Strategi Efisiensi Energi dan Penyeimbangan Beban Kerja Layanan Cloud Computing Melalui Konsolidasi Mesin Virtual Dinamis

Abdullah Fadil[1*]

[1]Pusat Sistem Teknologi Informasi dan Pangkalan Data UIN Sunan Ampel, Surabaya
**Corresponding Author*
E-mail: ziyanza@uinsby.ac.id[*]

Abstrak

Arsitektur data center di dalam cloud computing merupakan lingkungan yang heterogen dan terdistribusi, tersusun atas gugusan jaringan physical machine (PM) atau server dengan berbagai kapasitas sumber daya komputasi yang berbeda-beda di dalam PMnya. Kondisi permintaan (demand) dan ketersediaan (supply) pada layanan cloud yang fluktuatif tersebut membuat data center cloud harus dibuat elastis. Virtual Machine (VM) merupakan representasi dari ketersediaan sumber daya komputasi dinamis yang dapat dialokasikan dan direlokasikan sesuai dengan permintaan. VM yang berada di dalam data center cloud dapat dipindahkan dari satu PM ke PM lainnya menggunakan migrasi VM secara langsung (live VM migration) ataupun tidak langsung (off-line VM migration). Lingkungan cloud computing yang dinamis dan terdistribusi mengharuskan strategi pengambilan keputusan di dalam konsolidasi VM harus dibuat sedinamis mungkin atau bahkan adaptif dengan mempertimbangkan heterogenitas sumber daya virtual sesuai dengan layanan cloud computing yang disajikan. Sehingga, dalam penelitian ini diusulkan efisiensi energi sekaligus menjaga kinerja layanan cloud computing melalui penyeimbangan beban kerja dengan teknik migrasi VM yang terdapat pada prosedur konsolidasi VM dinamis. Strategi pengambilan keputusan pada prosedur konsolidasi virtual machine dinamis yang diusulkan, dapat meningkatkan kinerja layanan cloud computing sekaligus beban kerja physical machine menjadi seimbang karena keputusan pemilihan VM dan penempatan VM pada physical machine dipilih secara optimal melalui MADM. Konsumsi energi dari physical machine juga dapat di hemat dengan mematikannya karena statusnya idle.

Kata Kunci: Pusat Data, Komputasi Awan, Konsolidasi *Virtual Machine*, Penyeimbangan Beban, Efisiensi Energi.

Abstract

Data center architecture in cloud computing is a heterogeneous and distributed environment, made up of a cluster of physical machine (PM) networks or servers with different capacities of computing resources within their PM. Demand (demand) and availability (supply) conditions on the fluctuating cloud services make the cloud data center must be elastic. Virtual Machine (VM) is a representation of the availability of dynamic computing resources that can be allocated and relocated according to demand. VMs inside the cloud data center can be moved from one PM to another using live VM migration or off-line VM migration. A dynamic and distributed cloud computing environment requires that the decision-making strategy in VM consolidation must be made as dynamic as possible or even adaptive by considering the heterogeneity of virtual resources in accordance with the cloud computing services presented. Thus, in this study it is proposed that energy efficiency while maintaining the performance of cloud computing services through workload balancing with VM migration techniques contained in the dynamic VM consolidation procedure. The decision making strategy in the proposed dynamic virtual machine consolidation procedure can improve the performance of cloud computing services while the physical

machine workload becomes balanced because the decision of VM selection and VM placement on the physical machine are optimally selected through MADM. Energy consumption from physical machines can also be saved by turning it off because the status is idle.

***Keywords**: Data Center, Cloud Computing, Virtual Machine Consolidation, Load Balancing, Energy-Efficient.*

1. PENDAHULUAN

Arsitektur data center di dalam cloud computing merupakan lingkungan yang heterogen dan terdistribusi, tersusun atas gugusan jaringan physical machine (PM) atau server dengan berbagai kapasitas sumber daya komputasi yang berbeda-beda di dalam PMnya. Apabila kapasitas sumber daya komputasi yang disediakan oleh data center cloud lebih banyak daripada permintaan maka penyedia layanan cloud menderita kerugian akibat overhead biaya operasional (perawatan dan konsumsi daya) dan proses yang terjadi di data center cloud akan mengalami kekurangan pemanfaatan (underutilization). Sebaliknya, jika kapasitas sumber daya komputasi yang disediakan oleh data center cloud mengalami kekurangan akibat banyaknya permintaan, maka penyedia layanan akan kehilangan pendapatan akibat ditinggalkan oleh pelanggan dan proses yang terjadi di data center cloud akan mengalami kelebihan pemakaian (overutilization)[1][2]. Manajemen sumber daya komputasi yang tidak efektif dapat menyebabkan beban kerja menumpuk (workload hotspot) hanya di satu sisi sumber daya komputasi saja, sehingga utilisasinya menjadi tidak seimbang.

Di masa pandemi covid-19 yang mengharuskan 90% aktivitas normal dilakukan dari rumah menjadikan layanan cloud computing di serbu oleh pelanggan. Kemudahan yang ditawarkan oleh layanan cloud computing melalui model provisioning berdasarkan pemakaian membuat permintaan pelanggan terus meningkat. Berbagai macam layanan aplikasi berpindah ke platform cloud dengan tingkat kebutuhan sumber daya komputasi beraneka ragam yang juga mengakibatkan beban kerjanya semakin meningkat. Layanan cloud menjadi kebutuhan dan pelanggan menjadi ketergantungan yang membuat faktor ketersediaan menjadi sangat penting. Karakteristik pelanggan yang memanfaatkan momen waktu tertentu di dalam mengakses layanan cloud membuat jumlah pertukaran data melalui jaringan dan tersimpan di storage mengalami peningkatan yang juga dapat menyumbangkan masalah bottleneck.

Kondisi permintaan (demand) dan ketersediaan (supply) pada layanan cloud yang fluktuatif tersebut membuat 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 direlokasikan sesuai dengan permintaan. Virtualisasi juga dapat digunakan untuk menempatkan VM semaksimal mungkin pada seminimal mungkin Physical Machine (PM). Secara operasional hal tersebut dapat mengurangi kompleksitas dan menekan biaya perawatan dengan meminimalisir PM yang berjalan (running).

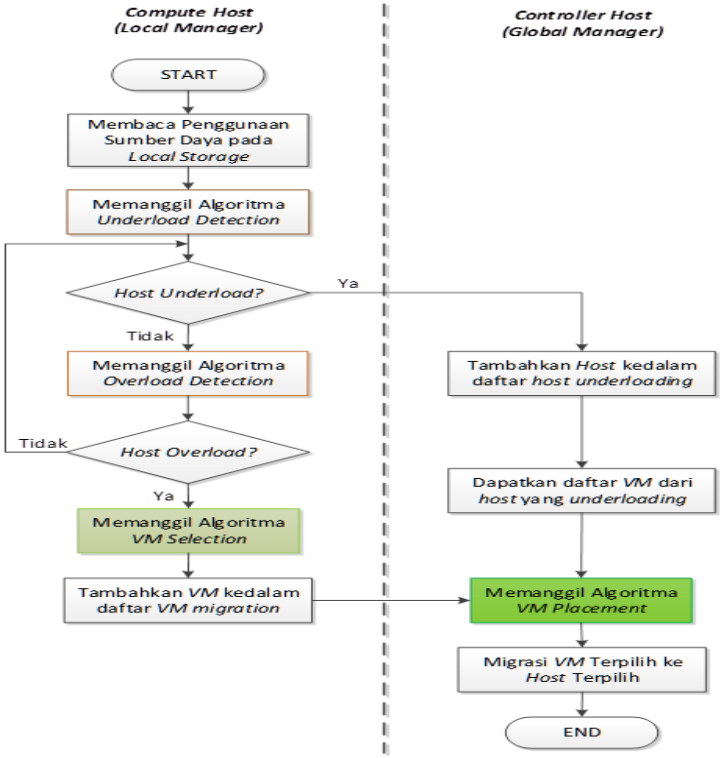
Permintaan layanan aplikasi yang beraneka ragam dapat menyebabkan VM dengan tingkat konsumsi sumber daya komputasi yang berbeda-beda misalnya CPU-intensive, Memory-intensive, Network-intensive dan Disk storage IO-intensive. Aplikasi pelanggan mungkin membutuhkan tingkat CPU yang tinggi dan memori yang rendah untuk menjalankanya ataupun kombinasi di antara sumber daya komputasi yang tersedia. PM pada data center cloud dapat memiliki sumber daya komputasi tertentu yang melimpah, tetapi sumber daya komputasi lainnya mengalami kekurangan. Akumulasi dari alokasi sumber daya komputasi pada VM yang tidak seimbang dapat menurunkan kinerja secara keseluruhan pada PM yang ada di data center cloud.

Hal tersebut berdampak pada beban kerja VM di dalam PM data center cloud menjadi tidak seimbang (load imbalance).

VM yang berada di dalam data center cloud dapat dipindahkan dari satu PM ke PM lainnya menggunakan migrasi VM secara langsung (live VM migration) ataupun tidak langsung (off-line VM migration). Live VM migration dilakukan di antara PM yang terdapat di dalam data center cloud ketika PM tersebut dalam keadaan hidup. Live VM migration dapat digunakan sebagai strategi konsolidasi untuk menyeimbangkan beban kerja di antara PM yang mendasari layanan cloud di atasnya, menghemat penggunaan daya data center cloud dengan menonaktifkan PM yang idle setelah VM di migrasikan dan memaksimalkan utilisasi dari VM itu sendiri dalam melayani akses permintaan dari pelanggan.

1.1 Konsolidasi Virtual Machine Dinamis

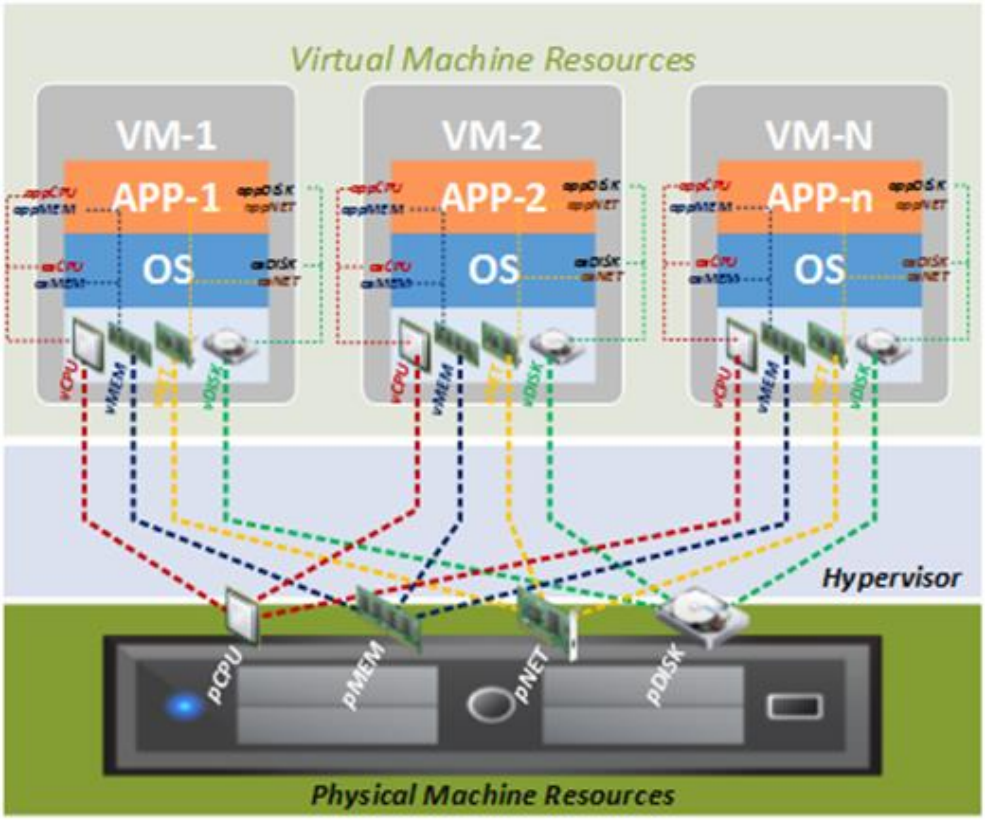
Secara garis besar [3][4] mengelompokkan strategi konsolidasi VM menjadi: konsolidasi VM statis dan konsolidasi VM dinamis. Konsolidasi VM tersebut dapat dilakukan dengan mempertimbangkan kapan, yang mana dan kemana VM harus dimigrasikan diantara PMnya atau host di dalam data center. Prosedur konsolidasi VM dinamis oleh [5][6][7] diuraikan ke dalam sub pengambilan keputusan yaitu host overloading detection, host underloading detection, VM selection dan VM Placement. Seperti pada gambar 1 host overloading detection dan host underloading detection digunakan untuk menentukan kapan suatu host mengalami overloaded ataupun underloaded. Dalam kondisi ini, satu atau lebih VM harus bermigrasi dari host tersebut. Keputusan VM mana yang harus dimigrasikan dari suatu host ke host lainnya akan dilakukan pada tahapan VM selection (pemilihan VM). VM placement (penempatan VM) akan memetakan kemana VM yang terpilih harus di migrasikan pada suatu host yang relatif underloaded dengan host lainnya.



Gambar 1. Prosedur Konsolidasi VM Dinamis

Di dalam prosedur konsolidasi VM dinamis, deteksi host yang overloading sering kali menjadi fokus utama guna mengurangi beban kerja yang dikaitkan dengan tujuan penghematan konsumsi energi yang ada di data center cloud[4]. Keputusan yang diambil dalam setiap prosedur konsolidasi VM dinamis didasarkan pada pendefinisian aturan yang diambil dari utilisasi sumber daya komputasi PM. Aturan yang ditetapkan biasanya menggunakan nilai yang cenderung konstan atau statis dengan mempertimbangkan bobot tertinggi dan terendah dari beban kerja yang menggunakan sumber daya komputasi, sehingga dapat dimungkinkan terjadinya overhead di dalam konsolidasi VM dinamis ketika memigrasikan VM. Overhead ini muncul sebagai akibat dari migrasi VM yang seharusnya tidak di perlukan atau VM yang seharusnya di migrasikan tetapi tidak dilakukan sebagai akibat dari penetapan policy dari host yang overloading dan underloading yang kurang sesuai. Deteksi host yang overload dan underload merupakan keputusan awal di dalam prosedur konsolidasi VM dinamis, jika keputusan awal tersebut tidak tepat maka akan mempengaruhi pengambilan keputusan pada langkah selanjutnya yaitu pemilihan VM yang akan di migrasikan dan penempatan VM yang terpilih pada host yang yang terpilih juga.

Utilisasi sumber daya cloud computing merupakan dasar yang dijadikan dalam pengambilan keputusan pada setiap prosedur konsolidasi VM dinamis. Sumber daya tersebut terdiri dari sumber daya fisik dan sumber daya virtual seperti yang ditunjukkan pada gambar 2.



Gambar 2. Sumber Daya Pada Cloud Computing

Sumber daya virtual memiliki keanekaragaman berkaitan dengan jenis aplikasi atau service yang dijalankan pada virtual machine. Variasi dari jenis aplikasi ini, akan menghasilkan workload dengan tingkat konsumsi sumber daya yang berbeda-berbeda. Selain itu, lingkungan cloud computing yang dinamis dan terdistribusi mengharuskan strategi pengambilan keputusan di dalam konsolidasi VM harus dibuat sedinamis mungkin atau bahkan adaptif dengan

 E-ISSN: 2621-4474, ISSN:2621-4458.

mempertimbangkan heterogenitas sumber daya virtual sesuai dengan layanan cloud computing yang disajikan. Sehingga tujuan efisiensi energi sekaligus menjaga kinerja layanan cloud computing melalui penyeimbangan beban kerja dengan teknik migrasi VM yang terdapat pada prosedur konsolidasi VM dinamis dapat terpenuhi. Pengambilan keputusan pada[1] rata-rata hanya menggunakan satu kriteria yaitu CPU sebagai parameter di dalam pengambilan keputusan pada prosedur konsolidasi VM. Sehingga, masih diperlukan usaha pengambilan keputusan pada setiap langkah konsolidasi VM dinamis yang mempertimbangkan beberapa kriteria sekaligus (tidak hanya satu kriteria) dari sumber daya cloud computing agar nantinya keputusannya lebih optimal.

1.2 Kriteria-kriteria Sumber Daya Cloud Computing

Sistem Cloud Computing pada dasarnya terdiri dari perangkat keras yang berupa physical machine seperti halnya komputer server dan perangkat lunak mesin virtual (virtual machine). Baik mesin fisik maupun virtual tentunya memiliki komponen sumber daya untuk pemrosesan, penyimpanan dan konektivitas. Komponen-komponen tersebut sangat berpengaruh terhadap pengambilan keputusan pada prosedur konsolidasi VM. Untuk tujuan efisiensi energi maupun peningkatan kinerja layanan cloud, parameter yang sering digunakan adalah processor saja seperti halnya pada [5] untuk diterapkan pada pengambilan keputusan hanya di salah satu prosedur konsolidasi VM. Padahal layanan cloud computing yang ditawarkan bervariasi jenisnya dengan tingkat konsumsi sumber daya yang berbeda-beda.

Oleh karena itu multi kriteria akan dijadikan sebagai pertimbangan dasar di dalam pengambilan keputusan pada prosedur konsolidasi VM. Pada PM kriteria-kriteria yang akan digunakan seperti pada pada tabel 1 berikut ini:

Tabel 1. Kriteria Physical Machine

Kriteria	Nama Atribut	Deskripsi
C1	$pCPU_u$ %	Prosentase penggunaan CPU PM
C2	$pRAM_u$ %	Prosentasi penggunaan RAM PM
C3	$pNET_u$ %	Prosentase penggunaan Network PM
C4	$pDISK_u$ %	Prosentase penggunaan *disk storage* PM
C5	$pDISK_{io}$ %	Prosentase *disk storage IO* PM
C6	pCPUc	Cadangan kapasitas CPU PM
C7	pRAMc	Cadangan kapasitas RAM PM
C8	pNETc	Cadangan kapasitas *bandwidth* PM
C9	pDISKc	Cadangan kapasitas *disk* PM
C10	VM	Jumlah VM pada PM

Kriteria pada tabel 1 dipetakan ke dalam jumlah physical machine yang akan digunakan seperti pada tabel 2.

Tabel 2. Pemetaan kriteria PM dengan PM yang digunakan

Alternatif	Atribut (kriteria)					
	pC1	pC2	pC3	pC4	pC5	pCn
PM 1	P1C1	P1C2	P1C3	P1C4	P1C5	P1Cn
PM 2	P2C1	P2C2	P2C3	P2C4	P2C5	P2Cn
PM 3	P3C1	P3C2	P3C3	P3C4	P3C5	P3Cn
PM 4	P4C1	P4C2	P4C3	P4C4	P4C5	P4Cn
PM 5	P5C1	P5C2	P5C3	P5C4	P5C5	P5Cn
PM m	PmC1	PmC2	PmC3	PmC4	PmC5	PmCn

Kriteria-kriteria physical machine tersebut akan digunakan dalam pengambilan keputusan Kriteria-kriteria physical machine tersebut akan digunakan dalam pengambilan keputusan pada prosedur konsolidasi VM di tahapan deteksi PM atau host yang overload dan underload serta penempatan VM agar lebih optimal.

Sedangkan pada *Virtual Machine* (VM) kriteria-kriterianya adalah sebagai berikut pada *Virtual Machine* (VM) kriteria-kriterianya adalah sebagai berikut:

Tabel 3. Kriteria Virtual Machine

Kriteria	Nama Atribut	Deskripsi
vC1	$vCPU_u$ %	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	vCPUc	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

Kriteria pada tabel 3 dikombinasikan dengan jumlah virtual machine yang digunakan akan tampak seperti pada tabel 4.

Tabel 4. Pemetaan Kriteria VM dengan jumlah VM yang digunakan

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

Kriteria-kriteria VM tersebut digunakan untuk pengambilan keputusan pada prosedur konsolidasi VM ketika tahapan pemilihan VM yang akan dimigrasikan agar lebih optimal.

2. METODOLOGI

Zimmerman (1991) dalam[8], pengambilan keputusan berdasarkan multi kriteria terdiri dari *Multi Attribute Decision Making (MADM)* dan *Multi Objective Decision Making (MODM)*. MADM digunakan untuk menyelesaikan permasalahan dalam ruang diskret dengan melakukan penilaian atau seleksi terhadap beberapa alternative dalam jumlah terbatas. Sedangkan MODM digunakan untuk menyelesaikan masalah-masalah pada ruang kontinyu seperti pada pemrograman matematika. MADM menyeleksi alternative terbaik dari sejumlah *alternative*, sedangkan MODM merancang alternative terbaik. Dari tipe datanya, multi kriteria pengambilan keputusan dibagi menjadi *deterministic*, stokastik atau *fuzzy*.

Medote MADM yang akan digunakan untuk menyelesaikan masalah pengambilan keputusan pada prosedur konsolidasi VM dinamis yang sesuai adalah *Technique for Order Preference by Similarity to Ideal Solution (TOPSIS)*. Menurut hwang (1982), zeleny (1982) dalam [5], TOPSIS didasarkan pada konsep dimana alternatif terpilih yang terbaik tidak hanya memiliki jarak terpendek dari solusi ideal positif, namun juga memiliki jarak terpanjang dari solusi ideal negatif. Langkah-langkah penyelesaikan masalah menggunakan TOPSIS adalah:

1. Menghitung matriks keputusan.

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

2. Normalisasi matriks keputusan.

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

3. Menentukan kriteria matriks pembobotan.

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

4. Normalisasi pembobotan matriks keputusan.

$$v_{ij} = w_i r_{ij} = W \; x \; R, i = 1, \dots m, dan \; j = 1, \dots, n$$

dimana w_i adalah bobot dari kriteria j dan $\sum_{i=1}^{n} w_i$=1, (4)

5. Mendapatkan solusi ideal positif dan solusi ideal negative.

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

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

6. Menentukan Euclidean Distance matriks berdimensi n.

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

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

7. Menentukan jarak kedekatan relative dengan solusi ideal.

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

8. Perengkingan alternatif ditinjau dari kedekatan relative dengan solusi ideal.

2.1 Penerapan TOPSIS Pada Prosesur Konsolidasi VM

Deteksi PM yang *underload* dilakukan jika nilai utilisasi dari masing-masing sumber daya kurang dari nilai ambang batas yang di tetapkan maka statusnya adalah underload. Implementasi dari algoritma *underload detection* adalah sebagai berikut:

Tabel 5. Algoritma *Underload Detection*

Input	: utlisasi sumber daya pada suatu host dan threshold
Output	: status underload
1	**if** host_resources_util <= threshold :
2	host underload

Deteksi PM yang *overload* merupakan kebalikan dari deteksi PM yang underload yaitu apabila nilai utilisasi dari masing-masing sumber daya yang di konsumsi oleh suatu host lebih dari nilai ambang batas yang di tetapkan maka statusnya adalah *overload*. Implementasi dari algoritma *overload detection* adalah sebagai berikut:

Tabel 6. Algoritma *Overload Detection*

Input	: utlisasi sumber daya pada suatu host dan threshold
Output	: status overload
1	**if** host_resources_util > threshold :
2	host overload

Pemilihan *Virtual Machine* (VM) dilakukan pada setiap kondisi dimana host mengalami status underload atau overload. Utilisasi dari sumber daya yang digunakan oleh VM akan di jadikan sebagai masukan untuk menentukan VM mana yang akan di pilih untuk di migrasikan. Pseudocode dari algoritma pemilihan VM yaitu:

Tabel 7. Algoritma Pemilihan VM

Input	: sumber daya vm, bobot, rules
Output	: kandidat vm yang akan di pindahkan
1	vm_res = [cpu, memory, disk_io, net_io]
2	matriks_d = numpy.array(vm_res)
3	matriks_r = topsis.trans_to_matriks_r(matriks_d)
4	matriks_v = topsis.trans_to_matriks_v(matriks_r, bobot)
5	solusi_ideal = topsis.trans_to_A(matiks_v, rules)
6	jarak_alternatif = topsis.trans_to_S(matriks_v, solusi_ideal)
7	preferensi = topsis.trans_to_C(jarak_alternatif)
8	preferensi_user = sorted(preferensi)
9	vm_uuids = preferensi_user

Untuk mendapatkan VM yang akan di migrasikan dari host yang mengalami underload atau overload, maka di terapkan pemilihan VM dengan menggunakan *Multi Attribute Decision Making (MADM)* berupa TOPSIS. Sebagai masukannya adalah utilisasi dari penggunaan sumber daya oleh vm yang kemudian di transformasikan kedalam bentuk matriks_d menggunakan modul *numpy* dari *python*. Seletah matriks_d di dapatkan, maka di jadikan sebagai masukan dari matriks_r. matriks_r ini di jadikan sebagai masukan untuk mendapatkan matriks_v dengan menambahkan bobot yang telah di tentukan. Solusi idela di dapatkan dengan memasukkan matriks_v dan menambahkan aturan apakah kriteria yang digunakan itu sebagai biaya atau keuntungan (*rules*). Pada TOPSIS, solusi_ideal yang di dapatkan masih di proses guna

mendapatkan jarak_alternatif dengan memasukkan matriks_v dan nilai solusi_ideal tersebut. Jarak_alternatif inilah yang akan digunakan sebagai masukkan untuk mendapatkan nilai preferensinya. Nilai preferensi_user merupakan hasil dari pengurutan nilai preferensi yang merupakan vm_uuids sebagai kandidat vm yang akan di pindahkan. vm_uuids ini di parsing ke global manager guna menentukan host yang akan di tuju untuk menempatkan vm tersebut.

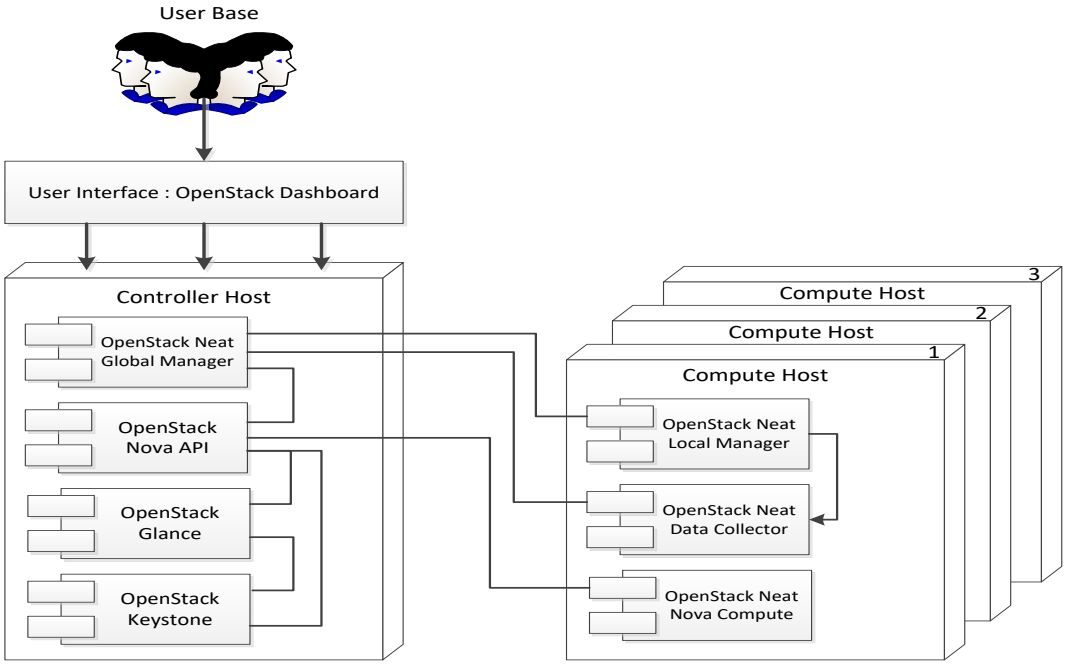
Penempatan VM yang terpilih pada dasarnya sama dengan prosedur pemilihan VM, hanya masukannya berdasarkan dari sumber daya yang digunakan oleh suatu host. Penempatan VM menghasilkan kandidat dari host yang akan di jadikan sebagai tujuan dari VM yang di migrasikan. Pseudocode dari algoritma penempatan VM adalah:

Tabel 8. Algoritma Penempatan VM

Input : sumber daya host, bobot, rules	
Output : kandidat host yang akan di tuju untuk migrasi vm	
1	vm_res = [cpu, memory, disk_io, net_io]
2	matriks_d = numpy.array(vm_res)
3	matriks_r = topsis.trans_to_matriks_r(matriks_d)
4	matriks_v = topsis.trans_to_matriks_v(matriks_r, bobot)
5	solusi_ideal = topsis.trans_to_A(matiks_v, rules)
6	jarak_alternatif = topsis.trans_to_S(matriks_v, solusi_ideal)
7	preferensi = topsis.trans_to_C(jarak_alternatif)
8	preferensi_user = sorted(preferensi)
9	vm_uuids = preferensi_user

2.2 Implementasi Sistem

Konsolidasi VM dinamis diimplementasikan kedalam lingkungan *cloud computing* secara riil menggunakan *OpenStack* dengan konsolidasi VM dinamis berbasis OpenStack-Neat [9][10] sesuai dengan topologi yang di tunjukkan pada gambar 3.



Gambar 3. Lingkungan *Cloud Computing* Berbasis OpenStack

Spesifikasi perangkat *controller host* yang di gunakan dalam di tunjukkan pada tabel 5.

Tabel 8. Spesifikasi *ControllerHhost* dan *User Base*

	Controller-host	***User-base***
Processor	Intel i3-2330M, 2 Core, 4 Threads, CPU @ 2.20 GHz	Intel i3-2330M, 2 Core, 4 Threads, CPU @ 2.20 GHz
RAM	4 GB	4 GB
Hardisk	500 GB	500 GB
Sistem Operasi	CentOS 6.5	Windows
Jumlah	1	1

Sedangkan spesifikasi perangkat *compute host* yang di gunakan di tunjukkan pada tabel 6.

Tabel 9. Spesifikasi *compute host*

Spesifikasi	***CPU-Intensive***	***MEM-Intensive***	***NET-Intensive***	***Disk-Intensive***
Hostname	*Compute-host2*	*Compute-host3*	*Compute-host4*	*Compute-host1*
Processor	Intel i3-2330M, 2 Core, 4 Threads, CPU @ 2.20 GHz	Intel i3-2330M, 2 Core, 4 Threads, CPU @ 2.20 GHz	Intel i3-2330M, 2 Core, 4 Threads, CPU @ 2.20 GHz	Intel i3-2330M, 2 Core, 4 Threads, CPU @ 2.20 GHz
RAM	4 GB	4 GB	4 GB	4 GB
Hardisk	500 GB	500 GB	500 GB	500 GB
Sistem Operasi	CentOS 6.5	CentOS 6.5	CentOS 6.5	CentOS 6.5
Jumlah	1	1	1	1

3. HASIL DAN PEMBAHASAN

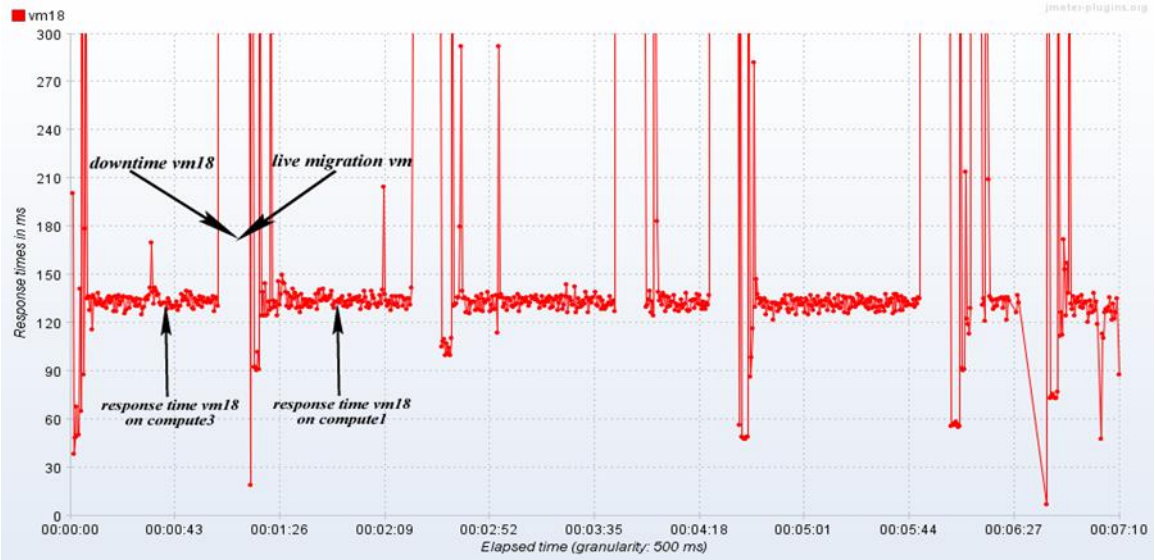
Hasil uji coba melalui pemberiaan peningkatan beban kerja secara periodik dengan waktu percobaan 300 detik guna mengetahui *workload* maksimum yang mampu diproses oleh suatu VM sesuai dengan layanannya. Pada *instance* VM selain layanan WEB (apache2+PHP5) ditambahkan layanan ServerAgent guna mendapatkan informasi tentang penggunaan sumber daya oleh VM. Hasil yang didapatkan dari hasil percobaan seperti pada Tabel 4.

Tabel 10. Hasil Uji Coba Pemberian *Workload* pada VM Selama 5 Menit

Percobaan	**Threads (Users)**	**Ramp-up (Sec)**	**Response Times (ms)**	**Throughput (Bytes/sec)**		**Latency (ms)**
				Bytes Recv	Bytes Sent	
1	1	1	5,3	6800000	6500	2,2
2	10	1	37,5	11195000	11830	15,8
3	20	1	76,5	11350000	11300	33,5
4	30	1	115,5	11500000	11450	52,2
5	35	1	139,5	11700000	11350	63,2
6	40	1	158,5	11450000	11250	74,4

Pada table menunjukkan penurunan *response times* selama percobaan ke-1 hingga ke-6. Untuk jumlah threads sebanyak 40, ServerAgent tidak dapat memberikan informasi tentang penggunaan sumber daya dari VM karena kekurangan jumlah CPU dan *service* secara otomatis mati. Sehingga jumlah *threads* maksimum yang mampu diproses oleh suatu vm selama 1 detik sebanyak 35 users.

Dengan meningkatkan beban kerja menjadi 35 *threads* dengan ramp-up 1 detik dan dilakukan selama periode waktu berulang. Penerapan prosedur konsoldiasi VM yang di usulkan tampak seperti pada gambar 4.



Gambar 4. *Response Time* pada *Live Migration* VM

Dari gambar 4 vm18 terpilih untuk dimigrasikan dan ketika menempati *compute3 response times* nya adalah 134 ms dan cenderung sama ketika menempati compute1 dan *compute2*. Vm18 cenderung mengalami peningkatan *response time* hingga 115 ms, ketika migrasi ke compute4 dikarenakan pada *compute4* cenderung *idle*. *Down time* ketika terjadi proses *live migration* vm18 di antara *compute node* adalah 13 detik.

4. KESIMPULAN

Strategi pengambilan keputusan pada prosedur konsolidasi *virtual machine* dinamis yang diusulkan, dapat meningkatkan kinerja layanan cloud computing. Hal ini dibuktikan dengan peningkatan response time dari 134 ms menjadi 115 ms ketika sebelum dan setelah proses migrasi *virtual machine*. Beban kerja physical machine menjadi seimbang karena keputusan pemilihan VM dan penempatan VM pada physical machine dipilih secara optimal melalui MADM. Hal tersebut dapat dilihat bahwa VM terpilih mengalami peningkatan response time ketika menempati compute node yang cenderung idle.

Konsumsi energi dari *physical machine* dapat di hemat dengan mematikannya karena statusnya idle sebagai akibat dari beban kerjanya menurun drastis setelah proses migrasi *virtual machine*. Selanjutnya masih terdapat potensi optimasi penyeimbangan beban kerja di lingkungan *cloud computing* misalnya dengan menggunakan *Artificial Intellegence* (AI) atupun *Machine Learning* (ML). Peluang efisiensi energi data center cloud juga masih terbuka guna menjawab sampai berapa persen penghematannya.

5. DAFTAR PUSTAKA

[1] A. Fox *et al.*, "Above the clouds: A berkeley view of cloud computing," *Dept. Electr. Eng. Comput. Sci. Univ. California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009.

[2] M. Gusev, G. Velkoski, S. Ristov, and M. Simjanoska, "Web service CPU overutilization in the cloud," in *ICIT 2013 The 6th International Conference on Information Technology. Cloud Computing*, 2013, pp. 8–10.

[3] M. A. Khan, A. Paplinski, A. M. Khan, M. Murshed, and R. Buyya, "Dynamic virtual

machine consolidation algorithms for energy-efficient cloud resource management: a review," in *Sustainable cloud and energy services*, Springer, 2018, pp. 135–165.

[4] B. Bermejo, C. Juiz, and C. Guerrero, "Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance," *J. Supercomput.*, vol. 75, no. 2, pp. 808–836, 2019.

[5] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurr. Comput. Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, 2012.

[6] S. Kusumadewi, S. Hartati, A. Harjoko, and R. Wardoyo, "Fuzzy Multi-Attribute Decision Making (Fuzzy MADM)," *Yogyakarta Graha Ilmu*, pp. 78–79, 2006.

[7] S. Wibowo, H. Deng, and W. Xu, "Evaluation of cloud services: A fuzzy multi-criteria group decision making method," *Algorithms*, vol. 9, no. 4, p. 84, 2016.

[8] H.-J. Zimmermann and H.-J. Sebastian, "Intelligent system design support by fuzzy-multi-criteria decision making and/or evolutionary algorithms," in *Proceedings of 1995 IEEE International Conference on Fuzzy Systems.*, 1995, vol. 1, pp. 367–374.

[9] A. Beloglazov and R. Buyya, "OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds," *Concurr. Comput. Pract. Exp.*, vol. 27, no. 5, pp. 1310–1333, 2015.

[10] F. F. Moges and S. L. Abebe, "Energy-aware VM placement algorithms for the OpenStack Neat consolidation framework," *J. Cloud Comput.*, vol. 8, no. 1, p. 2, 2019.