# A Comparative Study of VM Placement in Cloud Data Center in terms of Energy Efficiency

## Tugas Akhir

Disampaikan Sebagai Bagian Dari Persyaratan Kelulusan Diploma 3
Program Studi Teknik Komputer

**Oleh:**


13315014 Johan Reynaldi Sirait
13315017 Adedimita
13315025 Soni Pratama



**Institut Teknologi Del**

**2017/2018**

**Lembar Pengesahan Tugas Akhir**

**Institut Teknologi Del**

# A Comparative Study of VM Placement in Cloud Data Center in terms of Energy Efficiency

**Oleh:**


13315014 Johan Reynaldi Sirait

13315017 Adedimita

13315025 Soni Pratama


Sitoluama, 21 Juni 2018

| Pembimbing I | Pembimbing II |
|---|---|
| Eka Stephani Sinambela, SST., M.Sc | Gerry Italiano Wowiling, S.Tr.Kom |

**Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan**

**Sebagai**

**Laporan Tugas Akhir Diploma 3**

**Program Studi Teknik Komputer**

**Institut Teknologi Del**

# Prakata

Puji dan syukur kepada Tuhan Yang Maha Esa atas berkatNya yang menyertai penulis selama pengerjaan tugas akhir, sehingga tugas akhir ini dapat diselesaikan tepat pada waktunya dengan judul "*A Comparative Study of* VM *Placement in Cloud Data Center in terms of Energy Efficiency*". Tugas Akhir ini disusun dengan tujuan untuk memenuhi salah satu persyaratan untuk menyelesaikan Program Diploma 3 Jurusan Teknik Komputer di Institut Teknologi Del. Penulis menyampaikan terima kasih kepada Bapak Pandapotan Siagian, ST, M.Eng selaku penguji Tugas Akhir penulis, Ibu Eka Stephani Sinambela, SST., M.Sc., selaku pembimbing I dan Bapak Gerry Italiano Wowiling, S.Tr.Kom selaku pembimbing II penulis yang telah memberikan arahan dan bimbingan selama pelaksanaan Tugas Akhir.

Penulis juga menyampaikan terima kasih kepada seluruh Bapak/Ibu Dosen Institut Teknologi Del khususnya pada Jurusan Teknik Komputer yang telah membekali penulis dengan disiplin ilmu yang berguna tidak hanya di kampus Institut Teknologi Del, melainkan untuk mempersiapkan penulis ke dunia kerja. Penulis juga mengucapkan terima kasih kepada orangtua, saudara, sahabat, teman-teman angkatan 2015 yang telah memberi banyak dukungan dalam pengerjaan Tugas Akhir ini. Penulis menyadari bahwa Tugas Akhir ini masih memiliki kekurangan dan kesalahan. Karena itu kritik dan saran yang membangun akan diterima dengan senang hati. Semoga dengan adanya Tugas Akhir ini dapat bermanfaat dan menambah wawasan bagi semua pihak, khususnya tentang mengukur efisiensi dari *network energy*.

Akhir kata penulis mengucapkan terimakasih.

Sitoluama, 21 Juni 2018

13315014 Johan Reynaldi Sirait
13315017 Adedimita
13315025 Soni Pratama

# Abstrak

*Data center* adalah sumber daya komputasi yang menggunakan jaringan komunikasi sebagai layanan untuk mengelola aplikasi dan menyimpan data. Layanan tersebut adalah *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), and *Software as a Service* (SaaS) yang tersedia dengan sistem prabayar. Dengan keadaan tersebut, maka *data center* harus menyediakan kinerja yang dapat diandalkan untuk memenuhi beragam kebutuhan para pengguna *cloud computing*. Dengan menggunakan *data center*, terjadi peningkatan penggunaan energi karena dalam sebuah *data center* terjadi transmisi paket, maka semakin banyak energi yang diperlukan untuk melakukan proses transmisi paket tersebut.

Untuk melakukan transmisi paket, dibutuhkan energi untuk menjalankan PM (*Phyical Machine*) dan membutuhkan energi jaringan untuk mengirimkan paket-paket data tersebut. Oleh karena itu, *data center* membutuhkan *Virtual Machine Placement (*VM *Placement)* untuk menempatkan VM pada PM *(Physical Machine) / host* yang tepat, sehingga jumlah PM yang dijalankan dapat diminimalisir sesuai dengan yang dibutuhkan. *Data Center Network Architecture* yang tepat juga diperlukan untuk mencapai target efisiensi energi pada *network*.

Oleh karena itu, dalam Tugas Akhir yang berjudul "*A Comparative Study of* VM *Placement in Cloud Data Center in terms of Energy Efficiency*" ini, penulis melakukan perhitungan transmisi paket untuk mendapatkan efisiensi energi. Untuk mencapai tujuan diatas, penulis melakukan pengujian menggunakan *topology Fat-Tree*. Untuk melakukan pengujian tersebut, penulis menggunakan CloudSimSDN, karena dengan menggunakan CloudSimSDN, penulis tidak hanya bisa menghitung jumlah energi daya yang digunakan, tetapi CloudSimSDN dapat membantu penulis untuk menghitung energi jaringan yang digunakan dalam proses simulasi.

**Kata Kunci** : *Data Center, Fat-Tree*, VM *Placement*, CloudSimSDN.

# Daftar Isi

# Daftar Tabel

# Daftar Gambar

# BAB I

# Pendahuluan

Bab ini berisi penjelasan mengenai latar belakang, tujuan pelaksanaan, ruang lingkup, pendekatan yang dilakukan dan sistematika penyajian Tugas Akhir.

## 1.1 Latar Belakang

*Data center* adalah sumber daya komputasi yang menggunakan jaringan komunikasi sebagai layanan untuk mengelola aplikasi dan menyimpan data [1]. Layanan tersebut adalah *Infrastructure as a Service* (*IaaS*), *Platform as a Service* (*PaaS*), and *Software as a Service* (*SaaS*) [2], yang tersedia dengan sistem prabayar. Oleh karena itu, *data center* harus menyediakan kinerja yang dapat diandalkan untuk memenuhi beragam kebutuhan pengguna. Untuk mencapai tujuan ini, dibutuhkan sebuah mesin yang sering disebut sebagai PM (*Physical Machine)* dan perangkat jaringan (*network device)*, contohnya adalah *switch*. Dengan penggunaan mesin tersebut, maka konsumsi energi yang dibutuhkan meningkat. Pada tahun 2012, konsumsi energi oleh *data center* diperkirakan meningkat dalam jumlah 270 TWh penggunaan total tenaga listrik di seluruh dunia [3].

Dengan menggunakan *data center*, terjadi peningkatan penggunaan energi karena dalam sebuah *data center* terjadi transmisi paket, maka semakin banyak energi yang diperlukan untuk melakukan proses transmisi paket tersebut. Selain itu, dengan menggunakan *data center*, maka energi jaringan mengkonsumsi 10% – 20% dari energi normal [4]. Untuk melakukan transmisi paket, dibutuhkan energi untuk menjalankan PM (*Phyical Machine*). Tidak hanya membutuhkan energi untuk menjalankan PM, tetapi pada layanan *data center* dibutuhkan energi jaringan untuk mengirimkan transmisi paket-paket data [5]. PM yang dimaksudkan dalam pengujian ini adalah *host*. Oleh karena itu, *data center* membutuhkan *Virtual Machine Placement* (VM *Placement*) untuk menempatkan VM pada PM yang tepat, sehingga jumlah PM yang dijalankan dapat diminimalisir sesuai dengan yang dibutuhkan. *Data Center Network Architecture* yang tepat juga diperlukan untuk mencapai target efisiensi energi pada *network*. Dengan demikian, penempatan VM dan *Data Center Network Architecture* yang tepat, energi dapat dihemat secara lebih signifikan.

Oleh karena itu, dalam Tugas Akhir ini penulis berkontribusi untuk menghitung transmisi paket untuk mendapatkan efisiensi energi dengan membandingkan algoritma *bestfit* dan *worstfit* pada proses VM *Placement*. Untuk mencapai tujuan diatas, penulis melakukan pengujian terhadap CloudSimSDN dengan skenario penelitian dan *topology Fat-Tree*.

Untuk melakukan pengujian tersebut, penulis menggunakan CloudSimSDN, karena dengan menggunakan CloudSimSDN, penulis tidak hanya bisa menghitung jumlah energi daya yang digunakan, tetapi CloudSimSDN dapat membantu penulis untuk menghitung energi jaringan yang digunakan dalam proses simulasi.

Untuk mencapai tujuan tersebut, penulis merumuskan pertanyaan utama dalam penelitian pada Tugas Akhir ini:

1. Mengapa dalam beberapa penelitian *Data Center Network Architecture,* para peneliti lebih sering menggunakan *topology Fat-Tree*?
2. Apakah ada dampak yang berbeda ketika VM *Placement* diterapkan ke dalam *Data Center Network Architecture*?

## 1.2 Tujuan

Adapun tujuan dari Tugas Akhir ini adalah membandingkan algoritma *bestfit* dan *worstfit* yang digunakan dalam VM *Placement* pada perhitungan transmisi paket yang digunakan dalam sebuah *data center* untuk memperoleh efisiensi energi jaringan.

## 1.3 Lingkup

Batasan dan ruang lingkup dalam pengerjaan Tugas Akhir adalah:

1. Tidak mengoptimasi VM *Placement Policy*, melainkan hanya menggunakan VM *Placement Policy* yang sudah ada pada CloudSimSDN.
2. Pengujian dilakukan menggunakan *topology Fat-Tree* dan menggunakan simulator CloudSimSDN.
3. Penelitian dilakukan berdasarkan penelitian yang menjalankan skenario penelitian.

## 1.4 Pendekatan

Pendekatan yang dilakukan pada Tugas Akhir ini terdiri dari:

**1. Mempelajari *literatur***

Melakukan *study literatur* mengenai *Data Center Network Architecture* dengan membaca dan memahami beberapa jurnal yang berkaitan dengan Tugas Akhir.

**2. Membuat gambaran *system overview***

Menggambarkan cara kerja *Simulator* CloudSim yang digunakan penulis sebagai *Simulator* untuk menghitung jumlah daya yang akan digunakan dalam *data center*.

**3. *Experiment Planning***

Merancang percobaan untuk membandingkan satu *topology* satu dengan *topology* yang lain untuk menemukan *topology* yang paling tepat untuk proses *energy efficiency*.

**4. *Testing and Analysis***

Melakukan pengujian terhadap implementasi yang telah dilakukan dan juga melakukan analisis terhadap hasil implementasi.

**5. Membuat Dokumentasi**

Menuliskan laporan dan dokumentasi selama pengerjaan Tugas Akhir.

## 1.5 Sistematika Penyajian

Secara garis besar dokumen ini disajikan dalam enam bab, laporan Tugas Akhir ini disusun dengan sistematika berikut:

**1. Bab I Pendahuluan**

Pada bab ini berisi penjelasan mengenai latar belakang, tujuan pelaksanaan, ruang lingkup, pendekatan yang dilakukan dan sistematika penyajian Tugas Akhir.

**2. Bab II Tinjauan Pustaka**

Pada bab ini menguraikan setiap dasar teori dan perangkat keras yang akan digunakan pada pengerjaan Tugas Akhir.

**3. Bab III Penelitian dan Analisis**

Pada bab ini akan dijelaskan mengenai penjelasan *experiment definition* dan *experiment planning*.

**4. Bab IV Implementasi**

Bab ini berisi uraian implementasi yang dilakukan pada Tugas Akhir. Proses implementasi yang dilakukan berupa perbandingan *Data Center Network Architecture*.

**5. Bab V Pengujian dan Analisis**

Pada bab ini berisi tentang pengujian terhadap implementasi yang telah dilakukan serta analisis dari hasil pengerjaan Tugas Akhir yang telah dilakukan.

**6. Bab IV Kesimpulan dan Saran**

Pada bab ini dituliskan mengenai kesimpulan dari pengerjaan Tugas Akhir ini serta saran-saran yang dibutuhkan untuk pengembangan penelitian.

# BAB II

# Tinjauan Pustaka

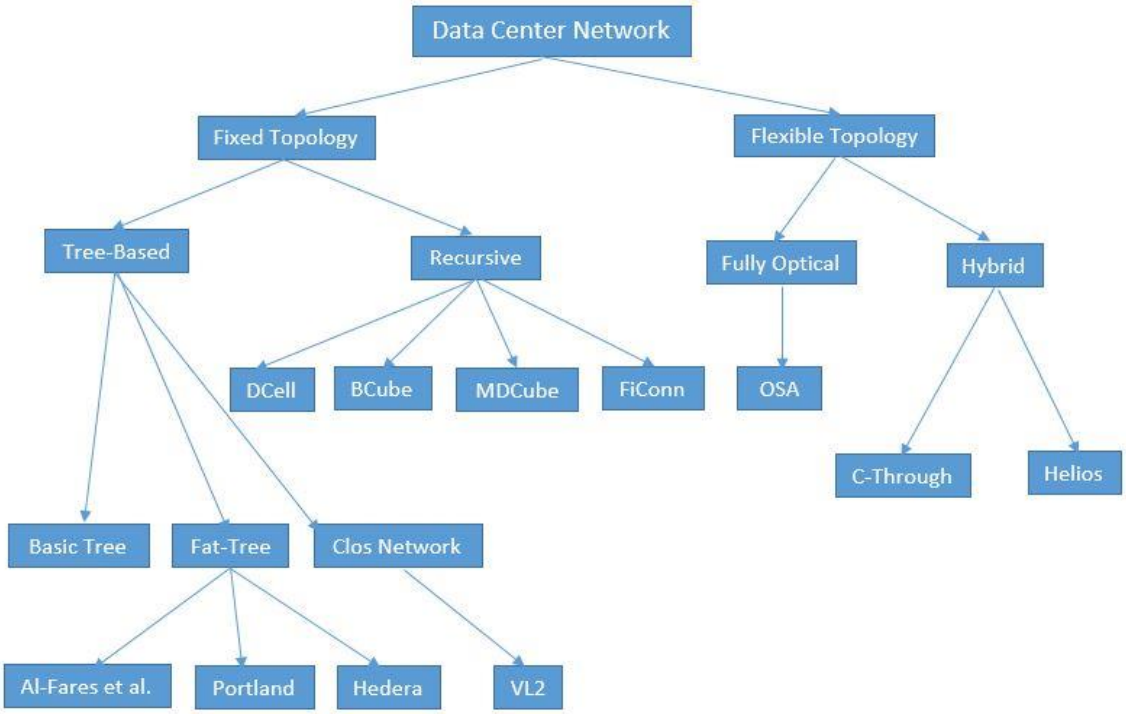
Pada bab ini dijelaskan tinjauan pustaka yang digunakan sebagai dasar teori untuk mengerjakan Tugas Akhir.

## 2.1 Landasan Teori

Pada bagian landasan teori ini diuraikan mengenai komponen-komponen *Data Center Network Architecture* yang digunakan untuk membandingkan konsumsi daya yang digunakan oleh *data center*.

### 2.1.1 *Data Center Network Architecture*

Arsitektur *data center* merupakan rancangan dalam membangun sebuah struktur *data center*. Arsitektur *data center* menentukan dimana dan bagaimana *host*, penyimpanan jaringan, dan sumber daya lain yang ditempatkan secara *physically*. Arsitektur *data center* tersebut dapat dilihat pada Gambar 1 dibawah ini.




**Gambar 1 *Data Center Network Architecture***

Terdapat dua kategori jaringan *data center* yang dibahas pada bagian ini yaitu *fixed topology* dan *flexible topology*. Jika jaringan *data center* tidak dapat di modifikasi setelah jaringan ditetapkan

pada sebuah aplikasi disebut sebagai *fixed topology,* tetapi sebaliknya jika jaringan *data center* dapat di modifikasi disebut *flexible topology.*

### 2.1.2 *Data Center Topology*

*Topology data center* adalah sebuah desain, konsep, struktur dalam membangun sebuah jaringan yang menghubungkan perangkat satu ke perangkat yang lain menggunakan suatu media, dengan adanya *topology data center* turut mempengaruhi tingkat kecepatan dan transfer data antar perangkat. *Topology data center* yang paling sering digunakan pada *Data Center Network Architecture* adalah sebagai berikut:

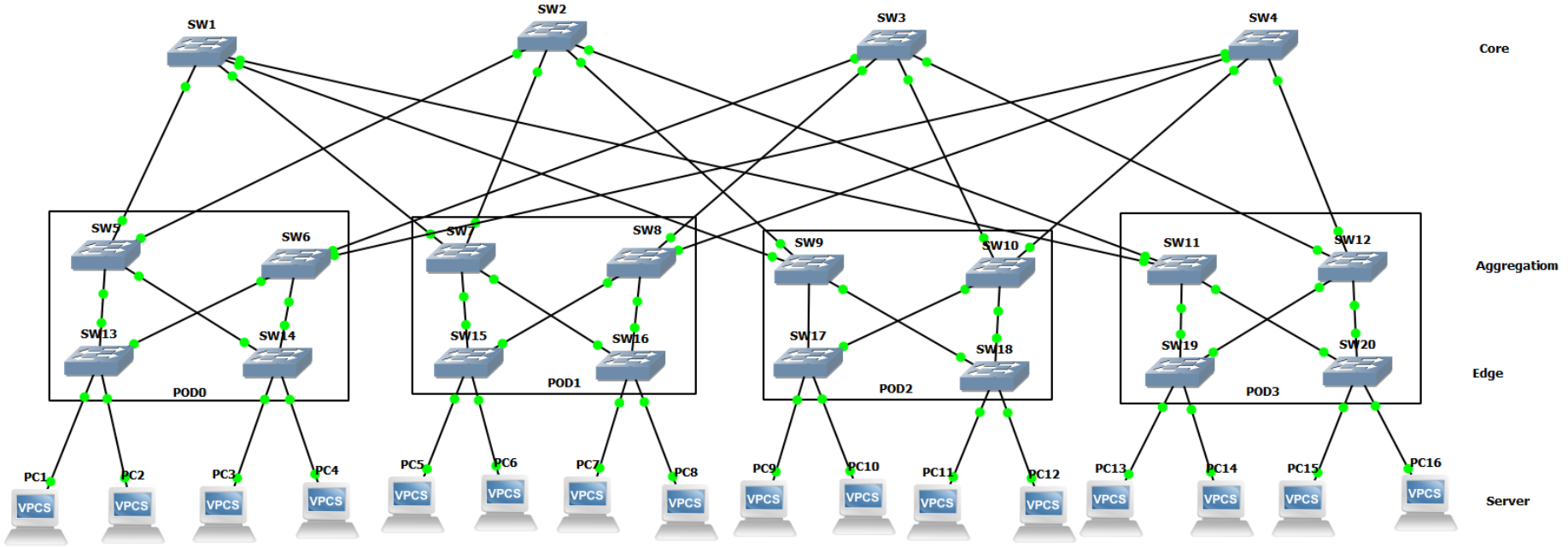
**1. *Fat-Tree Topology***

Pada penelitian ini, penulis menggunakan *Topology Fat-Tree*, karena jenis *topology* ini mudah untuk dikembangkan. Selain itu, *Fat-Tree* merupakan *topology* yang kompleks ketika dilakukan penambahan *switch* dan *links* [6]. *Fat-Tree* merupakan sebuah *topology* yang memiliki *routing* dengan tujuan untuk mendistribusikan *traffic* secara merata diantara inti *switch*. *Fat-Tree* juga merupakan sebuah jaringan berdasarkan *binary tree* yang lengkap. *Topology Fat-Tree* sering digunakan oleh beberapa peneliti karena dengan menggunakan karena *Fat-Tree* merupakan sebuah *topology* dengan sistem algoritma *simple routing* [7]. Karena dalam sebuah jaringan, mekanisme *routing* memiliki peranan yang sangat penting.

Misalkan:

1. n = jumlah *port* pada *switch*
2. N = jumlah total *host* yaitu sebanyak 4 buah [8]

Untuk setiap *switch* pada lapisan *edge* dengan *port* n terhubung ke $\frac{n}{2}$ *host*, sedangkan $\frac{n}{2}$ *port* yang lain terhubung ke *switch* dalam lapisan *aggregation*, masing-masing *switch* pada lapisan *edge* dan *aggregation* yang terhubung antara yang satu dengan yang lainnya dikelompokkan dalam *pod* yang sama yang dimana pada umumnya berbagi alamat *subnet* yang sama. Pada lapisan atas, ada *switch* $\left(\frac{n}{2}\right)^2$ *port* n yang disebut lapisan *core*. Masing-masing *switch* yang berada pada lapisan *core* terhubung pada satu *pod* dari *switch aggregation* pada setiap *pod*. Jumlah terbanyak dari *host* terhubung ke *port* n *switch edge* yang bisa dihitung dengan N = $\frac{n^3}{4}$. Sebagai contoh *topology Fat-Tree* dapat dilihat pada Gambar 2 dibawah ini.




**Gambar 2 *Fat-Tree Topology***

### 2.1.3 *Simulation Tools*

Pada bagian ini, menjelaskan mengenai komponen-komponen yang berbentuk *software* yang mendukung proses simulasi pada Tugas Akhir ini.

#### 2.1.3.1 *Cloud Simulator*

Tingginya penggunaan dari *data center* mengakibatkan peningkatan penggunaan energi. Karena itu, diperlukan perhitungan dan analisis efisiensi energi jarinan. *Cloud simulator* biasanya digunakan untuk menghitung dan menganalisis konsumsi energi pada *data center*. *Cloud simulator* yang paling populer untuk melakukan simulasi pada ruang lingkup *cloud* adalah CloudSim. Karena lebih populer dari *cloud simulator* yang lain, maka pada tugas ini akan menggunakan CloudSim sebagai *simulation tools*. Dalam penelitian ini, *simulator* digunakan sebagai model terhadap lingkungan berbasis *cloud* karena keterbatasan akses terhadap *data center*. Dengan demikian, *simulator* diperlukan untuk penelitian dalam lingkup *cloud*. Perbedaan antara CloudSim dan CloudSimSDN dapat dilihat pada Tabel 1 berikut [9]:

**Tabel 1 Perbandingan Analisis *Simulator***

| ***Simulator*** | ***Base Platform*** | ***Support SDN*** | ***Availability*** | ***Prog. Language*** | ***Cost Model*** | ***GUI*** | ***Simulation Time*** | ***Energy Model*** |
|---|---|---|---|---|---|---|---|---|
| CloudSim | SimJava | *No* | *Open Source* | *Java* | *Yes* | *No* | *Seconds* | *Yes* |
| CloudSimSDN | CloudSim | *Yes* | *Open Source* | *Java* | *Yes* | *Yes* | *Seconds* | *Yes* |

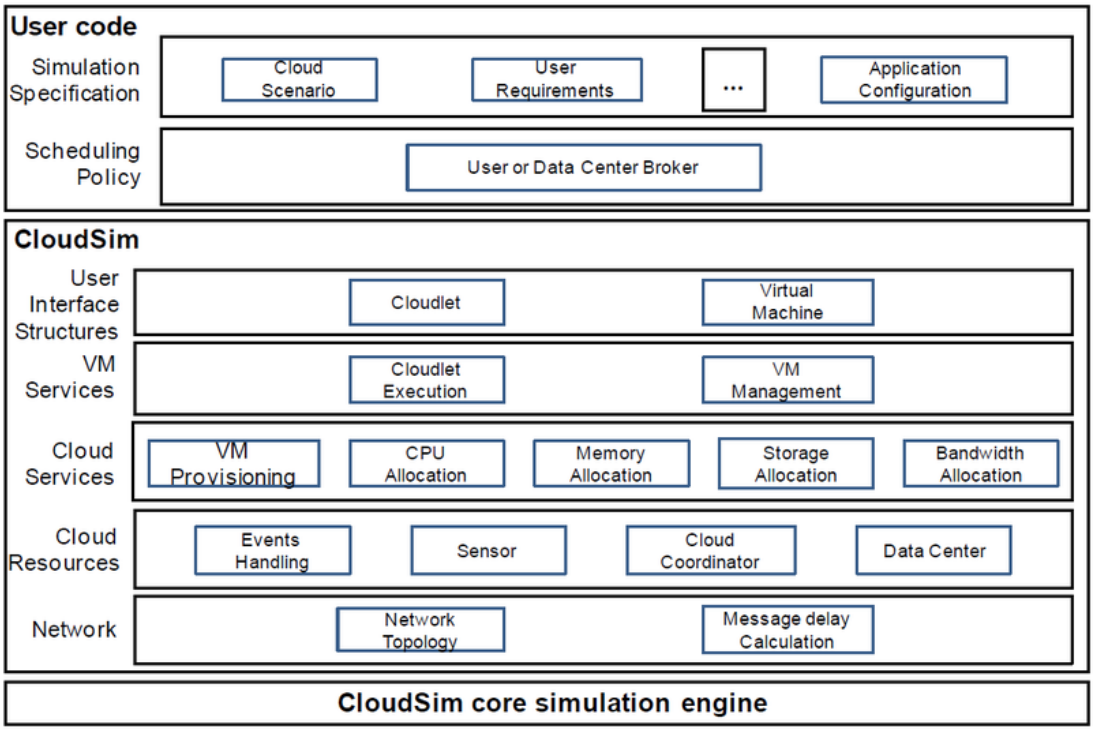
*Simulator* yang mendukung terbagi menjadi 2 jenis dengan memiliki spesifikasi perbedaan sebagai berikut:

**1. CloudSim**

CloudSim adalah program berbasis *Java* untuk melakukan simulasi *Cloud*. CloudSim dapat melakukan *modeling* infrastuktur *cloud*, simulasi, dan *experiment* pada ruang lingkup *cloud computing*. CloudSim juga dapat menampilkan konsumsi *power*, konsumsi *traffic*, dan *result time* dari hasil simulasi *experiment*. CloudSim menyediakan *class* dasar untuk menggambarkan *data center*, *virtual machines*, aplikasi, *user*, *resource* komputasi, dan kebijakan (*policy*) untuk mengelola bagian - bagian yang berbeda pada sistem.

## A. CloudSim *Architecture*

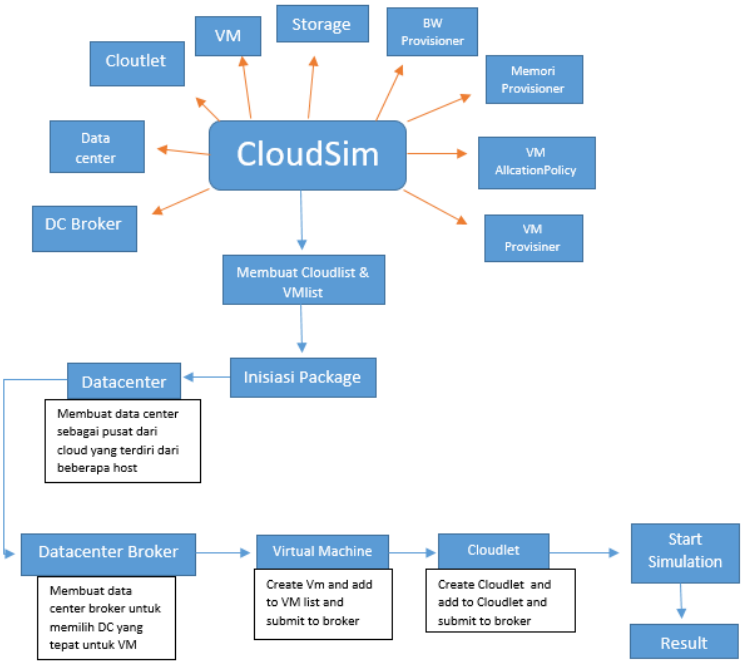
Pada Gambar 3 dijelaskan setiap komponen CloudSim. Setiap *layer* pada CloudSim menyediakan pemodelan dan simulasi terhadap virtualisasi *cloud* termasuk manajemen dalam VM, *memory*, *bandwidth*, dan *storage* [10]. CloudSim juga mendukung pengalokasian *host* untuk VM. Pada *layer* atas memaparkan dasar entitas *host* seperti jumlah *host* dan spesifikasi *host*, aplikasi, VM, jumlah pengguna, tipe aplikasi dan *broker scheduling*.




**Gambar 3 Arsitektur CloudSim**

## B. CloudSim Data *Flow*

Pada Gambar 4 dijelaskan setiap tahapan simulasi yang dilakukan oleh CloudSim dengan bantuan beberapa komponen CloudSim.




**Gambar 4 CloudSim *Data Flow***

Berikut adalah beberapa komponen yang terdapat dalam CloudSim, yang dapat dilihat pada Gambar 4:

**a. *Data Center***

*Data center* merupakan pusat dari *cloud* yang terdiri dari beberapa *host*, yang mampu mengalokasikan *bandwidth*, *memory* dan *storage*.

**b. *Data Center Broker***

*Broker* memiliki peranan sebagai mediator antara pengguna dan penyedia layanan, dimana *broker* memilih *Data center* yang tepat terhadap VM.

**c. *SANStorage***

*SanStorage* merupakan penyimpanan data yang digunakan oleh *data center*. Namun pengaksesan data pada saat pengelolaan *simulator* akan menyebabkan penundaan pelaksanaan transfer data melalui jaringan internal.

**d. *Virtual Machine***

*Virtual Machine* dikelola oleh *host*, sehingga setiap VM juga memiliki akses ke dalam komponen yang menyimpan karakteristik yang terkait dengan VM, misalnya seperti *memory*, *storage*, *processor* dan juga penjadwalan VM yang disebut VM *scheduling.*

**e. *Cloudlet***

*Cloudlet* merupakan sebuah data yang akan dikirimkan ke VM.

**f. *BWProvisioner***

Merupakan sebuah kebijakan yang mengalokasikan *bandwidth* untuk semua VM pada *data center*.

**g. *MemoryProvisioner***

Semua VM dijalankan dan disebar dalam sebuah *host*, jika *MemoryProvisioner* menentukan bahwa *host* memiliki ketersediaan memori yang cukup sesuai dengan yang diinginkan oleh VM.

**h. VM*Provisioner***

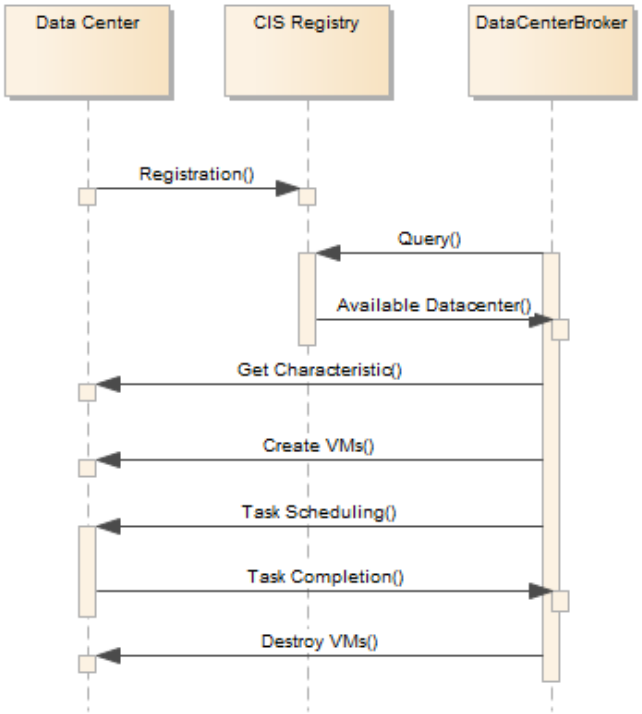
VM*Provisioner* bertanggung jawab untuk mengalokasikan VM ke setiap *host*.

**i. VM*AllocationPolicy***

Pengimplementasian komponen *host* untuk pemodelan kebijakan pembagian ruang dan waktu. Kebijakan ini diperlukan untuk mengalokasikan pengolahan *host* untuk VM.

### C. Komunikasi Antar Entitas CloudSim

Komunikasi yang terjadi dalam CloudSim dapat dilihat pada Gambar 5.




**Gambar 5 Komunikasi Antar Entitas**

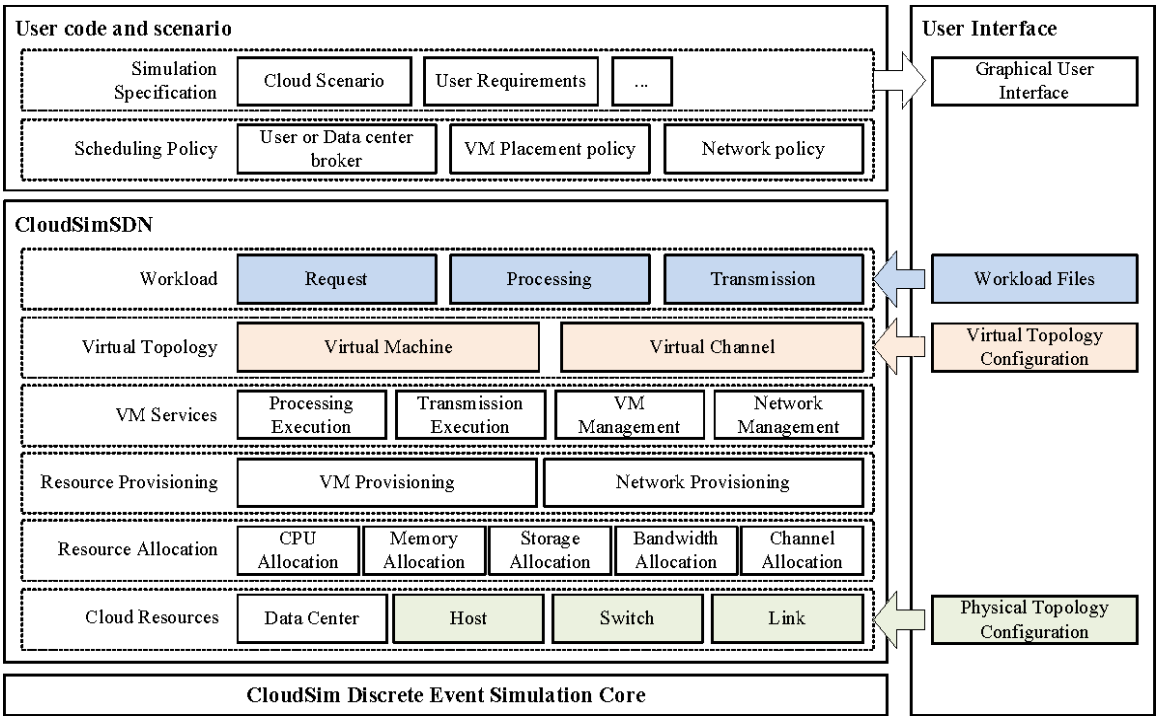
Pada saat simulasi dimulai, *data center* mendaftar ke CIS (*Cloud Service Information*). CIS berperan sebagai *mediator* untuk menyesuaikan pengguna yang sesuai dengan penyedia *Cloud Service*. Dalam hal ini, *broker* bertindak untuk mewakili pengguna menemukan *cloud* yang sesuai dengan kebutuhan pengguna. Jika penyedia *cloud* sudah sesuai, maka broker akan menyebarkan aplikasi ke *data center* yang dipilih.

### 2. CloudSimSDN

Permintaan untuk *scalable* dan efisiensi biaya pada *cloud computing* telah menjadi alasan kemunculan dari SDN (*Software Defined Networking*). CloudSimSDN adalah program *Simulator* yang dibangun dari *Simulator* CloudSim yang telah dikembangkan. CloudSimSDN adalah ektensi dari CloudSim. Perbedaan mendasar antara kedua *simulator* tersebut adalah CloudSimSDN dapat melakukan simulasi pengaturan dalam *joint allocation* pada *compute* dan *network resource.* CloudSimSDN dapat melakukan simulasi pada *cloud data center, switch, network, virtual topology*, dan *physical machines* dengan tujuan untuk mengukur konsumsi energi yang digunakan untuk meningkatkan ramah lingkungan dan pengurangan biaya dengan tujuan untuk tercapainya *energy network* yang efisien.

### A. Arsitektur CloudSimSDN

Pada Gambar 6, ditampilkan arsitektur CloudSimSDN yang terdiri dari enam lapisan, dimana yang pertama adalah dua lapisan untuk *File Workload* dan Konfigurasi *Virtual Topology*. Sementara itu, pada lapisan terakhir ditujukan untuk mengakomodasi Konfigurasi *Physical Topology.* [11]




**Gambar 6 Arsitektur CloudSimSDN [12]**

### 1. *Physical Topology (Data Center Configuration)*

Konfigurasi *host*, *switch*, dan *link* yang terdapat dalam *data center* dengan kemampuan SDN. Konfigurasi ini dapat menginput sebagai *fil e* JSON.

```
{
    "nodes":[
        {
            "ram":10240,
            "pes":16,
            "name":"h_0_0",
            "bw":1000000000,
            "mips":4000,
            "type":"host",
            "storage":10000000
        }
    ]
}
```

**Gambar 7 Contoh *file* JSON yang mendeskripsikan *host***

Pada Gambar 7, menjelaskan contoh karakteristik *host* dalam *physical topology*. Dalam sebuah *physical topology* yang lengkap, mencakup *host*, *switch*, dan *link*. VM akan dialokasikan ke *host*, kemudian paket-paket jaringan akan dikirim ke *host* lain melalui

*switch*. *Link* akan menghubungkan *host* ke *switch* atau *switch* ke *switch* menggunakan *bandwidth* dengan jumlah tertentu.

**2. *Virtual Topology (Resource Deployment Request)***

Dalam *virtual topology* dapat menginput VM dan *virtual links*, VM yang dijelaskan adalah komputasi daya, memori, dan ukuran penyimpanan. *Virtual Links* menghubungkan beberapa VM dengan *bandwidth*. Dalam *virtual topology* dapat mengkonfigurasi VM dan *Virtual Channel*.


**Gambar 8 Contoh *file* JSON yang mendeskripsikan VM**

Pada Gambar 8, menjelaskan karakteristik VM dalam *virtual topology*. Karakteristik tersebut adalah waktu VM memulai eksekusi sebuah aplikasi dan akhir dari waktu simulasi. Selain itu, dalam melakukan eksekusi, VM membutuhkan *memory*, PE, *bandwidth*, dengan ukuran tertentu.

**3. *Workload***

Setelah VM dibuat dalam *data center*, komputasi dan transmisi jaringan *workload* dari pengguna yang dikirimkan ke VM untuk diproses. *Workload* dapat melakukan penghitungan setiap pengolahan dan transmisi jaringan. *Workload* memiliki sebuah transmisi paket antara VM pada virtual jaringan yang sama. *Workload* dapat menginput sebagai *file* CSV. Karena, peneliti harus mengukur konsumsi daya yang digunakan oleh *switch*, maka transmisi data antara VM diperlukan untuk membiarkan *switch* tetap bekerja dalam waktu percobaan. *File workload* dalam CloudSimSDN adalah *request, processing,* dan *transmission.* Secara teknis, dalam CloudSimSDN, setiap VM dianggap sebagai jenis tertentu dari sebuah aplikasi. Sebuah komunikasi VM berkomunikasi dengan membentuk pengelompokan aplikasi. Pengelompokan tersebut disebut sebagai *Tier*.

Ada 4 jenis *Tier* dalam VM, yaitu:

1. *Tier* 1 terdiri dari VM dengan jenis aplikasi seperti “web” dan “app”
2. *Tier* 2 terdiri dari VM dengan jenis aplikasi seperti “web”, “app”, dan “db”
3. *Tier* 3 terdiri dari VM dengan jenis aplikasi seperti “web”, “app”, “db” dan “proxy”
4. *Tier* 4 terdiri dari VM dengan jenis aplikasi seperti “web”, “app”, “db”, “proxy” dan “firewall”

Selain dua konfigurasi infrastruktur yang telah disebutkan sebelumnya, yaitu *physical topology* dan *virtual topology*. Pada Gambar 9, ditampilkan contoh CSV *File*.

```
start, source, z, w1, link, dest, psize, w2
0.0,web0,0,1,default,app0,1000000000000000,1
```

**Gambar 9 Contoh *file* CSV yang mendeskripsikan *workload***

Struktur *file workload* terdiri dari waktu memulai transmisi, sumber VM, ukuran paket transmisi ke sumber VM, perhitungan *workload* untuk ke sumber VM, nama *virtual link* untuk mentransfer paket untuk tujuan VM, VM target, ukuran data transmisi target VM, dan komputasi *workload* untuk target VM. Selain itu, untuk VM *Placement* pada *physical machine* (*host*) dan bagaimana aliran jaringan dikomunikasikan VM pada *physical topology*, VM dialokasikan pada jaringan yang diperlukan. Setelah VM ditempatkan pada *host* yang sesuai, dan komunikasi didirikan pada VM yang telah dihubungkan oleh *link*. *Workload* dijalankan untuk menjalankan transmisi paket dari sumber ke VM target (antara *host* ke *host* dimana sumber dan tujuan VM dialokasikan) melalui *switch* yang dipilih. Dari proses ini, total konsumsi energi DC dapat ditemukan dengan menjumlahkan total konsumsi energi yang dimanfaatkan *host* dan *switch*.

### 2.1.4 Hasil yang diharapkan dalam simulasi CloudSimSDN

Agar simulasi dapat berjalan berjalan dengan baik, maka dibutuhkan 3 *file* seperti *physical topology*, *virtual topology* dan *workload*. Pada Gambar 10, ditampilkan jumlah *power consumption* dan *utilization level* dalam setiap *host* dan *switch*. Dan diharapkan hasil total energi yang digunakan seperti Gambar 11.

```
Host #0: 29653.168930555563
0.0, 4000.0
0.0, 16000.0
0.0, 35200.0
0.0, 51200.0
2390.0, 55200.0
2423.0, 59200.0
...
Switch #103: 27511.461264316662
22660.21001, 2
90180.21001, 3
502117.0, 2
1458312.66651, 0
```

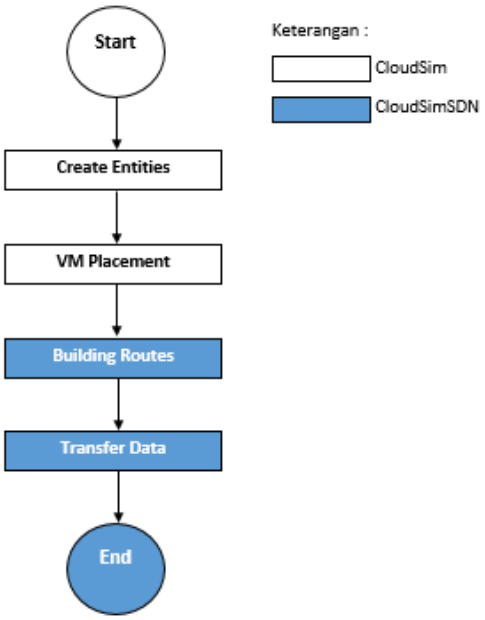
**Gambar 10 *Power Consumption dan Utilization Level***

```
========== TOTAL POWER CONSUMPTION ===========
Host energy consumed: 1848038.3846250002
Switch energy consumed: 92493.37391543222
Total energy consumed: 1940531.7585404324
Simultanously used hosts:30
```

**Gambar 11 Hasil Simulasi**

### 2.1.5 Tahapan dalam CloudSim dan CloudSimSDN

CloudSim dan CloudSimSDN adalah sebuah *simulator* yang mensimulasikan lingkungan berbasis *cloud*. Namun, keduanya memiliki beberapa tahapan simulasi yang berbeda. Pada Gambar 12, ditampilkan tahapan simulasi untuk kedua *Simulator* tersebut. Kotak putih mewakili tahapan yang dimiliki oleh CloudSim, sedangkan kotak biru mewakili tahapan yang ada dalam CloudSimSDN.




**Gambar 12 Tahapan dalam CloudSim dan CloudSimSDN**

Pada tahap pertama dari Gambar 12, dibuat sebuah entitas untuk *Simulator*, sebagai contohnya adalah *data center, broker, host,* VM, *Network Operating System* (NOS), dan SDN *Host*. NOS dan SDN *Host* merupakan entitas dalam CloudSimSDN. Dalam tahap kedua, penempatan VM dimulai setelah semua entitas dibuat kemudian VM dialokasikan ke dalam *host*. Dalam CloudSimSDN, VM tidak hanya diciptakan dalam *host* tetapi dalam SDN *Host* juga. Penempatan VM (VM *Placement*) merupakan tahapan terakhir dalam CloudSim, sedangkan untuk *Simulator* CloudSimSDN tahapan akan dilanjutkan. Pada tahap *building routes*, jaringan aliran routing akan dilakukan dalam NOS. Karena, semua proses jaringan dikelola dalam NOS. Tahap terakhir adalah melakukan transfer paket terhadap paket-paket yang telah siap untuk ditransfer melalui saluran dengan mengikuti *routing* yang sudah didefenisikan dalam tahapan sebelumnya yaitu *building routes*.

### 2.1.6 *Power Model*

*Data center* adalah bagian penting dari internet dan *cloud computing.* Dikarenakan semakin besarnya penggunaan energi elektrikal pada *data center*, maka diperlukan penghitungan jumlah energi yang sangat besar. Pada bagian ini, dijelaskan power model yang digunakan untuk menghitung penggunaan energi *host* dan *switch* pada *data center.* Satuan yang sering digunakan untuk menilai efisiensi energi adalah *Power (P)*, *Energy (E),* dan Periode (T). *Power* (P) adalah jumlah energi yang dikonsumsi per detik. Sementara, *Energy* (E) adalah total jumlah power yang dikonsumsi selama periode waktu (T) [13]

$$\mathbf{E = P.T}$$

*Energy* memiliki satuan *Joule* (*watt second*). Sementara Power memiliki satuan *watt* dan *Time* memiliki satuan detik. Cara menentukan perhitungan waktu adalah *end time* (t1) – *start time* (t1) seperti rumus berikut:

$$(\Delta t = t1 - t0)$$

### 2.1.7 Konsumsi Energi pada *Host* dan *Switch*

*Host* dan *switch* adalah komponen utama pada *data center*. Tugas akhir ini berfokus pada mendapatkan efisiensi energi dari *host* dan *switch*. Konsumsi energi pada *host* dihitung berdasarkan utilisasi CPU. Kemudian *Simulator* akan utilisasi CPU pada *host* tersebut akan ditampilkan dengan MIPS (Million Instructions per Second). CloudSimSDN menyediakan formula untuk menghitung *energy consumption* pada setiap *host* dengan cara menghitung persentasi utilisasi MIPS (total penggunaan MIPS dibagi jumlah MIPS pada *host*). Dan menghitung waktu antara *start time* dengan *end time* pada saat deploy VM

$$EHi = \sum (Pi * Ti)$$

Ada beberapa faktor untuk memperoleh konsumsi energi pada *switch*, seperti:

1. Jumlah port yang tersedia pada setiap *switch*
2. Konsumsi power pada setiap port yang aktif
3. Konsumsi power pada saat idle, idle yang dimaksudkan adalah pada saat tidak ada koneksi yang melewati *switch*.
4. Durasi *switch* pada saat *off*, jika tidak ada koneksi selama waktu tertentu.

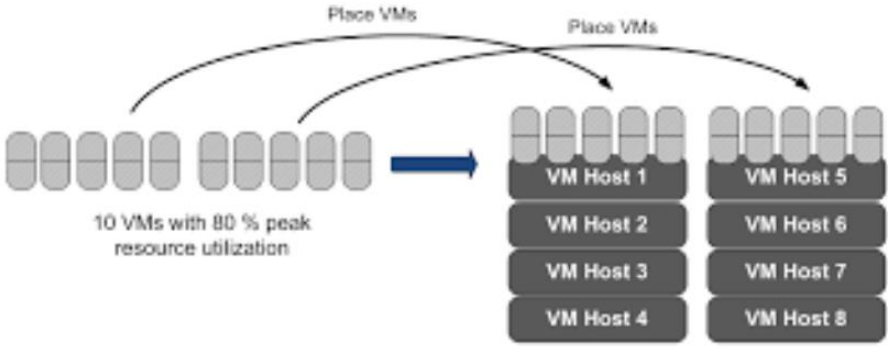
$$P = PowerConsumptionIdle + PowerConsumptionPerPort * numberOfActivePrt$$

Durasi waktu dibutuhkan untuk menghitung energi pada *switch*, dengan cara menghitung delta waktu dari transmisi data pengiriman paket sampai ke node yang dituju. Maka, formula yang dapat digunakan untuk menghitung energi pada *switch* adalah sebagai berikut:

$$ESi = \sum (Pi * Ti)$$

### 2.1.8 VM *Placement*

Pada saat *Virtual Machine* melakukan *deploy* ke *host*, maka dilakukan proses pemilihan VM yang paling tepat untuk *host* tersebut. Proses ini dikenal dengan *Virtual Machine Placement*. Pada saat dilakukan penempatan VM (VM Placement), VM akan dinilai berdasarkan kebutuhan *hardware* dan *resource requirements* untuk *host* tersebut. Proses penempatan VM dapat dilihat pada Gambar 13.




**Gambar 13 VM *Placement***

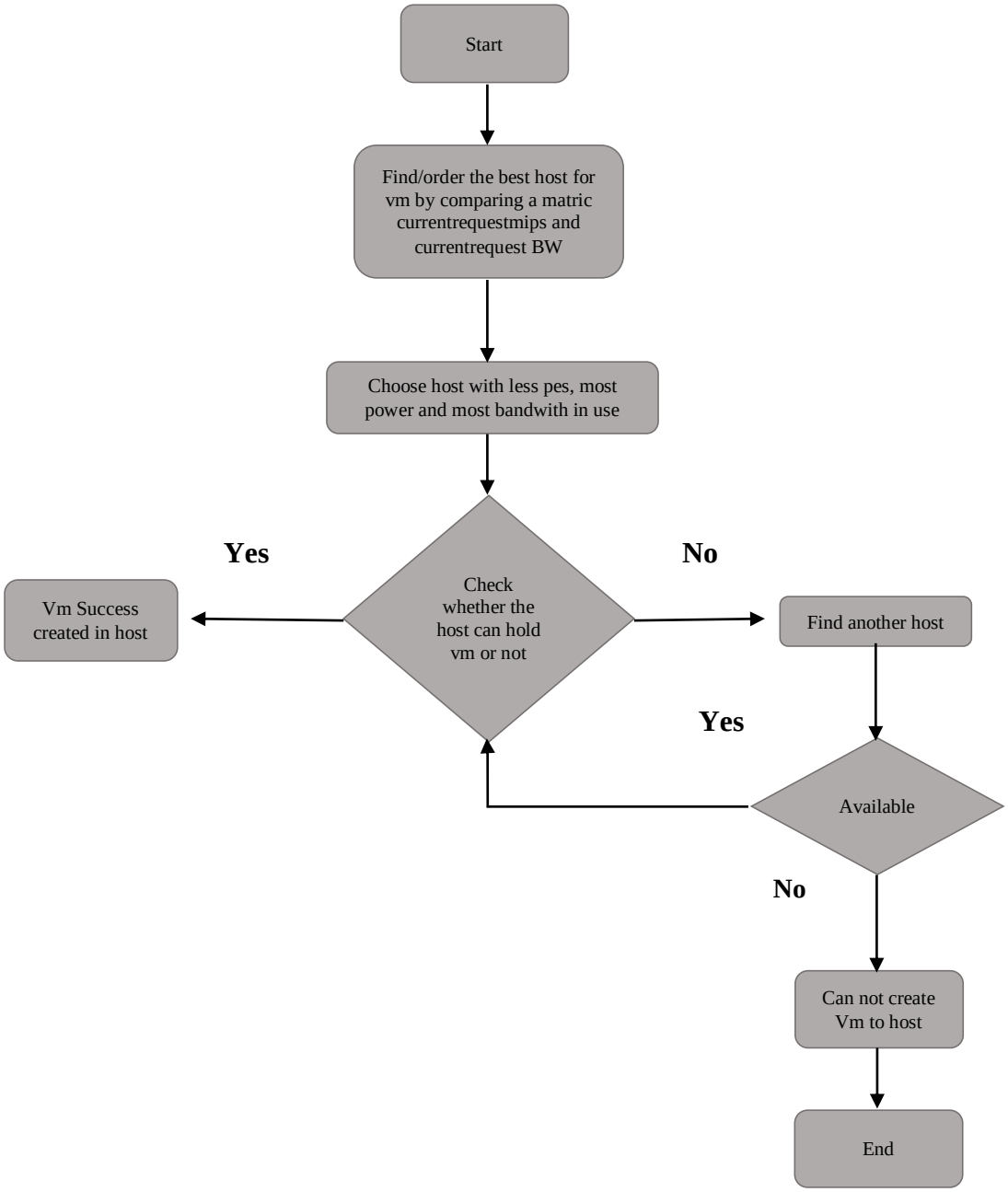
Program CloudSim memiliki sebuah *class* untuk melakukan VM *Placement* yang sering disebut dengan VMAllocationPolicy. Karena, pada sebuah *host* terdiri dari beberapa PE. PE (*Processing Element*) adalah CPU *core* pada *physical machine*. PE terdiri dari jutaan instruksi per detik. PE didefinisikan dalam MIPS (*Million Instruction per Second*) yaitu

ukuran kecepatan *processor* pada PE. VMAllocationPolicy memiliki dua algoritma dalam memilih *host,* yaitu:

**1. *Bestfit***

Algoritma *bestfit* adalah sebuah algoritma yang memilih *host* yang paling meminimalisir penggunaan PE (*Processing Element*). Algoritma *bestfit* juga disebut sebagai algoritma *Most Full First* (MFF). Algoritma *bestfit* juga memilih *host* dengan jumlah sumber daya terkecil. [10] Algoritma *bestfit* berfungsi untuk memilih *host* yang paling tepat untuk memenuhi permintaan terhadap VM, tentunya *host* tersebut memiliki sumber daya yang sesuai dengan persyaratan terhadap permintaan VM. [9] Dalam pendekatan ini, VM cenderung digabungkan ke sejumlah *host* yang lebih kecil, dan jaringan antara *host* dapat dikurangi karena lebih banyak VM yang ditempatkan dalam *host*.

Sehingga, setiap permintaan VM yang datang, diharapkan menggunakan *bestfit* agar dapat meminimalisir penggunaan *host*. Dalam penggunaan algoritma *bestfit*, setiap permintaan VM yang datang, semua VM dialokasikan kedalam satu *host* hingga mampu memenuhi persyaratan sesuai kebijakan *bestfit*, kemudian *host* tersebut dipilih. Cara kerja tersebut dilakukan sampai keseluruh *host*. Algoritma bestfit memilih host yang paling lengkap dalam hal power dan bandwidth. Cara kerja algoritma *bestfit* dapat dilihat pada Gambar 14 berikut:

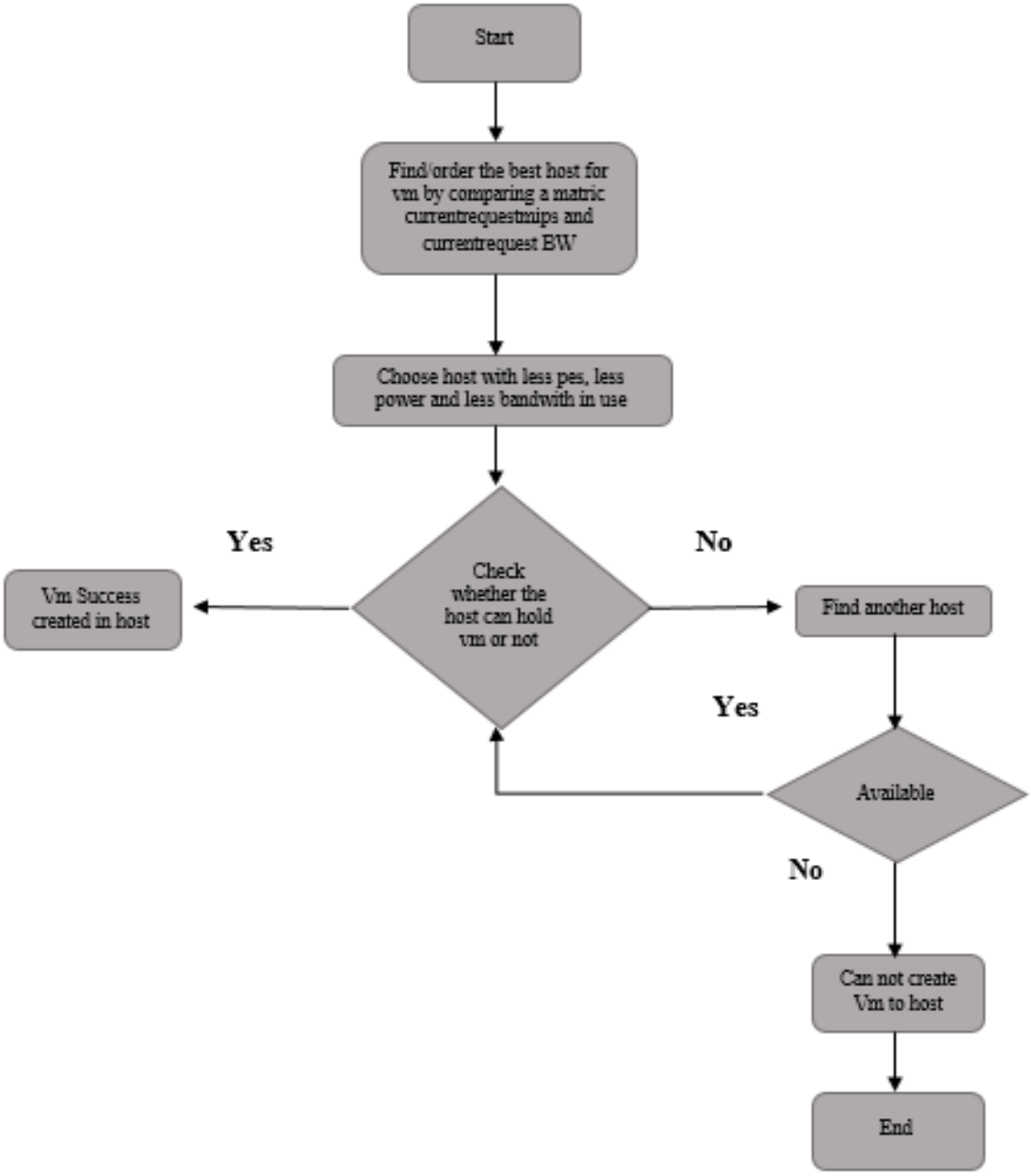



**Gambar 14 *Flowchart* Algoritma *Bestfit***

## 2. *Worstfit*

Algoritma *Worstfit* adalah sebuah algoritma memilih *host* yang menggunakan PE dengan jumlah yang lebih banyak dibandingkan dengan *bestfit*. Algoritma *bestfit* juga disebut sebagai algoritma *Least Full First* (LFF). Algoritma *worsfit* memilih *host* dengan jumlah sumber daya yang paling besar. [10] *Worstfit* berfungsi untuk memilih *host* yang memiliki

kapasitas sumber daya maksimum, sehingga dapat memaksimalkan daya komputasi. [9] Cara kerja algoritma *bestfit* dapat dilihat pada Gambar 15 berikut:




**Gambar 15 *Flowchart* Algoritma *Worstfit***

# BAB III

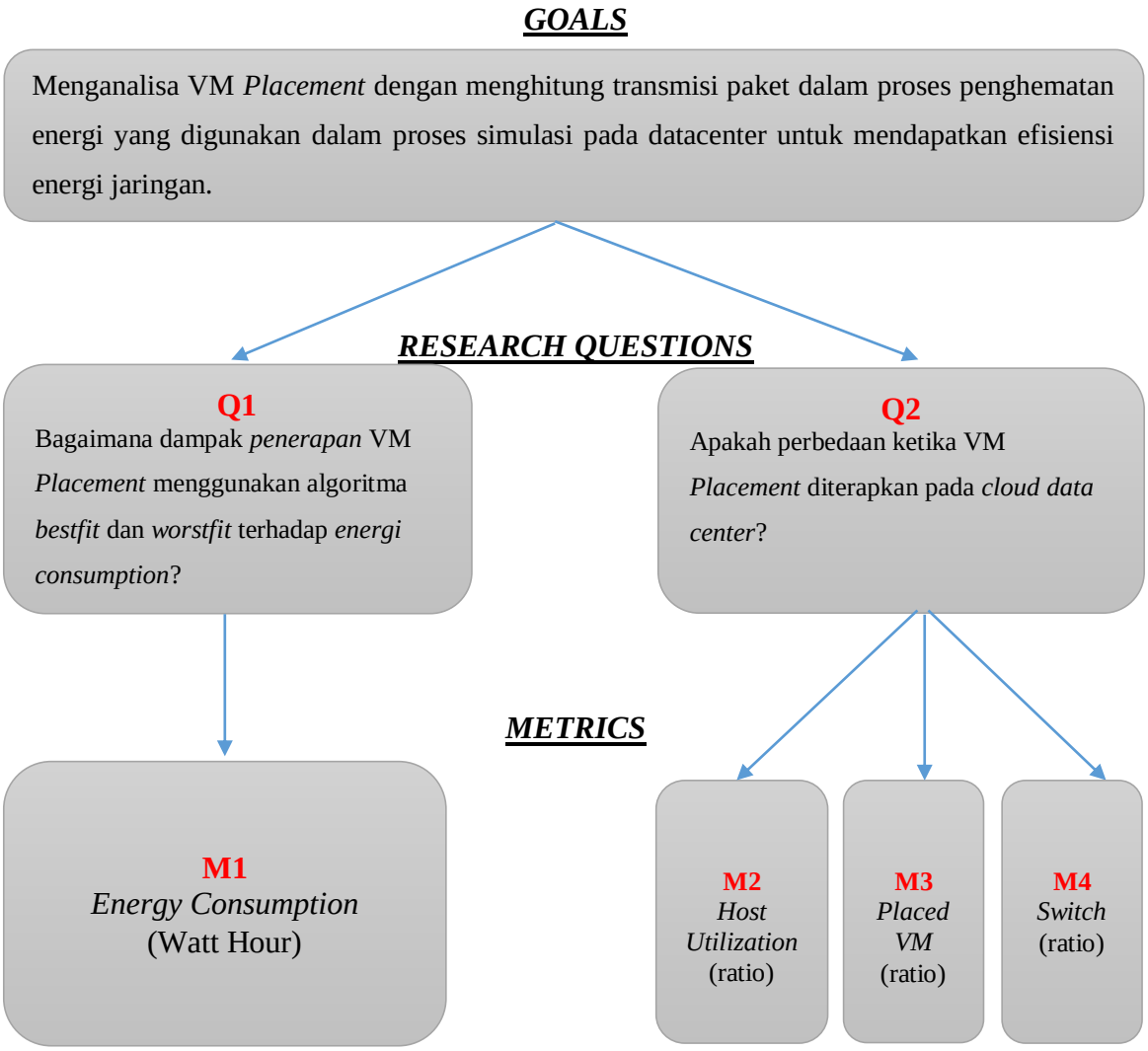
# Analisis dan Penelitian

## 3.1 *Experiment Definition*

Pada bagian ini menjelaskan mengenai pembahasan hubungan antara pertanyaan utama dengan metric untuk mencapai tujuan dalam Tugas Akhir ini.

### 3.1.1 *Goal Question Metric Tree*

Pada *Goal Question Metric Tree* pada Gambar 16 menjelaskan mengenai pertanyaan dan metric yang digunakan untuk mencapai tujuan dari penelitian ini.




**Gambar 16 *GQM Metrics***

### 3.1.2 Target Penelitian

Bagian ini menjelaskan target yang dicapai dari penelitian yang didefinisikan sebagai berikut:

**Menganalisa VM *Placement* dengan menghitung transmisi paket dalam proses penghematan energi yang digunakan dalam proses simulasi pada datacenter untuk mendapatkan efisiensi energi jaringan.**

Berdasarkan target diatas, eksperimentasi didorong oleh pertanyaan-pertanyaan penelitian berikut (*Research Questions*):

1. Bagaimana dampak penerapan VM *Placement* menggunakan algoritma *bestfit* dan *worstfit* terhadap *energy consumption*?
2. Apakah perbedaan ketika VM *Placement* diterapkan pada *cloud data center*?

Pertanyaan penelitian ini dijawab dengan menghitung tingkat konsumsi energi jaringan sistem yang diimplementasikan dengan algoritma VM *Placement* pada CloudSim dan CloudSimSDN yang digunakan untuk mengukur penggunaan energi jaringan. Sedangkan, metrik digunakan untuk menjawab kedua pertanyaan tersebut yang terkait dengan energi metrik yang mengukur konsumsi daya dari waktu ke waktu.

### 3.1.3 *Metric*

*Metric* yang mengukur setiap pertanyaan diatas, seperti berikut:

**1. [M1] *Energy Consumption***

Konsumsi energi berasal dari metrik yang mengukur konsumsi energi oleh *switch* dan *host* yang diukur dari waktu ke waktu. Waktu pelaksanaan diukur dalam detik ($\Delta$t), sehingga unit metrik adalah joule (*watt hour*).

**2. [M2] *Host* Utilization**

Dengan pemanfaatan *host*, dapat mengukur jumlah *physical machine* yang digunakan untuk menempatkan semua VM.

**3. [M3] *Placed* VM**

Penempatan VM yang sesuai dengan VM pada *physical machine*. Unit pengukuran untuk metric ini adalah rasio.

**4. [M4] *Switch***

*Switch* merupakan sebuah perangkat jaringan pada komputer yang menghubungkan perangkat dengan menggunakan transmisi paket untuk menerima, memproses dan meneruskan data ke perangkat yang dituju. Selain itu, *switch* digunakan sebagai penghubung antar *host*.

## 3.2 *Experiment Planning*

Pada bagian ini, menjelaskan mengenai perencanaan penelitian yang dilakukan dalam Tugas Akhir ini. Pada penelitian ini, beberapa alur dijalankan dalam dua konfigurasi yang berbeda dengan menganalisis *Data Center Network Architecture Fat-Tree* dengan menggunakan CloudSimSDN. Setelah itu, hasil perhitungan akan diuji pada hasil akhir untuk menjawab semua pertanyaan penelitian.

### 3.2.1 Cara Kerja *Fat-Tree Topology*

Sebagai contoh dalam cara kerja *Fat-Tree* adalah sebagai berikut:

Pada Gambar 2, menunjukkan contoh *topology Fat-Tree* dengan n = 4 dengan 3 lapisan *topology* yaitu *edge*, *aggregation* dan *core*. Lapisan *core* memiliki $\left(\frac{n}{2}\right)^2$ = 4 *switch,* lapisan *aggregation* memiliki *core switch* x 2 = 8 *switch,* dan lapisan *edge* memiliki *core switch* x 2 = 8 *switch.* Ada 4 *pod* yang dimana masing-masing memiliki 4 *host* yang menghasilkan N = $\frac{n^3}{4}$ = 16 *host.* Jumlah *pod* sama dengan jumlah *port* yaitu n = 4. Sebagai contoh perhitungan *topology Fat-Tree* dapat dilihat pada Tabel 2 berikut:

**Tabel 2 Contoh Perhitungan *Fat-Tree***

| N | *Edge* | Aggregation | *Core* | *Pods* | *Hosts* |
|---|---|---|---|---|---|
| 4 | 8 | 8 | 4 | 4 | 16 |
| 6 | 18 | 18 | 9 | 6 | 54 |
| 36 | 648 | 648 | 324 | 36 | 11664 |
| 46 | 1058 | 1058 | 529 | 46 | 24334 |

### 3.2.2 *Simulation Scenario*

Untuk menjawab kedua pertanyaan dalam penelitian yang akan dilakukan oleh penulis. Maka, dibuat skenario berbeda untuk mewakili setiap pertanyaan pada subbab 3.1.2. Oleh karena itu, skenario ini dimaksudkan untuk mengumpulkan data yang berbeda tergantung pada metrik penelitian. Pada pengujian yang akan dilakukan, terdapat 2 skenario seperti berikut:

1. **[SC1]** Skenario pertama mengakomodasi pengujian terhadap penggunaan VM *Placement* terhadap *Data Center Network Architecture.*
2. **[SC2]** Skenario kedua mengakomodasi pengukuran *energy consumption* pada *Data Center Network Architecture*.

Pemetaan skenario untuk pertanyaan penelitian dan metrik digambarkan pada Tabel 3.

**Tabel 3 Pemetaan Pertanyaan dan Metrik Penelitian**

| Questions | Scenario | Metrics |
|---|---|---|
| [Q1] Bagaimana dampak penerapan VM *Placement* menggunakan algoritma *bestfit* dan *worstfit* terhadap *energy consumption*? | SC1 | M1, M2, M3, dan M4 |
| [Q2] Apakah perbedaan ketika VM *Placement* diterapkan pada *cloud data center*? | SC2 | M1, M2, M3, dan M4 |

### 3.2.3 *Testbed Configuration*

Kecepatan konfigurasi *link* antara *core* dengan *edge switch*, dan antara *edge switch* dengan *host* ditampilkan pada Tabel 4.

**Tabel 4 *Link* Konfigurasi Percobaan Validasi**

| ***Link*** | ***Bandwidth*** |
|---|---|
| *Core* ↔ *Edge Switch*es | 10 Mbps |
| *Edge Switch*es ↔ *Host*s | 10 Mbps |

Pada Tabel 5, ditunjukkan skenario yang ada dalam transmisi yang ditetapkan untuk memulai transmisi pada saat yang sama. Oleh karena itu, jika ukuran data tidak sama, maka beberapa transmisi akan berakhir lebih awal dari transmisi lain, kemudian *link* yang sama dengan transmisi akan dihentikan, dengan memiliki *bandwidth* yang lebih banyak selama sisa koneksi.

**Tabel 5 Skenario untuk Validasi**

| *Scenario* | *Packet Size* |
|---|---|
| Skenario 1 | 10 MB |
| Skenario 2 | Bervariasi dalam distribusi yang seragam. (a= 10 MB, b = 20 MB) |

### 3.2.4. Skenario Pengujian

Delapan skenario yang digunakan dalam pengujian ini dapat dilihat pada Tabel 6.

**Tabel 6 Skenario Pengujian**

| Scenario Configuration | Algoritma *Bestfit* | Algoritma *Worstfit* |
|---|---|---|
| **Skenario 1.A & 1. B** | 10 MB *Data Size* | 10 MB *Data Size* |
| **Skenario 1.C & 1. D** | 10 MB *Data Size* | 10 MB *Data Size* |
| **Skenario 2.A & 2. B** | *Random* | *Random* |
| **Skenario 2.C & 2. D** | *Random* | *Random* |

Sementara itu, Virtual Machine yang digunakan dalam pengujian menggunakan konfigurasi yang dapat dilihat pada Tabel 7.

**Tabel 7 Konfigurasi VM**

| Items | Parameters |
|---|---|
| *MIPS* | 1000 |
| *Cores* | *Random* |
| *RAM* | *Random* |

### 3.2.4 *Scenario Parameters*

Untuk menguji skenario yang ada pada Tabel 6, berbagai parameter dirancang pada Tabel 8. Terdapat dua ukuran yang berbeda dari *host*, yaitu 16 dan 54 *host*. Masing-masing *host* memiliki 8 *core* dan 16 GB RAM. Karakteristik VM, setiap VM memerlukan jumlah *core* dalam rentang 1, 2, dan 4. Dan beberapa parameter yang dapat dilihat pada Tabel 8.

**Tabel 8 *Scenario Parameters***

| Items | Parameters |
|---|---|
| *Number of Cloud Service* (CS) | 10 |
| *Number of Host* | 16 , 54 |
| *Number of Max* VM | 30 |
| *Host* RAM (GB) | 16 |
| *Host Core* | 8 |
| VM *Core* | 1, 2, 4 |
| VM RAM | 1024, 2048, 4096 |
| *Number of Switch Port* | 4,6 |
| *Network Topology* | *Fat-Tree* |
| *Algorithm* | *Worst Fit* dan *Best Fit* |

### 3.2.5. Spesifikasi *Hardware* dan *Software*

Pada bagian ini, *hardware* dan *software* yang digunakan untuk menjalankan skenario simulasi adalah sebagai berikut:

**A. Spesifikasi *Hardware***

Alat yang digunakan pada penelitian ini dapat dilihat pada Tabel 9.

**Tabel 9 Spesifikasi *Hardware***

| Resources | Specification |
|---|---|
| Physical Machine | Lenovo G50 |
| CPU | Intel *Core* i5-5200U |
| Memory | 8 GB |
| HDD | 1000 GB |
| *Network* Adapter | Qualcomm Atheros Communications Inc |
| OS | Windows 10 Pro 64-bit |

**B. Deskripsi *Software***

*Software* yang digunakan yaitu CloudSim versi 4.0 dan CloudSimSDN. CloudSim memiliki kemampuan untuk mengukur setiap jumlah *host* yang digunakan, dan penempatan VM yang sesuai dengan VM *Placement*. Sedangkan, CloudSimSDN memiliki kemampuan untuk mengukur setiap konsumsi energi dan semua dan data VM *Placement*.

# BAB IV

# Implementasi dan Pengujian

## 4.1 Implementasi

Berdasarkan tahapan simulasi yang dilakukan oleh CloudSim seperti yang sudah dirancang pada subbab 3.2. Beberapa tahapan pengujian tersebut diimplementasikan sebagai berikut:

### 4.1.1 Create *Data Center*

Pada tahap pembuatan *Data center*, dibutuhkan beberapa *host*. *Host* terdiri dari spesifikasi seperti RAM, PE, bandwidth, MIPS, dan storage. Spesifikasi tersebut diimplementasikan pada file PhysicalTopolofyGenerator.java sebagai berikut:

```
public HostSpec createHostSpec(int pe, long mips, int ram, long storage,
long bw) {
      return new HostSpec(pe, mips, ram, storage, bw);
}
```

Spesifikasi *host* pada pembuatan *data center* diimplementasikan dengan jumlah sebagai berikut:

```
int pe = 8;
long mips = 2000;
int ram = 10240;
long storage = 10000000;
long bw = 1000000000;
```

Pada file JSON Physical Topology, spesifikasi *host* terdiri dari beberapa jenis spesifikasi sebagai berikut:

```
{
      "name": "host01",
      "type" : "host",
      "pes" : 1,
      "mips" : 30000000,
      "ram" : 10240,
      "storage" : 10000000,
      "bw" : 12500000,
    },

{
```

```
      "name": "host02",
      "type" : "host",
      "pes" : 1,
      "mips" : 30000000,
      "ram" : 10240,
      "storage" : 10000000,
      "bw" : 12500000,
    },
```

**a. *Set Data center***

Proses pembuatan *data center* terdapat pada file SDNExample.java. *Data center* memiliki karakteristik seperti arch, os, VMm, *hostList*, time_zone, cost, costPerMem, costPerStorage, costPerBw. [12]

```
/**
 * Creates the Data center.
 *
 * @param name the name
 *
 * @return the Data center
 */
      protected static NetworkOperatingSystem nos;
      protected static PowerUtilizationMaxHostInterface
      maxHostHandler = null;
      protected static SDNData center createSDNData center(String
      name, String physicalTopology, NetworkOperatingSystem snos,
      VMAllocationPolicyFactory VMAllocationFactory) {
      // In order to get Host information, pre-create NOS.
            nos=snos;
            List<Host> hostList = nos.getHostList();

            String arch = "x86"; // system architecture
            String os = "Linux"; // operating system
            String VMm = "Xen";

            double time_zone = 10.0;
              // time zone this resource located
            double cost = 3.0;
              // the cost of using processing in this resource
```

```
            double costPerMem = 0.05;
              // the cost of using memory in this resource
            double costPerStorage = 0.001;
              // the cost of using storage in this

            //resource
            double costPerBw = 0.0;
               // the cost of using bw in this resource
            LinkedList<Storage> storageList = new
            LinkedList<Storage>();
               // we are not adding SAN

            // devices by now

      Data centerCharacteristics characteristics = new Data
      centerCharacteristics(
             arch, os, VMm, hostList, time_zone, cost, costPerMem,
             costPerStorage, costPerBw);
}
```

Pada gambar diatas terdapat karakteristik *Data center* yang terdiri dari *host list*, maka untuk menciptakan *hostlist*, penulis harus melakukan tahapan konfigurasi *host* pada *Physical Topology*.

### 4.1.2 *Physical Topology*

Pada *physical topology*, terdapat *host*, *switch* dan link. VM/VMs akan dialokasikan untuk *host*, kemudian paket akan dikirim dari satu *host* ke *host* lain melalui *switch*. Link akan menghubungkan *host switch* atau beralih untuk *switch* dengan bandwidth dengan jumlah tertentu. *Host*, *switch*, dan *link* akan dikonfigurasi didalam *physical topology*. Konfigurasi tersebut diimplementasikan kedalam beberapa bagian berikut:

**1. Konfigurasi *Host***

Konfigurasi *host* adalah proses pembuatan *host* sesuai dengan spesifikasi *host* yang telah ditentukan dalam pembuatan *data center*. Berikut merupakan implementasi konfigurasi *host* pada file PhysicalTopologyGenerator.java

```
class HostSpec extends NodeSpec {
      int pe;
```

```
	long mips;
	int ram;
	long storage;

	@SuppressWarnings("unchecked")
	JSONObject toJSON() {
		HostSpec o = this;
		JSONObject obj = new JSONObject();
		obj.put("name", o.name);
		obj.put("type", o.type);
		obj.put("storage", o.storage);
		obj.put("pes", o.pe);
		obj.put("mips", o.mips);
		obj.put("ram", new Integer(o.ram));
		obj.put("bw", o.bw);
		return obj;
	}

public HostSpec(int pe, long mips, int ram, long storage, long bw)
	{
		this.pe = pe;
		this.mips = mips;
		this.ram = ram;
		this.storage = storage;
		this.bw = bw;
		this.type = "host";
	}
}
```

## 2. Konfigurasi *Switch*

Selain dilakukan konfigurasi *host*, dalam *physical topology* terdapat konfigurasi *switch* yang akan memberikan output pada file JSON.

```
class SwitchSpec extends NodeSpec {

            long iops;

            @SuppressWarnings("unchecked")

            JSONObject toJSON() {

                  SwitchSpec o = this;

                  JSONObject obj = new JSONObject();

                  obj.put("name", o.name);

                  obj.put("type", o.type);

                  obj.put("iops", o.iops);

                  obj.put("bw", o.bw);

                  return obj;

            }

      }
```

### 3. Konfigurasi Link

Pada *physical topology*, link berfungsi sebagai penghubung antara *host* dengan *switch*.

```
class LinkSpec {
            String source;
            String destination;
            double latency;

public LinkSpec(String source,String destination,double latency2) {
            this.source = source;
            this.destination = destination;
            this.latency = latency2; }
            @SuppressWarnings("unchecked")
            JSONObject toJSON() {
                  LinkSpec link = this;
                  JSONObject obj = new JSONObject();
                  obj.put("source", link.source);
                  obj.put("destination", link.destination);
                  obj.put("latency", link.latency);
```

```
                return obj;
            }
      }
```

### 4.1.3 Pembentukan *Topology Fat-Tree*

Pada Physical Topology terjadi pembentukan topology *Fat-Tree*, dimana untuk membentuk sebuah *topology Fat-Tree* dibutuhkan beberapa karakteristik pendukung seperti *host specification*, IOPS, *bandwidth*, *pod*, dan *latency*. Berikut merupakan implementasi pembentukan *topology Fat-Tree*. Pada implementasi *topology Fat Tree*, pengujian dilakukan tanpa memperhitungkan latency dan bandwidth. Pembentukan *topology Fat-Tree* diimplementasikan pada file PhysicalTopologyGenerator.java.

```
public void createTopologyFatTree(HostSpec hostSpec, long swIops, long
swBw, int numpods, double latency) {
      SwitchSpec [][] c = new SwitchSpec [numpods/2][numpods/2];
            for(int i=0; i<numpods/2; i++) {
                  for(int j=0; j<numpods/2; j++) {
                     c[i][j]  =  addSwitch("c_"+i+"_"+j,  "core",  swBw,
swIops);
                  }
            }

            for(int k=0; k<numpods; k++) {
                  SwitchSpec [] e = new SwitchSpec[numpods/2];
                  SwitchSpec [] a = new SwitchSpec[numpods/2];

      for(int i=0; i<numpods/2; i++) {
            e[i] = addSwitch("e_"+k+"_"+i, "edge", swBw, swIops);
            a[i] = addSwitch("a_"+k+"_"+i, "aggregate", swBw, swIops);
            addLink(a[i], e[i], latency);

            for(int j=0; j<i; j++) {
                  addLink(a[i], e[j], latency);
                  addLink(a[j], e[i], latency);
            }

            for(int j=0; j<numpods/2; j++) {
                  addLink(a[i], c[i][j], latency);
            }
```

```
            for(int j=0; j<numpods/2; j++) {
                  String hostname = "h_" + k+"_"+ i + "_" + j;
                  HostSpec h = addHost(hostname, hostSpec);
                  addLink(e[i], h, latency);
                        }
                  }
            }
      }
```

### 4.1.4 *Create Data Center Broker*

Proses terbentuknya *data center* broker melalui beberapa tahapan, dimulai dari penentuan *Virtual Machine* (VM) List, pembentukan cloudlist, selanjutnya VM membuat permintaan data center, hingga memberikan data center yang tepat kepada VM. Sehingga dari tahapan tersebut, sehingga dapat diketahui bahwa tugas dari *data center* broker adalah memilih *Data center* yang tepat untuk VM. Pembentukan *broker* diimplementasikan pada file SDNExample.java seperti berikut:

```
/**
 * Creates the broker.
 *
 * @return the Data center broker
 */
protected static SDNBroker createBroker() {
      SDNBroker broker = null;
      try {
            broker = new SDNBroker("Broker");
      } catch (Exception e) {


            e.printStackTrace();
            return null;
      }
      return broker;
}
```

### 4.1.5 Virtual Topology

Pada *virtual topology*, diperlukan beberapa komponen seperti VM, link dan workload. Link sendiri berfungsi sebagai penghubung pengalokasian VM ke *host*. Pengalokasian VM ke *host* tersebut menggunakan algoritma Least Full First (LFF) dan Most Full First (MFF). Sehingga VM yang paling sesuai yang diberikan kepada *host* sesuai dengan kebutuhan *host*. Pembentukan *virtual topology* dapat dilihat pada *file* VirtualTopologyGenerator.java

```
public class VirtualTopologyGenerator {

private List<VMSpec> VMs = new ArrayList<VMSpec>();

private List<LinkSpec> links = new ArrayList<LinkSpec>();

private List<DummyWorkloadSpec> dummyWorkload = new
ArrayList<DummyWorkloadSpec>();



public VMSpec addVM(String name, VMSpec spec) {

      return addVM(name, spec.pe, spec.mips, spec.ram, spec.size,
spec.bw, spec.starttime, spec.endtime);

}

public VMSpec addVM(String name, int pes, long mips, int ram, long
storage, long bw, double starttime, double endtime) {

      VMSpec VM = new VMSpec(pes, mips, ram, storage, bw, starttime,
endtime);

      VM.name = name;



      VMs.add(VM);

      return VM;

}



public LinkSpec addLink(String linkname, VMSpec source, VMSpec dest,
Long bw) {

LinkSpec link = new LinkSpec(linkname, source.name,dest.name, bw);
```

```
        links.add(link);



        addWorkload(linkname, source, dest);

        return link;

}

public void addWorkload(String linkname, VMSpec source, VMSpec dest)
{

        DummyWorkloadSpec wl = new DummyWorkloadSpec(source.starttime,
                source.name,dest.name, linkname);

        this.dummyWorkload.add(wl);

}



public VMSpec createVMSpec(int pe, long mips, int ram, long storage,
long bw, double starttime, double endtime) {

        return new VMSpec(pe, mips, ram, storage, bw, starttime,
                endtime);

}
```

### 4.1.6 *Workload*

Pada *workload* terjadi transmisi paket antar VM. *Workload* sendiri memiliki karakteristik seperti *start time*, *link* dan penghitungan total energi yang digunakan selama transmisi paket antar VM yang dapat dilihat pada *file* Workload.java:

```
public class Workload implements Comparable<Workload> {
        public int workloadId;
        public int appId;
        public double time;
        public int submitVmId;
        public int submitPktSize;
        public Request request;

        public WorkloadResultWriter resultWriter;
```

```
	public Workload(int workloadId, WorkloadResultWriter writer) {
		this.workloadId = workloadId;
		this.resultWriter = writer;
	}

	public void writeResult() {
		this.resultWriter.writeResult(this);
	}

	@Override
	public int compareTo(Workload that) {
		return this.workloadId - that.workloadId;
	}
}
```

### 4.1.7 *VM Allocation Policy*

Pada CloudSimSDN terdapat pengalokasian VM yang tepat kepada *host* dikenal dengan *Virtual Machine Placement*. Proses pengalokasian VM menggunakan VMAllocationPolicy. Implementasi VM allocation policy adalah sebagai berikut:

```
/ Create Data center with previously set parameters
SDNData center Data center = null;
	try {
	  VMAllocationPolicy VMPolicy
	  VMAllocationFactory.create(hostList);
	  maxHostHandler =
	   (
	     PowerUtilizationMaxHostInterface)VMPolicy;
	   Data center = new SDNData center(name, characteristics,
	   VMPolicy, storageList, 0, nos);
	   nos.setData center(Data center);}
	       catch (Exception e)
	   {
	       e.printStackTrace();
	   }
```

```
                return Data center;
            }
```

VM Allocation Policy terdiri dari 2 jenis algoritma untuk memilih *host* yang paling tepat, dua algoritma tersebut adalah sebagai berikut:

**1. MFF (*Most Full First*)**

Algoritma MFF merupakan algoritma yang dikenal sebagai algoritma *bestfit*. Dimana algoritma *bestfit* memilih PE yang paling sedikit. Algoritma MFF dapat dilihat pada *file* VMAllocationPolicyCombinedMostFullFirst.java

```
public VMAllocationPolicyCombinedMostFullFirst(List<? extends Host> list) {
       super(list);

       setFreePes(new ArrayList<Integer>());
       setFreeMips(new ArrayList<Long>());
       setFreeBw(new ArrayList<Long>());

       for (Host host : getHostList()) {
              getFreePes().add(host.getNumberOfPes());
              getFreeMips().add((long)host.getTotalMips());
              getFreeBw().add(host.getBw());
       }
       hostTotalMips = getHostList().get(0).getTotalMips();
       hostTotalBw =  getHostList().get(0).getBw();
       hostTotalPes =  getHostList().get(0).getNumberOfPes();

       setVMTable(new HashMap<String, Host>());
       setUsedPes(new HashMap<String, Integer>());
       setUsedMips(new HashMap<String, Long>());
       setUsedBw(new HashMap<String, Long>());
}
```

**2. LFF (*Least Full First*)**

Algoritma LFF merupakan algoritma yang dikenal sebagai algoritma *Worstfit*. Dimana algoritma *Worstfit* memilih PE yang paling banyak. Algoritma LFF dapat dilihat pada *file* VMAllocationPolicyCombinedLeastFullFirst.java

```
public class VMAllocationPolicyCombinedLeastFullFirst extends
VMAllocationPolicyCombinedMostFullFirst{
```

```
	public VMAllocationPolicyCombinedLeastFullFirst(List<? extends Host>
list) {
		super(list);
	}
```

### 4.1.8 *Energy Consumption*

Sesuai dengan rancangan penilitian pada Bab 3, untuk melakukan penghitungan energi yang digunakan dalam melakukan transmisi paket, peneliti menggunakan power model. Implementasi power model pada file PowerUtilizationEnergyModelHostLinear.java seperti berikut:

```
public double calculateEnergyConsumption(double duration, double
utilization) {

	double power = calculatePower(utilization);

	double energyConsumption = power * duration;


// Assume that the host is turned off when duration is long enough

	if(duration > powerOffDuration && utilization == 0)

		energyConsumption = 0;

	return energyConsumption / 3600;

}
```

### 4.1.9 Waktu Pengujian

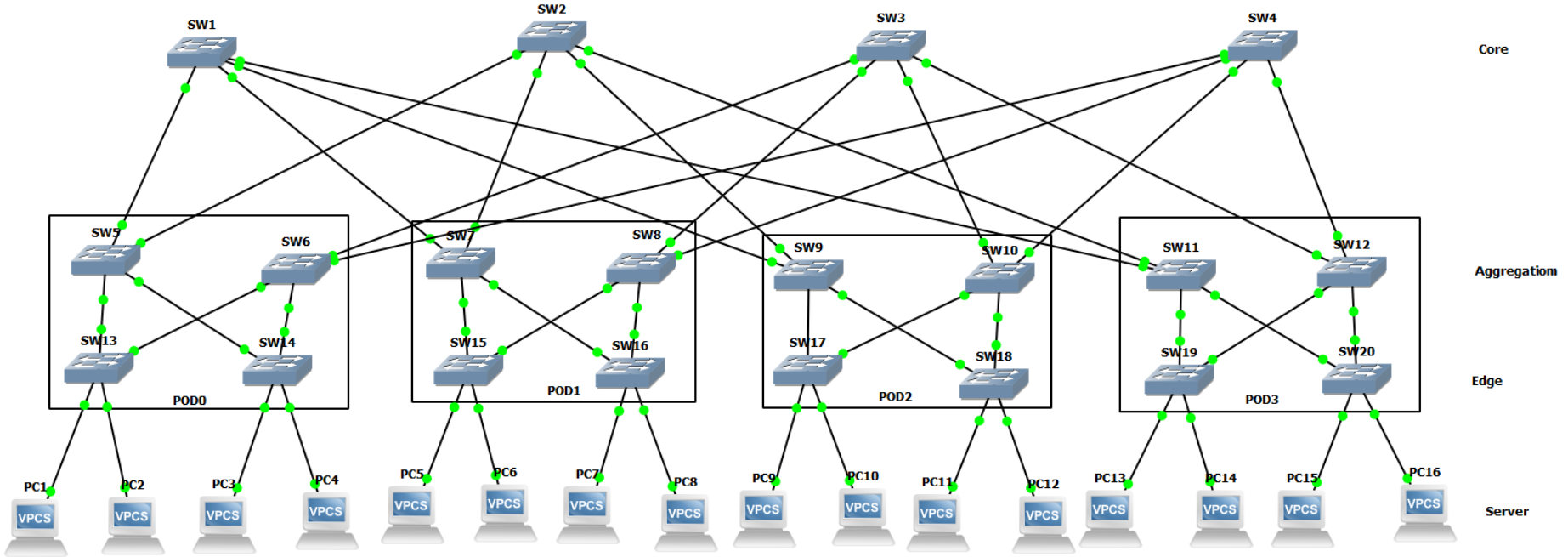
Penggunaan jumlah *energy* pada periode waktu tertentu dapat dihitung menggunakan rumus: waktu berakhirnya pengujian (t1) – waktu memulai pengujian (t0) seperti yang sudah dirancang pada subbab 2.1.6. Implementasi perhitungan dapat dilihat pada tabel berikut:

```
public double addPowerConsumption(double currentTime, double
cpuUtilizationOfLastPeriod) {

		double duration = currentTime - previousTime;
```

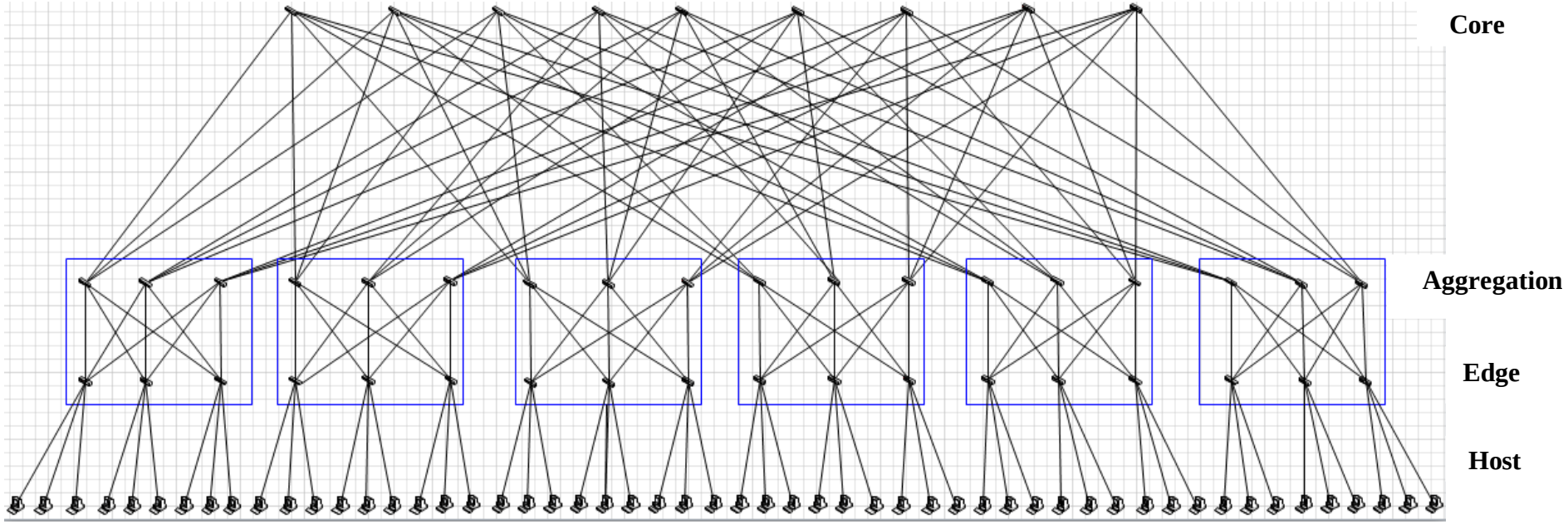
```
            double                    energyConsumption                    =
energyModel.calculateEnergyConsumption(duration,
cpuUtilizationOfLastPeriod);

            totalEnergy += energyConsumption;
            previousTime = currentTime;

            return energyConsumption;
      }
```

## 4.2 Pengujian

Penulis melakukan pengujian dengan melakukan perhitungan energi dengan menggunakan implementasi seperti pada subbab 4.1. Pengujian menggunakan skenario yang sudah dirancang pada subbab 3.2.2 pada Tabel 5. Berikut merupakan gambar *topology Fat-Tree* menggunakan 16 *host* dan 54 *host* yang digunakan pada pengujian:




**Gambar 17 *Fat-Tree Topology* 16 *Host***




**Gambar 18 *Fat-Tree Topology* 54 *Host***

### 4.2.1 Hasil Pengujian *Physical Topology*

Pada *physical topology*, dilakukan pengujian terhadap *edge*, *aggregation*, *core* dan *host*. Selain itu, pada *physical topology* terjadi pembuatan *data center*. *Physical topology* merupakan hasil seleksi antar node. Hasil pengujian dapat dilihat pada beberapa bagian berikut:

**1. Pengujian 1**

Skenario pertama menggunakan SC 1 pada subbab 3.2.3 yaitu pada skenario validasi Tabel 5 dan menggunakan 16 *host*, maka diperoleh hasil *physical topology* dengan sebagai berikut:

**Tabel 10 Hasil *Physical Topology* menggunakan Pengujian 1**

| **Menggunakan 16 Host [SC1]** |
|---|
| **1. h ditujukan untuk *host*.**<br>**h15** ↔ *Switch*: e7 <-> SDNHost 15<br>**h14** ↔ *Switch*: e7 <-> SDNHost 14<br>**h13** ↔ *Switch*: e6 <-> SDNHost 13<br>**h12** ↔ *Switch*: e6 <-> SDNHost 12<br>**h11** ↔ *Switch*: e5 <-> SDNHost 11<br>**h10** ↔ *Switch*: e5 <-> SDNHost 10<br>**h9** ↔ *Switch*: e4 <-> SDNHost 9<br>**h8** ↔ *Switch*: e4 <-> SDNHost 8<br>**h7** ↔ *Switch*: e3 <-> SDNHost 7<br>**h6** ↔ *Switch*: e3 <-> SDNHost 6<br>**h5** ↔ *Switch*: e2 <-> SDNHost 5<br>**h4** ↔ *Switch*: e2 <-> SDNHost 4<br>**h3** ↔ *Switch*: e2 <-> SDNHost 3<br>**h2** ↔ *Switch*: e1 <-> SDNHost 2<br>**h1** ↔ *Switch*: e1 <-> SDNHost 1<br>**h0** ↔ *Switch*: e0 <-> SDNHost 0 |
| **2. e ditujukan untuk *edge***<br>**a. *edge* 7**<br>- *Switch*: e7 <-> SDNHost 14<br>- *Switch*: a7 <-> *Switch*: e7<br>- *Switch*: a6 <-> *Switch*: e7<br>- *Switch*: e7 <-> SDNHost 15<br>**b. *edge* 6**<br>- *Switch*: a7 <-> *Switch*: e6<br>- *Switch*: a6 <-> *Switch*: e6<br>- *Switch*: e6 <-> SDNHost 12<br>- *Switch*: e6 <-> SDNHost 13<br>**c. *edge* 5**<br>- *Switch*: a4 <-> *Switch*: e5<br>- *Switch*: a5 <-> *Switch*: e5 |

- *Switch*: e5 <-> SDNHost 10
- *Switch*: e5 <-> SDNHost 11

**d. *edge* 4**

- *Switch*: a4 <-> *Switch*: e4
- *Switch*: a5 <-> *Switch*: e4
- *Switch*: e4 <-> SDNHost 9
- *Switch*: e4 <-> SDNHost 8

**e. *edge* 3**

- *Switch*: e3 <-> SDNHost 6
- *Switch*: a3 <-> *Switch*: e3
- *Switch*: a2 <-> *Switch*: e3
- *Switch*: e3 <-> SDNHost 7

**f. *edge* 2**

- *Switch*: a2 <-> *Switch*: e2
- *Switch*: a3 <-> *Switch*: e2
- *Switch*: e2 <-> SDNHost 5
- *Switch*: e2 <-> SDNHost 4

**g. *edge* 1**

- *Switch*: a0 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: e1 <-> SDNHost 2
- *Switch*: e1 <-> SDNHost 3

**h. *edge* 1**

- *Switch*: a0 <-> *Switch*: e0
- *Switch*: a1 <-> *Switch*: e0
- *Switch*: e0 <-> SDNHost 1
- *Switch*: e0 <-> SDNHost 0

**3. a ditujukan untuk *aggregation***

**a. *aggregation* 7**

- *Switch*: a7 <-> *Switch*: e7
- *Switch*: c2 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a7
- *Switch*: a7 <-> *Switch*: e7
- *Switch*: a7 <-> *Switch*: e6
- *Switch*: a7 <-> *Switch*: e6

**b. *aggregation* 6**

- *Switch*: a6 <-> *Switch*: e7
- *Switch*: c1 <-> *Switch*: a6

- *Switch*: c0 <-> *Switch*: a6
- *Switch*: a6 <-> *Switch*: e7
- *Switch*: a6 <-> *Switch*: e6
- *Switch*: a6 <-> *Switch*: e6

**c. *aggregation* 5**

- *Switch*: c2 <-> *Switch*: a5
- *Switch*: c3 <-> *Switch*: a5
- *Switch*: a5 <-> *Switch*: e5
- *Switch*: a5 <-> *Switch*: e4
- *Switch*: a5 <-> *Switch*: e5
- *Switch*: a5 <-> *Switch*: e4

**d. *aggregation* 4**

- *Switch*: c0 <-> *Switch*: a4
- *Switch*: c1 <-> *Switch*: a4
- *Switch*: a4 <-> *Switch*: e5
- *Switch*: a4 <-> *Switch*: e4
- *Switch*: a4 <-> *Switch*: e5
- *Switch*: a4 <-> *Switch*: e4

**e. *aggregation* 3**

- *Switch*: a3 <-> *Switch*: e3
- *Switch*: c3 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: a3 <-> *Switch*: e2
- *Switch*: a3 <-> *Switch*: e2
- *Switch*: a3 <-> *Switch*: e3

**f. *aggregation* 2**

- *Switch*: a2 <-> *Switch*: e3
- *Switch*: c1 <-> *Switch*: a2
- *Switch*: c0 <-> *Switch*: a2
- *Switch*: a2 <-> *Switch*: e2
- *Switch*: a2 <-> *Switch*: e2
- *Switch*: a2 <-> *Switch*: e3

**g. *aggregation* 1**

- *Switch*: c2 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e0
- *Switch*: a1 <-> *Switch*: e0

### h. *aggregation* 0

- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: a0 <-> *Switch*: e1
- *Switch*: a0 <-> *Switch*: e1
- *Switch*: a0 <-> *Switch*: e0
- *Switch*: a0 <-> *Switch*: e0

## 4. c ditujukan untuk *core*

### a. *core* 3

- *Switch*: c3 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a3
- *Switch*: c3 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a3
- *Switch*: c3 <-> *Switch*: a3
- *Switch*: c3 <-> *Switch*: a3
- *Switch*: c3 <-> *Switch*: a5
- *Switch*: c3 <-> *Switch*: a5
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a5
- *Switch*: c3 <-> *Switch*: a5
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a7

### b. *core* 2

- *Switch*: c2 <-> *Switch*: a7
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a7
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a5
- *Switch*: c2 <-> *Switch*: a5
- *Switch*: c2 <-> *Switch*: a5
- *Switch*: c2 <-> *Switch*: a1
- *Switch*: c2 <-> *Switch*: a1
- *Switch*: c2 <-> *Switch*: a1
- *Switch*: c2 <-> *Switch*: a7

- *Switch*: c2 <-> *Switch*: a5
- *Switch*: c2 <-> *Switch*: a5
- *Switch*: c2 <-> *Switch*: a1
- *Switch*: c2 <-> *Switch*: a7

### c. *core* 1

- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a2
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a2
- *Switch*: c1 <-> *Switch*: a2
- *Switch*: c1 <-> *Switch*: a2
- *Switch*: c1 <-> *Switch*: a4
- *Switch*: c1 <-> *Switch*: a4
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a4
- *Switch*: c1 <-> *Switch*: a4
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a6

### d. *core* 0

- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a2
- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a2
- *Switch*: c0 <-> *Switch*: a2
- *Switch*: c0 <-> *Switch*: a2
- *Switch*: c0 <-> *Switch*: a4
- *Switch*: c0 <-> *Switch*: a4
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a4
- *Switch*: c0 <-> *Switch*: a4
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a6

**2. Pengujian 1**

Skenario pertama menggunakan SC 1 pada subbab 3.2.3 yaitu pada skenario validasi Tabel 5 dan menggunakan 16 *host*, maka diperoleh hasil *physical topology* dengan sebagai berikut:

| **Menggunakan 54 Host [SC1]** |
|---|
| **1. h ditujukan untuk *host*.** |
| **h53** ↔ *Switch*: e17 <-> SDNHost 53 |
| **h52** ↔ *Switch*: e17 <-> SDNHost 52 |
| **h51** ↔ *Switch*: e17 <-> SDNHost 51 |
| **h50** ↔ *Switch*: e16 <-> SDNHost 50 |
| **h49** ↔ *Switch*: e16 <-> SDNHost 49 |
| **h48** ↔ *Switch*: e16 <-> SDNHost 48 |
| **h47** ↔ *Switch*: e15 <-> SDNHost 47 |
| **h46** ↔ *Switch*: e15 <-> SDNHost 46 |
| **h45** ↔ *Switch*: e15 <-> SDNHost 45 |
| **h44** ↔ *Switch*: e14 <-> SDNHost 44 |
| **h43** ↔ *Switch*: e14 <-> SDNHost 43 |
| **h42** ↔ *Switch*: e14 <-> SDNHost 42 |
| **h41** ↔ *Switch*: e13 <-> SDNHost 41 |
| **h40** ↔ *Switch*: e13 <-> SDNHost 40 |
| **h39** ↔ *Switch*: e13 <-> SDNHost 39 |
| **h38** ↔ *Switch*: e12 <-> SDNHost 38 |
| **h37** ↔ *Switch*: e12 <-> SDNHost 37 |
| **h36** ↔ *Switch*: e12 <-> SDNHost 36 |
| **h35** ↔ *Switch*: e11 <-> SDNHost 35 |
| **h34** ↔ *Switch*: e11 <-> SDNHost 34 |
| **h33** ↔ *Switch*: e11 <-> SDNHost 33 |
| **h32** ↔ *Switch*: e10 <-> SDNHost 32 |
| **h31** ↔ *Switch*: e10 <-> SDNHost 31 |
| **h30** ↔ *Switch*: e10 <-> SDNHost 30 |
| **h29** ↔ *Switch*: e9 <-> SDNHost 29 |
| **h28** ↔ *Switch*: e9 <-> SDNHost 28 |
| **h27** ↔ *Switch*: e9 <-> SDNHost 27 |
| **h26** ↔ *Switch*: e8 <-> SDNHost 26 |
| **h25** ↔ *Switch*: e8 <-> SDNHost 25 |
| **h24** ↔ *Switch*: e8 <-> SDNHost 24 |
| **h23** ↔ *Switch*: e7 <-> SDNHost 23 |
| **h22** ↔ *Switch*: e7 <-> SDNHost 22 |

**h21** ↔ *Switch*: e7 <-> SDNHost 21

**h20** ↔ *Switch*: e6 <-> SDNHost 20

**h19** ↔ *Switch*: e6 <-> SDNHost 19

**h18** ↔ *Switch*: e6 <-> SDNHost 18

**h17** ↔ *Switch*: e5 <-> SDNHost 17

**h16** ↔ *Switch*: e5 <-> SDNHost 16

**h15** ↔ *Switch*: e5 <-> SDNHost 15

**h14** ↔ *Switch*: e4 <-> SDNHost 14

**h13** ↔ *Switch*: e4 <-> SDNHost 13

**h12** ↔ *Switch*: e4 <-> SDNHost 12

**h11** ↔ *Switch*: e3 <-> SDNHost 11

**h10** ↔ *Switch*: e3 <-> SDNHost 10

**h9** ↔ *Switch*: e3 <-> SDNHost 9

**h8** ↔ *Switch*: e2 <-> SDNHost 8

**h7** ↔ *Switch*: e2 <-> SDNHost 7

**h6** ↔ *Switch*: e2 <-> SDNHost 6

**h5** ↔ *Switch*: e1 <-> SDNHost 5

**h4** ↔ *Switch*: e1 <-> SDNHost 4

**h3** ↔ *Switch*: e1 <-> SDNHost 3

**h2** ↔ *Switch*: e0 <-> SDNHost 2

**h1** ↔ *Switch*: e0 <-> SDNHost 1

**h0** ↔ *Switch*: e0 <-> SDNHost 0

### 2. e ditujukan untuk *edge*

### a. *edge* 17

- *Switch*: a15 <-> *Switch*: e17
- *Switch*: a16 <-> *Switch*: e17
- *Switch*: a17 <-> *Switch*: e17
- *Switch*: e17 <-> SDNHost 52
- *Switch*: e17 <-> SDNHost 52
- *Switch*: e17 <-> SDNHost 53

### b. *edge* 16

- *Switch*: e16 <-> SDNHost 49
- *Switch*: a15 <-> *Switch*: e16
- *Switch*: a16 <-> *Switch*: e16
- *Switch*: a17 <-> *Switch*: e16
- *Switch*: e16 <-> SDNHost 50
- *Switch*: e16 <-> SDNHost 48

### c. *edge* 15

- *Switch*: e16 <-> SDNHost 48

- *Switch*: a16 <-> *Switch*: e15
- *Switch*: a15 <-> *Switch*: e15
- *Switch*: e15 <-> SDNHost 46
- *Switch*: e15 <-> SDNHost 47
- *Switch*: e15 <-> SDNHost 45

### d. edge 14

- *Switch*: a12 <-> *Switch*: e14
- *Switch*: a14 <-> *Switch*: e14
- *Switch*: a13 <-> *Switch*: e14
- *Switch*: e14 <-> SDNHost 43
- *Switch*: e14 <-> SDNHost 42
- *Switch*: e14 <-> SDNHost 44

### e. edge 13

- *Switch*: a14 <-> *Switch*: e13
- *Switch*: a13 <-> *Switch*: e13
- *Switch*: a12 <-> *Switch*: e13
- *Switch*: e13 <-> SDNHost 41
- *Switch*: e13 <-> SDNHost 40
- *Switch*: e13 <-> SDNHost 39

### f. edge 12

- *Switch*: a13 <-> *Switch*: e12
- *Switch*: a14 <-> *Switch*: e12
- *Switch*: a12 <-> *Switch*: e12
- *Switch*: e12 <-> SDNHost 36
- *Switch*: e12 <-> SDNHost 37
- *Switch*: e12 <-> SDNHost 38

### g. edge 11

- *Switch*: e12 <-> SDNHost 38
- *Switch*: a11 <-> *Switch*: e11
- *Switch*: a10 <-> *Switch*: e11
- *Switch*: e11 <-> SDNHost 33
- *Switch*: e11 <-> SDNHost 34
- *Switch*: e11 <-> SDNHost 35

### h. edge 10

- *Switch*: a9 <-> *Switch*: e10
- *Switch*: a10 <-> *Switch*: e10
- *Switch*: a11 <-> *Switch*: e10
- *Switch*: e10 <-> SDNHost 30
- *Switch*: e10 <-> SDNHost 31

- *Switch*: e10 <-> SDNHost 32

### i. edge 9

- *Switch*: a10 <-> *Switch*: e9
- *Switch*: a11 <-> *Switch*: e9
- *Switch*: a9 <-> *Switch*: e9
- *Switch*: e9 <-> SDNHost 28
- *Switch*: e9 <-> SDNHost 27
- *Switch*: e9 <-> SDNHost 29

### j. edge 8

- *Switch*: a8 <-> *Switch*: e8
- *Switch*: a7 <-> *Switch*: e8
- *Switch*: a6 <-> *Switch*: e8
- *Switch*: e8 <-> SDNHost 26
- *Switch*: e8 <-> SDNHost 24
- *Switch*: e8 <-> SDNHost 25

### k. edge 7

- *Switch*: a6 <-> *Switch*: e7
- *Switch*: a8 <-> *Switch*: e7
- *Switch*: a7 <-> *Switch*: e7
- *Switch*: e7 <-> SDNHost 22
- *Switch*: e7 <-> SDNHost 23
- *Switch*: e7 <-> SDNHost 21

### l. *edge* 6

- *Switch*: a7 <-> *Switch*: e6
- *Switch*: a6 <-> *Switch*: e6
- *Switch*: a8 <-> *Switch*: e6
- *Switch*: e6 <-> SDNHost 20
- *Switch*: e6 <-> SDNHost 19
- *Switch*: e6 <-> SDNHost 18

### m. *edge* 5

- *Switch*: a7 <-> *Switch*: e6
- *Switch*: a6 <-> *Switch*: e6
- *Switch*: a8 <-> *Switch*: e6
- *Switch*: e6 <-> SDNHost 20
- *Switch*: e6 <-> SDNHost 19
- *Switch*: e6 <-> SDNHost 18

### n. edge 4

- *Switch*: e5 <-> SDNHost 15
- *Switch*: a3 <-> *Switch*: e4

- *Switch*: a5 <-> *Switch*: e4
- *Switch*: e4 <-> SDNHost 12
- *Switch*: e4 <-> SDNHost 13
- *Switch*: e4 <-> SDNHost 14

### o. *edge* 3

- *Switch*: a5 <-> *Switch*: e3
- *Switch*: a4 <-> *Switch*: e3
- *Switch*: a3 <-> *Switch*: e3
- *Switch*: e3 <-> SDNHost 10
- *Switch*: e3 <-> SDNHost 9
- *Switch*: e3 <-> SDNHost 11

### p. *edge* 2

- *Switch*: a1 <-> *Switch*: e2
- *Switch*: a2 <-> *Switch*: e2
- *Switch*: a0 <-> *Switch*: e2
- *Switch*: e2 <-> SDNHost 6
- *Switch*: e2 <-> SDNHost 7
- *Switch*: e2 <-> SDNHost 8

### q. *edge* 1

- *Switch*: a0 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: a2 <-> *Switch*: e1
- *Switch*: e1 <-> SDNHost 4
- *Switch*: e1 <-> SDNHost 5
- *Switch*: e1 <-> SDNHost 3

### r. *edge* 0

- *Switch*: a0 <-> *Switch*: e0
- *Switch*: a1 <-> *Switch*: e0
- *Switch*: a2 <-> *Switch*: e0
- *Switch*: e0 <-> SDNHost 0
- *Switch*: e0 <-> SDNHost 2
- *Switch*: e0 <-> SDNHost 1

## 3. a ditujukan untuk *aggregation*

## 1. *aggregation* 17

- *Switch*: a17 <-> *Switch*: e16
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c8 <-> *Switch*: a17
- *Switch*: a17 <-> *Switch*: e15

- *Switch*: a17 <-> *Switch*: e17
- *Switch*: a17 <-> *Switch*: e17
- *Switch*: a17 <-> *Switch*: e16
- *Switch*: a17 <-> *Switch*: e15
- *Switch*: a17 <-> *Switch*: e17
- *Switch*: a17 <-> *Switch*: e15
- *Switch*: a17 <-> *Switch*: e16

### 2. *aggregation* 16

- *Switch*: a16 <-> *Switch*: e16
- *Switch*: c8 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: a16 <-> *Switch*: e15
- *Switch*: a16 <-> *Switch*: e17
- *Switch*: a16 <-> *Switch*: e17
- *Switch*: a16 <-> *Switch*: e16
- *Switch*: a16 <-> *Switch*: e15
- *Switch*: a16 <-> *Switch*: e17
- *Switch*: a16 <-> *Switch*: e15
- *Switch*: a16 <-> *Switch*: e16

### 3. *aggregation* 15

- *Switch*: a15 <-> *Switch*: e16
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a15
- *Switch*: a15 <-> *Switch*: e15
- *Switch*: a15 <-> *Switch*: e17
- *Switch*: a15 <-> *Switch*: e17
- *Switch*: a15 <-> *Switch*: e16
- *Switch*: a15 <-> *Switch*: e15
- *Switch*: a15 <-> *Switch*: e17
- *Switch*: a15 <-> *Switch*: e15
- *Switch*: a15 <-> *Switch*: e16

### 4. *aggregation* 14

- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c8 <-> *Switch*: a14
- *Switch*: a14 <-> *Switch*: e13
- *Switch*: a14 <-> *Switch*: e13
- *Switch*: a14 <-> *Switch*: e12

- *Switch*: a14 <-> *Switch*: e12
- *Switch*: a14 <-> *Switch*: e14
- *Switch*: a14 <-> *Switch*: e13
- *Switch*: a14 <-> *Switch*: e14
- *Switch*: a14 <-> *Switch*: e12
- *Switch*: a14 <-> *Switch*: e14

### 5. *aggregation* 13

- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c8 <-> *Switch*: a13
- *Switch*: a13 <-> *Switch*: e13
- *Switch*: a13 <-> *Switch*: e13
- *Switch*: a13 <-> *Switch*: e12
- *Switch*: a13 <-> *Switch*: e12
- *Switch*: a13 <-> *Switch*: e14
- *Switch*: a13 <-> *Switch*: e13
- *Switch*: a13 <-> *Switch*: e14
- *Switch*: a13 <-> *Switch*: e12
- *Switch*: a13 <-> *Switch*: e14

### 6. *aggregation* 12

- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: a12 <-> *Switch*: e13
- *Switch*: a12 <-> *Switch*: e13
- *Switch*: a12 <-> *Switch*: e12
- *Switch*: a12 <-> *Switch*: e12
- *Switch*: a12 <-> *Switch*: e14
- *Switch*: a12 <-> *Switch*: e13
- *Switch*: a12 <-> *Switch*: e14
- *Switch*: a12 <-> *Switch*: e12
- *Switch*: a12 <-> *Switch*: e14

### 7. *aggregation* 11

- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c8 <-> *Switch*: a11
- *Switch*: a11 <-> *Switch*: e11
- *Switch*: a11 <-> *Switch*: e10
- *Switch*: a11 <-> *Switch*: e10
- *Switch*: a11 <-> *Switch*: e9

- *Switch*: a11 <-> *Switch*: e9
- *Switch*: a11 <-> *Switch*: e10
- *Switch*: a11 <-> *Switch*: e9
- *Switch*: a11 <-> *Switch*: e9
- *Switch*: a11 <-> *Switch*: e11
- *Switch*: c2 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e0
- *Switch*: a1 <-> *Switch*: e0

### 8. *aggregation* 10

- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c8 <-> *Switch*: a10
- *Switch*: a10 <-> *Switch*: e11
- *Switch*: a10 <-> *Switch*: e10
- *Switch*: a10 <-> *Switch*: e10
- *Switch*: a10 <-> *Switch*: e9
- *Switch*: a10 <-> *Switch*: e9
- *Switch*: a10 <-> *Switch*: e10
- *Switch*: a10 <-> *Switch*: e9
- *Switch*: a10 <-> *Switch*: e11
- *Switch*: a10 <-> *Switch*: e11

### 9. *aggregation* 9

- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: a9 <-> *Switch*: e11
- *Switch*: a9 <-> *Switch*: e11
- *Switch*: a9 <-> *Switch*: e10
- *Switch*: a9 <-> *Switch*: e9
- *Switch*: a9 <-> *Switch*: e9
- *Switch*: a9 <-> *Switch*: e10
- *Switch*: a9 <-> *Switch*: e9
- *Switch*: a9 <-> *Switch*: e11
- *Switch*: a9 <-> *Switch*: e11

### 10. *aggregation* 8

- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c8 <-> *Switch*: a8

- *Switch*: c7 <-> *Switch*: a8
- *Switch*: a8 <-> *Switch*: e7
- *Switch*: a8 <-> *Switch*: e8
- *Switch*: a8 <-> *Switch*: e8
- *Switch*: a8 <-> *Switch*: e7
- *Switch*: a8 <-> *Switch*: e6
- *Switch*: a8 <-> *Switch*: e6
- *Switch*: a8 <-> *Switch*: e8
- *Switch*: a8 <-> *Switch*: e6
- *Switch*: a8 <-> *Switch*: e7

### 11. *aggregation* 7

- *Switch*: c4 <-> *Switch*: a7
- *Switch*: c5 <-> *Switch*: a7
- *Switch*: c8 <-> *Switch*: a7
- *Switch*: a7 <-> *Switch*: e7
- *Switch*: a7 <-> *Switch*: e8
- *Switch*: a7 <-> *Switch*: e8
- *Switch*: a7 <-> *Switch*: e7
- *Switch*: a7 <-> *Switch*: e6
- *Switch*: a7 <-> *Switch*: e6
- *Switch*: a7 <-> *Switch*: e8
- *Switch*: a7 <-> *Switch*: e6
- *Switch*: a7 <-> *Switch*: e7

### 12. *aggregation* 6

- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: a6 <-> *Switch*: e7
- *Switch*: a6 <-> *Switch*: e8
- *Switch*: a6 <-> *Switch*: e8
- *Switch*: a6 <-> *Switch*: e7
- *Switch*: a6 <-> *Switch*: e6
- *Switch*: a6 <-> *Switch*: e6
- *Switch*: a6 <-> *Switch*: e8
- *Switch*: a6 <-> *Switch*: e6
- *Switch*: a6 <-> *Switch*: e7

### 13. *aggregation* 5

- *Switch*: a5 <-> *Switch*: e5
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c8 <-> *Switch*: a5

- *Switch*: c7 <-> *Switch*: a5
- *Switch*: a5 <-> *Switch*: e3
- *Switch*: a5 <-> *Switch*: e3
- *Switch*: a5 <-> *Switch*: e5
- *Switch*: a5 <-> *Switch*: e5
- *Switch*: a5 <-> *Switch*: e4
- *Switch*: a5 <-> *Switch*: e3
- *Switch*: a5 <-> *Switch*: e4
- *Switch*: a5 <-> *Switch*: e4

### 14. *aggregation* 4

- *Switch*: a4 <-> *Switch*: e5
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c8 <-> *Switch*: a4
- *Switch*: a4 <-> *Switch*: e3
- *Switch*: a4 <-> *Switch*: e3
- *Switch*: a4 <-> *Switch*: e5
- *Switch*: a4 <-> *Switch*: e5
- *Switch*: a4 <-> *Switch*: e4
- *Switch*: a4 <-> *Switch*: e3
- *Switch*: a4 <-> *Switch*: e4
- *Switch*: a4 <-> *Switch*: e4

### 15. *aggregation* 3

- *Switch*: a3 <-> *Switch*: e5
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: a3 <-> *Switch*: e3
- *Switch*: a3 <-> *Switch*: e3
- *Switch*: a3 <-> *Switch*: e5
- *Switch*: a3 <-> *Switch*: e5
- *Switch*: a3 <-> *Switch*: e4
- *Switch*: a3 <-> *Switch*: e3
- *Switch*: a3 <-> *Switch*: e4
- *Switch*: a3 <-> *Switch*: e4

### 16. *aggregation* 2

- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c8 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a2
- *Switch*: a2 <-> *Switch*: e0

- *Switch*: a2 <-> *Switch*: e2
- *Switch*: a2 <-> *Switch*: e0
- *Switch*: a2 <-> *Switch*: e1
- *Switch*: a2 <-> *Switch*: e0
- *Switch*: a2 <-> *Switch*: e1
- *Switch*: a2 <-> *Switch*: e1
- *Switch*: a2 <-> *Switch*: e2
- *Switch*: a2 <-> *Switch*: e2

### 17. *aggregation* 1

- *Switch*: c8 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: a1 <-> *Switch*: e0
- *Switch*: a1 <-> *Switch*: e2
- *Switch*: a1 <-> *Switch*: e0
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e0
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e1
- *Switch*: a1 <-> *Switch*: e2
- *Switch*: a1 <-> *Switch*: e2

### 18. *aggregation* 0

- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: a0 <-> *Switch*: e0
- *Switch*: a0 <-> *Switch*: e2
- *Switch*: a0 <-> *Switch*: e0
- *Switch*: a0 <-> *Switch*: e1
- *Switch*: a0 <-> *Switch*: e0
- *Switch*: a0 <-> *Switch*: e1
- *Switch*: a0 <-> *Switch*: e1
- *Switch*: a0 <-> *Switch*: e2
- *Switch*: a0 <-> *Switch*: e2

## 4. c ditujukan untuk *core*

### a. *core* 8

- *Switch*: c8 <-> *Switch*: a14
- *Switch*: c8 <-> *Switch*: a5
- *Switch*: c8 <-> *Switch*: a2

- *Switch*: c8 <-> *Switch*: a11
- *Switch*: c8 <-> *Switch*: a5
- *Switch*: c8 <-> *Switch*: a5
- *Switch*: c8 <-> *Switch*: a5
- *Switch*: c8 <-> *Switch*: a8
- *Switch*: c8 <-> *Switch*: a8
- *Switch*: c8 <-> *Switch*: a2
- *Switch*: c8 <-> *Switch*: a11
- *Switch*: c8 <-> *Switch*: a5
- *Switch*: c8 <-> *Switch*: a2
- *Switch*: c8 <-> *Switch*: a14
- *Switch*: c8 <-> *Switch*: a11
- *Switch*: c8 <-> *Switch*: a17
- *Switch*: c8 <-> *Switch*: a2
- *Switch*: c8 <-> *Switch*: a2
- *Switch*: c8 <-> *Switch*: a14
- *Switch*: c8 <-> *Switch*: a14
- *Switch*: c8 <-> *Switch*: a17
- *Switch*: c8 <-> *Switch*: a11
- *Switch*: c8 <-> *Switch*: a8
- *Switch*: c8 <-> *Switch*: a8
- *Switch*: c8 <-> *Switch*: a8
- *Switch*: c8 <-> *Switch*: a5
- *Switch*: c8 <-> *Switch*: a2
- *Switch*: c8 <-> *Switch*: a14
- *Switch*: c8 <-> *Switch*: a17
- *Switch*: c8 <-> *Switch*: a11

**b. *core* 7**

- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a5
- *Switch*: c7 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a5
- *Switch*: c7 <-> *Switch*: a5
- *Switch*: c7 <-> *Switch*: a5
- *Switch*: c7 <-> *Switch*: a8
- *Switch*: c7 <-> *Switch*: a8
- *Switch*: c7 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a5

- *Switch*: c7 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a8
- *Switch*: c7 <-> *Switch*: a8
- *Switch*: c7 <-> *Switch*: a8
- *Switch*: c7 <-> *Switch*: a5
- *Switch*: c7 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a5
- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a2
- *Switch*: c7 <-> *Switch*: a5
- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a8
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a11
- *Switch*: c7 <-> *Switch*: a14
- *Switch*: c7 <-> *Switch*: a8
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a5
- *Switch*: c7 <-> *Switch*: a8
- *Switch*: c7 <-> *Switch*: a2

- *Switch*: c7 <-> *Switch*: a17
- *Switch*: c7 <-> *Switch*: a18

**e. *core* 6**

- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a11

- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a11
- *Switch*: c6 <-> *Switch*: a14
- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a5
- *Switch*: c6 <-> *Switch*: a8
- *Switch*: c6 <-> *Switch*: a2
- *Switch*: c6 <-> *Switch*: a17
- *Switch*: c6 <-> *Switch*: a8

**f. *core* 5**

- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c5 <-> *Switch*: a7
- *Switch*: c5 <-> *Switch*: a7
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a7
- *Switch*: c5 <-> *Switch*: a7

- *Switch*: c5 <-> *Switch*: a7
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a7
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a10
- *Switch*: c5 <-> *Switch*: a13
- *Switch*: c5 <-> *Switch*: a7
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a4
- *Switch*: c5 <-> *Switch*: a7
- *Switch*: c5 <-> *Switch*: a1
- *Switch*: c5 <-> *Switch*: a16
- *Switch*: c5 <-> *Switch*: a7

**g. *core* 4**

- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a7
- *Switch*: c4 <-> *Switch*: a7

- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a7
- *Switch*: c4 <-> *Switch*: a7
- *Switch*: c4 <-> *Switch*: a7
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a7
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a10
- *Switch*: c4 <-> *Switch*: a13
- *Switch*: c4 <-> *Switch*: a7
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a16

- *Switch*: c4 <-> *Switch*: a4
- *Switch*: c4 <-> *Switch*: a7
- *Switch*: c4 <-> *Switch*: a1
- *Switch*: c4 <-> *Switch*: a16
- *Switch*: c4 <-> *Switch*: a7

**h. *core* 3**

- *Switch*: c3 <-> *Switch*: a13
- *Switch*: c3 <-> *Switch*: a4
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a10
- *Switch*: c3 <-> *Switch*: a4
- *Switch*: c3 <-> *Switch*: a4
- *Switch*: c3 <-> *Switch*: a4
- *Switch*: c3 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a10
- *Switch*: c3 <-> *Switch*: a4
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a13
- *Switch*: c3 <-> *Switch*: a10
- *Switch*: c3 <-> *Switch*: a16
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a13
- *Switch*: c3 <-> *Switch*: a13
- *Switch*: c3 <-> *Switch*: a16
- *Switch*: c3 <-> *Switch*: a10
- *Switch*: c3 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a7
- *Switch*: c3 <-> *Switch*: a4
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a13
- *Switch*: c3 <-> *Switch*: a16
- *Switch*: c3 <-> *Switch*: a10
- *Switch*: c3 <-> *Switch*: a10
- *Switch*: c3 <-> *Switch*: a1
- *Switch*: c3 <-> *Switch*: a13
- *Switch*: c3 <-> *Switch*: a4

- *Switch*: c3 <-> *Switch*: a13

**i. *core* 2**

- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a12

- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a9
- *Switch*: c2 <-> *Switch*: a12
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a3
- *Switch*: c2 <-> *Switch*: a6
- *Switch*: c2 <-> *Switch*: a0
- *Switch*: c2 <-> *Switch*: a15
- *Switch*: c2 <-> *Switch*: a6

**j. *core* 1**

- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a6

- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a9
- *Switch*: c1 <-> *Switch*: a12
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a3
- *Switch*: c1 <-> *Switch*: a6
- *Switch*: c1 <-> *Switch*: a0
- *Switch*: c1 <-> *Switch*: a15
- *Switch*: c1 <-> *Switch*: a6

**k. *core* 0**

- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a6

- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a9
- *Switch*: c0 <-> *Switch*: a12
- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a15

- *Switch*: c0 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a3
- *Switch*: c0 <-> *Switch*: a6
- *Switch*: c0 <-> *Switch*: a0
- *Switch*: c0 <-> *Switch*: a15
- *Switch*: c0 <-> *Switch*: a6

### 4.2.2. Hasil Pengujian *Virtual Topology*

Pada virtual topology, proses pengalokasian VM kedalam *host* yang tepat dilakukan menggunakan dua algoritma yaitu *bestfit* (MFF) dan *worstfit* (LFF).

### A. Menggunakan Algoritma *Bestfit*

Dengan menggunakan algoritma *bestfit*, maka proses penempatan VM (VMs *placed*) dilakukan dengan mengalokasikan VM secara maksimal kedalam sebuah host. Selain itu, dengan algoritma *bestfit*, VM akan ditempatkan ke dalam sebuah *host* yang memiliki *power* dan *bandwidth* yang paling lengkap. Pengujian menggunakan algoritma *bestfit* dapat dilihat pada pengujian berikut:

#### 1. Pengujian 1

Skenario pertama menggunakan *data size* sebesar 10 MB, dengan jumlah *number of cloud* sebanyak 10 buah, 16 *host* dan menggunakan algoritma *bestfit*. Untuk melakukan pengecekan terhadap hasil pengujian 1, dapat dilihat pada Lampiran 1.

**Tabel 11 Hasil *Virtual Topology* menggunakan Pengujian 1**

| |
|---|
| VM Created: 6 in SDNHost 0 |
| VM Created: 13 in SDNHost 0 |
| VM Created: 16 in SDNHost 0 |
| VM Created: 28 in SDNHost 0 |
| VM Created: 0 in SDNHost 1 |
| VM Created: 17 in SDNHost 1 |
| VM Created: 21 in SDNHost 1 |
| VM Created: 22 in SDNHost 2 |
| VM Created: 3 in SDNHost 1 |
| VM Created: 11 in SDNHost 2 |
| VM Created: 18 in SDNHost 2 |
| VM Created: 19 in SDNHost 3 |
| VM Created: 20 in SDNHost 3 |
| VM Created: 23 in SDNHost 4 |

| |
|---|
| VM Created: 26 in SDNHost 5 |
| VM Created: 1 in SDNHost 4 |
| VM Created: 5 in SDNHost 6 |
| VM Created: 9 in SDNHost 3 |
| VM Created: 10 in SDNHost 6 |
| VM Created: 14 in SDNHost 7 |
| VM Created: 4 in SDNHost 7 |
| VM Created: 7 in SDNHost 5 |
| VM Created: 8 in SDNHost 8 |
| VM Created: 15 in SDNHost 6 |
| VM Created: 2 in SDNHost 9 |
| VM Created: 24 in SDNHost 8 |
| VM Created: 12 in SDNHost 8 |
| VM Created: 25 in SDNHost 9 |
| VM Created: 27 in SDNHost 9 |
| VM Created: 29 in SDNHost 10 |

## 2. Pengujian 3

Skenario kedua menggunakan *data size* sebesar 10 MB, dengan jumlah *number of cloud* sebanyak 10 buah, 54 *host* dan menggunakan algoritma *bestfit*. Untuk melakukan pengecekan terhadap hasil pengujian 3, dapat dilihat pada Lampiran 3.

**Tabel 12 Hasil *Virtual Topology* menggunakan Pengujian 3**

| |
|---|
| VM Created: 6 in SDNHost 0 |
| VM Created: 13 in SDNHost 0 |
| VM Created: 16 in SDNHost 0 |
| VM Created: 28 in SDNHost 0 |
| VM Created: 0 in SDNHost 1 |
| VM Created: 17 in SDNHost 1 |
| VM Created: 21 in SDNHost 1 |
| VM Created: 22 in SDNHost 2 |
| VM Created: 3 in SDNHost 1 |
| VM Created: 11 in SDNHost 2 |
| VM Created: 18 in SDNHost 2 |
| VM Created: 19 in SDNHost 3 |
| VM Created: 20 in SDNHost 3 |
| VM Created: 23 in SDNHost 4 |
| VM Created: 26 in SDNHost 5 |
| VM Created: 1 in SDNHost 4 |
| VM Created: 5 in SDNHost 6 |
| VM Created: 9 in SDNHost 3 |

| |
|---|
| VM Created: 10 in SDNHost 6 |
| VM Created: 14 in SDNHost 7 |
| VM Created: 4 in SDNHost 7 |
| VM Created: 7 in SDNHost 5 |
| VM Created: 8 in SDNHost 8 |
| VM Created: 15 in SDNHost 6 |
| VM Created: 2 in SDNHost 9 |
| VM Created: 24 in SDNHost 8 |
| VM Created: 12 in SDNHost 8 |
| VM Created: 25 in SDNHost 9 |
| VM Created: 27 in SDNHost 9 |
| VM Created: 29 in SDNHost 10 |

### 3. Pengujian 5

Skenario pertama menggunakan *data size random*, dengan jumlah *number of cloud* sebanyak 10 buah, 16 *host* dan menggunakan algoritma *bestfit*. Untuk melakukan pengecekan terhadap hasil pengujian 5, dapat dilihat pada Lampiran 1.

**Tabel 13 Hasil *Virtual Topology* menggunakan Pengujian 5**

| |
|---|
| VM Created: 6 in SDNHost 0 |
| VM Created: 13 in SDNHost 0 |
| VM Created: 16 in SDNHost 0 |
| VM Created: 28 in SDNHost 0 |
| VM Created: 0 in SDNHost 1 |
| VM Created: 17 in SDNHost 1 |
| VM Created: 21 in SDNHost 1 |
| VM Created: 22 in SDNHost 2 |
| VM Created: 3 in SDNHost 1 |
| VM Created: 11 in SDNHost 2 |
| VM Created: 18 in SDNHost 2 |
| VM Created: 19 in SDNHost 3 |
| VM Created: 20 in SDNHost 3 |
| VM Created: 23 in SDNHost 4 |
| VM Created: 26 in SDNHost 5 |
| VM Created: 1 in SDNHost 4 |
| VM Created: 5 in SDNHost 6 |
| VM Created: 9 in SDNHost 3 |
| VM Created: 10 in SDNHost 6 |
| VM Created: 14 in SDNHost 7 |
| VM Created: 4 in SDNHost 7 |

| |
|---|
| VM Created: 7 in SDNHost 5 |
| VM Created: 8 in SDNHost 8 |
| VM Created: 15 in SDNHost 6 |
| VM Created: 2 in SDNHost 9 |
| VM Created: 24 in SDNHost 8 |
| VM Created: 12 in SDNHost 8 |
| VM Created: 25 in SDNHost 9 |
| VM Created: 27 in SDNHost 9 |
| VM Created: 29 in SDNHost 10 |

**4. Pengujian 7**

Skenario pertama menggunakan *data size random*, dengan jumlah *number of cloud* sebanyak 10 buah, 54 *host* dan menggunakan algoritma *bestfit*. Untuk melakukan pengecekan terhadap hasil pengujian 7, dapat dilihat pada Lampiran 3.

**Tabel 14 Hasil *Virtual Topology* menggunakan Pengujian 7**

| |
|---|
| VM Created: 6 in SDNHost 0 |
| VM Created: 13 in SDNHost 0 |
| VM Created: 16 in SDNHost 0 |
| VM Created: 28 in SDNHost 0 |
| VM Created: 0 in SDNHost 1 |
| VM Created: 17 in SDNHost 1 |
| VM Created: 21 in SDNHost 1 |
| VM Created: 22 in SDNHost 2 |
| VM Created: 3 in SDNHost 1 |
| VM Created: 11 in SDNHost 2 |
| VM Created: 18 in SDNHost 2 |
| VM Created: 19 in SDNHost 3 |
| VM Created: 20 in SDNHost 3 |
| VM Created: 23 in SDNHost 4 |
| VM Created: 26 in SDNHost 5 |
| VM Created: 1 in SDNHost 4 |
| VM Created: 5 in SDNHost 6 |
| VM Created: 9 in SDNHost 3 |
| VM Created: 10 in SDNHost 6 |
| VM Created: 14 in SDNHost 7 |
| VM Created: 4 in SDNHost 7 |
| VM Created: 7 in SDNHost 5 |
| VM Created: 8 in SDNHost 8 |
| VM Created: 15 in SDNHost 6 |

| |
|---|
| VM Created: 2 in SDNHost 9 |
| VM Created: 24 in SDNHost 8 |
| VM Created: 12 in SDNHost 8 |
| VM Created: 25 in SDNHost 9 |
| VM Created: 27 in SDNHost 9 |
| VM Created: 29 in SDNHost 10 |

## B. Menggunakan Algoritma *Worstfit*

Dengan menggunakan algoritma *worstfit*, maka proses penempatan VM (VMs *placed*) dilakukan dengan mengalokasikan VM secara merata pada setiap VM. Sehingga penggunaan *host* semakin meningkat dibandingkan ketika menggunakan algoritma *bestfit*. Hasil pengujian menggunakan algortima *worstfit* dapat dilihat pada pengujian berikut:

### 1. Pengujian 2

Skenario pertama menggunakan *data size* sebesar 10 MB, dengan jumlah *number of cloud* sebanyak 10 buah, 16 *host* dan menggunakan algoritma *worstfit*. Untuk melakukan pengecekan terhadap hasil pengujian 2, dapat dilihat pada Lampiran 2.

**Tabel 15 Hasil *Virtual Topology* menggunakan Pengujian 2**

| |
|---|
| VM Created: 6 in SDNHost 0 |
| VM Created: 13 in SDNHost 1 |
| VM Created: 16 in SDNHost 2 |
| VM Created: 28 in SDNHost 3 |
| VM Created: 0 in SDNHost 4 |
| VM Created: 17 in SDNHost 5 |
| VM Created: 21 in SDNHost 6 |
| VM Created: 22 in SDNHost 7 |
| VM Created: 3 in SDNHost 8 |
| VM Created: 11 in SDNHost 9 |
| VM Created: 18 in SDNHost 10 |
| VM Created: 19 in SDNHost 11 |
| VM Created: 20 in SDNHost 12 |
| VM Created: 23 in SDNHost 13 |
| VM Created: 26 in SDNHost 14 |
| VM Created: 1 in SDNHost 15 |
| VM Created: 5 in SDNHost 0 |
| VM Created: 9 in SDNHost 14 |
| VM Created: 10 in SDNHost 2 |

| |
|---|
| VM Created: 14 in SDNHost 3 |
| VM Created: 4 in SDNHost 5 |
| VM Created: 7 in SDNHost 7 |
| VM Created: 8 in SDNHost 10 |
| VM Created: 15 in SDNHost 9 |
| VM Created: 2 in SDNHost 6 |
| VM Created: 24 in SDNHost 12 |
| VM Created: 12 in SDNHost 8 |
| VM Created: 25 in SDNHost 12 |
| VM Created: 27 in SDNHost 5 |
| VM Created: 29 in SDNHost 8 |

## 2. Pengujian 4

Skenario kedua menggunakan *data size* sebesar 10 MB, dengan jumlah *number of cloud* sebanyak 10 buah, 54 *host* dan menggunakan algoritma *worstfit*. Untuk melakukan pengecekan terhadap hasil pengujian 4, dapat dilihat pada Lampiran 4.

**Tabel 16 Hasil *Virtual Topology* menggunakan Pengujian 4**

| |
|---|
| VM Created: 6 in SDNHost 0 |
| VM Created: 13 in SDNHost 1 |
| VM Created: 16 in SDNHost 2 |
| VM Created: 28 in SDNHost 3 |
| VM Created: 0 in SDNHost 4 |
| VM Created: 17 in SDNHost 5 |
| VM Created: 21 in SDNHost 6 |
| VM Created: 22 in SDNHost 7 |
| VM Created: 3 in SDNHost 8 |
| VM Created: 11 in SDNHost 9 |
| VM Created: 18 in SDNHost 10 |
| VM Created: 19 in SDNHost 11 |
| VM Created: 20 in SDNHost 12 |
| VM Created: 23 in SDNHost 13 |
| VM Created: 26 in SDNHost 14 |
| VM Created: 1 in SDNHost 15 |
| VM Created: 5 in SDNHost 16 |
| VM Created: 9 in SDNHost 17 |
| VM Created: 10 in SDNHost 18 |
| VM Created: 14 in SDNHost 19 |
| VM Created: 4 in SDNHost 20 |
| VM Created: 7 in SDNHost 21 |

| |
|---|
| VM Created: 8 in SDNHost 22 |
| VM Created: 15 in SDNHost 23 |
| VM Created: 2 in SDNHost 24 |
| VM Created: 24 in SDNHost 25 |
| VM Created: 12 in SDNHost 26 |
| VM Created: 25 in SDNHost 27 |
| VM Created: 27 in SDNHost 28 |
| VM Created: 29 in SDNHost 29 |

### 3. Pengujian 6

Skenario pertama menggunakan *data size random*, dengan jumlah *number of cloud* sebanyak 10 buah, 16 *host* dan menggunakan algoritma *worstfit*. Untuk melakukan pengecekan terhadap hasil pengujian 6, dapat dilihat pada Lampiran 2.

**Tabel 17 Hasil *Virtual Topology* menggunakan Pengujian 6**

| |
|---|
| VM Created: 6 in SDNHost 0 |
| VM Created: 13 in SDNHost 1 |
| VM Created: 16 in SDNHost 2 |
| VM Created: 28 in SDNHost 3 |
| VM Created: 0 in SDNHost 4 |
| VM Created: 17 in SDNHost 5 |
| VM Created: 21 in SDNHost 6 |
| VM Created: 22 in SDNHost 7 |
| VM Created: 3 in SDNHost 8 |
| VM Created: 11 in SDNHost 9 |
| VM Created: 18 in SDNHost 10 |
| VM Created: 19 in SDNHost 11 |
| VM Created: 20 in SDNHost 12 |
| VM Created: 23 in SDNHost 13 |
| VM Created: 26 in SDNHost 14 |
| VM Created: 1 in SDNHost 15 |
| VM Created: 5 in SDNHost 0 |
| VM Created: 9 in SDNHost 14 |
| VM Created: 10 in SDNHost 2 |
| VM Created: 14 in SDNHost 3 |
| VM Created: 4 in SDNHost 5 |
| VM Created: 7 in SDNHost 7 |

| |
|---|
| VM Created: 8 in SDNHost 10 |
| VM Created: 15 in SDNHost 9 |
| VM Created: 2 in SDNHost 6 |
| VM Created: 24 in SDNHost 12 |
| VM Created: 12 in SDNHost 8 |
| VM Created: 25 in SDNHost 12 |
| VM Created: 27 in SDNHost 5 |
| VM Created: 29 in SDNHost 8 |

**4. Pengujian 8**

Skenario pertama menggunakan *data size random*, dengan jumlah *number of cloud* sebanyak 10 buah, 54 *host* dan menggunakan algoritma *worstfit*. Untuk melakukan pengecekan terhadap hasil pengujian 8, dapat dilihat pada Lampiran 4.

**Tabel 18 Hasil *Virtual Topology* menggunakan Pengujian 8**

| |
|---|
| VM Created: 6 in SDNHost 0 |
| VM Created: 13 in SDNHost 1 |
| VM Created: 16 in SDNHost 2 |
| VM Created: 28 in SDNHost 3 |
| VM Created: 0 in SDNHost 4 |
| VM Created: 17 in SDNHost 5 |
| VM Created: 21 in SDNHost 6 |
| VM Created: 22 in SDNHost 7 |
| VM Created: 3 in SDNHost 8 |
| VM Created: 11 in SDNHost 9 |
| VM Created: 18 in SDNHost 10 |
| VM Created: 19 in SDNHost 11 |
| VM Created: 20 in SDNHost 12 |
| VM Created: 23 in SDNHost 13 |
| VM Created: 26 in SDNHost 14 |
| VM Created: 1 in SDNHost 15 |
| VM Created: 5 in SDNHost 16 |
| VM Created: 9 in SDNHost 17 |
| VM Created: 10 in SDNHost 18 |
| VM Created: 14 in SDNHost 19 |
| VM Created: 4 in SDNHost 20 |
| VM Created: 7 in SDNHost 21 |

```
VM Created: 8 in SDNHost 22
VM Created: 15 in SDNHost 23
VM Created: 2 in SDNHost 24
VM Created: 24 in SDNHost 25
VM Created: 12 in SDNHost 26
VM Created: 25 in SDNHost 27
VM Created: 27 in SDNHost 28
VM Created: 29 in SDNHost 29
```

# BAB V

# Hasil dan Pembahasan

## 5.1 Hasil

Simulasi menggunakan CloudSimSDN memperoleh hasil pengujian sebagai berikut:

### 5.1.1 Total Konsumsi Energi

Menunjukkan total konsumsi energi atas data center dengan *host* maksimum yang digunakan pada jangka waktu yang sama. Pengujian terhadap konsumsi energi bertujuan untuk menilai konsumsi energi yang digunakan ketika penempatan VM (VM *Placement*) berdasarkan algoritma *bestfit* dan *worsfit*. Penulis membandingkan hasil dari dua algoritma pada dua metrik yaitu konsumsi energi dari *host* dan *switch*. Secara khusus, penulis berfokus untuk melihat energi yang dikonsumsi oleh *switch* karena energi yang dikonsumsi oleh *switch* merupakan bagian dari konsumsi energi jaringan.

Untuk mendukung hasil dari energi yang dikonsumsi oleh *switch*, penulis menyediakan jumlah *switch* digunakan (*on*) dan jumlah *switch* yang tidak digunakan (off). Percobaan ini ditujukan untuk menjawab pertanyaan penelitian pertama (RQ1). Skenario pengujian yang digunakan adalah skenario pengujian yang telah disusun pada subbab 3.2.4 pada Tabel 6. Pada pengujian ini, penulis menggunakan dua *host* dengn ukuran yang berbeda yaitu 16 *host* dan 54 *host*. Masing-masing *host* dikonfigurasikan dengan *core* 8 dan 16 GB RAM. Skenario 1A hingga 1D diterapkan kepada 16 *host* dan skenario 2A sampai 2D diterapkan kepada 54 *host*. Hasil pengujian terhadap 8 skenario diatas dapat dilihat pada tabel dibawah ini:

**A. Pengujian dengan Menggunakan Skenario 1**

Berikut merupakan hasil pengujian menggunakan skenario 1, dengan *data size* berukuran 10 MB, dan *host* sejumlah 16 dan 54 *host*. Hasil pengujian tersebut dapat dilihat pada Tabel 19 dan Tabel 20 berikut:

**Tabel 19 Total Konsumsi Energi Menggunakan 16 Host Dengan Skenario 1**

| *Scenario Configuration* | *Algorithm* | *Energy Consumption (Wh)* | | | *Nodes* | |
|---|---|---|---|---|---|---|
| | | *Host* | *Switch* | *Total* | *Utilized Switch* | *Off Switch* |
| Scenario 1.A | *Bestfit* | 66128.8934548611 | 32263.864583333336 | 98392.75803819444 | 16 | 4 |
| | | | | | | |
| Scenario 1.B | *Worstfit* | 84128.89345486111 | 45799.73847222223 | 129928.63192708333 | 20 | 0 |

**Tabel 20 Total Konsumsi Energi Menggunakan 54 Host Dengan Skenario 1**

| *Scenario Configuration* | *Algorithm* | *Energy Consumption (Wh)* | | | *Nodes* | |
|---|---|---|---|---|---|---|
| | | *Host* | *Switch* | *Total* | *Utilized Switch* | *Off Switch* |
| Scenario 1.C | *Bestfit* | 66128.8934548611 | 21634.134527777776 | 87763.02798263887 | 11 | 34 |
| | | | | | | |
| Scenario 1.D | *Worstfit* | 168128.8934548611 | 32928.02622222223 | 201056.91967708332 | 27 | 18 |

**B. Pengujian dengan Menggunakan Skenario 2**

Berikut merupakan hasil pengujian menggunakan skenario 2, dengan *data size random*, dan *host* sejumlah 16 dan 54 *host*. Hasil pengujian tersebut dapat dilihat pada Tabel 20 dan Tabel 21 berikut:

**Tabel 21 Total Konsumsi Energi Menggunakan 16 Host Dengan Skenario 2**

| *Scenario Configuration* | *Algorithm* | *Energy Consumption (Wh)* | | | *Nodes* | |
|---|---|---|---|---|---|---|
| | | *Host* | *Switch* | *Total* | *Utilized Switch* | *Off Switch* |
| Scenario 2.A | *Bestfit* | 66188.44045572917 | 31705.46997222223 | 97893.9104279514 | 16 | 4 |
| | | | | | | |
| Scenario 2.B | *Worstfit* | 84188.44045572917 | 44517.132777777784 | 128705.57323350696 | 20 | 0 |

**Tabel 22 Total Konsumsi Energi Menggunakan 54 Host Dengan Skenario 2**

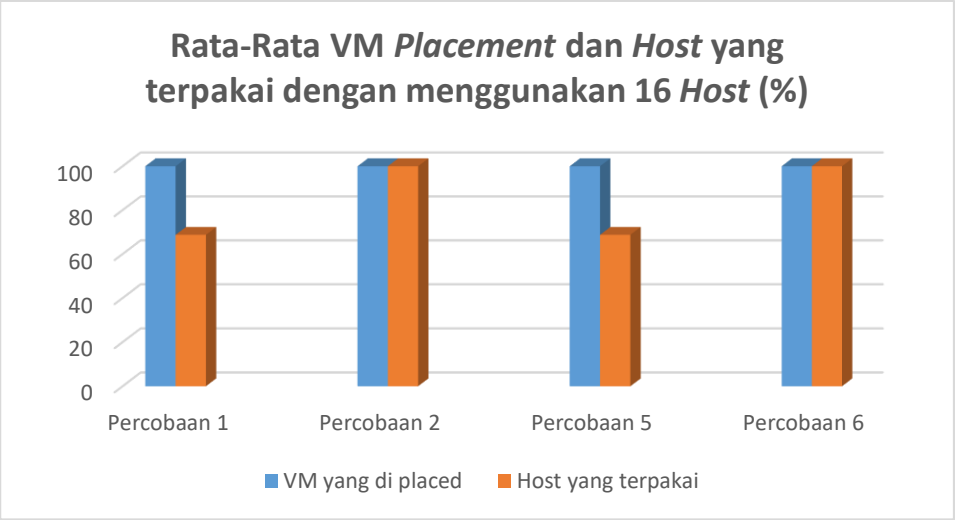
| *Scenario Configuration* | *Algorithm* | *Energy Consumption (Wh)* | | | *Nodes* | |
|---|---|---|---|---|---|---|
| | | *Host* | *Switch* | *Total* | *Utilized Switch* | *Off Switch* |
| Scenario 2.C | *Bestfit* | 66188.44045572917 | 21508.889777777782 | 87697.33023350695 | 31 | 11 |
| | | | | | | |
| Scenario 2.D | *Worstfit* | 168188.44045572917 | 30869.757527777772 | 199058.19798350695 | 27 | 18 |

## 5.2 Pembahasan

Berdasarkan pengujian yang telah dilakukan, maka diperoleh hasil bahwa algoritma *Most Full First* (MFF) atau yang sering disebut sebagai *bestfit* dapat menghemat energi lebih banyak dibandingkan dengan algoritma *worstfit*, karena algoritma *bestfit* dapat meminimalisir penggunaan *host* pada setiap VM ditempatkan secara maksimal kedalam sebuah *host*. Sehingga penggunaan *host* pada algoritma *bestfit* dapat dikatakan lebih stabil, berbeda dengan algoritma *worstfit*. Karena, dengan menggunakan algoritma *worstfit*, VM ditempatkan secara merata tanpa memperhitungkan berapa banyak jumlah *host* yang digunakan.

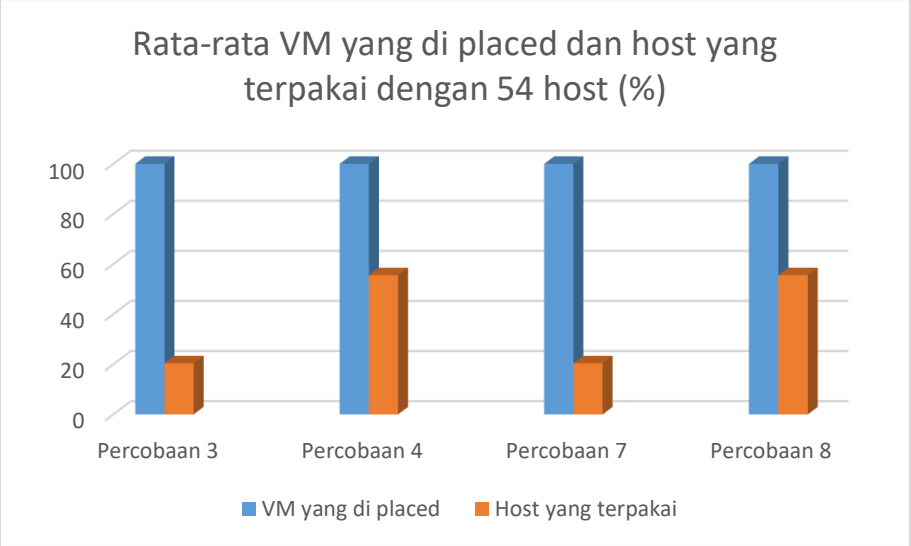
### 5.2.1 Persentase

Pada subbab 5.1.2 terdapat total konsumsi energi yang dibagi menjadi 2 bagian pengujian, yaitu menggunakan skenario 1 dengan kapasitas *data size* sebesar 10 MB, dan skenario 2 dengan kapasitas *data size random* antara 10 MB dan 20 MB. Berikut hasil pengujian tersebut disediakan dalam bentuk diagram:




**Gambar 19 Diagram Persentase *Placed* VMs dan *Utilized* 16 *Host***

Berdasarkan diagram pada Gambar 19, maka dapat diamati bahwa persentase penggunaan *host* menggunakan 16 *host* dengan 10 *cloud service* dan 30 maksimum VM meningkat. Persentase penggunaan *host* meningkat 31, 25 %. Ketika menggunakan SC2 menunjukkan hasil yang serupa. Persentase host yang digunakan mencapai 100 % sehingga persentase penempatan VM (*Placed* VMs) dalam keadaan stabil dalam angka 100 %. Dengan demikian, dapat diperoleh kesimpulan bahwa *host* yang tersedia mampu menampung seluruh VM.




**Gambar 20 Diagram Persentase *Placed VMs* dan *Utilized* 54 *Host***

Berdasarkan diagram pada Gambar 20, maka dapat diamati bahwa persentase penggunaan *host* menggunakan 54 *host* dengan 10 *cloud service* dan 30 maksimum VM meningkat. Persentase penggunaan *host* meningkat 35, 18 %. Ketika menggunakan SC2 menunjukkan hasil yang serupa. Persentase *host* yang digunakan mencapai 55, 55 % sehingga persentase penggunaan *host* dapat dikatakan kurang stabil, karena keseluruhan host digunaka

# BAB VI

# Kesimpulan dan Saran

## 6.1 Kesimpulan

Setelah melakukan perancangan dan implementasi kemudian dilakukan pengujian, maka dapat diambil beberapa kesimpulan mengenai pengujian yang telah dilakukan terhadap penggunaan energi dan energi jaringan pada transmisi paket dengan menggunakan algoritma *bestfit* dan *worstfit* dengan *topology Fat-Tree* antara lain sebagai berikut:

1. Pengujian menggunakan 16 *host* dan 54 *host* dengan membandingkan energi yang digunakan antara algoritma *bestfit* dengan algoritma *worstfit* pada proses penempatan VM atau yang disebut sebagai VM *Placement* menggunakan *topologi Fat-Tree*, memperoleh hasil yaitu algoritma yang lebih menghemat energi untuk mendapatkan *energy efficiency* jaringan yang paling menghemat energi jaringan secara maksimal adalah ketika pengujian menggunakan algoritma *bestfit*.
2. Ketika menggunakan 16 *host* dalam pengujian dapat menampung keseluruhan VM, sehingga keadaan penggunaan *host* dapat dikatakan stabil. Sedangkan, ketika menggunakan 54 *host*, *host* yang digunakan hanya sebesar 55, 55%.
3. Pengujian yang dilakukan sudah sesuai dengan ruang lingkup yang sudah dijelaskan pada subbab 1.3 yaitu tidak mengoptimasi setiap VM *Placement*, melainkan hanya menggunakan setiap *default* yang tersedia pada CloudSimSDN. Selain itu, pengujian yang dilakukan sudah menggunakan skenario penelitian yang sesuai seperti yang dijelaskan pada subbab 3.2.2.
4. Pada proses pengiriman paket atau transmisi paket, dapat diperoleh kesimpulan bahwa jumlah setiap paket yang dikirimkan oleh pengirim kepada penerima serupa dengan jumlah paket yang diterima oleh penerima paket.
5. Penggunaan *simulator* CloudSimSDN dalam pengujian ini sangat baik, karena tidak ditemukan *error* pada saat pengujian.

## 6.2 Saran

Setelah dilakukan pengujian, maka penulis mendapatkan saran agar penjelasan terhadap proses pengujian sebaiknya disediakan lebih terperinci agar tidak hanya pengguna yang sudah memahami mengenai *Cloud Computing* saja yang dapat mengerti alur pengujian, melainkan pengguna awam juga mampu untuk memahami dengan lebih mudah.

# DAFTAR PUSTAKA


[1] K. Bilal, S. Khan, J. Kolodziej, and L. Zhang, "A Comparative Study Of Data Center Network Architectures," Proceeding 26th Eur. Conf. Model. Simul., vol. 0, no. Cd, pp. 1–7, 2012.

[2] S. Kumar and R. H. Goudar, "Cloud Computing – Research Issues, Challenges, Architecture, Platforms and Applications: A Survey," Int. J. Futur. Comput. Commun., vol. 1, no. 4, pp. 356–360, 2012.

[3] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Trends in worldwide ICT electricity consumption from 2007 to 2012," Comput. Commun., vol. 50, pp. 64–76, 2014.

[4] Q. Yi and S. Singh, "Minimizing energy consumption of fat-tree data center networks," ACM SIGMETRICS Perform. Eval. Rev., vol. 42, no. 3, pp. 67–72, 2014.

[5] A. Carrega, S. Singh, R. Bruschi, and R. Bolla, "Traffic merging for energy-efficient Data center networks," Perform. Eval. Comput. Telecommun. Syst. (SPECTS), 2012 Int. Symp., pp. 1–5, 2012

[6] C. Paper, H. I. View, I. Conference, C. Engineering, and U. K. View, "Simulasi Mobility pada Software Defined Simulasi Mobility pada Software Defined Networking," vol. 2018, no. November 2016, pp. 26–27, 2018.

[7] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: Proceedings of the ACM SIGCOMM 2008 conference on Data communication, SIGCOMM '08, ACM, New York, NY, USA, 2008, pp. 63–74. doi:10.1145/1402958.1402967. URL http://doi.acm.org/10.1145/1402958.1402967

[8] Charles E. Leiserson, Zahi S. Abuhamdeh, David C. Douglas, Carl R. Feynman, Mahesh N. Ganmukhi, Jeffrey V. Hill, Daniel Hillis, Bradley C. Kuszmaul, Margaret A. St. Pierre, David S. Wells,Monica C. Wong, Shaw-Wen Yang, and Robert Zak. The network architecture of the connection machine cm-5. In Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures, SPAA '92, pages 272–285, New York, NY, USA, 1992. ACM.

[9] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "CloudSimSDN: Modeling and simulation of Software-defined cloud data centers," Proc. - 2015 IEEE/ACM 15th Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2015, pp. 475–484, 2015.

[10] J. Teixeira, G. Antichi, D. Adami, A. Del Chiaro, S. Giordano, and A. Santos, "Data center in a box: Test your SDN cloud-Data center controller at home," Proc. - 2013 2nd Eur. Work. Softw. Defin. Networks, EWSDN 2013, pp. 99–104, 2013.

[11] R. Buyya and M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing," Concurr. Comput. Pract. Exp., vol. 14, no. 13–15, pp. 1175–1220, 2002.

[12] P. S. Suryateja, "A Comparative Analysis of Cloud Simulators," Int. J. Mod. Educ. Comput. Sci., vol. 8, no. 4, pp. 64–71, 2016.

[13] L. Krug, M. Shackleton, and F. Saffre, "Understanding the environmental costs of fixed line networking," in Proc. 5th Int. Conf. Future e-Energy Syst., 2014, pp. 87–95.

[14] B. Gohil, "A Comparative Analysis of Virtual Machine Placement Techniques in the Cloud Environment," vol. 156, no. 14, pp. 12–18, 2016.

[15] CloudSimSDN - https://github.com/fogony/cloudsimsdn (last accessed on Feb, 2018)