Politeknik Caltex Riau

LAPORAN PROYEK AKHIR

PERBANDINGAN KINERJA MANAJEMEN STORAGE POOL DAN FILE SYSTEM MENGGUNAKAN STRATIS DAN LVM

Sarah Celia Jelly
NIM. 2056301032

Pembimbing
Muhammad Arif Fadhly Ridha, S.Kom., M.T.

PROGRAM STUDI TEKNOLOGI REKAYASA KOMPUTER
POLITEKNIK CALTEX RIAU
2024

Politeknik Caltex Riau

LAPORAN PROYEK AKHIR

PERBANDINGAN KINERJA MANAJEMEN STORAGE POOL DAN FILE SYSTEM MENGGUNAKAN STRATIS DAN LVM

Sarah Celia Jelly
NIM. 2056301032

Pembimbing
Muhammad Arif Fadhly Ridha, S.Kom., M.T.

PROGRAM STUDI TEKNOLOGI REKAYASA KOMPUTER
POLITEKNIK CALTEX RIAU
2024

HALAMAN PENGESAHAN

PERBANDINGAN KINERJA MANAJEMEN STORAGE POOL DAN FILE SYSTEM MENGGUNAKAN STRATIS DAN LVM

Sarah Celia Jelly
NIM. 2056301032

Proyek Akhir ini diajukan sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Terapan Teknik (S.Tr.T) di Politeknik Caltex Riau

Pekanbaru, 31 Mei 2024

Disetujui oleh:

Pembimbing,

Muhammad Arif Fadhly Ridha, S.Kom., M.T.
NIP. 138701

Penguji,

1. Dr. Agus Urip Ari Wibowo, S.T., M.T.
   NIP. 007001
2. Yuli Fitrisia, S.T., M.T.
   NIP. 088503

Mengetahui,
Ketua Program Studi Teknologi Rekayasa Komputer

Dr. Eng. Ananda Alim Syahbana, S.T., M.Sc.
NIP. 178809

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam proyek akhir yang berjudul :

“Perbandingan Kinerja Manajemen Storage Pool dan File System menggunakan Stratis dan LVM”

Adalah benar hasil karya saya, dan tidak mengandung karya ilmiah atau tulisan yang pernah diajukan di suatu Perguruan Tinggi.

Setiap kata yang dituliskan tidak mengandung plagiat, pernah ditulis maupun diterbitkan orang lain kecuali yang secara tertulis diacu dalam laporan proyek akhir ini dan disebutkan pada daftar pustaka. Saya siap menanggung seluruh akibat apabila terbukti melakukan plagiat.

Pekanbaru, 31 Mei 2024

Penulis

ABSTRAK

Media penyimpanan yang besar dibutuhkan bagi sebuah perusahaan, terutama untuk pengelolaan dan penyimpanan file. Seiring bertambahnya ukuran data yang semakin besar dari waktu ke waktu, hal ini harus didukung oleh media penyimpanan berkapasitas tinggi. Dengan permasalahan tersebut maka dibangun sebuah penyimpanan dengan menggabungkan *storage* dari beberapa komputer menggunakan aplikasi Stratis dan LVM. Dengan memanfaatkan penyimpanan yang ada di beberapa komputer dapat memaksimalkan perangkat keras yang ada dan mengurangi kebutuhan untuk membeli perangkat baru. Pengujian yang dilakukan adalah pengujian ukuran file yang menunjukkan hasil storage setelah digabungkan dengan *file system* Stratis dan LVM serta membandingkan kecepatan *upload* dan *download* file dan *write/read* file. Client dapat melakukan *transfer file* dengan menggunakan sistem operasi yang sama yaitu *CentOS 9* dengan berbagai ukuran file: 100, 200, 300, 400, dan 500 MB. Pengujian write/read file menggunakan perintah `iostat`. Pengujian *transfer file* dilakukan 10 kali dengan menggunakan aplikasi Teracopy. Dengan pengujian diperoleh hasil bahwa bahwa Stratis dan LVM sama-sama dapat menggabungkan storage dengan kapasitas yang lebih besar menggunakan *file system* XFS. LVM adalah *file system* yang lebih baik untuk kinerja kecepatan *transfer file* dengan rata-rata *upload* 82,8 MB/s dan *download* 79,72 MB/s sedangkan Stratis lebih baik untuk kinerja kecepatan tulis *disk* dengan rata-rata 254,9 MB/s menggunakan *tools* `dd`.

Kata Kunci: *LVM, Stratis, Secure copy, Iostat.*

ABSTRACT

Large storage media is needed for a company, especially for file management and storage. As data sizes increase over time, this must be supported by high capacity storage media. With these problems, a storage was built by combining storage from several computers using the Stratis and LVM applications. By utilizing existing storage on multiple computers you can maximize existing hardware and reduce the need to buy new devices. The test carried out is a file size test which shows the storage results after combining it with the Stratis and LVM file system as well as comparing the speed of uploading and downloading files and writing/reading files. Clients can transfer files using the same operating system, namely CentOS 9 with various file sizes: 100, 200, 300, 400, and 500 MB. Testing write/read files using the iostat command. File transfer testing was carried out 10 times using the Teracopy application. Testing results show that Stratis and LVM can both combine storage with a larger capacity using the XFS file system. LVM is a better file system for file transfer speed efficiency with an average upload of 82.8 MB/s and download of 79.72 MB/s while Stratis is better for disk write speed capability with an average of 254.9 MB/s using tools dd.

Keywords: *LVM, Stratis, Secure copy, Iostat.*

KATA PENGANTAR

Segala puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan proyek akhir yang berjudul "PERBANDINGAN KINERJA MANAJEMEN STORAGE POOL DAN FILE SYSTEM MENGGUNAKAN STRATIS DAN LVM". Proyek akhir ini disusun sebagai salah satu syarat untuk menyelesaikan jenjang pendidikan Diploma IV pada Program Studi Teknologi Rekayasa Komputer Politeknik Caltex Riau.

Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada pihak yang telah banyak memberikan bantuan dan dukungan yang tiada terhingga baik secara langsung maupun tidak langsung. Ucapan terima kasih tersebut penulis tujukan kepada:

1. Allah SWT atas rahmat dan karunia-Nya, sehingga penulis bisa menyelesaikan tugas akhir ini tepat waktu.
2. Kedua orang tua dan kakak penulis yang selalu mendo'akan dan memberikan semangat serta dorongan motivasi kepada penulis, sehingga penulis bisa menyelesaikan tugas akhir tepat waktu.
3. Bapak Muhammad Arif Fadhly Ridha, S.Kom., M.T. selaku pembimbing PA yang telah memberikan ilmu dan bimbingan kepada penulis dalam menyelesaikan proyek akhir.
4. Bapak Dr. Agus Urip Ari Wibowo, S.T., M.T dan ibu Yuli Fitrisia, S.T., M.T. selaku dosen penguji yang telah memberikan masukan dan saran dalam menyelesaikan proyek akhir.

Penulis sangat menyadari sepenuhnya bahwa laporan proyek akhir ini masih jauh dari sempurna, oleh karena itu segala jenis kritik, saran dan masukan yang membangun sangat penulis harapkan agar dapat memberikan wawasan bagi pembaca dan yang paling utama penulis sendiri.

Pekanbaru, 31 Mei 2024

Penulis

UCAPAN TERIMAKASIH

1. Bapak Dr. Dadang Syarif Sihabudin Sahid, S.Si,M.Sc. selaku Direktur Politeknik Caltex Riau yang telah memberikan dukungan moral dalam menyelesaikan proyek akhir ini.
2. Bapak Yoanda Alim Syahbana, S.T., M.Sc. selaku Ketua Program Studi Teknologi Rekayasa Komputer yang telah memberikan izin untuk menyelesaikan proyek akhir.
3. Bapak Memen Akbar, S.Si., M.T. selaku wali dosen 4TRK yang telah memberikan ilmu dan bimbingan dengan penuh kesabaran dalam menyelesaikan proyek akhir.
4. Seluruh Dosen Program Studi Teknologi Rekayasa Komputer yang telah memberikan bekal ilmu kepada penulis dalam menyelesaikan proyek akhir.
5. Muhammad Rafli Ramadhan yang selalu mendukung, menemani, menyemangati, membantu, serta memotivasi saya untuk menyelesaikan proyek akhir ini.
6. Teman-teman THE TEAM E atas hari-hari terbaik dalam mengerjakan proyek akhir bersama dengan selalu menyemangati dan membantu selama proses pengerjaan.
7. Terimakasih warga 4TRK, generasi 20 Teknologi Rekayasa Komputer dan seluruh pihak yang terlibat dalam menyelesaikan studi di Politeknik Caltex Riau yang telah memberikan dukungan dan semangat dalam menyelesaikan proyek akhir ini

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi telah banyak memberikan kemudahan terutama dalam mendapatkan informasi. Selain itu, banyak aspek kehidupan yang bergantung pada teknologi yang semakin berkembang yang harus didukung dengan media penyimpanan dengan kapasitas yang besar, karena semakin lama perkembangan *file* juga semakin banyak dan memiliki ukuran yang besar. Dengan perkembangan teknologi saat ini, media penyimpanan yang dibutuhkan juga harus semakin besar. Media penyimpanan merupakan perangkat yang dapat digunakan untuk menyimpan *file*/informasi yang merupakan salah satu poin penting, karena semua media disimpan di dalam media penyimpanan. *File* disini dapat berupa dokumen, program atau lainnya.

Kondisi penyimpanan pada saat ini menggunakan lebih dari satu hard *disk* yang *standalone*. File-file dengan kriteria sama yang disimpan pada hard*disk*-hard*disk* tersebut menjadi tidak teratur dan tidak dapat dikumpulkan dalam satu *path* / folder. Oleh karena itu, muncul permasalahan yaitu membuat *file* tersebut tersusun secara tepisah antara masing-masing *disk*. Permasalahan tersebut mengakibatkan tidak dapat dilakukannya manajemen untuk menggabungkan *hard disk*. Dengan menggabungkan *harddisk yang standalone* tersebut, maka akan mendapatkan media penyimpanan yang lebih besar pada satu *path* / folder.

Dari permasalahan tersebut, terdapat beberapa teknologi atau teknik sebagai solusi untuk penyimpanan *file* yang cukup besar dan manajemen *file system* yang memiliki kinerja yang baik yaitu dengan teknologi Stratis dan LVM pada Proxmox karena teknologi tersebut menyediakan fleksibilitas yang tinggi dalam mengelola penyimpanan. Teknologi tersebut memungkinkan pengguna untuk menggabungkan, membagi, dan mengubah ukuran volume logis dengan mudah tanpa memengaruhi *file* yang ada. Kemudian teknologi tersebut memiliki kemampuan untuk mengalokasikan ruang penyimpanan yang efisien

dan dapat mengelola beberapa *hard disk* atau penyimpanan yang tersebar dalam satu tempat, melakukan tugas administratif seperti membuat, mengubah ukuran, atau menghapus volume logis dengan mudah melalui antarmuka yang konsisten. Ketika menggunakan *Storage Pool* seperti Stratis atau LVM yang mana nantinya akan dikumpulkan menjadi satu kolam / *pool*, maka *file system* akan mengatur sendiri posisi *file* tersebut dimana akan tersusun dengan rata dan terstruktur, yang pastinya akan menambah *space* dan memudahkan pengguna untuk membuat partisi.

Metode Stratis adalah solusi mengotomatiskan manajemen penyimpanan lokal yang memungkinkan beberapa sistem *file* logis berbagi *file* kumpulan penyimpanan dan dialokasikan dari satu atau lebih perangkat blok yang digunakan untuk memudahkan mengkonfigurasi *pool*s dan *file system* dengan fungsionalitas penyimpanan yang ditingkatkan dan berfungsi dalam tumpukan manajemen penyimpanan Linux (Keefe, 2023). Kemudian teknologi kedua yaitu LVM yang merupakan pengelolaan *hard disk* yang memungkinkan penggabungan beberapa *hard disk* pada satu komputer untuk memperoleh kapasitas yang lebih besar (Hartawan & Iswara, 2015). LVM dapat memberikan kemudahan kepada system administrator dalam mengatur alokasi space partisi.

Berdasarkan penjelasan di atas maka akan dibandingkan kinerja manajemen penggabungan hard*disk Storage Pool* dan *file system* dengan dibuatnya **"Perbandingan Kinerja Manajemen *Storage Pool* dan *File System* menggunakan Stratis dan LVM"**, yang nantinya akan dikumpulkan menjadi satu *Storage Pool* untuk mengetahui kinerja dari Stratis dan LVM sehingga dapat membuktikan teknologi mana yang paling tepat untuk manajemen *file system* ini. Pada penelitian ini akan menggabungkan hard*disk*-hard*disk standalone* menjadi sebuah hard*disk* secara logik. Dengan manajemen penggabungan hard*disk*, diharapkan file-file dengan kriteria sama dapat dikumpulkan dalam satu *path*/folder. Sistem ini berjalan dengan menggabungkan dua node tersebut saling terhubung satu dengan yang lain melalui jaringan sehingga membentuk sebuah *pool* tunggal dari seluruh penyimpanan *file*. Maka hasil akhirnya membandingkan parameter pengukuran kinerja *storage* pada ke dua *file system* dengan *tools*nya. Kemudian, dilakukan perbandingan serta

menganalisis sumber daya dengan parameter berupa kecepatan *transfer file* dan write/read *file*. Berdasarkan hasil analisis tersebut diharapkan dapat mengetahui bagaimana kinerja dari ke dua *file system*.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah diuraikan, maka perumusan masalah dalam proyek akhir ini adalah sebagai berikut:

1) Bagaimana menerapkan *management file system* pada *Linux*.
2) Bagaimana membandingkan pengukuran kinerja *file system* dan *Storage Pool* (media penyimpanan) dari dua jenis *file system* Stratis dan LVM.
3) Bagaimana menggabungkan *storage* (media penyimpanan) dari dua *disk* pada 1 *path* sehingga kapasitas penyimpanan yang tersedia menjadi lebih besar.

1.3 Batasan Masalah

Adapun batasan masalah dalam pembuatan proyek akhir ini adalah:

1) Menggunakan dua jenis *file system* yaitu Stratis dan LVM.
2) Sistem operasi yang digunakan adalah *CentOS 9.*
3) Tidak membahas tentang keamanan pada server dan aplikasinya.
4) Pembuatan *cloud computing* dibuat dengan 2 komputer fisik dan 2 node atau virtualisasi.
5) Proyek akhir ini tidak membahas masalah network security, seperti file hilang, file corrupt dll.
6) Jumlah har*disk* yang digunakan masing – masing *file system* adalah 2 buah.
7) Teknologi virtualisasi yang digunakan adalah Proxmox.

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Adapun tujuan dari penelitian proyek akhir ini adalah:

1) Membandingkan dan menganalisis kinerja read/write dari kedua jenis manajemen *file system* yaitu Stratis dan LVM.
2) Menggabungkan *storage* dari beberapa *disk* pada satu *path* sehingga kapasitas penyimpanan yang tersedia menjadi lebih besar dengan menggunakan *file system* Stratis dan LVM.

1.4.2 Manfaat

Manfaat yang diharapkan dari penelitian proyek akhir ini adalah:

1) Memberikan hasil perbandingan kinerja dari kedua jenis *file system* yaitu Stratis dan LVM.
2) Mendapatkan hasil kinerja *file system* yang bekerja secara optimal antara Stratis dan LVM.
3) Menghasilkan sebuah *storage* yang besar agar dapat digunakan untuk kebutuhan file yang besar.
4) Mendapatkan pemahaman kelemahan dan kelebihan dari Stratis dan LVM.

1.5 Metodologi Penelitian

Metode penelitian yang dipakai dalam pembuatan proyek akhir ini adalah:

1) Studi Literatur
   Pada tahapan ini dilakukan kegiatan seperti membaca, mempelajari literatur penelitian terdahulu yang memiliki keterkaitan pada permasalahan yang dibahas, dan menganalisis buku-buku, jurnal dan artikel yang berkaitan dengan pelaksanaan proyek akhir.
2) Perencanaan dan Desain

Perencanaan dan perancangan sistem dilakukan sesuai dengan tujuan dan manfaat dari perbandingan kinerja sistem menggunakan Stratis dan LVM

3) Implementasi
   Pada tahap implementasi, akan diterapkan perencanaan dan perancangan yang telah dibuat sebelumnya.
4) Analsis dan Evaluasi
   Setelah dilakukan implementasi, akan dilakukan analisis dan evaluasi pada perbandingan kinerja sistem berdasarkan metode pengujian.
5) Penulisan Laporan
   Dalam penulisan laporan ini mengacu pada pedoman penulisan ilmiah, dalam hal ini penulisan proyek akhir yang bentuk bakunya telah diatur oleh pihak Politeknik Caltex Riau.

1.6 Sistematika Penulisan

Sistematika penulisan laporan proyek akhir ini secara keseluruhan terdiri dari empat bab, masing-masing terdiri dari beberapa sub bab. Adapun pokok pembahasan dari masing-masing bab tersebut secara garis besar sebagai berikut:

BAB I PENDAHULUAN

Bab ini menguraikan tentang latar belakang masalah, perumusan masalah dan ruang lingkup masalah, tujuan dan manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menguraikan lima hasil penelitian terdahulu dan sebelas landasan teori yang diperlukan terkait proyek akhir ini seperti Stratis, LVM, Proxmox, *Storage Pool*, *File System*, dll. Kemudian juga terdapat tabel perbandingan penelitian terdahulu dengan sekarang.

BAB III PERANCANGAN

Bab ini menjelaskan tentang perancangan sistem terdiri dari perancangan sistem yang akan dibangun meliputi jumlah node, jumlah hard *disk*, jumlah virtual machine, PC Client. Kemudian juga terdapat

skenario tentang perancangan sistem yang akan dibangun, skenario IP, kebutuhan perangkat yang digunakan, dan metode pengujian.

BAB IV PENGUJIAN DAN ANALISIS

Bab ini berisi informasi mengenai pengujian pada kecepatan *upload* file, write/read file serta *tools* yang digunakan dan analisisnya.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran setelah melaksanakan proyek akhir.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Dalam penelitian terdahulu ini diambil lima penelitian terdahulu yang sebelumnya telah dilakukan pengujian dan terdapat sembilan landasan teori yang digunakan.

Tabel 2.1 Penelitian terdahulu dan saat ini

Nama Peneliti	Teknik Virtualisasi	File System	Digunakan pada	Tujuan
Furliana Novriyanti Sigiro (Sigiro et al., 2017)	Cloud Computing	MooseFS dan GlusterFS	File *transfer*	Menggabungkan *storage* pada beberapa node dengan teknologi *clustered file system* yaitu *MooseFS* dan *GlusterFS*.
Kasero Ade Pratama (Pratama, 2017)	Cloud Computing	NFS	File *Transfer*	Membangun *Data center* menggunakan *cloud computing.*
Erin Fatmasari (Fatmasari, 2020)	Cloud Computing	OrangeFS dan Lustre	File *Transfer*	Menguji kelayakan *data center* yang dibangun.
Sugeng Purwantoro ESGS (Purwantoro E.S.G.S, 2022)	Cloud Computing	GlusterFS dan Ceph	File *Transfer*, Write/Read File	Memperbesar kapasitas penyimpanan dengan metode *Clustered file system.*

Nama Peneliti	Teknik Virtualisasi	File System	Digunakan pada	Tujuan
M. Farhan Adhitya (Adhitya, 2022)	Cloud Computing	Btrfs, ZFS, dan LVM	Write/Read File	Membandingkan kinerja dari setiap file system yang berbasiskan LXD.
Sarah Celia Jelly	Cloud Computing	Stratis dan LVM	File *Transfer*, Write/Read File	Pengujian ukuran file menggunakan *file system* Stratis dan LVM.

Pada topik penelitian pertama, telah dilakukan oleh Furliana Novrianti Sigiro dkk. (Sigiro et al., 2017). Pada penelitian ini melakukan aktifitas *transfer* file *upload* dan *download* untuk mengetahui nilai kecepatan rata-rata *transfer* file dan penggunaan CPU, *memory* dan throughput menggunakan aplikasi wireshark dan SNMP.

Pada topik penelitian kedua dilakukan oleh Kasero Ade Pratama (Pratama, 2017). Pada penelitian ini menggunakan 4 buah node virtual yaitu 1 node berperan sebagai pengaksesan ke file center dan juga berfungsi sebagai admin node, 1 node berperan sebagai node CEPH MON yang berfungsi untuk memonitoring dan memetakan cluster, 2 node sebagai node CEPH OSD yang berfungsi sebagai tempat penyimpanan file. Hasil yang didapat dari penelitian ini yaitu layanan file center dan file yang tersimpan pada file center tetap dapat diakses walaupun salah satu node dalam keadaan mati dan mencegah terjadinya single point of failure.

Pada topik penelitian ketiga dilakukan oleh Erin Fatmasari (Fatmasari, 2020). Penelitian ini akan menggabungkan *storage* pada beberapa node sehingga akan menjadi sebuah *storage* yang besar dan akan dibandingkan kinerja dari kedua perangkat lunak yang akan digunakan untuk membangun sebuah Clustered *File system*. Maka akan dibangun sebuah penyimpanan untuk file center dengan menggabungkan *storage* dari beberapa komputer dan memanfaatkan

sebuah teknologi clustered *file system* dengan menggunakan aplikasi OrangeFS dan Lustre.

Pada topik penelitian keempat dilakukan oleh Erick Wijaya (Wijaya & Sinisuka, 2022). Pada penelitian ini yaitu membandingkan kecepatan *upload* dan *download* file dan write/read file pada GlusterFS yang menggunakan 2 node sebagai peer dan CephFS menggunakan 3 node yaitu 1 node berperan sebagai MON yang berfungsi untuk memonitoring dan memetakan cluster, 2 node sebagai OSD yang berfungsi sebagai tempat penyimpanan file. Client dapat melakukan *upload* dan *download* file dengan menggunakan layanan *storage* yaitu Samba.

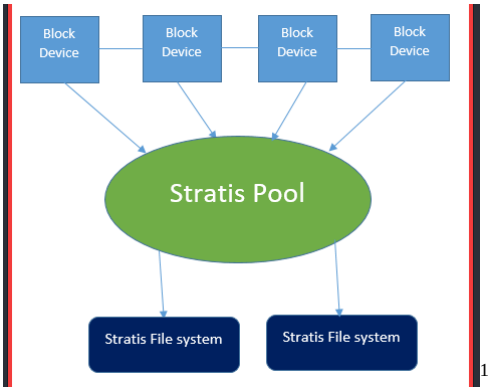
Pada topik penelitian kelima dilakukan oleh Farhan Adhitya (Adhitya, 2022). Pada penelitian ini menggunakan LXD container sebagai wadah untuk menganalisis kinerja dari setiap *file system*. Dengan menggunakan virtualisasi VMware Workstation, maka di pengujian ini membuat tiga mesin virtual yang telah diinstal LXD. Ukuran file yang digunakan untuk pengujian ini sebesar 1 GB. File berukuran 1 GB akan diuji 5 kali pada setiap *file system* untuk mendapatkan hasil kecepatan *transfer* dan kecepatan read/write. Dari pengujian ini, maka didapatkan hasil kecepatan *transfer* file pada Btrfs 1035.184 MB/s jauh lebih cepat dibanding ZFS 526.75 MB/s dan LVM 187.028 MB/s. Untuk kecepatan write, ZFS jauh lebih cepat yaitu 283.54 kB/s dibandingkan LVM 249.92 kB/s dan Btrfs 112.02 kB/s. Sedangkan untuk kecepatan read ZFS jauh lebih cepat yaitu 1248.15 kB/s dibandingkan LVM 1092. kB/s dan Btrfs 432,13 kB/s.

Pada penelitian proyek akhir ini akan dilakukan Perbandingan kinerja manajemen *Storage Pool* dan *file system* menggunakan Stratis dan LVM. Penelitian ini akan menggabungkan *storage* pada beberapa node sehingga menjadi sebuah *storage* yang besar dan membandingkan kinerja *file system* tersebut. Penelitian ini diharapkan dapat memberikan hasil yang akan membantu dalam pemilihan *file system* dengan kinerja terbaik dalam menggabungkan hard*disk* dari beberapa komputer agar memperoleh storge yang besar yang dapat mempercepat komputasi. Perangkat lunak yang digunakan pada penelitian ini adalah Stratis dan LVM.

2.2 Landasan Teori

2.2.1 *Stratis*

Stratis adalah solusi mengotomatiskan manajemen penyimpanan lokal yang memungkinkan beberapa sistem *file* logis berbagi *file* kumpulan penyimpanan dan dialokasikan dari satu atau lebih perangkat blok yang digunakan untuk memudahkan mengkonfigurasi *pools* dan *file system* dengan fungsionalitas penyimpanan yang ditingkatkan dan berfungsi dalam tumpukan manajemen penyimpanan *Linux* (Keefe, 2023).



[1]

Gambar 2.1 Stratis

Sumber : (Victor, 2024)

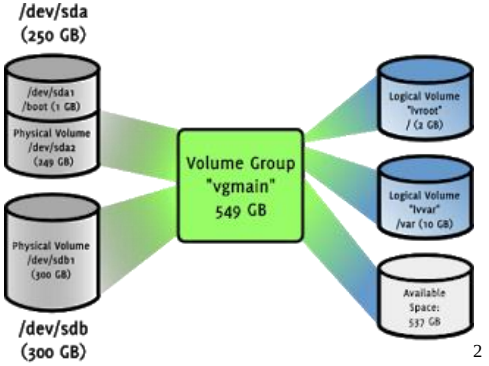
Untuk mencapai hal ini, Stratis memprioritaskan pengalaman baris perintah langsung, API yang kaya, dan pendekatan manajemen penyimpanan yang sepenuhnya otomatis. Itu dibangun di atas elemen tumpukan penyimpanan yang ada sebanyak mungkin. Secara khusus, Stratis menggunakan device-mapper, LUKS, XFS, dan Clevis. Stratis juga dapat menggabungkan teknologi tambahan di masa menfileng. Stratis dapat mengonfigurasi kumpulan penyimpanan terenkripsi atau tidak terenkripsi dengan satu atau lebih sistem file dengan cepat dan

[1] Victor, Managing Layered Storage with Stratis in Linux – RHEL 8/CentOS 8, https://tekneed.com/managing-layered-storage-with-stratis-in-linux/ diakses pada tanggal 19 Juni 2024

tanpa pengetahuan sebelumnya tentang banyak lapisan dan perintah penyimpanan. *Linux* memiliki sejumlah teknologi penyimpanan yang menyediakan fungsionalitas tingkat lanjut pada aplikasi untuk mengakses dan menyimpan file.

2.2.2 *LVM*

LVM yang merupakan singkatan dari Logical Volume Manager yang merupakan pengelolaan hard *disk* yang memungkinkan penggabungan beberapa hard *disk* pada satu komputer untuk memperoleh kapasitas yang lebih besar (Hartawan & Iswara, 2015).



[2]

Gambar 2.2 LVM

Sumber : (Marc GUILLAUME, 2024)

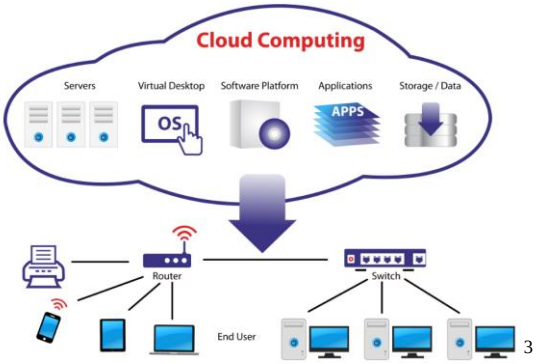
LVM dapat memungkinkan untuk membuat layer layer pada *harddisk*/ partisi yang digunakannya. Dengan LVM, hdd atau partisi dibuat menjadi satu buah Logical Volume yang terdiri dari beberapa hdd/ partisi. LVM tidak mempengaruhi OS dikarenakan LVM hanya akan memberitahu Volume Group (*disk*) dan Logical Volume (partisi) yang dibuat. Sebelum adanya LVM, suatu har*disk* akan dibagi ke beberapa partisi sesuai kebutuhan server. Ada partisi /boot, partisi /root, partisi /home dan seterusnya. Pada saat melakukan partisi ini,

[2] Marc GUILLAUME, Basic Debian Installation - Debian 5.0 Lenny, https://www.yakati.com/art/installation-debian-de-base-debian-5-0-lenny.html diakses pada tanggal 19 Juni 2024

harus dilakukan dengan hati-hati karena sulit sekali untuk melakukan resize. LVM file untuk membuat partisi yang lebih fleksibel. Dengan LVM, beberapa *harddisk* dapat digabungkan dengan har*disk* lain dan menjadi sebuah partisi baru. Partisi baru ini dipecah lagi menjadi partisi-partisi baru yang mudak dilakukan resize.

2.2.3 *Cloud computing*

Cloud computing adalah sebuah model yang memungkinkan untuk ubiquitous yang mana *On-demand* akses jaringan ke sumber daya komputasi yang dapat cepat dirilis atau ditambahkan.



Gambar 2.3 *Cloud Computing*

Sumber : (A. Maier, 2024)

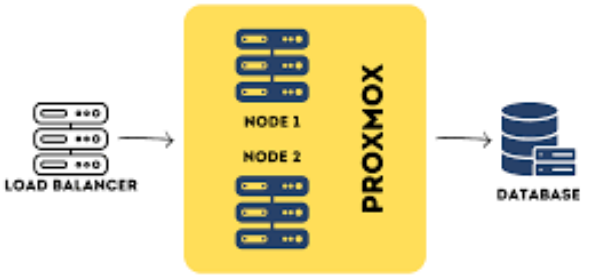
Menurut Peter Mell (Mell & Grance, 2012) Komputasi Cloud merupakan sebuah model yang memungkinkan untuk ubiquitous (Dimanapun dan Kapanpun), Nyaman, On-demand akses jaringan ke sumber daya komputasi (Contoh : jaringan, server, *storage*, aplikasi dan layanan) yang dapat dengan cepat dirilis atau ditambahkan.

2.2.4 *Cloud proxmox*

Proxmox adalah sebuah distro Linux virtualisasi berbasis Debian (64 bit) yang mengusung OpenVZ dan KVM. Proxmox

[3] A. Maier, Understanding the Fundamentals of a Cloud Computing Architecture, https://codecoda.com/en/blog/entry/understanding-the-fundamentals-of-a-cloud-computing-architecture diakses pada 20 Juni 2024

memungkinkan untuk melakukan manajemen terpusat dari banyak server fisik. Sebuah proxmox terdiri dari minimal satu master dan beberapa node (minimal satu master dan satu node) (Julianti et al., 2019).



[4]

Gambar 2.4 *Cloud Proxmox*

Sumber : (Admin, 2024)

Proxmox adalah solusi open-source yang mendukung teknologi virtualisasi seperti KVM dan LXC. Dengan menggunakan Proxmox, pengguna dapat mengatur dan mengelola VM dan kontainer mereka secara efisien melalui antarmuka web yang mudah digunakan. Selain itu, Proxmox juga menyediakan fitur-fitur seperti manajemen penyimpanan, pengelolaan jaringan, dan pemantauan kinerja sistem yang membantu pengguna dalam mengoptimalkan infrastruktur virtual mereka.

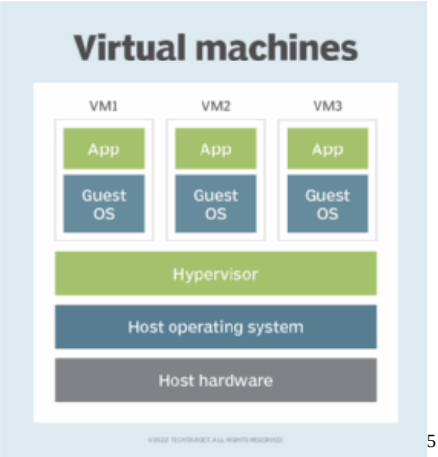
Proxmox juga mendukung integrasi dengan teknologi cloud, sehingga pengguna dapat menghubungkan mesin virtual mereka dengan layanan cloud seperti *Amazon Web Services* (AWS) dan Microsoft Azure. Hal ini membuat Proxmox menjadi pilihan yang populer bagi organisasi yang ingin mengembangkan infrastruktur cloud mereka sendiri dengan menggunakan teknologi virtualisasi yang andal dan efisien. Proxmox VE adalah platform open source lengkap

[4] Admin, Cara menggunakan Cluster di Proxmox, https://generalsolusindo.com/cara-menggunakan-cluster-di-proxmox/ diakses pada 21 Juni 2024

untuk semua perusahaan virtualisasi inklusif yang mengintegrasikan hypervisor KVM dan LXC dengan baik, penyimpanan yang ditentukan perangkat lunak dan fungsionalitas jaringan pada satu platform.

2.2.5 *Virtual machine*

Virtual Machine (VM) adalah perangkat lunak komputer yang berfungsi seperti komputer fisik dimana dapat menjalankan sistem operasi dan aplikasi. VM tersusun dari sekumpulan berkas konfigurasi dan spesifikasi yang didukung oleh physical resources milik host (Iswara & Yasa, 2021).



[5]

Gambar 2.5 *Virtual Machine*

Sumber : (Kinza Yasar, 2024)

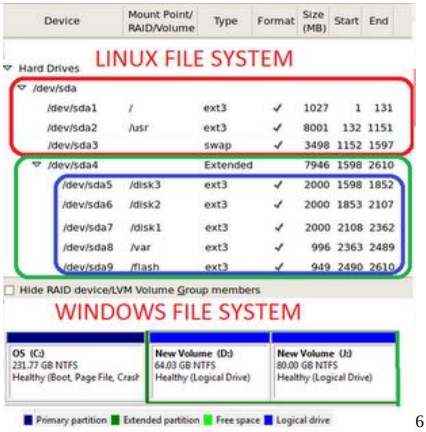
VM ditujukan untuk pengujian, mencadangkan file, atau menjalankan aplikasi berbasis perangkat lunak (SaaS). Secara harfiah, VM adalah wadah perangkat lunak yang terisolasi ketat yang dapat menjalankan sistem operasi dan aplikasi seolah-olah itu adalah komputer fisik. Meskipun tidak ada perangkat keras yang diperlukan

[5] Kinza Yasar, *Virtual Server*, https://www.techtarget.com/searchnetworking/definition/virtual-server diakses pada 21 Juni 2024

VM untuk beroperasi, itu masih berisi *Central Processing Unit* (CPU), *Random Access Memory* (RAM), *hard disk* dan kartu antarmuka jaringan (NIC), seperti komputer fisik. VM memungkinkan bisnis menjalankan sistem operasi yang berperilaku seperti komputer yang benar-benar terpisah, baik di jendela aplikasi atau di desktop. Penggunaan umum VM termasuk menjalankan perangkat lunak yang memerlukan sistem operasi berbeda atau untuk menguji aplikasi di lingkungan yang aman.

2.2.6 *File System*

File system mengatur bagaimana suatu file disimpan dalam suatu media penyimpanan file.



[6]

Gambar 2.6 *File System*

Sumber : (Rahul Panwar, 2024)

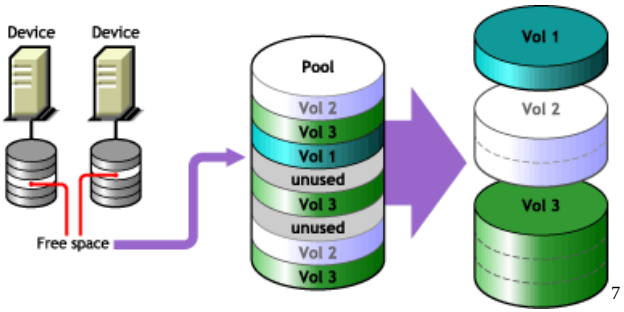
Seiring dengan perkembangan zaman, muncul berbagai macam *file system* yang mana masingmasing *file system* tersebut memiliki metode penyimpanan tersendiri disesuaikan dengan kebutuhan masing-masing. *File system* mengatur bagaimana suatu file disimpan dalam suatu media penyimpanan file. Seiring dengan

[6] Rahul Panwar, Linux File System and Windows File System, Difference, https://linuxexplore.com/2012/10/01/linux-file-system-and-windows-file-system-difference/ diakses pada 21 Juni 2024

perkembangan zaman, muncul berbagai macam *file system* yang mana masing-masing *file system* tersebut memiliki metode penyimpanan tersendiri disesuaikan dengan kebutuhan masing-masing. Perkembangan *file system* juga diimbangi dengan teknologi *storage* yang dipakai (Maktum et al., 2012).

2.2.7 *Storage Pool*

Storage Pool adalah kumpulan kapasitas dari sumber daya penyimpanan fisik yang berbeda dalam lingkungan penyimpanan bersama.



[7]

Gambar 2.7 *Storage Pool*

Sumber : (Anna, 2024)

Storage Pool dapat dikonfigurasi dalam berbagai ukuran dan memberikan sejumlah manfaat, termasuk peningkatan kinerja, manajemen, dan perlindungan file. Kumpulan dapat disediakan untuk menyertakan sejumlah kapasitas dan menggunakan kombinasi ruang penyimpanan fisik apa pun dalam jaringan area penyimpanan (SAN). Di lingkungan server virtual, mesin virtual (VM) dapat disimpan di kumpulan khusus, memastikan VM kritis memiliki akses ke jumlah penyimpanan yang tepat. *Storage Pool* juga merupakan volume penyimpanan. Volume penyimpanan adalah unit dasar penyimpanan, seperti ruang yang dialokasikan pada *disk* atau kartrid pita tunggal.

[7] Anna, ADM Novell W2 File System, https://admzs3.wordpress.com/2013/11/13/adm-novell-system-plikow/ diakses pada 21 Juni 2024

Server menggunakan volume penyimpanan untuk menyimpan file yang dicadangkan, diarsipkan, atau dikelola ruang. Server menyediakan tiga jenis kumpulan penyimpanan yang melayani tujuan berbeda: kumpulan penyimpanan utama, kumpulan penyimpanan salinan, dan kumpulan file aktif.

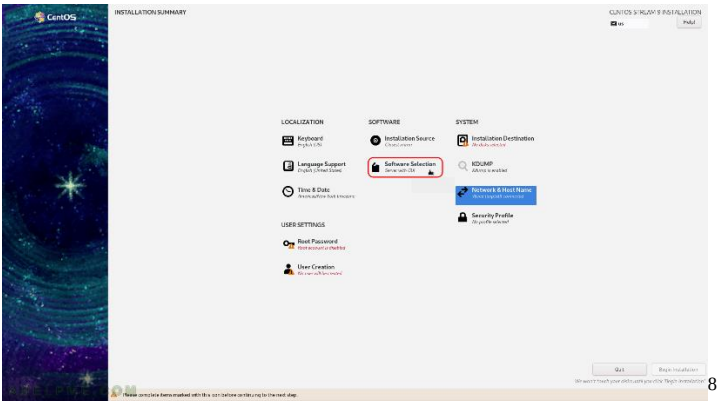
2.2.8 *CentOS*

Community Enterprise Operating System (CentOS) adalah Distro Linux yang cocok dipakai dalam skala Enterprise atau skala perusahaan yang bebas biaya atau Gratis. CentOS sendiri dikembangkan dari source code Red Hat Enterprise (RHEL) yang dikembangkan dalam sebuah komunitas yang disebut CentOS Project. Tujuan pengembangan awal sistem operasi CentOS yaitu untuk menyediakan platform komputasi skala perusahaan secara gratis dengan mempertahankan 100% kompabilitas terhadap Red Hat Linux atau CentOS dapat dikatakan sebagai kloningan dari RHEL. RHL awalannya disebut sebagai Red Hat Commercial Linux yang merupakan distro linux pertama yang memakai sistem RPM Package Manager, selanjutnya diikuti sebagian perusahaan lain, seperti Mandriva Linux serta SUSE Linux (Sulthon, 2023).

Desktop didasarkan pada GNOME 40 (GNOME 3.28 termasuk dalam RHEL 8) dan perpustakaan GTK 4. Di GNOME 40, desktop virtual dalam mode Tinjauan Aktivitas diatur ke orientasi lanskap dan muncul sebagai loop berkelanjutan dari kiri ke kanan. Secara default, Menu mulai GRUB disembunyikan jika RHEL adalah satu-satunya distribusi di sistem dan jika awal sebelumnya berhasil. Untuk menampilkan menu saat boot, cukup tahan tombol Shift atau tekan tombol Esc atau F8 beberapa kali. Kinerja SELinux telah ditingkatkan secara signifikan dan dengan konsumsi memori yang lebih sedikit. Di /etc/selinux/config, dukungan untuk pengaturan "SELINUX = disabled" untuk menonaktifkan SELinux telah dihapus.

Tabel 2.2 Persyaratan RAM Minimum

Jenis Instalasi	RAM Minimum
Instalasi media local (USB, DVD)	768 MiB
Instalasi jaringan NFS	768 MiB
Instalasi jaringan HTTP, HTTPS atau FTP	1,5 GB



[8]

Gambar 2.8 *CentOS*

Sumber : (NeoX, 2024)

2.2.9 *Ubuntu*

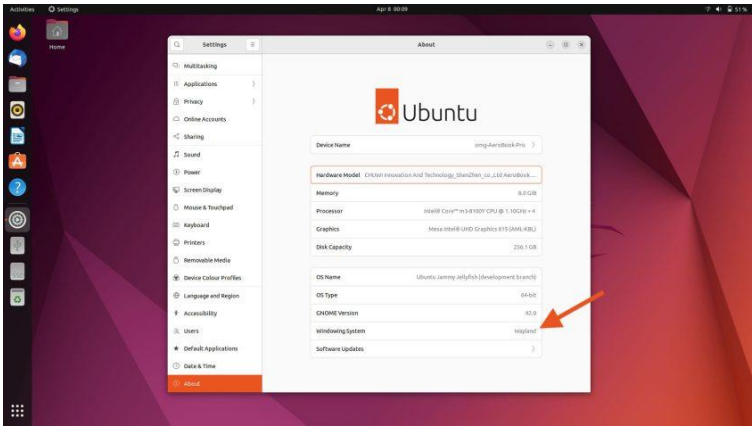
Ubuntu adalah sistem operasi turunan dari distro Linux jenis Debian unstable (sid), *Ubuntu* merupakan project untuk komunitas, yang bertujuan untuk menciptakan sebuah sistem operasi beserta dengan paket aplikasinya yang bersifat free dan open source, karena Ubuntu mempunyai prinsip untuk selamanya bersifat gratis *(free of charge)* dan tidak ada tambahan untuk versi enterprise edition. Ubuntu memiliki berbagai kelebihan distribusi debian diantaranya adalah :

a. Pemaketan (*Packaging*)
b. Pemilihan aplikasi yang luas (*Application choice*)

[8] neoX, Install CentOS Stream 9 Workstation (Gnome GUI), https://ahelpme.com/linux/centos-stream-9/install-centos-stream-9-workstation-gnome-gui/ diakses pada 23 Juli 2024

c. Siklus pembaharuan dilakukan secara rutin (*Updates*)
d. Dikenal stabilitas dan kualitasnya terutama di sisi Server (*Stability and quality*)

Ubuntu ini dikembangkan oleh Canonical Ltd yang merupakan sebuah perusahaan menyediakan pembaruan keamanan dan dukungan untuk setiap rilis *Ubuntu* (Muhyidin & Candra, 2016).



[9]

Gambar 2.9 *Ubuntu*

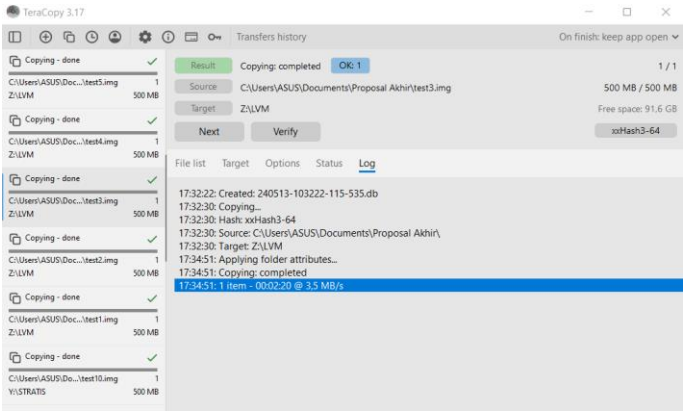
Sumber : (1000logos, 2024)

2.2.10 *Teracopy*

Teracopy adalah sebuah aplikasi yang digunakan untuk mengcopy file secara maksimal, dan teracopy juga dapat mem-pause sampai meng-skip file yang sedang berjalan dan dapat melanjutkan Kembali filefile yang sedang dicopy (Pratama, 2017). Dalam penelitian dibutuhkan *tools*yang dipergunakanuntuk mengumpulkan data-data. Data yang dikumpulkan adalah data sekunder yang berasal dari pencatatan *tools* yang dipergunakan. Didalam *tools* tersebut data yang di*transfer* akan menghasilkan *transfer rate* dan waktu *transfer* data sehingga nantinya data ini akan dijadikan

[9] 1000logos.net, UBUNTU LOGO, https://1000logos.net/ubuntu-logo/ diakses pada 21 Juni 2024

sebagai tolok ukur kualitas *transfer* data terhadap masing-masing fileyang di*transfer*.



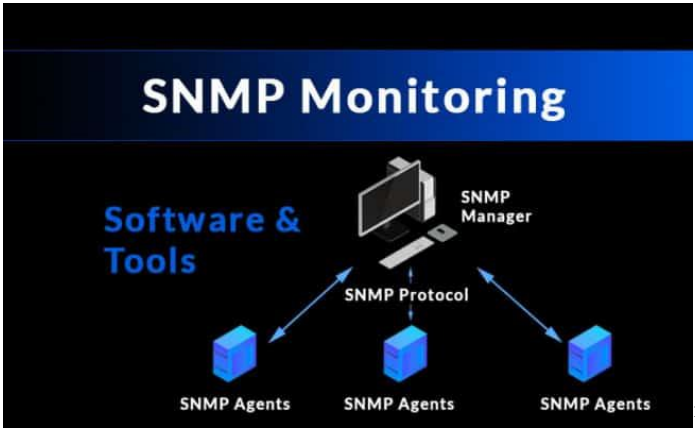
Gambar 2.10 *Teracopy*

2.2.11 *Simple Network Monitoring Protocol (SNMP)*

Simple Network Management Protocol (SNMP) adalah sebuah protokol yang digunakan untuk memonitor perangkat pada jaringan, seperti memonitor peralatan jaringan (seperti *switch*), peralatan komputer, dan perangkat lainnya (Mauro & Schmidt, 2005). Pradikta (2013) memaparkan perancangan sistem pemantauan menggunakan protokol SNMP. Protokol SNMP digunakan untuk mengumpulkan data perangkat dengan melakukan permintaan informasi pada perangkat. Informasi yang dikumpulkan digunakan untuk menghitung nilai *availability* pada perangkat. Basis data digunakan untuk menampung hasil dari permintaan SNMP.

Secara umum SNMP adalah sebuah protokol yang didesain untuk memberikan kemampuan pengumpulan data manajemen perangkat jaringan dan pengkonfigurasian perangkat jaringan secara jarak jauh (remotely). Pengelolaan ini dilakukan dengan cara melakukan polling dan setting variabel-variabel elemen jaringan yang dikelolanya. SNMP dirancang oleh Internet Engineering Task Force (IETF) untuk pemakaian di internet. SNMP memanfaatkan datagram UDP untuk pesannya pada perangkat jaringan. Karena pesan UDP bersifat unreliable (tidak dapat diandalkan) maka SNMP

menggunakan prosedur time out dan retry count untuk memecahkan masalah ini.



[10]

Gambar 2.11 SNMP

Sumber : (Brett Knight, 2024)

2.2.12 *Paessler Router Traffic Grapher (PRTG)*

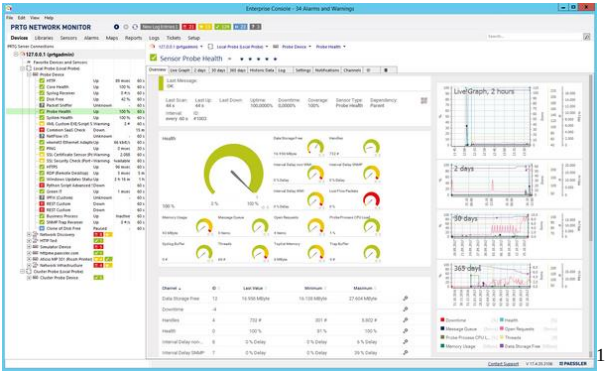
Monitor Jaringan Paessler Router Traffic Grapher (PRTG) adalah perangkat lunak pemantauan dan manajemen infrastruktur tanpa agen. Sistem ini dirancang untuk mempermudah proses pemantauan seluruh jaringan komputer dan perangkat IoT, sehingga dapat langsung mengetahui jika ada sesuatu yang terhubung atau berjalan dengan baik. Perangkat lunak Paessler PRTG membantu meningkatkan efisiensi jaringan karena memonitor bandwidth dan melacak berapa banyak sumber daya yang digunakan.

Paessler memiliki opsi PRTG lokal dan berbasis-cloud, dan paket bervariasi berdasarkan jumlah sensor, atau metrik pada setiap perangkat, yang ingin dipantau. PRTG mudah dipasang dan membantu mendeteksi kerentanan di jaringan. Terutama jika memiliki lokasi yang tersebar, penting untuk memiliki gambaran umum tentang

[10] Brett Knight, Best SNMP Monitoring Software & Tools for Windows & Linux of 2020, https://softwareportal.com/snmp-monitoring-tools-for-windows-and-linux/ diakses pada 4 Juli 2024

pemanfaatan rute WAN agar dapat bertindak cepat jika terjadi pemanfaatan jaringan yang tinggi. Dengan cara ini, dapat mencegah waktu habis atau bahkan waktu henti jaringan yang lengkap. Sensor Lalu Lintas memantau lalu lintas pada perangkat menggunakan SNMP dan dapat membuatnya pada perangkat yang menyediakan data lalu lintas. PRTG membuat satu sensor lalu lintas untuk setiap port individual. Terdapat fitur utama pada PRTG:

a. Pengukuran lalu lintas dan konsumsi bandwidth
b. Kemudahan penyebaran multi-situs
c. Dapat disesuaikan dengan kemungkinan untuk membuat sensor Anda sendiri
d. Pengaturan di semua perangkat: desktop, laptop, VM
e. Pemantauan jaringan modern yang mempertimbangkan IoT
f. Akses dari ponsel Anda ke solusi pemantauan Anda melalui aplikasi iOS dan Android
g. Cukup pantau aplikasi dan layanan: cadangan, server email, database, layanan dan keamanan Windows, aplikasi web
h. Sistem notifikasi yang efektif (eskalasi, notifikasi berkelompok, email, SMS, push ...)



[11]

Gambar 2.12 PRTG *Network Monitor*

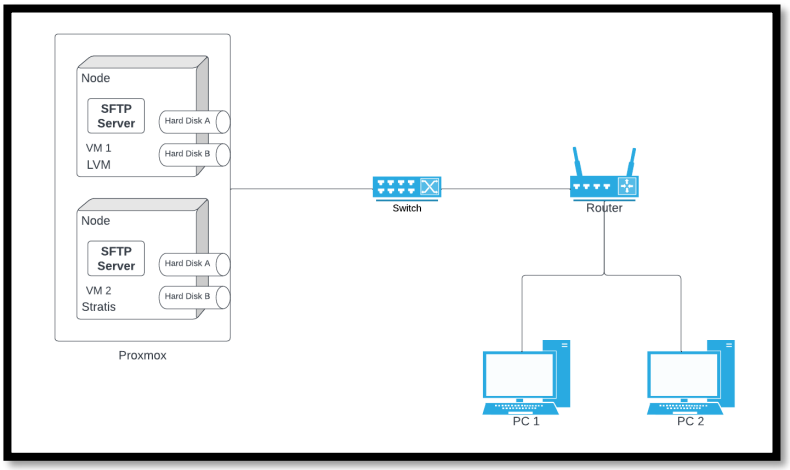
Sumber : (Shauli Zacks, 2024)

[11] Shauli Zacks, PRTG Manual: General Layout, https://www.paessler.com/manuals/prtg/general_layout_enterprise_console diakses pada 4 Juli 2024

BAB III
PERANCANGAN

3.1 Topologi

Berikut adalah perancangan topologi jaringan yang akan dibuat, beserta tabel pengalamatan yang akan diimplementasikan dalam proyek akhir ini.



Gambar 3.1 Rancangan Topologi

Rancangan topologi di atas ini menjelaskan bahwa terdapat sebuah PC Server, yang mana di dalamnya dibuat 2 node yaitu LVM dan Stratis dengan Sistem Operasi *CentOS 9*. Terdapat 1 *hard disk* fisik dan pada tiap node memiliki 2 buah *hard disk*. Masing – masing node akan dibangun dengan menggunakan *tools* Proxmox yang diinstal pada masing-masing node nya dengan menggunakan IP public sehingga dapat berjalan dan dapat diakses selama terhubung dengan internet dan terdapat SFTP *Server* yang digunakan untuk men*transfer* file antar komputer di jaringan. PC ini kemudian akan terhubung ke *Switch, Router* dan 2 PC Client. PC Client disini digunakan untuk mengkonfigurasi dalam pengujian dan mengakses layanan *hard disk*. Pada topologi ini, LVM dan Stratis akan dikumpulkan menjadi satu

Storage Pool untuk mengetahui kinerja dari Stratis dan LVM sehingga dapat membuktikan teknologi mana yang kinerjanya lebih baik untuk manajemen *file system* ini.

3.2 Skenario

Sebuah PC Server mempunyai 2 node yaitu LVM dan Stratis yang masing masing node akan dibangun dengan menggunakan *tools* Proxmox. Dengan menggunakan kedua teknologi tersebut, dapat menggabungkan dua *hard disk* dalam satu manajemen menggunakan manajemen *file system*. *File system* akan mengatur sendiri posisi *file* tersebut dimana ia akan tersusun dengan rata dan terstruktur, akan menambah *space* dan memudahkan untuk membuat partisi. Teknologi tersebut dapat memberikan kesederhanaan konseptual dari pengelolaan volume sistem *file*, serta integrasi dengan manajemen penyimpanan tingkat tinggi kerangka kerja yang akan dikumpulkan menjadi 1 *Storage Pool* untuk melakukan perbandingan pada pengujian kinerja *File system* dan *Storage Pool*. Hal ini dilakukan untuk mengetahui kinerja dari Stratis dan LVM, sehingga dapat membuktikan teknologi mana yang paling tepat untuk manajemen *file system* ini.

Pada skenario management *Storage Pool* menggunakan LVM dan Stratis, pertama-tama pengguna harus mempersiapkan sistem operasi *Linux* yang akan digunakan dan memastikan ada beberapa *disk* atau *partition* yang tersedia untuk digunakan sebagai ruang penyimpanan. Selanjutnya, pengguna dapat membuat *Physical Volume* pada partisi menggunakan perintah pvcreate dan membuat Volume Group pada *Physical Volume* tersebut menggunakan perintah vgcreate.

Setelah itu, pengguna dapat membuat Logical Volume pada Volume Group menggunakan perintah lvcreate dan membuat *file system* pada Logical Volume tersebut menggunakan perintah mkfs. Logical Volume dapat di-mount ke dalam sistem menggunakan perintah mount. Untuk menggunakan Stratis, pengguna dapat membuat *pool* Stratis menggunakan perintah `stratis pool create` dan membuat Stratis *file system* pada *pool* tersebut

menggunakan perintah `stratis file system create`. *File system* dapat di-mount ke dalam sistem menggunakan perintah mount.

Teknologi LVM dan Stratis memungkinkan pengguna untuk mengelola *Storage Pool* secara fleksibel dan efisien. Logical Volume pada LVM dapat digunakan untuk mengatur partisi secara dinamis, sedangkan Stratis dapat digunakan untuk membuat *pool* penyimpanan baru dengan mudah.

3.3 Skenario IP

Berikut adalah Skenario IP yang akan digunakan dalam penelitian ini:

Tabel 3.1 Skenario IP

Perangkat	Interface	IP Address	Subnet Mask	Software
SRV	Eth0	IP PUBLIC PCR		Proxmox
VM 1	Eth0			Stratis
VM 2	Eth0			LVM
PC CLIENT		IP PRIVATE		

3.4 Kebutuhan Perangkat

1) Proxmox

Tabel 3.2 Kebutuhan Perangkat Server

Spesifikasi	
Processor	4 core
Hard disk	1 TB
Memory	8 GB
Operating System	*Ubuntu* 22.04
Software	Proxmox

2) PC-Client

Tabel 3.3 Kebutuhan PC-Client

Spesifikasi	
Processor	Intel Core i5
Hard disk	1 TB
Memory	8 GB
Software	Windows 11
IP Address	IP Private

3) Node Stratis

Tabel 3.4 Kebutuhan Stratis

Spesifikasi	
Processor	2 core
Hard disk	Virtual SSI 50 GB
Memory	4 GB
Operating System	*CentOS 9*

4) Node LVM

Tabel 3.5 Kebutuhan LVM

Spesifikasi	
Processor	2 core
Hard disk	Virtual SSI 50 GB
Memory	4 GB
Operating System	*CentOS 9*

5) Software

Tabel 3.6 Kebutuhan Software

Spesifikasi	
Proxmox	7.4
Stratis	3.0
LVM	2.4
Operating System	*CentOS 9*

3.5 Metode Pengujian

Metode Pengujian Metode pengujian yang dilakukan pada proyek akhir ini yaitu dengan pengujian pada *disk* untuk melihat bagaimana kinerja *file system.*

3.5.1 Pengujian *Storage Pool*

Pengujian ini dilakukan pada *disk* untuk melihat hasil penggabungan *Storage Pool*. Pengujian yang dilakukan adalah pengujian ukuran file yang dapat melihat hasil *storage* yang telah digabungkan dengan menggunakan *file system* Stratis dan LVM. Pada pengujian ini bertujuan untuk menggabungkan 2 *disk* dari 1 node untuk membentuk sebuah *storage* dengan dilakukan penggabungan antara *hard disk*, kemudian dilakukan konfigurasi untuk membuat sebuah *pool*. Hasil penggabungan dilihat dengan menggunakan *tools* Linux yaitu `DF` (*Disk File system)*. Selain itu dilakukan pengukuran tingkat keberhasilan pada layanan *storage* dengan ukuran folder yang sama yaitu sebesar 100 MB, 200 MB, 300 MB, 400 MB dan 500 MB.

3.5.2 Pengujian kinerja *File System*

Parameter pengujian yang digunakan pada penelitian ini adalah Kinerja pada *file system*. Pengujian ini dilakukan pada *disk* untuk melihat kinerjansi Stratis dan LVM, mendata kecepatan rata-rata dalam melakukan *upload* dan *download* file pada Stratis dan LVM. Selain itu dilakukan pengukuran tingkat keberhasilan pada layanan *storage.* Setelah berhasil maka dilakukan pengujian untuk

mengukur seberapa besar kecepatan *transfer file* dengan menggunakan layanan *storage* yang dipakai yaitu SFTP Server. Pada masing-masing file system yang akan dilakukan write/read file system dengan ukuran 100 MB, 200 MB, 300 MB, 400 MB, dan 500 MB yang masing-masing dilakukan sebanyak 10 kali. Parameter yang diukur pada pengujian ini adalah kinerja pada *storage*nya kemudian difile kecepatan dalam men*transfer* file masing-masing dan dicari rata-rata kecepatan *transfer* file dari keseluruhan file yang di*transfer*. Pengujian ini dilakukan pada *disk* dengan menggunakan *tools* `dd` (File Duplicator) untuk mengambil file kecepatan tulis *disk* dan *tools* `iostat` untuk membandingkan write/read file pada *disk*.

BAB IV
PENGUJIAN DAN ANALISIS

4.1 Hasil Perancangan

Pengujian yang dilakukan pada Proyek Akhir ini adalah dengan melihat hasil *storage* yang telah digabungkan pada Stratis dan LVM dengan *environment* yang sama menggunakan perintah `df -Th`, aktifitas *transfer* file dan write/read file, kondisi CPU dan *memory* sebelum dan sesudah *storage* digabungkan. Selama kegiatan *transfer* file berlangsung maka dilakukan pengambilan data yaitu untuk mengetahui nilai kecepatan rata-rata *transfer* data dan penggunaan *disk* menggunakan *tools* `dd` dan `df`.

4.1.1 Kondisi environment yang sama

Pastikan bahwa sistem operasi yang digunakan menggunakan *CentOS 9.*

```
[sarah@lvm ~]$ more /etc/os-release
NAME="CentOS Stream"
VERSION="9"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="9"
PLATFORM_ID="platform:el9"
PRETTY_NAME="CentOS Stream 9"
ANSI_COLOR="0;31"
LOGO="fedora-logo-icon"
CPE_NAME="cpe:/o:centos:centos:9"
HOME_URL="https://centos.org/"
BUG_REPORT_URL="https://issues.redhat.com/"
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux 9"
REDHAT_SUPPORT_PRODUCT_VERSION="CentOS Stream"
```

Gambar 4.1 OS pada LVM

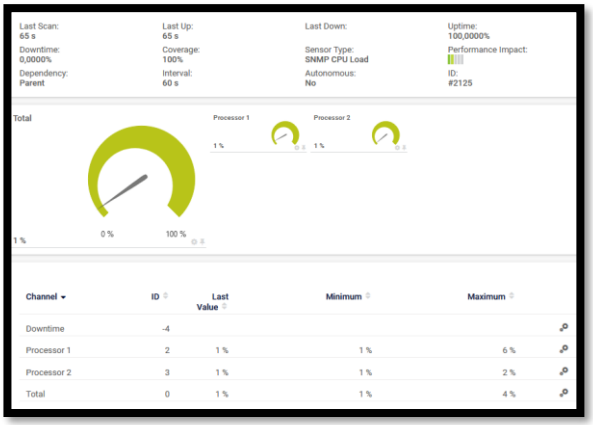
```
[sarah@stratis ~]$ more /etc/os-release
NAME="CentOS Stream"
VERSION="9"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="9"
PLATFORM_ID="platform:el9"
PRETTY_NAME="CentOS Stream 9"
ANSI_COLOR="0;31"
LOGO="fedora-logo-icon"
CPE_NAME="cpe:/o:centos:centos:9"
HOME_URL="https://centos.org/"
BUG_REPORT_URL="https://issues.redhat.com/"
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux 9"
REDHAT_SUPPORT_PRODUCT_VERSION="CentOS Stream"
```

Gambar 4.2 OS pada Stratis

Perintah `more /etc/os-release` adalah cara sederhana untuk menampilkan informasi tentang distribusi Linux yang berjalan pada sistem, menggunakan utilitas `more` untuk navigasi yang mudah. Dapat dipastikan bahwa pada LVM dan Stratis menggunakan CentOS Linux versi 9.

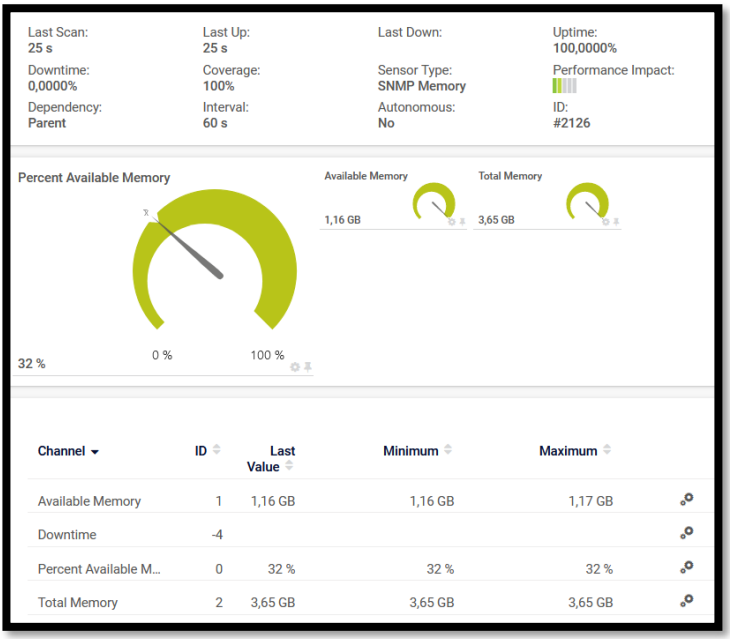
4.1.2 Kondisi CPU dan *Memory* sebelum dan sesudah *Storage Pool*

a. Kondisi CPU dan *Memory* LVM sebelum *Storage Pool*



Gambar 4.3 *Overview* CPU Load LVM sebelum *Storage Pool*

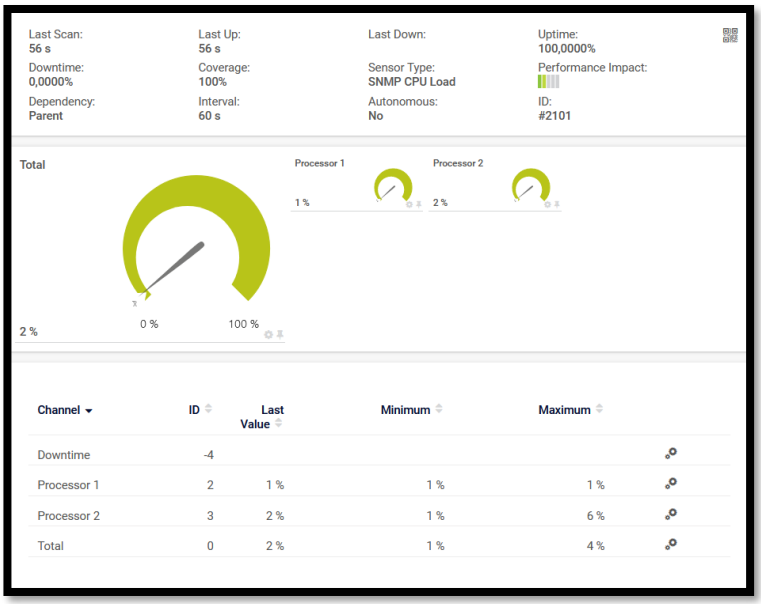
Gambar 4.3 tersebut menunjukkan data beban CPU dari sistem monitoring menggunakan sensor tipe SNMP CPU Load pada LVM. Berdasarkan data yang ditampilkan, sistem memiliki uptime 100% tanpa adanya downtime yang tercatat, yang menunjukkan stabilitas tinggi dan tidak adanya gangguan atau restart dalam periode monitoring. Total beban CPU adalah 1%, dengan masing-masing processor (Processor 1 dan Processor 2) menunjukkan beban yang sama sebesar 1%. Ini mengindikasikan bahwa sistem berada dalam kondisi idle atau tidak menjalankan tugas berat. Beban CPU terdistribusi merata antara kedua processor, menunjukkan tidak ada ketidakseimbangan beban kerja. Meskipun Processor 1 memiliki variasi beban yang sedikit lebih tinggi (maksimum 6%) dibandingkan Processor 2 (maksimum 2%), beban keseluruhan tetap rendah. Kesimpulannya, sebelum penggabungan *Storage Pool*, sistem berada dalam kondisi optimal dengan beban CPU yang sangat rendah dan stabilitas tinggi, menunjukkan bahwa sistem mampu menangani operasi tanpa menimbulkan tekanan berlebihan pada processor. Secara keseluruhan, sistem menunjukkan kinerja yang sangat stabil dan efisien dengan beban CPU yang minimal sebelum penggabungan *Storage Pool*, memastikan bahwa sistem dapat melanjutkan operasi dengan kinerja optimal tanpa menimbulkan tekanan berlebih pada CPU.



Gambar 4.4 *Overview Memory* LVM sebelum *Storage Pool*

Gambar 4.4 ini menunjukkan data penggunaan memori dari sistem monitoring menggunakan sensor tipe SNMP *Memory* pada LVM sebelum penggabungan *Storage Pool*. Sistem memiliki total memori sebesar 3,65 GB, dengan memori yang tersedia sebesar 1,16 GB, atau sekitar 32% dari total memori. Ini berarti sistem menggunakan sekitar 68% dari total memori yang tersedia. Data menunjukkan bahwa sistem sangat stabil dengan uptime 100% dan tidak ada downtime yang tercatat. Penggunaan memori juga konsisten, berkisar antara 1,16 GB hingga 1,17 GB, menunjukkan stabilitas tanpa perubahan yang signifikan. Secara keseluruhan, sistem memiliki ketersediaan memori yang cukup dan stabilitas tinggi, mengindikasikan bahwa sistem berada dalam kondisi optimal dan siap untuk melanjutkan operasi lebih lanjut, termasuk penggabungan *Storage Pool*, tanpa risiko kehabisan memori atau gangguan signifikan.

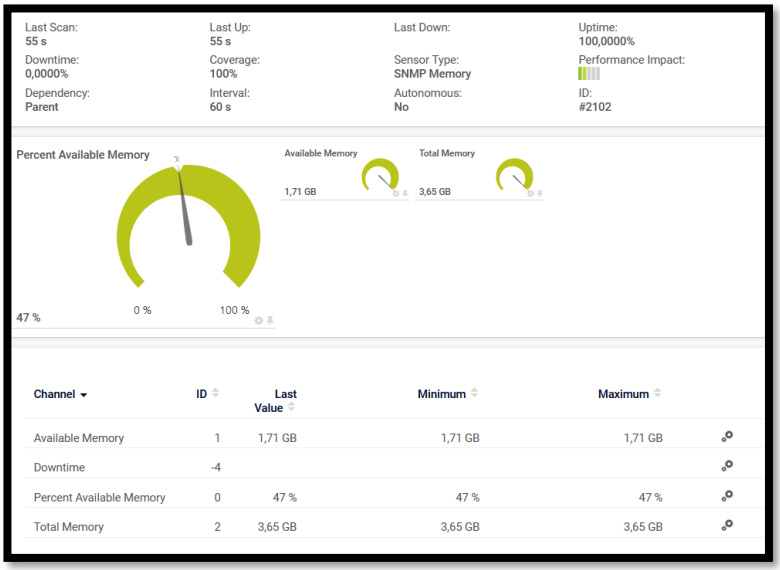
b. Kondisi CPU dan *Memory* Stratis sebelum *Storage Pool*



Gambar 4.5 *Overview* CPU Load Stratis sebelum *Storage Pool*

Gambar 4.5 tersebut menunjukkan data beban CPU dari sistem monitoring menggunakan sensor tipe SNMP CPU Load pada Stratis sebelum penggabungan *Storage Pool*. Berdasarkan data yang ditampilkan, sistem memiliki uptime 100% tanpa adanya downtime yang tercatat, yang menunjukkan stabilitas tinggi dan tidak adanya gangguan atau restart dalam periode monitoring. Total beban CPU adalah 2%, Processor 1 menunjukkan beban sebesar 1% dan Processor 2 menunjukkan beban sebesar 2%. Ini mengindikasikan bahwa sistem berada dalam kondisi idle atau tidak menjalankan tugas berat. Beban CPU terdistribusi merata antara kedua processor, menunjukkan tidak ada ketidakseimbangan beban kerja. Processor 1 memiliki variasi beban yang sedikit lebih rendah (maksimum 1%) dibandingkan Processor 2 (maksimum 6%). Kesimpulannya, sebelum penggabungan *Storage Pool*, sistem berada dalam kondisi optimal

dengan beban CPU yang sangat rendah dan stabilitas tinggi, menunjukkan bahwa sistem mampu menangani operasi tanpa menimbulkan tekanan berlebihan pada processor. Secara keseluruhan, sistem menunjukkan kinerja yang sangat stabil dan efisien dengan beban CPU yang minimal sebelum penggabungan *Storage Pool*, memastikan bahwa sistem dapat melanjutkan operasi dengan kinerja optimal tanpa menimbulkan tekanan berlebih pada CPU.

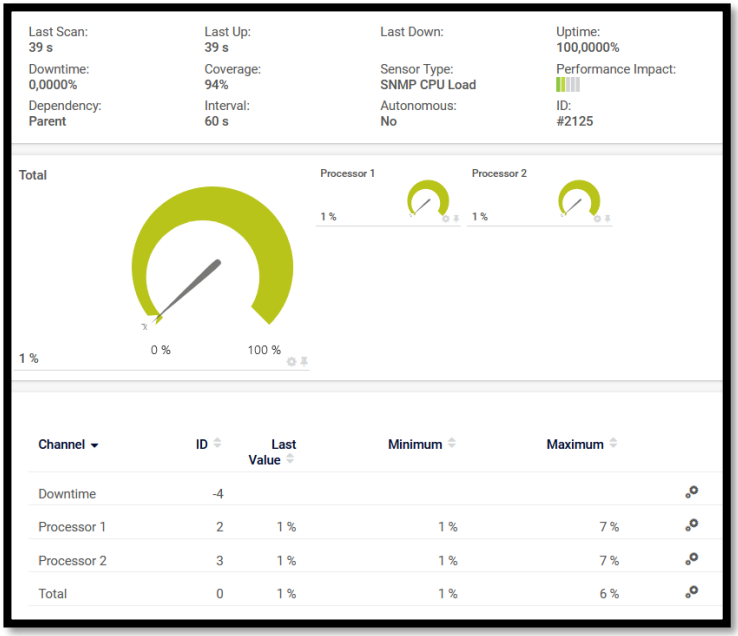


Gambar 4.6 *Overview Memory* Stratis sebelum *Storage Pool*

Gambar 4.6 ini menunjukkan data penggunaan memori dari sistem monitoring menggunakan sensor tipe SNMP *Memory* pada Stratis sebelum penggabungan *Storage Pool*. Sistem memiliki total memori sebesar 3,65 GB, dengan memori yang tersedia sebesar 1,71 GB, atau sekitar 47% dari total memori. Ini berarti sistem menggunakan sekitar 53% dari total memori yang tersedia. Data menunjukkan bahwa sistem sangat stabil dengan uptime 100% dan tidak ada downtime yang tercatat. Penggunaan memori juga konsisten, berkisar antara 1,71 GB, menunjukkan stabilitas tanpa perubahan yang

signifikan. Secara keseluruhan, sistem memiliki ketersediaan memori yang cukup dan stabilitas tinggi, mengindikasikan bahwa sistem berada dalam kondisi optimal dan siap untuk melanjutkan operasi lebih lanjut, termasuk penggabungan *Storage Pool*, tanpa risiko kehabisan memori atau gangguan signifikan.

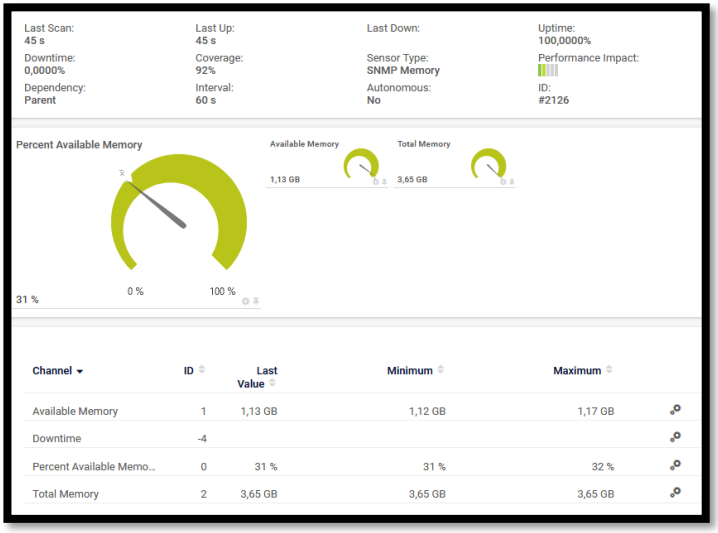
c. Kondisi CPU dan *Memory* LVM setelah *Storage Pool*



Gambar 4.7 *Overview* CPU Load LVM setelah *Storage Pool*

Gambar 4.7 tersebut menunjukkan data beban CPU dari sistem monitoring menggunakan sensor tipe SNMP CPU Load pada LVM setelah penggabungan *Storage Pool*. Berdasarkan data yang ditampilkan, sistem memiliki uptime 100% tanpa adanya downtime yang tercatat, yang menunjukkan stabilitas tinggi dan tidak adanya gangguan atau restart dalam periode monitoring. Total beban CPU adalah 1%, Processor 1 menunjukkan beban sebesar 1% dan Processor 2 menunjukkan beban sebesar 1%. Ini mengindikasikan bahwa sistem

berada dalam kondisi idle atau tidak menjalankan tugas berat. Beban CPU terdistribusi merata antara kedua processor, menunjukkan tidak ada ketidakseimbangan beban kerja. Processor 1 memiliki variasi beban yang sedikit lebih rendah (maksimum 1%) dibandingkan Processor 2 (maksimum 6%), maka beban keseluruhan tetap rendah. Kesimpulannya, sebelum penggabungan *Storage Pool*, sistem berada dalam kondisi optimal dengan beban CPU yang sangat rendah dan stabilitas tinggi, menunjukkan bahwa sistem mampu menangani operasi tanpa menimbulkan tekanan berlebihan pada processor. Secara keseluruhan, sistem menunjukkan kinerja yang sangat stabil dan efisien dengan beban CPU yang minimal setelah penggabungan *Storage Pool*, memastikan bahwa sistem dapat melanjutkan operasi dengan kinerja optimal tanpa menimbulkan tekanan berlebih pada CPU.

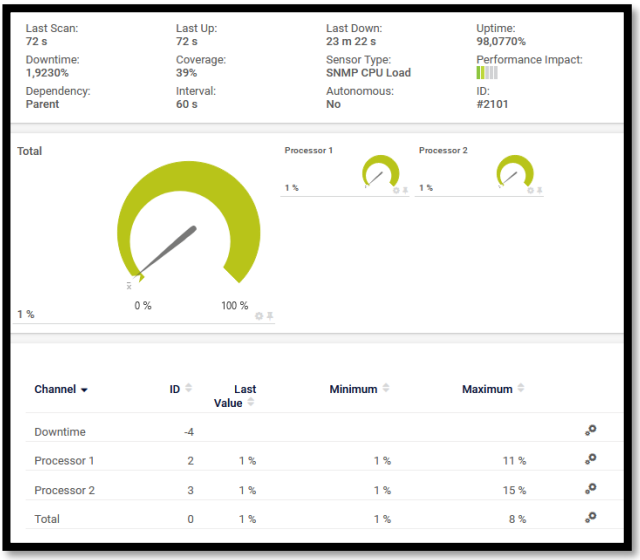


Gambar 4.8 *Overview Memory* LVM setelah *Storage Pool*

Gambar 4.8 ini menunjukkan data penggunaan memori dari sistem monitoring menggunakan sensor tipe SNMP *Memory* pada LVM setelah penggabungan *Storage Pool*. Sistem memiliki total

memori sebesar 3,65 GB, dengan memori yang tersedia sebesar 1,13 GB, atau sekitar 31% dari total memori. Ini berarti sistem menggunakan sekitar 69% dari total memori yang tersedia. Data menunjukkan bahwa sistem sangat stabil dengan uptime 100% dan tidak ada downtime yang tercatat. Penggunaan memori juga konsisten, berkisar antara 1,13 GB hingga 1,17 GB, menunjukkan stabilitas tanpa perubahan yang signifikan. Secara keseluruhan, sistem memiliki ketersediaan memori yang cukup dan stabilitas tinggi, mengindikasikan bahwa sistem berada dalam kondisi optimal dan siap untuk melanjutkan operasi lebih lanjut, termasuk penggabungan *Storage Pool*, tanpa risiko kehabisan memori atau gangguan signifikan.

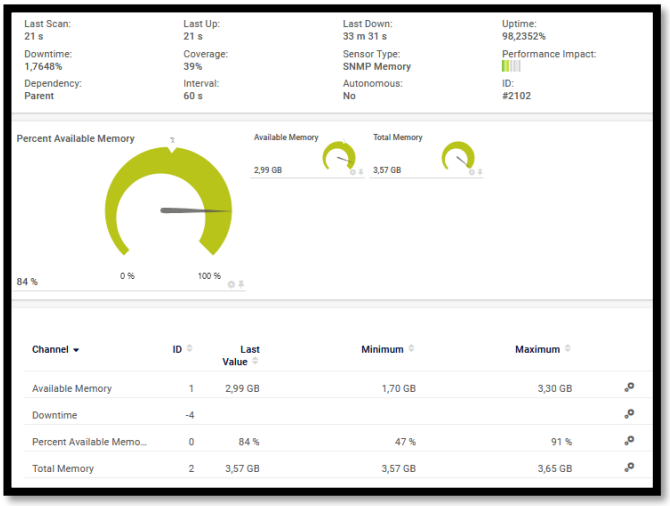
d. Kondisi CPU dan *Memory* Stratis setelah *Storage Pool*



Gambar 4.9 *Overview* CPU Load Stratis setelah *Storage Pool*

Gambar 4.9 tersebut menunjukkan data beban CPU dari sistem monitoring menggunakan sensor tipe SNMP CPU Load pada Stratis setelah penggabungan *Storage Pool*. Berdasarkan data yang

ditampilkan, sistem memiliki uptime 98,0770% dan 1,9230% downtime yang tercatat, yang menunjukkan stabilitas tinggi dan tidak adanya gangguan atau restart dalam periode monitoring. Total beban CPU adalah 1%, dengan masing-masing processor (Processor 1 dan Processor 2) menunjukkan beban yang sama sebesar 1%. Ini mengindikasikan bahwa sistem berada dalam kondisi idle atau tidak menjalankan tugas berat. Beban CPU terdistribusi merata antara kedua processor, menunjukkan tidak ada ketidakseimbangan beban kerja. Processor 1 memiliki variasi beban yang sedikit lebih rendah (maksimum 11%) dibandingkan Processor 2 (maksimum 15%), maka beban keseluruhan tetap rendah. Kesimpulannya, sebelum setelah penggabungan *Storage Pool*, sistem berada dalam kondisi optimal dengan beban CPU yang sangat rendah dan stabilitas tinggi, menunjukkan bahwa sistem mampu menangani operasi tanpa menimbulkan tekanan berlebihan pada processor. Secara keseluruhan, sistem menunjukkan kinerja yang sangat stabil dan efisien dengan beban CPU yang minimal setelah penggabungan *Storage Pool*, memastikan bahwa sistem dapat melanjutkan operasi dengan kinerja optimal tanpa menimbulkan tekanan berlebih pada CPU.



Gambar 4.10 *Overview Memory* Stratis setelah *Storage Pool*

Gambar 4.10 ini menunjukkan data penggunaan memori dari sistem monitoring menggunakan sensor tipe SNMP *Memory* pada Stratis setelah penggabungan *Storage Pool*. Sistem memiliki total memori sebesar 3,57 GB, dengan memori yang tersedia sebesar 2,99 GB, atau sekitar 84% dari total memori. Ini berarti sistem menggunakan sekitar 16% dari total memori yang tersedia. Data menunjukkan bahwa sistem sangat stabil dengan uptime 98,2352% dan downtime1,7648%. Penggunaan memori juga konsisten, berkisar antara 2,99 GB hingga 3,30 GB, menunjukkan stabilitas tanpa perubahan yang signifikan. Secara keseluruhan, sistem memiliki ketersediaan memori yang cukup dan stabilitas tinggi, mengindikasikan bahwa sistem berada dalam kondisi optimal dan siap untuk melanjutkan operasi lebih lanjut, termasuk penggabungan *Storage Pool*, tanpa risiko kehabisan memori atau gangguan signifikan.

4.1.3 Hasil salah satu perancangan *Storage Pool*

Gambar di bawah menunjukkan salah satu hasil perancangan *Storage Pool*. Dalam perancangan ini, terdapat dua perangkat, *Device* 1 dan *Device* 2 masing-masing dengan kapasitas penyimpanan sebesar 50 GB pada /dev/sdb dan /dev/sdc.

Gambar 4.11 *Storage Pool*

Berikut adalah penjelasan mengenai *Storage Pool* berdasarkan gambar tersebut:

a. *Device* 1 dan *Device* 2:

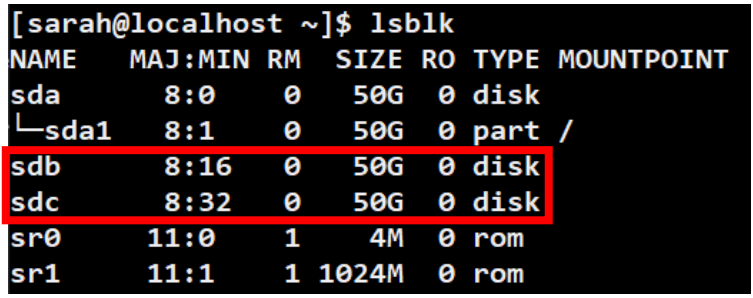
Terdapat dua perangkat yang masing-masing memiliki *disk* dengan ukuran 50 GB yang teridentifikasi sebagai `/dev/sdb`.

b. Physical Volumes (PV):
   Disk `/dev/sdb` pada kedua perangkat tersebut digunakan sebagai Physical Volumes dalam konfigurasi LVM.
   Dalam gambar, juga terlihat ada *disk* lain yang teridentifikasi sebagai `/dev/sdc`.
c. Volume Group (VG):
   Physical Volumes (`/dev/sdb` dan `/dev/sdc`) dari dua perangkat tersebut digabungkan ke dalam satu Volume Group yang diberi nama `VG 01`. Volume Group adalah kumpulan dari beberapa Physical Volumes yang bertindak sebagai *Storage Pool* tunggal.
d. Logical Volume (LV):
   Di dalam Volume Group `VG 01`, dibuat satu atau lebih Logical Volumes. Pada gambar ini, ada satu Logical Volume yang diberi nama `LV 01`. Logical Volume berfungsi sebagai ruang penyimpanan yang bisa digunakan oleh sistem operasi dan bisa diatur ukuran dan lokasinya tanpa terikat pada batasan fisik dari *disk*.

4.2 Pengujian

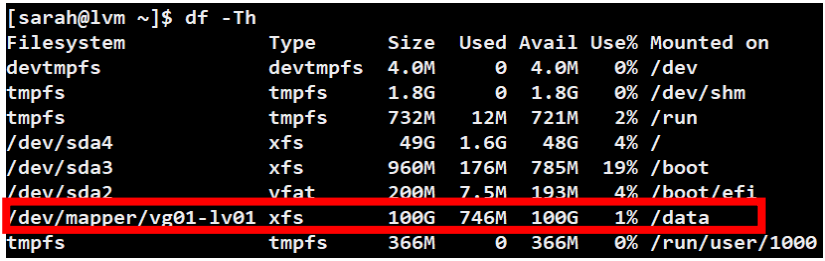
4.2.1 Pengujian *Storage Pool* pada Stratis dan LVM

Pengujian ini dilakukan untuk melihat cara pembagian atau pemetaan data antar node. Selain itu, pengujian ini dilakukan untuk melihat hasil *storage* yang telah digabungkan dengan menggunakan Stratis dan LVM dengan menggunakan perintah `df -Th`. `df` merupakan singkatan dari "*disk* free", perintah ini digunakan untuk menampilkan informasi tentang penggunaan *disk space*. `-Th` adalah opsi dari perintah `df`. Dalam hal ini, `-T` untuk menampilkan jenis dari masing-masing *file system*, sedangkan `-h` untuk menampilkan ukuran dalam format yang lebih mudah dibaca (misalnya, dalam bentuk gigabyte, megabyte, dan sebagainya).



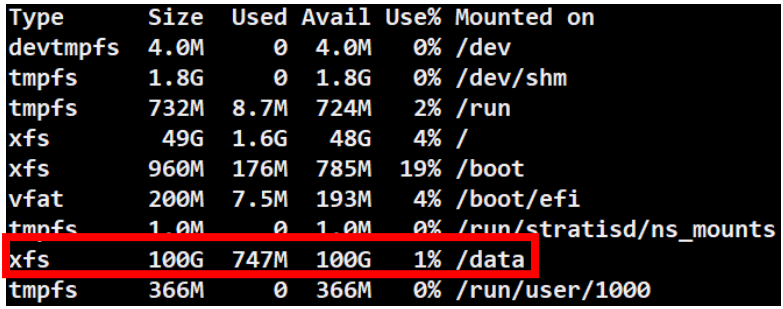
Gambar 4.12 Perangkat blok LVM sebelum digabung

Berdasarkan gambar di atas dapat dilihat kondisi ukuran perangkat blok `sdb` dan `sdc` masing - masing yaitu 50 GB sebelum digabung menggunakan LVM dan Stratis.



Gambar 4.13 LVM setelah digabung

Pada Gambar 4.13 dapat dilihat bahwa output `df -Th` pada LVM, ukuran *file system* `/dev/mapper/vg01-lv01` adalah 100 GB dengan 746M digunakan dan sisa yang tersedia juga menunjukkan 100 GB. Ini terjadi karena sistem file `XFS` yang digunakan oleh logical volume ini tidak mempengaruhi total kapasitas dari *Storage Pool* yang dibuat. Ketika dua *disk* masing-masing berukuran 50 GB digabungkan, total kapasitas *Storage Pool* dalam *volume group* adalah 100 GB. Logical volume yang dibuat dari *volume group* ini akan memiliki kapasitas total yang sama dengan jumlah kapasitas physical volumes yang tergabung di dalamnya.



Gambar 4.14 Stratis setelah digabung

Pada Gambar 4.14 dapat dilihat bahwa *output* `df -Th` pada Stratis menunjukkan bahwa sistem telah menggabungkan dua *disk* (`sdb` dan `sdc`) masing-masing berkapasitas 50 GB menjadi satu logical volume dengan kapasitas total 100 GB menggunakan Stratis. Logical volume ini menggunakan sistem file `XFS` dan terpasang di `/data`. Kapasitas ini terlihat di output sebagai `100 GB`, dengan penggunaan sekitar 747M, menunjukkan bahwa gabungan *storage* bekerja sesuai harapan. Ketika Stratis menggabungkan dua *disk* masing-masing berukuran 50 GB ke dalam satu *pool*, hasilnya adalah *pool* dengan total kapasitas 100 GB. *File system* yang dibuat dalam *pool* ini akan mencerminkan total kapasitas yang sama.

Dari kedua hasil penggabungan yang telah dilakukan, pada Gambar 4.13 dan Gambar 4.14 dapat dilihat bahwa LVM dan Stratis sama sama merupakan server yang dapat menggabungkan *storage* dengan kapasitas 100 GB menggunakan *file system* XFS.

File system yang digunakan, seperti XFS dalam kasus ini, tidak mempengaruhi total kapasitas dari *Storage Pool* yang dibuat. XFS hanya berfungsi sebagai sistem file yang mengatur cara data disimpan dan diakses dalam logical volume atau *pool*.

LVM dan Stratis keduanya menggabungkan kapasitas fisik dari dua *disk* (`sdb` dan `sdc`) menjadi satu unit penyimpanan logis yang lebih besar. Total kapasitas adalah hasil langsung dari penjumlahan kapasitas *disk* fisik yang tergabung. Maka, hasil *Storage Pool* dari dua *disk* (`sdb` dan `sdc`) yang masing-masing memiliki ukuran 50 GB akan menghasilkan size total 100 GB setelah digabungkan dalam sebuah *Storage Pool*, baik menggunakan LVM

maupun Stratis, karena cara kerja kedua teknologi tersebut dalam mengelola dan menggabungkan penyimpanan adalah serupa.

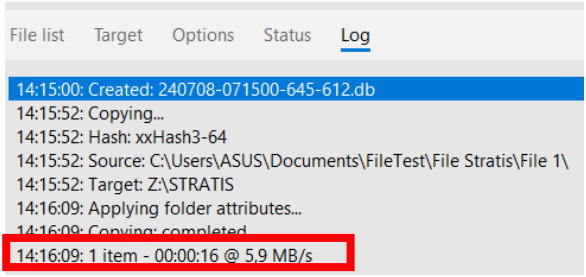
4.2.2 Pengujian *Transfer File* pada Stratis dan LVM

Pengujian ini dilakukan pada *disk* untuk melihat kinerjansi Stratis dan LVM, mendata kecepatan rata-rata dalam melakukan *upload* dan *download* file pada Stratis dan LVM tersebut. Selain itu dilakukan pengukuran tingkat keberhasilan pada layanan *storage* dengan ukuran folder yang sama yaitu sebesar 100 MB, 200 MB, 300 MB, 400 MB dan 500 MB yang dilakukan masing – masing selama 10 kali pengujian secara parallel. Setelah berhasil maka dilakukan pengujian untuk mengukur seberapa besar kecepatan *transfer file*. Untuk melakukan pengujian tersebut digunakan layanan *storage* dengan mengakses layanan *secure copy* menggunakan IP Address pada LVM dan IP Address pada Stratis.

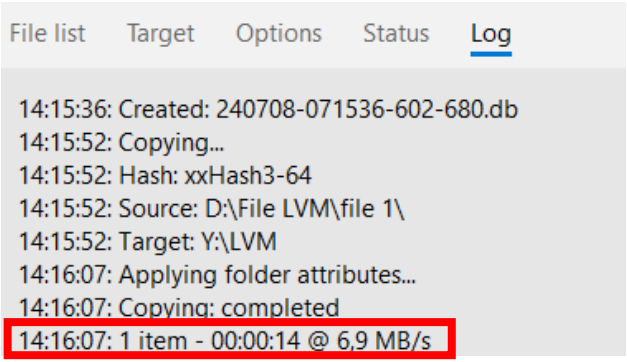
1. Pengujian *Upload* File

A. *Upload* File 100 MB/s

Pada Gambar 4.15 dan 4.16 merupakan tampilan percoban pertama Teracopy pada pengujian *upload* file dengan ukuran 100 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *upload* file dengan ukuran 100 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 6,9 MB/s selama 14 detik sedangkan Stratis sebesar 5,9 MB/s selama 16 detik.



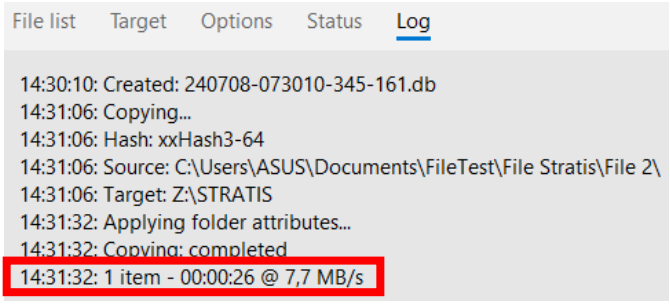
Gambar 4.15 *Upload* file 100 MB/s pada Stratis



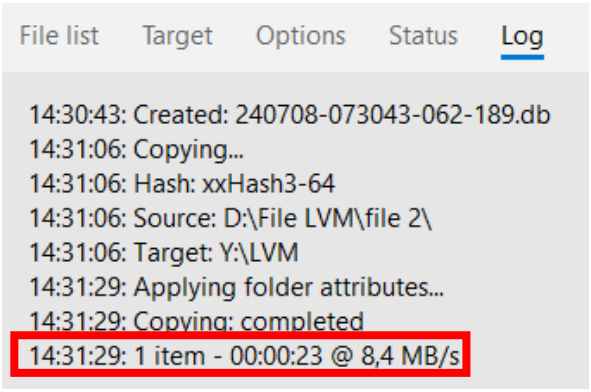
Gambar 4.16 *Upload* file 100 MB/s pada LVM

B. *Upload* File 200 MB/s

Pada Gambar 4.17 dan 4.18 merupakan tampilan percoban pertama Teracopy pada pengujian *upload* file dengan ukuran 200 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *upload* file dengan ukuran 200 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 8,4 MB/s selama 23 detik sedangkan Stratis sebesar 7,7 MB/s selama 26 detik.



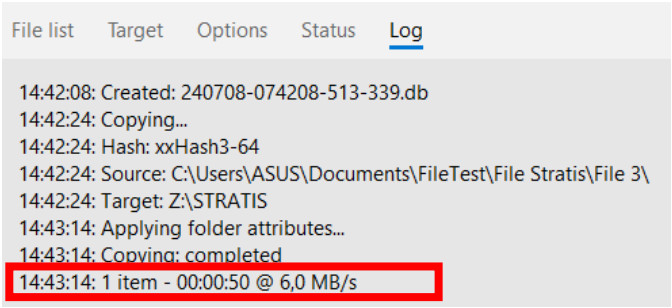
Gambar 4.17 *Upload* file 200 MB/s pada Stratis



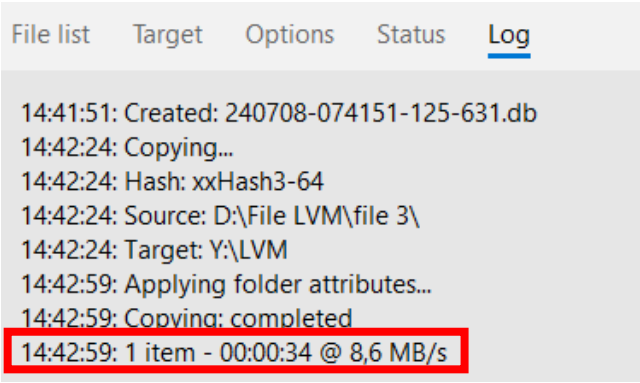
Gambar 4.18 *Upload* file 200 MB/s pada LVM

C. *Upload* File 300 MB/s

Pada Gambar 4.19 dan 4.20 merupakan tampilan percoban pertama Teracopy pada pengujian *upload* file dengan ukuran 300 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *upload* file dengan ukuran 300 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 8,6 MB/s selama 34 detik sedangkan Stratis sebesar 6,0 MB/s selama 50 detik.



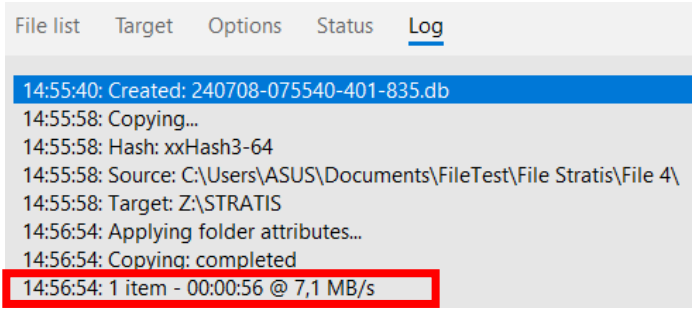
Gambar 4.19 *Upload* file 300 MB/s pada Stratis



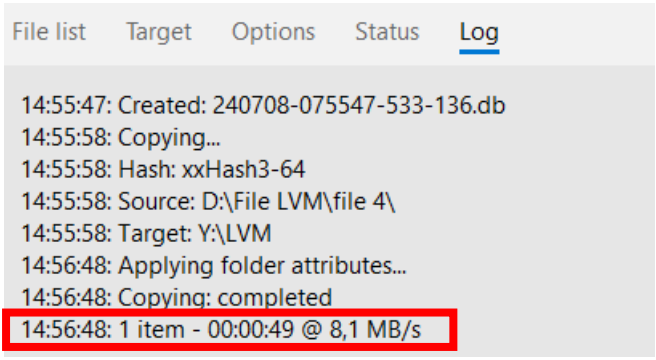
Gambar 4.20 *Upload* file 300 MB/s pada LVM

D. *Upload* File 400 MB/s

Pada Gambar 4.21 dan 4.22 merupakan tampilan percoban pertama Teracopy pada pengujian *upload* file dengan ukuran 400 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *upload* file dengan ukuran 400 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 8,1 MB/s selama 49 detik sedangkan Stratis sebesar 7,1 MB/s selama 56 detik.



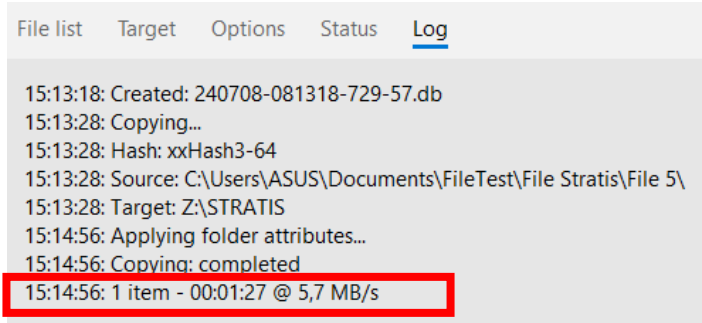
Gambar 4.21 *Upload* file 400 MB/s pada Stratis



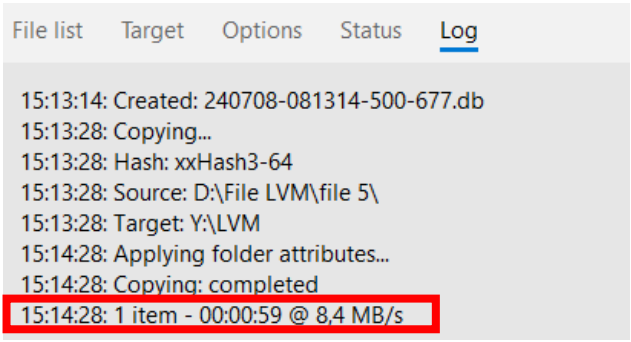
Gambar 4.22 *Upload* file 400 MB/s pada LVM

E. *Upload* File 500 MB/s

Pada Gambar 4.23 dan 4.24 merupakan tampilan percoban pertama Teracopy pada pengujian *upload* file dengan ukuran 500 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *upload* file dengan ukuran 500 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 8,4 MB/s selama 59 detik sedangkan Stratis sebesar 5,7 MB/s selama 1 menit 27 detik.



Gambar 4.23 *Upload* file 500 MB/s pada Stratis

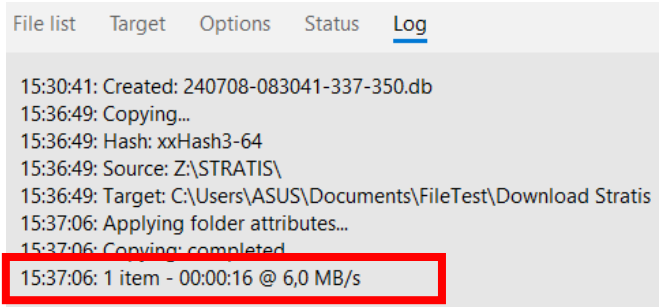


Gambar 4.24 *Upload* file 500 MB/s pada LVM

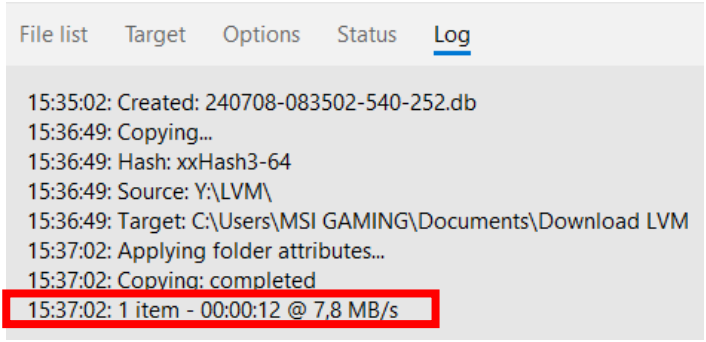
2. Pengujian *Download* File

A. *Download* File 100 MB/s

Pada Gambar 4.25 dan 4.26 merupakan tampilan percoban pertama Teracopy pada pengujian *download* file dengan ukuran 100 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *download* file dengan ukuran 100 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 7,8 MB/s selama 12 detik sedangkan Stratis sebesar 6,0 MB/s selama 16 detik.



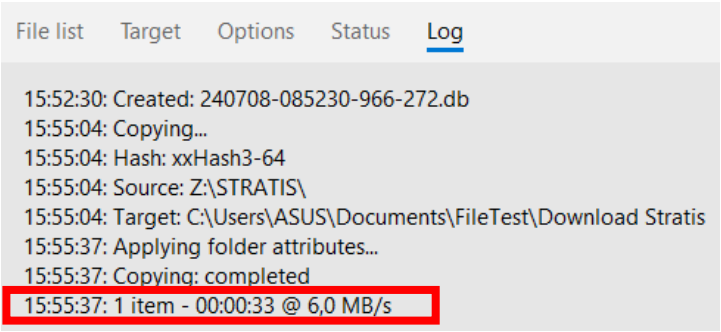
Gambar 4.25 *Download* file 100 MB/s pada Stratis



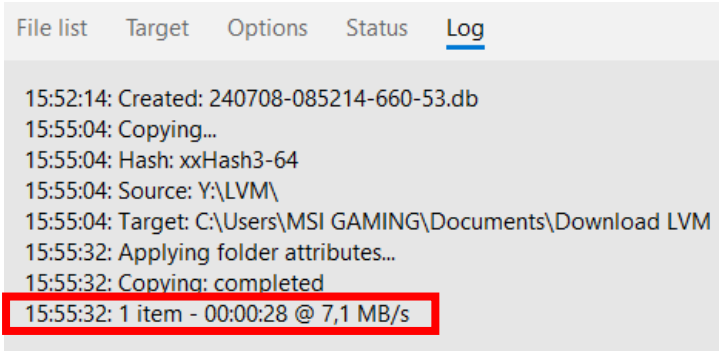
Gambar 4.26 *Download* file 100 MB/s pada LVM

B. *Download* File 200 MB/s

Pada Gambar 4.27 dan 4.28 merupakan tampilan percoban pertama Teracopy pada pengujian *download* file dengan ukuran 200 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *download* file dengan ukuran 200 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 7,1 MB/s selama 28 detik sedangkan Stratis sebesar 6,0 MB/s selama 33 detik.



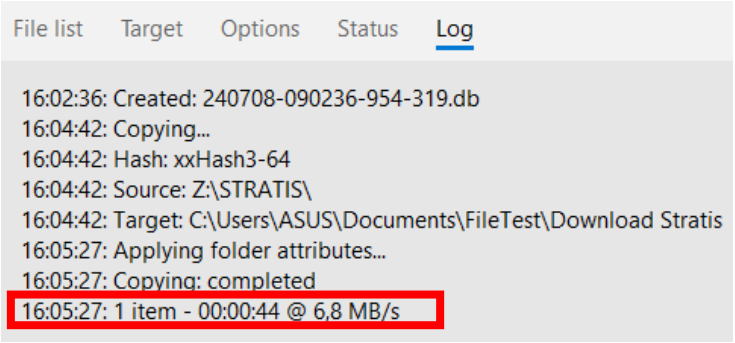
Gambar 4.27 *Download* file 200 MB/s pada Stratis



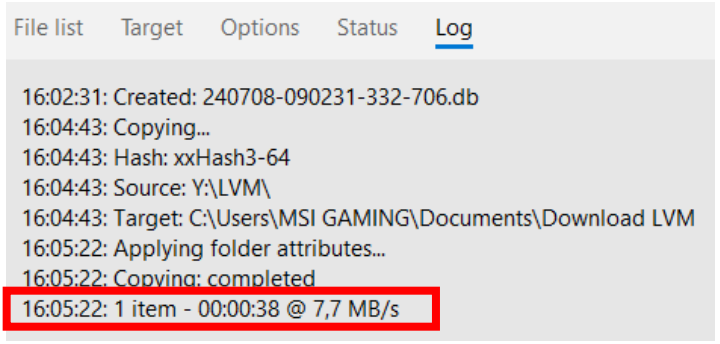
Gambar 4.28 *Download* file 200 MB/s pada LVM

C. *Download* File 300 MB/s

Pada Gambar 4.29 dan 4.30 merupakan tampilan percoban pertama Teracopy pada pengujian *download* file dengan ukuran 300 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *download* file dengan ukuran 300 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 7,7 MB/s selama 38 detik sedangkan Stratis sebesar 6,8 MB/s selama 44 detik.



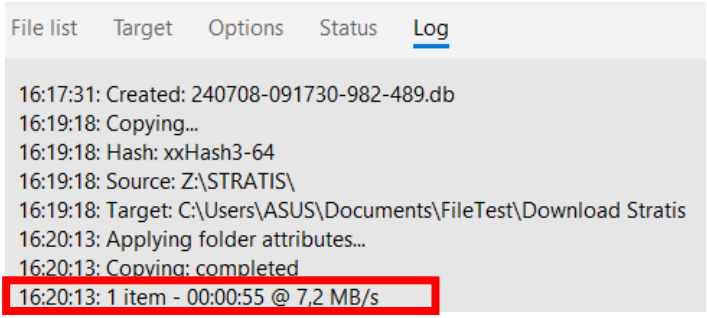
Gambar 4.29 *Download* file 300 MB/s pada Stratis



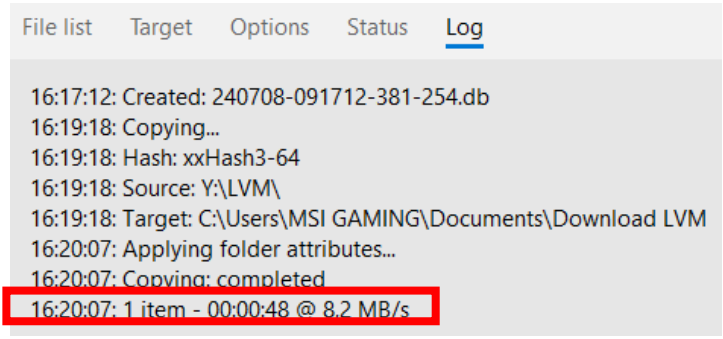
Gambar 4.30 *Download* file 300 MB/s pada LVM

D. *Download* File 400 MB/s

Pada Gambar 4.31 dan 4.32 merupakan tampilan percoban pertama Teracopy pada pengujian *download* file dengan ukuran 400 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *download* file dengan ukuran 400 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 8,2 MB/s selama 48 detik sedangkan Stratis sebesar 7,2 MB/s selama 55 detik.



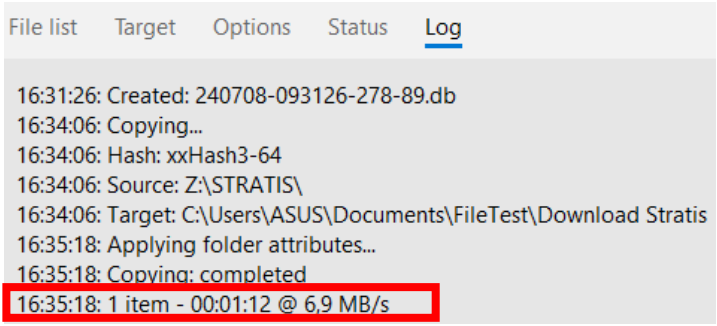
Gambar 4.31 *Download* file 400 MB/s pada Stratis



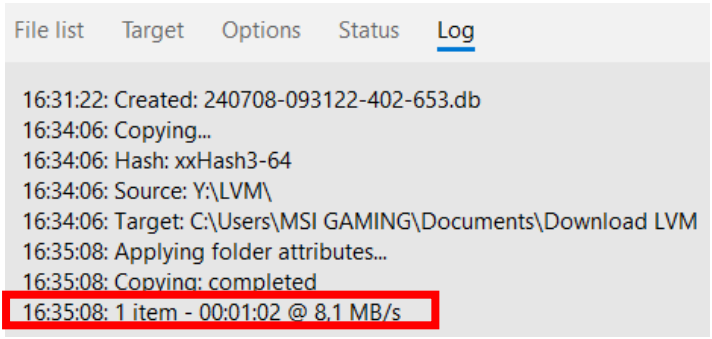
Gambar 4.32 *Download* file 400 MB/s pada LVM

E. *Download* File 500 MB/s

Pada Gambar 4.33 dan 4.34 merupakan tampilan percoban pertama Teracopy pada pengujian *download* file dengan ukuran 500 MB yang dilakukan pada Stratis dan LVM. Dapat dilihat kecepatan *download* file dengan ukuran 500 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 8,1 MB/s selama 1 menit 2 detik sedangkan Stratis sebesar 6,9 MB/s selama 1 menit 12 detik.



Gambar 4.33 *Download* file 500 MB/s pada Stratis



Gambar 4.34 *Download* file 500 MB/s pada LVM

Gambar di atas merupakan cara mengakses layanan secure *copy* dengan IP Address Stratis dan IP Address LVM. Setelah itu folder pada layanan secure copy dapat digunakan untuk melakukan *upload* dan *download* file. Percobaan dilakukan dengan pengulangan selama 10 kali dan setelah itu dihitung rata-rata kecepatan *transfer* dan write/read file.

4.2.3 Pengujian pada *disk* menggunakan *Tools*

Pengujian ini dilakukan pada *disk* dengan menggunakan *tools* `dd` untuk mengambil data kecepatan tulis *disk* dengan menulis atau membaca blok data tertentu dan menghasilkan kecepatan rata-rata langsung. Kemudian, *tools* `iostat` untuk untuk memantau statistik input/output sistem, yang memberikan informasi tentang kinerja

perangkat penyimpanan serta membandingkan write/read file pada *disk* dengan ukuran 100 MB, 200 MB, 300 MB, 400 MB dan 500 MB yang dilakukan masing – masing selama 10 kali pengujian. Berikut perintah dalam pengujian *disk* menggunakan *tools* `dd` dan `iostat`.

1. Menjalankan perintah `dd`

   a. Menjalankan perintah `dd` dengan ukuran 100 MB pada Stratis dan LVM

```
#dd if=/dev/urandom of=/data/100mb.img bs=1M count=100 status=progress
```

```
[root@stratis sarah]# dd if=/dev/urandom of=/data/100mb.img bs=1M count=100 status=progress
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.498624 s, 210 MB/s
```

Gambar 4.35 Perintah dd 100 MB/s pada Stratis

```
[root@lvm sarah]# dd if=/dev/urandom of=/data/100mb.img bs=1M count=100 status=progress
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.368774 s, 284 MB/s
```

Gambar 4.36 Perintah dd 100 MB/s pada LVM

Pada Gambar 4.35 dan 4.36 merupakan tampilan percoban pertama dd untuk mengambil data kecepatan tulis *disk* dengan ukuran 100 MB yang dilakukan pada Stratis dan LVM. Perintah `if=/dev/urandom` menentukan file input sebagai `/dev/urandom`, yaitu sebuah pseudo-random number generator yang menghasilkan data acak. Perintah `of=/data/100mb.img` menentukan file output sebagai `/data/100mb.img`, yaitu file yang akan dihasilkan dengan data acak yang diambil dari `/dev/urandom`. Perintah `bs=1M` menetapkan ukuran blok (block size) sebesar 1 Megabyte, yang berarti setiap kali `dd` membaca atau menulis data, ukuran bloknya adalah 1 MB. Dengan perintah `count=100`, ditentukan bahwa `dd` akan menulis 100 blok, masing-masing sebesar 1 MB, sehingga total ukuran file output adalah 100 MB. Terakhir, `status=progress` digunakan untuk menampilkan kemajuan (progress) selama proses copy berlangsung. Dapat dilihat

kecepatan tulis *disk* dengan ukuran 100 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 284 MB/s sedangkan Stratis sebesar 210 MB/s.

b. Menjalankan perintah `dd` dengan ukuran 200 MB pada Stratis dan LVM

```
#dd if=/dev/urandom of=/data/200mb.img bs=1M count=200 status=progress
```

```
[root@stratis sarah]# dd if=/dev/urandom of=/data/200mb.img bs=1M count=200 status=progress
200+0 records in
200+0 records out
209715200 bytes (210 MB, 200 MiB) copied, 0.700024 s, 300 MB/s
```

Gambar 4.37 Perintah dd 200 MB/s pada Stratis

```
[root@lvm sarah]# dd if=/dev/urandom of=/data/200mb.img bs=1M count=200 status=progress
200+0 records in
200+0 records out
209715200 bytes (210 MB, 200 MiB) copied, 0.701213 s, 299 MB/s
```

Gambar 4.38 Perintah dd 200 MB/s pada LVM

Pada Gambar 4.37 dan 4.38 merupakan tampilan percoban pertama dd untuk mengambil data kecepatan tulis *disk* dengan ukuran 200 MB yang dilakukan pada Stratis dan LVM. Perintah `if=/dev/urandom` menentukan file input sebagai `/dev/urandom`, yaitu sebuah pseudo-random number generator yang menghasilkan data acak. Perintah `of=/data/200mb.img` menentukan file output sebagai `/data/200mb.img`, yaitu file yang akan dihasilkan dengan data acak yang diambil dari `/dev/urandom`. Perintah `bs=1M` menetapkan ukuran blok (block size) sebesar 1 Megabyte, yang berarti setiap kali `dd` membaca atau menulis data, ukuran bloknya adalah 1 MB. Dengan perintah `count=200`, ditentukan bahwa `dd` akan menulis 200 blok, masing-masing sebesar 1 MB, sehingga total ukuran file output adalah 200 MB. Terakhir, `status=progress` digunakan untuk menampilkan kemajuan (progress) selama proses copy berlangsung. Dapat dilihat kecepatan tulis *disk* dengan ukuran 200 MB pada Stratis lebih cepat dibandingkan dengan LVM. Stratis memiliki nilai kecepatan sebesar 300 MB/s sedangkan LVM sebesar 299 MB/s.

c. Menjalankan perintah `dd` dengan ukuran 300 MB pada Stratis dan LVM

```
#dd if=/dev/urandom of=/data/300mb.img bs=1M count=300 status=progress
```

```
[root@stratis sarah]# dd if=/dev/urandom of=/data/300mb.img bs=1M count=300 status=progress
286261248 bytes (286 MB, 273 MiB) copied, 1 s, 286 MB/s
300+0 records in
300+0 records out
314572800 bytes (315 MB, 300 MiB) copied, 1.09813 s, 286 MB/s
```

Gambar 4.39 Perintah dd 300 MB/s pada Stratis

```
[root@lvm sarah]# dd if=/dev/urandom of=/data/300mb.img bs=1M count=300 status=progress
296747008 bytes (297 MB, 283 MiB) copied, 1 s, 296 MB/s
300+0 records in
300+0 records out
314572800 bytes (315 MB, 300 MiB) copied, 1.0631 s, 296 MB/s
```

Gambar 4.40 Perintah dd 300 MB/s pada LVM

Pada Gambar 4.39 dan 4.40 merupakan tampilan percoban pertama dd untuk mengambil data kecepatan tulis *disk* dengan ukuran 300 MB yang dilakukan pada Stratis dan LVM. Perintah `if=/dev/urandom` menentukan file input sebagai `/dev/urandom`, yaitu sebuah pseudo-random number generator yang menghasilkan data acak. Perintah `of=/data/300mb.img` menentukan file output sebagai `/data/300mb.img`, yaitu file yang akan dihasilkan dengan data acak yang diambil dari `/dev/urandom`. Perintah `bs=1M` menetapkan ukuran blok (block size) sebesar 1 Megabyte, yang berarti setiap kali `dd` membaca atau menulis data, ukuran bloknya adalah 1 MB. Dengan perintah `count=300`, ditentukan bahwa `dd` akan menulis 300 blok, masing-masing sebesar 1 MB, sehingga total ukuran file output adalah 300 MB. Terakhir, `status=progress` digunakan untuk menampilkan kemajuan (progress) selama proses copy berlangsung. Dapat dilihat kecepatan tulis *disk* dengan ukuran 300 MB pada LVM lebih cepat dibandingkan dengan Stratis. LVM memiliki nilai kecepatan sebesar 296 MB/s sedangkan Stratis sebesar 286 MB/s.

d. Menjalankan perintah `dd` dengan ukuran 400 MB pada Stratis dan LVM

```
#dd if=/dev/urandom of=/data/400mb.img bs=1M
count=400 status=progress
```

```
[root@stratis sarah]# dd if=/dev/urandom of=/data/400mb.img bs=1M count=400 status=progress
291504128 bytes (292 MB, 278 MiB) copied, 1 s, 291 MB/s
400+0 records in
400+0 records out
419430400 bytes (419 MB, 400 MiB) copied, 1.44606 s, 290 MB/s
```

Gambar 4.41 Perintah dd 400 MB/s pada Stratis

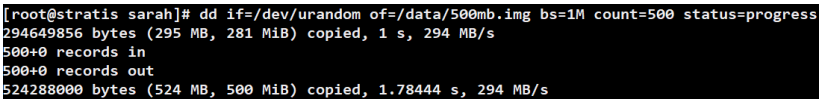
```
[root@lvm sarah]# dd if=/dev/urandom of=/data/400mb.img bs=1M count=400 status=progress
287309824 bytes (287 MB, 274 MiB) copied, 1 s, 287 MB/s
400+0 records in
400+0 records out
419430400 bytes (419 MB, 400 MiB) copied, 1.45685 s, 288 MB/s
```

Gambar 4.42 Perintah dd 400 MB/s pada LVM

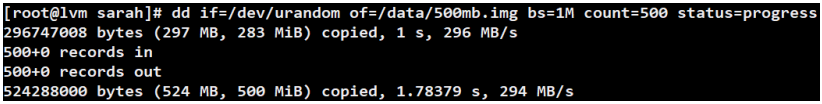
Pada Gambar 4.41 dan 4.42 merupakan tampilan percoban pertama dd untuk mengambil data kecepatan tulis *disk* dengan ukuran 400 MB yang dilakukan pada Stratis dan LVM. Perintah `if=/dev/urandom` menentukan file input sebagai `/dev/urandom`, yaitu sebuah pseudo-random number generator yang menghasilkan data acak. Perintah `of=/data/400mb.img` menentukan file output sebagai `/data/400mb.img`, yaitu file yang akan dihasilkan dengan data acak yang diambil dari `/dev/urandom`. Perintah `bs=1M` menetapkan ukuran blok (block size) sebesar 1 Megabyte, yang berarti setiap kali `dd` membaca atau menulis data, ukuran bloknya adalah 1 MB. Dengan perintah `count=400`, ditentukan bahwa `dd` akan menulis 400 blok, masing-masing sebesar 1 MB, sehingga total ukuran file output adalah 400 MB. Terakhir, `status=progress` digunakan untuk menampilkan kemajuan (progress) selama proses copy berlangsung. Dapat dilihat kecepatan tulis *disk* dengan ukuran 400 MB pada Stratis lebih cepat dibandingkan dengan LVM. Stratis memiliki nilai kecepatan sebesar 290 MB/s sedangkan LVM sebesar 288 MB/s.

e. Menjalankan perintah `dd` dengan ukuran 500 MB pada Stratis dan LVM

```
#dd if=/dev/urandom of=/data/500mb.img bs=1M
count=500 status=progress
```



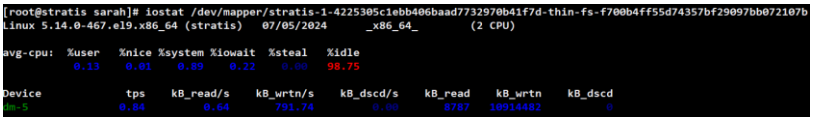
Gambar 4.43 Perintah dd 500 MB/s pada Stratis



Gambar 4.44 Perintah dd 500 MB/s pada LVM

Pada Gambar 4.43 dan 4.44 merupakan tampilan percoban pertama dd untuk mengambil data kecepatan tulis *disk* dengan ukuran 500 MB yang dilakukan pada Stratis dan LVM. Perintah `if=/dev/urandom` menentukan file input sebagai `/dev/urandom`, yaitu sebuah pseudo-random number generator yang menghasilkan data acak. Perintah `of=/data/500mb.img` menentukan file output sebagai `/data/500mb.img`, yaitu file yang akan dihasilkan dengan data acak yang diambil dari `/dev/urandom`. Perintah `bs=1M` menetapkan ukuran blok (block size) sebesar 1 Megabyte, yang berarti setiap kali `dd` membaca atau menulis data, ukuran bloknya adalah 1 MB. Dengan perintah `count=500`, ditentukan bahwa `dd` akan menulis 500 blok, masing-masing sebesar 1 MB, sehingga total ukuran file output adalah 500 MB. Terakhir, `status=progress` digunakan untuk menampilkan kemajuan (progress) selama proses copy berlangsung. Dapat dilihat kecepatan tulis *disk* dengan ukuran 500 MB pada Stratis sebanding dengan LVM. Stratis dan LVM sama sama memiliki nilai kecepatan sebesar 294 MB/s.

2. Menjalankan perintah `iostat`



Gambar 4.45 Perintah iostat pada Stratis

Tampilan di atas merupakan hasil untuk pengujian pada write/read file pada *device* Stratis. Device (dm-5) adalah perangkat yang sedang dimonitor. Perangkat dm-5 mengalami 0,84 transaksi I/O per detik. Pada *device* Stratis membaca 0,64 kilobyte per detik dan menulis 791,74 kilobyte per detik. Dapat dilihat bahwa sebanyak 8787 kilobyte yang telah dibaca dari perangkat serta total 10914482 kilobyte yang telah ditulis ke perangkat. Dari data ini, terlihat bahwa perangkat dm-5 lebih banyak digunakan untuk operasi penulisan daripada pembacaan, dengan jumlah transaksi I/O yang cukup rendah. Persentase waktu idle CPU sangat tinggi (98,75%), menunjukkan bahwa CPU sebagian besar tidak digunakan.

```
[root@lvm sarah]# iostat /dev/mapper/vg01-lv01
Linux 5.14.0-467.el9.x86_64 (lvm)       07/05/2024      _x86_64_        (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.14    0.03    0.64    0.33    0.00   98.86

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
dm-0              0.11         0.18      1112.33         0.00       2494   15411695          0
```

Gambar 4.46 Perintah iostat pada LVM

Tampilan di atas merupakan hasil untuk pengujian pada write/read file pada *device* LVM. Device (dm-0) adalah perangkat yang sedang dimonitor. Perangkat dm-0 mengalami 0,11 transaksi I/O per detik. Pada *device* LVM membaca 0,18 kilobyte per detik dan menulis 1112,33 kilobyte per detik. Dapat dilihat bahwa sebanyak 2494 kilobyte yang telah dibaca dari perangkat serta total 15411695 kilobyte yang telah ditulis ke perangkat. Dari data ini, terlihat bahwa perangkat dm-0 lebih banyak digunakan untuk operasi penulisan daripada pembacaan, dengan jumlah transaksi I/O yang cukup rendah. Persentase waktu idle CPU sangat tinggi (98,86%), menunjukkan bahwa CPU sebagian besar tidak digunakan.

4.3 Analisis

Berdasarkan hasil pengujian yang dilakukan pada kedua file system yaitu Stratis dan LVM, maka selanjutnya akan dilakukan analisis terhadap hasil dari pengujian.

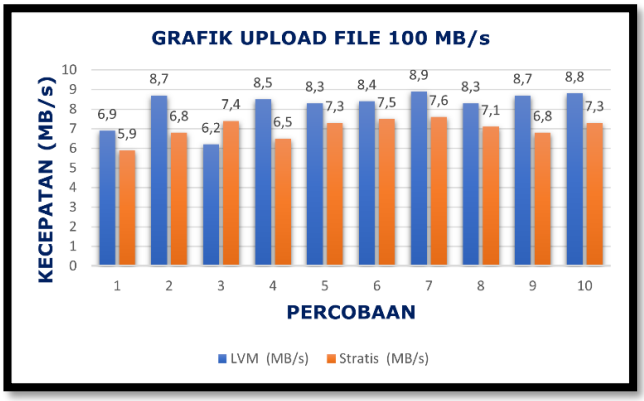
4.3.1 Analisis kecepatan *upload* dan *download file*

Pengujian ini memperoleh hasil kecepatan *upload* file pada masing-masing *file system* yang diuji sebanyak 10 kali percobaan dengan ukuran file sebesar 100 MB, 200 MB, 300 MB, 400 MB dan 500 MB yang dilakukan secara parallel.

1. Analisis kecepatan *upload* file pada Stratis dan LVM

   a. Analisis kecepatan *upload* file 100 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.47 tersebut menunjukkan perbandingan kecepatan *upload* file berukuran 100 MB antara LVM dan Stratis selama 10 kali percobaan mengindikasikan bahwa LVM secara konsisten memiliki kecepatan yang lebih tinggi dibandingkan Stratis. Rata-rata kecepatan *upload* LVM adalah 8,17 MB/s, sedangkan Stratis hanya 7,02 MB/s. Dalam semua percobaan, LVM unggul atas Stratis, dengan perbedaan yang signifikan terutama terlihat pada percobaan 2, 4, 7, 9, dan 10. LVM cenderung lebih cepat dibandingkan Stratis dalam pengujian I/O seperti *upload* file 100 MB karena metode pembagian data dan pemrosesan bersamaan operasi I/O yang lebih efisien. LVM menggunakan teknik seperti *striping* yang memungkinkan pembagian file menjadi beberapa bagian yang didistribusikan ke beberapa perangkat penyimpanan, meningkatkan kecepatan *transfer* data secara keseluruhan. Di sisi lain, Stratis yang memprioritaskan manajemen otomatis dan penempatan data efisien dapat mengalami sedikit penurunan kinerja karena overhead dari proses-proses tersebut. Selain itu, manajemen metadata, *caching*, dan *buffering* yang lebih efisien di LVM juga berkontribusi pada kecepatan I/O yang lebih tinggi dibandingkan Stratis.



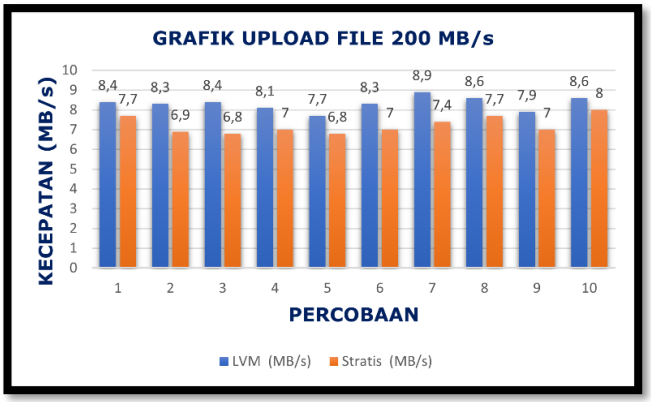
Gambar 4.47 Grafik *upload* file 100 MB/s

Meskipun kecepatan *transfer* pada LVM lebih tinggi, kecepatannya tidak selalu stabil karena faktor lain seperti beban kerja dan konfigurasi perangkat keras seperti kecepatan dan jenis dari *storage* yang digunakan untuk menyimpan volume fisik (physical volumes) begitu pun dengan Stratis yang mengalami peningkatan kecepatan pada proses percobaan pertama hingga ke tiga, kemudian mengalami penurunan pada percobaan ke tiga hingga ke empat, lalu mengalami pertambahan kembali pada percobaan ke lima hingga ke tujuh dan kembali mengalami penurunan sampai percobaan ke sepuluh. Tetapi di sisi lain, Stratis memungkinkan administrator untuk menentukan strategi penempatan (placement strategy) yang efisien. Selain itu, Stratis memiliki kemampuan yang mudah untuk beradaptasi dengan penambahan atau pengurangan ukuran data, sehingga lebih fleksibel karena mampu menyimpan data dengan kapasitas besar maupun kecil. Oleh karena itu, jika kecepatan *upload* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 100 MB/s ini.

b. Analisis kecepatan *upload* file 200 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.48 tersebut menunjukkan bahwa LVM secara konsisten memiliki kecepatan yang lebih tinggi

dibandingkan Stratis. Rata-rata kecepatan *upload* LVM adalah 8,32 MB/s, sedangkan Stratis hanya 7,23 MB/s. Dalam beberapa percobaan, LVM unggul atas Stratis, dengan perbedaan yang signifikan terutama terlihat pada percobaan 2, 3 dan 7. Hal ini dapat disebabkan oleh penggunaan teknik seperti striping pada LVM yang memungkinkan pembagian file menjadi beberapa bagian yang didistribusikan ke beberapa perangkat penyimpanan, meningkatkan kecepatan *transfer* data secara keseluruhan. Sedangkan Stratis mengelola penempatan data efisien dan memprioritaskan metadata secara otomatis, dapat mengalami sedikit penurunan kinerja karena overhead dari proses-proses tersebut. Selain itu, manajemen metadata, *caching*, dan *buffering* yang lebih efisien di LVM juga berkontribusi pada kecepatan I/O yang lebih tinggi dibandingkan Stratis. Stratis memiliki kemampuan yang mudah untuk beradaptasi dengan penambahan atau pengurangan ukuran data, sehingga lebih fleksibel karena mampu menyimpan data dengan kapasitas besar maupun kecil.



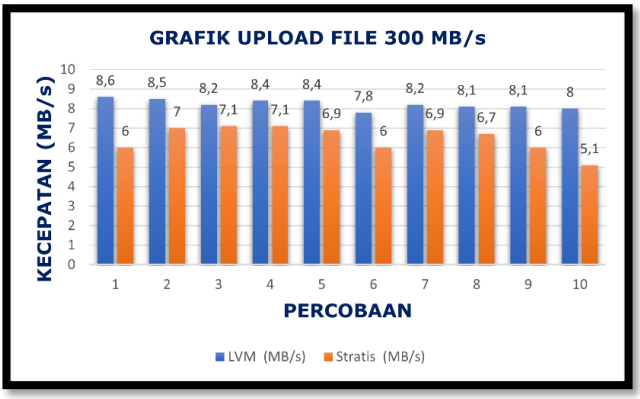
Gambar 4.48 Grafik *upload* file 200 MB/s

Namun perubahan kecepatan *upload* pada setiap percobaan tidak stabil, baik menggunakan LVM atapun Stratis. LVM yang memiliki kecepatan lebih tinggi pada saat proses percobaan pertama hingga ke tujuh kecepatan semakin bertambah dan mengalami penurunan pada proses percobaan kedelapan hingga ke sepuluh. Hal ini terjadi dikarenakan faktor lain seperti beban kerja dan konfigurasi perangkat keras seperti kecepatan dan jenis dari *storage* yang

digunakan untuk menyimpan volume fisik (physical volumes) begitu pun dengan Stratis yang mengalami penurunan pada proses percobaan pertama hingga ke lima dan semakin bertambah pada percobaan ke enam hingga ke sepuluh. Tetapi Stratis memungkinkan administrator untuk menentukan strategi penempatan (placement strategy) yang efisien. Oleh karena itu, jika kecepatan *upload* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 200 MB/s ini.

c. Analisis kecepatan *upload* file 300 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.49 tersebut menunjukkan bahwa LVM secara konsisten memiliki kecepatan yang lebih tinggi dibandingkan Stratis. Rata-rata kecepatan *upload* LVM adalah 8,23 MB/s, sedangkan Stratis hanya 6,48 MB/s. Dalam beberapa percobaan, LVM unggul atas Stratis, dengan perbedaan yang signifikan terutama terlihat pada percobaan 1, 2, 9 dan 10. Hal ini dapat disebabkan oleh penggunaan teknik seperti striping pada LVM yang memungkinkan pembagian file menjadi beberapa bagian yang didistribusikan ke beberapa perangkat penyimpanan, meningkatkan kecepatan *transfer* data secara keseluruhan. Sedangkan Stratis mengelola penempatan data efisien dan memprioritaskan metadata secara otomatis, dapat mengalami sedikit penurunan kinerja karena overhead dari proses-proses tersebut. Selain itu, manajemen metadata, *caching*, dan *buffering* yang lebih efisien di LVM juga berkontribusi pada kecepatan I/O yang lebih tinggi dibandingkan Stratis. Stratis memiliki kemampuan yang mudah untuk beradaptasi dengan penambahan atau pengurangan ukuran data, sehingga lebih fleksibel karena mampu menyimpan data dengan kapasitas besar maupun kecil.



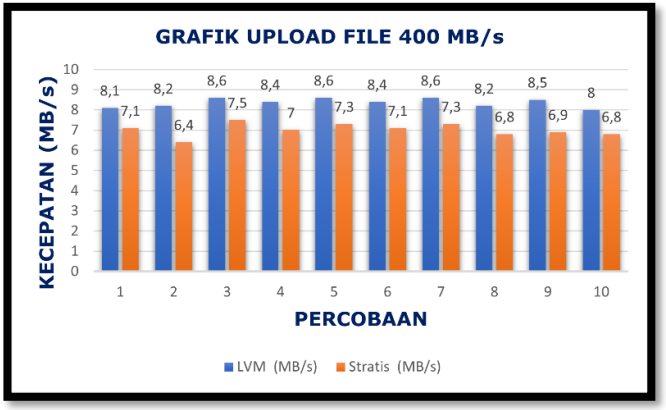
Gambar 4.49 Grafik *upload* file 300 MB/s

Namun perubahan kecepatan *upload* pada setiap percobaan tidak stabil, baik menggunakan LVM atapun Stratis. LVM yang memiliki kecepatan lebih tinggi pada saat proses percobaan pertama dan mengalami penurunan pada proses percobaan kedua hingga ke sepuluh. Hal ini terjadi dikarenakan faktor lain seperti beban kerja dan konfigurasi perangkat keras seperti kecepatan dan jenis dari *storage* yang digunakan untuk menyimpan volume fisik (physical volumes) begitu pun dengan Stratis yang memiliki kecepatan lebih tinggi pada saat proses percobaan pertama hingga ke empat dan mengalami penurunan pada proses percobaan ke lima hingga ke sepuluh. Tetapi Stratis memungkinkan administrator untuk menentukan strategi penempatan (placement strategy) yang efisien. Oleh karena itu, jika kecepatan *upload* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 300 MB/s ini.

d. Analisis kecepatan *upload* file 400 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.50 dapat dilihat perubahan kecepatan *upload* pada setiap percobaan, baik pada LVM maupun Stratis adalah stabil dan tidak jauh berbeda. Namun, dari percobaan pertama hingga percobaan kesepuluh terlihat bahwa LVM memiliki

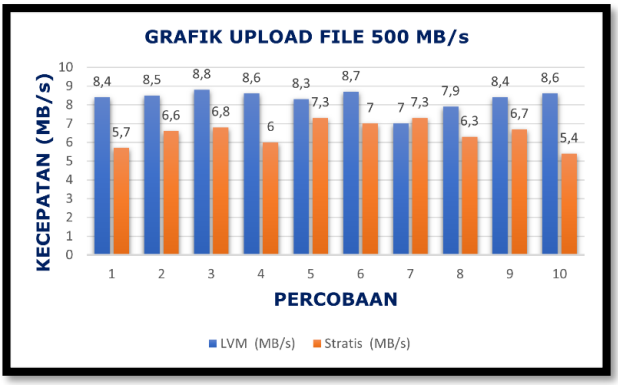
kecepatan yang lebih tinggi dibandingkan Stratis. Rata-rata kecepatan *upload* LVM adalah 8,36 MB/s, sedangkan Stratis hanya 7,02 MB/s. Dalam beberapa percobaan, LVM unggul atas Stratis, dengan perbedaan yang signifikan terutama terlihat pada percobaan 2 dan 4. Hal ini dapat disebabkan oleh penggunaan teknik seperti striping pada LVM yang memungkinkan pembagian file menjadi beberapa bagian yang didistribusikan ke beberapa perangkat penyimpanan, meningkatkan kecepatan *transfer* data secara keseluruhan. Sedangkan Stratis mengelola penempatan data efisien dan memprioritaskan metadata secara otomatis, dapat mengalami sedikit penurunan kinerja karena overhead dari proses-proses tersebut. Selain itu, manajemen metadata, *caching*, dan *buffering* yang lebih efisien di LVM juga berkontribusi pada kecepatan I/O yang lebih tinggi dibandingkan Stratis. Stratis memiliki kemampuan yang mudah untuk beradaptasi dengan penambahan atau pengurangan ukuran data, sehingga lebih fleksibel karena mampu menyimpan data dengan kapasitas besar maupun kecil. Stratis memungkinkan administrator untuk menentukan strategi penempatan (placement strategy) yang efisien. Oleh karena itu, jika kecepatan *upload* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 400 MB/s ini.



Gambar 4.50 Grafik *upload* file 400 MB/s

e. Analisis kecepatan *upload* file 500 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.51 tersebut menunjukkan bahwa LVM secara konsisten memiliki kecepatan yang lebih tinggi dibandingkan Stratis. Rata-rata kecepatan *upload* LVM adalah 8,32 MB/s, sedangkan Stratis hanya 6,51 MB/s. Dalam beberapa percobaan, LVM unggul atas Stratis, dengan perbedaan yang signifikan terutama terlihat pada percobaan 1, 4 dan 10. Hal ini dapat disebabkan oleh penggunaan teknik seperti striping pada LVM yang memungkinkan pembagian file menjadi beberapa bagian yang didistribusikan ke beberapa perangkat penyimpanan, meningkatkan kecepatan *transfer* data secara keseluruhan. Sedangkan Stratis mengelola penempatan data efisien dan memprioritaskan metadata secara otomatis, dapat mengalami sedikit penurunan kinerja karena overhead dari proses-proses tersebut. Selain itu, manajemen metadata, *caching*, dan *buffering* yang lebih efisien di LVM juga berkontribusi pada kecepatan I/O yang lebih tinggi dibandingkan Stratis. Stratis memiliki kemampuan yang mudah untuk beradaptasi dengan penambahan atau pengurangan ukuran data, sehingga lebih fleksibel karena mampu menyimpan data dengan kapasitas besar maupun kecil.



Gambar 4.51 Grafik *upload* file 500 MB/s

Namun perubahan kecepatan *upload* pada setiap percobaan tidak stabil, baik menggunakan LVM atapun Stratis. LVM yang memiliki kecepatan lebih tinggi pada saat proses percobaan pertama hingga percobaan ke tiga dan mengalami penurunan dan kenaikan

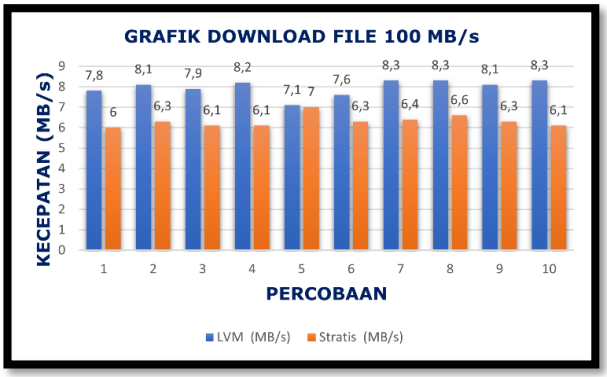
pada proses percobaan selanjutnya. Hal ini terjadi dikarenakan faktor lain seperti beban kerja dan konfigurasi perangkat keras seperti kecepatan dan jenis dari *storage* yang digunakan untuk menyimpan volume fisik (physical volumes) begitu pun dengan Stratis yang memiliki kecepatan lebih tinggi pada saat proses percobaan pertama hingga ke tiga dan mengalami penurunan dan kenaikan pada proses percobaan selanjutnya. Tetapi Stratis memungkinkan administrator untuk menentukan strategi penempatan (placement strategy) yang efisien. Oleh karena itu, jika kecepatan *upload* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 500 MB/s ini.

2. Analisis kecepatan *download* file pada Stratis dan LVM

    a. Analisis kecepatan *download* file 100 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.52 analisis menunjukkan bahwa kecepatan *download* file berukuran 100 MB/s pada LVM dan Stratis dilakukan selama 10 kali pengujian mengindikasikan bahwa LVM secara konsisten memiliki kecepatan yang lebih tinggi dibandingkan Stratis. Rata-rata kecepatan *download* pada LVM adalah sekitar 7,97 MB/s, sedangkan pada Stratis hanya sekitar 6.32 MB/s. Konsistensi kecepatan pada LVM terlihat jelas dengan kecepatan di atas 7 MB/s dalam semua percobaan. Secara keseluruhan, LVM menunjukkan kinerja yang lebih unggul dalam hal kecepatan *download* dibandingkan Stratis. Hal ini disebabkan oleh metode pembagian data dan pemrosesan I/O yang lebih efisien pada LVM, seperti teknik striping yang memungkinkan pembagian file menjadi beberapa bagian dan mendistribusikannya ke beberapa perangkat penyimpanan, sehingga meningkatkan kecepatan *transfer* data. Stratis, di sisi lain, mengelola penempatan data dan metadata secara otomatis, yang dapat menambah overhead. Meskipun Stratis lebih lambat, ia tetap menunjukkan kinerja yang stabil dalam setiap percobaan. Selain itu, manajemen metadata, *caching*, dan *buffering* yang lebih efisien di LVM juga berkontribusi pada kecepatan I/O yang lebih tinggi dibandingkan Stratis. Tetapi Stratis memungkinkan

administrator untuk menentukan strategi penempatan (placement strategy) yang efisien. Stratis memiliki kemampuan yang mudah untuk beradaptasi dengan penambahan atau pengurangan ukuran data, sehingga lebih fleksibel karena mampu menyimpan data dengan kapasitas besar maupun kecil. Oleh karena itu, jika kecepatan *download* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 100 MB/s ini.



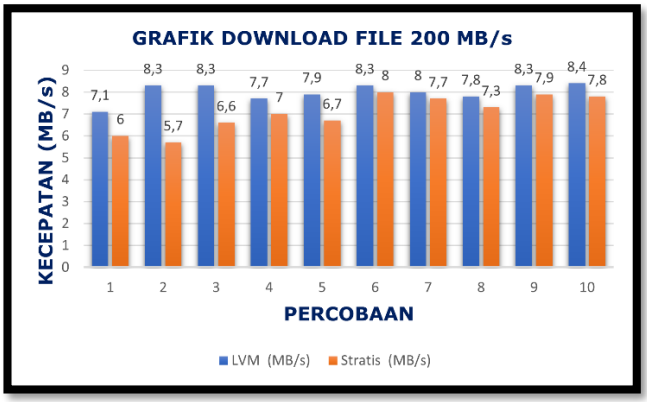
Gambar 4.52 Grafik *download* file 100 MB/s

b. Analisis kecepatan *download* file 200 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.53 analisis menunjukkan bahwa kecepatan *download* file berukuran 200 MB/s pada LVM dan Stratis dilakukan selama 10 kali pengujian mengindikasikan bahwa LVM secara konsisten tetap memiliki kecepatan yang lebih tinggi dibandingkan Stratis. Rata-rata kecepatan *download* pada LVM adalah sekitar 8,01 MB/s, sedangkan pada Stratis hanya sekitar 7,07 MB/s. Konsistensi kecepatan pada LVM terlihat jelas dengan kecepatan di atas 7 MB/s dalam semua percobaan. Secara keseluruhan, LVM menunjukkan kinerja yang lebih unggul dalam hal kecepatan *download* dibandingkan Stratis. Hal ini disebabkan oleh metode pembagian data dan pemrosesan I/O yang lebih efisien pada LVM, seperti teknik *striping* yang memungkinkan pembagian file menjadi beberapa bagian dan mendistribusikannya ke beberapa

perangkat penyimpanan, sehingga meningkatkan kecepatan *transfer* data. Sedangkan Stratis mengelola penempatan data efisien dan memprioritaskan metadata secara otomatis, dapat mengalami sedikit penurunan kinerja karena overhead dari proses-proses tersebut. Selain itu, manajemen metadata, *caching*, dan *buffering* yang lebih efisien di LVM juga berkontribusi pada kecepatan I/O yang lebih tinggi dibandingkan Stratis. Stratis memiliki kemampuan yang mudah untuk beradaptasi dengan penambahan atau pengurangan ukuran data, sehingga lebih fleksibel karena mampu menyimpan data dengan kapasitas besar maupun kecil.

Namun perubahan kecepatan *download* pada setiap percobaan tidak stabil, baik menggunakan LVM atapun Stratis. LVM yang memiliki peningkatan kecepatan pada saat proses percobaan pertama hingga percobaan ke tiga dan mengalami penurunan dan kenaikan pada proses percobaan selanjutnya. Hal ini terjadi dikarenakan faktor lain seperti beban kerja dan konfigurasi perangkat keras seperti kecepatan dan jenis dari *storage* yang digunakan untuk menyimpan volume fisik (physical volumes) begitu pun dengan Stratis yang mengalami penurunan kecepatan pada saat proses percobaan pertama hingga ke dua dan mengalami kenaikan dan penurunan pada proses percobaan selanjutnya. Tetapi Stratis memungkinkan administrator untuk menentukan strategi penempatan (*placement strategy*) yang efisien. Oleh karena itu, jika kecepatan *download* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 200 MB/s ini.

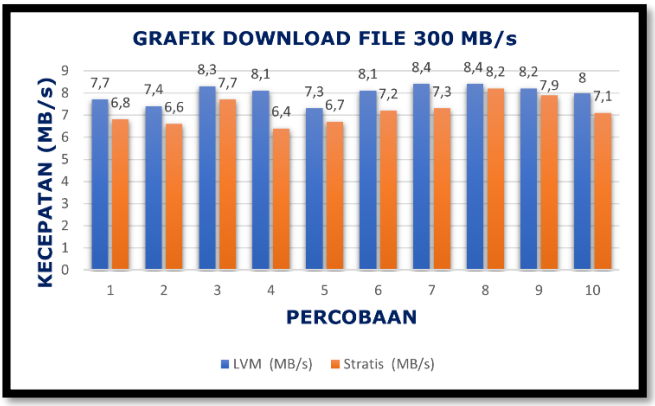


Gambar 4.53 Grafik *download* file 200 MB/s

c. Analisis kecepatan *download* file 300 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.54 analisis menunjukkan bahwa kecepatan *download* file berukuran 300 MB/s pada LVM dan Stratis dilakukan selama 10 kali pengujian mengindikasikan bahwa LVM secara konsisten tetap memiliki kecepatan yang lebih tinggi dibandingkan Stratis. Rata-rata kecepatan *download* pada LVM adalah sekitar 7,99 MB/s, sedangkan pada Stratis hanya sekitar 7,19 MB/s. Konsistensi kecepatan pada LVM terlihat jelas dengan kecepatan di atas 7 MB/s dalam semua percobaan. Secara keseluruhan, LVM menunjukkan kinerja yang lebih unggul dalam hal kecepatan *download* dibandingkan Stratis. Hal ini disebabkan LVM menggunakan teknik striping yang membagi data menjadi beberapa bagian kecil dan mendistribusikannya ke berbagai perangkat penyimpanan, memungkinkan akses data secara paralel dan meningkatkan kecepatan *transfer* data secara keseluruhan. Selain itu, manajemen metadata di LVM cenderung lebih sederhana dan efisien, yang membantu mempercepat operasi I/O dengan mengurangi latensi saat mengakses atau menulis data. Mekanisme caching dan buffering yang lebih efisien di LVM juga berkontribusi pada peningkatan kecepatan akses data, terutama untuk operasi yang sering diulang. Di sisi lain, Stratis mengelola penempatan data dan metadata secara

otomatis, yang menambah overhead dan mempengaruhi kecepatan I/O. Fokus Stratis pada manajemen otomatis dan fleksibilitas penyimpanan juga menambah kompleksitas dalam pengelolaan data, yang bisa memperlambat kecepatan *download*. Oleh karena itu, efisiensi dalam manajemen data dan minimnya overhead dari fitur manajemen otomatis membuat LVM menunjukkan kinerja kecepatan *download* yang lebih tinggi dibandingkan dengan Stratis.

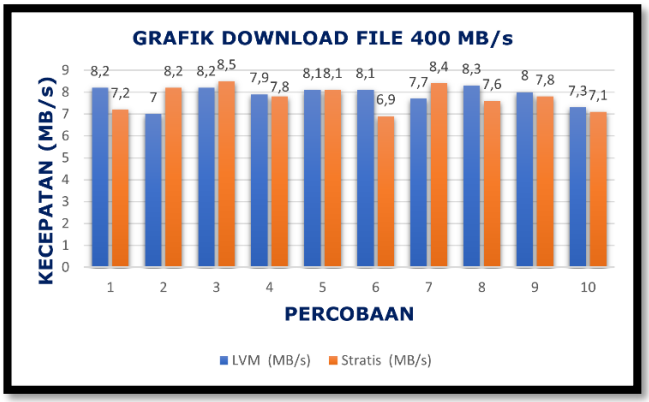


Gambar 4.54 Grafik *download* file 300 MB/s

Namun perubahan kecepatan *download* pada setiap percobaan tidak stabil, baik menggunakan LVM atapun Stratis. LVM yang memiliki penurunan kecepatan pada saat proses percobaan pertama hingga percobaan ke dua dan mengalami kenaikan dan penurunan pada proses percobaan selanjutnya. Hal ini terjadi dikarenakan faktor lain seperti beban kerja dan konfigurasi perangkat keras seperti kecepatan dan jenis dari *storage* yang digunakan untuk menyimpan volume fisik (physical volumes) begitu pun dengan Stratis yang mengalami penurunan kecepatan pada saat proses percobaan pertama hingga ke dua dan mengalami kenaikan dan penurunan pada proses percobaan selanjutnya. Tetapi Stratis memungkinkan administrator untuk menentukan strategi penempatan (placement strategy) yang efisien. Oleh karena itu, jika kecepatan *download* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 300 MB/s ini.

d. Analisis kecepatan *download* file 400 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.55 analisis menunjukkan bahwa kecepatan *download* file berukuran 400 MB/s pada LVM dan Stratis dilakukan selama 10 kali pengujian mengindikasikan bahwa LVM secara konsisten tetap memiliki kecepatan yang lebih tinggi dibandingkan Stratis. Rata-rata kecepatan *download* pada LVM adalah sekitar 7,88 MB/s, sedangkan pada Stratis hanya sekitar 7,76 MB/s. Konsistensi kecepatan pada LVM terlihat jelas dengan kecepatan di atas 7 MB/s dalam semua percobaan. Secara keseluruhan, LVM menunjukkan kinerja yang lebih unggul dalam hal kecepatan *download* dibandingkan Stratis. Hal ini disebabkan LVM menggunakan teknik striping yang membagi data menjadi beberapa bagian kecil dan mendistribusikannya ke berbagai perangkat penyimpanan, memungkinkan akses data secara paralel dan meningkatkan kecepatan *transfer* data secara keseluruhan. Selain itu, manajemen metadata di LVM cenderung lebih sederhana dan efisien, yang membantu mempercepat operasi I/O dengan mengurangi latensi saat mengakses atau menulis data. Mekanisme caching dan buffering yang lebih efisien di LVM juga berkontribusi pada peningkatan kecepatan akses data, terutama untuk operasi yang sering diulang. Di sisi lain, Stratis mengelola penempatan data dan metadata secara otomatis, yang menambah overhead dan mempengaruhi kecepatan I/O. Fokus Stratis pada manajemen otomatis dan fleksibilitas penyimpanan juga menambah kompleksitas dalam pengelolaan data, yang bisa memperlambat kecepatan *download*. Oleh karena itu, efisiensi dalam manajemen data dan minimnya overhead dari fitur manajemen otomatis membuat LVM menunjukkan kinerja kecepatan *download* yang lebih tinggi dibandingkan dengan Stratis.



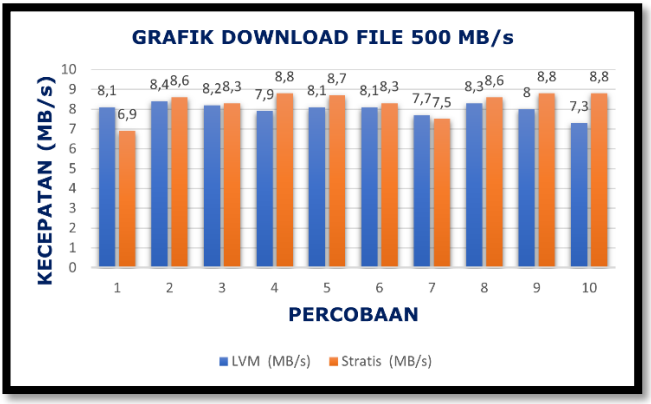
Gambar 4.55 Grafik *download* file 400 MB/s

Namun perubahan kecepatan *download* pada setiap percobaan tidak stabil, baik menggunakan LVM atapun Stratis. LVM yang memiliki penurunan kecepatan pada saat proses percobaan pertama hingga percobaan ke dua dan mengalami kenaikan dan penurunan pada proses percobaan selanjutnya. Hal ini terjadi dikarenakan faktor lain seperti beban kerja dan konfigurasi perangkat keras seperti kecepatan dan jenis dari *storage* yang digunakan untuk menyimpan volume fisik (physical volumes) begitu pun dengan Stratis yang mengalami peningkatan kecepatan pada saat proses percobaan pertama hingga ke tiga dan mengalami penurunan dan kenaikan pada proses percobaan selanjutnya. Tetapi Stratis memungkinkan administrator untuk menentukan strategi penempatan (placement strategy) yang efisien. Oleh karena itu, jika kecepatan *download* merupakan prioritas utama, LVM adalah pilihan yang lebih baik berdasarkan hasil percobaan dengan ukuran 400 MB/s ini.

e. Analisis kecepatan *download* file 500 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.56 analisis menunjukkan bahwa kecepatan *download* file berukuran 500 MB/s pada LVM dan Stratis dilakukan selama 10 kali pengujian mengindikasikan bahwa Stratis memiliki kecepatan yang lebih tinggi dibandingkan LVM.

Rata-rata kecepatan *download* pada Stratis adalah sekitar 8,33 MB/s, sedangkan pada LVM hanya sekitar 8,01 MB/s. Perbedaan kecepatan antara LVM dan Stratis saat mengunduh file dari berbagai ukuran bisa disebabkan oleh beberapa faktor utama. Pada file yang lebih kecil seperti 100 MB hingga 400 MB, kecepatan *download* LVM lebih cepat karena lebih sedikit overhead dalam pengelolaan data dan dapat memanfaatkan cache sistem dengan lebih efektif. Namun, ketika ukuran file mencapai 500 MB, Stratis menunjukkan kinerja yang lebih baik karena kemampuannya dalam mengelola data dan metadata secara lebih efisien pada ukuran besar, kemampuan cache dan buffering lebih efektif, dan memiliki mekanisme untuk mendistribusikan beban kerja secara lebih optimal. Ketika ukuran file meningkat, kemampuan Stratis untuk mendistribusikan beban kerja secara lebih merata lebih efektif dibandingkan LVM. Dengan demikian, meskipun LVM unggul pada ukuran file yang lebih kecil, Stratis dapat lebih efektif dalam menangani file yang lebih besar karena pengelolaan data yang lebih baik dan kemampuan skalabilitasnya.



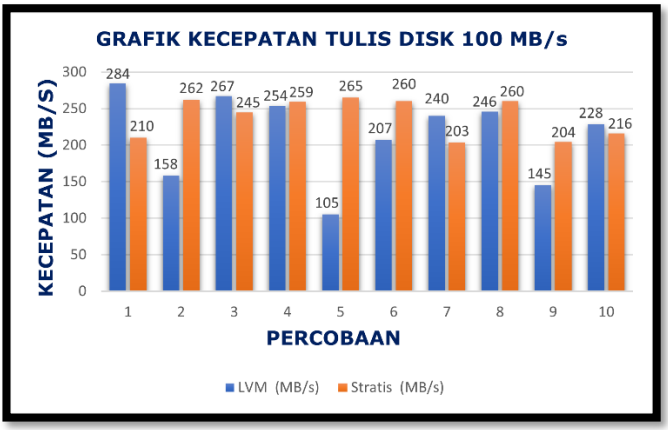
Gambar 4.56 Grafik *download* file 500 MB/s

4.3.2 Analisis kecepatan tulis *disk*

Pengujian ini memperoleh hasil kecepatan tulis *disk* pada masing-masing file system yang diuji sebanyak 10 kali percobaan dengan ukuran file sebesar sebesar 100 MB, 200 MB, 300 MB, 400 MB dan 500 MB.

a. Analisis kecepatan tulis *disk* 100 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.57 menunjukkan perbandingan kecepatan tulis *disk* antara LVM dan Stratis dalam 10 percobaan dengan ukuran 100 MB/s menggunakan *tools* `dd`. Analisis grafik pengujian kecepatan tulis *disk* menunjukkan bahwa Stratis memiliki rata-rata kecepatan tulis yang lebih tinggi yaitu 238,4 MB/s dibandingkan LVM hanya 213,4 MB/s. Hal ini dapat dijelaskan oleh beberapa faktor. Pertama, Stratis menunjukkan kecepatan yang relatif stabil pada sebagian besar percobaan, kecuali pada percobaan ke lima. Sebaliknya, LVM memiliki satu nilai ekstrem yang sangat tinggi pada percobaan pertama yaitu 284 MB/s tetapi juga beberapa nilai rendah seperti 105 MB/s pada percobaan ke lima dan 145 MB/s pada percobaan ke sembilan, yang menurunkan rata-rata keseluruhannya. Selain itu, Stratis menggunakan algoritma pengelolaan data yang lebih efisien dan memiliki mekanisme pengelolaan cache yang lebih baik, yang membantu menjaga kecepatan tulis yang tinggi. Overhead sistem yang lebih rendah pada Stratis dibandingkan dengan LVM juga dapat berkontribusi pada kinerja yang lebih baik, mengurangi beban kerja yang tidak perlu dan meningkatkan efisiensi penulisan data. Mekanisme penanganan I/O Stratis yang lebih efektif juga memungkinkan aliran data yang lebih optimal selama operasi tulis. Kombinasi faktor-faktor ini membuat Stratis mampu mencapai rata-rata kecepatan tulis yang lebih tinggi dibandingkan LVM dalam pengujian tersebut. Oleh karena itu, Stratis adalah pilihan yang lebih baik dalam pengujian kecepatan tulis *disk* berdasarkan hasil percobaan dengan ukuran 100 MB/s ini.

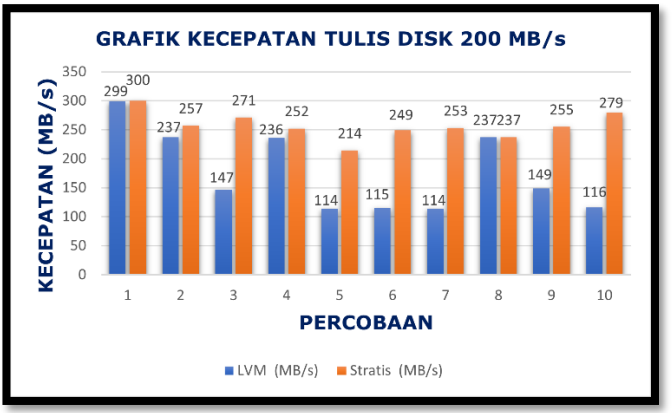


Gambar 4.57 Grafik tulis *disk* 100 MB/s

b. Analisis kecepatan tulis *disk* 200 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.58 menunjukkan perbandingan kecepatan tulis *disk* antara LVM dan Stratis dalam 10 percobaan dengan ukuran 200 MB/s menggunakan *tools* `dd`. Analisis grafik pengujian kecepatan tulis *disk* menunjukkan bahwa Stratis memiliki rata-rata kecepatan tulis yang lebih tinggi yaitu 256,7 MB/s dibandingkan LVM hanya 176,4 MB/s. Ini menunjukkan bahwa Stratis secara konsisten memberikan kinerja yang lebih baik dalam pengujian ini. Stratis menunjukkan kinerja yang lebih konsisten dengan kecepatan tulis yang relatif stabil pada setiap percobaan. Nilai terendah untuk Stratis adalah 214 MB/s, sementara LVM memiliki beberapa percobaan dengan kecepatan tulis di bawah 150 MB/s. LVM menunjukkan variasi yang lebih besar dalam kecepatan tulis, dengan kecepatan tertinggi 299 MB/s dan terendah 114 MB/s. Stratis memiliki rentang variasi yang lebih sempit, dengan kecepatan tertinggi 300 MB/s dan terendah 214 MB/s. Hal ini disebabkan algoritma pengelolaan data dan mekanisme pengelolaan cache yang digunakan oleh Stratis tampaknya lebih efisien, memungkinkan kecepatan tulis yang lebih tinggi dan konsisten. Overhead sistem yang lebih rendah pada Stratis dibandingkan dengan LVM juga dapat berkontribusi pada kinerja yang lebih baik, mengurangi beban kerja yang tidak perlu dan

meningkatkan efisiensi penulisan data. Mekanisme penanganan I/O Stratis yang lebih efektif juga memungkinkan aliran data yang lebih optimal selama operasi tulis. Kombinasi faktor-faktor ini membuat Stratis mampu mencapai rata-rata kecepatan tulis yang lebih tinggi dibandingkan LVM dalam pengujian tersebut. Oleh karena itu, Stratis adalah pilihan yang lebih baik dalam pengujian kecepatan tulis *disk* berdasarkan hasil percobaan dengan ukuran 100 MB/s ini.

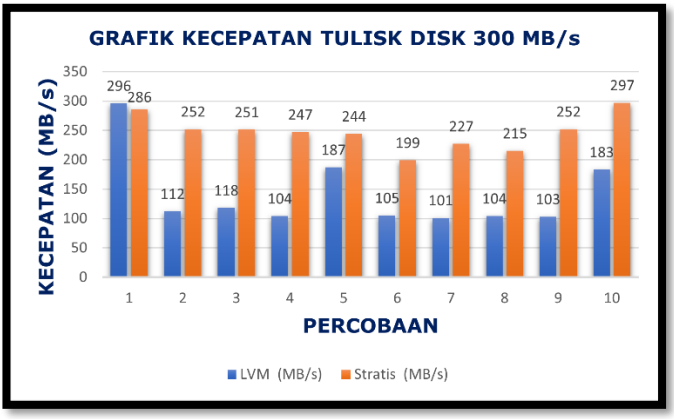


Gambar 4.58 Grafik tulis *disk* 200 MB/s

a. Analisis kecepatan tulis *disk* 300 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.59 menunjukkan perbandingan kecepatan tulis *disk* antara LVM dan Stratis dalam 10 percobaan dengan ukuran 300 MB/s menggunakan *tools* `dd`. Analisis grafik pengujian kecepatan tulis *disk* menunjukkan bahwa Stratis memiliki rata-rata kecepatan tulis yang lebih tinggi yaitu 247 MB/s dibandingkan LVM hanya 141,3 MB/s. Hal ini dapat dijelaskan oleh beberapa faktor. Analisis perbandingan juga menunjukkan variasi yang lebih besar dalam kecepatan tulis LVM, dengan rentang dari 101 MB/s hingga 296 MB/s. Di sisi lain, Stratis menunjukkan variasi yang lebih kecil, dari 199 MB/s hingga 297 MB/s. Hal ini menandakan bahwa Stratis cenderung memberikan kecepatan tulis yang lebih konsisten dibandingkan LVM dalam pengujian ini. Meskipun demikian, perlu diperhatikan bahwa dalam beberapa percobaan

tertentu, seperti pada percobaan kedua, ketiga, dan keenam, LVM mengalami kecepatan tulis yang lebih rendah daripada Stratis. Namun, secara keseluruhan, Stratis menawarkan kinerja yang lebih baik dan lebih stabil dalam konteks pengujian kecepatan tulis *disk* menggunakan *tools* `dd` dengan ukuran 300 MB/s. Hal ini disebabkan Stratis menggunakan algoritma pengelolaan data yang lebih efisien dan memiliki mekanisme pengelolaan cache yang lebih baik, yang membantu menjaga kecepatan tulis yang tinggi. Overhead sistem yang lebih rendah pada Stratis dibandingkan dengan LVM juga dapat berkontribusi pada kinerja yang lebih baik, mengurangi beban kerja yang tidak perlu dan meningkatkan efisiensi penulisan data. Mekanisme penanganan I/O Stratis yang lebih efektif juga memungkinkan aliran data yang lebih optimal selama operasi tulis. Kombinasi faktor-faktor ini membuat Stratis mampu mencapai rata-rata kecepatan tulis yang lebih tinggi dibandingkan LVM dalam pengujian tersebut. Oleh karena itu, Stratis adalah pilihan yang lebih baik dalam pengujian kecepatan tulis *disk* berdasarkan hasil percobaan dengan ukuran 300 MB/s ini.

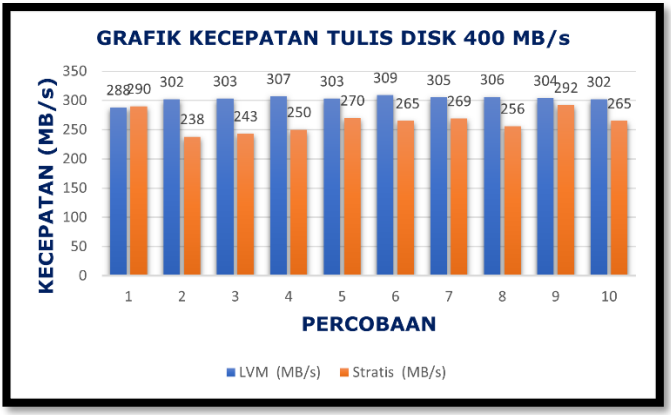


Gambar 4.59 Grafik tulis *disk* 300 MB/s

b. Analisis kecepatan tulis *disk* 400 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.60 menunjukkan perbandingan kecepatan tulis *disk* antara LVM dan Stratis dalam 10

percobaan dengan ukuran 400 MB/s menggunakan *tools* `dd`. Pada pengujian kali ini dapat dilihat perubahan kecepatan tulis *disk* pada setiap percobaan, baik pada LVM maupun Stratis adalah stabil dan tidak jauh berbeda. Berbeda dengan pengujian pada ukuran sebelumnya, analisis grafik pengujian kecepatan tulis *disk* menunjukkan bahwa LVM memiliki rata-rata kecepatan tulis yang lebih tinggi yaitu 302,9 MB/s dibandingkan Stratis hanya 263,8 MB/s. Hal ini disebabkan LVM dan Stratis menggunakan algoritma pengelolaan data yang berbeda. Pada kecepatan tulis tinggi seperti 400 MB/s, algoritma yang digunakan oleh LVM lebih efisien atau lebih cocok untuk jenis operasi yang dilakukan pada pengujian tersebut, sehingga menghasilkan kecepatan yang lebih tinggi. LVM memiliki mekanisme cache yang lebih optimal untuk kecepatan tinggi, yang memungkinkan untuk menangani operasi tulis dengan lebih cepat pada kecepatan 400 MB/s. Di situasi ini, LVM memiliki overhead yang lebih rendah pada kecepatan tulis 400 MB/s, sementara Stratis mungkin mengalami overhead yang lebih tinggi. Hal ini menandakan bahwa LVM cenderung memberikan kecepatan tulis yang lebih konsisten dibandingkan Stratis dalam pengujian ini. Kombinasi faktor-faktor ini membuat LVM mampu mencapai rata-rata kecepatan tulis yang lebih tinggi dibandingkan Stratis dalam pengujian tersebut.



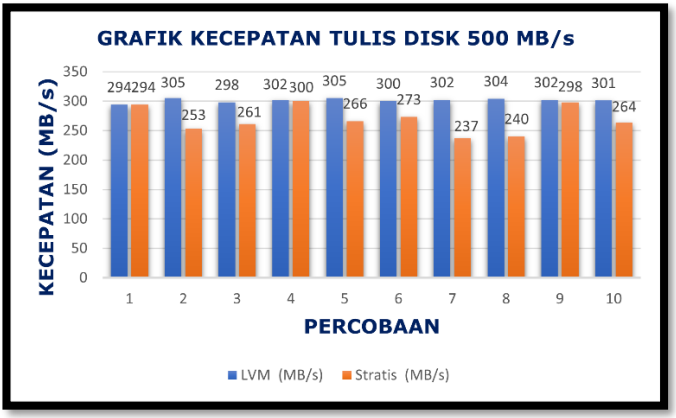
Gambar 4.60 Grafik tulis *disk* 400 MB/s

Oleh karena itu, LVM adalah pilihan yang lebih baik dalam pengujian kecepatan tulis *disk* berdasarkan hasil percobaan dengan ukuran 400 MB/s ini. Di sisi lain, pada kecepatan tulis yang lebih rendah seperti 100 MB/s, 200 MB/s, dan 300 MB/s, Stratis menunjukkan kinerja yang lebih baik karena konfigurasi atau algoritma pengelolaan data yang berbeda yang lebih efisien pada rentang kecepatan tersebut. Stratis juga dapat memiliki cara yang lebih baik dalam memanfaatkan cache atau mengelola overhead pada kecepatan tulis yang lebih rendah, yang membuatnya lebih cepat daripada LVM dalam situasi-situasi tersebut. Secara keseluruhan, perbedaan dalam kinerja antara LVM dan Stratis pada berbagai ukuran pengujian terutama dipengaruhi oleh bagaimana masing-masing sistem mengelola dan memproses data pada kecepatan tulis yang berbeda.

c. Analisis kecepatan tulis *disk* 500 MB/s pada Stratis dan LVM

Berdasarkan grafik dari Gambar 4.61 menunjukkan perbandingan kecepatan tulis *disk* antara LVM dan Stratis dalam 10 percobaan dengan ukuran 500 MB/s menggunakan *tools* `dd`. Pada pengujian kali ini dapat dilihat perubahan kecepatan tulis *disk* pada setiap percobaan, baik pada LVM maupun Stratis adalah stabil dan tidak jauh berbeda. Sama seperti pengujian pada ukuran 400 MB/s, analisis grafik pengujian kecepatan tulis *disk* menunjukkan bahwa LVM memiliki rata-rata kecepatan tulis yang lebih tinggi yaitu 301,3 MB/s dibandingkan Stratis hanya 268,6 MB/s. Hal ini disebabkan LVM dan Stratis menggunakan algoritma pengelolaan data yang berbeda. Pada kecepatan tulis yang lebih tinggi seperti 400 MB/s dan 500 MB/s, LVM cenderung menunjukkan keunggulan karena algoritma pengelolaan data yang lebih efisien dan pengaturan cache yang lebih optimal untuk menangani throughput data yang besar. Hal ini memungkinkan LVM untuk mengirimkan data dengan kecepatan yang lebih tinggi secara konsisten. LVM memiliki mekanisme cache yang lebih optimal untuk kecepatan tinggi, yang memungkinkan untuk menangani operasi tulis dengan lebih cepat pada kecepatan 500 MB/s. Di situasi ini, LVM memiliki overhead yang lebih rendah pada kecepatan tulis 500 MB/s, sementara Stratis mungkin mengalami

overhead yang lebih tinggi. Hal ini menandakan bahwa LVM cenderung memberikan kecepatan tulis yang lebih konsisten dibandingkan Stratis dalam pengujian ini. Kombinasi faktor-faktor ini membuat LVM mampu mencapai rata-rata kecepatan tulis yang lebih tinggi dibandingkan Stratis dalam pengujian tersebut. Di sisi lain, pada kecepatan tulis yang lebih rendah seperti 100 MB/s, 200 MB/s, dan 300 MB/s, Stratis juga dapat memiliki cara yang lebih baik dalam memanfaatkan cache atau mengelola overhead pada kecepatan tulis yang lebih rendah, yang membuatnya lebih cepat daripada LVM dalam situasi-situasi tersebut.

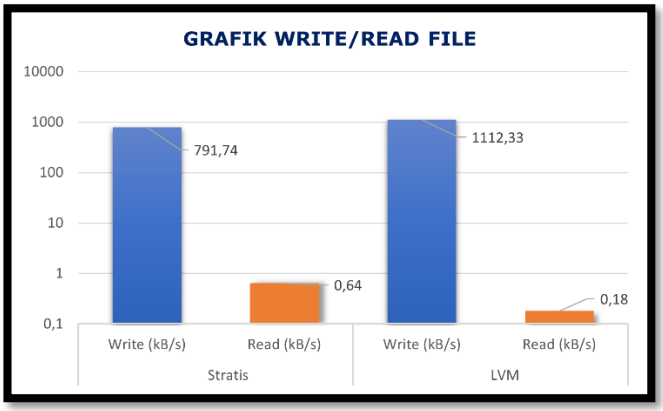


Gambar 4.61 Grafik tulis *disk* 500 MB/s

Oleh karena itu, LVM adalah pilihan yang lebih baik dalam pengujian kecepatan tulis *disk* berdasarkan hasil percobaan dengan ukuran 500 MB/s ini. Secara keseluruhan, perbedaan dalam kinerja antara LVM dan Stratis pada berbagai ukuran pengujian terutama dipengaruhi oleh bagaimana masing-masing sistem mengelola dan memproses data pada kecepatan tulis yang berbeda.

4.3.3 Analisis *write/read file*

Pengujian ini memperoleh hasil kecepatan tulis *disk* pada masing-masing file system yang diuji sebanyak 10 kali percobaan dengan ukuran file sebesar 500 MB.



Gambar 4.62 Grafik kecepatan tulis *disk*

Berdasarkan grafik dari Gambar 4.62 tersebut dapat dilihat perbandingan kecepatan write/read file menggunakan *tools* `iostat`. Berdasarkan hasil pengujian, perbandingan kecepatan tulis (write) dan baca (read) file antara Stratis dan LVM menunjukkan karakteristik yang berbeda. Pada aspek kecepatan tulis, LVM mencatatkan kinerja yang lebih tinggi dengan rata-rata 1112,33 kB/s, dibandingkan dengan Stratis yang hanya mencapai 791,74 kB/s. Perbedaan ini menunjukkan bahwa LVM mampu mengirimkan data ke *disk* dengan lebih cepat dalam pengujian ini. LVM menunjukkan kecepatan write yang lebih tinggi karena implementasi algoritma pengelolaan data yang dioptimalkan atau konfigurasi cache yang lebih efisien dalam menangani operasi tulis data. Algoritma ini dapat dirancang khusus untuk mengoptimalkan throughput data pada berbagai ukuran dan jenis file, sehingga LVM mampu men*transfer* data ke media penyimpanan dengan lebih cepat dalam pengujian tertentu. Di sisi lain, dalam hal kecepatan baca, Stratis menunjukkan angka yang sedikit lebih tinggi dengan 0,64 kB/s dibandingkan dengan 0,18 kB/s yang dicapai oleh LVM, meskipun perbedaannya sangat kecil. Stratis memiliki keunggulan dalam kecepatan read file karena strategi pengelolaan I/O yang lebih canggih. Ini dapat mencakup pengaturan cache yang optimal untuk mengurangi latensi akses dan meningkatkan

efisiensi baca data. Stratis juga mungkin menggunakan teknik penjadwalan I/O yang lebih baik, yang memungkinkan untuk respons cepat terhadap permintaan baca data dari media penyimpanan.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, ada beberapa hal yang dapat disimpulkan dari penelitian ini, yaitu:

1. Stratis dan LVM merupakan aplikasi *open-source* yang mudah digunakan untuk penggabungkan *storage*.
2. LVM dan Stratis sama-sama merupakan server yang dapat menggabungkan *storage* dengan kapasitas 100 GB menggunakan *file system* XFS, dikarenakan XFS tidak mempengaruhi total kapasitas dari *Storage Pool* yang dibuat. XFS hanya berfungsi sebagai sistem file yang mengatur cara data disimpan dan diakses dalam logical volume atau *pool*.
3. LVM memiliki kecepatan *upload* dan *download* rata-rata lebih tinggi dibandingkan Stratis. Hal ini dapat disebabkan oleh penggunaan teknik seperti *striping* pada LVM yang memungkinkan pembagian file menjadi beberapa bagian yang didistribusikan ke beberapa perangkat penyimpanan, dan meningkatkan kecepatan *transfer* data secara keseluruhan. Sedangkan Stratis mengelola penempatan data efisien dan memprioritaskan metadata secara otomatis, dapat mengalami sedikit penurunan kinerja karena overhead dari proses-proses tersebut.
4. Pada kecepatan tulis yang lebih tinggi seperti 400 MB dan 500 MB, LVM cenderung menunjukkan keunggulan karena algoritma pengelolaan data yang lebih efisien dan pengaturan cache yang lebih optimal untuk menangani throughput data yang besar. Hal ini memungkinkan LVM untuk mengirimkan data dengan kecepatan yang lebih tinggi secara konsisten. Di sisi lain, pada kecepatan tulis yang lebih rendah seperti 100 MB, 200 MB, dan 300 MB, Stratis lebih unggul karena kemampuannya untuk mengelola overhead sistem dengan lebih baik atau menggunakan algoritma yang lebih efektif dalam konteks kecepatan tersebut.

5. LVM adalah pilihan *file system* yang lebih baik dibandingkan Stratis untuk kinerja kecepatan *transfer* file berdasarkan hasil pengujian percobaan dengan berbagai ukuran file.
6. Stratis adalah pilihan *file system* yang lebih baik dibandingkan LVM untuk kinerja kecepatan tulis *disk* menggunakan *tools* `dd` berdasarkan hasil pengujian percobaan dengan berbagai ukuran file.
7. Dengan menggunakan *tools* `iostat`, LVM menunjukkan kecepatan write yang lebih tinggi dibandingkan Stratis. Namun Stratis menunjukkan kecepatan read yang lebih tinggi dibandingkan LVM dikarenakan Stratis memiliki keunggulan dalam kecepatan read file karena strategi pengelolaan I/O yang lebih canggih.

5.2 Saran

Adapun saran yang penulis berikan guna pengembangan penelitian selanjutnya adalah sebagai berikut:

1. Menambahkan banyak *user* untuk membandingkan *file system*.
2. Menambahkan variasi tipe file pada pengujian jenis file teks binary, JPG/JPEG, dan lain lain.

DAFTAR PUSTAKA

1000logos. (2024, May 21). *UBUNTU LOGO*. 1000logos.Net. https://1000logos.net/ubuntu-logo/

A. Maier. (2024, May 20). *Understanding the Fundamentals of a Cloud Computing Architecture*. Codecoda.Com. https://codecoda.com/en/blog/entry/understanding-the-fundamentals-of-a-cloud-computing-architecture

Adhitya, F. (2022). *Analisa Perbandingan Kinerja Filesystem Btrfs, ZFS, dan LVM pada Container berbasis Linux Daemon*. Skripsi Sarjana, Pustaka Politeknik Caltex Riau. https://opac.lib.pcr.ac.id/index.php?p=show_detail&id=14971&keywords=farhan+adhitya

Admin. (2024, May 21). *Cara Menggunakan Cluster di Proxmox*. Generalsolusindo.Com. https://generalsolusindo.com/cara-menggunakan-cluster-di-proxmox/

Anna. (2024, May 21). *ADM Novell W2 File System*. ADM. https://admzs3.wordpress.com/2013/11/13/adm-novell-system-plikow/

Brett Knight. (2024, June 4). *Best SNMP Monitoring Software & Tools for Windows & Linux of 2020*. Software Portal. https://softwareportal.com/snmp-monitoring-tools-for-windows-and-linux/

Fatmasari, E. (2020). Analisa Perbandingan Clustered File System menggunakan OrangeFS dan Lustre. *Jurnal Aksara Komputer Terapan*, *9*(1), 20–26.

Hartawan, I. N. B., & Iswara, I. B. A. I. (2015). Analisis Penerapan AoE dan LVM sebagai Teknologi Berbagi Media Penyimpanan pada Multi Server. *S@CIES*, *5*(2), 91–95. https://doi.org/10.31598/sacies.v5i2.61

Iswara, I. B. A. I., & Yasa, I. P. P. K. (2021). Analisis dan Perbandingan Quality of Service Video Conference Jitsi dan Bigbluebutton pada Virtual Private Server. *Jurnal RESISTOR (Rekayasa Sistem Komputer)*, *4*(2), 192–203.

Julianti, M. R., Ramdhan, S., & Mulyana, A. (2019). Perancangan Server Cloud Computing Model Infrastructure As A Service berbasis Proxmox pada PT Fortuna Mediatama. *Academic Journal of Computer Science Research*, *1*(1), 1–6.

Keefe, D. (2023). *Deskripsi Stratis*. Github.Io. https://stratis-storage.github.io/

Kinza Yasar. (2024, May 21). *Virtual Server*. TechTarget. https://www.techtarget.com/searchnetworking/definition/virtual-server

Maktum, F., Mazharuddin, A., & Suadi, W. (2012). Hybrid File System pada NAND-Flash SSD dan HDD menggunakan File System in User Space (FUSE). *Jurnal Teknik Pomits*, *1*(1), 1–6.

Marc GUILLAUME. (2024, May 19). *Basic Debian Installation - Debian 5.0 Lenny*. Yakati.Com. https://www.yakati.com/art/installation-debian-de-base-debian-5-0-lenny.html

Mauro, D., & Schmidt, K. (2005). *Essential SNMP: Help for System and Network Administrators*. " O'Reilly Media, Inc."

Mell, P., & Grance, T. (2012, April 18). *The NIST Definition of Cloud Computing - Winthrop University*. The National Institute of Standards and Technology (NIST). http://faculty.winthrop.edu/domanm/csci411/handouts/nist.pdf

Muhyidin, A., & Candra, M. (2016). *Buku Ubuntu Server Fundamental (Ubuntu Camp 2016)* (A. Muhyidin, Ed.). Yogyakarta, ID Networkers.

NeoX. (2024, July 23). *Install CentOS Stream 9 Workstation (Gnome GUI)*. Ahelpme.Com. https://ahelpme.com/linux/centos-stream-9/install-centos-stream-9-workstation-gnome-gui/

Pratama, K. A. (2017). *Implementasi Clustered File System pada Datacenter berbasis Cloud Computing menggunakan CEPH*. Skripsi Sarjana, Politeknik Caltex Riau. https://opac.lib.pcr.ac.id/index.php?p=show_detail&id=9078&keywords=Implementasi+Clustered+File+System+pada+Datacenter+berbasis+Cloud+Computing+menggunakan+CEPH

Purwantoro E.S.G.S, S. (2022). Perbandingan Kinerja Clustered File System pada Cloud Storage menggunakan GlusterFS dan Ceph. *INOVTEK Polbeng - Seri Informatika*, *7*(2), 319. https://doi.org/10.35314/isi.v7i2.2753

Rahul Panwar. (2024, May 21). *Linux File System and Windows File System, Difference*. TechTarget. https://linuxexplore.com/2012/10/01/linux-file-system-and-windows-file-system-difference/

Shauli Zacks. (2024, July 4). *PRTG Manual: General Layout*. Kisslabs.Ch. https://www.kisslabs.ch/en/solution-informatique-prtg-network-monitor

Sigiro, F. N., Suhatman, R., & Ridha, M. A. F. (2017). Analisa Perbandingan Clustered File System menggunakan MooseFS dan GlusterFS. *Jurnal Aksara Komputer Terapan*, *6*(2), 28–31.

Sulthon, A. (2023, June 15). *Apa itu CentOS ?* DomaiNesia. https://www.domainesia.com/tips/apa-itu-centos/

Wijaya, E., & Sinisuka, S. P. E. S. G. (2022). Perbandingan Kinerja Clustered File System pada Cloud Storage menggunakan GlusterFS dan Ceph. *INOVTEK Polbeng-Seri Informatika*, *7*(2), 319–333. https://doi.org/10.35314/isi.v7i2.2753

LAMPIRAN A
Prosedur Instalasi dan Konfigurasi LVM

1. Install perangkat lunak manajemen Volume Logis (LVM).

```
[root@localhost sarah]# sudo dnf install lvm2 -y
CentOS Linux 8 - AppStream                      2.8 MB/s | 8.4 MB     00:03
CentOS Linux 8 - BaseOS                         2.1 MB/s | 4.6 MB     00:02
CentOS Linux 8 - Extras                          12 kB/s |  10 kB     00:00
Package lvm2-8:2.03.12-10.el8.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

2. Untuk menggunakan *disk* dalam grup volume, beri label sebagai volume fisik LVM dengan perintah `pvcreate`.

```
#pvcreate /dev/sdb /dev/sdc
```

```
[root@localhost sarah]# pvcreate /dev/sdb /dev/sdc
  Physical volume "/dev/sdb" successfully created.
  Physical volume "/dev/sdc" successfully created.
```

3. Gunakan perintah `pvs` untuk menampilkan hasil atribut grup volume baru.

```
#pvs
```

```
[root@localhost sarah]# pvs
  PV         VG Fmt  Attr PSize  PFree
  /dev/sdb      lvm2 ---  50.00g 50.00g
  /dev/sdc      lvm2 ---  50.00g 50.00g
```

4. Buat grup volume yang terdiri dari volume fisik LVM yang telah dibuat. Perintah berikut membuat grup volume vg01.

```
#vgcreate vg01 /dev/sdb /dev/sdc
```

```
[root@localhost sarah]# vgcreate vg01 /dev/sdb /dev/sdc
  Volume group "vg01" successfully created
```

5. Gunakan perintah `vgs` untuk menampilkan atribut grup volume baru.

```
#vgs
```

```
[root@localhost sarah]# vgs
  VG   #PV #LV #SN Attr   VSize  VFree
  vg01   2   0   0 wz--n- 99.99g 99.99g
```

6. Buat volume logis dari grup volume yang telah dibuat. Perintah berikut digunakan untuk membuat Logical Volume (LV) baru dengan menggunakan semua ruang yang tersedia dalam Volume Group (VG) yang ditentukan.

```
#lvcreate -l 100%FREE -n lv01 vg01
```

```
[root@localhost sarah]# lvcreate -l 100%FREE -n lv01 vg01
  Logical volume "lv01" created.
```

Perintah `-l` (extent size) pada `-l 100%FREE` digunakan untuk menentukan ukuran Logical Volume dalam unit extents. `100%FREE` berarti menggunakan semua ruang kosong yang tersedia di Volume Group. Perintah `-n` (name) pada `-n lv01` digunakan untuk memberikan nama pada Logical Volume yang baru, dalam hal ini `lv01`.

7. Gunakan perintah lvs untuk menampilkan informasi tentang *logical volumes* (volume logis), termasuk ukuran, status, dan grup volume tempat mereka berada.

```
#lvs
```

```
[root@localhost sarah]# lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv01 vg01 -wi-a----- 99.99g
```

8. Membuat sebuah direktori untuk tujuan *mounting* dengan perintah:

```
#mkdir /data
```

9. Buat sistem file pada volume logis. Perintah berikut membuat sistem file XFS pada volume logis.

```
#mkfs.xfs /dev/mapper/vg01-lv01
```

```
[root@localhost sarah]# mkfs.xfs /dev/mapper/vg01-lv01
meta-data=/dev/mapper/vg01-lv01  isize=512    agcount=4, agsize=6553088 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=26212352, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=12799, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

10. Konfigurasi file /etc/fstab dengan perintah nano

```
#nano /etc/fstab
```

Isi dengan baris berikut :

```
/dev/mapper/vg01-lv01 /data xfs  defaults
0 0
```

```
GNU nano 2.9.8                    /etc/fstab                         Modified

#
# /etc/fstab
# Created by anaconda on Fri Dec  4 17:37:12 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=efba87f2-a6d9-4aa8-9e3c-91de25086f64 /                       xfs     defaults        0 0
/dev/mapper/vg01-lv01       /data    xfs   defaults   0 0
```

11. Aktifkan direktori /data dengan perintah mount agar volume logis yang baru diformat dapat diakses.

```
#mount /data/
```

```
[root@localhost sarah]# mount /data/
mount: /data: /dev/mapper/vg01-lv01 already mounted on /data.
```

12. Tampilkan informasi tentang semua perangkat blok termasuk partisi dan volume logis.

```
[root@localhost sarah]# lsblk
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda            8:0    0   50G  0 disk
└─sda1         8:1    0   50G  0 part /
sdb            8:16   0   50G  0 disk
└─vg01-lv01 253:0    0  100G  0 lvm  /data
sdc            8:32   0   50G  0 disk
└─vg01-lv01 253:0    0  100G  0 lvm  /data
sr0           11:0    1    4M  0 rom
sr1           11:1    1 1024M  0 rom
```

LAMPIRAN B
Prosedur Instalasi dan Konfigurasi Stratis

1. Install paket yang menyediakan layanan Stratis.

```
#yum install stratisd stratis-cli
```

```
[root@localhost sarah]# yum install stratisd stratis-cli
Last metadata expiration check: 1:40:31 ago on Tue 02 Jul 2024 03:17:32 AM UTC.
Dependencies resolved.
================================================================================ Package
               Architecture Version              Repository       Size
================================================================================Installing:
 stratis-cli                                noarch       2.4.2-1.el8          appstream        86 k
 stratisd                                   x86_64       2.4.2-2.el8          appstream       4.1 M
Installing dependencies:
 clevis                                     x86_64       15-1.el8             appstream        57 k
 cryptsetup                                 x86_64       2.3.3-4.el8          baseos          190 k
 device-mapper-persistent-data              x86_64       0.9.0-4.el8          baseos          925 k
 jose                                       x86_64       10-2.el8             appstream        58 k
 jq                                         x86_64       1.5-12.el8           appstream       161 k
 libaio                                     x86_64       0.3.112-1.el8        baseos           33 k
 libjose                                    x86_64       10-2.el8             appstream        64 k
 libluksmeta                                x86_64       9-4.el8              appstream        27 k
 luksmeta                                   x86_64       9-4.el8              appstream        22 k
 oniguruma                                  x86_64       6.8.2-2.el8          appstream       187 k
 python3-dbus-client-gen                    noarch       0.4-1.el8            appstream        26 k
 python3-dbus-python-client-gen             noarch       0.7-3.el8            appstream        28 k
 python3-dbus-signature-pyparsing           noarch       0.03-2.el8           appstream        18 k
 python3-into-dbus-python                   noarch       0.06-2.el8           appstream        27 k

Installed:
  clevis-15-1.el8.x86_64                                clevis-luks-15-1.el8.x86_64
  cryptsetup-2.3.3-4.el8.x86_64                         device-mapper-persistent-data-0.9.0-4.el8.x86_64
  jose-10-2.el8.x86_64                                  jq-1.5-12.el8.x86_64
  libaio-0.3.112-1.el8.x86_64                           libjose-10-2.el8.x86_64
  libluksmeta-9-4.el8.x86_64                            luksmeta-9-4.el8.x86_64
  oniguruma-6.8.2-2.el8.x86_64                          python3-dbus-client-gen-0.4-1.el8.noarch
  python3-dbus-python-client-gen-0.7-3.el8.noarch       python3-dbus-signature-pyparsing-0.03-2.el8.noarch
  python3-into-dbus-python-0.06-2.el8.noarch            python3-justbases-0.14-4.el8.noarch
  python3-justbytes-0.14-2.el8.noarch                   python3-psutil-5.4.3-11.el8.x86_64
  python3-pyparsing-2.1.10-7.el8.noarch                 stratis-cli-2.4.2-1.el8.noarch
  stratisd-2.4.2-2.el8.x86_64                           tpm2-tools-4.1.1-5.el8.x86_64

Complete!
```

2. Verifikasi bahwa layanan stratisd diaktifkan:

```
#systemctl enable --now stratisd
```

```
[root@localhost sarah]# systemctl enable --now stratisd
```

3. Hapus semua sistem file atau tabel partisi yang ada di setiap perangkat blok yang ingin digunakan di *pool* Stratis:

```
#wipefs --all /dev/sdb /dev/sdc
```

```
[root@localhost sarah]# wipefs --all /dev/sdb /dev/sdc
```

4. Buat *pool* Stratis baru yang tidak terenkripsi pada perangkat blok yang dipilih, di mana perangkat blok adalah jalur ke perangkat blok yang kosong atau dihapus.

```
#stratis pool create my-pool /dev/sdb
/dev/sdc
```

```
[root@localhost sarah]# stratis pool create my-pool /dev/sdb /dev/sdc
```

5. Verifikasi bahwa kumpulan Stratis baru telah dibuat:

```
#stratis pool list
```

```
[root@localhost sarah]# stratis pool list
Name                    Total Physical   Properties                                UUID
my-pool   100 GiB / 49.66 MiB / 99.95 GiB      ~Ca,~Cr   8da7369f-5f6e-4232-a440-f6b90681fc9c
```

6. Untuk membuat sistem file Stratis di dalam *pool*, gunakan:

```
#stratis filesystem create --size 100GiB
my-pool filesystem1
```

```
[root@localhost sarah]# stratis filesystem create my-pool filesystem1
```

7. List sistem file dalam *pool* untuk memeriksa apakah sistem file Stratis telah dibuat:

```
#stratis fs list my-pool
```

```
[root@localhost sarah]# stratis fs list my-pool
Pool Name   Name         Used     Created            Device                          UUID
my-pool     filesystem1  546 MiB  Jul 02 2024 05:22  /dev/stratis/my-pool/filesystem1  4269cd6e-23ba-47e8-9808-e735eef2c88a
```

8. Gunakan perintah mkdir untuk membuat direktori baru di dalam sistem file.

```
#mkdir /data
```

```
[root@localhost sarah]# mkdir /data
```

9. Lakukan mounting pada sistem file di direktori /dev/stratis:

```
#mount    /dev/stratis/my-pool/filesystem1
/data
```

```
[root@localhost sarah]# mount /dev/stratis/my-pool/filesystem1 /data
```

10. Gunakan perintah pwd untuk mengecek direktori kerja saat ini di dalam system file.

```
#pwd
```

```
[root@localhost sarah]# pwd
/home/sarah
```

11. Gunakan perintah lsblk untuk menampilkan perangkat blok yang dikelola oleh Stratis, termasuk *pool* penyimpanan, volume logis Stratis, dan volume file.

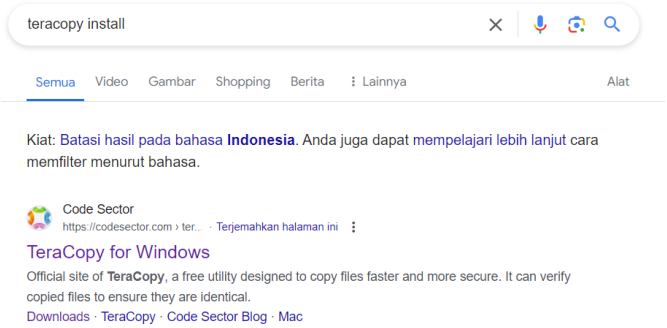
```
#lsblk
```

```
[root@stratis sarah]# lsblk
NAME                                                                                                  MAJ:MIN RM  SIZE RO TYPE    MOUNTPOINTS
sda                                                                                                     8:0    0   50G  0 disk
├─sda1                                                                                                  8:1    0    1M  0 part
├─sda2                                                                                                  8:2    0  200M  0 part    /efi
│                                                                                                                               /boot/efi
├─sda3                                                                                                  8:3    0    1G  0 part    /boot
└─sda4                                                                                                  8:4    0 48.8G  0 part    /
sdb                                                                                                     8:16   0   50G  0 disk
└─stratis-1-private-4225305c1ebb406baad7732970b41f7d-physical-originsub                               253:0    0 50.6G  0 stratis
  ├─stratis-1-private-4225305c1ebb406baad7732970b41f7d-flex-thinmeta                                  253:1    0   29M  0 stratis
  │ └─stratis-1-private-4225305c1ebb406baad7732970b41f7d-thinpool-pool                                253:4    0   50G  0 stratis
  │   └─stratis-1-4225305c1ebb406baad7732970b41f7d-thin-fs-f700b4ff55d74357bf29097bb072107b          253:5    0  100G  0 stratis /data
  ├─stratis-1-private-4225305c1ebb406baad7732970b41f7d-flex-thindata                                  253:2    0   50G  0 stratis
  │ └─stratis-1-private-4225305c1ebb406baad7732970b41f7d-thinpool-pool                                253:4    0   50G  0 stratis
  │   └─stratis-1-4225305c1ebb406baad7732970b41f7d-thin-fs-f700b4ff55d74357bf29097bb072107b          253:5    0  100G  0 stratis /data
  └─stratis-1-private-4225305c1ebb406baad7732970b41f7d-flex-mdv                                       253:3    0  512M  0 stratis
sdc                                                                                                     8:32   0   50G  0 disk
└─stratis-1-private-4225305c1ebb406baad7732970b41f7d-physical-originsub                               253:0    0 50.6G  0 stratis
  ├─stratis-1-private-4225305c1ebb406baad7732970b41f7d-flex-thinmeta                                  253:1    0   29M  0 stratis
  │ └─stratis-1-private-4225305c1ebb406baad7732970b41f7d-thinpool-pool                                253:4    0   50G  0 stratis
  │   └─stratis-1-4225305c1ebb406baad7732970b41f7d-thin-fs-f700b4ff55d74357bf29097bb072107b          253:5    0  100G  0 stratis /data
  ├─stratis-1-private-4225305c1ebb406baad7732970b41f7d-flex-thindata                                  253:2    0   50G  0 stratis
  │ └─stratis-1-private-4225305c1ebb406baad7732970b41f7d-thinpool-pool                                253:4    0   50G  0 stratis
  │   └─stratis-1-4225305c1ebb406baad7732970b41f7d-thin-fs-f700b4ff55d74357bf29097bb072107b          253:5    0  100G  0 stratis /data
  └─stratis-1-private-4225305c1ebb406baad7732970b41f7d-flex-mdv                                       253:3    0  512M  0 stratis
sr0                                                                                                    11:0    1    4M  0 rom
sr1                                                                                                    11:1    1 1024M  0 rom
```

LAMPIRAN C
Prosedur Instalasi Teracopy

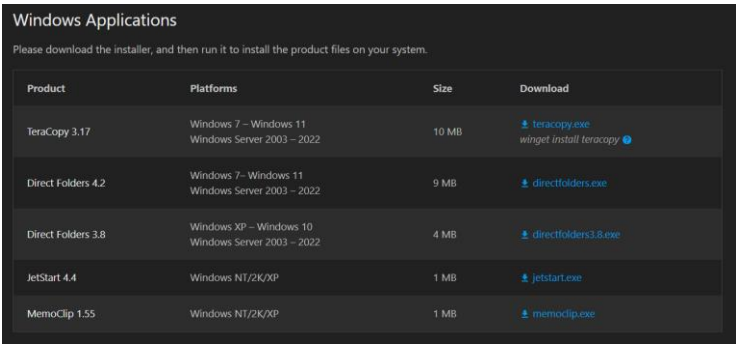
A. Prosedur Instalasi Teracopy

1. Kunjungi situs resmi Teracopy pada Code Sector Teracopy for Windows.

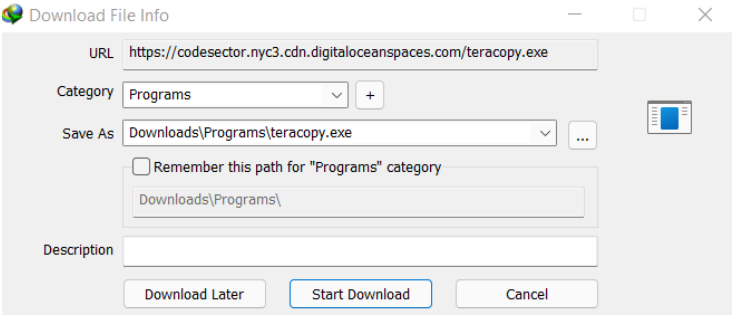


2. Pada halaman utama situs TeraCopy, cari dan klik tombol "*Download*". Pastikan memilih versi yang sesuai untuk sistem operasi (Windows).

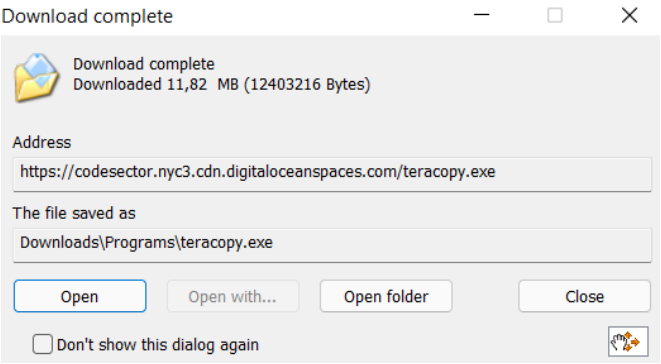
3. Pada bagian Windows Application terdapat berbagai *product*, lalu pilih yang Teracopy 3.17 dengan platform Windows 7 – Windows 11, dan klik *download* teracopy.exe.



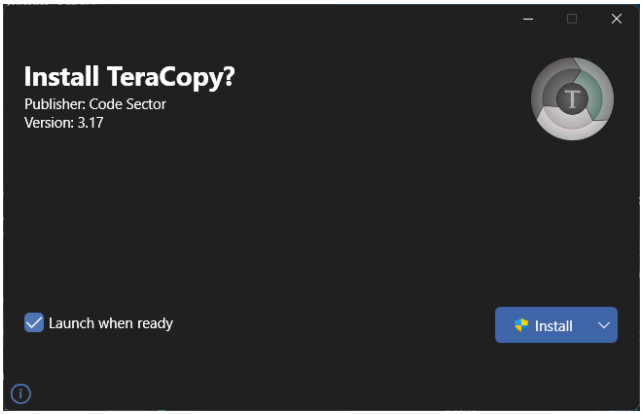
4. Lalu, atur tempat penyimpanan aplikasi pada *Downloads*\Programs\teracopy.exe, dan klik start *download*.



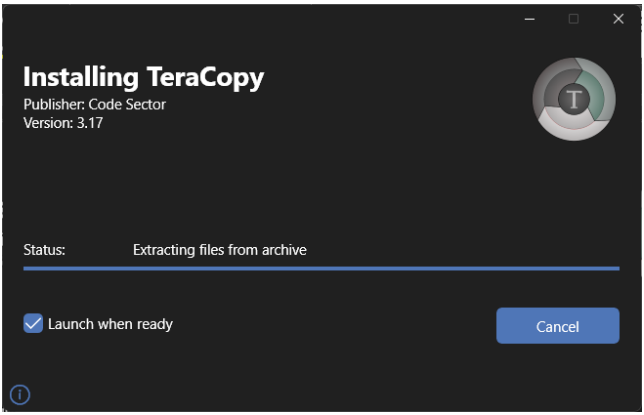
5. Setelah unduhan selesai, buka folder tempat file diunduh dan klik “Open”.



6. Klik dua kali pada file installer yang telah diunduh untuk menjalankannya. Ikuti petunjuk yang muncul di layar untuk menginstal TeraCopy. Ini biasanya termasuk menyetujui syarat dan ketentuan, memilih lokasi instalasi, dan memilih apakah ingin membuat pintasan di desktop.



7. Setelah proses instalasi selesai, klik tombol "Finish". TeraCopy sekarang sudah terinstal di komputer Anda.



8. Lakukan pengujian kecepatan *transfer* file.