pISSN: 2442-3386 eISSN: 2442-4293
Melek IT
Vol 11 No 1 Jan – Jun 2024, 87 - 100
INFORMATION TECHNOLOGY JOURNAL

SYSTEMATIC LITERARTUR REVIEW DENGAN METODE PRISMA : MANAJEMEN MEMORI EFISIEN PADA SISTEM OPERASI LINUX

Rexy Putra Nur Laksana*[1], Herbert Siregar[2]
[1]Program Studi Ilmu Komputer, Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam, Universitas Pendidikan Indonesia, rexy.putra005@upi.edu
[2]Program Studi Ilmu Komputer, Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam, Universitas Pendidikan Indonesia, herbert@upi.edu
*)Korespondensi: rexy.putra005@upi.edu

Abstrak

Pengelolaan memori yang optimal menjadi aspek penting dalam menjaga performa sistem operasi Linux, terutama dalam menghadapi tantangan arsitektur komputasi *modern*. Studi ini menerapkan pendekatan PRISMA untuk melakukan tinjauan literatur sistematis yang mengevaluasi strategi-strategi efisiensi manajemen memori dalam *kernel* Linux. Dari hasil seleksi ketat terhadap literatur terkini, ditemukan empat pendekatan utama yang relevan: perluasan memori terdistribusi (COMEX), kontrol alokasi berbasis NUMA, manajemen memori heterogen CPU-GPU (HMM), serta pengujian performa melalui *benchmark* XSBench. Keempat teknik tersebut menunjukkan efektivitas dalam mengurangi latensi, meningkatkan pemanfaatan sumber daya, dan menyederhanakan pengelolaan memori pada sistem berskala besar. Temuan ini memberikan kontribusi penting bagi pengembangan kebijakan memori adaptif di masa mendatang dan menunjukkan pentingnya kolaborasi antara desain perangkat keras, *kernel*, dan aplikasi.
Kata Kunci: Manajemen memori, Kernel Linux, PRISMA, NUMA, HMM

Abstract

Optimal memory management is a critical aspect in maintaining the performance of Linux operating systems, especially when facing the challenges of modern computing architectures. This study applies the PRISMA approach to conduct a systematic literature review evaluating efficiency strategies in Linux kernel memory management. Through a rigorous selection of recent literature, four main relevant approaches were identified: distributed memory expansion (COMEX), NUMA-based allocation control, heterogeneous CPU-GPU memory management (HMM), and performance testing using the XSBench benchmark. These four techniques demonstrate effectiveness in reducing latency, improving resource utilization, and simplifying memory handling in large-scale systems. The findings make a significant contribution to the development of future adaptive memory policies and highlight the importance of collaboration between hardware design, the kernel, and applications.
Keywords: *Memory management, Linux kernel, PRISMA, NUMA, HMM*

I. PENDAHULUAN

Manajemen memori yang efisien merupakan aspek krusial dalam sistem operasi, terutama pada Linux yang banyak digunakan dalam berbagai aplikasi komputasi berskala besar dan heterogen [1]. Efisiensi dalam pengelolaan memori sangat penting untuk menjaga kinerja sistem, mencegah latensi berlebih, dan memastikan penggunaan sumber daya secara optimal. Metode *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* (PRISMA) digunakan dalam studi ini karena dirancang untuk meningkatkan transparansi dan kualitas pelaporan tinjauan sistematis dan meta-analisis [2]. PRISMA 2020, yang merupakan pembaruan dari PRISMA 2009, memperkenalkan sejumlah modifikasi penting guna meningkatkan kejelasan dan efektivitas pelaporan [3]. Studi ini bertujuan untuk mengidentifikasi, mengevaluasi, dan mensintesis pendekatan manajemen memori yang efisien dalam sistem operasi Linux melalui tinjauan literatur sistematis.

Dalam tinjauan literatur ini, fokus utama adalah pada berbagai pendekatan dan teknik yang digunakan untuk mengelola memori di sistem operasi Linux. Beberapa penelitian telah mengusulkan metode baru untuk meningkatkan efisiensi manajemen memori, seperti *Improved Contiguous Memory Allocation* (ICMA) yang bertujuan untuk mengoptimalkan penggunaan memori, mengurangi kesalahan alokasi, dan mengurangi *overhead* sistem [4]. Penelitian terbaru juga memperkenalkan pendekatan PET dan Memtis yang mampu meningkatkan efisiensi pengelolaan memori secara signifikan pada sistem bertingkat, melalui metode prediktif dan klasifikasi halaman yang dinamis [5][6]. Selain itu, kebijakan perangkat lunak untuk manajemen memori heterogen juga telah dikembangkan untuk meningkatkan keseimbangan antara kinerja dan konsumsi daya [7].

Penelitian lain menunjukkan bahwa algoritma manajemen memori yang saat ini digunakan di Linux, dan diadopsi oleh Android, memiliki beberapa kelemahan, seperti tingginya rasio *page re-fault* yang

menunjukkan bahwa banyak halaman yang dihapus oleh algoritma tersebut diakses kembali dalam waktu dekat. Untuk mengatasi masalah ini, beberapa ide telah diusulkan untuk meningkatkan kinerja manajemen memori pada perangkat Android [8].

Dengan menggunakan metode PRISMA, tinjauan literatur ini akan mengidentifikasi dan mengevaluasi berbagai pendekatan yang telah diusulkan untuk meningkatkan efisiensi manajemen memori di sistem operasi Linux. Hasil dari tinjauan ini diharapkan dapat memberikan wawasan yang berharga bagi pengembangan strategi manajemen memori yang lebih efisien dan efektif di masa depan.

II. KERANGKA TEORI / KAJIAN

2.1. Studi Pustaka

Linux adalah sistem operasi yang sangat portabel dan dapat dioperasikan di berbagai perangkat mulai dari komputer genggam, komputer pribadi, hingga komputer berukuran sedang [9]. Linux memiliki karakteristik *open source*, stabilitas, dan efisiensi yang membuatnya menjadi platform yang baik untuk penelitian di bidang ilmu komputer [10]. Selain itu, Linux juga dikenal karena keamanannya yang tinggi, meskipun ada beberapa tantangan terkait keamanan yang terus menjadi perhatian [11]. Penggunaan Linux dalam pendidikan juga semakin meningkat, dengan berbagai program pendidikan yang dirancang untuk meningkatkan kemampuan profesional di bidang Linux [12].

Sistem manajemen memori Linux menggunakan teknik memori virtual yang sudah matang, namun kompleksitasnya membuat *kernel* rentan terhadap bug dan kinerja yang tidak terduga [13]. Berdasarkan analisis patch *kernel* (versi 2.6.32–4.0), [13] menemukan bahwa sebagian besar perubahan pada modul memori Linux terpusat pada fungsi-fungsi penting seperti mengalokasi memori, penanganan kesalahan halaman, dan pengelolaan sumber daya memori [13]. Studi yang sama juga mencatat bahwa sekitar 80% *patch* diarahkan hanya pada 25% kode sumber inti, menandakan fokus pengembangan yang sangat terkonsentrasi pada area-area tertentu sistem memori virtual Linux [13]. Dengan demikian, meski sistem memori Linux telah banyak berkembang, kompleksitas strukturalnya tetap berkontribusi pada kerentanan bug dan perilaku kinerja yang sulit diprediksi [13].

Banyak penelitian menyoroti pentingnya efisiensi penggunaan sumber daya dalam desain sistem operasi. [14] menemukan bahwa sistem komputasi data intensif modern lebih memprioritaskan skalabilitas daripada efisiensi, orientasi ini mengakibatkan laju pemrosesan aktual per-*node* yang jauh lebih rendah dibanding potensi teoritisnya. [15] menambahkan bahwa meskipun perangkat lunak paralel dapat memaksimalkan pemanfaatan kemampuan hardware paralel, hal ini sering kali menyebabkan pemborosan sumber daya karena eksekusi pekerjaan yang redundant. [16] secara praktis menunjukkan bahwa efisiensi OS dapat diukur melalui serangkaian tes kinerja sederhana; misalnya, durasi komunikasi antartugas, penjadwalan tugas, dan penanganan interupsi diukur untuk menghasilkan tolok ukur kuantitatif efisiensi berbagai sistem operasi.

Tinjauan literatur sistematis *Systematic Literature Review* (SLR) didefinisikan sebagai metodologi penelitian yang secara sistematis mengumpulkan, mengidentifikasi, dan menganalisis secara kritis studi-studi penelitian yang relevan mengenai suatu topik [17]. [18] menekankan bahwa SLR harus dilakukan dengan protokol tinjauan yang terperinci, termasuk perumusan pertanyaan penelitian, strategi pencarian komprehensif, serta kriteria inklusi dan eksklusi eksplisit. Pendekatan ini menjamin tinjauan yang komprehensif dan dapat direproduksi, strategi pencarian yang terdokumentasi lengkap memungkinkan peneliti memasukkan semua studi relevan dan meminimalkan bias (misalnya dengan melaporkan temuan yang mendukung maupun yang menentang hipotesis). Lebih lanjut, SLR bertujuan merangkum bukti empiris yang ada dan mengidentifikasi kesenjangan penelitian dalam literatur, sehingga hasilnya dapat menjadi landasan bagi penelitian lebih lanjut.

Sejumlah penelitian terkini menggarisbawahi kompleksitas pengelolaan memori di *kernel* Linux akibat munculnya konfigurasi memori baru (misalnya memori bertingkat dan heterogen). [19] misalnya memperkenalkan konsep *File-Based Memory Management* (FBMM) di mana sistem manajemen memori ditulis sebagai filesystem melalui lapisan VFS untuk mendukung ekstensibilitas. Pendekatan ini memungkinkan banyak ekstensi memori seperti dukungan tiering diimplementasikan tanpa mengubah kode inti *kernel*. Selain itu, pengalokasian memori kontigu dinamis terus dikaji. [20] mencatat bahwa Linux menggunakan *Contiguous Memory Allocator* (CMA) untuk penyediaan memori kontinu secara dinamis, namun mekanisme ini masih memiliki kekurangan (karena tidak menjamin ketersediaan memori kontinu di masa depan).

Kendala *overhead* penanganan *page fault* juga menjadi fokus penelitian mutakhir. [21] melaporkan bahwa beban akibat *page fault minor* (*lazy allocation*) dapat mencapai ≈29% dari waktu eksekusi aplikasi, sehingga mereka mengusulkan desain ko-kerja *hardware* / perangkat lunak untuk menguranginya. Sebagai contoh, mereka memperkenalkan *Minor Fault Offload Engine* (MFOE), sebuah akselerator hardware per-inti CPU dengan tabel *frame* halaman pre-alokasi yang melayani *page fault* secara paralel, penerapan MFOE pada prototipe Linux menunjukkan peningkatan latensi penanganan *page fault* rata-rata hingga 33× lebih cepat dibandingkan mekanisme konvensional. Selain itu, sistem seperti EMD dan eBPF-mm memungkinkan optimalisasi dalam pengelolaan *page*, baik yang bersifat transparan maupun berbasis *userspace* [22][23].

pISSN: 2442-3386 eISSN: 2442-4293
Vol 11 No 1 Jan – Jun 2024, 89 - 100
Melek IT
INFORMATION TECHNOLOGY JOURNAL

Dalam arsitektur berbasis prosesor heterogen, pengoptimalan penjadwalan CPU dan penempatan data terutama dalam sistem NUMA sangat krusial untuk meningkatkan kinerja. [24] memperkenalkan PMCSched, sebuah kerangka kerja modul *kernel* Linux yang memungkinkan penjadwalan khusus pada sistem *asymmetric multicore* seperti Intel Alder Lake menggunakan *Performance Monitoring Counters* (PMC), tanpa perlu mengubah kode *kernel* inti. Dengan memanfaatkan informasi runtime dari PMC dan teknologi seperti Intel Thread Director, PMCSched mampu mengimplementasikan kebijakan penempatan *thread* dan pemanfaatan *cache* secara adaptif. Sementara itu, Toast dan HyperAlloc mengeksplorasi pengelolaan memori virtual pada sistem berbasis kontainer dan hypervisor dengan fokus pada overhead yang minimal serta kemampuan *elastic scaling* [25][26]. Evaluasi eksperimen menunjukkan bahwa PMCSched dapat meningkatkan hasil penjadwalan *multithreaded* dengan signifikan, mendekati potensi optimasi *scheduling* yang sebelumnya hanya tersedia melalui *patch kernel* langsung.

Tidak kalah penting, di domain *memory tiering*, [27] memperkenalkan *Transparent Memory Tiering System* (TMTS) pada Linux v5.18, sebuah sistem penempatan halaman otomatis yang menggabungkan DRAM dan memori lambat misalnya CXL-enabled. Penelitian terbaru seperti Memstrata dan M5 menunjukkan bahwa pendekatan berbasis CXL mampu meningkatkan efisiensi serta isolasi data antar tenant secara signifikan di lingkungan *virtualisasi* [28][29]. TMTS mengalokasikan sekitar 25% memori ke tier lambat dan menggunakan heuristik berbasis metrik performa untuk mempromosikan halaman panas secara dinamis ke DRAM, sambil menjaga penurunan performa di bawah 5%—sebuah terobosan efisiensi yang valid untuk sistem skala besar.

2.2. Systematic Literature Review

Systematic Literature Review (SLR) adalah metode yang digunakan untuk menganalisis literatur yang ada secara sistematis guna menjawab pertanyaan penelitian tertentu. Dalam bidang rekayasa perangkat lunak, SLR telah mencapai tingkat adopsi yang signifikan [30]. Proses SLR melibatkan beberapa tahap yang terdefinisi dengan baik, termasuk perencanaan, pelaksanaan, dan dokumentasi ulasan [31]. Tantangan utama dalam melakukan SLR meliputi strategi pencarian, basis data online, perencanaan, dan ekstraksi data [32]. Penelitian terkini juga menyoroti pentingnya *profiling* dan *benchmarking* alokasi memori, seperti yang dilakukan dalam MemPerf, untuk menganalisis *overhead* yang dihasilkan oleh alokator memori modern [33].

SLR dalam rekayasa perangkat lunak sering kali memerlukan adaptasi protokol untuk mengakomodasi karakteristik spesifik domain [34]. Selain itu, kualitas abstrak yang buruk dan kurangnya terminologi standar dalam makalah rekayasa perangkat lunak sering kali menjadi hambatan [32]. Untuk mengatasi tantangan ini, beberapa metode baru untuk mengevaluasi dan memvalidasi SLR telah diusulkan, termasuk pengembangan daftar pertanyaan dan kriteria evaluasi [35].

Dalam penelitian ini, SLR digunakan untuk menyelidiki dan menetapkan atribut yang mendefinisikan fitur kualitas perangkat lunak Sistem Manajemen Pengetahuan (KMS), metode yang digunakan untuk mengukur kualitas tersebut, dan panduan penting untuk menilai kualitas perangkat lunak KMS [31]. Pengalaman dari peneliti yang telah menggunakan metodologi SLR dapat sangat bermanfaat bagi peneliti pemula dalam mengatasi tantangan yang dihadapi selama proses SLR [32].

2.3. Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA)

Bagian dari *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* (PRISMA) mengenai manajemen memori efisien pada sistem operasi Linux dapat dijelaskan sebagai berikut, Manajemen memori yang efisien pada sistem operasi Linux melibatkan berbagai pendekatan untuk mengoptimalkan alokasi dan penggunaan memori. Salah satu metode yang diusulkan adalah pendekatan kolaboratif di mana aplikasi memberikan panduan kepada sistem operasi mengenai alokasi dan daur ulang memori fisik.

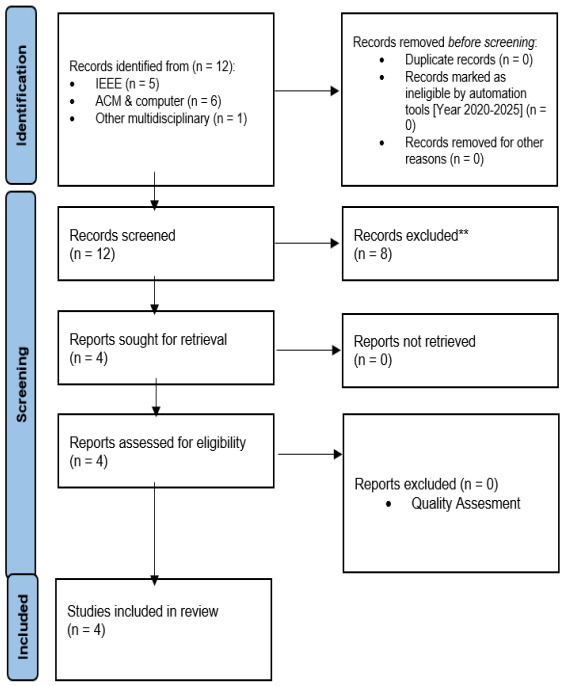
Pendekatan ini memungkinkan aplikasi untuk menginformasikan sistem operasi tentang tingkat akses relatif dan pola penggunaan ruang alamatnya, sehingga sistem operasi dapat menjadwalkan memori lebih efektif dan efisien [36]. Selain itu, penggunaan kurva rasio *miss* halaman (MRC) untuk melacak permintaan memori dinamis aplikasi secara real-time juga diusulkan. Metode ini dapat meningkatkan alokasi memori dan manajemen energi memori, yang terbukti mempercepat waktu eksekusi aplikasi hingga 5,86 kali dan mengurangi jumlah *page fault* hingga 63,1% [37]. Implementasi skema *prefetch* adaptif juga dapat meningkatkan kinerja sistem dengan memprediksi status memori secara lebih akurat dan dinamis, yang menunjukkan peningkatan kinerja rata-rata sebesar 10% [38]. Pendekatan-pendekatan ini menunjukkan bahwa kolaborasi antara perangkat keras, perangkat lunak pengawas, dan tugas aplikasi dapat menghasilkan manajemen memori yang lebih efisien pada sistem operasi Linux.

III. METODOLOGI PENELITIAN

Dengan mengikuti panduan PRISMA, penelitian ini melakukan *Systematic Literature Review* (SLR) untuk mendalami topik manajemen memori efisien pada Linux. Tahap awal melibatkan perancangan strategi pencarian dan penetapan kriteria eksklusi. Selanjutnya, dilakukan seleksi literatur secara cermat melalui identifikasi, penyaringan, dan penilaian kualitas publikasi. Data yang relevan diekstraksi dari studi-studi

yang lolos tahap akhir. Dalam pelaksanaannya, seluruh proses ini memanfaatkan alat bantu Parsifal untuk mempermudah proses perancangan, pemilihan artikel, dan evaluasi literatur.

Penelitian ini bertujuan untuk mengidentifikasi dan menghimpun literatur ilmiah yang relevan dengan manajemen memori yang efisien pada sistem operasi Linux. Fokus utama adalah memilih studi-studi yang tidak hanya sesuai topik, tetapi juga telah dievaluasi kelayakannya serta diperiksa mutu ilmiahnya. Seluruh tahapan dalam proses seleksi ini, mulai dari pencarian hingga pemilihan akhir, divisualisasikan melalui diagram alur PRISMA pada Gambar 1. Diagram ini memberikan gambaran yang jelas mengenai jumlah artikel yang ditemukan, disaring pada setiap tahap, dan akhirnya disertakan dalam tinjauan sistematis ini.



Gambar 1. *PRISMA flow diagram for the selection process*

Diagram alur PRISMA pada Gambar 1 memberikan gambaran visual yang ringkas mengenai seluruh tahapan seleksi literatur dalam kajian ini, mulai dari tahap identifikasi hingga proses inklusi akhir. Informasi lengkap mengenai jumlah publikasi pada masing-masing tahap seleksi serta alasan pengecualian akan dijelaskan secara terperinci pada Bab V, bagian Hasil dan Pembahasan.

IV. HASIL DAN PEMBAHASAN

4.1. Strategi Pencarian

Strategi pencarian dalam studi ini mencakup pemilihan basis data dan perumusan *base string* yang terdiri dari kata kunci relevan terkait manajemen memori efisien pada sistem operasi Linux. Kata kunci yang digunakan antara lain mencakup "*efficient memory management*," "*memory optimization*," "*Linux kernel*," dan "*memory allocation*," yang dikombinasikan dengan operator Boolean dan simbol *truncation* (misalnya tanda bintang) untuk mencakup berbagai variasi bentuk kata. Contoh *base string* yang digunakan adalah

("efficient memory management" OR "memory optimization") AND ("Linux kernel" OR "memory allocation")**.

Penggunaan *truncation* memungkinkan pencarian yang lebih luas, misalnya mencakup istilah seperti *management strategies*, *allocation policies*, hingga *kernel-level memory handling*, sehingga meningkatkan peluang menjaring publikasi yang relevan. Tabel 1 menampilkan basis data yang dipilih karena reputasinya dalam menyediakan literatur ilmiah di bidang ilmu komputer dan sistem operasi, seperti ACM Digital *Library* dan IEEE *Xplore* yang kaya akan publikasi jurnal dan prosiding konferensi. *Dimensions* digunakan untuk menjangkau studi multidisipliner, sementara *arXiv* dipertimbangkan untuk *pre-print* terkait sistem operasi dan manajemen memori. *Springer* menyediakan akses ke jurnal dan buku ilmiah yang ditinjau sejawat, dan sumber eksternal seperti tesis doktoral serta paten juga dijadikan referensi awal untuk memperluas cakupan tinjauan. Meskipun *pre-print* dari *arXiv* yang tidak ditinjau sejawat akhirnya dikeluarkan pada tahap penilaian kualitas, keberadaannya tetap dipertimbangkan untuk mencegah luputnya studi yang potensial. Basis data lain seperti *Web of Science* dan Scopus tidak disertakan karena fokusnya kurang spesifik terhadap topik sistem operasi dan efisiensi memori. Pemilihan basis data yang terbatas namun relevan ini bertujuan untuk menjaga keseimbangan antara kelengkapan dan efisiensi pencarian literatur.

4.2. Kriteria Pengecualian

Kriteria eksklusi ditetapkan untuk menyaring literatur yang tidak memenuhi syarat khusus dalam tinjauan ini. Tabel 2 merangkum kriteria eksklusi yang digunakan untuk menentukan kelayakan publikasi dalam proses seleksi literatur. Kolom E# pada tabel menunjukkan penanda unik untuk setiap kriteria eksklusi, yang diberi label dari E1 hingga E5. Sementara itu, kolom "*Statement*" memberikan penjelasan rinci tentang masing-masing kriteria beserta referensi pendukungnya.

Dalam konteks studi bertema *Manajemen Memori Efisien di Sistem Operasi Linux*, kriteria E1 hingga E3 diterapkan pada tahap identifikasi untuk mengecualikan publikasi yang tidak sesuai, seperti artikel yang tidak berbahasa Inggris (E1), diterbitkan sebelum tahun 2018 (E2), atau hanya berisi judul dari *prosiding*, *simposium*, atau kumpulan acara lainnya alih-alih penelitian individual yang ditinjau sejawat (E3). Hal ini penting untuk memastikan hanya literatur terkini dan relevan yang diikutsertakan, terutama sejak perkembangan signifikan terkait efisiensi manajemen memori dalam Linux terjadi dalam lima tahun terakhir.

pISSN: 2442-3386 eISSN: 2442-4293
Vol 11 No 1 Jan – Jun 2024, 91 - 100
Melek IT
INFORMATION TECHNOLOGY JOURNAL

Tabel 1. Kriteria pengecualian untuk proses seleksi

	Kriteria	Penjelasan
E1	Artikel dengan judul yang tidak menggunakan bahasa Inggris	Kriteria ini mengacu pada referensi yang relevan dan ditetapkan untuk memastikan bahwa hanya literatur berbahasa Inggris yang disertakan agar dapat dipahami secara luas dan mendukung konsistensi kajian literatur.
E2	Artikel yang diterbitkan sebelum tahun 2018	Perkembangan signifikan terkait efisiensi manajemen memori dalam *kernel* Linux terjadi dalam beberapa tahun terakhir. Oleh karena itu, pembatasan tahun publikasi sejak 2018 bertujuan untuk menjaring literatur yang relevan dan terkini.
E3	Judul artikel hanya mencerminkan event/kegiatan seperti prosiding, simposium, atau buku kolektif	Kriteria ini ditujukan untuk menghindari indeksasi judul dari acara atau kumpulan tanpa kontribusi penelitian individual. Hanya studi peer-*reviewed* yang relevan secara substansi terhadap manajemen memori di sistem operasi Linux yang disertakan.
E4	Judul maupun abstrak tidak menyebutkan minimal satu kata kunci utama	Artikel yang tidak mencantumkan istilah seperti "*efficient memory management*", "*Linux kernel*", atau "*memory allocation*" dalam judul atau abstrak dikecualikan, karena tidak relevan langsung dengan fokus kajian.
E5	Teks penuh dari artikel tidak dapat diakses	Kriteria ini ditetapkan agar hanya artikel yang dapat diakses secara penuh yang digunakan dalam tinjauan, guna memungkinkan proses analisis yang mendalam dan komprehensif terhadap isi literatur.

Selanjutnya, kriteria E4 dan E5 diterapkan pada tahap penyaringan. E4 mengecualikan artikel yang tidak menyebutkan topik inti seperti "*efficient memory management*," "*memory allocation*," atau "*Linux kernel*" dalam judul atau abstraknya, sedangkan E5 menyingkirkan artikel yang tidak memiliki akses ke teks penuh. Dengan pendekatan ini, tinjauan sistematis dapat fokus pada publikasi yang benar-benar berkaitan dengan tujuan penelitian dan menjamin kualitas serta signifikansi hasil kajian, seperti pada Tabel 1.

4.3. Identifikasi

Tahap awal dalam proses seleksi studi ini adalah Identifikasi sesuai pada Gambar 1. Tujuan utama dari tahap ini adalah untuk mengumpulkan sebanyak mungkin publikasi awal yang berpotensi relevan dengan topik manajemen memori efisien pada sistem operasi Linux. Pencarian dilakukan secara sistematis pada basis data ilmiah utama, yaitu IEEE, ACM & *computer*, serta sumber-sumber multidisipliner lainnya. Dari upaya pencarian ini, berhasil dihimpun sebanyak 12 publikasi awal, dengan sebaran 5 publikasi berasal dari IEEE, 6 publikasi dari ACM & *computer*, dan 1 publikasi dari sumber lain.

Sebelum melanjutkan ke penyaringan yang lebih mendalam, ke-12 publikasi ini menjalani pemeriksaan pra-penyaringan. Pemeriksaan ini bertujuan untuk mengeliminasi catatan duplikat dan catatan yang jelas-jelas tidak memenuhi syarat berdasarkan kriteria otomatis, seperti batasan tahun publikasi (2020-2025). Hasilnya menunjukkan bahwa tidak ada catatan duplikat yang ditemukan (n = 0), tidak ada yang dieksklusi oleh alat otomatis (n = 0), dan tidak ada pula yang dihapus karena alasan-alasan lainnya (n = 0). Oleh karena itu, semua 12 studi yang teridentifikasi ini lolos dari pra-penyaringan dan siap untuk dinilai lebih lanjut pada tahap penyaringan (*screening*).

4.4. Penyaringan

Memasuki tahap Penyaringan (*Screening*), ke-12 studi yang teridentifikasi sebelumnya dievaluasi lebih lanjut. Proses ini melibatkan pembacaan judul dan abstrak secara seksama untuk setiap studi. Tujuannya adalah untuk menerapkan kriteria pengecualian (seperti yang dijelaskan pada Tabel 1) dan menyaring studi yang paling relevan untuk menjawab pertanyaan penelitian mengenai manajemen memori efisien pada Linux.

Fokus utama pada penyaringan ini adalah memastikan studi memiliki relevansi langsung, terutama dengan menerapkan kriteria E4 (Judul maupun abstrak tidak menyebutkan minimal satu kata kunci utama). Kriteria lain seperti E1 (Bahasa), E2 (Tahun), dan E3 (Jenis Event) juga diperiksa pada tahap ini. Hasil dari proses ini adalah sebanyak 8 studi dieksklusi karena tidak memenuhi satu atau lebih kriteria tersebut. Oleh karena itu, hanya 4 studi yang lolos penyaringan awal ini dan akan dilanjutkan ke tahap pencarian teks lengkap untuk penilaian kelayakan lebih lanjut.

4.5. Penilaian Kualitas

Setelah berhasil mendapatkan teks lengkap, keempat studi memasuki tahap akhir, yaitu Penilaian Kelayakan. Pada fase ini, setiap laporan dibaca secara menyeluruh, dari pendahuluan hingga kesimpulan. Tujuannya adalah untuk melakukan verifikasi final terhadap semua kriteria inklusi dan eksklusi yang telah ditetapkan sebelumnya, memastikan bahwa setiap studi benar-benar relevan dan sesuai dengan lingkup penelitian.

Salah satu aspek penting dalam penilaian ini adalah Penilaian Kualitas. Meskipun kriteria kualitas diterapkan, hasil evaluasi menunjukkan bahwa keempat studi tersebut memiliki metodologi yang cukup solid dan data yang valid sesuai standar yang dibutuhkan. Karena itu, diputuskan bahwa tidak ada studi yang perlu dikecualikan (n = 0) pada tahap ini. Dengan demikian, keempat studi ini secara resmi ditetapkan sebagai studi akhir yang akan dianalisis dan disintesis dalam tinjauan sistematis ini.

4.6. Ekstraksi Data

Setelah keempat studi final berhasil diidentifikasi dan lolos tahap penilaian kualitas, langkah selanjutnya dalam tinjauan sistematis ini adalah ekstraksi data. Proses ini bertujuan untuk mengumpulkan informasi penting dari masing-masing studi terpilih secara terstruktur dan konsisten. Dengan ekstraksi data yang sistematis, diharapkan dapat mempermudah proses analisis serta sintesis temuan guna menjawab pertanyaan penelitian utama, yaitu mengenai strategi-strategi efisiensi manajemen memori dalam *kernel* Linux.

Komponen data yang diekstraksi meliputi pendekatan penelitian, temuan utama, keterbatasan yang diakui oleh penulis, serta kesimpulan dan implikasi dari masing-masing studi. Ringkasan hasil ekstraksi tersebut ditampilkan dalam Tabel 2, yang memberikan gambaran komparatif empat pendekatan utama manajemen memori efisien dalam sistem operasi Linux. Format ini dirancang untuk memudahkan pembaca dalam memahami perbedaan dan kesamaan antar studi, serta mengevaluasi relevansi dan kontribusi masing-masing pendekatan terhadap isu manajemen memori modern.

Tabel 2. List Data Ekstraksi

Artikel	Pendekatan	Temuan	Keterbatasan	Implikasi
COMEX [39]	Ekstensi *kernel* Linux dengan RDMA antar-*node* untuk memori terdistribusi	Peningkatan performa hingga 170x saat memori lokal tidak cukup	Memerlukan perangkat keras RDMA, kompleksitas *page fault* jarak jauh	Efektif untuk HPC dan *cloud computing*
NUMA API [44]	Pengaturan alokasi memori dengan *mbind()*, *set_mempolicy()*	Meningkatkan *locality* dan mengurangi latensi memori	Perlu konfigurasi manual karena bisa menyebabkan fragmentasi	Berguna untuk server basis data dan *workload* paralel
HMM [47]	*Shared Virtual Memory* antara CPU-GPU	Menghilangkan salinan eksplisit CPU-GPU, mendukung *Unified Memory*	Memerlukan *driver* GPU khusus dan arsitektur tertentu	Mempermudah pengembangan aplikasi AI dan *deep learning*
XSBench [21]	*Benchmark* simulasi memori intensif berbasis *OpenMC*	Alat evaluasi performa memori yang portabel dan representatif	Bukan solusi manajemen memori adalah tidak reflektif pada aplikasi nyata	Berguna untuk *profiling* performa sistem memori

4.7. Pembahasan

Selain empat jurnal utama yang dianalisis sebagai hasil seleksi sistematis menggunakan metode PRISMA, pembahasan ini juga dilengkapi dengan referensi tambahan dari literatur ilmiah terbaru (2022–2025). Referensi pendukung ini bertujuan untuk memperkaya konteks analisis, membandingkan temuan utama, serta mengaitkan hasil penelitian dengan perkembangan terkini dalam pengelolaan memori pada sistem operasi Linux. Fokus tambahan mencakup berbagai aspek seperti *tiered memory*, alokator yang *NUMA-aware*, teknik *memory reclaim* dalam lingkungan mesin virtual (VM), serta aktivitas *benchmarking* dan *profiling* berbasis *kernel* Linux.

Efisiensi manajemen memori di sistem operasi Linux merupakan isu sentral dalam pengembangan infrastruktur komputasi modern, terutama seiring meningkatnya kebutuhan akan pemrosesan data berskala besar dan arsitektur heterogen. Keempat artikel yang dianalisis dalam tinjauan ini menyajikan pendekatan berbeda namun saling melengkapi dalam mengatasi keterbatasan memori fisik dan meningkatkan performa sistem.

Artikel pertama oleh [39] memperkenalkan *Cooperative Memory Expansion* (COMEX), sebuah ekstensi *kernel* Linux yang memungkinkan perluasan memori secara kolektif antar simpul dalam klaster menggunakan *Remote Direct Memory Access* (RDMA). Dengan mengintegrasikan memori dari node-node lain melalui perluasan *page table*, COMEX mengatasi keterbatasan memori lokal dan menghindari ketergantungan pada swap tradisional berbasis *disk*. Efisiensi ditingkatkan karena halaman-halaman yang tidak aktif (*cold pages*) dapat dipindahkan secara otomatis ke *node* lain tanpa modifikasi kode aplikasi. Bahkan, COMEX menunjukkan peningkatan kecepatan eksekusi aplikasi hingga 170 kali ketika *footprint* aplikasi 10 kali lebih

pISSN: 2442-3386 eISSN: 2442-4293
Vol 11 No 1 Jan – Jun 2024, 93 - 100
Melek IT
INFORMATION TECHNOLOGY JOURNAL

besar dari kapasitas memori lokal. Sementara itu, *Transparent Page Placement* (TPP) [40] mengusulkan mekanisme penempatan halaman otomatis pada sistem memori berjenjang berbasis CXL. Pendekatan ini memungkinkan promosi dan demosi halaman dilakukan secara efisien berdasarkan pola akses memori. Sebagai solusi pelengkap, Mercury [41] memperkenalkan sistem memori berjenjang berbasis *Quality of Service* (QoS), yang mengelola pembagian memori antar-*tenant* menggunakan mekanisme *admission control* adaptif. Teknik ini mendukung pengendalian performa yang lebih halus dan dinamis dalam skenario *multi-tenant* .

Pendekatan serupa juga dikembangkan dalam sistem Memstrata [28], yang menerapkan mekanisme pengelolaan memori terdistribusi pada arsitektur virtualisasi berbasis CXL. Sistem ini fokus pada isolasi memori antar-tenant serta melakukan *tiering* otomatis dengan latensi yang rendah. Di sisi lain, *Proactive Demotion for Efficient Tiered Memory* (PET) [5] menggunakan teknik prediktif untuk memperkirakan akses halaman, sehingga dapat secara proaktif memindahkan *cold pages* ke lapisan memori bawah, guna meningkatkan efisiensi alokasi memori pada sistem bertingkat. Dalam penelitian terbaru, HybridTier [42] diperkenalkan sebagai solusi *tiered memory* adaptif berbasis CXL yang mampu melacak intensitas akses data baik jangka pendek maupun jangka panjang. Sistem ini mendukung migrasi halaman secara proaktif, yaitu dengan mempromosikan *hot pages* ke lapisan memori cepat dan mendemisikan *cold pages* ke lapisan yang lebih lambat namun lebih murah. Evaluasi menunjukkan peningkatan kinerja hingga 91% dibandingkan metode konvensional dalam lingkungan *multi-tenant*. Sistem ODRP [43] memperluas konsep *disaggregated memory* berbasis RDMA dengan memperkenalkan mekanisme *remote paging* yang diimplementasikan secara *offload* ke dalam perangkat NIC (*Network Interface Card*). Pendekatan ini berhasil menekan *overhead* CPU dan meningkatkan utilisasi memori fisik di antara simpul-simpul dalam sistem terdistribusi. Karena sifatnya yang efisien dan skalabel, ODRP dapat diintegrasikan sebagai bagian dari arah pengembangan arsitektur sistem berikutnya, seperti COMEX.

Sementara itu, pendekatan yang lebih granular diadopsi dalam artikel [44] melalui pengembangan NUMA API untuk Linux. *Non-Uniform Memory Access* (NUMA) memungkinkan kontrol atas lokasi alokasi memori, baik pada node lokal maupun *interleaved* di beberapa *node*. Hal ini penting karena akses memori dari *node* lokal memberikan latensi yang lebih rendah dibanding akses ke memori *remote*. NUMA API menyediakan fleksibilitas melalui kebijakan seperti *bind, preferred*, dan *interleave*, yang bisa diatur baik per proses maupun per blok memori. Dengan demikian, pengembang aplikasi dapat mengoptimalkan latensi dan *bandwidth* sesuai karakteristik beban kerja aplikasinya.

Pengembangan alokator yang *NUMA-aware* juga diimplementasikan dalam sistem NUMAlloc [45], yang memperbaiki mekanisme alokasi heap dengan mempertimbangkan lokasi NUMA melalui deteksi otomatis. NUMAlloc mampu meningkatkan *throughput* hingga 1,4 kali lipat pada aplikasi *multithreaded*, tanpa memerlukan konfigurasi manual terhadap kebijakan alokasi *kernel*. Penelitian Phoenix [46] menunjukkan bahwa pengaturan *thread* dan *page table* secara bersamaan dapat meningkatkan *memory locality* serta mengurangi latensi pada sistem NUMA, terutama dalam konfigurasi dengan jumlah inti (*core*) yang besar.

Selanjutnya, Glisse dan Hubbard dalam presentasi mereka di [47] membahas pendekatan *Heterogeneous Memory Management* (HMM), yang menjadi solusi atas pemisahan spasial antara memori CPU dan GPU. HMM memungkinkan *page fault* dari GPU ditangani oleh *kernel* Linux dengan mentransfer halaman yang dibutuhkan dari CPU ke GPU secara transparan. Hal ini sangat penting dalam konteks *Unified Memory*, di mana aplikasi tidak perlu lagi mengatur eksplisit lokasi data antara CPU dan GPU. Selain itu, integrasi HMM dengan *driver* NVIDIA UVM menjadikan proses migrasi halaman lebih efisien, dengan meminimalisasi latensi yang sebelumnya menjadi hambatan utama pada arsitektur heterogen. Dalam karya terbaru, Cooper et al. [48] mengkaji implementasi *Shared Virtual Memory* (SVM) pada arsitektur GPU AMD, yang memungkinkan berbagi data antara CPU dan GPU tanpa memerlukan penyalinan data secara eksplisit. Pendekatan ini semakin memperkokoh fondasi konsep *Unified Memory* dan membuka ruang bagi peningkatan efisiensi dalam *workload* yang bersifat heterogen, terutama yang melibatkan eksekusi bersama CPU-GPU.

Artikel terakhir, XSBench oleh [21], meskipun bukan secara langsung merupakan teknik manajemen memori, memberikan konteks penting dalam pengujian efisiensi memori melalui simulasi beban kerja intensif memori. XSBench menyederhanakan *kernel* perhitungan dari *OpenMC* dan memodelkan beban kerja yang sensitif terhadap pola akses memori. Melalui studi *scaling multithread*, ditemukan bahwa efisiensi menurun pada *thread* lebih tinggi, mengindikasikan adanya hambatan baik dari sisi memori maupun komunikasi antar thread. Untuk memahami kinerja dalam skenario memori besar dan *tiered memory* , pendekatan *benchmark* seperti GPAC dan HotMem juga digunakan untuk mengevaluasi efisiensi *memory reclaim* serta konfigurasi yang *VM-aware* [49][50]. Hal ini penting karena menunjukkan bagaimana batas performa terkait memori dapat dianalisis lebih transparan sebelum dioptimalkan lebih lanjut melalui teknik seperti NUMA atau *COMEX*.

Tabel 3. Perbandingan empat pendekatan utama

Pendekatan	Arsitektur / Lingkungan	Teknik Manajemen Memori	Kelebihan	Kekurangan	Skenario Penggunaan
COMEX	Klaster server terdistribusi (*disaggregated memory*)	Ekstensi *kernel* Linux dengan RDMA antar-*node* untuk memperluas memori virtual secara kolektif	• Transparan bagi aplikasi (tidak perlu modifikasi kode) • Peningkatan performa drastis saat memori lokal tidak cukup • Mengurangi kebutuhan swap ke disk	• Memerlukan perangkat keras RDMA (*Infiniband* / RoCE) • Kompleksitas manajemen *page-fault* jarak jauh • Belum tersedia di kernel Linux arus utama	HPC, *cloud computing*, *big-data analytics* dengan *footprint* memori besar
NUMA API	Server *multi-socket* / arsitektur NUMA	Pengaturan alokasi memori dengan *mbind()*, *set_mempolicy()*, dan *numactl* berdasarkan *node*	• Kontrol granular atas lokasi alokasi memori • Meningkatkan *locality* dan mengurangi latensi memori	• Memerlukan konfigurasi manual • Tidak berlaku lintas mesin • Dapat menyebabkan fragmentasi jika salah setel	Server basis data, VM berbeban berat, *workload* paralel lokal
HMM	Sistem heterogen CPU-GPU / SoC	*Shared Virtual Memory* (SVM) Linux untuk memungkinkan CPU dan GPU berbagi alamat virtual	• Menghilangkan kebutuhan salinan eksplisit antara CPU-GPU • Mendukung *Unified Memory* Mempermudah pengembangan aplikasi akselerasi	• Memerlukan dukungan dari *kernel*, driver GPU (UVM), dan arsitektur tertentu • Kinerja sensitif terhadap latensi *page migration*	*Deep learning*, *embedded* AI (*Jetson*, *Snapdragon*), *workload* GPU
XSBench	*Benchmark* simulasi memori (*mini-app*)	Simulasi pola akses memori acak dari *OpenMC* digunakan untuk *profiling* performa	• Portabel, ringan, dan mudah digunakan • Representatif untuk aplikasi memori intensif	• Bukan pendekatan solusi, hanya alat evaluasi • Tidak mencerminkan kompleksitas aplikasi penuh	Evaluasi performa memori sistem (NUMA, *tiered memory*, HMM,

Tabel 4. Ringkasan artikel literatur

Penulis	Pendekatan	Metode	Fungsi
Srinuan et al., 2020	*Cooperative Memory Expansion* (COMEX)	Ekstensi *kernel* Linux dengan pemanfaatan RDMA antar simpul dalam klaster	Memperluas kapasitas memori efektif antar *node* untuk menghindari penggunaan *swap disk* dan meningkatkan *throughput*
Kammerdiener et al., 2025	NUMA API	Penyediaan antarmuka dan kebijakan *bind*, *preferred*, dan interleave via *libnuma* dan *numactl*	Mengurangi latensi dan/atau meningkatkan *bandwidth* dengan mengatur lokasi alokasi memori secara eksplisit
Allen et al., 2024	*Heterogeneous Memory Management* (HMM)	Integrasi *kernel* Linux untuk mendukung *Unified Memory* antara CPU dan GPU dengan penanganan *page fault* otomatis	Memungkinkan migrasi halaman memori secara transparan; mengoptimalkan akses lintas CPU-GPU tanpa modifikasi kode
Tirumalasetty et al., 2022	*Proxy Benchmark* (XSBench)	Pembuatan aplikasi ringan yang merepresentasikan pola akses memori intensif dari *OpenMC*	Memberikan dasar analisis performa sistem terhadap beban kerja memori tinggi, mengidentifikasi *bottleneck* skala besar

pISSN: 2442-3386 eISSN: 2442-4293
Vol 11 No 1 Jan – Jun 2024, 95 - 100
Melek IT
INFORMATION TECHNOLOGY JOURNAL

Selain XSBench, studi *benchmarking* terbaru juga mengandalkan pendekatan seperti GPAC [50] dan HotMem [49] untuk mengevaluasi efisiensi *reclaim* memori dalam konteks *tiered memory* dan *serverless* virtual machines. Di sisi lain, MemPerf [33] memperkenalkan teknik *profiling* ringan untuk mengukur overhead allocator, memungkinkan analisis bottleneck memori secara lebih presisi dalam eksperimen *kernel-level*. Penelitian oleh Ghaemi [51] bahkan lebih lanjut mengembangkan MemScope, sebuah *toolkit benchmarking* heterogen yang dirancang untuk mengukur *bandwidth* dan latensi memori secara langsung pada tingkat *kernel*, memberikan analisis yang lebih rinci dan akurat terhadap kinerja subsistem memori.

Agar lebih mudah dipahami, pada Tabel 3 adalah untuk membandingkan keempat pendekatan utama berdasarkan beberapa aspek penting seperti arsitektur yang digunakan, teknik pengelolaan memori, kelebihan dan kelemahan, serta skenario penerapannya masing-masing.

Tabel 4 merupakan ringkasan setiap artikel, pendekatan manajemen memori yang dibahas, serta fungsi/peran metode tersebut dalam konteks efisiensi memori di sistem operasi Linux. Setiap entri mengacu pada pembahasan tematik di atas dan dilengkapi dengan sitasi.

Pada Tabel 4 telah diberikan gambaran lengkap dari semua pendekatan yang sudah dibahas sebelumnya. Selain itu, tabel ini juga menunjukkan kontribusi masing-masing pendekatan dalam meningkatkan efisiensi pengelolaan memori di sistem operasi Linux. Meskipun keempat metode tersebut memiliki perbedaan dalam cara kerja dan lingkungan penerapannya, semuanya punya peran penting dalam mengatasi berbagai tantangan memori modern. Mulai dari sistem terdistribusi, NUMA, komputasi heterogen, hingga dalam pengukuran performa sistem.

4.7.1. Analisis Perbandingan COMEX vs NUMA API

COMEX menawarkan solusi inovatif untuk sistem klaster dengan memperluas kapasitas memori virtual melalui RDMA antar-*node*. Dalam skenario di mana memori lokal tidak mencukupi, COMEX dapat meningkatkan performa hingga 170x karena kemampuannya mengakses memori *node* lain secara transparan tanpa modifikasi aplikasi. Selain itu, COMEX mengurangi kebutuhan swap ke disk, yang sering menjadi *bottleneck* dalam sistem besar.

Sebaliknya, NUMA API hanya berlaku dalam satu mesin fisik dan tidak mampu mengakses memori di luar *node*. Meskipun NUMA efektif dalam meningkatkan *locality* dan mengurangi latensi pada server *multi-socket*, ia memiliki keterbatasan dalam hal skalabilitas dan fleksibilitas distribusi memori. Misalnya, NUMA API memerlukan konfigurasi manual dan rentan terhadap fragmentasi jika tidak dikonfigurasi dengan benar.

Oleh karena itu, COMEX lebih unggul dalam konteks sistem terdistribusi, sedangkan NUMA API tetap menjadi pilihan utama untuk optimasi memori dalam *single-node* sistem berskala besar. Pemilihan pendekatan tergantung pada arsitektur sistem dan jenis beban kerja.

4.7.2. Benchmark XSBench dalam Evaluasi COMEX dan HMM

XSBench merupakan *mini-benchmark* yang mensimulasikan pola akses memori intensif seperti pada *OpenMC*. *Benchmark* ini dapat digunakan untuk mengevaluasi efektivitas COMEX dalam memperluas kapasitas memori virtual serta HMM dalam mengelola migrasi halaman antara CPU dan GPU.

Dalam pengujian COMEX, XSBench dapat mengukur penurunan jumlah *page fault* saat *footprint* memori melebihi batas lokal. Sedangkan pada HMM, XSBench membantu menilai *overhead page migration* antara CPU dan GPU, serta efisiensi *Unified Memory*. Hasil *benchmark* ini dapat menjadi dasar dalam memilih pendekatan manajemen memori yang paling sesuai dengan beban kerja tertentu, baik dalam sistem heterogen maupun terdistribusi.

V. PENUTUP

5.1. Kesimpulan

Penelitian ini berhasil mengidentifikasi empat pendekatan utama yang berkontribusi pada efisiensi manajemen memori dalam sistem operasi Linux. Teknik-teknik seperti COMEX, NUMA API, HMM, dan *benchmark* XSBench menunjukkan bahwa pengelolaan memori yang adaptif, baik pada level perangkat keras maupun perangkat lunak, sangat penting untuk meningkatkan kinerja sistem. Pendekatan-pendekatan tersebut menawarkan solusi terhadap keterbatasan memori fisik, optimalisasi lokasi alokasi memori, dan efisiensi pemrosesan dalam arsitektur heterogen. Dengan demikian, studi ini memberikan landasan konseptual dan teknis bagi pengembangan strategi manajemen memori yang lebih canggih dan responsif terhadap kebutuhan komputasi masa kini.

5.2. Saran

Pengembangan sistem operasi Linux ke depan hendaknya memperhatikan integrasi pendekatan manajemen memori yang adaptif guna mengatasi keterbatasan memori fisik serta meningkatkan efisiensi alokasi dalam berbagai skenario komputasi modern. Pendekatan seperti COMEX dan NUMA API memiliki potensi besar untuk dikembangkan lebih lanjut, baik secara individual maupun dalam kombinasi. Optimasi pemanfaatan RDMA dalam COMEX dapat memberikan peningkatan performa signifikan pada lingkungan klaster server, sementara otomatisasi kebijakan distribusi dalam NUMA akan mempermudah penggunaannya

dalam sistem berskala besar tanpa mengorbankan kontrol granular atas lokasi alokasi memori.

Selain itu, perluasan dukungan untuk arsitektur memori heterogen seperti *Heterogeneous Memory Management* (HMM) menjadi sangat penting. Dengan semakin populernya beban kerja AI / ML dan aplikasi berbasis GPU, kemampuan migrasi halaman transparan antara CPU dan GPU harus ditingkatkan agar tidak menyebabkan *bottleneck* atau *overhead* tinggi. Penggunaan mekanisme *Unified Memory* dalam HMM juga perlu didukung oleh *driver* GPU dan arsitektur *hardware* yang lebih luas agar implementasinya tidak terbatas hanya pada platform tertentu.

Sebagai alat evaluasi yang objektif dan representatif, *benchmark* seperti XSBench sebaiknya digunakan secara rutin dalam tahap awal pengembangan strategi manajemen memori. Hal ini akan membantu identifikasi potensi masalah performa sejak dini, sehingga pendekatan manajemen memori yang diterapkan lebih responsif terhadap karakteristik beban kerja masa kini. Penelitian seperti *Motivating Next-Generation OS Memory Management for Terabyte NVMMs* [52] menyarankan bahwa pendekatan tradisional dalam pengelolaan memori perlu direvisi dan dirancang ulang agar mampu beradaptasi dengan perkembangan teknologi memori non-volatil berskala besar.

Lebih jauh lagi, pendekatan manajemen memori yang telah dianalisis dalam penelitian ini juga berpotensi untuk diterapkan dalam domain lain yang memiliki keterbatasan sumber daya, seperti Android, *Internet of Things* (IoT), dan *embedded* Linux Sistem operasi pada platform-platform tersebut umumnya memiliki batasan energi, kapasitas memori, serta kebutuhan real-time, sehingga adaptasi teknik seperti HMM untuk CPU-GPU SoC atau alokator *NUMA-aware* versi ringan dapat memberikan efisiensi tambahan tanpa membebani sistem.

DAFTAR PUSTAKA

[1] R. Prabhu, A. Nayak, J. Mohan, R. Ramjee, and A. Panwar, "vAttention: Dynamic memory management for serving LLMs without pagedattention," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2024, pp. 1133–1150. doi: 10.1145/3669940.3707256.

[2] D. Moher, D. G. Altman, and J. Tetzlaff, *PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses)*. 2014. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85178329658&partnerID=40&md5=9f5d0e12fab5decab7d997e7451124e9

[3] M. J. Page *et al.*, "Updating guidance for reporting systematic reviews: development of the PRISMA 2020 statement," *J. Clin. Epidemiol.*, vol. 134, pp. 103–112, 2021, doi: 10.1016/j.jclinepi.2021.02.003.

[4] A. Suryavanshi and S. K. Sharma, "Result analysis of the improved contiguous memory allocation (ICMA) approach in the Linux kernel research," *Int. J. Performability Eng.*, vol. 19, no. 11, pp. 753–761, 2023, doi: 10.23940/ijpe.23.11.p6.753761.

[5] W. Doh *et al.*, "PET: Proactive demotion for efficient tiered memory management," in *EuroSys 2025 - Proceedings of the 2025 20th European Conference on Computer Systems*, 2025, pp. 854–869. doi: 10.1145/3689031.3717471.

[6] T. Lee, S. K. Monga, C. Min, and Y. I. Eom, "MEMTIS: Efficient memory tiering with dynamic page classification and page size determination," in *SOSP 2023 - Proceedings of the 29th ACM Symposium on Operating Systems Principles*, 2023, pp. 17–34. doi: 10.1145/3600006.3613167.

[7] M. Giardino, K. Doshi, and B. Ferri, "Soft2LM: Application guided heterogeneous memory management," in *2016 IEEE International Conference on Networking Architecture and Storage, NAS 2016 - Proceedings*, 2016. doi: 10.1109/NAS.2016.7549421.

[8] Y. Liang, Q. Li, and C. J. Xue, "Mismatched memory management of android smartphones," in *11th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2019, co-located with USENIX ATC 2019*, 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85088227803&partnerID=40&md5=224001910e0705fe622a939f9187e7f1

[9] J. Zhang and C. Jiang, "Primary exploration on teaching of Linux operating system course," in *WIT Transactions on Information and Communication Technologies*, 2014, pp. 971–976. doi: 10.2495/ICTE131172.

[10] H. Zhao, "A design of information teaching platform based on Linux operating system," in *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/2138/1/012018.

[11] P. Hunter, "Linux security: Separating myth from reality," *Netw. Secur.*, vol. 2004, no. 8, pp. 8 – 9, 2004, doi: 10.1016/S1353-4858(04)00116-3.

[12] M. Blank, S. Brunner, T. Fuhrmann, H. Meier, and M. Niemetz, "Embedded Linux in engineering education," in *IEEE Global Engineering Education Conference, EDUCON*, 2015, pp. 145–150. doi:

10.1109/EDUCON.2015.7095964.

[13] J. Huang, M. K. Qureshi, and K. Schwan, "An evolutionary study of linux memory management for fun and profit," in *Proceedings of the 2016 USENIX Annual Technical Conference, USENIX ATC 2016*, 2016, pp. 465–478. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85021948275&partnerID=40&md5=e40f41db099eb558cdfe1e111889b091

[14] E. Anderson and J. Tucek, "Efficiency matters!," *Oper. Syst. Rev.*, vol. 44, pp. 40–45, 2010, doi: 10.1145/1740390.1740400.

[15] V. Vasudevan, D. G. Andersen, and M. Kaminsky, "The case for VOS: The vector operating system," in *13th Workshop on Hot Topics in Operating Systems, HotOS 2011*, 2011, p. 31.

[16] K. Sacha, "Measuring the real-time operating system performance," 1995, pp. 34–40. doi: 10.1109/EMWRTS.1995.514289.

[17] Q. Huang, "Exploring the use of ChatGPT for a systematic literature review: A design-based research," 2024. doi: 10.48550/arXiv.2409.17426.

[18] K. Staffs, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[19] B. Tabatabai, M. Mansi, and M. M. Swift, "FBMM: Using the VFS for extensibility in kernel memory management," in *HotOS 2023 - Proceedings of the 19th Workshop on Hot Topics in Operating Systems*, 2023, pp. 181–187. doi: 10.1145/3593856.3595908.

[20] A. Suryavanshi and S. Sharma, "An approach towards improvement of contiguous memory allocation linux kernel: A review," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 25, no. 3, pp. 1607–1614, 2022, doi: 10.11591/ijeecs.v25.i3.pp1607-1614.

[21] C. Tirumalasetty, C. C. Chou, N. Reddy, P. Gratz, and A. Abouelwafa, "Reducing minor page fault overheads through enhanced page walker," *ACM Trans. Archit. Code Optim.*, vol. 19, no. 4, 2022, doi: 10.1145/3547142.

[22] P. Gangar, A. Panwar, and K. Gopinath, "EMD: Fair and efficient dynamic memory de-bloating of transparent huge pages," in *Proceedings of the 2025 ACM SIGPLAN International Symposium on Memory Management*, 2025, pp. 1–13. doi: 10.1145/3735950.3735952.

[23] K. Mores, S. Psomadakis, and G. Goumas, "eBPF-mm: Userspace-guided memory management in Linux with eBPF," *arXiv Prepr. arXiv2409.11220*, 2024.

[24] C. Bilbao, J. C. Saez, and M. Prieto-Matias, "Flexible system software scheduling for asymmetric multicore systems with PMCSched: A case for intel alder lake," *Concurr. Comput. Pract. Exp.*, vol. 35, no. 25, 2023, doi: 10.1002/cpe.7814.

[25] M. Bailleu, D. Stavrakakis, R. Rocha, S. Chakraborty, D. Garg, and P. Bhatotia, "Toast: A heterogeneous memory management system," in *Parallel Architectures and Compilation Techniques - Conference Proceedings, PACT*, 2024, pp. 53–65. doi: 10.1145/3656019.3676944.

[26] L. Wrenger, K. Albes, M. Wurps, C. Dietrich, and D. Lohmann, "HyperAlloc: Efficient VM memory de/inflation via hypervisor-shared page-frame allocators," in *EuroSys 2025 - Proceedings of the 2025 20th European Conference on Computer Systems*, 2025, pp. 702–719. doi: 10.1145/3689031.3717484.

[27] P. Duraisamy *et al.*, "Towards an adaptable systems architecture for memory tiering at warehouse-scale," in *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, 2023, pp. 727–741. doi: 10.1145/3582016.3582031.

[28] Y. Zhong *et al.*, "Managing memory tiers with CXL in virtualized environments," in *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024*, 2024, pp. 37–56.

[29] Y. Sun *et al.*, "M5: Mastering page migration and memory management for CXL-based tiered memory systems," in *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, 2025, pp. 604–621. doi: 10.1145/3676641.3711999.

[30] A. Hinderks, F. Jose, D. Mayo, J. Thomaschewski, and M. J. Escalona, "An SLR-tool: Search process in practice : To conduct and manage systematic literature review (SLR)," in *Proceedings - 2020 ACM/IEEE 42nd International Conference on Software Engineering: Companion, ICSE-Companion 2020*, 2020, pp. 81–84. doi: 10.1145/3377812.3382137.

[31] W. Gunathilake and T. Neligwa, "A quality assessment framework for KMS software: Reflections on conducting a systematic

literature review," in *Proceedings - KIM 2013, Knowledge and Information Management Conference: Sustainable Quality*, 2013, pp. 13 – 23. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84910685977&partnerID=40&md5=5cb29e2fc4054e031e6bf7878e19c206

[32] S. Imtiaz, M. Bano, N. Ikram, and M. Niazi, "A tertiary study: Experiences of conducting systematic literature reviews in software engineering," in *ACM International Conference Proceeding Series*, 2013, pp. 177 – 182. doi: 10.1145/2460999.2461025.

[33] J. Zhou *et al.*, "MemPerf: Profiling allocator-induced performance slowdowns," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA2, pp. 1418–1441, 2023, doi: 10.1145/3622848.

[34] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *J. Syst. Softw.*, vol. 80, no. 4, pp. 571 – 583, 2007, doi: 10.1016/j.jss.2006.07.009.

[35] R. Asyrofi, M. R. Dewi, M. I. Lutfhi, and P. Wibowo, "Systematic literature review langchain proposed," in *IES 2023 - International Electronics Symposium: Unlocking the Potential of Immersive Technology to Live a Better Life, Proceeding*, 2023, pp. 533–537. doi: 10.1109/IES59143.2023.10242497.

[36] M. R. Jantz, C. Strickland, K. Kumar, M. Dimitrov, and K. A. Doshi, "A framework for application guidance in virtual memory systems," in *VEE 2013 - Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2013, pp. 155 – 165. doi: 10.1145/2451512.2451543.

[37] Z. Pin, V. Pandey, J. Sundaresan, A. Raghuraman, Z. Yuanyuan, and S. Kumar, "Dynamic tracking of page miss ratio curve for memory management," in *Operating Systems Review (ACM)*, 2004, pp. 177 – 188. doi: 10.1145/1037949.1024415.

[38] H. K. Lee, B. S. An, and E. J. Kim, "Adaptive prefetching scheme using web log mining in Cluster-based web systems," in *2009 IEEE International Conference on Web Services, ICWS 2009*, 2009, pp. 903 – 910. doi: 10.1109/ICWS.2009.127.

[39] P. Srinuan, X. Yuan, and N. F. Tzeng, "Cooperative memory expansion via OS kernel support for networked computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 11, pp. 2650–2667, 2020, doi: 10.1109/TPDS.2020.2999507.

[40] H. Al Maruf *et al.*, "TPP: Transparent page placement for CXL-enabled tiered-memory," in *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, 2023, pp. 742–755. doi: 10.1145/3582016.3582063.

[41] J. Lu *et al.*, "Mercury: QoS-aware tiered memory system," *arXiv Prepr. arXiv2412.08938*, 2024.

[42] K. Song, J. Yang, S. Liu, and G. Pekhimenko, "Lightweight frequency-based tiering for CXL memory systems," *arXiv Prepr. arXiv2312.04789*, 2023, [Online]. Available: http://arxiv.org/abs/2312.04789

[43] Z. Wang, X. Wei, J. Gu, H. Xie, R. Chen, and H. Chen, "ODRP: On-demand remote paging with programmable RDMA," in *Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation, NSDI 2025*, 2025, pp. 1101–1115.

[44] B. Kammerdiener, J. Z. McMichael, M. Jantz, K. Doshi, and T. Jones, "Flexible and effective object tiering for heterogeneous memory systems," *ACM Trans. Archit. Code Optim.*, vol. 22, no. 1, 2025, doi: 10.1145/3708540.

[45] H. Yang *et al.*, "NUMAlloc: A faster NUMA memory allocator," in *International Symposium on Memory Management, ISMM*, 2023, pp. 97–110. doi: 10.1145/3591195.3595276.

[46] M. Siavashi, A. Sanaee, M. Sharifi, and G. Antichi, "Phoenix--A novel technique for performance-aware orchestration of thread and page table placement in NUMA systems," *arXiv Prepr. arXiv2502.10923*, 2025.

[47] T. Allen, B. Cooper, and R. Ge, "Fine-grain quantitative analysis of demand paging in unified virtual memory," *ACM Trans. Archit. Code Optim.*, vol. 21, no. 1, 2024, doi: 10.1145/3632953.

[48] B. Cooper, T. R. W. Scogland, and R. Ge, "Shared virtual memory: Its design and performance implications for diverse applications," in *Proceedings of the International Conference on Supercomputing*, 2024, pp. 26–37. doi: 10.1145/3650200.3656608.

[49] O. Lagkas Nikolos, C. Alverti, S. Psomadakis, G. Goumas, and N. Koziris, "Fast and efficient memory reclamation for serverless MicroVMs," *arXiv e-prints*, p. arXiv--2411, 2024.

[50] C. Prakash, A. Prasad, S. Kumar, and S. Subramoney, "Efficient memory tiering in a

pISSN: 2442-3386 eISSN: 2442-4293
Melek IT
Vol 11 No 1 Jan – Jun 2024, 99 - 100
INFORMATION TECHNOLOGY JOURNAL

virtual machine," *arXiv Prepr. arXiv2506.06067*, 2025, [Online]. Available: http://arxiv.org/abs/2506.06067

[51] G. Ghaemi, K. Taram, and R. Mancuso, "Heterogeneous memory benchmarking toolkit," *arXiv Prepr. arXiv2505.00901*, 2025.

[Halaman ini dibiarkan kosong]