Analisis Perbandingan Performa Web Server Apache Menggunakan Docker Dan Podman

1st Septian Essra Al Rasheed
Direktorat Universitas Telkom Purwokerto
Universitas Telkom Purwokerto
Purwokerto, Indonesia
septianessra@student.telkomuniversity.ac.id

2nd Muhammad Fajar Sidiq, S.T., M.T.
Direktorat Universitas Telkom Purwokerto
Universitas Telkom Purwokerto
Purwokerto, Indonesia
mfsidiq@telkomuniversity.ac.id

3rd Gunawan Wibisono, S.Kom., M.Kom.
Direktorat Universitas Telkom Purwokerto
Universitas Telkom Purwokerto
Purwokerto, Indonesia
gunawanw@telkomuniversity.ac.id

Abstrak — Dalam era modern, penggunaan container menjadi solusi utama dalam pengelolaan aplikasi berbasis server. Docker dan Podman adalah dua teknologi container yang populer dengan kelebihan masing-masing dalam efisiensi dan fleksibilitas pengelolaan aplikasi. Penelitian ini bertujuan untuk membandingkan performa web server Apache yang dijalankan dalam lingkungan Docker dan Podman berdasarkan parameter server response time dan penggunaan CPU. Pengujian dilakukan menggunakan Apache Benchmark dengan berbagai skenario beban kerja, termasuk 500, 600, 750, dan 1000 concurrent requests. Hasil penelitian menunjukkan bahwa Docker memiliki performa yang lebih stabil dalam menangani beban kerja tinggi, sementara Podman menunjukkan efisiensi yang lebih baik dalam penggunaan sumber daya CPU. Analisis ini memberikan wawasan bagi pengembang dan administrator sistem dalam memilih teknologi container yang optimal untuk pengelolaan web server Apache.

Kata kunci— Apache, Container, Docker, Podman, Web Server

I. PENDAHULUAN

Dalam era modern ini, penggunaan container menjadi pilihan utama dalam pengelolaan dan distribusi aplikasi. Container seperti Docker dan Podman menawarkan sebuah lingkungan yang berbeda, terisolasi, dan fleksibel untuk menjalankan aplikasi tanpa mengkhawatirkan konflik dependensi atau masalah lingkungan[1]. Docker memiliki keunggulan berupa kontainer yang ringan dan sudah mencakup semua komponen yang dibutuhkan untuk menjalankan aplikasi, sehingga tidak bergantung pada perangkat lunak yang terpasang di host[2]. Podman sendiri memiliki keunggulan dimana para developer dapat mengelola container tanpa perlu menggunakan daemon[3]. Dengan kelebihan dari masing-masing container para developer dapat meningkatkan produktivitas dalam membuat software yang berkualitas.

Salah satu aspek penting dalam penerapan kontainerisasi adalah performa dari server tempat container dijalankan. Apache, Nginx, Cloudflare Server, LightTPD, Hiawatha, Cherokee, dan Apache Tomcat adalah beberapa contoh aplikasi web server yang umum digunakan di banyak belahan dunia[4]. Menurut data dari W3Techs hingga November 2023, Nginx menjadi web server urutan pertama yang paling banyak digunakan yaitu sebesar 34,2% disusul urutan kedua ada Apache yaitu sebesar 30,8% dan di urutan ketiga ada Cloudflare Server yaitu sebesar 21,2%[5][6].

Pada penelitian sebelumnya dalam menguji web server seperti penelitian yang berjudul "Analisis Perbandingan Performa Apache Web Server Dan Nginx Menggunakan Apache Jmeter". Penelitian ini menyatakan bahwa Nginx lebih efisien dari segi response time dan lebih baik dari sisi throughputnya[7]. Namun penelitian ini tidak menyebutkan secara detail jenis container yang digunakan. Lalu, terdapat penelitian yang membandingkan antara Docker dan Podman seperti penelitian yang berjudul "Analisis Perbandingan Kinerja Layanan Container As A Service (CAAS) Studi Kasus : Docker dan Podman". Hasil penelitian ini menunjukkan bahwa Docker memiliki performa yang lebih baik dibandingkan Podman ketika menjalankan 1-6 aplikasi. Setelah menerapkan tindakan penanggulangan terhadap serangan container, Podman menunjukkan keunggulan dalam performa dan penggunaan sumber daya dibandingkan Docker[1]. Namun penelitian ini tidak menyebutkan secara detail web server yang digunakan.

Berdasarkan latar belakang yang dijabarkan di atas, maka dilakukanlah penelitian yang nantinya menghasilkan container mana yang terbaik dengan menggunakan Apache sebagai web server-nya dan Docker dengan Podman sebagai container-nya. Penelitian ini menggunakan tools dari Apache Benchmark untuk mengukur server response time dan throughput yang bisa dilakukan oleh web server

II. KAJIAN TEORI

A. VirtualBox

VirtualBox adalah software open-source yang dikembangkan oleh Oracle yang menggunakan teknologi virtualisasi untuk menjalankan sistem operasi guest beserta pustaka dan aplikasinya pada computer dengan sistem operasi y dan hypervisor. VirtualBox memungkinkan pengguna untuk memvirtualisasikan sistem operasi yang berbeda di atas sistem operasi utama[8].

B. Web Server

Web server adalah sebuah software yang berfungsi untuk melayani permintaan data dari klien melalui protokol HTTP atau HTTPS. Permintaan ini biasanya dilakukan melalui aplikasi browser web. Setelah menerima permintaan, server akan mengirimkan data dalam bentuk halaman web, yang umumnya berformat HTML. Halaman web tersebut dapat mencakup berbagai jenis file, seperti teks, gambar, video, dan lainnya[9]. Fungsi utama web server adalah menangani permintaan dari klien berdasarkan protokol yang telah ditentukan, kemudian mengirimkan kembali data yang

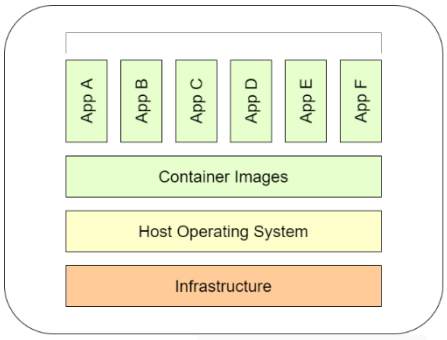
diminta dalam bentuk halaman web beserta kontennya[10].

C. Apache

Sebagai server web yang berjalan di sistem berbasis Unix, Apache awalnya dikembangkan dari kode NCSA HTTPD 1.3 dan kemudian diprogram ulang hingga menjadi salah satu server web yang paling banyak digunakan saat ini. Apache menawarkan berbagai fitur unggulan, termasuk kinerja yang optimal, fungsionalitas yang luas, efisiensi, serta kecepatan yang baik. Selain itu, Apache bersifat open source[11].

D. Container

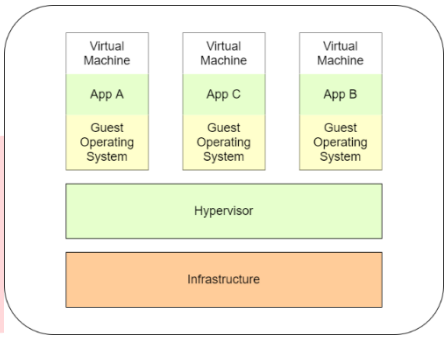
Container merupakan abstraksi pada lapisan aplikasi yang menggabungkan kode beserta dependensinya dalam satu paket. Beberapa container dapat dijalankan secara bersamaan pada satu mesin dan berbagi kernel OS dengan container lainnya, tetapi tetap beroperasi sebagai proses yang terisolasi di ruang pengguna. Dibandingkan dengan virtual machine, container memerlukan ruang yang lebih kecil (biasanya hanya berukuran puluhan MB), mampu menjalankan lebih banyak aplikasi, serta mengurangi kebutuhan akan virtual machine dan sistem operasi[12].



GAMBAR 1
Skema Container[12]

Gambar 1 menggambarkan arsitektur berbasis kontainer yang terdiri dari beberapa lapisan. Lapisan paling bawah adalah Infrastructure (Infrastruktur), yaitu perangkat keras fisik atau mesin virtual tempat sistem berjalan, seperti data center lokal, cloud provider (misalnya AWS, Google Cloud, atau Azure), atau kombinasi keduanya. Di atasnya terdapat Host Operating System (Sistem Operasi Host), yaitu sistem operasi yang diinstal pada infrastruktur tersebut sebagai dasar untuk menjalankan teknologi kontainer seperti Docker atau container runtime lainnya. Selanjutnya, terdapat lapisan Container Images (Citra Kontainer) yang berisi paket-paket aplikasi dengan semua dependensi yang diperlukan untuk menjalankan aplikasi. Lapisan paling atas adalah Apps (Aplikasi), yang menunjukkan aplikasi-aplikasi yang berjalan dalam lingkungan kontainer. Setiap aplikasi berada dalam kontainer terpisah, memberikan isolasi sehingga aplikasi dapat berjalan secara independen tanpa saling mengganggu. Pendekatan ini memastikan stabilitas aplikasi di berbagai lingkungan, dengan isolasi yang baik dan tanpa konflik antar aplikasi.

Virtual machine merupakan representasi virtual dari perangkat keras fisik yang memungkinkan satu server berfungsi seolah-olah terdiri dari beberapa server. Dengan bantuan hypervisor, beberapa VM dapat dijalankan secara bersamaan pada satu perangkat. Setiap VM memiliki sistem operasi, aplikasi, biner, dan library secara lengkap, yang dapat memakan ruang penyimpanan hingga puluhan gigabyte. Selain itu, proses booting VM cenderung memakan waktu lebih lama[12].



GAMBAR 2
Skema Virtual Machine[12]

Gambar 2 menunjukkan skema arsitektur berbasis mesin virtual (virtual machine). Pada lapisan paling bawah terdapat Infrastructure (Infrastruktur), yaitu perangkat keras fisik atau sumber daya komputasi yang menyediakan fondasi untuk menjalankan mesin virtual. Di atasnya terdapat Hypervisor, yang merupakan perangkat lunak atau lapisan virtualisasi yang memungkinkan beberapa mesin virtual berjalan di atas satu infrastruktur fisik. Hypervisor bertanggung jawab untuk mengalokasikan sumber daya perangkat keras ke masing-masing mesin virtual.

Setiap Virtual Machine (Mesin Virtual) terdiri dari dua bagian utama: Guest Operating System (Sistem Operasi Tamu) dan aplikasi. Sistem operasi tamu adalah sistem operasi penuh yang berjalan di dalam mesin virtual dan berfungsi untuk mendukung aplikasi yang berjalan di atasnya, seperti App A, App B, atau App C. Setiap mesin virtual memiliki isolasi penuh, sehingga aplikasi dalam satu mesin virtual tidak memengaruhi aplikasi di mesin virtual lainnya. Namun, karena setiap mesin virtual memiliki salinan lengkap sistem operasi, metode ini cenderung lebih berat dibandingkan arsitektur berbasis kontainer.

E. Docker

Docker, sebuah platform aplikasi yang memanfaatkan teknologi kontainer, pertama kali diperkenalkan sebagai perangkat lunak sumber terbuka pada tahun 2013. Docker berfungsi sebagai platform kontainer yang berjalan di atas perangkat keras dan sistem operasi host. Di dalam sistem operasi host tersebut, Docker engine merupakan bagian terpadu dari Docker, bertindak sebagai lingkungan tempat Docker container dapat dijalankan[13].

Docker Container merupakan perkembangan dari konsep kontainer LXC, dengan perbedaan utamanya terletak pada pendekatan virtualisasi yang lebih spesifik terhadap aplikasi. Docker membuat template dari

berbagai sistem operasi (OS) dengan cara yang sederhana untuk mendukung eksekusi layanan tertentu. Hal ini menjadikan Docker sebagai solusi virtualisasi yang lebih efisien, karena tidak mengisolasi keseluruhan sistem operasi seperti yang dilakukan oleh LXC[14].

F. Podman

Podman adalah mesin container yang dapat menggantikan peran Docker. Pengembangan mesin ini dilakukan oleh Red Hat (kini dimiliki oleh IBM) dengan tujuan untuk menggerakkannya secara bertahap[15][16]. Podman memanfaatkan perangkat lunak sumber terbuka yang berguna untuk mengelola container secara lokal. Dengan Podman, Anda dapat mencari, menjalankan, membangun, atau menyebarkan container[17].

G. Ubuntu

Ubuntu adalah sistem operasi yang dikembangkan dari distribusi Linux, atau yang lebih dikenal dengan istilah "distro", yang berbasis Debian dan dilengkapi dengan berbagai fitur lengkap. Sifat opensource dari sistem operasi Ubuntu memungkinkan penggunaannya secara gratis oleh siapa pun[18].

H. Apache Benchmark

Apache Benchmarking Tool merupakan perangkat yang dikembangkan oleh Apache Organization untuk mengukur kinerja server web yang menggunakan protokol Hypertext Transfer Protocol (HTTP). Alat ini mampu menghitung jumlah permintaan per detik yang dapat diproses oleh server web yang diuji[19].

I. Grafana

Grafana adalah platform open-source untuk pemantauan dan observabilitas. Platform ini menyediakan kemampuan visualisasi yang kuat dan terintegrasi mulus dengan prometheus. Grafana memiliki fitur seperti visualisasi data, pembuatan dashboard, integrasi dengan Promtheus, dll. Keunggulan Grafana adalah dashboard yang dapat disesuaikan, pilihan visual yang banyak, antarmuka yang ramah pengguna dan masih banyak keunggulan lainnya[20].

J. Prometheus

Prometheus adalah perangkat lunak open source yang digunakan untuk pemantauan dan pemberian peringatan. Salah satu keunggulan utama Prometheus dibandingkan perangkat lunak pemantauan lainnya adalah kemampuannya menyediakan berbagai metrik yang dibutuhkan untuk memantau sistem. Fleksibilitas Prometheus dalam penyesuaian juga membuatnya sangat berguna ketika digabungkan dengan perangkat lunak lain, seperti saat digunakan bersama Grafana untuk visualisasi data[21].

K. Node Exporter

Node Exporter adalah alat dalam ekosistem Prometheus yang berfungsi untuk mengumpulkan berbagai metrik guna memantau kinerja dan kondisi sistem operasi. Alat ini mengumpulkan data tentang penggunaan CPU, memori, I/O disk, jaringan, dan metrik lain terkait dengan performa infrastruktur[22].

L. Time Taken for Tests

Pengujian ini bertujuan untuk menghitung durasi waktu yang diperlukan sejak Apache Benchmark pertama kali terhubung ke server hingga menerima respons terakhir dalam satu percobaan[23].

M. Requests per Second

Pengujian ini bertujuan untuk mengukur jumlah permintaan layanan yang dapat diproses per detik yang dikirimkan oleh klien ke server web[23].

N. Average Time per Request

Pengujian ini bertujuan untuk mengukur rata-rata waktu yang dibutuhkan dalam memproses sejumlah permintaan secara bersamaan[23].

O. Average Time per Request (concurrent)

Pengujian ini ditujukkan untuk menentukan nilai rata-rata permintaan secara bersama-sama (concurrent)[24].

P. Transfer Rate

Pengujian ini ditujukkan untuk menghitung nilai transaksi per second yang dihasilkan dari memenuhi permintaan user[23].

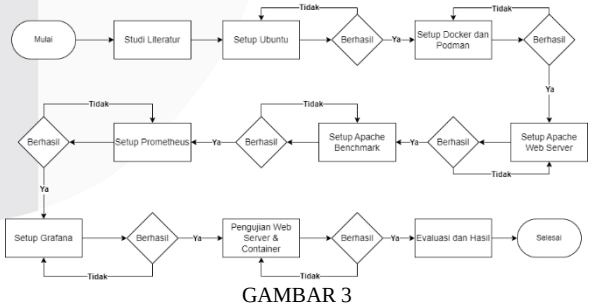
III. METODE

A. Subyek dan Obyek Penelitian

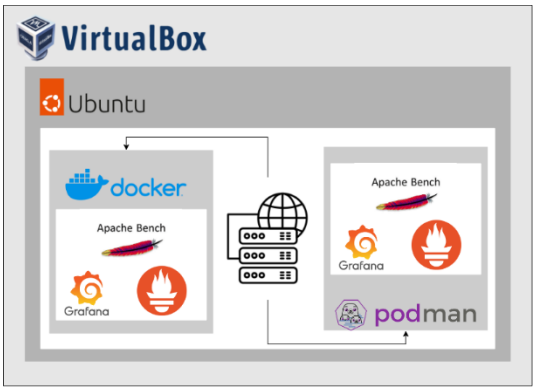
Berdasarkan data dan informasi yang sudah diperoleh subjek penelitian ini yaitu berupa container Docker dan Podman.

Objek Penelitian ini yaitu berfokus pada perbandingan performa dari web server Apache.

B. Diagram Alir Penelitian



GAMBAR 3
Diagram Alir Penelitian



GAMBAR 4
Skema Pengoperasian Tools

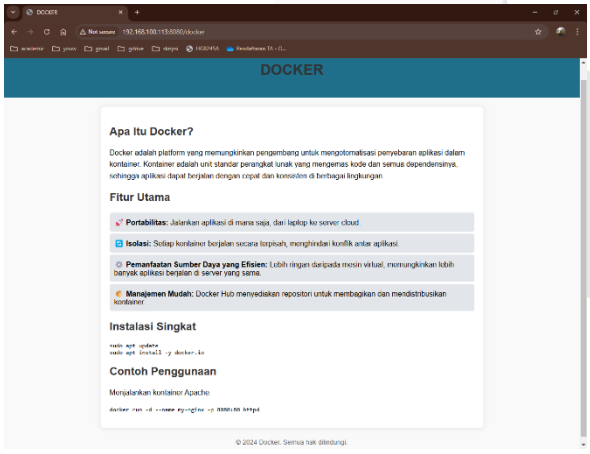
Gambar 4 adalah skema dari alat-alat yang digunakan. VirtualBox sebagai yang didalamnya sistem operasi Ubuntu. Web server Apache yang akan beroperasi di dalam container Docker dan Podman.

IV. HASIL DAN PEMBAHASAN

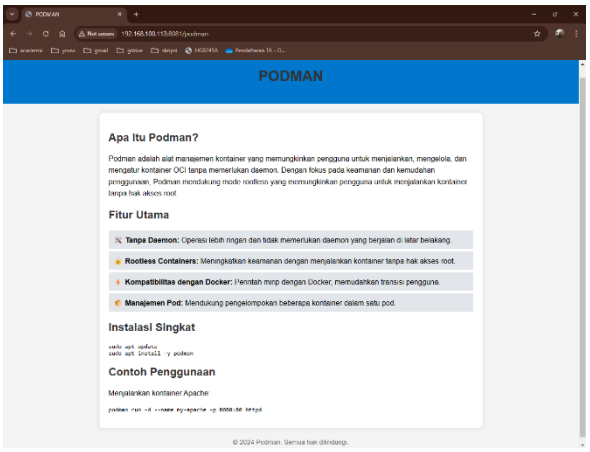
Hasil pengujian berulang sebanyak 5 kali per concurrent dan per container dengan nilai request 5000 yang telah ditentukan dibahas dalam bab ini. Hasil yang ditujukkan adalah rata-rata yang telah diolah, sehingga dapat menarik observasi dan kesimpulan dengan membandingkan data individu dengan data lainnya. Hasil concurrent 500, 600, 750, dan 1000 diperoleh dengan memeriksa dari pengujian Apache Benchmark dan utilisasi CPU yang digunakan untuk setiap container.

A. Hasil dan Perancangan Sistem

Penulis menggunakan port yang ditentukan untuk memeriksa apakah web server berhasil diinstal dan dapat diakses melalui web browser menggunakan Docker port 8080 dan Podman port 8081. Gambar 5 dan 6 merupakan tampilan website statis dari Apache yang sudah dibuat.



GAMBAR 5
Website Statis Docker



GAMBAR 6
Website Statis Podman

Service yang telah dibuat dalam Prometheus dan disambungkan dalam sistem Grafana buat menerima data monitoring penggunaan CPU yang dipakai selama pengujian berlangsung. Berikut gambar 7 yang merupakan tampilan monitoring Grafana.

GAMBAR 7
Grafik Monitoring CPU

Gambar 7 memaparkan visualisasi monitoring CPU container Docker dan Podman yang disambungkan menggunakan Prometheus. Menggunakan data CPU Idle atau yang tidak digunakan memperlihatkan grafik yang lebih stabil daripada data CPU Usage atau yang digunakan.

B. Pengujian Server Response Time

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan oleh server untuk merespon user request.

```
ab -n 5000 -c 500 http://192.168.1.13:8080/docker
```

GAMBAR 8
Sintaks Pengujian Apache Benchmark

Gambar 8 merupakan instruksi dari Apache Benchmark untuk melakukan pengujian terhadap server web dengan parameter yang ditentukan.

a. Time Taken for Tests

Pengujian ini dilakukan dengan jumlah 5000 permintaan dan berulang sebanyak 5 kali pengujian per parameter. Berikut tabel data dari yang telah diuji:

TABEL 1
Data Time Taken for Tests

Concurrent	Docker (s)	Podman (s)
500	3,929	2,796
600	3,883	3,308
750	4,197	3,179
1000	4,886	4,924

Tabel 1 menunjukkan waktu yang dibutuhkan oleh Docker dan Podman untuk menjalankan sejumlah proses secara concurrent. Data dalam tabel mencerminkan perbandingan performa antara kedua container dalam menangani jumlah permintaan yang berbeda. Berikut poin-poin dari tabel diatas:

- Pada nilai concurrent 500, Podman lebih cepat dengan waktu eksekusi 2,796 detik dibandingkan Docker yang memerlukan 3,929 detik.
- Pada nilai concurrent 600 dan 750, performa Docker sedikit meningkat pada 600 konkurensi, tetapi kemudian meningkat lebih tinggi dibandingkan Podman pada 750 konkurensi.
- Pada nilai concurrent 1000, waktu eksekusi Docker dan Podman hampir sama, dengan Docker membutuhkan 4,886 detik dan Podman 4,924 detik.

Secara umum, Podman menunjukkan keunggulan dalam eksekusi dengan jumlah konkurensi yang lebih rendah, sedangkan Docker mulai mendekati atau melampaui performa Podman pada tingkat konkurensi yang lebih tinggi.

b. Requests per Second

Uji ini dilaksanakan dengan 5000 permintaan dan diulang sebanyak 5 kali untuk setiap parameter. Berikut tabel data dari requests per second yang telah diuji:

TABEL 2
Data Requests per Second

Concurrent	Docker	Podman
500	1288,18	1861.15
600	1292,14	1800.27
750	1206,40	1466.91
1000	1074,30	1435.19

Tabel 2 menunjukkan jumlah permintaan per detik (Requests per Second) yang dapat ditangani oleh Docker dan Podman pada berbagai tingkat konkurensi. Data ini memberikan wawasan tentang performa kedua platform dalam menangani beban kerja yang meningkat. Berikut poin-poin dari tabel diatas:

- Pada nilai concurrent 500, Podman menangani lebih banyak permintaan per detik yaitu 1791,17 permintaan dibandingkan Docker yaitu 1288,18 permintaan.
- Pada nilai concurrent 600, performa Docker sedikit meningkat menjadi 1292,14 permintaan, sementara Podman mengalami sedikit penurunan menjadi 1552,69 permintaan.
- Pada nilai concurrent 750, Docker mengalami penurunan ke 1206,40 permintaan, sementara Podman kembali naik ke 1589,94 permintaan.
- Pada nilai concurrent 1000, baik Docker maupun Podman mengalami penurunan, dengan masing-masing menangani 1074,30 dan 1130,07 permintaan per detik.

Secara keseluruhan, Podman cenderung lebih unggul dalam menangani lebih banyak permintaan per detik dibandingkan Docker di semua tingkat konkurensi.

c. Average Time per Request

Pengujian ini dilakukan dengan 5000 permintaan dan diulang lima kali untuk setiap parameter. Berikut tabel data dari average time per request yang telah diuji:

TABEL 3
Data Average Time per Request

Concurrent	Docker (ms)	Podman (ms)
500	392,870	279,580
600	466,000	396,945
750	629,562	476,941
1000	977,166	984,839

Tabel 3 menunjukkan waktu rata-rata per permintaan (Average Time per Request) yang diperlukan oleh Docker dan Podman pada berbagai tingkat concurrent. Data ini memberikan wawasan mengenai efisiensi kedua platform dalam menangani peningkatan jumlah concurrent. Berikut poin-poin dari tabel di atas:

- Pada nilai concurrent 500, Podman memiliki waktu eksekusi lebih cepat, yaitu 279,580 ms dibandingkan Docker yang membutuhkan 392,870 ms.
- Pada nilai concurrent 600, waktu eksekusi Docker meningkat menjadi 466,000 ms, sementara Podman tetap lebih cepat dengan 396,945 ms.
- Pada nilai concurrent 750, Docker mengalami peningkatan waktu menjadi 629,562 ms, sementara Podman tetap lebih unggul dengan waktu 476,941 ms.

- Pada nilai concurrent 1000, kedua platform menunjukkan waktu eksekusi yang hampir sama, dengan Docker di 977,166 ms dan Podman di 984,839 ms.

Secara keseluruhan, Podman cenderung memiliki waktu eksekusi lebih cepat dibandingkan Docker pada semua tingkat concurrent, kecuali pada 1000 concurrent, di mana kinerja keduanya hampir setara.

d. Average Time per Request (concurrent)

Uji ini dilakukan dengan 5000 permintaan yang diulang lima kali untuk setiap parameternya. Berikut tabel data dari average time per request (concurrent) yang telah diuji:

TABEL 4
Data Average Time per Request (concurrent)

Concurrent	Docker (ms)	Podman (ms)
500	0,786	0,559
600	0,777	0,662
750	0,840	0,636
1000	0,977	0,985

Tabel 4 menyajikan data rata-rata waktu pemrosesan per permintaan (Average Time per Request) untuk Docker dan Podman pada berbagai tingkat konkurensi. Informasi ini memberikan gambaran tentang bagaimana kedua platform menangani peningkatan jumlah permintaan secara simultan. Berikut adalah beberapa poin utama yang dapat diambil dari tabel:

- Pada tingkat concurrent 500, Podman menunjukkan performa lebih baik dengan waktu pemrosesan 0,559 ms, lebih cepat dibandingkan Docker yang membutuhkan 0,786 ms.
- Pada tingkat concurrent 600, waktu eksekusi Docker sedikit menurun menjadi 0,777 ms, sementara Podman meningkat menjadi 0,662 ms, tetapi tetap lebih unggul dibandingkan Docker.
- Saat jumlah permintaan meningkat ke 750, Docker mengalami peningkatan waktu pemrosesan menjadi 0,840 ms, sedangkan Podman tetap menunjukkan efisiensi yang lebih baik dengan waktu 0,636 ms.
- Pada tingkat konkurensi 1000, selisih antara kedua platform semakin mengecil, dengan Docker mencatatkan waktu 0,977 ms dan Podman 0,985 ms, yang menunjukkan bahwa kinerja keduanya hampir setara pada beban kerja yang lebih tinggi.

Secara keseluruhan, Podman cenderung lebih cepat dalam menangani permintaan dibandingkan Docker, terutama pada tingkat konkurensi yang lebih rendah hingga menengah. Namun, pada tingkat permintaan yang sangat tinggi, perbedaan kinerja antara keduanya menjadi semakin kecil.

e. Transfer Rate

Pengujian ini dilaksanakan dengan 5000 permintaan yang diulang lima kali untuk tiap parameternya. Berikut tabel data dari transfer rate yang telah diuji:

TABEL 5
Data Transfer Rate

Concurrent	Docker (Kb/s)	Podman (Kb/s)
500	3517,35	4843,51
600	3528,15	4198,63
750	3294,03	4299,37
1000	2933,34	3055,82

Tabel 5 menyajikan data Transfer Rate dalam satuan Kb/s untuk Docker dan Podman pada berbagai tingkat konkurensi. Informasi ini memberikan wawasan mengenai seberapa cepat kedua platform mentransfer data saat menangani peningkatan jumlah permintaan secara bersamaan. Berikut adalah beberapa poin utama dari tabel tersebut:

- Pada tingkat concurrent 500, Podman menunjukkan kinerja lebih unggul dengan kecepatan transfer 4843,51 Kb/s, lebih tinggi dibandingkan Docker yang mencapai 3517,35 Kb/s.
- Saat tingkat concurrent meningkat ke 600, performa Docker sedikit naik menjadi 3528,15 Kb/s, tetapi masih tertinggal dari Podman yang mencatatkan 4198,63 Kb/s.
- Pada concurrent 750, Docker mengalami sedikit penurunan kecepatan menjadi 3294,03 Kb/s, sedangkan Podman tetap mempertahankan keunggulannya dengan 4299,37 Kb/s.
- Pada tingkat concurrent 1000, kedua platform mengalami sedikit perlambatan, dengan Docker turun menjadi 2933,34 Kb/s, sementara Podman masih lebih cepat dengan 3055,82 Kb/s.

Secara keseluruhan, Podman memiliki kecepatan transfer data yang lebih baik dibandingkan Docker pada semua tingkat konkurensi. Meskipun keduanya mengalami sedikit penurunan performa saat jumlah permintaan meningkat, Podman tetap menunjukkan efisiensi yang lebih tinggi dalam proses transfer data.

C. Penggunaan CPU

Proses evaluasi ini dilakukan untuk mengukur seberapa besar CPU yang digunakan oleh sistem. Dikarenakan pemrosesan ini dilakukan secara

virtualisasi dan mempunyai batas hanya beroperasi pada RAM 5098 MB dan 2 processor.

TABEL 6
Prngujian CPU

Concurrent	Docker (*node*)	Podman (*node*)
500	1,67	1,88
600	5,15	1,58
750	3,85	1,35
1000	3,49	2,07

Tabel 6 menyajikan informasi mengenai rata-rata dari konsumsi CPU oleh Docker dan Podman sebanyak 5 percobaan pada berbagai tingkat konkurensi. Data ini memberikan wawasan tentang seberapa efisien kedua platform dalam mengelola beban kerja yang meningkat.

- Pada tingkat concurrent 500, Docker mengonsumsi 1,67 node, sedangkan Podman sedikit lebih tinggi dengan 1,88 node.
- Pada concurrent 600, penggunaan CPU Docker melonjak signifikan hingga 5,15 node, sementara Podman justru mengalami penurunan menjadi 1,58 node.
- Pada concurrent 750, konsumsi CPU Docker berkurang menjadi 3,85 node, sementara Podman juga menurun menjadi 1,35 node.
- Pada concurrent 1000, Docker menggunakan 3,49 node, sementara Podman mengalami peningkatan menjadi 2,07 node.

Secara keseluruhan, penggunaan CPU Docker cenderung lebih fluktuatif dibandingkan Podman. Sebaliknya, Podman menunjukkan pola yang lebih stabil dan umumnya lebih hemat dalam penggunaan CPU pada tingkat konkurensi yang lebih tinggi.

D. Hasil Pengujian

TABEL 7
Hasil Pengujian

Nama Pengujian	Docker	Podman
Time Taken for Tests	0	1
Requests per Second	0	1
Average Time per Request	0	1
Transfer Rate	0	1
CPU Usage	0	1
Jumlah	0	5

Tabel 4.7 Hasil Pengujian menyajikan perbandingan antara Docker dan Podman berdasarkan lima parameter pengujian, yaitu Time Taken for Tests, Requests per Second, Average Time per Request, Average Time per Request (concurrent), Transfer Rate, dan CPU Usage. Tabel 4.7 menunjukkan bahwa Podman memiliki performa lebih baik dalam setiap aspek yang diuji. Hal ini mengindikasikan bahwa dalam seluruh parameter yang diuji, Podman menunjukkan keunggulan dibandingkan Docker. Dengan demikian, berdasarkan hasil pengujian ini, Podman lebih unggul dalam hal kinerja dibandingkan Docker.

V. KESIMPULAN

Berdasarkan rumusan masalah pada penelitian Analisis Perbandigan Performa Web Server Apache Menggunakan Docker dan Podman, penelitian ini menghasilkan kesimpulan sebagai berikut:

Hasil pengujian menunjukkan bahwa Podman memiliki performa yang lebih baik dibandingkan Docker dalam hal waktu respons server (server response time), jumlah request per detik (requests per second), rata-rata waktu per request (average time per request), concurrent dan tingkat transfer data (transfer rate). Podman secara konsisten mencatat waktu eksekusi yang lebih cepat dan lebih efisien dalam menangani permintaan dalam jumlah besar.

Pengujian penggunaan CPU menunjukkan bahwa Podman memiliki pola penggunaan CPU yang lebih stabil dibandingkan Docker. Docker mengalami fluktuasi yang lebih signifikan, terutama pada tingkat konkurensi yang lebih tinggi. Sebaliknya, Podman menunjukkan efisiensi dalam pengelolaan sumber daya CPU, yang mengindikasikan bahwa Podman lebih hemat dalam penggunaan sumber daya pada skenario uji coba ini.

Berdasarkan hasil ini, Podman dapat direkomendasikan sebagai alternatif yang lebih efisien dibandingkan Docker dalam menjalankan Apache Web Server, terutama dalam skenario yang membutuhkan respons cepat dan efisiensi penggunaan CPU. Kesimpulan ini sejalan dengan tujuan penelitian, yaitu mengidentifikasi platform container yang lebih unggul dalam menjalankan web server Apache berdasarkan pengujian response time dan penggunaan CPU.

REFERENSI

[1] C. Mukmin, T. Naraloka, and Q. H. Andriyanto, "Analisis Perbandingan Kinerja Layanan Container As A Service (CAAS) Studi Kasus : Docker dan Podman," *Kumpul. J. Ilmu Komput.*, vol. 08, no. 2, pp. 152–161, 2021.

[2] Docker, "The Docker platform," Docker Docs. Accessed: Nov. 29, 2023. [Online]. Available: https://docs.docker.com/get-started/overview/

[3] Podman, "What is Podman?," Podman. Accessed: Nov. 29, 2023. [Online]. Available: https://docs.podman.io/en/latest/

[4] A. Y. Chandra, "Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request," *J. Sist. dan Inform.*, vol. 14, no. 1, pp. 48–56, 2019, doi: 10.30864/jsi.v14i1.248.

[5] Stackscale, "Which are the most used web servers?," Stackscale. Accessed: Nov. 11, 2023. [Online]. Available: https://www.stackscale.com/blog/top-web-servers/

[6] W3Techs, "Historical quarterly trends in the usage statistics of web servers," W3Techs. Accessed: Nov.

11, 2023. [Online]. Available: https://w3techs.com/technologies/history_overview/web_server/ms/q

[7] A. R. Busran, "Analisis Perbandingan Performa Apache Web Server Dan Nginx Menggunakan Apache Jmeter," *J. Teknoif ITP*, vol. 8, no. 2, pp. 87–92, 2020, [Online]. Available: https://randi.itp.ac.id/index.php/teknoif/article/view/48

[8] I. P. A. E. Pratama and I. M. S. Raharja, "Node.js Performance Benchmarking and Analysis at Virtualbox, Docker, and Podman Environment Using Node-Bench Method," *JOIV Int. J. Informatics Vis.*, vol. 7, no. 4, pp. 2240–2246, Dec. 2023, doi: 10.30630/joiv.7.4.1762.

[9] M. Hasibuan and A. M. Elhanafi, "Penetration Testing Sistem Jaringan Komputer Menggunakan Kali Linux untuk Mengetahui Kerentanan Keamanan Server dengan Metode Black Box," *sudo J. Tek. Inform.*, vol. 1, no. 4, pp. 171–177, 2022, doi: 10.56211/sudo.v1i4.160.

[10] S. Dwiyatno, E. Rachmat, A. P. Sari, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," *PROSISKO J. Pengemb. Ris. dan Obs. Sist. Komput.*, vol. 7, no. 2, pp. 165–175, 2020, doi: 10.30656/prosisko.v7i2.2520.

[11] H. Sandra, "Performance Analysis of Openlitespeed and Apache Web Servers in Serving Client Requests," *Knowbase Int. J. Knowl. Database*, vol. 2, no. 2, p. 114, 2022, doi: 10.30983/ijokid.v2i2.5306.

[12] Docker, "What is a Container?," Docker. Accessed: Nov. 21, 2023. [Online]. Available: https://www.docker.com/resources/what-container/

[13] Y. Cahyaningrum and I. R. Widiasari, "Analisis Performa Container Berplatform Docker atas SeranganMalicious Software (Malware)," *J. Buana Inform.*, vol. 11, no. 1, p. 47, 2020, doi: 10.24002/jbi.v11i1.3279.

[14] M. V. Harris, I. R. Munadi, and E. F. Dewanta, "Perbandingan Kinerja Linux Container dan Docker Container untuk Skema Migrasi Layanan Comparison of Linux Container and Docker Container Performance for Service Migration Scheme," vol. 7, no. 3, pp. 9152–9165, 2020.

[15] A. N. I. Nadyanto, M. A. W., Nizar, M. V. A., & Purwanto, "Membangun Mail Server Berbasis Linux Menggunakan Postfix," *Ilm. Tek. Inform. dan Komun.*, vol. 1, no. 1, pp. 1–8, 2021.

[16] T. T. Taufik, A., Sudarsono, G., Sudaryana, I. K., & Muryono, *Pengantar teknologi informasi Sutarman*. 2022. [Online]. Available: http://badanpenerbit.org/index.php/dpipress/article/view/18

[17] A. Andriansyah, R. A. F., Nurhayati, "Pelatihan Red Hat OpenShift Development Introduction to Containers With Podman Untuk Mahasiswa Universitas Mikroskil," *PENGABDIANMASYARAKAT*, vol. 2, no. 1, pp. 53–57, 2023.

[18] A. Faradilla, "Apa Itu Ubuntu? Ini Penjelasan, Kelebihan, & Kekurangannya," Hostinger. Accessed: Nov. 14, 2023. [Online]. Available: https://www.hostinger.co.id/tutorial/ubuntu-adalah

[19] I. K. S. Satwika and K. N. Semadi, "Perbandingan Performansi Web Server Apache Dan Nginx Dengan Menggunakan Ipv6," *SCAN - J. Teknol. Inf. dan Komun.*, vol. 15, no. 1, pp. 10–15, 2020, doi: 10.33005/scan.v15i1.1847.

[20] Y. Jani, "Unified Monitoring for Microservices: Implementing Prometheus and Grafana for Scalable Solutions," *J. Artif. Intell. Mach. Learn. Data Sci.*, vol. 2, no. 1, pp. 848–852, Mar. 2024, doi: 10.51219/JAIMLD/yash-jani/206.

[21] Ramadoni, Mahmud Zunus Amirudin, Rifki Fahmi, Ema Utami, and Muhammad Syukri Mustafa, "Evaluasi Penggunaan Prometheus dan Grafana Untuk Monitoring Database Mongodb," *J. Inform. Polinema*, vol. 7, no. 2, pp. 43–50, 2021, doi: 10.33795/jip.v7i2.530.

[22] B. Rasyidi and F. Pratama, "Server Monitoring System at PT . XYZ Media Indonesia Based on Grafana and Prometheus Sistem Monitoring Server di PT . XYZ Media Indonesia Berbasis Grafana dan Prometheus," *Indones. J. Mach. Learn. Comput. Sci.*, vol. 4, no. October, pp. 1456–1465, 2024.

[23] Y. N. Khamidah, B. Arifwidodo, and M. A. Amanaf, "Pengaruh Client Request Pada Web Server Apache dan Nginx Dengan Ipv6 Menggunakan Apache Benchmark," *Techno (Jurnal Fak. Tek. Univ. Muhammadiyah Purwokerto)*, vol. 25, no. 1, p. 1, 2024, doi: 10.30595/techno.v25i1.14056.

[24] H. N. Amriansyah, "Analisis Perbandingan Performa Web Server (Nginx) Menggunakan Docker Dan Podman" Institut Teknologi Telkom Purwokerto, 2023.