Jurnal Kajian Ilmiah (JKI)
e-ISSN: 2597-792X, ISSN: 1410-9794
Vol. 20 No. 2 (Mei 2020), Halaman: 137 – 144
Terakreditasi Peringkat 5 (SINTA 5) sesuai SK RISTEKDIKTI Nomor. 3/E/KPT/2019

Service High Availability Pada Native Server dan Virtual Server Menggunakan Proxmox VE

Dadan Irwan [1,*], Heru Sukoco [2], Sri Wahjuni [2]

[1] Fakultas Teknik; Universitas Islam 45; Jl. Cut Meutia No. 83, Bekasi Timur, Bekasi, Jawa Barat 17113. Telp: 021-8801027, 8802015, Fax: 021-880119236; e-mail: dadanirwan@unismabekasi.ac.id
[2] Departemen Ilmu Komputer; Institut Pertanian Bogor; Bogor, 0251-8625584; e-mail: hsrkom@ipb.ac.id; email: my_juni04@yahoo.com

* Korespondensi: e-mail: dadanirwan@unismabekasi.ac.id

Abstract

Virtualization technology can improve service capabilities to two or more server machines virtually. Failover and failback systems are high availability techniques in reducing service failures on the main server. This study aims to analyze the ability of high availability service with a failover and failback system on two different server architectures namely native server and virtual server using the Proxmox VE virtualization system. The research methodology uses the stages of problem analysis and system requirements, architecture design and implementation, testing by measuring the capability of the system, analyzing the results. Based on the testing process, high availability on virtual server systems has a lower performance rate of 4.15% on average with native server systems. In conclusion, virtualization systems have the advantage of being able to provide two different virtual server services on one native server machine so that they are more efficient in terms of budget and more effective in managing server administration systems.

***Keywords**:* virtualisasi, *virtual server*, *native server*, *high availability*, *failover failback*

Abstrak

Teknologi virtualisasi dapat meningkatkan kemampuan layanan menjadi dua mesin server atau lebih secara *virtual*. Sistem *failover* dan *failback* merupakan teknik *high availability* dalam mengatasi terjadinya kegagalan layanan pada master server. Penelitian ini bertujuan untuk menganalisis kemampuan *service high availability* dengan sistem *failover* dan *failback* pada dua arsitektur *server* yang berbeda yaitu *native server* dan *virtual server* menggunakan sistem virtualisasi Proxmox VE. Metodologi penelitian menggunakan tahapan analisa permasalahan dan kebutuhan sistem, disain arsitektur dan implementasi, pengujian dengan pengukuran kemampuan sistem, dan analisa hasil. Berdasarkan proses pengujian, *high availability* pada sistem *virtual server* memiliki tingkat kinerja lebih rendah rata-rata 4.15% dengan sistem *native server*. Kesimpulannya sistem virtualisasi memiliki keunggulan karena dapat memberikan dua layanan *server* virtual yang berbeda pada satu mesin *native server* sehingga lebih efisien dari sisi anggaran dan lebih efektif dalam pengelolaan sistem administrasi server.

Kata kunci: virtualisasi, *virtual server*, *native server*, *high availability*, *failover failback*

1. Pendahuluan

Server merupakan perangkat utama pada jaringan komputer yang bekerja untuk pengolahan data dan menyediakan layanan informasi, sehingga dapat memberikan layanan yang tinggi atau yang biasa disebut dengan istilah *high availability.* Perkembangan teknologi pada mesin *server* sudah mendukung sistem virtualisasi yang bertujuan untuk memaksimalkan sumber daya yang terdapat pada *server* (Harijanto & Ariyanto, 2015). Pada sistem virtualisasi terdapat

Submitted: **20 Maret 2020**; Revised: **10 April 2020**; Accepted: **24 April 2020**; Published: **26 Mei 2020**

teknologi *hypervisor* yang memiliki kemampuan memaksimalkan kelebihan sumber daya pada *server* seperti *processor*, *memory*, dan *harddisk* sehingga sebuah mesin *server* dapat menghasilkan beberapa *hardware* virtual pada semua *software* yang berjalan (Fajrin & Sabiq, 2016). Selain itu, teknologi virtualisasi mampu memberikan efisiensi biaya dalam pengadaan infrastruktur perangkat keras *server* serta pemeliharaan sistem lebih efektif dengan adanya sistem *backup* dan *recovery* (Sriyanta, Winarno, & Sudarmawan, 2018). Proxmox VE merupakan salah satu *software* virtualisasi *hypervisor type* 1 yang bersifat *open source* dan tidak berbayar dengan keunggulan dalam memaksimalkan *CPU load*, *memory usage, network traffic,* dan *respon time* (Surahmat & Tenggono, 2019).

Pesatnya perkembangan teknologi pada mesin *server* saat ini menjadi keuntungan para pengelola *server* dalam menyediakan infrastruktur layanan kepada para penggunanya. Spesifikasi yang tinggi pada sebuah mesin *server* (*native server*) dapat dioptimalkan untuk menyediakan beberapa mesin *server virtual* dengan layanan yang berbeda (Widarma & Siregar, 2019).

Penerapan teknologi virtualisasi pada sebuah *native server* menjadi kelebihan dalam aspek jumlah layanan untuk pengelola tetapi menjadi sebuah kekhawatiran pada aspek penyediaan data dan informasi untuk pengguna. *High avalilabity* diimplementasikan untuk meningkatkan ketersediaan layanan yang tinggi yang disediakan oleh sebuah *server* pada sebuah infrastruktur jaringan komputer (Rosalia, Munadi, & Mayasari, 2016). Ketersediaan layanan oleh *server* dipengaruhi oleh kemampuan mesin *server* yang dapat bekerja tanpa mengalami gangguan terutama gangguan dari dalam seperti kerusakan pada salah satu komponen *hardware*.

Pada penelitian (Sriyanta et al., 2018), penelitian tersebut menitikberatkan pada bagaimana memaksimalkan sebuah perangkat *server* menjadi beberapa mesin *server virtual* sehingga diperoleh beberapa layanan yang baru.

Penelitian yang dilakukan oleh (Rosalia et al., 2016), bertujuan untuk meningkatkan kehandalan dan ketersediaan layanan pada mesin *virtual server* dan membandingkan performansi dua *software* yaitu *load balancing* dan *failover*. Penelitian ini bertujuan untuk mengetahui tingkat *high availability* pada mesin *native* server dan pada mesin *virtual server* dengan menggunakan aplikasi *hypervisor full virtualization* Proxmox VE. Pada implementasi virtualisasi *hypervisor* ditambahkan teknik *failover* dan *failback* sebagai parameter dalam *high availability.*

Teknik *failover* bekerja ketika terjadi kerusakan dan menimbulkan kegagalan layanan pada *master server* sehingga *slave server* mengambil alih layanan. Sementara teknik *failback* adalah proses pengambil alihan layanan dari *slave server* oleh *master server*. Pada penelitian ini menggunakan dua buah mesin *native* server yang memiliki spesifikasi berbeda untuk implementasi sistem *failover* dann *failback* dengan dukungan sistem jaringan komputer.

2. Metode Penelitian

2.1. Analisa Permasalahan dan Kebutuhan

Pada tahapan ini, peneliti melakukan analisis terhadap permasalahan yang terjadi pada sistem *server* yang sedang berjalan seperti terjadinya kehilangan data, informasi, dan layanan lainnya pada pengguna. Hal ini terjadi karena ketika *master server* mengalami gangguan maka tidak ada *server* lain yang mampu mengambil alih layanan dari *master server*. Kemudian untuk mengatasi permasalahan tersebut perlu dilakukan analisis kebutuhan baik kebutuhan *hardware* maupun *software* termasuk aspek sistem jaringan komputer.

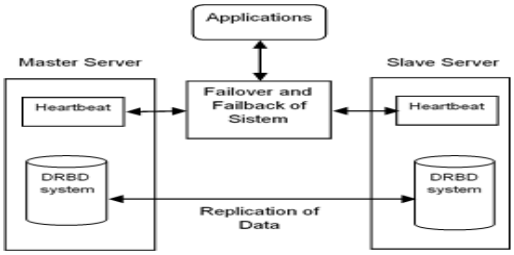
2.2. Disain dan Implementasi Sistem

2.2.1. Disain Arsitektur Sistem dan Jaringan

Disain arsitektur dilakukan konfigurasi sistem *failover clustering* baik pada sistem *native server* maupun sistem *virtual server*. Konfigurasi dilakukan dengan membuat satu *server* sebagai *master server* dan satu *server* sebagai *slave server* dimana pada dimana saat *server* dalam keadaan normal *master server* menangani semua *request* dari *client*. *Slave server* akan mengambil alih tugas *master server* apabila *master server* tidak berfungsi atau *down*.

Kegagalan pada sistem *server* tidak akan disadari oleh *client* karena tersedianya *server* lain sebagai *backup* sehingga dapat mengatasi kegagalan atau *failure* pada *web server* itu sendiri (Pribadi, Pn, & Irwansyah, 2020). Terdapat dua arsitektur pada *failover clustering* yaitu *synchronous* dan *asynchronous*.

Pada penelitian ini menggunakan arsitektur *asynchronous*. Arsitektur *asynchronous* merupakan arsitektur yang proses sinkronisasinya dilakukan oleh *server* aktif kepada *server* pasif. Arsitektur *asynchronous* adalah jenis *failover* yang tidak memerlukan *server* identik dan spesifikasi sama (Kang, Park, Hyoung, & Oh, 2014).



Sumber: Hasil Penelitian (2017)

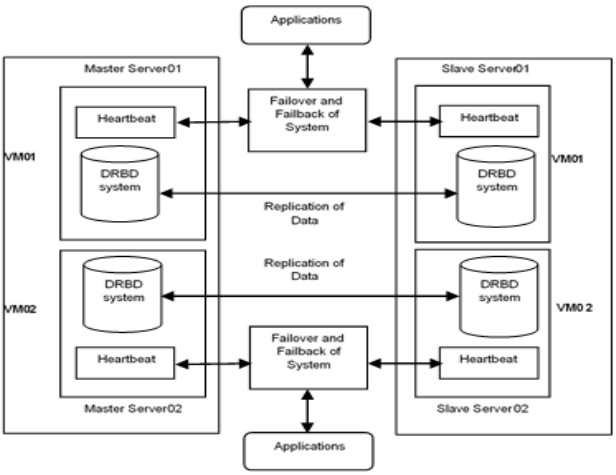
Gambar 1. Arsitektur sistem pada *native server*

Pada Gambar 1 ditunjukkan disain arsitektur *failover clustering* pada *native server* dimana terdapat dua *server*, satu *server* bertindak sebagai *master server* dan *server* lainnya sebagai *slave server*. Pada arsitektur yang sama terdapat dua aplikasi yaitu *Heartbeat* yang berfungsi sebagai *software* penyedia sistem *failover* dan *Duplicated Replicated Block Device* (DRBD) berfungsi sebagai *software* penyedia proses duplikasi dan replikasi data.

Arsitektur topologi jaringan pada *native server* dan *virtual server* memiliki disain yang sama. Terdapat koneksi langsung antara *master server* dengan *slave server* yang digunakan untuk

Submitted: **20 Maret 2020**; Revised: **10 April 2020**; Accepted: **24 April 2020**; Published: **26 Mei 2020**

proses duplikasi dan repilkasi data atau DRBD. Setiap *server* terhubung ke jaringan Internet untuk proses implementasi dan pengujian sistem.

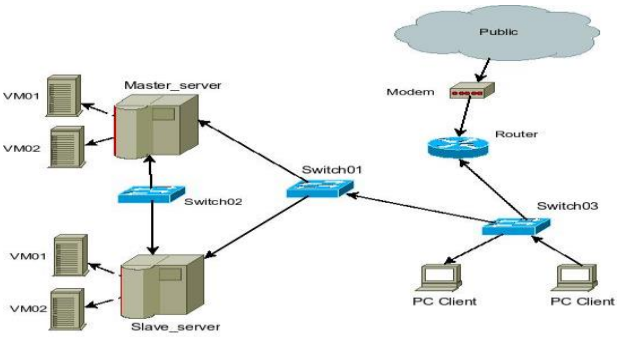


Sumber: Hasil Penelitian (2017)

Gambar 2. Arsitektur sistem pada *virtual server*

Pada Gambar 2 ditunjukkan arsitektur sistem *failover clustering* pada *virtual server*. Pada arsitektur ini terdapat dua mesin *virtual* yang dibangun pada masing-masing *server*. Arsitektur yang dibangun pada *virtual server* menyerupai arsitektur pada *native server*. Perbedaan antara kedua arsitektur ini adalah terletak pada penggunaan dua *virtual* mesin pada *virtual server.*

Pada Gambar 3 ditampilkan arsitektur topologi jaringan sistem *failover clustering* pada *native server* dan *virtual server*. Pada masing-masing server terpasang 2 buah *network interface card* (NIC) dengan fungsi yang berbeda. NIC pertama berfungsi untuk menghubungkan setiap server ke jaringan computer dan NIC kedua berfungsi untuk proses terjadinya duplikasi dan replikasi data antara *master server* dan *slave server*.



Sumber: Hasil Penelitian (2017)

Gambar 3. Arsitektur Topologi Pada Sistem Jaringan

2.2.2. Implementasi Sistem

Pada tahap ini adalah implementasi sistem *failover clustering* pada *native server* dan *virtual server*. Implementasi sistem *failover clustering* menggunakan dua *server*, dimana sebuah *server* sebagai *master server* dan *server* lainnya sebagai *slave server*. Terdapat dua proses yang berbeda pada implementasi sistem *failover clustering*. Proses pertama adalah implementasi pada *native server* dengan menggunakan sistem operasi Turnkey Lamp 64 bit versi 4.2. Pada sistem

operasi ini sudah menyediakan dukungan *web server*, *database server*, MySQL *server*, dan terdapat sistem informasi perpustakaan sebagai alamat website yang digunakan untuk proses pengujian. Pada Tabel 1 ditunjukkan daftar kebutuhan *software* untuk implementasi sistem *failover clustering* pada *native server* dan *virtual server*. Proses duplikasi dan replikasi data memerlukan *harddisk drive* tambahan dengan kapasitas yang sama baik pada *master server* maupun *slave server.*

Tabel 1. Daftar kebutuhan *software*

No	Jenis aplikasi *server*	*Master server*	*Server* cadangan
1	Sistem Operasi	*Turnkey Lamp*	*Turnkey Lamp*
2	*Failover*	*Heartbeat*	*Heartbeat*
3	Replikasi Data	DRBD	DRBD
4	*Web server*	*Apache2*	*Apache2*
5	*Database server*	*MySQL*	*MySQL*
6	Aplikasi *website*	SIM Perpustakaan	-
7	*Virtualization Management*	Proxmox	Proxmox

Sumber: Hasil Penelitian (2017)

Tahap berikutnya adalah implementasi sistem *failover clustering* pada *virtual server* menggunakan aplikasi *hypervisor* sebagai manajemen virtualisasi. Pada kedua *server* telah dibangun 2 (dua) *virtual machine* (VM), 2 VM pada *master server* berfungsi sebagai *master* dan 2 VM pada *slave server* berfungsi sebagai cadangan. Pada Tabel 2 ditampilkan daftar spesifikasi kedua VM pada masing-masing *virtual server*.

Tabel 2. Spesifikasi Logik *Virtual Server* Pada *Master Server* Dan *Slave Server*

No	Spesifikasi logik	*Master server*		*Slave server*	
		VM01	VM02	VM01	VM02
1	*Processor*	1 *core*	1 *core*	1 *core*	1 *core*
2	*Memory* (RAM)	512 Mb	512 Mb	512 Mb	512 Mb
3	*Harddisk Drive* 01	50 Gb	50 Gb	50 Gb	50 Gb
4	*Harddisk Drive* 02	10 Gb	10 Gb	10 Gb	10 Gb
5	*Network Interface Card* (NIC)	Eth0	Eth1	Eth0	Eth1

Sumber: Hasil Penelitian (2017)

Implementasi IP *address* pada *native server* menggunakan 5 buah IP *address* dengan dua alamat *network* berbeda. IP *address* untuk sistem *failover clustering* menggunakan 3 buah IP *address* dan DRBD menggunakan 2 buah IP *address*. IP *address* untuk *virtual server* menggunakan 14 buah IP *address* dengan 2 alamat *network* yang berbeda. Empat IP *address* untuk *interface* vmbr, 8 buah IP *address* untuk VM dan 2 buah IP *address* untuk IP *virtual*.

2.3. Pengujian Sistem

2.3.1 Skenario Pengujian

Proses pengujian ini memerlukan beberapa skenario yaitu dengan memberikan pembebanan ke *master server* dengan mengirimkan sejumlah *request rate* menggunakan aplikasi *Httperf generator*. Pada Tabel 3 ditampilkan skenario pengujian *service availability* pada *master server*.

Pada proses pengujian sistem *failover* menggunakan parameter meliputi *request rate*, *reply rate* dan *error rate*. *Request rate* adalah total data koneksi*, reply rate* adalah data yang

Submitted: **20 Maret 2020**; Revised: **10 April 2020**; Accepted: **24 April 2020**; Published: **26 Mei 2020**

berhasil direspon, dan *error rate* adalah data yang mengalami kegagalan koneksi. Pada Tabel 3 ditampilkan skenario pembebanan pada *server*.

Tabel 3. Skenario Pengujian Sistem

No	Jenis pengujian	Skenario pengujian	*server* yang diuji
1	Pengujian *failover* dan *failback* (*service availability*)	Melakukan penonaktifan dan pengaktifan layanan *Heartbeat*	*native server* dan *virtual server*
2.		Melakukan pembebanan pada *server* dengan mengirim jumlah koneksi maksimal 2000 dan *request rate* antara 10 sampai 50 konekasi/detik, variasi jumlah beban 10 koneksi, setiap pengujian dilakukan pengulangan sebanyak 5 kali (lihat Lampiran 3 dan 4).	

Sumber: Hasil Penelitian (2017)

Tabel 4. Data variasi jumlah *request rate* pada pengujian *failover*

No	*Request rate* (*conn/second*)	IP *address virtual*	*File target*
1	10	83.1.0.103	*index.php*
2	20		
3	30		
4	40		
5	50		

Sumber: Hasil Penelitian (2017)

Pada Tabel 4 ditampilkan variasi jumlah *request rate* dari *client* ke *server*. Pada proses ini, komputer klien mengakses alamat *virtual* pada *server*. Sementara itu, index.php merupakan *file* yang menjadi target untuk menghubungkan antara komputer *client* dengan aplikasi web pada *server*.

3. Hasil dan Pembahasan

3.1. Analisis Proses *Failover* dan *Failback*

Proses pengujian sistem *failover* pada *native server* dapat berjalan dengan baik. Sementara itu, proses pengujian sistem *failover* pada *virtual server* hanya menggunakan satu mesin *virtual*, sedangkan mesin *virtual* lainnya pada keadaan *standby*. Pada proses pengujian ke semua mesin *virtual* mengalami kegagalan karena sistem berjalan sangat lambat dan tidak stabil. Penyebab kegagalan terjadi karena perangkat *input/output* pada *virtual server* memiliki arsitektur yang bersifat *virtual*, sehingga mempengaruhi sistem kerja dan transmisi data dari *server* ke klien.

Pada Tabel 5 ditampilkan data rata-rata yang didapat pada pengujian *failover* dan *failback* pada *native server* dan *virtual server*. Data pada *failover* merupakan data yang didapat dari proses kerja *slave server* dan pada *failback* merupakan data yang didapat dari proses kerja *master server*.

Durasi proses untuk *request rate* (RR) 10 koneksi/detik pada total 2000 koneksi memerlukan waktu sekitar 2 detik dan pada RR 50 koneksi/detik sekitar 40 detik. Data *reply* merupakan *success connection* pada *failover* dan *failback* dengan jumlah *request rate*. Berdasarkan pengujian yang dilakukan bahwa semakin tinggi jumlah RR maka waktu yang

diperlukan semakin lama dan jumlah *lost connection* semakin tinggi. Pada Tabel 6 ditampilkan data *reply* (*success of ratio*) pada *native server* dan *virtual server* untuk 2000 koneksi.

Tabel 5. Data hasil pengukuran *failover* dan *failback* pada *native server* dan *virtual server* dengan 2000 koneksi

No	*Request Rate* (*conn/second*)	Parameter	*Failover* (conn)		*Failback* (conn)	
			Native server	*Virtual server*	*Native server*	*Virtual server*
1		*Request*	1941	1837	1891	1698
	10	*Reply*	1940	1836	1857	1641
		Error	60	165	143	198
2		*Request*	1884	1673	1472	1592
	20	*Reply*	1882	1670	1200	1566
		Error	118	330	800	434
3		*Request*	1806	1573	1171	1395
	30	*Reply*	1774	1542	868	1056
		Error	226	458	1132	944
4		*Request*	1716	1236	1326	1509
	40	*Reply*	1682	1175	1061	1268
		Error	318	825	939	732
5		*Request*	1581	1076	1322	1214
	50	*Reply*	1516	1006	1170	1010
		Error	484	994	830	990

Sumber: Hasil Penelitian (2017)

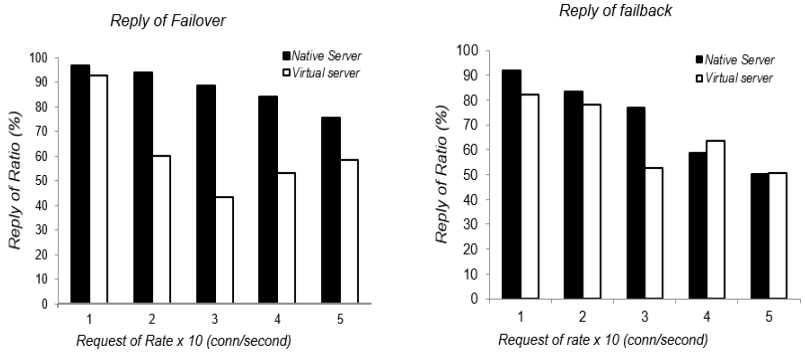
Berdasarkan pengujian pada jumlah RR 10 koneksi/detik, *service availability* pada *server* menghasilkan nilai yang cukup tinggi, tapi ketika *server* menerima beban lebih besar terjadi penurunan baik pada *native server* dan *virtual server*.

Tabel 6. Data *reply failover* dan *failback* pada *native server* dan *virtual server* dengan total jumlah koneksi 2000

No	*Request Rate* (*conn/second*)	*Reply of failover (conn)*		*Reply of failback (conn)*	
		Native server	*Virtual server*	*Native server*	*Virtual server*
1	10	1940	1857	1836	1641
2	20	1882	1200	1670	1566
3	30	1774	868	1542	1056
4	40	1682	1061	1175	1268
5	50	1516	1170	1006	1010

Sumber: Hasil Penelitian (2017)

Pada Gambar 4 adalah perbandingan presentase *service availability* antara *native server* dan *virtual server*. Data didapat dengan menghitung parameter *reply* dan membaginya dengan jumlah total koneksi dalam satuan persen.



Sumber: Hasil Penelitian (2017)

Gambar 4. Presentase *service availability* pada *native server* dan *virtual server*

Submitted: **20 Maret 2020**; Revised: **10 April 2020**; Accepted: **24 April 2020**; Published: **26 Mei 2020**

Berdasarkan proses pengujian yang dilakukan bahwa jumlah RR dan total koneksi sangat mempengaruhi proses *failover* (*downtime*) dan *failback* (*uptime*). Proses pengujian *failover* dan *failback* pada *native server* menunjukkan nilai yang lebih baik dari *virtual server*. Sementara itu, jika nilai RR meningkat maka nilai *service availability* pada server terjadi penurunan secara linear. Sistem *failover* pada native server 4.15% lebih baik dari *virtual server*, sedangkan proses *failback* pada kedua jenis server cenderung memiki presentase yang sama.

4. Kesimpulan

Service high availability pada sistem *virtual server* memiliki tingkat kinerja lebih rendah rata-rata 4.15% dibandingkan dengan sistem *native server*. Tetapi pada sistem *virtual server* memiliki keunggulan dapat memberikan beberapa layanan *server* yang berbeda dengan tersedianya 2 atau lebih mesin *server* virtual pada satu mesin *native server* sehingga lebih efisien dari sisi anggaran dan lebih efektif dalam pengelolaan sistem administrasi server.

Daftar Pustaka

Fajrin, F., & Sabiq, A. (2016). Private Cloud Menggunakan Proxmox Ve Pada Sistem Informasi Akademik Politeknik Purbaya. *Jurnal Teknologi Informasi YARSI*, *3*(1), 40–45. https://doi.org/10.33476/jtiy.v3i1.777

Harijanto, B., & Ariyanto, Y. (2015). *Server Menggunakan Proxmox Virtual*. *5*(1), 17–24.

Kang, K., Park, J., Hyoung, S., & Oh, J. Y. (2014). *(12) United States Patent*. *2*(12).

Pribadi, Y., Pn, A. B., & Irwansyah, M. A. (2020). *Analysis of the Use of the Failover Clustering Method to Achieve High Availability on a Web Server (Case Study : Informatics Department Building) Analysis of the Use of the Failover Clustering Method to Achieve High Availability on a Web Server (Case Study : Informatics Department Building)*. *8*(2). https://doi.org/10.26418/justin.v8i2.31965

Rosalia, M., Munadi, R., & Mayasari, R. (2016). Load Balancing Dan Failover Pada Virtual Web Server Cluster Implementation of High Availability Server Using Load Balancing and. *EProceesings of Engineering*, *3*(3), 4496–4503.

Sriyanta, Winarno, W. W., & Sudarmawan. (2018). Optimalisasi Penggunaan Hardware Server Mempergunakan Virtualisasi Server di SMAN 1 Wonosari. *Jurnal INFORMA Politeknik Indonusa*, *4*(2), 35–42.

Surahmat, S., & Tenggono, A. (2019). Analisis Perbandingan Kinerja Layanan Infrastructure As A Service Cloud Computing Pada Proxmox dan Xenserver. *MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, *19*(1), 9–16. https://doi.org/10.30812/matrik.v19i1.434

Widarma, A., & Siregar, Y. H. (2019). *Analisis Kinerja Teknologi Virtualisasi Server (Study Kasus : Universitas Asahan)*. (Vm), 688–698.