Pembuatan Aplikasi Penjadwalan dan Reservasi untuk Penggunaan *Private Cloud Computing*

Albert Halim[1], Henry Novianus Palit;[2], Agustinus Noertjahyana[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: lim_bert_lie@yahoo.co.id[1], hnpalit@petra.ac.id [2], agust@petra.ac.id [3]

ABSTRAK

Cloud computing adalah teknologi baru yang sedang hangat dibahas oleh pakar teknologi informasi. Teknologi *cloud computing* hadir untuk menjawab tantangan akan kebutuhan teknologi komputasi yang semakin efisien. Sehingga terciptalah ide untuk membangun sebuah sistem *cloud* di laboratorium komputer Universitas Kristen Petra. Namun implementsi *cloud* yang ada dapat mengganggu pemakaian lab yang ada. Program ini bertujuan untuk membantu *user* memanajemen sistem *cloud* yang ada agar berjalan sesuai dengan jadwal yang diinginkan.

Dalam pembuatan program dilakukan pengujian terhadap sistem *scheduling* yang telah tersedia, dan melakukan pengembangan terhadap sistem *filter scheduler* pada *scheduler* yang telah ada, serta membuat *user interface* berupa *web* dengan bentuk tampilan kalender, dengan fitur-fitur yang dibutuhkan.

Hasil pada program ini, berupa sebuah sistem *scheduler* yang dapat membatasi penggunaan *instance* pada OpenStack agar berjalan sesuai dengan jadwal yang dibuat oleh sistem administrator. Sistem *scheduler* yang telah dibuat mempunyai *latency* (*overhead*) yang relative kecil sehingga tidak begitu memberatkan sistem *scheduler* yang ada pada OpenStack.

Kata Kunci: *Cloud Computing, Openstack, Scheduling, Filter Scheduler*.

ABSTRACT

Cloud computing is a new technology that is being hotly discussed by experts in information technology. Cloud computing technology is here to address the challenge of more efficient computing technologies. So an idea has come about to build a cloud system in the Petra Christian University's computer laboratory. However implementing the cloud could interfere with the use of existing lab. The application aims to help the user manage the cloud system to run according to the desired schedule.

The application development involved evaluation on the scheduling system, enhancement on the filter scheduler to accommodate a time schedule, and creation of the web-based user interface to manage the time schedule (in the form of a calendar).

The application can restrict the OpenStack's instance deployment only on the time schedule allowed by the system administrator. In addition, the system scheduler has negligible latency (overhead), so that it will not affect the system scheduler's performance.

***Keywords**: Cloud Computing, Openstack, Scheduling, Filter Scheduler*.

1. PENDAHULUAN

Pada saat ini, komputer memiliki teknologi yang canggih dan murah, serta telah menjamur dimana-mana. Komputer - komputer juga telah terhubung satu sama lain melalui *network* (yang sering disebut paralel *computing*) dan saling berkomunikasi satu sama lain melalui *passing message* dan berbagi proses komputasi yang ada dengan komputer lain, yang sering disebut *distributed computing*.

Dengan adanya teknologi dan sistem yang berkembang tersebut, maka timbullah kebutuhan untuk lebih memaksimalkan pemakaian sistem komputer terdistribusi dengan melakukan pembagian sumber daya komputasi ke banyak *user*. Komponen yang sering digunakan untuk kebutuhan ini diantaranya: *resource allocation*, *job scheduler* (termasuk di dalamnya, *advanced reservation*), dan *resource monitoring*.

Lab - lab di Jurusan Teknik Informatika U.K. Petra ditunjang dengan komputer-komputer dengan spesifikasi yang memadai serta ditunjang dengan Internet dengan *speed* yang cukup cepat namun belum dimanfaatkan dengan maksimal. Selama ini lab yang ada digunakan oleh mahasiswa, tidak digunakan secara penuh (utilitas masih rendah) dimana penggunaannya hanya terbatas pada saat ada praktikum. Dalam kondisi ini biasanya komputer dibiarkan tetap menyala dan tanpa digunakan (dalam kondisi *idle*). Kondisi demikian, memberikan ide untuk memanfaatkan komputer – komputer di lab tersebut sebagai *platform private cloud computing*. Agar penerapan *platform private cloud computing* yang ada tidak mengganggu pemakaian lab untuk keperluan belajar mengajar mahasiswa, maka diperlukan sistem penjadwalan dan reservasi yang akan mengatur berjalannya kedua aktifitas di lab Jurusan Teknik Informatika U.K. Petra

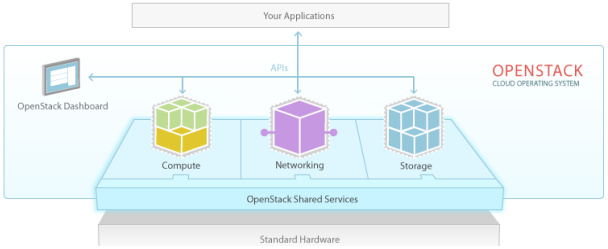
2. DASAR TEORI

2.1. Konsep *Scheduling* dan *Reservation*

Penjadwalan adalah alokasi sumber daya bersama dari waktu ke waktu untuk *computing activities*. *Scheduling* telah menjadi subyek dari sejumlah besar *literatur* di bidang *operations research*. Dalam masalah-masalah *machine scheduling*, pekerjaan (*job*) mewakili aktivitas dan mesin mewakili sumber daya (*resource*); setiap mesin dapat memproses paling banyak satu pekerjaan (*job*) pada satu waktu [8].

2.2. Konsep OpenStack

OpenStack adalah perangkat lunak *open source* untuk membangun *cloud computing* (*private* dan *public*). [5]



Gambar 1 OpenStack *Components* pada Konsep OpenStack

Sumber:http://www.openstack.org/themes/openstack/images/openstack-software-diagram.png

Manfaat *OpenStack*:

- *Control and Flexibility*. *Open source platform* berarti kita tidak pernah terikat ke *proprietary vendor*, dan desain modular yang dapat diintegrasikan dengan teknologi – teknologi lama (*legacy*) atau pihak ketiga untuk memenuhi kebutuhan bisnis kita.
- Skalabilitas (*Scalability*). Dengan *public cloud* yang berskala besar dan kemampuan memyimpanan di skala *petabyte*, OpenStack sudah digunakan di perusahaan – perusahaan global yang menyediakan infrastruktur *cloud computing* yang aman.
- *Open Industry Standard*. Lebih dari 75 perusahaan terkemuka turut berpartisipasi dalam OpenStack, termasuk Cisco, Citrix, Dell, Intel dan Microsoft.
- *Openness and Compatibility*. Menghindari *vendor lock-in* dengan penggunaan lisensi Apache untuk *source code* yang memberikan kompatibilitas dengan ribuan *cloud computing* yang ada untuk transisi yang mulus dari *cloud* ke *cloud*.
- *Flexible Technology*. Ekosistem global dari *vendor* - *vendor* terkemuka memberikan dukungan terintegrasi untuk berbagai fitur dalam *cloud*. Sebagai contoh, dukungan hypervisor yang mencakup ESX, Hyper-V, KVM, LXC, QEMU, UML, Xen, dan XenServer.

OpenStack terdiri dari banyak bagian yang memiliki fungsi yang berbeda. Karena sifatnya yang terbuka, siapapun dapat menambahkan komponen tambahan untuk OpenStack untuk membantu untuk memenuhi kebutuhan yang ada [2]

Sembilan komponen kunci yang merupakan bagian dari "inti" dari OpenStack adalah sebagai berikut: [3]

- ***Nova*** adalah mesin komputasi utama di balik OpenStack. Ini adalah sebuah *"fabric controller"* yang digunakan untuk menyebarkan dan mengelola sejumlah besar mesin *virtual* dan *instances* untuk menangani tugas-tugas komputasi.
- ***Swift*** adalah sistem penyimpanan untuk objek dan file.
- ***Cinder*** adalah komponen penyimpanan blok, yang lebih bersifat analog dengan gagasan komputer tradisional yang dapat mengakses lokasi tertentu pada *disk drive*.
- ***Neutron*** menyediakan kemampuan jaringan untuk OpenStack. Ini membantu untuk memastikan bahwa setiap komponen dari penyebaran OpenStack dapat berkomunikasi satu sama lain dengan cepat dan efisien.
- ***Horizon*** adalah *dashboard* OpenStack. Ini adalah satu-satunya *interface* grafis untuk OpenStack, sehingga bagi pengguna yang ingin mencoba OpenStack, ini mungkin merupakan komponen pertama mereka yang benar-benar dapat "dilihat." Pengembang dapat mengakses semua komponen OpenStack individual melalui *Application Programming Interface* (API), tapi *dashboard* menyediakan sistem administrator untuk melihat apa yang sedang terjadi di *cloud*, dan untuk mengelolanya sesuai kebutuhan.
- ***Keystone*** menyediakan identitas layanan untuk OpenStack. Ini pada dasarnya adalah daftar sentral dari semua pengguna dari OpenStack *cloud*, yang dipetakan terhadap semua layanan yang diberikan oleh *cloud* yang memiliki izin untuk menggunakan layanan tersebut.
- ***Glance*** menyediakan layanan *image* untuk *OpenStack*. Dalam kasus ini, "gambar" mengacu pada gambar (atau salinan *virtual*) dari *hard disk*. *Glance* memungkinkan gambar-gambar (*images*) ini akan digunakan sebagai *template* ketika *deploying virtual machine instances* baru.
- ***Ceilometer*** menyediakan layanan telemetri, yang memungkinkan *cloud* untuk menyediakan *billing service* kepada pengguna individu dari *cloud*.
- ***Heat*** adalah komponen orkestrasi dari OpenStack, yang memungkinkan pengembang untuk menyimpan persyaratan (*requirements*) aplikasi *cloud* dalam sebuah *file* yang mendefinisikan sumber daya apa saja yang diperlukan untuk aplikasi tersebut. Dengan cara ini, hal ini membantu untuk mengelola infrastruktur yang diperlukan untuk menjalankan layanan *cloud*.

2.3. Nova Scheduler

OpenStack menggunakan *nova scheduler* untuk menentukan bagaimana memenuhi permintaan - permintaan komputasi (*compute*) dan ruang penyimpanan (*storage*). Sebagai contoh, layanan nova-scheduler menentukan di *host* mana suatu VM akan diluncurkan. *Scheduler* dapat dikonfigrasikan dengan mengeset beberapa opsi. Opsi – opsi ini biasanya disimpan dalam file /etc/nova/nova.conf [1].

Filter scheduler adalah *scheduler default* untuk penjadwalan sebuah *virtual machine instances*. *Filter scheduler* mendukung *filtering* dan *weighting* untuk membuat keputusan mengenai di mana *instance* baru harus dibuat. Ketika *filter scheduler* menerima permintaan untuk sumber daya yang ada, pertama kali *filter scheduler* menerapkan *filter* untuk menentukan *host* yang memenuhi syarat untuk dipertimbangkan saat pengiriman sumber daya. *Filter* adalah biner: baik sebuah *host* diterima oleh *filter*, atau ditolak. *Host* yang diterima oleh *filter* kemudian diproses oleh algoritma yang berbeda untuk menentukan *host* yang akan digunakan untuk permintaan itu [4].

Secara *default*, *scheduler_driver* dikonfigurasikan sebagai *filter scheduler*. Dalam konfigurasi *default*, *scheduler* ini mempertimbangkan *hosts* yang memenuhi semua kriteria berikut:[6]

- Belum pernah dijadwalkan sebelumnya (RetryFilter).
- Berada dalam *availability zone* yang diminta (AvailabilityZoneFilter).
- Memiliki RAM yang cukup (RamFilter).
- Dapat melayani permintaan (*request*) (ComputeFilter).
- Memenuhi spesifikasi tambahan yang terkait dengan tipe instansi (ComputeCapabilitiesFilter).
- Memenuhi properti - properti arsitektur, jenis *hypervisor*, atau *mode virtual machine* yang dispesifikasikan dalam *image properties* dari instansi (ImagePropertiesFilter).

- Berada di *host* yang berbeda dari grup instansi – instansi lain (jika diminta) (ServerGroupAntiAffinityFilter).
- Berada dalam *host*s yang berada di dalam satu grup (jika diminta) (ServerGroupAffinityFilter).

Untuk membuat *filter* baru diperlukan membuat *class* baru yang merupakan turunan dari *BaseHostFilter* dan menerapkan sebuah metode: *host_passes*. Metode ini seharusnya mengembalikan nilai *True* jika *host* memenuhi syarat *filter*. Metode ini membutuhkan parameter yaitu *host_state* dan *filter_properties*. Sebagai contoh pada *class RamFilter* (catatan: *class* standar yang sudah tersedia di nova) sebagai berikut:

```
class RamFilter(filters.BaseHostFilter):
  """Ram Filter with over subscription flag"""
  def host_passes(self, host_state, \
    filter_properties):
    """Only return hosts with sufficient
    available RAM."""
    instance_type = \
      filter_properties.get('instance_type')
    requested_ram = instance_type['memory_mb']
    free_ram_mb = host_state.free_ram_mb
    total_usable_ram_mb = \
      host_state.total_usable_ram_mb
    used_ram_mb = total_usable_ram_mb - free_ram_mb
    return total_usable_ram_mb * \
      FLAGS.ram_allocation_ratio - \
      used_ram_mb >= requested_ram
```

Sebagai contoh ketika mengimplementasikan *custom filter* dengan Python bernama *myfilter*. *MyFilter* bersamaan dengan *filter* yang sudah ada pada nova-*scheduler*, maka dapat ditambahkan pada nova.conf tanpa menghapus *filter* yang sudah ada. Sehingga *nova.conf* mengisikan 2 buah filter seperti berikut:

- scheduler_driver=nova.scheduler.FilterScheduler
- scheduler_available_filters=nova.scheduler.filters.standard_filters
- scheduler_available_filters=myfilter.MyFilter
- scheduler_default_filters=RamFilter,ComputeFilter,MyFilter

Dengan pengaturan ini, *nova scheduler* akan menggunakan *Filter Scheduler* untuk *driver scheduler*. Standar *Nova filter* dan *MyFilter* tersedia untuk *Filter Scheduler*. *RamFilter*, *ComputeFilter*, dan *MyFilter* digunakan secara *default* ketika tidak ada *filter* yang ditentukan dalam permintaan. Contoh modifikasi *filter* menggunakan nama "test_filter.py":

```
from nova.scheduler import filters
from nova.openstack.common import log as logging

LOG = logging.getLogger(__name__)

class TestFilter(filters.BaseHostFilter):
    """NOOP host filter. Returns all hosts."""

    def host_passes(self, host_state, \
       filter_properties):
       LOG.debug("COMING FROM: \
         nova/scheduler/filters/test_filter.py")
       return True
```

3. RANCANGAN SISTEM

3.1. Desain Jaringan

Universitas Kristen Petra memiliki *network* yang tersebar melalui kabel LAN (*Local Area Network*), yang memiliki DHCP (*Dynamic Host Configuration Protocol*) dan memiliki pengaturan HTTP *proxy* yaitu http://proxy.petra.ac.id:8080/ yang tersebar ke seluruh jaringan yang ada.

Lab Jurusan Teknik Informatika U.K. Petra terhubung ke jaringan Universitas Kristen Petra melalui kabel LAN yang memiliki IP *static* pada setiap komputernya. Dimana penempatan nomor IP tersebut disesuaikan dengan nomor yang ada pada tiap komputer yang ada.

3.2. Spesifikasi Komputer

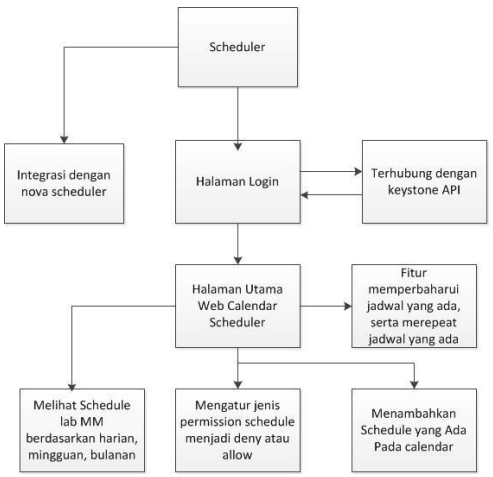
Infrastruktur komputer merupakan kebutuhan dasar berupa komputer, yang dibutuhkan untuk melakukan instalasi Openstack. Sebelum melakukan instalasi Openstack, terlebih dahulu komputer yang ada dilakukan instalasi sistem operasi Ubuntu Server 14.04 LTS. Setelah komputer yang ada menggunakan sistem operasi ubuntu, barulah instalasi openstack dilakukan. Komputer-komputer yang ada di laboratorium-MM memiliki spesifikasi sebagai berikut:

Processor	: Intel (R) Core (TM) i5-3340 CPU @ 3.10GHz 3.10GHz
RAM	: 16.0 GB
Ethernet Card	: 1 Gbps
Storage	: 250 GB

3.3. Rancangan Instalasi Openstack

Openstack adalah *Cloud Operating System* (Cloud OS) yang berfungsi untuk mengatur sekumpulan *resource* fungsi *compute*, *storage*, dan *network*. Semua ini dapat diatur dengan *dashboard* yang dapat diakses melalui web *interface*. OpenStack merupakan sekumpulan *software* yang terbagi menjadi modul-modul sesuai dengan fungsinya masing-masing (*compute*, *storage*, *network*). Openstack yang digunakan dalam pengerjaan skripsi ini adalah Openstack versi Juno. Dalam percobaan awal, dilakukan instalasi Openstack di laboratorium MM dengan sejumlah lima komputer, dimana satu komputer sebagai *controller node* dan dan empat komputer sebagai *compute node*.

3.4. Garis Besar Web Scheduler



Gambar 2 Diagram Blok Garis Besar *Web Scheduler*

Diagram blok pada Gambar 2 merupakan penjelasan rancangan dari cara penggunaan *scheduler* yang dibuat. Penggunaan *scheduler* dimulai setelah pengaturan dan penambahan *file* di nova scheduler filter, nova.conf dan *interface web scheduler* selesai ditambahkan, lalu setelah itu pengguna (admin) dapat masuk ke halaman utama scheduler setelah melakukan *login* dengan menggunakan *username* dan *password* yang telah dibuat di OpenStack. Setelah *login*, pengguna dapat langsung memasuki halaman utama *scheduler* dan melihat *list* dari jadwal yang telah ada, pengguna juga dapat melihat jadwal berdasarkan harian, mingguan ataupun

bulanan. Pengguna (*admin*) juga dapat melakukan penambahan jadwal baru pada *scheduler*. Selain menambahkan jadwal yang ada, pengguna juga dapat mengedit jadwal dan me-*repeat* jadwal yang telah dibuat. Untuk membantu pengguna, di dalam aplikasi juga terdapat fitur untuk mencetak jadwal yang telah dibuat, serta melihat informasi *resource* yang tersedia.

3.5. Desain Database

Desain *database* dari *scheduler* yang dibuat mengikuti desain *database* yang telah disediakan oleh Openstack. Aplikasi *scheduler* yang dibuat menggunakan *database* MariaDB sebagai pengelola data yang dipakai oleh *scheduler*.

4. IMPLEMENTASI SISTEM

4.1. Pembuatan Sistem di OpenStack

Pembuatan sistem di OpenStack mencakup pengaturan sistem yang ada pada OpenStack dan membuat file *filter* yang dibutuhkan dalam proses *scheduling* pada nova scheduler.

4.2. Pengaturan Database

Langkah pertama yang paling penting sebelum kita mengimplementasikan sistem dari aplikasi ini, adalah membuat *database* aplikasi dan pengaturan koneksi ke *database*. *Database* yang digunakan yaitu MariaDB. Untuk mempermudah manajemen *database*, menggunakan *database* manajemen phpmyadmin di ubuntu, yang di *install* dengan cara apt-get install phpmyadmin. Setelah itu dibuatlah *database calendar* yang memiliki tabel jqcalendar, *repeat*, dan *permission*.

4.3. Implementasi *user interface* aplikasi yang dibuat

Pembuatan aplikasi skripsi ini menggunakan Python (*Programming language*) dan PHP (*Hypertext Preprocessor*). Bahasa pemrograman python digunakan untuk mengatur sistem *filtering* dan *weighting* yang ada di nova-scheduler. Sedangkan PHP digunakan untuk *user interface* (*web*) dari *scheduler* yang akan dibuat. Implementasi sistem dimulai dengan instalasi *service* nova *scheduler* di OpenStack.

4.4. Implementasi untuk koneksi *database*

Bagian ini menjelaskan *file-file* PHP yang berguna untuk melakukan pengambilan data dari *database Calendar scheduler*. Pengambilan data terjadi ketika aplikasi memanggil *file* PHP mengambil data *schedule* dari *database*.

4.5. Implementasi fungsi-fungsi untuk sistem pada *Calendar Scheduler*

Dalam pembuatan *web Calendar Scheduler* dilakukan pembuatan *file-file* PHP yang berisi fungsi-fungsi dari sistem *Calendar Schedular*.

4.6. Implementasi Javascript

Javascript memiliki peran yang cukup penting bagi jalannya *web Calendar Scheduler*. Javascript diperlukan sebagai bagian yang menjalankan fungsi-fungsi yang ada dalam *user interface* aplikasi.

4.7. *Plugin* WdCalendar

Plugin WdCalendar, merupakan *plugin* yang digunakan untuk membuat tampilan *calendar* dari *Web Calendar Scheduler* yang telah dibuat.

4.8. *Plugin* Data Tables

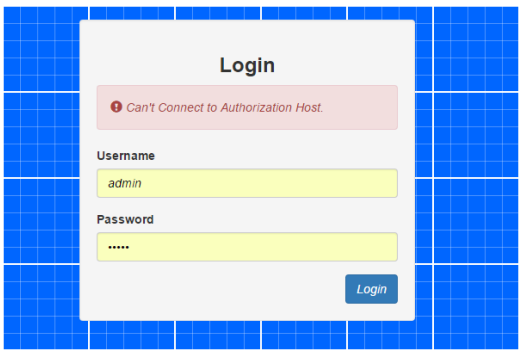
Merupakan *plugin* yang digunakan untuk membuat tampilan berupa tabel untuk data-data yang ada di laporan *instance queue* dan *instance report*.

5. PENGUJIAN SISTEM

Dalam pengujian ini dilakukan pengujian dalam beberapa tahapan yaitu menguji *user interface* web *Calendar Scheduler* yang dibuat, dan menguji sistem yang ada pada OpenStack pada saat sebelum dilakukan pemasangan sistem *scheduling* yang dibuat. Pengujian juga dilakukan setelah pemasangan sistem *scheduling* yang dibuat.

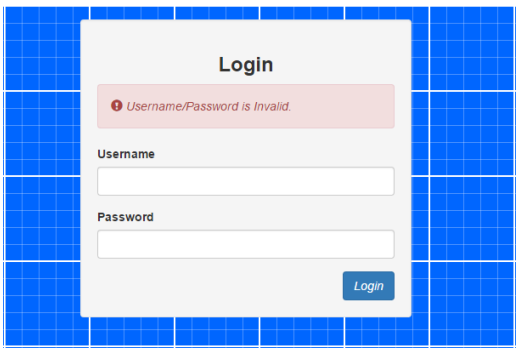
5.1. Pengujian Sistem *Login*

Pengujian proses *login*, dimana situasi yang dibuat adalah *form login* tidak dihubungkan dengan *database* keystone, dapat dilihat pada Gambar 3.



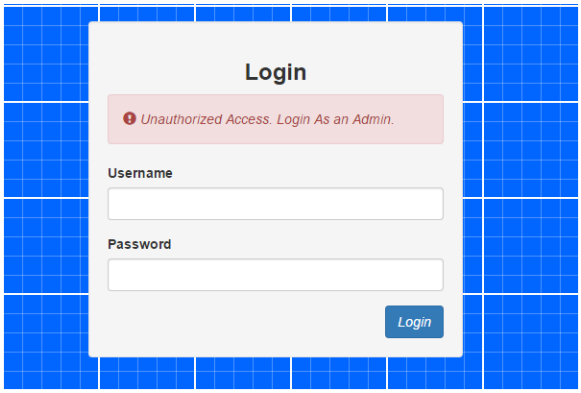
Gambar 3 Tanggapan dari halaman *login*, jika halaman *login* tidak terhubung dengan *database* keystone

Pengujian proses *login* dimana *username* atau *password* yang dimasukkan salah. Setelah *username* dan *password* dimasukkan maka akan terjadi respon seperti pada Gambar 4.



Gambar 4 Tanggapan dari halaman *login* jika *username* atau *password* salah

Pengujian proses *login* dimana situasi yang dibuat adalah *user* yang melakukan *login* bukanlah *user* dengan hak akses admin, hasil tanggapan dari sistem akan seperti Gambar 5.

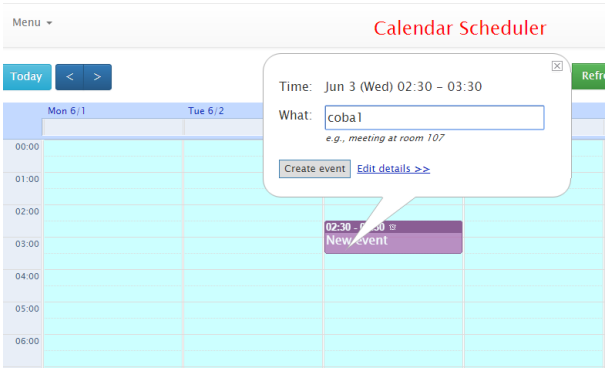


Gambar 5 Tanggapan dari halaman login jika *user* yang melakukan *login* tidak memiliki *role* sebagai *admin*

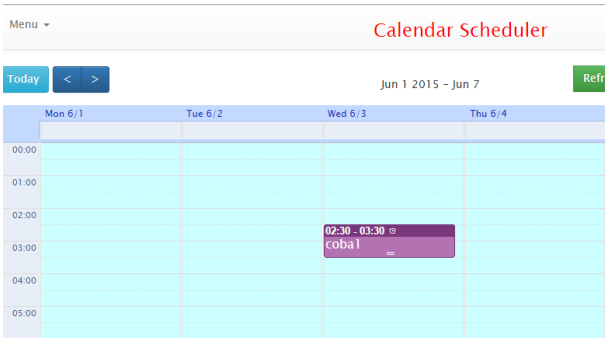
Jika *user* yang melakukan *login* menggunakan *username* dan *password* yang benar serta memiliki hak akses sebagai *admin* maka *user* akan diarahkan langsung ke halaman main.php

5.2. Pengujian Prosess *add* pada *Scheduler*

Pengujian proses *add* dimana uji coba dilakukan dengan langsung mengisi *form input* berupa *subject* yang diinginkan, dimana *subject* yang dicoba bernama coba1 yang dapat dilihat pada Gambar 6 dan hasil *add* dapat dilihat pada Gambar 7.



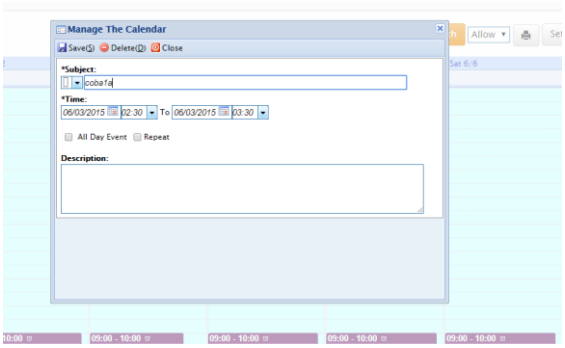
Gambar 6 Mengisi *Form add*



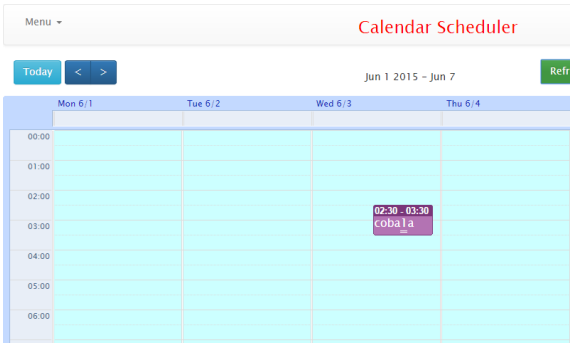
Gambar 7 Hasil *add* dengan nama *subject* coba1

5.3. Pengujian Prosess *update* pada *Scheduler*

Pada pengujian ini dilakukan uji coba untuk mengubah nama *subject* coba1 dari hasil *add* pada Gambar 7 menjadi *subject* dengan nama coba1a. Dimana terlebih dahulu kita mengisi *form* edit seperti pada Gambar 8 dan hasil *edit* atau *update* yang dilakukan dapat dilihat pada Gambar 9.



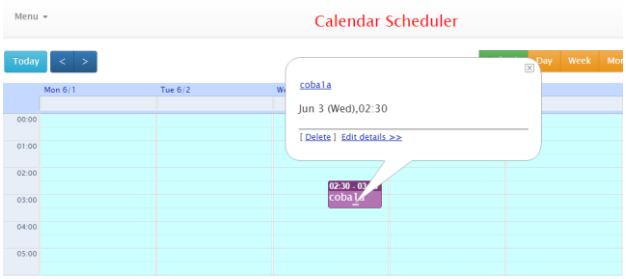
Gambar 8 Mengisi *form edit* untuk mengubah nama jadwal



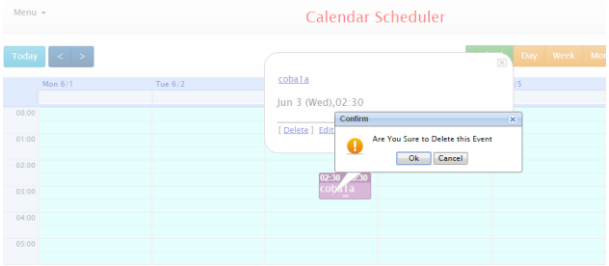
Gambar 9 Hasil Mengubah nama jadwal

5.4. Pengujian Prosess *delete* Pada *Scheduler*

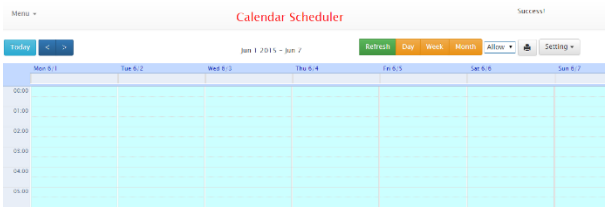
Pada pengujian ini *user* melakukan klik pada jadwal yang telah dibuat dan akan muncul *form* awal yang menampilkan pilihan *edit* dan *delete*. Lalu memilih pilihan *delete* yang ada seperti pada Gambar 10, dan akan muncul peringatan seperti Gambar 11 dan jika memilih *Ok* untuk melakukan *delete*, jadwal coba1 dihapus dan hasilnya dapat dilihat pada Gambar 12.



Gambar 10 Memilih menu *delete* setelah klik jadwal yang ingin dihapus



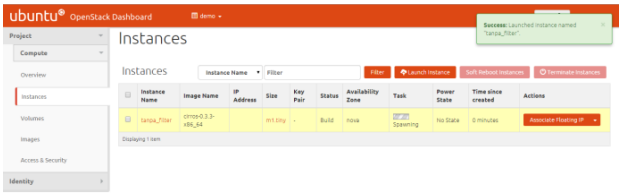
Gambar 11 Peringatan yang muncul setelah memilih *delete*



Gambar 12 Hasil *delete* setelah memilih menghapus jadwal coba1a

5.5. Pengujian *Launch Instance* pada OpenStack

Pada pengujian yang pertama dilakukan *launch instance* dengan tanpa menggunakan sistem *filter* yang telah dibuat, yaitu calendar_filter.py yang telah dibuat. Pada proses ini dilakukan *launch instance* di horizon *dashboard* yang dapat dilihat pada Gambar 13. Hasilnya *instance* dapat dijalankan dengan baik yang menunjukkan sistem *launch* OpenStack berjalan dengan baik.. Dari hasil percobaan ini diketahui bahwa sistem *scheduler* yang ada membutuhkan waktu 32.4 *milliseconds* untuk melakukan *launch instance*.

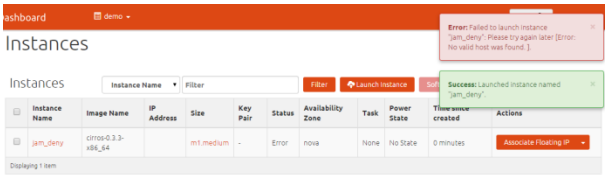


Gambar 13 Hasil *launch instance* tanpa *filtering*

Hasil *log* dari pengujian pada Gambar 13 adalah sebagai berikut:

```
2015-06-05 19:07:17.752 987 DEBUG nova.filters [req-
1265c81e-aa1c-4d27-b781-70e78c3d39e6 None] Starting
with 7 host(s) get_filtered_objects
/usr/lib/python2.7/dist-packages/nova/filters.py:70
...
2015-06-05 19:07:17.783 987 DEBUG
nova.scheduler.filter_scheduler [req-1265c81e-aa1c-
4d27-b781-70e78c3d39e6 None] Weighed [WeighedHost
[host: (mm-17, mm-17) ram:15479 disk:221184 io_ops:0
instances:0, weight: 1.0], WeighedHost [host: (mm-20,
mm-20)
```

Pada pengujian yang kedua dilakukan *launch instance* dengan menggunakan sistem *filter* yang telah dibuat, yaitu calendar_filter.py yang telah dibuat. Pengujian ini dilakukan dalam mode *deny*, *dan instance* dijalankan pada saat jam *deny* sehingga proses *launch instance* tidak diijinkan. Proses *launch instance* di horizon *dashboard* yang dilakukan dapat dilihat pada Gambar 14. Dari hasil percobaan ini diketahui bahwa sistem *scheduler* yang ada membutuhkan waktu 42.3 *milliseconds* untuk melakukan *launch instance*.

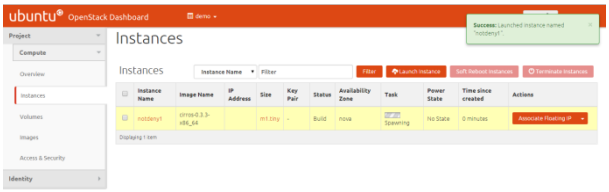


Gambar 14 *Host* Dijalankan pada jam *deny*

Hasil *log* dari pengujian pada Gambar 14 ketika *scheduler* dalam mode *deny* adalah sebagai berikut:

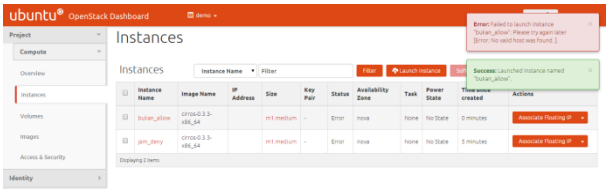
```
2015-05-29 12:51:58.214 3183 DEBUG nova.filters
[req-473010a6-e439-4fb3-8fc8-212e8af01c09 None]
Starting with 7 host(s) get_filtered_objects
/usr/lib/python2.7/dist-packages/nova/filters.py:70
..........
2015-05-29 12:52:27.086 3183 DEBUG
nova.openstack.common.periodic_task [-] Running
periodic task SchedulerManager._run_periodic_tasks
run_periodic_tasks /usr/lib/python2.7/dist-
packages/nova/openstack/common/periodic_task.py:193
```

Pada pengujian yang berikutnya, dilakukan *launch instance* diluar jadwal *deny* yang telah dibuat akan tetapi *scheduler* yang ada berada pada *default permission allow*. Sehingga proses *launch instance* yang dilakukan berhasil yang dapat dilihat pada Gambar 15. Pengujian ini dilakukan sebanyak 10 kali, dimana dari hasil pengujian diketahui rata-rata waktu yang dibutuhkan untuk melakukan *launch instance* adalah 48.2 *milliseconds*.



Gambar 15 Hasil *launch instance* diluar jam *deny*

Pada pengujian yang berikutnya dilakukan *launch instance* dengan menggunakan sistem *filter* yang telah dibuat, yaitu calendar_filter.py yang telah dibuat. Pengujian ini dilakukan dalam mode *allow*, tetapi *instance* dijalankan diluar dari jadwal jam *allow yang dibuat*. Proses *launch instance* di horizon *dashboard* yang dilakukan dapat dilihat pada Gambar 16. Dari hasil percobaan ini diketahui bahwa sistem *scheduler* yang ada membutuhkan waktu 44.6 *milliseconds* untuk melakukan *launch* instance.

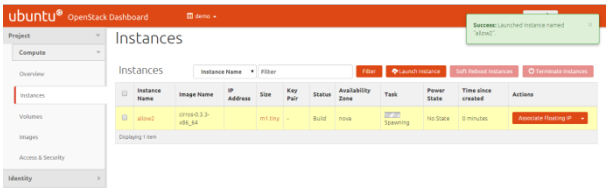


Gambar 16 Host dijalankan bukan pada jam *allow*

Hasil *log* dari pengujian pada Gambar 16 adalah sebagai berikut:

```
2015-05-29 12:57:22.503 3183 DEBUG nova.filters [req-
1b721125-1972-4047-8441-0354403df474 None] Starting
with 7 host(s) get_filtered_objects
/usr/lib/python2.7/dist-packages/nova/filters.py:70
.......
2015-05-29 12:57:22.528 3183 INFO nova.filters [req-
1b721125-1972-4047-8441-0354403df474 None] Filter
CalendarFilter returned 0 hosts
```

Pada pengujian yang terakhir, dilakukan *launch instance* pada jadwal *allow*, tetapi dilakukan di luar jam *allow*. Sehingga proses *launch instance* yang dilakukan berhasil yang dapat dilihat pada Gambar 17. Pengujian ini dilakukan sebanyak 10 kali, dimana dari hasil pengujian diketahui rata-rata waktu yang dibutuhkan untuk melakukan *launch instance* adalah 45.2 *milliseconds*. Data hasil pengujiannya dapat dilihat pada Table 1.



Gambar 17 Proses *launch instance* success pada *allowed time*

5.6. Data Hasil Penelitian

Dari Proses pengujian *launch instance* terhadap sistem yang telah dilakukan diperoleh data-data hasil pengujian yang dapat dilihat pada Table 1.

Table 1 Data hasil pengujian launch instance

Testing	Original	Not Allow	Allow	Not Deny	Deny
1	31	42	42	40	44
2	35	41	50	49	41
3	37	40	47	45	46
4	32	48	41	46	42
5	31	41	48	48	47
6	31	41	43	49	38
7	31	48	41	63	42
8	33	48	52	48	41
9	31	53	44	52	40
10	32	44	44	42	42
Total	324	446	452	482	423
Average	32.4	44.6	45.2	48.2	42.3
Overhead	-	0.376	0.395	0.487	0.305

Table 1 menunjukkan pengujian yang dilakukan dalam beberapa tahapan yaitu: pengujian orginal, allow, not allow, deny, dan *not deny*. Dari hasil Pengujian diketahui bahwa, sistem *scheduler* yang dibuat tidak memerlukan waktu yang cukup lama untuk memproses *launch instance* yang dilakukan oleh *user*, hal ini dapat dilihat dari hasil *overhead* (*latency*) dari sistem yang telah dibuat.

6. KESIMPULAN

Berdasarkan hasil pengujian dapat disimpulkan beberapa hal sebagai berikut:

- Dengan aplikasi *scheduler* yang dibuat, proses *resource scheduling* di lab Universitas Kristen Petra dapat dilakukan dengan membuat jadwal (*schedule*) yang dilakukan oleh admin untuk membatasi *launch instance* di *Calendar Scheduler*.
- Proses reservasi terhadap *resource* yang ada di laboratorium dilakukan berdasarkan jadwal pada *Calendar Scheduler* yang dibuat oleh admin dan melalui proses *filtering* dan *weighting* pada sistem *scheduling* yang ada.
- Sistem *scheduling* yang ada dapat memberikan *availbility zone* yang dapat menentukan *host* (komputer) tertentu yang ingin digunakan oleh *user*. *Scheduler* juga melakukan proses *filter* terhadap ukuran *job* dari *user* dan menentukan komputer yang akan digunakan, serta hanya memperbolehkan *user* untuk melakukan *launch instance* pada waktu dan jam yang telah disediakan oleh *admin*.
- Pencatatan terhadap pekerjaan (*job*) yang telah dilakukan dimasukkan kedalam *database*, dan ditampilkan dalam laporan *instance* yang ada pada *web Calendar Scheduler* dan horizon *dashboard*.
- Pengecekan sistem *login* yang ada sudah terhubung dengan *database* keystone dari OpenStack dengan menggunakan API.
- Aplikasi dapat menampilkan jadwal (*schedule*) dan isinya sesuai dengan data jadwal yang ada dalam *database Calendar Scheduler*.
- Sistem scheduler yang dibuat memiliki *overhead* yang tidak terlalu banyak. Sehingga tidak terlalu memberatkan sistem *scheduler* yang telah ada.

7. REFERENSI

[1] Ahmid, A., & Andersson, E. 2014. *OpenStack Networking Scheduler*. STOCKHOLM: KTH Royal Institute of Technology.

[2] Chen, G. 2014. *KVM – Open Source Virtualization for the Enterprise and.* Open Virtualization Alliance.

[3] Fifield, T., Fleming, D., Gentle, A., Hochstein, L., Proulx, J., Toews. 2014. *OpenStack Operations Guide.* O'Reilly Media.

[4] Foundation, O. 2015. *nova 2015.2.0.dev339 documentation*. URI=http://docs.openstack.org/developer/nova/devref/filter_scheduler.html

[5] Intel. 2011. Open Source Software for Building Private and Public Clouds. Dalam Intel. U.S: Intel Corporation.

[6] Jha, A., D, J., Murari, K., Raju, M., Cherian, V., & Girikumar, Y. 2012 *OpenStack Beginner's Guide*. URI=http://arccn.ru/knowledge-base?pdf=50f6707855f16.pdf

[7] OpenStack. 2011. *HypervisorSupportMatrix - OpenStack*. URI= https://wiki.openstack.org/wiki/HypervisorSupportMatrix

[8] Yamada, T., & Nakano, R. 1997. *Job-shop scheduling*. The Institution of Electrical Engineers.