IMPLEMENTASI DAN ANALISIS *LOAD BALANCING AS A SERVICE* PADA FTP SERVER MENGGUNAKAN OPENSTACK

IMPLEMENTATION AND ANALYSIS OF LOAD BALANCING AS A SERVICE ON FTP SERVER USING OPENSTACK

Fikri Elang Kesuma[1], Dr. Sofia Naning Hertiana, Ir., M.T.[2], Fardan, S.T., M.Sc.[3]
[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]**fikriekesuma@student.telkomuniversity.ac.id**, [2]**sofiananing@telkomuniversity.ac.id**,
[3]**fardanfnn@telkomuniversity.ac.id**

Abstrak

***Transfer* data dalam jumlah yang besar serta dalam waktu yang cepat sudah menjadi kebutuhan dalam kehidupan di masa sekarang ini. *File Transfer Protocol* (FTP) merupakan suatu layanan yang umum digunakan dalam melakukan *transfer* data. Semakin banyaknya *request* yang diterima dari *client* terhadap FTP server, membuat beban kerja pada FTP server menjadi berlebih sehingga menyebabkan waktu pengiriman menjadi lama bahkan hingga *overload*. Untuk menghindari hal tersebut pada FTP server dibutuhkan metode *load balancing*.**

***Load balancing* merupakan suatu metode yang berfungsi untuk membagi beban kerja server ke server lain yang sedang dalam kondisi *idle*. Dengan *load balancing*, beban *traffic* akan dibebankan kepada beberapa jalur koneksi. Hal ini akan mempercepat proses pengiriman data untuk jumlah besar serta dapat mencegah dari adanya beban yang berlebih di suatu server. *Load balancing as a service* merupakan suatu layanan infrastruktur berbasis *cloud computing* yang terdapat pada *openstack*.**

Telah berhasil dilakukan implementasi penggunaan *LBaaSv2 octavia* pada FTP server dengan menggunakan *openstack*. Dari hasil pengujian diketahui bahwa sistem *load balancer* memiliki performa yang lebih baik dibandingkan dengan single FTP server untuk pengiriman file dalam jumlah yang besar. Hal ini ditunjukan dari selisih nilai rata – rata dari *total time* sebesar 42.6 untuk 1 GB, 76.6 detik untuk 3 GB dan 119.8 detik untuk 6 GB. Pada parmeter *throughput* memiliki selisih sebesar 5.04 MB/s untuk 1 GB, 2.31 MB/s untuk 3 GB dan 1.55 MB/s untuk 6 GB. Kemudian berdasarkan hasil rata - rata pengujian *CPU usage, load balancer* juga memiliki performa yang lebih baik dibandingkan dengan single FTP server dengan adanya selsish rata – rata CPU sebesar 14.16% untuk 100 MB, 25.38% untuk 200 MB, 30.82% untuk 500 MB, 34.63% untuk 1 GB, 44.75% untuk 3 GB dan 60.63% untuk 6GB.

Kata kunci : ***FTP, Load Balancing, Cloud Computing, Openstack***

Abstract

Transferring large amounts of data in a fast time has become a necessity in the current era. File Transfer Protocol (FTP) is a service that is commonly used in transferring data. The increasing number of requests received from the client to the FTP server, makes the workload on the FTP server excessive, causing long delivery times and even overload. To avoid this, the FTP server requires a load balancing method.

Load balancing is a method that functions to share server workload to other servers that are in an idle condition. With load balancing, the traffic load will be charged to several connection lines. This will speed up the process of sending data for large amounts and can prevent excessive load on a server. Load balancing as a service is a cloud computing based infrastructure service that found on Openstack.

The implementation of LBaaSv2 octavia on the FTP server using openstack has been successfully implemented. From the test results, it is known that the load balancer system has better performance than a single FTP server for sending large amounts of files. This is indicated by the difference in the average value of the total time of 42.6 for 1 GB, 76.6 seconds for 3 GB and 119.8 seconds for 6 GB. The throughput parmeter has a difference of 5.04 MB / s for 1 GB, 2.31 MB / s for 3 GB and 1.55 MB / s for 6 GB. Then based on the average results of CPU usage testing, load balancers also have better performance compared to a single FTP server with an average CPU selsish of 14.16% for 100 MB, 25.38% for 200 MB, 30.82% for 500 MB, 34.63% for 1 GB, 44.75% for 3 GB and 60.63% for 6GB.

Keywords: ***FTP, Load Balancing, Cloud Computing, Openstack***

1. Pendahuluan

FTP masih banyak digunakan karena dapat digunakan untuk mengirim data dalam jumlah yang besar serta lebih efisien [1] yang dapat mendukung pada era ini karena *transfer* data menjadi suatu kebutuhan yang penting dikarenakan banyak kegiatan yang sudah dapat dikerjakan dari rumah ataupun dari tempat yang jauh. Kebutuhan akan *transfer* data dalam jumlah yang besar akan menyebabkan suatu server kelebihan beban, untuk beberapa kasus beberapa menit waktu henti server dapat mengakibatkan hilangnya peluang, ketidakpuasan pelanggan, dan hilangnya pendapatan[2]. Untuk mengantisipasi hal tersebut maka dibutuhkan metode *load balancing*.

Berdasarkan dari penelitian sebelumnya tentang implementasi *load balancing as a service* pada *Openstack*[3] dengan menggunakan layanan web server dan melakukan perbandingan kinerja antara algoritma *round robin*, least *connection*, dan *source ip*. Diketahui penggunaan *load balancing* algoritma *round robin* dapat meningkatkan kinerja server dengan baik. Namun pada penelitian tersebut menggunakan *LBaaSv1 neutron*, dimana pada versi tersebut telah usang dan sekarang digantikan oleh *LBaaSv2 octavia*. Oleh karena itu dilakukan penelitian tentang layanan FTP server yang memiliki peranan penting dalam melakukan *tranfer* data. Digunakan *openstack* sebagai *cloud computing* karena merupakan sutau *software* yang *open source* serta memiliki layanan infrastruktur *load balancing as a service* didalamnya.

Pada penelitian kali ini akan dilakukan implementasi penggunaan *load balancing* pada layanan FTP server dengan menggunakan *LBaaSv2 octavia* di *openstack* serta menganalisa kinerja dari LBaaSv2 octavia terhadap FTP server. Dengan memanfaatkan *LBaaSv2*, diharapkan dapat mempercepat waktu respon server serta dapat mencegah dari terjadinya kelebihan beban pada suatu server.

2. Dasar Teori

2.1 FTP Server

FTP adalah sebuah layanan yang dapat memberi akses kepada *client* dan server untuk saling malakukan pertukaran data [4]. FTP server merupakan suatu server yang menjalankan layanan FTP sebagai media pertukaran data. *Client* merupakan sebuah perangkat yang akan melakukan *request* ke FTP server untuk melakukan transfer data. FTP adalah protokol *client* - server. *Client* akan meminta file, dan server memberikannya. Dalam membuat suatu koneksi FTP juga membutuhkan dua saluran dasar , yaitu saluran perintah (sebagai instruksi dan respons) dan saluran data (melakukan proses distribusi data). Untuk membuat koneksi antara pengguna dan server FTP biasanya memakai port nomor 20 dan 21 sebagai mode komunikasi *default*-nya, FTP juga memiliki dua mode koneksi FTP dasar, yaitu aktif dan pasif[4].

2.2 *Load Balancing*

Load balancing merupakan suatu metode sebagai penyeimbang dalam jaringan, yaitu dengan melakukan proses pendistribusian beban kerja di beberapa komputasi node untuk memastikan bahwa tidak ada satu node yang kelebihan beban dengan tugas yang sedang berlangsung saat node tidak bekerja. *Load Balancing* adalah suatu teknik yang memastikan tidak ada server atau node komputer di jaringan yang digunakan secara berlebihan, *load balancing* menggunakan mekanisme selektif dengan mengirim pekerjaan masuk ke node yang menganggur untuk mencegah mereka pergi ke node yang sudah kelebihan beban. *Load balancing* sebagai penyeimbangan beban memiliki peranan penting dalam pusat data di mana tujuan sistem adalah untuk melayani permintaan *client* yang masuk dengan waktu penyelesaian paling sedikit[5]. Pada saat *client* melakukan *request* ke server, maka *load balancing* akan membagi permintaan tersebut ke beberapa server agar tidak mengalami kelebihan beban. Dengan membagi pekerjaan ke beberapa server membuat kinerja server akan menjadi lebih ringan serta dapat membuat waktu proses pemenuhan dari permintaan *client* menjadi lebih cepat.

2.3 Algoritma *Round Robin*

Pada algoritma ini setiap sistem mengambil bagian yang sama dari sesuatu pada gilirannya. Algoritma ini mendistribusikan beban ke node dalam urutan *round-robin* dan menetapkan beban yang sama untuk setiap node secara melingkar. Setiap node pada jaringan memelihara sebuah *load index independent* dari *remote* node. Algoritma *round-robin* merupakan algoritma yang sederhana sederhana dan tidak memerlukan komunikasi antar proses yang memberikan kinerja terbaik untuk aplikasi tujuan khusus[5]. Algoritma *round robin* akan mendistribusikan beban *request* dari *client* secara merata dengan menetapkan beban yang sama untuk setiap server. Dengan menggunakan algoritma *round robin load balancer* akan mengirimkan *request* secara berurutan. *Request 1* akan dikirimkan ke FTP server 1 lalu *request 2* akan dikirimkan ke FTP server 2 dan *resquest 3* akan dikirimkan ke FTP server 3. Request 4 akan dimulai kembali pada FTP server 1 begitu pula pada *request – request* selanjutnya.

2.4 *Cloud Computing*

Cloud computing adalah suatu teknologi yang dapat memberikan akses jaringan di mana-mana melalui internet dan dapat disediakan serta dirilis dengan cepat tanpa perlu adanya interaksi dengan penyedia layanan. Layanan yang disediakan meliputi server, media penyimpanan, database, software dan lain sebagainya sehingga memudahkan pengguna untuk mengakses dan mengelola data.yang dapat dengan cepat disediakan dan dirilis dengan upaya manajemen minimal atau interaksi penyedia layanan[6]. *Cloud Computing* juga dapat diartikan sebagai sebuah teknologi yang menjadikan internet sebagai *database* untuk mengelola data dan juga aplikasi pengguna. *Cloud Computing* memberikan kemudahan kepada penggunanya untuk dapat menjalankan suatu program tanpa harus melakukan instalasi aplikasi terlebih dahulu. Terdapat tiga layanan yang ada pada cloud computing[7] yaitu:

a. Software as a Service (SaaS)

SaaS adalah suatu layanan dari *Cloud Computing* dimana pelanggan dari SaaS dapat menggunakan software atau aplikasi yang telah disediakan oleh *cloud provider*. Pelanggan cukup tahu bahwa perangkat lunak bisa berjalan dan bisa digunakan dengan baik. Contoh dari layanan SaaS ini antara lain adalah Office365, GoogleDocs, Adobe Creative Cloud.

b. Platform as a Service (PaaS)

PaaS adalah layanan dari *Cloud Computing* dalam bentuk *platform* sehingga dapat dimanfaatkan untuk membantu melakukan *testing, development*, serta implementasi. Dapat digambarkan seperti menyewa "rumah" berikut lingkungannya, untuk menjalankan aplikasi yang telah dibuat. Pelanggan tidak perlu memikirkan untuk menyiapkan "rumah" dan memelihara "rumah" tersebut. Pemeliharaan "rumah" ini seperti sistem operasi, *network*, *database engine, framework* aplikasi, dll akan menjadi tanggung jawab dari penyedia layanan. Contoh penyedia layanan PaaS: Amazon Web Service, Microsoft Azure, dan GoogleApp Engine.

c. Infrastructure as a Service (IaaS)

IaaS adalah layanan dari *Cloud Computing* dengan memberikan kendali penuh terhadap infrastruktur yang digunakan, mulai dari *server cloud, network*, sistem operasi hingga *storage,* sehingga dapat menjalankan suatu *software* secara bebas. Untuk lebih mudahnya, layanan IaaS ini adalah seperti menyewa komputer yang masih kosong. Kita sendiri yang mengkonfigurasi komputer ini untuk digunakan sesuai dengan kebutuhan kita dan bisa kita install sistem operasi dan aplikasi apapun diatasnya. Contoh penyedia layanan IaaS : AWS, Rackspace Cloud, Windows Azure.

2.5 Openstack

Openstack merupakan seperangkat *software tools* untuk membangun dan mengelola *cloud computing* untuk *public* dan *private clouds*. *OpenStack* dikelola oleh *Openstack Foundation* yang dibentuk pada tahun 2012 sebagai badan independen yang menyediakan sumber daya bersama untuk melindungi, memberdayakan, dan mempromosikan *OpenStack* pada komunitas di sekitarnya. *Openstack* adalah *software open source*, yang berarti bahwa siapa pun dapat mengakses kode sumber dan membuat perubahan atau modifikasi apa pun menyesuaikan lingkungan *cloud* mereka sendiri[8]. Semua layanan menyediakan akses melalui *Application Programming Interface* (API) sehingga semua sumber daya dapat dikelola melalui satu *dashboard (Horizon)* dengan demikian, memberikan kontrol kepada administrator *cloud Openstack*, juga memberdayakan pengguna untuk penyediaan sumber daya melalui antarmuka web.

2.6 FileZilla

FileZilla merupakan suatu *software opensource* FTP klien yang dapat digunakan untuk memudahkan kita dalam melakukan *upload, download* serta mengelola file dari suatu server yang kita miliki [9]. Filezilla juga memiliki beberapa fitur yang dapat digunakan untuk mengukur kinerja suatu server. Fitur tersebut meliputi *queued files* yang menunjukan status setiap antrian file secara *realtime*, *failed transfer* untuk melihat berapa banyak file yang gagal untuk dikirimkan lalu *successful transfer* yang dapat menampilkan semua file yang sudah terkirim serta waktu awal hingga berakhir setiap file yang dikirim sehingga dapat dihitung nilai *total time*.

2.7 Htop

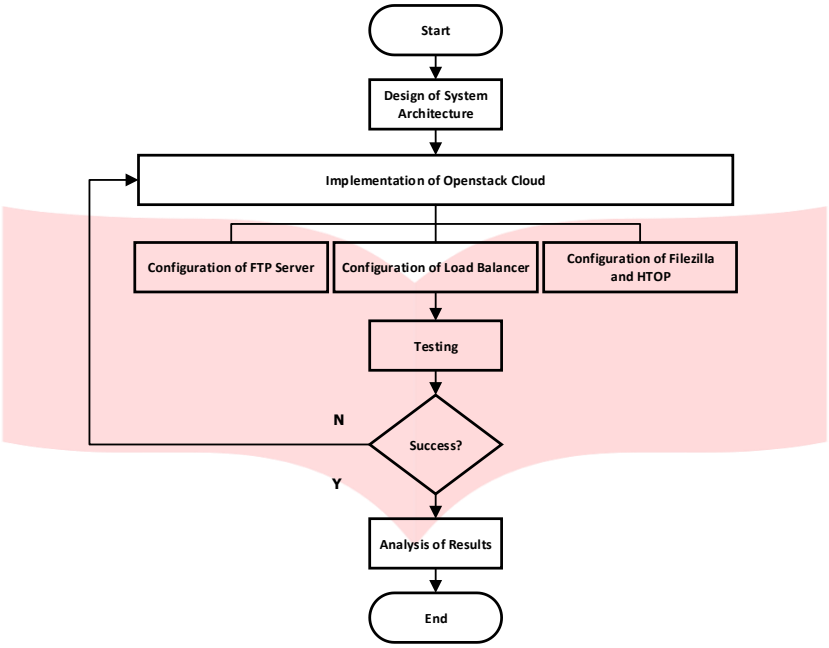
Htop adalah suatu aplikasi pemantauan proses *realtime* yang interaktif untuk sistem operasi Linux dengan distro Debian atau Ubuntu. Htop juga alternatif dari perintah Top, yang merupakan alat pemantauan proses *default* yang dilengkapi dengan pra-instalasi pada semua sistem operasi Linux. Htop tidak jauh berbeda dengan command top, namun htop lebih interaktif dan lebih mudah digunakan dan htop lebih *user friendly* karena dapat menampilkan penggunaan dari CPU, memori, *swap* dan *resource* yang lain[10].

2.8 ProFTPD

ProFTPD atau Pro FTP Daemon merupakan salah satu contoh aplikasi server FTP yang berfungsi sebagai media *transfer file* antar komputer [11]. Prinsip kerja dari suatu FTP Server adalah dengan menggunakan autentikasi *standard* seperti *username* dan *password* untuk *login* dan mengakses suatu file pada direktori server. Komputer yang sudah terpasang dan dikonfigurasi ProFTPD akan berlaku menjadi server, sedangkan komputer lain dapat berlaku sebagai *FTP Client* atau pengguna.

3. Pembahasan

3.1 Diagram Alir Pengerjaan Sistem

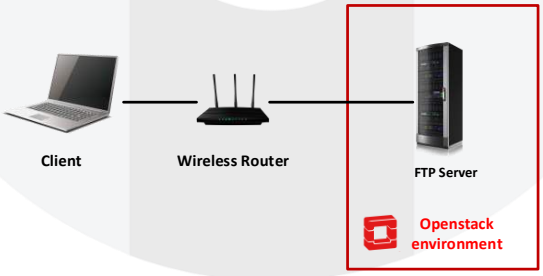


Gambar 1. Diagram Alir Perancangan Sistem

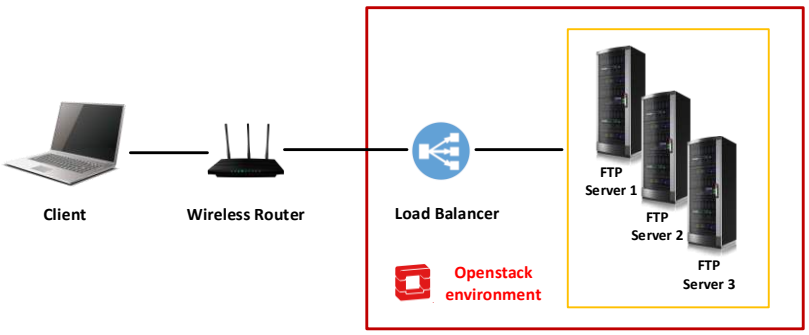
Berdasarkan gambar 1. *platform* yang akan digunakan untuk melakukan metode *Load Balancing as a Service* adalah *Openstack Cloud*. *Load Balancing as a Service* yang telah dibuat pada *openstack cloud* akan dikonfigurasi dengan FTP server. Setelah LbaaS berhasil terkonfigurasi dengan FTP Server maka akan dilakukan pengujian terhadap performansi dari LbaaS dan melakukan analisis terhadap hasil pengujian dengan berdasarkan parameter *total time*, *throughput*, *CPU usage* serta *packet loss*.

3.2 Desain Topologi Jaringan

Dalam topologi jaringan pada penelitian ini menggunakan 2 buah laptop yang akan digunakan untuk membangun *openstack environment*, laptop pertama terdiri dari *network node* untuk menyediakan jaringan agar setiap intance dapat berkomunikasi dengan jaringan luar dan *compute node* yang digunakan sebagai platform virtualisasi dari setiap instance. Kemudian untuk laptop kedua terdiri dari *controller node* yang berfungsi untuk mengontrol setiap layanan yang ada didalam *openstack*.



Gambar 2. Desain topologi jaringan single server FTP

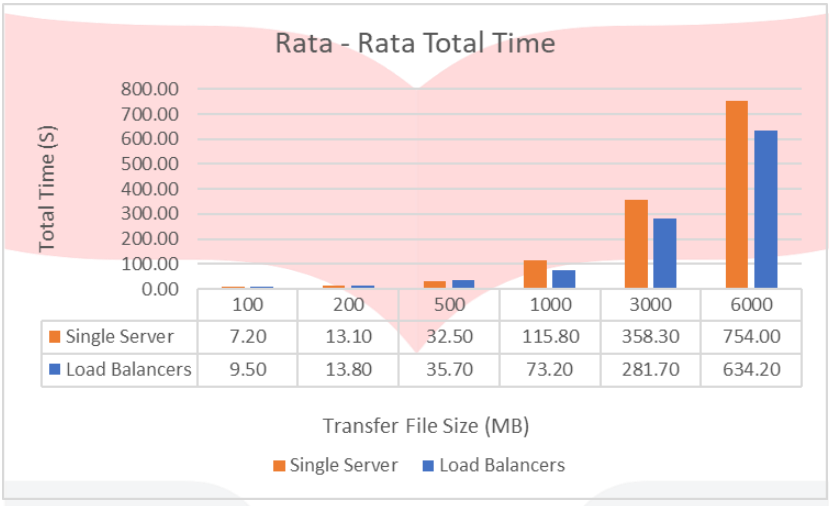


Gambar 3. Desain Topologi Jaringan Load Balancer

Berdasarkan gambar 2. dan gambar 3. dapat dilihat bahwa terdapat beberapa komponen seperti *client*, *wireless router*, dan *openstack environment* dengan fungsi dari masing – masing sebagai berikut:Server merupakan satu kesatuan dari server *openstack*. Server tersebut berjalan diatas *virtual machine* berupa VMWare.

a. Load balancer yang berada didalam openstack envirornment dan berjalan diatas virtual machine berupa VMWare.
b. FTP server berupa ProFTPD yang berada pada satu cluster *openstack*.
c. *Client* yang telah terinstall aplikasi FileZilla yang berfungsi untuk membangkitkan *request* dari *client*.
d. Satu buah *wireless router* yang digunakan sebagai media penghubung antar *device*.
e. *Network Interface Card (NIC) virtual* pada masing-masing VM dengan adapter *bridge*.

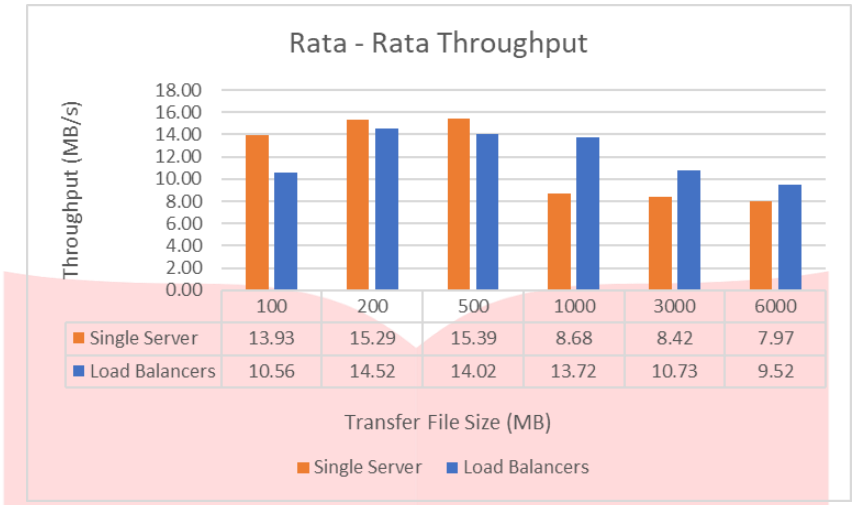
3.3 Pengujian *Total Time*



Gambar 4. Hasil Pengujian *Total Time*

Gambar 4. menunjukkan *total time* FTP server dan *load balancer* untuk melayani FTP *request* yang diajukan oleh *client*. Nilai dari *total time* ini akan semakin baik jika nilainya semakin kecil. *Single* FTP server memiliki performa yang lebih baik dibandingkan dengan *Load Balancer* untuk pengiriman file dalam jumlah yang kecil. Berdasarkan pengujian dengan beban file sejumlah 100 MB – 500 MB didapatkan selisih *total time* dari *single* FTP server terhadap *load balancer* sebesar 2.3 detik untuk file sebesar 100 MB, kemudian 0.7 detik untuk file sebesar 200 MB dan 3.2 detik untuk file sebesar 500 MB. untuk beban file dalam jumlah yang besar *load balancer* dengan 3 server memiliki performa yang lebih baik dibandingkan dengan *single* FTP server. Berdasarkan pengujian dengan beban file sejumlah 1 GB – 6 GB didapatkan selisih *total time* dari *load balancer* terhadap *single FTP server* sebesar 42.6 detik untuk file sebesar 1 GB, kemudian 76.6 detik untuk file sebesar 3 GB dan 119.8 detik untuk file sebesar 6 GB. Perbedaan performa tersebut dipengaruhi oleh bertambahnya jumlah hop pada jaringan, yang pada awalnya bisa langsung menuju ke server akan tetapi harus melalui *load balancer* terlebih dahulu.

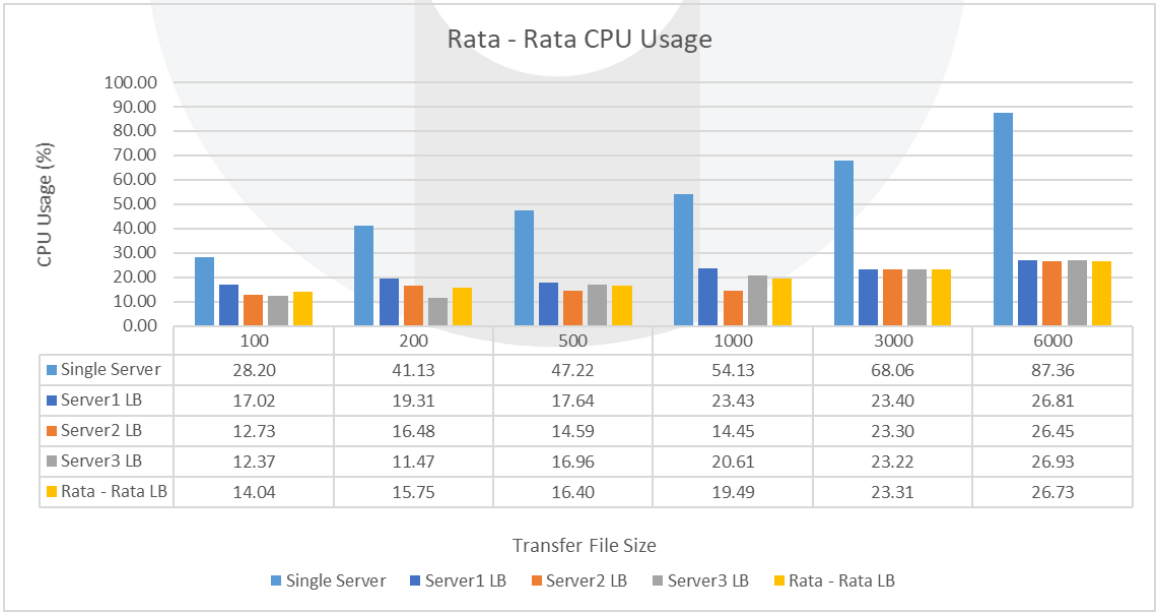
3.4 Pengujian *Throughput*



Gambar 5. Hasil Pengujian *Throughput*

Gambar 5. menunjukkan hasil dari *throughput single* FTP server dan *load balancer* untuk melayani FTP *request* yang diajukan oleh client. Nilai dari *throughput* ini akan semakin baik jika nilainya semakin besar. Sama seperti dengan *total time*, *single* FTP server memiliki performa yang lebih baik dibandingkan dengan *load balancer* untuk pengiriman file dalam jumlah yang kecil. Berdasarkan pengujian dengan beban file sejumlah 100 MB, 200 MB hingga 500 MB didapatkan selisih *throughput* dari *single* FTP server terhadap *load balancer* sebesar 3.37 MB/s untuk file sebesar 100 MB, kemudian 0.77 MB/s untuk file sebesar 200 MB dan 1.37 MB/s untuk file sebesar 500 MB. Sama juga seperti pada *total time* untuk beban file dalam jumlah yang besar *load balancer* dengan 3 server memiliki performa yang lebih baik dibandingkan dengan *single* FTP server. Berdasarkan pengujian dengan beban file sejumlah 1 GB, 3 GB, 6 GB didapatkan selisih *throughput* dari *load balancer* terhadap *single* FTP server sebesar 5.04 MB/s untuk file sebesar 1 GB, kemudian 2.31 MB/s untuk file sebesar 3 GB dan 1.55 MB/s untuk file sebesar 6 GB. Sama seperti pada parameter *total time* perbedaan performa tersebut juga dipengaruhi oleh bertambahnya jumlah hop pada jaringan, yang pada awalnya bisa langsung menuju ke server akan tetapi harus melalui *load balancer* terlebih dahulu. Sehingga untuk file berukuran kecil yang sebenarnya masih dapat diatasi dengan baik oleh satu server sedangkan pada *load balancer* harus melewati satu hop tambahan.

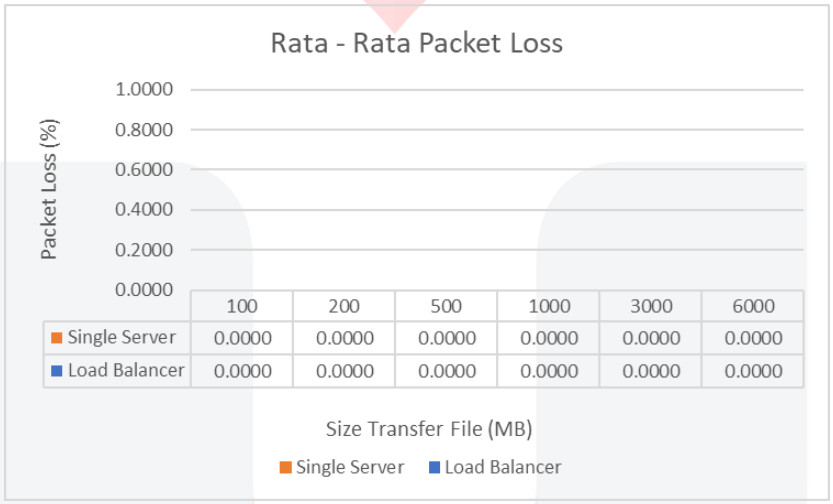
3.5 Pengujian CPU *Usage*



Gambar 6. Hasil Pengujian *CPU Usage*

Gambar 6. menunjukkan hasil *CPU usage* dari *single FTP server* dan *load balancer* dalam melayani *request* yang diajukan oleh *client*. Nilai dari *CPU usage* ini akan semakin baik jika persentasenya semakin kecil. *Load balancer* dengan 3 server memiliki performa yang lebih baik jika dibandingkan dengan performa *single FTP server*. Berdasarkan pengujian dengan beban file sejumlah 100 MB didapatkan nilai *CPU usage* dari *load balancer* sebesar 14.04% dan *single server* FTP sebesar 28.2%, untuk beban file sejumlah 200 MB didapatkan nilai dari *CPU usage* sebesar 15.75% dan *single server* FTP sebesar 41.13%, lalu untuk file sejumlah 500MB didapatkan nilai *CPU usage* dari *load balancer* sebesar 16.4% dan *single server* FTP sebesar 47.22%, lalu pada file sejumlah 1GB didapatkan nilai *CPU usage* dari *load balancer* sebesar 19.49% dan *single server* FTP sebesar 54.13%, kemudian untuk file sejumlah 3GB didapatkan nilai *CPU usage* dari *load balancer* sebesar 23.31% dan *single server* FTP sebesar 68.06% dan yang terkahir untuk file sejumlah 6GB didapatkan nilai CPU usage dari *load balancer* sebesar 26.73% dan *single server* FTP sebesar 87.36%. Perbedaan performa tersebut dipengaruhi oleh adanya pembagian beban ke beberapa server pada *load balancer* seperti yang sudah ditampilkan pada gambar 4.26. Jadi pada *load balancer, request* dari *client* dibagi kedalam tiga FTP server sehingga server - server tidak terlalu terbebani seperti pada *single FTP server* yang memiliki *CPU usage* mencapai 87% untuk pengiriman file dalam jumlah yang besar.

3.6 Pengujian *Packet Loss*



Gambar 7. Hasil Pengujian Packet Loss

Gambar 7. menunjukkan hasil *packet loss* dari sistem FTP server dan *load balancer* dalam melayani *request* yang diajukan oleh *client*. Nilai dari *packet loss* ini akan semakin baik jika persentasenya semakin kecil atau mendekati 0% yang berarti tidak ada data yang gagal untuk dikirim. Pada pengujian ini *single FTP server* dan *load balancer* memiliki performa yang seimbang dengan tidak adanya *packet loss* selama melayani *request* dari *client*. Hal ini dipengaruhi oleh penggunaan protokol TCP yang berarti apabila suatu file gagal untuk dikirim maka pengiriman tersebut akan diulang hingga proses *transfer* tersebut berhasil dilakukan oleh *client* terhadap server.

4. Kesimpulan dan Saran

4.1 Kesimpulan

Berdasarkan hasil penelitian selama pengerjaan Tugas Akhir ini dapat ditarik kesimpulan bahwa:

a. Berhasil melakukan implementasi *load balancing as a service* pada FTP server menggunakan *Openstack Cloud*.
b. Kinerja *load balancing as a service* pada FTP server:
    1. Hasil pengujian *total time load balancer* memiliki performa yang lebih baik untuk pengiriman file dalam jumlah yang besar dengan adanya selisih waktu sebesar 42.6 detik untuk file sebesar 1 GB, kemudian 76.6 detik untuk file sebesar 3 GB dan 119.8 detik untuk file sebesar 6 GB.
    2. Hasil pengujian *throughput load balancer* memiliki performa yang lebih baik untuk pengiriman file dalam jumlah yang besar dengan adanya selisih sebesar 5.04 MB/s untuk file sebesar 1 GB, kemudian 2.31 MB/s untuk file sebesar 3 GB dan 1.55 MB/s untuk file sebesar 6 GB.

3. Hasil pengujian *CPU usage load balancer* memiliki performa yang lebih baik dibandingkan dengan *single FTP server* dengan adanya rata – rata *CPU usage* dengan selisih sebesar 14.16% untuk pengiriman file sebesar 100 MB, selisih sebesar 25.38% untuk pengiriman file sebesar 200 MB, selisih sebesar 30.82% untuk pengiriman filesebesar 500 MB, selisih sebesar 34.63% untuk pengiriman file sebesar 1 GB, selisih sebesar 44.75% untuk pengiriman file sebesar 3 GB dan selisih sebesar 60.63% untuk pengiriman file sebesar 6GB
4. Hasil pengujian *packet loss load balancer* dan *single FTP server* memiliki performa yang sama dengan tidak adanya paket yang gagal dikirim oleh *load balancer* dan *single FTP server*.

4.2 Saran

Berikut ini merupakan beberapa saran yang dapat diberikan dalam penelitian Tugas Akhir ini.

a. Pada penelitian selanjutnya agar ditambahkan fitur sinkronisasi antar server agar setiap server yang ada dapat memiliki file yang sama untuk memudahkan dalam proses *download* dari server ke client.
b. Untuk memudahkan dalam konfigurasi setiap node, dapat menggunakan satu *device* yang memiliki jumlah memori besar dan jumlah *core* yang banyak agar semua node dapat dikonfigurasi didalam satu *device*.

Daftar Pustaka:

[1] Mansour, Abdoulaye, Abdullahi, 2016, “IUT FTP SERVER”, Bangladesh : Islamic University of Technology.
[2] Load Balancing File Transfer Services, [online], https://www.jscape.com/blog/load-balancing-file-transfer-services, diakses 11 Juni 2023.
[3] Ivan Hidayah, 2019, “Implementasi High-Availability Web Server Menggunakan Load Balancing As A Service Pada Openstack Cloud”, Bandung : Universitas Telkom.
[4] Rico Satria, 2009, “Membangun FTP Server Pada Windows Server 2008 Di Lembaga Penyiaran Publik TVRI”, Bogor : Institut Pertanian Bogor.
[5] Momodou Njie, 2018, “Dynamic Load Balancing With Openstack Cloud”, Venezia : Università Ca’ Foscari Venezia.
[6] Lee Badger, Tim Grance, Robert Patt-Corner, Jeff Voas , 2012, “Cloud Computing Synopsis and Recommendations”, Gaithersburg : National Institute of Standards and Technology.
[7] Alex Budiyanto, 2012, “Pengantar Cloud Computing”, cloud indonesia.
[8] Openstack [online] https://www.openstack.org/, diakses pada 13 November 2021
[9] FileZilla, [online], https://wiki.filezilla-project.org/Main_Page, diakses pada 04 Februari 2023.
[10] Amalia Intan Safura, Adityas Widjajarto, Avon Budiono, 2019, “Analisis Penggunaan Memori Sistem Pada Migrasi Aplikasi Dalam Linux Container (LXD) Menggunakan LXD API”, Bandung : Universitas Telkom.
[11] ProFTPD [online] http://www.proftpd.org/, diakses pada 11 Desember 2021