# IMPLEMENTASI KUBERNETES CLUSTER MENGGUNAKAN LXD CONTAINER

**Sarah Syifa Putri[1)], Muhammad Arif Fadhly Ridha[2)]**

[1]Program Studi Teknik Informatika, Politeknik Caltex Riau, Jl. Umbansari (Patin) No.1 Rumbai, Pekanbaru, 28265

[2]Program Studi Teknik Informatika, Politeknik Caltex Riau, Jl. Umbansari (Patin) No.1 Rumbai, Pekanbaru, 28265

E-mail: sarah17ti@mahasiswa.pcr.ac.id, fadhli@pcr.ac.id

**Abstract**

The use of cloud computing service enables clients to access data and information without having to install an application which increases the number of client requests that leads to the increase of server load and could result in overload. In clustering technology, the server workload will be distributed evenly to each server. To run clustering, a platform called Kubernetes is needed. Kubernetes is used for container management. To run a clustering with Kubernetes requires virtualization using LXD. In this research compares the performance of conventional model services and Kubernetes clusters. The test results of two servers in standby condition, the technology that uses the highest CPU and memory is the Kubernetes cluster which reached 104% on CPU usage, while memory usage is 28%. It is because of the nodes that are running on the virtual machine. In busy server conditions, conventional technology is the one that uses the highest CPU and memory due to the high, which reached 1289% on CPU usage while memory usage is 46%. Number of client requests that increase CPU and memory resources.

**Keywords:** *Cluster computing, Kubernetes, LXD, CPU, Memory*.

**Abstrak**

Penggunaan layanan *cloud computing* dapat memudahkan *client* untuk mengakses data dan informasi tanpa harus menginstall aplikasi. Penyebabnya peningkatan jumlah *request client* yang akan mengakibatkan beban *server* menjadi lebih berat dan mengakibatkan *overload*. Teknologi *cluster* hadir untuk menangani beban kerja *server* yang meningkat akan dibagi secara merata atau seimbang kepada masing-masing *server*. Namun, untuk menjalankan clustering, membutuhkan sebuah platform yang dinamakan Kubernetes. Kubernetes digunakan untuk manajemen container, untuk menjalankan cluster menggunakan virtualisasi Kubernetes menggunakan LXD. Pada penelitian ini dibandingkan performa layanan model konvensioanl dan Kubernetes *cluster*. Diperoleh hasil pengujian kedua *server* saat *server* dalam keadaan *standby*, teknologi yang menggunakan CPU dan *memory* paling tinggi adalah Kubernetes cluster yang mencapai 104% pada penggunaan CPU, sedangkan penggunaan memori yaitu 28%. Hal ini disebabkan terdapat node-node yang berjalan diatas mesin virtual. Pada saat *server busy*, teknologi konvensional adalah penggunaan CPU dan *memory* paling tinggi, yang mencapai 1289% pada penggunaan CPU sedangkan penggunaan memori yaitu 46%. Dikarenakan *request* klien yang tinggi dapat meningkatkan sumber daya CPU dan *memory*.

**Kata Kunci:** *Cluster computing, Kubernetes, LXD, CPU, Memory*.



**ISSN: 2339 – 2053**

Pekanbaru, 25 Agustus 2021

## PENDAHULUAN

Seiring dengan perkembangan teknologi segala sesuatu yang dilakukan menjadi lebih mudah, berbeda dengan sebelum adanya teknologi, segala sesuatu masih dilakukan secara manual. Seperti pada penggunaan teknologi komputasi. Teknologi komputasi adalah pemanfaatan atau penggunaan teknologi komputer, jaringan, *server* dan aplikasi berbasis internet (Kominfo, 2013). Salah satu jenis dari komputasi modern adalah *cloud computing*. *Cloud computing* merupakan teknologi yang menjadikan internet sebagai pusat *server* untuk mengelola data dan juga aplikasi pengguna. Selain itu, *cloud computing* memudahkan penggunanya untuk menjalankan program tanpa harus menginstall aplikasi terlebih dahulu dan mengakses data dan informasi melalui internet (IdCloudHost, 2019).

Layanan *cloud computing* tersimpan pada *server*. Semakin meningkatnya jumlah *requst client* akan menambah beban *server* menjadi lebih berat, sehingga dapat mengakibatkan *overload*. Dari hal tersebut perlu meningkatkan kinerja *server* ketika terjadi masalah dalam satu *node*, maka *virtual machine* didalamnya akan berpindah ke node lain untuk mengatasi gangguan pada layanan yang diakses *client*. Layanan-layanan *server* dijalankan pada mesin-mesin *server* virtual di dalam mesin *server* fisik. Jumlah layanan yang banyak, data-data penting, dan tingkat ketergantungan kinerja dari perusahaan, instansi, atau organisasi yang tinggi terhadap layanan *server* membuat *server* harus dapat melayani secara terus menerus. (Adi, Nurhayati, & Widianto, 2016). Penggunaan teknologi *cluster* maka akan mengurangi beban kerja *server*. *Cluster computing* adalah teknologi yang memanfaatkan beberapa sumber daya komputer untuk bekerja secara bersamaan, sehingga terlihat seperti satu sistem yang saling terintegrasi. Oleh karena itu, diperlukan sebuah teknologi *cluster computing* yang bertujuan mengurangi beban kerja *server* (Apriliana dkk., 2018).

Disebabkan perkembangan teknologi semakin pesat, maka virtualisasi berbasis *container* hadir untuk menjalankan aplikasi, sehingga membutuhkan beberapa *container* untuk menjalankan *services* atau layanan yang sama. Salah satu *platform* untuk mengelola teknologi *container* adalah Docker. Docker merupakan *project open source* yang



**ISSN: 2339 – 2053**

Pekanbaru, 25 Agustus 2021

menyediakan *platform* terbuka untuk pengembangan dan administrator agar dapat membangun, mengemas, dan menjalankan aplikasi dalam lokasi manapun sebagai sebuah (*container*) yang ringan. Untuk membangun suatu layanan diperlukan sebuah *software cluster orchestration* yaitu Kubernetes. Penelitian mengenai kubernetes sudah pernah dilakukan yaitu oleh Rendy Willianto (2020) yang berjudul "Manajemen Kontainer Berbasis Docker Menggunakan Kubernetes dan Portainer". Kubernetes merupakan *platform open source* yang digunakan untuk manajemen *container*.

Teknologi virtualisasi *server* diterapkan ke *server* dengan prosesor inti Multipel, dapat digunakan untuk menjalankan aplikasi-aplikasi dan layanan virtualisasi. Dengan menggunakan teknologi virtualisasi masalah mengenai kebutuhan terhadap *hardware* dapat teratasi. Sehingga penggunaan teknologi virtualisasi lebih menguntungkan dibandingkan penggunaan mesin non-virtual. Salah satu teknologi virtualisasi yaitu LXD *Container*. LXD *Container* merupakan sekumpulan satu atau lebih suatu proses yang diisolasi dari seluruh sistem (Saputra, Budiono, & Widjajarto, 2019). Selain itu, LXD sebanding dengan virtual *machine* dengan kemampuan untuk meng-*host* beberapa *container* OS di satu *host* tunggal. LXD juga menggunakan Linux *Container* (LXC) APIs melalui liblxc dan satu set *Go bindings* untuk membuat dan mengelola *container* (Pratama, Mayasari, & Sanjoyo, 2018).

Berdasarkan uraian diatas, maka penulis melakukan penelitian mengenai "Implementasi Kubernetes *Cluster* menggunakan LXD *Container*" dengan tujuan membangun cluster Kubernetes menggunakan virtualisasi LXD dan membandingkan kinerja sumber daya perangkat keras (CPU dan memori) selama virtualisasi server web standby dan sibuk antara teknologi konvensional dan teknologi cluster Kubernetes.

## METODE PENELITIAN

Adapun metode penelitian yang akan dilakukan pada penelitian ini adalah:

A. Pengujian *High-Availability*

Pengujian akan dilakukan sebanyak 5 kali pengujian agar dapat melihat tingkat keberhasilan dari pengujian *load balancer*, *failover* dan *failback*.



**ISSN: 2339 – 2053**

Pekanbaru, 25 Agustus 2021

1. Pengujian Load Balancer dan Failover

   Tingkat keberhasilan dari pengujian *load balancer* dan *failover* dilakukan dengan mengofflinekan salah satu node. Pada saat salah satu node telah *offline*, maka *resource* akan berpindah ke node yang tersedia (*running*).

2. Pengujian Failback

   Pengujian ini melihat apakah layanan web *server* yang sebelumnya digantikan oleh node lainnya akan kembali lagi.

B. Pengujian *Performance*

1. Pengujian *Stress Testing*

   Pengujian *Stress Testing* bertujuan untuk mengetahui dan mengukur kekuatan sebuah *web server* dalam menangani *request client* secara bersamaan dalam satu waktu pada server yang menggunakan aplikasi JMeter.

2. Pengujian performa virtualisasi web *server* dalam keadaan *standby* dan *busy*
   Pengujian ini dilakukan untuk memonitoring persentase penggunaan CPU dan *memory* ketika web *server* tidak diakses klien (*standby)* dan diakses oleh *client* (*busy*), dengan menggunakan aplikasi protokol SNMP.

3. Pengujian *Scalability*

   *Scalability* bertujuan untuk melihat beban layanan yang meningkat dengan cara penambahan pod, dan dapat melakukan pengurangan pod ketika beban layanan menurun. Perubahan tersebut dapat mengatasi beban layanan sesuai kebutuhan.

## HASIL DAN PEMBAHASAN

A. Hasil Pengujian *High-Availability*

Pada pengujian *failover* dan *Load Balancer* ini akan dilakukan skenario jika worker2 diofflinekan atau dimatikan apakah node lainnya akan menggantikan fungsi layanan dari worker2 dan web *server* masih dapat diakses oleh client atau tidak. Pengujian ini dilakukan sebanyak 5 kali percobaan. Hasil pengujian dapat dilihat pada Tabel 1.



**ISSN: 2339 – 2053**

Pekanbaru, 25 Agustus 2021

Tabel 1. Hasil Pengujian *Failover* dan *Load Balancing*

| Pengujian Ke - | Node Master | Worker 1 | Worker 2 | Status Web Server |
|---|---|---|---|---|
| 1 | Aktif | Aktif | Mati | Aktif |
| 2 | Aktif | Aktif | Mati | Aktif |
| 3 | Aktif | Aktif | Mati | Aktif |
| 4 | Aktif | Aktif | Mati | Aktif |
| 5 | Aktif | Aktif | Mati | Aktif |

Tabel 1 menunjukkan bahwa status *web server* tetap aktif dan dapat diakses *user* ketika worker2 telah diofflinekan atau dimatikan. Layanan *web server* diambil fungsi oleh worker1. Pengujian ini melakukan simulasi *user* sebanyak 20, 40, dan 60 *user*. Simulasi menggunakan perangkat lunak JMeter. Percobaan dilakukan sebanyak 5 kali.
Berikut adalah hasil pengujian *failback* yang dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian *Failover* dan *Load Balancing*

| Pengujian Ke - | Node Master | Worker 1 | Worker 2 | Status Web Server |
|---|---|---|---|---|
| 1 | Aktif | Aktif | Aktif | Aktif |
| 2 | Aktif | Aktif | Aktif | Aktif |
| 3 | Aktif | Aktif | Aktif | Aktif |
| 4 | Aktif | Aktif | Aktif | Aktif |
| 5 | Aktif | Aktif | Aktif | Aktif |

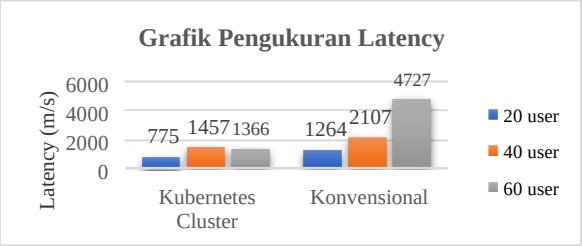
B. Hasil Pengujian *Performance*

1. *Stress testing*

Pengujian ini diakses *user* dengan menggunakan JMeter selama 5 menit (300 detik) dengan hitungan perdetik. Pengukuran tingkat *latency* dilakukan dengan 3 kali tahapan pengujian yaitu dengan 20 klien, 40 klien dan 60 klien dengan masingmasing pengujian dilakukan 5 kali percobaan pengambilan data.



**ISSN: 2339 – 2053**

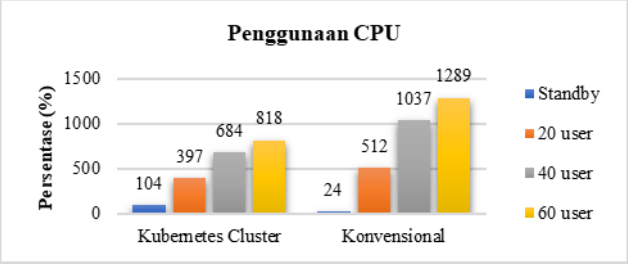



Gambar 1. Grafik rata-rata pengujian *stress testing*

Berdasarkan pengujian yang telah dilakukan diperoleh grafik seperti pada gambar 1 yang merupakan jumlah *latency* dari masing-masing *server*. Pada teknologi konvensional menunjukkan jumlah *latency* paling tinggi yaitu mencapai 4727m/s. Hal ini disebabkan karena dilakukannya *stress request* yang tinggi menyebabkan tingkat *delay* pada web *server* yang semakin besar. Sedangkan jumlah *latency* paling sedikit adalah teknologi Kubernetes cluster hanya 775m/s. Hal ini disebabkan karena adanya pembagian beban kerja pada node-node saat *stress testing* yang menerima *request* dari klien, sehingga tingkat *delay* pada web *server* semakin kecil.

2. Hasil CPU dan *Memory*

Berdasarkan pengujian yang dilakukan pada saat *server* keadaan *standby* dan *busy* diperoleh data penggunaan CPU. Penggunan CPU berikut dilakukan dengan keadaan *standby*, 20 klien, 40 klien dan 60 klien dengan masing-masing pengujian dilakukan 5 kali percobaan pengambilan data.



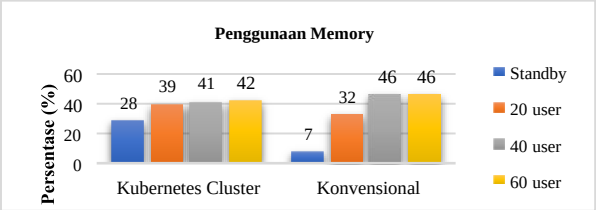

Gambar 2. Grafik penggunaan CPU saat *standby* dan *busy*

Berdasarkan pengujian yang telah dilakukan diperoleh grafik seperti pada gambar 2 yang merupakan penggunaan CPU keadaan *standby* dan *busy*. Pada saat keadaan *busy*, teknologi konvensional menunjukkan penggunaan CPU paling tinggi yaitu mencapai



**ISSN: 2339 – 2053**

1289% dikarenakan *request* klien meningkat saat dilakukannya *stress testing*. Sedangkan penggunaan CPU paling sedikit saat menerima *request* klien adalah teknologi kubernetes cluster hanya 818%. Hal ini disebabkan karena pembagian beban kerja pada node saat menerima *request* dari klien. Pada saat keadaan *standby*, teknologi Kubernetes cluster menunjukkan CPU paling tinggi yaitu mencapai 104% dikarenakan terdapat node-node yang berjalan di atas mesin virtual. Sedangkan penggunaan CPU paling sedikit adalah teknologi konvensional hanya 24%. Pengujian dengan membedakan jumlah klien ternyata memberikan hasil penggunaan CPU yang berbeda. Semakin banyak klien yang mengakses maka akan naik penggunaan CPU. Selanjutnya adalah analisa untuk penggunaan *memory* pada masing-masing *server* pada saat standby dan diakses oleh tiga tahapan user yaitu 20 user, 40 user dan 60 user.




Gambar 3. Grafik penggunaan memory saat *standby* dan *busy*

Berdasarkan pengujian yang telah dilakukan diperoleh grafik seperti gambar 3 yang merupakan penggunaan *memory* keadaan *standby* dan *busy*. Pada pengujian ini ternyata menunjukkan hasil yang sama dengan penggunaan *memory*, yaitu pada saat keadaan *busy* teknologi konvensional menunjukkan penggunaan *memory* paling tinggi yaitu mencapai 46%. Tentu penyebabnya sama yaitu karena *request* klien meningkat saat dilakukannya *stress testing*. Sedangkan penggunaan *memory* paling sedikit saat menerima *request* klien adalah teknologi kubernetes cluster hanya 42%. Hal ini disebabkan karena pembagian beban kerja pada *node* saat menerima *request* dari klien. Pada saat keadaan *standby*, teknologi Kubernetes cluster menunjukkan *memory* paling tinggi yaitu mencapai 28%. Penyebabnya sama dengan sebelumnya yaitu dikarenakan terdapat node-node yang berjalan di atas mesin virtual. Sedangkan penggunaan *memory* paling sedikit adalah teknologi konvensional hanya 7%.



**ISSN: 2339 – 2053**

Pekanbaru, 25 Agustus 2021

Pengujian dengan membedakan jumlah klien ternyata memberikan hasil penggunaan *memory* yang berbeda. Semakin banyak klien yang mengakses maka akan naik penggunaan *memory*.

3. Hasil Pengujian *Scalability*

Pada pengujian ini untuk melihat perubahan dari pod untuk mengatasi beban layanan sesuai kebutuhan dengan fitur *built-in* Kubernetes yaitu *Horizonatal Pod Scaler* (HPA).


Gambar 4. Persentase HPA saat dilakukan stress testing

Pada Gambar 4 merupakan pembuktikan bahwa keberhasilan dari penambahan pod ketika dilakukannya pengujian *stress test* terhadap layanan, sehingga persentase target meningkat hingga melebihi kapasitas 50%.


Gambar 5. Persentase HPA saat stress testing menurun

Pada Gambar 5 membuktikan bahwa ketika pengujian *stress test* terhadap layanan menurun maka akan terjadi pengurangan pod dan penurunan persentase target.

## SIMPULAN

Berdasarkan hasil pengujian yang telah dilakukan, ada beberapa hal yang dapat disimpulkan dari penelitian ini, yaitu:

1. Kuberneter cluster dapat dibangun didalam virtualisasi LXD.
2. Teknologi Kubernetes cluster menunjukkan jumlah *latency* terendah yaitu hanya 775m/s dibandingkan dengan konvensional berdasarkan semua pengujian yang dilakukan.
3. Teknologi konvensional menggunakan sumber daya lebih besar seperti CPU dan *Memory* yaitu mencapai 1289% pada penggunaan CPU dan penggunaan memori yaitu 46% dibandingkan dengan penggunaan sumber daya Kubernetes cluster berdasarkan semua pengujian yang dilakukan.



**ISSN: 2339 – 2053**

Pekanbaru, 25 Agustus 2021

4. Semakin banyak jumlah klien yang mengakses web *server* maka semakin tinggi penggunaan sumber daya perangkat keras pada kedua *server*.

## DAFTAR PUSTAKA

Adi, Cornell, Nurhayati, & Widianto. (2016). Perancangan Sistem Cluster Server untuk Jaminan Ketersediaan Layanan Tinggi pada Lingkungan Virtual. *JNTETI,* 69, 23014156.

Apriliana, L., Darusalam, U. D., & Nathasia, N. D. (2018). Clustering Server Pada Cloud Computing Berbasis Proxmox VE Menggunakan Metode High Availability. *JOINTECS (Journal of Information Technology and Computer Science)*. https://doi.org/10.31328/jointecs.v3i1.498

IDCloudHost, (2019). *Mengenal Apa Itu Cloud Computing: Defenisi, Fungsi, dan Cara Kerja*. *The New York Times*. idcloudhost.com. http:// idcloudhost.com/mengenalapa-itu-cloud-computing-defenisi-fungsi-dan-cara-kerja/

Kominfo, (2013). *Penerapan Komputasi Awan*. Kominfo.go.id http://kominfo.go.id/content/detail/3459/kominfo-masih-pelajari-penerapan komputasi-awan/0/berita_satker

Rendy, W., (2020). Manajemen Kontainer Berbasis Docker Menggunakan Kubernetes dan Portainer.

Saputra, Budiono, & Widjajarto. (2019). Analisis Penggunaan System Process Pada Migrasi Aplikasi Dalam Linux Container (LXD) Menggunakan LXD API. *eProceeding of Engineering*, 7862, 2355-9365.

Pratama, Mayasari, & Sanjoyo. (2018). Implementasi Web Server Cluster Menggunakan Metode Load Balancing Pada Container Docker, LXC, dan LXD. *eProceeding of Engineering*, 5028, 2355-9365.



**ISSN: 2339 – 2053**

Pekanbaru, 25 Agustus 2021