Analisis Performa Write/Read Kompresi Delta Encoding pada Data Logging Menggunakan Go Benchmark

I Putu Gede Mahardika Adi Putra[a1], I Ketut Gede Suhartana[a2]

[a]Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Udayana
Jalan Raya Kampus Udayana, Bukit Jimbaran, Kuta Selatan, Badung, Bali Indonesia
[1]gedemahardika@student.unud.ac.id
[2]ikg.suhartana@unud.ac.id (Corresponding Author)

Abstract

Compression for data logging in environmental health monitoring is a serious concern. Recording of environmental health history is carried out by monitoring the fluctuations of the physical parameters. The record stored in a warehouse from the logging system will increase over time. So, that's the reason why compression on time-series data logging is needed. But, the addition of compression algorithm like delta encoding allows for increased latency. Therefore, the performance of write/read of delta encoding must be analyzed. One method to analyze is the Go Benchmark. The test parameter used in this paper is the number of iterations and times per iteration taken from the Go Benchmark's output. The other test parameter is compression ratio and storage saving taken from the size of data before and after compression. There are 4 tests case used: single data write, single data read, multiple data write, and multiple data read. As the result, single data write/read and multiple data read work optimally in delta compression with the similar test result with baseline. But multiple data write not working optimally with times per iteration 10,254 times higher than baseline.

Keywords: *Data Compression, Go Benchmark, Delta Encoding, Data Logging, Performance.*

1. Pendahuluan

Kompresi pada data logging pemantauan kesehatan lingkungan menjadi hal krusial yang perlu perhatian serius. Pemantauan kesehatan lingkungan melalui parameter kondisi fisik lingkungan menghasilkan data time-series dalam jumlah yang besar. Parameter fisik yang dapat dijadikan acuan untuk mengetahui kesehatan lingkungan adalah kandungan gas di udara, kelembapan, suhu, dan ada atau tidaknya cahaya. Kandungan gas di udara dapat memberikan informasi keberadaan gas beracun untuk manusia. Kelembapan dan suhu memberikan informasi kecenderungan berkembangnya bakteri [1]. Data hasil pemantauan dari beberapa parameter dapat memberikan informasi yang utuh mengenai kondisi lingkungan terkini. Status lingkungan layak ditinggali, diperlukan perbaikan, atau tidak layak ditinggali dapat diketahui dengan lebih cepat sehingga mendukung keputusan yang akan dilakukan selanjutnya [2]. Selain status kesehatan lingkungan pada satu waktu, pemantauan juga berfungsi untuk mencatat riwayat kesehatan lingkungan. Riwayat ini berfungsi untuk melihat fluktuasi parameter-parameter fisik lingkungan serta hubungan antarparameter yang dipantau. Karena pentingnya fungsi data pemantauan kesehatan lingkungan, maka diperlukan sebuah metode kompresi yang dapat digunakan untuk mereduksi ukuran data time series yang didapatkan dari pemantauan.

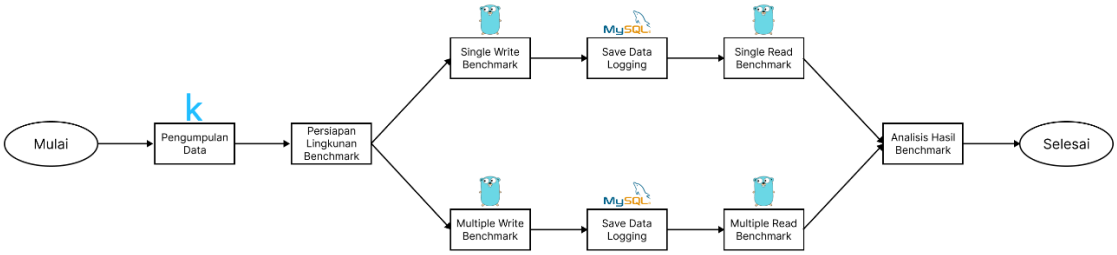
Perangkat Internet of Things (IoT) yang mengirimkan data time series termasuk ke dalam perangkat data logger. Sistem data logger adalah sistem yang dapat memonitor parameter fisik seperti suhu dan radiasi dalam jumlah yang masif [3]. Data logging yang dihasilkan oleh data logger selanjutnya disimpan ke sistem basis data. Data logging digunakan untuk tujuan analisis dan sistem pendukung keputusan dalam proses pemantauan keadaan lingkungan. Data yang disimpan di sistem basis data akan membentuk sebuah gudang data (warehouse) yang menggambarkan keadaan lingkungan dari waktu ke waktu. Semakin lama waktu pemantauan,

semakin banyak data time series yang didapatkan. Besarnya porsi data dan terus bertambahnya ukuran data memicu pembengkakan ukuran data yang disimpan pada basis data.

Metode kompresi dapat digunakan untuk mengatasi pembengkakan ukuran data. Mengecilnya ukuran data memungkinkan pengurangan biaya jaringan dan penyimpanan data di server [4]. Salah satu metode kompresi pada data time series adalah delta encoding. Metode ini termasuk ke dalam lossless compression, yaitu metode kompresi yang menjaga agar tidak ada data yang hilang pada proses dekompresi data yang terkompresi [5]. Namun, penambahan algoritma kompresi terkadang dapat menambah latensi dan penggunaan memori pada server [4]. Oleh karena itu, perlu dianalisis perbedaan performa write dan read data dengan dan tanpa menggunakan delta encoding. Salah satu metode yang dapat digunakan untuk mengetahui perbedaan performanya, terutama pada server adalah dengan menggunakan Go Benchmark.

2. Metode Penelitian

2.1 Alur Penelitian



Gambar 1. Alur Penelitian

Alur penelitian digambarkan pada Gambar 1. Penelitian dimulai dari pengumpulan data pemantauan keadaan lingkungan. Data didapatkan dari penyedia dataset, yaitu Kaggle. Data timestamp dan identitas perangkat pengirim yang tersedia pada dataset dapat digunakan untuk menyimulasikan data yang dikirim oleh beberapa data logger. Sebelum menuju tahap benchmark, disiapkan lingkungan benchmark terlebih dahulu. Lingkungan benchmark yang disiapkan berupa sistem operasi dan text editor atau integrated development environment (IDE) untuk menjalankan benchmark. Selanjutnya, tahap pengujian dilakukan dengan menggunakan Go Benchmark dengan database management system (DBMS) MySQL. Pengujian terjadi dalam dua tahap, yang mana pada masing-masing tahap terjadi tahap write dan read data. Tahap pertama adalah pengujian single write/read data dan tahap kedua adalah pengujian multiple write/read data.

Tahap pertama digunakan untuk mengetahui konsistensi performa write dan read data untuk satu data yang sama. Satu data tersebut secara terus menerus di-write ke basis data berdasarkan fungsi benchmark, baik dengan tambahan proses kompresi maupun tidak. Selanjutnya, proses read merupakan kebalikan proses write-nya. Satu data yang sama, baik data terkompresi maupun tidak, dibaca terus menerus dengan menggunakan fungsi benchmark.

Tahap kedua digunakan untuk mengetahui performa read dan write untuk multiple data, di mana sebanyak 10.000 data ditulis ke basis data menggunakan fungsi benchmark. Proses tulis data ini dilakukan secara terus menerus, baik untuk yang disertai proses kompresi maupun tidak. Selanjutnya, proses read juga dilakukan untuk 10.000 data, baik untuk data terkompresi maupun tidak.

2.2 Pengumpulan Data

Data logging pemantauan keadaan lingkungan yang digunakan pada penelitian ini adalah data sekunder yang didapatkan dari Kaggle. Sebagai komunitas, Kaggle mengizinkan penggunanya untuk mendapatkan dataset yang telah dibagikan oleh pengguna lain [6]. Parameter-parameter fisik lingkungan yang tersedia pada dataset adalah suhu, kelembapan udara, kandungan gas

karbon monoksida dan liquid petroleum gas (LPG) di udara, kandungan asap di udara, dan ketersediaan cahaya. Data logging ini dikumpulkan selama satu minggu oleh pemilik, mulai dari 12 Juli 2020 pukul 00.00 UTC hingga 19 Juli 2020 pukul 23.59 UTC. Pemantauan dilakukan dengan menggunakan tiga buah perangkat IoT yang sejenis, dilengkapi dengan alamat masing-masing perangkat. Total data logging yang disediakan sebanyak 405.184 data [7].

2.3 Go Benchmark

Bahasa pemrograman Go memiliki fungsionalitas bawaan untuk melakukan benchmark yang berada pada package testing. Benchmark pada Go menggunakan aksioma yang dimulai dengan kata “Benchmark”. Aksioma ini menandakan fungsi yang dibuat adalah sebuah fungsi benchmark yang menerapkan fungsionalitas benchmark. Eksekusi fungsi benchmark menggunakan perintah “go test” dengan penanda ”-bench” [8]. Fungsi benchmark menjalankan kode target sebanyak b.N kali. Selama fungsi benchmark dijalankan, b.N disesuaikan hingga fungsi benchmark berlangsung cukup lama untuk diukur waktunya. Secara bawaan, benchmark pada Go dijalankan selama satu detik untuk mendapatkan hasil yang signifikan secara statistik [8], [9]. Benchmark akan dijalankan sampai fungsi menghasilkan return, fail, atau skip.

2.4 Delta Encoding

Delta encoding (juga disebut delta compression) merupakan salah satu algoritma yang termasuk ke dalam lossless compression. Algoritma ini dapat digunakan untuk mengurangi biaya penyimpanan dan mengurangi bandwidth untuk mengirimkan data. Ide utama pada algoritma ini adalah untuk meningkatkan kesamaan data sehingga rentang dari dataset akan lebih kecil daripada data asli [10]. Berikut merupakan rumusan delta encoding.

$$\sigma = x_i - x_{i-1} \quad (1)$$

2.5 Perangkat Pengujian

Benchmark proses write/read penggunaan metode kompresi delta encoding dilakukan dengan menggunakan Go Benchmark. Proses pengujian dijalankan dengan metode Go Benchmark pada DBMS MySQL. Pada saat pengujian dilangsungkan, beban kerja perangkat disamakan dengan tidak membuka aplikasi selain Visual Studio Code dan LAMPP yang menyediakan paket MySQL server. Spesifikasi perangkat pengujian dapat dilihat pada Tabel 1 di bawah.

Tabel 1. Spesifikasi Perangkat Pengujian

Bagian	Nama dan Spesifikasi
Sistem Operasi	Kali Linux 2023.1
Arsitektur	64-bit
Prosesor	Intel Core i3-2350M
RAM	8 GB

2.6 Parameter Pengujian

Parameter yang digunakan untuk sebagai pembanding dalam proses benchmark adalah number of iteration dan time per iteration. Number of iteration menunjukkan banyaknya iterasi proses write/read yang dijalankan oleh fungsi benchmark, sedangkan time per iteration menunjukkan waktu yang dibutuhkan untuk melakukan sekali iterasi. Semakin tinggi nilai number of iteration dan semakin rendah nilai time per iteration menunjukkan kasus yang lebih baik. Ukuran data sebelum dan setelah kompresi juga digunakan sebagai parameter pengujian. Parameter ukuran data digunakan untuk mengetahui rasio kompresi dan penghematan ruang yang menunjukkan kinerja algoritma kompresi. Rasio kompresi adalah perbandingan ukuran sebelum kompresi dengan ukuran data setelah kompresi. Rasio ini adalah 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]. Berikut formula rasio kompresi dan penghematan ruang.

$$\text{Compression ratio} = \frac{\text{Uncompressed file size}}{\text{Compressed file size}} \quad (2)$$

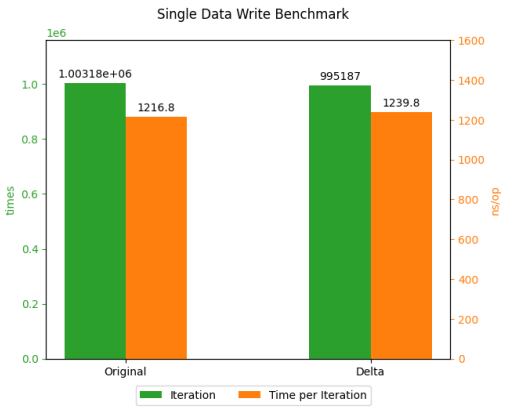
$$\text{Space saving} = 1 - \frac{\text{Compressed file size}}{\text{Uncompressed file size}} \quad (3)$$

3. Hasil dan Pembahasan

```
└─$ go test -bench BenchmarkInsertDeltaData
goos: linux
goarch: amd64
pkg: go-delta
cpu: Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz
BenchmarkInsertDeltaData-4        998538              1208 ns/op
PASS
ok      go-delta        2.041s
```

Gambar 2. Proses Go Benchmark

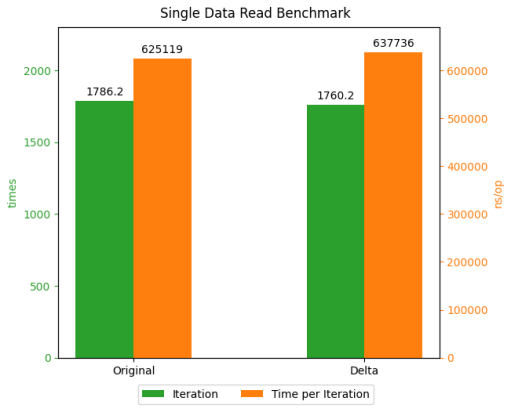
Terdapat beberapa informasi yang didapatkan saat menjalankan Go Benchmark. Pada Gambar 2 terdapat informasi sistem operasi yang digunakan, arsitektur sistem operasi, package yang sedang menjalankan benchmark, prosesor perangkat, fungsi benchmark yang dijalankan, jumlah core prosesor yang digunakan, jumlah iterasi pada proses benchmark, waktu untuk menjalankan satu iterasi, status benchmark “pass” atau “failed”, dan total waktu yang diperlukan untuk menjalankan benchmark. Pengujian menggunakan Go Benchmark dijalankan menggunakan beberapa kasus. Kasus-kasus yang diuji adalah single data write, single data read, multiple data write, dan multiple data read. Setiap kasus benchmark dijalankan sebanyak lima kali untuk mendapatkan nilai benchmark yang lebih presisi dengan menggunakan nilai rata-rata dari kelima pengujian. Setiap kasus juga menerapkan write/read untuk proses tanpa kompresi sebagai baseline dan proses kompresi menggunakan delta encoding.



Gambar 3. Grafik Perbandingan Single Data Write pada Baseline dan Delta Encoding

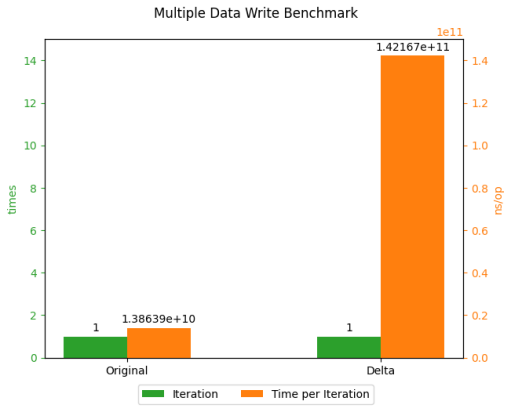
Kasus single data write pada grafik di Gambar 3 menunjukkan proses write data asli (baseline) dan delta encoding memiliki nilai rata-rata iterasi yang tinggi. Saat dijalankan Go Benchmark, rata-rata nilai iterasi baseline adalah 1.003.184 kali dan delta encoding dengan rata-rata nilai iterasi 995.187 kali. Meskipun nilai iterasi pada delta encoding lebih rendah, nilainya hanya terpaut 7.997 kali iterasi. Selanjutnya, pada parameter time per iteration, baseline dapat menyelesaikan satu iterasi dengan waktu 1216,8 nanodetik, sedangkan metode delta encoding memiliki nilai yang lebih tinggi, yaitu 1239,8 nanodetik, terpaut 23 nanodetik. Pada kasus ini, rata-

rata nilai iterasi metode delta encoding saat benchmark adalah 0,992 kali lebih sedikit daripada baseline dan dengan rata-rata waktu per iterasi lebih tinggi 1,019 kali daripada baseline. Perbedaan perbandingan metode delta encoding yang tidak terpaut jauh dari baseline menunjukkan delta encoding efektif diterapkan untuk kasus single data write.



Gambar 4. Grafik Perbandingan Single Data Read pada Baseline dan Delta Encoding

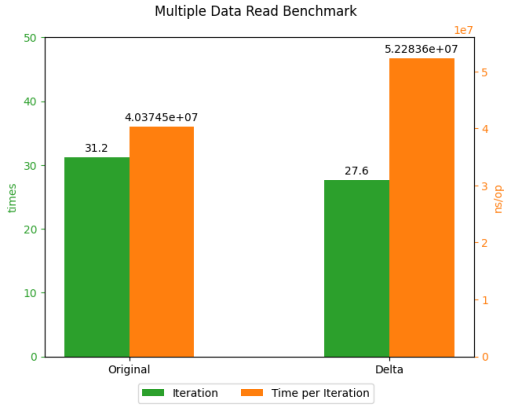
Kasus single data read merupakan kebalikan dari single data write. Satu data, baik data asli maupun yang telah di-delta encoding disimpan ke MySQL lalu dibaca dan di-benchmark. Grafik pada Gambar 4 merupakan hasil benchmark single data read. Pembacaan data asli (baseline) dapat menghasilkan iterasi rata-rata sebanyak 1.786,2 kali, sedangkan pembacaan data yang telah di-delta encoding menghasilkan iterasi rata-rata sebanyak 1.760,2 kali, terpaut 26 kali. Pembacaan data asli memerlukan waktu rata-rata 625.119 nanodetik untuk menjalankan satu kali iterasi, sedangkan pembacaan data yang telah di-delta encoding memerlukan waktu rata-rata 637.736 nanodetik untuk satu kali iterasi. Waktu ini terpaut 12.617 nanodetik. Pada kasus ini, nilai iterasi metode delta encoding saat benchmark adalah 0,985 kali lebih rendah dari baseline. Sedangkan, waktu per iterasi metode delta encoding lebih tinggi 1,02 kali daripada baseline. Perbedaan perbandingan parameter iteration dan time per iteration yang kecil (tidak mencapai 2 kali) menunjukkan proses decode data yang telah di-delta encoding tidak jauh lebih berat daripada membaca data asli yang tidak dikenai proses kompresi.



Gambar 5. Grafik Perbandingan Multiple Data Write pada Baseline dan Delta Encoding

Kasus selanjutnya adalah multiple data write. Pada kasus ini, benchmark dilakukan dengan menuliskan sebanyak 10.000 data, baik data asli (baseline) maupun yang telah di-delta encoding. Grafik pada Gambar 5 menunjukkan proses penulisan 10.000 data ke database merupakan proses yang berat. Benchmark pada baseline maupun delta encoding untuk kasus ini hanya terjadi 1 kali iterasi. Namun, time per operation menunjukkan baseline dan delta encoding memiliki waktu yang terpaut jauh. Baseline memerlukan waktu rata-rata 13.863.918.358,2

nanodetik (±13 detik) untuk satu iterasi, sedangkan delta encoding memerlukan waktu rata-rata 142.167.422.453,4 nanodetik (±142 detik) untuk satu iterasi. Terpaut ±129 detik, di mana delta encoding memerlukan waktu untuk satu iterasi 10,254 kali lebih tinggi daripada baseline. Tingginya waktu untuk satu iterasi ini disebabkan karena proses encode pada data mengharuskan untuk mengambil jumlah nilai delta pada masing-masing field terlebih dahulu dari basis data. Jumlah nilai delta pada data-data sebelumnya digunakan untuk mencari nilai delta untuk data yang akan dimasukkan ke basis data. Perbedaan time per iteration pada proses delta encoding menunjukkan proses kompresi data dalam jumlah yang banyak sekaligus tidak efektif untuk dilakukan dengan menggunakan delta encoding.



Gambar 6. Grafik Perbandingan Multiple Data Read pada Baseline dan Delta Encoding

Kasus multiple data read merupakan proses pembacaan pada data sensor yang telah dilakukan delta encoding. Grafik pada Gambar 6 menggambarkan benchmark untuk multiple data read. Proses pembacaan data asli (baseline) saat di-benchmark dapat dilakukan iterasi rata-rata sebanyak 31,2 kali, sedangkan untuk data yang telah di-delta encoding rata-rata sebanyak 27,6 kali. Proses pembacaan data dengan delta encoding memiliki rata-rata iterasi yang lebih kecil, yaitu lebih sedikit 3,6 kali dibandingkan dengan baseline. Waktu satu iterasi yang diperlukan untuk membaca 10.000 data pada baseline adalah 40.374.522,6 nanodetik (±40 milidetik), sedangkan pada data dengan delta encoding adalah 52.283.589,4 nanodetik (±52 milidetik). Waktu satu iterasi ini terpaut 11.909.066,8 nanodetik (±11 milidetik). Pada kasus ini, nilai iterasi pada metode delta encoding lebih rendah 0,88 kali dibandingkan baseline. Sedangkan waktu rata-rata per iterasi pada metode delta encoding lebih tinggi 1,294 kali dibandingkan baseline. Mirip dengan kasus multiple data write, waktu rata-rata per iterasi yang lebih tinggi pada metode delta encoding disebabkan karena adanya proses decode dengan mencari jumlah data pada setiap nilai delta untuk mendapatkan data asli.

Pada parameter ukuran data, data logging tanpa proses kompresi memiliki ukuran 1.293.040 byte, sedangkan data logging yang telah dikompresi dengan menggunakan delta encoding memiliki ukuran 722.873 byte. Rasio kompresi pada 10.000 data logging sensor adalah sebesar 1,789 (1,789:1) dengan penghematan ruang sebesar 44,096%. Nilai penghematan ruang ini menunjukkan delta encoding dapat digunakan untuk mengompresi data dengan baik, yaitu hampir setengah dari ukuran data sebelum dikompresi. Namun, perlu diperhatikan bahwa waktu yang diperlukan akan semakin bertambah untuk mengompresi data dengan jumlah yang semakin banyak pula.

4. Kesimpulan

Penelitian yang dilakukan bertujuan untuk menganalisis performa write/read algoritma delta encoding terhadap 10.000 data logging sensor pendeteksi kesehatan lingkungan. Berdasarkan parameter pengujian, yaitu number of iteration, time per iteration, rasio kompresi, dan penghematan ruang, delta encoding menunjukkan performa yang optimal, terutama pada kasus single data write, single data read, dan multiple data read. Performa optimal ini ditunjukkan oleh

nilai pada masing-masing parameter pengujian yang tidak terpaut jauh dengan baseline. Namun perlu diperhatikan saat menggunakan delta encoding untuk mengompresi beberapa data dalam jumlah yang besar sekaligus. Waktu yang diperlukan akan jauh lebih tinggi dibandingkan dengan baseline, yaitu mencapai 10,254 kali lebih tinggi. Oleh karena itu, delta encoding akan lebih baik digunakan untuk proses write satu data pemantauan (single data write) sehingga beban kerja server akan lebih kecil jika dibandingkan dengan menuliskan beberapa data pemantauan sekaligus, terlebih lagi data dengan jumlah yang besar. Selanjutnya, jika diperlukan untuk write banyak data pemantauan sekaligus (multiple data write), maka perlu dipertimbangkan untuk membagi data menjadi beberapa klaster untuk mengurangi waktu write data untuk satu kali eksekusi.

Daftar Pustaka

[1] O. R. Pinontoan dan O. J. Sumampouw, Dasar Kesehatan Lingkungan. Sleman: Deepublish Publisher, 2019. Diakses: 29 April 2023. [Daring]. Tersedia pada: https://books.google.co.id/books?hl=en&lr=&id=kl3HDwAAQBAJ&oi=fnd&pg=PR5&dq=parameter+kesehatan+lingkungan&ots=rPxKC0NxhI&sig=Z7B4cieoBi9Vvc4NfktA6fpzjw4&redir_esc=y#v=onepage&q&f=false

[2] R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, dan B. Qureshi, “An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques,” Sensors, vol. 20, no. 21, hal. 6076, Okt 2020, doi: 10.3390/S20216076.

[3] O. A. Ahmad, H. Sayed, K. A. Jalal, D. Y. Mahmood, W. H. Habeeb, dan O. A. Ahmed, “Design and implementation of an indoor solar emulator based low-cost autonomous data logger for PV system monitoring,” Int. J. Power Electron. Drive Syst., vol. 10, no. 3, hal. 1645–1654, 2019, doi: 10.11591/ijpeds.v10.i3.pp1645-1654.

[4] D. Blalock, S. Madden, dan J. Guttag, “Sprintz: Time Series Compression for the Internet,” Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol., vol. 2, no. 3, hal. 1–23, Sep 2018, doi: 10.1145/3264903.

[5] H. Devi Kotha, M. Tummanapally, dan V. K. Upadhyay, “Review on Lossless Compression Techniques,” J. Phys. Conf. Ser., vol. 1228, no. 1, hal. 012007, Mei 2019, doi: 10.1088/1742-6596/1228/1/012007.

[6] M. H. Mahmoud, “What is a Kaggle?,” 1 Juni 2022. https://www.kaggle.com/general/328265 (diakses 1 Mei 2023).

[7] G. A. Stafford, “Environmental Sensor Telemetry Data,” 21 Juli 2020. https://www.kaggle.com/datasets/garystafford/environmental-sensor-data-132k (diakses 1 Mei 2023).

[8] B. Strecansky, Hands-On High Performance with Go. Birmingham: Packt Publishing Ltd., 2020. Diakses: 3 Mei 2023. [Daring]. Tersedia pada: https://books.google.co.id/books?hl=id&lr=&id=5C7ZDwAAQBAJ&oi=fnd&pg=PP1&dq=golang+benchmark&ots=IFrriR9L-z&sig=gJymaEU5rPZ924sGVVNtzn23Mdk&redir_esc=y#v=onepage&q=golang benchmark&f=false

[9] Go, “testing package,” 2 Mei 2023. https://pkg.go.dev/testing (diakses 3 Mei 2023).

[10] A. Saidani, X. Jianwen, dan D. Mansouri, “A Lossless Compression Approach Based on Delta Encoding and T-RLE in WSNs,” Wirel. Commun. Mob. Comput., vol. 2020, hal. 1–10, Sep 2020, doi: 10.1155/2020/8824954.

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

Halaman ini sengaja dibiarkan kosong