KINETIK, Vol.1, No.2, Agustus 2016, Hal. 79-90
ISSN : 2503-2259,
E-ISSN : 2503-2267

Pemilihan dan Imigrasi Virtual Machine Menggunakan Multi-Criteria Decision Making untuk Peningkatan Kinerja Layanan pada Cloud Computing

Abdullah Fadil[*1], Waskitho Wibisono[2]
[1] Universitas Islam Negeri Sunan Ampel, Surabaya
[2] Institut Teknologi Sepuluh Nopember, Surabaya
fadil@uinsby.ac.id[*1], waswib@if.its.ac.id[2]

Abstrak

Komputasi awan atau cloud computing merupakan lingkungan yang heterogen dan terdistribusi, tersusun atas gugusan jaringan server dengan berbagai kapasitas sumber daya komputasi yang berbeda-beda guna menopang model layanan yang ada di atasnya. Virtual Machine (VM) dijadikan sebagai representasi dari ketersediaan sumber daya komputasi dinamis yang dapat dialokasikan dan direalokasikan sesuai dengan permintaan. Mekanisme live migration VM di antara server fisik yang terdapat di dalam data center Cloud digunakan untuk mencapai konsolidasi dan memaksimalkan utilisasi VM. Pada prosedur konsoidasi VM, pemilihan dan penempatan VM sering kali menggunakan kriteria tunggal dan statis. Dalam penelitian ini diusulkan pemilihan dan penempatan VM menggunakan multi-criteria decision making (MCDM) pada prosedur konsolidasi VM dinamis di lingkungan Cloud data center guna meningkatkan layanan cloud computing. Pendekatan praktis digunakan dalam mengembangkan lingkungan cloud computing berbasis OpenStack Cloud dengan mengintegrasikan VM selection dan VM Placement pada prosedur konsolidasi VM menggunakan OpenStack-Neat. Hasil penelitian menunjukkan bahwa metode pemilihan dan penempatan VM melalui live migration mampu menggantikan kerugian yang disebabkan oleh down-times sebesar 11,994 detik dari waktu responnya. Peningkatan response times terjadi sebesar 6 ms ketika terjadi proses live migration VM dari host asal ke host tujuan. Response times rata-rata setiap VM yang tersebar pada compute node setelah terjadi proses live migration sebesar 67 ms yang menunjukkan keseimbangan beban pada sistem cloud computing.

Kata Kunci: *Cloud computing, Virtual Machine migration, load balancing, dynamic VM consolidation*

Abstract

Cloud computing is a heterogeneous and well-distributed circumstances, consists of server network clusters in various capacity of computing resources to sustain the service mode on it. Virtual Machine (VM) is used as the representative of dynamic computing resources availability, which can be allocated and reallocated according to the demands. VM live migration mechanism, among the physical server in Cloud data center, is used to afford the consolidation and optimizes the VM utilization. In VM consolidation procedure, the VM selection and placement is using single and static criteria. This research suggests to select and place the VM using multi-criteria decision making (MCDM) for dynamic VM consolidation procedure in Cloud data center to improve the cloud computing performance. Practical approach is used to develop the OpenStack Cloud-based by integrating the VM selection and VM placement in VM consolidation procedure using OpenStack-Neat. The findings show that the VM selection and placement via live migration can substitute the waste caused by down-times for 11,994 second per response. Response-times is increasing about 6ms during VM live migration process from the originating host to the destination host. The average response-times for each VM in compute node is 67ms which indicates the load balance in cloud computing system.

Keywords: *Cloud computing, virtual machine migration, workload balancing, dynamic consolidation*

Makalah dikirim 30 Juni 2016; Revisi 30 Juli 2016; Diterima 1 Agustus 2016

1. Pendahuluan

Komputasi awan atau *cloud computing* merupakan model layanan yang memberikan kemudahan kepada *end-users* dan *broker*. Layanan-layanan tersebut dikemas dalam pangkalan sumber daya komputasi yang dapat diakses melalui jaringan internet sesuai dengan permintaan. Sumber daya tersebut disajikan kepada pelanggan secara swalayan melalui usaha pengelolaan dan interaksi dengan penyedia layanan yang minimal [1].

Untuk mengatasi masalah permintaan dan ketersedian pada layanan *cloud* yang fluktuatif, maka data *center cloud* harus dibuat elastis. Teknologi yang memungkinkan untuk merealisasikan hal tersebut adalah virtualisasi. *Virtual Machine* (VM) merupakan representasi dari ketersediaan sumber daya komputasi dinamis yang dapat dialokasikan dan direalokasikan sesuai dengan permintaan.

Usaha untuk menyeimbangkan beban kerja dan meningkatkan utilisasi VM di dalam server fisik yang ada di data *center cloud* telah banyak dilakukan. Selain penyeimbangkan beban kerja melalui mekanisme alokasi distribusi beban di antara VM yang terdapat di dalam server fisik pada *data center* di lingkungan *cloud* melalui pendekatan penjadwalan [2], prosedur lain yang dapat dilakukan adalah memindahkan VM dari satu server fisik ke server fisik lainnya menggunakan migrasi VM secara langsung (*live* VM *migration*) ataupun tidak langsung *(offline* VM *migration)*. *Live* VM *migration* dilakukan diantara server fisik yang terdapat di dalam data *center cloud* guna mencapai konsolidasi dan memaksimalkan utilisasi VM itu sendiri.

Strategi konsolidasi VM dapat dilakukan dengan mempertimbangkan kapan, yang mana dan kemana VM harus dimigrasikan. Secara formal [3] menguraikannya ke dalam sub pengambilan keputusan, yaitu *host overloading detection, host underloading detection,* VM *selection* dan VM *Placement*. *Host overloading detection* dan *host underloading detection* digunakan untuk menentukan kapan suatu *host* dianggap mengalami *overloaded* ataupun *underloaded*. Dalam kondisi ini, satu atau lebih VM harus bermigrasi dari *host* tersebut. VM *selection* mengidentifikasi dan menentukan VM yang mana harus bermigrasi dari suatu *host* yang *overloaded* atau *underloaded*. VM *placement* akan memetakan kemana VM terpilih harus dimigrasikan pada suatu *host* relatif *underloaded* dengan *host* lainnya.

Lingkungan *cloud computing* yang dinamis dan terdistribusi mengharuskan prosedur dan strategi pengambilan keputusan di dalam konsolidasi VM harus dibuat dinamis juga. Sehingga dalam penelitian ini diusulkan optimasi kebijakan pemilihan dan penempatan VM pada prosedur konsolidasi VM pada lingkungan *cloud* data *center* dengan mempertimbangkan *hosting* jenis *service* aplikasi tertentu pada *instance* VM yang membutuhkan level sumber daya komputasi berbeda-beda. Akumulasi penggunaan sumber daya komputasi pada VM yang tidak seimbang dalam server fisik dapat direduksi dengan menggunakan *live* VM *migration* guna mencapai *workload balancing*. Pendekatan praktis akan digunakan dalam mengembangkan lingkungan *cloud computing* berbasis *OpenStack Cloud* dengan mengintegrasikan VM *selection* dan VM *Placement* pada prosedur konsolidasi VM menggunakan *OpenStack-Neat*.

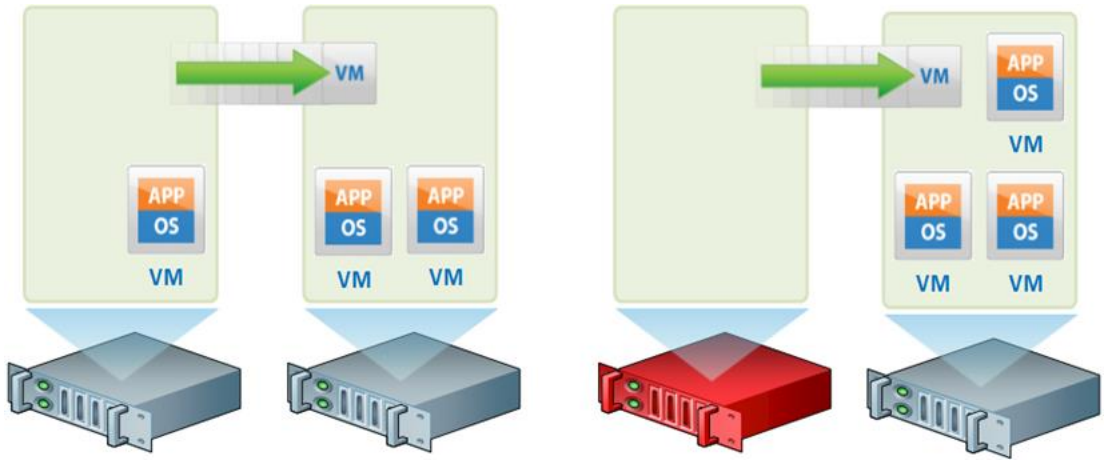
1.1 Komputasi Awan (Cloud Computing)

Komputasi awan atau *cloud computing* merupakan suatu model *client-server*, dimana sumber daya (*resources)* seperti *server, storage, network* dan *software* dapat dipandang sebagai layanan yang dapat diakses oleh pengguna secara *remote* dan setiap saat [4]. Pengguna dapat menikmati berbagai layanan yang disediakan oleh *provider cloud computing* tanpa perlu terlalu banyak meminta bantuan teknis atau *support* dari pihak *provider*. Model layanan *cloud computing* dapat diilustrasikan dalam Gambar 1 [1].

1.2 Migrasi VM

Migrasi VM dapat digunakan untuk konsolidasi VM secara dinamis berdasarkan pada analisis data historis dari penggunaan sumber daya oleh VM seperti yang ditunjukkan pada Gambar 1.

Masalahnya dibagi menjadi empat bagian: (1) menentukan kapan *host* dianggap *underloaded* yang mengarah kepada kebutuhan untuk migrasi semua VM dari *host* ini dan mengalihkan *host* ke dalam *mode sleep*, (2) menentukan kapan *host* dianggap *overloaded* yang mengarah kepada kebutuhan migrasi dari satu atau lebih VM dari *host* ini untuk mengurangi beban, (3) memilih VM yang harus bermigrasi dari *host* yang kelebihan beban, dan (4) menemukan penempatan baru dari VM yang dipilih untuk migrasi dari *host* yang *overloaded* [5].



Gambar 1. Migrasi VM untuk Konsolidasi

1.3 Multi-Criteria Decision Making (MCDM)

Multi-Criteria Decision Making (MCDM) adalah suatu metode pengambilan keputusan untuk menetapkan alternatif terbaik dari sejumlah alternatif berdasarkan beberapa kriteria tertentu. Kriteria biasanya berupa ukuran-ukuran,aturan-aturan, atau standar yang digunakan dalam pengambilan keputusan. Secara umum dapat dikatakan bahwa MCDM menyeleksi alternatif terbaik dari sejumlah alternatif [6].

Dalam [6] menyebutkan terdapat beberapa fitur umum yang digunakan dalam MCDM, yaitu:

- Alternatif, alternatif adalah objek-objek yang berbeda dan memiliki kesempatan sama untuk dipilih oleh pengambil keputusan.
- Atribut, atribut sering juga disebut sebagai kriteria keputusan.
- Konflik antar kriteria, bebrapa kriteria biasanya mempunyai konflik antara satu dengan yang lainnya, misalnya kriteria keuntungan akan mengalami konflik dengan kriteria biaya.
- Bobot keputusan, bobot keputusan manunjukkan kepentingan relatif dari setiap kriteria, W = ($w1$, $w2$, $w3$,…, wn).
- Matriks keputusan, suatu matriks keputusan X yang berukuran m x n, berisi elemen-elemen xij yang merepesentasikan rating dari alternatif $Ai;i$=1,2,3,…,m terhadap kriteria $Cj;j$=1,2,3,…,n.

Technique for Order Preference by Similiarity to Ideal Solution atau TOPSIS merupakan salah satu pendekatan yang popular di dalam MCDM. Metode tersebut menggunakan pemeringkatan pada setiap alternatif dalam sekumpulan kriteria. Misalnya, permasalahan MCDM dengan alternatif m, A_1, …, A_m dan n merupakan kriteria atau atribut keputusan C_1, …, C_n. setiap alternatif yang dievaluasi berdasarkan kriteria atau atribut n. Semua nilai yang diberikan sebagai alternatif dengan mempertimbangkan pada setiap kriteria dalam bentuk *decision matrix* yang dirumuskan dengan $X = (x_{ij})_{m\,x\,n}$. Misalkan $W = (w_1 \ldots w_n)$ merupakan *vector* bobot yang relatif untuk suatu kriteria, nilai kepuasan dapat ditentukan melalui $\sum_{i=1}^{n} w_i$=1, kemudian metode dapat dirumuskan di Persamaan sebagai berikut:

1) Menghitung *decision matrix* (D) sebagai:

$$D = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ldots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \quad (1)$$

2) Menghitung normalisasi *decision matrix* atau R matriks. Nilai normalisasi r_{ij} dihitung sebagai:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \; dengan\; i = 1, \ldots, m \; dan \; j = 1, \ldots, n \quad (2)$$

3) Menghitung kriteria matriks pembobotan sebagai:

$$W = \begin{bmatrix} w_{11} & \cdots & 0 \\ \vdots & w_2 \dots & \vdots \\ 0 & \cdots & w_n \end{bmatrix} \quad (3)$$

4) Menghitung normaliasi pembobotan *decision matrix*. Normalisasi nilai pembobotan v_{ij} dihitung berdasarkan:

$v_{ij} = w_i r_{ij} = W\ x\ R, i = 1, \dots m, dan\ j = 1, \dots, n$ =, dimana w_i adalah bobot dari kriteria j dan $\sum_{i=1}^{n} w_i$=1, (4)

5) Menentukan solusi ideal positif dan negatif, A^+ dan A^- yang diharapkan.

$$A^+ = \left\{\left(\max v_{ij} | j \in J\right), \left(\min v_{ij} | j \in j'\right) | i = 1,2, \dots, m\right\} = \left\{v_1^+, v_2^+, \dots, v_j^+, \dots, v_n^+\right\}, \quad (5)$$

$$A^- = \left\{\left(\min v_{ij} | j \in J\right), \left(\max v_{ij} | j \in j'\right) | i = 1,2, \dots, m\right\} = \left\{v_1^-, v_2^-, \dots, v_j^-, \dots, v_n^-\right\}, \quad (6)$$

Dimana J berkaitan dengana kriteria benefit, dan j' berkaitandengan *criteria cost*.

6) Menghitung pengukuran pemisah menggunakan *Euclidean distance* n-dimensi. *Distance* masing-masing alternatif dari solusi ideal rumuskan:

$$d_i^+ = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^+\right)^2}\ \ i = 1, \dots, m\ , \quad (7)$$

Untuk *distance* dari solusi ideal negatif dirumuskan:

$$d_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2}\ \ i = 1, \dots, m\ , \quad (8)$$

7) Menghitung kedekatan relatif dengan solusi ideal. Kedekatan relatif dari alternatif A_i dengan A^+ di definisikan dengan :

$$RC_i = \frac{d_i^-}{d_i^+ + d_i^-}\ dengan\ i = 1, \dots, m\ , \quad (9)$$

Karena $d_i^- \geq 0$ dan $d_i^+ \geq 0$ maka $RC_i \in [0,1]$

8) Pemeringkatan alternatif berdasarkan kedekatan relatif dengan solusi ideal. Serta RC_i yang lebih rendah merupakan alternatif A_i yang lebih didalam sistem.

1.4 Cloud Computing Berbasis OpenStack

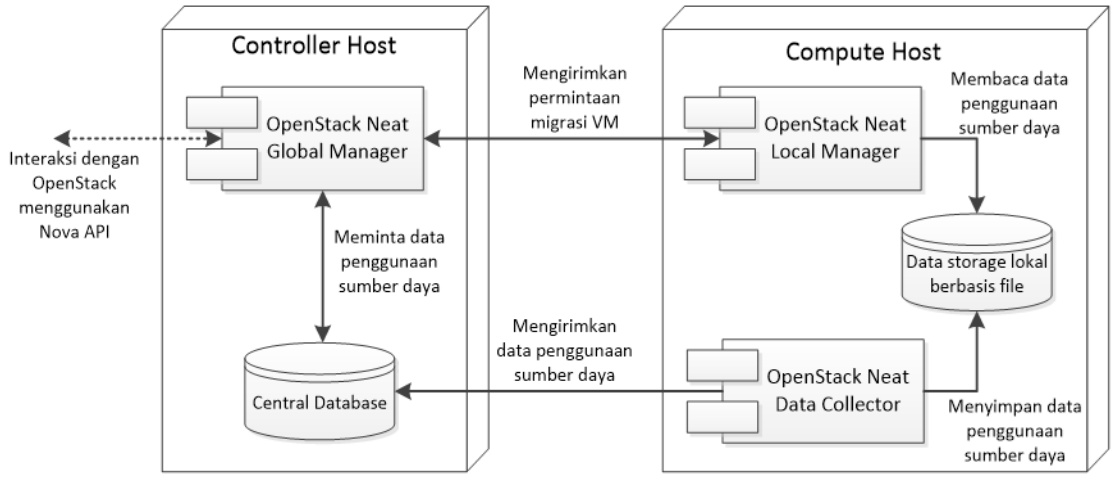
OpenStack menjadi *software* standar untuk infrastruktur *Cloud* yang memenuhi kesederhanaan dan skalabilitas [7]. Beberapa komponen inti dari *OpenStack* adalah:

- *Image (Glance)* menyediakan katalog dan repositori untuk *virtual disk image*. *Disk image* ini sebagian besar digunakan dalam *OpenStack Compute*. Layanan ini secara teknis opsional, berapapun ukuran dari *Cloud* memerlukan *glance* tersebut.
- *Compute (Nova)* menyediakan *server virtual* sesuai dengan permintaan. *Rackspace* dan HP menyediakan layanan komputasi komersial dibangun di Nova dan digunakan secara internal di perusahaan-perusahaan seperti Mercado Libre dan NASA.
- *Dashboard (Horizon)* menyediakan *user-interface* modular berbasis *web* untuk semua layanan *OpenStack*. Dengan *web* GUI ini, dapat dilakukan sebagian besar operasional *Cloud* seperti menyajikan *instance*, menentukan alamat IP dan pengaturan kontrol akses.
- *Identity (Keystone)* menyediakan otentikasi dan otorisasi untuk semua layanan *OpenStack*. Hal ini juga menyediakan katalog layanan pada *OpenStack Cloud* tertentu.

1.5 Konsolidasi VM Pada OpenStack Neat

OpenStack-Neat merupakan perluasan pada *OpenStack Cloud* mengimplementasikan konsolidasi dinamis *Virtual Machine* (VM) menggunakan *live migration*. Tujuan utama konsolidasi VM dinamis adalah untuk meningkatkan pemanfaatan sumber daya server fisik dan mengurangi konsumsi energi dengan mengalokasikan kembali VM menggunakan *live migration* sesuai dengan permintaan sumber daya *real-time* dan mengalihkan *host* yang *idle* ke-*mode sleep* [8].

OpenStack-Neat terdiri dari sejumlah komponen dan unit penyimpan data, beberapa diantaranya digunakan pada *compute host*, dan beberapa pada *controller host* Gambar 2.



Gambar 2. Framework OpenStack-Neat [9]

Seperti ditunjukkan dalam Gambar 2, sistem ini terdiri dari tiga komponen utama:

- Global *manager*-komponen yang digunakan pada *management host* untuk membuat keputusan manajemen global.
- *Local manager*-komponen yang digunakan pada setiap *compute host* untuk membuat keputusan lokal.
- Data *collector*-komponen yang digunakan pada setiap *compute host* dan bertanggung jawab untuk mengumpulkan data tentang penggunaan sumber daya oleh VM, serta menyimpan data tersebut secara lokal dan mengirimkan data ke *central database*.

2. Metode Penelitian

2.1 Desain dan Implementasi

Pada penelitian ini sistem konsolidasi VM yang akan dikembangkan adalah pada tahapan pemilihan dan penempatan VM melalui mekanisme *live migration*. Kandidat VM dan *host* yang terpilih didasarkan pada multi kriteria parameter dari sumber daya VM dan *host* yang digunakan.

(1) Sumber daya fisik *(Physical Machine)*

Sumber daya fisik merupakan unit perangkat keras untuk pemrosesan, penyimpanan dan komunikasi yang terdapat pada *Physical Machine (PM)* atau *host* berpengaruh dalam pengambilan keputusan prosedur konsolidasi VM, khususnya penentuan *host overloading, underloading detection,* dan VM *placement*. Dalam penelitian ini, sumber daya fisik yang digunakan adalah *CPU, memory, network traffic,* dan *disk storage IO* yang melekat pada *physical machine* atau *host*.

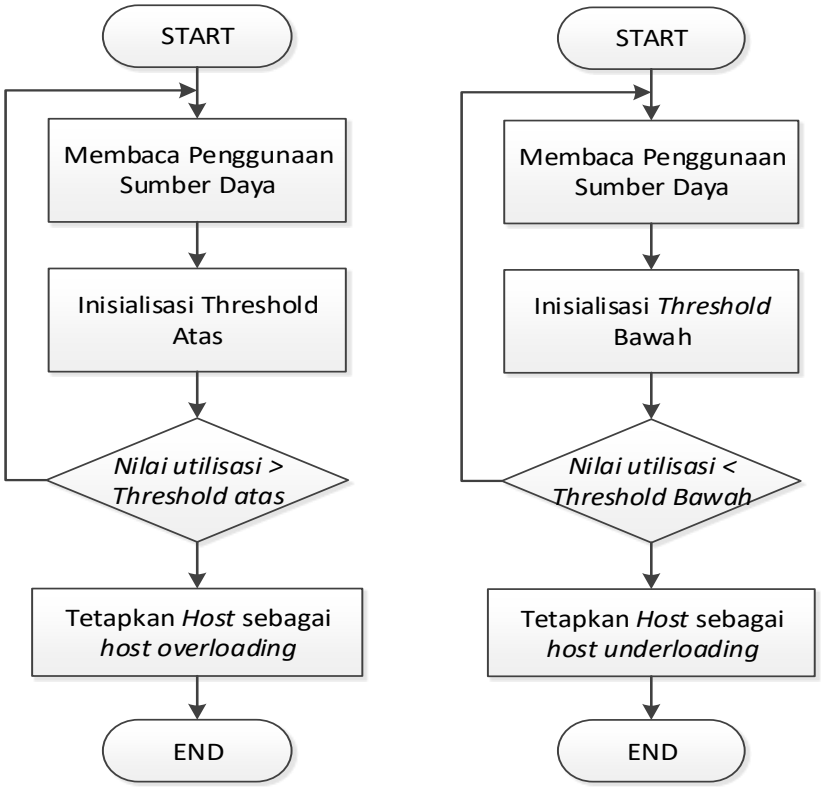
(2) Sumber daya virtual *(Virtual Machine)*

Sumber daya *virtual* merupakan unit *logical* untuk diproses, penyimpanan, dan komunikasi yang terdapat pada *Virtual Machine (*VM*)* berpengaruh dalam pengambilan keputusan pada prosedur konsolidasi VM, khususnya VM *selection*. Dalam penelitian ini, sumber daya *virtual* yang digunakan adalah *CPU, memory, network io,* dan *disk storage IO* yang melekat pada *Virtual Machine.*

(3) Heterogenitas sumber daya

Keanekaragaman sumber daya pada *Virtual Machine* berkaitan dengan jenis aplikasi atau *service* yang dijalankan pada *guest os* dalam suatu *Virtual Machine*. Variasi dari jenis

aplikasi ini, akan menghasilkan *workload* dengan tingkat konsumsi sumber daya yang berbeda-beda. Dalam penelitian ini, tingkat konsumsi sumber daya yang digunakan oleh aplikasi dibedakan menjadi CPU-*intensive, memory-intensive, network io-intensive* dan *disk storage IO-intensive*. Aplikasi CPU-*intensive* akan relatif mengkonsumsi CPU lebih besar dibandingkan dengan sumber daya lainnya.



Gambar 3. Flowchart Deteksi Host yang (a) Overloading dan (b) Underloading

2.2 Konsolidasi VM

Setelah definisi paramater awal ditetapkan, prosedur konsolidasi VM secara dinamis, seperti pada Gambar 4 yang dijelaskan sebagai berikut:

(1) *Host overloading detection*

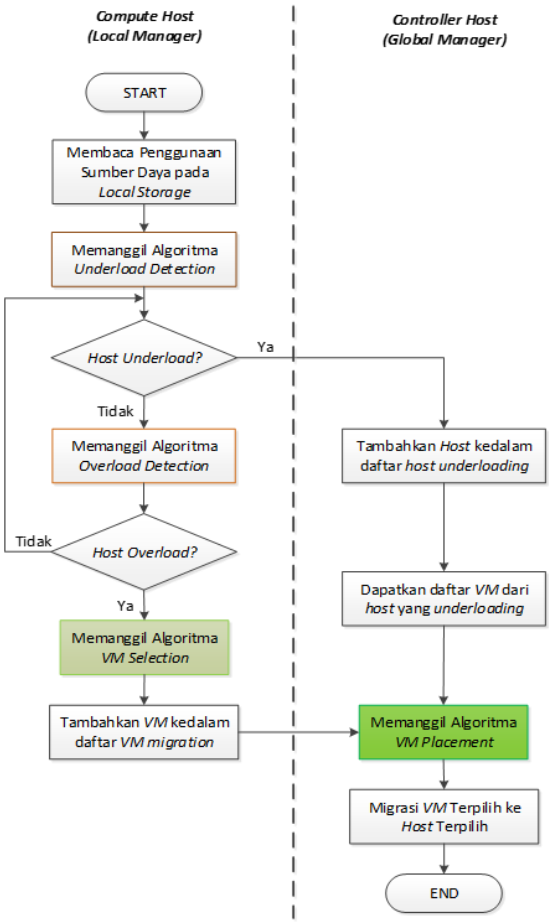
Ditentukan berdasarkan perbandingan utilisasi dengan *workload* yang beraneka ragam dari sumber daya komputasi (*CPU-intensive*, *Memory-intensive*, *Disk storage IO-intensive,* dan *Network-intensive*) dibandingkan dengan *threshold* batas atas yang akan ditetapkan dari utilisasi masing-masing sumber daya tersebut dalam percobaan. *Flowchart* deteksi *host* yang *overloading* seperti yang ditunjukkan pada Gambar 3(a). Keputusan *host* yang *overloading* ditentukan berdasarkan intensitas penggunaan pada masing-masing sumber daya yang melebihi *threshold* batas atas yang telah ditetapkan.

(2) *Host underloading* detection

Prosedur deteksi *host* yang *underloading* merupakan proses kebalikan dari deteksi *host* yang *overloading*. Deteksi *host* yang *underloading* ditentukan berdasarkan perbandingan utilisasi dengan *workload* yang beraneka ragam dari sumber daya komputasi (*CPU-intensive*, *Memory-intensive*, *Disk storage IO-intensive,* dan *Network-intensive*) dibandingkan dengan *threshold* batas bawah yang akan ditetapkan dari utilisasi masing-masing sumber daya tersebut dalam percobaan. *Flowchart* deteksi *host* yang *underloading* seperti yang ditunjukkan pada Gambar 3(b). Keputusan *host* yang *underloading* ditentukan berdasarkan intensitas penggunaan pada masing-masing sumber daya yang melebihi ambang batas bawah yang telah ditetapkan.

(3) VM Selection

Ketika *host* telah ditetapkan sebagai *host* yang *overloading* maka pemilihan VM dilakukan pada *host* tersebut menggunakan multi kriteria parameter seperti pada Tabel 1.



Gambar 4. Bagan Alir Konsolidasi VM secara Dinamis

Tabel 1. Kriteria pemilihan VM pada host yang overloading

Kriteria	Nama Atribut	Deskripsi
vC1	v*CPU*$_u$ %	Prosentase penggunaan CPU VM
vC2	vRAM$_u$ %	Prosentase penggunaan RAM VM
vC3	vNET$_u$ %	Prosentase penggunaan *Network* VM
vC4	vDISK$_u$ %	Prosentase penggunaan *disk storage* VM
vC5	vDISK$_{io}$ %	Prosentase aktivitas *disk storage* IO VM
vC6	v*CPU*c	Cadangan kapasitas CPU VM
vC7	vRAMc	Cadangan kapasitas RAM VM
vC8	vNETc	Cadangan kapasitas *net bandwidth* VM
vC9	vDISKc	Cadangan kapasistas *disk storage* VM
vC10	vAPPu	Konsumsi *resources* VM oleh APP

Masing-masing kriteria yang terdapat pada Tabel 1, dikombinasikan dengan masing-masing VM seperti pada Tabel 2.

Tabel 2. Kombinasi VM dengan setiap Kriteria pada VM

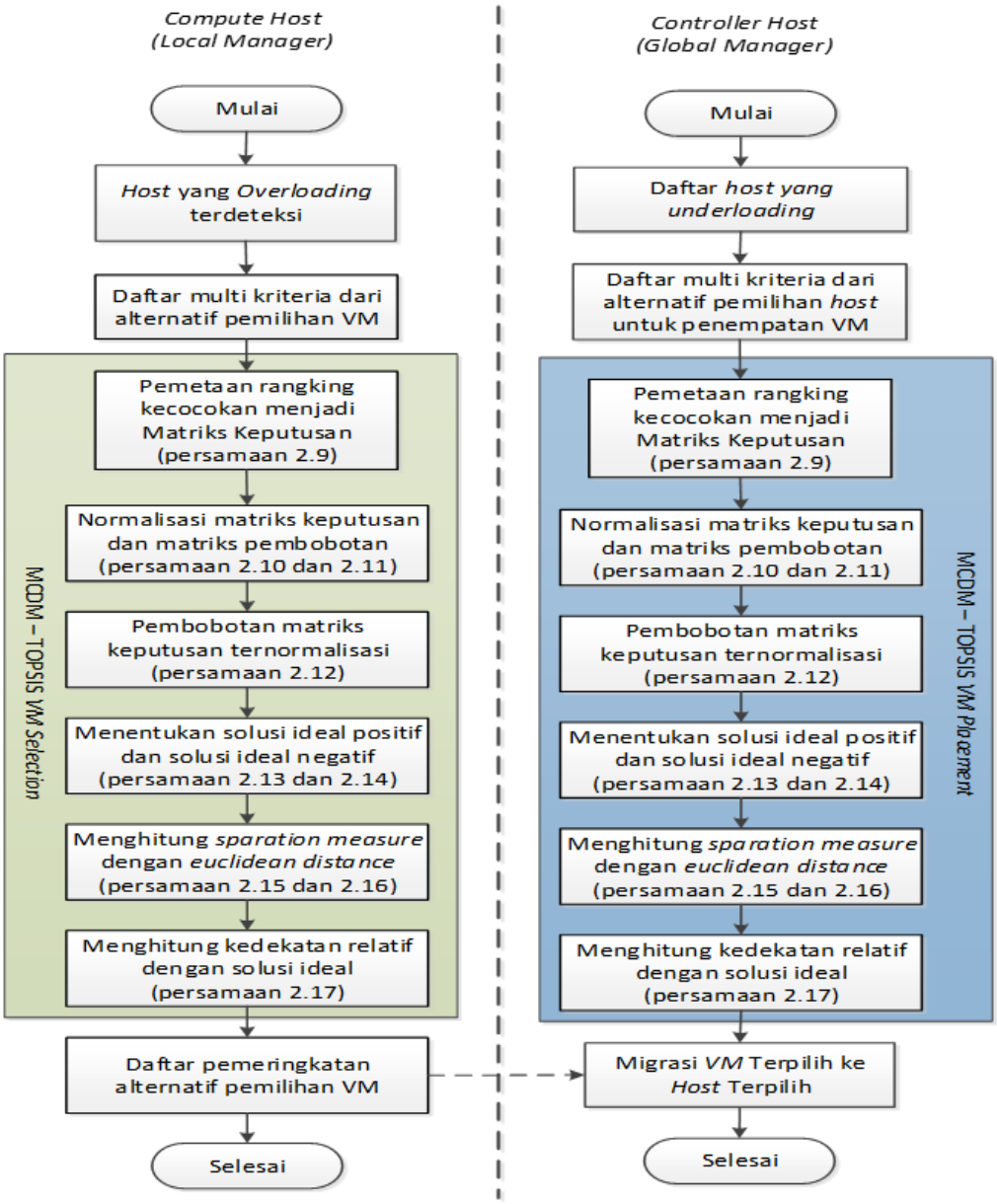
Alternatif	Atribut (kriteria)					
	vC1	vC2	vC3	vC4	vC5	vCn
VM1	V1C1	V1C2	V1C3	V1C4	V1C5	V1Cn
VM2	V2C1	V2C2	V2C3	V2C4	V2C5	V2Cn
VM3	V3C1	V3C2	V3C3	V3C4	V3C5	V3Cn
VM4	V4C1	V4C2	V4C3	V4C4	V4C5	V4Cn
VMm	VMC1	VMC2	VMC3	VMC4	VMC5	VMCn

Rangking kecocokan setiap alternatif pemilihan VM pada setiap kriteria, dinilai dengan 1 sampai 5 seperti pada Tabel 3. Nilai kecocokan dari setiap kriteria pada VM nantinya disesuaikan dengan jenis *workload* yang dampaknya berpengaruh pada tingkat kinerja layanan sistem *cloud computing*. Kriteria yang berdampak besar pada penurunan kinerja layanan akan diberi nilai bobot rendah dan sebaliknya. Nilai masing-masing kombinasi antara VM dan kriteria pada Tabel 2 dapat diperoleh dari tabel solusi ideal atau rangking kecocokan untuk membentuk matrik keputusan. Alternatif pemilihan VM akan ditentukan berdasarkan peringkat yang dihitung sesuai dengan langkah-langkah seperti pada Gambar 5(a).

Tabel 3. Tabel Transformasi Solusi Ideal

Rangking	Nilai
Rendah	1
Cukup rendah	2
Sedang	3
Cukup tinggi	4
Tinggi	5

(4) VM Placement

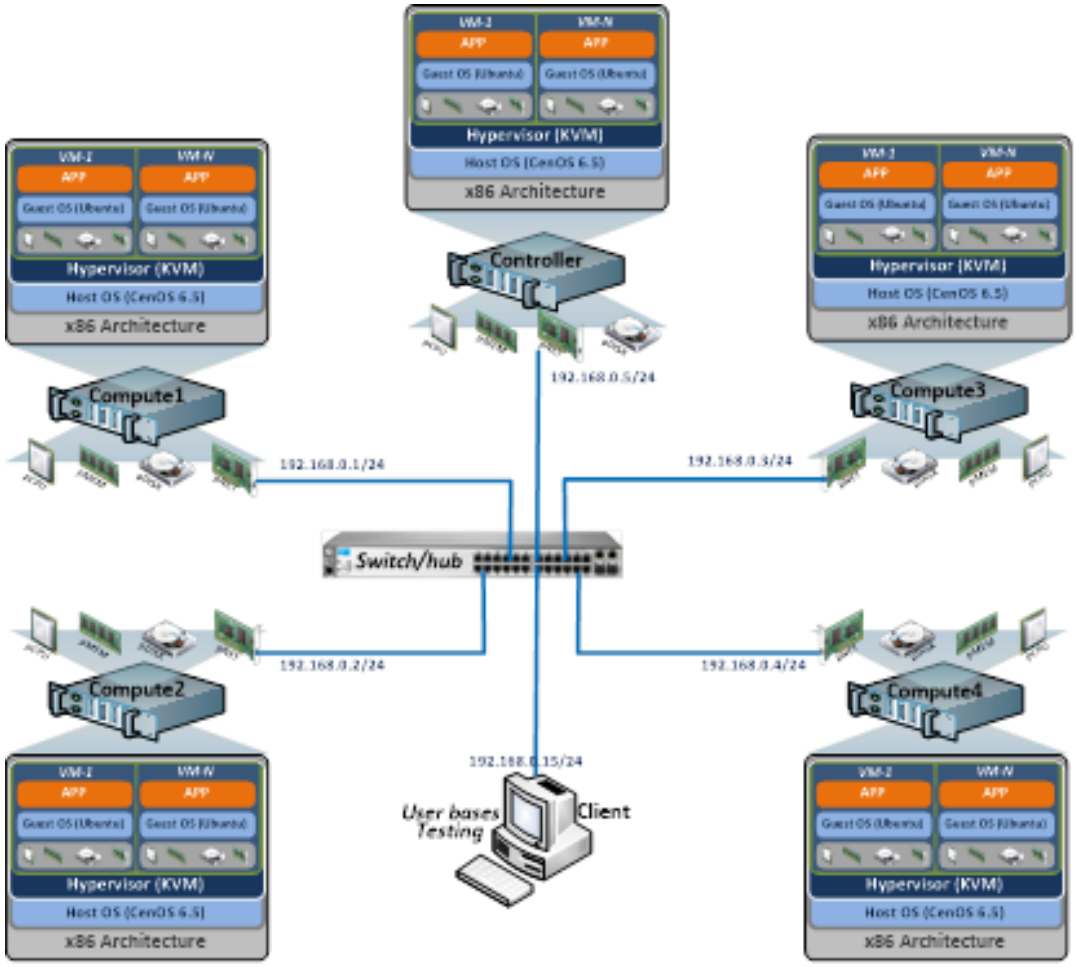


Gambar 5. Flowchart MCDM (a) VM Selection dan (b) VM Placement

Ketika VM sudah terpilih, sistem mencari *host* yang digunakan sebagai tempat untuk menerima migrasi dari VM terpilih. Pemilihan *host* ditentukan juga dengan menggunakan multi kriteria parameter yang sama dengan pemilihan VM sesuai dengan Tabel 1 dan Tabel 2. Rangking kecocokan setiap alternatif pemilihan *host* pada setiap kriteria ditentukan berdasarkan pada Tabel 3. Kriteria yang mengakibatkan penurunan kinerja layanan diberi nilai bobot rendah dan sebaliknya. Nilai setiap kombinasi antara *host* dan kriteria pada Tabel 2 dapat diperoleh dari tabel solusi ideal untuk membentuk matrik keptutusan. Alternatif pemilihan *host* untuk penempatan VM ditentukan berdasarkan pemeringkatan sesuai dengan tahapan pada Gambar 5(b).

2.3 Implementasi Sistem (Testbed)

Desain topologi jaringan *testbed* untuk implementasi sistem *cloud computing* dapat dilihat pada Gambar 6. Lingkungan ini dibangun secara riil diatas perangkat keras *(physical machine)* dengan *plaform Cloud* berbasis *OpenStack Cloud* dan *Framework OpenStack Neat* sebagai *consolidation* VM dinamis. *Physical machine* secara fungsional akan bertindak sebagai *controller* dan *compute host* pada lingkungan *cloud computing. Controller host* akan berperan dalam pengambilan keputusan *global* seperti migrasi dan penempatan VM. *Compute host* bertindak sebagai pengambil keputusan *local* seperti deteksi *host* yang *overloading* dan *underloading*. Spesifikasi perangkat *controller host dan compute host* serta *testing* yang digunakan dalam *testbed* adalah Intel i3-2330M dengan RAM 4GB dan HDD 500GB.



Gambar 6. Topologi Jaringan Testbed

3. Hasil Penelitian dan Pembahasan

Analisis performa *live migration* VM didasarkan pada *response times*, durasi migrasi dan *down times*. Pada VM akan diberikan beban mulai dari 1 *threads* hingga 35 *threads*. Setiap 7 detik ditambahkan 1 *users* selama periode *ramp-up* 30 detik dan ditahan hingga 600 detik

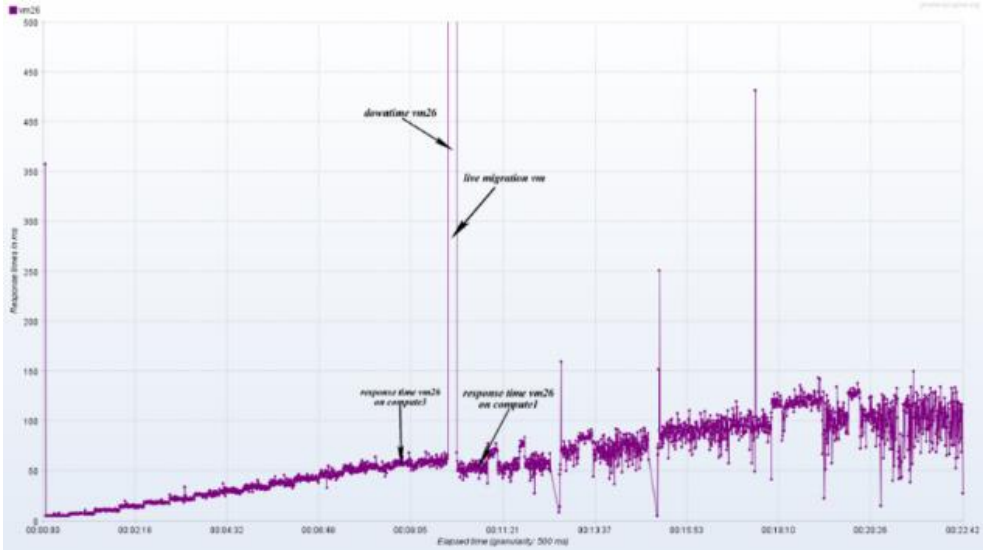
ketika mencapai 35 *users*. Hasilnya adalah pola pembebanan secara bertingkat membentuk tangga.

Pada uji coba ini juga digunakan metode yang diusulkan dan diimplementasikan ke dalam OpenStack-*Neat*. Konfigurasi yang digunakan pada *OpenStack-Neat* adalah seperti pada Tabel 4.

Dari hasil uji coba yang dilakukan seperti pada Gambar 7. Pada proses uji coba berdasarkan *log* yang ada, penggunaan *resources* berada di bawah ambang batas bawah penggunaan *resources*, hal ini disebabkan oleh penetapan ambang batas atas didasarkan pada hasil uji coba yaitu kenaikan 55% dari penggunaan *resources* dari 1 VM menjadi 7 VM pada compute3 masih belum mencapai kapasitas maksimum dari *resources* yang tersedia di *compute*3. Pada kondisi tersebut status yang dijalankan adalah *underload detection*. Pada kondisi *underload* maupun *overload detection* prosedur selanjutnya adalah pemilihan VM yang pada dasarnya merujuk pada algoritma pemilihan VM yang sama.

Tabel 4. Konfigurasi OpenStack-Neat

Modul	Keterangan
Data *collector*	Setiap 5 detik
Local manager	Setiap 5 detik
Batas Atas penggunaan *Resources*	55%
Batas Bawah penggunaan *Resource*	23%
Algoritma Pemilihan VM	TOPSIS
Algoritma Penempatan VM	TOPSIS



Gambar 7. Response Times pada Live Migration VM sesuai Metode

Pada pemilihan dan penempatan VM, algoritma yang digunakan adalah sama yaitu TOPSIS dengan memberikan bobot sama-sama penting pada setiap kriteria yang dipakai. Dari hasil uji coba di dapatkan VM26 sebagai kandidat yang dimigrasikan hal ini dikarenakan *resources* yang digunakan oleh VM yang menempati *compute*3 cenderung sama satu sama lainnya. Prosedur selanjutnya adalah penempatan VM26 di antara *pilihan compute*1*, compute*2 dan *compute*4. Algoritma penempatan VM memilih *compute*1 sebagai *host* untuk VM26 dengan *down time* pada proses *live migration* adalah 12 detik. Sedangkan pada *response times* terjadi peningkatan sebesar 6 ms dari 60 ms ketika menempati *compute*3 dan 54 Ms ketika migrasi ke *compute*1.

Dalam periode waktu tertentu setelah terjadi proses *live migration* VM di antara *compute node* yang tersedia, rata-rata *response times* nya adalah 67 ms yang menunjukkan terjadi *load balancing* diantara VM yang tersebar pada *compute node* seperti ditunjukkan pada Gambar 8.

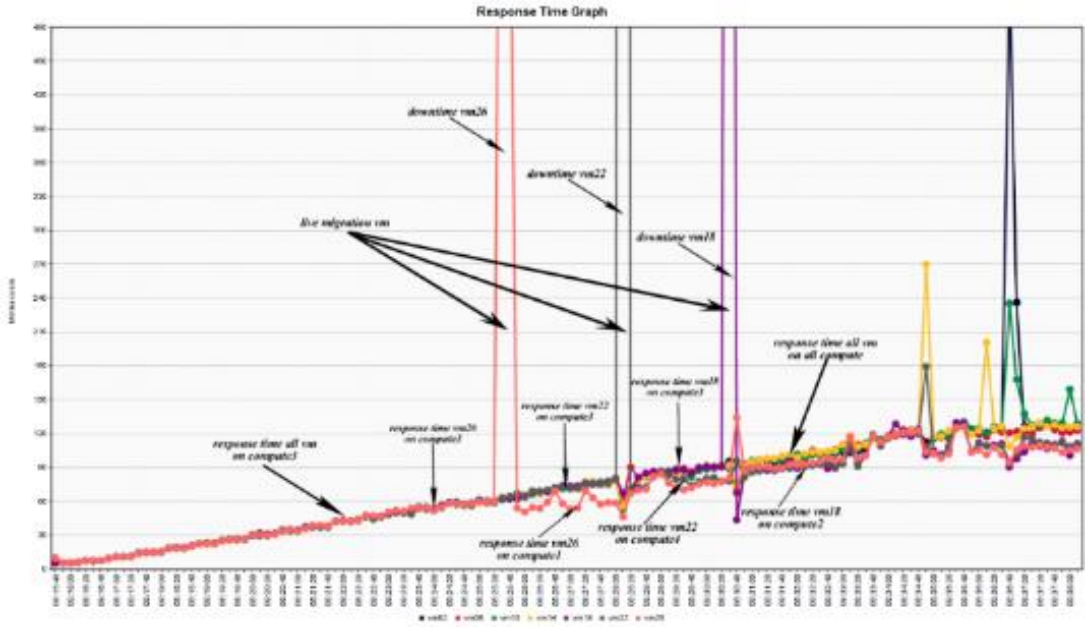
4. Kesimpulan

Performa dari layanan yang disajikan di dalam sistem *cloud computing* dapat dioptimalkan dengan *live migration* VM di antara *host* yang di tempatinya. Hal ini dibuktikan

dengan keberhasilan metode yang diusulkan melalui kebijakan pemilihan dan penempatan VM menggunakan MCDM pada prosedur konsolidasi VM yang mencapai *zero down-time*.

Berdasarkan hasil analisis uji coba, perbandingan antara tingkat *down-time* pada saat terjadi proses *live migration* VM adalah 12 s dengan *response times* 54 Ms ketika menempati *host* baru yang menunjukkan bahwa waktu *down-time* cenderung tidak berpengaruh terhadap *up-time* dari suatu layanan. Bahkan *response times* dari ketika menempati *host* asal sebelum migrasi dengan ketika menempati *host* tujuan mengalami peningkatan 6 Ms. Hal ini membuktikan bahwa pemilihan dan penempatan VM melalui MCDM berdasarkan paramater sumber daya yang digunakan *(CPU times, memory, disk-io, net-io)* mampu menyesuaikan beban di antara VM dan *host* yang di tempatinya dari *response time* yang dihasilkan.

Dari analisis terhadap VM lain yang tersebar di antara *compute node,* terlibat dalam periode waktu tertentu setelah mengalami proses *live migration* menunjukkan bahwa *response times* rata-ratanya 67 Ms. Hal ini menunjukkan mekanisme *live migration* VM pada konsolidasi VM mampu menyeimbangkan beban di antara *physical machine* yang ada pada sistem *cloud computing*.



Gambar 8. Kondisi setelah Proses Live Migration VM

Deteksi *overload* dan *underload* sebagai prosedur sebelum pemilihan dan penempatan VM melaui *live migration* VM pada sistem *cloud computing* dapat di tingkatkan dengan menggunakan metode yang sama seperti pada pemilihan dan penempatan VM guna yang menggunakan metode *thresholding*. Pada *OpenStack-Neat* dapat dilengkapi dengan metode-metode lain pada prosedur pemilihan dan penempatan VM untuk optimasi.

Referensi

[1] Mell, P., & Grance, T. (2011). “The NIST Definition of Cloud Computing.” National Institute of Standard and Technology

[2] Khiyaita, A., Zbakh, M., Bakkali, H., & Kettani, D. (2012). “Load Balancing Cloud Computing:state of art.” Network Security and Systems (JNS2), 106-109.

[3] Beloglazov, A., & Buyya, R. (2012). “Optimal Online Deterministic Algorithms and Adaptive Heuristics.” Concurrency and Computation: Practice and Expreience, 1397-1420.

[4] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwindsky, A., Zaharia, M. (2009). “Above the Clouds: A Berkeley View of Cloud Computing.” Electrical Engineering and Compute Sciences Unversity of California at Berkely.

[5] Beloglazov, A., & Buyaa, R. (2013). “Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers Under Quality of Service Constraints.” IEE Transactions on parallel and distributed system.

[6] Kusumadewi, S. (2006). “Fuzzy Multi-Attribute Decision Making (FMADM).” Yogyakarta: Graha Ilmu.

[7] OpenStack. (2013). OpenStack Cloud Administrator Guide. docs.openstack.org
[8] Beloglazov, A., & Buyya, R. (2013). OpenStack Neat: A Framework for Dynamic and Energy-Efficient Consolidation of Virtual Machines in OpenStack.
[9] Beloglazov, A. (2014, April). OpenStack Neat. Retrieved from http://www.openstack-neat.org