JUSTIN
Jurnal Sistem dan Teknologi Informasi
p-ISSN : 2460-3562 / e-ISSN : 2620-8989

Vol. 8, No. 1, Januari 2020

Analisa Perbandingan Performa *Openstack* dan *Apache Cloudstack* dalam Model *Cloud Computing* Berbasis *Infrastructure As a Service*

Hari Triyanto[#1], Arif Bijaksana Putra Negara[#2], Muhammad Azhar Irwansyah[#3]

[#]*Program Studi Informatika Fakultas Teknik Universitas Tanjungpura*
Jl. Prof. Dr. H. Hadari Nawawi, Kota Pontianak, 78115
[1]hari.triyanto22@gmail.com
[2]arifbpn@gmail.com
[3]irwansyah.azhar@gmail.com

Abstrak*— Efisiensi pemanfaatan sumber daya komputasi dapat di lakukan dengan melakukan virtualisasi pada mesin fisik komputer. Pada *cloud computing*, sumber daya seperti CPU, memori, *storage* dan jaringan dapat dipandang sebagai suatu layanan dengan virtualisasi sebagai jantungnya. Openstack dan Cloudstack merupakan salah satu *opensource* untuk membangun *cloud computing* dengan model IaaS. Penelitian ini bertujuan untuk membandingkan Openstack dan Cloudstack dalam perancangan *private cloud computing* dengan aspek pengujian *web server*, komputasi, oltp *database* dan jaringan. Pengujian menggunakan metrik skalabilitas dengan metode pengujian *overhead* dan *linearity*. Implementasi dilakukan pada satu *server* dengan menggunakan satu *router* dan satu laptop sebagai *client*. *Client* bertugas untuk menjalankan pengujian menggunakan *tools* seperti Httperf, Sysbench dan Iperf. Sejumlah beban pengujian diberikan untuk tiap tiap *instance* berdasarkan skenario pengujian yang telah di buat. Hasil pengujian menunjukan bahwa waktu eksekusi yang diperlukan untuk mengakses *web server* dan komputasi pada *instance* Openstack lebih rendah daripada *instance* Cloudstack. Sedangkan pada pengujian oltp *database* dan jaringan, hasil pengujian menunjukan bahwa *instance* pada Cloudstack lebih unggul dengan waktu eksekusi oltp *database* yang lebih rendah, *throughput* yang lebih tinggi serta *jitter* yang lebih rendah. Oleh karena itu, Openstack unggul dalam penggunaan aplikasi berbasis web dan komputasi sedangkan Cloudstack unggul dalam aplikasi dengan transaksi *database* yang tinggi serta membutuhkan jaringan yang baik.

***Kata kunci*— IaaS, Openstack, Cloudstack, Cloud Computing, Studi Komparatif**

I. PENDAHULUAN

Perkembangan komputasi telah beralih dari komputasi tradisional ke arah komputasi awan. Komputasi awan (*cloud computing*) adalah sebuah model komputasi baru yang membawa disiplin, teknologi dan model bisnis untuk menyajikan informasi teknologi sesuai permintaan. *Cloud computing* merupakan perkembangan dari komputasi tradisional. Pada komputasi tradisional, seluruh sumber daya seperti CPU, memori, penyimpanan dan jaringan digabung menjadi satu untuk menghasilkan daya komputasi yang besar. Sedangkan pada *cloud computing*, sumber daya yang besar tersebut dipecah menjadi beberapa mesin virtual agar penggunaannya lebih efisien.

Pada *cloud computing* terdapat model pengembangan *public cloud* dan *private cloud*. *Public cloud* memberikan kemudahan pada pengguna, organisai ataupun perusahaan untuk mengeluarkan biaya sesuai dengan sumber daya yang digunakan serta terbebas dari biaya perancangan infrastruktur. Pengguna tersebut dapat mengatur jumlah pembiayaan yang dikeluarkan agar proses bisnis tetap berjalan. Akan tetapi semakin berkembangnya kebutuhan dari pengguna maka diperlukan sumber daya yang lebih besar pula. Selain itu, permasalahan seperti keamanan data, kontrol dan perawatan menjadi salah satu poin yang diperhitungkan dalam menggunakan *public cloud*. Penggunaan p*rivate cloud* merupakan salah satu solusi dalam menjaga keamanan data vital pengguna. Selain itu, model *private cloud* memungkinkan pengguna untuk melacak permasalahan dalam menjalankan sistem.

Dalam perancangan *private cloud* terdapat beberapa perangkat lunak *opensource* yang dapat digunakan untuk perancangan private cloud seperti *Eucalyptus*, *Opennebula*, *Openstack* dan *Cloudstack*. Perangkat lunak tersebut melakukan manajemen terhadap pengelolaan CPU, memori, penyimpanan dan jaringan. Openstack dan Apache Cloudstack merupakan perangkat lunak yang dapat digunakan untuk membangun *private cloud computing* berbasis IaaS. Openstack dan Apache Cloudstack digunakan oleh organisasi, perusahaan, pemerintah dan akademisi. Openstack dan Apache Cloudstack dapat digunakan pada sistem operasi Ubuntu, Centos, Redhat serta berbagai *hypervisor* seperti KVM dan Xen sebagai jantung virtualisasinya. Ubuntu memiliki dukungan komunitas yang besar sehingga memudahkan

Submitted 18-03-2019; Revised 26-07-2019; Accepted 06-01-2020

pengguna dalam melakukan konfigurasi. Dalam virtualisasi menggunakan *hypervisor*, Xen dianggap sebagai hypervisor yang lebih stabil pada I/O dan jaringan sedangkan KVM unggul pada daya komputasinya.

Kemunculan *opensource cloud computing* yang berbeda memerlukan keputusan untuk memilih perangkat lunak *cloud computing* yang paling cocok dengan kebutuhan pengguna. Perbedaan komponen dan arsiktektur perangkat lunak *cloud computing* mengakibatkan perbedaan performa pada CPU, memori, R/W *hard disk* dan QoS jaringan yang diberikan. Diperlukan suatu analisa untuk mengetahui perbandingan performa cloud computing yang digunakan. Analisa dilakukan menggunakan metrik skalabilitas dengan metode *overhead* dan *linearity* untuk mengukur performa sesuai dengan aspek yang akan diuji. Diharapkan dengan adanya analisa perbandingan performa tersebut dapat membantu pengguna dalam pemilihan perangkat lunak berbasis IaaS.

II. URAIAN PENELITIAN

A. Cloud Computing

Cloud computing adalah sebuah *client-server* dimana resource seperti *server, storage, network* dan *software* dapat dipandang sebagai layanan yang dapat diakses oleh pengguna secara *remote* dan setiap saat [1]. Pengguna dapat menikmati berbagai layanan yang disediakan oleh *provider cloud computing* tanpa perlu terlalu banyak meminta bantuan teknis atau *support* dari pihak *provider*

Komputasi awan atau *cloud computing* adalah teknologi yang cukup baru di dalam dunia IT (*Information Technology*) di mana komputasi awan ini berfungsi sebagai layanan berupa (jaringan, server, aplikasi, penyimpanan dan lain-lain) yang disediakan kepada para pengguna internet untuk kemudahan dalam beraktifitas [2]. Ada tiga jenis bentuk layanan cloud computing yang diberikan kepada pengguna. Layanan tersebut sebagai berikut:

1) Software as a Service: *Software as a Service* atau SaaS merupakan Layanan yang diberikan dengan menyediakan *software* maupun aplikasi yang dapat diakses pelanggan via internet [3]. Penyedia layanan *cloud computing* berinteraksi dengan pengguna dan pelanggan melalui sebuah *front-end panel*. SaaS menghapus kebutuhan organisasi untuk melakukan instalasi dan menjalankan aplikasi pada komputer atau *data center* mereka. Pengguna dapat menggunakan aplikasi tanpa harus mengerti bagaimana data disimpan, melakukan perawatan dan pengembangan aplikasi tersebut

2) Platform as a Service: *Platform as a Service* atau PaaS merupakan layanan perangkat lunak perantara pada lingkungan *cloud* untuk memfasilitasi berjalannya program aplikasi aplikasi lainnya [4]. PaaS menjadi wadah bagi pengguna untuk menjalankan aplikasinya. Pada PaaS, penyediaan *platform* bagi *developer* yang dikelola oleh pihak *provider* dan diakses melalui internet. PaaS memungkinkan kita untuk membangun aplikasi, *upload* aplikasi, melakukan *testing* aplikasi, ataupun mengatur konfigurasi yang dibutuhkan dalam proses pengembangan aplikasi. Pengguna hanya perlu fokus pada pengembangan aplikasi karena perawatan sistem dikelola oleh pihak *provider*.

3) Infrastructure as a Service: *Infrastructure as a Service* atau IaaS adalah salah satu dari layanan dasar dari *cloud computing* yang menyediakan layanan berupa infrastruktur IT seperti CPU, *storage, memory, network*, dan sebagainya yang bersifat *scalable* dimana sumber daya yang disewakan dapat diubah ubah sesuai kebutuhan pelanggan dengan mudah [5]. IaaS menyediakan sumber daya komputer yang tervirtualisasi dan dapat diakses melalui internet. Sumber daya tersebut dipesan oleh pengguna sesuai dengan kebutuhan dan biaya yang dimiliki oleh pengguna. Pengguna dapat memilih sistem dengan berbagai varian cpu, memori dan penyimpanan. Selain itu pengguna juga dapat melakukan konfigurasi jaringan meskipun bersifat terbatas pada kebijakan *provider*. IaaS memungkinkan pengguna dalam memilih dan mengkonfigurasikan sistem operasi yang disediakan oleh *provider*. Pengguna dapat mengakses *terminal* dan melakukan instalasi *package* untuk mendukung pengembangan aplikasi.

B. Openstack

OpenStack merupakan *open source cloud computing* software untuk membangun i*nfrastruktur cloud* yang dapat diandalkan [5].

1) Keystone: Keystone (*Identity Service*) menyediakan layanan identitas dan akses kebijakan untuk semua komponen dalam keluarga OpenStack [6]. Keystone menyediakan otentikasi dan otorisasi untuk semua komponen OpenStack. Otorisasi akan memverifikasi apakah pengguna yang terotentikasi memiliki akses ke layanannya yang dia minta atau tidak.

2) Glance: Glance (*Image Service*) OpenStack *Imaging Service* adalah salah satu produk dari OpenStack yang digunakan untuk layanan *virtual disk images* [6].

3) Nova: Nova (*Compute Service*) bertindak sebagai *platform management* yang mengelola sumber daya komputasi, jaringan, otorisasi, dan kebutuhan skalabilitas dari OpenStack *cloud* [6]. Nova menangani seluruh kegiatan yang berkaitan dengan siklus hidup *instance.*

4) Neutron: Neutron (*Networking Service*) adalah salah satu komponen Openstack yang menyediakan layanan *cloud Network as a Service* dan memungkinkan pengguna untuk melakukan konfigurasi jaringan pada *cloud* [6]. Neutron memungkinkan pengguna untuk menghubungkan perangkat antarmuka yang dikelola oleh Openstack ke perangkat lainnya. Neutron berinteraksi dengan Nova untuk menyediakan jaringan dan konektivitas dengan *instance.*

5) Horizon: Horizon (*User Interface Service*) merupakan suatu layanan *user interface* dalam infrastruktur Openstack yang memberikan akses visualisasi bagi *user* dalam menciptakan *cloud* [6]. Horizon menyediakan suatu layanan *administrator* dengan

sebuah GUI untuk mengakses *instance*, melakukan *provisioning* dan otomatisasi sumber daya *cloud*.

6) Cinder: Cinder (*Block Storage Service*) adalah komponen penyimpanan blok yang lebih analog dengan gagasan tradisional komputer yang dapat mengakses lokasi tertentu pada *disk drive* serta menyediakan perangkat penyimpanan untuk digunakan dengan *instances* pada OpenStack [6].

C. Apache Cloudstack

Apache Cloudstack merupakan suatu perangkat lunak *opensource* yang digunakan untuk membangun *private, public* atau *hybrid cloud* IaaS dengan mengelola sumber daya komputer [7]. Cloudstack dikembangkan dan didukung oleh komunitas dan beberapa *provider cloud* saat ini. Sebagai salah satu perangkat lunak IAAS, Cloudstack didukung Apache Software Foundation (ASF) sebagai projek utamanya di tahun 2013.

1) Host: *Host* merupakan suatu komputer yang menyediakan daya komputasi untuk menjalankan virtual machine. *Host* pada Cloudstack menggunakan *hypervisor* seperti Xen Server dan KVM untuk mengelola *virtual machine*. *Host* merupakan zona terkecil dalam pengembangan *cloud computing* menggunakan Cloudstack.

2) Cluster: *Cluster* merupakan kumpulan dari beberapa *host*. Satu *cluster* terdiri dari beberapa perangkat yang identik dan menjalankan *hypervisor* yang sama, di satu *subnet* yang sama dan saling berbagi media penyimpanan. Setiap VM yang berjalan di satu *cluster* dapat bermigrasi antar *host* tanpa mengganggu layanan pengguna.

3) Pods: *Pods* merupakan satu kumpulan *cluster* dalam 1 rak. Tiap-tiap host di *pod* yang sama menggunakan *subnet* jaringan yang sama.

4) Zone: *Zone* memiliki kedudukan terbesar kedua dalam perancangan *cloud computing* menggunakan Cloudstack. *Zone* terdiri dari sebuah *pod* atau lebih dan dimungkinkan untuk memiliki penyimpanan redundan. Suatu *zone* dapat dianggap sebagai sebuah *datacenter*.

D. Httperf

Httperf adalah program untuk mengukur kinerja atau performansi dari *web server* yang dibuat oleh David Mosberger dari HP Labs [8]. Httperf menyediakan fitur yang fleksibel dalam pembuatan beban kerja sesuai dengan variable yang diberikan padanya

E. Sysbench

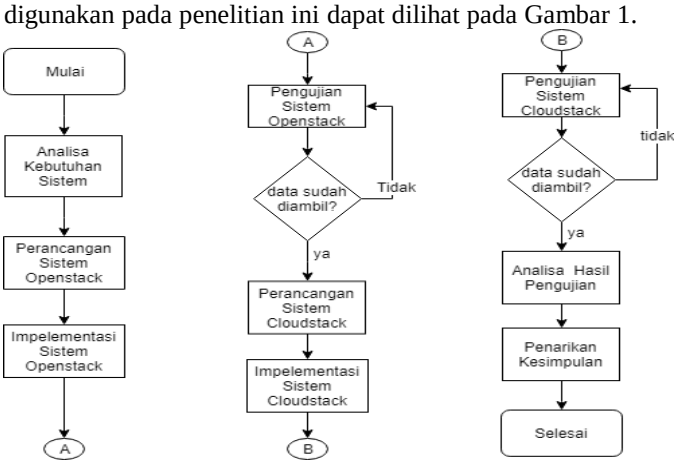
Sysbench adalah aplikasi *benchmark* yang ditujukan untuk mengukur performansi dari suatu *database* yang dijalankan melalui *command prompt* atau *terminal* pada sistem operasi linux [9]. Sysbench dapat melakukan tes pada CPU dan *database*. Pengujian yang dilakukan pada tes CPU berupa mencari bilangan prima dalam jumlah yang telah ditentukan. Sysbench akan menverifikasi suatu bilangan prima dengan membagi bilangan tersebut dengan rentang angka dua hingga akar dari bilangan tersebut. Pengujian yang dilakukan pada tes *database* dilakukan dengan melakukan *insert, read, update* dan *delete* pada *database*.

F. Iperf

Iperf adalah perintah yang digunakan untuk mengukur kinerja *bandwidth* TCP dan UDP untuk menampilkan *throughput, delay jitter*, dan *packet loss* [10]. Iperf dapat digunakan untuk mengukur performa jaringan atau koneksi baik antar *instance* maupun *instance* ke luar melalui terminal.

G. Metodologi Penelitian

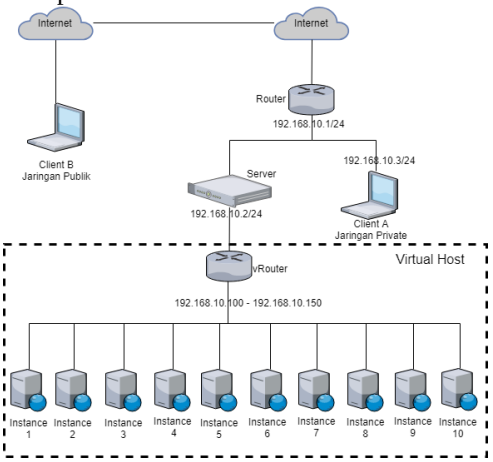
Diagram alir (*flowchart*) penelitian yang digunakan untuk menjelaskan langkah-langkah perencanaan dalam melakukan penelitian. Diagram alir (*flowchart*) yang digunakan pada penelitian ini dapat dilihat pada Gambar 1.



Gambar. 1 Diagram alir penelitian

H. Rancang Arsitektur Jaringan

Arsitektur jaringan yang digunakan dalam penelitian ini dapat dilihat pada Gambar 2.

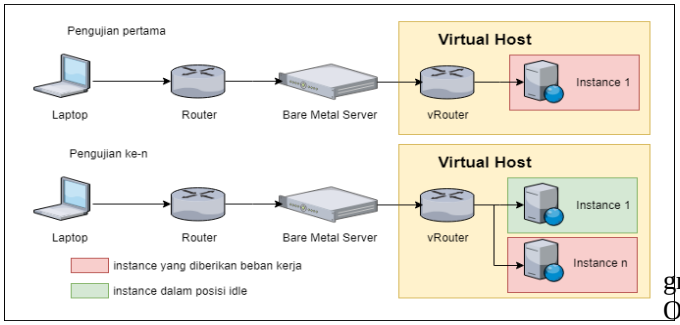


Gambar. 2 Arsitektur jaringan

I. Rancang Pengujian Overhead

Overhead pada virtualisasi *server* adalah seberapa sering dan lamanya waktu yang dibutuhkan oleh *hypervisor* untuk menyelesaikan suatu proses dan menjalankan kembali *virtual machine* [11]. Pengujian

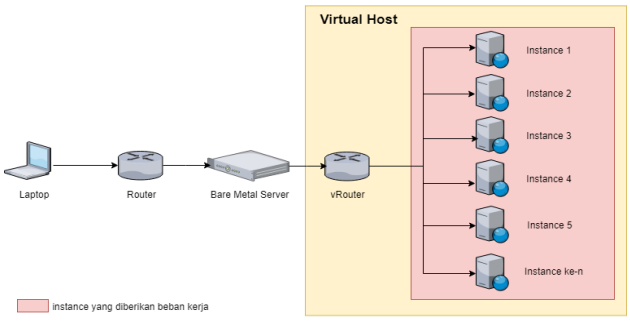
overhead dilakukan dengan cara menambahkan *instance* dan melakukan pengujian pada *instance* tersebut. Setiap *instance* yang di tambahkan akan di uji secara bertahap. Aspek yang diuji pada *instance* tersebut yaitu kecepatan komputasi, *web server*, *database* dan kualitas jaringan. Pengujian dijalankan oleh *client* yang terhubung ke *instance* yang di uji. Simulasi pengujian *overhead* dapat dilihat pada Gambar 3.



Gambar. 3 Skenario pengujian *overhead*

J. Rancang Pengujian Linearity

Linearitas pada virtualisasi *server* dapat diartikan sama dengan *overhead*, akan tetapi dalam pengujian linearitas, *virtual machine* yang ditambahkan diberikan aplikasi yang sama [11]. Pengujian linearitas mengarah pada penilaian kuantitatif tetang waktu yang diperlukan oleh suatu sistem untuk memproses beban kerja sesuai dengan jumlah mesin virtual yang digunakan. Pengujian linearitas dilakukan dengan cara memberikan beban kerja untuk setiap *instance* yang ditambahkan. Aspek yang diuji pada linearitas yaitu waktu eksekusi perintah pada *web server*, komputasi, *database* dan jaringan. Setiap pengujian dilakukan bersamaan untuk setiap mesin virtual yang dijalankan.



Gambar. 4 Skenario pengujian *linearity*

III. HASIL DAN PEMBAHASAN

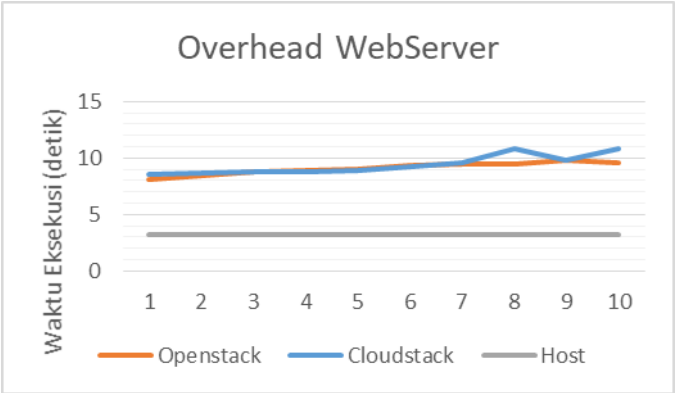
A. Hasil dan Analisa Webserver

Webserver adalah sebuah bentuk *server* yang khusus digunakan untuk menyimpan halaman *website* atau *homepage* [12]. Pengujian *webserver* menggunakan Httperf untuk mendapatkan waktu eksekusi dengan jumlah koneksi sebanyak dua puluh ribu koneksi. Adapun pengujian dilakukan dengan pada *localhost instance* dengan *web* awal pada saat pertama kali dilakukan instalasi Apache *webserver*. Hasil pengujian *overhead webserver* dapat dilihat pada Tabel 1.

TABEL I
HASIL PENGUJIAN OVERHEAD WEBSERVER

Instance ke	Openstack (detik)	Cloudstack (detik)	Host (detik)
1	8.136	8.4986	3.2256
2	8.4672	8.6648	3.2256
3	8.8068	8.7632	3.2256
4	8.8868	8.8048	3.2256
5	9.0304	8.9424	3.2256
6	9.3322	9.2532	3.2256
7	9.4234	9.5312	3.2256
8	9.4598	10.8376	3.2256
9	9.8056	9.767	3.2256
10	9.5902	10.7682	3.2256

Berdasarkan hasil pengujian pada Tabel 1, dibuatlah grafik untuk menampilkan hasil perbandingan performa Openstack dan Cloudstack dalam bentuk visual pada Gambar 5.



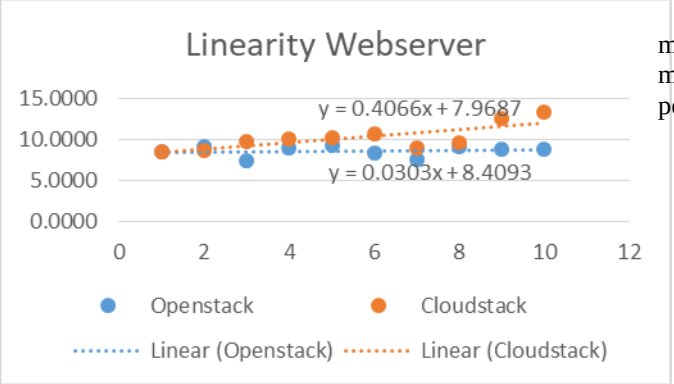
Gambar. 5 Grafik pengujian *overhead webserver*

Pada pengujian *linearity*, seluruh *instance* yang sedang aktif menjalankan httperf secara bersamaan. Hasil dari pengujian *linearity web server* dapat dilihat pada Tabel 2 berikut.

TABEL II
HASIL PENGUJIAN LINEARITY WEBSERVER

Instance ke	Openstack (detik)	Cloudstack (detik)
1	8.4528	8.4986
2	9.0643	8.611
3	7.3717	9.7761
4	8.9705	9.9766
5	9.2102	10.2115
6	8.3576	10.6006
7	7.6143	8.954
8	9.1757	9.5712
9	8.7316	12.5537
10	8.8121	13.2984

Berdasarkan hasil pengujian pada Tabel 2, dibuatlah grafik untuk menampilkan hasil perbandingan performa Openstack dan Cloudstack dalam bentuk visual pada Gambar 6.



Gambar. 6 Grafik pengujian *linearity webserver*

Pengujian httperf berfokus pada waktu eksekusi yang diperlukan oleh *instance* untuk melakukan *connection*, *request* dan *reply*. Httperf menggunakan kemampuan CPU untuk membuat permintaan tersebut. Tujuan dari pengujian ini adalah untuk mendapatkan waktu eksekusi yang lebih rendah antara instance Openstack dan Cloudstack dengan *host* sebagai media pembandingnya. Dengan waktu eksekusi yang lebih rendah maka lebih banyak permintaan http yang dapat dibuat per detik oleh *web server* tersebut.

Berdasarkan pengujian *overhead web server* yang dilakukan pada Openstack dan Cloudstack, didapatkan hasil bahwa waktu eksekusi yang diperlukan *instance* cenderung naik seiring dengan bertambahnya jumlah *instance* dijalankan. Kenaikan tersebut dapaat kita perhatikan pada Gambar 5. Hasil pengujian menunjukan bahwa waktu eksekusi antara Cloudstack dan Openstack relatif sama pada *instance* pertama hingga *instance* ketujuh. Namun pada *instance* delapan hingga sepuluh Openstack memiliki waktu eksekusi yang lebih rendah daripada Cloudstack. Jika dibandingkan dengan *host*, waktu eksekusi pada Openstack dan Cloudstack lebih dari dua kali lipat untuk pengujian tiap tiap *instance*. Waktu eksekusi *web server* pada *host* memang lebih baik, namun jika salah satu *web* pada *host* tersebut menggunakan seluruh sumber daya maka aka berdampak pada kinerja *web* yang lain. Pada *instance* Openstack dan Cloudstack, seluruh sumber daya tersebut terpisah, sehingga apabila salah satu *instance* mengalami kegagalan, maka *instance* lain masih dapat bekerja dnegan baik.

Berdasarkan pengujian *linearity web server* yang dilakukan pada Openstack dan Cloudstack, didapatkan hasil bahwa *instance* pada Cloudstack mengalami penambahan waktu eksekusi sebanding dengan jumlah instance yang digunakan sedangkan *instance* pada Openstack cenderung relatif sama untuk rata rata waktu eksekusinya. Kenaikan waktu eksekusi tersebut dapat dilihat pada grafik pada Gambar 6. Pada pengujian *linearity web server* dapat diambil kesimpulan bahwa Openstack lebih unggul dari Cloudstack karena waktu eksekusi yang lebih relatif lebih rendah.

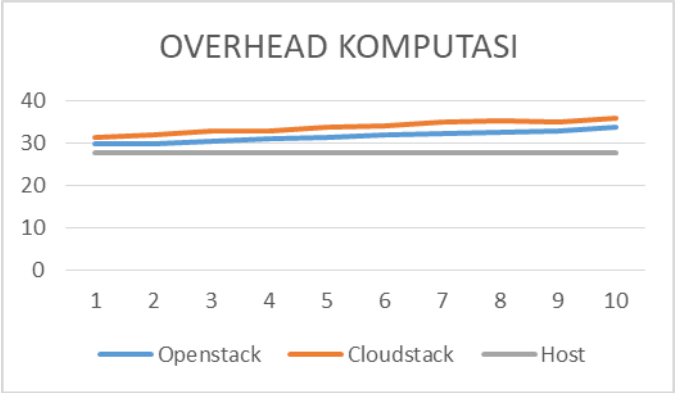
B. Hasil dan Analisa Komputasi

Pengujian Komputasi menggunakan Sysbench untuk menghitung waktu yang diperlukan oleh *instance* untuk menemukan dua puluh ribu bilangan prima. Hasil pengujian komputasi dapat dilihat pada Tabel 3 berikut.

TABEL III
HASIL PENGUJIAN OVERHEAD WEBSERVER

Instance ke	Openstack (detik)	Cloudstack (detik)	Host (detik)
1	29.81958	31.3641	27.58122
2	29.83824	32.02746	27.58122
3	30.47266	32.82794	27.58122
4	30.87008	32.92118	27.58122
5	31.19182	33.6778	27.58122
6	31.79386	34.05862	27.58122
7	32.22864	35.01746	27.58122
8	32.35438	35.17498	27.58122
9	32.93492	35.06258	27.58122
10	33.6388	35.71812	27.58122

Berdasarkan hasil pengujian pada Tabel III, dibuatlah grafik untuk menampilkan hasil perbandingan performa Openstack dan Cloudstack pada Gambar 7.



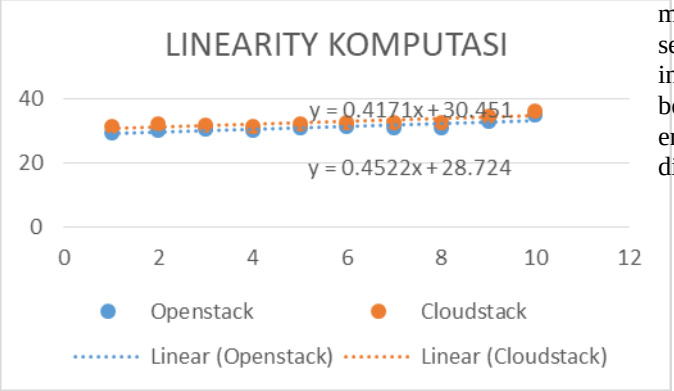
Gambar. 7 Grafik pengujian *overhead* komputasi

Hasil pengujian *linearity* komputasi dapat dilihat pada Tabel IV. Pada tabel tersebut, seluruh instance menjalankan Sysbench secara bersamaan untuk mencari dua puluh ribu bilangan prima.

TABEL IV
HASIL PENGUJIAN LINEARITY WEBSERVER

Instance ke	Openstack (detik)	Cloudstack (detik)
1	29.4861	31.3641
2	30.1301	31.9734
3	30.3317	31.8808
4	30.2267	31.4976
5	30.8826	32.1455
6	31.2808	32.6480
7	30.9684	32.6050
8	30.8422	32.5666
9	32.9786	34.6349
10	34.9860	36.1353

Berdasarkan hasil pengujian pada Tabel IV, dibuatlah grafik untuk menampilkan hasil perbandingan performa Openstack dan Cloudstack pada Gambar 8.



Gambar 8 Grafik pengujian *linearity* komputasi

Pengujian komputasi bertujuan untuk mengetahui waktu eksekusi yang dibutuhkan untuk menyelesaikan suatu perhitungan antara *instance* Openstack dan Cloudstack. Beberapa aplikasi berbasis web dan dekstop memerlukan CPU untuk menyelesaikan algoritma pemrogramannya. Selain itu, *service* lain yang bertugas untuk menjalankan sistem operasi juga bergantung pada kinerja CPU tersebut. Dengan kinerja CPU yang lebih baik maka pemrosesan layanan lain seperti ftp, *web service* dan perhitungan komplek menjadi lebih baik.

Berdasarkan hasil pengujian *overhead* komputasi yang dilakukan pada Openstack dan Cloudstack, waktu eksekusi mengalami kenaikan yang stabil sesuai dengan penambahan jumlah *instance* yang dijalankan. Pada pengujian ini terlihat bahwa *instance* pada Openstack relatif lebih baik daripada *instance* pada Cloudstack. Hal ini dikarenakan waktu eksekusi Openstack yang lebih rendah dibandingkan dengan Cloudstack. Waktu eksekusi tersebut dapat dilihat pada Gambar 7 yang menunjukan grafik pengujian. Pada pengujian ini, waktu eksekusi antara *instance* Openstack dan Cloudstack relatif mendekati waktu eksekusi *host*. Hal ini dikarenakan pengujian CPU pada Sysbench hanya menjalankan satu *thread* pengujian sehingga CPU yang bekerja antara *host*, Openstack dan Cloudstack sama sama berjumlah 1 *core* CPU.

Berdasarkan pengujian *linearity* komputasi yang dilakukan pada Openstack dan Cloudstack, didapatkan hasil bahwa *instance* pada Cloudstack mengalami penambahan waktu eksekusi sebanding dengan jumlah *instance* yang digunakan sedangkan *instance* pada Openstack cenderung relatif sama untuk rata rata waktu eksekusinya. Kenaikan waktu eksekusi tersebut dapat dilihat pada grafik pada Gambar 8. Pada pengujian *linearity* komputasi dapat diambil kesimpulan bahwa Openstack lebih unggul dari Cloudstack karena waktu eksekusi yang lebih relatif lebih rendah.

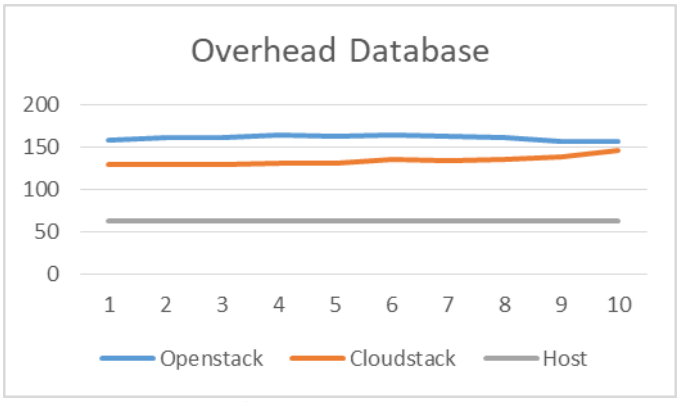
C. Hasil dan Analisa Database

Pengujian *database* menggunakan Sysbench untuk menghitung waktu yang diperlukan oleh *instance* untuk melakukan pengujian OLTP dengan jumlah data dibaca sebanyak dua puluh ribu data. OLTP adalah kelas sistem informasi yang memfasilitasi dan mengelola aplikasi berorientasi transaksi, biasanya untuk pemrosesan data entry dan *retrieval* [13]. Hasil pengujian komputasi dapat dilihat pada Tabel V berikut.

TABEL V
HASIL PENGUJIAN OVERHEAD DATABASE

Instance ke	Openstack (detik)	Cloudstack (detik)	Host (detik)
1	158.29834	129.28872	62.22126
2	161.04316	129.61084	62.22126
3	161.42964	129.69062	62.22126
4	164.13384	131.01852	62.22126
5	163.05474	130.70722	62.22126
6	164.30342	134.70512	62.22126
7	162.08348	134.29838	62.22126
8	160.37508	135.6868	62.22126
9	156.2803	138.5282	62.22126
10	156.88988	145.6028	62.22126

Berdasarkan hasil pengujian pada Tabel V, dibuatlah grafik untuk menampilkan hasil perbandingan performa Openstack dan Cloudstack dalam bentuk visual pada Gambar 9.



Gambar. 9 Grafik pengujian *overhead database*

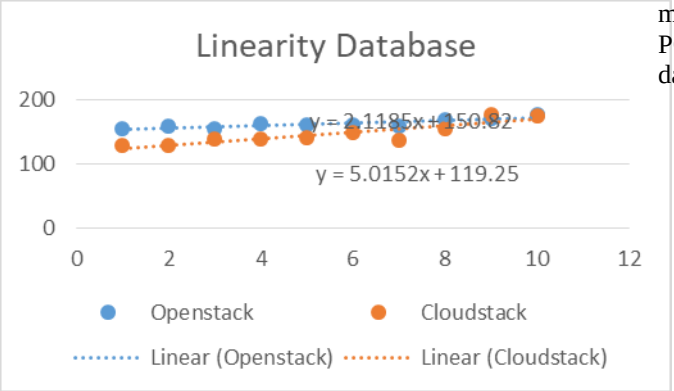
Pengujian *linearity database* menggunakan Sysbench untuk melakukan pengujian OLTP oleh *instance* aktif secara bersamaan. Berikut hasil pengujian *lienarity database* dapat dilihat pada Tabel 6.

TABEL VI
HASIL PENGUJIAN LINEARITY DATABASE

Instance ke	Openstack (detik)	Cloudstack (detik)
1	154.8445	129.2887
2	158.7755	129.0991
3	153.7133	138.0587
4	161.7316	139.3046
5	160.3493	140.8554
6	159.5351	148.5788
7	159.4898	136.8081
8	168.6968	154.6885
9	170.7129	176.5241
10	176.9130	175.0828

Berdasarkan hasil pengujian pada Tabel VI, maka dibuatlah grafik untuk menampilkan hasil visual dari

pengujian yang telah dilakukan. Adapun grafik tersebut dapat dilihat pada Gambar 10.



Gambar. 10 Grafik pengujian *linearity database*

Pengujian *database* dilakukan untuk mendapatkan waktu eksekusi yang dibutuhkan untuk melakukan beberapa tes OLTP pada *database*. Beberapa aplikasi seperti *online store*, inventori barang, aplikasi penjualan serta sistem informasi mengandalkan *cache* memori untuk menyelesaikan transaksi pada *database*. Dengan kecepatan transaksi *database* yang lebih baik maka diharapkan dapat menambahkan, menghapus, mengubah dan menampilkan data dengan lebih baik.

Berdasarkan hasil pengujian *overhead database* yang dilakukan pada Openstack dan Cloudstack, waktu eksekusi sama sama mengalami kenaikan yang stabil sesuai dengan penambahan jumlah *instance* yang dijalankan. Pada pengujian ini terlihat bahwa waktu eksekusi oltp *database* pada *instance* Cloudstack lebih cepat daripada *instance* Openstack. Hal tersebut dapat dilihat pada Gambar 9.

Berdasarkan hasil pengujian *linearity database* yang dilakukan pada Openstack dan Cloudstack, didapatkan hasil bahwa waktu ekselusi mengalami kenaikan sesuai dengan penambahan jumlah *instance* dan pengujian yang dijalankan. Pada pengujian ini, waktu eksekusi cenderung stabil pada *instance* Openstack. Sedangkan pada instance Cloudstack, kenaikan waktu eksekusi stabil hingga *instance* kelima. *Instance* keenam dan selanjutnya mengalami kenaikan yang cenderung kurang stabil. Pada pengujian *linearity database* ini, *instance* pada Cloudstack lebih unggul daripada instance pada Openstack karena waktu eksekusi OLTP *database* yang lebih rendah. Hal ini dapat dilihat pada Gambar 10.

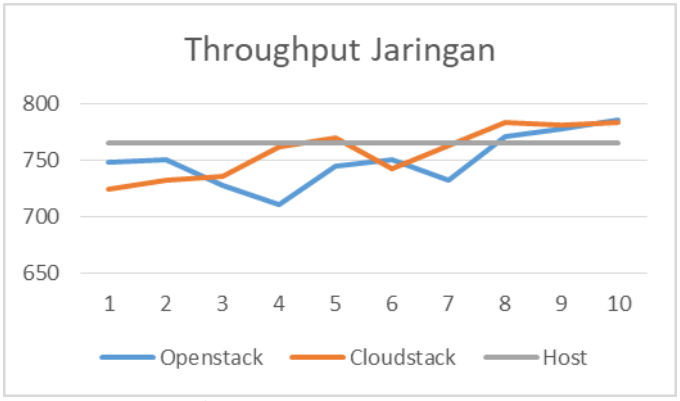
D. Hasil dan Analisa Jaringan

Pengujian jaringan menggunakan Iperf untuk menghitung *throughput*, *jitter* dan *packet loss*. *Throughput* merupakan salah satu parameter dalam *Quality of Service* dimana fungsinya untuk melihat perbandingan jumlah data yang di proses kirim dalam waktu pengamatan tertentu [14]. *Jitter* adalah perbedaan selang waktu kedatangan antar paket di terminal tujuan. *Jitter* dapat disebabkan oleh terjadinya kongesti, kurangnya kapasitas jaringan, variasi ukuran paket, serta ketidakakuratan paket [15]. Pengujian jaringan menggunakan Iperf sebagai salah satu alat pengujian. *Instance* Openstack dan Cloudstack menjalankan Iperf *server* sedangkan *client* menggunakan PC/Laptop (lihat Gambar 2). Hasil pengujian *throughput* dapat dilihat pada Tabel 7 berikut.

TABEL VII
HASIL PENGUJIAN OVERHEAD THROUGHPUT

Insance ke	Openstck (Mbps)	Cloudsack (Mbps)	Host (Mbps)
1	748	723.6	764.6
2	750.2	732	764.6
3	728	735	764.6
4	710.4	762.2	764.6
5	744.6	769.4	764.6
6	750.8	741.8	764.6
7	731.8	762.4	764.6
8	771.2	783.2	764.6
9	777.8	781.2	764.6
10	785.4	782.8	764.6

Berdasarkan hasil pengujian pada Tabel VII, dibuatlah grafik untuk menampilkan hasil perbandingan performa Openstack dan Cloudstack pada Gambar 11.



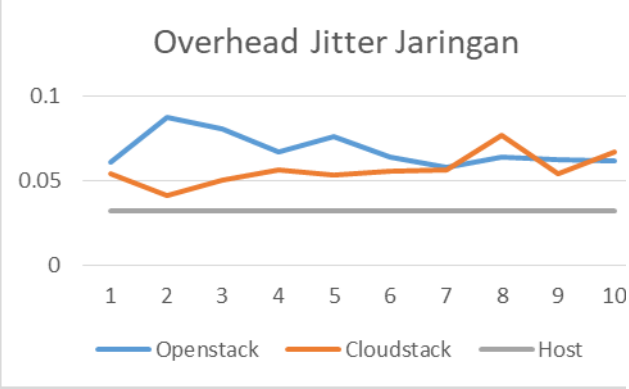
Gambar. 11 Grafik pengujian *overhead throughput* jaringan

Hasil Pengujian *jitter* dapat dilihat pada Tabel 8 berikut ini.

TABEL VIII
HASIL PENGUJIAN OVERHEAD JITTER

Instance ke	Openstack (ms)	Cloudstack (ms)	Host (ms)
1	0.0608	0.0542	0.0322
2	0.087	0.0414	0.0322
3	0.0808	0.0505	0.0322
4	0.0666	0.056	0.0322
5	0.076	0.0532	0.0322
6	0.0638	0.0558	0.0322
7	0.058	0.0566	0.0322
8	0.0638	0.0766	0.0322
9	0.0622	0.0542	0.0322
10	0.06175	0.0666	0.0322

Berdasarkan hasil pengujian pada Tabel VIII, dibuatlah grafik untuk menampilkan hasil perbandingan performa Openstack dan Cloudstack dalam bentuk visual pada Gambar 12 berikut.



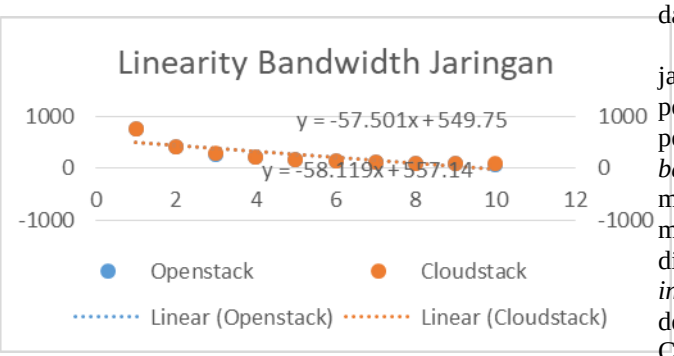
Gambar. 12 Grafik pengujian *overhead jitter* jaringan

Pada pengujian *linearity throughput* jaringan, seluruh instance aktif bertugas sebagai Iperf *server* dan PC menjadi Iperf *client*. Hasil pengujian *linearity throughput* dapat dilihat pada Tabel 9 berikut.

TABEL IX
HASIL PENGUJIAN LINEARITY THROUGHPUT

Instance ke	Openstack (Mbps)	Cloudstack (Mbps)
1	756.60	766.00
2	400.80	409.90
3	276.80	279.00
4	206.95	209.60
5	164.24	167.40
6	136.90	138.95
7	117.40	118.87
8	101.07	104.11
9	89.47	92.95
10	80.97	82.88

Berikut grafik untuk menampilkan hasil visual dari pengujian *linearity throughput* pada Gambar 13.



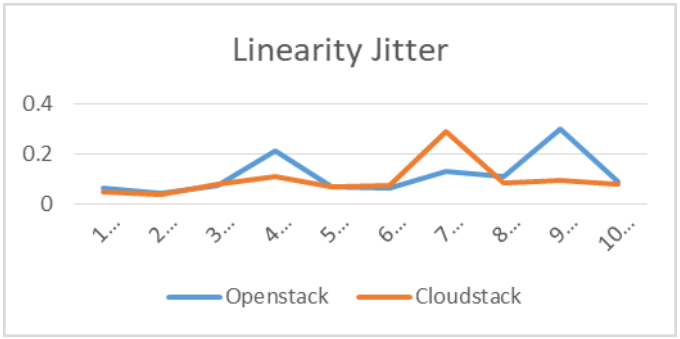
Gambar. 13 Grafik pengujian *linearity throughput* jaringan

Hasil pengujian dari *linearity jitter* dapat dilihat pada Tabel 10 berikut ini.

TABEL X
HASIL PENGUJIAN LINEARITY THROUGHPUT

Instance ke	**Openstack (ms)**	**Cloudstack (ms)**
1	756.60	766.00
2	400.80	409.90
3	276.80	279.00
4	206.95	209.60
5	164.24	167.40
6	136.90	138.95
7	117.40	118.87
8	101.07	104.11
9	89.47	92.95
10	80.97	82.88

Tampilan visual dari hasil pengujian *linearity jitter* dapat dilihat pada Gambar 14 berikut ini.



Gambar. 14 Grafik pengujian *linearity throughput* jaringan

Pengujian pada jaringan *private* bertujuan untuk mendapatkan kecepatan *throughput*, variasi *delay* atau *jitter* dan persentase *packet loss* dalam suatu lingkungan internal kerja yang berada dalam satu gedung atau satu ruangan. Beberapa layanan seperti ftp dan rdp memerlukan *throughput* yang tinggi serta *jitter* dan *packet loss* yang rendah. *Throughput* merupakan kecepatan aktual dari pengiriman data, *jitter* bertanggung jawab pada kondisi kestabilan jaringan sedangkan *packet loss* mengarah pada data yang gagal sampai ke tempat tujuan. Iperf menggunakan protokol TCP untuk menghitung *throughput* serta protokol UDP untuk menghitung *jitter* dan *packet loss*.

Berdasarkan hasil pengujian *overhead throughput* jaringan pada Openstack dan Cloudstack, kenaikan dan penurunan kecepatan cenderung tidak stabil. Pada pengujian yang telah dilakukan, tidak diberikan limitasi *bandwidth*. Hal ini dimaksudkan agar *throughput* jaringan melalui protokol TCP dapat mencapai nilai kecepatan maksimal sesuai dengan batasan kemampuan mesin yang digunakan. Pada Openstack kecepatan throughput *instance* berkisar antara 710,4 Mbps hingga 785,4 Mbps dengan kecepatan rata rata 749.82 Mbps . Pada Cloudstack kecepatan throughput *instance* berkisar antara antara 723,6 Mbps hingga 783,2 Mbps dengan kecepatan rata rata 757.36 Mbps. Berdasarkan grafik pada Gambar 12, hasil pengujian *jitter* pada Cloudstack relatif lebih rendah dibandingkan dengan Openstack. Pengujian *overhead* pada Openstack dan Cloudstack pada jaringan *private* tidak mengalami *packet loss* untuk tiap tiap hasil pengujian.

Berdasarkan hasil pengujian *linearity throughput* jaringan yang dilakukan pada Openstack dan Cloudstack, didapatkan bawah kecepatan transfer mengalami

penurunan sesuai dengan penambahan jumlah instance dan beban pengujian yang dijalankan. Pada pengujian ini, throughput pada Openstack dan Cloudstack relatif sama. Hal ini dapat dilihat pada Gambar 13. Pada Pengujian jaringan menggunakan protokol UDP pada Openstack dan Cloudstack, didapatkan hasil bahwa jitter jaringan mengalami kenaikan dan penurunan yang kurang stabil. Hal ini dapat dilihat pada Gambar 14. Sedangkan untuk packet loss, persentasi paket yang hilang tetap sebesar 0%.

IV. KESIMPULAN

Berdasarkan hasil pengujian dan analisa terhadap performa Openstack dan Cloudstack dalam aspek *webserver*, komputasi, *database* dan jaringan, maka dapat diambil kesimpulan sebagai berikut.

1. *Instance* pada Openstack unggul dalam aspek pengujian komputasi dan *weserver*, sehinga pada penggunaan yang berfokus pada pengembangan aplikasi berbasis *web* ataupun yang memerlukan daya komputasi yang lebih tinggi disarankan untuk menggunakan Openstack.
2. *Instance* pada Cloudstack unggul dalam aspek pengujian *database* dan jaringan. Cloudstack unggul pada *throughput* jaringan *private* dan publik. Sedangkan untuk *jitter* dan *packet loss*, pengujian pada *instance* Cloudstack relatif lebih rendah dibandingkan Openstack. Pada pengujian *database*, waktu eksekusi pada Cloudstack lebih baik daripada Openstack.

REFERENSI

[1] I. Sofana, Teori dan Praktik Cloud Computing (OpenNebula, VMWare dan Amazon AWS), Bandung: Informatika Bandung, 2012.

[2] M. Ibrahim and Kusnawi, "Analisis dan Implementasi Owncloud Sebagai Media Penyimpanan Pada Yayasan Salman Salman Al-Farisi Yogyakarta," *Jurnal Ilmiah DASI*, vol. 14, no. 04 Desember 2013, pp. 32-37, 2013.

[3] E. Retnaningsih and B. E. Purnama, "Pelacakan Lokasi Hosting Web Perguruan Tinggi Studi Kasus: Perguruan Tinggi Kopertis 6 Jawa Tengah," *Seruni FTI UNSA* , vol. I, pp. J1-J8, 2012.

[4] Y. Cancer and Z. Alim, "Platform as a Service (PaaS) sebagai layanan sistem operasi cloud computing," *Jurnal TIMES*, vol. V, no. 1, pp. 32-35, 2016.

[5] T. W. Ananda, Rusmani and A. Mulyana, "Desain dan realisasi sistem grid computing pada infrastructure as a service menggunakan cloud platform Openstack," *e-Proceeding of Engineering*, vol. 3, no. 1, pp. 743-748, 2016.

[6] M. Fauzan, A. Fiade and F. E. M. A, "Analisis dan perancangan infrastruktur private cloud dengan Openstack," *Jurnal Pseudocode*, vol. IV, no. 2, pp. 180-189, 2017.

[7] R. Kumar, K. Jain, H. Maharwal, N. Jain and A. Dadhich, "Apache CloudStack: Open Source Infrastructure as a Service Cloud," *International Journal of Advancement in Engineering Technology, Management and Applied Science*, vol. 1, no. 2, pp. 111-116, 2014.

[8] D. S. Kristian, A. F. Rochim and E. D. Widianto, "Pengembangan Sistem Replikasi dan Redundansi Untuk Meningkatkan Kehandalan Basisdata Mysql," *Jurnal Teknologi dan Sistem Komputer* , vol. 3, no. 4, pp. 523-529, 2015.

[9] T. A. Gani, A. Arafat and Melinda, "Analisis Kinerja MySQL Cluster Menggunakan Metode Load Balancing," *Jurnal Rekayasa Elektrika*, vol. 11, no. 4, pp. 129-134, 2015.

[10] Wahyudi and Supini, "Monitoring dan Analisa Traffik Jaringan Dengan Menggunakan Mikrotik RouterOS," *Jurnal Teknologi Informasi*, vol. 5, no. 2, pp. 269-276, 2017.

[11] A. Arfriandi, "Perancangan, Implementasi, dan Analisis Kinerja Virtualisasi Server Menggunakan Proxmox, Vmware ESX, dan Openstack," *Jurnal Teknologi*, vol. 5, no. 2, pp. 182-191, 2012.

[12] P. A. Nugraha, M. A. Irwansyah and H. Priyanto, "Rancang Bangun Web Server Berabsis Linux Dengan Metode Load Balancing (Studi kasus : Laboratorium Teknik Informatika)," *Jurnal Sistem dan Teknologi Informasi*, vol. 3, no. 1, pp. 1-5, 2016.

[13] Y. Firdaus, S. B. Premapasha and S. Yulias, "Analisis dan Perancangan Sistem Online Transaction Processing (OLTP) Menggunakan SCRUM (Studi Kasus Rumah Sakit Puti Bungsu)," *e-Proceeding of Engineering*, vol. 4, no. 2, pp. 3130-3137, 2017.

[14] B. Arifwidodo, "Analisis Quality of Service pemanfaatan Ethernet Over IP(EoIP) Tunnel di MikrotikRouterOS dengan Routing Protocol OSPF," *Journal of Informatics, Information System, Software Engineering and Applications* , vol. 1, no. 1, pp. 1-8, 2018.

[15] Fitri, M. Yamin and L. B. Aksara, "Perbandingan Metode Differentiated Service dengan Metode Integrated Service Untuk Analisa Quality of Service (QoS Video Streaming) Pada Jaringan Multi Protocol Label Switching (MPLS)," *semanTIK*, vol. 3, no. 1, pp. 135-142, 2017.