbandingan ukuran data sebelum kompresi dengan ukuran data setelah kompresi. Rasio menggambarkan perhitungan pengurangan ukuran data relatif oleh algoritma kompresi. Selanjutnya, penghematan ruang menentukan pemotongan ukuran data relatif terhadap ukuran data yang tidak terkompresi. Penghematan ruang biasanya dinotasikan dalam bentuk persentase[11]. Formula (1) dan (2) di menjadi tolok ukur efektivitas kompresi. Semakin tinggi rasio kompresi, maka semakin baik algoritma kompresi. Berbanding lurus, persentase penghematan ruang.

$$rasio\ kompresi = \frac{ukuran\ file\ sebelum\ kompresi}{ukuran\ file\ setelah\ kompresi} \quad (1)$$

$$penghematan\ ruang = 1 - \frac{ukuran\ file\ setelah\ kompresi}{ukuran\ file\ sebelum\ kompresi} \quad (2)$$

Metode BZIP2 digunakan untuk membandingkan kompresi GZIP dalam mengompresi *dump data logging*. Masing-masing *dataset*, baik *dataset* awal dan *dataset* duplikasi dikenakan kompresi GZIP dan BZIP2. Rasio kompresi dan penghematan ruang dibandingkan dari kedua metode kompresi. Metode dengan rasio kompresi dan penghematan ruang yang lebih tinggi adalah metode kompresi yang lebih baik. *Dataset* yang digunakan untuk membandingkan kedua metode kompresi adalah *dataset* awal, 6x duplikasi, dan 12x duplikasi. Penggunaan dataset yang terduplikasi memungkinkan metode kompresi dapat dievaluasi dengan lebih detail. Tabel 4 menunjukkan perbedaan ukuran setelah dilakukan duplikasi data.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Tabel 4. Perbedaan Ukuran *Dump* untuk Evaluasi Kompresi

Keterangan	Ukuran	Jumlah Baris
Dataset Awal	9,4 MB	49.785
Duplikasi 6x	56,5 MB	298.710
Duplikasi 12x	113 MB	597.420

Pengujian *transfer time* juga dilakukan untuk mengetahui efektivitas metode kompresi yang digunakan. *Transfer time* dibagi menjadi 3 bagian, yaitu durasi *parsing*, durasi dekompresi, dan durasi *merge*. Durasi *parsing* adalah durasi yang diperlukan oleh sistem pada *warehouse* untuk menerima *dump data logging* dari *gateway*. Durasi dekompresi adalah durasi yang diperlukan untuk mendekompresi *dump data logging* terkompresi menjadi *dump data logging* tidak terkompresi. Terakhir, durasi *merge* adalah durasi yang diperlukan untuk menggabungkan *dump data logging* ke dalam HDFS. *Dataset* awal, 6x duplikasi, dan 12x duplikasi digunakan pada pengujian *transfer time*. Iterasi dilakukan sebanyak 10 kali untuk masing-masing *dataset* untuk masing-masing skenario. Skenario pertama adalah pengiriman tanpa kompresi, skenario kedua adalah pengiriman dengan kompresi GZIP, dan ketiga adalah pengiriman dengan kompresi BZIP2. Rata-rata *transfer time* untuk masing-masing skenario dihitung dari tiap iterasi untuk mengetahui metode kompresi yang lebih baik.

b. *Load Testing*

Load testing digunakan untuk menguji sistem pada *gateway* berdasarkan beban kerja real yang akan dihadapi oleh sistem. Pengujian *load testing* dapat dilakukan dengan bantuan perangkat lunak Grafana k6. Perangkat lunak ini bersifat *open source* sehingga bebas digunakan dan memiliki utilitas yang lengkap untuk melakukan *load testing*. Grafana k6 memiliki beberapa metrik yang menunjukkan kinerja sistem saat dilakukan *load testing*. Terdapat beberapa metrik yang dimunculkan secara *default*, yaitu jumlah data dikirim dan diterima, *response time*, pengiriman data sukses dan gagal, serta banyaknya *virtual users* (VUs). Pada pengujian *load testing*, jumlah VU ditingkatkan dari 25 VU, menuju 50 VU, 75 VU, hingga 100 VU.

3. Hasil dan Diskusi

3.1 Implementasi Kompresi GZIP pada Kompresi *Dump Data Logging*

a. Algoritma Lempel-Ziv 1977 (LZ77)

Algoritma LZ77 pada kompresi GZIP berfungsi untuk mengurangi ukuran data. Mekanisme yang digunakan algoritma ini adalah dengan mengganti urutan karakter berulang dengan referensi ke urutan karakter berulang sebelumnya. Algoritma ini menghapus data duplikat dengan referensi ke lokasi data yang telah ditemukan sebelumnya. LZ77 bekerja dengan menggunakan mekanisme *sliding window* untuk mendeteksi adanya karakter berulang pada data. *Sliding window* terdiri atas 3 bagian, yaitu *coding position, search buffer* dan *lookahead buffer*. *Coding position* menunjukkan posisi karakter yang sedang dikodekan dari masukan. *Search buffer* berisi riwayat karakter yang telah diproses. *Search buffer* memiliki panjang yang tetap. LZ77 mencari kecocokan karakter pada *lookahead buffer* di *search buffer*. Terakhir, *lookahead buffer* berisi kumpulan karakter yang belum diproses (karakter setelah karakter saat ini). LZ77 mencari kecocokan karakter terpanjang yang memungkinkan dari *lookahead buffer*. Keluaran dari algoritma ini adalah *tuple* dengan 3 literal, yaitu (*offset, length,* dan *next character*). Bahkan, *tuple* ini dapat diperpendek menjadi 2 *literal* dengan mengeluarkan *next character*. *Tuple* akhir yang didapatkan adalah (*offset, length*).

b. Algoritma Huffman Coding

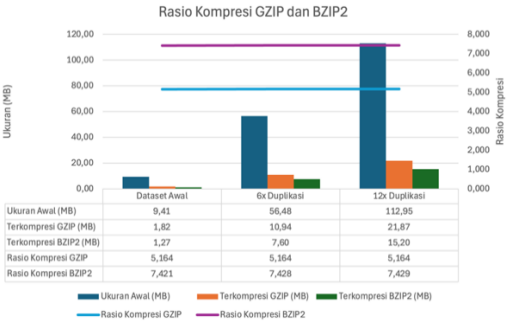
Huffman Coding berfungsi untuk mengompresi data dengan menetapkan kode dengan panjang variabel pada karakter berdasarkan frekuensi kemunculannya. Karakter yang lebih sering muncul mendapatkan kode yang lebih pendek. Sebaliknya, karakter yang jarang muncul mendapatkan kode yang lebih panjang. Pendekatan ini bertujuan untuk mengurangi jumlah bit yang dibutuhkan untuk merepresentasikan data. Hal ini menjadikan Huffman Coding sebagai bentuk kompresi *lossless* yang tidak menghilangkan data. Pembentukan *frequency map* berdasarkan frekuensi kemunculan karakter, pembentukan pohon Huffman, pembentukan *code map*, dan membuat kode Huffman berdasarkan pohon yang telah dibentuk.

3.2 Perbandingan Efektivitas Metode Kompresi GZIP dengan BZIP2

a. Rasio Kompresi dan Persentase Penghematan Ruang Penyimpanan

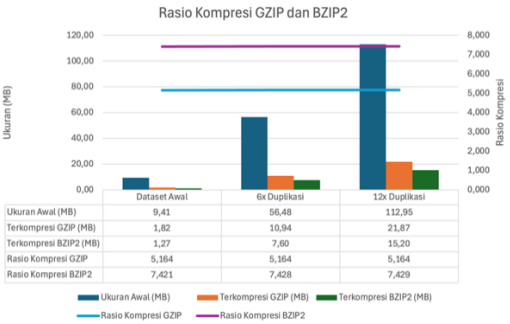
Rasio kompresi menggambarkan pengurangan ukuran data relatif oleh algoritma kompresi. Gambar 6 menunjukkan grafik ukuran awal, ukuran terkompresi GZIP, ukuran terkompresi

BZIP2, dan rasio kompresi masing-masing metode kompresi. *Dataset* awal yang memiliki ukuran 9,41 MB dikompresi menjadi 1,82 MB oleh metode GZIP dan 1,27 MB oleh metode BZIP2. Rasio kompresi yang didapatkan pada *dataset* awal adalah 5,164 untuk metode GZIP dan 7,421 untuk metode BZIP2. *Dataset* dengan 6 kali duplikasi memiliki ukuran 56,48 MB dikompresi menjadi 10,94 MB oleh metode GZIP dan 7,60 MB oleh metode BZIP2. Rasio kompresi untuk *dataset* dengan 6 kali duplikasi adalah 5,164 untuk metode GZIP dan 7,428 untuk metode BZIP2. Terakhir, *dataset* dengan 12 kali duplikasi memiliki ukuran 112,95 MB dikompresi menjadi 21,87 MB oleh metode GZIP dan 15,20 MB oleh metode BZIP2. Rasio kompresi untuk metode GZIP adalah 5,164 dan 7,429 untuk metode BZIP2. Rasio kompresi rata-rata metode GZIP adalah 5,164, lebih rendah daripada metode BZIP2 yang mencapai 7,426.



Gambar 6. Rasio Kompresi Metode GZIP dan BZIP2

Penghematan ruang penyimpanan merupakan pemotongan ukuran data relatif terhadap ukuran data yang tidak terkompresi. Penghematan ruang merujuk pada persentase ruang penyimpanan yang dihemat setelah dilakukan kompresi. Persentase penghematan ruang penyimpanan setelah dilakukan kompresi GZIP dan BZIP2 pada masing-masing *dataset* dapat dilihat pada Gambar 7. Penggunaan penyimpanan untuk *dataset* awal dihemat sebesar 80,6% oleh metode GZIP dan 86,5% oleh metode BZIP2. Penyimpanan untuk *dataset* dengan 6 kali duplikasi dihemat sebesar 80,6% oleh metode GZIP dan 86,5% oleh metode BZIP2. Terakhir, penyimpanan untuk *dataset* dengan 12 kali duplikasi dihemat sebesar 80,6% oleh metode GZIP dan 86,5% oleh metode BZIP2. Metode GZIP menghemat ruang penyimpanan rata-rata hingga 80,6% untuk masing-masing *dataset*. Lebih tinggi dari GZIP, metode BZIP2 dapat menghemat ruang penyimpanan rata-rata hingga 86,5% untuk masing-masing *dataset*.



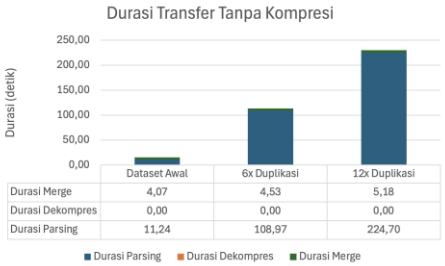
Gambar 7. Rasio Kompresi Metode GZIP dan BZIP2

Berdasarkan parameter rasio kompresi dan penghematan ruang penyimpanan, metode BZIP2 lebih efektif digunakan untuk mengompresi *dump data logging* pendeteksi kelayakan tanam padi. BZIP2 lebih efektif untuk mengurangi ukuran data karena menerapkan lebih dari satu algoritma kompresi, yaitu *Burrows-Wheeler Transform* (BWT), *Move-to-Front* (MTF), dan *Run-Length Encoding* (RLE). BWT menyusun ulang data untuk mengelompokkan karakter yang mirip, hal ini akan meningkatkan kompresibilitas. MTF membuat karakter yang sering digunakan muncul lebih sering dalam suatu ukuran. Terakhir, RLE digunakan untuk mengompresi karakter yang berulang [12]. Berbeda dengan GZIP yang hanya menerapkan algoritma LZ77 untuk melakukan kompresi data. Meskipun algoritma terakhir yang digunakan adalah Huffman Coding, BZIP2 unggul karena menerapkan tiga algoritma berbeda sebelum masuk ke Huffman Coding.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

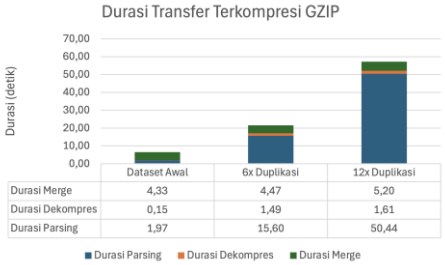
b. Perbedaan *Transfer Time Dump Data Logging* Menuju *Warehouse*

Efektivitas metode GZIP dapat diketahui dengan membandingkan *transfer time dump data logging* terkompresi BZIP2. Pengujian *transfer time* dilakukan dalam 10 kali iterasi untuk tiap perlakuan dan tiap jenis *dataset*. Perlakuan pertama adalah pengiriman tanpa kompresi. *Dump data logging* tidak terkompresi dikirim menuju *warehouse* mulai dari *dataset awal*, 6 kali duplikasi, dan 12 kali duplikasi. Pengiriman dilakukan sebanyak 10 kali iterasi. *Dump data logging* terkompresi GZIP juga dikirim dengan 10 kali iterasi untuk *dataset* awal, 6 kali duplikasi, dan 12 kali duplikasi. Terakhir, *dump data logging* terkompresi BZIP2 juga dikirim dengan 10 kali iterasi untuk tiap-tiap *dataset*. Parameter yang digunakan untuk membandingkan efektivitas metode kompresi dalam pengiriman menuju *warehouse* adalah *transfer time* rata-rata. Nilai ini didapatkan dengan merata-ratakan durasi yang didapatkan pada tiap iterasi.



Gambar 8. Rata-rata *Transfer Time Dump* Tidak Terkompresi

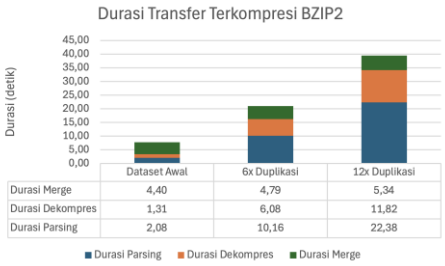
Rata-rata *transfer time* pada tiap iterasi untuk tiap *dataset* dapat dilihat pada Gambar 8. *Dataset* awal berukuran 9,4 MB memiliki *transfer time* rata-rata 15,31 detik. Detail *transfer time*-nya adalah 11,24 detik untuk durasi *parsing* dan 4,07 detik untuk durasi *merge*. *Dataset* dengan 6 kali duplikasi berukuran 56,68 MB memiliki *transfer time* rata-rata sebesar 113,5 detik. Detail *transfer time*-nya adalah 108,97 detik untuk durasi *parsing* dan 4,53 detik untuk durasi *merge*. Terakhir, *dataset* dengan 12 kali duplikasi berukuran 112,95 MB memiliki *transfer time* rata-rata 229,88 detik. Detail *transfer time*-nya adalah 224,7 detik untuk durasi *parsing* dan 5,18 detik untuk durasi *merge*. Hampir 90% *transfer time* rata-rata dipengaruhi oleh durasi *parsing*.



Gambar 9. Rata-rata *Transfer Time Dump* Terkompresi GZIP

Gambar 9 menunjukkan grafik *tranfer time* rata-rata setelah diterapkan kompresi GZIP. Rata-rata *transfer time* menurun dari 15,31 detik menjadi 6,45 detik untuk *dataset* awal. GZIP juga menurunkan rata-rata transfer *time dataset* dengan 6 kali duplikasi dari 113,5 detik menjadi 21,55 detik untuk *dataset* dengan 6 kali duplikasi. Terakhir, GZIP menurunkan rata-rata *transfer time* dari 229,88 detik menjadi 57,26 detik untuk *dataset* dengan 12 kali duplikasi. Kompresi GZIP memberikan penurunan *transfer time* rata-rata mencapai 70%. Penambahan durasi dekompresi juga tidak meningkatkan *tranfer time* secara signifikan karena hanya durasi kompresi tidak mencapai 2 detik.

Gambar 10 menunjukkan grafik *transfer time* rata-rata kompresi BZIP2. *Dataset* awal memiliki *transfer time* rata-rata 7,79 detik, lebih tinggi 17,2% daripada kompresi GZIP. *Dataset* dengan 6 kali duplikasi memiliki *transfer time* rata-rata 21,04 detik, lebih rendah 2,4% daripada kompresi GZIP. Terakhir, *dataset* dengan 12 kali duplikasi memiliki *transfer time* rata-rata 39,53 detik, lebih rendah 30,6% daripada kompresi GZIP.



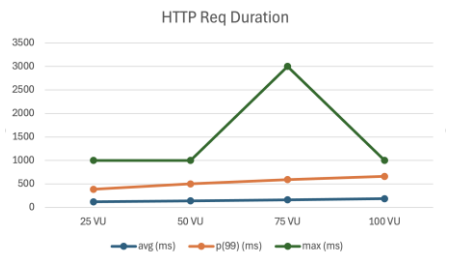
Gambar 10. Rata-rata *Transfer Time Dump* Terkompresi BZIP2

Berdasarkan pengujian *transfer time*, GZIP efektif untuk mentransfer *dump data logging* dengan ukuran yang relatif kecil, yaitu pada ukuran 9,4 MB (sebelum dikompresi). Hal ini dikarenakan GZIP dan BZIP2 memiliki ukuran terkompresi yang mirip, yaitu 1,82 MB untuk GZIP dan 1,27 MB untuk BZIP2. GZIP pun lebih unggul karena memiliki waktu dekompresi yang lebih rendah dengan ukuran file kecil. Sebaliknya, BZIP2 efektif untuk mentransfer *dump data logging* dengan ukuran lebih besar, yaitu 56,48 MB dan 112,95 MB (sebelum kompresi). Pada ukuran file ini, BZIP2 mencapai ukuran terkompresi yang lebih kecil daripada GZIP. Meskipun memberikan durasi dekompresi yang lebih tinggi daripada GZIP, BZIP unggul dalam durasi *parsing*. Hasil akhirnya adalah *transfer time* yang lebih rendah daripada kompresi GZIP.

3.3 Implementasi *Load Testing* Sistem Kompresi Menggunakan Grafana k6

Load testing difokuskan pada *route* /sensors. *Route* ini menangani penerimaan *data logging* dan juga secara otomatis memulai proses kompresi sesuai dengan *threshold* dan selang waktu pengecekan ukuran basis data. Kode testing dibuat menggunakan bahasa pemrograman TypeScript. Tabel 4.7 menunjukkan kode program *load testing* pada *gateway*. Proses *load testing* dimulai dengan inisialisasi. Proses ini memuat *data logging* yang diambil dari dataset. 500 baris data digunakan dalam load testing. *Stage* pada *load testing* didefinisikan agar menyimulasikan jumlah pengguna dan durasinya. Terdapat 3 stage: 1) meningkatkan jumlah pengguna dari 0 hingga N selama 5 menit; 2) mempertahankan N jumlah pengguna selama 30 menit; dan 3) mengurangi jumlah pengguna hingga 0 dalam 5 menit.

Load testing menggunakan Grafana k6 memberikan banyak metrik untuk mengetahui kemampuan sistem pada *load* yang diperkirakan. Beberapa metrik yang dapat digunakan adalah http_req_duration, http_req_sending, http_req_waiting, http_req_receiving, dan http_reqs. http_req_duration menunjukkan waktu total permintaan. Metrik ini adalah jumlah dari http_req_sending, http_req_waiting, dan http_req_receiving. http_req_sending adalah waktu yang diperlukan untuk mengirim data ke *remote host*. http_req_waiting adalah waktu yang diperlukan untuk menunggu respons dari *remote host*. http_req_receiving adalah waktu yang diperlukan untuk menerima respons dari *remote host*. Terakhir, http_reqs menunjukkan jumlah permintaan HTTP yang dibuat oleh k6. Metrik-metrik yang berasal dari k6 dapat dipadukan dengan *load* CPU untuk mengetahui kinerja fisik server.

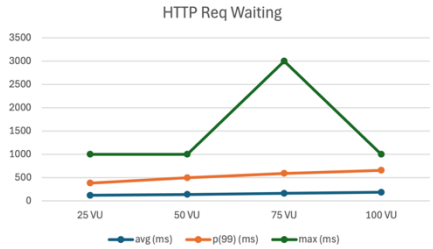


Gambar 11. Grafik Metrik *Load Testing* http_req_duration

Load testing untuk sistem pada *gateway* dilakukan sebanyak 4 kali. Jumlah *virtual user* (VU) ditingkatkan mulai dari 25, 50, 75, dan 100. Metrik http_req_duration yang ditunjukkan oleh grafik pada Gambar 11 menggambarkan durasi permintaan HTTP yang meningkat seiring bertambahnya VU. Rata-rata durasi permintaan HTTP meningkat secara berkala, mulai dari 121 ms untuk 25 VU, 140 ms untuk 50 VU, 163 VU untuk 75 VU, dan 186 ms untuk 100 VU. Meskipun rata-rata durasi permintaan HTTP berada di bawah 200 ms, 99% durasi permintaan berada di atas 300 ms. Mulai dari 386 ms untuk 25 VU, 501 ms untuk 50 VU, 593 ms untuk 75 VU, bahkan 662 ms (mendekati 1 detik)

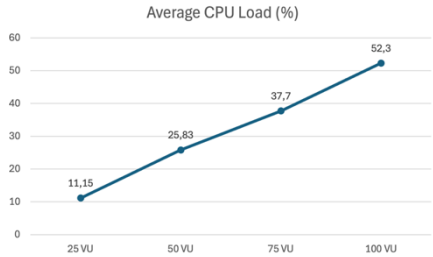
p-ISSN: 2301-5373
e-ISSN: 2654-5101

untuk 100 VU. Durasi ini menunjukkan bahwa total pengguna (*nodes*) yang dapat ditangani agar durasi permintaan tidak menyentuh 1 detik adalah 100 pengguna.



Gambar 12. Grafik Metrik *Load Testing* http_req_waiting.

Metrik req_http_waiting digunakan untuk memastikan nilai-nilai pada metrik http_req_duration muncul karena menunggu respons dari s*gateway*. req_http_waiting mengindikasikan pemrosesan yang dilakukan oleh *gateway* saat menerima *data logging*, baik itu proses *parsing*, transformasi, atau *insert* ke basis data. Gambar 12 menunjukkan metrik http_req_waiting pada tiap-tiap kondisi *load testing*. 99% permintaan pada metrik req_http_waiting untuk tiap kondisi memiliki nilai yang mirip dengan metrik req_http_duration. Kemiripan ini menandakan proses yang terjadi pada sistem *gateway* yang berpengaruh besar pada durasi permintaan, dibandingkan dengan pengiriman data, atau penerimaan respons dari *gateway*.



Gambar 13. Penggunaan CPU Orange Pi 3 saat *Load Testing*

Penggunaan CPU juga dipantau selama melakukan *load testing*. Pemantauan dilakukan saat fase *steady-state*, yaitu saat jumlah VU konsisten. Perintah htop digunakan untuk memonitor penggunaan *resource* Orange Pi 3 LTS. Gambar 13 menunjukkan grafik penggunaan CPU pada masing-masing VU. Penggunaan CPU meningkat seiring meningkatnya VU pada *load testing*. 25 VU meningkatkan penggunaan CPU menjadi 11,15%, 50 VU meningkatkan penggunaan CPU menjadi 25,83%. 75 VU meningkatkan penggunaan CPU menjadi 37,7%. Terakhir, 100 VU meningkatkan penggunaan CPU menjadi 52,3%. Berdasarkan grafik pada Gambar 5, Orange Pi 3 LTS dapat menangani permintaan dari 100 VU tanpa terjadi penurunan performa. Utilitas CPU dipertahankan di bawah 60% agar proses kompresi yang terjadi secara asinkron tidak terganggu.

.

4. Kesimpulan

Berdasarkan penelitian yang telah dilaksanakan, maka didapatkan simpulan sebagai berikut.

a. Kompresi GZIP pada *dump data logging* pendeteksi kelayakan tanam padi diimplementasikan menggunakan bahasa pemrograman Go dengan dua algoritma utama, yaitu LZ77 dan Huffman Coding. LZ77 menggunakan mekanisme *sliding window* yang terdiri atas *coding position, search buffer,* dan *lookahead buffer*. Keluaran algoritma ini adalah *tuple* dengan 2 literal (*offset, length*). Algoritma kedua adalah Huffman Coding. Terdiri atas 4 fungsi, yaitu *buildFrequency* untuk membentuk *frequency map*, *buildHuffman* untuk membentuk pohon Huffman, *generateCodes* untuk membentuk *code map Huffman*, dan *encodeHuffman* berfungsi membentuk hasil akhir proses Huffman Coding. Hasil akhirnya adalah *code map*, dan data yang berbentuk biner.
b. Berdasarkan parameter rasio kompresi dan persentase penghematan ruang penyimpanan, kompresi BZIP2 lebih efektif daripada GZIP. Rasio kompresi rata-rata BZIP2 mencapai 7,426, lebih tinggi daripada GZIP yang hanya 5,164. Persentase penghematan ruang penyimpanan rata-rata BZIP2 mencapai 86,5%, lebih tinggi daripada GZIP yang hanya 80,6%. Berdasarkan pengujian *transfer time*, metode GZIP hanya efektif untuk mentransfer *dump* dengan ukuran relatif kecil. Sebaliknya, BZIP2 efektif untuk mentransfer *dump* dengan ukuran yang lebih besar. BZIP2 dapat menurunkan *transfer time* hingga 39,53 detik, lebih rendah 30,6% dari

GZIP dan 82,8% lebih rendah daripada transfer tidak terkompresi pada skenario *dataset* dengan 12 kali duplikasi.

c. Berdasarkan *load testing* yang dilakukan menggunakan Grafana k6, sistem kompresi data *logging* pada *gateway* Orange Pi 3 dapat menangani hingga 100 *nodes*. Jumlah ini didapatkan dengan pertimbangan agar 99% durasi permintaan HTTP tetap berada di bawah 1 detik. Jumlah 100 *nodes* juga menjaga penggunaan CPU Orange Pi 3 tetap berada di bawah 60% agar CPU tetap tersedia saat proses kompresi berjalan secara asinkron. Hasil *load testing* menunjukkan sistem pada *gateway* dapat menangani lebih dari jumlah rata-rata petak sawah per subak, yaitu 25 petak (*nodes*).

Daftar Pustaka

[1] G. Idoje, T. Dagiuklas, dan M. Iqbal, “Survey for smart farming technologies: Challenges and issues,” *Computers & Electrical Engineering*, vol. 92, hlm. 107104, Jun 2021, doi: 10.1016/j.compeleceng.2021.107104.

[2] I. G. A. A. Suryawati dan I. G. N. N. Santhiarsa, “Literasi Budaya Bali : Kajian Filsafat Ilmu Tentang Keadilan dalam Sistem Subak,” *Jurnal Nomosleca*, vol. 6, no. 1, Apr 2020, doi: 10.26905/nomosleca.v6i1.3960.

[3] R. Singh, A. Gehlot, S. Vaseem Akram, A. Kumar Thakur, D. Buddhi, dan P. Kumar Das, “Forest 4.0: Digitalization of forest using the Internet of Things (IoT),” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, hlm. 5587–5601, Sep 2022, doi: 10.1016/j.jksuci.2021.02.009.

[4] H. Hu, Y. Dong, Y. Jiang, Q. Chen, dan J. Zhang, “On the Age of Information and Energy Efficiency in Cellular IoT Networks With Data Compression,” *IEEE Internet Things J*, vol. 10, no. 6, hlm. 5226–5239, Mar 2023, doi: 10.1109/JIOT.2022.3222343.

[5] S. K. Routray, A. Javali, A. Sahoo, W. Semunigus, dan M. Pappa, “Lossless Compression Techniques for Low Bandwidth IoTs,” dalam *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, IEEE, Okt 2020, hlm. 177–181. doi: 10.1109/I-SMAC49090.2020.9243457.

[6] S. Bharadwaj, “Using Convolutional Neural Networks to Detect Compression Algorithms,” dalam *Proceedings of International Conference on Communication and Computational Technologies*, Springer, Singapore, Nov 2023, hlm. 33–45. doi: 10.1007/978-981-19-3951-8_3.

[7] B. Vijayalakshmi dan N. Sasirekha, “Comparative Analysis of Lossless Text Compression Methods with Novel Tamil Compression Technique,” *International Journal of Research in Engineering and Science (IJRES) ISSN*, vol. 9, no. 7, hlm. 38–44, 2021, Diakses: 23 Mei 2023. [Daring]. Tersedia pada: www.ijres.org

[8] Z. Ali Syed dan T. Rahim Soomro, “Compression Algorithms: Brotli, Gzip and Zopfli Perspective,” *Indian J Sci Technol*, vol. 11, no. 45, hlm. 1–4, Des 2018, doi: 10.17485/ijst/2018/v11i45/117921.

[9] R. S. Bahar, Basuki Wibawa, *Rekayasa Perangkat Lunak: Pendekatan Terstruktur & Berorientasi Objek*. Jakarta: Universitas Negeri Jakarta, 2021.

[10] C. Larman, *Agile & Iterative Development: A Manager’s Guide*. Boston: Pearson Education, Inc, 2004. [Daring]. Tersedia pada: https://books.google.co.id/books?id=76rnV5Exs50C&printsec=frontcover&hl=id&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[11] A. Gopinath dan M. Ravisankar, “Comparison of Lossless Data Compression Techniques,” dalam *2020 International Conference on Inventive Computation Technologies (ICICT)*, Amritapuri: IEEE, Feb 2020, hlm. 628–633. doi: 10.1109/ICICT48043.2020.9112516.

[12] X. Liu *dkk.*, “Refactoring BZIP2 on the new-generation sunway supercomputer,” *Engineering Reports*, vol. 7, no. 1, Okt 2023, doi: 10.1002/eng2.12806.