IMPLEMENTASI *DISTRIBUTED BARTERING ALGORITHM* PADA *LIVE MIGRATION* MESIN VIRTUAL

Meliza Gratia Dorothy Latuputty [1], Sabriansyah Rizqika Akbar, S.T, M.Eng [2], Achmad Basuki, S.T, M.MG, Ph.D [3]
Informatika
Fakultas Ilmu Komputer
Universitas Brawijaya
Malang 65145, Indonesia
Email: meliza.latuputty29@gmail.com[1], sabrian@ub.ac.id [2], abazh@ub.ac.id [3]

ABSTRAK

Munculnya *cloud computing* telah menyebabkan perkembangan minat dalam penyebaran dan penggunaan berbagai macam aplikasi pada lingkungan komputasi. Keberhasilan *cloud computing* didorong dengan adanya penggunaan virtualisasi sebagai teknologi yang mendasari mereka. Namun, dalam beberapa aplikasi, perhitungan analisis ilmiah dan data yang memiliki pola yang rumit dari komunikasi sehingga membutuhkan pertukaran data dalam jumlah besar untuk melaksanakan perhitungan yang akan dilakukan. Untuk aplikasi tersebut, pola komunikasi antar komponen yang berbeda merupakan faktor kunci yang harus diperhatikan selama pengalokasian sumber daya. Selain itu, sering ditemukan *bandwidth* jaringan menjadi sumber kemacetan dibanyak platform virtualisasi yang ada. Berdasarkan permasalahan tersebut, dilakukan percobaan proses komunikasi yang dapat mengusulkan perpindahan mesin virtual ke server lain. Penulis menggunakan *distributed bartering algorithm* dimana server fisik secara independen bernegosiasi untuk penempatan virtual machine (VM) atas dasar informasi lokal, berdasarkan gangguan afinitas yang berbeda antar pasangan VM, dan negosiasi penempatan yang lebih baik untuk VM. Jika proses negosiasi selesai maka dilakukan migrasi VM yang akan saling bertukar volume pada trafik jaringan yang lebih dekat satu sama lain. Hasil pengujian menunjukkan proses pengimplementasian *distributed bartering algorithm* dapat berjalan dengan baik. *Distributed bartering algorithm* dapat menjadi salah satu solusi untuk mengurangi kemacetan dan *overhead* jaringan. Hasil pengujian memperlihatkan bahwa rata-rata dari tiap skenario tidak melebihi ambang batas yaitu 400 Mbits/sec sehingga hal ini menunjukan bahwa metode yang diusulkan mampu menyeimbangkan beban trafik data antar HVM dan meminimalisir terjadi kemacetan dan *overhead* jaringan. Metode negosiasi dan *swapping* antar HVM juga dapat diterapkan dengan baik dan mampu menyeimbangkan kelebihan beban yang terjadi antar HVM.
Kata kunci: Virtualisasi, *Distributed bartering algorithm, Live migration*

ABSTRACT

The emergence of cloud computing has led to the development of interest in the deployment and the use of a wide range of applications in the computing environment. The success of cloud computing is driven by their use of virtualization as their underlying technology. However, in some applications, scientific analysis and calculation of data which have complex patterns of communication and exchange depedensi data requires large amounts of data to perform calculations to be performed. For such applications, the pattern of communication between the different components is a key factor that must be considered during the allocation of resources. Additionally, it is often found that network bandwidth become source of bottlenecks in many existing virtualization platform. Based on these problems, the authors conducted communication process experiments to propose a transfer of virtual machines to other servers. The author uses a distributed bartering algorithm where physical server independently negotiate for VM placement on the basis of local information, based on the different affinity interference between the pair VM, and negotiating better placement for the VM. If the negotiation process is completed then performed VM migration that would exchange traffic volume on the network closer to each other. The test results showed that distributes bartering algorithm implementation can run well. Distributed bartering algorithm can be one solution to reduce congestion and network overhead. The test results also showed that the average of each scenario did not exceed the threshold of 400 Mbits/sec so this shows that the proposed method is able to balance the load of data traffic occurs between HVM and minimize congestion and network overhead. Negotiation and swapping between HVM also be applied properly and are able to balance the overload that occurs between HVM.

Keywords: Virtualisation, *Distributed bartering algorithm, Live migration*

1. PENDAHULUAN

1.1. Latar Belakang

Saat ini perkembangan teknologi informasi memegang peranan penting pada kehidupan manusia. Seiring dengan pesatnya pembangunan yang ditandai dengan semakin canggihnya arsitektur komputer saat ini, teknologi internet dapat dikembangkan menjadi komputasi awan. Dengan komputasi awan, pengguna dapat mengakses semua aplikasi dan dokumen dari tempat manapun dan menggunakan gadget apapun. Komputasi awan juga menyajikan tren teknologi yang signifikan, dan secara jelas membentuk kembali proses teknologi informasi dan pasar IT (Furth, 2010).

Dengan keberhasilan cloud, telah mendorong adanya penggunaan virtualisasi sebagai teknologi yang mendasarinya. Virtualisasi merupakan fitur kunci dari komputasi awan yang memungkinkan beberapa mesin virtual untuk berbagi sumber daya pada mesin fisik tunggal yang dapat meningkatkan pemanfaatan sumber daya data center (Nilesh & Rajesh, 2013). Mesin virtual (VM) memberikan fleksibilitas dan mobilitas melalui proses migrasi yang mudah, yang memungkinkan pemetaan yang dinamis pada mesin virtual (VM) ke sumber daya yang tersedia. Mesin virtual (VM) juga menyediakan isolasi dan keamanan kinerja yang memfasilitasi multiplexing dan pemanfaatan pembagian sumber daya. Namun, dalam beberapa aplikasi perhitungan analisis ilmiah dan data yang memiliki pola yang rumit dari komunikasi dan depedensi data, membutuhkan pertukaran data dalam jumlah besar untuk melaksanakan perhitungan yang akan dilakukan. Untuk aplikasi tersebut, pola komunikasi antar komponen yang berbeda merupakan faktor kunci yang harus diperhatikan selama pengalokasian sumber daya. Selain itu, di banyak platform virtualisasi yang ada sering ditemukan *bandwidth* jaringan yang menjadi sumber kemacetan (Armbrust, 2009).

Akibat dari permasalahan itu, penyedia cloud berusaha untuk mengurangi *overhead* jaringan sebanyak mungkin. Selain mempertimbangkan karakter fisik (kecepatan CPU, memori dan penyimpanan) dari node yang ada pada mesin virtual yang ditempatkan, topologi jaringan juga harus menjadi pertimbangan dalam meningkatkan efisiensi platform dan mengurangi permasalahan jaringan.

Berdasarkan permasalahan tersebut, dilakukan proses komunikasi yang dapat mengusulkan perpidahan mesin virtual ke server lain. Penulis menggunakan *distributed bartering*

algorithm yang dimana server fisik secara independen bernegosiasi untuk penempatan VM atas dasar informasi lokal, berdasarkan gangguan afinitas yang berbeda antar pasangan VM, negosiasi penempatan yang lebih baik untuk VM. Jika proses negosiasi selesai maka dilakukan migrasi VM yang dimana akan saling bertukar volume pada trafik jaringan yang lebih dekat satu sama lain.

Harapan yang ingin dicapai dari penelitian ini adalah penerapan *distributed bartering algorithm* dapat berhasil dengan baik dan dapat melakukan migrasi serta proses negosiasi penempatan VM juga dapat berjalan dengan optimal sehingga dapat mengatasi permasalahan biaya komunikasi yang tinggi pada server fisik.

1.2. Rumusan Masalah

Berdasarkan pada permasalahan yang diangkat, maka dapat dirumuskan masalah untuk penelitian ini adalah sebagai berikut:

1. Bagaimana menerapkan *distributed bartering algorithm* pada *live migration* mesin virtual?
2. Bagaimana metode negosiasi dan *swapping* melakukan optimasi *live migration* mesin virtual pada lingkungan komputasi awan?

1.3 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah menerapkan *distributed bartering algorithm* pada proses *live migration* sehingga dapat mengurangi kemacetan jaringan dan meminimalisir biaya komunikasi jaringan.

1.4 Manfaat

Pada penelitian ini proses migrasi menggunakan *distributed bartering algorithm* diharapkan dapat membantu proses negosiasi penempatan VM sehingga VM dapat bertukar besar volume lalu lintas jaringan yang lebih dekat satu dengan lain.

1.5 Batasan Masalah

Pembatasan masalah penelitian ini adalah sebagai berikut.

1. Semua komputer yang digunakan dalam penelitian memiliki spesifikasi yang sama dan sistem dibangun pada segmentasi jaringan lokal.
2. Sumber daya komputasi hanya fokus pada penggunaan *bandwidth*.
3. Semua mesin virtual yang dibangun berbagi satu media penyimpanan yang sama.
4. Pembuatan program perangkat lunak tambahan menggunakan Bash Shell.
5. Pengujian sistem dilakukan dengan membebani server virtual yaitu melakukan ping secara terus menerus antar VM.

2. KAJIAN PUSTAKA DAN DASAR TEORI

2.1. Kajian Pustaka

Penelitian yang terkait dengan pengoptimalan kinerja *live migration* mesin salah satunya telah dilakukan oleh Angga Aditya (2012) dengan judul "Optimasi Penggunaan Sumber Daya Komputasi Di Lingkungan Komputasi Awan". Dalam penelitian tersebut penulis menjabarkan bahwa sistem dibangun dengan menggunakan metode *load balancing* untuk melakukan penyebaran beban kerja dengan memindahkan mesin virtual pada server yang memiliki beban kerja CPU tinggi ke server yang memiliki beban kerja CPU rendah. Sedangkan metode *load aggregation* digunakan jika sistem mendeteksi penurunan beban kerja CPU dalam satu atau lebih server fisik, beberapa mesin virtual akan digabungkan kembali ke dalam sebuah server fisik virtualisasi. Berdasarkan penelitian tersebut, sistem yang dibangun dapat mengoptimalkan pengunaan sumber daya komputasi dan menyeimbangkan beban kerja CPU dalam semua server fisik.

Penelitian terkait lainnya mengenai pengoptimalan *live migration* salah satunya dilakukan oleh Jason Sonnek (2010) dengan judul "*Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration*". Penelitian tersebut menyajikan teknik migrasi desentralisasi affinity-aware yang menerapkan *distributed bartering algorithm* pada proses *live migration* mesin virtual secara dinamis menyesuaikan penempatan VM sehingga biaya *overhead* komunikasi dapat diminimalkan.

Dengan melihat beberapa penelitian yang telah dilakukan sebelumnya maka penelitian ini

berfokus tentang bagaimana cara untuk melakukan optimasi *live migration* mesin virtual dengan menggunakan *distributed bartering algorithm*.

2.2 Virtualisasi

Virtualisasi adalah teknologi arsitektur komputer dengan yang beberapa mesin virtual (VM) multiplexing dalam mesin hardware yang sama. Tujuan dari VM adalah untuk meningkatkan berbagi sumber daya oleh banyak pengguna dan meningkatkan kinerja komputer dalam hal pemanfaatan sumber daya dan fleksibilitas aplikasi. Hardware sumber daya (CPU, memori, I/O perangkat, dll) atau sumber daya perangkat lunak (sistem operasi dan perpustakaan software) dapat virtual di berbagai lapisan fungsional (Kai Hwang, 2012).

Pada lingkungan komputasi awan, sebagian besar penyedia layanan cloud mengunakan teknik virtualisasi. Hal ini dikarenakan dengan menggunakan teknologi virtualisasi, pemanfaatan sumber daya komputasi lebih efisien dan memungkinkan beberapa penguna untuk bebragi infrastuktur fisik sama serta sumber data yang lainya. Selain itu, terdapat beberapa kelebihan teknik virtualisasi lainya, yaitu efektifitas perbaikan proses pemulihan kegagalan sistem, pengurangan kebutuhan konsumsi energi, kebutuhan tempat, beban jaringan, serta pengurangan biaya tes dan pengembangan perangkat lunak.

2.3 Pendekatan Virtualisasi

Terdapat dua jenis pendekatan teknik virtualisasi, yaitu full virtualization dan paravirtualization:

1. *Full virtualization*
   Full virtualization adalah salah satu teknik virtualisasi yang seluruh sistem komputer dibuat ke dalam kontruksi software, dimana konstruksi tersebut berfungsi sebagai hardware aslinya (Karen, 2011).
2. *Paravirtualization*
   Paravirtualization merupakan teknik virtualisasi yang efisien dan ringan yang implementasinya digunakan pada berbagai macam lingkungan mesin virtual, dengan lingkungan mesin virtual hanya disediakan simulasi perangkat keras secara sebagian.

2.4 *Hypervisor*

Hypervisor menciptakan beberapa virtual server dalam satu server fisik. Hypervisor memungkinkan penyatuan prosesor dan sumber daya memori. Instalasi hypervisor pada server host memungkinkan untuk menjalankan beberapa sistem operasi secara bersamaan menggunakan virtualisasi menggunakan teknik virtualisasi yang menyediakan dukungan infrastruktur untuk beberapa mesin virtual dengan virtualisasi sumber daya fisik perangkat keras (Sanjay, 2012).

Hypervisor mempunyai dua tipe: Tipe 1 (native, bare metal) mewakili teknik virtualisasi full virtualization sedangkan hypervisor tipe 2 (hosted) mewakili paravirtualization. Dalam penelitian ini, akan menerapkan teknik virtualisasi full virtualization dengan mengunakan hypervisor Kernel-Based Mesin virtual (KVM).

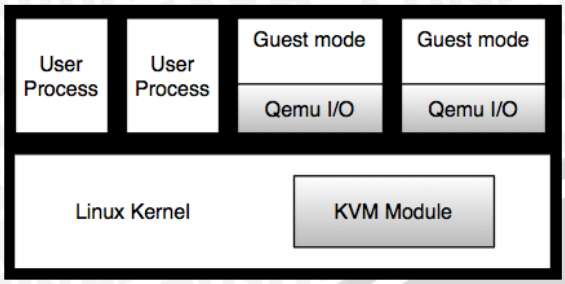
2.5 *Kernel-Based Virtual Machine* (KVM)

Kernel-Based Virtual Machine (KVM) adalah solusi virtualisasi kreatif berbasis Linux. Tidak seperti hypervisors umum lainnya yang melakukan dasar penjadwalan, manajemen memori sendiri, KVM menemukan kesamaan modul antara VM dan kernel sistem operasi, sehingga beralih ke kernel Linux untuk semua fungsi umum.

Terdapat dua prinsip yang membuat KVM menjadi hypervisor dengan kinerja tinggi dan melampaui open source hypervisors lainnya (Khoa, 2013).

1. Prinsip desain pertama meliputi:
   a. Memanfaatkan semua kemampuan virtualisasi hardware-dibantu disediakan oleh Intel® Virtualization Technology (VT) dan AMD® Secure Virtual Machine (SVM)
   b. Fitur ekstensi virtualisasi hardware terbaru.
   c. Eksploitasi kemampuan hardware sekaligus menjaga *overhead* virtualisasi KVM
2. Prinsip desain kedua meliputi:
   a. Memanfaatkan sistem operasi Linux
   b. Memenuhi banyak komponen yang dibutuhkan oleh hypervisor, seperti

manajemen memori, penjadwalan, I/O stack, dan device driver dan sebagainya tidak perlu menulis dari awal sehingga lebih efisien.



Gambar 2.1 Arsitektur Virtualisasi KVM
Sumber : Timo (2011).

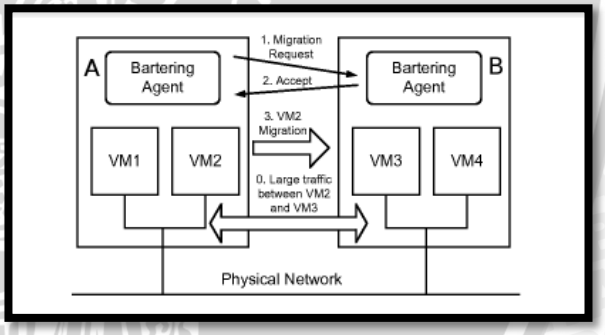
2.6 *Live migration*

Penggunaan fitur *live migration* ini adalah bentuk upaya yang dilakukan untuk melakukan penambahan sumber daya komputasi melalui ekspansi jumlah data center dengan pemindahan layanan dari data center yang ada ke data center yang baru (Al-Kiswany, 2011). Saat proses *live migration*, klien tidak ikut terlibat dalam proses ini sehingga tetap dapat mengakses layanan secara normal tanpa merasakan adanya perpindahan server. Berikut merupakan urutan alogritma umum *live migration* mesin virtual antara salah satu mesin fisik ke mesin fisik yang lainya (Lublin, 2007):

1. *Request* migrasi mesin virtual diterima (migrasi mesin virtual dari mesin fisik A ke mesin fisik B)
   a. Membuat perintah-perintah eksternal.
   b. Menghubungkan dan mengirimkan header.
   c. Mengalokasikan pengaturan dan sumber daya.
2. Pemindahan *memory*
   a. Proses pertama, *memory pages* (iterasi pertama) ditransfer secara menyeluruh
   b. Untuk iterasi selanjutnya,mentransfer seluruh *dirty pages* dari iterasi i-1
   c. Proses dilakukan secara terus menerus hingga selesai
3. Mesin virtual dihentikan
4. Pemindahan *VM State*
   a. Setiap perangkat dipindahkan ke state masing-masing
   b. Pemindahan *dirty pages* dari iterasi terakhir
5. Melanjutkan mesin virtual
   a. Pada mesin B, jika proses migrasi berhasil Mengirimkan paket broadcast melalui Ethernet untuk memberikan informasi lokasi mesin virtual yang baru
   b. Pada localhost mesin A jika proses migrasi gagal

2.7 *Distributed bartering algorithm*

Distributed bartering algorithm diusulkan untuk meminimalkan *overhead* dan meningkatkan skalabilitas yang memungkinkan VM untuk bernegosiasi penempatan pada server fisik yang lebih dekat pada data yang mereka butuhkan. Algoritma ini menghindari kemacetan dan titik pusat kegagalan yang terkait dengan solusi terpusat, dan juga dapat beradaptasi lebih cepat untuk perubahan dinamis lokal pada perubahan pola dan topologi jaringan. Selain itu pendekatan desentralisasi tersebut memungkinkan penggunaan untuk kebijakan yang berbeda dengan agen yang berbeda berdasarkan lokasi serta persyaratan aplikasi secara spesifik dari VMS yang dikelola.



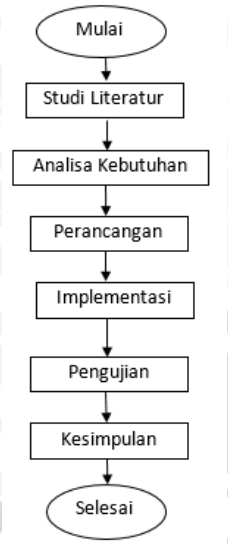
Gambar 2.2 Sistem *Distributed bartering algorithm*
Sumber: (Sonnek, 2009)

Ketika total trafik antar mesin virtual dan server fisik melebihi ambang batas maka agen barter akan melakukan negosiasi untuk relokasi dengan server yang diinginkan dan memulai migrasi jika berhasil.

3. METODE PENELITIAN

Langkah-langkah yang dilakukan dalam perancangan, implementasi, analisis dan pengujian dari sistem yang dibuat ditunjukkan pada Gambar 3.1. Kesimpulan dan saran disertakan sebagai catatan atas sistem dan

kemungkinan arah pengembangan sistem selanjutnya.



Gambar 3. 1. Diagram Alir Penelitian

3.1 Studi Literatur

Studi literatur berupa tinjauan teori yang berkaitan dengan penelitian ini. Teori-teori tersebut berkaitan dengan penelitian yang sudah pernah dilakukan, virtualisasi, pendekatan virtualisasi, *hypervisor*, *Kernel-Based Mesin virtual*, *live migration*, dan *distributed bartering algorithm* seperti yang tertuang dalam point 2.

3.2 Analisa Kebutuhan

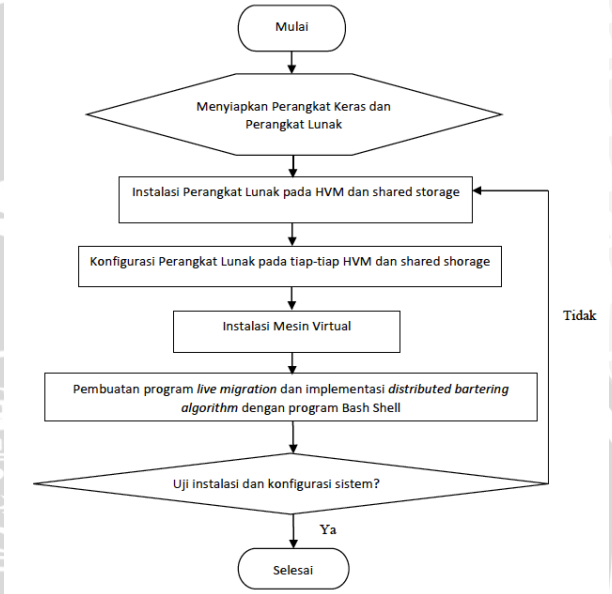
Analisis kebutuhan dilakukan dengan mengindentifikasi semua kebutuhan sistem, seperti perangkat keras, perangkat lunak, kebutuhan pendukung lainnya. Analisis juga dilakukan untuk mengetahui kondisi lapangan yang ada dan menjadi dasar dari pelaksanaan perancangan dan implementasi sistem yang akan dilanjutkan pada tahap selanjutnya.

3.3 Perancangan Sistem

Perancangan implementasi *distributed bartering algorithm* pada *live migration* ini dilakukan setelah tahap analisis kebutuhan selesai dilakukan. Tahap ini diperlukan agar peneliti lebih mudah dan tepat dalam melakukan proses implementasi pada tahap selanjutnya. Dalam penelitian ini, perancangan sistem terbagi atas dua bagian yaitu perancangan jaringan yaitu bagaimana membangun komunikasi antar perangkat keras yang ada pada sistem *live migration* dan perancangan perangkat lunak yaitu merancang program tambahan dengan bahasa Bash Shell.

3.4 Implementasi

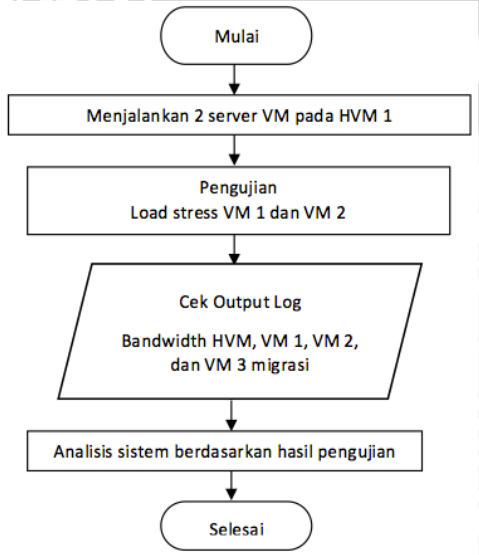
Implementasi dilakukan berdasarkan perancangan yang telah ditentukan. Tahap implementasi dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram Alir Implementasi

3.5 Pengujian dan Analisis Sistem

Pengujian dilakukan sesuai dengan diagram alir Gambar 3.3 berikut ini:



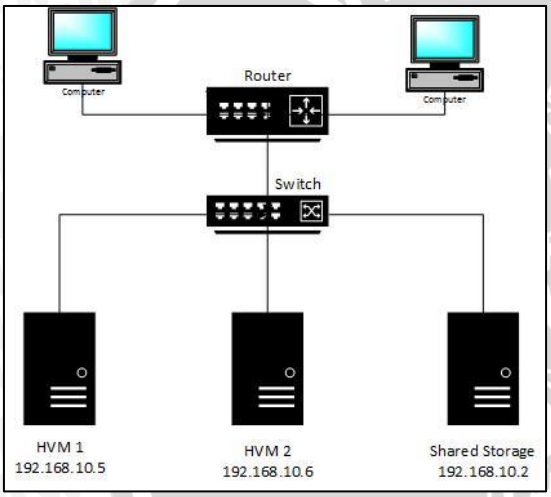
Gambar 3.3 Diagram Alir Pengujian

4 IMPLEMENTASI

4.1 Perancangan

4.1.1 Analisis Kebutuhan Perangkat Keras dan Lunak

Perancangan perangkat keras ini terdiri atas instalasi komputer HVM, komputer shared storage, media penghubung (jaringan), dan perangkat keras lainnya serta dibangun pada sebuah segmen jaringan lokal yang sama. Proses perancangan dilakukan dengan menghubungkan seluruh perangkat keras dalam jaringan lokal dengan kabel UTP dan memberikan alamat jaringan ke setiap HVM dan shared storage. Pada penelitian ini mengunakan alamat jaringan 192.168.10.0 dan subnet mask 255.255.255.0 untuk setiap HVM dan shared storage, seperti yang terlihat pada Gambar 4.1



Gambar 4. 1 Perancangan Jaringan Sistem

4.1.2 Perancangan Perangkat Lunak

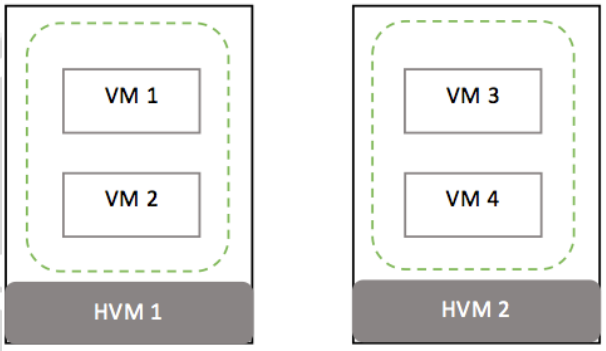
Pada penelitian ini, penulis memasang empat buah mesin virtual. Adapun alokasi sumber daya komputasi pada masing-masing server fisik dan mesin virtual adalah sebagai berikut:

Tabel 4.1 Alokasi Sumber Daya Komputasi

Server	CPU/vCPU	Memory	HDD
HVM 1	1	2 GB	100 GB
HVM 2	1	2 GB	100 GB
VM 1	1	1 GB	50 GB
VM 2	1	1 GB	50 GB
VM 3	1	1 GB	50 GB
VM 4	1	1 GB	50 GB

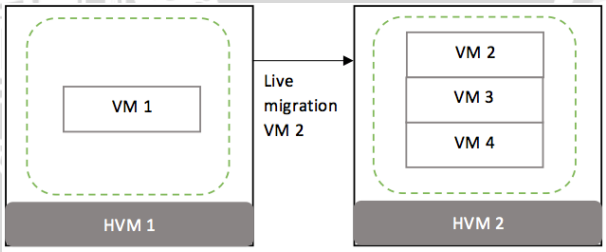
Ilustrasi cara kerja sistem:

- Pada gambar dibawah ini dijelaskan bahwa sistem ketika HVM 1 dan HVM 2 dalam keadaan trafik jaringan yang stabil, sehingga baik HVM 1 dan HVM 2 tidak ada yang kelebihan beban maka tidak ada kegiatan yang dilakukan.



Gambar 4.2 Trafik jaringan dalam keadaan stabil

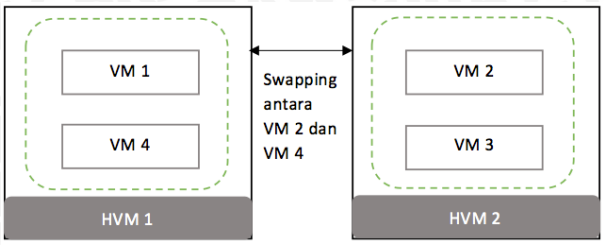
- Selanjutnya, berdasarkan parameter trafik jaringan yang tercatat pada log sistem maka dapat diperoleh VM yang dikategorikan memiliki beban trafik tinggi. Dalam penelitian ini, penulis menentukan ambang batas beban trafik adalah 400 Mbits/sec. Dari hasil perolehan tersebut maka VM yang melebihi ambang batas/kelebihan beban akan dipindahkan atau dialokasikan pada HVM yang lebih proporsional. Berikut merupakan gambaran secara umum proses perpindahan VM dari HVM 1 ke HVM 2.



Gambar 4.3 Proses Migrasi VM

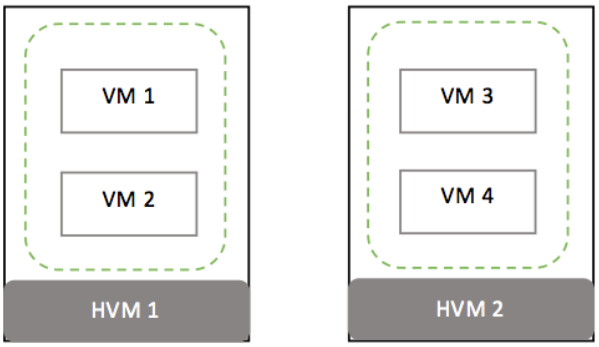
- Dalam penelitian ini ditentukan juga untuk memulai proses *swapping* maka ambang batas beban trafik adalah sekitar 200-300 Mbits/sec. Proses *swapping* merupakan salah satu bentuk pengimplementasian dari *distributed bartering algorithm*. Pada Gambar 4.4 dijelaskan bahwa proses *swapping* terjadi ketika HVM 1 memiliki trafik jaringan yang tinggi, jumlah beban dari VM 1 dan VM 2 melewati ambang batas, yang

sudah ditentukan sedangkan HVM 2 memiliki trafik yang stabil sehingga terjadi proses komunikasi antara VM dari HVM 1 yang memiliki throughput tinggi dengan VM dari HVM 2 yang memiliki throughput sesuai dengan batas yang ditentukan.



Gambar 4.4 Proses *Swapping* VM

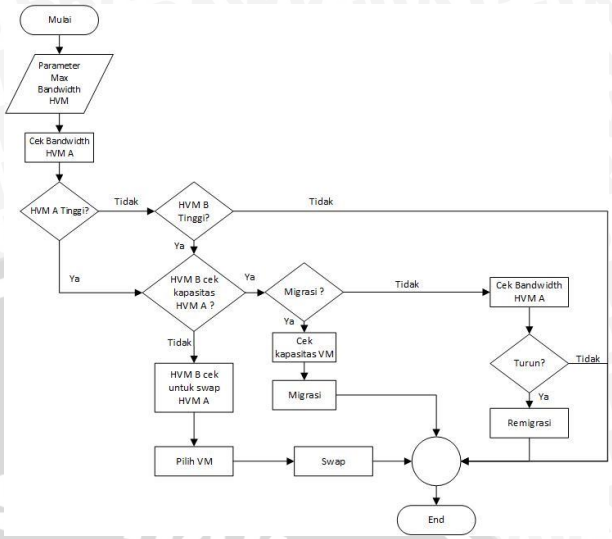
- Selanjutnya ketika trafik jaringan HVM 1 dan HVM 2 sama-sama tinggi maka tidak ada proses migrasi maupun swap yang dilakukan. Hal ini mengingat karena jika kedua HVM memiliki trafik yang tinggi maka tidak ada hal yang bisa dilakukan.



Gambar 4.5 Proses No Migration VM
Sumber: (Perancangan)

4.1.3 Algoritma Program

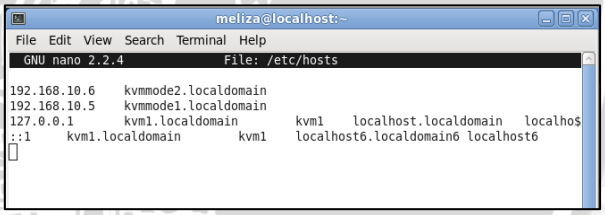
Untuk dapat melakukan proses implementasi *distributed bartering algortihm* memerlukan sebuah program tambahan *Bash Shell* yang dijalankan secara berkala dan terpusat pada HVM 1. Adapun diagram alir algoritma program tambahan tersebut



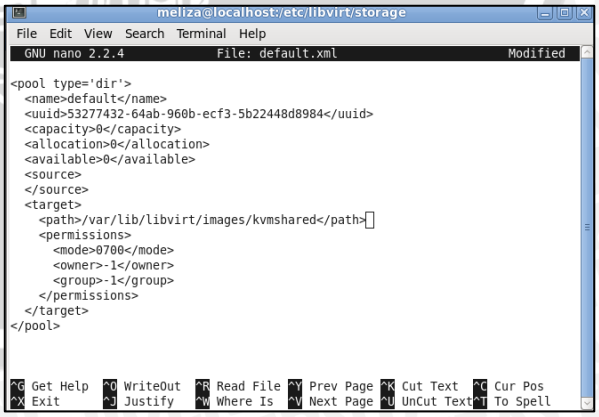
Gambar 4.6 Algoritma Program

4.2 Implementasi Sistem

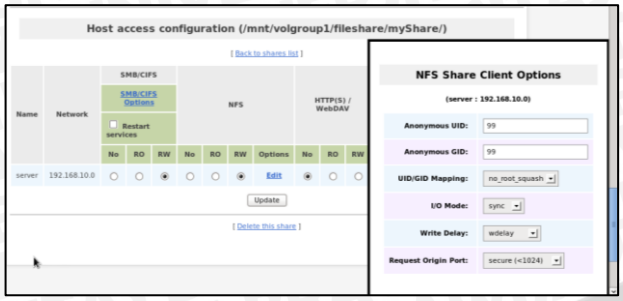
Implementasi sistem dalam penelitian ini meliputi instalasi HVM, konfigurasi perangkat lunak seperti pada Gambar 4.7 dan Gambar 4.8, implementasi *shared storage* seperti terlihat pada Gambar 4.9, dan implementasi mesin virtual yang terlihat pada Gambar 4.10 dan Gambar 4.11 .



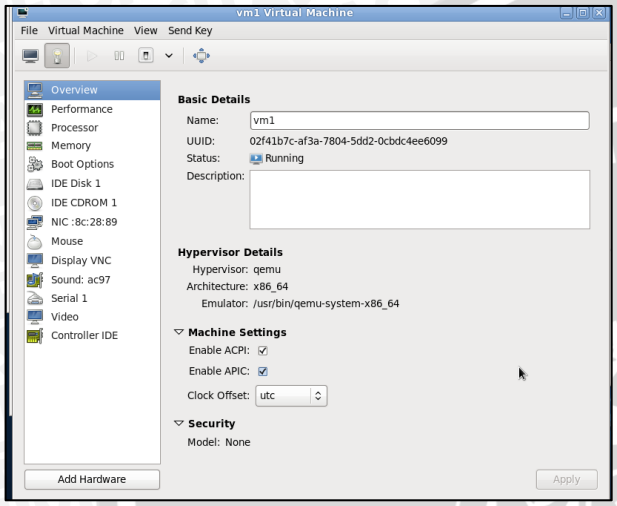
Gambar 4.7 Konfigrasi /etc/hosts



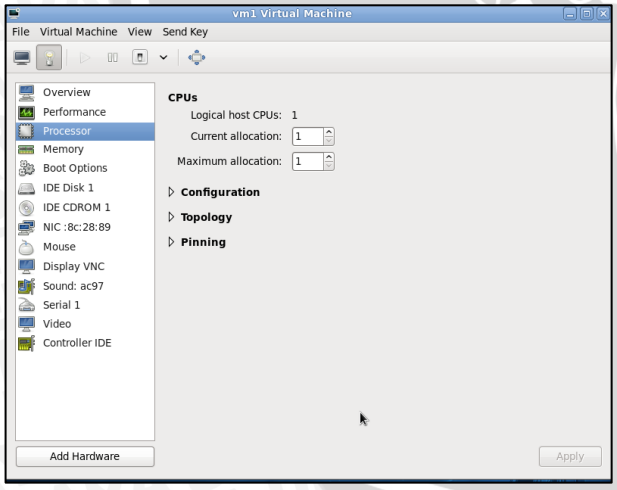
Gambar 4.8 Konfigrasi /etc/libvirt/storage/



Gambar 4.9 Konfigurasi NFS pada *shared storage*



Gambar 4.10 Detail Umum Spesifikasi Mesin Virtual 1

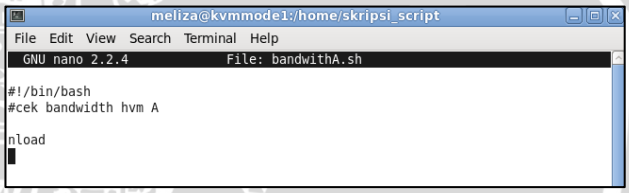


Gambar 4.11 Detail Processor Spesifikasi Mesin Virtual 1

Mengingat alokasi yang diberikan tidak besar maka diperlukan sistem operasi yang simple, dan tidak berat. Maka sistem operasi yang tepat dipasang pada mesin virtual tersebut adalah sistem operasi Damn Small Linux. Mesin virtual tidak memerlukan konfigurasi khusus karena IP dapat diperoleh secara otomatis dengan jaringan bridge.

Implementasi yang dilakukan selanjutnya adalah pembuatan program tambahan yang berupa berkas Bash Shell dan teks. Berkas-berkas tersebut berisi perintah-perintah shell yang digunakan untuk membangun sistem pada penelitian ini, dimana tiap-tiap berkas memiliki fungsi tersendiri. Berkas tersebut disimpan pada direktori /home/skripsi_script HVM 1 yang merupakan HVM utama. Berikut daftar berkas-berkas untuk program tambahan yang dibuat penulis.

- *bandwidth*A.sh : berkas shell ini berisi perintah untuk memonitoring *bandwidth* yang berjalan pada HVM 1. Proses monitoring dilakukan dengan menggunakan tool *nload*. Perintah monitoring tersebut dapat dilihat pada Gambar 4.12 berikut ini.



Gambar 4.12 Hasil *bandwidth*A.sh

- main.sh : merupakan berkas shell utama yang mengatur semua proses yang ada pada sistem penelitian ini. Isi dari main.sh adalah perintah parameter variable, pengambilan data, klasifikasi mesin virtual, migrasi atau remigrasi, proses *swapping* dan pencatatan log sistem seperti terlihat pada Gambar 4.13, Gambar 4.14, Gambar 4.15, Gambar 4.16, dan Gambar 4.17.

```
#!/bin/bash
fileTotalVM='/home/skripsi_script/totalvm_run.txt'
buf=copy-$1
```

Gambar 4.13. Perintah memasukkan variable

```
echo 'cleaning log : '$1 ', result on :'$buf
sed -i '/MIGRATION\ RUNNING/d' $buf
sed -i '/^$/d' $buf
sed -i 's/^.*VM1_L/VM1_L/g' $buf
sed -i 's/\t/\ /g' $buf
sed -i 's/\ \ /\ /g' $buf
IFS=$'\r\n' GLOBIGNORE='*' :; line=($(cat $buf))
bandwidth=$(( ${#line[@]} / 12 ))
echo ${#line[@]} " / 12 = "  $bandwidth
menit=0
idx=1
```

Gambar 4.14. Pengambilan data

```
if [ $(echo "$bandwidthvm1 >= 400" | bc -l) ]
then
    echo "vm1 bandwidth"
    IFS=$'\r\n' GLOBIGNORE='*' :; arr1line=($(cat arr1))
elif [ $(echo "$bandwidthvm2 >= 200-300" | bc -l) ]
then
    echo "vm2 bandwidth"
    IFS=$'\r\n' GLOBIGNORE='*' :; arr1line=($(cat arr2))
fi
```

Gambar 4.15. Perintah proses *swapping*

```
if [[ "$prev" == "low" && "$curr" == "high" ]]; then
        echo "*/"$menit" * * * * virsh migrate --live ds1_1
qemu+ssh://kvmmode2.localdomain/system" >> xyz
    elif [[ "$prev" == "high" && "$curr" == "low" ]]; then
        echo "*/"$menit" * * * * virsh --connect
qemu+ssh:///system" >> xyz
    fi
```

Gambar 16. Perintah proses migrasi atau remigrasi

- totalvm.sh : berkas shell yang berisi perintah virsh list yaitu untuk mencari tahu berapa jumlah mesin virtual yang berjalan di HVM 1.
- log.txt: berkas berisi pencatatan hasil eksekusi main.sh yang dieksekusi tiap menit. Pencatatan tersebut adalah waktu eksekusi, proses perpindahan mesin virtual (nama VM yang pindah dan HVM tujuan), *bandwidth* HVM 1 dan HVM 2, proses *swapping* antar VM, beban trafik jaringan VM 1, VM 2 dan VM 3.
- totalvm_run.txt : berkas ini berisi jumlah mesin virtual yang berjalan pada HVM 1.

5 PENGUJIAN DAN ANALISIS

5.1 PENGUJIAN

Pengujian dilakukan sesuai dengan perancangan topologi yang sudah dibahas pada point sebelumnya serta tidak memakai alat uji atau program uji tambahan tetapi cukup dengan membuat sibuk trafik jaringan dengan melakukan ping antar server secara terus menerus.

Pengujian dilakukan dengan mengunakan tiga skenario *stress load* yang mengacu pada diagram alir pengujian. Pada setiap skenario pengujian yang dilakukan, penulis mengambil rentan waktu selama 100 menit. Dalam rentan waktu 100 menit tersebut sistem akan bekerja secara reaktif terhadap trafik aliran data.

Selanjutnya, data hasil pengujian akan diperoleh melalui berkas log.txt yang telah dibuat oleh penulis. Berkas log.txt tersebut akan merekam beberapa data, antara lain *bandwidth* HVM 1 dan HVM 2, serta proses perpindahan mesin virtual dari HVM satu ke HVM yang lainnya, baik itu proses migrasi, *swapping*, maupun remigrasi. Migrasi adalah proses dimana terdapat mesin virtual yang berpindah dari HVM 1 ke HVM 2, *swapping* adalah proses dimana terdapat mesin virtual yang saling bertukar antara HVM 1 dan HVM 2 sedangkan remigrasi adalah proses dimana mesin virtual berpindah kembali dari HVM 2 ke HVM 1. Data yang diperoleh dari hasil pengujian dianalisis dan digunakan untuk menarik kesimpulan dan saran.

Pada pengujian skenario satu, *stress load* yang dilakukan hanya ditujukan pada HVM 1. Pada proses ini, HVM 1 yang terdiri dari VM 1 dan VM 2 diberikan *stress load* sesuai dengan hasil *bandwidth* yang diperoleh dengan menggunakan n*load*. Tujuan dari pengujian skenario satu ini dilakukan adalah untuk mengetahui proses migrasi dengan mengetahui perilaku sistem saat HVM 1 ketika diberikan *stress load* sesuai dengan karakteristik beban kerjanya. *Stress load* dilakukan pada VM 1 dengan menjalankan ping terhadap mesin virtual lainnya secara bersamaan dalam kurun waktu 100 menit. Hasil dari pengujian skenario satu adalah analisis fungsional sistem serta proses migrasi ketika *bandwidth* pada HVM 1 terkadang sibuk dan terkadang stabil.

Pada pengujian skenario dua, *stress load* ditujukan pada HVM 1 yang memiliki *stress load* yang sangat tinggi sedangkan pada HVM 2 memiliki *stress load* yang berada pada rentan 200-300 Mbits/sec. Pengujian ini bertujuan untuk mengetahui perilaku sistem ketika dua buah mesin virtual berjalan dengan *stress load* yang cukup tinggi serta melihat proses pertukaran mesin (*swapping*) yang dilakukan.

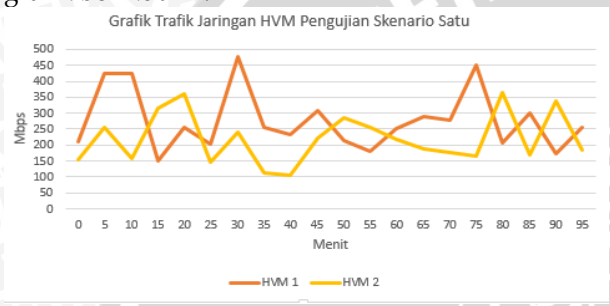
Pada pengujian skenario tiga, *stress load* ditujukan masih pada empat buah mesin virtual yaitu VM 1, VM 2, VM 3 dan VM 4 tetapi dengan keadaan *stress load* yang dipusatkan pada VM 3. Untuk VM 1 dan VM 2 *stress load* dilakukan sesuai dengan karakteristik beban kerjanya sedangkan VM 3 dan VM 4, dilakukan dengan beragam *stress load*.

5.2 ANALISIS

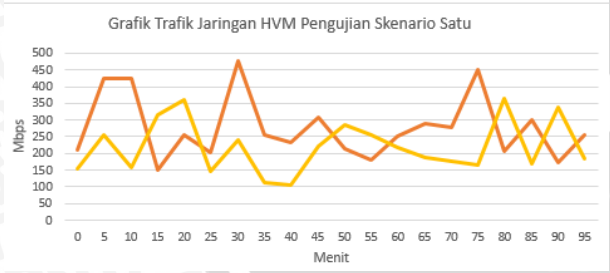
Hasil dari pengujian skenario satu, dua, dan tiga yang telah dilakukan sebelumnya menunjukan bahwa sistem yang telah dibangun

dapat berjalan dengan baik secara fungsional. Sistem dapat berjalan lancar baik yaitu karena dapat melakukan *live migration* antar VM dan dapat melalukan proses *swapping* sesuai dengan ambang batas yang ditentukan.

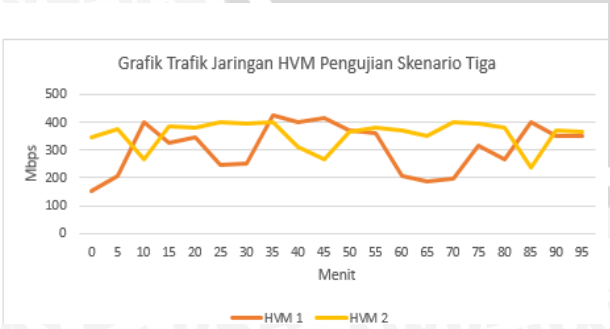
Berdasarkan hasil pengujian sistem ini juga memperlihatkan hasil bahwa beban *bandwidth* HVM 1 selalu dalam keadaan stabil yaitu beban *bandwidth* HVM 1 tidak melebihi angka 400 Mbits/sec. Hal ini membuktikan bahwa sistem mampu menyeimbangkan beban *bandwidth* serta penerapan *live migration* mesin virtual berjalan secara optimal. Perubahan-perubahan beban pada HVM 1 dan HVM 2 saat pengujian skenario satu, skenario dua, dan skenario tiga dapat dilihat pada grafik berikut ini.



Gambar 5.1 Trafik Jaringan HVM Pengujian Skenario 1



Gambar 5.2 Trafik Jaringan HVM Pengujian Skenario Dua



Gambar 5. 3 Trafik Jaringan HVM Pengujian Skenario Tiga

6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitan yang telah dilakukan dan telah dibahas pada bab sebelumnya, maka dapat ditarik kesimpulan sebagai berikut:

1. Pada sistem yang dibangun, dari hasil pengujian proses pengimplementasian *distributed bartering algorithm* dapat berjalan dengan baik. *Distributed bartering algorithm* dapat menjadi salah satu solusi untuk mengurangi kemacetan dan *overhead* jaringan. Melihat hasil pengujian, bahwa rata-rata dari tiap skenario tidak melebihi ambang batas yaitu 400 Mbits/sec sehingga hal ini menunjukan bahwa metode yang diusulkan mampu menyeimbangkan beban trafik data antar HVM dan meminimalisir terjadi kemacetan dan *overhead* jaringan.
2. Metode negosiasi dan *swapping* antar HVM juga dapat diterapkan dengan baik. Proses *swapping* mampu menyeimbangkan kelebihan beban yang terjadi antar HVM. Dengan adanya proses *swapping* ini, trafik jaringan yang ada antar HVM dapat lebih stabil sehingga proses *swapping* dapat digunakan sebagai salah satu metode selain *live migration*.

6.1 Saran

Diperlukannya penelitian yang lebih lanjut terkait *distributed bartering algorithm* berdasarkan kemacetan jaringan, CPU *Load* maupun Memory *Load* sehingga hasilnya nanti dapat dijadikan perbandingan dari penelitian ini

DAFTAR PUSTAKA

Aditya Angga Agung. 2012. "*Optimasi Penggunaan Sumber Daya Komputasi Di Lingkungan Cloud Computing*". Malang: PTIIK Universitas Brawijaya

Alex Budiyanto, 2012. *E-Book Pengantar Cloud Computing*, Cloud Indonesia, hal 3.

Al-Kiswany, Samer, dkk, 2011."*VMFlock: Mesin virtual Co-Migration for the Cloud"*. Proceedings of the 20th international symposium on High performance distributed computing, New York.

Anja Strunk and Waltenegus Dargie, 2011. "*Does Live migration of Virtual Machines Cost Energy?*". Technical University of Dresden, Dresden, Germany.

Furth, Borko, &Escalante, A.,2010. "*Handbook of Cloud Computing*", Springer, hal v.

J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra, 2010 "Starling: Minimizing Communication *Overhead* in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration," 39th International Conference on Parallel Processing

Kai hwang, Geoffrey C.Fox, Jack J. Dongarra, 2012. "*Distributed and Cloud Computing"*, Morgan Kaufmann, USA.

Karen Scarfone, Murugiah Suoppaya, Paul Hoffman, 2011. "*Guide to Security for Full Virtualization Technologies.*

Khoa Huynh, Ph.D, Andrew Theurer & Stefan Hajnoczi, 2013."*KVM Virtualized I/O Performance*".

Lublin, Uri, & Ligouri, Antony, 2007. "*KVM Live migration*" Qumranet & IBM.

Microsoft-TechNet, 2012. *Hypervisor* [image online] Tersedia di: https://blogs.technet.microsoft.com/chenley/2011/02/09/hypervisors/ [Diakses 20 Desember 2015]

M. R. Garey and D. S Johnson, 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman.

Myint, May T. H & Thandar Thein, 2010. "*Availability Improvement in Virtualized Multiple Servers with Software Rejuvenation and Virtualization", University of Computer Studies, Yangon, Myanmar.*

Nilesh Pachorkar, Rajesh Ingle, 2013. "*Multi-dimensional Affinity Aware VM Placement Algorithm in Cloud Computing*", International Journal of Advanced Computer Research.

Sanjay P. Ahuja, Suganya Sridharan, 2012. "*Performance Evaluation of Hypervisors for Cloud Computing",* International Journal of Cloud Applications and Computing, USA.

Sarna, David E.Y., 2011, Implementing and Developing Cloud Computing Applications. Taylor and Francis Group, LLC. Boca Raton.

Shirley Radack, 2012. "*Cloud Computing: A Review of Features, Benefits, and Risk, and Recommendations for Secure, Efficient Implementations*", National Institute of Standarts and Technology.

Suryatama, Indra, 2012. "*Membangun Infrastruktur Komputasi Awan Privat Mengunakan Ubuntu Enterprise Cloud*". Andi: Yogyakarta.

Stage, Alexander & Thomas Setzer, 2009. "*Network-aware migration control and scheduling of differentiated mesin virtual workloads".*Technische Universitat Munchen: Germany.

Timo Hirt, 2011. "KVM - The kernel-based virtual Machine".