Jurnal SimanteC
Vol. 4, No. 3 Juni 2015 ISSN 2088-2130

CONTAINER DAN DOCKER: TEKNIK VIRTUALISASI DALAM PENGELOLAAN BANYAK APLIKASI WEB

Firmansyah Adiputra
Program Studi Manajemen Informatika, Fakultas Teknik, Universitas Trunojoyo Madura
Jl. Raya Telang, PO BOX 2 Kamal, Bangkalan
E-mail : frm.adiputra@trunojoyo.ac.id

ABSTRAK

Sistem web hosting modern, di dalam setiap servernya, mengelola banyak aplikasi web. Teknologi *virtual machine* dimanfaatkan untuk menyelesaikan masalah heterogenitas (perbedaan versi *library* atau *tool* dari beberapa aplikasi web). Peningkatan jumlah aplikasi web yang harus dihosting harus diikuti dengan peningkatan kualitas ataupun kuantitas sumber daya, terlebih saat hadirnya kebutuhan high availability dari layanan web tersebut. Teknik kontainerisasi (virtualisasi berbasis *container*) hadir sebagai solusi dan menjadi trend saat ini. Docker adalah salah satu software yang mengadopsi teknik kontainerisasi dan semakin banyak diterapkan di dalam lingkungan web hosting. Tulisan ini mencoba untuk melakukan kajian literatur terhadap teknologi virtualisasi di atas, baik virtual machine maupun container dan kemudian merangkum perbandingannya. Arsitektur container di dalam Docker merupakan fokus dari tulisan ini, termasuk perkembangan dan keunggulan dari Docker yang sudah diteliti dan diimplementasikan dalam dua tahun terakhir. Review ini dirasakan sangat penting bagi pengembang dan *system administrator* yang mengelola banyak aplikasi web, terutama aplikasi-aplikasi yang memiliki heterogenitas tinggi dan berjalan di dalam satu mesin server fisik yang sama.

Kata kunci: mesin virtual, container, web hosting, pengelolaan aplikasi web, docker.

ABSTRACT

Modern web hosting system manages many web applications in each of its servers. Virtual machine technology have been used as a solution to software heterogeneity. An increased number of hosted web application must be balanced by enhancement in resources quality or quantity, especially if high availability must be met. Container based virtualization is now becoming a trend. Docker is a platform that implements container based virtualization, and many web hosting services are starting use it. This paper reviews and compares the two virtualization technology, virtual machine and container based virtualization. This paper focused on the architecture of Docker's container, the advantages of Docker, and the development of Docker in the last two years. The review will be useful for software developers and system administrators who maintain many web applications which have high heterogeneity and runs on the same physical server.

Keyword: *virtual machine, container, web hosting, web application management, docker.*

PENDAHULUAN

Menurut Wikipedia, layanan hosting Internet adalah suatu layanan yang menjalankan server-server Internet sehingga suatu organisasi atau individu dapat menyediakan layanan tertentu yang dapat diakses dari Internet. Ada banyak jenis dan tingkatan layanan hosting Internet, salah satunya adalah web hosting. Web hosting adalah layanan yang memungkinkan pengguna Internet mempunyai situs atau aplikasi web yang dapat diakses melalui World Wide Web (WWW atau Web). Web host adalah perusahaan yang menyediakan ruang pada server untuk meletakkan file-file web pengguna sekaligus juga menyediakan koneksi Internet. Ini menghasilkan suatu infrastruktur bernama *data center*.

Secara teknis, layanan web disediakan oleh software web server seperti Apache dan nGinX. Secara default, setiap web server menyediakan situs web untuk satu domain tertentu, misalnya www.trunojoyo.ac.id (situs web utama Universitas Trunojoyo Madura). Metode *virtual hosting* digunakan untuk menempatkan banyak nama domain atau sub-domain di dalam satu web server. Ini memungkinkan banyak aplikasi web berbagi pakai sumber daya komputasi seperti siklus memori dan *processor*nya. Pada pendekatan *virtual hosting*, file-file dari setiap aplikasi web diisolasi di dalam direktori masing-masing sehingga jika terjadi tindakan kejahatan terhadap suatu aplikasi web maka tidak berpengaruh terhadap aplikasi web lain.

Masalah yang muncul pada pendekatan *virtual hosting* adalah meningkatnya beban kerja dari web server sejalan dengan bertambahnya jumlah aplikasi web dan pengguna (pengunjung) yang harus ditanganinya. Data yang digunakan oleh banyak aplikasi web biasaya disimpan di dalam satu server database. Ini memungkinkan kerusakan data dari semua aplikasi saat server database berhasil dimasuki oleh penyusup. Masalah terakhir yang sering muncul adalah perbedaan versi library atau software dari beberapa aplikasi di bawah tanggungjawab web server. Aplikasi tertentu, misalnya X.trunojoyo.ac.id dibuat menggunakan bahasa pemrograman (interpreter) PHP versi 4.x dan terhubung ke server database MySQL versi 4.x. Aplikasi lain, misalnya Y.trunojoyo.ac.id dibuat menggunakan PHP versi terbaru (5.6.3) dan datanya tersimpan di server MySQL terbaru pula (5.7). Artinya server tersebut harus menginstal 2 versi interpreter dan server MySQL. Ini sangat sulit diwujudkan. Solusinya adalah memisahkan aplikasi dengan kebutuhan khusus pada server khusus. Pendekatan ini memerlukan biaya tinggi untuk pengadaan server, instalasi dan perawatannya.

Penyelesaian masalah di atas banyak menggunakan konsep virtualisasi berbasis mesin virtual (*virtual machine*, VM). Suatu server dapat diisi beberapa server virtual. Sebagaimana server riil, server virtual harus mempunyai sistem operasi (SO) dan berbagai perangkat lunak lain agar layanan web hosting dapat berjalan. Dengan adanya server virtual, setiap aplikasi web berjalan pada mesinnya masing-masing dimana file pustaka, interpreter, server database setiap server disesuaikan dengan kebutuhan dari aplikasi. Pendekatan ini memungkinkan terwujudnya isolasi terhadap suatu aplikasi dan datanya. Penyusup yang dapat menembus server virtual tertentu tidak serta merta dapat mengakses server virtual lain meskipun berada dalam satu server riil yang sama. Kelemahan utama pendekatan ini adalah kebutuhan memory, processor dan storage yang semakin tinggi seiring bertambahnya mesin virtual di dalam suatu server. Hal lain yang perlu diperhitungkan adalah lamanya waktu *startup* dari setiap mesin virtual yang rata-rata lebih panjang daripada waktu startup server riil. Meskipun hadir dengan beberapa kekurangan, pendekatan mesin virtual ini adalah yang paling

banyak digunakan saat ini, termasuk di data center Universitas Trunojoyo Madura.

Kebutuhan akan perangkat server semakin bertambah saat *availability* (ketersediaan) layanan dari aplikasi web menjadi prioritas. Aplikasi dan data yang dianggap penting diperbanyak dengan menginstalnya pada dua atau lebih server riil atau virtual. Jika aplikasi Y.trunojoyo.ac.id dipasang pada dua server fisik berbeda, maka saat terjadi kerusakan atau penghentian layanan pada server pertama, server kedua tetap hadir menyediakan layanan dari aplikasi Y tersebut. Semakin tinggi tingkat ketersediaan yang diharapkan maka semakin banyak server yang harus disediakan, dikonfigurasi dan dirawat. Di sisi lain, jumlah aplikasi web yang harus di*online*kan ke Internet terus bertambah. Server riil mahal dan tidak dapat diadakan setiap waktu. Penambahan server virtual akan semakin memberatkan beban kerja server riil. Penambahan jumlah server juga mengakibatkan peningkatan kebutuhan listrik, ruangan dan jumlah serta kualitas pengelola jaringan.

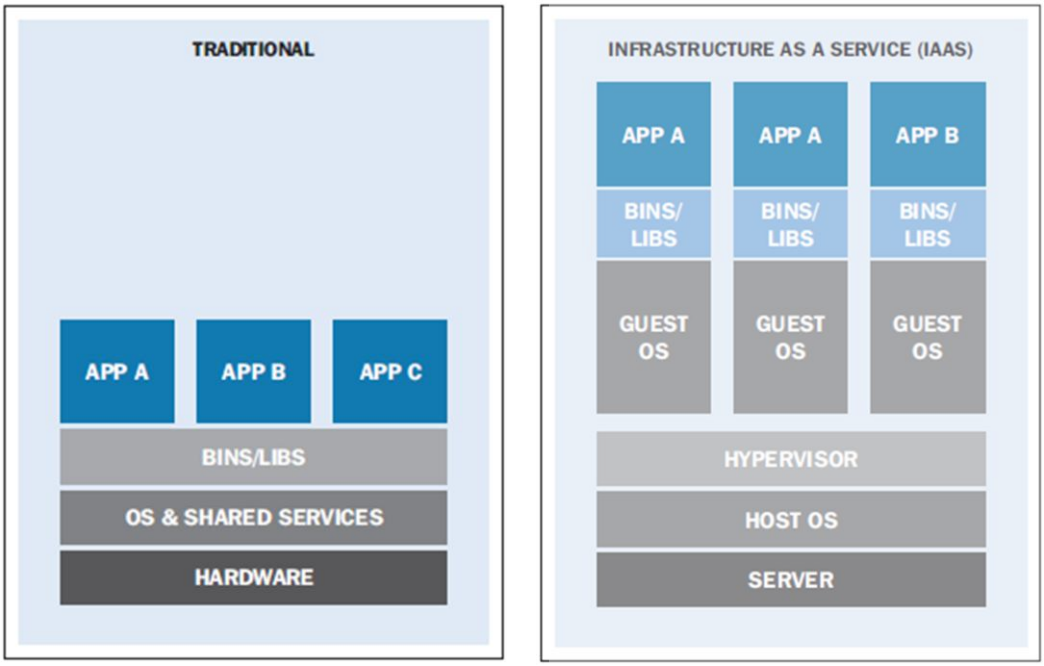
Berdasarkan analisis yang telah dilakukan, teknologi yang tepat untuk mewujudkan sistem web hosting yang *high availabity* adalah dengan mengadopsi teknologi virtualiasi sistem operasi berbasis *Container*, bukan berbasis mesin virtual. Teknologi ini relatif baru dan cepat perkembangannya. Salah satu software yang mengadopsi konsep ini adalah LXC (*Linux Container*). Impementasi dalam LXC tersebut kemudian diadopsi lebih lanjut oleh Docker. Teknologi ini tidak membangun mesin virtual sendiri, lebih hemat memory, *processor* dan *storage*. Waktu yang diperlukan untuk startup Docker juga sangat cepat bahkan jauh lebih cepat daripada server riil. Ini terjadi karena Docker berbagi pakai kernel Linux dari server riil, tidak memerlukan instalasi OS di dalam *container*. *Container* hanya berisi aplikasi web yang akan di*online*kan dan beberapa pustaka atau software yang dibutuhkan oleh aplikasi web.

METODOLOGI PENELITIAN

Tulisan ini mencoba untuk melakukan kajian literatur terhadap teknologi virtualisasi dalam pengelolaan banyak aplikasi web yang biasanya dikelola dalam suatu sistem web hosting. Telaah dimulai dengan melihat konsep hosting aplikasi web tradisional dan *virtual machine* (VM). Beberapa kelemahan dari VM dirangkum dan kemudian membandingkannya dengan konsep virtualisasi sistem operasi (SO) berbasis cointainer. Kontainerisasi aplikasi web menjadi trend dalam pengelolaan web hosting karena hadirnya software bernama Docker dan pendukungnya di bahas pada bagian berikutnya. Terakhir, dijelaskan pula perkembangan dan keunggulan dari Docker yang sudah diteliti dan diimplementasikan dalam 2 tahun terakhir. Review ini dirasakan sangat penting bagi pengembang dan pengelola banyak aplikasi web, terutama aplikasi-aplikasi yang melibatkan heterogenitas tool dan library saat banyak aplikasi berjalan di dalam satu mesin server fisik yang sama.

Virtualisasi untuk Aplikasi Web

Awalnya sebagian besar aplikasi di-*deploy* secara langsung ke suatu server (host) secara fisik. Sistem operasi yang berjalan di atas host akan mengeksekusi aplikasi tersebut. Pada pendekatan tradisional ini (Gambar 1.a), hanya ada satu ruang pengguna (*user space*), *runtime* harus dibagi-pakai (*share*) antar aplikasi, *deployment*nya stabil, *hardware-centric*, siklus perawatan



(a) Pendekatan hosting tradisional

(b) Pendekatan hosting berbasis mesin virtual (VM)

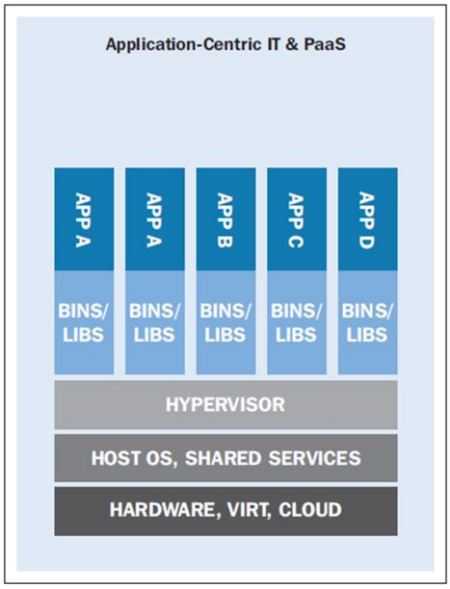
Gambar 1. Perbandingan arsitektural antara pendekatan hosting tradisional dan hosting berbasis mesin virtual

panjang, pemanfaatan *hardware* pada waktu normal sangat rendah [1]. Setiap aplikasi berbagi pakai sistem operasi, layanan dari sistem operasi, file biner dan pustaka. Masalah utamanya adalah kebutuhan berbeda dari setiap aplikasi, misalnya perbedaan versi pustaka atau server database.

Ada empat hal yang menjadi masalah utama dalam pengembangan dan *deployment* aplikasi web [2] yaitu: (1) *Dependency*: suatu *software* bergantung pada banyak *software* lain terutama berbentuk *library* untuk versi tertentu saja. (2) Dokumen yang tidak lengkap atau tidak menyelesaikan masalah instalasi dan operasional awal. (3) *Code rot*. Berbeda versi library, sistem operasi (kernel) atau bahasa pengembangan (*interpreter*) dapat berbeda pula hasil yang diberikan oleh aplikasi. Misalnya *update bug* terhadap suatu kernel atau *library* ternyata dapat membuat *software* yang telah dibuat untuk menangani *error* sebelumnya malah memunculkan masalah baru. (4) Hambatan dalam adopsi dan penggunaan ulang solusi yang sudah ada sebelumnya.

Kesulitan yang hadir pada pendekatan tradisional di atas dapat diselesaikan oleh teknologi virtualisasi (Gambar 1.b). Menurut Zhang [3], virtualisasi merupakan bagian penting dari infrastruktur cloud modern seperti Amazon's Elastic Compute Cloud (EC2) dan Google's App Engine. Sebagian besar pusat data komputasi cloud menjalankan Hypervisor (software komputer di atas sistem operasi yang membuat dan menjalankan mesin virtual) seperti KVM, X.

VM membawa payload yang berat [4]. Setiap VM berisi aplikasi yang ukurannya hanya beberapa Megabyte, file biner dan pustaka, serta sebuah OS lengkap yang mungkin meminta ruang harddisk puluhan Gigabyte sehingga dapat menghabiskan sumber daya terutama saat terdapat banyak VM yang berjalan di dalam server hosting. Hal lain yang perlu dipertimbangkan adalah lamanya boot (*startup time*) yang dilakukan OS dalam VM. Ini terjadi



Gambar 2. Arsitektur teknologi container di dalam server web hosting

karena hardware virtual dari VM dan berbagi-pakainya memory, processor dan harddisk pada Host dengan semua VM yang berjalan. Saat menjalankan aplikasi, SO biasanya meminta memory dan harddisk lebih besar daripada kebutahan sesungguhnya dari aplikasi tersebut.

Penyedia infrastruktur cloud tradisional seperti Amazon Web Services (AWS), Google Computer Engine telah menawarkan VM. VM menyediakan kemampuan untuk ekspansi (up atau down), sumber daya komputasional yang hampir penuh tergaransi, isolasi keamanan, dan akses ke API untuk *provisioning*, tanpa adanya overhead berkaitan dengan pengelolaan server secara fisik. Namun, semakin banyak VM yang dideploy maka makin banyak overhead yang dibayarkan untuk menjalankan image SO *full-blown* dari setiap VM. Pendekatan ini menjadi mahal karena bertambahnya jumlah aplikasi web yang harus dipublikasikan ke Internet.

Kondisi di atas menyebabkan terjadinya pergeseran fokus teknologi informasi dari *hardware-centric* menuju *application-centric* [1]. Virtualisasi mesin bergeser menjadi virtulisasi SO. Lapisan *guest OS* dibuang untuk mengurangi emulasi hardware dan kompleksitas. Aplikasi-aplikasi dipaketkan bersama dengan lingkungan *runtime*nya dan dideploy menggunakan kontainer. OpenVZ, Solaris Zones, dan LXC merupakan contoh dari teknologi ini. SO dari setiap *container* adalah SO dari host. Kebutuhan khusus dari setiap aplikasi dipenuhi oleh file-file biner/pustaka (Bins/Libs) di dalam *container* masing-masing (Gambar 2).

Pernah dilakukan analisis komparatif terhadap implementasi virtualisasi berbasis mesin virtual dan kontainerisasi [5] dan hasilnya diperlihatkan pada tabel 1.

Kontainerisasi Aplikasi Web

Teknologi *container* telah diimplementasikan oleh banyak penyedia layanan online di bidang *cloud computing* dengan pendekatan dan kelebihan/kekurangannya masing-masing. Berikut ini adalah penjelasan mengenai beberapa implementasi teknologi *container* yang popular:

Linux Containers

Linux Containers (LXC) adalah teknologi virtualisasi kernel atau sistem operasi sehingga mampu untuk menjalankan banyak proses di dalam lingkungan terisolasinya masing-masing. LXC dikenal juga sebagai virtualisasi berbasis *container* [6]. Sampai saat ini, LXC berjalan dengan baik di atas OS Linux, terutama pada kelompok distribusi Ubuntu dan Red Hat [5].

Warden Container

Warden *container* menawarkan implementasi containment tidak tergantung kernel yang dapat dipasangkan ke banyak OS host yang mendasari. Implementasi *container* ini digunakan oleh proyek Cloud Foundry untuk meng-*host*-kan banyak aplikasi.

Docker

Docker adalah suatu daemon yang menyediakan kemampuan untuk mengelola *container* Linux sebagai

image tersendiri. Docker memanfaatkan

Tabel 1. Perbandingan antara VM dan kontainerisas

Parameter	Virtual Machines	Containers
Guest OS	Setiap VM berjalan pada hardware dan kernel virtual yang dimuatkan ke dalam wilayah memorynya sendiri.	Semua guest berbagi pakai SO dan kernel yang sama. Image kernel dimuatkan ke dalam memory fisik.
Komunikasi	Melalui perangkat ethernet	Meknisme IPC standard seperti Signal, Pipe dan Socket
Keamanan	Tergantung pada implementasi dari hypervisor	Kontrol akses *mandatory* dapat dimanfaatkan
Kinerja	VM mengalami overhead kecil karena instruksi mesin diterjemahkan dari *guest* OS ke Host.	Container menyediakan kinerja mendekati natif dibandingkan SO host yang mendasari.
Isolasi	Berbagi pakai pustaka, file-file antar guest dan antara guest dengan host tidaklah mungkin	Subdirektori dapat secara transparan dimount dan dibagi-pakaikan.
Waktu startup	Perlu beberapa menit untuk memulai (boot)	Dapat diboot dalam beberapa detik
Storage	Perlu lebih besar storage karena kernel OS lengkap dan program-program yang berasoasiasi harus diinstal dan dijalankan.	Storage lebih kecil karena OS basis dibagi-pakaikan.

LXC untuk implementasi *container* dan menambahkan kemampuan manajemen *image* dan *Union File System* ke dalamnya.

Google lmctfy

Google lmctfy menyediakan suatu API untuk mengkonfigurasi sumber daya sehingga manajemen *container* menjadi lebih sederhana. File konfigurasi lmctfy berbasis ***intent*** tanpa perlu memahami cgroups dan menghilangkan kesulitan dari API LXC yang tidak stabil. Google juga menyediakan *resource sharing* dan dapat mendukung jaminan kinerja.

OpenVZ

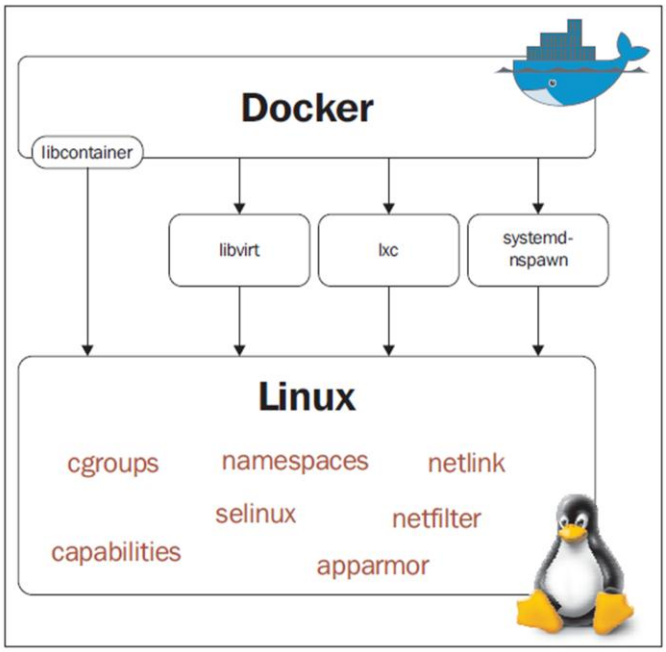
OpenVZ menggunakan kernel Linux yang telah dimodifikasi/dilengkapi sehingga mampu mengelola banyak server fisik dan virtual melalui pemartisian *real- time* dinamis. OpenVZ mempunyai sedikit *overhead* dan menawarkan kinerja lebih tinggi dan dapat dikelola lebih baik dari pada teknologi Hypervisor. Sebagai *container*, OpenVZ juga menggunakan cgroups dan namespaces. Software ini telah menyediakan template untuk memudahkan pembuatan lingkungan virtualnya.

Docker

Docker adalah suatu platform terbuka bagi pengembang perangkat lunak dan pengelola sistem jaringan untuk membangun, mengirimkan dan menjalanan aplikasi-aplikasi terdistribusi [7]. Definisi tersebut membawa pengertian praktis bahwa Docker merupakan suatu cara memasukkan layanan ke dalam lingkungan terisolasi bernama *container*, sehingga layanan tersebut dapat dipaketkan menjadi satu bersama dengan semua pustaka dan software lain yang dibutuhkan [8]. Docker mempengaruhi pengembang

sehingga yakin bahwa layanan tersebut akan berjalan dimanapun Docker berjalan. Ada dua masalah penting yang diselesaikan oleh Docker [9], yang pertama adalah beratnya dan besarnya sumber daya komputer yang digunakan oleh salinan OS yang berjalan di atas hypervisor yang berjalan di atas hardware fisik. Yang kedua, ungkapan "tadi aplikasi ini bekerja di komputer saya tetapi sekarang tidak bekerja seperti tadi". Dengan docker, pengembang aplikasi bekerja dengan anggapan "apa yang dibangun dan dijalankan saat pengembangan dan test adalah sama dengan yang dibangun dan jalankan saat produksi". Ini sejalan dengan Miell & Sayers yang menyatakan bahwa Docker adalah solusi standard untuk menyelesaikan salah satu area berbiaya tinggi dalam siklus pengembangan perangkat lunak, yaitu *deployment* [10].

Docker memberikan beberapa keuntungan bagi pengembang perangkat lunak, termasuk (1) dapat menggantikan peran dari VM, (2) memudahkan pembuatan prototipe banyak software, dengan setiap software dan file terkait ada di *container* terisolasi, (3) menyederhanakan pemaketan software sesuai dengan kemampuan pengembang, bukan mengikuti kemampuan administrator web hosting, (4) mengaplikasikan arsitektur *micro-service*, (5) memodelkan jaringan (terutama data center), (6) memungkinkan produktifitas *full-stack* ketika offline, (7) mengurangi biaya debugging, (8) memudahkan dokumentasi ketergantungan dan *touch-points* dari software, (9) memungkin delivery berkelanjutan. Hal tersebut di disampaikan juga oleh Matthias dan Kane [11]. Boettiger melengkapinya [2] dengan menyatakan bahwa Docker mampu (1) melakukan virtualisasi pada level sistem operasi, (2) men-*deploy container* secara portabel meskipun lintas platform, (3) menyediakan fitur pemanfaatan ulang komponen, (4) *sharing*, (5) *archiving*, dan (6) *versioning* dari *image container*.



Gambar 3. Driver eksekusi dan fitur kernel yang digunakan oleh Docker

Secara teknis, Docker memanfaatkan fitur-fitur dari kernel Linux dimana daemonnya berjalan demi terwujudnya *container* yang ringan dan stabil [1]. Gambar 3 memperlihatkan driver-driver eksekusi dan fitur-fitur kernel yang dapat digunakan oleh Docker.

Fitur kernel yang sangat terkait dengan eksekusi *container* adalah namespace, cgroups dan sistem File UNION. Namespaces merupakan *building blocks* dari suatu *container*. Ada banyak jenis namespaces dan semua mengisolasi *container* aplikasi. Namespaces tersebut dibuat menggunakan *system call* clone. Suatu namespace dapat disambungkan ke namespace. Beberapa namespace yang digunakan oleh Docker adalah PID, Net, IPC, Mnt, Uts dan User. Control Groups (cgroups) menyediakan pembatasan dan *accounting* terhadap sumber daya dari *container*. Dokumentasi Kernel Linux menyatakan bahwa croups menyediakan mekanisme untuk mengagregasi/ mempartisi himpunan tugas dan semua sub-tugas berikutnya ke dalam kelompok hirarkis sesuai dengan perilakunya. cgroups ini dapat dibandingkan dengan perintah shell **ulimit** atau *system call* **setrlimit**. cgroups membolehkan pembatasan sumberdaya ke suatu grup proses, tidak hanya terhadap proses tunggal. cgroups dapat dipecah ke dalam beberapa subsistem seperti CPU, himpunan CPU, blok memory dan I/O.

PENELITIAN TERKAIT

Docker dan LXC dalam sistem web hosting merupakan teknologi yang baru dan belum ditemukan referensi penerapannya di Indonesia. Beberapa publikasi ilmiah di IEEE dan Google Scholars memperlihatkan bahwa adopsi Docker dalam penelitian infrastruktur data center baru dimulai tahun 2014. Teknologi virtualisasi Xen dan LXC pernah diperbandingkan untuk menjalankan beberapa komponen aplikasi [12]. Keduanya mampu menyediakan portabilitas, isolasi dan optimisasi sumber daya hardware. LXC memerlukan sumber daya yang jauh lebih kecil sehingga sangat tepat untuk mengekskusi proses-proses kecil yang terisolasi.

Terdapat hasil pengujian [13] yang mengatakan bahwa image dari Docker jauh lebih kecil daripada KVM (VM) sehingga mempengaruhi (1) waktu start-up: Docker 1.5 detik dan KVM 11.5 detik, (2) kecepatan operasi: Docker menyelesaikan 100000! selama 4.546 detik (deviasi standar 0.02 detik) dan KVM memerlukan 4.793 detik (deviasi standard 0.05 detik). Sedangkan pengujian lainnya [14] memperlihatkan kinerja LXC yang lebih baik dari pada KVM. Pengujian tersebut menggunakan Phoronics Test Suite memperlihatkan bahwa sistem Linux natif mampu memroses 9708.01 request per detik, LXC 76000.65 request per detik dan KVM 7124.55 request per detik. KVM memberikan kinerja sedikit lebih baik untuk operasi baca tulis record dalam ukuran kecil. Namun saat ukuran record diperbesar, kinerjanya menurun di bawah LXC.

Terdapat pula eksperimen [15] yang membangun aplikasi web berupa blog (wordpress) dalam dua versi: pada mesin Linux natif dan berbasis *container* Docker. Hasilnya menunjukkan bahwa kinerja kedua pendekatan ini hampir sama. Sejalan dengan itu, sebuah penelitian lain [15] membandingkan kinerja 5 konfigurasi server MySQL yang melayani banyak akses secara *concurrent*: (1) MySQL berjalan secara normal pada mesin Linux (natif), (2) MySQL di bawah Docker menggunakan host networking dan volume (Docker net=host volume), (3) menggunakan volume tetapi networking Docker normal (Docker NAT volume), (4) menyimpan database di dalam sistem file *container* (Docker NAT AUFS) dan (5) MySQL berjalan di bawah mesin virtual KVM. Hasil pengujian memperlihatkan kinerja yang hampir sama antara MySQL pada Linux natif dengan MySQL yang dijalankan di dalam *container* Docker (Gambar 4). KVM memerlukan *overhead* (biaya) lebih dari 40% dalam semua kasus yang diukur. AUFS meminta *overhead* yang signifikan karena operasi I/O melalui beberapa layer.

SIMPULAN

Artikel ini telah mereview beberapa tulisan mutakhir mengenai teknik virtualisasi, baik berbentuk mesin virtual maupun *container*. Perbandingan keduanya memperlihatkan bahwa teknik *container* merupakan solusi tepat dapat mengelolaan banyak aplikasi web pada suatu sistem hosting. Salah satu software yang fokus dalam kontainerisasi aplikasi web ini adalah Docker. Kontaineriasi yang dilakukan dengan Docker terbukti mampu menghadirkan layanan aplikasi web yang unggul dari sisi kinerja, ketersediaannya tinggi dan hemat sumber daya server. Pada penelitian lebih lanjut, akan dirancang dan diimplementasikan *high availability web hosting system* yang melibatkan teknologi *container*, replikasi, *reverse proxy* dan *load balancing*.

DAFTAR PUSTAKA

[1] Khare, N., 2015. Docker Cookbook. Packt Publishing.

[2] Boettiger, C., 2015. An introduction to Docker for reproducible research , with examples from the R environment. ACM SIGOPS Operating Systems Review, Special Issue on Repeatability and Sharing of Experimental Artifacts.

[3] Zhang, Q., Cheng, L., dan Boutaba, R., 2010. Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, 7(18).

[4] Pamidi, M. R., & Vasudeva, A., 2015. *Impact of Containers on Data Center Virtualization*. Website: http://www.itnewswire.us/Containers.pdf, diakses tanggal 24 April 2015.

[5] Dua, R., Raja, A. R., & Kakadia, D., 2014. Virtualization vs Containerization to support PaaS. *International Conference on Cloud Engineering* (pp. 610–614). IEEE.

[6] Lxc, 2015. *Linux Continer*. Website: http://linuxcontainers.org, diakses tanggal 24 April 2015.

[7] Docker, 2015. *Docker Docs*. Website: https://docs.docker.com/, diakses tanggal 15 Mei 2014.

[8] Hane, O., 2015. Build Your Own PaaS with Docker. Packt Publishing.

[9] Anderson, C., 2015. Docker. Software Engineering, IEEE, 102–105.

[10] Miell, I., dan Sayers, A.H., 2015. Docker in Practice, MEAP Edition Version 3. Manning Publications.

[11] Matthias, K., & Kane, S. P., 2015. Docker Up & Running. OReilly.

[12] Scheepers, M. J., 2014. Virtualization and Containerization of Application Infrastructure : A Comparison. *21st Twente Student Conference on IT* (pp. 1–7). Enschede,The Netherlands: University of Twente, Faculty of Electrical Engineer ing, Mathematics and Computer Science.

[13] Seo, K., Hwang, H., Moon, I., Kwon, O., & Kim, B., 2014. Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud Related Research. Advanced Science and Technology Letters, 66, 105–111.

[14] Sudha, M., Harish, G. M., & Usha, J., 2014. Performance Analysis of Linux Containers - An Alternative Approach to Virtual Machines. International Journal of Advanced Research in Computer Science and

Software Engineering, 4(1), 820–824.

[15] Christner, B., 2015. *WordPress Bare Metal vs WordPress Docker Performance Comparison*. Website: http://blog.loadimpact.com/blog/wordpress-bare-metal-vs-wordpress-docker-performance-comparison/ diakses tanggal 14 Mei 2015.

[16] Felter, W., Ferreira, A., Rajamony, R., & Rubio, J., 2015. An Updated Performance Comparison of Virtual Machines and Linux Containers (pp. 171–172). IEEE.