PERANCANGAN N-CLUSTERING HIGH AVAILABILITY WEB SERVER DENGAN LOAD BALANCING DAN FAILOVER

Sumarna[1]; Hafis Nurdin[2]; Felix Wuryo Handono[3]

[1]Program Studi Sistem Informasi
STMIK Nusa Mandiri
http://nusamandiri.ac.id
sumarna@nusamandiri.ac.id

[2]Program Studi Sistem Informasi
STMIK Nusa Mandiri
http://nusamandiri.ac.id
hafis.nnr@nusamandiri.ac.id

[3]Program Studi Sistem Informasi
Universitas BSI
http://bsi.ac.id
felix@bsi.ac.id

Abstract—*The Web is a media for used in the delivery of information and serve as a media competency test in various sector. The growth in the sector of information technology are getting advanced and increasing the access of the web becoming a need improved performance from the web server. The more a web accessible so the time required will be getting slower because of the use of the resource. If it takes place continuously then the web server will be overloaded traffic and cause a web service is down. Clustering is one of the solutions to prevent the occurrence of an overload of traffic on the web server. With the clustering system load will be shared by multiple servers that will be working simultaneously but is from one single system. There are two methods that are contained in clustering system, i.e. the load balancing and failover. Method of load balancing will work by dividing the traffic load evenly to the servers in the cluster. Whereas the method of failover works when there is a system failure at the primary server, then the traffic will be diverted to a backup server so that the web service can still be accessed. After implementing the methods of traffic load is reduced due to some shared server so that access time web servers become more quickly accessible though many clients simultaneously.*

Keywords: *clustering, high availability, web server.*

Intisari—*Web* merupakan media yang dijadikan sarana dalam penyampaian suatu informasi maupun dijadikan sebagai media uji kompentensi di berbagai bidang. Pertumbuhan di bidang teknologi informasi yang semakin canggih dan bertambahnya yang mengakses *web* menjadi suatu kebutuhan peningkatan kinerja dari *web server*. Semakin banyak suatu *web* diakses maka waktu yang dibutuhkan akan semakin lambat dikarenakan penggunaan sumber daya yang tinggi. Jika hal tersebut berlangsung secara terus menerus maka *web server* tersebut akan kelebihan beban trafik dan menyebabkan layanan *web* terhenti. *Clustering* merupakan salah satu solusi untuk mencegah terjadinya kelebihan beban trafik pada *web server*. Dengan sistem *cluster* beban akan dibagi dengan beberapa *server* yang nantinya akan bekerja secara bersamaan tapi merupakan dari satu kesatuan sistem tunggal. Ada dua metode yang terdapat pada sistem *cluster*, yaitu *load balancing* dan *failover*. Metode *load balancing* akan bekerja dengan membagi beban trafik secara merata kepada *server-server* yang tergabung dalam *cluster*. Sedangkan metode *failover* bekerja disaat ada kegagalan sistem pada *server* utama, maka trafik akan dialihkan ke *server* cadangan sehingga layanan *web* tetap dapat diakses. Setelah diterapkannya metode tersebut beban trafik berkurang karena dibagi ke beberapa *server* sehingga waktu akses *web server* menjadi lebih cepat walau diakses banyak klien secara bersamaan

Kata Kunci: *clustering, high availability, web server.*

PENDAHULUAN

Perkembangan teknologi yang bertumbuh semakin canggih memberikan kemudahan kita dalam memperoleh informasi. Hampir semua informasi dapat kita ketahui melalui media *web*.

Dengan banyaknya yang mengakses suatu *web* akan berdampak pada beban *web server* tersebut, sehingga waktu aksesnya menjadi lebih lama bahkan dapat menjadikan *web server* tersebut mati dan tidak dapat melayani permintaan akses. Hal tersebut sangat mungkin terjadi jika *web server* merupakan *server* tunggal dalam melayani permintaan akses (Hasim & Riadi, 2013).

Ada dua metode untuk meningkatkan layanan *web server*, yaitu pengembangan perangkat keras dan pengembangan berbasis perangkat lunak (Juliharta, Supedana, & Hostiadi, 2015). Pengembangan berbasis perangkat lunak memiliki keuntungan lebih, yaitu biaya rendah dan skala yang lebih tinggi dalam ketersediaan dan efektifitas daripada pengembangan perangkat keras (Mondéjar, García-López, Pairot, & Pamies-Juarez, 2013). Pengembangan berbasis perangkat lunak juga membutuhkan penambahan *server* yang nanti berfungsi untuk membantu beban yang terdapat di *server* utama. Teknologi yang akan digunakan untuk meningkatkan kinerja dari *web server* dikenal dengan *clustering*. Dengan *cluster* ada dua metode yang dapat diterapkan, yaitu: *load balancing* dan *failover* yang diharapkan dapat menjaga ketersediaan layanan *web server* dalam memberikan layanannya secara cepat walaupun diakses secara bersamaan oleh banyak klien.

Load balancing bekerja dengan cara mendistribusikan beban kerja secara merata ke beberapa *server* yang bekerja sebagai *back-end server*, sehingga beban kerjanya bisa menjadi lebih ringan. Dengan menerapkan *load balancing* dapat mempersingkat waktu akses terhadap *web server* dan memiliki ketersediaan layanan yang tinggi (Kahanwal & Singh, 2012).

Beberapa algoritma *load balancing* yang dapat diterapkan, yaitu *Round Robin, Least Connecction* dan *Source*. Algoritma *Round Robin* mendistribusikan beban kepada semua *server* yang terdapat pada *cluster* sehingga semua *server* mendapatkan beban yang sama dalam waktu yang sama. Algoritma ini dapat diterapkan jika semua *server* yang terdapat pada *cluster* memiliki kemampuan *processing* yang sama, jika tidak beberapa *server* yang memiliki kemampuan lebih rendah akan mengalami kelebihan beban. Algoritma *Least Connection* meneruskan permintaan dari klien pada *server cluster* yang memiliki koneksi paling sedikit. Sedangkan Algoritma *Source* bekerja dengan melakukan pencatatan alamat *IP address* dari klien yang ingin mengakses ke *web server*. Pencatatan *IP address* tersebut digunakan untuk mengarahkan klien tersebut ke *server* yang sama selama *server* tersebut aktif dan tidak pindah ke *server* lainnya kecuali *server* yang pertama diakses mati. Algoritma *source* bekerja statis, artinya tidak berpengaruh terhadap beban yang dimiliki pada setiap *server cluster* (Sakul, Rumagit, & Sugiarso, 2014).

Sedangkan *failover* biasa dikenal dengan istilah *High Availability*, umumnya bertujuan untuk meningkatkan ketersediaan layanan yang disediakan oleh suatu *server*. Elemen *failover* memiliki *server-server* redundan yang akan digunakan untuk menyediakan layanan ketika salah satu komponen mengalami kegagalan (Kahanwal & Singh, 2012).

Dengan menggabungkan kedua teknologi tersebut, kumpulan *server* yang bertindak sebagai entitas tunggal dalam menyediakan sumber daya dan layanan ke jaringan dengan tujuan menjaga ketersediaan sumber daya bagi klien yang mengakses ketika terjadi kegagalan system ataupun perangkat keras pada *server*.

Penelitian-penelitian terdahulu yang dilakukan oleh (Rosalia, Munadi, & Mayasari, 2016) yang menerapkan *load balancing* algoritma *least connection*. Hasil penelitian didapatkan kesimpulan penggunaan *load balancing* jauh lebih baik dibandingkan *server* tunggal serta dapat menurunkan penggunaan CPU.

Penelitian selanjutnya dilakukan oleh (Juliharta et al., 2015), penelitian ini merancang *web server* dengan teknik *failover clustering.* Kesimpulan yang didapat yaitu: ketersediaan *web server* terjamin dengan menggunakan sistem *failover*, klien tidak akan mengalami gangguan jika ada *server* yang sedang tidak aktif.

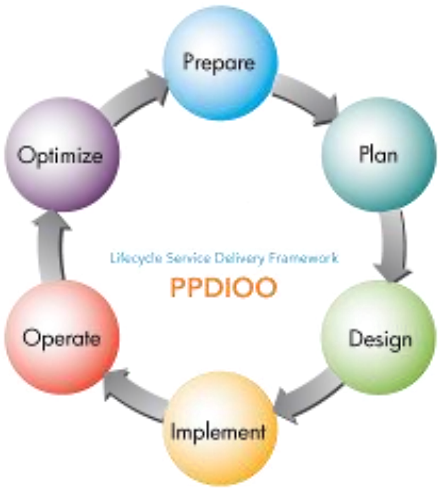
Penelitian lainnya dilakukan oleh (Moniruzzaman, Waliullah, & Rahman, 2014), pada penelitian ini menggunakan *Linux Virtual Server* (LVS) yang digabungkan dengan teknologi penyimpanan bersama untuk merancang *web server cluster*. Hasil penelitian teknologi ini tidak hanya dapat meningkatkan ketersediaan layanan, tetapi juga terhadap keamanan dan performa dari layanan yang diberikan.

Penelitian yang dilakukan oleh (Noviyanto, Kumalasari, & Hamzah, 2015) menggunakan HAProxy sebagai *load balancer*. Algoritma yang diterapkan adalah *round robin*. Permasalahan beban trafik dapat diatasi karena pembagian beban yang merata.

Dari penelitian-penelitian tersebut semua bertujuan untuk meningkatkan ketersediaan layanan dan keamanannya. Dengan menggunakan *load balancing* dapat meringankan beban kerja yang diberikan ke *server*, tetapi disini muncul pertanyaan bagaimana jika *load balancing* yang diterapkan menggunakan 2 lapisan. Dengan menerapkan 2 lapisan *load balancing*, maka terdapat lebih dari satu *cluster* yang dimiliki untuk menyediakan ketersediaan layanan.

BAHAN DAN METODE

Metodologi penelitian yang akan dipakai adalah metode *Cisco Lifecycle Services* yang dikembangkan oleh *Cisco*. Metode ini biasa juga disebut dengan *Prepare, Plan, Design, Implement, Operate, Optimize* (PPDIOO), merupakan metode yang dirancang untuk mendukung perkembangan jaringan komputer (Lubis, Kurniawan, & Yunan, 2018)



Sumber: (NPR, 2018)

Gambar 1. Model PPDIOO

Langkah-langkah penelitian cluster web server dengan metode PPDIOO adalah:

1. Persiapan

Tahap awal dalam penelitian ini adalah persiapan, beberapa hal yang dilakukan yaitu melakukan rencana kerja, pengumpulan data dan bahan pendukung.

2. Perencanaan cluster

Tahap ini dilakukan analisis kebutuhan perangkat keras dan perangkat lunak yang akan digunakan.

Perancangan cluster Sumber : (Sumarna, Nurdin, & Handono, 2019)

3.

Pada tahap perancangan *cluster* adalah melakukan rancangan jaringan secara lebih spesifik dan rinci untuk mendukung implementasi penelitian.

4. Implementasi cluster

Dalam implementasi *cluster* dilakukan instalasi perangkat keras, perangkat lunak serta jaringan komputer.

5. Pengujian cluster

Melakukan pengujian terhadap rancangan, mendeteksi kesalahan dan pemantauan kinerja yang terjadi selama *cluster* dijalankan untuk dijadikan bahan untuk tahap optimasi.

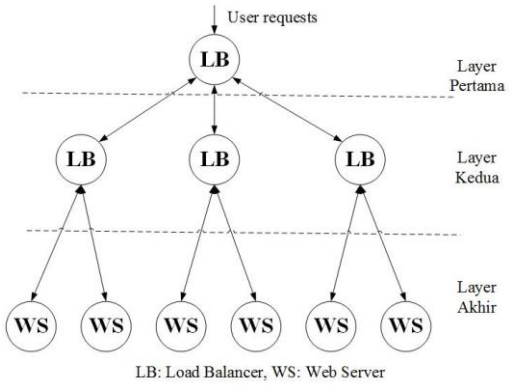
6. Optimasi cluster

Untuk optimasi *cluster* dilakukan identifikasi kesalahan dan melakukan perbaikan-perbaikan agar mengurangi tingkat kegagalan. Tahap optimasi ini dapat dilakukan perancangan ulang jika terjadi banyak kesalahan yang muncul.

HASIL DAN PEMBAHASAN

Untuk menyelesaikan masalah yang terjadi pada beban *web server*, dilakukan pengumpulan data dan solusi dari penelitian-penelitian terdahulu yang akan dijadikan sebagai bahan pendukung dalam menyelesaikan permasalahan. Setelah mempelajari penelitian-penelitian terdahulu didapatkan kesimpulan untuk menyelesaikan permasalahan dibutuhkan tambahan perangkat keras dan perangkat lunak yang nanti akan dikonfigurasi menjadi sebuah *cluster*. Kebutuhan perangkat lunak meliputi: sistem operasi, haproxy sebagai *load balancer* (LB), dan aplikasi *Web Application Performance Testing* (WAPT).

Perancangan *cluster* akan menggunakan 10 unit perangkat *server*. Struktur rancangan *cluster* yang akan diuji adalah dengan struktur *N-Clustering*, dimana akan menggunakan *load balancer* sejumlah 4 unit *server* dan *back-end server* sebanyak 6 unit *server*.

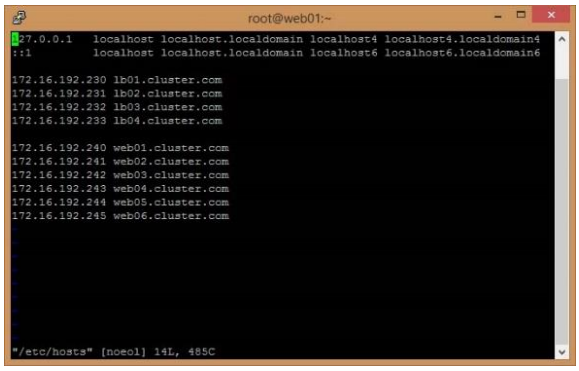


Sumber : (Sumarna et al., 2019)

Gambar 2. Struktur *N-Clustering*

Load balancer dibagi menjadi dua lapisan, struktur tersebut memungkinkan untuk menerapkan dua algoritma *load balancing* bersamaan seperti *round robin* dan *least connection* yang akan menambah performa dan skalabilitas *web server*. Setelah melewati 2 lapisan *load balancer,* permintaan akses akan diteruskan ke lapisan akhir yang merupakan *back-end web server*.

Dalam implemetasi *cluster* digunakan sistem operasi linux CentOS versi 6.3. Setiap *server* akan dikonfigurasi seperti yang ada pada gambar 3 agar setiap *server* bisa saling mengenal dengan menambahkan daftar *hostname* dan *IP Address* semua *server* yang ada dalam *cluster*.



Sumber : (Sumarna et al., 2019)
Gambar 3. Konfigurasi hosts.conf

Sedangkan untuk implementasi *load balancer* akan menggunakan haproxy. Untuk meningkatkan ketersediaan layanan *web server* dalam melayani permintaan maka diterapkan *failover cluster* antar *load balancer*. Untuk konfigurasi yang akan diterapkan ditampilkan pada gambar 4, semua *server* dimasukkan kedalam daftar agar dapat melakukan proses pembagian beban yang diakses oleh klien ke *server load balancing* tersebut.

```
global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        maxconn 4096

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        retries 3
        option  redispatch
        maxconn 2000
      timeout connect 5000000
     timeout client 5000000
     timeout server 5000000

listen load_balance *:80
        mode    http
        stats   enable
        stats   refresh 10s
        stats   auth    monitoring:MONITORIN9

          balance roundrobin

        cookie  JSESSIONID prefix  |
        option  httpclose
        option  forwardfor
        option  httpchk HEAD /check.txt HTTP/1.0
          stats uri     /grafik?status

      server web01 172.16.192.240:80 weight 1 maxconn 512 check
      server web02 172.16.192.241:80 weight 1 maxconn 512 check
      server web03 172.16.192.242:80 weight 1 maxconn 512 check
      server web04 172.16.192.243:80 weight 1 maxconn 512 check
      server web05 172.16.192.244:80 weight 1 maxconn 512 check
      server web06 172.16.192.245:80 weight 1 maxconn 512 check

      appsession JSESSIONID len 52 timeout 3h
```

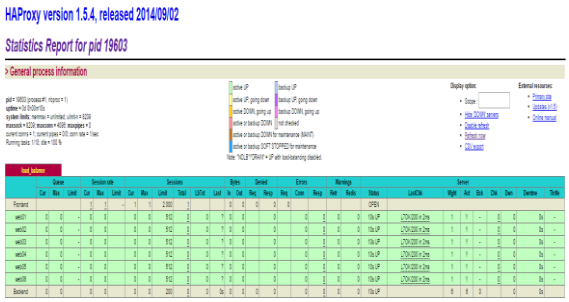
Sumber : (Sumarna et al., 2019)
Gambar 4. Konfigurasi Haproxy.conf

Untuk mengetahui kinerja dari penerapan *cluster web server* dilakukan pengujian dengan menggunakan aplikasi WAPT. Aplikasi WAPT yang akan digunakan adalah versi demo, sehingga pengujian hanya terbatas pada jumlah koneksi yang mengakses. Pengujian akan dilakukan secara bertahap mulai dari pengujian *web server cluster* dengan *load balancing* algoritma *round robin* (rr), pengujian dengan *load balancing* algoritma *least connection* (lc), dan terakhir pengujian struktur *N-Clustering* yang akan menggunakan dua lapisan *load balancing* dan menggunakan dua algoritma. Untuk menguji ketersediaan layanan disimulasikan dengan cara mematikan salah satu atau beberapa *back-end web server* secara langsung saat layanan *web server* aktif melayani permintaan dari klien.

Berikut adalah hasil uji dari beberapa konfigurasi algoritma *load balancing* yang diterapkan pada *web server cluster*:

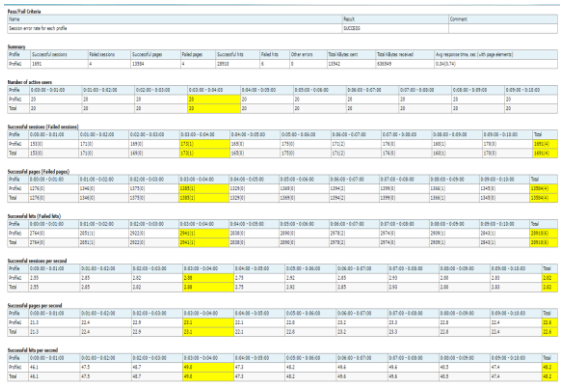
1. Pengujian *cluster* dengan satu *load balancer*

Untuk Pengujian dengan menggunakan satu *load balancer* (LB) dilakukan sebanyak dua kali dengan membedakan algoritma yang dipakai. Sebelum melakukan pengujian, dilakukan pengecekan dengan melihat *monitoring* dari *server* haproxy apakah setiap *back-end server* sudah siap dan sesuai dengan konfigurasi yang akan diujikan seperti pada gambar 5.

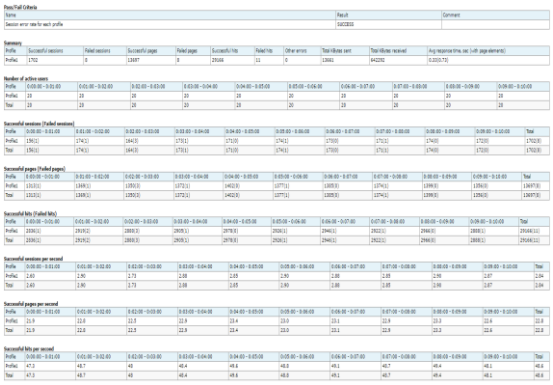


Sumber : (Sumarna et al., 2019)
Gambar 5. Haproxy satu LB

Hasil pengujian pertama menggunakan algoritma *round robin* didapatkan hasil seperti pada gambar 6. Pengujian kedua dengan mmenggunakan algoritma *Least Connection* pada gambar 7. Berdasarkan hasil pengujian tersebut didapatkan seberapa berhasilnya *clustering* untuk dapat menyediakan ketersediaan layanan terhadap banyaknya klien yang mengakses ke *web server* tersebut.



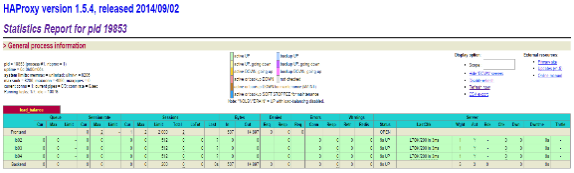
Sumber : (Sumarna et al., 2019)
Gambar 6. Hasil pengujian satu LB (rr)



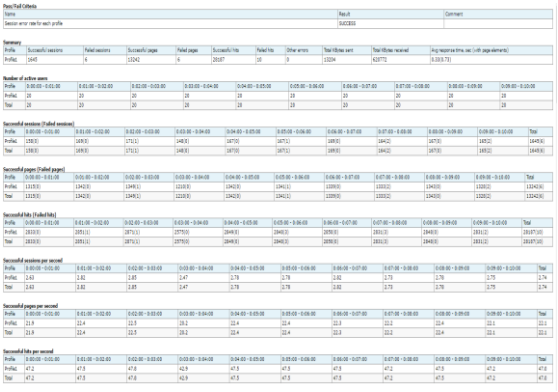
Sumber : (Sumarna et al., 2019)
Gambar 7. Hasil pengujian satu LB (lc)

2. Pengujian *cluster* dua *load balancer*

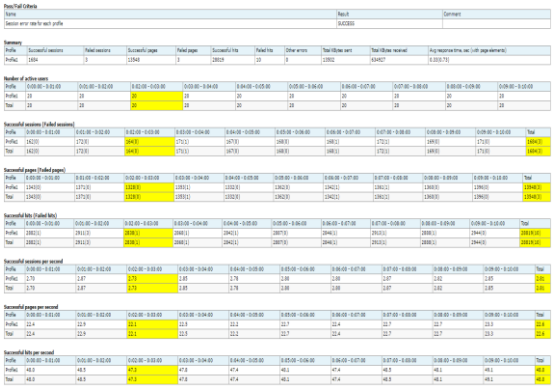
Pengujian pada tahap kedua menggunakan dua lapisan *load balancer*, pengujian juga dilakukan sebanyak dua kali dengan menerapkan beberapa algoritma yang dipakai. Pada gambar 8 terliat jumlah *back-end server* yang berbeda dengan sebelumnya, hal ini dikarenakan *server load balancer* lapisan pertama akan mengarahkan ke *server load balancer* lapisan kedua saja. Permintaan akses yang diterima oleh *server load balancer* lapisan kedua baru akan diteruskan ke *back-end server* yang berfungsi sebagai *web server*. Hasil pengujian *clustering* dengan menggunakan dua lapisan *load balancer* dari setiap algoritma ditampilkan pada gambar 9 dan 10.



Sumber : (Sumarna et al., 2019)
Gambar 8. Haproxy 2 lapisan LB



Sumber : (Sumarna et al., 2019)
Gambar 9. Hasil pengujian LB (rr-lc)



Sumber : (Sumarna et al., 2019)
Gambar 10. Hasil pengujian LB (rr-rr)

Tabel 1. Pengukuran hasil uji

No	Algortima Load Balance	Failed Session	Session /sec	Failed Pages
1	LeastConn	8	2.84	8
2	Round Robin	4	2.82	4
3	Double Round Robin	3	2.81	3

Sumber : (Sumarna et al., 2019)

3. Pengujian ketersediaan layanan

Untuk mengetahui fungsi dari *failover* yang ada di *cluster* dilakukan dengan cara mematikan salah satu atau beberapa *back-end web server*, lalu dilakukan pengecekan apakah layanan *web server* tetap berjalan. Hasilnya seperti yang ditampilkan pada gambar 11, terdeteksi adanya *server* yang mati ditandai dengan warna merah di status *monitoring* haproxy tetapi *server-server* lainnya yang masih aktif melakukan ambil alih tugas layanan dari *server* yang mati tersebut sehingga layanan *web server* tetap berjalan.

Sumber : (Sumarna et al., 2019)
Gambar 11. Hasil pengujian failover

Untuk optimasi *cluster* dilakukan pengujian secara rutin selama *cluster* tersebut dipakai, dilakukan analisis berdasarkan data hasil trafik dan beban yang ada pada setiap *server cluster*. Jika *cluster* sudah melebihi beban, maka dapat ditambahkan beberapa *back-end web server* lagi sampai beban yang didapatkan tidak berlebihan.

KESIMPULAN

Berdasarkan dari data-data diatas didapatkan hasil analisis penerapan struktur *n-clustering* pada *cluster web server*, maka dapat disimpulkan penerapan struktur *n-clustering* pada *cluster web server* dapat mengurangi beban trafik karena pembagian beban dilakukan secara dua langkah *load balancing* sebelum diberikan ke *back-end web server*. Ketersediaan layanan dalam *cluster web server* menjadi lebih tinggi karena adanya proses ambil alih permintaan terhadap *web server* yang tidak aktif. Waktu akses yang dirasakan oleh klien menjadi lebih cepat dan dapat menambah jumlah maksimal klien tanpa harus melakukan penggantian perangkat keras secara menyeluruh.

REFERENSI

Hasim, M., & Riadi, I. (2013). ANALISIS PERBANDINGAN KINERJA SERVER PADA DATA CENTER IIX DAN INTERNATIONAL MENGGUNAKAN TEKNOLOGI CLOUD SERVER. *Sarjana Teknik Informatika*, *1*(2). https://doi.org/10.1017/CBO9781107415324.004

Juliharta, I. G. P. K., Supedana, W., & Hostiadi, D. P. (2015). High Availability Web Server Berbasis Open Source dengan Failover Clustering. *Seminar Nasional Teknologi Informasi Dan Multimedia 2015*, *3*(1), 6–8.

Kahanwal, B., & Singh, T. P. (2012). The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle. *International Journal of Latest Research in Science and Technology*, *1*(2), 183–187.

Lubis, Y. H., Kurniawan, M. T., & Yunan, U. (2018). ANALISIS DAN PERANCANGAN POWER DISTRIBUTION DALAM RANCANGAN SUB DATA CENTER DI DISKOMINFOKABUPATEN BANDUNG DENGAN MENGGUNAKAN STANDAR EN 50600 DAN METODOLOGI PPDIOO LIFE-CYCLE APPROACH. *E-Proceeding of Engineering*, *5*(2), 3130–3139.

Mondéjar, R., García-López, P., Pairot, C., & Pamies-Juarez, L. (2013). CloudSNAP: A transparent infrastructure for decentralized web deployment using distributed interception. *Future Generation Computer Systems*, *29*(1), 370–380. https://doi.org/10.1016/j.future.2011.08.013

Moniruzzaman, Waliullah, & Rahman, S. (2014). A High Availability Clusters Model Combined with Load Balancing and Shared Storage Technologies for Web Servers. *International Journal of Scientific & Engineering Research*, *5*(12). https://doi.org/10.1108/eb057387

Noviyanto, A. B., Kumalasari, E., & Hamzah, A. (2015). Jurnal JARKOM Vol . 3 No . 1 Desember 2015 ISSN : 2338-6313 Jurnal JARKOM Vol . 3 No . 1 Desember 2015 ISSN : 2338-6313. *Jarkom*, *3*(1), 21–31.

NPR. (2018). Solution. Retrieved from http://www.nprsolutions.com/solutions.html

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).

Sakul, A. S., Rumagit, A. M., & Sugiarso, B. A. (2014). Studi Performa PC Cluster. *Teknik Elektro Dan Komputer*, *3*(5).

Sumarna, S., Nurdin, H., & Handono, F. W. (2019). *Laporan Akhir Penelitian - Perancangan N-Clustering High Availability Web Server Dengan Load Balancing Dan Failover*. Jakarta.