**UNIVERSITAS INDONESIA**

# IMPLEMENTASI TEKNOLOGI CLOUD COMPUTING MENGGUNAKAN CLOUDSIM UNTUK IMPLEMENTASI KONSEP TIK HIJAU

**SKRIPSI**

**RYAN A.P. ARMANDA**

**0606078506**

**FAKULTAS TEKNIK**
**PROGRAM STUDI TEKNIK KOMPUTER**
**DEPOK**
**JUNI 2010**

**UNIVERSITAS INDONESIA**

# IMPLEMENTASI TEKNOLOGI CLOUD COMPUTING MENGGUNAKAN CLOUDSIM UNTUK IMPLEMENTASI KONSEP TIK HIJAU

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik**

**RYAN A.P. ARMANDA**

**0606078506**

**FAKULTAS TEKNIK**
**DEPARTEMEN ELEKTRO**
**PROGRAM STUDI TEKNIK KOMPUTER**
**DEPOK**
**JUNI 2010**

# HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,**
**dan semua sumber baik yang dikutip maupun dirujuk**
**telah saya nyatakan dengan benar.**

| | |
|---|---|
| **Nama** | **: Ryan A.P. Armanda** |
| **NPM** | **: 0606078506** |
| **Tanda Tangan** | **:** |
| **Tanggal** | **:15 Juni 2010** |

# HALAMAN PENGESAHAN

| | |
|---|---|
| Skripsi ini diajukan oleh | : |
| Nama | : Ryan A.P. Armanda |
| NPM | : 0606078506 |
| Program Studi | : Teknik Komputer |
| Judul Skripsi | : Implementasi *Cloud Computing* berbasis *CloudSim* untuk Implementasi Konsep TIK Hijau |

**Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia**

**DEWAN PENGUJI**

| | | |
|---|---|---|
| Pembimbing | : Prof. Dr. Ir. Riri Fitri Sari M.Sc, M.M. | ( ) |
| Penguji | : Prof. Dr. Ir. Bagio Budiardjo M.Sc. | ( ) |
| Penguji | : Prof. Dr.-Ing. Ir. Kalamullah Ramli M.Eng. | ( ) |

Ditetapkan di : Depok

Tanggal : 1 Juli 2010

# KATA PENGANTAR/UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Tuhan Yang Maha Kuasa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan Skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Program Studi Teknik Komputer pada Fakultas Teknik Universitas Indonesia.Saya menyadari, banyak pihak yang telah membantu dalam penyelesaian skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

1. Ibu Prof. Dr. Ir. Riri Fitri Sari MM, Msc. , selaku dosen pembimbing, atas arahan, ide, dan pengertian serta kesabaran dari beliau, yang menjadi kunci keberhasilan saya menyelesaikan Tugas Akhir ini.
2. Orang tua beserta keluarga saya, yang telah memberikan bantuan baik moril maupun materil dan keyakinan sehingga saya dapat menyelesaikan Skripsi ini.
3. Teman – teman yang sudah banyak membantu dan menemani selama hampir 4 tahun perkuliahan, yang tidak dapat saya sebutkan satu persatu.
4. Karenita, untuk dukungan, kepercayaan dan motivasi yang luar biasa.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu.Semogaskripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Depok, Juni 2010

Penulis

# HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

| | |
|---|---|
| Nama | :Ryan Aditya Perdana Armanda |
| NPM | :0606078506 |
| Program Studi | :Teknik Komputer |
| Departemen | :Teknik Elektro |
| Fakultas | :Teknik |
| Jenis karya | :Skripsi |

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul:
**Implementasi Teknologi *Cloud Computing* Menggunakan *CloudSim* untuk Implementasi Konsep TIK Hijau**
beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format- kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di :Depok
Pada tanggal :14 Juni 2010
Yang menyatakan

( Ryan A.P. Armanda)

# ABSTRAK

Nama : Ryan A.P. Armanda
Program Studi : Teknik Komputer
Judul : Implementasi Teknologi *Cloud Computing* Menggunakan *CloudSim* untuk Implementasi Konsep TIK Hijau

Tugas Akhir ini membahas tentang teknologi *cloud computing*, yaitu teknologi pemanfaatan *resource* komputasi, baik perangkat keras, maupun perangkat lunak, sebagai *service* melalui media Internet. Teknologi Informasi dan Komunikasi Hijau merupakan sebuah konsep teknologi komunikasi dan informasi yang bersifat ramah lingkungan. *Cloud computing* dalam tugas akhir ini, disimulasikan dengan bantuan program *CloudSim* dari *University of Melbourne.* Berkaitan dengan Teknologi Informasi dan Komunikasi Hijau, pada program *CloudSim* tersebut ditambahkan modul konsumsi tenaga, dan disipasi panas. Hal ini dilakukan dengan cara menambahkan modul dan fungsi berbasis *Java* pada program *CloudSim*. Pada Tugas Akhir ini dibuat simulasi penggunaan *CloudSim* dalam bentuk *GreenCloudSimulation,* dengan memodifikasi berbagai modul *CloudSim*. Selanjutnya dilakukan analisa kinerja berdasar kedua simulasi tersebut. Kemudian simulasi dijalankan pada arsitektur *cloud computing* dari Amazon, yaitu Amazon AWS. Hasil yang didapat adalah seiring dengan bertambahnya penggunaan *resource*, maka bertambah pula konsumsi daya pada simulasi. Pertambahan daya terjadi secara bertahap, seperti ditunjukkan pada hasil simulasi. Konsumsi daya meningkat dari 16,32 Watt menjadi 23,26 Watt. Disipasi panas meningkat dari 55,68 BTU menjadi 79.38 BTU. Sedangkan hasil simulasi pada arsitektur Amazon AWS menunjukkan hasil serupa dengan ketika dijalankan pada simulasi *CloudSim*, walaupun terdapat kendala kecepatan akses karena lokasi *datacenter* yang jauh.

Kata kunci:
Internet, *Web Service, Cloud Computing,* TIK Hijau, Amazon AWS, Konsumsi Tenaga, Disipasi Panas

# ABSTRACT

Name : Ryan A.P. Armanda
Study Program : Computer Engineering
Title : Implementation of Cloud Computing using CloudSim for Implementing Green ICT Concept

This final project reviews the cloud computing, which is technology of accessing resource, hardware or software, as a services through Internet. Green Information and Communication technology (Green ICT) is a concept of environmentally friendly ICT. Cloud computing on this project is simulated by CloudSim program. In conjunction with Green Information and Communication Technology, on the CloudSim program, we have added and customized the module about power consumption and heat dissipation. This is being done by adding a customized Java module to CloudSim source program. For the analysis part, we have made the CloudSim simulation project, based on GreenCloudSimulation execution, and so we can analyzed performance of the CloudSim based on the two programs above. Subsequently the simulation has also been executed on Amazon AWS cloud computing platform. The result of the simulation shows that along with resource increase usage, the power consumption and heat dissipation also being inclined. The increase of power and heat generated from this simulation happened gradually, as shown by the simulation, power consumption increase from 16,32 Watt to 23,26 Watt. The heat dissipation is increased from 55,68 BTU to 79,38 BTU. On the other hand, the simulation on the Amazon AWS platform shows the same results as the result of the CloudSim simulation, even though some problem occured due to the distance location of the datacenters.

Key Words:

Internet, Web Service, Cloud Computing, Green ICT, Amazon AWS, Power Consumption, Heat Dissipation

**DAFTAR ISI**

# DAFTAR GAMBAR

# BAB 1 PENDAHULUAN

## 1.1 LATAR BELAKANG

Perkembangan teknologi khususnya dibidang teknologi informasi membuat seluruh bidang dalam sebuah sistem perekonomian tidak lagi hanya membutuhkan teknologi informasi sebagai sarana pendukung, tetapi telah menjadi salah satu pilar utama dalam perancangan sebuah sistem ekonomi secara keseluruhan.Teknologi informasi telah bermetamorfosa menjadi sebuah basis penting dimana hal – hal substansial dari sebuah perusahaan, baik perusahaan kecil, hingga perusahaan multinasional, didokumentasikan dan disimpan dalam sebuah unit basis data. Dalam pelaksanaannya, basis data ini, beserta dengan aplikasi lainnya seringkali membutuhkan *resource* CPU yang tidak sedikit, dan membutuhkan perawatan yang bernilai tinggi.

Untuk semua hal – hal tersebut, sistem *Cloud Computing* dinilai sangat bermanfaat dan berguna bagi sebuah perusahaan, maupun pengguna pribadi. Hal ini disebabkan sistem *cloud computing* yang berupa sistem *resource* pihak ketiga yang dapat diakses secara *online*, maka penggunaan aplikasi tersebut dapat disewa dari pihak ketiga dengan sistem yang sangat fleksibel.

Perkembangan dari penggunaan *resource* secara kovensional dengan sistem membeli sebuah *resource*, proses instalasi sebuah *resource*, dan dilanjutkan dengan eksekusi dan perawatan sebuah *resource*, dapat dipangkas dengan seignifikan dengan penggunaan *cloud computing*. *Resource* tidak lagi hanya dapat diklasifikasikan sebagai "Pemilikan *Resource*" , namun dengan teknologi ini sebuah *resource* dapat diklasifikasikan sebagai "Penggunaan *Resource*" , dikarenakan pihak pengguna tidak lagi dibutuhkan untuk memiliki suatu *resource* secara khusus. Segala *resource* yang dibutuhkan dapat dimanfaatkan dengan sistem *service based*, yakni pengguna hanya perlu menyewa *resource* tersebut sesuai dengan kebutuhan dan keperluan dari pengguna itu sendiri [1].

Dalam penggunaannya cloud computing dapat diklasifikasikan dengan :

1. *Infrastructure as a Service* (IaaS) yaitu konsep tertua dimana pengimplementasiannya banyak dilakukan mulai dari penggunaan atau penyewaan jaringan untuk akses Internet, layanan *Disaster Recovery Center,* dsb.
2. *Platform as a Service* (PaaS) yaitu konsepnya hampir serupa dengan IaaS. Namun *Platform* disini adalah penggunaan *operating system* dan infrastruktur pendukungnya. Yang cukup terkenal adalah layanan dari situs *Force.com* serta layanan dari para *vendor server*.
3. *Software as a Service* (SaaS) yaitu berada satu tingkat diatas PaaS dan IaaS, dimana disini yang ditawarkan adalah *software* atau suatu aplikasi bisnis tertentu. Contoh yang paling mutakhir adalah *SalesForce.com*, *Service-Now.com*, Google Apps, dsb.

## 1.2 TUJUAN PENULISAN

Tujuan dari penulisan Tugas Akhir ini adalah pengenalan sekaligus mempelajari sebuah teknologi yang sangat ditunggu oleh pihak pihak dalam dunia komputer, yakni teknologi *cloud computing.* Dalam tulisan ini akan dibahas mengenai infrastruktur dan mekanisme teknologi *cloud computing*, serta implementasi dan masa depan teknologi ini.

## 1.3. BATASAN MASALAH

Pada Tugas Akhir ini pembatasan masalah akan dibatasi pada implementasi teknologi *cloud computing* yang berkaitan dengan teknologi informasi dan komunikasi hijau. Hal yang akan dievaluasi adalah modul disipasi panas dan penggunaan daya pada simulasi *CloudSim*.

## 1.4 SISTEMATIKA PENULISAN

Sistematika penulisan pada Tugas Akhir ini adalah :

1. **BAB 1 PENDAHULUAN**

   Pada Bab ini berisi Latar Belakang, Tujuan Penulisan, Batasan Masalah, dan Sistematika Penulisan.

2. **BAB 2 DASAR TEORI**

   Pada Bab ini berisi Pengenalan teknologi *Cloud computing*, mekanisme dan arsitektur, serta infrastuktur dari teknologi *cloud computing*.

3. **BAB 3 IMPLEMENTASI TEKNOLOGI CLOUD COMPUTING**

   Pada Bab ini akan dibahas mengenai implementasi teknologi *cloud computing*, perbandingan penyedia layanan, dan masa depan teknologi *cloud computing*.

4. **BAB 4 ANALISA**

   Pada Bab ini akan analisa mengenai simulasi *Cloud Computing* yang telah dirancang dari berbagai aspek

5. **Bab 5 KESIMPULAN**

   Pada Bab ini akan dijelaskan mengenai kesimpulan yang didapat dari Skripsi ini.

# BAB 2

# DASAR TEORI TEKNOLOGI CLOUD COMPUTING

## 2.1. Pengenalan Teknologi Cloud Computing

Evolusi, atau langkah awal dari teknologi *cloud computing*, berasal dari periode tahun 1960. *Time sharing* dan *multitasking*, yang menjadi fitur umum pada sistem operasi dewasa ini, merupakan fasilitas utama yang memungkinkan terciptanya teknologi *cloud computing*. Pada masa awal revolusi komputer, IBM merupakan pengembang terbesar dalam teknologi ini. *Mainframe* IBM merupakan sistem yang secara komersial dan utuh dapat diakses oleh publik, walau pada sisi pengguna, hanya dapat memproses beberapa perintah sederhana.

Pada dekade tahun 1970, *Tymeshare* sebagai penyedia layanan mengembangkan sistem penyewaan ruang *mainframe* yang dapat diakses melalui jalur telepon.Perusahaan tersebut menyediakan layanan komputer tingkat tinggi kepada perusahaan – perusahaan besar saat itu.

Dekade 1980 dan 1990 merupakan kelahiran banyak perusahaan yang menciptakan *mainframe* besar dan komputer dalam skala kecil. *Hardware – hardware* yang mereka diciptakan kemudian disewakan kepada perusahaan guna kebutuhan *datacenter*, dangan biaya perbulan atau per tahun. Perusahaan-perusahaan penyewa ini terhubung dengan *datacenter* melalui jalur ISDN, bersamaan dengan itu kemampuan kempampuan komputer lainnya juga disediakan untuk pelanggan. Data pelanggan di *back up* dalam sebuah kaset, dan dapat dipergunakan jika ada kerusakan sistem. Pada masa ini pula karena berkembangnya sistem penyewaan *resource*, berkembang sistem SLA atau *Service Level Agreement*. Dekade 1980 dan 1990 juga merupakan dekade kelahiran dari *Application Service Providers* (ASP), yang pada masa itu dikenal sebagai layanan pihak ketiga yang berfungsi dalam hal pendelegasian

Data, manajemen data, dan hosting *software* dalam layanan berbasis sewa. ASP melahirkan kesempatan untuk perusahaan berskala kecil untuk menggunakan teknologi terbaru, dengan biaya yang relatif rendah. Harga, pelayanan konsumen, dan RAS (*Realibility*, *Avaibility*, *Serviceability*) merupakan parameter dalam ASP [3].

Dekade 2000 merupakan tahun – tahun dimana penggunaan ASP melonjak tajam. Dimana perusahaan besar seperti AT&T dan Oracle mengintegrasikan layanan ASP dalam bisnis mereka. Namun layanan ini hanya terbatas dalam pengguna perusahaan, dan gagal menyentuh pengguna akhir sebagai konsumen. Hal ini dikarenakan ASP cukup sulit dan membutuhkan biaya yang besar untuk diinstallasi dan dirawat. Sumber daya manusia yang banyak dibutuhkan untuk memberikan bantuan kepada pengguna akhir, sejalan dengan merawat sistem ini. Oleh karena itu banyak perusahaan ASP yang berguguran, dan hanya sedikit yang mampu bertahan.

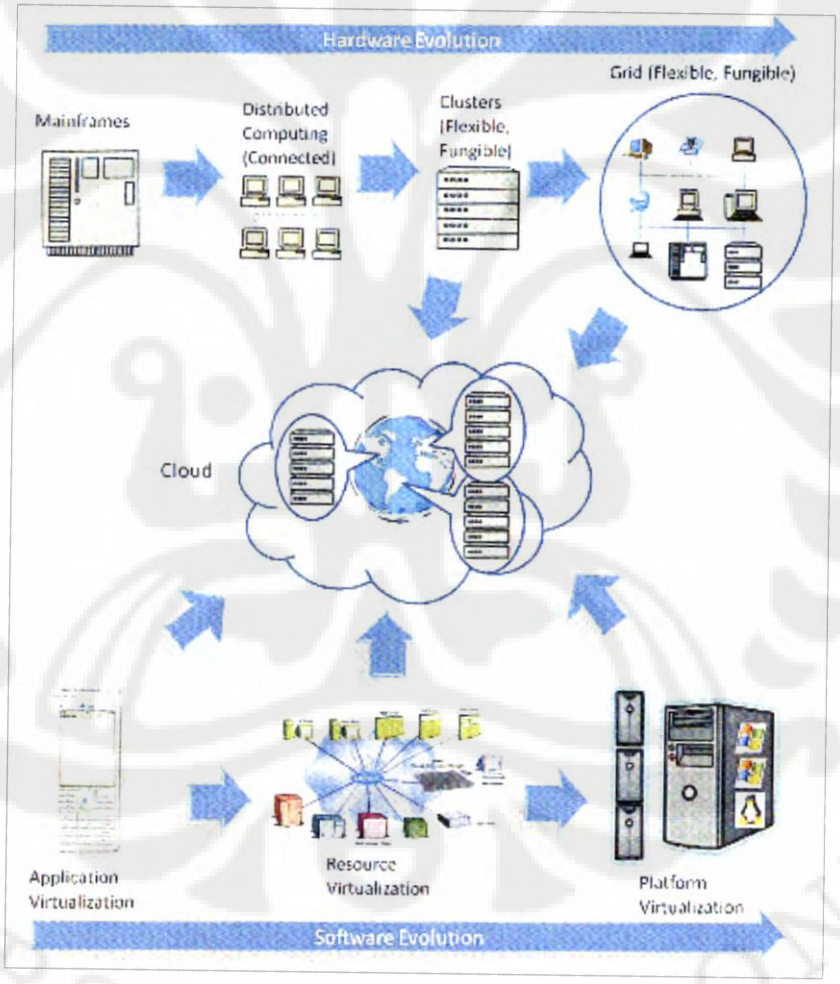
Dan keseluruh periode evolusi sistem komputer tersebut, pada akhirnya melahirkan dua konsep dasar yang nantinya akan menciptakan teknologi baru bernama *Cloud Computing*. Konsep tersebut yakni;

### 2.1.1 Web Hosting

Perkembangan yang sangat cepat dari sistem web melahirkan banyak aplikasi *e-commerce* dan aplikasi berbasis web lainnya. dalam banyak kasus, aplikasi berbasis web ini diletakkan pada infrastuktur yang dimiliki oleh penyedia lain. Aplikasi atau konten diletakkan pada satu penyedia (*hosting provider*) dan dimiliki oleh pengguna. Akses menuju aplikasi ini dilakukan melalui Web (menggunakan HTTP). Layanan ini biasanya akan menyediakan pengguna dengan pilihan sistem operasi apa yang hendak digunakan, hal ini dilakukan dengan tujuan tidak terjadi konflik sistem dengan aplikasi yang dikembangkan atau dimiliki pengguna, dan juga memberikan dukungan terhadap bahasa pemrograman apa yang digunakan pengguna, seperti Perl, *Python*, PhP, Ruby, dan lainnya. Pihak penyedia biasanya juga akan menawarkan registrasi nama domain, backup data, dan layanan tambahan lainnya.

### 2.1.2 Aplikasi Web Native

Aplikasi web pertama muncul ke permukaan sesaat setelah revolusi besar internat pada tahun 1995-1996. *E-mail* merupakan *software* utuh pertama yang tersedia dalam bentuk layanan berbasis *web*. *Hotmail*, yang dikemudian hari diakuisisi oleh microsoft, merupakan salah satu penyedia layanan e-*mail* cuma – cuma pertama di dunia. Sejalan dengan munculnya *e-mail*, aplikasi pendukung lain pun bermunculan di Internet, seperti kalender, *chat*, *talk*, pemetaan. Aplikasi ini yang dikemudian hari akan menjadi tiang pancang bagi terciptanya teknologi baru bernama *Cloud computing*.




Gambar 2.1 Evolusi Menuju layanan Cloud [2].

*Cloud computing*, yang merupakan mimpi dari seluruh insinyur komputer, memiliki prinsip sederhana, yakni menjadikan sebuah komputer menjadi sebuah *service client.* Sebuah aplikasi yang dapat diakses dari tempat yang berbeda. Para

pengembang aplikasi komputer dengan yang memiliki jaringan Internet tidak lagi membutuhkan investasi yang tinggi terhada perangkat keras komputer dan sumber daya manusia yang mampu untuk mengoperasikannya. Perusahaan–perusahaan besar dengan banyak pekerjaan yang berorientasi *resource* komputer, dapat mendapatkan hasil yang maksimal, dengan perbandingan, menggunakan 1000 *server* dalam satu jam, memiliki harga yang sebanding dengan menggunakan satu *server* untuk 1000 jam. Elastisitas *resource* ini, dengan keuntungan tidak harus membayar investasi yang besar untuk jaringan berskala besar, adalah hal yang tidak diperhitungkan dalam dunia teknologi informasi.

*Cloud computing* memiliki dua konsep, yakni konsep pada aplikasi yang dijalankan sebagai *service* dengan bantuan *Internet*, dan konsep perangkat keras yang berada pada pusat *datacenter* yang menyediakan akses bagi konsep sebelumnya.Konsep aplikasi sebagai sebuah *service*, telah lama dikenal dengan sebutan *Software as a Service (SaaS). Hardware* yang berada pada *datacenter* dan menjalankan aplikasi tersebut disebut *Cloud.* Ketika *Cloud* menjadi tersedia untuk layanan berbayar bagi kalangan umum, maka hal tersebut disebut dengan *Public Cloud*. S*ervice* yang dijual disebut *Utility Computing.* Gabungan antara konsep *Cloud*, *Software as a Service*, dan *Utility Computing*, kita kenal dengan nama *Cloud Computing.*

Dari perspektif *hardware*, ada tiga aspek baru dalam *cloud computing*:

1. *Resource komputer yang tak terbatas, dan tersedia jika dibutuhkan,* hal ini mengeliminasi kemungkinan terjadinya keterbatasan *resource* dan kebutuhan untuk perencanaan perawatan jangka panjang pada sebuah *resource*, dari pihak pengguna *cloud computing*.
2. *Eliminasi dari komitmen besar di awal instalasi.* Hal ini menyediakan kesempatan bagi perusahaan untuk memulai dengan dana yang minimum, dan meningkatkan *resource hardware* hanya ketika dibutuhkan.
3. *Kesempatan untuk membayar resource komputer dalam penggunaan jangka pendek.* Contohnya yaitu penggunaan prosesor per jam, dan penyimpanan data per hari, dan membuat *resource* yang tidak diperlukan dapat dilepaskan jika *resource* tersebut tak lagi berguna bagi pemakai.

Dari keuntungan dan hal diatas, maka dapat disimpulkan bahwa teknologi ini

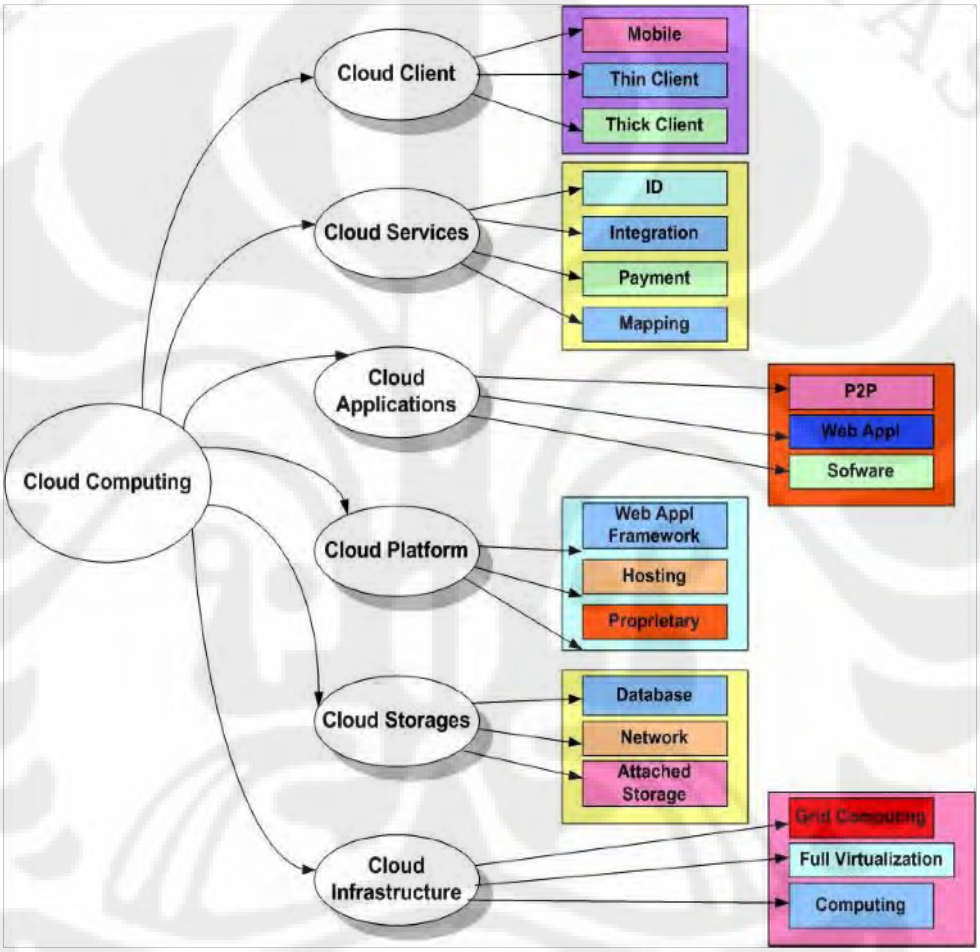
tidak hanya akan membawa perubahan yang signifikan kepada dunia computer. Perubahan yang signifikan juga akan terjadi pada dunia bisnis, dimana perusahaan dapat menghemat investasi dengan jumlah yang cukup besar.

Ada dua contoh perusahaan penyedia cloud computing yang dapat dikatakan sukses dalam menjalankan teknologi ini, yakni Amazon, dam Google. Dalam Tugas Akhir ini, contoh dari penerapan *cloud computing* akan lebih ditinjau dari layanan kedua perusahaan besar ini. *Elastic Compute Cloud* (EC2) dari *Amazon Web Service* (AWS) menjual sebuah resource berkekuatan 1.0-GHz x86 dengan harga $0.1 per jam. Dalam media penyimpanan, Amazon menyewakan data *storage* $0.12-$0.15 per *Gigabyte*/Bulan. Hal ini tentu cukup menguntungkan bagi pihak perusahaan, dikarenakan biaya yang dikeluarkan akan menjadi biaya tetap, tidak lagi seperti metode konvensional selama ini, dimana dana yang dikeluarkan berupa dana pembelian *resource*, ditambah dana tak terduga seperti perawatan dan *troubleshooting*, dan juga biaya pengoperasiaannya oleh tenaga kerja yang terampil.

Sebagai teknologi, *cloud computing* juga menghasilkan banyak kemungkinan kemungkinan baru, yang sulit dicapai dengan teknologi yang sudah ada. Hal tersebut dapat berupa implementasi pada perangkat bergerak (*mobile devices*), dan juga sistem *parallel processing* serta pemrograman dan perancangan sistem tingkat tinggi lainnya. Hal tersebut akan dijelaskan dalam bab-bab berikutnya [7].

## 2.2. Arsitektur Cloud Computing.

Arsitektur yang unik dari *cloud computing* memiliki peranan yang cukup besar pada proses tercipta dan berkembangnya teknologi ini. Penerapan *Software as a Service* dan *Client as a service* memiliki dampak yang sangat besar kepada tingginya harapan terhadap perkembangan teknologi ini.




Gambar 2.2 Struktur *Cloud Computing* [2].

Dari gambar diatas dapat kita perhatikan, ada beberapa arsitektur dasar dari sistem *cloud computing* secara umum, yakni *Cloud Client, Cloud Services, Cloud Applications, Cloud Platform, Cloud Storages*, dan *Cloud Infrastructure*. Arsitektur dasar *cloud computing* terdapat pada bagian *Cloud Infrastucture* dan *Cloud Platform*. Dari kedua bagian ini dapat diperhatikan, terdapat satu bagian yang menjadi inti dari sistem *cloud computing* secara keseluruhan, yakni *Virtualization*.

### 2.2.1 Virtual Machine

Dalam beberapa tahun terakhir, *virtual machine* telah menjadi standar bagi pengembangan aplikasi. *Virtualization* meningkatkan fleksibilitas karena *virtualization* meningkatnya kemampuan *software* sehingga dapat dijalankan tanpa harus terikat dengan sebuah *server* secara spesifik. *Virtualization* memungkinkan terciptanya sebuah *datacenter* dinamis yang ada ketika dibutuhkan, dan memiliki kemampuan dari beragam aplikasi untuk menghitung, menyimpan data, serta konvergensi jaringan dimana dapat berubah secara dinamis menyesuaikan kebutuhan sebuah organisasi. Dengan pemasangan sebuah aplikasi yang bersifat langsung dari sebuah *virtualization server*, aplikasi dapat dipasang dan menyesuaikan diri dengan cepat dengan keadaan *resource* yang tersedia, tanpa harus men-*setting server* fisik yang ada [8].

*Virtualization* membuat kemampuan sebuah komputer direpresentasikan dalam sebuah entitas logikal. Komputer *virtual* ini dapat berupa sebuah mesin, beberapa mesin yang terhubung dengan jaringan, atau bagian dari mesin yang memiliki cukup kemampuan untuk dibagi dengan beberapa pengguna yang membutuhkan kemampuan komputer. *Virtualization* sudah ada sejak generasi *mainframe* IBM. Beberapa perusahaan baru seperti VM Ware menciptakan generasi baru dari *virtualization software*. Perusahaan seperti Xen menyediakan *platform* sehingga perusahaan pengembang seperti *Sun Microsystems*, *Citrix, Oracle Systems* membuat sendiri sistem *virtualization* mereka. Perkembangan terakhir dari perusahan ini adalah membuat sebuah sistem yang dapat memuat lebih dari satu sistem operasi pada sebuah *hardware*. Hal inilah yang menjadi keuntungan dari konsep *cloud computing*, dengan menggunakan *virtualization* para penyedia jasa dapat memenuhi kebutuhan sistem operasi pengguna yang beragam dengan menggunakan *hardware* yang sama pada *datacenter*. Hal ini mereduksi biaya pada manajemen sebuah *datacenter*, untuk memberikan kemampuan kepada *software* berkomunikasi dengan *software* lainnya dalam sebuah jaringan, pihak penyedia layanan bahkan telah merancang sebuat *software* yang bertindak sebagai *router*. Menurut sebuah artikel dari *New York Times*, Cisco sebagai salah satu produsen terbesar *hardware* jaringan komputer telah

bekerja sama dengan penyedia layanan *cloud computing*, untuk membuat sebuah *virtual switch*.

*Virtual appliance*, yakni *virtual machine* yang berisi *software* yang telah dikonfigurasi penuh, atau dikonfigurasi parsial untuk melakukan sebuah tugas tertentu, seperti server *web*, atau basis data, dapat meningkatkan kemampuan untuk menciptakan dan memasang sebuah aplikasi dengan cepat. Kombinasi dari v*irtual machine* dan *virtual apliance* sebagai standar pemasangan sebuah aplikasi adalah salah satu fitur terpenting dari keseluruhan sistem *Cloud Computing*. [3]

### 2.2.2Application Programming Interface.

Arsitektur lain yang tak kalah penting dalam proses perancangan sistem cloud computing adalah *Application Programming Interface* (API). Menurut Lew Tucker, *Chief Technology Officer* dari *Sun Microsystems Cloud Computing Division*, API merupakan aspek yang seringkali dilupakan oleh para pengguna layanan *cloud computing*. Aplikasi yang tersedia pada *cloud computing* dapat diakses melalui Internet dikarenakan peran API sebagai fitur dari aplikasi tersebut. Hal ini tidak hanya berarti kita dapat menggunakan *browser* untuk berinteraksi dengan layanan yang ada pada Internet, kita dapat langsung terhubung dengan layanan yang ada dengan API. Teknologi seperti SOAP, XML, WSDL telah menjadi format standar untuk pertukaran data dan algoritma antara dua sistem yang berbeda. Protokol HTTP biasanya digunakan sebagai sarana transpor. Hal ini lah yang menjadi dasar perancangan layanan berbasis Internet. Dalam sistem *cloud computing* , terdapat sebuah istilah yang bernama RESTful[2] API yang sering digunakan. Istilah ini memiliki artian *Representational State transfer* yang mengubah data dalam sebuah layanan berbasis Internet menjadi sebuah *resource* yang memiliki URI yang unik. Penggunaan protokol HTTP *resource* ini dapat diciptakan, dibaca, di *update*, dan di hapus dan dimanipulasi dalam sebuah layanan berbasis *web* [8].

## 2.3. Mekanisme Cloud Computing.

### 2.3.1 Infrastructure as a Service (Iaas)

Konsep ini secara umum didefinisikan sebagai penggunaan infrastuktur komputer sebagai sebuah *service*. *Service* ini mencakup kemampuan dasar komputer, basis data, jaringan, load balancer, dan lainnya. Hal ini mengurangi keperluan dari sebuah perusahaan untuk memiliki sebuah *datacenter*.Dibanding membeli seperangkat *server*, *software*, ruangan *datacenter*, atau perangkat jaringan, sebuah perusahaan dapat mengakses kebutuhan itu dari sebuah penyedia layanan *Cloud Computing*. Service yang dijual oleh perusahaan ini biasanya berupa konsep *pay as you go* dan jumlah berapa *resource* yang digunakan.

IaaS dapat berjalan pada klien dengan bantuan software APIs.Pada umumnya *RESTful* dan SOAP APIs *software* digunakan untuk mengkses infrastruktur ini. IaaS sangat bergantung kepada teknologi *virtualization* pada *platform* untuk menjalankan sebuah *virtual machine* tertentu.

Implementasi dari IaaS dapat termasuk pada jaringan komputer yang berupa *firewall*, *load balancer* yang dibutuhkan pada *datacenter* untuk menjamin tingkat *security* dan performa yang tinggi dari sebuah aplikasi. Penyedia layanan *cloud* seperti Amazon menciptakan komunitas pengguna yang besar dimana para pengguna service Amazon ini telah menciptakan sebuah konsep API yang dijalankan pada *web service*. Perusahaan lainnya seperti Google menyediakan tampilan antarmuka yang berupa *browser*, yang memberikan keleluasaan bagi pengguna untuk membangun sebuah infrastuktur dengan sistem *drag-and-drop* widget yang melambangkan CPU, basis data, media penyimpanan, *loadbalancer, firewall*, dan lainnya [2].

Berbagai penyedia layanan ini membedakan layanan mereka pada jumlah Sistem Operasi pada *platform* yang mereka *support*, *software* yang diinstalasi pada sistem operasi, harga, dan level *service agreement*.

### 2.3.2 Platform as a Service (PaaS)

*Platform as a service* (Paas) adalah sebuah konsep komputasi dimana seluruh fasilitas yang ada dibutuhkan untuk memenuhi seluruh *life cycle* dari sebuah aplikasi, yaitu membangun aplikasi, melakukan test aplikasi, dan finalisasi serta penyelesaian sebuah aplikasi dalam jaringan Internet dari sebuah *cloud*, tanpa membutuhkan proses *download* dan instalasi *software* dari pihak pengembang dan pengguna. Menurut wikipedia, hal ini disebut *cloudware*. PaaS adalah langkah berikutnya dari sistem yang kita kenal sebagai *mashups*. Pengguna dapat menciptakan aplikasi dengan menambahkan fitur fitur yang tersedia dari *google maps*, *google calendar* dan *web service* lainnya untuk menjadikan fitur ini sebagai bagian dari aplikasi mereka. Dalam PaaS seorang pengembang tidak perlu menulis setiap *source code* pada setiap aplikasi yang ingin dikembangkan, dikarenakan hal ini dapat dilakukan dengan metode logis dan tervirtualisasi seperti yang disediakan oleh *platform*.

Sistem PaaS berisi fasilitas dan perangkat yang dibutuhkan untuk merancang sebuah aplikasi, mengembangkan aplikasi, *testing*, pemasangan aplikasi, dan *hosting*, sejalan dengan *applicationservices* seperti *team collaboration*, integrasi *web service*, integrasi *database*, *security*, skalabilitas, penyimpanan, dan manajemen aplikasi. Konsep ini serupa jika diumpamakan sebagai versi web dari bahasa visual basic, dimana pengembang dapat merancang sebuah aplikasi secara visual serta menambah kode sesuai dengan kebutuhan.

Dalam sebuah konsep tradisional dari pengembangan sebuah *software*, aplikasi ditulis dalam sebuah sistem, melalui tahap testing pada sistem lainnya, dan dipasangkan pada sistem lainnya guna distribusi. Selain dana besar yang dibutuhkan untuk membangun, konfigurasi, dan perawatan dari sistem yang berbeda ini, aplikasi harus selalu dimonitor,dimana dana lain dibutuhkan untuk hal tersebut. Dalam sistem PaaS, seluruh *software lifecycle* dilakukan pada sebuah sistem yang sama, dan dana yang dikeluarkan akan berkurang secara signifikan dalam bidang pengembangan dan perawatan, distribusi, dan risiko pengerjaan. PaaS memberikan keleluasaan kepada pengembang untuk menciptakan *software*

yang mereka inginkan, tanpa harus memikirkan sistem sistem pendukung yang harus diciptakan, konfigurasi yang rumit, dan biaya yang besar [6].

Karakteristik lain dari PaaS adalah fakta bahwa PaaS memberikan integrasi kepada *Web Services* lainnya yang tidak tercakup dalam PaaS. Sebagai contoh, aplikasi yang dikembangkan dengan basis Google *AppEngine* dapat dihubungkan dengan aplikasi lain yang berada pada *Cloud*.

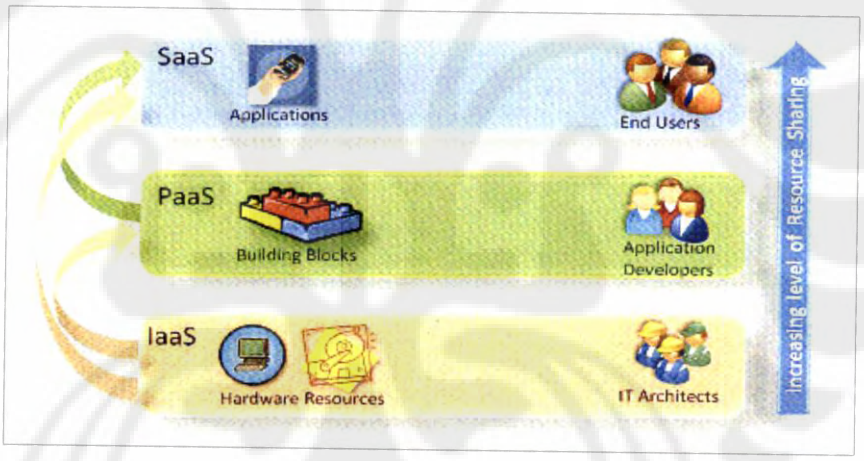
### 2.3.3 Software as a Service (SaaS)

Konsep ini dapat didefinisikan sebagai proses *delivery* dari sebuah aplikasi melalui Internet. Aplikasi yang diciptakan sebagai aplikasi web atau *service* dapat di akses oleh pengguna melalui antarmuka *browser* atau antarmuka yang disediakan oleh penyedia layanan tersebut. Beberapa *service* yang ditawarkan tersedia secara gratis. Sebagian besar aplikasi yang berbasis *client server* dapat diciptakan melalui konsep ini. *Browser* berfungsi sebagai klien. *Service* lainnya, dan aplikasi non *browser* juga bertindak sebagai klien. Sebuah service dapat diletakkan pada *datacenter* manapun selama hal tersebut tidak terhubung ke jaringan. Perusahaan seperti Google, Microsoft dan Yahoo menyediakan *service* umum seperti *mail*, kalender, *mapping*, dan lainnya dari *datacenter* mereka [7].

Konsep SaaS mengeliminasi kebutuhan untuk meng*install* dan menjalankan sebuah aplikasi pada sisi pengguna. Konsep ini juga mengurangi beban pengguna dalam hal perawatan aplikasi dan *support*. Sisi negatifnya terletak pada hilangnya kemampuan pengguna untuk merubah versi *software*. Problem utama dari *upgrade* dari versi sebuah *software* secara konvensional adalah permasalahan kompabilitas data terhadap versi *software* yang baru. Dalam konsep SaaS, data disimpan bersama dengan *software* pada superkomputer yang dimiliki oleh penyadia layanan. Merupakan tanggung jawab dari pihak penyedia layanan untuk memastikan jika ada perubahan versi dari sebuah aplikasi, data yang ada dapat kompatibel secara penuh dengan versi baru aplikasi tersebut.SaaS mengurangi biaya yang harus dikeluarkan pengguna untuk membeli sebuah software dengan konsep penyewaan bulanan yang dimiliki penyedia layanan. Jika *software* digunakan sebagai *service*, maka *software* tersebut tidak dapat dibajak.

Dari sisi vendor SaaS memiliki kemampuan yang sempurna untuk memberikan proteksi yang maksimal kepada properti intelektual yang dimiliki [8].

Aplikasi seperti *Customer Relationship Management* (CRM), *video conferen*ce, sumber daya manusia, IT *service management*, *accounting*, keamanan IT, *web analytics*, manajemen *web content*, *e-mail*, dan kalender merupakan aplikasi yang menjadi pilar kesuksesan dari konsep SaaS. Beberapa penyedia layanan SaaS telah mengembangkan aplikasi baru seperti *Billing as a Service*, dan *Monitoring as a Service*. Perbedaan antara SaaS dan aplikasi berbasis Internet sebelum generasi SaaS, adalah SaaS dikembangkan secara spesifik untuk memaksimalkan teknologi *web* seperti *browser*, dan menyediakan antarmuka alternatif kepada pengguna. Pengguna tidak perlu menginstall *source code* setiap aplikasi. Dengan SaaS aplikasi kini merupakan sebuah *service* yang dapat diakses dari browser dan melalui APIs dari *service* lainnya.



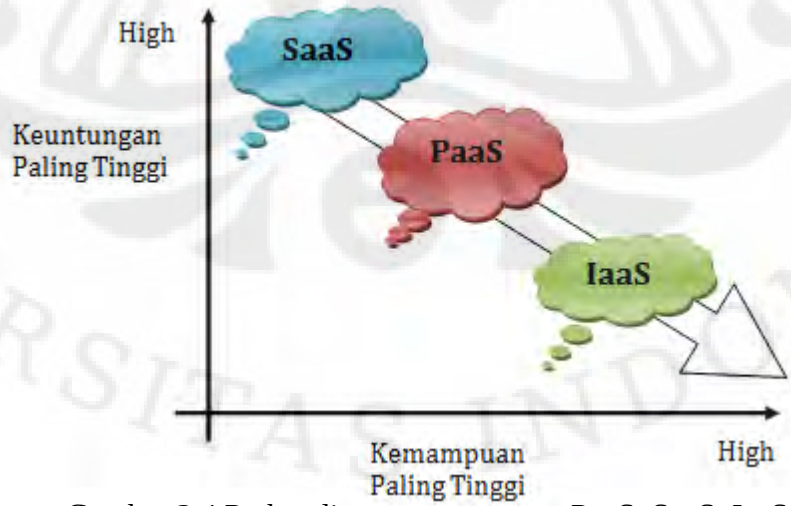

Gambar 2.3 SaaS, PaaS, IaaS [1].

Dengan klasifikasi dari mekanisme *cloud computing* dari tipe *resource* yang disediakan seperti disebutkan diatas, dapat kita lihat bahwa implementasi dari SaaS dapat diletakkan dengan basis layanan IaaS maupun PaaS. Dengan semakin tinggi level pelayanan, dari PaaS ke IaaS level pembagian *resource* dan level utilisasi *resource* semakin meningkat. Pada level IaaS, pengguna dapat menciptakan dan menjalankan seluruh sistem operasi yang di*support*, untuk kemudian menginstall seluruh *software* yang mereka butuhkan untuk digunakan. Pada level PaaS pengembang menggunakan *service* untuk menciptakan aplikasi

menggunakan *software development kit* yang dibutuhkan namun terbatas dari segi antarmuka, bahasa dan fitur yang ditawarkan oleh penyedia layanan PaaS. Pada level SaaS, pengguna akhir dibatasi untuk menggunakan aplikasi secara spesifik yang ditawarkan oleh penyedia, dan memiliki kapabilitas yang terbatas untuk memodifikasi antarmuka layanan tersebut [11].

Dalam hal keuntungan setiap level, kita dapat melihat keuntungan yang terbesar ada pada level SaaS. Level SaaS menyediakan kemungkinan yang paling tinggi untuk penggunaan *sharing resource* komputer, dari mulai *hardware* sampai *software* yang di *share* ke beberapa user. User tidak perlu membeli lisensi dengan harga yang tetap, dan hanya perlu membayar lisensi per penggunaan ketika mereka menggunakan aplikasi tersebut. Tidak ada *upgrade* maupun *patch* yang harus dikhawatirkan oleh pengguna dan tidak ada perawatan yang dibutuhkan untuk *backup security* dan lainnya. Dari sisi pengembang aplikasi, melakukan *patching*dan *upgrade* menjadi semakin mudah dikarenakan hanya *upgrade* tersentralisasi yang perlu dilakukan. Hal ini memungkinkan proses *patch* dan *upgrade* lebih efektif dan efisien [3].

Pada level IaaS, pengguna mendapatkan keuntungan dengan mampu menggunakan *resource hardware* sesuai dengan kebutuhan penggunaan. Pada level ini merupakan hak pengguna untuk menggunakan *software* yang dibutuhkan untuk mengkonfigurasi dan menggunakan metode ini sesuai yang diinginkan.




Gambar 2.4 Perbandingan penggunaan PaaS, SaaS, IaaS

Gambar diatas menjelaskan tentang perbandingan antara keuntungan yang didapat dengan penggunaan *cloud* dibanding dengan kemampuan paling tinggi yang dapat dilakukan dalam sebuah arsitektur *cloud.*

### 2.3.4 Kemiripan Fitur

Ketiga layanan *Cloud Computing* yang disebutkan diatas memiliki beberpa kesamaan fitur.

Fitur utama dari layanan *cloud computing* yang berupa kemudahan untuk tidak perlu menginstalasi atau meng*update* aplikasi atau *software* yang digunakan. Layanan yang digunakan tersedia dalam konsep *on demand*. *Delivery* dari sebuah aplikasi lebih menyerupai konsep *one to many* dibandingkan konsep *one to one*, termasuk arsitektur, harga penggunaan, dan manajemen [4].

Aktivitas dari sebuah *Cloud Computing* diatur dari satu atau lebih lokasi, bukan dari tempat setiap pengguna. Hal ini juga berarti seluruh fitur dari aplikasi di *update* dari satu lokasi sentral tanpa pengguna harus melakukan apapun. Hal ini mencakup *upgrade* dan *patch*.

Penggunaan dari layanan *cloud* harus dapat diukur. Atau layanan ini tidak dapat menggunakan konsep harga *pay per use*. Dalam setiap waktu utilisasi dari setiap *resource* dapat dimonitor dari sisi pengguna. Seluruh hal terkecil dari sebuah utilisasi *resource* juga perlu untuk dapat dimonitor merupakan hal yang sangat penting. Hal ini membuat penyedia layanan dapat merubah dasar harga yang dikenakan ke pengguna tanpa harus merubah *software monitoring* secara keseluruhan [4].

Dari sisi pengguna layanan *cloud computing*, sampai saat ini belum ditemukan sebuah permasalahan yang berarti. Untuk mengantisipasi kegagalan sistem, dibuat sebuah *backup* terhadap aplikasi yang siap untuk mengambil alih sistem tanpa ada delay (disebut dengan *failover*). Karena hal tersebut, dalam sistem ini terdapat perjanjian bahwa segalanya harus memiliki *backup*, dikarenakan ketika adanya kegagalan sistem, dan *backup* menjadi aplikasi utama, sistem akan menciptakan *backup* baru, dan mempertahankan reabilitas dalam jaringan tersebut [14].

Salah satu fitur penting dari keseluruhan sistem *cloud computing* adalah fakta bahwa klien–klien berbeda mengakses infrastruktur layanan yang sama tanpa harus mengenal satu sama lain. Hal ini disebut dengan *Multi Tenancy.* Dengan arsitektur *multitenant*, sebuah aplikasi dirancang untuk mempartisi konfigurasi dan data yang dimiliki untuk setiap pengguna yang menggunakan aplikasi tersebut, hal ini dapat diumpamakan dengan beberapa orang yang tinggal pada rumah yang sama.

## 2.4 Inovasi *Software Lifecycle*

Perbedaan antara *cloud computing* dan aplikasi *software* konvensional, adalah perubahan dan inovasi yang diciptakan dengan lahirnya teknologi baru ini, kepada metode pengembangan *software* secara umum.

Salah satunya, pada metode pengembangan *software lifecycle* tradisional, baik *agile* maupun *waterfall*, akan menjadi lebih singkat ketika *cloud computing* diadaptasi kepada metode tersebut. Hal ini dimungkinkan karena beberapa fungsi yang dijalankan pada satu mesin kini dapat disediakan oleh penyedia layanan *cloud computing*. Hal ini meliputi *instalasi, patching, upgrade*, dan lainnya. Sebagian besar permasalahan yang meliputi performa dan skalabilitas dapat diserahkan kepada penyedia layanan *cloud*, terlebih jika sebelumnya *software* hanya dikembangkan oleh satu mesin [13].

Proses penjaminan mutu juga menjadi lebih pendek dalam beberapa kasus. Secara khusus jika perusahaan yang sebelumnya mengembangkan *software* secara lokal pada satu mesin, memutuskan untuk menggunakan layanan *cloud*. Penyedia layanan *cloud* biasanya menyediakan *sandbox* ketika proses *testing*, dimana dengan adanya sistem tersebut tidak ada waktu yang terbuang. Perusahaan yang memindahkan aplikasi mereka ke *cloud*, dapat melakukan tes kecocokan dengan sistem yang baru tersebut dengan mudah, tanpa harus membeli sebuah hardware fisik apapun. Sebagai contoh, jika aplikasi akan digunakan pada *platform* Amazon EC2, pengguna dapat memasang aplikasi pada sebanyak mungkin *platform* EC2, dan melakukan test pada *platform- platform* tersebut, dan sistem akan sevara

otomatis menghilangkan *platform–platform* itu dan mengeluarkan hasil test. Hal ini menghemat banyak waktu, *resource*, dan biaya dari sisi pengembang [4].

Sebagai contoh, pada *SalesForce.com*, *software* dapat dibuat dengan waktu paling lama 90 hari. Masa pengembangan itu akan dibagi dengan 3 tahap. Pada 30 hari pertama, aplikasi akan dirancang dan diimplementasikan. Pada 30 hari berikutnya tim pengembang akan membahas sisi bisnis dan visual dari *software* ketika *software* selesai dikembangkan, tidak diperlukan proses *rebuild* dan lainnya. Pada 30 hari terakhir merupakan proses penambahan fitur fitur tambahan yang ingin diimplementasikan pada *software*, diluar fungsi utama dari *software* tersebut. Hal ini merupakan sesuatu yang mustahil dilakukan dengan konsep *software lifecycle* tradisional [4].

Diantara konsep *waterfall* dan *agile* pada konsep perancangan sebuah *software*, konsep *agile* lebih condong untuk mudah diaplikasikan dengan teknologi baru, seperti SaaS.Konsep *Agile* memungkinkan terciptanya sebuah *software* dengan *resource* yang minimum, dengan rencana jangka pendek, dibanding dengan rencana jangka panjang. Kelebihan *agile* tersebut meliputi seluruh tahapan *development cycle*, termasuk *planning, requirements analysis, design, coding, unit testing,* dan *acceptance testing* ketika *software* diujicobakan dihadapan pemegang saham. Hal ini sangat cocok pada layanan *cloud*, dikarenakan infrastuktur yang dibutuhkan untuk tes, pemasangan maupun validasi diurus sepenuhnya oleh *provider*. Layanan *cloud*, setidaknya dalam teori, akan selalu mampu untuk membuat *software* tersebut bekerja dan memiliki kapabilitas yang tak terbatas [5].

Ketika *software* ingin dikembangkan berbasis *cloud*, sebuah perusahaan harus memikirkan tentang strategi *deployment* pada awal *software lifecycle*, yaitu pada proses *design*. Hal ini dikarenakan penyedia layanan *cloud* mungkin memiliki *resource* yang terdiri dari beberapa set API yang bersifat tetap, dan setiap *software* yang dikembangkan dalam *cloud* harus memenuhi API tersebut. *Software* harus memiliki algoritma tambahan dimana memungkinkan digunakannya *virtual machine* atau *virtual storage* ketika semakin banyak data yang dihasilkan oleh *software* tersebut. Hal ini merupakan sesuatu yang kontras

dengan pengembangan *software* secara tradisional dimana situasi tersebut lebih sederhana [6].

### 2.5 CloudSim 1.0 Simulation

Pada level *Infrastructure as a Service,* terdapat sebuah *software* simulasi *cloud computing* yang diciptakan oleh Dr. Rajkumar Buyya, seorang pengajar di University of Melbourne, Australia. Software ini secara khusus mensimulasikan proses pengiriman data pada *datacenter* menuju host. Dalam *software* ini, terdapat beberapa metode simulasi, yakni:

- Satu *datacenter* dengan satu *Host* yang menjalankan satu *Cloud*
- Satu *datacenter* dengan satu *Host* yang menjalankan dua *Cloud*
- Satu *datacenter* dengan dua *Host*, yang menjalankan dua *Cloud*
- Dua *datacenter* dengan masing-masing satu *Host*.
- Simulasi *Scalable*

*Software* simulasi ini, berawal dari masa *Grid Computing* yang merupakan basis dari teknologi *Cloud Computing*. Pada masa *Grid Computing*, banyak *software* simulasi yang diciptakan, antara lain *GridSim*, *SimGrid*, dan *GangSim*. *Software* ini, dikemudian hari, tidak mampu mensimulasikan infrastuktur teknologi *Cloud Computing*, dikarenakan kurangnya kapabilitas. Oleh sebab itu, *CloudSim* diciptakan. *CloudSim* diciptakan dengan mengambil basis fundamental *GridSim*, dan menambah fungsionalitas utamanya agar dapat mengadaptasi layer *Cloud Computing*. Layer pada *CloudSim* mensimulasikan *environment* pada sebuah datacenter pada *cloud*, yakni *Virtual Machine, Memory, Storage* dan *Bandwidth*. Dengan menggunakan *CloudSim*, para developer teknologi ini dapat melakukan tes pada skenario dan konfigurasi tertentu, dengan tujuan pengembangan teknologi *cloud computing* secara keseluruhan [15].

# BAB 3 PERANCANGAN DAN SIMULASI CLOUD COMPUTING

## 3.1 *CloudSim* Simulation

Simulasi teknologi *Cloud Computing* yang akan dilakukan pada Tugas Akhir ini, akan dilakukan dengan mengambil dasar dari simulasi *CloudSim*, yang berjalan diatas *platform* Java. Simulasi *CloudSim* ini, nantinya akan ditambahkan dengan fungsi *heat dissipation* dan *power consumption* pada setiap eksekusi yang dijalankan. Sebagai percontohan pada simulasi ini, akan digunakan sebuah konsep *GreenCloudSimulation*, yaitu simulasi *resource* yang ada pada *Cloud Computing*.

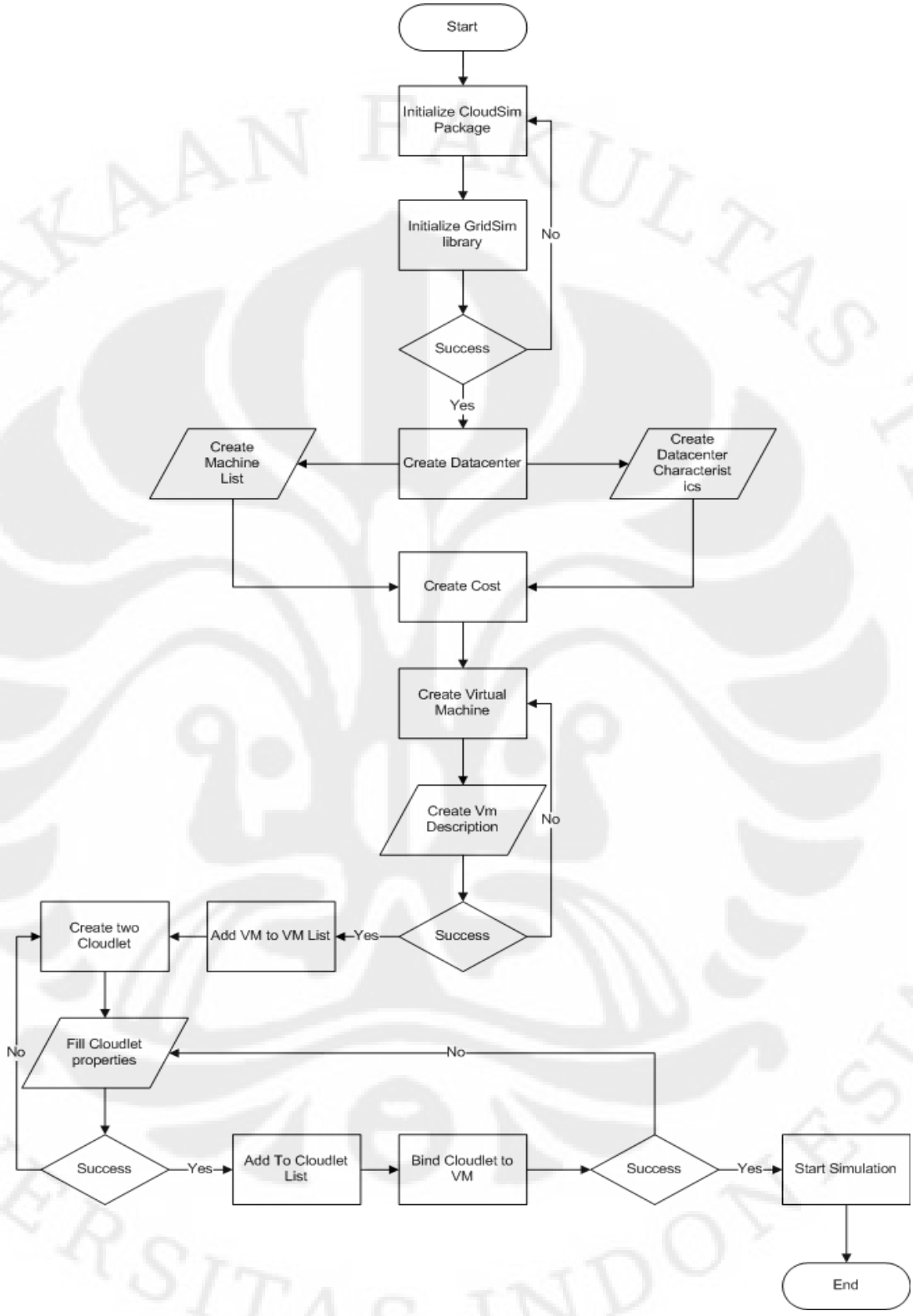
### 3.1.1 Perancangan Simulasi *CloudSim*

*CloudSim* merupakan simulasi yang terdiri dari berbagai modul, dan kemudian pada penggunaannya, modul tersebut akan dipanggil untuk mengeksekusi fungsi fungsi tertentu yang ada pada program. Adapun modul yang tersedia pada *CloudSim* adalah:

*CloudSim*.Cloudlet;
*CloudSim*.CloudletList;
*CloudSim*.DataCenter;
*CloudSim*.DatacenterBroker;
*CloudSim*.DatacenterCharacteristics;
*CloudSim*.Host;
*CloudSim*.SimpleBWProvisioner;
*CloudSim*.SimpleMemoryProvisioner;
*CloudSim*.SimpleVMProvisioner;
*CloudSim*.TimeSharedWithPriorityAllocationPolicy;
*CloudSim*.TimeSharedVMScheduler;
*CloudSim*.VMCharacteristics;
*CloudSim*.VirtualMachine;
*CloudSim*.VirtualMachineList;

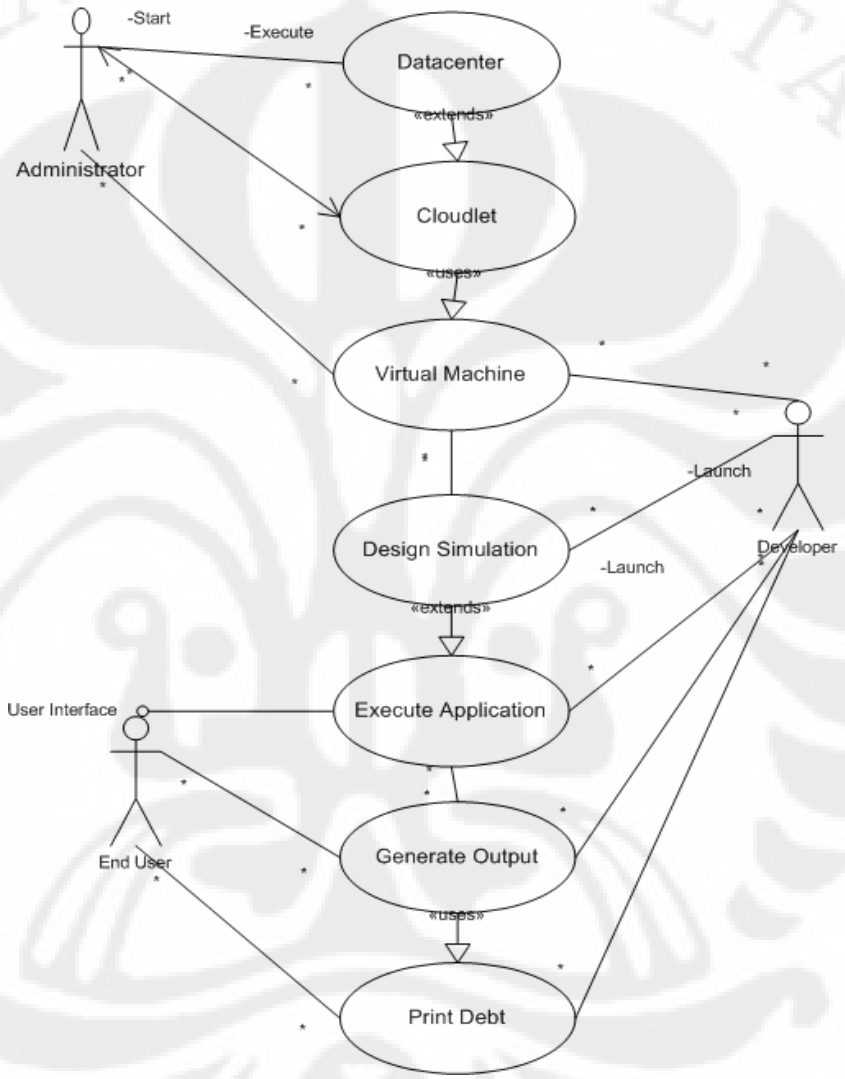
Gambar 3.1 Modul yang ada pada *CloudSim*

Tiap modul ini mewakili atribut dan fungsi yang ada pada simulasi *CloudSim*. Keseluruhan modul memiliki format Java, dan dapat berjalan menggunakan *compiler* dan ekstraktor *Java*. Selain modul tersebut pada program simulasi, akan ditambahkan input input manual yang berbeda untuk setiap simulasi, seperti *hardware* yang digunakan, *bandwidth* yang dibutuhkan, dan sebagainya.




Gambar 3.2 Diagram Alir *CloudSim*

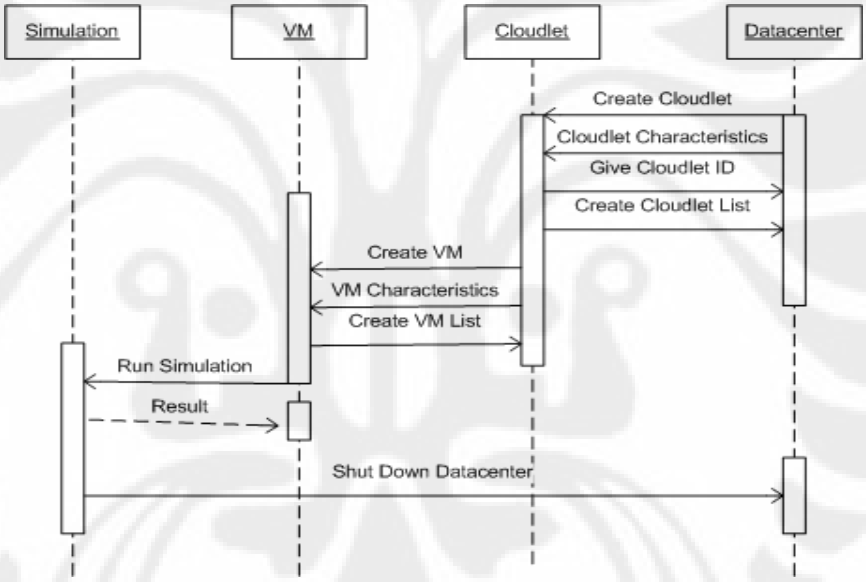
Dapat dilihat dari diagram diatas, dijelaskan alur *CloudSim* dari mulai proses inisialisasi hingga proses eksekusi program. Keseluruhan perjalanan program *CloudSim* menggunakan konsep yang serupa, yakni inisialisasi paket, ekstraksi *Cloudlet*, dan pembuatan VM atau *Virtual Machine*.




Gambar 3.3 *Use-Case* diagram Program *CloudSim*

Gambar diatas merupakan *use-case scenario* pada *CloudSim*, dapat terlihat ada tiga elemen utama pada sebuah arsitektur *cloud*, yakni *administrator datacenter*, pengembang program, dan *end user*. *Administrator* secara garis besar

bertugas sebagai pengoperasi perangkat pada *datacenter*, sedangkan pengembang bekerja pada perancangan simulasi, pembuatan *Virtual Machine*, dan membuat *output* yang terkait dengan simulasi tersebut. Sedangkan *end user*, atau pengguna, adalah entitas terakhir yang hanya memiliki akses kepada *ouput* program. Dalam *use case diagram* tersebut juga dapat dijelaskan tentang mekanisme dasar dari program *CloudSim*, yakni perancangan simulasi seluruhnya dilakukan pada tahap *Virtual Machine* yang dijalankan oleh *cloudlet* pada *datacenter*, sehingga seluruh *Virtual machine* yang ada tidak langsung berada di bawah *datacenter*, seperti layaknya *datacenter* konvensional, melainkan terlebih dahulu melewati mekanisme *cloudlet*.




Gambar 3.4 *Sequence* diagram pada program *CloudSim*

Gambar diatas merupakan *sequence* atau langkah yang dibutuhkan dalam simulasi, dari tahap awal hingga tahap akhir, langkah tersebut dimulai dari *datacenter* yang membuat *cloudlet*, dan diteruskan dari *cloudlet* yang berisi berbagai VM, dan didalam VM tersebut simulasi dijalankan dan dikonfigurasi. Pada tahap akhir, simulasi memberi *command shutdown* kepada *datacenter* sekaligus memberikan *output* kepada pengguna

### 3.2 Elemen *GreenCloudSimulation* Pada *CloudSim*

Sebagai simulasi dari program *CloudSim*, dirancang sebuah simulasi berbentuk *GreenCloudSimulation*. *Green Cloud,* yang merupakan teknologi *cloud computing* yang ramah lingkungan, akan disimulasikan dengan bantuan dua jenis eksekusi *resource*, yakni *resource* yang membutuhkan spesifikasi tinggi dan rendah. Kemudian akan dibandingkan penggunaan daya pada kedua jenis *resource* tersebut.

#### *High Resource*

Pada simulasi ini akan dicontohkan eksekusi aplikasi yang membutuhkan *resource* tinggi, *resource* tersebut dapat berupa *storage*, RAM, dan prosesor .
Spesifikasi yang dibutuhkan:

- Prosesor 2.4 GHz,
- RAM 4 GB
- *Storage* 10 GB

```
private static List<Vm> createVms(int brokerId) {
    List<Vm> vms = new ArrayList<Vm>();


    int[] mips = { 250, 500, 750, 1000 };
    int pesNumber = 1;
    int ram = 4096;
    long bw = 2500;
    long size = 10000;
    int priority = 1;
    String vmm = "HighResource";
```

Gambar 3.5 Kode program pada Simulasi *High Resource*

***LowResource***

Pada simulasi ini akan dicontohkan eksekusi program yang membutuhkan resource normal atau rendah. Hal ini dibuat untuk membandingkan penggunaan konsumsi daya antara kedua jenis *resource* tersebut.

Spesifikasi yang dibutuhkan:

- Prosesor2.4 GHz
- RAM 1 GB
- *Storage* 4 GB

```
private static List<Vm> createVms(int brokerId) {
    List<Vm> vms = new ArrayList<Vm>();


    int[] mips = { 250, 500, 750, 1000 };
    int pesNumber = 1;
    int ram = 1024;
    long bw = 2500;
    long size = 4000;
    int priority = 1;
    String vmm = "LowResource";
```

Gambar 3.6 Kode program pada simulasi *LowResource*

Dan pada penggunaannya kedua program ini akan dihubungkan dengan konsep *Energy* dan akan dianalisa *output* dan kinerja nya sehubungan dengan konsep tersebut.

## 2.5 Elemen *Energy* pada *CloudSim*

Salah satu topik yang cukup diperhatikan oleh kalangan pakar teknologi informasi dewasa ini adalah konsep teknologi informasi dan komunikasi hijau.Untuk konsep itu dibuat sebuah modul baru yang berupa modul konsumsi tenaga dan disipasi panas terhadap simulasi yang dijalankan pada *CloudSim*. Kedua elemen ini dibuat dengan memodifikasi *source code CloudSim* yang ada, dan menambahkannya ke dalam program, sedemikian rupa hingga program yang ada, akan menampilkan konsumsi tenaga dan disipasi panas yang terjadi.

### 3.3.1 Power Consumption

Elemen *power consumption* pada simulasi ini didapatkan dengan memanfaatkan modul *power* yang ada pada *CoudSim*, yaitu menambahkan kode berbasis MIPS pada simulasi *CloudSim*. MIPS, atau *Million Instruction Per Second*, merupakan jumlah kinerja *resource* pada simulasi tersebut. Langkah berikutnya yakni mengalikan koefisien MIPS yang ada pada datacenter ke waktu yang digunakan untuk melakukan simulasi. Hasilnya merupakan koefisien tenaga yang dibutuhkan untuk menjalankan simulasi tersebut.

```
        if (mips.get(0) != 0) {
            numberOfAllocations++;
            totalRequested += mips.get(0);
            totalAllocated += mips.get(1);
            double _sla = (mips.get(0) - mips.get(1)) / mips.get(0) * timediff;
            if (_sla > 0) {
                sla.add(_sla);
            }
        }
    }
```

Gambar 3.7 Kode program modul Power Consumption

Tahap selanjutnya yakni melakukan penambahan fungsi ini kepada output akhir program simulasi. Hasil yang telah didapat dari MIPS yang ada ditampilkan pada hasil akhir dengan dibagi dengan koefisien jam. Sehingga hasilnya merupakan entitas dalam satuan jam. Hal ini berguna agar dapat dicari disipasi panasnya, yang akan dibahas pada bagian selanjutnya. Hasil keluaran konsumsi tenaga, dalam pengembangan selanjutnya akan sangat berguna untuk menerapkan prinsip Teknologi Informasi dan Komunikasi Hijau, dikarenakan dengan diketahuinya konsumsi tenaga yang dibutuhkan, akan dapat dilakukan efisiensi terhadap *resource* yang digunakan.

Nilai *Power Consumption* yang didapat, kemudian akan dicetak pada *output* simulasi program bersamaan dengan *ouput* lainnya, sekaligus nilai ini, akan menjadi basis untuk penghitungan disipasi panas.

### 3.3.2 Heat Dissipation

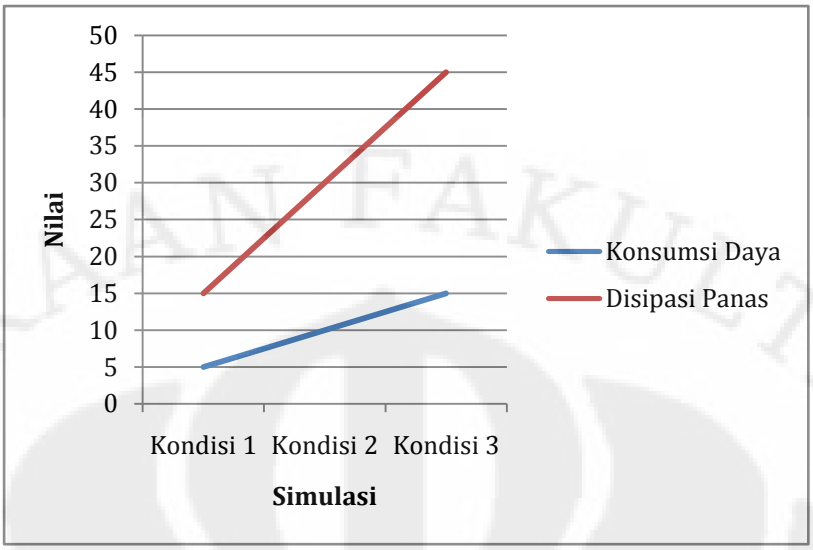
Disipasi panas yaitu jumlah panas yang terjadi pada sebuah perangkat keras yang harus diemitasi dan oleh perangkat tersebut. Prinsip disipasi panas

sepenuhnya bersumber kepada konsumsi tenaga yang digunakan, dan melakukan penghitungan berdasarkan satuan BTU, yakni *British Thermal Unit.* Sehingga satuannya berupa BTU/hr atau BTU per jam yang diemitasi oleh perangkat keras. Satuan dari BTU adalah : 1 Watt = 3.412 BTU [9].

```
for (Entry<String, List<List<Double>>> entry : datacenter.getUnderAllocatedMips().entrySet()) {
    List<List<Double>> underAllocatedMips = entry.getValue();
    double totalRequested = 0;
    double totalAllocated = 0;
    for (List<Double> mips : underAllocatedMips) {
        if (mips.get(0) != 0) {
            numberOfAllocations++;
            totalRequested += mips.get(0);
            totalAllocated += mips.get(1);
            double _sla = (mips.get(0) - mips.get(1)) / mips.get(0) * 100;
            if (_sla > 0) {
                sla.add(_sla);
            }
        }
    }
    totalTotalRequested += totalRequested;
    totalTotalAllocated += totalAllocated;
}


datacenter.printDebts();

Log.printLine();
Log.printLine(String.format("Total simulation time: %.2f sec", lastClock));
Log.printLine(String.format("Energy consumption: %.2f Watt/hr", datacenter.getPower() / ( 3600 )));
Log.printLine(String.format("Heat Dissipation: %.2f BTU/hr", (datacenter.getPower() / (3600)) * 3.412));
Log.printLine();
```

Gambar 3.8 Kode program modul Heat Dissipation

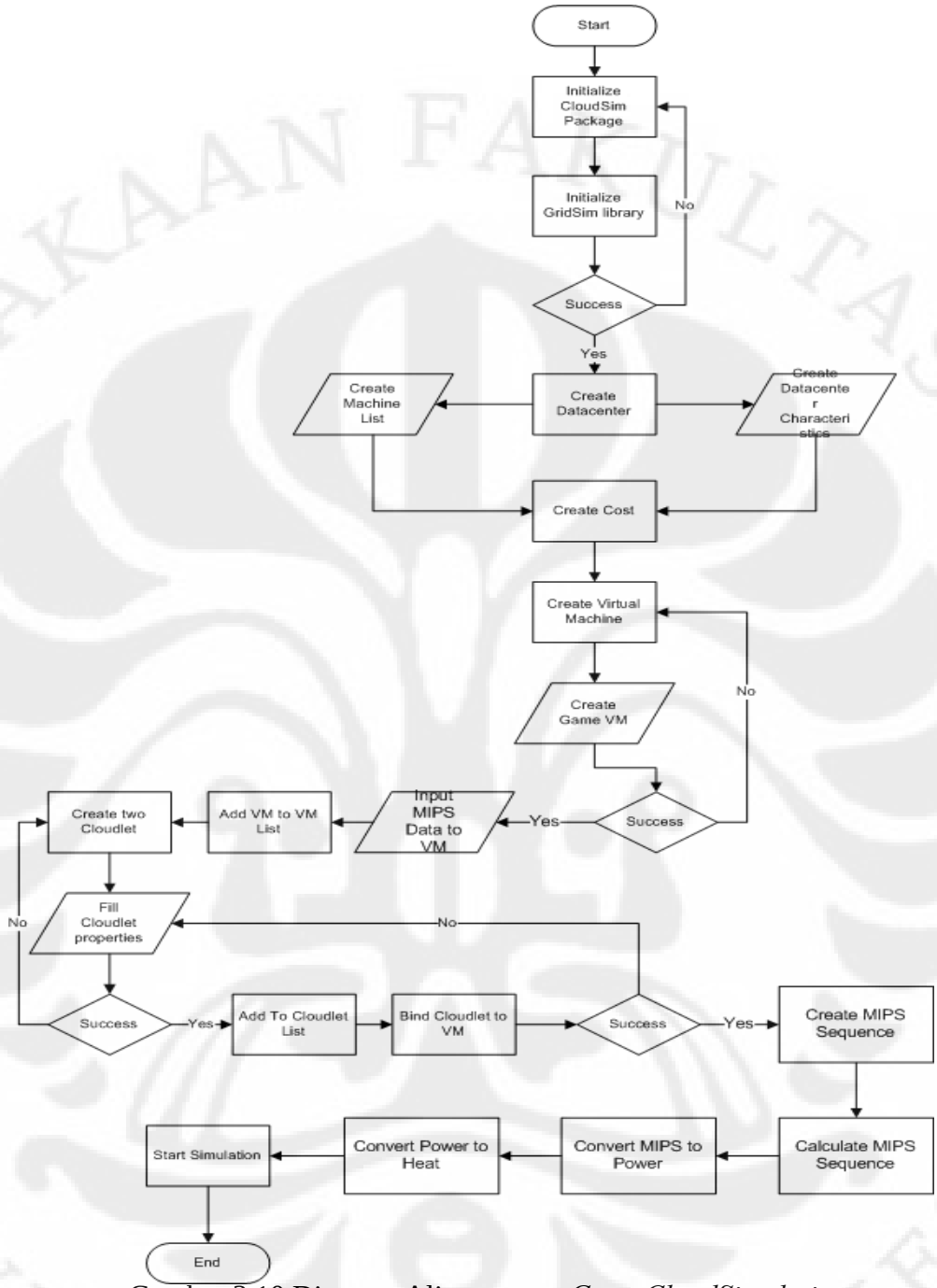
Seperti dapat kita lihat dari kode diatas, disipasi panas sepenuhnya bergantung kepada konsumsi tenaga. Dengan hubungannya dengan teknologi Informasi dan Komunikasi Hijau, disipasi panas memainkan peranan yang cukup penting. Disipasi panas menentukan jumlah panas yang diemitasi oleh perangkat keras dan harus diserap atau dibuang oleh pendingin udara. Oleh karena itu mengetahui jumlah disipasi panas yang ada merupakan hal yang cukup penting. Dengan diketahuinya disipasi panas dari sebuah perangkat keras maka akan dapat dilakukan efisiensi dan penyerapan panas lebih baik, dan apabila diperlukan dapat dilakukan migrasi dari satu *datacenter* ke *datacenter* lainnya pada jaringan *cloud* tersebut.




Gambar 3.9 Peningkatan searah Konsumsi Daya dan Disipasi Panas

Dapat dilihat dari grafik diatas, *heat dissipation* akan selalu bertambah mengikuti bertambahnya *power consumption*. Hal ini berbanding lurus dikarenakan *heat dissipation* didapatkan dari hasil kali *power consumption*, oleh karena itu setiap bertambahnya pemakaian daya, maka akan bertambah pula panas yang harus diemitasi oleh perangkat.

Elemen konsumsi tenaga, dan disipasi panas ini merupakan salah satu konsep dasar pada system *Green Cloud* . Sistem *cloud computing* yang berbasis kepada Teknologi Informasi dan Komunikasi Hijau dan ramah lingkungan. Oleh karena itu, mekanisme ini dapat dikembangkan nantinya ke berbagai bidang, dikarenakan nilai konsumsi tenaga dan disipasi panas telah diketahui, dapat dilakukan migrasi VM, dimana VM yang tidak penuh terisi, akan dimaksimalkan pengisiannya, agar tenaga yang digunakan pun lebih efektif [2].




Gambar 3.10 Diagram Alir program *GreenCloudSimulation*

Grafik diatas merupakan diagram alir program *GreenCloudSimulation* dimana pada diagram tersebut dapat terlihat elemen yang telah ditambahkan kepada program asli *CloudSim*, yakni berupa penghitungan MIPS yang menjadi basis dari penghitungan konsumsi tenaga dan disipasi panas pada program tersebut.

# BAB 4

# ANALISA SIMULASI CLOUD COMPUTING

Teknologi *Cloud Computing* yang tergolong teknologi baru dan mutakhir, tidak memiliki teknologi serupa yang dapat di komparasi terhadap teknologi tersebut, salah satu teknologi yang paling dekat hubungannya, adalah teknologi *Grid Computing*. Namun banyak perbedaan fundamental yang menjadi basis *cloud* dan *grid.*Untuk itu, sebagai metode analisa akan di analisa performa dan hasil keluaran berdasarkan aplikasi *CloudSim* yang telah dimodifikasi sedemikian rupa.

## 4.1 Analisa *CloudSim* dan Green*CloudSim*ulation

*CloudSim* menjadi basis dasar program *GreenCloudSimulation* yang dibuat, merupakan program *open-source* yang dibuat oleh Dr. Rajkummar Buyya. Untuk itu akan dianalisa performa program awal *CloudSim* terhadap performa program yang telah dimodifikasi ini, untuk dapat mengetahui apakah program baru yang telah dirancang memiliki performa lebih baik dari program lama.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    1        SUCCESS        2             1        105.1        0         105.1
    0        SUCCESS        2             0        210.11       0         210.11
*****PowerDatacenter: Datacenter_0*****
User id         Debt
3               71.2
************************************
CloudSimExample2 finished!
```

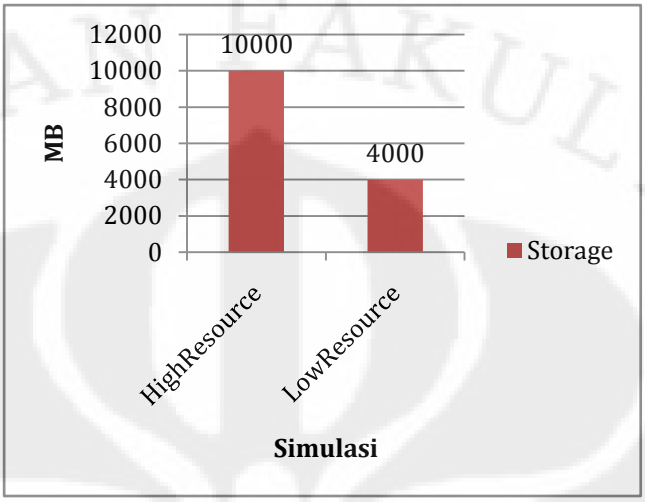
Gambar 4.1 Kode program *CloudSim*

Dapat dilihat dari gambar diatas, program ini memiliki *output* berupa:

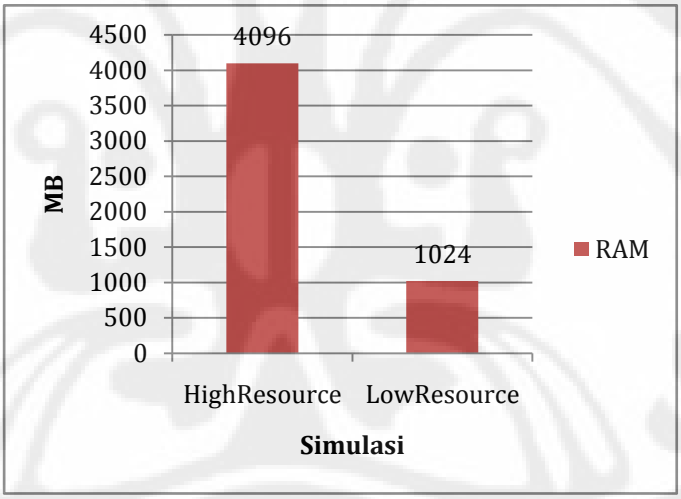
- *Cloudlet ID*
- Status
- *Datacenter ID*
- *VM ID*
- *Time*
- *Start Time*
- *Finish Time*
- *User ID*
- *Debt*

Hal yang dapat dicermati dari *output* diatas adalah *execution time*, yang merupakan waktu yang dibutuhkan program ini untuk dapat melakukan sebuah

eksekusi. Dalam program *CloudSim*, terlihat waktu yang dibutuhkan adalah 105 detik untuk melakukan sebuah instruksi pada sebuah mesin komputer.




Gambar 4.2 Perbedaan storage pada kedua simulasi




Gambar 4.3 Perbedaan RAM pada kedua simulasi

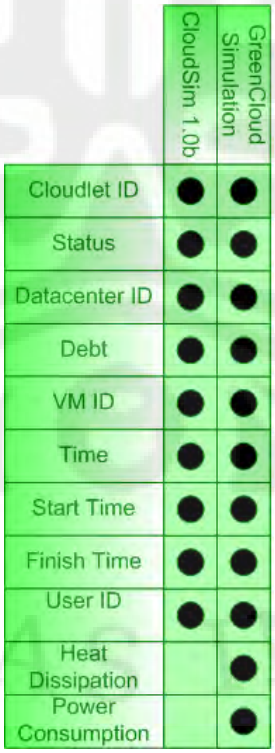
Dalam program Green*CloudSim*ulation yang telah dimodifikasi, dimasukkan fungsi tambahan berupa fungsi *Power Consumption*dan *Heat Dissipation* kedalam kode asal program *CloudSim*, setelah ditambahkan fungsi tersebut, ditambahkan juga konfigurasi perangkat keras yang dibutuhkan untuk menjalankan sebuah program pada jaringan *Cloud.*

```
========== OUTPUT ==========
Cloudlet ID     STATUS  Resource ID     VM ID   Time    Start Time      Finish Time
        1       SUCCESS         2       1       165     0               165
        0       SUCCESS         2       0       335     0               335
*****PowerDatacenter: HighResource*****
User id         Debt
3               429,6
**********************************

Total simulation time: 335,00 sec
**************************************
Heat Dissipation: 79,38 BTU/hr
**************************************
Power consumption: 23,26 Watt/hr
**************************************

HighResource Simulation Finished !!
```

Gambar 4.4 Kode program Green*CloudSim*ulation

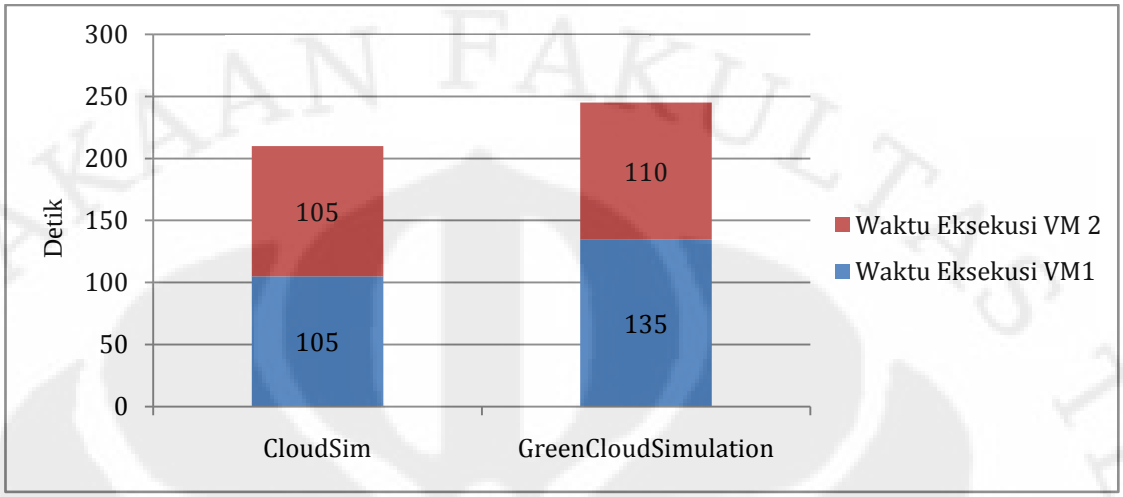
Dari gambar diatas dapat kita perhatikan, bahwa terdapat *output* berupa

- Cloudlet ID
- Status
- *Resource* ID
- *Virtual Machine* ID
- *Time*
- *Start Time*
- *Finish Time*
- *User* ID
- *Debt*
- *Heat Dissipation*
- *Power Consumption*




Gambar 4.5 Perbandingan Fitur *CloudSim* dan Green*CloudSim*ulation

Seperti terlihat pada gambar diatas, dijabarkan secara lengkap perbedaan fitur keluaran yang ada pada *CloudSim* dan *GreenCloudSimulation*.




Gambar 4.6 Perbandingan waktu eksekusi *CloudSim* dan Green*CloudSim*ulation

Dapat dilihat secara perbandingan waktu eksekusi pada grafik diatas, terlihat sedikit perbedaan, hal ini seluruhnya dikarenakan adanya tambahan fitur yang dieksekusi oleh *GreenCloudSimulation*, sehingga menyebabkan bertambahnya waktu eksekusi yang ada.

*Execution times* atau waktu eksekusi yang menjadi acuan akan berapa waktu yang dibutuhkan program ini, bertambah sekitar 20 detik, menjadi 135 Detik, hal ini merupakan hal yang dapat dimaklumi dikarenakan adanya penambahan fitur dan modul terhadap program ini.

### 4.1.1 Analisa Elemen Simulasi

Elemen simulasi yang dipilih, merupakan dua contoh eksekusi yang mungkin dijalankan pada *datacenter* berbasis Cloud. Dipilihnya kedua *program* ini, berdasar dari perbedaan signifikan akan perangkat keras yang dibutuhkan. *HigheResource* melambangkan *program* dan aplikasi yang membutuhkan *resource* tinggi. Sedangkan Program *LowResource* melambangkan program dan aplikasi yang membutuhkan *resource* normal. Secara keseluruhan eksekusi kedua program ini pada *cloud* akan dapat memberikan hasil yang jelas akan kinerja dari *datacenter*.

```
========== OUTPUT ==========
Cloudlet ID     STATUS  Resource ID     VM ID   Time    Start Time      Finish Time
        1       SUCCESS         2       1       165     0               165
        0       SUCCESS         2       0       335     0               335
*****PowerDatacenter: HighResource*****
User id         Debt
3               429,6
**********************************

Total simulation time: 335,00 sec
**************************************
Heat Dissipation: 79,38 BTU/hr
**************************************
Power consumption: 23,26 Watt/hr
**************************************

HighResource Simulation Finished !!
```

Gambar 4.7 Kode Program *High Resource*

Seperti terlihat pada gambar diatas, Program *High Resource* mengkonsumsi 23.26 Watt per jam tenaga listrik, dan mengeluarkan 23,26 BTU per jam disipasi panas, jika dijalankan pada sebuah *datacenter* berarsitektur *Cloud*.

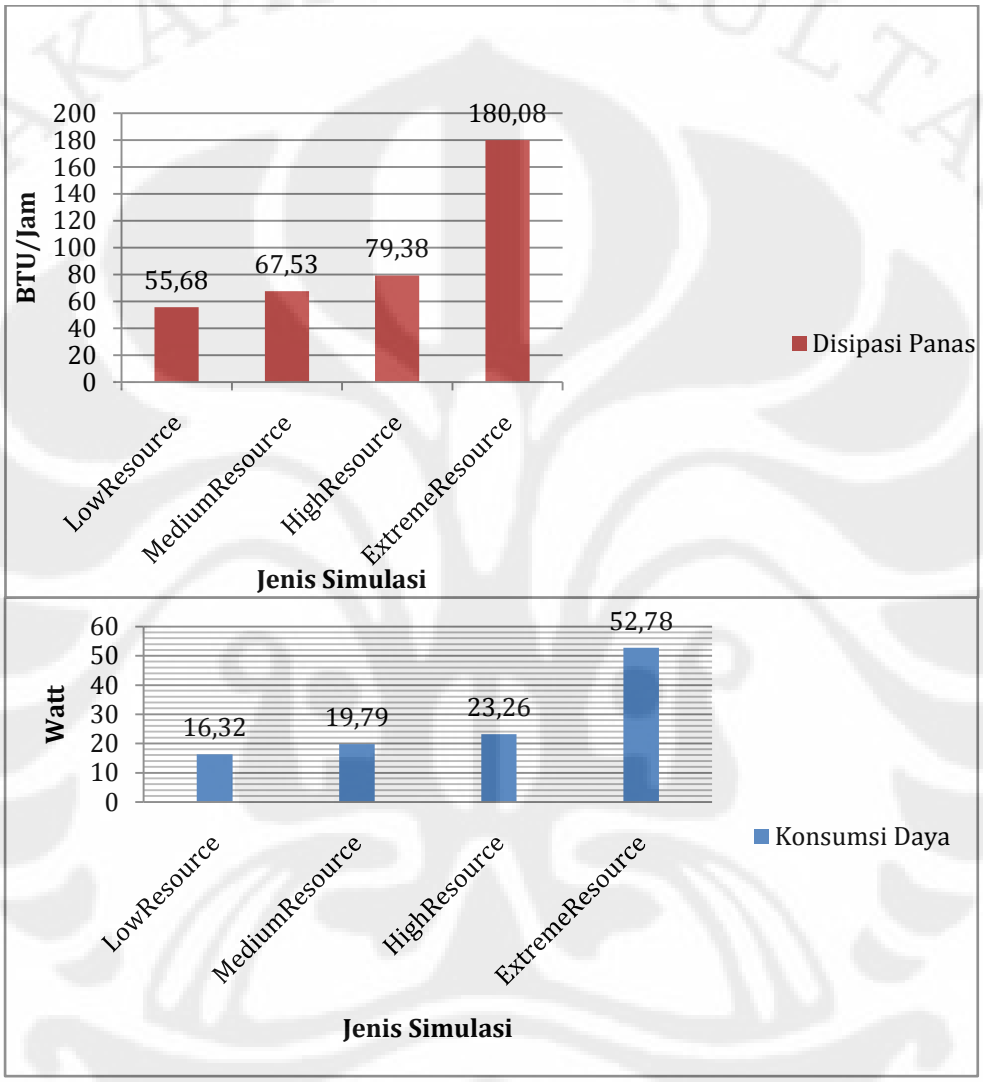
```
========== OUTPUT ==========
Cloudlet ID     STATUS  Resource ID     VM ID   Time    Start Time      Finish Time
        1       SUCCESS         2       1       120     0               120
        0       SUCCESS         2       0       235     0               235
*****PowerDatacenter: LowResource*****
User id         Debt
3               110,4
**********************************

Total simulation time: 235,00 sec
**************************************
Heat Dissipation: 55,68 BTU/hr
**************************************
Power consumption: 16,32 Watt/hr
**************************************

LowResource Simulation Finished !!
```

Gambar 4.8 Kode Program *LowResource*

Dari gambar diatas, terlihat program *LowResource* hanya mengkonsumsi 16.32 Watt perjam tenaga listrik, dan mengemitasi panas sebesar 55.68 BTU per jam, jika dijalankan pada sebuah *datacenter* berarsitektur Cloud.

Perbedaan yang terjadi cukup signifikan, hal tersebut terjadi antara lain dikarenakan program *HighResource* menggunakan *resource* dan tenaga yang jauh lebih besar dibanding dengan program *LowResource*. Sebagai contoh, program

*HighResource* menggunakan RAM sebesar 4096 Mb, sedangkan Program *LowResource* hanya menggunakan RAM sebesar 1024 Mb. Perbedaan yang terjadi pun bertambah dikarenakan besarnya MIPS yang diproses oleh program yang membutuhkan *resource* yang lebih besar.




Gambar 4.9 Grafik Perbedaan Energi Program

Dari grafik diatas, dapat dianalisa bahwa penggunaan *resource* yang besar akan berdampak kepada penggunaan tenaga yang besar pula. Namun hal ini tidak terjadi dengan sistem berbanding lurus. Sebagai contoh, *program* yang menggunakan RAM 4096 Mb, tidak serta merta membutuhkan tenaga empat kali lipat lebih banyak dari *program* yang membutuhkan tenaga sebesar 1024 Mb. Hal ini dapat terjadi dikarenakan penggunaan *resource* mengalami kenaikan tenaga

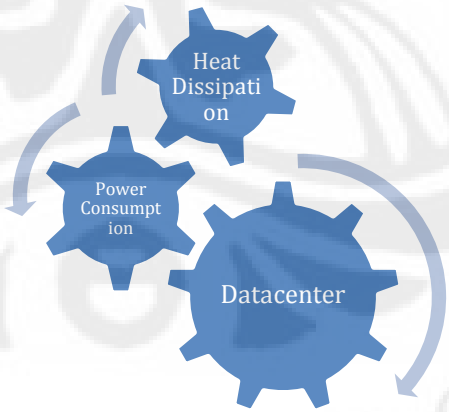
secara bertahap, tergantung dari daya yang dibutuhkan oleh masing masing perangkat.

Dalam grafik diatas diambil dua buah contoh lain sebagai metode pembanding dengan program ini, hal ini dilaukan agar mengetahui kenaikan panas dari simulasi ini. agar terlihat kenaikannya yang tidak berbanding lurus. Yaitu *MediumResource* dengan RAM 2048 dan *ExtremeResource* yang melakukan eksekusi sebanyak 100.000 MIPS.

Disipasi panas yang dihasilkan kedua perangkat tersebut pun memiliki perbedaan yang searah dengan perbedaan yanga da pada konsumsi tenaga, hal ini sepenuhnya terjadi karena disipasi panas sepenuhnya diambil dari konsumsi tenaga, sehingga perbedaan yang terjadi akan berbanding lurus dengan perbedaan pada konsumsi tenaga. Disipasi panas yang muncul sepenuhnya dapat diserap oleh *datacenter*, dikarenakan daya yang termasuk kecil, akan menghasilkan disipasi panas yang tidak besar pula.

### 4.1.2 Analisa Elemen *Energy*

Elemen baru yang ditambahkan pada simulasi ini adalah elemen *energy-friendly*, yaitu elemen *power consumption* sebagai penghitungan daya yang dibutuhkan, dan elemen *heat dissipasion*, sebagai elemen disipasi panas yang diemitasi oleh perangkat ini.

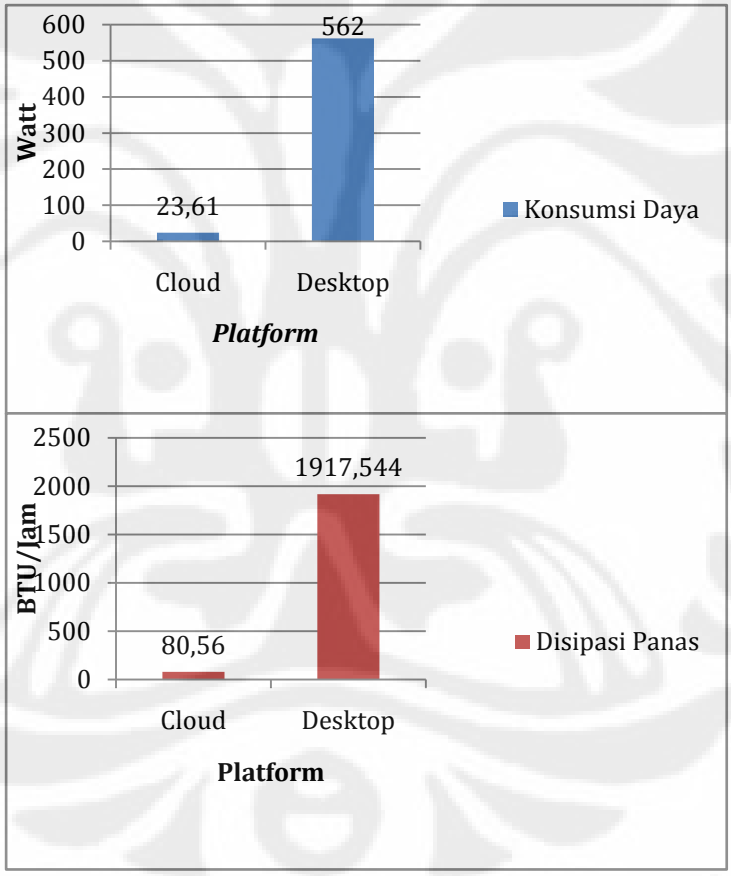



Gambar 4.10 Mekanisme modul energi yang bersinggungan

Elemen *energy friendly* pada *datacenter* yang digunakan, dapat dianalisa dan dibandingkan dengan jika sebuah aplikasi dijalankan secara tunggal pada

sebuah komputer dekstop, maka akan dapat terlihat perbedaannya, dikarenakan pada jaringan *cloud*, sebuah *hardware* dapat menangani puluhan VM sekaligus, sehingga *load resource* dapat dibagi dengan para pengguna layanan tersebut. Sebagai contoh, sebuah *desktop high – end* yang memiliki Prosesor Core 2 Quad, VGA Card 512 MB, dan RAM 4 GB, jika dijumlahkan kebutuhan daya yang tercetak pada kemasan produk tersebut, akan membutuhkan daya maksimum sebesar 562 Watt. Akan menghasilkan disipasi panas sebagai berikut [10].

Disipasi Panas = Konsumsi Daya x 3.142
= 562 Watt x 3.412
= 1917,544 BTU/Jam




Gambar 4.11 Perbedaan Konsumsi daya HighResource *Cloud* dan *Desktop*

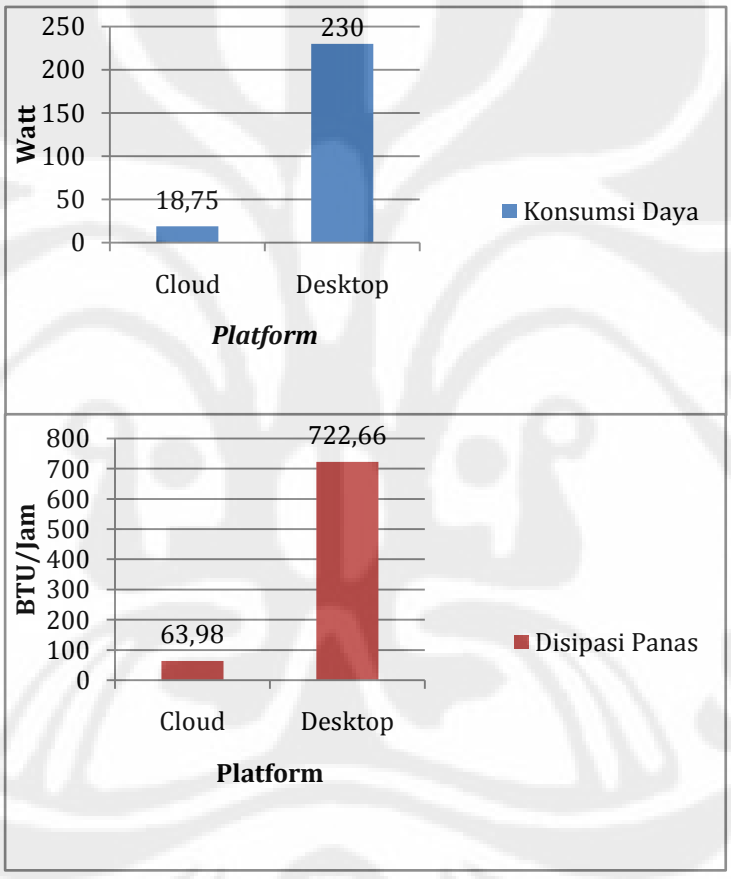
Sedangkan untuk konsumsi pengguna biasa, yakni dengan *load resource* yang kecil, dapat dibandingkan dengan desktop yang memiliki kapabilitas sebagai berikut, Prosesor Core 2 Duo, RAM 1 GB , dan VGA terintegrasi. Jika dijumlahkan kebutuhan daya desktop tersebut, adalah maksimal 230 Watt. [10]

Disipasi Panas = Konsumsi Daya x 3.142

= 230 Watt x 3.412

= 722,66 BTU/Jam

Jika dibandingkan dengan simulasi *LowResource,* tetap terlihat perbedaan yang sangat jauh beda konsumsi tenaganya, hal ini disebabkan karena banyak hal, antara lain seorang pengguna cloud tidak menanggung seluruh konsumsi daya sendiri, melinkan dibagi dengan pengguna cloud lainnya, berbeda dengan pengguna desktop, yang harus menanggung semua konsumsi daya yang digunakan.

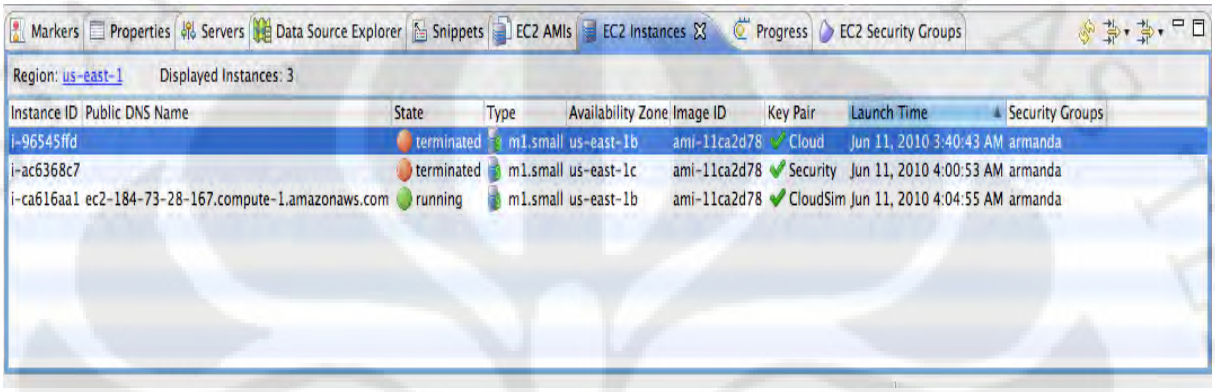



Gambar 4.12 Perbedaan Konsumsi daya *LowResource* Cloud dan Desktop

### 4.1.3 Analisa Eksekusi Program pada arsitektur Cloud

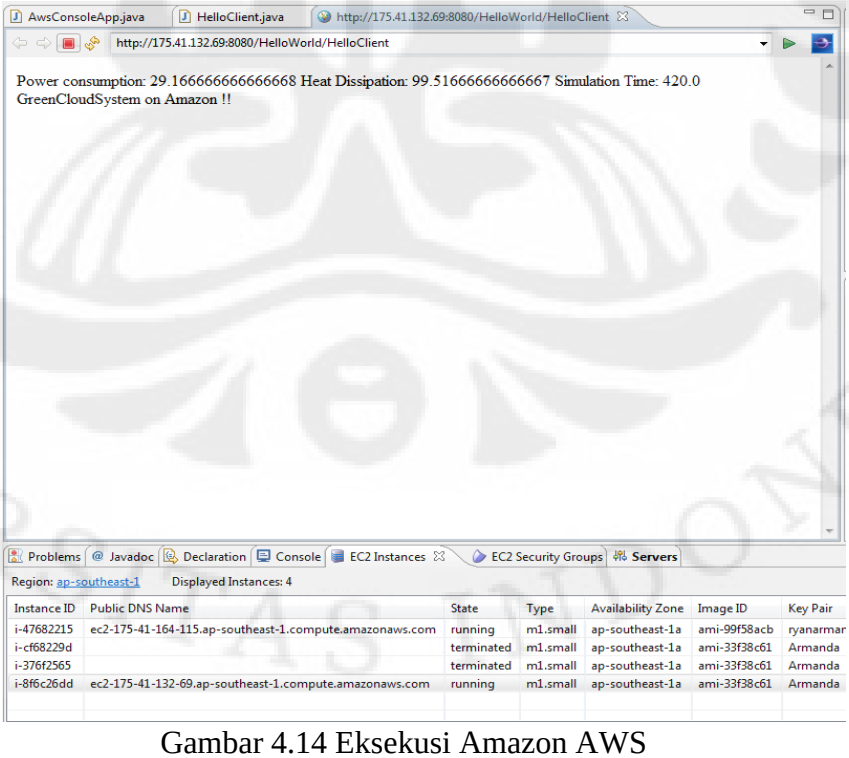
Pada bagian ini akan diujicoba eksekusi program Green*CloudSim*ulation pada arsitektur *Cloud* yang sebenarnya, yakni *Amazon Elastic Compute Cloud* (Amazon EC2) yang disediakan penyedia layanan *cloud* berbayar *Amazon Web Services* (Selanjutnya akan disebut Amazon AWS). Eksekusi program

*GreenCloudSimulation* via Amazon AWS ini dapat dilakukan karena adanya ekstensi Amazon AWS pada program *compiler* Java *Eclipse*. Program ini akan dijalankan sepenuh nya menggunakan arsitektur *cloud* dan disimpan pada *storage* Amazon AWS yang berada di Amerika Serikat. Kemudian akan dianalisa *output* nya apakah program ini dapat berjalan seperti pada media *desktop* atau akan menemui *performance defect*.



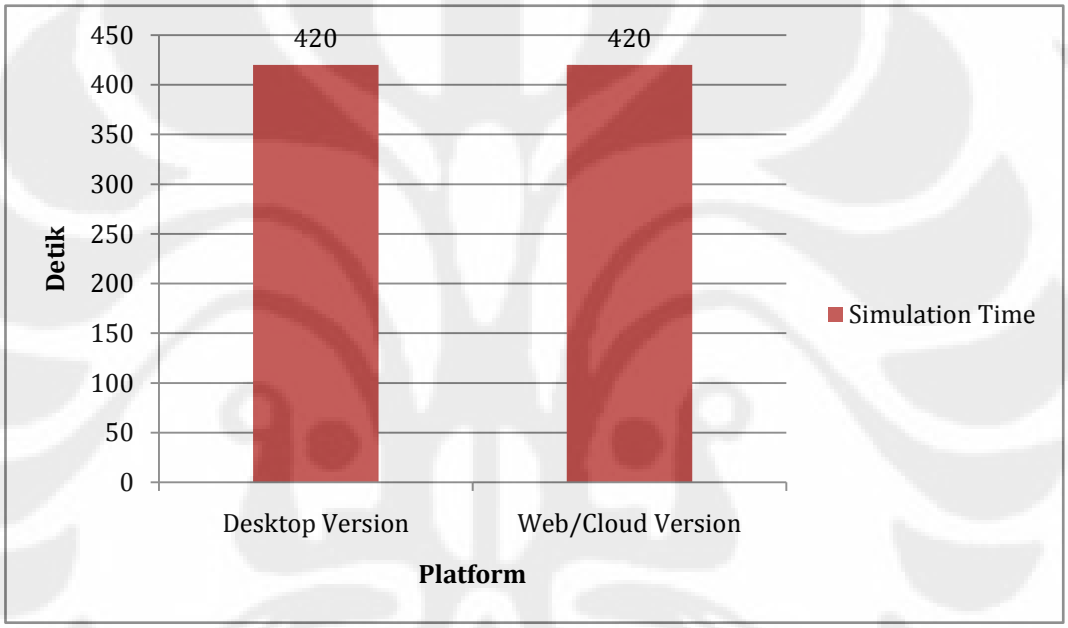

Gambar 4.13 Server Amazon AWS

Proses dimulai dengan menjalankan Amazon AWS pada *eclipse*, seperti terlihat pada gambar diatas, proses tersebut yaitu menjalankan *server* yang ada pada Amazon AWS, dengan cara membuat *server* dan memberikan elastic IP, atau IP yang didapatkan dari Amazon AWS pada *cloud*.




Gambar 4.14 Eksekusi Amazon AWS

Eksekusi program dalam arsitektur *Cloud* dapat dijalankan dengan baik, walau menemui cukup banyak kendala, diantara lain sangat lemahnya koneksi terhadap *Server* Amazon AWS, dan sangat mudah untuk terjadi putus koneksi terhadap server. Hal ini terjadi dikarenakan letak geografis server yang jauh di Amerika Serikat, dan koneksi Internet Indonesia yang tidak stabil. Selain itu karena besarnya *load* yang harus dijalankan. Kinerja *Cloud* ini dapat dimaklumi, dikarenakan sifat teknologi yang baru dan rumit, teknologi ini baru mencapai tahap pengembangan, dan dibutuhkan lebih banyak riset dan implementasi agar sempurna.




Gambar 4.15 Perbandingan waktu simulasi *desktop* vs *cloud*

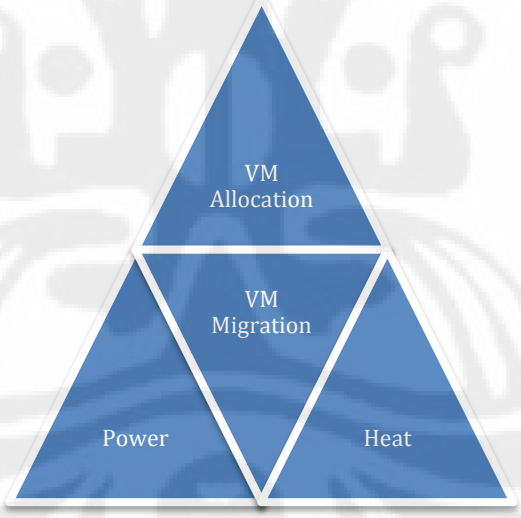
## 4.2 Analisa Keseluruhan Program dan Pengembangan Selanjutnya Terhadap *Green Cloud*

Pada bagian ini akan dianalisa performa *CloudSim* secara keseluruhan terhadap konsep TIK Hijau. Performa yang akan dianalisa merupakan *CloudSim* yang telah ditambahkan elemen simulasi, dan modul disipasi panas dan *power consumption.* Modul *CloudSim* ini dapat dieksekusi dengan menggunakan perangkat *compiler* Java. Penambahan modul ini ditujukan untuk sebagai langkah awal dalam pengembangan *Cloud Computing* yang berbasis Teknologi informasi dan Komunikasi Hijau

Pengembangan selanjutnya yang dapat dilakukan dari program ini memiliki potensi yang cukup besar, yakni melakukan migrasi VM dari yang memiliki load lebih banyak ke lebih rendah, agar efisiensi konsumsi tenaga dapat dilakukan, dan lainnya. Adapun hal lain yang dapat dilakukan adalah membentuk jaringan cloud antar bangsa yang berbasis TIK Hijau, hal ini cukup mungkin dilakukan mengingat teknologi *cloud* merupakan teknologi berbasis Internet.

Selain itu, disipasi panas yang sering menjadi kekhawatiran para *administrator datacenter*, dapat diantisipasi dengan adanya simulasi ini, adanya hasil keluaran disipasi panas yang harus ditangani oleh para *administrator* turut memudahkan dan memberikan efisiensi terhadap persoalan panas dan pengalihan energi.

Tidak hanya fitur tersebut, fitur lain yang terdapat pada *CloudSim* juga berfungsi secara penuh pada simulasi *GreenCloudSimulation*, informasi biaya yang harus dibayar konsumen, dan simulasi eksekusi program masih menjadi fitur utama yang ada di program ini.




Gambar 4.16 Arah perkembangan Simulasi *Cloud*

Diharapkan dengan adanya program simulasi akan lebih membuka arah dan pola pikir pengambang menuju TIK Hijau, dan memanfaatkan teknologi *cloud computing* secara keseluruhan untuk mencapai tujuan tersebut.

# BAB V

# KESIMPULAN

Setelah dilakukan perancangan dan analisa terhadap *CloudSim* dan *GreenCloudSimulation*, serta implementasi langsung terhadap arsitektur *Cloud* Amazon AWS, maka dapat diambil beberapa kesimpulan, yaitu:

1. Kinerja *GreenCloudSimulation* berbeda dalam batas yang wajar dengan *CloudSim*, hal ini dikarenakan adanya penambahan fitur dan fungsi baru, yakni penghitungan daya dan disipasi panas terhadap *datacenter* ketika simulasi dilakukan. Perbedaan ini dikarenakan bertambahnya beban kerja program dikarenakan proses penghitungan intsruksi per detik yang dilakukan untuk mendapatkan fungsi konsumsi daya.
2. Konsumsi daya yang dibutuhkan dan disipasi panas yang dikeluarkan tergolong sangat kecil jika dibandingkan dengan penggunaan pada komputer desktop. Pada arsitektur *Cloud* program yang membutuhkan *resource* tinggi hanya membutuhkan 23,61 Watt,dan mengeluarkan 80,56 BTU disipasi panas. Sedangkan pada desktop membutuhkan 562 Watt daya, dan mengeluarkan 1917,544 BTU panas.
3. Disipasi panas adalah jumlah panas yang dikeluarkan oleh perangkat dan harus diserap oleh lingkungan. Disipasi panas dihitung dalam satuan *British thermal Unit* (BTU) dengan rumus 1 BTU = Watt x 3.412.
4. Perbandingan antara kedua program yang berbeda karakteristik *resource* tersebut menunjukkan hasil bahwa program yang membutuhkan *resource* lebih besar, akan menghabiskan daya yang lebih besar. Namun perbedaan yang terjadi dalam batas wajar, karena tidak bersifat eksponensial, namun bertahap sesuai dengan perangkat keras yang digunakan.
5. Eksekusi program dalam arsitektur *cloud* yang sebenarnya, yakni *Amazon Web Services* (AWS) berjalan dengan lancar, dan menunjukkan hasil yang sama dengan versi *desktop*. Namun ada beberapa kendala yakni *server* amazon yang berada di Amerika membuat proses koneksi menjadi sangat lambat.

# DAFTAR ACUAN


[1] Ratnadeep Bhattacharjee, *An Analysis of the Cloud Computing Platform* . Thesis. Massachusetts Institute of Technology (MIT), January 2009, http://dspace.mit.edu/bitstream/handle/1721.1/47864, diakses terakhir tanggal 29 Desember 2009

[2] Leonard Francis, *Cloud Computing: Implications for Enterprise Software Vendors (ESV)*. Thesis. Massachusetts Institute of Technology (MIT), January 2009,http://dspace.mit.edu/handle/1721.1/47862, diakses terakhir tanggal 29 Desember 2009

[3] Willis, John M. Cloud Computing. *IT Management and Cloud Blog.* December2008. http://wwwjohnmwillis.com., diakses terakhir tanggal 3 Januari 2010

[4] Perry, Geva. *How Cloud and Utility Computing Are Different*. GigaOm, February 2008, http://gigaom.com/2008/02/28/how-cloud-utility-computing-are-different, diakses terakhir tanggal 15 Desember 2009

[5] Jay Heiser, Mark Nicolett. *Assessing the Security Risks of Cloud Computing.*Stamford : Gartner, 2008

[6] *Gartner Says Cloud Computing Will Be As Influential As E-business.* Stamford:Gartner, 2008.

[7] Christensen, Clayton M. and Raynor, Michael E. *The Innovator's Solution.*Cambridge : Harvard Business School Press, 2003.

[8] Information Week vol.10 2008 http://www.informationweek.com/cloud-computing/blog/archives /2008/1 0/willmicrosoft_2.html, diakses terakhir tanggal 27 Desember 2009

[9] Energy Measurements and Conversions Structure http://www.extension.iastate.edu/agdm/wholefarm/html/c6-86.html, diakses terakhir tanggal 11 Juni 2010

[10] Peripherals Specification Reviews http://www.tomshardware.com/charts ,diakses terakhir tanggal 11 Juni 2010

[11] Fine, Charles H. *Clock Speed -Winning Industry Control in the Age of Temporary Advantage.* Reading: Perseus Books, 1998.

[12] Yefim V. Natis, Nicholas Gall, David W. Cearley, Lydia Leong,Robert P. Desisto,Benoit J.Lheureux, David Mitchell Smith. *Cloud,SaaS, Hosting and Other Off-Premises Computing Models.*Gartner, 2008.

[13] Daryl C. Plummer, Thomas J. Bittman, Tom Austin, David W. Cearley, David Mitchell Smith. *Cloud Computing: Defining and Describing an Emerging Phenomenon.*Stamford : Gartner, 2008.

[14] DeCandi A, G. , Has Torun, D. , Jampani, M. , Kakul Apat I, G Lakshman, *Dynamo: Amazon's highly available key-value store*.ACM Press New York, NY, USA, 2009

[15] Michael Miller. *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online.* Que Books,2008