ANALISIS PENGGUNAAN MEMORI SISTEM PADA MIGRASI APLIKASI DALAM LINUX CONTAINER (LXD) MENGGUNAKAN LXD API

ANALYSIS OF SYSTEM MEMORY UTILIZATION OF APPLICATION MIGRATION IN LINUX CONTAINER (LXD) USING LXD API

Amalia Intan Safura[1], Adityas Widjajarto, S.T.,M.T.[2], Avon Budiono, S.T.,M.T.[3]

[1,3]Prodi S1 Sistem Informasi, Fakultas Rekayasa Industri, Universitas Telkom
[1]amaliafura@telkomuniversity.ac.id, [2] adtwjrt@telkomuniveristy.co.id, [3]avonbudi@telkomuniversity.ac.id

Abstrak

Teknologi *virtual* yang saat ini digemari konsumen salah satunya yaitu *container*. *Container* menyediakan fleksibilitas secara keseluruhan, dengan dapat langsung berjalan diatas sistem operasi tanpa adanya menggunakan *hypervisor*. Salah satunya LXD *Container*, pada penggunaan biasanya ditemukan kondisi responsif ataupun kondisi tidak responsif yang salah satunya disebabkan kebutuhan akan *resource* yang tidak tepat. LXD sebagai wadah aplikasi yang menyediakan layanan. Maka dari itu untuk mengetahui bagaimana penggunaan *resource* serta ancaman terhadap rusaknya program pada LXD , maka dilakukan penelitian menggunakan LXD yang berfungsi sebagai wadah aplikasi untuk menjalankan aplikasi layanan yang disediakan untuk kebutuhan *user*, dan dilakukan pengujian serta analisis terhadap penggunaan memori sistem pada LXD lalu menjalankan proses migrasi dengan LXD API pada LXD dari *platform* satu ke *platform* lainnya. Selanjutnya mengukur penggunaan memori pada waktu sebelum, dan setelah migrasi yang nantinya dapat menjadi acuan dalam penggunaan LXD *Container* yang tepat dan sesuai berdasarkan kebutuhan *user*. Pada penelitian ini dihasilkan bahwa kapasitas memori dan mengingkatnya *core* pada *processor* tidak berpengaruh secara signifikan pada jangka waktu proses migrasi LXD *container* dilakukan. Namun penggunaan memori sangat berpengaruh pada kondisi responsif dan tidak responsifnya sistem disaat mengakses banyak aplikasi layanan yang terdapat pada LXD *container*.

Kata kunci : Migrasi, Linux Ubuntu, LXD, Memori.

Abstract

Virtual technology that is currently favored by consumers is one of them, namely container. Container provides overall flexibility, by being able to directly run on the operating system without using a hypervisor. One of them is LXD Container, the use is usually found responsive conditions or unresponsive conditions, one of which is due to the need for inappropriate resources because it does not know how the characteristics of LXD as a container application that provides services. Therefore to find out how to use the right resource and also to know how the characteristics of LXD are operated as well as the threat to program damage in LXD, then research is done using LXD which functions as an application container to run service applications provided for user needs, and test and analysis of system memory usage in LXD and then run the migration process with LXD API on LXD from one Platform to another. Furthermore, measuring memory usage at the time before, and after migration to find out how the characteristics of LXD can later become a reference in the use of the right and appropriate LXD Container based on user needs. In this study it was produced that the memory capacity and increasing core on the processor did not significantly influence the duration of the LXD container migration process. However, memory usage is very influential on the responsive and unresponsive condition of the system when accessing many service applications contained in the LXD container.

Keywords: Migration, Linux Ubuntu, LXD, Memory

1. Pendahuluan

Saat ini teknologi berbasis *virtual* telah banyak menarik minat dan perhatian dari konsumen pada berbagai bidang seperti bidang industri, pendidikan maupun penelitian, salah satunya ditujukan dalam memenuhi kebutuhan memproses program data dan mengakses data semudah mungkin tanpa harus menambah biaya mahal dalam pengoperasiannya. Teknologi virtualisasi dengan *container* menawarkan banyak fitur-fitur yang dibutuhkan seperti lebih hemat ruang, hemat tenaga, isolasi, akuntabilitas, pengalokasian dan pembagian sumber daya yang tepat dan lain sebagainya. Salah satu *container* yang terdapat pada Linux adalah LXD yang merupakan *container* generasi selanjutnya dari LXC. Dengan LXD, sebuah host dapat menjalankan banyak sistem *container* LXC hanya daemon sistem tunggal, dan dapat memiliki keamanan yang lebih baik yang mana dapat memanfaatkan fitur kemanan tingkat host agar *container* lebih aman. [4]. Banyak keuntungan dan kefektifitas yang disediakan oleh LXD, maka dalam penggunaannya, pengguna dapat mencegah datangnya

ancaman atau insiden yang tidak terduga seperti tidak responsifnya sistem saat berjalan, rusaknya program yang dapat menganggu operasional pada LXD dengan melakukan pencegahan seperti menggunakan *resource* yang sesuai kebutuhan *user* dan melakukan migrasi LXD dengan *resource* yang tepat. Migrasi merupakan proses untuk memindahkan suatu *virtual machine* ke host fisik yang lain sehingga data-data dapat dibackup. Umumnya, pada sistem akan terjadi kerusakan atau *error*, yang disebabkan oleh faktor internal ataupun eksternal. Melakukan migrasi pada *container* LXD adalah hal yang tepat dan penting dalam meminimalisir hilangnya data serta aplikasi-aplikasi yang dibutuhkan oleh pengguna dan dapat meminimalisir downtime pada sistem.

Berdasarkan uraian tersebut, maka pada penelitian tugas akhir ini akan dilakukan proses migrasi aplikasi dalam LXD dari satu host (*Platform* 1) ke host (*Platform* lain) lainnya menggunakan LXD API[6] dan melakukan analisis bagaimana penggunaan memori sistem pada proses tersebut dengan mengukur penggunaan memori pada waktu disaat dilakukan migrasi.

2. Landasan Teori

2.1 Penjelasan LXD

LXD merupakan generasi pengaturan *system container* selanjutnya dari LXC yang menawarkan *user experience* yang mirip *virtual machine* namun menggunakan linux *container* sebagai gantinya. LXD merupakan REST API yang terhubung ada liblxc, yaitu perpustakaan perangkat lunak LXC. LXD tidak memakai *template* seperti halnya yang terdapat pada LXC. [2] Dalam membuat linux *container* LXD menggunakan *image* yang memiliki ukuran file lebih kecil dibandingkan dengan LXC. Keamanan pada LXD cukup baik karena memiliki keamanan selevel dengan keamanan pada sistem operasi jka dibandingkan dengan LXC. [4]

2.2 Migrasi

Migrasi adalah proses *transfer* suatu teknologi *virtual* tanpa menganggu operasi normalnya. Migrasi memungkinkan porting teknologi *virtual* dan dilakukan secara sistematis untuk memastikan sehingga meminimalisir *downtime* pada operasionalnya.

Pada umumnya, migrasi dilakukan ketika komputer fisik/server host memerlukan *maintenance*, pembaharuan, dan atau untuk beralih antar host yang berbeda. Untuk memulainya, data dalam memori teknologi *virtual* ini pertama dipindahkan ke mesin fisik targetnya,. Setelah proses penyalinan memori telah selesai, status sumber daya operasioanal yang berupa CPU, memori, dan penyimpanan dibuat pada mesin tujuan. Selanjutnya, teknologi *virtual* ini akan di*suspend* pada site asli dan dipindah serta diinisiasi pada mesin tujuan bersama aplikasi yang telah dipasang.

2.3 Memori

Memori merupakan perangkat yang bertanggungjawab untuk menyimpan data dan aplikasi secara sementara ataupun permanen. Dengan memori memungkinkan pengguna untuk menyimpan informasi yang tersimpan dalam komputer. Jika tanpa adanya memori, maka *processor* tidak akan dapat menemukan tempat yang diperlukan dalam menyimpan proses dan perhitungan.

Memori utama dibagi menjadi dua jenis, sebagai berikut:

1. *Random Access* Memori (RAM)
   Untuk tipe jenis memori ini, bertanggung jawab dalam menyimpan data sementara sehingga dapat langsung di akses *processor* sebagian dan bila diperlukan.
2. *Read Only* Memori (ROM)
   Tidak seperti RAM, ROM merupakan penyimpanan permanen. Untuk memori ROM ini, akan tetap aktif dan berjalan walaupun sistem dimatikan.

2.5 *Swap Memory*

Pada dasarnya *swap memory* merupakan *memory virtual* yang digunakan saat memori fisik (RAM) telah penuh, dan akan ditangani oleh *swap*. *Swap* memori yaitu ruang yang terdapat pada *hardisk* yang dijadikan sebagai *virtual* memori. *Swap* memori dapat berguna jika memiliki kapasitas memori utama/fisik yang minim. Biasanya ukuran *swap* memori dua kali lipat dari ukuran memori fisik yang terpasang, namun hal tersebut tidak berlaku pada komputer saat ini. Justru dengan menggunakan ukuran memori yang dua kali lipat atau lebih besar dibandingkan memori utama maka menjadi sebuah pemborosan pada *hardisk.* Jika sebuah aplikasi lebih memakai banyak memori *swap* dapat menurunkan kineja aplikasi saat berjalan karena tidak dapat mengakses data dengan kecepatan yang sama seperti memori fisik, yang menyebabkan apikasi lambat dalam operasionalnya.

2.6 LXD API

Merupakan protokol yang menyediakan semua komunikasi antara LXD *container* dan kliennya . REST API dapat diakses melalui soket unix lokal atau melalui HTTPs. Pada *unix socket* tidak mebutuhkan autentifikasi beda halnya saat kita melalui https membutuhkan autentifikasi kata sandi *client.* [4]

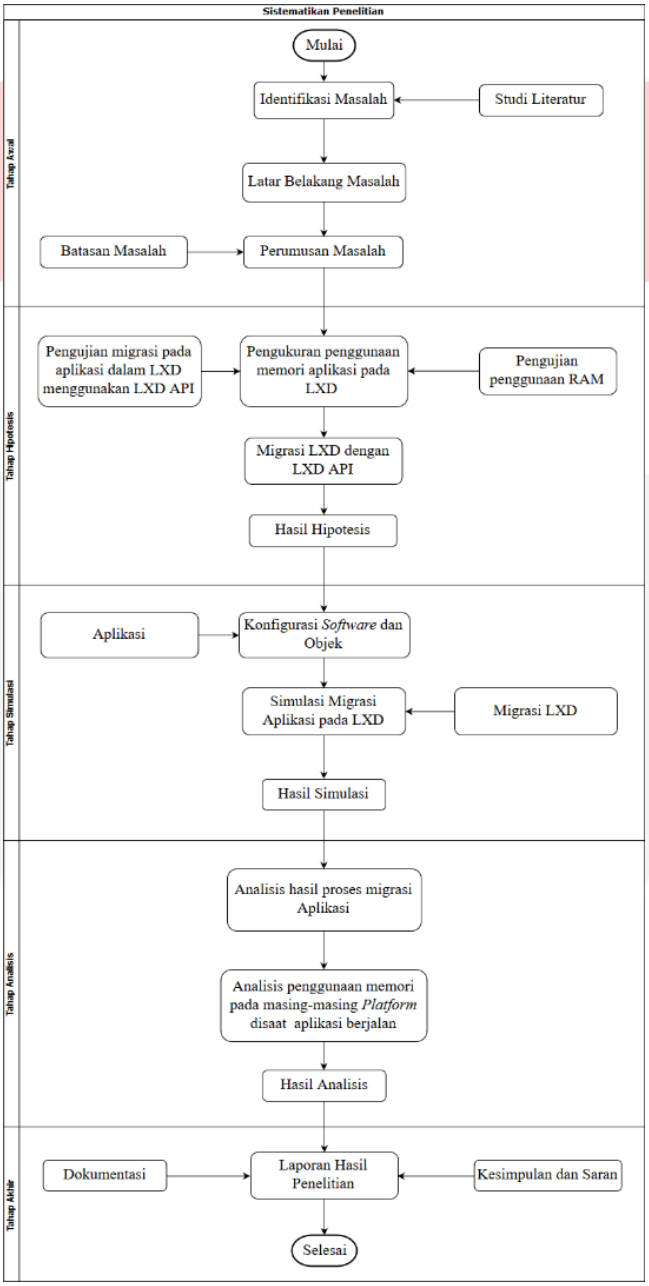
2.7 Htop

Htop adalah salah satu *monitoring tools* dan pengukuran proses yang terjadi dalam sistem operasi Linux dengan distro Debian atau Ubuntu. Htop tidak jauh berbeda dengan *command*, namun htop lebih interaktif dan lebih mudah digunakan dan htop lebih *user friendly* karena telah terdapat dukungan mouse dalam operasionalnya. Pada htop dtampilkan penggunaan dari CPU, memori, *swap* dan *resource* yang lain.

3. Metodologi Penelitian

3.1. Sistematika Penelitian

Pada sub bab sistematika penelitian dikemukakan beberapa langkah yang dilakukan dalam mengerjakan penelitian ini yang bersifat terstruktur serta sistematis. Penelitian ini dimulai dari dilakukannya identifikasi masalah hingga laporan hasil penelitian. Sistematika penelitian analisis penggunaan memori sistem pada migrasi aplikasi pada Linux *Container* (LXD) menggunakan LXD API.



Gambar 1 Sistematika Pengujian

4. Pengujian Sistem dan Skenario Pengujian

4.1. Instrumen Pengujian

Sistem yang dibangun merupakan implementasi dalam bentuk simulasi pengujian yang dilakukan pada *virtual machine* dengan sistem operasi Linux Ubuntu. Sebelum, dilakukannya simulasi pengujian akan didefinisikan beberapa instrumen pengujian yang bertujuan untuk mendukung jalannya simulasi pengujian yang dilakukan, instrumen pengujian ini berupa alat-alat yang dibutuhkan dalam simulasi pengujian, dapat berupa instrumen perangkat lunak (*software*) ataupun perangkat keras (*hardware*).

a. Instrumen Perangkat Keras

Instrumen perangkat keras adalah alat-alat fisik yang digunakan dan dibutuhkan dalam menjalankan skenario pengujian. Berikut beberapa instrumen perangkat keras yang digunakan:

Tabel 1. Instrumen Perangkat Keras

No.	Perangkat	Model/Seri	Spesifikasi
1.	*Platform* 1	Asus A456U	1. *Processor*: Intel® Core™ i5-7200U *Processor* (2.5 GHz, 3M *Cache*) *up to* 3.10 GHz 2. Memori: 8GB DDR4 3. GPU: *Intel HD Graphics* 620 dan NVIDIA GeForce GT930MX 2GB 4. Hardisk: 1TB HDD 5. OS: Windows 10 64-bit
2.	*Platform 2*	Asus X550V	1. *Processor*: Intel® Core™ i5-7700U *Processor up to* 3.8 GHz 2. Memori: 8GB DDR4 3. GPU: Intel HD *Graphics* 630 dan NVIDIA GeForce GT950MX 2GB 4. Hardisk: 1TB HDD 5. OS: Windows 10 64-bit

b. Instrumen Perangkat Lunak

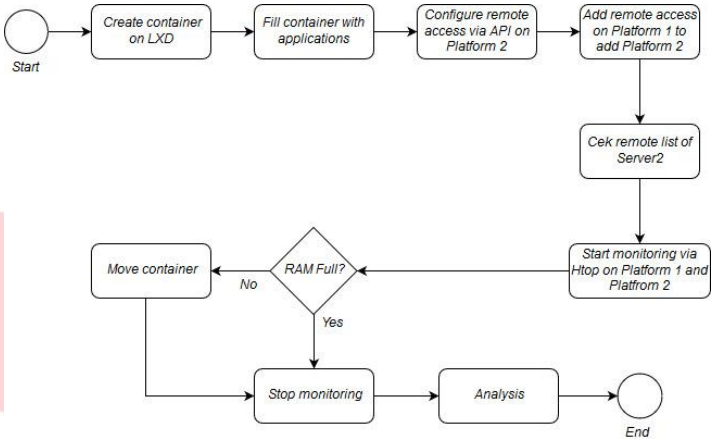
Instrumen perangkat lunak merupakan program (*software*) yang digunakan pada simulasi pengujian, yang bertujuan untuk mendukung jalannya pengujian sesuai dengan aspek yang dibutuhkan. Berikut ini instrument perangkat lunak yang digunakan:

Tabel 2. Instrumen Perangkat Lunak

No.	Tipe	Nama Software	Versi
1.	Sistem Operasi	Linux Ubuntu	16.04.03 LTS
2.	Objek Migrasi	*Container*	2.0.11
		Odoo	Odoo 10
		Nginx	1.10.0
		Apache	2.2.15
		WordPress	3.7.1
		PhpMyAdmin	5.2
		MySql	5
3.	*Third Party Software*	LXD *Container*	2.0.11
		VMware Workstation Pro	12.5.4build-5192485
		htop	2.6.3

4.2. Skenario Pengujian

Dalam melakukan pengujian, diperlukan adanya skenario pengujian yang berfungsi untuk gambaran pengujian yang akan dilakukan. Yaitu pengujian migrasi aplikasi pada LXD dengan menggunakan LXD API. Pengujian ini dilakukan untuk mengetahui bagaimana proses migrasi aplikasi pada LXD dengan menggunakan LXD API dan pengaruh penggunaan memori yang terkait pada proses tersebut.



Gambar 2 Skenario Pengujian

4.2.1 Skenario 1: Migrasi Aplikasi Platform 1 – Platform 2 (Satu Arah)

Pada skenario pertama, akan dilakukan pengujian proses migrasi aplikasi pada LXD dengan menggunakan LXD API. Terdapat dua *platform* yang saling terhubung yaitu *Platform* 1 dan *latform* 2. *Platform* 1 melakukan pemindahan *container*, sedangkan *platform* 2 melakukan penerimaan LXD. Pengujian ini juga menghitung dan mengukur penggunaan memori RAM pada aplikasi dalam LXD sesaat dimigrasikan.

4.2.2 Skenario 2: Migrasi Aplikasi Platform 2 – Platform 1 (Satu Arah)

Pada skenario kedua, akan dilakukan pengujian proses Migrasi aplikasi pada LXD dengan menggunakan LXD API. Terdapat dua *platform* yang saling terhubung yaitu *Platform* 1 dan *Platform* 2. *Platform* 2 melakukan pemindahan *container*, sedangkan *Platform* 1 melakukan penerimaan LXD. Pengujian ini juga menghitung dan mengukur penggunaan memori RAM pada aplikasi dalam LXD sesaat dimigrasikan.

4.2.3 Migrasi Aplikasi Dua Arah I

Pada skenario ketiga, akan dilakukan pengujian proses migrasi aplikasi dalam LXD dengan menggunakan LXD API. Terdapat dua *platform* yang saling terhubung yaitu *Platform* 1 dan *Platform* 2. *Platform* 1 dan *Platform* 2 melakukan pemindahan dan penerimaan LXD secara bersamaan. Pada pengujian ini menghitung dan mengukur penggunaan memori RAM pada aplikasi dalam LXD sesaat dimigrasikan.

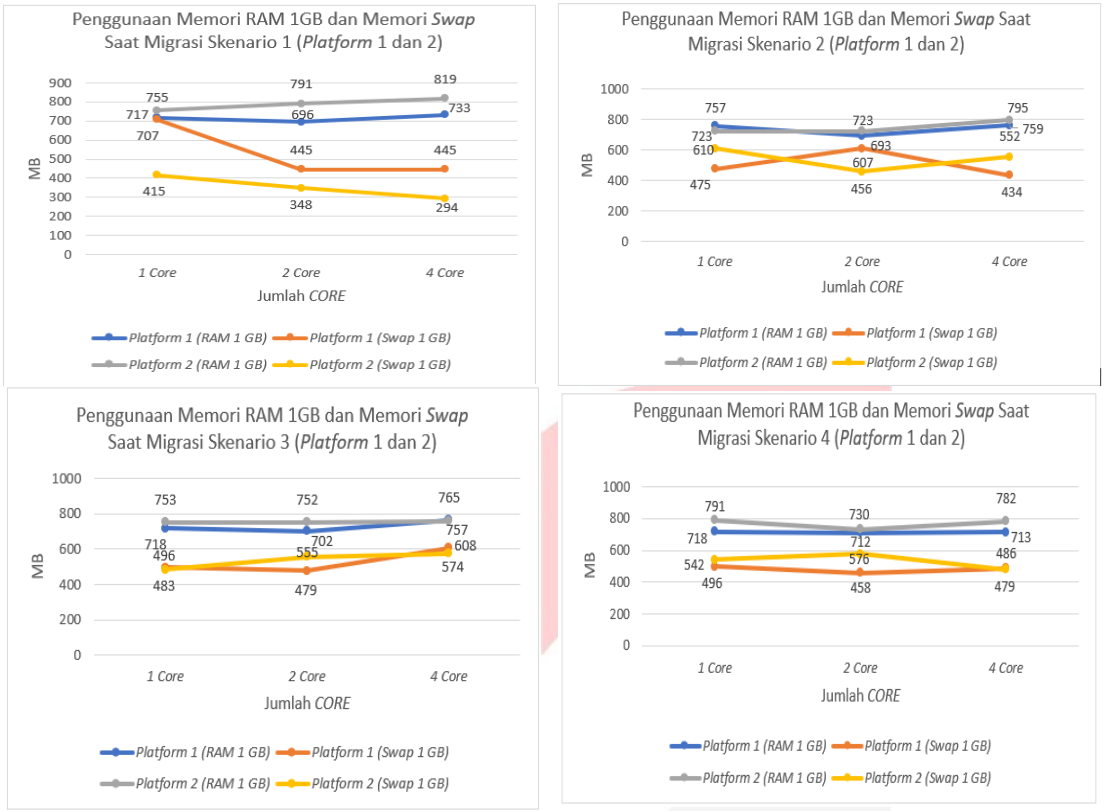
4.2.4 Migrasi Aplikasi Dua Arah II

Skenario keempat merupakan skenario yang dilakukan disaat skenario ketiga berhasil dan telah selesai dilakukan. Pada skenario ini *platform* 1 dan *platform* 2 melakukan pemindahan dan penerimaan LXD kedua yang artinya LXD yang telah dikirimkan akan dipulangkan kembali pada *platform* masing-masing. Untuk pengujian ini menghitung dan mengukur penggunaan memori RAM pada aplikasi dalam LXD sesaat dimigrasikan

5. Pengujian Sistem dan Analisis

Pengujian sistem pada migrasi LXD *Container* dilakukan dengan menggunakan LXD API dibantu dengan CRIU yang dijalankan pada sistem operasi Linux Ubuntu pada sebuah *virtual machine.* Pengukuran memori dilakukan saat LXD *container* telah terpasang aplikasi, pada kondisi tersebut jika memori menunjukkan angka stabil dan sistem masih dapat berjalan dengan baik maka akan dilakukan migrasi aplikasi yang berada didalam LXD, pada jalannya migrasi LXD ini juga dilakukan *monitoring* pada masing-masing *Platform* terhadap penggunaan memori lalu setelah selesai dilakukannya migrasi kembali melakukan *monitoring* pada masing-masing *Platform* . Pada pengujian berikut melakukan migrasi terhadap lima aplikasi layanan yang dijalankan antara lain Odoo, Apache, PhpMyAdmin, Mysql dan Wordpress.

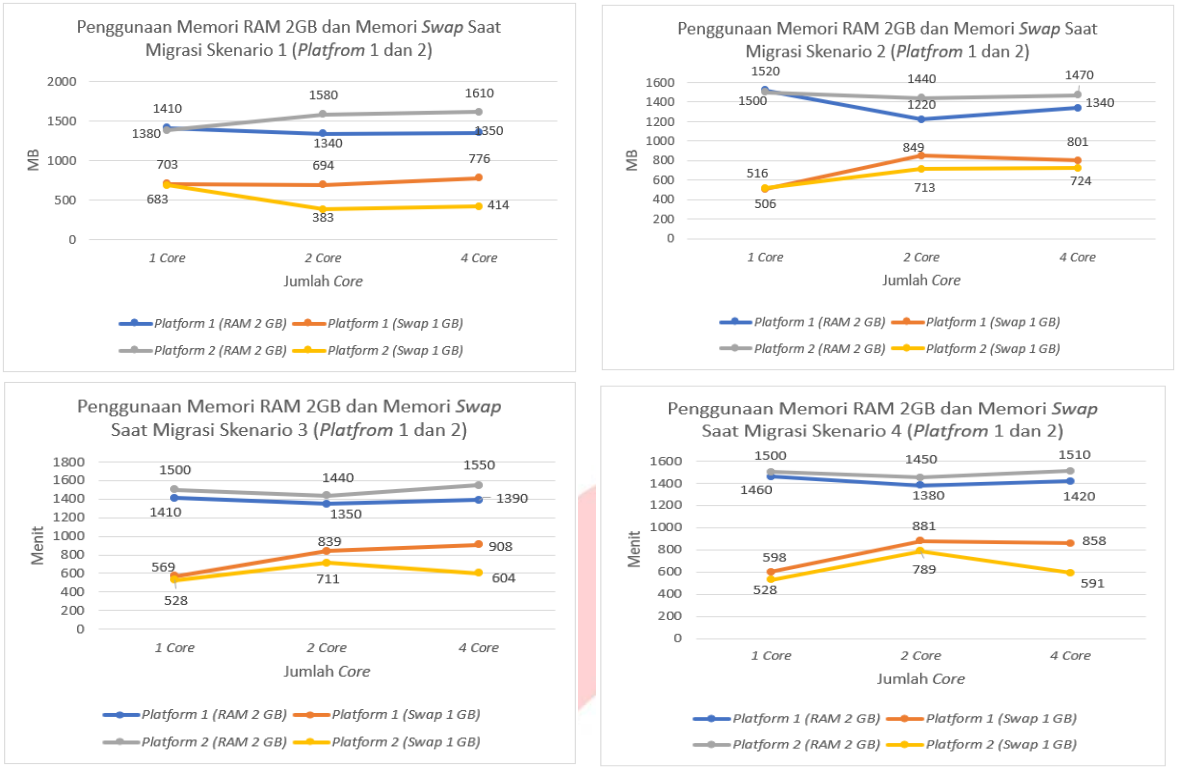
5.1. Hasil Pengukuran Memori Kapasitas 1 GB saat Migrasi Aplikasi pada 4 skenario



Gambar 3 Jumlah Penggunaan Memori 1GB dan Memori *Swap* Saat Migrasi 4 Skenario

Pada gambar 3 merupakan penggunaan memori RAM pada 4 skenario dengan perubahan *core* cenderung stabil pada kedua *platform* di saat kondisi migrasi berlangsung sehingga perubahan *core* terhadap penggunaan memori pada saat migrasi berlangsung tidak mempengaruhi secara signifikan. Pada penggunaan memori *swap* bergantung pada kemampuan memori RAM saat melakukan proses penyimpanan data sehingga dapat mencegah terjadinya *lag* atau *crash* yang disebabkan oleh penggunaan memori RAM yang melebihi kapasitas, biasanya semakin besar memori *swap* yang dipakai maka memori RAM yang dipakai akan berkurang maupun sebaliknya.

5.2. Hasil Pengukuran Memori Kapasitas 2 GB saat Migrasi Aplikasi pada 4 Skenario



Gambar 4 Jumlah Penggunaan Memori 2GB dan Memori *Swap* Saat Migrasi 4 Skenario

Gambar 4 merupakan penggunaan memori RAM 2GB dan memori *swap* 1GB pada saat proses migrasi 4 skenario dilakukan antara *platform* 1 dan *platform* 2 pada masing-masing *core* yang berbeda. Dalam pengukuran memori diambil disaat proses migrasi dimulai dan mencapai memori RAM tertinggi serta memori *swap* yang digunakan. Pada skenario 1 penggunaan memori RAM dengan perubahan *core* cenderung stabil pada kedua *platform* di saat kondisi migrasi berlangsung sehingga perubahan *core* terhadap penggunaan memori pada saat migrasi berlangsung tidak mempengaruhi secara signifikan. Pada penggunaan memori *swap* bergantung pada kemampuan memori RAM saat melakukan proses penyimpanan data sehingga dapat mencegah terjadinya *lag* atau *crash* yang disebabkan oleh penggunaan memori RAM yang melebihi kapasitas, biasanya semakin besar memori *swap* yang dipakai maka memori RAM yang dipakai akan berkurang maupun sebaliknya.

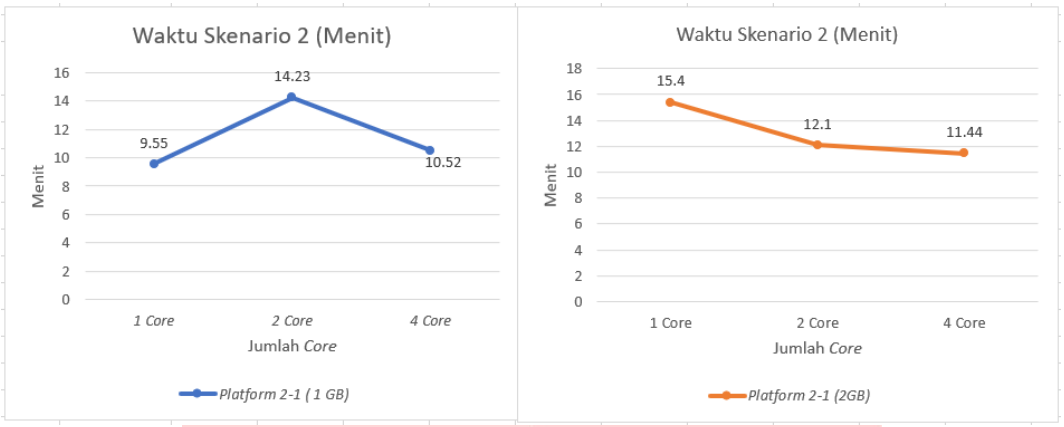
5.2. Hasil Waktu Terpakai pada Spesifikasi RAM 1 GB dan 2 GB pada 4 Skenario

Gambar 5 Waktu Terpakai Skenario 1

Gambar 5 berisi mengenai jumlah waktu yang dibutuhkan dalam melakukan migrasi dengan spesifikasi RAM 1GB dan 2GB pada *platform* 1 sebagai pengirim aplikasi dalam LXD dalam migrasi skenario 1. Pada spesifikasi memori RAM 1GB membutuhkan waktu 12.25 menit di 1 *core*, 16.51 menit di 2 *core* dan 12.44 menit di 4 *core*. Pada spesifikasi memori RAM 2GB membutuhkan waktu 16.51 menit di 1 *core*, 12.41 menit di 2 *core*

dan 13.34 menit di 4 *core* dengan rata-rata waktu yang dipakai pada spesifikasi RAM 1 GB untuk melakukan migrasi yaitu 13,73 menit, sedangkan rata-rata waktu yang dipakai pada spesifikasi RAM 2 GB untuk melakukan migrasi yaitu 14,08 menit.

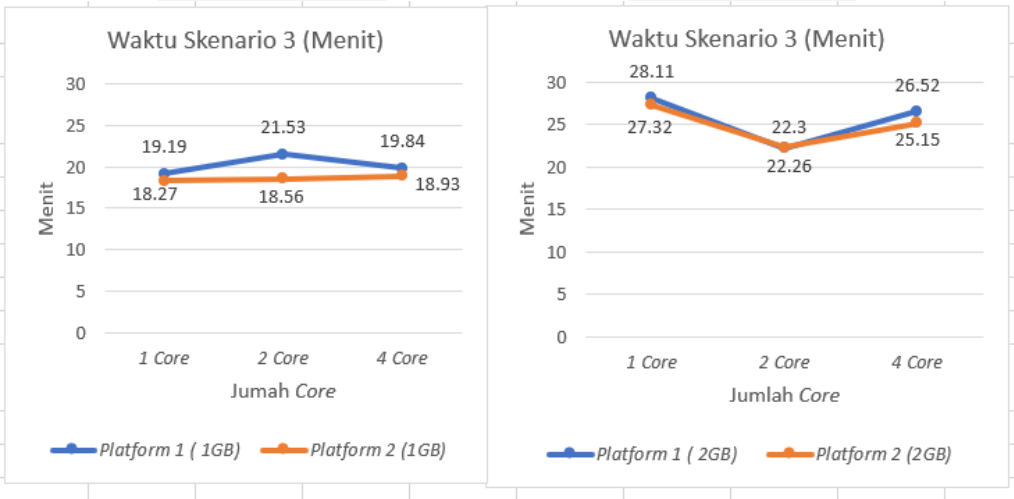
Pada pengujian skenario 1, pada spesifikasi RAM 1 GB dan 2 GB dengan 1 *core*, 2 *core*, dan 4 *core* memiliki jangka waktu migrasi dengan kondisi fluktuatif, sehingga dapat disimpulkan secara sementara bahwa perubahan *core* pada spesifikasi RAM 1 GB dan 2 GB tidak berpengaruh pada jangka waktu migrasi yang dilakukan.



Gambar 6 Waktu Terpakai Skenario 2

Gambar 6 berisi mengenai jumlah waktu yang dibutuhkan dalam melakukan migrasi dengan spesifikasi RAM 1GB dan 2GB pada *platform* 2 sebagai pengirim aplikasi dalam LXD dalam migrasi skenario 2. Pada spesifikasi memori RAM 1GB membutuhkan waktu 9.55 menit di 1 *core*, 14.23 menit di 2 *core* dan 10.52 menit di 4 *core*. Pada spesifikasi memori RAM 2GB membutuhkan waktu 15.4 menit di 1 *core*, 12.1 menit di 2 *core* dan 11.44 menit di 4 *core* dengan rata-rata waktu yang dipakai pada spesifikasi RAM 1 GB untuk melakukan migrasi yaitu 11,43 menit, sedangkan rata-rata waktu yang dipakai pada spesifikasi RAM 2 GB untuk melakukan migrasi yaitu 12,96 menit.

Pada pengujian skenario 2, pada spesifikasi RAM 1 GB dengan spesifikasi *processor* 1 *core*, 2 *core*, dan 4 *core* memiliki jangka waktu migrasi dengan kondisi fluktuatif, maka dapat disimpulkan secara sementara bahwa perubahan *core* pada spesifikasi RAM 1 GB tidak berpengaruh pada jangka waktu migrasi yang dilakukan sedangkan pada spesifikasi RAM 2 GB dengan spesifikasi *processor* 1 *core*, 2 *core*, dan 4 *core* memiliki jangka waktu yang semakin cepat terhadap peningkatan *core*.



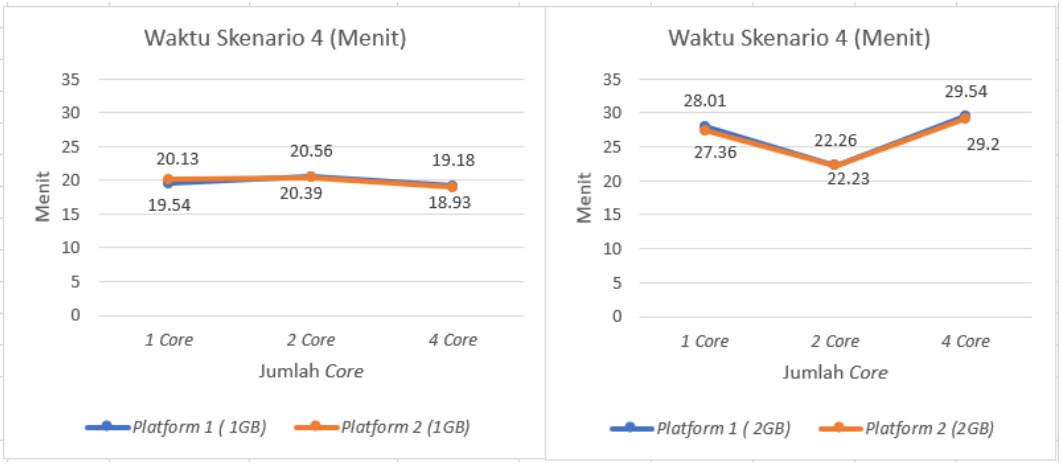
Gambar 7 Waktu Terpakai Skenario 3

Gambar 7 berisi mengenai jumlah waktu yang dibutuhkan dalam melakukan migrasi dengan spesifikasi RAM 1GB dan 2GB pada *platform* 1 dan 2 sebagai pengirim aplikasi dalam LXD dalam migrasi skenario 3. Pada spesifikasi memori RAM 1GB *platform* 1 membutuhkan waktu 19.19 menit di 1 *core*, 21.53 menit di 2 *core* dan 19.84 menit di 4 *core*, sedangkan pada *platform* 2 membutuhkan waktu 18.27 menit di 1 *core*, 18.56 menit di 2

core dan 18.93 menit di 4 *core* dengan rata-rata waktu yang dipakai pada spesifikasi RAM 1 GB untuk melakukan migrasi yaitu 20.18 menit pada *platform* 1 dan *platform* 2 untuk melakukan migrasi yaitu 19.52 menit.

Pada spesifikasi memori RAM 2GB *platform* 1 membutuhkan waktu 28.11 menit di 1 *core*, 22.26 menit di 2 *core* dan 26.52 menit di 4 *core* sedangkan pada *platform* 2 membutuhkan waktu 28.11 menit di 1 *core*, 22.3 menit di 2 *core* dan 25.15 menit di 4 *core* dengan rata-rata waktu yang dipakai pada spesifikasi RAM 1 GB untuk melakukan migrasi yaitu 25.63 menit pada *platform* 1 dan *platform* 2 untuk melakukan migrasi yaitu 25.18 menit.

Pada skenario 3 hasil waktu pengujian spesifikasi 1 GB dan 2 GB pada *platform* 1 dan *platform* 2 memiliki waktu migrasi dengan kondisi fluktuatif, sehingga dapat disimpulkan sementara bahwa perubahan *core* pada spesifikasi RAM 1 GB dan 2 GB tidak berpengaruh pada jangka waktu migrasi yang dilakukan.



Gambar 8 Waktu Terpakai Skenario 4

Gambar 8 berisi mengenai jumlah waktu yang dibutuhkan dalam melakukan migrasi dengan spesifikasi RAM 1GB dan 2GB pada *platform* 1 dan *platform* 2 sebagai pengirim aplikasi dalam LXD dalam migrasi skenario 4. Pada spesifikasi memori RAM 1GB *platform* 1 membutuhkan waktu 19.54 menit di 1 core, 20.56 menit di 2 core dan 19.18 menit di 4 core, sedangkan pada *platform* 2 membutuhkan waktu 20.13 menit di 1 *core*, 20.39 menit di 2 *core* dan 18.93 menit di 4 *core* dengan rata-rata waktu yang dipakai pada spesifikasi RAM 1 GB untuk melakukan migrasi yaitu 19.76 menit pada *platform* 1 dan *platform* 2 untuk melakukan migrasi yaitu 19.81 menit.

Pada spesifikasi memori RAM 2GB *platform* 1 membutuhkan waktu 28.01 menit di 1 *core*, 22.26 menit di 2 *core* dan 29.54 menit di 4 *core* sedangkan pada *platform* 2 membutuhkan waktu 27.36 menit di 1 *core*, 22.23 menit di 2 *core* dan 29.2 menit di 4 *core* dengan rata-rata waktu yang dipakai pada spesifikasi RAM 1 GB untuk melakukan migrasi yaitu 26.60 menit pada *platform* 1 dan *platform* 2 untuk melakukan migrasi yaitu 26.26 menit.

Pada skenario 4 hasil waktu pengujian spesifikasi 1 GB dan 2 GB pada *platform* 1 dan *platform* 2 mengalami kondisi fluktuatif, sehingga dapat disimpulkan sementara bahwa perubahan core pada spesifikasi RAM 1 GB dan 2 GB tidak berpengaruh pada jangka waktu migrasi yang dilakukan.

6. Kesimpulan

Dari hasil analisis penggunaan memori terhadap pengujian terhadap proses migrasi aplikasi dalam LXD dengan menggunakan LXD API di antara dua *platform* dengan spesifikasi sama., maka dapat diambil kesimpulan:

1. Proses migrasi aplikasi menggunakan LXD API dilakukan dengan melakukan *remote* antar *platform* melalui pengenalan IP, untuk IP kedua *platform* diharuskan dalam satu jaringan yang sama dan masing-masing telah melakukan installasi LXD. Dengan menggunakan LXD API, proses migrasi *container* berisi aplikasi harus dalam keadaan status berhenti *(stopped)* terlebih dahulu selanjutnya *container* tersebut dimigrasikan, migrasi akan gagal jika container masih dalam status *running.* Dengan melakukan pengujian migrasi dapat memindahkan layanan menuju tempat lainnya jika terdapat kondisi yang tak terduga terjadi.
2. Jumlah RAM yang tersedia pada sebuah *platform* tidak mempengaruhi kecepatan dan jangka waktu proses migrasi aplikasi dalam LXD *container* diantara dua *platform.* Selanjutnya pada pengujian penggunaan memori yang telah dilakukan berdasarkan kapasitas memori RAM 1 GB dan 2 GB serta pada *processor* 1 *core,* 2 *core* dan 4 *core* mendapatkan hasil bahwa perubahan *core* pada setiap memori RAM 1 GB dan 2 GB tidak mempengaruhi secara signifikan penggunaan memori pada saat mengakses

aplikasi dan melakukan pengujian 4 skenario migrasi aplikasi dalam LXD *container*¸ maka dari itu penggunaan memori mengalami perbedaan namun tidak signifikan terhadap rendah atau tingginya jumlah *core* pada *processor* yang terpakai, dan penggunaan memori RAM dan memori *swap* merupakan salah satu faktor berpengaruh dalam menentukan kondisi responsif suatu sistem yang menjalankan aplikasi layanan dalam LXD *container*.

Daftar Pustaka:

[1] Alauddin Fikri M, Ijtihadie Muslim R, Husni M (2017) : Implementasi Virtual *Data center* Menggunakan Linux *Container* Berbasis Docker dan SDN. JURNAL TEKNIK ITS Vol. 6, No. 2

[2] Xenitellis, S. (2018). *How to Set Up and Use LXD on Ubuntu 16.04 | DigitalOcean*. [online] Digitalocean.com. Available at: https://www.digitalocean.com/community/tutorials/how-to-set-up-and-use-lxd-on-ubuntu-16-04

[3] Usman Hijarah F, "*Disaster Recovery Strategy* menggunakan *software* Bacula dengan metode *Differential Backup-Restore*" 2018.

[4] S.Gupta, D. Gera "*A Comparison of LXD, Docker and Virtual Machine*" 2018.

[6] Azzam, T. (2018). *Container VS VM (Virtual Machine)*. [online] Medium. Available at: https://medium.com/core-network-laboratory-tech-page/*container*-vs-vm-virtual-machine-72c2dc406355 [Diakses 14 Jun. 2019].

[7] Classic A. (2016): Inilah Besaran Kapasitas Memori dari Terkecil Hingga Terbesar! https://www.kreasitekno.com/kapasitas-memori-terkecil-dan-terbesar/ (diakses/diunduh) pada 23 Oktober 2018.

[8] Enthil Kumaran S. "*Practical LXC and LXD"*, Tamil Nadu, India 2016.

[9] Prof. Ann Mary Joy, "*Performance Comparison Between Linux Containers and Virtual Machines"*,Ghaziabad, India, 2015

[10] Auliya Y A, Nurdinsyah Y and Wulandari D A R, "*Performance Comparison of Docker and LXD with ApacheBench*". 2018

[11] Graber, S. (2016). LXD 2.0: Remote hosts and container migration [6/12] | Stéphane Graber's website. Dari https://stgraber.org/2016/04/12/lxd-2-0-remote-hosts-and-container-migration-612/ (diakses/diunduh) pada 11 Juni 2019.