A Web Portal for Management of Aneka-Based MultiCloud Environments

Mohammed Alrokayan and Rajkumar Buyya

Cloud Computing and Distributed Systems (CLOUDS) Laboratory,
Department of Computing and Information Systems,
The University of Melbourne, Parkville, Victoria 3010, Australia

Emails: a@alrokayan.com, rbuyya@unimelb.edu.au

Abstract

Many Cloud providers offer services with different sets of configurations and settings. This makes it difficult for their clients to seamlessly integrate various services from different Cloud providers. To simplify, we developed an extendible Cloud Web Portal (CWP), a comprehensive open source Cloud management portal that aims to deliver a foundation for researchers and developers to prove a research concept, test a code or deliver a product in a fast and easy to use graphical user interface. Also, it aims to seamlessly integrate different Cloud services by providing a flexible architecture and design system. CWP is based on Aneka, which allows developers to use a set of .Net-based APIs for monitoring, billing/accounting, scheduling, and provisioning. Aneka provision services from private, public or hybrid Clouds. Our evaluation results show that Aneka scheduling algorithm performs efficiently for executing tasks in distributed machines.

Keywords: Cloud Management, Scheduling, Provisioning, MultiCloud, Portal, Cloud Services.

1 Introduction

Contemporary Cloud computing solutions, both research projects and commercial products, have mainly focused on Infrastructure as a Service (IaaS) model due to the uncertainty in the other models like Platform as a Service (PaaS). This uncertainty is caused by the lack of proper standards in the IaaS level especially in terms of APIs for federated Clouds. This drives the users to develop their own platforms and portals from scratch trying to use multiple IaaS providers' APIs.

As a result, users need an open portal that is flexible to adapt this uncertainty and regular variations in Cloud infrastructures. Our Cloud Web Portal (CWP) aims to provide an open source portal for easy adjustment to adapt the frequent changes in Cloud computing technology. CWP has been developed using Microsoft ASP.NET MVC 4[1] with Razor syntax (Galloway et al. 2011). It has the capability to build, test, deploy, and scale applications easily and rapidly.

There are three main common types of Cloud PaaS: Firstly, PaaS for application deployment, where a product is deployed in distributed resources over the Cloud, for example: web applications, which is the most common type of application to be deployed on the Cloud. Secondly, PaaS for batch processing and scalable grid computing over the Cloud, where a batch of files is processed in distributed machines to accelerate the application performance. Thirdly, PaaS for multi and hybrid Cloud management where users manage multiple resources from different providers through one interface. CWP is not for application deployment as in type one of Cloud PaaS. However, it supports the other two types. It supports the second type of Cloud PaaS by using the Aneka framework [2] for three different kind of programming models: Thread, Task and MapReduce. Also, it supports the third type of Cloud PaaS by using the provisioning library of Aneka which support three different providers: Amazon AWS, Microsoft Windows Azure and GoGrid (Wei et al. 2011).

CWP provides several services and components for developers, researchers and deployment teams to integrate their work through a graphical web portal. Those different services and components will be discussed in Section 4. Our evaluation method, encompassing cases with up to 80 experiments using three different parameters, shows that the Aneka scheduling algorithm perform efficiently for batch processing in distributed machines, especially when the number of workers is increased. Surprisingly, with all the network latency and overhead to send and receive data, the 49 image rendering tasks does not have significant effect on Aneka performance as shown in Section 8.

The key **contributions** of our paper are: 1) an extensible architecture for Web portal for cloud computing environments, 2) a methodology for creation of adapters or widgets for monitoring or interacting with different cloud platforms, 3) a prototype software system demonstrating these capabilities and their mapping to the Aneka cloud application platform, and 4) a detailed evaluation and demonstration of our portal functionalities by deploying in a hybrid cloud environment by utilizing Melbourne private cloud and Amazon EC2 resources.

The rest of the paper is organized as follows: In next section, we present various open source cloud projects and how they are compared to CWP. Then in Section 3 we discuss the motivations behind CWP. Section 4 describes the different components and services of CWP and how they are integrated with Aneka. Then we discuss the CWP graphical user interface usability for the end-users and the flexibility of the underlying code for developers in Section 5. An example of a page request has been illustrated in a sequence diagram in Section 6. Then in Section 7, based on the NIST definition of Cloud Deployment

[1]ASP.NET MVC 3: http://www.asp.net/mvc/mvc3

[2]Manjrasoft Pty Ltd http://manjrasoft.com

Models, we summarize how CWP and Aneka support all levels of deployment models. In Section 8, an evaluation of CWP shows the Aneka scheduling performance using several parameters and statistics. We close in Section 9 with conclusions and future work.

2 Related Works

Cloud computing has been driven mainly by the industry; as a result the most common and useful services can be found there. This section is a result of a study researching seven Cloud computing projects. It shows two main features: the service and deployment models. A summary of the related works is shown in Table 1 along with Cloud Web Portal for comparison. In our research, we focus on the deployment model which will be explained in Section 7. The following is a summary of the seven projects:

CloudFoundry[3]

It is a portal to deploy applications in distributed machines over a Cloud. It supports limited number of public Cloud providers. Also, it supports OpenStack[4] private Clouds. A commercial public Cloud version of CloudFoundry project by VMWare can be found at CloudFoundry.com[5], which supports only one public Cloud provider even though the open source version of the project supports Multi-Public Cloud. CloudFoundry is for application deployment, not for MultiCloud management like Delta Cloud project.

Delta Cloud[6] and jCloud[7]

These two projects are different than the others, they are libraries to manage Multi-Public Cloud providers and manage hybrid infrastructure services using one API instead of dealing with multiple APIs for different providers. Delta Cloud is written in Ruby and it supports wide range of private Cloud projects and public Cloud providers, which allows users to manage several instances through one REST-based API for simple any-platform access. jCloud is almost the same as Delta Cloud except than it is written in Java.

3 Motivations

Not all Cloud platforms and portals fall into one category, CloudFoundry, Microsoft Windows Azure[8], CloudBees[9], and Google App Engine[10], for example, are platforms for application deployment and they vary according to which programming language the user want to deploy on the Cloud. Other platforms, like RightScale[11], is for Multi-Public Cloud management where multiple VM instances from multiple providers and monitoring can be provisioned via one management console.

Cloud Web Portal (CWP) is different from the others, on top of Multi-Public Cloud management, it allows users to run batch processing jobs over the Cloud in parallel and distributed manner leveraging Aneka framework (Vecchiola et al. 2012). Aneka supports three different programming models: Task, Thread and MapReduce. CWP provides a flexible user interface for Cloud administrators, developers and researchers to manage multiple nodes in a Cloud. Also, it aims to give non-technical users an easy to use fully functional dashboard to view a summary of the current system status, and allow them to apply changes to the Cloud system according to their granted permissions. CWP allows the Cloud developers to add any feature to the portal or develop an application by implementing what we call "widgets" (which is Detailed in Section 5) and add it to the portal to be used instantly.

4 CWP Architecture and Services

CWP has been designed and developed to provide developers with an easy to understand code structure. Also, it has a flexible architecture for any future changes or adjustment to the portal. CWP gives portal users a Multi-Public Cloud support to provision public VMs from multiple different providers. Also, it has the capability to execute batch processing jobs over the Cloud according to the scheduling policy that the user chooses using Aneka APIs.

The CWP architecture in Figure 1 shows the different CWP services that users and developers can use and integrate. Aneka provides the main services for CWP such as monitoring, SLA-Resource management and resource provisioning whether those resources are local desktop machines, private Cloud, public Cloud or hybrid Cloud. CWP graphical user interface and its usability is described in the next section.

4.1 Provisioning

CWP provisions resources from all different deployment models using Aneka (?). Aneka uses different scheduling algorithms, for example, the deadline scheduling algorithm (Vecchiola et al. 2012) which allows Aneka to provision to public Clouds dynamically when the local desktop grid is not enough to execute the job within its given deadline. Another example of an algorithm is the budget-based algorithm that aims to limit the resources that are provisioned and chooses a set of resources that can finish the job within the specific budget.

4.2 Monitoring

Aneka monitors the running nodes through a heartbeat-based approach that performs periodic checks on the availability of the node. If one of the worker nodes fails executing a task, Aneka migrates that task to another node to be executed. If a master node fails, a discovery system looks for another master within the defined Cloud domain to take over and continue scheduling the tasks starting from where the previous master node stopped.

Service Measurements

CWP measures the services from different Cloud providers. CWP suggests the best service(s) to provision based on the users' QoS, budget, and monitored data. The SMICloud (Garg et al. 2011) algorithm has been used for multi-criteria selection of different Cloud services but has not been integrated with Aneka yet.

[3]Cloud Foundry Open Source: http://cloudfoundry.org
[4]Apache OpenStack: http://www.openstack.org
[5]Cloud Foundry: http://cloudfoundry.com
[6]Apache Deltacloud API: http://deltacloud.apache.org
[7]Apache jclouds: http://www.jclouds.org
[8]Microsoft Windows Azure: http://www.windowsazure.com
[9]CloudBees Java PaaS: http://www.cloudbees.com
[10]Google App Engine: https://cloud.google.com/products
[11]RightScale Cloud Management: http://www.rightscale.com

Table 1: Related Works Projects

Project	Service Model	Deployment Model	Language	Licence
CloudFoundry	Portal for application deployment	Private/Multi-Public Cloud management	Java	Apache
Delta Cloud	Ruby Library for Multi-Public Cloud support	Multi-Public Cloud	Ruby	Apache
jCloud	Java Library for Multi-Public Cloud support	Multi-Public Cloud	Java	Apache
Cloud Web Portal	Cloud portal for Multi-Public Cloud management and batch processing	Private/Multi-Public Cloud/Hybride-Cloud	.Net	Apache

SLA-Based Resource Allocation and Provisioning

Aneka can allocate resources based on the users' QoS. Unfortunately, due to the current limitations, almost all public Cloud service providers provide a static SLA, mainly for availability, that can not be negotiated. However, an Aneka prototype performance results show the feasibility and effectiveness of SLA-based resource provisioning in Clouds (Buyya et al. 2011).

4.3 Scheduling

Aneka support three different programming models: Task, Thread, and MapReduce associated with different scheduling algorithms based on time (Vecchiola et al. 2012) and budget. A user or a developer can implement or execute applications mixing any of the programming models with any of the scheduling algorithms. This kind of scheduling is important, for example, for scientist to execute batch processing type of applications and for graphical designers to render images or videos in a shorter time.

4.4 Billing/Accounting

Aneka is Market-oriented system, which can be used by a Cloud broker. Aneka has a fully functional billing and accounting system mainly for desktop grids. There are two different models: Pay-Per-Task or Pay-Per-Resource. The first one is to set a price for a task on each node to charge the users or the brokers depending on how many tasks they have executed. The second model is to set a price for the hourly usage on a node regardless on how many tasks the user will execute.

5 CWP Interface and Component Usability

CWP helps developers to create Cloud applications with decoupled components (input logic, GUI logic, and business logic). The loose coupling among the three main components of CWP provides the ability for parallel development, flexibility in changes and fast debugging. CWP infrastructure has been designed and implemented not just for the developers and researchers but also for any non-expert users to use the portal easily. Also, it brings flexibility in development and usability for the end-users to use its graphical user interface especially the concept of widgets and dashboard. This flexibility and usability in design allows the Cloud developers to edit any existing feature, widget or application and add almost any desired one easily.

The error messages are shown on the top right corner of the portal, which we call Issues Centre. It

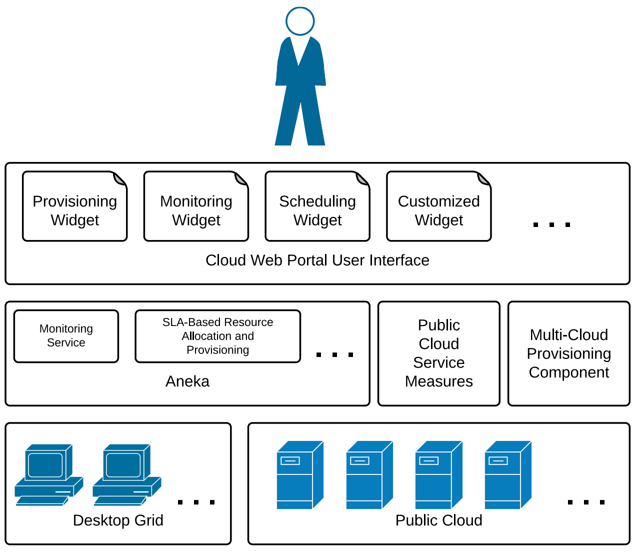

Provisioning Widget
Monitoring Widget
Scheduling Widget
Customized Widget
. . .
Cloud Web Portal User Interface
Monitoring Service
SLA-Based Resource Allocation and Provisioning
. . .
Aneka
Public Cloud Service Measures
Multi-Cloud Provisioning Component
. . .
Desktop Grid
. . .
Public Cloud

Figure 1: CWP Architecture

gives the portal users a summary and a quick overview of what are the current errors and warnings. As soon as the user clicks on any of the errors or warnings it shows a dialog to solve the problem easily. In addition to the Issues Centre, CWP has the Activity Centre that shows to users a summary and a quick overview on the current running tasks. Next we discuss further the Widgets, Dashboard, Issues Centre and Activity Centre.

5.1 Widgets and Dashboard

Widgets give the developers a quicker method to develop Cloud applications leveraging all the components that have been mentioned in Section 4. The concept of widgets gives the users the flexibility to adjust the graphical user interface and to modify the business logic easily. Each widget is designed to be wrapped-up with <*article*> and <*section*> HTML tags; those wrapped-up tags specify the widgets' configurations, such as the width. A widget creates a box of information or form depending on the developer design. The width of a widget is between 1 and 12, so 12 is the full width of the browser window, see Figure 2 for example.

The HTML code in Figure 2 displays two widgets, each one fills half the width of the user's browser window (*class="grid_6"*). The first widget that will be shown is the *_Clouds* controller and *"Details"* action to shows details for a specific Cloud as shown in Figure 4 (we are also passing the Cloud id *id* = *@Model.CloudId*). The code of this widget can be found in the file *"Controllers_CloudsController.cs"*

```
<article class="container_12">
  <section class="grid_6"><br />
    <div class="block-border">
      @Html.Action("Details", "_Clouds",
        new { id = @Model.CloudId })
    </div>
  </section>

  <section class="grid_6"><br />
    <div class="block-border">
      @Html.Action("CPU_Utilization_Range",
        "_Charts",
        new { id = @Model.CloudId })
    </div>
  </section>
</article>
```

Figure 2: An example of an HTML code to display two widgets

```
<div class="block-content">
    <h1>New CWP Widget Title</h1>
    <div class="infos">
        //Any HTML or ASP.Net/RAZOR code
    </div>
</div>
```

Figure 3: An example of a widget code

in function *"Details"*, which returns a View object. The second widget is almost the same as the first one, it calls the *"CPU_Utilization_Range"* controller and *"_Charts"* action to draw a CPU utilization chart for a specific Cloud as shown in Figure 5. This shows how easy it is to customize and edit any widget in CWP. A new widget can be added by creating a new HTML file and following a specific format to match the CSS of CWP, for example, an HTML file content is shown in Figure 3

5.2 Issue and Activity Center

The issues and activity centers were designed to monitor the Cloud resources and to keep checking the status of Aneka workers and masters. Also, it shows the ongoing tasks for the provisioned machines, masters and/or workers.

Issue Center

The issues centre, as shown in Figure 6, checks the status of the CWP resources, whether they are active or failed last execution. Also, it displays an error or a warning message according to how critical the issue is. Issue center gives the users the ability to fix any issue easily by clicking on any of the listed issues to popup a dialog box to solve it.

Activity Center

The activity centre, as shown in Figure 7, checks if there is any tasks in progress. Each one of those tasks is clickable to open a popup dialog box to show the users more information about the selected task.

6 CWP Sequence Diagram

CWP sequence diagram shows how its components operate with each other and in what order. The sequence diagram in Figure 8 was simplified to show details of only one widget (_Clouds\Details). The rest of the widgets follow almost the same approach.

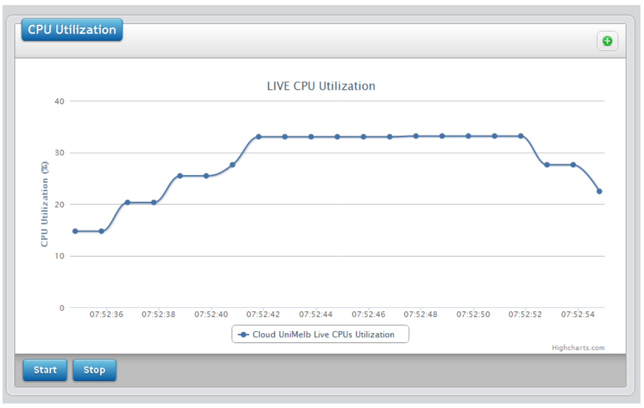
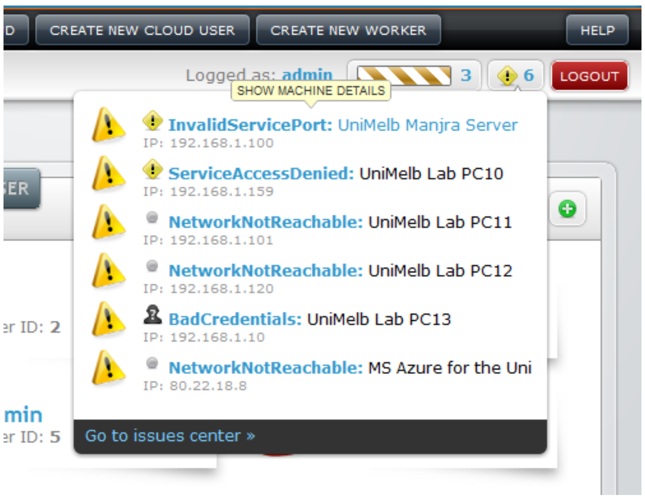
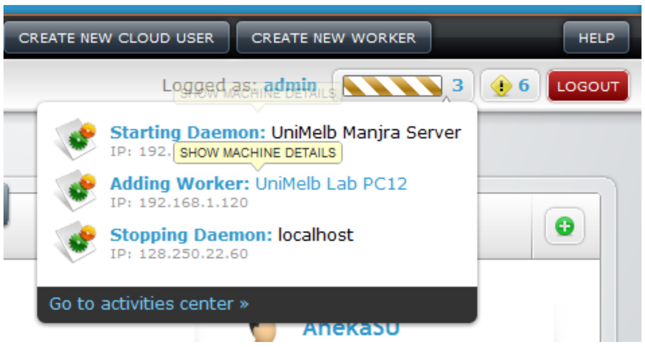


Figure 5: Live CPU utilization widget



InvalidServicePort: UniMelb Manjra Server
IP: 192.168.1.100
ServiceAccessDenied: UniMelb Lab PC10
IP: 192.168.1.159
NetworkNotReachable: UniMelb Lab PC11
IP: 192.168.1.101
NetworkNotReachable: UniMelb Lab PC12
IP: 192.168.1.120
BadCredentials: UniMelb Lab PC13
IP: 192.168.1.10
NetworkNotReachable: MS Azure for the Uni
IP: 80.22.18.8
Go to issues center »

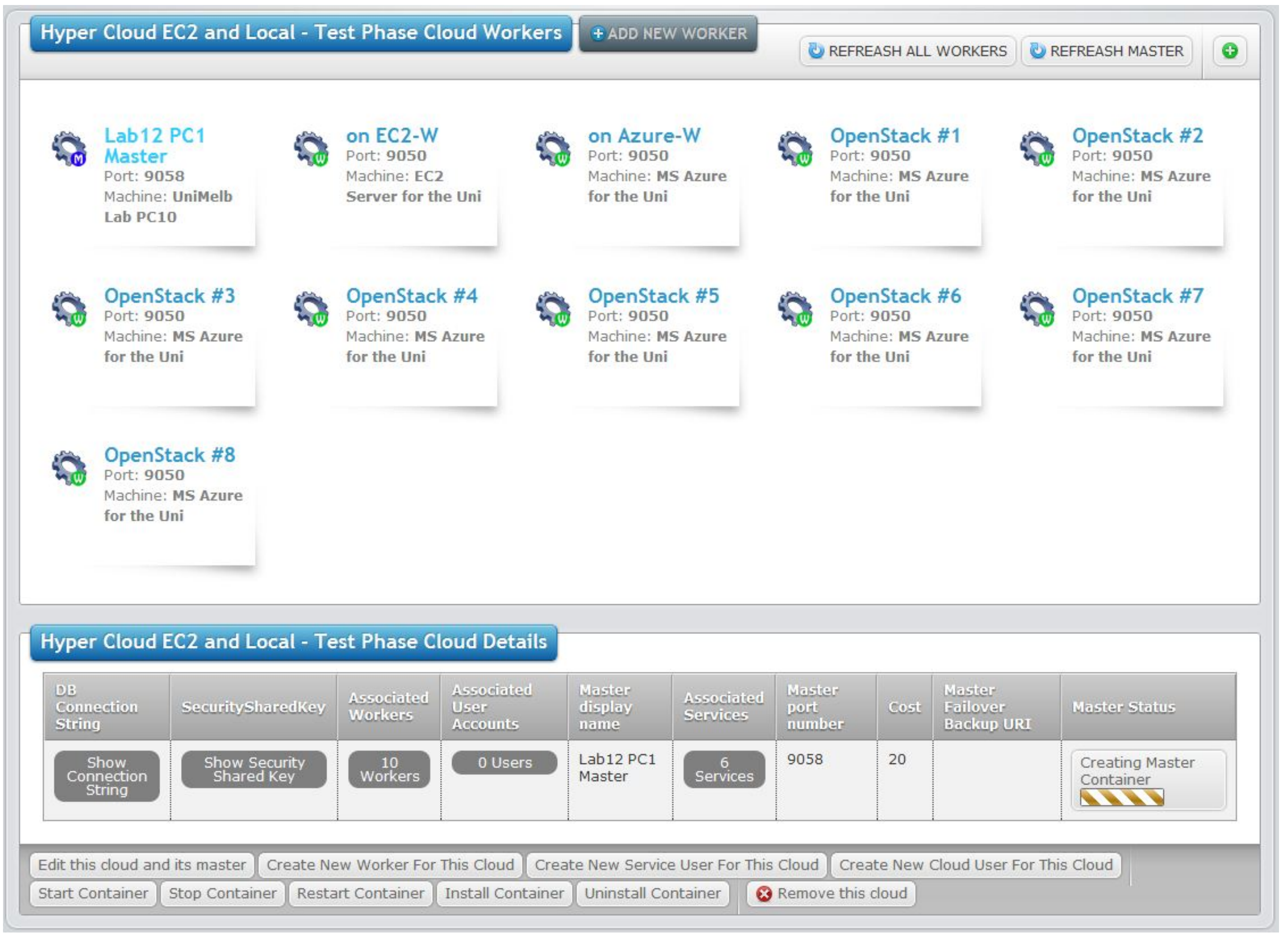
Figure 6: Different issues have been addressed by CWP Issue Center



Starting Daemon: UniMelb Manjra Server
IP: 192.
Adding Worker: UniMelb Lab PC12
IP: 192.168.1.120
Stopping Daemon: localhost
IP: 128.250.22.60
Go to activities center »

Figure 7: Activity Center shows different stages of ongoing tasks

The sequence diagram in this section is an example of a request to this page: "\CloudManagement\CloudDetails\1", where the web server calls "CloudDetails" action in "CloudManagement" controller passing the Cloud id "1". The controller requests a Cloud object from the Entity Framework (EF) to send it to the CWP view: "CloudManagement\CloudDetails". This view calls three widgets as shown in Figure 8. The first widget is: "_Clouds\Details\1", which shows the master and a list of workers along with some information about the selected Cloud, like running services. The widget sends several requests to the Entity Framework to get such information. Then the widget returns



Lab12 PC1 Master
Port: 9058
Machine: UniMelb Lab PC10

on EC2-W
Port: 9050
Machine: EC2 Server for the Uni

on Azure-W
Port: 9050
Machine: MS Azure for the Uni

OpenStack #1
Port: 9050
Machine: MS Azure for the Uni

OpenStack #2
Port: 9050
Machine: MS Azure for the Uni

OpenStack #3
Port: 9050
Machine: MS Azure for the Uni

OpenStack #4
Port: 9050
Machine: MS Azure for the Uni

OpenStack #5
Port: 9050
Machine: MS Azure for the Uni

OpenStack #6
Port: 9050
Machine: MS Azure for the Uni

OpenStack #7
Port: 9050
Machine: MS Azure for the Uni

OpenStack #8
Port: 9050
Machine: MS Azure for the Uni

Hyper Cloud EC2 and Local - Test Phase Cloud Details

DB Connection String	SecuritySharedKey	Associated Workers	Associated User Accounts	Master display name	Associated Services	Master port number	Cost	Master Failover Backup URI	Master Status
Show Connection String	Show Security Shared Key	10 Workers	0 Users	Lab12 PC1 Master	6 Services	9058	20		Creating Master Container

Edit this cloud and its master
Create New Worker For This Cloud
Create New Service User For This Cloud
Create New Cloud User For This Cloud
Start Container
Stop Container
Restart Container
Install Container
Uninstall Container
Remove this cloud

Figure 4: Cloud Details widget shows 10 workers and one master in a Cloud

HTML content to the view. The view repeats the same approach to get the HTML content from all the widgets. Finally, after the view wraps all the HTML contents, it sends the completed HTML page to the user.

Components in Figure 8 are decoupled to be replaced or extended. This gives the portal users and developers a flexibility to build adapters to suit their work or research.

7 Deployment Models

NIST defines four different Cloud deployment models (?), which CWP supports via Aneka along with the Desktop Grid model.

7.1 Desktop Grid

Aneka utilizes the unused computational power of desktop Personal Computers (PCs) connected on local area networks (LAN), virtual LAN, or over the Cloud. The main objective of a desktop grid is to accelerate application execution, especially distributed-aware applications that split a job into tasks. Aneka provides several .Net libraries for developers to develop this type of applications to be distributed among Aneka workers. Desktop grid allows organizations to utilize unused PCs resources without affecting the productivity of PC users.

7.2 Public Cloud

This is the most common model of the Cloud which allows several users to share the same infrastructure to reduce the cost and utilize the shared resources. The main features of the public Cloud are the resource availability, elasticity and cost efficiently. NIST defines public Cloud as "The Cloud infrastructure is provisioned for open use by the general public" (?). Aneka provisions public Cloud resources from Amazon EC2, Microsoft Windows Azure and GoGrid (Wei et al. 2011) to execute batch processing among them.

Multi-Public Cloud

Research on MultiCloud (Xiong et al. 2011), InterCloud and Cloud Federation (Celesti et al. 2010) have emerged questioning the openness of the Cloud and discussing avoiding vendor lock-in. Also, deploying application on MultiCloud reduces the chance of outage by leveraging multiple running application servers on different providers, and switching to the most efficient one in case of any failure. In web applications, for example, MultiCloud application servers allow the system to redirect the users requests to where they can get the best experience by shortening the response time and increasing the availability[12]. CWP uses Aneka to provision infrastructure Multi-Public Cloud resources from any of Amazon EC2, Microsoft Windows Azure and/or GoGrid (Wei et al. 2011). Those resources can be managed in a single resource pool or multiple resource pools.

[12]Cedexis: http://www.cedexis.com/country-reports/

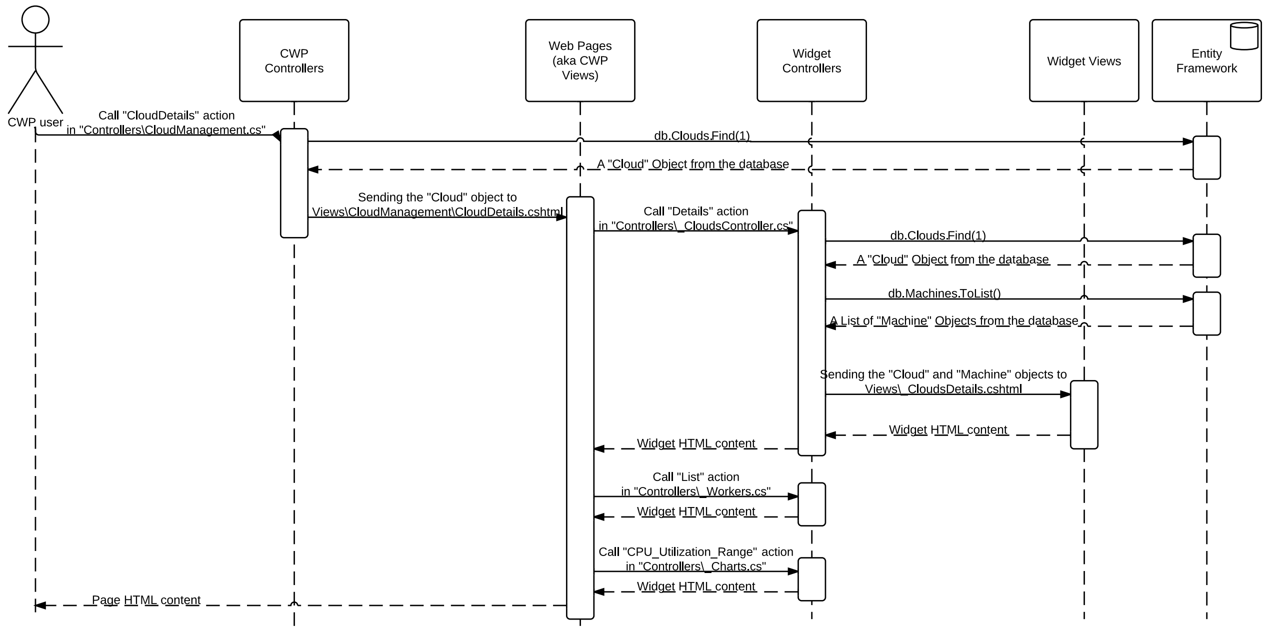

CWP user
CWP Controllers
Web Pages (aka CWP Views)
Widget Controllers
Widget Views
Entity Framework
Call "CloudDetails" action in "Controllers\CloudManagement.cs"
db.Clouds.Find(1)
A "Cloud" Object from the database
Sending the "Cloud" object to Views\CloudManagement\CloudDetails.cshtml
Call "Details" action in "Controllers_CloudsController.cs"
db.Clouds.Find(1)
A "Cloud" Object from the database
db.Machines.ToList()
A List of "Machine" Objects from the database
Sending the "Cloud" and "Machine" objects to Views_CloudsDetails.cshtml
Widget HTML content
Widget HTML content
Call "List" action in "Controllers_Workers.cs"
Widget HTML content
Call "CPU_Utilization_Range" action in "Controllers_Charts.cs"
Widget HTML content
Page HTML content

Figure 8: CWP Sequence Diagram Requests Page: "\CloudManagement\CloudDetails\1"

7.3 Private Cloud

Many open source projects (Cloud Stack[13], Deltacloud, Eucalyptus (Nurmi et al. 2009), Open Nebula[14], Open Stack, and Cloud Foundry) emerged giving organizations the opportunities to host a private Cloud. It provides more control on the data and increases security. NIST defines private Cloud as: "The Cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units)." (?). Almost any private Cloud project can be public when hosted and exposed to the general public as a service. Manjrasoft Aneka has an ongoing project to support automated provisioning for OpenStack private Cloud.

Outsourced Private Cloud

Private Cloud usually refers to the on-premise private Cloud where an organization hosts the head node locally and the reset of the resources on the public Cloud. However, some providers offer private Cloud service that can be almost 100% outsourced, which is basically a public Cloud service that has been adjusted to isolate the infrastructure of the service to be dedicated to single organization. This model offers the security strength of the private Cloud and the availability, elasticity and cost efficient of the public Cloud.

7.4 Hybrid Cloud

NIST defines hybrid Cloud as: "The [hybrid] Cloud infrastructure is a composition of two or more distinct Cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., Cloud bursting for load balancing between Clouds)" (?). Aneka has the ability to provision resources in hybrid Cloud (Vecchiola et al. 2012) where multiple tasks are distributed between local desktop grid PCs and also Multi-Public Cloud.

8 Performance Evaluation

We evaluate Aneka performance, and the efficiency of its scheduling algorithm. Aneka schedules jobs that consist of groups of tasks among workers via a master node.

Seven small size Amazon EC2 instances were used, one master and six workers. All the machines have the same specification. Our evaluation method encompasses cases with up to 80 experiments using three different parameters: number of workers, programming models, and number of tasks. The parameter of interest is the time in seconds that Aneka takes to finish executing a job. Four different numbers of workers have been used: one (representing the sequential execution), two, four and six workers. Two programming models were used: Thread, using the Mandelbrot application, and Task using a distributed version of POV-Ray application. Both applications, Mandelbrot and POV-Ray, use Aneka APIs for scheduling. Two number of tasks have been used: 25 tasks (which is rendering an image with 5 columns and 5 rows) and 49 (which is 7x7 image rendering).

Comparing the sequential execution of a job (number of workers = 1) with the parallel execution (number of workers = 2, 3, and 6), Tukey simultaneous tests shows P-Values close to zero, which means statistically that there is a significant difference between the different number of workers. Also, ANOVA General Linear Model test for the time in second versus the number of workers shows *P-Value <0.001*, which means that the time and number of workers are strongly related and each one of them affects the other, so having more workers means a huge reduction in time for executing a job.

Also, Two-Sample T-Test for the time and number of tasks shows a *P-Value <0.001* and *95% CI for difference (34.51, 61.35)*, which means that the number of the tasks effects significantly the time that Aneka needs to execute a job. As a result, both number of workers and number of tasks have a great impact

[13]Cloud Stack: http://cloudstack.org/
[14]Open Nebula: http://opennebula.org

```
For Task Programming Model:
Time = 60.1007 - 9.56779 * Num of workers
+ 0.780371 * Num of tasks

For Thread Programming Model:
Time = 12.1695 - 9.56779 * Num of workers
+ 0.780371 * Num of tasks
```

Figure 9: Regression analysis for the two programming models with the number of tasks and the number of workers

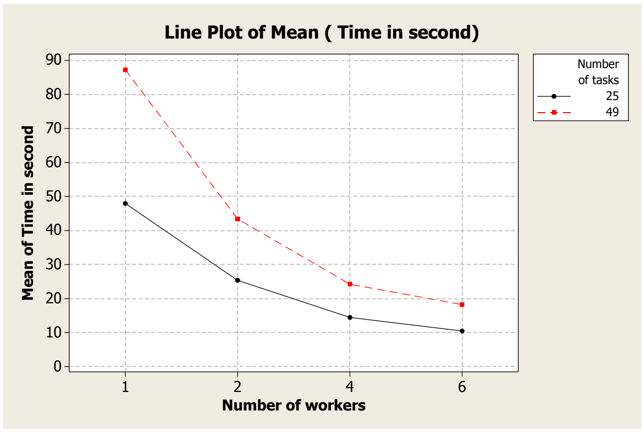


Figure 10: LinePlot of mean time in second

on the time for Aneka to finish executing a job, and a correlation analysis has been preformed that supports this result.

Looking at the two programming models, the regression analysis shows the equations in Figure 9. That means both models have the same spread but the thread programming model (*60.1007*) always executes faster than the task programming model (*12.1695*).

The line plot (Figure 10) shows how the increase of the number of workers to execute the tasks in parallel reduces the time that Aneka took to execute a job. Also, the sequence execution (number of workers = 1) has a wide gap between 25 and 49 tasks, while it is the opposite on parallel execution when we have 2, 4 or 6 workers, and the gap get closer when we have more workers.

The box plot (Figure 11) of the time Aneka takes to execute a job grouped in the number of tasks and the number of workers - shows how the number of workers decreased the time the Aneka takes to execute a job significantly. Also, the difference between the two box plots within the same number of workers is large in the sequence execution (number of workers = 1) while it becomes narrower when we increase the number of workers.

As a result of this experiment, Aneka scheduling algorithm has been proven to perform efficiently for executing tasks in distributed machines, especially when the number of workers is increased. Surprisingly, with all the network latency and overhead to send and receive data, the 49 image rendering tasks does not have significant effect on Aneka performance compared to 25 tasks as shown in the box plot Figure 11.

9 Conclusion and Future Work

Cloud Web Portal (CWP) is an open source Cloud management portal for researchers and developers to prove a research concept, test a code or deliver a product. CWP uses Aneka as it is framework, which gives CWP several features especially in monitoring, billing/accounting, scheduling, and provisioning of local desktop PCs, private, public or hybrid Cloud. Aneka can be used by the developers using its APIs. Our evaluation method encompassing cases with up to 80 experiments using three different parameters show that Aneka scheduling algorithm perform efficiently for executing tasks in distributed machines, especially when the number of workers is increased. Surprisingly, with all the network latency and overhead to send and receive data, the 49 image rendering tasks do not have significant effect on Aneka performance compared to the 25 tasks.

As CWP is extendible, it is possible to build adapters for mapping its capability to other Cloud platforms. One of the undergoing and future works is the support for Multi-Public Cloud service measurement and keep tracking of the Cloud condition to select and allocate Multi-Public Cloud resources more accurately based on the users QoS and budget – by implementing SMICloud (Garg et al. 2011) on CWP.

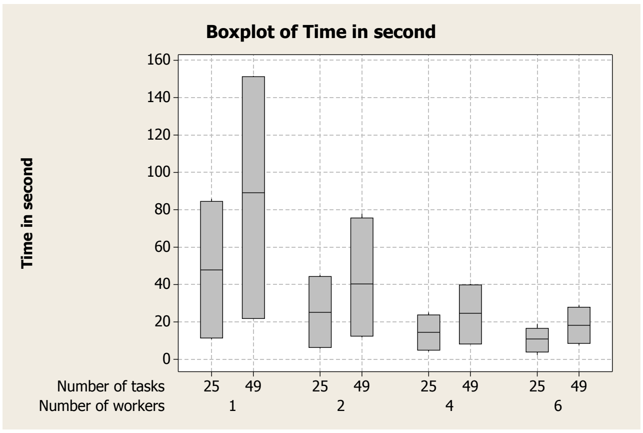


Figure 11: Boxplot of time in second

Software Availability

A software of Cloud Web Portal (CWP) represented in this paper can be downloaded from http://www.cloudbus.org/cwp.

Acknowledgements

The authors would like to thank Dr.Rodrigo Calheiros and Nikolay Grozev for their valuable comments for improving the paper.

References

Buyya, R., Garg, S. & Calheiros, R. (2011), SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture, and Solutions, *in* ‘Cloud and Service Computing (CSC), 2011 International Conference on’, pp. 1 –10.

Celesti, A., Tusa, F., Villari, M. & Puliafito, A. (2010), How to Enhance Cloud Architectures to Enable Cross-Federation, *in* ‘Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on’, pp. 337 –345.

Galloway, J., Haack, P., Wilson, B. & Allen, K. S. (2011), *Professional ASP.NET MVC 3*, 1 edn, Wrox. The book in Kindle.

Garg, S., Versteeg, S. & Buyya, R. (2011), SMICloud: A Framework for Comparing and Ranking Cloud Services, *in* 'Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on', pp. 210 –218.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. & Zagorodnov, D. (2009), The Eucalyptus Open-Source Cloud-Computing System, *in* 'Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on', pp. 124 –131.

Vecchiola, C., Calheiros, R. N., Karunamoorthy, D. & Buyya, R. (2012), 'Deadline-Driven Provisioning of Resources for Scientific Applications in Hybrid Clouds with Aneka', *Future Generation Computer Systems* **28**, 58 – 65.

Wei, Y., Sukumar, K., Vecchiola, C., Karunamoorthy, D. & Buyya, R. (2011), 'Aneka Cloud Application Platform and Its Integration with Windows Azure', *CoRR* **abs/1103.2590**.

Xiong, N., Rindos, A., Russell, M. L., Robinson, K. P., Vandenberg, A. & Pan, Y. (2011), 'Sharing Computing Resources to Satisfy Multi-Cloud User Requirements', *International Journal of Cloud Computing* **1**, 81–100.