Migrasi Mikroservis pada *Fog Computing* untuk Mendukung Kinerja Komputasi Robot dengan Cakupan Pergerakan yang Luas

Favian Dewanta [1], Reza Afshari [2]
[1,2] Fakultas Teknik Elektro, Universitas Telkom
Jl. Telekomunikasi Terusan Buah Batu No 1, 40257, Bandung, Jawa Barat, Indonesia
[1]E-mail:favian@telkomuniversity.ac.id

Diterima : 23 Juli 2020. Disetujui: 15 Agustus 2020. Dipublikasikan: 19 Agustus 2020

Abstract – Computational power depends on the type of processor, hard disk, memory, and also the type of software/programming language used to execute algorithms in the robot. The more complex the algorithm is, the more the computation power is required by the robot. Moreover, for some robots which are demanded to respond fast from sensing to executing, the requirement of high computational power cannot be neglected anymore. In order to fulfill the requirement of computational power, some companies / institutions employ a fog computing as a supporting system. In this research, we discuss microservices as parts of fog computing infrastructure for solving the lack of computational power in the robot. As a part of research project, we intentionally discuss microservices migration for overcoming robot's mobility from one fog computing coverage to another fog computing coverage. As for the research method, we create checkpoint-restore file from the running container and transfer that file together with imagesnapshot to the new fog node by using SSH and FTP protocol. The results show that the delay of sending microservices imagecontributes a lot in microservices migration as if compared with the delay of checkpoint creation and the delay of resuming microservices imagein a new fog computing server in which the checkpoint and resuming delay vary from 2.4 until 6.3 seconds. In addition, the delay of sending microservices imageincreases significantly in the scenario of sending the imageby employing SSH protocol, which are around 71.2 until 208.2 seconds, with respect to sending the imageby employing FTP protocol, which are around 30.9 until 48.4 seconds.

Keywords: microservices migratio; fog computing; agv robot

Abstrak—Kekuatan komputasi robot sangat bergantung pada jenis *processor, hard disk, memory*, serta jenis *software*/bahasa pemrograman yang mengeksekusi algoritma pada robot tersebut. Semakin kompleks algoritma yang dikerjakan suatu robot, semakin besar pula kebutuhan komputasi yang harus dipenuhi. Terlebih lagi jika robot tersebut dituntut untuk memberikan respon yang cepat dari mulai pembacaan sensor hingga eksekusi oleh aktuator, maka kebutuhan komputasi yang tinggi sudah tidak bisa ditawar lagi. Untuk mengatasi permasalahan kekurangan komputasi, beberapa perusahaan / institusi menggunakan *fog computing* sebagai pendukung sistem. Pada penelitian ini, kami membahas mikroservis sebagai bagian infrastruktur *fog computing* untuk mengatasi kekurangan kekuatan komputasi pada robot. Sebagai satu bagian proyek penelitian, kami secara khusus membahas migrasi mikroservis untuk mendukung mobilitas robot yang berpindah dari area cakupan suatu *fog computing* menuju suatu *fog computing* yang lainnya. Adapun metode yang digunakan adalah dengan cara membuat *checkpoint-restore file* dari *container* yang sedang berjalan dan kemudian mengirimkannya bersama dengan *snapshot image*pada *fog node* tujuan dengan menggunakan protokol SSH dan FTP. Hasil eksperimen menunjukkan *delay* proses pengiriman *image*mikroservis berkontribusi paling banyak pada proses migrasi mikroservis jika dibandingkan dengan *delay* pembuatan *checkpoint* dan instalasi *image*mikroservis pada server *fog computing* tujuan dengan nilai sekitar 2,4 sampai 6,3 detik. *delay* pengiriman tersebut bertambah sangat signifikan jika proses pengiriman *image*mikroservis menggunakan protokol SSH di mana nilainya sebesar 71,2 sampai 208,2 detik, jika dibandingkan dengan protokol FTP yang berkisar 30,9 sampai 48,4 detik.

Kata kunci: migrasi mikroservis; *fog computing*; robot agv

I. PENDAHULUAN

A. Latar Belakang

Robot adalah salah satu perangkat penting dalam proses produksi barang secara massal. Adanya robot akan membuat pekerjaan menjadi lebih cepat dan efisien. Terlebih, robot dapat bekerja secara konsisten dan presisi sehingga dapat meningkatkan kualitas produksi dan menekan produk yang cacat.

Berbagai macam jenis robot digunakan dalam proses produksi, mulai dari robot yang statis dan berlengan [1] [2], robot yang bergerak dinamis menggunakan roda [3]. Setiap robot tersebut memiliki keunggulan masing-masing dan fungsi yang berbeda. Fungsi tersebut berkaitan erat dengan lokasi tempat robot itu bekerja, jenis beban yang harus ditangani, serta tingkat presisi yang diharapkan. Namun yang jelas, kesemua aspek

fungsionalitas tersebut sama-sama membutuhkan kolaborasi sensor, prosesor, hingga aktuator yang mumpuni untuk mencapai tujuan.

Ada cukup banyak publikasi penelitian yang pernah membahas desain sistem robot dalam melakukan suatu pekerjaan atau mengatasi tantangan seperti yang dibahas dalam publikasi berikut ini [4] [5]. Tentunya bahasan desain sistem robot disajikan per bagian / fitur dikarenakan kompleksitas analisisnya. Terlebih lagi, penelitian satu fitur dalam robot melibatkan kolaborasi sensor hingga aktuator untuk membentuk suatu sistem yang terkontrol dalam *loop* tertutup. Dalam skala yang lebih besar, di mana melibatkan banyak sensor dan aktuator, sistem tertutup pada robot tersebut bahkan didesain untuk bekerja menggunakan jaringan yang lebih sering disebut dengan istilah *networked control system* [6].

Terlepas dari banyaknya bahasan mengenai desain robot tersebut, tetap saja robot memerlukan komputasi yang kuat untuk dapat menjalankan algoritma dari sang arsitek. Dalam hal komputasi, kita dapat berdebat mengenai fakta ini yaitu semakin kompleks suatu pekerjaan yang harus ditangani suatu robot, maka semakin kompleks pula pemrograman ataupun algoritmanya. Hal tersebut tentu saja berpengaruh pada kebutuhan komputasi pada robot tersebut yang meliputi jenis prosesor, jenis memori, hingga ke media penyimpanan data.

Sebagai ilustrasi, mari kita ambil contoh dua buah robot berikut ini [7] [8]. Pada penelitian [7], pergerakan robot AGV didesain untuk mengikuti jalur yang ditandai dengan RFID. Sistem tersebut cukup simpel dan hanya menggunakan prosesor yang cukup minim komputasinya (mikrokontroller 8 bit), di mana dia hanya bertugas untuk mengenali jenis RFID yang berkorelasi dengan arah rute dan pergerakan roda robot. Kondisi tersebut tentu saja sangat jauh berbeda jika robot harus mengenali lingkungan dengan menggunakan kamera dan bergerak dengan kaki seperti pada [8]. Dalam penjelasan spesifikasinya, kita dapat melihat bahwa robot yang memiliki tugas yang kompleks tersebut dilengkapi dengan prosesor 64 bit.

Dengan adanya ilustrasi tersebut, maka kualitas dan fungsionalitas dari robot sangat bergantung pada kekuatan komputasi yang dimiliki. Sehingga untuk meningkatkan kapasitas robot, pemilik dan pengguna selain harus memodifikasi sensor dan aktuator, juga seringkali harus memodifikasi perangkat keras yang digunakan. Tentu saja modifikasi perangkat keras akan berpengaruh pada ukuran fisik dan tampilan suatu robot dikarenakan ukuran dan bentuk *hardware* yang berbeda, sebagai contoh *motherboard* prosesor 8 bit seperti Arduino dan Atmel [9] tentu berbeda dibandingkan dengan prosesor 64 bit seperti Intel Atom [10] atau Arm Cortex [11]. Tak hanya permasalahan tampilan, harga juga seringkali jadi pertimbangan untuk meningkatkan kapasitas suatu robot.

Terkait dengan permasalahan komputasi, perkembangan teknologi komputasi terkini telah berubah cukup jauh. Dengan bergabung dengan teknologi jaringan, komputer kini bisa saling terhubung sehingga komputasi yang kompleks bisa didistribusikan kepada banyak komputer. Konsekuensinya, komputer yang memiliki komputasi yang terbatas dapat meningkatkan kapasitasnya secara virtual. Hal ini seringkali disebut dengan *cloud computing* jika servernya secara geografis jauh dari pengguna dan juga *fog computing* jika servernya dekat dengan pengguna secara geografis [12] [13].

Tentu saja teknologi *fog computing* sangat bermanfaat untuk mendukung kinerja komputasi robot. Terlebih lagi jika pemilik / pengguna enggan untuk mengganti perangkat keras robot yang membutuhkan investasi cukup besar. Strateginya adalah cukup melakukan *task offloading* [14] pada pekerjaan yang memerlukan komputasi yang kompleks. Selebihnya pekerjaan yang mudah cukup ditangani dengan kekuatan komputasi robot yang ada.

Namun solusi ini tidaklah begitu simpel, terutama untuk objek yang bergerak dinamis dengan area pergerakan yang luas seperti pada robot. *Fog computing* seperti diketahui bersama, memiliki batasan cakupan pelayanan. Maka dengan kondisi tersebut, layanan *task offloading* pada *fog computing* harus mendukung skema *handover* [14] atau migrasi layanan seperti yang umumnya dilakukan pada telepon seluler.

Pada penelitian ini, kami mendesain suatu skema untuk melakukan migrasi mikroservis pada *fog computing* untuk mendukung kinerja komputasi robot yang bergerak dengan cakupan yang luas. Dukungan komputasi pada robot diberikan secara khusus oleh kontainer yang berisi suatu *bundle* program / perangkat lunak yang umumnya disebut sebagai mikroservis [15]. Penggunaan istilah mikroservis digunakan karena fitur atau layanan pada robot nantinya akan dikembangkan secara terpisah / independen namun harapannya semua fitur tersebut dapat dengan mudah dirangkaikan melalui suatu *application programming interface* (API).

B. Tujuan dan Rumusan Masalah

Adapun riset ini bertujuan untuk menginvestigasi performa proses migrasi mikroservis yang nantinya akan mendukung kinerja robot saat berpindah dari satu cakupan servis *fog computing* menuju cakupan servis *fog computing* lainnya. Secara khusus, penelitian ini bertujuan

untuk mencari besarnya *delay* yang diperlukan saat proses migrasi tersebut dilangsungkan.

Adapun rumusan masalah yang ingin dipecahkan adalah kaitan antara besar *image*mikroservis terhadap besarnya *delay* yang diperlukan. Selain itu, penelitian ini juga berusaha menganalisis permasalahan pemilihan protokol komunikasi yang diperlukan dalam proses migrasi mikroservis yang dilakukan.

II. METODE PENELITIAN

A. Model Sistem

Dalam penelitian ini, sistem yang akan diuji dapat dimodelkan menjadi tiga buah entitas yang masing-masingnya memiliki kekuatan komputasi sebagaimana disebutkan berikut ini.

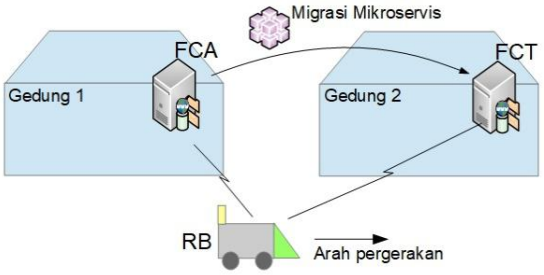
- Robot (RB) merupakan sebuah entitas yang memiliki kekuatan komputasi minimalis dan terbatas. Walaupun dalam paktiknya robot dapat memiliki kekuatan komputasi setara laptop / notebook, namun pada dasarnya kekuatan komputasi tersebut hanya digunakan untuk mengontrol dan memproses sinyal dari dan ke sensor ataupun aktuator yang tertanam di tubuh robot tersebut. Adapun modifikasi atau penambahan kapasitas seringkali harus mengubah perangkat robot secara keseluruhan. Oleh karena itu, dalam penelitian ini robot diasumsikan sebagai entitas dengan kapasitas komputasi yang terbatas.
- *Fog Computing* Asal (FCA) merupakan *server* penyedia layanan mikroservis *task offloading* pada robot pada suatu bangunan tertentu sebelum robot berpindah menuju cakupan layanan *fog computing* lainnya. Entitas ini diasumsikan memiliki kekuatan komputasi yang cukup besar untuk melayani robot yang membutuhkan pada suatu area.
- *Fog Computing* Tujuan (FCT) merupakan server penyedia layanan mikroservis setelah robot meninggalkan cakupan layanan FCA. Sama seperti penjelasan sebelumnya, entitas ini juga diasumsikan memiliki kekuatan komputasi yang dapat melayani seluruh klien yang membutuhkan layanan mikroservis.

Dalam kondisi riil, FCA dan FCT masing-masingnya dapat berupa sebuah server yang terdiri dari satu hingga beberapa baremetal yang saling terhubung menggunakan jaringan internal server. Kemudian sumber daya komputasi tersebut dikontrol menggunakan *cloud operating system* seperti pada Amazon Web Services [16], Google Cloud Platform [17], maupun Openstack [18]. Secara sederhananya, FCA dan FCB merupakan kumpulan beberapa *Virtual Machine* (VM) dan juga *container* dalam satu server yang saling terhubung untuk melayani permintaan komputasi RB. Secara lebih detailnya kami sarankan pembaca untuk membaca beberapa referensi berikut untuk pengayaan [19] [20] [21].

Adapun RB dapat diimplementasikan menjadi bermacam-macam robot dengan segala macam fungsi, baik beroda maupun berkaki. Namun sebagai prasyarat dalam melakukan *task offloading*, RB harus memiliki antar muka jaringan *wireless* (WiFi). Selain itu, RB juga harus mampu berkomunikasi dengan FCA dan FCT menggunakan pelbagai API (*application programming interface*) yang mendukung komunikasi antara RB dan FCA / FCB. Di antara sekian banyak API, yang paling popular digunakan antara lain adalah Rest API [22], SOAP API [23], dan gRPC API [24].

Untuk memperjelas mengenai model sistem yang dikembangkan, pembaca bisa melihat Gambar 1. Pada gambar tersebut, FCA dan FCT digambarkan sebagai *server* yang terpasang pada suatu gedung manufaktur. RB adalah sebuah robot AGV yang beroperasi antar gedung untuk mengantarkan suatu barang atau bahan untuk proses produksi. Koneksi antara RB dengan FCA maupun RB dengan FCT adalah menggunakan jaringan WiFi dan suatu protokol komunikasi (API) tertentu.

Proses migrasi dimulai ketika RB akan meninggalkan FCA. Masih di Gambar 1, FCA diilustrasikan mengirimkan mikroservis kepada FCT. Sedangkan, RB di saat yang bersamaan juga terhubung pada dua buah server untuk proses *hand-shaking* layanan *task offloading*. Di akhir, proses migrasi dikatakan berhasil jika RB dapat Kembali terhubung dan menggunakan mikroservis pada *server* FCT.



Gambar 1 Ilustrasi Model Sistem Migrasi Mikroservis

Lain halnya dengan konsep *handover* pada layanan 4G yang mengedepankan proses yang terjadi secara *real-time*, migrasi mikroservis pada penelitian ini tidak disyaratkan secara *real-time*. Hal ini dikarenakan proses perpindahan dari satu gedung ke gedung lainnya diperkirakan memakan waktu yang tidak sebentar yakni sekitar 10 – 20 menit, mengingat robot akan berjalan dengan kecepatan yang lambat agar barang yang dibawa tidak jatuh dan stabil di atas robot. Sehingga pada

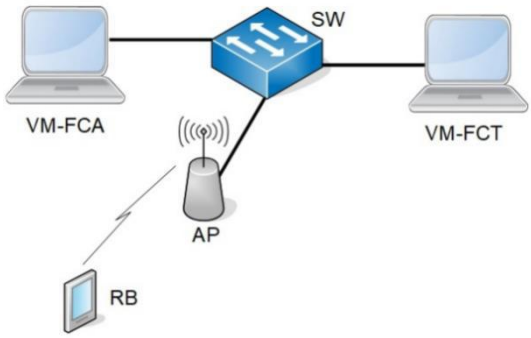
saat robot tiba di gedung dan terhubung dengan FCT, dia bisa langsung menggunakan mikroservis yang sudah diterima oleh FCT.

B. Desain Eksperimen

Sebelum kami menjelaskan desain eksperimen dalam penelitian ini, kami perlu menyampaikan beberapa asumsi dan batasan penelitian yang digunakan. Adapun uraian asumsi dan batasan penelitian adalah sebagai berikut ini.

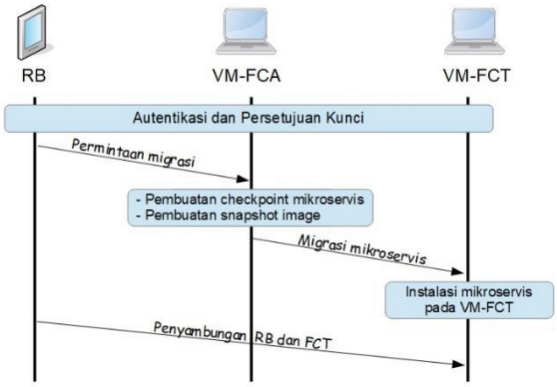
- *Server* FCA dan FCT direalisasikan menggunakan dua buah VM pada satu PC host yang sama.
- *Server* FCA dan FCT dipasang pada jaringan yang masih satu subnet sehingga tidak diperlukan proses perutean IP dalam berkomunikasi antar server tersebut. Hal ini dilakukan karena dalam penelitian ini faktor jenis protokol perutean IP dan juga lama peruteannya tidak menjadi perhatian dalam penelitian ini.
- Penelitian ini tidak akan membahas proses autentikasi dan *key agreement* walaupun dalam praktiknya ketiga entitas FCA, FCT, dan RB tersebut pasti akan saling mengautentikasi sebelum memulai migrasi mikroservis. Hal ini dikarenakan bahasan akan menjadi tidak fokus dan sangat panjang. Oleh karena itu, kami merekomendasikan untuk membaca hasil penelitian kami sebelumnya yang membahas autentikasi dan cara membangun kepercayaan antar entitas [14] [25].
- Proses sinkronisasi dan *node discovery* di antara RB, FCA, dan FCT juga tidak akan kami bahas. Kami asumsikan bahwa ketiga entitas tersebut telah memiliki informasi mengenai lokasi geografis maupun alamat jaringan (IP / MAC address) dan juga *port* layanan mikroservisnya.
- Keamanan jaringan dan juga kebijakan perlindungan entitas serta transaksi di dalamnya adalah hal yang tidak menjadi perhatian dalam riset ini walaupun pada kondisi riilnya ada kemungkinan serangan internal dari dalam jaringan sendiri (insider attack). Kami mengasumsikan bahwa semua entitas yang berada dalam jaringan adalah entitas yang terpercaya dan tidak mencurigakan. Hal ini dilakukan mengingat sudah banyak riset dan perangkat yang digunakan untuk melindungi aset siber.

Topologi jaringan yang digunakan adalah seperti yang diberikan pada Gambar 2. Dua buah virtual machine VM-FCA dan VM-FCT terhubung ke jaringan menggunakan sebuah *switch virtual* SW. Untuk mensimulasikan robot, kami menggunakan sebuah *mobile phone* RB yang terhubung melalui access point AP. Karena menggunakan switch, ketiga entitas tersebut secara logis berada dalam satu jaringan yang sama sebagaimana disebutkan dalam uraian batasan riset sebelumnya.



Gambar 2 Topologi Jaringan dalam Eksperimen Migrasi Mikroservis

Komunikasi antara RB, FCA, dan FCT dilakukan dengan menggunakan *socket programming* serta SSH dan FTP. Secara khusus, komunikasi RB dan FCA serta RB dengan FCT dilakukan dengan *socket programming* dengan model hubungan *client-server*. RB diposisikan sebagai *client* yang menginisiasi koneksi sedangkan FCA dan FCT sebagai servernya yang menerima dan membalas permintaan layanan. Selain *socket programming*, protokol SSH [26] dan FTP [27] digunakan untuk berkomunikasi dan migrasi mikroservis antara FCA dan FCT. SSH adalah *secure shell* yang sering digunakan untuk komunikasi yang aman dan terenkripsi menggunakan algoritma enkripsi kunci publik RSA [28] sedangkan FTP adalah protokol yang tidak terenkripsi. Kedua protokol pengiriman data tersebut berbasis protokol TCP pada layer 4 OSI serta menggunakan *port* 22 dan 21.

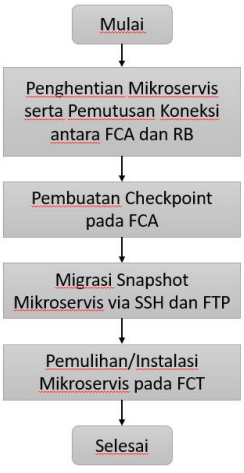


Gambar 3 Alur Proses Migrasi Mikroservis pada Eksperimen

Adapun proses migrasi yang dilakukan dalam eksperimen ini adalah sepeti yang diberikan

pada Gambar 3 di atas. Pada awalnya semua entitas akan melakukan autentikasi dan persetujuan kunci. Setelah itu ketiga entitas tersebut dapat saling berkomunikasi dan begitu juga RB yang sedang dalam posisi terhubung ke FCA dan menggunakan layanan mikroservis. Selain Gambar 3 tersebut, kami juga menyajikan diagram alir eksperimen pada Gambar 4 untuk semakin mempermudah dalam memahami proses migrasi. Kemudian detail proses migrasinya adalah sebagai berikut ini

- Pada saat akan meninggalkan Gedung 1 dan menuju Gedung 2 seperti ditunjukkan pada Gambar 1 sebelumnya, RB akan meminta VM-FCA untuk memigrasi mikroservis kepada VM-FCT.
- Setelah menerima permintaan tersebut, VM-FCA akan membuat *checkpoint* posisi komputasi CPU maupun memory yang sedang dilakukan oleh mikroservis yang terkait.
- Kemudian VM-FCA akan menghentikan mikroservis dan membuat *snapshot image*dari mikroservis tersebut
- Pada proses selanjutnya, *snapshot* mikroservis tersebut akan dimigrasikan ke VM-FCT menggunakan protokol pengiriman data SSH dan juga FTP.
- VM-FCT kemudian akan melakukan instalasi *snapshot image*VM-FCT tersebut pada sistemnya dan mengembalikan mikroservis pada kondisi terakhir sebelum dikirimkan.
- Terakhir, RB akan dapat menggunakan mikroservis tersebut setelah tersambung dengan terautentikasi oleh VM-FCT.



Gambar 4 Diagram Alir Eksperimen

Untuk melengkapi bahasan desain eksperimen, kami menyajikan TABEL I di bawah ini. Dalam tabel tersebut, kami menggunakan bahasa pemrograman python versi 3.7 untuk menangani proses di dalam *fog computing* VM-FCA dan VM-FCT, serta melayani *task offloading* yang diminta oleh RB kepada mikroservis yang dipasang pada dua buah *fog computing* tersebut. Pemilihan bahasa pemrograman python tersebut dilandasi oleh keadaan di mana banyak *library* aplikasi yang berguna bagi RB yg disediakan oleh bahasa pemrograman tersebut, antara lain *computer vision*, *machine learning*, pemrosesan Big Data, *socket programming*, dan lain sebagainya.

TABEL I PARAMETER DALAM EKSPERIMEN MIGRASI MIKROSERVIS

No	Parameter	Keterangan
1	Software container	Docker
2	OS pada VM-FCA dan FCT	Ubuntu 16.04
3	Hypervisor	Virtualbox
4	Docker *image*/ mikroservis	Python 3.7
5	Jumlah core CPU pada VM-FCA & VM-FCT	2 core
6	Memory RAM	3 GB
7	Ukuran snapshot image	90MB, 170MB, dan 240MB
8	Protokol migrasi	SSH & FTP

Software kontainer yang digunakan adalah Docker [29], mengingat *software* ini begitu popular dan banyak digunakan oleh perusahaan IT. Alasannya sederhana, konsep *software* monolitik sudah semakin usang dan biayanya cukup mahal. Hal tersebut akhirnya mendorong banyak perusahaan IT untuk beralih menuju *software* dengan konsep arsitektur non-monolitik (mikroservis). Di antara sekian banyak *software* kontainer, Docker adalah yang paling popular karena sifatnya yg *open-source* dan juga didukung oleh banyak kontributor serta komunitas. Sehingga, kontainer Docker bisa dijalankan di hampir semua *cloud platform*, termasuk Google cloud, AWS, Microsoft Azure, maupun openstack.

Kemudian variabel yang diamati dalam eksperimen ini adalah ukuran *image*mikroservis yang diatur sebesar 90, 170, dan 240 MB. Dalam penelitian ini, besarnya mikroservis berkaitan erat dengan jenis layanan serta frekuensi dalam pencatatan hasil penginderaan oleh sensor-sensor. Sebagai ilustrasi, untuk aplikasi yang berbasis pada machine learning, seringkali dibutuhkan data latih yang banyak. Semakin banyak data latih, maka semakin akurat keluaran dari sistem tersebut. Sehingga secara otomatis, *machine learning* akan membutuhkan *metafile* yang begitu besar, terkadang hingga puluhan atau ratusan MB tergantung pada jenis datanya (*text, image, video*). Tak hanya itu, kombinasi dengan akumulasi pencatatan hasil penginderaan (*sensing log*) juga seringkali berkontribusi besar terhadap ukuran kontainer yang digunakan.

Terakhir, ada dua metode pengiriman mikroservis yang digunakan, yakni SSH dan juga

FTP. Walaupun di awal kami menyebutkan tidak akan membahas masalah keamanan siber dan autentikasi, namun melalui variabel ini kami ingin menyajikan sudut pandang praktis pemakaian dua buah variabel tersebut bagi, terutama terkait dengan kualitas layanan (QoS). Di antara sekian banyak protokol pengiriman data yang tergolong handal (*reliable*), SSH dan FTP dipilih dalam eksperimen ini karena dua protokol tersebut sering digunakan dan mudah pengimplementasiannya dalam Bahasa pemrograman python 3.7.

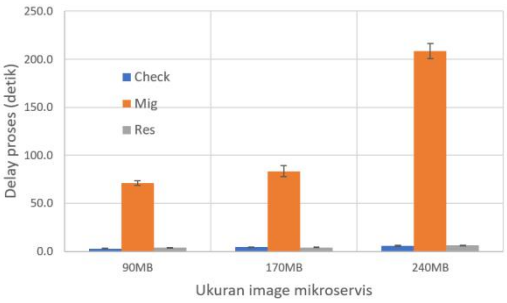
III. HASIL DAN PEMBAHASAN

A. *Hasil Eksperimen*

Dalam eksperimen ini, kami secara khusus mengamati performa proses migrasi melalui empat jenis *delay* yang dijelaskan berikut ini.

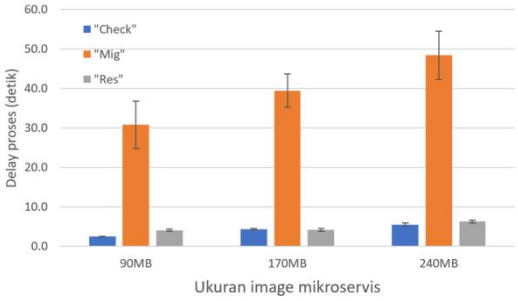
- *Delay checkpoint* (*Check*), merupakan *delay* yang dialami oleh sistem pada saat VM-FCA menghentikan komputasi mikroservis, membuat *checkpoint* pada *memory* yang menunjukkan posisi terakhir perintah yang dijalankan, serta membuat *snapshot image*mikroservis sebelum dikirimkan ke VM-FCT.
- *Delay* migrasi (Mig), merupakan *delay* yang dialami saat pengiriman *image*mikroservis dari VM-FCA ke VM-FCT.
- *Delay restore* (Res), merupakan *delay* yang saat melakukan proses pengembalian *image*mikroservis menjadi mikroservis yang utuh kembali di VM-FCT, sehingga mikroservis dapat beroperasi dengan normal di server *fog computing* yang baru.
- *delay* total merupakan *delay* akumulasi dari proses migrasi mikroservis mulai dari pembuatan *checkpoint*, pengiriman *image*mikroservis, hingga instalasi *image*mikroservis tersebut pada *fog computing* yang baru.

Bahasan eksperimen kami sajikan dalam tiga buah sudut pandang, yakni perbandingan penggunaan protokol SSH dan FTP, perbandingan ukuran *image* mikroservis, serta komponen yang paling berkontribusi dalam menyumbang *delay* yang besar dalam proses migrasi. Ketiga hal tersebut akan dielaborasi dengan bantuan beberapa grafik seperti yang diberikan di bawah ini.



Gambar 5 Perbandingan Performa Migrasi Mikroservis pada Protokol SSH

Pada Gambar 4 tersebut, pembaca dapat melihat adanya hubungan yang erat antara besar *image* mikroservis dengan besarnya *delay* rata-rata proses migrasi mikroservis. Secara umum terlihat bahwa semakin besar ukuran *image*, maka *delay* yang dibutuhkan akan semakin besar. Di antara ketiga komponen *delay* tersebut, pembaca dapat melihat bahwa *delay* Mig adalah yang paling dominan pada semua ukuran *image* dengan masing-masing sebesar 71,2 detik, 83,4 detik, dan 208,2 detik untuk ukuran *image*90MB, 170MB, dan 240MB. Sementara itu, *delay* Check dan Res berbeda cukup tipis, yakni 2,9 dan 3,9 detik pada 90MB, 4,4 dan 4,3 detik pada 170MB, serta 58 dan 6,4 detik pada 240MB.

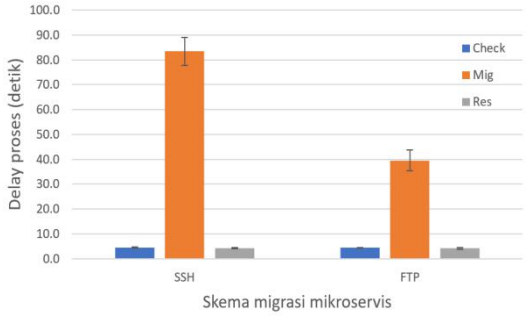


Gambar 6 Perbandingan Performa Migrasi Mikroservis pada Protokol FTP

Sama halnya dengan protokol SSH, pada penggunaan protokol FTP, pembaca dapat melihat dengan jelas adanya hubungan yang erat antara ukuran *image* mikroservis dengan lamanya waktu migrasi. Pada kondisi ini, komponen yang paling berkontribusi juga didominasi oleh *delay* Mig dengan nilai 30,9 detik pada 90MB, 39,5 detik pada 170MB, dan 48,4 detik pada 240MB. Adapun *delay* Check dan Res berbeda cukup tipis walaupun masih terlihat jelas hubungan antara *delay* dan ukuran *image*, dengan nilai 2,4 dan 4,1 detik pada 90MB, 4,3 dan 4,2 detik pada 170MB, serta 5,5 dan 6,3 detik pada 240MB.

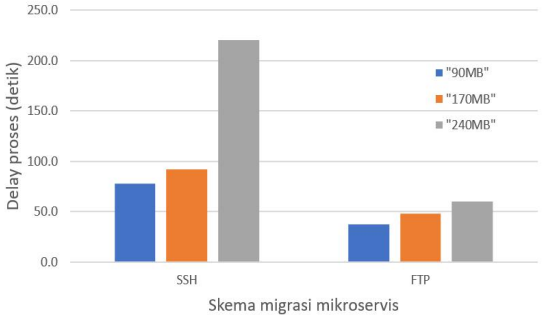
Adapun perbandingan antara penggunaan kedua protokol pengiriman *image*mikroservis (SSH dan FTP) dapat dilihat pada Gambar 6. Dalam gambar tersebut, pembaca dapat melihat dengan

jelas bahwa SSH memiliki selisih *delay* yang cukup signifikan dibandingkan dengan FTP, yakni sekitar 44 detik. Hal tersebut dapat dimaklumi mengingat SSH menggunakan enkripsi-dekripsi untuk menutupi data yang dikirimkan. Secara otomatis hal tersebut akan menambah lama proses pengiriman *image* mikroservis, dari yang awalnya langsung mengirimkan *snapshot image* menjadi mengenkripsi terlebih dahulu baru kemudian mengirimkan *snapshot image* yang terenkripsi. Apalagi SSH menggunakan algoritma enkripsi RSA yang mengandalkan operasi pemangkatan bilangan *integer* yang cukup Panjang, yakni 512, 1024, dan 2048. Sehingga *delay* komputasi pun akhirnya bertambah cukup signifikan dibandingkan tanpa enkripsi RSA, dan bahkan dengan enkripsi simetrik (AES dan DES) sekalipun.



Gambar 7 Perbandingan Performa Migrasi Mikroservis Antara SSH dan FTP pada Ukuran *image*170MB

Pada Gambar 7, perbandingan *delay* akumulatif untuk semua ukuran *image* mikroservis semakin menegaskan bahwa penggunaan SSH sangat berpengaruh pada lamanya migrasi mikroservis. Pada semua skenario ukuran *image* mikroservis, *delay* akumulatif SSH jauh lebih besar dibandingkan dengan *delay* akumulatif FTP. Sehingga rasanya tidak berlebihan jika, *delay* yang cukup besar itu adalah hasil kompensasi dari fasilitas pengiriman *image* mikroservis yang aman.



Gambar 8 Perbandingan Performa Migrasi Mikroservis Secara Akumulatif pada Protokol SSH dan FTP

B. Diskusi dan Interpretasi Hasil

Sebagaimana dijabarkan pada sub-bagian III.A, migrasi mikroservis adalah proses yang sangat bergantung pada besarnya data yang harus dikirimkan. Semakin besar data, maka semakin besar pula biaya pengiriman (*delay*) yang harus dibayar. Oleh karena itu, migrasi mikroservis konvensional sebagaimana diperagakan dalam tulisan ini tidak disarankan diadopsi untuk aplikasi yang bersifat *real-time* dan juga mensyaratkan respon yang cepat saat *handover* dari satu cakupan layanan *fog computing* menuju satu cakupan layanan *fog computing* yang lain.

Adapun penambahan fitur keamanan bisa dipastikan akan menambah besarnya *delay* migrasi mikroservis sebagaimana ditunjukkan pada hasil ekseperimen di atas. Namun, besarnya penambahan *delay* tidaklah selamanya begitu signifikan, karena masih banyak cara dan metode yang dapat digunakan. Yang paling mudah adalah dengan menggunakan algoritma enkripsi simetrik seperti DES dan AES. Pengurangan *delay* bisa juga diaplikasikan dengan *stream cypher*, seperti RC4, A5/1, dan lain sebagainya.

Oleh karena itu, desain sistem migrasi mikroservis sudah semestinya mempertimbangkan besar *delay* yang diharapkan serta tingkat keamanan ataupun resiko serangan pada suatu area tertentu. Walaupun *fog computing* pada awalnya juga ditujukan untung mengurangi potensi serangan dari luar jaringan, namun hal tersebut tidaklah menjadi jaminan bahwa *fog computing* pasti aman. Karena potensi serangan internal masih ada dan bisa jadi lebih berbahaya bila dibandingkan serangan dari luar jaringan / internet. Maka, menurut hemat kami, tingkat resiko suatu arsitektur *fog computing* bersifat unik dan sangat bervariatif serta tergantung dari karakter pengguna di dalam *fog computing* itu sendiri.

Dengan mempertimbangkan hal tersebut, maka pengaplikasian algoritma enkripsi bisa dibuat lebih adaptif dengan bergantung pada penilaian tingkat resiko serangan serta kebutuhan *delay* yang ingin dicapai. Jika dirasa aman, maka pengguna dapat memilih enkripsi yang tidak terlalu kompleks sehingga *delay* yang bertambah cukup kecil jika dibandingkan dengan pengiriman tanpa algoritma enkripsi.

IV. KESIMPULAN DAN SARAN

Migrasi mikroservis pada robot dengan mobilitas tinggi dan cakupan area yang luas adalah sebuah kebutuhan yang tidak dapat dibantah. Migrasi tersebut dibutuhkan agar robot dapat terus beroperasi dengan optimal dengan bantuan komputasi pada *server fog computing* yang cakupan area layanannya terbatas. Hasil eksperimen menunjukkan bahwa komponen yang paling berkontribusi besar terhadap migrasi mikroservis adalah *delay* pengiriman *image* mikroservis dengan hasil eksperimen berkisar antara 30,9 sampai 48,4 detik pada FTP dan 71,2 sampai 208,2 detik pada

SSH. Adapun *delay* pembuatan *checkpoint* serta instalasi *image* mikroservis pada server *fog computing* yang baru tidaklah terlalu signifikan besarnya dan cenderung stabil di selang nilai sekitar 2,4 sampai 6,3 detik.

Untuk penelitian selanjutnya, kami menyarankan agar metode enkripsi yang adaptif dapat diriset sehingga didapatkan skema migrasi mikroservis yang dapat mendukung aplikasi yang bersifat *real-time*.

V. REFERENSI

[1] Kuka, Industrial Robot, Accessed: July 2020, [Online]. Available: https://www.kuka.com/en-de/products/robot-systems/industrial-robots/kr-40-pa

[2] Y. Inoue, F. Kato and S. Tachi, "Master-Slave Robot Hand Control Method based on Congruence of Vectors for Telexistence Hand Manipulation," 2019 IEEE International Symposium on Measurement and Control in Robotics (ISMCR), Houston, TX, USA, 2019, pp. A1-1-1-1-A1-1-1-4, doi: 10.1109/ISMCR47492.2019.8955688.

[3] M. Antony, M. Parameswaran, N. Mathew, S. V.S, J. Joseph and C. M. Jacob, "Design and Implementation of Automatic Guided Vehicle for Hospital Application," 2020 5th International Conference on Communication and Electronics Systems (ICCES), COIMBATORE, India, 2020, pp. 1031-1036, doi: 10.1109/ICCES48766.2020.9137867.

[4] F. Liu, X. Li and Y. Wang, "Design of Automatic Guided Vehicle Motion Control System Based on Magnetic Navigation," 2018 Chinese Control And Decision Conference (CCDC), Shenyang, 2018, pp. 4775-4779, doi: 10.1109/CCDC.2018.8407957.

[5] Silvirianti, A. S. R. Krisna, A. Rusdinar, S. Yuwono and R. Nugraha, "Speed control system design using fuzzy-pid for load variation of automated guided vehicle (AGV)," 2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST), Shenzhen, 2017, pp. 426-430, doi: 10.1109/ICFST.2017.8210549.

[6] P. D. Wibawa, E. Susanto and F. Dewanta, "Maximum allowable time *delay* on networked control system using guaranteed cost method," 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), Bandung, 2016, pp. 91-95, doi: 10.1109/ICCEREC.2016.7814982.

[7] WALDY, Ibnu; RUSDINAR, Angga; ESTANANTO, Estananto. Design and Implementation System Automatic Guided Vehicle (AGV) Using RFID for Position Information. Journal of Measurements, Electronics, Communications, and Systems, [S.l.], v. 1, n. 1, p. D1-6, dec. 2015. ISSN 2477-7986

[8] Nao Robot, Accessed: July 2020, [Online]. Available: https://robots.ieee.org/robots/nao/

[9] Arduino Uno Rev 3, Accessed: July 2020, [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3

[10] Intel Atom Processor, Accessed: July 2020, [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/78475/intel-atom-processor-e3845-2m-cache-1-91-ghz.html

[11] Arm Cortex A35, Accessed: July 2020, [Online]. Available: https://developer.arm.com/ip-products/processors/cortex-a/cortex-a35

[12] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.:A survey and analysis of security threats and challenges," Future Generation Computer Systems, vol. 78, pp. 680 – 698, 2018.

[13] M. Chen and Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network," in IEEE Journal on Selected Areas in Communications, vol. 36, no. 3, pp. 587-597, March 2018, doi: 10.1109/JSAC.2018.2815360.

[14] F. Dewanta and M. Mambo, "A Mutual Authentication Scheme for Secure *fog computing* Service Handover in Vehicular Network Environment," in IEEE Access, vol. 7, pp. 103095-103114, 2019, doi: 10.1109/ACCESS.2019.2931217.

[15] Microservices, Accessed: July 2020, [Online]. Available: https://microservices.io/

[16] Amazon Web Services, Accessed: July 2020, [Online]. Available: https://aws.amazon.com/

[17] Google Cloud Platform, Accessed: July 2020, [Online]. Available: https://cloud.google.com/

[18] Openstack, Accessed: July 2020, [Online]. Available: https://www.openstack.org/

[19] O. Consortium, "Openfog reference architecture for *fog computing*," OpenFog Consortium, Tech. Rep., 2017. [Online]. Available: https://www.openfogconsortium.org/wpcontent/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL-1.pdf

[20] Zaigham Mahmood et al., "*fog computing*: Concepts, Frameworks and Technologies", Springer, 2018

[21] Guy Pujolle, "Software Networks: Virtualization, SDN, 5G and Security", Wiley, 2015

[22] "What is REST", Accessed: July 2020, [Online]. Available: https://restfulapi.net/

[23] "Comparing SOAP vs REST API", Accessed: July 2020, [Online]. Available: https://restfulapi.net/soap-vs-rest-apis/

[24] "Core concepts, architecture and lifecycle", Accessed: July 2020, [Online]. Available: https://grpc.io/docs/what-is-grpc/core-concepts/

[25] F. Dewanta and M. Mambo, "Bidding Price-Based Transaction: Trust Establishment for Vehicular *fog computing* Service in Rural Area," 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 2019, pp. 882-887, doi: 10.1109/PERCOMW.2019.8730830.

[26] SSH.COM, "SSH (Secure Shell)", Accessed: July 2020, [Online]. Available: https://www.ssh.com/ssh/

[27] Pamela Statz, "FTP for Beginners", Accessed: July 2020, [Online]. Available: https://www.wired.com/2010/02/ftp_for_beginners/

[28] E. Ochoa-Jiménez, L. Rivera-Zamarripa, N. Cruz-Cortés and F. Rodríguez-Henríquez, "Implementation of RSA Signatures on GPU and CPU Architectures," in IEEE Access, vol. 8, pp. 9928-9941, 2020, doi: 10.1109/ACCESS.2019.2963826.

[29] Docker, Accessed: July 2020, [Online]. Available: https://www.docker.com/