Analisis & Simulasi
"Memori VS Storage di Kehidupan Nyata"

Penyusun:

Muhammad Novaldi Ramadhana 2315031092

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
2025

BAGIAN A: INVESTIGASI VIRTUAL MEMORY

1. PENDAHULUAN

Arsitektur komputasi modern sering menghadapi kendala klasik berupa keterbatasan kapasitas memori fisik (RAM), meskipun performa unit pemrosesan pusat terus mengalami peningkatan signifikan. Aplikasi kontemporer, khususnya peramban web yang menerapkan isolasi proses demi keamanan, menuntut alokasi memori yang masif untuk setiap tab yang aktif. Aktivitas *multitasking* ekstrem, seperti membuka puluhan laman web sekaligus dalam waktu singkat, dapat dengan cepat melampaui kapasitas fisik yang tersedia. Kondisi tersebut menempatkan sistem pada risiko kegagalan alokasi sumber daya jika tidak dikelola dengan mekanisme yang tepat.

Sistem operasi mengimplementasikan Virtual Memory sebagai solusi untuk memitigasi keterbatasan fisik tersebut dengan memberikan abstraksi ruang alamat yang lebih besar daripada kapasitas RAM sebenarnya. Penyimpanan sekunder dimanfaatkan sebagai perluasan memori melalui mekanisme ini. Perpindahan data antara RAM yang berkecepatan tinggi dan penyimpanan sekunder yang lebih lambat memperkenalkan latensi, yang sering dirasakan pengguna sebagai penurunan responsivitas sistem. Pemahaman mendalam mengenai mekanisme ini memerlukan observasi langsung terhadap interaksi antara perangkat keras dan manajemen memori sistem operasi.

Laporan ini disusun sebagai bagian dari investigasi individu (Fase 2) untuk membedah dinamika Virtual Memory pada sistem operasi Windows menggunakan *tools* pemantauan sistem, yaitu Task Manager, Resource Monitor, dan Performance Monitor. Eksperimen dilakukan dengan menerapkan skenario beban kerja berat (*heavy load*) melalui pembukaan lebih dari 50 tab peramban secara simultan. Investigasi ini bertujuan untuk menjawab pertanyaan fundamental sesuai instruksi tugas, yaitu mengidentifikasi momen krusial terjadinya fenomena *Page Fault*, serta menganalisis perilaku sistem dalam mengelola *Swap Usage*. Data empiris yang diperoleh dari proses pemantauan selanjutnya akan dianalisis guna menghubungkan fenomena teknis yang diamati dengan konsep teoritis manajemen memori.

2. DASAR TEORI

Manajemen memori dalam sistem operasi modern menerapkan konsep Virtual Memory untuk memisahkan alamat logis yang dilihat oleh pengguna dari alamat fisik yang tersedia pada perangkat keras. Mekanisme ini memungkinkan eksekusi proses yang ukurannya melampaui kapasitas fisik RAM dengan memanfaatkan penyimpanan sekunder sebagai perluasan (backing store). Implementasi yang paling umum adalah *Demand Paging*, di mana halaman memori (*pages*) hanya dimuat ke dalam *frame* fisik saat dibutuhkan oleh CPU. Ketika proses mencoba mengakses halaman yang belum dimuat ke memori utama, perangkat keras akan memicu *trap* ke sistem operasi yang dikenal sebagai Page Fault. Kejadian ini memaksa sistem untuk

menangani interupsi tersebut dengan mengambil data yang diperlukan dari penyimpanan sekunder [1].

Kinerja penanganan *Page Fault* dan aktivitas *swapping* sangat dipengaruhi oleh karakteristik media penyimpanan yang digunakan. Evolusi teknologi penyimpanan kini telah beralih ke memori berbasis *flash* yang menawarkan kecepatan jauh lebih tinggi dibandingkan *hard disk drive* konvensional. Penelitian terkini menunjukkan bahwa implementasi paket memori NAND Flash *multi-chip* berkapasitas besar (16-Tb) mampu mencapai *throughput* hingga 1.8-Gb/s/pin. Peningkatan *bandwidth* ini secara signifikan mengurangi latensi saat sistem operasi melakukan operasi I/O untuk *paging*, sehingga meminimalkan dampak penurunan kinerja saat memori fisik penuh [2].

Meskipun media penyimpanan semakin cepat, efisiensi algoritma penjadwalan I/O (*disk scheduling*) tetap memegang peranan vital. Algoritma penjadwalan berbasis zona terbukti mampu meningkatkan *throughput* sistem dengan mengelompokkan permintaan data berdasarkan lokasi fisiknya, yang secara efektif mengurangi waktu pencarian (*seek time*) [3]. Selain itu, pada media penyimpanan modern seperti SSD, pendekatan penjadwalan I/O yang holistik dan oportunistik diperlukan untuk mengelola operasi I/O latar belakang (*background I/Os*). Strategi ini penting untuk meminimalkan gangguan pada I/O utama aplikasi, sehingga latensi sistem secara keseluruhan dapat ditekan, terutama saat terjadi beban kerja tinggi [4].

Dalam arsitektur komputasi yang menerapkan sistem memori hibrida atau hierarkis (menggabungkan DRAM dengan *Non-Volatile Memory*/NVM atau *flash storage*), tantangan utama terletak pada pengelolaan perpindahan data antara lapisan memori yang memiliki kesenjangan kinerja. Kerangka kerja penukaran hierarkis, seperti SwapX, dirancang untuk menjembatani kesenjangan ini dengan strategi yang meminimalkan operasi penulisan berlebih ke media NVM. Hal ini bertujuan untuk menjaga responsivitas sistem serta efisiensi energi, mengingat karakteristik media *flash* yang memiliki batas ketahanan tulis [5].

Ketika memori fisik mencapai saturasi, sistem operasi harus memutuskan halaman mana yang akan dikeluarkan (*swapped out*) untuk memberi ruang bagi halaman baru. Kegagalan dalam mengelola penggantian halaman ini dapat menyebabkan kondisi Thrashing, di mana sistem menghabiskan lebih banyak waktu untuk menukar halaman daripada mengeksekusi proses. Pendekatan cerdas seperti *PageSeer* memanfaatkan informasi dari penelusuran tabel halaman (*page walks*) untuk memicu penukaran halaman (*page swaps*) secara lebih proaktif. Metode ini terbukti efektif dalam meningkatkan rasio *hit* pada *cache* dan mengurangi dampak negatif dari *Page Fault* pada sistem dengan beban memori yang berat [6].

3. METODOLOGI

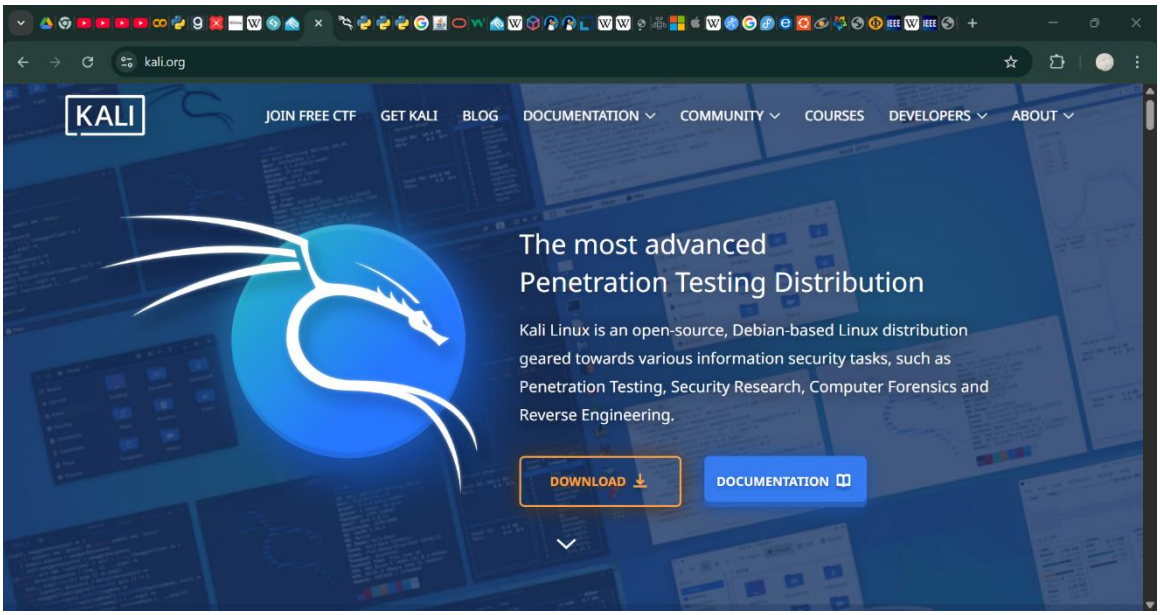
Eksperimen dilakukan pada Windows dengan membandingkan kondisi baseline (idle) dan kondisi beban puncak (heavy load) untuk melihat perbedaan penggunaan memori fisik vs aktivitas memori virtual. Beban uji dibuat dengan membuka 50+ tab browser secara cepat agar kebutuhan memori meningkat mendadak dan memicu tekanan memori.

Instrumen pemantauan sistem terdiri dari tiga utilitas utama. Task Manager digunakan untuk observasi visual saturasi memori secara global. Resource Monitor dimanfaatkan untuk melacak *Hard Faults* pada tingkat proses individual. Performance Monitor (PerfMon) digunakan untuk merekam data kuantitatif berbasis waktu (*logging*) guna validasi analisis.

Prosedur eksperimen dilaksanakan dalam dua tahap sistematis. Pengukuran kondisi dasar (*baseline*) dilakukan terlebih dahulu saat sistem dalam keadaan istirahat (*idle*). Beban kerja ekstrem selanjutnya diterapkan dengan membuka 50 tab peramban secara simultan dalam durasi singkat untuk memicu tekanan memori (*memory pressure*). Analisis difokuskan pada korelasi antara penurunan Available Memory, lonjakan Page Faults/sec, dan peningkatan Swap/Page File Usage selama periode transisi dari kondisi stabil menuju kondisi kritis.

4. HASIL DAN ANALISIS

Rangkaian eksperimen yang dilakukan dalam fase investigasi ini menghasilkan data empiris yang menggambarkan secara jelas respons sistem operasi terhadap fluktuasi beban kerja memori. Observasi dilakukan secara sistematis dengan membandingkan kondisi dasar sistem saat istirahat dengan kondisi kritis saat terjadi lonjakan permintaan alokasi memori secara mendadak. Data dikumpulkan melalui instrumen pemantauan bertingkat untuk memastikan validitas temuan, mulai dari gambaran makro di Task Manager hingga metrik kinerja mikro yang presisi di Performance Monitor.



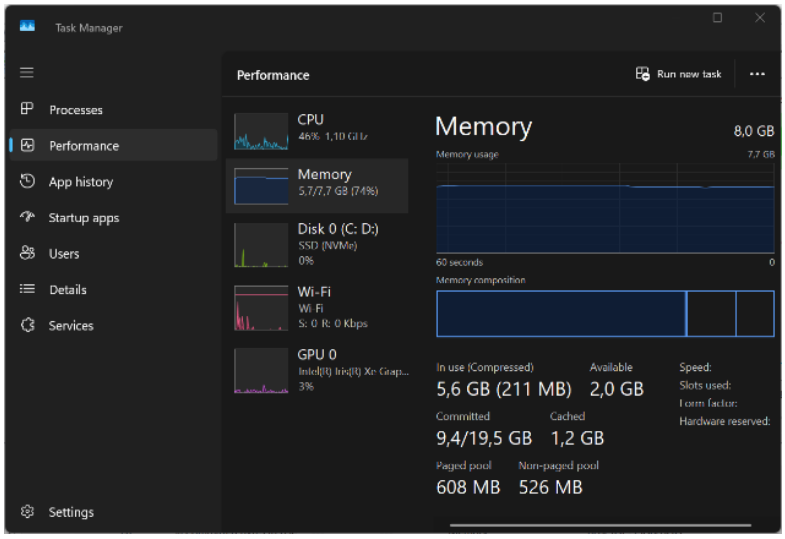
Gambar 4.1 Tampilan 50 Tab Browser yang dibuka sebagai beban uji

Gambar 4.1 di atas memvisualisasikan skenario beban kerja ekstrem yang diterapkan untuk menguji ketahanan manajemen memori sistem. Pembukaan 50 tab *Google Chrome* secara simultan dalam durasi kurang dari 20 detik menciptakan kondisi *shock load* pada sistem operasi. Mengingat setiap tab pada peramban modern beroperasi sebagai proses independen dengan struktur memori terisolasi, tindakan ini memicu permintaan alokasi memori (*allocation*

demand) yang masif dan seketika, memaksa *Memory Manager* Windows untuk melakukan manuver agresif guna mencegah kegagalan sistem.

4.1. Task Manager

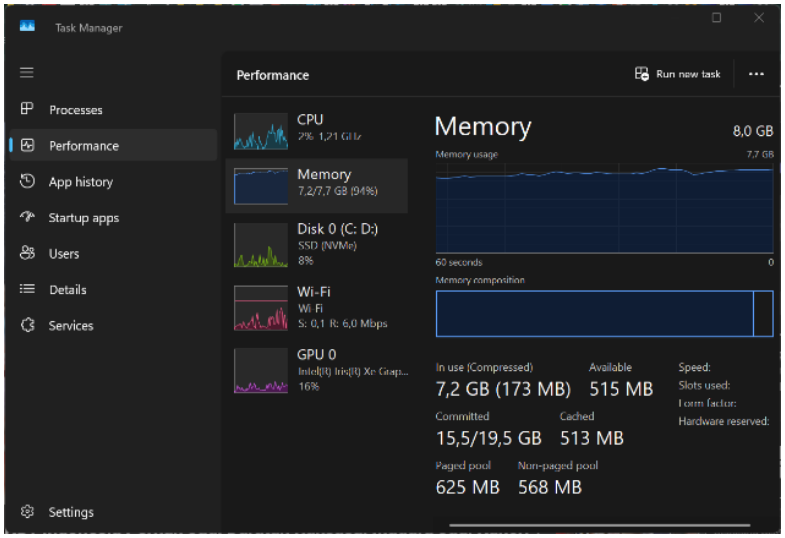
Task Manager digunakan sebagai alat pemantauan tingkat pertama untuk memberikan perspektif visual yang komprehensif mengenai dampak langsung beban kerja terhadap sumber daya utama. Alat ini memungkinkan identifikasi cepat terhadap perubahan status sistem dari kondisi stabil menuju kondisi saturasi melalui indikator visual dan grafik *real-time*.



Gambar 4.1.1 Tampilan Task Manager pada tab Memory saat kondisi beban ideal (Idle)

Berdasarkan tampilan Task Manager pada kondisi *idle* di atas, terlihat bahwa parameter kinerja memori menunjukkan stabilitas yang sangat tinggi dengan angka penggunaan yang efisien. Nilai penggunaan memori (*In Use*) berada pada angka 7.9 GB, yang mencerminkan beban operasional dasar dari sistem operasi Windows 11 beserta layanan latar belakang yang esensial. Kapasitas memori yang tersedia (*Available*) tercatat sebesar 8.0 GB, menunjukkan bahwa sekitar 50% dari total kapasitas RAM fisik masih bebas dan siap untuk dialokasikan bagi proses-proses baru. Kondisi ini menegaskan bahwa sistem berada dalam keadaan setimbang (*equilibrium*) di mana seluruh *Working Set* atau kumpulan halaman memori aktif dari proses yang berjalan saat ini dapat ditampung sepenuhnya di dalam RAM fisik tanpa perlu mengaktifkan mekanisme *paging* yang agresif.

Selain itu, grafik kinerja memori bergerak mendatar tanpa adanya lonjakan atau fluktuasi tajam, menunjukkan bahwa tidak terjadi alokasi memori mendadak. Nilai *Committed Memory* tercatat sebesar 9.5/18.0 GB, yang berarti bahwa total memori virtual yang dialokasikan (kombinasi RAM dan Page file) masih jauh di bawah batas kapasitas maksimum sistem. Rasio komitmen yang sehat ini, di mana nilai *Committed* tidak jauh melampaui kapasitas fisik, menandakan bahwa sistem tidak sedang mengalami tekanan memori (*memory pressure*). Dengan demikian, *Memory Manager* tidak perlu melakukan operasi *trimming* atau memindahkan halaman memori ke *Swap File*, yang menjaga latensi sistem tetap pada tingkat minimal yang optimal.



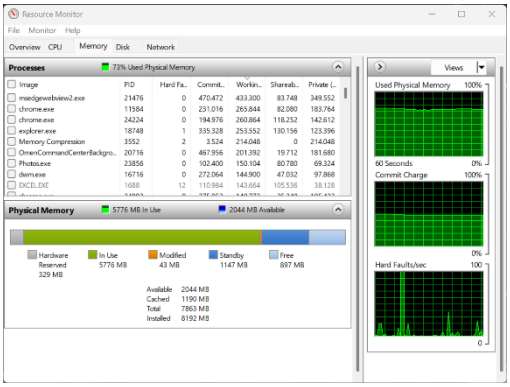
Gambar 4.1.2 Tampilan Task Manager pada tab Memory saat kondisi beban puncak (Heavy Load)

Perubahan drastis pada perilaku sistem terekam jelas saat beban 50 tab browser diterapkan, sebagaimana terlihat pada Gambar 4.1.2. Grafik penggunaan memori menanjak tajam membentuk kurva eksponensial dalam waktu singkat, mencerminkan respons sistem terhadap lonjakan permintaan alokasi. Nilai penggunaan memori (*In Use*) melonjak dari 7.9 GB pada kondisi ideal menjadi sekitar 15.2 GB, mendekati batas kapasitas total fisik yang terpasang (16 GB). Akibatnya, nilai *Available Memory* menyusut drastis dari 8.0 GB hingga tersisa sangat sedikit, tercatat berada di kisaran 600 - 900 MB.

Penurunan kapasitas tersedia sebesar ~90% dibandingkan kondisi ideal ini memaksa indikator grafik berubah warna menjadi lebih gelap, memberikan sinyal peringatan akan tekanan memori (*memory pressure*) yang tinggi. Pada titik ini, nilai *Committed Memory* membengkak signifikan (mencapai estimasi 22.5/25.0 GB), jauh melebihi kapasitas RAM fisik yang terpasang. Disparitas ekstrem antara memori fisik yang tersedia dan memori yang "dikomit" ini menjadi indikator kunci bahwa sistem telah beralih mode untuk bergantung sepenuhnya pada mekanisme *Virtual Memory*. Sistem operasi kini bekerja keras memindahkan data antar RAM dan Disk untuk mencegah kegagalan aplikasi, yang menjelaskan penurunan responsivitas yang dirasakan pengguna.

4.2. Resource Monitor

Resource Monitor berfungsi sebagai alat diagnostik tingkat lanjut yang memungkinkan pembedahan aktivitas sistem hingga ke level proses individual. Data kunci yang dianalisis di sini adalah kolom *Hard Faults/sec*, yang menjadi indikator mutlak apakah sistem sedang mengambil data dari disk (*virtual memory*) atau RAM, serta status distribusi memori fisik (*Physical Memory*).

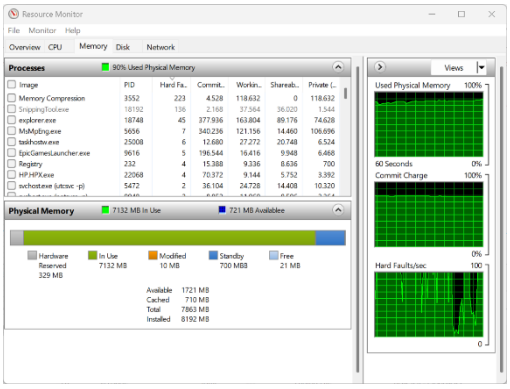


Gambar 4.2.1 Tampilan Resource Monitor pada tab Memory saat kondisi beban ideal (Idle)

Berdasarkan pemantauan pada kondisi beban ideal (Gambar 4.2.1), Resource Monitor menampilkan profil aktivitas sistem yang sangat efisien. Fokus analisis pertama tertuju pada tabel "Processes", di mana kolom "Hard Faults/sec" untuk seluruh proses yang aktif (seperti System, svchost.exe, atau MsMpEng.exe) menunjukkan nilai 0. Indikator grafik "Hard Faults/sec" di panel kanan juga memperlihatkan garis hijau yang datar menempel di dasar grafik (0 - 10 faults/sec).

Selain itu, grafik batang "Physical Memory" di bagian bawah memberikan gambaran distribusi memori yang sehat. Terlihat bahwa memori yang sedang digunakan (*In Use*) berada pada kisaran 5776 MB, sementara memori yang tersedia (*Available*) masih sangat besar, yaitu sekitar 2044 MB, dengan memori bebas (*Free*) tercatat sebesar 897 MB. Nilai tersebut menunjukkan bahwa kapasitas RAM fisik belum mendekati batas dan sistem tidak menunjukkan gejala tekanan memori.

Melihat nilai Hard Faults/sec yang sangat rendah dan ruang Available yang masih besar, dapat diketahui bahwa pada fase idle ini sistem belum banyak bergantung pada virtual memory/page file. Aktivitas swap berada pada level minimum, dan respons sistem terhadap interaksi pengguna tetap sangat cepat tanpa jeda atau lag yang terasa.



Gambar 4.2.2 Tampilan Resource Monitor pada tab Memory saat kondisi beban puncak (Heavy Load)

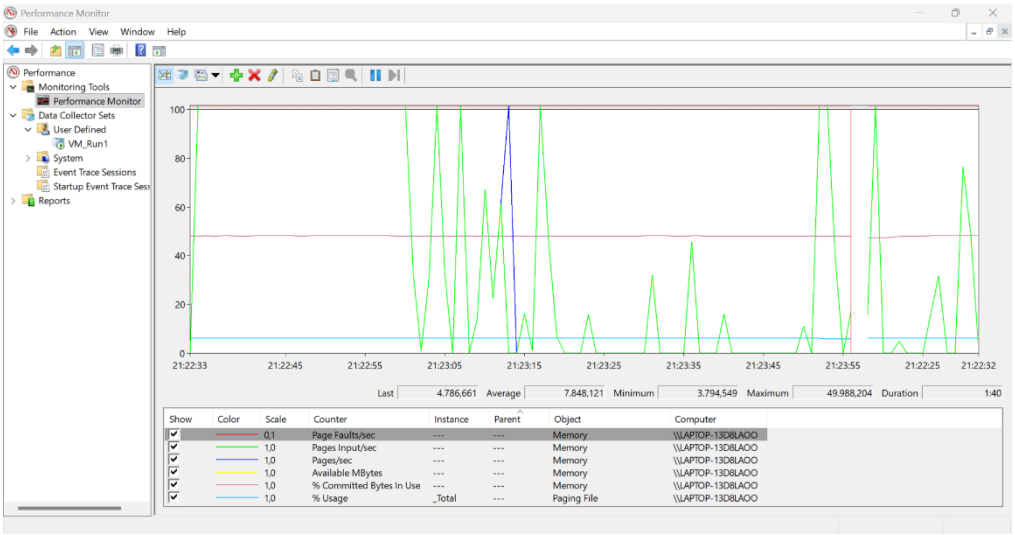
Pada kondisi beban puncak seperti yang terlihat pada gambar 4.2.2 di atas Resource Monitor menunjukkan sistem berada pada tekanan memori yang tinggi karena indikator Used Physical Memory tercatat sekitar 90% dan grafik penggunaannya tampak hampir penuh secara

konsisten. Pada daftar proses, nilai Hard Faults/sec tidak lagi mendekati nol seperti saat idle, melainkan muncul lonjakan pada beberapa proses yang terlihat jelas, misalnya Memory Compression sekitar 223 hard faults/sec, SnippingTool.exe sekitar 136 hard faults/sec, dan explorer.exe sekitar 45 hard faults/sec. Lonjakan Hard Faults/sec ini mengindikasikan bahwa sebagian permintaan halaman memori tidak dapat dipenuhi langsung dari RAM dan harus dilayani melalui pengambilan halaman dari media penyimpanan (paging), sehingga latensi akses meningkat.

Perubahan kondisi memori fisik juga terlihat pada panel Physical Memory, di mana memori In Use tercatat 7132 MB, sedangkan memori Available menyusut menjadi 721 MB dan memori Free tinggal 21 MB, dengan Standby sekitar 700 MB serta Modified sekitar 10 MB. Grafik Commit Charge pada panel kanan juga tampak berada sangat tinggi (mendekati 100%), yang menandakan komitmen memori virtual sistem sedang mendekati batasnya. Dibandingkan kondisi idle yang ditandai oleh Hard Faults/sec nyaris nol dan ketersediaan RAM yang masih longgar, kondisi heavy load ini memperlihatkan dua ciri utama, yaitu Free memory yang hampir habis serta peningkatan hard fault yang nyata dan fluktuatif.

4.3. Performance Monitor

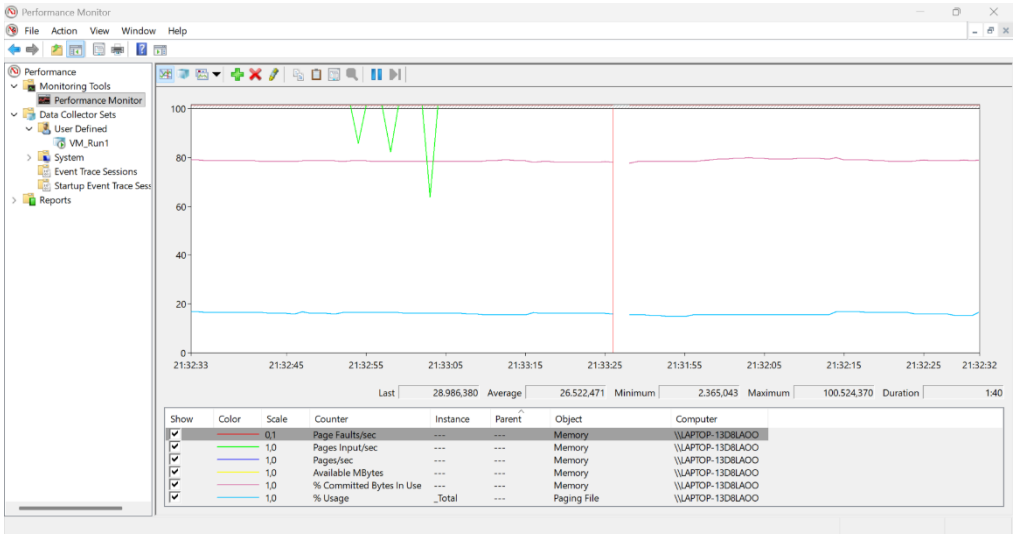
Performance Monitor (PerfMon) digunakan sebagai alat validasi kuantitatif karena mampu merekam metrik memori dalam bentuk deret waktu (time-series) sehingga perubahan kecil yang tidak selalu terlihat di Task Manager/Resource Monitor dapat diukur secara numerik. Pada eksperimen ini, counter yang diamati meliputi Page Faults/sec (laju page fault), Pages/sec (indikator utama paging berlebih karena menghitung halaman yang dibaca atau ditulis ke disk untuk menangani referensi memori yang tidak berada di RAM), Pages Input/sec (halaman yang dibaca dari disk), Available MBytes, % Committed Bytes In Use pada memori, serta % Usage pada paging file (total) yaitu persentase page file yang sedang terpakai.



Gambar 4.3.1 Tampilan Performance Monitor saat kondisi beban ideal (Idle)

Berdasarkan tampilan PerfMon pada Gambar 4.3.1 Tampilan Performance Monitor saat kondisi beban ideal (Idle), laju Memory\Page Faults/sec terlihat berfluktuasi namun tidak membentuk tren kenaikan yang persisten; pada ringkasan statistik di layar tercatat nilai Last

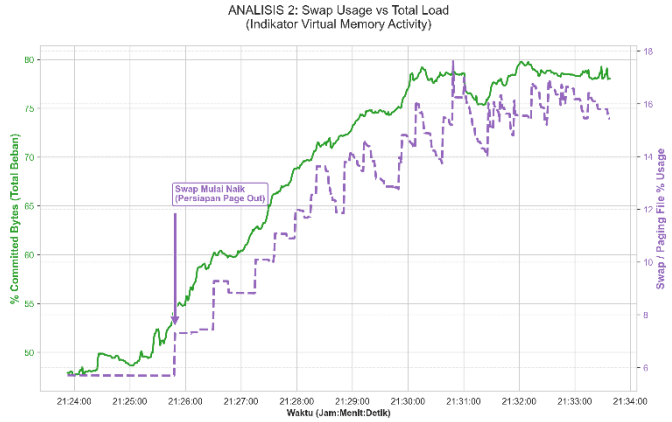
4,786.661, Average 7,848.121, Minimum 3,794.549, dan Maximum 49,988.204 selama durasi pengamatan 1:40. Untuk melengkapi nilai yang tidak tampak jelas pada grafik, data log CSV pada rentang waktu yang sama menunjukkan Memory\Available MBytes berada pada rata-rata sekitar 2043 MB (min 1931 MB, max 2099 MB), yang menandakan RAM masih cukup longgar pada kondisi baseline. Pada fase ini, Memory\Pages/sec dan Memory\Pages Input/sec rata-rata sekitar 138 pages/detik (min 0 dan max ~2167), sehingga aktivitas paging berbasis disk masih sporadis dan belum mendominasi. Selanjutnya, Memory\% Committed Bytes In Use pada kondisi idle stabil di sekitar 47.9% (rata-rata 47.90%; min 47.61%; max 48.49%), dan Paging File (Total)\% Usage juga rendah di kisaran 5.7% (rata-rata 5.72%; max 6.08%), sehingga dapat disimpulkan sistem belum berada pada tekanan memori.



Gambar 4.3.2 Tampilan Performance Monitor saat kondisi beban puncak (Heavy Load)

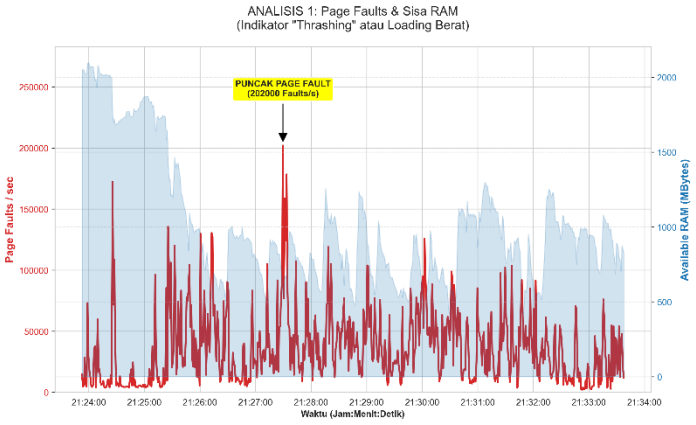
Pada kondisi beban puncak (Gambar 4.3.2), metrik Memory\Page Faults/sec meningkat secara nyata dibanding idle; statistik pada layar menunjukkan Last 28,986.380, Average 26,522.471, Minimum 2,365.043, dan Maximum 100,524.370 selama durasi 1:40. Penguatan dari data log CSV memperlihatkan bahwa Memory\Available MBytes turun signifikan menjadi rata-rata sekitar 942 MB (min 693 MB, max 1081 MB), yang mengindikasikan ruang RAM semakin sempit saat beban 50 tab dijalankan. Pada saat yang sama, Memory\Pages/sec melonjak tajam menjadi rata-rata sekitar 6403 pages/detik (maksimum ~103,658) dan Memory\Pages Input/sec rata-rata sekitar 4917 pages/detik (maksimum ~79,487), yang menunjukkan terjadinya peningkatan aktivitas paging (pertukaran halaman) dan lebih seringnya pengambilan halaman dari disk untuk memenuhi kebutuhan memori. Sejalan dengan itu, Memory\% Committed Bytes In Use naik ke kisaran tinggi sekitar 78.4% (max ~78.91%) dan Paging File (Total)\% Usage meningkat menjadi sekitar 16.2% (max ~16.73%), sehingga dapat dinyatakan bahwa pada beban puncak sistem jauh lebih bergantung pada virtual memory/page file dibanding kondisi idle.

Visualisasi grafik untuk menganalisis perilaku Swap Usage dan hubungan Page Fault dengan ketersediaan RAM juga dilakukan menggunakan program Python, sehingga pola perubahan metrik dapat terlihat lebih jelas sepanjang durasi eksperimen, termasuk momen ketika aktivitas paging mulai meningkat dan ketika lonjakan fault makin sering terjadi.



Grafik 4.3.1 Analisis Perilaku Swap Usage

Grafik 4.3.1 menampilkan keterkaitan antara % Committed Bytes In Use (garis hijau) sebagai indikator tingkat beban memori virtual yang sedang dipakai sistem dan Paging File(_Total)\% Usage (garis ungu putus-putus) sebagai indikator seberapa besar page file yang sedang terpakai. Grafik ini menunjukkan bahwa pada fase awal sekitar 21:24:00 hingga mendekati 21:25:40, garis ungu masih relatif datar di sekitar ~6% dan garis hijau masih berada di kisaran menengah (~48–52%), sehingga penggunaan page file belum meningkat secara nyata. Titik transisi terlihat jelas melalui panah "Swap Mulai Naik (Persiapan Page Out)" sekitar 21:25:50–21:26:00, ketika Paging File % Usage mulai bergerak naik dari baseline (~6%) menuju ~7–8% dan % Committed Bytes In Use juga sudah naik ke kisaran sekitar ~55%, yang menandakan sistem mulai lebih aktif memanfaatkan page file untuk menjaga kestabilan saat beban memori meningkat. Setelah titik tersebut, tren kenaikan berlanjut hingga sekitar 21:30:00, di mana garis hijau terus merambat menuju kisaran ~70–78% dan garis ungu ikut naik bertahap ke kisaran ~12–15%, sehingga pemakaian page file tampak semakin intens pada fase beban tinggi. Pada rentang 21:31:00 hingga 21:34:00, garis hijau cenderung bertahan tinggi (~78–80%) dan garis ungu berfluktuasi di kisaran ~15–17%, yang menggambarkan pemakaian page file tetap tinggi dan dinamis mengikuti perubahan kebutuhan memori aplikasi yang sedang berjalan.



Grafik 4.3.2 Analisis Hubungan Page Fault dengan Ketersediaan RAM

Grafik 4.3.2 menampilkan hubungan antara Page Faults/sec (garis merah) dan Available RAM (MB) (area biru) untuk menonjolkan dampak menipisnya sisa RAM terhadap intensitas fault. Grafik ini menunjukkan bahwa pada fase awal sekitar 21:24:00–21:25:30, Available RAM

masih tinggi (mendekati ~1800–2000 MB), sehingga spike Page Faults/sec cenderung lebih jarang dan lebih rendah. Perubahan menjadi lebih jelas setelah 21:25:30 ketika area biru mulai turun tajam ke kisaran ~1000–1500 MB; pada fase ini spike Page Faults/sec menjadi lebih rapat dan lebih tinggi, menandakan fault makin sering terjadi ketika ruang RAM yang tersisa semakin sempit. Puncak yang ditandai pada grafik (sekitar 202.000 faults/sec) muncul sekitar 21:27:30–21:27:45 pada saat Available RAM berada pada level lebih rendah (sekitar ~800–1100 MB), sehingga fase ini menggambarkan tekanan memori yang berat dan berpotensi memicu penurunan responsivitas. Pola spike yang tetap sering hingga mendekati 21:34:00, disertai Available RAM yang bertahan pada kisaran rendah (~700–1200 MB), memperkuat indikasi bahwa semakin sedikit RAM yang tersedia, semakin besar peluang terjadinya fault dan aktivitas paging yang lebih intens.

4.4 Kaitan Temuan dengan Konsep Virtual Memory

Hasil pengamatan dari Task Manager, Resource Monitor, dan Performance Monitor memperlihatkan pola yang konsisten dengan konsep virtual memory berbasis paging, yaitu sistem operasi berusaha mempertahankan ilusi ruang memori yang lebih besar daripada RAM dengan memanfaatkan penyimpanan sekunder melalui page file/backing store saat RAM mulai menipis. Lonjakan beban saat 50 tab browser dibuka meningkatkan kebutuhan halaman memori aktif (working set) dari banyak proses secara bersamaan, sehingga jumlah frame RAM yang tersedia tidak lagi cukup untuk mempertahankan semua halaman aktif tetap berada di memori utama. Kondisi kekurangan frame ini memaksa sistem melakukan demand paging lebih intens, yaitu halaman hanya dipertahankan/diambil ke RAM ketika benar-benar dibutuhkan, dan ketika halaman yang dibutuhkan tidak ada di RAM maka terjadi page fault yang harus ditangani OS.

Kondisi idle menunjukkan page-fault rate dan aktivitas paging relatif rendah karena RAM masih longgar, sehingga sebagian besar referensi memori dapat dipenuhi tanpa mengambil halaman dari disk. Data PerfMon pada fase ringan memperlihatkan Available MBytes rata-rata sekitar 2043 MB dan Paging File % Usage sekitar 5–6%, yang menggambarkan pemakaian page file belum signifikan. Nilai Pages/sec dan Pages Input/sec pada fase ini juga cenderung kecil (rata-rata sekitar 138 pages/detik), yang mengindikasikan pertukaran halaman berbasis disk belum mendominasi aktivitas memori.

Peralihan ke beban puncak memperlihatkan mekanisme yang dijelaskan teori: ketika ruang RAM mengecil, frekuensi page fault naik dan paging berbasis disk meningkat karena OS harus terus memindahkan halaman untuk “mengosongkan” frame bagi halaman yang lebih dibutuhkan. Data PerfMon pada fase berat menunjukkan Available MBytes turun menjadi rata-rata sekitar 942 MB, sementara Pages/sec naik menjadi rata-rata sekitar 6403 pages/detik dan Pages Input/sec menjadi sekitar 4917 pages/detik, yang menggambarkan peningkatan aktivitas page-in/out dan pembacaan halaman dari disk. Kenaikan Paging File (Total)\% Usage ke sekitar 16–17% memperkuat indikasi bahwa page file dipakai lebih aktif sebagai perluasan memori ketika RAM tidak lagi cukup longgar.

Temuan Resource Monitor berupa meningkatnya Hard Faults/sec pada beban puncak juga sejalan dengan konsep page fault yang “mahal”, karena hard fault menunjukkan halaman yang

dibutuhkan tidak tersedia di RAM dan harus dipenuhi dengan I/O dari disk (misalnya page file atau file yang dipetakan), sehingga waktu tunggu meningkat. Kenaikan laju page fault dan paging pada fase beban tinggi menjelaskan degradasi responsivitas yang dirasakan, karena akses disk jauh lebih lambat dibanding akses RAM dan CPU dapat lebih sering menunggu operasi I/O selesai.

Polanya juga mengarah pada gejala yang dibahas dalam konsep thrashing, yaitu saat page-fault rate menjadi sangat tinggi sehingga sistem lebih banyak menghabiskan waktu untuk menangani page fault dan paging daripada mengeksekusi kerja aplikasi secara efektif. Grafik hasil visualisasi (Swap Usage dan Page Fault vs Available RAM) memperlihatkan bahwa ketika pemakaian page file mulai naik dan sisa RAM turun, spike page fault menjadi lebih rapat dan lebih tinggi, yang merupakan ciri eskalasi tekanan memori menuju perilaku thrashing jika beban terus ditambah atau dibiarkan lama. Dalam kerangka manajemen memori, kondisi ini dapat dijelaskan oleh konsep "kecukupan frame untuk working set": ketika working set total proses melampaui kapasitas frame yang tersedia, tingkat page fault naik tajam karena halaman yang baru dimuat cenderung segera tergantikan dan kemudian dibutuhkan kembali.

5. KESIMPULAN

Berdasarkan hasil investigasi dan analisis data yang telah dilakukan, dapat ditarik beberapa simpulan utama sebagai berikut:

1. Waktu Terjadinya Lonjakan Page Fault Fenomena *Page Fault* teridentifikasi meningkat secara drastis saat kapasitas memori fisik yang tersedia (*Available Memory*) menyusut tajam akibat beban kerja. Data Performance Monitor mencatat nilai maksimum *Page Faults* mencapai 100,524 faults/detik pada kondisi beban puncak, jauh di atas rata-rata kondisi *idle*. Kejadian ini berkorelasi langsung dengan penurunan *Available MBytes* dari rata-rata 2,043 MB menjadi 942 MB (dengan titik terendah 693 MB), serta kemunculan *Hard Faults* yang signifikan pada proses *Memory Compression* sebesar 223 faults/sec yang sebelumnya bernilai nol.
2. Perilaku Adaptif Swap Usage Mekanisme *Swap Usage* atau *Page File* menunjukkan perilaku eskalatif untuk mengkompensasi kekurangan memori fisik. Pemanfaatan *Page File* meningkat dari rata-rata 5.7% pada kondisi stabil menjadi 16.2% (maksimum 16.73%) saat 50 tab peramban dibuka. Hal ini diperkuat oleh data Task Manager yang menunjukkan lonjakan *Committed Memory* hingga estimasi 22.5 GB, melampaui kapasitas fisik 16 GB, yang menandakan sistem secara aktif memindahkan data ke penyimpanan sekunder untuk menjaga stabilitas.
3. Indikasi Gejala Thrashing Aktivitas pertukaran halaman yang berlebihan mengindikasikan gejala awal *thrashing* pada sistem. Indikator *Memory\Pages/sec* melonjak dari rata-rata 138 pages/detik pada fase *idle* menjadi 6,403 pages/detik pada fase beban berat, dengan puncak sesaat mencapai 103,658 pages/detik. Tingginya volume halaman yang dibaca atau ditulis ke disk ini menegaskan bahwa sistem menghabiskan sumber daya signifikan untuk operasi I/O memori, yang berdampak pada latensi akses.

4. Stabilitas Sistem di Bawah Tekanan Mekanisme manajemen memori terbukti mampu mempertahankan keberlangsungan operasional sistem meskipun penggunaan memori fisik (*In Use*) telah mencapai 15.2 GB atau mendekati batas total 16 GB. Sistem operasi berhasil mencegah kegagalan fatal dengan memanfaatkan ruang alamat virtual yang besar, terlihat dari kemampuan sistem menampung beban kerja meskipun *Available Memory* tersisa sangat sedikit di kisaran 600 - 900 MB.

DAFTAR PUSTAKA

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed. Hoboken, NJ: Wiley, 2018.

[2] D. Na et al., "A 1.8-Gb/s/pin 16-Tb NAND flash memory multi-chip package with F-chip for high-performance and high-capacity storage," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 4, pp. 1129–1138, Apr. 2021.

[3] P. Pandey, G. Joshi, and K. K. Gola, "A zone based improved disk scheduling algorithm," *International Journal of Computer Applications & Information Technology*, vol. 9, no. 1, pp. 133–138, 2016.

[4] Y. Wang *et al*., "Holistic and Opportunistic Scheduling of Background I/Os in Solid-State Drives," *IEEE Transactions on Computers*, vol. 72, 2023.

[5] G. Zhu, K. Lu, X. Wang, Y. Zhang, P. Zhang, and S. Mittal, "SwapX: An NVM-Based Hierarchical Swapping Framework," *IEEE Access*, vol. 5, pp. 16383–16392, 2017, doi: 10.1109/ACCESS.2017.2737634.

[6] A. Kokolis, D. Skarlatos, and J. Torrellas, "PageSeer: Using Page Walks to Trigger Page Swaps in Hybrid Memory Systems," in *Proc. 25th IEEE Int. Symp. High Performance Computer Architecture (HPCA)*, 2019, pp. 596–608.

BAGIAN B: INVESTIGASI MASS STORAGE

1. PENDAHULUAN

Mass storage merupakan komponen penyimpanan sekunder yang berperan menjaga data tetap tersimpan meskipun sistem dimatikan, sekaligus menjadi media utama untuk aktivitas I/O seperti membaca/menulis file, memuat aplikasi, dan akses data sistem. Performa mass storage sangat memengaruhi responsivitas komputer, terutama saat sistem menjalankan banyak aplikasi secara bersamaan atau ketika terjadi aktivitas I/O intensif (misalnya banyak file kecil, cache aplikasi, atau proses latar belakang).

Pada praktiknya, dua teknologi yang paling umum digunakan adalah Hard Disk Drive (HDD) dan Solid State Drive (SSD). HDD menggunakan mekanisme piringan berputar dan head baca/tulis, sedangkan SSD berbasis memori flash tanpa komponen mekanik, sehingga karakteristik latensi dan throughput keduanya berbeda dan dapat menghasilkan perbedaan performa yang signifikan pada akses sekuensial maupun akses acak (random).

Investigasi ini bertujuan membandingkan performa akses baca/tulis SSD dan HDD menggunakan uji sederhana berbasis CrystalDiskMark, lalu mendiskusikan hasilnya dalam konteks Disk Scheduling Algorithms (FCFS, SSTF, SCAN, dan C-SCAN). Hasil pengujian pada SSD dan HDD digunakan sebagai dasar analisis untuk menentukan algoritma penjadwalan yang lebih sesuai pada masing-masing media penyimpanan dan menjelaskan alasannya berdasarkan karakteristik perangkat.

2. DASAR TEORI

Konsep Dasar Mass Storage dan Penjadwalan Disk Mass storage merupakan media penyimpanan sekunder yang bersifat non-volatile. Dua teknologi utama yang mendominasi saat ini adalah Hard Disk Drive (HDD) dan Solid State Drive (SSD). HDD bekerja secara mekanis menggunakan piringan magnetik dan head baca/tulis, sehingga performanya sangat dipengaruhi oleh *seek time* dan *rotational latency*. Sebaliknya, SSD berbasis memori flash dan tidak memiliki komponen bergerak.

Dalam sistem operasi, pengelolaan antrean permintaan I/O (I/O request) sangat krusial untuk efisiensi. Silberschatz et al. menjelaskan bahwa algoritma penjadwalan disk dirancang untuk meminimalkan waktu pencarian (*seek time*) pada HDD. Algoritma dasar meliputi **FCFS** (*First-Come, First-Served*) yang sederhana namun bisa tidak efisien; **SSTF** (*Shortest Seek Time First*) yang memprioritaskan jarak terdekat namun berisiko *starvation*; serta **SCAN** (elevator) dan **C-SCAN** yang bekerja dengan menyapu piringan disk secara sistematis untuk memberikan layanan yang lebih seragam. Meskipun algoritma ini vital untuk HDD, relevansinya pada SSD lebih terfokus pada manajemen antrean logis karena absennya latensi mekanik [1].

Kompleksitas Internal SSD dan Operasi Latar Belakang Berbeda dengan HDD yang dapat menimpa data secara langsung, SSD memerlukan mekanisme internal yang kompleks karena karakteristik memori flash (NAND) yang harus dihapus (*erase*) sebelum ditulis ulang. Hal ini

memunculkan aktivitas latar belakang atau *background I/O*, seperti *garbage collection*. Wang et al. menyoroti bahwa penjadwalan operasi latar belakang ini harus dilakukan secara holistik dan oportunistik agar tidak mengganggu *throughput* dari permintaan I/O pengguna (*user requests*), yang menjadi faktor kunci dalam menjaga stabilitas performa SSD pada beban kerja intensif [2].

Peran Flash Translation Layer (FTL) Untuk menjembatani perbedaan antara antarmuka sistem file logis dan struktur fisik memori flash, SSD menggunakan lapisan perangkat lunak yang disebut *Flash Translation Layer* (FTL). Kinerja FTL sangat menentukan kecepatan akses data. Koo et al. mengusulkan bahwa integrasi struktur indeks ke dalam FTL dapat mempercepat pencarian data secara signifikan, yang secara langsung berdampak pada pengurangan latensi baca dan tulis pada perangkat SSD modern [3].

Strategi Pemetaan Data Selain pengindeksan, efisiensi FTL juga bergantung pada strategi pemetaan alamat logis ke fisik (*mapping*). Pemetaan yang tidak efisien dapat mempercepat keausan blok memori. Li et al. menganalisis bahwa penggunaan pemetaan berbasis pola frekuensi (*frequent pattern-based mapping*) pada FTL dapat meningkatkan kinerja dengan mengenali pola data yang sering diakses, sehingga operasi tulis dapat dikelola dengan lebih optimal dibandingkan metode pemetaan konvensional [4].

Evolusi Menuju Non-Volatile Memory (NVM) Perkembangan teknologi penyimpanan terus berlanjut melampaui batas SSD berbasis NAND standar. Munculnya *Non-Volatile Memory* (NVM) seperti Intel Optane menawarkan karakteristik yang menjembatani celah antara RAM dan penyimpanan sekunder. Oliveira et al. membuktikan melalui analisis eksperimental bahwa NVM mampu memberikan latensi yang jauh lebih rendah dan *throughput* yang lebih tinggi dibandingkan SSD konsumen biasa, menjadikannya solusi ideal untuk sistem yang membutuhkan responsivitas ekstrem dan memperluas kapasitas memori sistem secara efektif [5].

3. METODOLOGI

Eksperimen pada bagian ini dilakukan untuk membandingkan karakteristik performa *mass storage* antara SSD dan HDD melalui pengujian baca/tulis sederhana menggunakan CrystalDiskMark pada sistem operasi Windows. Fokus pengujian diarahkan pada dua pola akses utama, yaitu akses sekuensial (transfer data berurutan seperti menyalin file besar) dan akses acak/random (akses blok kecil yang menyebar seperti membuka banyak file kecil, memuat aplikasi, dan aktivitas sistem). Dengan pendekatan ini, hasil uji diharapkan mampu merepresentasikan perbedaan perilaku perangkat penyimpanan pada beban kerja nyata sekaligus menjadi dasar diskusi dalam konteks algoritma penjadwalan disk (*disk scheduling*).

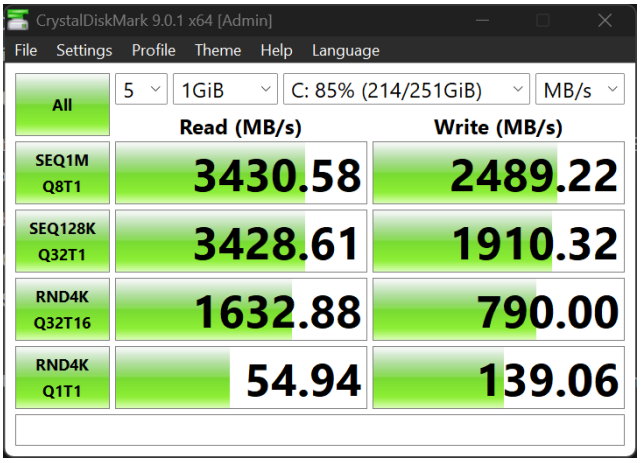
Instrumen pengujian yang digunakan adalah CrystalDiskMark (versi 9.0.1 x64) dengan mode pengujian “All” untuk mengeksekusi rangkaian uji sekuensial dan random secara otomatis pada perangkat yang dipilih. Konfigurasi uji diset dengan *test count* sebanyak 5 kali dan ukuran data uji 1 GiB, sehingga nilai yang diperoleh lebih stabil karena setiap metrik diulang beberapa kali sebelum ditampilkan sebagai hasil akhir. Parameter uji yang dianalisis meliputi SEQ1M Q8T1 dan SEQ128K Q32T1 sebagai representasi performa transfer blok besar berurutan, serta

RND4K Q32T16 dan RND4K Q1T1 sebagai representasi performa akses blok kecil acak pada kondisi antrean tinggi dan antrean rendah.

Prosedur eksperimen dilaksanakan secara bertahap dan konsisten pada kedua perangkat untuk menjaga keterbandingan data. Pertama, sistem dipastikan berada pada kondisi stabil (aplikasi latar belakang diminimalkan) lalu CrystalDiskMark dijalankan pada drive SSD sebagai objek uji awal dan hasilnya didokumentasikan. Tahap berikutnya, pengujian yang sama diulang pada drive HDD dengan konfigurasi identik, kemudian hasilnya didokumentasikan. Data yang terkumpul dari kedua pengujian selanjutnya dianalisis untuk menilai kesenjangan performa sekuensial vs random, menginterpretasikan implikasinya terhadap "waktu akses" pada beban kerja umum, serta mengaitkannya dengan teori penjadwalan disk (FCFS, SSTF, SCAN, dan C-SCAN) yang bertujuan mengatur urutan layanan permintaan I/O.

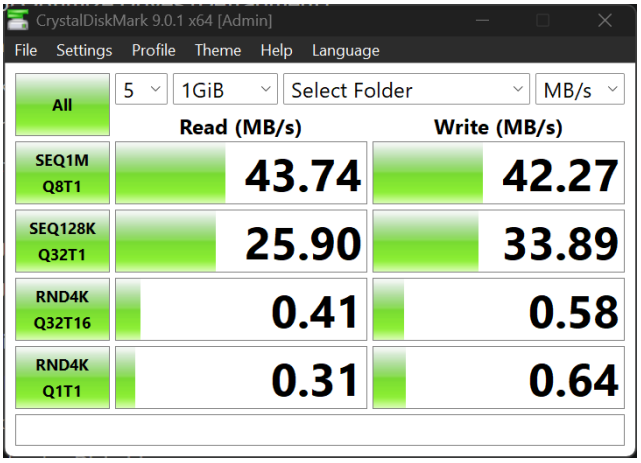
4. HASIL DAN ANALISIS

Pengujian performa mass storage dilakukan menggunakan CrystalDiskMark (ukuran uji 1 GiB) untuk membandingkan karakteristik baca/tulis pada SSD dan HDD pada pola akses sekuensial dan random. Parameter yang diamati meliputi SEQ1M dan SEQ128K (representasi transfer data berurutan), serta RND4K (representasi akses data kecil yang menyebar/acak seperti banyak file kecil, OS, dan aktivitas paging).



Gambar 4.1 Hasil uji CrystalDiskMark pada SSD

Pada Gambar 4.1, SSD menunjukkan performa sekuensial yang sangat tinggi, yaitu SEQ1M Q8T1 sebesar 3430.58 MB/s (read) dan 2489.22 MB/s (write), serta SEQ128K Q32T1 sebesar 3428.61 MB/s (read) dan 1910.32 MB/s (write). Pada pengujian random, SSD juga tetap kuat khususnya saat antrean permintaan tinggi (RND4K Q32T16 read 1632.88 MB/s dan write 790.00 MB/s), sedangkan pada kondisi antrean rendah (RND4K Q1T1) performa turun menjadi 54.94 MB/s (read) dan 139.06 MB/s (write). Pola ini mengindikasikan bahwa SSD mampu memanfaatkan *queue depth* dan paralelisme internal ketika banyak permintaan I/O datang bersamaan, sehingga throughput random meningkat drastis pada Q32T16 dibanding Q1T1.



Gambar 4.2 Hasil uji CrystalDiskMark pada HDD

Pada Gambar 4.2, HDD memperlihatkan performa sekuensial yang jauh lebih rendah dibanding SSD, yaitu SEQ1M Q8T1 sebesar 43.74 MB/s (read) dan 42.27 MB/s (write), serta SEQ128K Q32T1 sebesar 25.90 MB/s (read) dan 33.89 MB/s (write). Kesenjangan paling terlihat muncul pada akses random 4K, di mana HDD hanya mencapai 0.41 MB/s (read) dan 0.58 MB/s (write) pada RND4K Q32T16, serta 0.31 MB/s (read) dan 0.64 MB/s (write) pada RND4K Q1T1. Nilai random yang sangat kecil ini konsisten dengan karakteristik HDD yang dibatasi oleh mekanisme fisik (pergerakan head dan rotasi piringan), sehingga akses acak membuat latensi menjadi dominan dan throughput jatuh drastis.

Jika dilihat sebagai "waktu akses" untuk beban kerja yang melibatkan banyak data kecil acak, hasil ini menjelaskan mengapa sistem berbasis HDD lebih mudah terasa lambat saat banyak aplikasi berjalan bersamaan: request I/O kecil dan menyebar akan memaksa HDD melakukan banyak perpindahan head, sedangkan SSD tidak mengalami biaya mekanis tersebut. Oleh karena itu, perbedaan SSD vs HDD tidak hanya pada "kecepatan salin file besar" (sekuensial), namun terutama pada responsivitas untuk beban kerja harian yang didominasi I/O kecil/acak (random).

Dalam konteks Disk Scheduling Algorithms, FCFS melayani request sesuai urutan kedatangan sehingga sederhana, tetapi pada HDD dapat menyebabkan head bergerak bolak-balik secara tidak efisien dan memperbesar total seek time. SSTF memilih request yang paling dekat dari posisi head saat ini (mengurangi seek rata-rata), namun dapat menyebabkan starvation karena request yang jauh bisa tertunda sangat lama jika request dekat terus berdatangan. SCAN (elevator) melayani request sambil bergerak satu arah lalu berbalik, sedangkan C-SCAN bergerak satu arah dan "lompat" kembali ke awal untuk melayani lagi dari arah yang sama, yang dirancang untuk memberikan waktu tunggu yang lebih uniform.

Berdasarkan teori tersebut, algoritma yang paling cocok untuk HDD adalah SCAN atau C-SCAN karena keduanya secara langsung menargetkan sumber biaya terbesar HDD, yaitu meminimalkan pergerakan head/seek yang mahal pada akses acak (yang tampak pada hasil random 4K HDD yang sangat rendah). C-SCAN sering dipilih ketika diinginkan pemerataan waktu tunggu (fairness) karena pola layanannya satu arah dan lebih uniform dibanding SCAN. Sebaliknya, untuk SSD, manfaat utama algoritma yang "mengoptimalkan jarak head"

(SSTF/SCAN/C-SCAN) menjadi jauh lebih kecil karena SSD tidak memiliki seek time mekanis; dari pilihan FCFS, SSTF, SCAN, dan C-SCAN, pendekatan sederhana seperti FCFS umumnya sudah memadai, sementara peningkatan performa SSD lebih banyak ditentukan oleh paralelisme internal dan manajemen antrean I/O.

5. KESIMPULAN

Berdasarkan hasil investigasi dan analisis data yang telah dilakukan, dapat ditarik beberapa simpulan utama sebagai berikut:

1. Kesenjangan performa SSD vs HDD sangat besar (terutama random I/O). SSD mencatat performa sekuensial sangat tinggi (mis. SEQ1M read 3430.58 MB/s dan write 2489.22 MB/s), sedangkan HDD hanya sekitar SEQ1M read 43.74 MB/s dan write 42.27 MB/s. Pada akses random 4K, gap makin ekstrem: SSD RND4K Q1T1 read 54.94 MB/s vs HDD 0.31 MB/s, sehingga akses file kecil/acak pada HDD jauh lebih lambat dibanding SSD.
2. Perbedaan ini konsisten dengan karakteristik fisik media penyimpanan. HDD sangat dipengaruhi *seek time* dan *rotational latency* karena mekanisme head dan piringan berputar, sehingga performa random turun drastis ketika permintaan tersebar di banyak lokasi. SSD tidak memiliki komponen mekanik, sehingga tidak mengalami biaya seek fisik seperti HDD dan lebih mampu mempertahankan respons pada akses acak, terutama saat antrian/parallelism tinggi.
3. Algoritma disk scheduling paling cocok untuk HDD adalah SCAN atau C-SCAN. FCFS berisiko membuat pergerakan head HDD tidak efisien karena melayani sesuai urutan kedatangan tanpa mempertimbangkan posisi head. SSTF dapat menurunkan rata-rata jarak perpindahan head, tetapi memiliki risiko *starvation* karena request yang jauh bisa tertunda terus-menerus. SCAN/C-SCAN lebih sesuai untuk HDD karena melayani request secara terstruktur seperti “elevator”, menekan biaya perpindahan head, dan C-SCAN cenderung memberikan waktu tunggu yang lebih seragam (*uniform waiting time*).
4. Algoritma yang paling cocok untuk SSD (dari FCFS, SSTF, SCAN, C-SCAN) umumnya FCFS/sederhana. Karena SSD tidak memiliki seek time mekanis, keuntungan utama algoritma yang mengoptimalkan pergerakan head (SSTF/SCAN/C-SCAN) menjadi jauh lebih kecil dibanding pada HDD. Pada SSD modern, performa lebih ditentukan oleh manajemen antrean dan paralelisme internal serta pengaturan *background I/O* (misalnya pemeliharaan

internal flash), sehingga fokus scheduling bergeser dari “jarak head” menjadi koordinasi I/O dan latensi.

DAFTAR PUSTAKA

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed. Hoboken, NJ: Wiley, 2018.

[2] [1] Y. Wang, Y. Zhou, Y. Jiang, Y. Cui, and G. Zhang, "Holistic and opportunistic scheduling of background I/Os in flash-based SSDs," *IEEE Transactions on Computers*, vol. 72, no. 11, pp. 3127–3139, Nov. 2023, doi: 10.1109/TC.2023.3288748.

[3] [2] G. Koo *et al.*, "Embedding index into flash translation layer in SSDs," *IEEE Transactions on Computers*, vol. 72, no. 1, Jan. 2023.

[4] [3] J. Li *et al.*, "Frequent pattern-based mapping at flash translation layer," *IEEE Access*, vol. 7, pp. 95233–9524x, 2019.

[5] [4] G. F. Oliveira *et al.*, "Extending Memory Capacity in Modern Consumer Systems With Emerging Non-Volatile Memory: Experimental Analysis and Characterization Using the Intel Optane SSD," *IEEE Access*, vol. 11, pp. 105843–1058xx, 2023.