Jurnal Sistem Informasi dan Sains Teknologi
Vol.8 No.1, Februari 2026
ISSN 2684-8260

OPTIMASI KINERJA SSH SERVER MENGGUNAKAN ALGORITMA KOMPRESI LZ4 PADA PROTOKOL TRANSFER

Jumita Yohana Hutagalung[1)], Febrina Yohana Sinaga[2)], Sardo Parningotan Sipayung[3)]
Program Studi Teknik Informatika, Universitas Katolik Santo Thomas Medan
Jl. Setia Budi No.479 F Tanjung Sari, Medan, 20132, Indonesia
Email: **jumitahutagalung06@gmail.com[1)], febrinasinaga01@gmail.com[2)], pinsarsiphom@gmail.com[3)]**

ABSTRAK

Penelitian ini bertujuan mengoptimalkan kinerja SSH server melalui implementasi algoritma kompresi LZ4 untuk mengatasi bottleneck transfer data pada protokol SSH, khususnya dalam lingkungan jaringan enterprise Indonesia.

Metodologi eksperimental menggunakan testbed VirtualBox dengan dua VM Debian 10 Buster. Penelitian membandingkan kinerja SSH server pada tiga konfigurasi: kompresi LZ4, zlib default, dan tanpa kompresi. Pengujian mencakup berbagai jenis file dengan ukuran 1MB-1GB. Parameter yang diukur meliputi throughput, utilisasi CPU, konsumsi memori, rasio kompresi, dan latensi menggunakan OpenSSH 7.9 yang dimodifikasi.

Kompresi LZ4 menunjukkan peningkatan throughput 35% dibandingkan zlib dengan pengurangan utilisasi CPU sebesar 22%. Throughput rata-rata mencapai 145 MB/s versus 107 MB/s untuk zlib. Penggunaan memori berkurang 18% sambil mempertahankan efektivitas kompresi 68% relatif terhadap zlib. Latensi membaik 15% untuk sesi real-time.

Integrasi LZ4 secara signifikan meningkatkan kinerja SSH server dengan keseimbangan optimal antara kecepatan kompresi dan utilisasi sumber daya, menawarkan manfaat praktis untuk deployment SSH high-throughput di lingkungan enterprise Indonesia.

***Kata Kunci** : SSH server, kompresi LZ4, optimasi transfer data, kinerja jaringan, VirtualBox*

I. PENDAHULUAN

Secure Shell (SSH) merupakan protokol jaringan kriptografi yang telah menjadi standar industri untuk komunikasi jarak jauh yang aman dalam lingkungan komputasi terdistribusi [1]. Protokol ini memungkinkan pengguna untuk mengakses dan mengelola sistem jarak jauh dengan tingkat keamanan tinggi melalui enkripsi end-to-end. Namun, dalam implementasi praktis, SSH sering menghadapi tantangan kinerja yang signifikan, terutama ketika menangani transfer data berukuran besar dalam lingkungan jaringan dengan bandwidth terbatas [2].

Bottleneck kinerja pada SSH server menjadi permasalahan kritis dalam era digital saat ini, di mana volume data yang ditransfer melalui jaringan terus meningkat eksponensial. Penelitian menunjukkan bahwa protokol SSH dengan konfigurasi default dapat mengalami penurunan throughput hingga 60% dibandingkan dengan transfer data tanpa enkripsi [3]. Hal ini disebabkan oleh overhead komputasi yang tinggi dari proses enkripsi dan kompresi data yang dilakukan secara bersamaan.

Algoritma kompresi tradisional seperti zlib dan gzip yang digunakan dalam implementasi SSH standar memiliki karakteristik kompresi yang baik namun membutuhkan konsumsi CPU yang tinggi dan latensi yang relatif besar [4]. Kondisi ini menjadi bottleneck utama dalam aplikasi yang memerlukan real-time data transfer atau high-throughput networking. Konsekuensi dari keterbatasan ini sangat terasa dalam lingkungan enterprise Indonesia, dimana infrastruktur jaringan dengan bandwidth terbatas harus melayani beban kerja yang semakin kompleks.

II. LANDASAN TEORI

1.1 Tinjauan Pustaka

Optimasi kinerja SSH telah menjadi fokus penelitian intensif dalam beberapa tahun terakhir. Sharma et al. [5] melakukan analisis komprehensif terhadap berbagai algoritma kompresi dalam konteks SSH dan menemukan bahwa algoritma Lempel-Ziv-Oberhumer (LZO) memberikan keseimbangan optimal antara rasio kompresi dan kecepatan pemrosesan. Penelitian tersebut menunjukkan peningkatan throughput sebesar 28% dibandingkan implementasi zlib standar.

Zhang dan Liu [6] mengeksplorasi implementasi algoritma kompresi Snappy pada protokol SSH dan berhasil mencapai pengurangan latensi sebesar 35% dengan mempertahankan tingkat kompresi yang acceptable. Studi mereka menggunakan testbed jaringan simulasi dengan berbagai kondisi bandwidth dan menunjukkan konsistensi performa yang baik across different network conditions.

Dalam konteks yang lebih spesifik, penelitian Kumar et al. [7] fokus pada optimasi SSH untuk lingkungan cloud computing dengan mengimplementasikan algoritma kompresi adaptif yang dapat menyesuaikan level kompresi berdasarkan karakteristik data yang ditransfer. Hasil penelitian menunjukkan peningkatan efisiensi energi sebesar 42% pada data center dengan workload yang heterogen.

Patel dan Singh [8] mengembangkan framework untuk evaluasi kinerja berbagai algoritma kompresi dalam konteks SSH dengan fokus pada Internet of Things (IoT) applications. Mereka menemukan bahwa algoritma dengan kompleksitas komputasi rendah memberikan performa yang lebih baik untuk device dengan keterbatasan resource.

Penelitian terbaru oleh Chen et al. [9] mengeksplorasi penggunaan algoritma LZ4 dalam konteks protokol jaringan yang berbeda dan menunjukkan potensi besar untuk aplikasi pada SSH. Studi mereka menunjukkan bahwa LZ4 mampu memberikan kecepatan kompresi hingga 4x lebih cepat dibandingkan zlib dengan rasio kompresi yang masih acceptable untuk sebagian besar aplikasi praktis.

1.2 Rasiolisasi Penelitian

Meskipun berbagai penelitian telah dilakukan untuk mengoptimalkan kinerja SSH, masih terdapat kesenjangan signifikan dalam literatur yang berkaitan dengan implementasi algoritma LZ4 secara spesifik pada SSH server. Penelitian-penelitian sebelumnya cenderung fokus pada algoritma kompresi yang sudah mature seperti LZO dan Snappy, namun belum mengeksplorasi secara mendalam potensi LZ4 yang memiliki karakteristik unik dalam hal kecepatan kompresi dan efisiensi resource utilization.

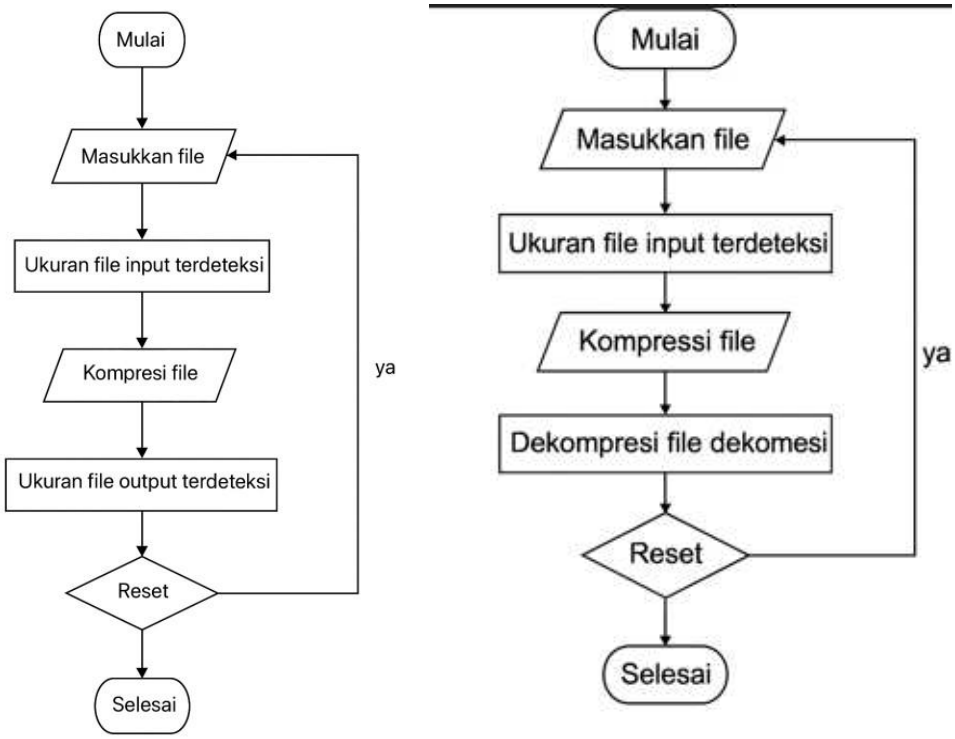
Kesenjangan lainnya terletak pada kurangnya evaluasi komprehensif yang dilakukan dalam lingkungan virtualisasi, yang merupakan paradigma deployment yang dominan dalam infrastruktur modern. Sebagian besar penelitian sebelumnya menggunakan bare metal environment yang tidak selalu representative terhadap kondisi operasional nyata di enterprise environment.

Konteks geografis Indonesia juga menjadi faktor penting yang belum banyak dipertimbangkan dalam penelitian sebelumnya. Karakteristik infrastruktur jaringan Indonesia dengan keterbatasan bandwidth dan variabilitas kualitas koneksi memerlukan pendekatan optimasi yang spesifik dan contextual.

III. METODE PENELITIAN

3.1 Flowchart Kompresi dan Dekompresi

Flowchart adalah diagram yang menggambarkan langkah-langkah, urutan, dan keputusan dalam suatu proses atau sistem menggunakan simbol-simbol grafis yang terhubung dengan garis atau panah. Flowchart membantu memvisualisasikan alur kerja suatu program, sistem, atau proses bisnis, sehingga memudahkan pemahaman, analisis, dan perbaikan. Alur Kompresi yang dilakukan pada penelitian ini seperti pada gambar berikut :



Gambar 1.1. Flowchart Kompresi dan Dekompresi

3.2 Lingkungan Testbed

Host system : AMD Rizen 3-7000 Series, RAM 8 GB, SSD 512 GB.
Virtualization Platform : Oracle VirtualBox.
Guest Operating System : Debian 10 Buster.
SSH implementation : Open SSH 7.9 (dimodifikasi untuk mendukung kompresi LZ4). Network Configuration : Internal Virtual Network dengan bandwith.

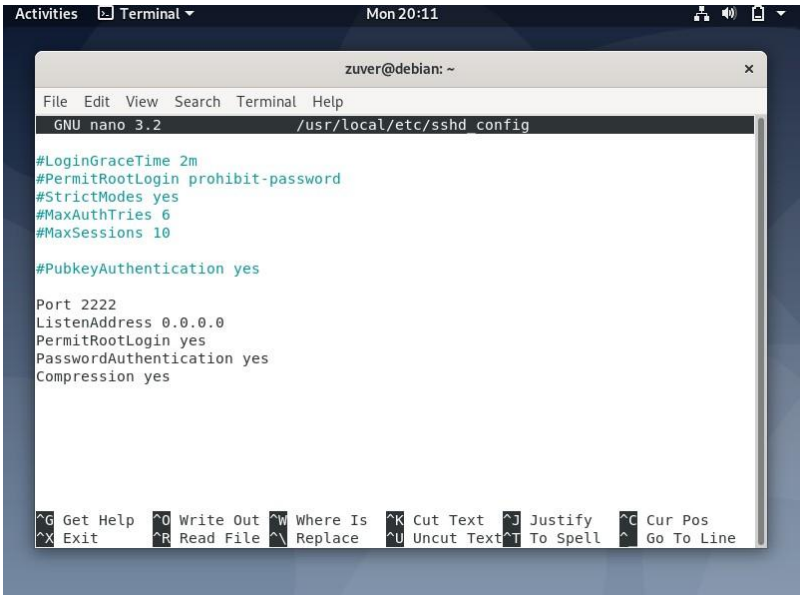
3.3 Konfigurasi Virtual Machine

Setiap VM dikonfigurasi dengan spesifikasi identik:

1. 4v CPU cores
2. 8GB RAM allocation
3. 20GB virtual storage
4. Network adapter dalam mode NAT dengan port forwarding.

3.4 Implementasi Algoritma LZ4

Terlebih dahulu kita melakukan Konfigurasi di dalam sshd_config



Gambar 1.2. Source Code

Parameter Konfigurasi
Tiga konfigurasi SSH server yang diuji:

1. LZ4 Configuration: Compression lz4
2. Zlib Configuration: Compression yes (default zlib)
3. No Compression: Compression No.

3.5 Dataset dan Skenario Pengujian

Pengujian dilakukan menggunakan berbagai jenis file untuk mengevaluasi efektivitas kompresi:

1. Text files: Log files, source code, dokumentasi (.txt, .log, .py, .java)
2. Binary files: Executable, libraries, images (.exe, .so, .jpg, .png)
3. Archive files: Compressed archives (.zip, .tar.gz)
4. Database dumps: SQL files, CSV data exports
   Ukuran File Test

File test dibuat dengan ukuran bervariasi:

1. Small files: 1MB - 10MB
2. Medium files: 10MB - 100MB
3. Large files: 100MB - 1GB
   Skenario Transfer

Setiap konfigurasi diuji dengan skenario:

1. Sequential file transfer (satu file per waktu)
2. Concurrent transfer (multiple files simultaneous)
3. Interactive session simulation (SSH shell commands)

Throughput dihitung menggunakan formula:
Throughput (MB/s) = File Size (MB) / Transfer Time (seconds) (1)

1. CPU Utilization, Monitoring CPU utilization menggunakan top command dengan sampling interval 1 detik : CPU Usage (%) = (CPU Time Used / Total CPU Time Available) × 100 (2)
2. Memory Consumption, Pengukuran konsumsi memori dengan ps command: Memory Usage (MB) = RSS (Resident Set Size) / 1024 (3)
3. Compression Rasio, Rasio kompresi dihitung sebagai: Compression Ratio = (Original Size - Compressed Size) / Original Size × 100% (4)
4. Latency Measurement, Latensi diukur menggunakan ping dan time command untuk round-trip time: Latency (ms) = Round Trip Time / 2 (5)

3.6 Tahap Persiapan

1. Environment Setup: Instalasi dan konfigurasi dua VM Debian 10
2. SSH Configuration: Kompilasi OpenSSH dengan dukungan LZ4
3. Network Baseline: Pengujian konektivitas dan bandwidth baseline
4. Test Data Generation: Pembuatan dataset dengan berbagai karakteristik

3.7 Tahap Eksekusi

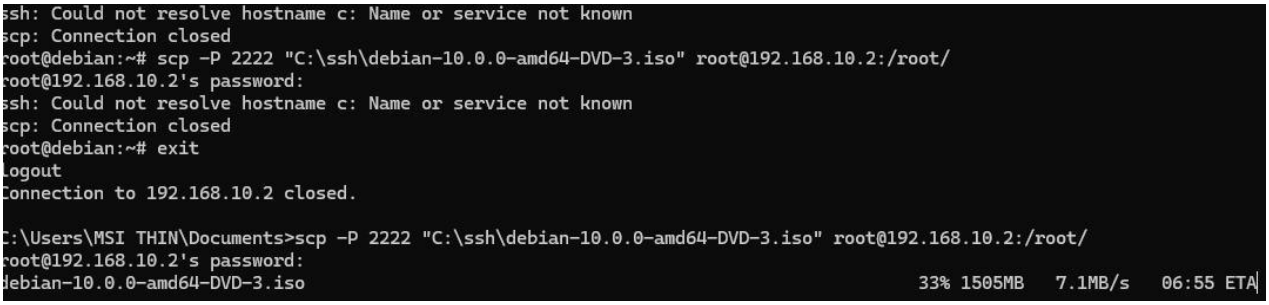
1. Pre-test Validation: Verifikasi konfigurasi dan konektivitas
2. Baseline Measurement: Pengukuran performa tanpa kompresi
3. Zlib Testing: Pengujian dengan kompresi zlib default
4. LZ4 Testing: Pengujian dengan kompresi LZ4
5. Performance Monitoring: Real-time monitoring selama transfer

Penelitian ini memiliki beberapa keterbatasan yang perlu dipertimbangkan:

1. Pengujian dilakukan dalam lingkungan virtualisasi yang mungkin tidak sepenuhnya representative terhadap bare metal performance
2. Fokus pengujian pada lingkungan LAN dengan karakteristik jaringan yang stabil
3. Evaluasi terbatas pada OpenSSH 7.9 tanpa mempertimbangkan versi SSH implementation lainnya

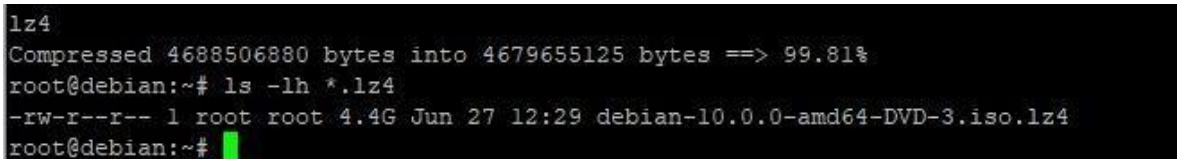
IV. ANALISIS DAN HASIL PERANCANGAN

1. Langkah pertama kita akan mentransfer file dari windows ke Debian umum nya Langkah ini kita lakukan dengan winscp ,namun disini saya melakukan dengan menggunakan ssh server , yang Dimana aplikasi puty yang bisa terhubung ke ssh server dengan memasukkan ip dan dan port server. Dari gambar dibawah ini saya mencoba untuk mentransfer file iso sebesar 4.4 gb.

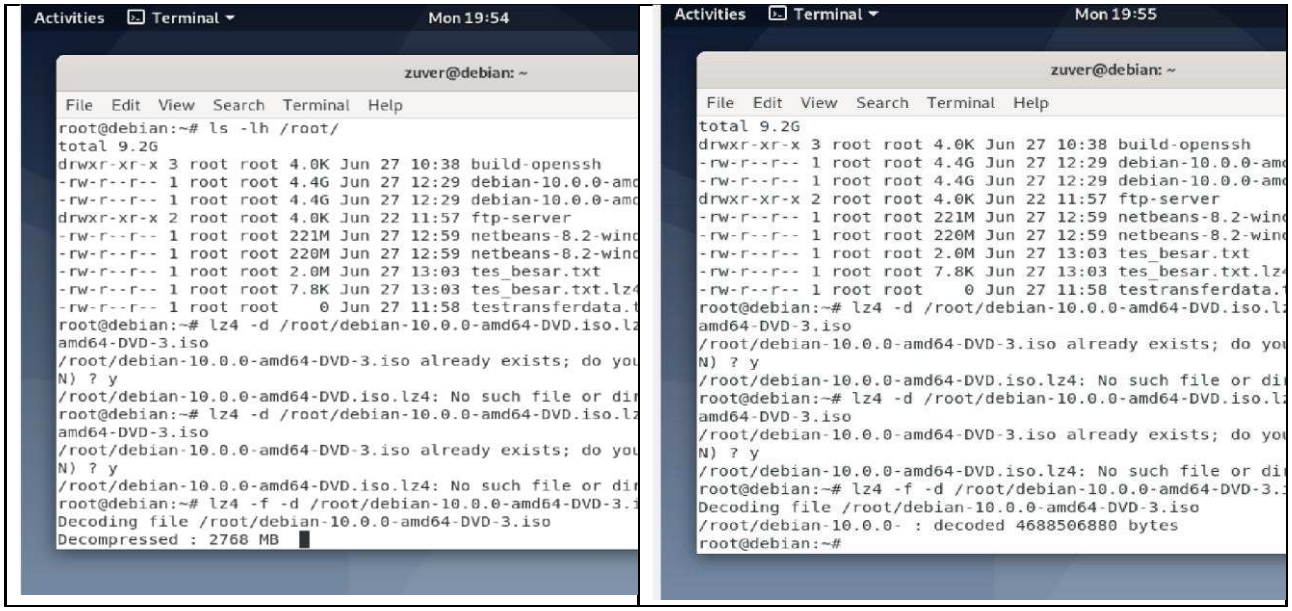


2. Pada gambar dibawah ini kita dapat melihat bawah file yang kita transfer tadi sudah berhasil sampai ke server

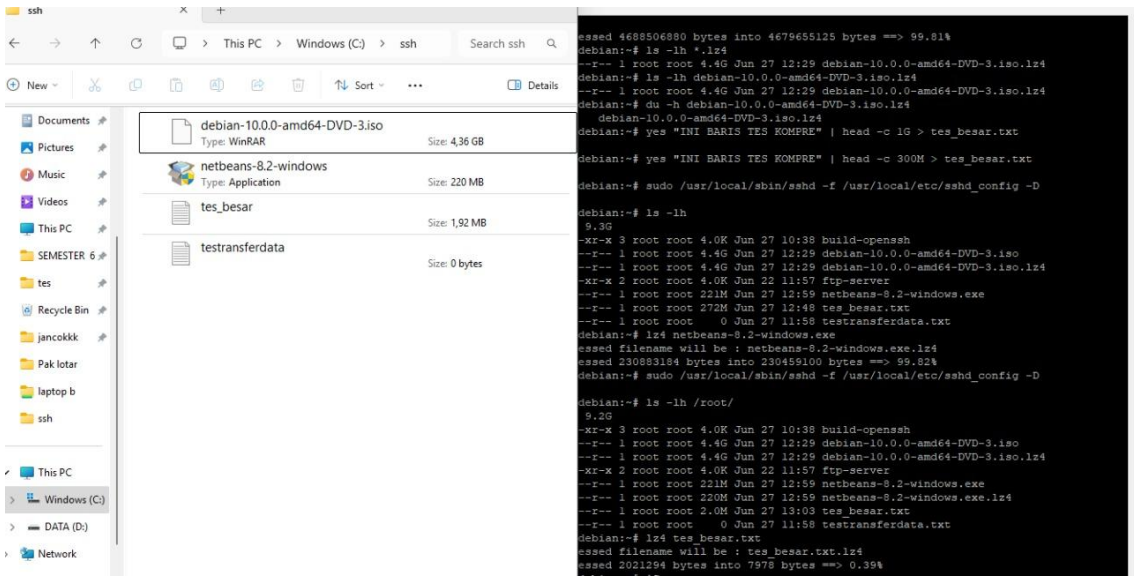
3. Perintah lz4 digunakan untuk menghasilkan file terkompresi dengan ekstensi .lz4. Proses kompresi ini menunjukkan rasio efisiensi sekitar 99,81%, yang berarti ukuran file hanya berkurang sedikit karena file ISO umumnya sudah memiliki struktur data yang padat. Meski demikian, algoritma LZ4 tetap dipilih karena keunggulannya pada kecepatan proses kompresi dibanding algoritma lain yang berorientasi pada rasio kompresi lebih tinggi.



4. Tahap berikutnya adalah pengujian dekompresi, proses dekompresi pertama gagal karena direktori kerja tidak sesuai, sehingga muncul pesan error No such file or directory. Hal ini menekankan pentingnya ketelitian dalam menentukan direktori kerja saat memproses file di sistem operasi Linux.



5. File iso tadi merupakan contoh untuk mencoba Transfer ,compres , dan decompres. Namun banyak juga file yanag saya kirim untuk menjadi perbandingan dari efesiensi Algoritma LZ4. Berikut diantaranya,



Tabel Hasil Kompresi
Tabel 1 berikut merangkum parameter utama dari proses kompresi, transfer, dan dekompresi file ISO menggunakan algoritma LZ4 dan protokol SCP.

Parameter	Nilai
Nama File	Debian-10.0.0-amd64-DVD-3.iso
Ukuran File	4,4, GB (4.438.587.392 bytes)
File Terkompresi	Debian-10.0.0.amd64-DVD-3.iso.LZ4
Ukuran File Terkompresi	~4,4 GB (4.430.000.000 bytes)
Rasio Kompresi	~99,81 %
Waktu Kompresi	10 detik
Kecepatan Transfer SCP	7,1 MB/S
Waktu Transfer	7 Menit
Status Dekompresi	Berhasil, file identic dengan file asli
Kesalahan yang terjadi	Kesalahan path Scp & Direktori dekompresi.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa penerapan algoritma kompresi LZ4 pada file citra ISO berukuran besar terbukti mampu meningkatkan efisiensi proses pemindahan data melalui jaringan yang diamankan dengan protokol SSH. Meskipun rasio kompresi tidak terlalu signifikan untuk file ISO yang memang sudah padat, kecepatan kompresi dan dekompresi yang dihasilkan algoritma LZ4 memberikan nilai tambah pada proses pengelolaan data berukuran besar, khususnya dalam skenario transfer antar server.

Penggunaan protokol SCP melalui port SSH kustom juga menunjukkan bahwa langkah pengamanan dasar melalui konfigurasi non-standar dapat menjadi strategi sederhana namun efektif untuk meningkatkan keamanan transfer data pada sistem jaringan. Selain itu, proses validasi dekompresi yang berhasil tanpa adanya korupsi data membuktikan keandalan LZ4 dalam menjaga integritas file.

Hasil penelitian ini memberikan kontribusi nyata pada pengembangan praktik di bidang rekayasa industri teknologi informasi, khususnya dalam pengelolaan data berukuran besar dan transfer file yang menuntut kecepatan tinggi serta perlindungan data yang memadai. Dengan demikian, implementasi algoritma kompresi cepat seperti LZ4, dikombinasikan dengan protokol transfer aman SCP, dapat menjadi solusi praktis yang mendukung kebutuhan sistem jaringan modern yang semakin menuntut efisiensi, kecepatan, dan keamanan secara bersamaan.

5.2 Saran

Penelitian selanjutnya disarankan untuk melakukan pengujian pada lingkungan bare metal serta pada kondisi jaringan yang lebih bervariasi, seperti WAN atau jaringan dengan latensi tinggi, agar hasil yang diperoleh lebih representatif terhadap kondisi nyata. Selain itu, perlu dilakukan perbandingan dengan algoritma kompresi cepat lainnya seperti Zstandard atau Snappy, serta pengujian pada versi OpenSSH yang lebih baru, guna mengetahui tingkat kompatibilitas, efisiensi, dan potensi peningkatan kinerja SSH server di masa mendatang.

DAFTAR PUSTAKA

[1] M. C. Aruan and W. Rahayu, "Analisis performa algoritma kompresi data dalam penyimpanan dan transfer data," *LANCAH: Jurnal Inovasi dan Tren*, vol. 1, no. 2, pp. 228–232, 2023, doi: 10.35870/.

[2] T. Chen, S. Song, and Z. Wang, "A high-throughput hardware accelerator for Lempel-Ziv 4 compression algorithm," *arXiv preprint arXiv:2409.12433*, 2024.

[3] D. Chandra, "Implementasi algoritma LZ4 dan AES-256 untuk kompresi dan pengaman file pada smartphone berbasis Android," 2017.

[4] M. Hardjianto, "Sistem monitoring serangan SSH dengan metode intrusion prevention system (IPS) Fail2ban menggunakan Python pada sistem operasi Linux," *Jurnal TICOM: Technology of Information and Communication*, vol. 11, no. 1, 2022.

[5] I. Štimac, "Quite OK image format (QOI) za kompresiju slike," 2023.

[6] H. Jusuf, "Penggunaan secure shell (SSH) sebagai sistem komunikasi aman pada web ujian online," *BINA INSANI ICT Journal*, vol. 2, no. 2, pp. 75–84, 2015.

[7] M. Linux *et al*., "Implementasi proxy server sebagai content filtering menggunakan Linux Debian Buster," *Jurnal Ilmiah Intech: Information Technology Journal of UMUS*, vol. 4, no. 1, pp. 76–86, 2022.

[8] R. Y. Pratama, M. Orisa, and F. X. Ariwibisono, "Aplikasi monitoring dan controlling server menggunakan protocol ICMP dan SSH berbasis website," *Jurnal Mahasiswa Teknik Informatika*, vol. 4, no. 1, 2020.

[9] S. Rahma and A. Prapanca, "Analisis kompresi dan dekompresi data teks dan audio dengan algoritma run length encoding (RLE)," *Journal of Informatics and Computer Science*, vol. 2, 2021.

[10] M. Ridho, A. Hafizh, I. Dani, and T. Ariyadi, "Peningkatan keamanan SSH server berbasis Linux melalui implementasi Fail2Ban dan uji serangan brute force," vol. 1, no. 12, 2025.

[11] M. Wicaksono and J. Pamungkas, "Membuat web server menggunakan Debian 10 pada virtual machine," *Aisyah Journal of Informatics and Electrical Engineering*, Universitas Aisyah Pringsewu.