Implementasi *Virtual Data Center* Menggungakan Linux Container Berbasis Docker dan SDN

Muhammad Fikri Alauddin, Royyana Muslim Ijtihadie, dan Muchammad Husni
Departemen Teknik Informatika, Fakultas Teknologi dan Informasi Institut Teknologi Sepuluh Nopember
e-mail: roy@if.its.ac.id,

***Abstrak*—Software Defined Networking (SDN) adalah teknik jaringan yang berbasis aplikasi sehingga pengguna hanya tinggal menjalankan saja dan software akan menghandle konfigurasi yang diperlukan, tidak seperti teknik jaringan konvensional yang memerlukan konfigurasi dari admin untuk masalah seperti routing. Oleh karena itu sangat cocok untuk diterapkan di Data Center berbasis virtual yang membutuhkan trafik yang sangat fleksible dan akses jaringan yang on-demand. Di era modern ini, teknologi jaringan dituntut untuk akses yang semakin cepat dan semakin dinamis untuk memenuhi kebutuhan dari pengguna yang semakin banyak. Sama halnya dengan Data Center. Oleh karena itu Studi ini mengimplementasi kan Data Center berbasis virtual menggunakan Linux Container bebasis Docker dan SDN untuk menjawab akan permintaan akses yang dinamis dan on-demand. Penggunaan Docker sebagai host dari Data Center berbasis virtual dinilai berhasil dan memang cocok untuk diterapkan. Dengan hasil performa yang dibilang berhasil dan dapat digunakan dan berjalan dengan baik. Docker terintegrasi dengan baik kedalam switch jaringan Data Center berbasis virtual yang dikontrol dengan controller yang berbasis dan menggunakan teknologi SDN.**

***Kata Kunci— Software Defined Networking, Virtual Data Center, Software Container,* Docker.**

I. PENDAHULUAN

ADANYA ledakan jumlah dari perangkat mobile dan kontennya, virtualisasi server, dan digembar gemborkannya teknologi *cloud computing* adalah beberapa alasan kenapa sangat dibutuhkannya arsitektur atau teknik networking baru yang dimana bisa mengatasi keinginan dari teknologi baru yang disebutkan diatas. Struktur *network* statis yang lawas mulai tidak dapat mengatasi permintaan teknologi terbaru yang sudah memakai komputasi dinamis yang membutuhkan *data storage* yang besar.

Penelitian ini mengaplikasikan salah satu teknik arsitektur terbaru yaitu SDN (*Software-defined Networking)*[1]. SDN adalah teknik jaringan yang berbasis aplikasi sehingga pengguna hanya tinggal menjalankan saja dan software akan menghandle konfigurasi yang diperlukan, tidak seperti teknik jaringan konvensional yang memerlukan konfigurasi dari admin untuk masalah seperti routing.

Untuk mempermudah vitualisasi jaringan digunakan Docker[2] sebagai *host* dari Mininet[3] yang berperan sebagai *container* untuk mempermudah mengelola dan mempermudah konfigurasi dari jaringan sendiri.

SDN digunakan karena menjawab permasalahan bahwa arsitektur statis dari jaringan konvensional sangat tidak cocok untuk diterapkan di dalam komputasi dan penyimpanan dinamis yang dimana dibutuhkan di *data center*, kampus, dan lingkungan kerja dewasa ini. Kunci masalah dari dibutuhkannya arsitektur jaringan baru yaitu :

- Perubahan pola trafik: Banyak aplikasi sekarang harus mengakses database dan server yang berada di tempat berbeda lewat *cloud.* Hal ini menyebabkan diharuskannya managemen trafik yang sangat fleksibel dan akses jaringan *on-demand.*
- Munculnya layanan *cloud*: User menginginkan akses kepada aplikasi, infrastruktur dan sumber daya IT yang *on-demand.*

Keuntungan yang didapat dari SDN yaitu dapat diprogram secara langsung, mudah beradaptasi, dapat dikelola secara terpusat, dapat dikonfigurasi langsung lewat program, dan settingan seragam antar jaringan sehingga cocok untuk menjawab permasalah diatas. SDN juga sangat menghemat waktu, menyerdehanakan konfigurasi yang ada, dan mengurangi kemungkinan untuk mesin melakukan error.

Penggunaan *software* virtualisasi jaringan berupa mininet untuk pengaplikasian *Virtual Data Center* berbasis SDN ini dikarenakan pertama, murah karena tidak diperlukannya alat tambahan, dan juga dapat merefleksikan jaringan fisik dengan sama persis karena konfigurasi dan kodingan didalam Mininet dibuat semirip mungkin dengan konfigurasi jaringan di dunia nyata. Dipilihnya Docker sebagai *container* karena Docker telah banyak digunakan dengan dukungan yang besar.

Makalah ini menawarkan sebuah solusi dari permasalahan yang dijabarkan diatas dengan membangun *Virtual Data Center* yang menggunakan Docker sebagai *host* dan platform Mininet sebagai dasarnya.

II. STUDI LITERATUR

Pada Tahap ini, dilakukan pencarian informasi dan studi literatur apa saja yang dapat dijadikan referensi untuk membantu penelitian ini. Informasi didapatkan dari buku dan literatur yang berhubungan dengan metode yang digunakan. Informasi yang dicari adalah *Software-Defined Networking*

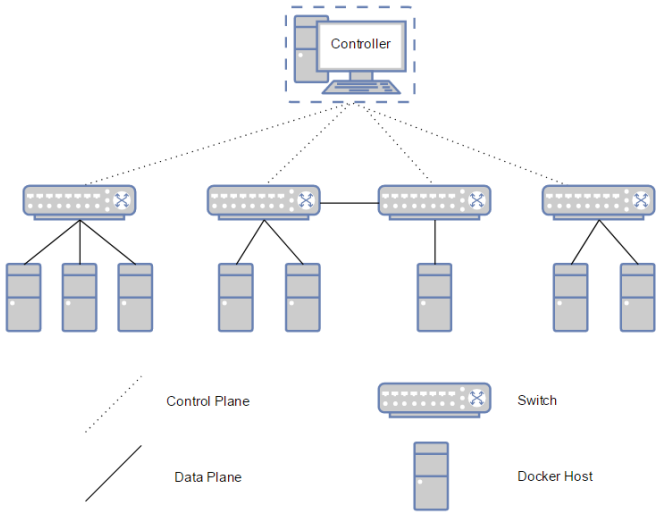
(SDN), Mininet, Containernet[4] yang dimana merupakan modifikasi Mininet agar Docker *container* dapat digunakan sebagai *host* dari Mininet, dan Docker.

Untuk menggabungkan antara *Virtual Data Center* dan web UI yang akan diakses user digunakan kerangka kerja yaitu Flask[5]. Flask sendiri merupakan kerangka kerja berbasis bahasa pemograman python yang sangat cocok untuk diintegrasikan bersama aplikasi berbasis *cloud*. Selain berfungsi sebagai penghubung antar *Virtual Data Center* dan web, Flask juga bertanggung jawab atas pengintegrasian database yang dimana disini dipilih MySql[6] sebagai database.

Penelitian ini juga mengacu pada literature jurnal karya Manuel Peuster, Holger Karl, dam Steven van Rossem yang berjudul "MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments." Yang muncul di konferensi IEEE tentang *Network Function Virtualization and Software Defined Network* (NFV-SDN), 2016.

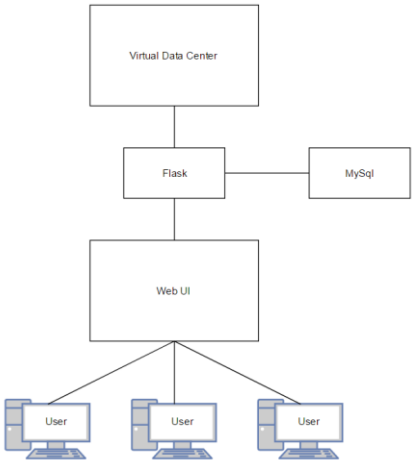
III. DESAIN UMUM SISTEM

Rancangan *Virtual Data Center* dari sistem dapat dilihat pada Gambar 1. Berdasarkan perancangan arsitektur sistem pada Gambar 1, menunjukkan sebuah arsitektur *Virtual Data Center* berbasis *Container* sederhana yang menggunakan *SDN Controller*. Terlihat terjadinya pemisahan antara *data plane* dan *control plane*. *SDN Controller* memiliki koneksi langsung berupa *control plane* dengan masing-masing *virtual switch*, sendangkan masing-masing *virtual switch* memiliki koneksi *data plane* dengan *virtual switch* lainnya yang digunakan untuk mengirim paket.



Gambar 1. Arsitektur Umum Sistem *Virtual Data Center*

Setiap switch di arsitektur pada Gambar 1 bersifat independen dimana setiap switch tidak dapat berkomunikasi satu sama lain kecuali dibuat jalur *data plane* antar switch yang ingin disambungkan. Hal ini dilakukan supaya *data plane* antar user tidak bocor satu sama lain, sehingga menjamin keamanan data antar user. Pada saat mendaftar user otomatis diberikan satu switch di dalam *Virtual Data Center* ini. Untuk menjawab permasalahan agilitas *cloud*, penulis memberikan user pilihan untuk menyambungkan *data plane* kepada switch user lain.

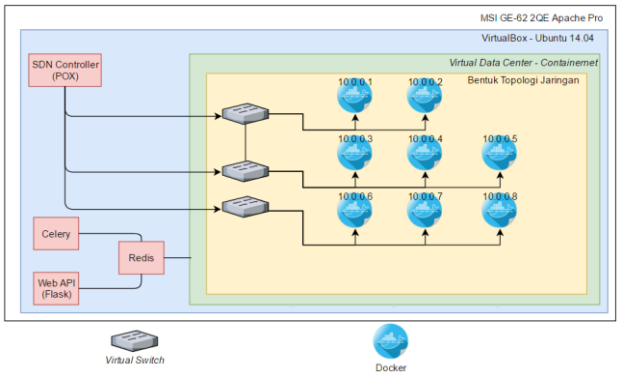


Gambar 2. Arsitektur Umum Sistem

Gambar 2 menunjukkan arsitektur umum dari sistem secara keseluruhan. Untuk memudahkan user mengakses *Virtual Data Center* penulis membuat UI berupa web berbasis Flask dan MySql. Flask dipilih karena berjalan diatas bahasa pemrograman Python sehingga lebih mudah diintegrasikan untuk mengontrol *Virtual Data Center*. Flask juga mengatur keluar masuknya data ke MySql untuk menyimpan inrformasi dan data dari user. Flask juga mengolah data untuk ditampilkan ke web untuk mempermudah user dalam memahami data tersebut.

IV. PENGUJIAN DAN PEMBAHASAN

Lingkungan uji coba akan merujuk pada Gambar 3. Uji coba dilakukan pada sebuah mesin virtual (Virtual Box) yang berjalan diatas perangkat fisik *laptop* dengan *processor* Intel(R) Core(TM) i7-5700 HQ @ 2.7 GHz (8 CPUs) dan 2 x 8 GB 1600 MHz DDR3 RAM.



Gambar 3. Bagan Lingkungan Pengujian

Pembahasan skenario uji coba akan merujuk pada Tabel 1. Berdasarkan scenario uji coba yang telah dilakukan dapat diketahui jumlah docker *container* yang dapat berjalan pada sistem dan juga performa kecepatan transfer data antar dua docker *container* dengan menggunakan *iperf*[7] dari sistem dengan dua spesifikasi sistem yang berbeda. Spesifikasi 1 yaitu dengan 4 GB Ram dan 1 CPU *core*, spesifikasi 2 dengan 8 GB Ram dan 4 CPU *core*.

Tabel 1.
Hasil uji coba performa sistem *Virtual Data Center*

	Skenario		
	Jumlah Docker *Container* Berjalan	Kecepatan Transfer Data (Gbits/sec)	
		2 Docker *Container* aktif	100 Docker *Container* aktif
Spesifikasi 1	619	28.7	12.1
Spesifikasi 2	755	44.6	42.4

Untuk uji coba performa dari MySql di dalam Docker *container* akan merujuk pada Tabel 2. Untuk menguji performa dari MySql digunakan *sysbench*[8].

Tabel 2.
Hasil uji coba performa MySql menggunakan *sysbench*

Tes	Nilai
OLTP statistik tes :	
Jumlah query yang dilakukan:	
Read (Jumlah query)	3536288
Write (Jumlah query)	1221609
Other (Jumlah query)	491162
Jumlah	5249059
Transaksi (Jumlah)	238570 (397.58 per detik)
Deadlocks (Jumlah)	14022 (23.37 per detik)
Permintaan *read/write* (Jumlah)	4757897 (7929.15 per detik)
Operasi yang lain (Jumlah)	491162 (818.53 per detik)
Rangkuman tes :	
Waktu eksekusi (detik)	600.0516 detik
Jumlah dari *event* (Jumlah)	238570
Total waktu dari eksekusi *event* (detik)	23998.5074 detik
Statistik *pre-request* :	
Minimum (millisecond)	2.20ms
Rata rata (millisecond)	100.59ms
Maksimum (millisecond)	1072.63ms
Perkiraan. 95 persen dari total (millisecond)	189.43ms
Thead Fairness :	
Events (Rata-rata/*stddev*)	5964.2500/49.48
Waktu eksekusi (Rata-rata/*stddev*)	599.9627/0.06

Untuk uji coba performa *file IO* dari Docker *container* itu sendiri akan merujuk pada Tabel 3. Untuk pengujian ini digunakan juga *sysbench* sebagai media pengujian.

Tabel 3.
Hasil uji coba performa *file IO* menggunakan *sysbench*

Tes	Nilai
OLTP statistik tes :	
Jumlah query yang dilakukan:	
Read (Jumlah query)	3536288
Write (Jumlah query)	1221609
Other (Jumlah query)	491162
Jumlah	5249059
Transaksi (Jumlah)	238570 (397.58 per detik)
Deadlocks (Jumlah)	14022 (23.37 per detik)
Permintaan *read/write* (Jumlah)	4757897 (7929.15 per detik)
Operasi yang lain (Jumlah)	491162 (818.53 per detik)
Rangkuman tes :	
Waktu eksekusi (detik)	600.0516 detik
Jumlah dari *event* (Jumlah)	238570
Total waktu dari eksekusi *event* (detik)	23998.5074 detik
Statistik *pre-request* :	
Minimum (millisecond)	2.20ms
Rata rata (millisecond)	100.59ms
Maksimum (millisecond)	1072.63ms
Perkiraan. 95 persen dari total (millisecond)	189.43ms
Thead Fairness :	
Events (Rata-rata/*stddev*)	5964.2500/49.48
Waktu eksekusi (Rata-rata/*stddev*)	599.9627/0.06

V. KESIMPULAN/RINGKASAN

Virtual Data Center dapat dibangun, diimplementasikan, dan dikelola dengan baik menggunakan Docker dan platform Containernet yang dimana merupakan implementasi lebih lanjut dari platform Mininet. Performa dari sistem *Virtual Data Center* yang telah dibangun dipengaruhi oleh jumlah dari Docker *machine* yang menyala dan juga jumlah dari Docker *machine* yang terkoneksi ke *switch*. Untuk performa dari MySql yang berjalan dalam Docker di sistem *Virtual Data Center* berhasil melakukan kurang lebih 8748 *query* per detik. Untuk performa *file IO* dari Docker *machine* di sistem *Virtual Data Center* sendiri berhasil mencetak kecepatan *read/write* sebesar 12.299Mb/detik. Untuk spesifikasi dari Docker *image* yang dapat digunakan di sistem *Virtual Data Center* yaitu harus sudah memliki *package net-tools* dan *iproute*. Sedangkan untuk menambah performa dari sistem sendiri dapat digunakan Docker *image* khusus yang dibuat seringan mungkin.

DAFTAR PUSTAKA

[1] "Software-Defined Networking (SDN) Definition - Open Networking Foundation." [Daring]. Tersedia pada: https://www.opennetworking.org/sdn-resources/sdn-definition. [Diakses: 27-Apr-2017].

[2] "What is a Container," *Docker*, 29-Jan-2017. [Daring]. Tersedia pada: https://www.docker.com/what-container. [Diakses: 02-Mei-2017].

[3] "Mininet Overview - Mininet." [Daring]. Tersedia pada: http://mininet.org/overview/. [Diakses: 02-Mei-2017].

[4] "mpeuster/containernet - Docker Hub." [Daring]. Tersedia pada: https://hub.docker.com/r/mpeuster/containernet/. [Diakses: 02-Mei-2017].

[5] "Welcome | Flask (A Python Microframework)." [Daring]. Tersedia pada: http://flask.pocoo.org/. [Diakses: 03-Mei-2017].

[6] "MySQL :: MySQL 5.7 Reference Manual :: 1.3.1 What is MySQL?" [Daring]. Tersedia pada: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html. [Diakses: 03-Mei-2017].

[7] "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool." [Daring]. Tersedia pada: https://iperf.fr/. [Diakses: 28-Mei-2017].

[8] "How To Benchmark Your System (CPU, File IO, MySQL) With sysbench," *Howtoforge*. [Daring]. Tersedia pada: https://www.howtoforge.com/how-to-benchmark-your-system-cpu-file-io-mysql-with-sysbench. [Diakses: 28-Mei-2017].