Analisis Perbandingan Kinerja Protokol NFS dan iSCSI pada Teknologi OpenStack Cinder

Reynaldo Chandra, Henry Novianus Palit, Agustinus Noertjahyana
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jln. Siwalankerto 121-131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) - 8417658
reynaldo.chandra0@gmail.com, hnpalit@petra.ac.id, agust@petra.ac.id

ABSTRAK

Istilah *cloud computing* sudah biasa digunakan pada hari-hari ini. Penggunaan *cloud computing* kini sudah menyebar di seluruh dunia dan digunakan oleh banyak perusahaan. Karena penggunaannya yang semakin besar, muncullah kebutuhan akan *cloud storage*. *Cloud storage* tidak hanya memanfaatkan sumber daya dengan lebih efektif dan efisien, namun juga dapat mempermudah dalam penggunaan *virtual machine*. Ada beberapa protokol alternatif yang dapat diimplementasikan untuk *cloud storage*, dimana masing-masing protokol tersebut memiliki kelebihannya sendiri, sehingga skripsi ini bertujuan untuk menentukan protokol alternatif yang paling baik untuk diimplementasikan pada OpenStack cinder dalam pemanfaatan *cloud storage*.

Implementasi *cloud computing* dilakukan di laboratorium komputer Universitas Kristen Petra menggunakan *private cloud*. Hal ini dikarenakan spesifikasi komputer di laboratorium tergolong bagus, namun tidak selalu digunakan untuk kelas sehingga dapat dimanfaatkan dengan lebih maksimal. Untuk penyedia *storage* digunakan NAS Synology DS416J. Implementasi menggunakan *framework cloud* OpenStack, yang menyediakan *Infrastructure as a Service* (IaaS) dalam penggunaannya. OpenStack cinder adalah salah satu project OpenStack yang menyediakan *cloud storage* dengan *persistent storage*. Cinder sendiri dapat diimplementasi menggunakan protokol *fibre channel*, NFS, dan iSCSI, namun protokol yang menjadi fokus pelitian ini adalah NFS dan iSCSI.

Setelah implementasi, pengujian dilakukan dengan cara mengukur performa protokol NFS dan iSCSI saat diimplementasikan pada OpenStack cinder menggunakan IOzone. Berdasarkan hasil pengujian tersebut dapat dilihat bahwa NFS memiliki keunggulan saat melakukan *write* pada *file* dengan *record size* yang kecil, sedangkan iSCSI unggul saat melakukan *write* pada *file* dengan *record size* yang besar, namun pada aktivitas *read* tidak terlihat perbedaan yang mencolok antara protokol NFS dan iSCSI. Dengan memperhitungkan hasil pengujian dan analisa sistem yang telah dibuat, pada skripsi ini diambil kesimpulan bahwa protokol iSCSI lebih baik untuk diimplementasikan pada OpenStack cinder dibandingkan NFS.

Kata Kunci: *Cloud computing*, OpenStack, OpenStack cinder, *persistent storage*, *Infrastructure as a Service*, NFS, iSCSI

ABSTRACT

Cloud computing term is commonly used nowadays. Cloud computing usage now already spread all over the world dan used by many companies. Because of high cloud computing usage, a need for a cloud storage now emerge. Cloud storage not only use resource more effective and efficient, but also make the usage of a virtual machine much more easier. There are several alternative protocols that can be implemented for cloud storage, where each of these protocols has its own advantages, so this thesis aims to determine the most suitable alternative protocol to implement on OpenStack cinder in utilizing cloud storage.

Cloud computing implementation is done in computer laboratorium at Petra Charistian University using private cloud. This is done because of the computer specification in the laboratorium is good, but not always used for class so it can be utilized maximally. For storage provider a NAS Synology DS416J was used. Implementation using OpenStack cloud framework, that provides Infrastructure as a Service (IaaS). OpenStack cinder one of OpenStack project that provides cloud storage with persistent storage. Cinder itself can be implemented using fibre channel, NFS, and iSCSI protocols, but the protocols that are the focus of this study are NFS and iSCSI.

After implementation, testing was carried out by measuring the performance of the NFS and iSCSI protocols when implemented on OpenStack cinder using IOzone. Based on the results of these tests it can be seen that NFS has the advantage when writing files with a small record size, whereas iSCSI has the advantage when writing files with a large record size, but in read activity there is no noticeable difference between the NFS and iSCSI protocols. By taking into account the results of testing and analysis of the systems that have been made, in this thesis the conclusion was the iSCSI protocol is better to be implemented on OpenStack cinder than NFS.

Keywords: *Cloud computing, OpenStack, OpenStack cinder, persistent storage, Infrastructure as a Service, NFS, iSCSI*

1. PENDAHULUAN

Pada jaman modern ini, teknologi berkembang dengan sangat cepat. Banyak orang ataupun perusahaan yang tidak berusaha untuk mengikuti perkembangan teknologi ini akan ketinggalan jaman. Hal tersebut dikarenakan teknologi saat ini menawarkan banyak kemudahan dan efisiensi dalam banyak hal, baik untuk pekerjaan maupun kehidupan sehari-hari saja. Salah satu teknologi yang ditawarkan saat ini adalah penggunaan cloud computing.

Cloud computing adalah metode untuk menyediakan satu set resource komputasi yang terdiri atas aplikasi, komputasi, storage, networking, development, dan deployment platform serta business process. Cloud computing yang mengubah aset komputasi tradisional menjadi resource yang dapat digunakan bersama untuk

melakukan suatu task sehingga proses komputasi bisa lebih cepat [3].

Cloud sendiri memiliki model yang berbeda-beda, sesuai dengan kebutuhan. Ada 2 model utama dalam implementasi cloud, yaitu public dan private [3]. Public cloud biasa digunakan oleh perusahaan atau organisasi yang ingin menghemat waktu karena tidak perlu melakukan melakukan management, maintaince, dan update data center yang digunakan untuk cloud, sedangkan private cloud dipilih karena keamanannya dan kemudahan dalam memindahkan dan menganalisa data.

Untuk mengatur penyimpanan data ini digunakanlah OpenStack cinder. OpenStack cinder ini adalah project block storage milik OpenStack yang bisa didapatkan dengan mudah karena bersifat open-source dan gratis. Kegunaannya untuk mengatur penyimpanan data dengan menggunakan volume yang dibagikan ke VM instance. Kelebihan cinder adalah fungsi persistent block storage dan volume snapshot yang disediakannya sehingga apabila VM (virtual machine) dihapus maka data yang telah disimpan tidak akan hilang [1].

Ada beberapa protokol yang bisa digunakan untuk OpenStack cinder ini, diantaranya Internet Small Computer Systems Interface (iSCSI), Network File System (NFS), dan Fibre Channel. Protokol yang menjadi fokus penelitian ini adalah iSCSI dan NFS sehingga diajukan judul skripsi ini yaitu Analisis Perbandingan Kinerja Protokol NFS dan iSCSI pada Teknologi OpenStack Cinder untuk menemukan protokol yang paling cocok untuk diimplementasikan pada OpenStack cinder.

2. LANDASAN TEORI

2.1 OpenStack

OpenStack adalah platform *cloud open-source* dengan komunitas yang sangat aktif. OpenStack sangat disupport oleh penyedia *cloud* komersial, seperti Rackspace, Canonical, Dreamhost, dan HP Cloud, dan sering digunakan untuk research [10].

OpenStack adalah kombinasi dari *open source tool* (yang dikenal sebagai *project*) yang menggunakan *pool* dari *virtual resource* untuk membuat dan me*manage private* dan *public cloud*. Enam dari *project* ini menghandle inti dari *cloud-computing service* yaitu komputasi, *networking*, *storage*, *identity*, dan *image service*, dengan banyak *project optional* yang bisa digunakan secara bersamaan dan membentuk cloud yang unik [12].

OpenStack menyediakan *software module* yang dibutuhkan untuk membangun *private cloud platform* terautomasi. Walaupun dulu OpenStack fokus pada penyediaan *Infrastructure as a Service* (IaaS) seperti *Amazon Web Services* (AWS), beberapa *project* baru telah diperkenalkan, yang akan mulai menyediakan kemampuan untuk lebih diasosiasikan dengan *Platform as a Service* (PaaS) [15].

OpenStack project keystone adalah sebuah OpenStack *service* yang menyediakan API *client authentication*, *service discovery*, dan *distributed multi-tenant authorization* dengan mengimplementasi OpenStack *Identity* API [9]. Keystone diperlukan sebagai sarana autentikasi semua *service* OpenStack.

OpenStack project glance menyiapkan sebuah *service* dimana *user* bisa menguploud dan menemukan *image* yang perlu digunakan untuk *virtual machine*. *Service* image glance termasuk *discovering*, me*register*, dan me*retrieve* image *virtual machine*. Glance memiliki RESTful API meng*query metadata* image *virtual machine* dan juga pengambilan image yang asli [9].

OpenStack project nova adalah OpenStack *project* yang menyediakan *compute instance*. Nova men*support* pembuatan *virtual machines*, *baremetal server* (apabila menggunakan OpenStack project ironic), dan mempunyai *support* terbatas untuk *system containers*. Nova berjalan sebagai sebuah set *daemon* diatas Linux server yang ada untuk menyediakan *service* tersebut [9].

OpenStack project neutron adalah sebuah OpenStack *project* yang menyediakan "*network connectivity as a service*" antara device interface (contohnya vNICs) yang di*manage* oleh OpenStack services yang lainnya (contohnya nova). Dilakukan dengan cara mengimplementasikan Neutron API [9]. Neutron diperlukan untuk menyiapkan network yang akan dipakai oleh *virtual machine*.

OpenStack project horizon adalah implementasi untuk OpenStack *dashboard*, yang menyediakan sebuah *user interface* berbasiskan *web* untuk *service-service* OpenStack termasuk nova, swift, keystone, dll [9].

Cinder adalah nama *project* untuk *OpenStack Block Storage*. Cinder menyediakan *persistent block storage* untuk *guest virtual machine* (VM). *Block storage* sering dibutuhkan untuk *expandable file system*, *maximum performance*, dan integrasi dengan *enterprise storage service* serta aplikasi yang membutuhkan akses ke *raw block-level storage*.

Sistem dapat *connect* ke *device*, me*manage creation*, *attachment* ke, dan *detachment* dari server. *Application programming interface* (API) juga memfasilitasi *snapshot management* yang bisa mem*backup volume* dari *block storage* [13].

Persistent storage tradisional disediakan pada OpenStack *workload* melalui Cinder *block storage component*. Life cycle dari *volume* Cinder di*maintain* secara independen oleh *compute instance*, dan *volume* bisa di*attach* atau *detach* ke satu atau lebih *compute instance* untuk menyediakan *backing store* untuk *filesystem-based storage* [15].

Beberapa fitur yang disediakan oleh Cinder adalah *volume management*, *snapshot management*, *attach* atau *detach volume* dari *instance*, *cloning volume*, membuat *volume* dari *snapshot*, dan meng*copy image* ke *volume* dan sebaliknya [5].

Program-program lain yang diperlukan agar semua service OpenStack dapat berjalan dengan baik adalah NTP, MariaDB, RabbitMQ, dan memcached.

NTP adalah singkatan dari *Network Time Protocol*, dan merupakan sebuah protokol internet yang digunakan untuk mensinkronisasi *clock* dari komputer ke sebuah referensi waktu. NTP adalah sebuah protokol standar internet yang awalnya di*develop* oleh Professor David L. Mills dari University of Delaware [8]. NTP diperlukan sebagai sarana sinkronisasi waktu untuk komputer *server* dan *client*.

MariaDB Server adalah salah satu *database* server yang popular di dunia. MariaDB dibuat oleh *developer* MySQL dan dijamin untuk tetap *open source*. Pengguna yang cukup terkenal adalah Wikipedia, WordPress.com and Google [6]. MariaDB diperlukan untuk menyediakan *database* bagi service-service OpenStack.

RabbitMQ adalah sebuah broker *messaging* atau sebuah perantara untuk *messaging*. RabbitMQ memberikan aplikasi sebuah *platform* untuk mengirim dan menerima *message*, dan memberikan tempat untuk *message* sampai *message* berhasil diterima [11]. RabbitMQ digunakan oleh *service-service* OpenStack untuk saling berkomunikasi.

Memcached bersifat *open source*, memiliki performa yang tinggi, *distributed memory object caching system*, ditujukan untuk mempercepat *dynamic web application* dengan cara mengurangi *load database*. Memcached adalah sebuah *in-memory key-value* yang menyimpan data (*string*, *object*) yang merupakan hasil dari *database call*, API *call*, atau *page rendering* [7].

2.2 *Network File System*

Network file system adalah abstraksi *network* melalui sebuah *file system* yang membuat *remote client* dapat mengaksesnya melalui sebuah *network* menggunakan cara yang mirip dengan *local file system*. Walaupun bukan sistem yang muncul pertama kali, namun NFS terus berkembang dan berevolusi sehingga banyak digunakan sebagai *network file system* di UNIX. NFS memperbolehkan sharing dari sebuah *file system* yang umum diantara banyak *user* dan memberikan keuntungan dari *data centralizing* untuk meminimalkan *storage* yang diperlukan [4].

2.3 *Internet Small Computer Systems Interface*

Internet Small Computer Systems Interface adalah sebuah protokol berbasis TCP/IP untuk mengirimkan perintah SCSI melalui *network* berbasis IP. Struktur iSCSI bisa terus dikembangkan melebihi local LAN dan digunakan pada WAN atau bahkan internet. iSCSI memperlakukan *external volume* sebagai *hard disk internal* [14].

2.4 Tinjauan Studi

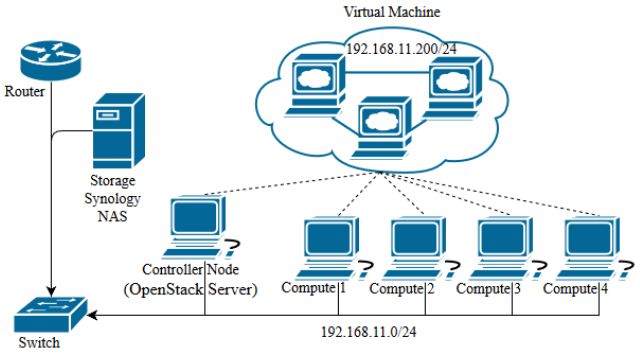
Agar dapat memahami apa yang harus dilakukan pada saat melakukan *testing* protokol NFS dan iSCSI maka dilakukan tinjauan studi pada penelitian Hashimoto yang berjudul Comparing Filesystem Performance in Virtual Machines [2]. Pengujian yang dilakukan menggunakan IOzone *filesystem benchmark*. Menurut hasil penelitian Hashimoto, performa read NFS sangat bagus untuk file dengan record size yang kecil. Hashimoto menduga bahwa NFS melakukan *read-ahead* dan memanfaatkan *cache* untuk bisa mendapatkan performa sebaik itu. Sedangkan untuk alasan mengapa *virtual machine* dapat memiliki performa yang lebih baik dibandingkan *host* fisiknya, Hashimoto menduga bahwa *hypervisor* telah melakukan *buffering* untuk *read* dari *virtual machine*. Pada hasil pengujian untuk write ini disimpulkan bahwa performa NFS menjadi sangat buruk akibat adanya keterbatasan pada network.

3. DESAIN SISTEM

3.1 Desain Sistem

Desain sistem yang dibuat terdiri dari sebuah *controller node* dan 4 komputer sebagai *compute node*. *Controller node* yang berperan sebagai server OpenStack tersebut menggunakan OpenStack project keystone (*identity*), glance (*image*), nova (*compute*), neutron (*network*), horizon (*dashboard*), dan cinder (*block storage*). Pada *compute node* digunakan OpenStack project nova.

Untuk dapat menyimpan data *virtual machine* (VM) yang dijalankan di *compute node* dan memakai data itu kembali diperlukan OpenStack project cinder karena adanya fasilitas *persistent storage*. Sedangkan penyimpanan data tersebut disimpan pada penyimpanan eksternal, yaitu *Network Attached Storage*. Untuk sementara *controller node* dibuat agar hanya dapat diakses melalui jaringan lokal saja. Desain sistem yang dibuat dapat dilihat pada Gambar 1.



Gambar 1. Desain sistem dan jaringan

3.1.1 Desain Controller Node

Langkah pertama adalah melakukan instalasi sistem operasi pada *controller node*. Sistem operasi yang digunakan adalah Ubuntu server 16.04 LTS 64 bit. Selanjutnya *controller node* di*install* dengan project-project OpenStack versi Pike, yaitu OpenStack project keystone, glance, nova, neutron, horizon, dan cinder. *Controller node* juga di*install* dengan MariaDB, RabbitMQ dan Memcached sebagai *pre-requirement* dari OpenStack versi Pike. Selain itu *Network Time Protocol* (NTP) juga perlu di*install* sebagai sarana untuk sinkronisasi waktu.

Instalasi NTP perlu dilakukan pada *controller node*. NTP pada *controller node* ini berfungsi untuk menyediakan sarana sinkronisasi waktu bagi semua *compute node* yang ada pada sistem sehingga seluruh *node* yang ada pada sistem memiliki waktu yang sama dan tidak memiliki masalah soal perbedaan *timestamp*.

Database bersifat wajib untuk di*install* karena digunakan untuk menyimpan data semua *service* yang dijalankan OpenStack. Instalasi MariaDB ini perlu dilakukan pada *controller node*. MariaDB sendiri adalah sebuah *relational database management system* (RDBMS) berbasis *open source* yang disarankan dalam penggunaan OpenStack.

Instalasi RabbitMQ perlu dilakukan pada *controller node*. RabbitMQ menyediakan sarana agar semua komponen *services* OpenStack dapat saling berkomunikasi. Semua *service* OpenStack yang berjalan di RabbitMQ akan bertindak sebagai *guest*.

Instalasi memcached dilakukan pada *controller node*. Memcached berfungsi untuk mempercepat akses data external (misalnya database dan API) dengan menggunakan sistem *distributed memory object caching*.

Instalasi keystone dilakukan pada *controller node*. Keystone ini berfungsi untuk menyimpan data user, service, dan endpoint semua OpenStack project lainnya dan juga sebagai *authenticator* untuk semua OpenStack *service*. Selain itu keystone juga berlaku sebagai user yang memiliki hak akses administrasi dan juga memiliki *role* administrasi. Semua *service* OpenStack lainnya harus memiliki *user*, *service*, dan *endpoint* yang terdaftar di keystone agar servicenya bisa digunakan.

Instalasi glance dilakukan pada *controller node*. Glance berfungsi menyediakan sebuah direktori khusus untuk menyimpan image

yang digunakan untuk menyalakan *virtual machine*. Glance juga harus memiliki database di MariaDB beserta akun dengan hak admin pada database tersebut.

Instalasi nova dilakukan pada *controller node*. Nova berfungsi untuk menyediakan *compute instance*. Nova juga membutuhkan database di MariaDB beserta akun dengan hak admin atas database tersebut.

Instalasi neutron dilakukan pada *controller node*. Neutron berfungsi untuk menyiapkan *virtual network* untuk *virtual machine* yang akan dijalankan nantinya.

Instalasi horizon dilakukan pada *controller node*. Horizon berfungsi untuk menyediakan *user interface* berbasis web untuk *service* OpenStack terutama nova, termasuk untuk memilih *flavor* dan menjalankan *virtual machine*.

Instalasi cinder dilakukan pada *controller node*. Cinder berfungsi untuk menyediakan *volume services* ke *end user* yaitu *virtual machine* melalui OpenStack nova.

3.1.2 Desain Compute Node

Langkah pertama adalah melakukan instalasi sistem operasi pada *compute node*. Sistem operasi yang digunakan adalah Ubuntu server 16.04 LTS 64 bit. *Compute node* menggunakan 4 komputer yang ada di dalam laboratorium multimedia yaitu compute1, compute2, compute3, dan compute4. Selanjutnya *compute node* di*install* dengan project OpenStack versi Pike yang dibutuhkan untuk melakukan komputasi, yaitu OpenStack project nova. *Network Time Protocol* (NTP) juga di*install* sebagai sarana untuk sinkronisasi waktu.

Instalasi NTP perlu dilakukan pada *compute node*. NTP pada *compute node* ini berfungsi untuk mengsinkronisasikan waktu *compute node* dengan waktu *controller node* sehingga tidak ada masalah soal perbedaan *timestamp*.

Instalasi nova dilakukan pada *compute node*. Nova pada *compute node* berfungsi agar *user* bisa me*launch virtual machine* menggunakan *compute service* yang disediakan.

3.2 Desain Jaringan

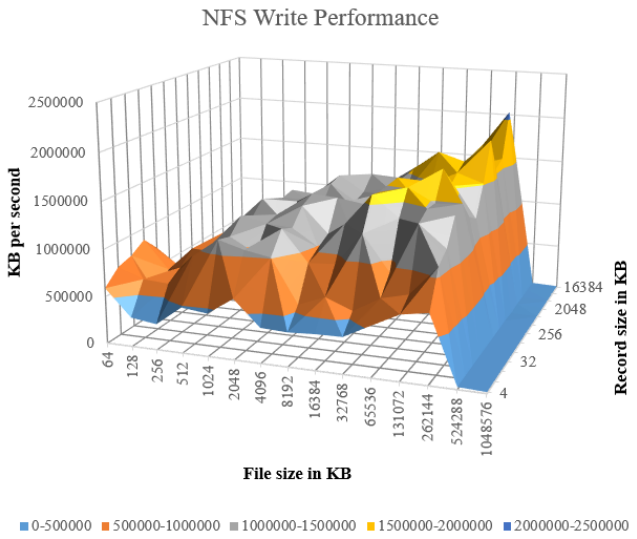
Untuk pembuatan skripsi ini digunakan sebuah *router* milik laboratorium multimedia yang memiliki network 192.168.11.0, *subnet mask* 255.255.255.0, *default gateway* 192.168.11.1, DNS server 203.189.120.4 dan 203.189.120.7, serta tidak membutuhkan *proxy* untuk mengakses internet. Rencananya network yang akan digunakan untuk server dan komputer *client* yang ada di dalam laboratorium multimedia adalah 192.168.11.0/24 dan network yang akan digunakan untuk *instance* VM menggunakan range IP 192.168.11.200/24 – 192.168.11.254/24 melalui router laboratorium multimedia sehingga network yang dibuat untuk keperluan penelitian ini tidak akan bertabrakan dengan jaringan yang ada di Universitas Kristen Petra.

Komputer-komputer yang ada di dalam laboratorium multimedia diatur dengan menggunakan IP *static* dengan komputer pertama dimulai dari nomor 6 menggunakan 192.168.11.6, komputer kedua adalah nomor 7 menggunakan 192.168.11.7 dan seterusnya. Desain jaringan yang telah direncanakan untuk digunakan oleh *controller node*, *compute node*, *virtual machine* dan storage NAS Synology dapat dilihat pada Gambar 1.

4. PENGUJIAN SISTEM

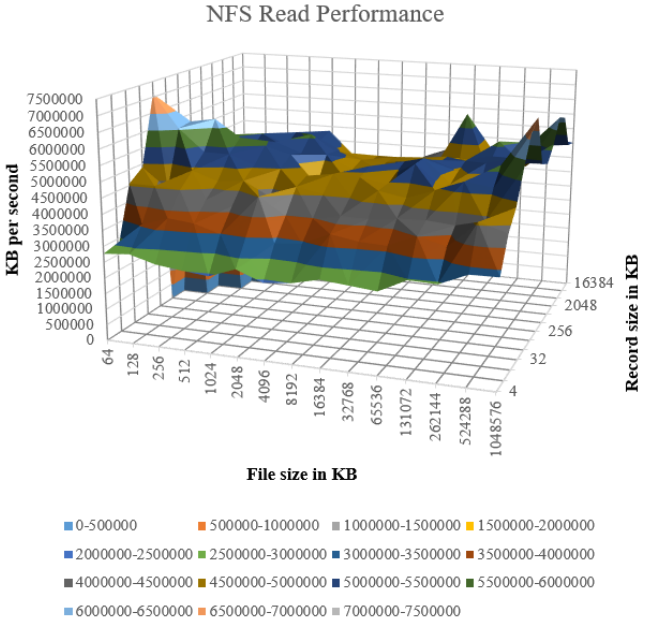
4.1 Hasil *Testing* Protokol NFS dan iSCSI

Pengujian ini dilakukan untuk membandingkan kinerja protokol NFS dan iSCSI. Semua testing protokol NFS dan iSCSI dilakukan pada *virtual machine*. Pada Gambar 2 dan 3 terlihat grafik hasil *testing* dari NFS yaitu performa *write* dan *read* dengan menggunakan tool IOzone.



Gambar 2. Performa *write* NFS

Pada Gambar 2 dapat dilihat bahwa kecepatan *throughput* menurun secara drastis pada saat *file size* berukuran sebesar 524288 Kb dan terus menurun, hal ini mungkin disebabkan oleh pengaruh keterbatasan *filesystem* ext4 dan keterbatasan pada *processor* yang digunakan oleh *virtual machine*.

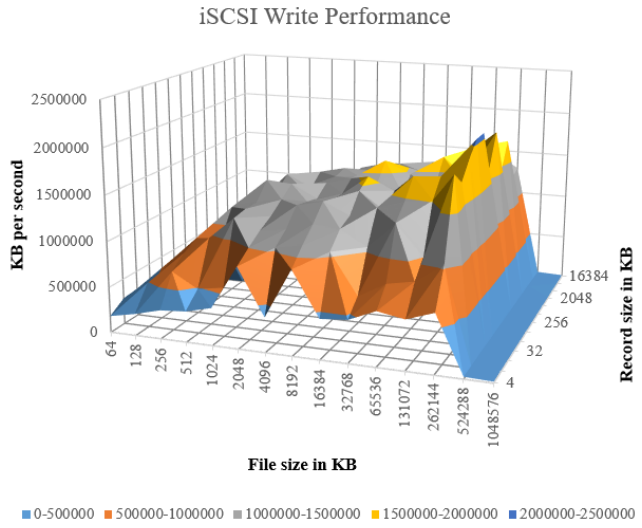


Gambar 3. Performa *read* NFS

Pada Gambar 3 dapat dilihat bahwa *throughput* untuk *read* pada *virtual machine* lebih tinggi daripada dilakukan pada *physical*

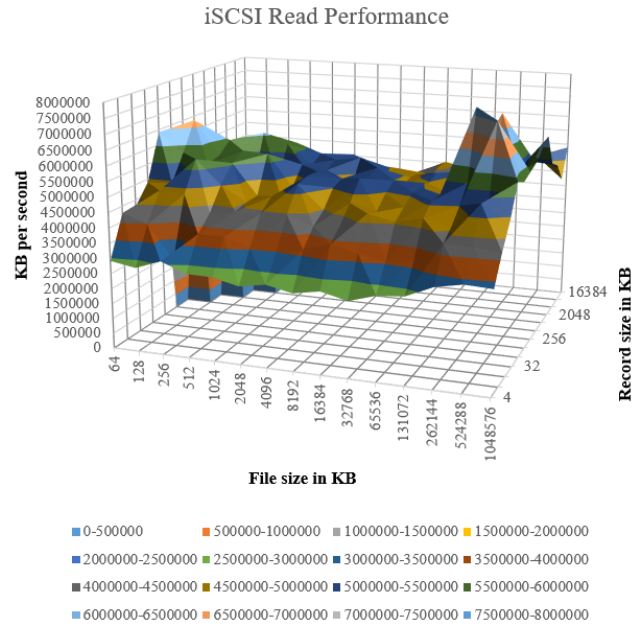
machine secara langsung. Hal ini mungkin terjadi akibat dari pengaruh aktivitas *buffering* yang dilakukan oleh *hypervisor*.

Selanjutnya pada Gambar 4 dan 5 terlihat grafik hasil *testing* dari iSCSI yaitu performa *write* dan *read* dengan menggunakan tool IOzone.



Gambar 4. Performa *write* iSCSI

Pada Gambar 4 dapat dilihat bahwa kecepatan *throughput* menurun secara drastis pada saat *file size* berukuran sebesar 524288 Kb dan terus menurun, hal ini mungkin disebabkan oleh pengaruh keterbatasan *filesystem* ext4 dan keterbatasan pada *processor* yang digunakan oleh *virtual machine*.



Gambar 5. Performa *read* iSCSI

Pada Gambar 5 dapat dilihat bahwa *throughput* untuk *read* pada *virtual machine* lebih tinggi daripada dilakukan pada *physical machine* secara langsung. Hal ini mungkin terjadi akibat dari pengaruh aktivitas *buffering* yang dilakukan oleh *hypervisor*.

Berdasarkan *throughput* yang dihasilkan oleh protokol NFS dan iSCSI, protokol iSCSI memiliki *throughput* yang lebih baik dibandingkan protokol NFS, baik dari performa saat melakukan aktivitas *write* maupun *read*. Karena itu diambil kesimpulan bahwa iSCSI adalah protokol yang lebih baik untuk diimplementasikan pada OpenStack cinder dibandingkan NFS karena protokol iSCSI lebih cepat daripada NFS.

5. KESIMPULAN

Dari penelitian yang dilakukan kesimpulan yang bisa ditarik sebagai berikut:

- NFS memiliki keunggulan saat melakukan *write file* yang memiliki *record size* yang kecil, sedangkan iSCSI memiliki keunggulan saat melakukan *write file* yang memiliki *record size* yang besar. Dalam melakukan *read* tidak terlihat perbedaan yang mencolok antara NFS dan iSCSI. Secara keseluruhan proses baik *write* maupun *read* iSCSI memiliki keunggulan kecil dalam *throughput*nya, karena itu pada skripsi ini diambil kesimpulan bahwa iSCSI adalah protokol yang lebih baik untuk diimplementasikan pada OpenStack cinder dibandingkan NFS.
- Konfigurasi yang paling baik untuk diimplementasikan adalah dengan menggunakan file konfigurasi yang sudah disediakan oleh OpenStack. Konfigurasi protokol NFS dan iSCSI pada sisi client dapat dilakukan secara langsung pada file konfigurasi cinder, sehingga tidak perlu melakukan konfigurasi manual pada komputer. Untuk konfigurasi protokol NFS dan iSCSI pada sisi server dilakukan secara langsung pada storage NAS Synology karena sistem NAS synology dapat menerima konfigurasi NFS dan membuat iSCSI LUN dan iSCSI target sendiri.

6. DAFTAR REFERENSI

[1] Akhtar, S. 2013. *Storage as a Service and Openstack Cinder*. URI = https://www.slideshare.net/openstackindia/storage-as-a-service-and-openstack-cinder

[2] Hashimoto, M. 2014. *Comparing Filesystem Performance in Virtual Machines*. URI = http://mitchellh.com/comparing-filesystem-performance-in-virtual-machines

[3] Hurwitz, J., Kaufman, M., & Halper, F. 2012. *Cloud Services For Dummies ® IBM Limited Edition*. New Jersey: John Wiley & Sons, Inc.

[4] Jones, M. 2010. *Network file systems and Linux*. URI = https://www.ibm.com/developerworks/library/l-network-filesystems/index.html

[5] Khedher, O., & Chowdhury, C. D. 2017. *Mastering OpenStack Second Edition*. Birmingham: Packt Publishing Ltd.

[6] MariaDB. About MariaDB. URI = https://mariadb.org/about/

[7] Memcached. *What is Memcached*. URI = https://memcached.org/

[8] NTP. *What is NTP*. URI = http://www.ntp.org/ntpfaq/NTP-s-def.htm

[9] OpenStack. 2018. *OpenStack Compute (nova)*. URI = https://docs.openstack.org/nova/pike/

[10] Pacevic, R., & Kaceniauskas, A. 2016. *The development of VisLT visualization service in Openstack cloud infrastructure*. Vilnius: Elsevier Ltd.

[11] RabbitMQ. *What can RabbitMQ do for you*. URI = https://www.rabbitmq.com/features.html

[12] Redhat. URI = https://www.redhat.com/en/topics/openstack

[13] Rhoton, J. 2013. *The Storage components Swift and Cinder*. URI = https://www.ibm.com/developerworks/cloud/library/cl-openstack-swift-cinder/index.html

[14] Singh, M. 2015. *Internet Small Computer Systems Interface*. URI = https://www.slideshare.net/manojsinghdhn/internet-small-computer-system-interface

[15] Solberg, M., & Silverman, B. 2017. *OpenStack for Architects*. Birmingham: Packt Publishing Ltd.