

Copyright (c) 2015 [DCS Lab, NCTU]

Written by Po-Yu Chung <chungchris.eecs98@g2.nctu.edu.tw>, July 2015

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

# 1 Introduction

A tool for helping the system managers to reconfigure their system easily. The work first gives a complete definition on reconfiguration command. This definition covers the common requirements in the reconfiguration issue, such as scaling direction and scaling policy. Second, the work implements the tool following that definition. Besides, the proposed tool is designed in the manner of to several principles, such as cross-cloud, plug-and-work, and discontinuity avoidance.

## 1.1 Single Command

```
# python RC.py [options]
```

## 1.2 Options

```
# python RC.py -h
```

	Options		Item	Value	Default Value	Comment
1	m	mode	Mode	u, b, rs, rx	-	Mode of using: Update, Beginning, Reconfigure single cluster, Reconfigure the system with xml description
2	k	key	Key file	key file name	KEY.xml	The file contains key information to access user's cloud account.
3	s	config	Configuration file	configuration description file name	reconfig.xml	The configuration description file in xml format to describe the configuration of the system in the clouds.
4	c	cluster	Target cluster	cluster name	-	The target cluster to be scaled
5	t	type	Scaling type	up, out	up	Scale the target cluster in which way, up for vertical, and out for horizontal
6	d	direct	Scaling direction	a, d	a	Scale the target cluster for the purpose of increasing resource or decreasing, "a" for adding resource, and "d" for reducing
7	o	handoff	Handoff	y, n	n	In vertically scaling, handoff the load to the new node before stop the original instance.
8	P	policy	Scaling policy	vcpu, ram, disk, bandwidth	ram	In vertically scaling, reconfigure the original instance to the new one in term of which resource.
9	i	iniscrit	Initialization script activation	y, n	y	Activate the initialization script (if existed) or not after starting the instance.
10	f	straightforward	Pass confirming	y, n	n	Straightforwardly go through the entire confirming step or not.
11	g	gui	Enable GUI	y, n	y	Enable the GUI as necessary or not

## 2 Environment

### 2.1 Python

Python 2 is suggested. Specifically, the version after Python2.6. Usually, the Python 2 is by default installed in most of OS of Linux.

### 2.2 Libcloud

The tool relies on Libcloud library to communicate with IaaS's API. The user can install it on Ubuntu through:

```
# pip install apache-libcloud  
# pip install --upgrade apache-libcloud
```

If the “pip” has not been installed yet, go pip's website (<https://pip.pypa.io/en/latest/installing.html>) to download the “get-pip.py”, and execute with:

```
# python get-pip.py  
# pip install pyinstaller
```

### 2.3 wxpython

The tool utilize wx as GUI displaying tool, installed with:

```
# apt-get install python-wxgtk2.8
```

### 2.5 Fabric

The tool uses Fabric to make SSH connection to the remote host. Install it with:

```
# pip install fabric
```

## 2.4 Get the Tool

Download from: <https://github.com/chungchris/cloud-system-reconfiguration-tool>

Extract it to the preferred work directory.

## 3 Set-up work

### 3.1 Give Keys

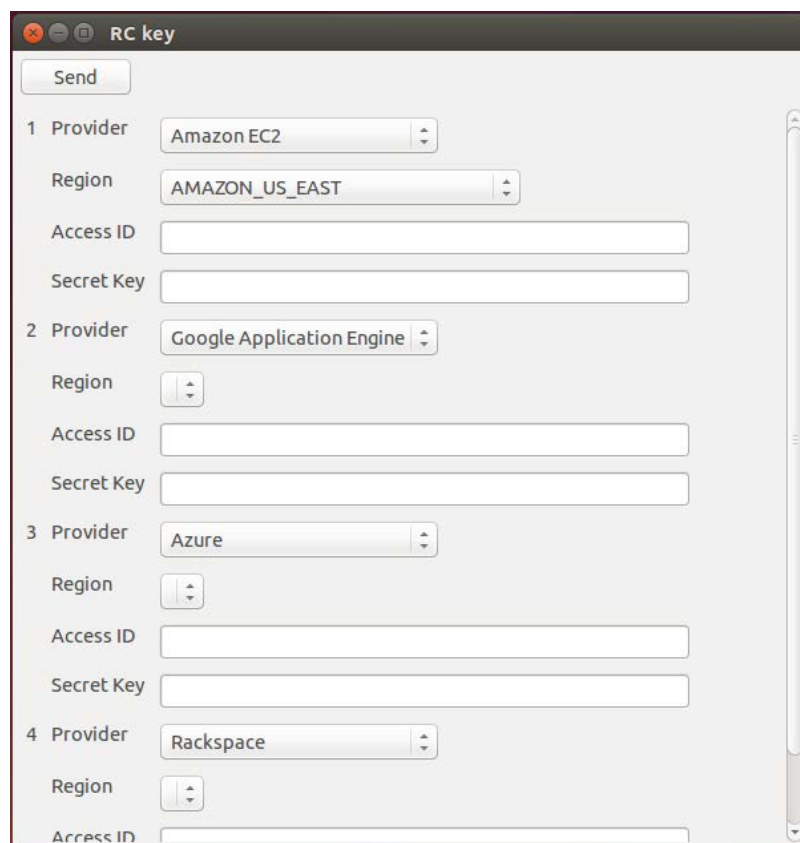
```
# python RC.py
```

The default value of key file is “KEY.xml”. (There is a templet in the work directory.)  
The user should edit this file to provide key information for accessing the cloud resource.

The user can specify the key file by option “-k”. For example:

```
# python RC.py -k mykey.xml
```

If the default or specified key file not existed in the work directory, a GUI will be shown as Figure 1 for you to give key information. (The option “-g” is enabled by default. If the user disabled it, the GUI will not pop up.)



The screenshot shows a window titled "RC key" with a "Send" button at the top left. Below the button are four sections, each for a different cloud provider. Each section contains a "Provider" dropdown menu, a "Region" dropdown menu, an "Access ID" text input field, and a "Secret Key" text input field. The providers listed are Amazon EC2, Google Application Engine, Azure, and Rackspace. The "Region" dropdowns for Google Application Engine, Azure, and Rackspace are currently empty, while the one for Amazon EC2 shows "AMAZON\_US\_EAST".

Figure 1 Give Keys

The key file is stored locally in the work directory in the XML format as Figure 2. The user can directly edit it or generate through GUI.

```
<?xml version="1.0"?>
<!-- Generate Time: 09/05/2015 00:17:55-->
<data>
  <Provider name="EC2" region="us-west-2">
    <AccessID>AKIAJ7B7P4QHL33GUPPA</AccessID>
    <SecretKey>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</SecretKey>
  </Provider>
  <Provider name="GCE" region="">
    <AccessID>178421981909-bc5dqjoiidp813e93hgbp7qf5sureotjq.apps.googleusercontent.com</AccessID>
    <SecretKey>f0f39e601480142fa557c1f060680bc9bf846b75</SecretKey>
  </Provider>
</data>
```

Figure 2 Key File

If the specified key file has already existed, the tool will directly use those keys to access user's cloud.

Once the user want to modify the keys, just directly edit the key file, or specifying another new key file, or delete the original key file and give new keys through GUI.

## 3.2 Give clustering structure information

The proposed tool depends on the configuration description file to construct the structure of user's system.

```
# python RC.py
```

The default value of configuration description file is "reconfig.xml". (There is a templet in the work directory.) The user should edit this file to provide clustering structure as the first-time use.

The user can specify the configuration description file by options "-s". For example:

```
# python RC.py -k mykey.xml -s myconfigdescrib.xml
```

In this command, the tool will first find the key file "mykey.xml". If existed, the tool will obtain all metadata from user's cloud. Then, the tool will find the configuration description file "myconfigdescrib.xml". If not existed, a GUI will pop up as Figure 3 to let user give clustering structure information.

In GUI, the user has to give unique name to each cluster. By default, each instance in the clouds would be considered as a cluster, i.e. each cluster contain just one instance,

and the cluster name will be set as instance name. Besides, the user have to give cluster information including:

- Scale Type- This cluster is aim to be vertically scaling or horizontally scaling.
- Load Balancer- If the cluster is set as “horizontally scaling” type, it should be assigned with a load balancer.
- Image- Once the tool have to generate new instance in the future, the tool will first check whether an image had been assigned. If yes, then use that image to launch the new instance. If not, the tool will chose an instance in the cluster to snapshot to generate an image, and further use that image to launch an instance.
- Initialization Script- As soon as a new instance is been generated, the tool will check whether has to send an initialization script to that new instance on behalf of the user. Note that the tool uses ssh to send the script line by line. And the first 2 lines of the script should be the username and the key file for ssh to connect the target host, e.g.  
`username=ubuntu`  
`keyfile=/home/tmp/Amazon/key/tpcw1.pem`
- Max Running Node- The user can limit the maximum number of running instance in the cluster to avoid too many instance generated by some kind of automatic process.

The screenshot shows the 'RC Clustering' application window. At the top, there is a 'System Name' input field. Below it, the 'Clusters' section contains two cluster configurations, each with a 'Submit' button at the top right.

**Cluster 1:**

- Name: `tpcw_tier1`
- Scale Type: `UP (Vertical)`
- Load Balancer: `TCPW`
- Image: `None`
- Initialization Script: `(No...)`
- Max Running node: `0`
- Nodes:
 

Node	ID	Prov
<input checked="" type="checkbox"/> tpcw_tier1_base	i-56b30e5b	Ama
<input type="checkbox"/> tpcw_tier2_base	i-250e39d2	Ama
<input type="checkbox"/> tpcw_tier3_base	i-1d192eea	Ama
<input type="checkbox"/> tp_redundant	i-0ada6507	Ama

**Cluster 2:**

- Name: `tpcw_tier2`
- Scale Type: `UP (Vertical)`
- Load Balancer: `TCPW`
- Image: `None`
- Initialization Script: `(No...)`
- Max Running node: `0`
- Nodes:
 

Node	ID	Prov
<input type="checkbox"/> tpcw_tier1_base	i-56b30e5b	Ama
<input checked="" type="checkbox"/> tpcw_tier2_base	i-250e39d2	Ama
<input type="checkbox"/> tpcw_tier3_base	i-1d192eea	Ama
<input type="checkbox"/> tp_redundant	i-0ada6507	Ama

Figure 3 Give Clustering Structure

Information of all instances in user's clouds will be obtained and listed on the right side. The user just has to check them to claim that some of them belong to the specific cluster.

To delete a cluster, just leave the cluster name field empty.

After click on "Submit", a configuration description file will be generated and stored locally as Figure 4.

If the "--gui" option is disabled, the GUI will not show even the specified configuration description file not existed. Instead, the "beginning mode" could be used to generate a basic configuration description file:

```
<?xml version="1.0"?>
<!-- Generate Time: 05/07/2015 17:47:46-->
<SYSTEM name="mycloudsystem">
  <!--><CLUSTER name="tpcw_tier1_base"><!--name is necessary-->
  <!--><SCALETYPE>UP</SCALETYPE><!--"OUT" or "UP"-->
  <!--><!--<IMAGE></IMAGE>-->
  <!--><!--<LOADBALANCER></LOADBALANCER>-->
  <!--><!--<INISCRIP></INISCRIP>--><!--The full path of the initi
  <!--><NUMBER>1</NUMBER><!--Do not Change-->
  <!--><RUNNING>1</RUNNING><!--How many running node in this cluster
  <!--><!--<RUNNING_MAX></RUNNING_MAX>--><!--The upper limit of the
  <!--><NODE name="tpcw_tier1_base" id="i-56b30e5b">
  <!--><!--<DRIVER>ec2_us_west_oregon</DRIVER><!--All node in a clus
  <!--><TYPE>t2.micro</TYPE>
  <!--><STATUS>running</STATUS>
  <!--></NODE>
  <!--></CLUSTER>
  <!--><CLUSTER name="tpcw_tier2_base"><!--name is necessary-->
  <!--><SCALETYPE>OUT</SCALETYPE><!--"OUT" or "UP"-->
  <!--><IMAGE>ami-f71b1dc7</IMAGE><!--Give the based image of the in
  <!--><LOADBALANCER>tpcw2</LOADBALANCER><!--Give the name of load-b
  <!--><INISCRIP>/home/chris/Dropbox/Thesis/tpcw2.txt</INISCRIP><!--
  <!--><NUMBER>1</NUMBER><!--Do not Change-->
  <!--><RUNNING>1</RUNNING><!--How many running node in this cluster
  <!--><!--<RUNNING_MAX></RUNNING_MAX>--><!--The upper limit of the
  <!--><NODE name="tpcw_tier2_base" id="i-250e39d2">
  <!--><!--<DRIVER>ec2_us_west_oregon</DRIVER><!--All node in a clus
  <!--><TYPE>t2.micro</TYPE>
  <!--><STATUS>running</STATUS>
  <!--></NODE>
  <!--></CLUSTER>
  <!--><CLUSTER name="tpcw_tier3_base"><!--name is necessary-->
  <!--><SCALETYPE>UP</SCALETYPE><!--"OUT" or "UP"-->
  <!--><!--<IMAGE></IMAGE>-->
  <!--><!--<LOADBALANCER></LOADBALANCER>-->
  <!--><!--<INISCRIP></INISCRIP>--><!--The full path of the initi
  <!--><NUMBER>1</NUMBER><!--Do not Change-->
  <!--><RUNNING>1</RUNNING><!--How many running node in this cluster
  <!--><!--<RUNNING_MAX></RUNNING_MAX>--><!--The upper limit of the
  <!--><NODE name="tpcw_tier3_base" id="i-1d192eea">
  <!--><!--<DRIVER>ec2_us_west_oregon</DRIVER><!--All node in a clus
  <!--><TYPE>t2.micro</TYPE>
  <!--><STATUS>running</STATUS>
  <!--></NODE>
  <!--></CLUSTER>
</SYSTEM>

<!--Pending Nodes-->
```

Figure 4 Configuration Description File



```
# python RC.py -k mykey.xml -s myconfigdescrib.xml -m b
```

Identically, by default, each instance in the clouds would be considered as a cluster, i.e. each cluster contain just one instance, and the cluster name will be set as instance name. This command will generate a basic configuration description file named as “myconfigdescrib.xml”. The user can further edit such file to give the correct description.

In the future use, if the specified configuration description file already existed, the tool will directly use such clustering structure information, combined with real-time instance state obtained from user’s cloud, to generate the necessary knowledge and conscious toward user’s system.

The tool provide “update mode” with options “-m”:

```
# python RC.py -k mykey.xml -s myconfigdescrib.xml -m u
```

The user can update the existed configuration description file according to real-time system state. And we will find that the instances in the clouds but not included in thany cluster would be listed as “pending node” in the tail of configuration description file as Figure 5.

```
<!--Pending Nodes-->
<!--
<NODE name="tp_redundant" id="i-0ada6507">
  <DRIVER>ec2_us_west_oregon</DRIVER>
  <TYPE>t2.micro</TYPE>
  <STATUS>stopped</STATUS>
</NODE>
<NODE name="tpcw3_redundant1" id="i-c72ald30">
  <DRIVER>ec2_us_west_oregon</DRIVER>
  <TYPE>t2.micro</TYPE>
  <STATUS>stopped</STATUS>
</NODE>
<NODE name="tpcw_tier2_base_1435767085.69" id="i-2ed7e1d9">
  <DRIVER>ec2_us_west_oregon</DRIVER>
  <TYPE>t2.micro</TYPE>
  <STATUS>running</STATUS>
</NODE>
<NODE name="tpcw_tier2_redundant" id="i-28488525">
  <DRIVER>ec2_us_west_oregon</DRIVER>
  <TYPE>t2.micro</TYPE>
  <STATUS>stopped</STATUS>
</NODE>
-->
```

Figure 5 Pending Nodes

# 4 Reconfiguration

## 4.1 Reconfigure single cluster

### 4.1.1 Horizontally Scaling

```
# python RC.py -k mykey.xml -s myconfigdescrib.xml -m rs -c tpcw_tier2_base -d a -  
g n
```

This command will scale the cluster named “tpcw\_tier2\_base” in the default direction “horizontally” because, by the Figure 4, this cluster is set as “OUT” scaling type. For the horizontally scaling, adding resource is “scale-out” while decreasing resource is “scale-in”, both in unit of virtual machine. Therefore, this command will add an instance to cluster “tpcw\_tier2\_base”. The “-g” option indicates that just using this command to do reconfiguration, don’t pop up the GUI.

A scale-out type cluster should be assigned with a load balancer compulsorily.

The instances in the scale-out-type cluster could be applied scale-up scaling on demand as well.

As scaling the scale-out-type cluster, the proposed tool would follow the steps below:

#### 4.1.1.1 STEP: Check Load Balancer

Each cluster defined as scale-out type should be assigned with a load balancer. The tool will first check the existence of the load balancer and see if it’s available and legal.

#### 4.1.1.2 STEP: Utilize Existed Instance

In the case of increasing-capability scaling, for the scale-out type cluster, it’s a work of allocating more instances to expand the cluster for sharing the load. If there were nodes in “stopped” state in the cluster, the tool would choose them to be started. Similarly, in the case of decreasing-capability scaling, if there are nodes in “running” state in the cluster, the tool would choose them to be stopped.

#### 4.1.1.3 STEP: Clone Instance

If the previous step is not enough for expanding or shrinking the cluster, the tool will then choose one of the instance in the cluster to be cloned to create a new instance. The tool will first choose the instance in stopped state in order not to affect the service. If all of the instances are in running state, the tool will choose an instance to be stopped temporarily.

The chosen instance will be snapshotted to generate an image. The tool will then use this image to create a new instance that holds the same configuration as the instance from which it's cloned. After the snapshotting, if the instance were that be stopped previously, it would be restarted for continuing on service.

For the decreasing-capability scaling case, if all of the instances in the cluster have been stopped, the tool would not delete the instance because the cloud resource is usually applied with pay-as-you-go model, the instance not in running state rarely brings cost.

#### 4.1.1.4 STEP: Reconfigure Load Balancer

The newly started or created instance will be attached (or detach in the case of stopping the instance) to the load balancer.

#### 4.1.1.5 STEP: Initialization

If the initialization script was given, the script will be sent to the instance to be executed after the instance is started.

After reconfiguration, the specified configuration description file would be updated corresponding to the newest system status.

If there is no “-g” option, it's by default evoke the GUI to help you do the single command configuration like Figure 6. Essentially, this GUI would finally generate a single command identical as what directly given in terminal command line. What's more is that the user could preview the possible consequence of performing reconfiguration before practical reconfiguration. In Figure 16, the cluster “tpcw3” is commanded to scaled up according to the memory capability. It's suggested by the proposed tool that the instance type of the instance “tpcw3” would be changed to “m3.large” from original “m3.medium”.

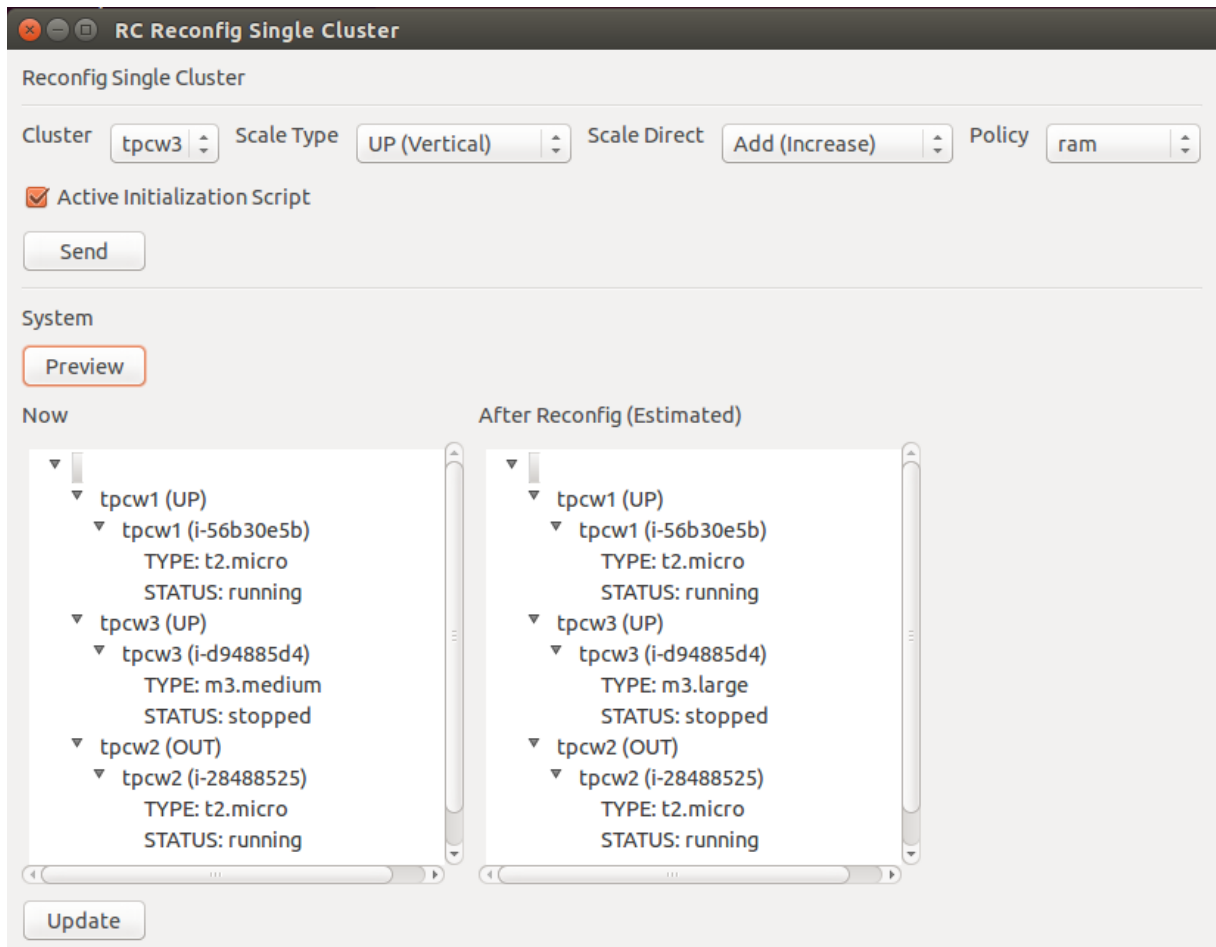


Figure 6 GUI of Reconfigure Single Cluster Mode

#### 4.1.2 Vertically Scaling

```
# python RC.py -k mykey.xml -s myconfigdescrib.xml -m rs -c tpcw_tier3_base -d a -
p ram -g n
```

This command will scale the cluster named “tpcw\_tier3\_base” in the default direction “vertically” because, by the Figure 4, this cluster is set as “UP” scaling type. For the vertically scaling, adding resource is “scale-up” while decreasing resource is “scale-down”. Therefore, this command will replace the original instance in the cluster to a more powerful or weaker instance in term of memory capability. The “-g” option indicates that just using this command to do reconfiguration, don’t pop up the GUI.

As scaling the scale-up-type cluster, the proposed tool would follow the steps below:

#### 4.1.2.1 STEP: Select a Instance to be Scaled

If there are more than one instance in the cluster, the tool will choose an instance to be scaled according to the specified policy. For example, in the case of increasing-capability scaling, if the scaling policy is set as “number of vcpu”, the tool will choose an instance with lowest computing capability as target instance to be scaled later. For example, if there are three instances in a scale-up-type cluster with low-end hardware configuration, say type-L instance, and the user issued the reconfiguration command triple times, the three instances in the cluster would be changed to the instance type with high-end hardware configuration, say type-H instance.

Similarly, in the case of reducing-capability scaling, the instance with highest capability would be chosen to be scaled later.

#### 4.1.2.2 STEP: Select an Appropriate Instance Type to be Changed to

From the example above, the main difference between type-L and type-H instance would be the resource specified as scaling policy. The tool can only choose the instance type from available type list. However, the proposed tool would try the best to select the most appropriate instance type to be reconfigured to in the manner of fine-grained scaling.

The tool will choose the instance type in the same family as original type according the scaling direction request. The chosen type should minimize the difference in term of scaling policy. For example, if the scaling policy is increasing the computing capability, the original instance type holds 1 VCPU, and there are two other instance types, one holds 2 VCPUs and the other holds 4 VCPUs, the one holds 2 VCPUs would be chosen, but not the one holds 4 VCPUs. If there are multiple instance types remained after the previous filtering, they will be ranked according to their price from low to high. The cheapest one would be chosen as the target instance type to be reconfigured to. The algorithm is designed in this way because the instance types in the same family usually hold the same characteristic. Therefore, we can easily choose a type with higher capability in term of specified scaling policy as the target type. We don't have to worry about whether this choice would cause any reduction on the resource other than the policy-specified resource. In other words, the choice would be just more powerful on the policy-specified resource as well as not weaker on the other resources than the original type. Please refer to appendix A and you will find out the similarity in between the instances in the same family.

If there is no appropriate instance type could be chosen from the previous

algorithm, the tool will then makes the first concession that allowing cross-family selection. However, none of the resources other than policy-specified resource should be diminished still. It means that the type selected by this algorithm is not in the same family as the original type though, but it's still not weaker on the resources other than policy-specified resource than the original type, and it's more powerful on the policy-specified resource at the same time.

Another matter worth mention is that not all configuration information of the instance type are provided by the IaaS directly. Some information could be obtained by the IaaS API while some could not. Take Amazon EC2 for example, the memory (RAM) and local storage (disk) information could be obtained through API while the VCPU amount and network bandwidth information could not. Practically, IaaS providers hold their way of counting the computing capability of the instance type. We can only get the mapping information about which instance type equaling to how many VCPU computing capability from the official website or document. Consequently, we could only set up a table of this computing capability mapping information according to the official claim. In the future, if the IaaS change the claim, the mapping table should be updated as well.

#### 4.1.2.3 STEP: Reconfiguration

The chosen instance would be changed to the selected type in this step. If the chosen instance is in running, it will be stopped temporarily and restarted after reconfiguration. It's called "without handoff" mode in vertically scaling. In fact, it's a process of directly replacing the original instance with a more powerful or weaker instance. The vertically scaling is by default "without handoff" mode.

The proposed tool provides another mode called "with handoff" mode in vertically scaling with option "-o" set to "y", i.e.

```
# python RC.py -k mykey.xml -s myconfigdescrib.xml -m rs -c tpcw_tier3_base -d a -  
p ram -o y -g n
```

We can consider this mode as implementing vertically scaling with horizontally scaling. The proposed tool will first use the pre-defined image to launch a new instance with target configuration. After proper initialization, the tool will then stop the original instance. Therefore, there is always an instance in service. That is the difference with "without handoff" mode, which would cause discontinuity while replacing. Nevertheless, in "with handoff" mode, not only an image should be assigned previously, but also a load balancer should be assigned to the cluster, even if

