E-ISSN: 2776-2343

High Availability Service dengan Multiple Master pada Kubernetes Cluster Menggunakan Virtualisasi Kernel Based Virtual Machine (KVM)

Nabila Firdha Aisyah[1)], Muhammad Arif Fadhly Ridha [2)]
[1]*Program Studi Teknik Informatika, Politeknik Caltex Riau, Pekanbaru, Indonesia*
[2]*Program Studi Teknik Informatika, Politeknik Caltex Riau, Pekanbaru, Indonesia*

E-mail: nabilafirdha@alumni.ac.id , fadhly@pcr.ac.id

Abstract: *Servers have an important role in managing and serving requests from clients. So that high availability is needed to handle risks in server use. Cluster technology is needed, so that the server load is shared evenly. However, to run clustering, it requires the application of a method, namely the HAProxy load balancer which works by evenly distributing the traffic load to the clustered physical servers. By implementing Kernel-Based Virtual Machine (KVM) virtualization technology. In KVM virtualization there is a technology used, namely Kubernetes. From the results of this final project, the results of the High Availability testing carried out are known that the server performance with scenario 5 access experiments 20-500 users with stress testing. Parameters measured are throughput and monitoring system usage of CPU, memory according to user access load. Which is the highest value of Througput on parameter 500 clients worth 1330.6/ sec. The presence of 2 servers provides performance test results. In standby, the lowest CPU usage and memory usage occurs on the main server, which is 0.07% and 24% because there are no nodes running on top of the virtual machine. In a busy state, the lowest CPU usage occurs on the main server is 1.06% due to the division of workload on virtual machines. The lowest memory usage occurs on the main server server is 24% because there are no nodes running on top of the virtual machine.*

Keywords: *Cluster computing, CPU, Failback, Failover, High Availability, HAProxy, Kubernetes, KVM, Load Balancing, Memory, Throughput.*

Abstrak: *Server mempunyai peran penting dalam mengatur dan melayani permintaan dari client. Sehingga dibutuhkannya high availability untuk menangani resiko dalam penggunaan server. Dibutuhkanlah teknologi cluster, bertujuan agar beban server dibagi secara merata. Namun, untuk menjalankan clustering dibutuhkan penerapan metode yaitu load balancer HAProxy yang bekerja dengan mendistribusikan secara merata beban trafik ke physical server yang tercluster. Dengan menerapkan teknologi virtualisasi Kernel-Based Virtual Machine (KVM). Di dalam virtualisasi KVM ada teknologi yang digunakan yaitu Kubernetes. Dari hasil penelitian Proyek Akhir ini didapatkan hasil dari pengujian* High Avaibility *yang dilakukan diketahui bahwa kinerja server dengan skenario 5 percobaan akses 20-500 user dengan strees testing. Parameter yang diukur yaitu throughput serta sistem monitoring besar penggunaan CPU, memory sesuai dengan beban akses user. Yang mana nilai tertinggi dari Througput pada parameter 500 client senilai 1330,6/ sec. Dengan adanya 2 buah server memberikan hasil pengujian performance Dalam keadaan standby, pemakaian CPU dan pemakaian memori terendah terjadi pada server utama adalah 0,07% dan 24% karena tidak ada node-node yang berjalan di atas mesin virtual. Dalam keadaan busy, pemakaian CPU terendah terjadi pada server utama adalah 1,06% karena adanya pembagian beban kerja terhadap mesin virtual. Pemakaian memory terendah terjadi pada server server utama adalah 24 % karena tidak ada node-node yang berjalan di atas mesin virtual.*

Kata kunci: *Cluster computing, CPU, Failback, Failover, High Availability, HAProxy, Kubernetes, KVM, Load Balancing, Memory, Throughput.*

1. Pendahuluan

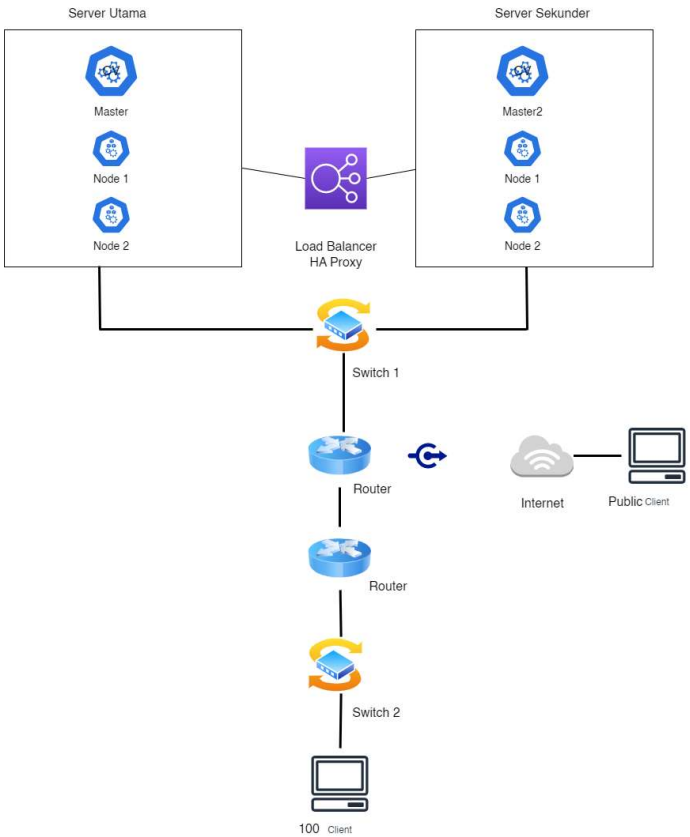
Perkembangan teknologi memberikan pengaruh bagi kehidupan, salah satunya adalah teknologi komputasi. Teknologi komputasi memberikan layanan yang dapat mengakses satu sama lainnya. Model komputasi modern yang digunakan saat ini adalah *cloud computing*. Layanan *cloud computing* tersimpan pada server, yang mana server memiliki kemampuan penyimpanan yang berbeda-beda tergantung dari kapasitas yang ada. Semakin banyak yang mengakses melalui suatu situs web membuat beban kerja yang lebih pada suatu penyedia layanan yang disebut web server dan menjadi kurang optimal. Suatu *single server* bisa mengalami kegagalan yang disebabkan oleh meningkatnya jumlah *request* yang mencapai ribuan bahkan jutaan pada waktu yang bersamaan atau disebut dengan *overload* [1]. Hal ini akan merugikan pihak yang mempercayakan situsnya pada suatu *web server*, karena situs tersebut tidak dapat diakses sesuai yang diinginkan dari *client*.

Untuk membangun *web server* yang dapat membagi beban dengan kualitas layanan yang baik, maka salah satu yang bisa diterapkan untuk mengatasi permasalahan tersebut yaitu Server Clustering. Suatu teknologi yang dapat membangun *physical server* dengan Multiple Master yang bekerja bersama-sama yang se olah-olah merupakan satu system tunggal [2]. Dengan didukung teknik *load balancing* yang diharapkan dapat menangani beban yang sangat berat dengan mendistribusikannya kepada server lain yang tercluster. Sehingga trafik dapat berjalan dengan optimal. Penerapan *load balancing* pada *web server* cluster sangat penting dan dapat menjadi solusi dalam menangani beban server yang terlalu berat dengan memanfaatkan HAProxy sebagai *load balancer* [3]. *Load balancer* dibangun dalam sebuah *container*, dengan menerapkan teknologi virtualisasi Kernel-Based Virtual Machine (KVM). Virtualisasi KVM menggunakan jenis virtualisasi *Full virtualization* yang bebas menggunakan OS apapun yang diinginkan, seperti Linux, windows maupun yang lainnya. Di dalam virtualisasi KVM ada teknologi yang digunakan yaitu Kubernetes. Kubernetes merupakan sistem *opensource* yang berfungsi untuk mengoptimalkan penyebaran maupun pengukuran *container* [4]. Dengan adanya teknologi tersebut dapat membantu mengatasi adanya masalah *distributed server*. Sehingga dengan kelebihan *container* yang mudah dikelola dan sangat ringan, memungkinkan untuk melakukan proses high avaibility pada *physical server*. Dengan demikian, tidak hanya beban server saja yang berkurang tetapi ketersediaan layanan juga terjamin. Sehingga terbentuklah sebuah layanan *cloud computing* dengan High availability yang memiliki tingkat performasi yang baik, handal, dan terjaga ketersediaannya [5].

Hal inilah yang melatar belakangi penulis untuk membuat judul proyek akhir “High Availability Service dengan Multiple Master pada Kubernetes Cluster menggunakan Virtualisasi Kernel- Based Virtual Machine (KVM)”.

2. Metode Pengujian

Dalam metode penelitian ini menggunakan multiple master dibangun dalam bentuk *physical server.* Adapun topologi yang dibangun seperti gambar dibawah ini,



Gambar 1. Topologi Jaringan

Metode pengujian pada proyek akhir ini dilakukan dengan beberapa cara yakni sebagai berikut:

A. Pengujian High-Availability

Pengujian high availability akan dilakukan melalui 5 pengujian agar dapat melihat tingkat keberhasilan dari *failover*, *failback* dan *load balancing* [7]. Skenario yang digunakan dalam pengujian yaitu dengan mematikan salah satu *node* yaitu worker2 untuk melihat tingkat keberhasilan dari *failover* dan menghidupkan kembali worker2 untuk melihat tingkat keberhasilan dari *failback*. Adapun *load balancer* diimplementasikan pada *pod* sehingga pada saat *pod* di-*replicas* secara otomatis oleh *horizontal pods autoscale* (HPA), *load balancer* akan melakukan distribusi *pod* secara seimbang pada setiap *node worker*.

B. Pengujian *Performance*

Pengujian *performance* akan dilakukan dengan beberapa cara seperti: *stress request* untuk mengetahui dan mengukur pembebanan pada server utama dan server sekunder cluster dalam menangani *request client* dalam jumlah yang sangat banyak dalam sebuah web server dengan menggunakan software JMeter [8]. Pada saat dilakukan *stress request*, kemudian dilakukan monitoring penggunaan CPU dan *memory* pada server utama dan server sekunder cluster dengan menggunakan protocol SNMP dan software yang digunakan adalah SNMP PRTG. Yang mana Pengujian kinerja virtualisasi web server dalam keadaan *standby* dan *busy* dilakukan untuk memantau persentase penggunaan CPU dan *memory* yang terpakai pada web server tidak di akses *client* (*standby*) dan di akses *client* (*busy)* pada server utama serta server sekunder. Pengujian *scalability* bertujuan untuk melihat beban layanan meningkat Penskalaan jumlah pod berdasar pemanfaatan CPU atau memori yang diamati [9]. Dalam kata lain horizontal pod autoscaler akan menambahkan wadah atau yang dikenal dengan sebutan container jika sudah melebihi batas maksimum. Untuk melakukan konfigurasi pada HPA dapat membuat skala pods berdasar *varied, external, and custom metrics*. Horizontal pod *autoscaler* lebih cocok digunakan ketika dibutuhkan kapasitas untuk *scalability* yang cukup banyak. Dengan mereplika *container-container* yang ada [10].

3. Hasil Dan Pembahasan

3.1 Hasil Pengujian High Avaibility

Pengujian *failback, loadbalancing* serta *failover* yang dilakukan untuk melihat keberhasilan dari failback, Master02 yang sebelumnya dimatikan akan dihidupkan kembali untuk melihat apakah layanan yang tadinya diambil alih oleh Server Utama akan kembali lagi pada Master02 (Server Sekunder). Adapun load balancer diimplementasikan pada pod, sehingga saat pod di-replicas secara otomatis oleh horizontal pods autoscale (HPA), load balancer akan melakukan distribusi pod secara seimbang pada setiap node. Berikut hasil pengujian tingkat keberhasilan failover dan load balancing dapat dilihat pada Tabel 1.

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

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

Pada Tabel 1 menunjukkan bahwa status *web server* tetap aktif dan dapat diakses oleh *client* ketika Master2 telah dimatikan. Layanan *web server* diambil fungsi oleh Server utama karena pada saat dimatikan Master2 (server sekunder) secara otomatis layanan *web server* diambil alih oleh server utamanya pada saat *running*. Pengujian ini dilakukan sebanyak lima kali percobaan pada *pyshical server* dengan simulasi menggunakan perangkat lunak JMeter. Berikut keberhasilan pengujian pada *failback* dapat dilihat pada Tabel 2.

Tabel 2 Hasil Pengujian *failback*

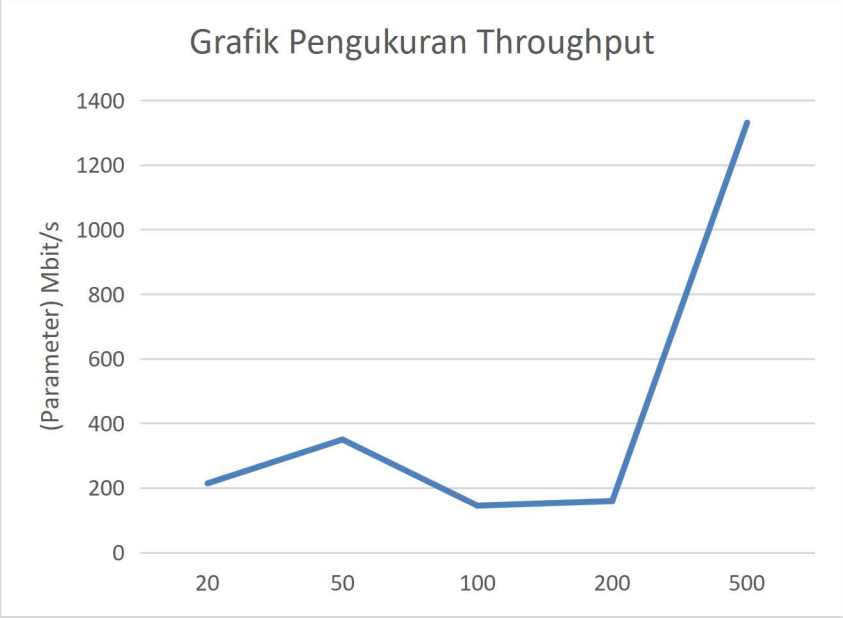
Pengujian Ke-	Node			Status Web Server
	Master	Worker 1	Worker 2	
1	Aktif	Aktif	Aktif	Aktif
2	Aktif	Aktif	Aktif	Aktif
3	Aktif	Aktif	Aktif	Aktif
4	Aktif	Aktif	Aktif	Aktif
5	Aktif	Aktif	Aktif	Aktif

Pada Tabel 2 ini menunjukkan bahwa status *web server* tetap aktif dan dapat diakses oleh *client* ketika Master2 telah dihidupkan kembali. Dimana terjadi pembagian beban pada ketiga node yang running yaitu master, worker 1 dan worker 2. Sehingga layanan *web server* dapat berjalan pada setiap *node*. Pengujian ini dilakukan sebanyak lima kali percobaan pada *pyshical server* dengan simulasi menggunakan perangkat lunak JMeter.

3.2 Hasil Pengujian *Performance*

1. Hasil Pengujian *Stress Testing*

Pengujian *stress testing* dilakukan sebanyak 5 skenario percobaan dengan menggunakan *software* JMeter bertujuan untuk mengetahui dan mengukur kinerja sebuah *web server* dalam menangani *request client*. Tediri dari beberapa percobaan *client*. Pada percobaan pertama yaitu 20 *client*. Kedua 50 *client*, ketiga 100 *client*, keempat 200 *client* dan kelima 500 *client* selama (30 detik) setiap satu kali percobaan. Setiap pengujian akan ditampilkan datanya sehingga menjadi hasil dari pengujian tersebut. Gambar 3 adalah hasil dari pengujian JMeter, berikut tabel hasil dari pengujian *stress request* menggunakan *software* JMeter.



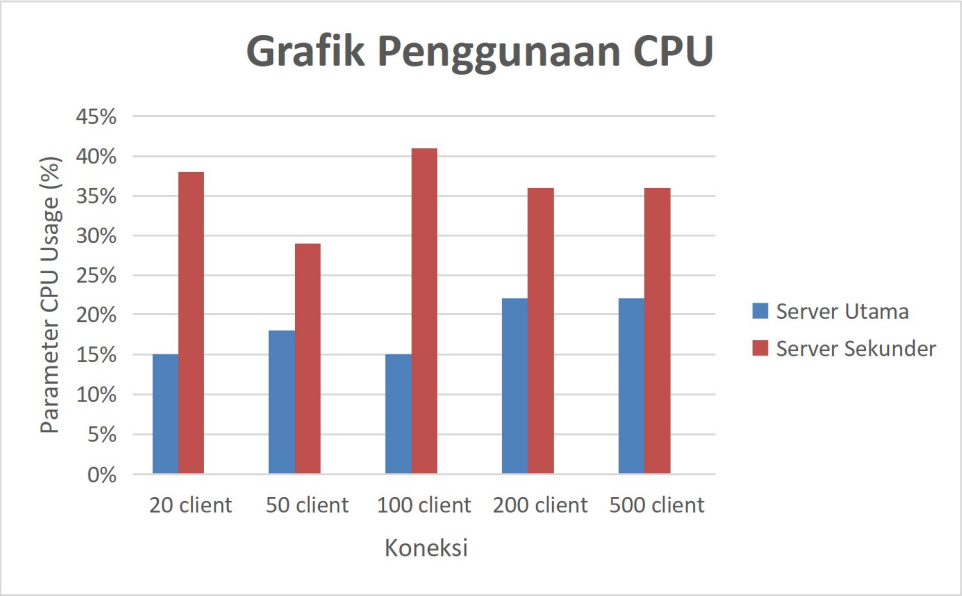
Gambar 1 Data Pengujian Jmetter

Berdasarkan pengujian yang telah dilakukan diperoleh grafik seperti pada Gambar 4. 8 Pada 20 koneksi, didapatkan nilai rata-rata throughput sebesar 213,9/sec yang nilainya mulai meningkat pada 50 koneksi menjadi 349,6/sec. Terjadi penurunan nilai throughput yang sangat kecil dan mengalami penurupa pada koneksi 100 dengan nilai throughput terendah sebesar 145,0/sec. Dan pengalami penaikan sedikit pada 200 koneksi menjadi 159,0/sec. Terjadi kenaikan signifikan pada 500 koneksi menjadi sebesar 1330,6/ sec. Kenaikan nilai throughput terjadi karena penambahan jumlah koneksi yang secara otomatis meningkatkan jumlah request terhadap web

server dalam waktu pengiriman yang pendek, sehingga terjadi peningkatan performa web server. Sementara penurunan *throughput* terjadi setelah 50 koneksi disebabkan karena faktor kenaikan pada koneksi dan pengguna jaringan pada jaringan yang sama pada saat pengujian.

2. Hasil CPU dan Memory

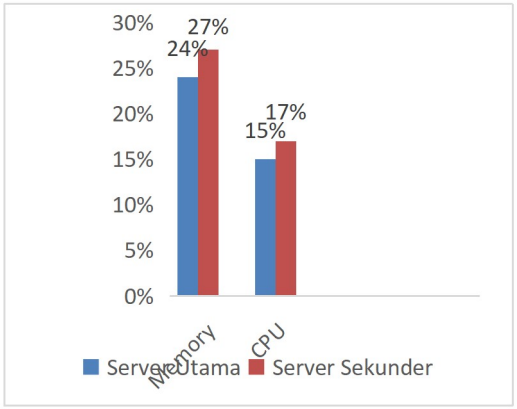
Berdasarkan pengujian yang dilakukan pada saat keadaan *server standby* dan *busy* diperoleh data penggunaan CPU. Penggunaan CPU dilakukan dengan keadaan *standby*, 20 *client*, 50 *client*, 100 *client*, 200 *client* dan 500 *client* dengan masing-masing pengujian dilakukan dalam waktu 30/sec dalam mengambil data. Gambar 4 Grafik penggunaan CPU saat *standby* dan *busy*



Gambar 4 Penggunaan CPU

Berdasarkan pengujian yang telah dilakukan diperoleh grafik seperti pada Gambar 4. 9 yang merupakan hasil rata-rata *persentase* penggunaan CPU dari semua pengujian yang dilakukan pada *server* Utama dan *server Sekunder* dalam keadaan *busy*.

Pada saat keadaan *busy*, Server Utama menunjukkan penggunaan CPU terendah yaitu 15% karena adanya pembagian beban kerja pada setiap *node* saat menerima *request client*, sedangkan penggunaan CPU



Gambar 5 Penggunaan CPU dan Memory

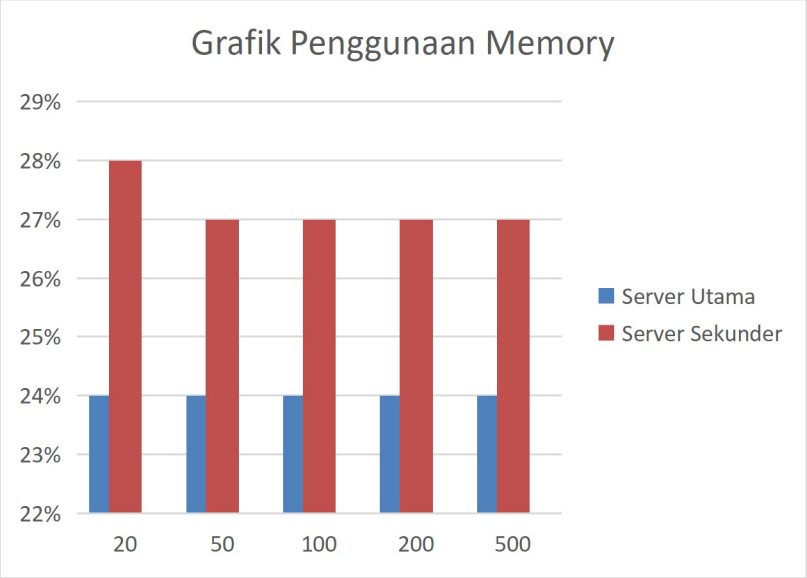
tertinggi adalah Server Sekunder yaitu 41% karena tidak adanya pembagian beban kerja terhadap mesin virtual.

Pad Gambar 5 saat keadaan *standby*, Server Utama menunjukkan penggunaan CPU terendah yaitu 0,07% karena tidak ada *node-node* yang berjalan di atas mesin virtual, sedangkan penggunaan CPU tertinggi adalah teknologi *Kubernetes cluster* yaitu 0,33% karena terdapat *node-node* yang berjalan di atas mesin virtual.

Pada saat keadaan *standby*, psyical server menunjukkan penggunaan *memory* rata-tertinggi yaitu 27% karena penggunaan storage pada *node* yang berjalan di atas mesin virtual, sedangkan penggunaan *CPU* tertinggi adalah pada server sekunder *cluster* yaitu 17% karena terdapat *node-node* yang berjalan di atas mesin virtual.

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

Selanjutnya adalah analisis penggunaan *memory* pada masing-masing *server* pada saat *standby* dan diakses oleh 100 *client*, 200 *client*, 300 *client*, 400 *client* dan 500 *client*.



Gambar 6 Grafik penggunaan *memory* saat *busy*

Berdasarkan pengujian yang telah dilakukan diperoleh grafik seperti pada Gambar 6 yang merupakan hasil rata-rata *persentase* penggunaan *memory* dari semua pengujian yang dilakukan pada *server utama* dan *server sekunder cluster* dalam keadaan *busy*.

Pada saat keadaan *standby*, server menunjukkan penggunaan *memory* rata-rata yaitu 24% karena tidak ada *node-node* yang berjalan di atas mesin virtual, sedangkan penggunaan *memory* tertinggi adalah server sekunder cluster yaitu 28% karena terdapat *node-node* yang berjalan di atas mesin virtual.

Pengujian dengan membedakan jumlah *client* ternyata memberikan hasil penggunaan *memory* yang berbeda. Semakin banyak klien yang mengakses maka akan tergantung pada beban server dalam penggunaan *memory*.

4. Kesimpulan

Dari hasil pengujian dan analisis yang telah dilakukan, maka dapat diambil beberapa kesimpulan sebagai berikut:

1) Kubernetes cluster dapat dibangun di dalam virtualisasi KVM.

2) Horizontal pods autoscaller (HPA) dapat melakukan replikasi pod secara otomatis pada teknologi Kubernetes cluster.
3) Penggunaan monitoring sesuai dengan yang di harapkan setelah dilakukan pengujian, sistem mendapatkan output yaitu hasil data monitoring dengan parameter grafik penggunaan CPU, *Memory* dan Throughput.
4) Througput tertinggi yaitu 1330,6/sec pada parameter 500 *client* berdasarkan pengujian sehingga semakin tinggi nilai Througput maka semakin baik kinerja server.
5) Failover dan Failback yang dilakukan dalam pengujian High Avaibility memiliki dengan keberhasilan
6) Penggunaan kemampuan pada system load balanching HAProxy pada multiple master dengan dua buah web server cluster dalam melayani request *client* web server cluster dalam melayani request *client*.
7) Dari hasil kinerja dari load balancing server web berbasis KVM bedasarkan penggunaan sumber daya *memory* dapat disimpulkan bahwa penggunaan load balancing berdasarkan penggunaan sumber daya *memory* dapat mengurangi traffic web server.
8) Dalam keadaan *busy*, pemakaian CPU dan pemakaian *memory* yang tinggi pada server sekunder yaitu 41% dan 28% pada server utama berdasarkan semua pengujian yang dilakukan.
9) Dalam keadaan *standby*, pemakaian CPU terendah pada server pertama adalah 0,07% dan pemakaian *memory* keduanya yaitu 27% berdasarkan semua pengujian yang dilakukan.
10) Semakin banyak jumlah *client* yang mengakses web server maka semakin tinggi penggunaan sumber daya perangkat keras pada kedua server.

Ucapan Terima Kasih

Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada pihak yang telah banyak memberikan bantuan dan dukungan yang tiada terhingga baik secara langsung maupun tidak langsung. Ucapan terima kasih tersebut penulis tujukan kepada:
1. Allah Azza Wa Jalla atas rahmat, kemudahan dan berkah yang telah dilimpahkan-Nya, sehingga penulis bisa menyelesaikan penelitian ini tepat waktu.
2. Kedua orang tua penulis, ibu dan ayah yang selalu mendo'akan serta dukungan dan kasih sayang nya, sehingga penulis bisa menyelesaikan penelitian tepat waktu.
3. Dr. Dadang Syarif Sihabudin Sahid, S.Si., M.Sc. selaku Direktur Politeknik Caltex Riau yang telah memberikan dukungan moral dalam menyelesaikan proyek akhir ini.
4. Ibu Kartina Diah Kusuma Wardhani, S.T., M.T. selaku Ketua Program Studi Teknik Informatika yang telah memberikan izin untuk menyelesaikan proyek akhir.
5. Ibu Rika Perdana Sari, S.T., M.Eng. selaku Koordinator Proyek Akhir dan Dosen Wali yang telah membantu dalam menyelesaikan proyek akhir ini.
6. Bapak Muhammad Arif Fadhly Ridha, S.Kom., M.T selaku dosen pembimbing yang telah memberikan ilmu dan bimbingan dengan penuh kesabaran kepada penulis dalam menyelesaikan proyek akhir.
7. Bapak Ibnu Surya, S.T., M.T. dan Bapak Rahmat Suhatman, S.T., M.T. selaku dosen penguji, yang telah memberikan masukan dan saran dalam menyelesaikan proyek akhir.
8. Seluruh dosen khususnya Program Studi Teknik Informatika dan seluruh dosen di Politeknik Caltex Riau pada umumnya yang telah memberikan bekal ilmu kepada penulis dalam menyelesaikan proyek akhir.

Rujukan

[1] Surbakti, N. K, "Implementasi Kubernetes Cluster Menggunakan KVM", ABEC Indonesia, 209-217, 2021.
[2] J Huang, "Implementasi Multi Server Data Storage Pada Cloud Computing," Pekanbaru, 2018.
[3] E Larsen., "Analisis Perbandingan High Availability Web Server Menggunakan Heartbeat Dan Pacemaker", Vol. 8 No. 2 (2019): JAKT
[4] A. Nugroho, W. Yahya, and Kasyful Amron, "Analisis Perbandingan Performa Algoritma Round Robin dan Least Connection untuk Load Balancing pada Software Defined Network," J. Pengemb. Teknol. Inf. dan Ilmu Komput., vol. 1, pp. 1568–1577, 2017.
[5] Kahanwal, B., Singh, T. P., "The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle". International Journal of Latest Research in Science and Technology Vol.1, Issue 2: Page No.183-187, 2012.
[6] Hoenisch, P., Weber, I., Schulte, S., Zhu, L., & Fekete, A. 2015. Four-Fold Auto-Scaling on a Contemporary Deployment Platform Using Docker Containers. ServiceOriented Computing Lecture Notes in Computer Science,316-323. doi:10.1007/978-3-662-48616-0_20

[7] Irfani and H. Sulistyanto, “Implementasi High Availability Server Dengan Teknik Failover Virtual Computer Cluster,” 2015.
[8] Rahmatulloh, A., & MSN, F. (2017). Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi. Jurnal Nasional Teknologi Dan Sistem Informasi, 3(2), 241–248. https://doi.org/10.25077/teknosi.v3i2.2017.241-248
[9] D. I. Permatasari, “Pengujian Aplikasi menggunakan metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian,” J. Sist. dan Teknol. Inf., vol. 8, no. 1, p. 135, 2020, doi: 10.26418/justin. v8i1.34452.
[10] S. Taherizadeh and V. Stankovski, “Dynamic Multi-level Auto-scaling Rules for Containerized Applications,” Comput. J., vol. 62, no. 2, pp. 174–197, Feb. 2019.