IMPLEMENTASI DAN ANALISIS MEDIA SERVER BERBASIS DOCKER MENGGUNAKAN PROTOKOL HLS DAN MPEG-DASH

IMPLEMENTATION AND ANALYSIS OF MEDIA SERVER BASED ON DOCKER USING PROTOCOL HLS AND MPEG-DASH

Derry Revy Pryana[1], Dr. Ir. Rendy Munadi, M.T.[2], Sri Ponco Kisworo, S.T., M.M.[3]
Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas
Telkomderryrvp@students.telkomuniversity.ac.id,
[2]**rendymunadi@telkomuniversity.ac.id**, [3]**ponco_k@telkom.co.id**

Abstrak
Meningkatkan kinerja layanan video streaming dengan cara membandingkan protokol HLS dan DASH dalam menyalurkan data kepada pengguna dan mengoptimalkan kinerja server dengan membangun layanan berbasis docker.

Kata kunci : Media server, HLS, DASH, virtualisasi, docker, dan http

Abstract
Improve the performance of video streaming services by comparing the HLS and DASH protocols in distributing data to users and optimizing server performance by building docker-based services.

Keywords: Media server, HLS, DASH, virtualization, docker, and http

1. Pendahuluan

Dengan berkembangnya teknologi memudahkan kita untuk mengakses video dari mana saja. Namun terkadang video yang diakses tidak berjalan dengan lancar atau terjadi *buffering*. Hal ini dapat terjadi dikarenakan banyak faktor, dari sisi pengguna salah satu penyebabnya karena koneksi yang lambat saat memutar video dengan resolusi tinggi. Sedangkan dari sisi penyedia layanan atau server dapat disebabkan oleh karakteristik data media, aplikasi, dan *hardware* (CPU, RAM, dan Storage) [1].

Berdasarkan permasalahan yang terjadi [1] dan pada percobaan sebelumnya [2], pernggunaan virtualisasi dapat meningkatkan performance pada media server. Seiring berkembangnya teknologi virtualisasi, serta pengembangan server berbasis virtual menjadi solusi untuk meningkatkan kinerja server dalam memproses banyak data dengan sumber daya minimum. Docker merupakan salah satu *platform container* yang saat ini banyak diguanakan oleh perusahaan-perusahaan dibidang *Computer Software* dan *IT Service*.

Selain itu pemilihan protokol yang digunakan nantinya akan mempengaruhi server dalam mengolah data yang akan dikirimkan kepada penggunanya dan pengaturan *bitrate* agar dapa menyesuaikan dengan jaringan yang disediakan oleh penggunanya, sehingga pengguna dapat menonton video tanpa menunggu proses *buffring* yang terlalu lama.

Pada tulisan ini, penulis akan mengimplementasikan media server menggunakan NGINX ke dalam virtualisasi berbasis Docker untuk meningkatkan performance pada server. Protokol HLS (*HTTP Live Streaming*) dan MPEG-DASH (*Dynamic Adaptive Streaming over HTTP*) digunakan untuk perbandingan.

2. Dasar Teori /Material dan Metodologi/perancangan

2.1 Media Server

Media server merupakan perangkat keras atau program yang menyimpan dan berbagi data media digital berupa video, audio, dan gambar. Media server biasanya digunakan untuk layanan VOD (*Video On Demand*) dan *video streaming*, dimana data video dalam penyimpanan dibagikan melalui jaringan untuk dapat diakses oleh para penggunanya.

2.2 Docker

Docker merupakan sebuah platform berbasis *open-source* untuk *developer* dan *sysadmin* agar dapat membangun, mengemas, dan menjalankan aplikasi dalam bentuk wadah yang disebut

container [4]. Docker termasuk ke dalam PaaS (*Platform as a Service*) yang digunakan untuk mengisolasi aplikasi dalam bentuk *container*. Fungsi Docker mirip dengan VM (*Virtual Machine*) akan tetapi Docker tidak melakukan virtualisasi seluruh pada sistem operasi, Docker bekerja dengan cara berbagi sistem dengan *host* pembawanya.

2.3 HTTP (*Hypertext Transfer Protocol Secure*)

HTTP berfungsi untuk mengatur format dan menentukan bagaimana data ditransmisikan. Pada data yang bersifat penting dan rahasia protokol HTTP menggunakan enkripsi sebagai metode untuk mengamankan data yang disebut HTTPS (*Hypertext Transfer Protocol Secure*). HTTPS memiliki fungsi sama dengan HTTP, yaitu mengatur bagaimana data diproses hanya saja lebih aman dibandingkan HTTP.

2.4 HLS (*HTTP Live Streaming*)

HLS (HTTP Live Streaming) merupakan protokol yang digunakan dalam mentransfer video dan audio. HLS memungkinkan penggunanya untuk menyesuaikan *bitrate* dengan kondisi jaringan yang disediakan, sehingga memunginkan penggunanya untuk melakukan *streaming* tanpa terputus [5]. HLS menggunakan teknik *progressive download* untuk mentransmisikan datanya. Spesifikasi *codec* yang digunakan pada HLS adalah AAC-LC, HE-AAC + v1 & v2, MP3 untuk audio dan H.264, H.265 untuk video.

2.5 MPEG-DASH (*Dynamic Adaptive Streaming over HTTP*)

MPEG-DASH merupakan protokol *streaming* yang dikembangkan oleh MPEG (*Moving Picture Expert Group*). MPEG mengembangkan sebuah protokol yang dapat pada mentrasmisikan video melalui HTTP dengan kemampuan merubah *bitrate* berdasarkan kemampuan jarinagn yang tersedia. MPEG-DASH membagi file video ke dalam segmen-segmen yang lebih kecil sebelum ditransmisikan kepada *client*. File segmen yang tersedia di transmisikan berdasarkan kemampuan jaringan sehingga proses *streaming* berjalan tanpa ada gangguan.

2.6 NGINX

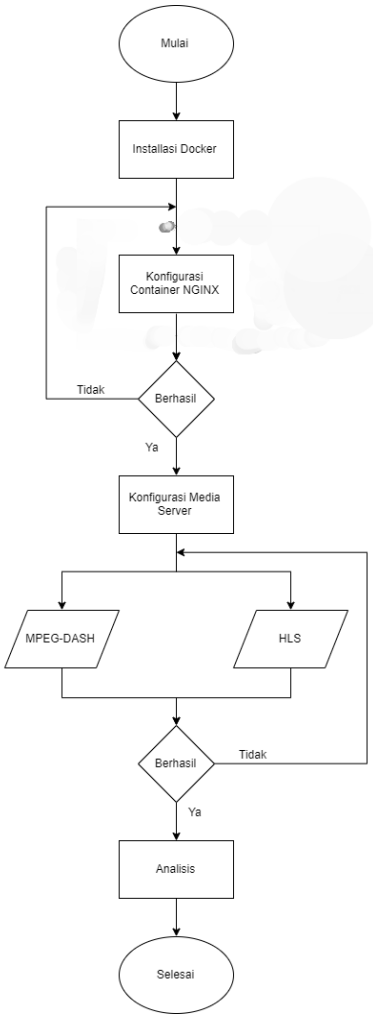
NGINX merupakan *software* yang digunakan untuk membangun *web server* berbasis *open source*. NGINX dapat berfungsi sebagai *reverse proxy*, HTTP *load balancer*, dan *email proxy*. Arsitektur NGINX menggunakan konsep *event-driven* dan asinkron sehingga memiliki kelebihan dalam kecepatan dan skalabilitasnya.

2.6 BASH

Bash merupakan interpreter dari perintah-perintah pada linux. Bash bisa juga disebut *shell* pada unix yang berisikan perintah-perintah. Bash memiliki kompetibel dengan sh (Bourne Shell). Perintah-perintah pada bash dapat dikumpulkan untuk menjadi sebuah perintah baru dalam format .sh. Bash dikembangkan oleh Brian Fox dalam proyek GNU sebagai pengganti perangkat lunak Bourne Shell.

3. Perancangan dan Analisis Sistem

3.1. Gambaran Sistem

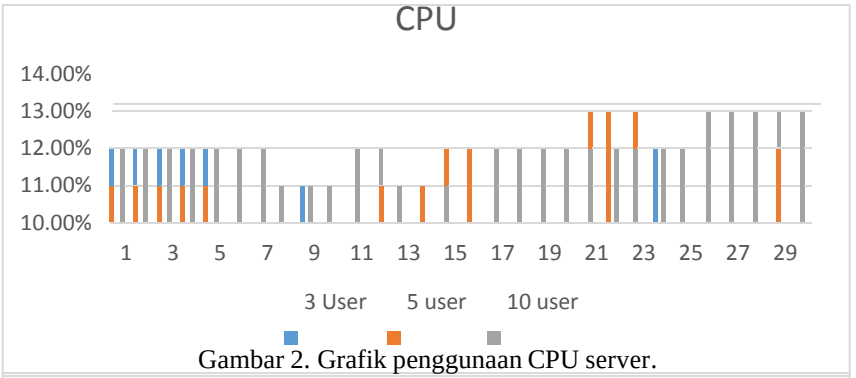


Gambar 1. Flowchart Skema yang Diusulkan.

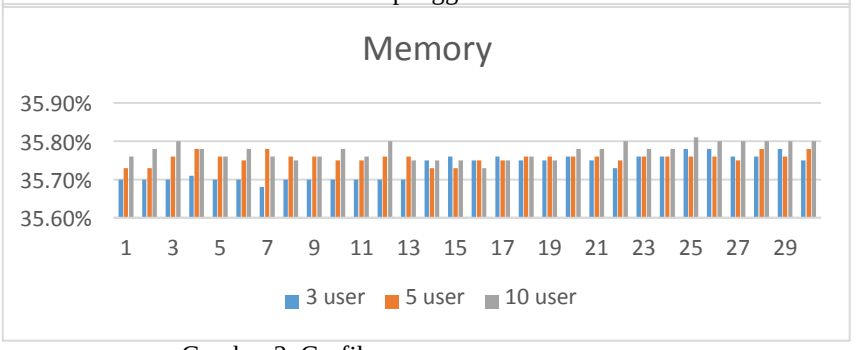
Pada jurnal ini akan dilakukan perancangan sistem media server berbasis *container* untuk layanan *streaming* dengan menggunakan protokol *streaming* MPEG-DASH dan HLS dengan pengaturan *bitrate adaptive*. Pada sistem ini server akan menerima data *video stream* dari encoder, kemudian data akan mengalami proses transcoder dan kompresi sebelum data tersebut ditransmisikan kepada *client*. Pada saat *client* mengakses video menggunakan protokol MPEG-DASH atau HLS kualitas *bitrate* video dapat menyesuaikan dengan kapasitas jaringan yang disediakan pada *client* tersebut, sehingga video dapat diputar dengan lancar. Pada sistem ini kualitas jaringan akan mempengaruhi kualitas *bitrate* video yang akan ditampilkan.

Dalam sistem yang ada pada jurnal ini penulis menambahkan sebuah program yang akan menjalankan *scipts* dengan fungsi mengambil data penggunaan CPU dan *memory*. Program ini dibuat menggunakan bash *scripts*. Tujuan pembuatan program ini adalah untuk mempermudah pengambilan data CPU dan *memory* pada server selama proses pengambilan data analisis. Data CPU dan *memory* akan ditulis ke dalam sebuah format file sebagai bahan penelitian, data tersebut akan dijadikan acuan untuk Tugas Akhir ini.

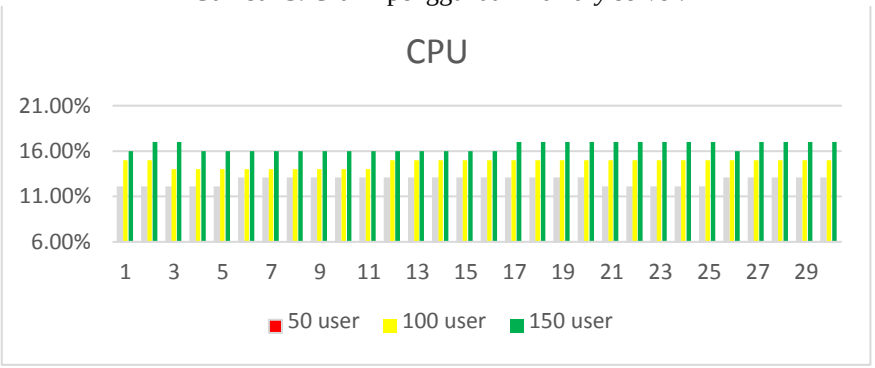
3.2. Analisis Performansi Server



Gambar 2. Grafik penggunaan CPU server.



Gambar 3. Grafik penggunaan *memory* server.



Gambar 4. Grafik penggunaan CPU server dengan jmeter.

Gambar 5. Grafik penggunaan *memory* server dengan jmeter.

Berdasarkan grafik 2 dan 3 dapat dilihat bahwa penggunaan CPU dan *memory* meningkat dengan semakin banyaknya jumlah *client* yang mengakses, peningkatan CPU dan *memory* disebabkan karena meningkatnya permintaan *client* untuk mengakses layanan *video streaming* sehingga meningkatkan proses kerja server. Pada grafik 4 dan 5 nilai penggunaan CPU dan *memory* juga meningkat seiring bertambahnya *client* yang diberikan. Pengujian menggunakan jmeter memungkinkan lebih banyak *client* yang dapat mengakses server. Pengujian pada 150 *user* menunjukkan penggunaan CPU mencapai 17% dan *memory* 41 % dari kapasitas yang diberikan.

3.3. Analisis *Delay*

Tabel 1. *delay* pada protokol HLS.

Bandwidth (kbps)	HLS (s)	DASH (s)
256	**0.328**	**0.047**
512	**0.211**	**0.025**
1000	**0.024**	**0.012**
3000	**0.018**	**0.004**
5000	**0.004**	**0.002**

Pada tabel 1 dan 2 menunjukka hasil penghitungan *delay* dengan variasi *background traffic* pada kedua protokol *streaming*. Hasil pengukuran menunjukkan bahwa besarnya *delay* berbanding terbalik dengan besarnya *background traffic* yang digunakan. Berdasarkan pengukuran pada layanan *streaming* didapatkan bahwa secara rata-rata protokol DASH memiliki nilai yang lebih kecil dibandingkan dengan protokol HLS.

3.4. Analisis *Packet Loss*

Tabel 3. *Packet Loss* pada protokol HLS.

Bandwidth (kbps)	HLS (%)	DASH (%)
256	**16.364**	**2.152**
512	**1.570**	**1.663**
1000	**1.145**	**1.252**
3000	**1.129**	**1.198**
5000	**0.889**	**1.098**

Pada parameter *packet loss* sangat dipengaruhi oleh banyaknya paket yang dikirim dan link yang digunakan pada jaringan. Pada pengujian *background traffic* 256 Kbps nilai *packet loss* cukup besar sekitar 2% pada protokol DASH dan 16% pada protokol HLS. Sedangkan pengujian menggunakan *background traffic* yang lebih tinggi nilai *packet loss* semakin kecil.

3.5. Analisis *Throughput*

Tabel 5. *Throughput* pada protokol HLS.

Bandwidth (kbps)	HLS (kb)	DASH (kb)
256	**258**	**176**
512	**484**	**507**
1000	**997**	**647**
3000	**1698**	**761**
5000	**2339**	**715**

Pada tabel 5 dan 6 menunjukkan hasil dari pengukuran *throughput* dengan variasi *background traffic* 256 Kbps, 512 Kbps, 1 Mbps, dan 5 Mbps. Dari hasil pengukuran didapat bahwa terjadi perubahan nilai *throughput* terhadap peningkatan *backround traffic*. Dari data yang didapat nilai *trhoughput* pada protokol HLS cenderung lebih besar dibandingkan dengan protokol DASH. Hal ini disebabkan karena protokol HLS memiliki data paket yang lebih banyak dibandingkan dengan protokol DASH.

4. Kesimpulan

Penggunaan CPU dan *memory* server mengalami kenaikan ketika *client* mengkses video. Penggunaan CPU dan *memory* bertambah seiring dengan bertambahnya jumlah akses video terhadap server, peningkatan tertinggi terjadi ketika melakukan percobaan menggunakan jmeter, penggunaan CPU mencapai 16% dan *memory* sebesar 32% dari kapasitas yang diberikan.

Penggunaan CPU dan *memory* lebih rendah dibandingkan dengan *virtual machine* berdasarkan penelitian sebelumnya. Hal ini disebabkan karena untuk menjalankan sebuah layanan atau aplikasi berbasis docker tidak dibutuhkan lagi untuk menginstal sistem operasi. Docker menjalankan *container* yang menggunakan sumber daya lebih kecil dibandingkan virtual machine, sehingga proses booting atau saat menjalankan servisnya pun cukup cepat dibandingkan dengan virtual machine.

Perubahan *bitrate* pada protokol DASH lebih baik dibandingkan dengan protokol HLS pada *background traffic* kecil. Hal ini disebabkan karena protocol HLS akan menampilkan video dengan resolusi paling besar di awal *streaming*. Sedangkan pada protokol DASH membutuhkan waktu lama untuk memulai *streaming* dibandingkan dengan protokol HLS. Hal ini disebabkan karena protokol DASH membutuhkan waktu untuk menyesuaikan *backround traffic* dengan resolusi yang akan ditampilkan. Berdasarkan nilai *delay* yang dihasilkan pada *background traffic* 256 kbps, delay pada protokol HLS mencapai 0.328 ms sedangkan *delay* pada protokol DASH hanya sebesar 0.047 ms.

Daftar Pustaka:

[1] CDN. URL: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/.

[2] Docker Inc. 2019. Docker Engine. URL: https://docs.docker.com/engine/.

[3] ITU-T, "GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-THINGS AND SMART CITIES" 2017.

[4] Jeong,J.W., Youngjae Lee,Y., & Kim,J.S. (2015). A High-Performance Media Streaming Architecture based on KVM. Euromicro International Conference on Parallel,23.

[5] NGINX. URL: https://nginx.org/en/.

[6] R. Pantos, HTTP Live Streaming draft-pantos-http-live-streaming-09 [Online]. Available: https://tools.ietf.o livestreaming-09.

[7] K. Pires, G. Simon, "DASH in Twitch: Adaptive Bitrate Streaming in Live Game Streaming Platforms", Pr Adaptive Video Streaming, ACM, Dec 2014, pp. 13-18, doi: 10.1145/2676652.2676657.

[8] X. Li, M. A. Salehi, M. Bayoumi, "VLSC: Video Live Streaming Using Cloud Services", Proc. IEEE International Conferences on Big Data and Cloud Computing, Social Computing and Networking, Sustainable Computing and Communications, IEEE, Oct. 2016, doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.93.

[9] Apple Inc. 2014. About HTTP Live Streaming. URL: https://developer.apple.com/library/archive/referencel LiveStreaming/about/about.html..

[10] Encoding Intelligence. 2016. MPEG-DASH An Overview. URL: https://www.encoding.com/mpeg-dash/.

[11] Docker Inc. 2019. Docker Engine. URL: https://docs.docker.com/engine/.

[12] Encoding Intelligence. 2016. MPEG-DASH An Overview. URL: https://www.encoding.com/mpeg-dash/..

[13] Mouat,A.(2016) Using Docker: Developing and Deploying Software with Containers, United States of America: O'Reilly Media, Inc.

[14] FFmpeg, URL: https://www.FFmpeg.org/.

[15] Tan-actichat, Taurin., Joseph, Pasquale. (2010). VNC in High – Latency Environments and Techniques for http://ieee.org/.