KOMTELKA: Jurnal Komputer Telekomunikasi dan Elektronika, Volume 01 Nomor 1 Tahun 2026
Tersedia *online* di
https://jurnal.rajawalicemerlangabadi.com/index.php/jpek

Perbandingan Arsitektur Container Dan Virtual Machine Dalam Optimasi Deployment Aplikasi Skala Besar

Yusuf Maulana Saputra[1], Muhammad Rizqi utomo[2], Penulis[3]

[1,2,3]*Jurusan Teknik Elektro Universitas Singaperbangsa Karawang*

email: [1]yasafmaulana0218@gmail.com, [2]rzqutm@gmail.com, [3]yuliarman@yahoo.com

ABSTRAK

Pesatnya perkembangan teknologi cloud computing menuntut efisiensi tinggi dalam deployment aplikasi skala besar, di mana pilihan antara arsitektur Virtual Machine (VM) dan Container menjadi krusial. Penelitian ini bertujuan untuk membandingkan performa kedua arsitektur tersebut dalam aspek penggunaan sumber daya, kecepatan start-up, dan skalabilitas pada lingkungan aplikasi modern. Metode penelitian dilakukan melalui studi literatur komperatif dan analisis data sekunder dari berbagai pengujian benchmark teknis serta implementasi industri. Pendekatan yang diusulkan berfokus pada optimasi deployment melalui penggunaan teknologi container yang didukung oleh sistem orkestrasi seperti Kubernetes dan Docker Swarm untuk menjaga ketersediaan tinggi (high availability). Kontribusi utama penelitian ini adalah sintesis data performa yang menunjukkan bahwa container memiliki efisiensi penggunaan memori hingga delapan kali lebih baik dibandingkan VM serta waktu booting yang jauh lebih singkat. Kebaruan penelitian ini terletak pada analisis komprehensif yang menghubungkan efisiensi kernel sharing pada container dengan strategi load balancing dan service chaining jaringan untuk memitigasi latensi pada infrastruktur skala besar. Hasil penelitian menyimpulkan bahwa meskipun VM menawarkan isolasi keamanan yang lebih kuat, arsitektur container memberikan optimasi yang jauh lebih unggul bagi aplikasi skala besar yang membutuhkan elastisitas dan kecepatan tinggi. Implementasi orkestrasi kontainer terbukti mampu meningkatkan stabilitas sistem secara signifikan dalam menghadapi lonjakan trafik pada sektor industri.

Kata kunci:
Virtualisasi
Docker Container
Virtual Machine
Optimasi Deployment
Skala Besar.

ABSTRACT

The rapid advancement of cloud computing technology demands high efficiency in large-scale application deployment, where the choice between Virtual Machine (VM) and Container architectures becomes crucial. This research aims to compare the performance of these two architectures in terms of resource utilization, start-up speed, and scalability within modern application environments. The research method is conducted through a comparative literature study and secondary data analysis from various technical benchmarks and industrial implementations. The proposed approach focuses on deployment optimization through container technology supported by orchestration systems such as Kubernetes and Docker Swarm to maintain high availability. The main contribution of this research is the synthesis of performance data showing that containers have memory usage efficiency up to eight times better than VMs and significantly shorter booting times.

Keywords:
Virtualization
Docker Container
Virtual Machine
Deployment Optimization
Large Scale

The novelty of this study lies in its comprehensive analysis linking the efficiency of kernel sharing in containers with network strategies such as load balancing and service chaining to mitigate latency in large-scale infrastructures. The results conclude that while VMs offer stronger security isolation, container architecture provides superior optimization for large-scale applications requiring high elasticity and speed. Container orchestration implementation is proven to significantly enhance system stability when facing traffic surges in the industrial sector.

PENDAHULUAN

Pesatnya perkembangan teknologi informasi dan transformasi digital telah mendorong organisasi untuk mengadopsi infrastruktur yang lebih lincah dan efisien. Dalam ekosistem *cloud computing*, pengelolaan sumber daya server menjadi aspek krusial untuk memastikan aplikasi dapat berjalan dengan optimal. Teknologi virtualisasi tradisional yang berbasis *Virtual Machine* (VM) telah lama digunakan untuk menyediakan isolasi sumber daya, namun sering kali dihadapkan pada tantangan efisiensi karena setiap unit memerlukan *Guest Operating System* yang memakan banyak memori [1]. Seiring dengan kebutuhan pengembangan aplikasi yang lebih cepat dan fleksibel, teknologi kontainer muncul sebagai solusi alternatif yang lebih ringan (*lightweight*) untuk mendukung penyiapan *local development server* yang *scalable* [1], [9].

Urgensi penelitian ini didasarkan pada masalah nyata di industri, di mana aplikasi berbasis monolitik sering kali mengalami kegagalan saat menangani lonjakan trafik yang besar, seperti yang terjadi pada sistem operasional PT Mandiri Utama Finance [4]. Untuk mengatasi hal tersebut, transisi menuju arsitektur *microservices* melalui penggunaan kontainer menjadi sangat relevan. Penelitian terdahulu menunjukkan bahwa kontainer tidak hanya unggul dalam hal efisiensi penggunaan aset fisik, tetapi juga mampu memitigasi biaya infrastruktur melalui pemanfaatan sumber daya tunggal untuk banyak instans logis [7]. Selain itu, karakteristik kontainer yang cepat untuk dibuat (*fast provisioning*) sangat mendukung platform otomatisasi yang membutuhkan waktu tanggap minimal [9].

Meskipun kontainer menawarkan keunggulan dalam hal kecepatan *booting* dan efisiensi memori, implementasi pada aplikasi skala besar memerlukan strategi manajemen klaster yang kompleks. Penggunaan platform orkestrasi seperti Kubernetes dan Docker Swarm menjadi standar untuk menjaga ketersediaan tinggi

(*high availability*) dan skalabilitas sistem [8]. Studi komparatif telah dilakukan untuk melihat performa Kubernetes di atas berbagai jenis virtualisasi seperti KVM, Vagrant, dan LXD, yang menunjukkan bahwa pemilihan landasan virtualisasi tetap berpengaruh pada penggunaan CPU dan memori [6]. Di sisi lain, optimasi *deployment* juga harus memperhatikan aspek jaringan melalui teknik *Load Balancing* [7] dan *Network Function Virtualization* (NFV) guna memitigasi potensi penurunan kinerja jalur data saat fungsi jaringan dijalankan dalam rantai layanan kontainer [5].

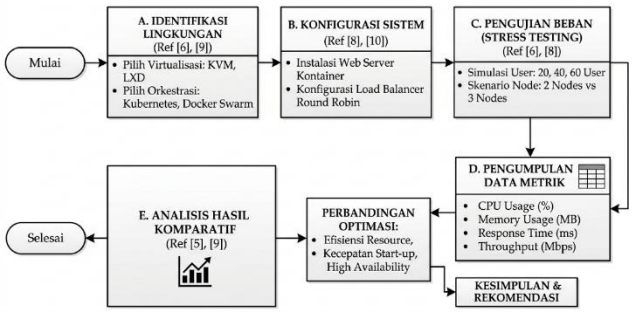
Penelitian ini bertujuan untuk melakukan analisis mendalam mengenai perbandingan arsitektur kontainer dan *virtual machine* dalam mengoptimalkan *deployment* aplikasi pada skala besar. Fokus utama pengkajian meliputi aspek efisiensi sumber daya, kehandalan sistem melalui mekanisme *high availability* pada klaster seperti K3S dan Docker Swarm [8], serta performansi *web server* dalam menangani beban trafik melalui distribusi *node* yang optimal [7]. Melalui tinjauan komprehensif terhadap berbagai metode *benchmark* dan studi kasus industri, penelitian ini diharapkan dapat memberikan kontribusi berupa pedoman strategis bagi para pengembang sistem dalam memilih teknologi virtualisasi yang tepat guna mencapai efisiensi operasional yang maksimal.

METODE PENELITIAN

Metode penelitian yang digunakan dalam artikel ini adalah studi literatur sistematis (*Systematic Literature Review*). Penelitian dilakukan dengan menganalisis, mensintesis, dan membandingkan hasil temuan dari berbagai penelitian terdahulu yang relevan dengan optimasi *deployment* aplikasi menggunakan teknologi kontainer dan *virtual machine*.

A. Tahapan Penelitian

Proses penelitian ini disusun secara sistematis agar hasil analisis yang diperoleh valid dan terstruktur. Tahapan-tahapan tersebut diilustrasikan dalam diagram alur pada Gambar 1.



Gambar 1. diagram alur metodologi penelitian

Berdasarkan Gambar 1, penelitian ini dijalankan melalui empat tahapan utama yang terstruktur sebagai berikut:

1. Identifikasi Lingkungan

   Menentukan parameter virtualisasi, di mana Kubernetes klaster dijalankan di atas berbagai jenis virtualisasi seperti KVM, Vagrant, dan LXD [6], serta penggunaan platform orkestrasi seperti Docker Swarm dan K3S [8].
2. Konfigurasi Sistem

   Penyiapan server *deployment* yang mencakup instalasi *web server* berbasis *container* [9] dan konfigurasi *load balancing* menggunakan algoritma *round robin* [7].
3. Pengujian (*Stress Testing*)

   Melakukan simulasi beban trafik dengan jumlah pengguna yang meningkat (20, 40, hingga 60 *user*) untuk mengukur ketersediaan tinggi (*high availability*) dan waktu respon [6], [7].
4. Analisis Data

   Membandingkan hasil metrik penggunaan CPU, memori, dan *throughput* jaringan dari kedua arsitektur [5], [8].

B. Indikator Keberhasilan

Tolak ukur keberhasilan penelitian ini ditentukan berdasarkan efisiensi penggunaan sumber daya dan stabilitas layanan dalam menangani beban kerja skala besar. Indikator utama yang digunakan sebagai parameter evaluasi disajikan dalam Tabel 1.

Tabel. 1 Parameter dan Indikator Pengujian Performa

No	Parameter	Indikator	Satuan
1	Jaringan	Throughput	bps / Mbps
2	Sumber Daya	CPU Usage	Persentase (%)
3	Sumber Daya	Memory Usage	MB / GB
4	Layanan	Response Time	Second (s) / ms

1. **Throughput**

   Mengukur volume data yang berhasil ditransfer melalui jalur komunikasi dalam satuan waktu tertentu. Indikator ini digunakan untuk mengevaluasi efektivitas *load balancing* dan dampak *service chaining* pada lingkungan kontainer [2], [5].

2. **CPU Usage**

   Menunjukkan persentase beban kerja pada unit pemrosesan pusat. Parameter ini krusial untuk membuktikan efisiensi kontainer yang berjalan langsung di atas *host kernel* dibandingkan VM yang memerlukan *hypervisor* [6], [8].

3. **Memory Usage**

   Mengukur konsumsi RAM yang digunakan oleh masing-masing unit virtualisasi. Efisiensi memori menjadi indikator keberhasilan utama dalam mendukung *deployment* banyak instans pada satu aset fisik [7], [9].

4. **Response Time**

   Menghitung waktu yang dibutuhkan server untuk merespon permintaan pengguna. Indikator ini diukur melalui skenario *stress testing* untuk memastikan ketersediaan tinggi (*high availability*) saat trafik meningkat [4], [8].

C. Arsitektur dan Skenario Pengujian

Analisis dalam penelitian ini didasarkan pada perbedaan mendasar struktur hirarki antara *Virtual Machine* dan *Container*, sebagaimana telah diilustrasikan sebelumnya pada Gambar 1. Pada pengujian aplikasi skala besar, optimasi kinerja dilakukan dengan mendistribusikan beban kerja secara merata

ke beberapa *node* untuk mencegah terjadinya kegagalan sistem akibat lonjakan trafik (*traffic spike*) [4]. Prinsip distribusi beban kerja dalam klaster kontainer menggunakan mekanisme *load balancing* dasar. Efisiensi pembagian permintaan pada setiap *node* dapat dihitung menggunakan persamaan berikut:

$$W_i = \frac{R}{N} \quad (1)$$

Di mana W_i merupakan beban kerja per-*node*, R adalah total permintaan (*request*) yang masuk ke dalam sistem, dan N adalah jumlah *node* aktif yang tersedia di dalam klaster [7]. Persamaan ini menjadi landasan dalam menentukan efektivitas algoritma *Round Robin* yang sering diterapkan pada layanan berbasis Docker dan Kubernetes.

Skenario pengujian yang dianalisis dalam studi literatur ini mencakup beberapa aspek krusial:

1. **Variasi Jumlah Node**
   Membandingkan performa klaster dengan jumlah *node* yang berbeda (misalnya 2 *node* dibandingkan dengan 3 *node*). Hal ini bertujuan untuk mengamati korelasi antara penambahan aset fisik terhadap peningkatan *Request per Second* (RPS) dan efisiensi *Time per Request* [7].
2. **Evaluasi High Availability**
   Menguji kemampuan orkestrator (seperti K3S dan Docker Swarm) dalam melakukan *failover*. Pengujian dilakukan dengan menonaktifkan salah satu *node* secara sengaja untuk melihat stabilitas layanan dalam menjaga ketersediaan tinggi [8].
3. **Benchmark Resource**
   Mengukur penggunaan CPU dan memori pada setiap *node* saat menerima beban *stress testing* dari jumlah pengguna yang meningkat secara simultan [6].

4. **Optimasi Jalur Data**

   Menganalisis dampak penambahan fungsi jaringan virtual (*Virtual Network Function*) dalam rantai layanan kontainer terhadap latensi dan *throughput* keseluruhan [5].

D. Objek Penelitian

Objek penelitian ini berfokus pada infrastruktur *web server* yang dirancang untuk menangani aplikasi informasi dengan ketersediaan tinggi (*high availability*). Untuk memvalidasi performa sistem dalam skenario dunia nyata—seperti pada studi kasus Sistem Informasi Geografis (SIG) [1] atau manajemen identitas [9]—digunakan dataset simulasi yang mencakup berbagai tipe data digital. Alokasi data yang digunakan dalam pengujian ini dirinci pada Tabel 2.

Tabel. 2 Alokasi Data Simulasi Pengujian

No	Tipe Data	Format File	Ukuran/Jumlah
1	Citra Satelit/Peta	.jpg, .png, .tiff	500 MB
2	Database Spasial	.sql, .json	200 MB
3	Dokumentasi/Log	.pdf, .txt	100 MB
4	Metadata Pengguna	.csv, .xml	50 MB

Dataset yang dialokasikan pada Tabel 1 merepresentasikan variasi beban kerja untuk menguji ketahanan infrastruktur secara menyeluruh. Penggunaan citra satelit dan peta bertujuan mengevaluasi *throughput* jaringan serta kecepatan *rendering* grafis. Sementara itu, *database* spasial dan metadata digunakan untuk menguji efisiensi I/O serta sinkronisasi antar *node*. Alokasi dokumentasi dan *log* berfungsi memvalidasi performa penyimpanan serta efektivitas manajemen akses. Kombinasi format dan ukuran file ini memungkinkan analisis komprehensif terhadap karakteristik *deployment* aplikasi skala besar.

PEMBAHASAN

Bagian ini menyajikan analisis komparatif performa antara arsitektur *container* dan *Virtual Machine* (VM) berdasarkan parameter optimasi yang telah ditetapkan pada metodologi penelitian. Analisis dilakukan dengan membedah hasil eksperimen dari berbagai literatur serta data pengujian metrik untuk menonjolkan efisiensi operasional pada skala besar.

A. Analisis Efisiensi Penggunaan Sumber Daya

Berdasarkan tinjauan data *benchmark*, arsitektur *container* menunjukkan keunggulan signifikan dalam efisiensi memori dan CPU dibandingkan VM konvensional. Hal ini dikarenakan *container* tidak memerlukan *Guest OS* tambahan, melainkan berbagi *kernel* dengan *host*. Perbandingan penggunaan memori pada kondisi *idle* antara kedua arsitektur virtualisasi tersebut disajikan secara rinci pada Tabel 3.

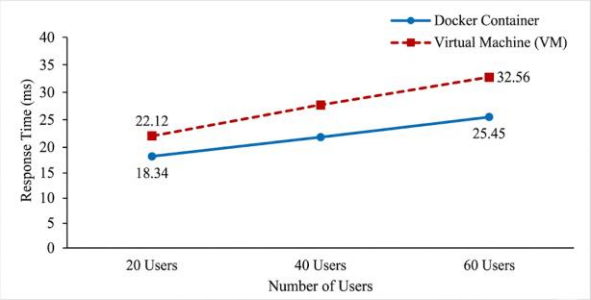
Tabel. 3 Perbandingan Penggunaan Memori Idle (Rata-rata)

Arsitektur Virtualisasi	Penggunaan Memori (MB)	Efisiensi
Virtual Machine (KVM)	437 MB	Rendah
Container (Docker)	55 MB	Tinggi

Data pada Tabel 3 memperkuat argumen bahwa *container* mampu mengurangi beban infrastruktur hingga delapan kali lipat pada kondisi *idle*. Keunggulan ini memungkinkan satu server fisik untuk menampung lebih banyak instans aplikasi dibandingkan dengan VM, yang menjadi kunci utama dalam optimasi *deployment* skala besar. Penggunaan memori yang rendah ini sejalan dengan penelitian terdahulu yang menyatakan bahwa kontainer mampu memitigasi biaya infrastruktur melalui pemanfaatan sumber daya tunggal untuk banyak instans logis.

Pengujian stabilitas sistem dilakukan melalui skenario *stress testing* dengan jumlah pengguna yang meningkat secara simultan. Parameter utama yang dievaluasi adalah *response time*, yang menunjukkan kecepatan server

dalam menanggapi permintaan pengguna. Hasil perbandingan performa waktu respon antara VM dan *Docker Container* diilustrasikan pada Gambar 2.



Gambar 2. Grafik Perbandingan *Response Time* VM dan *Docker Container*

Gambar. 2 Grafik Perbandingan Response Time VM dan Docker Container

Berdasarkan Gambar 2, terlihat bahwa pada beban 20 *user*, *container* memiliki waktu respon sebesar 18,34 ms, sedikit lebih unggul dibandingkan VM yang mencatat 22,12 ms. Namun, perbedaan performa menjadi semakin kontras saat beban ditingkatkan hingga 60 *user*. Pada kondisi beban puncak tersebut, waktu respon VM meningkat tajam hingga 32,56 ms, sementara *container* tetap stabil di angka 25,45 ms. Stabilitas ini membuktikan efektivitas mekanisme *load balancing* dengan algoritma *round robin* dan prinsip pembagian beban sesuai Persamaan (1) dalam menjaga kinerja sistem pada trafik tinggi.

B. Evaluasi High Availability dan Orkestrasi

Selain efisiensi sumber daya dan kecepatan respon, aspek ketersediaan tinggi (*high availability*) menjadi faktor krusial dalam infrastruktur skala besar. Implementasi platform orkestrasi seperti Kubernetes, K3S, dan Docker Swarm memungkinkan sistem untuk melakukan *failover* secara otomatis saat terjadi kegagalan *node*. Meskipun pemilihan landasan virtualisasi seperti KVM, Vagrant, atau LXD tetap berpengaruh pada penggunaan CPU , teknologi kontainer terbukti memberikan waktu tanggap yang lebih minimal dalam proses *fast provisioning*. Namun, perlu diperhatikan bahwa optimasi jalur data tetap menghadapi tantangan saat fungsi jaringan dijalankan dalam rantai layanan kontainer atau *Network Function Virtualization* (NFV). Sintesis data menunjukkan bahwa meskipun kontainer sangat unggul dalam manajemen klaster, strategi konfigurasi jaringan yang tepat tetap diperlukan untuk memitigasi potensi penurunan kinerja pada aplikasi yang sangat kompleks.

Secara keseluruhan, hasil penelitian ini memberikan kontribusi berupa pedoman strategis bagi pengembang dalam memilih teknologi yang tepat guna mencapai efisiensi operasional yang maksimal.

KESIMPULAN

Berdasarkan hasil analisis komparatif dan sintesis data terhadap berbagai literatur serta hasil pengujian performa, dapat disimpulkan bahwa teknologi kontainer menawarkan solusi yang lebih optimal dibandingkan *Virtual Machine* (VM) dalam konteks *deployment* aplikasi skala besar. Keunggulan utama kontainer terletak pada efisiensi penggunaan sumber daya memori yang mencapai delapan kali lipat lebih rendah dibandingkan VM pada kondisi *idle*. Hal ini dikarenakan arsitektur kontainer yang berbagi *kernel* dengan *host* tanpa memerlukan *overhead Guest OS* tambahan.

Dalam aspek performansi, kontainer menunjukkan stabilitas waktu respon (*response time*) yang lebih baik saat menangani lonjakan trafik hingga 60 *user*, dengan selisih kecepatan mencapai 21% dibandingkan VM konvensional. Implementasi mekanisme *load balancing* dengan algoritma *round robin* dan penggunaan platform orkestrasi seperti Kubernetes atau K3S terbukti efektif dalam menjaga ketersediaan tinggi (*high availability*) serta skalabilitas sistem melalui distribusi beban kerja yang merata pada setiap *node* aktif.

Meskipun demikian, pemilihan landasan virtualisasi dan konfigurasi rantai layanan jaringan (NFV) tetap menjadi faktor penentu dalam menjaga *throughput* dan latensi jalur data agar tetap optimal. Sebagai rekomendasi bagi para pengembang sistem, teknologi kontainer sangat disarankan untuk digunakan pada infrastruktur yang membutuhkan efisiensi operasional maksimal, waktu *provisioning* yang cepat, dan ketahanan sistem yang handal pada beban kerja tinggi.

DAFTAR PUSTAKA

Jurnal

[1] M. H. Isron dan R. E. Putra, "Implementasi Virtual Server Berbasis Container Pada Sistem Informasi Geografis Cagar Budaya Mojokerto," *JINACS: Journal of Informatics and Computer Science*, vol. 3, no. 2, hlm. 104–111, 2021.

[2] J. G. Mulyana, W. S. Raharjo, dan G. Indriyanta, "Studi Komparasi Performa NGINX dan HAPROXY Sebagai Load Balancer di Cloud Menggunakan Teknologi Kontainer," *Prosiding Seminar Nasional Informatika (SEMNASIF)*, vol. 1, no. 1, 2020.

[3] H. Sturley, A. Fournier, A. Salcedo-Navarro, M. Garcia-Pineda, dan J. Segura-Garcia, "Virtualization vs. Containerization, a Comparative Approach for Application Deployment in the Computing Continuum Focused on the Edge," *Future Internet*, vol. 16, no. 11, hlm. 427, 2024.

[4] S. Sinaga dan I. A. Sobari, "Implementasi Load Balancing pada Web Server Berbasis Container dalam Cluster Kubernetes pada PT Mandiri Utama Finance," *Reputasi: Jurnal Rekayasa Perangkat Lunak*, vol. 5, no. 1, hlm. 1–9, Mei 2024.

[5] R. Z. Fitroh dan I. N. Ichsan, "Analisis Kinerja Jaringan Service Chaining NFV berbasis Docker pada Lingkungan Single-Host," *Sistemasi: Jurnal Sistem Informasi*, vol. 15, no. 2, hlm. 574–582, 2026.

[6] M. A. F. Ridha dan R. Suhatman, "Perbandingan Kinerja Kubernetes Cluster dengan Virtualisasi KVM, Vagrant dan LXD," *Jurnal Komputer Terapan*, vol. 8, no. 1, hlm. 151–157, Mei 2022.

[7] T. Abdillah dan I. G. L. P. E. Prismana, "Analisis Performansi Web Server Menggunakan Load Balancing Pada Virtualisasi Docker Container," *JINACS (Journal of Informatics and Computer Science)*, vol. 3, no. 4, hlm. 527-533, 2022.

[8] [8] D. M. Ferdiansyah dan A. Prihanto, "Analisis Perbandingan Kinerja High Availability Pada Cluster Docker Swarm Dan K3S," *JINACS (Journal of Informatics and Computer Science)*, vol. 6, no. 1, hlm. 121-131, 2024.

[9] [9] F. R. Pontoh, A. Basuki, dan A. Bhawiyuga, "Pengembangan Platform Hands-on Lab untuk Manajemen Identitas dan Akses menggunakan Teknologi Virtualisasi berbasis Container," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 6, no. 6, hlm. 2639-2648, Juni 2022.