Analisis Performansi Proses Migrasi Pada Cloud Dengan Menggunakan Container Orchestration

Fauzan Rambang Poetra[1], Sidik Prabowo[2], Siti Amatullah Karimah[3]

[1,2,3] Fakultas Informatika, Universitas Telkom, Bandung, Indonesia
[1]fauzanrambang@students.telkomuniversity.ac.id, [2]pakwowo@telkomuniversity.ac.id,
[3]karimahsiti@telkomuniversity.ac.id

Abstrak

Penggunaan cloud computing sebagai infrastruktur suatu sistem sedang banyak digemari. Dikarenakan cloud computing telah beradaptasi dalam penggunaan berbagai macam aplikasi yang sangat berguna bagi perusahaan dalam mengurangi usaha, biaya, dan waktu dalam pembuatan suatu sistem. Dan dengan virtualisasi berbasis container, proses pembuatan maupun penggunaan sistem tersebut akan semakin mudah. Tetapi pada arsitektur cloud computing, semua layanan terletak di dalam satu sistem yang sama, ketika sistem tersebut mengalami system down, semua layanan tersebut akan terkena dampak yang sama. Oleh karena itu dibutuhkan suatu solusi untuk memindahkan layanan dari satu sistem cloud ke sistem cloud yang lain. Dan untuk mempercepat proses tersebut adalah dengan menggunakan arsitektur Container Orchestration atau arsitektur yang memiliki struktur yang memungkinkan untuk memindahkan layanan ke beberapa container dengan satu controller. Pada penelitian ini dilakukan perbandingan dua jenis arsitektur yaitu arsitektur container orchestration dengan arsitektur container sederhana. Hasil penelitian menunjukan contaner creating time pada arsitektur container orchestration 30x lebih cepat dan 3x lebih sedikit dalam menggunakan resource pada CPU Utilization jika dibandingkan dengan arsitektur container sederhana.

Kata kunci : Cloud, Migrasi, Container, Container Orchestration, Container Creating Time, CPU Utilization

Abstract

The usage of cloud computing as the infrastructure of a system is very popular. Because cloud computing has supported the use of various application that are very useful for companies in saving some costs, effort, and time in making a system. And with container based virtualization, the process of making or using this system will be even easier. However, because of the cloud computing archtecture, all services run on the same system, when the system experiences system down, all of these services will experience it too. Therefore, we need a solution to move services from one system cloud to another named migration process. And to accelerate that process is to use container orchestration architecture or a architecture that has a structure to manage containers with one controller. This resarch compare two types of architecture namely container orchestration architecture with docker architecture in general. The research shown that container creating time on container orchestration architecture is 30x more faster and 3x more efficient in CPU Utilization than docker architecture in general.

Keywords: Cloud, Migration, Container, Container Orchestration, Container Creating Time, CPU Utilization

1. Pendahuluan

Latar Belakang

Penggunaan cloud computing sebagai infrastruktur suatu sistem cukup digemari[1]. Dikarenakan cloud computing telah beradaptasi dalam menjalankan beberapa aplikasi seperti sistem otomatis suatu perusahaan yang sangat berpengaruh terhadap biaya yang dikeluarkan saat pembuatan sistem. Tetapi terdapat beberapa skema cloud, masih terdapat skema yang menerapkan hypervisor-based virtualization sebagai arsitektur utama mereka[2]. Oleh karena itu, berdasarkan penelitian sebelumnya[3], solusi untuk memperbaiki virtualisasi berbasis hypervisor adalah dengan menerapkan container-based virtualization.

Adapun masalah selanjutnya adalah pada arsitektur cloud computing, semua layanan bekerja dalam satu sistem[4]. Ketika sistem tersebut mengalami system down, semua layanan di dalam sistem tersebut akan mengalami hal yang sama. Untuk mengatasi masalah tersebut, dilakukan proses migrasi layanan dari satu sistem cloud ke sistem cloud

yang lain. Untuk mempercepat proses tersebut, dilakukan suatu penerapan skema yang disebut container orchestration[5]. Container Orchestration adalah suatu sistem open source yang berfungsi sebagai skema yang memiliki sistem utama yang berfungsi untuk mengendalikan beberapa container yang berisi layanan agar mempermudah proses scaling, repairing, dan migrating[6].

Pada penelitian terkait[7], telah diuji kemampuan container orchestration dalam hal ini menggunakan kubernetes dalam memperbaiki instansi yang berisikan container yang rusak. Tetapi, dalam penelitan tersebut, peneliti hanya berfokus pada perbandingan waktu yang dibutuhkan ketika instansi tersebut dirusak secara manual dengan dirusak menggunakan traffic generator.

Topik dan Batasan

Berdasarkan latar belakang yang sudah dipaparkan sebelumnya permasalahan yang ditarik adalah:

- Pada arsitektur cloud computing, semua layanan bekerja dalam satu sistem yang sama. Ketika sistem tersebut mengalami system down, semua layanan di dalam sistem tersebut akan mengalami hal yang sama.

Adapun batasan masalah yang terdapat pada tugas akhir ini adalah sebagai berikut :

- Container yang digunakan adalah Docker Container.
- Container Orchestration yang digunakan adalah Kubernetes.
- Parameter yang diukur adalah Container Creating Time dan CPU Utilization.
- Arsitektur yang digunakan adalah Private Cloud.
- Layanan yang digunakan di dalam container adalah layanan video streaming.

Tujuan

Untuk tujuan penelitian yang bisa diambil dari latar belakang adalah :

- Melakukan perbandingan container creating time pada arsitektur container orchestration yaitu kubernetes dengan arsitektur docker container pada umumnya dalam proses migrasi layanan.

Organisasi Tulisan

Urutan penulisan laporan ini adalah Bagian 2 menunjukkan penelitian-penelitian terkait dengan tugas akhir ini. Sistem yang diajukan untuk analisis proses migrasi pada sistem cloud akan dijelaskan di bagian 3. Pada bagian 4 akan didiskusikan mengenai hasil pengujian dan evaluasi sistem. Akhirnya, kesimpulan akan dipaparkan pada bagian 5.

2. Studi Terkait

2.1 Cloud Computing

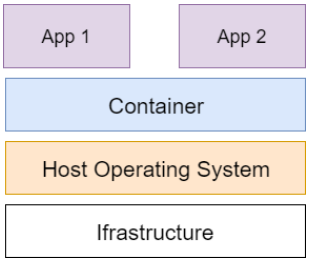
Cloud Computing adalah suatu teknologi yang mana berfungsi untuk menggunakan fungsi dari suatu komputer baik fungsi penyimpanan, aplikasi, ataupun fungsi-fungsi komputer lainnya melalui cloud platform dengan menggunakan internet[8]. Komputer penyedia jasa Cloud Computing berkerja dengan cara memiliki banyak server yang saling terhubung satu sama lain dengan memanfaatkan aplikasi virtual yang dapat menghasilkan suatu komputasi dan penyimpanan yang berskala besar yang mana dapat diakses oleh pengguna yang terhubung dengan sistem cloud tersebut.

2.2 Virtualisasi

Virtualisasi adalah teknologi dimana dapat menggunakan hardware dari Information Technology (IT) secara maksimal. Teknologi ini memungkinkan pengguna untuk saling berbagi sumber daya pada kondisi apapun[9]. Adapun dalam pengerjaannya, terdapat 2 basis utama yaitu Container-based Virtualization dan Hypervisor-based Virtualization[10].

2.2.1 Container-based Virtualization

Container adalah bentuk evolusi dari teknologi Hypervisor-based Virtualization yaitu bentuk virtualisasi yang dapat berbagi akses sistem operasi tanpa memerlukan mesin virtual lagi. Dikarenakan semua aplikasi yang berjalan sudah berbentuk container image[11].

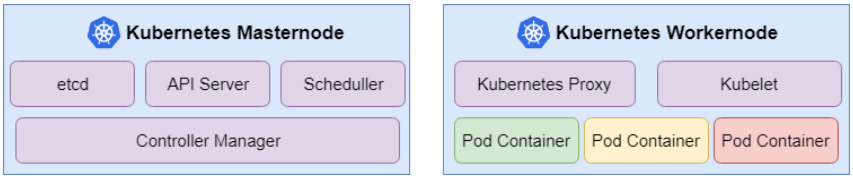


Gambar 1. Arsitektur Container

2.3 Kubernetes

Di jaman internet sekarang ini pengguna mengharapkan suatu layanan yang dapat berjalan selama 24/7 dan developer mengharapkan suatu layanan yang dapat di update kapanpun[12]. Oleh karena itu, google dan komunitas dari kubernetes itu sendiri menciptakan sistem yang bernama Kubernetes. Kubernetes adalah sebuah teknologi yang berguna untuk manajemen container. Kubernetes berfungsi sebagai pengendali dari container dengan cara membuat kubernetes cluster yang nantinya dapat berisi berbagai pod container. Dalam penerapannya, kubernetes menggunakan dua node (Master dan Worker). Master Node terdiri dari API, Scheduller, Controller[13]. Dan untuk Worker Node terdiri dari Kubelet, kub-proxy, Pod Container[14]. Adapun kegunaannya adalah sebagai berikut :

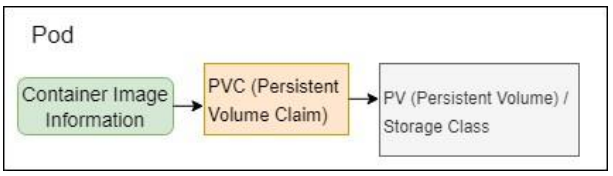
1. API Server berfungsi sebagai penghubung antara node master dengan node worker.
2. Scheduller bertanggung jawab untuk proses instalasi container pada cluster.
3. Controller sebagai kepala dari master node yang berfungsi sebagai pengambilan keputusan dari keadaan yang dikirimkan oleh Worker Node.
4. Kubelet berfungsi untuk mengirimkan state yang ingin dijalankan oleh Master Node.
5. Kub-Proxy berfungsi untuk menghubungkan Kubelet dengan Container Engine.
6. Pod Container adalah instansi di dalam kubernetes cluster yang berisi container image.



Gambar 2. Arsitektur Kubernetes

2.3.1 Pod Lifecycle

Pod Lifecycle adalah salah satu fitur yang disediakan oleh kubernetes yang memungkinkan pod untuk memperbaiki dirinya sendiri dengan bantuan Controller Manager yang berada pada Masternode Kubernetes[15]. Adapun detail prosesnya adalah dengan memasukan container image kedalam pod, informasi container image tersebut akan tersimpan di dalam storage class yang berada di dalam pod tersebut. Ketika pod tersebut rusak, storage class tersebut tidak akan ikut rusak. Lalu controller manager meminta informasi container image sebelumnya melalui Persistent Volume Claim (PVC) untuk di teruskan ke storage class dan dilanjutkan dengan proses pembuatan kembali pod baru[16].



Gambar 3. Proses pengambilan informasi Container Image

Adapun fungsi entitas pada Gambar 3 adalah sebagai berikut :

1. Container Image Information adalah informasi container image yang berada di dalam pod
2. Persistent Volume Claim (PVC) adalah objek yang terhubung dengan Storage Class / Persistent Volume (PV) untuk menyimpan Container Image Information kedalam Storage Class
3. Storage Class / Persistent Volume (PV) adalah tempat penyimpanan yang berisi tentang informasi setiap Container Image.

2.4 Penelitian Terkait

Pada penelitian sebelumnya[3], telah membandingkan teknologi hypervisor-based dengan container-based. Hasil yang didapatkan adalah container-based lebih ringan jika dibandingkan dengan hypersivor-based dikarenakan container-based tidak membutuhkan periferal lain dalam menjalankan aplikasi di dalamnya dan container-based memiliki sistem yang terisolasi dengan baik. Tetapi dari penelitian di atas tidak dikatakan bahwa manakah yang lebih baik ketika menangani system down. Sedangkan sistem virtualisasi dapat dikatakan cukup rentan apabila terkena serangan.

Pada penelitian berikutnya[5], telah dilakukan pembuatan sistem yang bernama Hybrid Cloud Service Broker (Hybrid CSB) yang berguna untuk mengintergrasikan private dengan public cloud. Tetapi dalam penelitian ini, tidak dijelaskan secara pasti tujuan intergrasi tersebut. Sedangkan proses ini berguna untuk proses migrasi layanan dari satu cloud ke cloud yang lain yang berguna dalam menanggulangi layanan di dalam cloud mati.

Pada penelitian yang lain[7], telah dibandingkan performansi container tanpa Container Orchestration dengan yang menggunakan Container Orchestration. Acuan dari penelitian ini adalah CPU Performance dan Disk I/O Performance. Pada penelitian ini dikatakan bahwa CPU Performance dan Disk I/O Performance dari container dengan Container Orchestration memiliki performansi yang dikatakan lebih efisien dikarenakan container dengan Container Orchestration memiliki karakteristik high density dan high elactisity. Sehingga penelitian ini dapat dijadikan acuan dalam menjalankan Container Orchestration dalam hal ini adalah kubernetes.

Pada kasus berikutnya[17], telah diuji kemampuan dari kubernetes dalam mengembalikan node-node yang hilang. Tetapi dari hanya membandingkan container creating time ketika pod yang berisi container di hapus secara manual dan dengan menggunakan traffic generator. Tidak membandingkan penggunaan arsitektur container orchestration dengan tidak menggunakannya. Jadi belum dapat dikatakan kubernetes sebagai solusi untuk mempercepat proses repair time. Untuk memperjelas perbandingan setiap penelitian, dapat dilihat pada tabel berikut:

Tabel 1. Perbandingan Penelitian

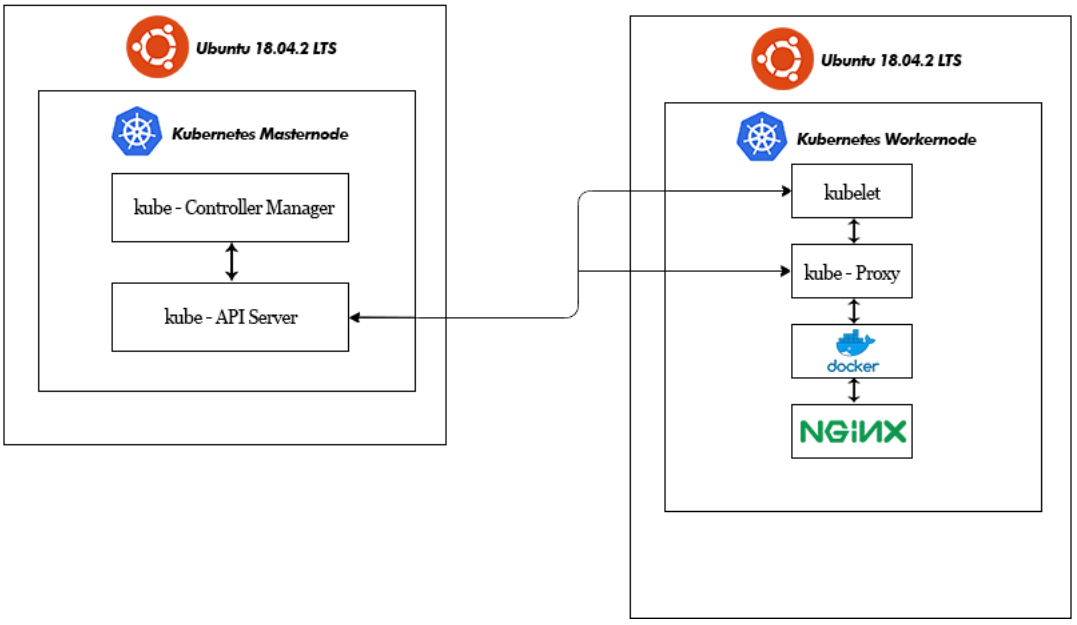
Judul Paper	Tahun	Sistem	Metode	Output
Hypervisor- vs. Container-based Virtualization[3]	2016	Hypervisor-based, Container-based	Single VM, Single Container	-
Approach for Cloud Recommendation and Integration to Construct User-Centric Hybrid Cloud[5]	2017	Cloud System	Hybrid Cloud Service Broker (Hybrid CSB)	Method Units dan Script Units
The Performance Analysis of Docker and Rkt Based on Kubernetes[7]	2017	Container-based	Container Orchestration, Kubernetes	CPU Performance dan Disk I/O Performance
Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned[17]	2018	Container-based	Container Orchestration, Kubernetes	Repair Time
Analisis Performansi Proses Migrasi Pada Cloud Dengan Menggunakan Container Orchestration (Judul Penelitian Ini)	2018	Container-based	Container Orchestration, Migration Service	Container Creating Time dan CPU Utilization

Berdasarkan tabel diatas, akan dilakukan perbandingan dua arsitektur yaitu arsitektur container orchestration dan arsitektur docker pada umumnya. Dengan mengacu pada penelitian sebelumnya[7], parameter yang digunakan adalah Container Creating Time dan CPU Utilization.

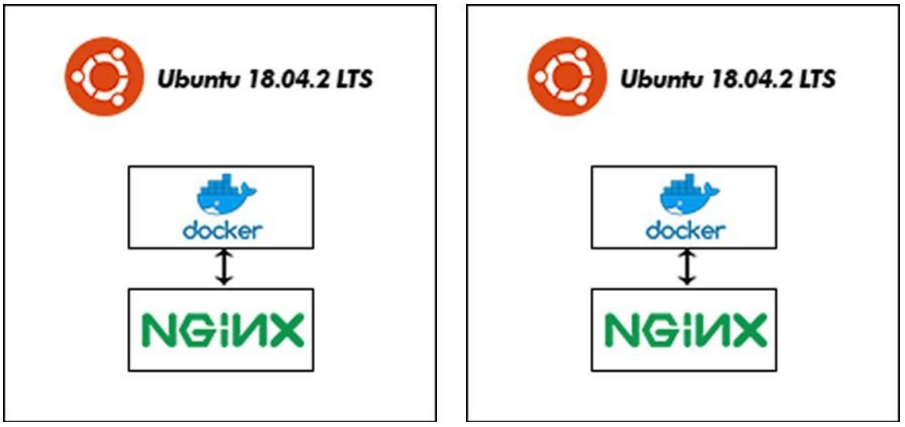
3. Sistem yang Dibangun

3.1 Topologi Sistem

Penelitian ini memiliki dua sistem arsitektur utama, yaitu dengan menggunakan arsitektur Kubernetes dengan arsitektur docker sederhana. Arsitektur pertama dimulai dengan memasang Kubernetes Cluster yaitu Master Node di dalam System Host 1 dan Worker Node di dalam System Host 2. Lalu memasang Docker Container Engine yang dilanjutkan dengan memasang container image nginx Video Streaming Server ke dalam container yang sudah dibuat pada Worker Node. Arsitektur kedua dimulai dengan memasang container engine pada system host yang dilanjutkan dengan memasang container image di dalam container engine yang sudah dibuat.



Gambar 4. Arsitektur Kubernetes



Gambar 5. Arsitektur Docker

3.2 Skenario Pengujian

Adapun parameter yang diukur pada penelitian ini adalah Container Creating Time dan CPU Utilization. Dengan melakukan skenario penghapusan container secara manual, dilakukan perbandingan waktu yang dibutuhkan dalam pembuatan kembali container yang hilang. Dalam hal ini menggunakan bash script image_migrate dan load_image yang berfungsi untuk menjalankan proses migrasi container pada arsitektur docker sederhana dan menggunakan perintah delete pod kubernetes pada arsitektur container orhcestration. Pada CPU Utilization dilakukan perbandingan persentase penggunaan resouce pada host ketika melakukan migrasi.

3.2.1 Bash Script

Adapun bash script yang akan digunakan di pengujian ini adalah image_migrate dan load_image. Adapun komponen yang ada di image_migrate script adalah sebagai berikut :

1. Masukkan nama container yang ingin dipindahkan
2. Menghentikan container yang memiliki nama container yang pada tahap satu di isikan
3. Setelah container terhenti, dilakukan proses penyimpanan container tersebut menjadi arsip file lokal.
4. Lalu dilanjutkan dengan memindahkan arsip file tersebut ke host tujuan.

Setelah image_migrate script berjalan, dilanjutkan dengan menjalankan load_image script pada host tujuan. Adapun detail script tersebut adalah sebagai berikut :

1. Memasukkan nama arsip file yang sudah di pindahkan dari host pertama.
2. Setelah itu, dilakukan proses docker load untuk menjadikan arsip file tersebut menjadi container image kembali.
3. Lalu dilanjutkan dengan proses menjalankan container image tersebut.

3.2.2 Container Creating Time

Untuk menghitung Container Creating Time pada kedua arsitektur, dilakukan uji coba sebanyak tiga puluh kali. Pada skema container orchestration dijalankan perintah kubectl delete pod namespace dan menjalankan bash script pada skema docker yang berisi kumpulan perintah docker stop, commit, save, scp filename host destination:file destination, docker load, sampai docker run.

3.2.3 CPU Utilization

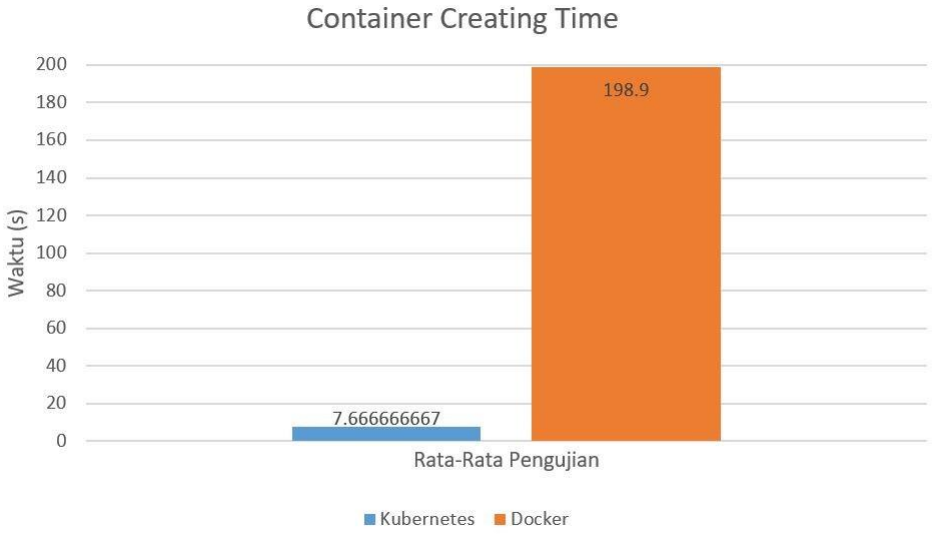
Untuk menghitung persentase CPU Utilization pada kedua arsitektur, dilakukan uji coba sebanyak sepuluh kali. CPU Utilization pada kedua arsitektur dihitung dengan cara menjalankan command TOP yang disediakan oleh sistem operasi ubuntu itu sendiri. Setelah merekam penggunaan resource, dilakukan penjumlahan persentase antara kedua host pada kedua arsitektur(Masternode dan Workernode pada Kubernetes dan Host 1 dan Host 2 pada docker.

4. Evaluasi

Bagian ini menampilkan hasil analisis penulis. Seperti yang dijelaskan sebelumnya, parameter yang dipilih untuk diukur adalah Container Creating Time.

4.1 Container Creating Time

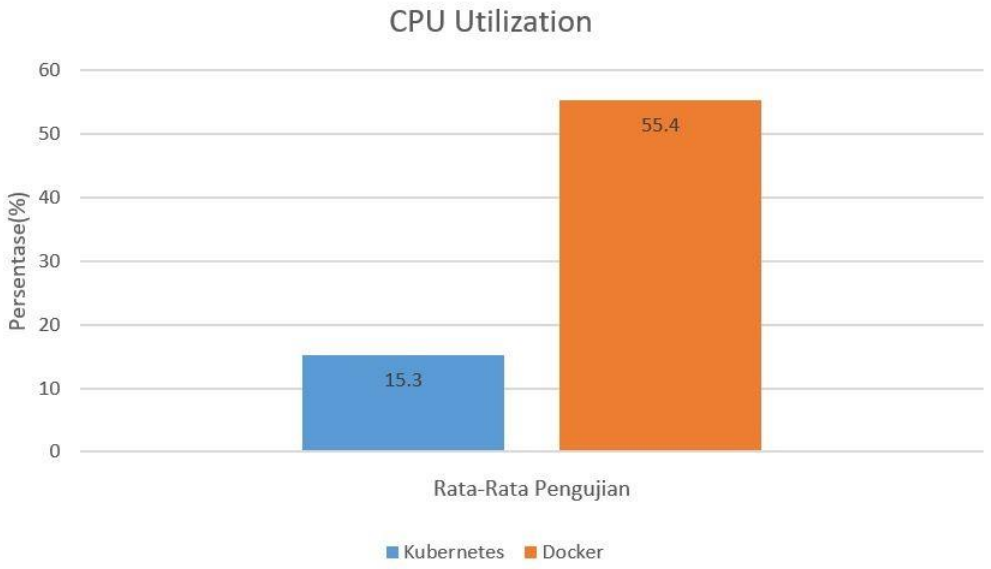
Terlihat dari gambar 6, perbandingan waktu container creating time pada arsitektur kubernetes dengan arsitektur docker sederhana sangat jauh berbeda. Dikarenakan pada arsitektur kubernetes terdapat pod yang memiliki fitur pod lifecycle yang memungkinkan pod yang berisikan container memiliki kemampuan untuk membuat instansi baru dengan menduplikasi informasi docker image di dalam container pada instansi sebelumnya yang tersimpan di dalam storage class. Berbeda halnya arsitektur docker pada umumnya yang mengharuskan docker container yang sedang berjalan harus diberhentikan terlebih dahulu, lalu container tersebut dijadikan docker image yang akan disimpan menjadi docker image file berbentuk .tar file yang dilanjutkan dengan menjalankan perintah secured copy files (SCP) melalui ssh yang saling terhubung antara dua host komputer yang berbeda dan dilakukan secara manual.



Gambar 6. Rata-Rata Perbandingan Container Creating Time

4.2 CPU Utiization

Terlihat dari gambar 7, perbandingan persentase CPU Utilization antara arsitektur kubernetes dengan arsitektur docker cukup berbeda. Dikarenakan pada arsitektur kubernetes penggunaan resource pada host memakan waktu yang cukup sedikit. Berbeda halnya dengan arsitektur docker yang memerlukan waktu yang cukup lama untuk sekali proses migrasi. Ini menyebabkan penggunaan resource terus berjalan sampai proses migrasi selesai. Dengan hasil persentase penggunaan resource pada arsitektur kubernetes adalah 15.3% dan 55.4% pada arsitektur docker sederhana, dapat dikatakan bahwa arsitektur kubernetes lebih kecil 3x lipat dalam penggunaan resource.



Gambar 7. Rata-Rata Perbandingan CPU Utilization

5. Kesimpulan

5.1 Kesimpulan

Hasil analisis dari seluruh pengujian yang dilakukan dalam penelitian tugas akhir ini dapat menarik kesimpulan sebagai berikut:

1. Container Creating Time

   Proses migrasi container dengan arsitektur kubernetes dengan cara melakukan pembuatan kembali pod yang berisi container menghasikan waktu sebesar 7,667 detik menjadikan waktu yang dibutuhkan dalam membuat kembali container lebih cepat dibandingkan dengan arsitektur docker yang menghasilkan waktu 198,9 detik. Hal ini disebabkan karena kubernetes memiliki fitur pod lifecycle. Berbeda halnya arsitektur docker yang membutuhkan dua komputer host yang berbeda untuk proses migrasi.

2. CPU Utilization

   Dengan menggunakan arsitektur kubernetes, rata-rata penggunaan resource CPU hanya sampai 15.3% yang menjadikan resource yang digunakan dalam pembentukan kembali pod yang berisi container lebih kecil jika dibandingkan dengan arsitektur docker sederhana yang mencapai 55.4%. Hal ini disebabkan penggunaan resource pada host akan berbanding lurus dengan waktu yang diperlukan untuk proses migrasi. Semakin banyak waktu yang diperlukan dalam proses migrasi, semakin banyak pula resource yang digunakan dalam pembentukan kembali container.

5.2 Saran

Adapun saran yang diberikan untuk pengembangan penelitian berikutnya adalah :

1. Menambahkan parameter yang diuji.
2. Menambahkan layanan yang diuji.
3. Mencoba arsitektur cloud lainnya seperti Hybrid Cloud.

Daftar Pustaka

[1] Yu Kaneko and Toshio Ito. A reliable cloud-based feedback control system. IEEE International Conference on Cloud Computing, CLOUD, pages 880–883, 2017.

[2] Alex Meade. Understanding Google Cloud Platform : Architecture. Global Knowledge, pages 1–9, 2017.

[3] Michael Eder and Holger Kinkelin. Hypervisor- vs. Container-based Virtualization. Network Architectures and Services, (July), 2016.

[4] Simplilearn. Cloud Computing Architecture, 2019.

[5] Joonseok Park, Donggyu Yun, Ungsoo Kim, and Keunhyuk Yeom. Approach for Cloud Recommendation and Integration to Construct User-Centric Hybrid Cloud. Proceedings - 2nd IEEE International Conference on Smart Cloud, SmartCloud 2017, pages 24–32, 2017.

[6] Scot Marvin. What is Kubernetes, 2019.

[7] Xiao Lan Xie, Peng Wang, and Qi Wang. The performance analysis of Docker and rkt based on Kubernetes. ICNC-FSKD 2017 - 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, pages 2137–2141, 2018.

[8] Amazon. What is Cloud Computing, 2019.

[9] Vitor Goncalves da Silva, Marite Kirikova, and Gundars Alksnis. Containers for Virtualization: An Overview. Applied Computer Systems, 23(1):21–27, 2018.

[10] VMWare. What is a Hypervisor, 2019.

[11] Docker. What is a Container, 2019.

[12] Shavidissa. Learn Kubernetes Basics, 2019.

[13] Pei-Hsuan Tsai, Hua-Jun Hong, An-Chieh Cheng, and Chen-Hsin Hsu. Distributed Analytics in Fog Computing Platforms Using TensorFlow and Kubernetes. IEEE International Conference on Cloud Computing, CLOUD, 2017.

[14] Carla Schroder. What Makes Up a Kubernetes Cluster, 2017.

[15] Adebayo Adesanya. Pod Lifecycle, 2019.

[16] Joep Piscaer. Kubernetes in the Enterprise. page 21, 2018.

[17] Leila Abdollahi Vayghan, Mohamed Aymen Saied, Maria Toeroe, and Ferhat Khendek. Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned. IEEE International Conference on Cloud Computing, CLOUD, 2018-July:970–973, 2018.