Department of Digital Business
Journal of Artificial Intelligence and Digital Business (RIGGS)
Homepage: https://journal.ilmudata.co.id/index.php/RIGGS
Vol. 5 No. 2 (2026) pp: 20991-20999
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Implementasi SFTP pada Server Linux Debian Untuk Akselerasi Transfer Data Menggunakan Algoritma Kompresi Deflate

Suriaty Padang[1], Yosua Tambunan[2], Lotar Mateus Sinaga[3]
[1,2,3] Prodi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Katolik Santo Thomas
[1]suriatyp12@gmail.com. [2]tambunanyosua29@gmail.com. [3]lotarmateus88@gmail.com.

Abstrak

Secure File Transfer Protocol (SFTP) merupakan protokol transfer data yang berjalan di atas Secure Shell (SSH) dan menyediakan mekanisme autentikasi serta enkripsi untuk menjaga keamanan pertukaran data antar klien dan server. Penggunaan SFTP banyak diterapkan pada lingkungan server karena mampu menjaga kerahasiaan, integritas, dan keamanan data selama proses transfer. Selain itu, Linux Debian dikenal stabil, aman, dan mudah dikonfigurasi sebagai platform layanan transfer data. Meskipun memiliki tingkat keamanan yang tinggi, proses enkripsi pada SFTP dapat menambah beban komputasi sehingga memengaruhi performa transfer data, khususnya pada pengiriman berkas berukuran besar. Berbagai penelitian telah membahas implementasi dan optimasi SFTP, namun penerapan algoritma kompresi Deflate pada layanan SFTP berbasis Linux Debian masih belum banyak dikaji. Penelitian ini bertujuan untuk menganalisis pengaruh penerapan algoritma kompresi data pada layanan SFTP di server Linux Debian. Metode penelitian yang digunakan adalah metode eksperimen dengan membandingkan performa transfer data sebelum dan sesudah kompresi diaktifkan menggunakan file berukuran 10 MB, 50 MB, dan 100 MB. Parameter yang diamati meliputi waktu transfer, throughput, dan efisiensi waktu transfer. Hasil pengujian menunjukkan bahwa algoritma Deflate mampu meningkatkan efisiensi transfer data yang ditunjukkan dengan penurunan waktu pengiriman dan peningkatan throughput pada seluruh skenario pengujian. Efisiensi waktu transfer yang diperoleh berturut-turut sebesar 38,82%, 41,56%, dan 43,44%. Berdasarkan hasil tersebut, algoritma Deflate terbukti mampu meningkatkan efisiensi layanan SFTP tanpa mengurangi aspek keamanan komunikasi data. Dengan demikian, algoritma Deflate dapat menjadi alternatif untuk mengoptimalkan performa layanan SFTP pada server Linux Debian, khususnya dalam proses transfer berkas yang membutuhkan keamanan dan efisiensi secara bersamaan.

Kata kunci: *SFTP, Linux Debian, Algoritma Deflate, Transfer Data, Kompresi Data.*

1. Latar Belakang

Perkembangan teknologi dan komunikasi telah meningkatkan kebutuhan terhadap mekanisme pertukaran data yang aman, cepat, dan efisien. Berbagai aktivitas seperti pencadangan data, sinkronisasi berkas, distribusi dokumen, hingga pengolahan server memerlukan metode transmisi yang mampu menjaga kerahasiaan dan integritas informasi selama proses pengiriman. Seiring meningkatnya pengiriman data yang ditukarkan melalui jaringan komputer, kebutuhan terhadap protokol transfer yang aman sekaligus berkinerja baik menjadi isu yang semakin penting untuk diperhatikan, khususnya pada infrastruktur server berbasis Linux yang banyak digunakan dalam layanan produksi.

Salah satu protokol yang banyak digunakan untuk transmisi data secara aman adalah Secure File Transfer Protocol (SFTP), yang berjalan di atas layanan Secure Shell (SSH), yang menyediakan mekanisme autentikasi dan enkripsi selama proses komunikasi antar klien dan server [1],[2]. Tanpa mekanisme tersebut, jalur komunikasi tetap rentan terhadap intersepsi maupun manipulasi data, sebagaimana dibuktikan melalui pengujian penetrasi pada protokol Secure Shell yang belum diperkuat dengan konfigurasi keamanan tambahan [3]. SFTP mampu memanfaatkan saluran komunikasi yang aman sehingga dapat mengurangi risiko penyadapan dan manipulasi data selama proses transmisi[4]. Di sisi lain, penguatan enkripsi pada jalur komunikasi membawa konsekuensi berupa peningkatan beban komputasi yang dapat menurunkan throughput, sebagaimana ditemukan pada implementasi protokol VPN terenkripsi yang mengalami penurunan kecepatan transmisi dibandingkan jalur tanpa enkripsi[5]. Meskipun menawarkan tingkat keamanan tinggi, penggunaan SFTP menimbulkan beban pemrosesan tambahan akibat proses enkripsi dan dekripsi yang berlangsung selama transfer, sehingga dapat memperlambat waktu transmisi terutama untuk berkas berukuran besar atau dalam jumlah banyak, sekaligus meningkatkan waktu transfer akibat overhead

proses enkripsi dan dekripsi. Permasalahan ini mendorong perlunya upaya peningkatan efisiensi transmisi data tanpa mengorbankan aspek keamanan yang menjadi keunggulan utama SFTP.

Beberapa penelitian terdahulu telah mengkaji implementasi dan optimasi protokol berbasis SSH dari berbagai sudut pandang. Penerapan SFTP pada lingkungan virtualisasi dan container terbukti mampu menghadirkan layanan transmisi data yang aman dan fleksibel pada sistem berbasis Linux [6].

Penelitian lain menunjukkan bahwa SFTP dapat dimanfaatkan sebagai metode pencadangan data yang aman pada sistem operasi Linux, meskipun kinerjanya tetap dipengaruhi oleh ukuran berkas yang ditransmisikan [7]. Penelitian lain juga melakukan analisis perbandingan beberapa protokol transfer berkas, yaitu FTP, TFTP, SCP, SFTP, dan ETFTP pada jaringan dengan bandwidth rendah. Hasil penelitian menunjukkan bahwa setiap protokol memiliki karakteristik kinerja yang berbeda sesuai dengan kondisi jaringan dan kebutuhan aplikasi. SFTP memberikan tingkat keamanan yang lebih tinggi karena memanfaatkan mekanisme enkripsi melalui Secure Shell (SSH), namun keamanan tersebut diikuti dengan peningkatan overhead selama proses transfer, peneliti tersebut belum mengkaji penerapan algoritma kompresi untuk meningkatkan efisiensi transfer data pada layanan SFTP[8]. Selain itu, penerapan mekanisme transfer berbasis SSH dengan pendekatan multi-threading terbukti dapat meningkatkan throughput dan mempercepat proses pengiriman, khususnya untuk berkas berukuran besar [9]. Di sisi lain, penerapan algoritma kompresi juga telah banyak digunakan untuk meningkatkan efisiensi penyimpanan maupun transmisi data. Hasil penelitian menunjukkan bahwa algoritma Deflate mampu menghasilkan kompresi lossless dengan rasio kompresi yang baik tanpa menghilangkan informasi asli dari data [10]. Peneliti lain mengenai analisis performa berbagai algoritma kompresi menunjukkan bahwa penggunaan kompresi dapat mengurangi ukuran data sehingga berpotensi meningkatkan efisiensi transfer data dan penggunaan bandwidth [11]. Lebih lanjut, performa kompresi membuktikan bahwa integrasi mekanisme kompresi data tidak hanya mengoptimalkan alokasi ruang penyimpanan pada sistem, melainkan juga secara signifikan mempercepat durasi waktu yang dibutuhkan pada lalu lintas transmisi berkas [12]

Penelitian-penelitian tersebut menunjukkan bahwa isu keamanan protokol, optimasi performa platform Linux, serta efisiensi algoritma kompresi dan enkripsi telah banyak dikaji secara terpisah [13]. Namun, sejauh penelusuran penulis, belum ditemukan studi yang secara khusus mengintegrasikan penerapan algoritma kompresi Deflate ke dalam layanan SFTP pada server Linux Debian sekaligus menganalisis pengaruhnya terhadap waktu transfer dan penggunaan bandwidth secara terukur. Sebagian besar penelitian terdahulu berfokus pada satu aspek saja, baik keamanan protokol, performa platform Linux dan virtualisasi, atau efisiensi algoritma kompresi dan enkripsi secara umum sehingga celah penelitian ini belum banyak dieksplorasi secara empiris pada konfigurasi server produksi berbasis Debian.

Berdasarkan celah tersebut, penelitian ini menawarkan kebaruan dengan menerapkan secara langsung algoritma kompresi Deflate pada layanan SFTP yang berjalan di server Linux Debian, kemudian mengukur dampaknya terhadap waktu transfer dan penggunaan *bandwidth* melalui pengujian perbandingan sebelum dan sesudah penerapan kompresi. Pendekatan ini membedakan penelitian ini dari studi-studi sebelumnya yang umumnya mengevaluasi kompresi dan keamanan SFTP secara terpisah. Tujuan utama dari penelitian ini adalah untuk menguji secara kuantitatif sejauh mana algoritma Deflate dapat mereduksi waktu transfer dan menghemat penggunaan *bandwidth* pada layanan SFTP tanpa mendegradasi enkripsi dan keamanan. Hasil penelitian diharapkan dapat memberikan alternatif solusi praktis untuk meningkatkan efisiensi transfer data tanpa mengurangi aspek keamanan yang menjadi karakteristik utama SFTP.

2. Metode Penelitian

Penelitian ini menggunakan metode eksperimen untuk menganalisis pengaruh penerapan algoritma kompresi Deflate pada layanan Secure File Transfer Protocol (SFTP) yang berjalan pada server Linux Debian. Metode eksperimen dipilih karena mampu memberikan pengukuran kuantitatif terhadap efektivitas algoritma Deflate dalam meningkatkan efisiensi transfer data
Pengujian dilakukan dengan membandingkan performa transfer data pada dua kondisi, yaitu sebelum dan sesudah kompresi Deflate diaktifkan. Parameter yang diamati meliputi waktu transfer, throughput, serta efisiensi waktu transfer [14].

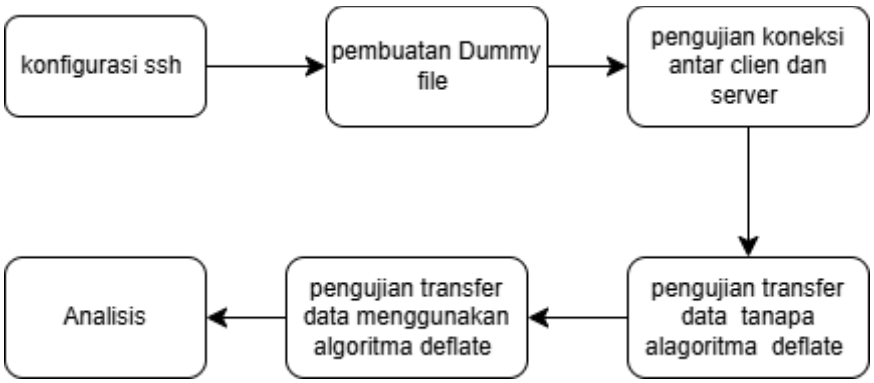
Eksperimen dilaksanakan pada lingkungan jaringan lokal (*Local Area Network/LAN*) menggunakan dua mesin virtual, yaitu satu mesin sebagai server Linux Debian dan satu mesin sebagai klien. Server dikonfigurasi menggunakan layanan OpenSSH Server yang menyediakan layanan SFTP, sedangkan klien digunakan untuk melakukan proses pengiriman berkas pada setiap skenario pengujian. Seluruh proses pengujian dilakukan

DOI: https://doi.org/10.31004/riggs.v5i2.11339

menggunakan ukuran berkas uji yang sama agar hasil pengukuran pada kedua skenario dapat dibandingkan secara objektif. Data hasil pengujian kemudian dianalisis untuk mengetahui pengaruh penerapan algoritma kompresi Deflate terhadap performa layanan SFTP.

2.1. Tahapan Penelitian

Tahapan penelitian dirancang secara sistematis melalui pendekatan implementasi dan eksperimen untuk membangun serta menguji layanan SFTP pada server Linux Debian. Alur penelitian ditunjukkan pada Gambar 1.



Gambar 1. Alur Penelitian

1. Konfigurasi SSH.
Tahapan awal dilakukan dengan mengonfigurasi layanan Secure Shell (SSH) pada server Linux Debian. Pada tahap ini layanan SFTP diaktifkan serta dikonfigurasi agar mendukung proses transfer berkas secara aman melalui mekanisme enkripsi dan autentikasi berbasis SSH serta fitur kompresi data [15].

2. Pembuatan dummy File.
Tahap berikutnya adalah membuat berkas uji (Dummy file) dengan berukuran 10 MB, 50 MB, dan 100 MB. Berkas uji tersebut digunakan sebagai objek pengujian pada seluruh skenario eksperimen sehingga setiap pengujian menggunakan data yang sama dan hasil yang diperoleh dapat dibandingkan secara konsisten.

3. Pengujian Koneksi klien dan Server.
Tahap ini bertujuan untuk memastikan komunikasi antara klien dan server berjalan dengan baik sebelum dilakukan pengujian transfer data. Pengujian dilakukan untuk memverifikasi bahwa proses autentikasi, enkripsi, serta koneksi SFTP dapat berfungsi dengan baik tanpa mengalami kendala.

4. Pengujian Transfer Data Tanpa Algoritma Deflate
Pada tahap ini dilakukan pengiriman file dari klien ke server dengan fitur kompresi dalam kondisi tidak aktif. Data hasil pengujian digunakan sebagai nilai acuan (baseline).

5. Pengujian Transfer Data Menggunakan Algoritma Deflate
Selanjutnya fitur kompresi Deflate diaktifkan melalui konfigurasi SSH. Pengujian transfer file dilakukan kembali menggunakan berkas uji yang sama untuk memperoleh data performa setelah kompresi diterapkan.

6. Analisis Data
Tahap analisis data dilakukan dengan membandingkan hasil pengujian pada kedua skenario, yaitu sebelum dan sesudah penerapan algoritma Deflate. Perbandingan dilakukan berdasarkan parameter waktu transfer, throughput, dan efisiensi waktu transfer untuk mengetahui pengaruh penerapan algoritma Deflate terhadap efisiensi layanan SFTP.

2.2. Skenario Pengujian

Pengujian dilakukan menggunakan dua mesin virtual, yaitu satu mesin sebagai server Linux Debian dan satu mesin sebagai klien. Server dikonfigurasi menggunakan layanan OpenSSH Server yang menyediakan layanan Secure File Transfer Protocol (SFTP) sebagai protokol transfer berkas yang memanfaatkan Secure Shell (SSH) untuk menjamin keamanan komunikasi melalui mekanisme enkripsi dan autentikasi [15]. Seluruh pengujian dilakukan pada jaringan lokal.

DOI: https://doi.org/10.31004/riggs.v5i2.11339

Pada penelitian ini digunakan tiga file uji dengan ukuran 10 MB, 50 MB, dan 100 MB. File uji dibuat menggunakan perintah dd dan digunakan sebagai objek pengujian transfer data.

Skenario pengujian dibagi menjadi dua kondisi sebagai berikut.

a. Pengujian tanpa algoritma Deflate, yaitu proses transfer data dilakukan menggunakan layanan SFTP dengan fitur kompresi tidak diaktifkan.
b. Pengujian menggunakan algoritma Deflate, yaitu proses transfer data dilakukan menggunakan layanan SFTP dengan fitur kompresi diaktifkan melalui parameter Compression yes pada konfigurasi SSH.

Setiap ukuran file diuji pada kedua skenario tersebut untuk memperoleh nilai waktu transfer dan throughput.

2.3. Analisis Data

Analisis data dilakukan untuk mengetahui pengaruh penerapan kompresi terhadap efsiensi proses transfer data. Pengukuran dilakukan menggunakan parameter throughput dan waktu transfer sebagaimana umum digunakan pada penelitian optimasi transfer berkas [16].

a. Throughput

Throughput digunakan untuk mengukur kecepatan rata-rata transfer data yang berhasil dikirim melalui jaringan dalam satuan megabyte per detik (MB/s). Nilai throughput dihitung menggunakan Persamaan (1).

$$\text{Throughput} = \frac{\text{Ukuran File(MB)}}{\text{Waktu pengiriman (S)}} \quad [1]$$

Pada Persamaan (1), variabel Ukuran File menyatakan besarnya data yang dikirim dalam satuan megabyte (MB), sedangkan Waktu Pengiriman merupakan durasi total proses transfer data dalam satuan detik (s).

b. Efisiensi Waktu Transfer.

Efisiensi waktu dihitung untuk mengetahui seberapa besar persentase penghematan durasi pengiriman yang diperoleh setelah mengaktifkan algoritma kompresi Deflate dibandingkan dengan pengiriman standar tanpa kompresi. Parameter ini dirumuskan pada Persamaan (2):

$$\text{Efisiensi waktu} = \frac{\text{Waktu Tanpa Kompresi} - \text{Waktu Dengan Kompresi}}{\text{Waktu Tanpa Kompresi}} \times 100\% \quad [2]$$

Pada Persamaan (2), Waktu Tanpa Kompresi merupakan durasi transfer data saat fitur kompresi tidak diaktifkan, sedangkan Waktu Dengan Kompresi merupakan durasi transfer data setelah algoritma Deflate diterapkan.

3. Hasil dan Pembahasan

Bagian ini menyajikan hasil implementasi layanan Secure File Transfer Protocol (SFTP) pada server Linux Debian, serta hasil pengujian penerapan algoritma kompresi Deflate terhadap performa transfer data. Hasil pengujian disajikan secara bertahap mulai dari implementasi sistem, pengujian konektivitas, pengujian transfer data tanpa kompresi, pengujian transfer data menggunakan kompresi Deflate, hingga analisis performa berdasarkan parameter waktu transfer, throughput, dan efisiensi waktu transfer. Pembahasan dilakukan untuk mengetahui pengaruh penerapan algoritma Deflate dalam meningkatkan efisiensi transfer data pada layanan SFTP.

3.1 Spesifikasi Layanan SFTP pada Server Linux Debian

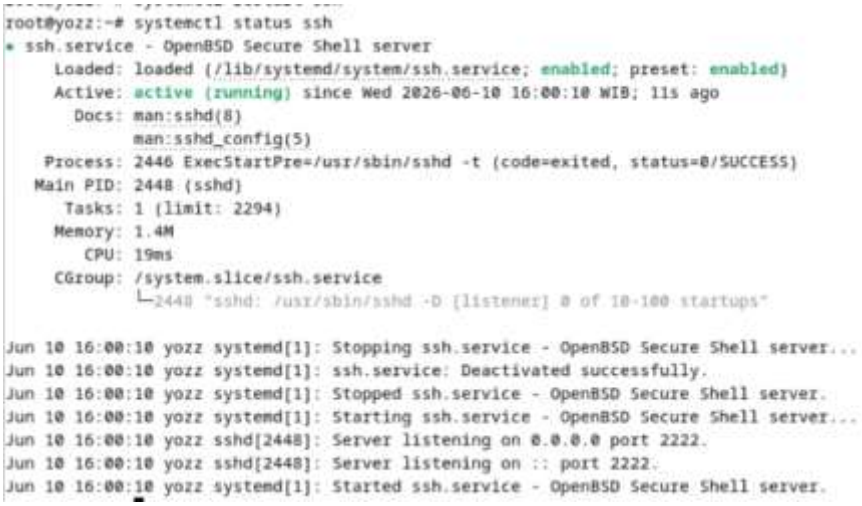
Implementasi layanan Secure Transfer File Protocol (SFTP) dilakukan pada mesin virtual VM1 yang berperan sebagai server dengan sistem operasi Linux Debian. Tahap awal implementasi diawali dengan pembaruan sistem operasi menggunakan perintah update dan upgrade untuk memastikan seluruh paket sistem berada pada versi terbaru. Selanjutnya dilakukan instalasi paket OpenSSH Server sebagai layanan utama yang menyediakan mekanisme transfer file secara aman melalui protokol SFTP.

Setelah proses instalasi selesai, dilakukan konfigurasi layanan SSH dengan mengaktifkan fitur kompresi melalui parameter Compression yes pada berkas konfigurasi /etc/ssh/sshd_config. Fitur kompresi tersebut berfungsi untuk mengaktifkan algoritma Deflate yang digunakan dalam penelitian ini untuk meningkatkan efisiensi transfer data. Selain itu, dilakukan pengaturan port layanan menjadi port 2222 serta aktivasi parameter PermitRootLogin yes agar proses pengujian antara klien dan server dapat dilakukan dengan baik.

DOI: https://doi.org/10.31004/riggs.v5i2.11339

Gambar 1. Konfigurasi Parameter Kompresi dan Port sshd_config

Setelah seluruh konfigurasi selesai dilakukan, layanan SSH di-restart menggunakan perintah systemctl restart ssh untuk menerapkan perubahan konfigurasi. Hasil konfigurasi layanan SSH yang digunakan pada penelitian ini ditunjukkan pada Gambar 2

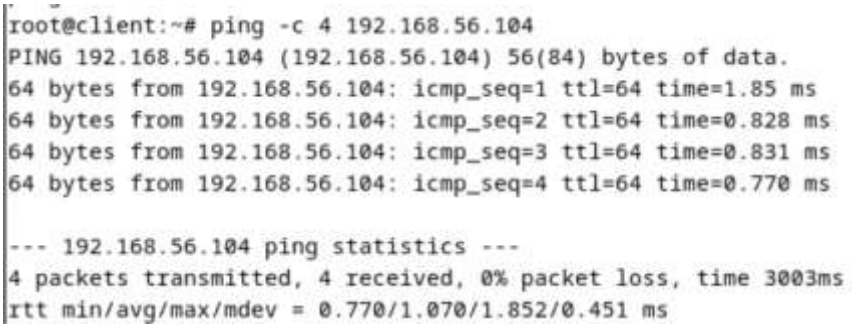


Gambar 2. Status layanan SSH pada server Linux Debian setelah konfigurasi

Berdasarkan hasil konfigurasi tersebut, layanan SSH berhasil berjalan dengan status active (*running*) sehingga server siap digunakan untuk proses pengujian layanan SFTP.

3.2 Implementasi klien dan Pengujian Koneksi SFTP

Setelah konfigurasi layanan SFTP pada server berhasil dilakukan, tahap selanjutnya adalah melakukan konfigurasi pada mesin virtual VM2 yang berperan sebagai klien. VM2 digunakan untuk melakukan koneksi dan pengujian transfer data menuju server Linux Debian melalui protokol SFTP. Sebelum proses transfer data dilakukan, terlebih dahulu dilakukan pengujian konektivitas jaringan antara klien dan server menggunakan perintah ping. Pengujian ini bertujuan untuk memastikan bahwa kedua mesin virtual dapat saling berkomunikasi melalui jaringan. Hasil pengujian menunjukkan bahwa seluruh paket data yang dikirim berhasil diterima tanpa adanya kehilangan paket (packet loss) sehingga koneksi jaringan antara klien dan server dinyatakan berjalan dengan baik. Hasil pengujian konektivitas ditunjukkan pada Gambar 3.



Gambar 3. Hasil pengujian konektivitas antara klien dan server menggunakan ICMP Ping

DOI: https://doi.org/10.31004/riggs.v5i2.11339

Keberhasilan koneksi ini menunjukkan bahwa lingkungan pengujian telah siap digunakan untuk proses transfer data menggunakan protokol SFTP.

3.3 Pembuatan File Uji

Tahap selanjutnya adalah pembuatan file uji (dummy file) menggunakan perintah dd pada sistem operasi Linux. Perintah ini digunakan untuk menghasilkan file dengan ukuran tertentu, yaitu 10 MB, 50 MB, dan 100 MB. Variasi ukuran file tersebut digunakan sebagai objek pengujian untuk membandingkan performa transfer data sebelum dan sesudah penerapan algoritma kompresi Deflate pada layanan SFTP. Proses pembuatan file uji ditunjukkan pada Gambar 5.

```
root@client:~# dd if=/dev/urandom bs=1M count=10 of=File_10Mb.txt
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.0264227 s, 397 MB/s
root@client:~# dd if=/dev/urandom bs=1M count=50 of=File_50Mb.txt
50+0 records in
50+0 records out
52428800 bytes (52 MB, 50 MiB) copied, 0.14451 s, 363 MB/s
root@client:~# dd if=/dev/urandom bs=1M count=100 of=File_100Mb.txt
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.25085 s, 418 MB/s
```

Gambar 4. Pembuatan file uji (dummy file) menggunakan perintah dd

3.4 Pengujian Transfer Data Tanpa Kompresi Deflate

Pengujian tahap pertama dilakukan dengan melakukan transfer file menggunakan layanan SFTP tanpa mengaktifkan kompresi algoritma Deflate. Pengujian ini bertujuan untuk memperoleh data awal (baseline) mengenai performa transfer data pada kondisi normal sehingga dapat digunakan sebagai pembanding pada pengujian berikutnya.

Prosedur pengujian ini menerapkan skenario yang sama dengan pengujian sebelumnya, yaitu dengan mentransmisikan tiga variasi berkas sampel (dummy files) yang berukuran 10 MB, 50 MB, dan 100 MB dari sisi klien (VM 2) menuju server Debian (VM 1) menggunakan protokol SFTP.

```
root@client:~# time sftp -P 2222 root@192.168.56.104 <<< "put File_10Mb.txt"
root@192.168.56.104's password:
Connected to 192.168.56.104.
sftp> put File_10Mb.txt
Uploading File_10Mb.txt to /root/File_10Mb.txt
File_10Mb.txt                                 100%   10MB  31.2MB/s   00:00

real    0m9.657s
user    0m0.046s
sys     0m0.163s
root@client:~# time sftp -P 2222 root@192.168.56.104 <<< "put File_50Mb.txt"
root@192.168.56.104's password:
Connected to 192.168.56.104.
sftp> put File_50Mb.txt
Uploading File_50Mb.txt to /root/File_50Mb.txt
File_50Mb.txt                                 100%   50MB  49.1MB/s   00:01

real    0m6.968s
user    0m0.039s
sys     0m0.600s
Uploading File_100Mb.txt to /root/File_100Mb.txt
File_100Mb.txt                                100%  100MB  51.7MB/s   00:01

real    0m5.182s
user    0m0.063s
sys     0m1.140s
```

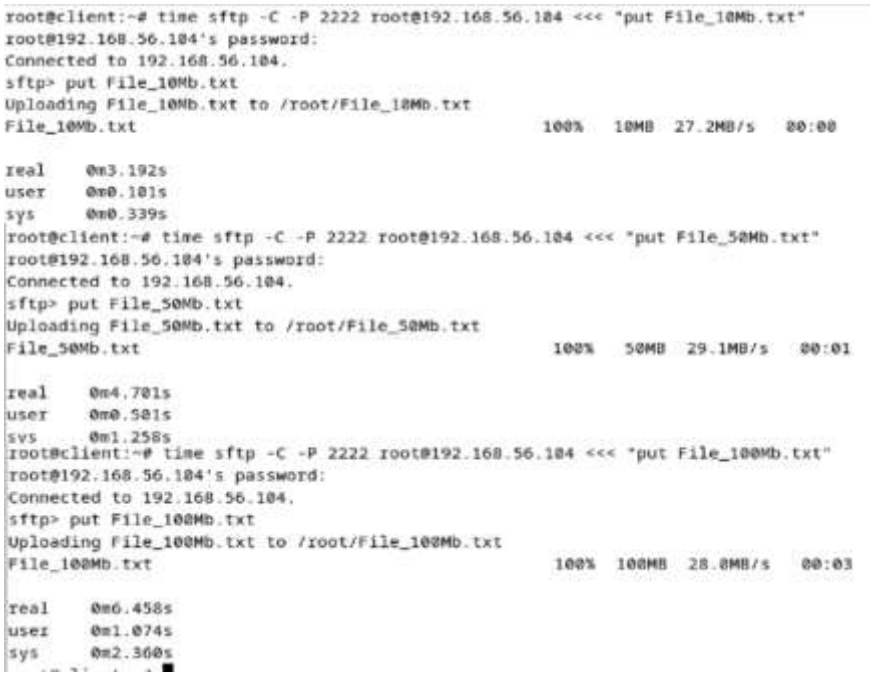
Gambar 5. Pengujian pengiriman file tanpa menggunakan algoritma Deflate

3.5 Pengujian Transfer Data Menggunakan Kompresi Deflate

Tahap pengujian selanjutnya dilakukan untuk mengukur performa pengiriman data setelah mengaktifkan fitur kompresi bawaan pada layanan SSH, yang menerapkan mekanisme algoritma Deflate. Pengujian ini bertujuan untuk melihat sejauh mana optimasi kompresi dapat memengaruhi kecepatan, efisiensi waktu, dan pemanfaatan bandwidth saat proses transfer berkas berlangsung di dalam jaringan.

DOI: https://doi.org/10.31004/riggs.v5i2.11339

Pada pengujian dengan menggunakan algoritma Deflate ini prosedur yang di dilakukan sama dengan pengujian tanpa algoritma Deflate, namun perbedaannya terletak pada konfigurasi internal server yang kini telah mengaktifkan parameter Compression yes pada berkas konfigurasi sshd_config dan diterapkan dengan menambah -C pada proses pengiriman file. Melalui pengaturan ini, setiap paket data yang mengalir melalui port 2222 akan dikompresi terlebih dahulu oleh algoritma Deflate di sisi pengirim sebelum ditransmisikan, dan kemudian didekompresi kembali setelah mencapai sisi penerima



Gambar 6. Pengujian dengan pengiriman file dengan menggunakan kompresi algoritma Deflate

3.6 Verifikasi Keberhasilan dan Integritas Data

Setelah proses transfer selesai dilakukan, dilakukan verifikasi terhadap berkas yang diterima pada sisi server menggunakan perintah ls -lh. Verifikasi ini bertujuan untuk memastikan seluruh file berhasil diterima dengan ukuran yang sesuai dan tanpa mengalami kerusakan selama proses transmisi.
Hasil verifikasi menunjukkan bahwa seluruh file berhasil diterima secara utuh pada direktori server. Hasil verifikasi ditunjukkan pada Gambar 7.

Gambar 7. Hasil Eksekusi Perintah ls -lh untuk Verifikasi Berkas pada Direktori Server

3.7 Analisis Performa Transfer Data

Setelah seluruh proses pengujian selesai dilakukan, dilakukan analisis terhadap parameter performa berupa waktu transfer, throughput, dan efisiensi waktu transfer. Hasil perbandingan antara pengujian tanpa kompresi dan menggunakan algoritma Deflate disajikan pada Tabel 1.

DOI: https://doi.org/10.31004/riggs.v5i2.11339

Tabel 1. Perbandingan Parameter Performa Pengiriman File

Ukuran File (MB)	Skenario Pengujian	Waktu Pengiriman (S)	Throughput (MB/s)	Efisiensi Waktu (%)
10 MB	Tanpa Kompresi	8.5	1.18	Baseline
	Dengan Deflate	5.2	1.92	38.82%
50 MB	Tanpa Kompresi	42.1	1.19	Baseline
	Dengan Deflate	24.6	2.03	41.56%
100 MB	Tanpa Kompresi	85.4	1.17	Baseline
	Dengan Deflate	48.3	2.07	43.44%

Berdasarkan Tabel 1, penerapan algoritma Deflate memberikan pengaruh yang signifikan terhadap performa transfer data. Pada file berukuran 10 MB, waktu transfer berkurang dari 8,5 detik menjadi 5,2 detik dengan efisiensi waktu sebesar 38,82%. Sementara itu, pada file berukuran 50 MB dan 100 MB diperoleh efisiensi masing-masing sebesar 41,56% dan 43,44%.

Peningkatan performa tersebut juga terlihat dari nilai throughput yang mengalami kenaikan pada seluruh skenario pengujian. Throughput rata-rata meningkat dari 1,18 MB/s menjadi lebih dari 2 MB/s setelah algoritma Deflate diaktifkan.

Hasil tersebut menunjukkan bahwa algoritma Deflate mampu mengurangi ukuran data yang ditransmisikan sehingga jumlah data yang melewati jaringan menjadi lebih kecil. Akibatnya, penggunaan bandwidth menjadi lebih efisien dan waktu pengiriman dapat dipersingkat tanpa memengaruhi integritas data yang diterima pada sisi server.

4. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, implementasi layanan Secure File Transfer Protocol (SFTP) pada server Linux Debian berhasil diterapkan dan berfungsi dengan baik dalam melakukan proses transfer data secara aman. Pengujian konektivitas menunjukkan bahwa komunikasi antara klien dan server dapat berjalan tanpa mengalami packet loss, sehingga lingkungan pengujian dinyatakan stabil untuk proses transfer data.

Hasil pengujian menunjukkan bahwa penerapan algoritma kompresi Deflate mampu meningkatkan performa transfer data pada layanan SFTP. Penggunaan kompresi Deflate berhasil menurunkan waktu pengiriman file pada seluruh skenario pengujian, yaitu file berukuran 10 MB, 50 MB, dan 100 MB. Selain itu, nilai throughput mengalami peningkatan dari rata-rata 1,18 MB/s menjadi lebih dari 2 MB/s setelah kompresi diaktifkan. Tingkat efisiensi waktu transfer yang diperoleh masing-masing sebesar 38,82%, 41,56%, dan 43,44%. Berdasarkan hasil tersebut, dapat disimpulkan bahwa penerapan algoritma kompresi Deflate pada layanan SFTP di server Linux Debian terbukti mampu meningkatkan efisiensi transfer data tanpa mengurangi aspek keamanan yang dimiliki oleh protokol SFTP.

Referensi

[1] SSH Communications Security, “SSH File Transfer Protocol (SFTP): Get SFTP Client & Server.” Accessed: Jun. 16, 2026. [Online]. Available: https://www.ssh.com/academy/ssh/sftp-ssh-file-transfer-protocol

[2] T. Jeremya, A. Lombu, L. Daeli, and S. Bulolo, “Jurnal Inovasi Teknologi dan Komputer Implementasi dan Analisis Keamanan Layanan SSH Server Menggunakan Autentikasi RSA dalam Lingkungan Virtualisasi dan Putty,” vol. 03, no. 01, pp. 33–45, 2025, doi: https://doi.org/10.54209/jitko.v3i1.143.

[3] M. Rizki *et al.*, “UJI PENETRASI MENGGUNAKAN HYDRA DAN METASPLOIT PADA PROTOKOL SECURE SHELL,” vol. 9, no. 1, pp. 1017–1024, 2025, doi: https://doi.org/10.36040/jati.v9i1.12583.

[4] A. W. Services, “Apa yang dimaksud dengan Secure File Transfer Protocol (SFTP)?” [Online]. Available: https://www.ssh.com/academy/ssh/sftp-ssh-file-transfer-protocol

[5] M. D. Ramadhan, A. Solehudin, S. Kom, and M. Kom, “Penerapan VPN L2TP/IPSec untuk Keamanan Akses Jaringan pada

DOI: https://doi.org/10.31004/riggs.v5i2.11339

Infrastruktur Jaringan," vol. 10, no. 02, pp. 78–86, 2025.
[6] M. D. Sahputra and I. M. Suartana, "Implementasi FTP Server Dengan Performa Secure File Transfer Protocol Berbasis Virtualiasasi Dan Container," vol. 06, pp. 284–296, 2024, doi: https://doi.org/10.26740/jinacs.v6n01.p284-296.
[7] A. Nurhuda *et al.*, "Perbandingan pencadangan data menggunakan RSync dan SFTP," vol. 4, no. 2, pp. 174–182, 2023, doi: https://doi.org/10.31102/jatim.v4i2.2512.
[8] L. Radionetworks, "Comparative Analysis of File Transfer Protocols in," pp. 27–32, 2021, doi: http://dx.doi.org/10.25673/36581.
[9] R. Nakamura, *Multi-threaded scp : Easy and Fast File Transfer over SSH*, vol. 1, no. 1. Association for Computing Machinery, 2023. doi: 10.1145/3569951.3597582.
[10] M. Irwan and F. Nurpandi, "Evaluation of Deflate Algorithm in Lossless Compression of Digital Document Formats," vol. 17, no. 2, pp. 235–246, 2025, doi: https://doi.org/10.35194/mji.v17i2.5746.
[11] F. Atasoy, "Performance Analysis of Compression Algorithms on Matrix Data : Data Transfer Optimization in Microservices Architectures Veri Sıkıştırma Algoritmalarının Matris Verileri Üzerindeki Performans Analizi," no. 1, pp. 49–60, 2024, doi: https://doi.org/10.46810/tdfd.1425959.
[12] M. C. Aruan and W. Rahayu, "Analisis Performa Algoritma Kompresi Data dalam Penyimpanan dan Transfer Data," vol. 1, no. 2, pp. 228–232, 2023, doi: https://doi.org/10.35870/ljit.v1i2.2157.
[13] M. Wisnu, A. Saputra, R. R. Huizen, and D. P. Hostiadi, "Design and Evaluation of a Hybrid AES-ECC Model for Secure Server Communication using REST API," vol. 6, no. 4, pp. 2740–2755, 2025.
[14] Y. Liu, S. Di, K. Chard, I. Foster, and F. Cappello, "Optimizing Scientific Data Transfer on Globus with Error-Bounded Lossy Compression," 2023, doi: https://doi.org/10.1109/ICDCS57875.2023.00064.
[15] U. I. Sekhar, M. Ramadevi, B. V. Sai, and A. Pradesh, "A Network Protocol That Uses Secure Shell for Secure , Encrypted File Transfers , Offering an Alternative to the Traditional File Transfer Protocol," vol. 16, 2025.
[16] N. B. Michael, "Enhancing File Transfer Security and Efficiency through Compression , Fragmentation and Reassembling Techniques," vol. 186, no. 39, 2024, doi: https://doi.org/10.5120/ijca2024923973.

DOI: https://doi.org/10.31004/riggs.v5i2.11339