International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:6, 2015

Developing a Web-Based Workflow Management System in Cloud Computing Platforms

Wang Shuen-Tai, Lin Yu-Ching, Chang Hsi-Ya

***Abstract*—**Cloud computing is the innovative and leading information technology model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort. In this paper, we aim at the development of workflow management system for cloud computing platforms based on our previous research on the dynamic allocation of the cloud computing resources and its workflow process. We took advantage of the HTML5 technology and developed web-based workflow interface. In order to enable the combination of many tasks running on the cloud platform in sequence, we designed a mechanism and developed an execution engine for workflow management on clouds. We also established a prediction model which was integrated with job queuing system to estimate the waiting time and cost of the individual tasks on different computing nodes, therefore helping users achieve maximum performance at lowest payment. This proposed effort has the potential to positively provide an efficient, resilience and elastic environment for cloud computing platform. This development also helps boost user productivity by promoting a flexible workflow interface that lets users design and control their tasks' flow from anywhere.

***Keywords*—**Web-based, workflow, HTML5, Cloud Computing, Queuing System.

I. Introduction

CLOUD computing [1] is becoming more and more matured over the last few years and consequently the demands for better cloud services is increasing rapidly. As proposed by [2] and shared by many researchers and practitioners, compared with conventional computing paradigms, cloud computing can provide "a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet". It has to be a highly elastic environment which provides stable services to users. In which services are delivered according to external users' needs via internet. Users can have access from a set of web interfaces that manage and monitor a pool of computing resources at any time and at any place in different economic, commercial and scientific fields and obviate or decrease costs related to storage and computation in their sites.

The numbers of cloud computing services are growing very fast; which also increase the cloud users. With this increase there is a need to manage all those cloud computing tasks from cloud users. One of the research topics to improve cloud services is the flexible allocation and automation of running and management for cloud computing tasks by adapting workflow system concept [3], [4]. Workflow systems are designed to support the process automation of large scale sequential or parallel applications, in whole or in part, during which applications are passed from one participant to another for action, according to a set of procedural rules. A workflow models a process as consisting of a series of steps that simplify the complexity of execution and management of applications [5]. On the other hand, task scheduling is an important part when to manage and control the computing tasks in a computing platform. Kamer [6] presented an approach to scheduling sequential tasks in a distributed system in which the only relationship between tasks is the need to access to common files and there is no dependence between tasks. So the workflow scheduling can be defined as the automation in workload scheduling where the workload is the requests and tasks generated by number of user s or clients in cloud.

The goal of this paper is to run a user-defined service workflow. We aim at the development of workflow management system for cloud computing platforms based on our previous research on the dynamic allocation of the cloud computing resources and its workflow process. We developed an execution engine for workflow management which allows an entire computation is partitioned by user and distributed over several computing nodes with a result of being able to process many computing tasks. We implemented an abstraction and designed mechanisms for workload scheduling in which independent jobs gets scheduled among various available processors of distributed computing for optimization. So all information is known in advance by the workflow engine and tasks are allocated according to the prior knowledge and will not be affected by the state of the system.

In difference to other workflow implementations, our system also targets on a unified interface. We implemented a sketch of clientless web-based interface that seeks to enable manageability among tasks, which is efficient, resilience and independent of the operating system. This workflow interface can be accessed from any OS platform through any HTML5 compliant browser like Internet Explorer, Mozilla Firefox, Google Chrome, Safari, Opera, etc. So our implementation can make such a service is simple and easy to use and allow users to access the workflow management from anywhere, any devices and without requiring the installation of special clients.

The rest of this paper is organized as follows. Section II lists the related works. Section III gives descriptions of architecture and platform. Section IV gives some details of the development. Section V discusses future work and concludes.

This work is supported by National Science Council, R.O.C., under the contract number of MOST-103-2221-E-492-027

S.T. Wang, Y.C. Lin and H.Y. Chang are with the National Center for High-Performance Computing, Taiwan, R.O.C. (e-mail: stwang@nchc.org.tw, 1203043@nchc.org.tw, jerry@nchc.org.tw).

II. Related Works

Currently, many research communities have realized the concept of task workflow management. Scheduling the tasks at the back-end servers is very difficult, because the compute jobs requesting is very large in number which required different resources to execute. The workflow has to optimal and good enough so that each request by the user gets response in time. There are various task workflow system developed to solve this problem. Following three implementations provide the good framework for scientific workflow.

A. Cloudbus Toolkit

Cloudbus Toolkit [7] is a well-known project that defines a complete architecture for creating market-oriented Clouds. A workflow engine is also mentioned in the designed architecture. This project realized the requirement and changes needed to be incorporated when moving scientific workflows to Clouds. It aims at a prototypical implementation of a workflow engine that executes a workflow composed of different Cloud services. The three key components of this architecture are a Cloud Broker, a Market Maker and an InterCloud. The Cloud Broker schedules applications on behalf of the user by specifying the desired quality of service requirements, whereas the Market-Maker acts as a mediator bringing together Cloud providers and customers. It aggregates infrastructure demands from the Cloud Broker and matches them against the available resources published by the Cloud providers. The Inter Cloud provides a scalable federated computing environment composed of heterogeneous interconnected Clouds enabling the Inter Cloud resource sharing. The goal of Cloudbus Toolkit is to simply provide a new functionality rather than to investigate a comprehensive solution. The architectural framework of this project is still under development.

B. CloudAnalyst

CloudAnalyst [8] is a graphical simulation tool built on top of the CloudSim Toolkit [9], developed at the University of Melbourne whose goal is to model and analyze the behavior of large social network applications according to geographic distribution of users and data centers. In this tool, communities of users and data centers supporting the social networks are characterized and, based on their location; parameters such as user experience while using the social network application and load on the data center are obtained. The Internet traffic routing between the user bases located in different geographic locations and the datacenters, is controlled in CloudAnalyst by a service broker that decides which datacenter should serve the requests from each user base based on different routing policies. The current version of CloudAnalyst implements three different routing policies, which are network-latency-based routing, response-time-based routing and dynamic-load-based routing. A CloudAnalyst simulation case study of the social network application proved how load balancing managed by a service broker optimizes the performance and cost of large scale Cloud applications.

C. Swin De W-C

SwinDeW-C (Swinburne Decentralized Workflow for Cloud) [10], is a peer to peer based Cloud workflow system for managing large scale workflow applications. SwinDeW-C is not built from the scratch but based on our existing SwinDeW-G. The general cloud architecture of SwinDeW-C includes four basic layers from top to bottom: application layer (user applications), platform layer (middleware cloud services to facilitate the development/deployment of user applications), unified resource layer (abstracted/encapsulated resources by virtualization) and fabric layer (physical hardware resources). In order to support large scale workflow applications, a novel Swin DeW-C architecture is also presented where the original fabric layer of SwinDeW-G is inherited with the extension of external commercial cloud service providers. Meanwhile, significant modifications are made and some functional components are enhanced at the platform layer to support the management of large scale workflow applications; the user interface is modified to support Web browser based access.

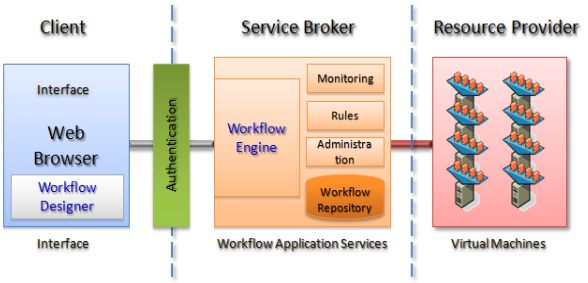

Fig. 1 System architecture

III. Architecture

A. System Architecture

Fig. 1 provides an overview of the logical architecture for a standard, three tiers implementation. The framework architecture is composed of three main parts: the Client, the Service Broker and Resource Provider. The internal components of every architecture part and their provided functionalities are discussed in the following:

1) Client: the Client provides Cloud users with an interactive user interface to submit their service requests to the broker by describing the functional computing job requirements. While our web-based interface is built using a Model-View-Controller based Google Web Toolkit [11] framework. The user is able to manage and monitor the service after its deployment through a single management console. If the requested service requires the involvement of other services, the workflow engine could be deployed to assist users in building and executing complex Cloud services.
2) Service Broker: the Service Broker builds the heart part of our implemented architecture. It includes a Workflow Engine. The engine is a software service that provides the run-time environment in order to create, manage and execute workflow instances. The engine handles the

representation of a workflow process in a form which supports automated manipulation, and interfaces to support interoperability between different workflow systems. This broker also provides the system monitoring and metric functions to facilitate the management of composite workflow application environments. Its main task is to find the most suitable computing resource provider while satisfying the users' service requirements.

3) Resource Provider: Provider is a pool of computing resources including machines, network, storage, operating system, application programs and development environments. When granted workflow requests, a part of the computing resources in the pool is dedicated to the requesting user until those resources are released. For providing computing resource, we employed virtualization technology. Virtualization [12] acts as a central component that can achieve the purpose of cloud platforms and services, and it is a promising approach to consolidating multiple services onto a smaller number of computing resources. A virtualized server environment allows computing resources to be shared among multiple performance-isolated platforms called virtual machines [13]. A virtual machine is a software implementation of a machine that executes related programs like a physical machine. Each virtual machine includes its own system kernel, OS, supporting libraries and applications. Users are allowed to customize their process workflow to support the dynamic resource allocation.

B. Software Architecture

Fig. 2 illustrates the software stack of our workflow management system. Our system is entirely web-based that way the end user doesn't need to download and install any tools or plugins on his/her computer. In particular, this enables accessing our interface from a wide range of devices, including mobile devices such as Smartphone or Pad. The key components are as follows:

1) Hardware: There are many physical devices including CPU, memory, hard disk, NIC (Network Interface Card), etc.
2) Hypervisor: We adopt KVM [14] to attain virtualization aim. KVM consists of a loadable kernel module that provides the core virtualization infrastructure and a processor specific module. KVM also requires a modified QEMU although work is underway to get the required changes upstream. Using KVM, we can run multiple virtual machines running unmodified Linux or Windows images.
3) Libvirt APIs: Libvirt [15] is an open source API, daemon and management tool for managing platform virtualization. It can work with a variety of hypervisors in the development of a cloud based solution. Thus, we employ these APIs to control and manage our KVM, and we can switch the underlying hypervisor technology at a later stage with minimal efforts.
4) PHP: We adopt the PHP program language to build all web pages. PHP is an open source server-side scripting language designed for Web development to produce.
5) MySQL: MySQL is the world's most used open source relational database management system (RDBMS) that run as a server providing multi-user access to a number of databases.
6) Google Web Toolkit: Our web based interface is built using a Model-View-Controller (MVC) based Google Web Toolkit (GWT) framework. GWT is a development toolkit for building and optimizing complex browser-based applications. The GWT SDK provides a set of core Java APIs and Widgets. These allow us to write AJAX applications in Java and then compile the source to highly optimized JavaScript that runs across all browsers.
7) jsPlumb: jsPlumb is a JavaScript library that provides a mean for a developer to visually connect elements on their web pages. It provides a way to connect elements of a UI into a nice graph. It uses a canvas stage to drag graphs, charts or different elements showing the way GUI elements or other objects interact with each other.

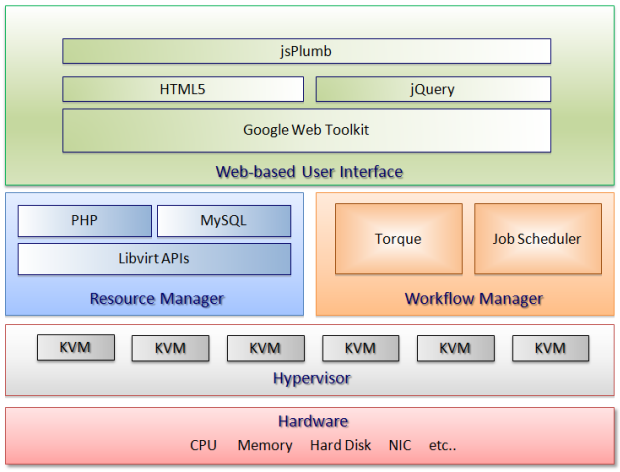

Fig. 2 Software architecture

TABLE I
FORMOSA 3 CLOUD CLUSTER SPECIFICATION

CPU	Intel Xeon x5660 six cores 2.8GHz
Hard Disk	80GB SSD
Memory	48GB DDR3 Registered ECC SDRAM
Network	4x QDR 40Gb Infiniband and Gigabit Ethernet
Operating System	CentOS 6.3
Hypervisor	Kernel-based Virtual Machine

C. Cloud Platform

Table I shows the specification information of our cloud platform named Formosa 3. Formosa 3 [16] is a 64bits high-performance Beowulf cluster located within Southern Business Unit of the National Center for High Performance Computing (NCHC) [17]. It consists of 76 IBM X3550M3 servers as its compute nodes. This self-made cluster was designed and constructed by the 'HPC Cluster Group' at NCHC for cloud service and came online in 2012. Each node has two Six-Core Intel Xeon x5660 2.8GHz processors and 48GB of DDR3 registered ECC SDRAM. All nodes were connected on

the InfiniBand high speed network and a private subnet with 1000 Mbits/s Gigabit Ethernet. An additional 4 nodes are used as front ends to interface with cluster, and 4 nodes as storage for the user file systems by Parallel File System.

IV. Development

We implemented the proposed workflow architecture as a Web-based interface, written in Java. Currently we adopt GWT to allow the use of existing Java knowledge and tools to build high performance, desktop web applications. GWT abstracts away many complexities of web application development by not requiring us to learn Javascript and HTML. It rests on today's web standards: AJAX, JSON and HTML5 as well. While implementing our cloud platform for integrating with workflow system, we came across several issues that have previously not been addressed. For example: how to build the cloud virtualization environment? Currently we adopt the Kernel-based Virtual Machine. By using KVM, we can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware devices. A virtualized server environment allows computing resources to be used by workflow system among multiple performance-isolated platforms. This section aims to explain the details of development involved with Workflow Designer, Translator, Workflow Engine and Session Controller.

A. Workflow Designer

The Workflow Designer provides intuitive user-friendly interface for users to design workflows as well as specify workflow parameters. It is a web-based GUI allowing users to compose, edit and submit workflows. As shown in Fig. 3, it consists of a working pool and component palette. The working pool is a HTML5 canvas, users can drag and drop components and input data parameters onto it and connect them to the workflow. User can press the setting button to launch a specific dialog to define settings, included: among the available resource providers, the number of computing nodes, which instance type for each computing node, estimated running time of the workflow, etc. Once resources are chosen, the user presses the 'Submit' button which sends a request to the Workflow Engine to run the workflow.

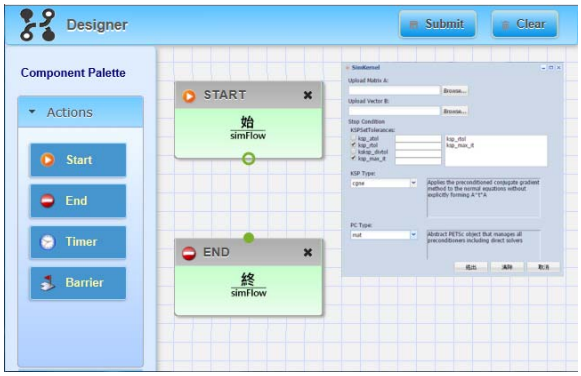

Fig. 3 Workflow Designer

B. Translator

Once workflow is composed and submitted, the workflow request will be sent to the Translator. The Translator module is responsible for producing executable representations of workflows from the specifications written in our XML-based format. These specifications are stored in the Workflow Repository. Fig. 4 illustrates example of workflow XML representation.

```
<workflow>
  <status>
    <state value="Active">
      <queue>n4p16</queue>
      <queue>n8p32</queue>
      <queue>n16p64</queue>
    </state>
    <state value="Closed">
      <queue>n128</queue>
    </state>
  </status>
  <flow>
    <unit>
      <job serial="1" queue="n4p16" speed="T1">
      <name>simKernel</name>
      <path>/home/ct01/oldpast/kely.math</path>
      <time>5</time>
      <parent>NULL</parent>
      <child>VASP</child>
    </unit>
    <unit>
      <job serial="2" queue="n4p16" speed="A2">
      <name>VASP</name>
      <path>/home/ct01/vasp/vasp_x86_proversion</path>
      <time>5</time>
      <parent>NULL</parent>
      <child>VASP</child>
    </unit>
  </flow>
</workflow>
```

Fig. 4 Workflow XML representation

C. Workflow Engine

The Workflow Engine is the heart of a workflow system and responsible for creating and executing workflow tasks accounting to the executable representations. It acts as a bridge between the Client and the backend Resource Provider. It supports the following functionalities: workflow xml compilation, workflow scheduling, resource acquisition, process scheme generation, serialization and saving of parameters, pre-execution of process. The workflow process scheme include: verification of the current status, determine the authority of users executing condition script. Fig. 5 shows the monitoring page which generated by Workflow Engine. The Workflow Engine keeps track of the statuses of individual tasks such as initialized, executing, finished, error.

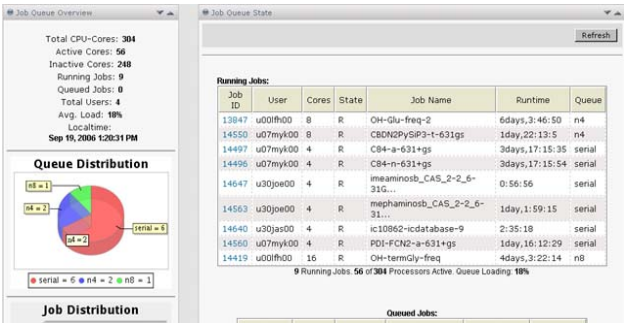

Fig. 5 Workflow monitoring page

D.Session Controller

For controlling the user's session, our workflow system has a session server to perform the role of workflow controller, and to manage user's connection and accounting information. The session controller also communicates with service broker to deliver the status. Besides, the session controller is deployed on the GWT to provide the single web-based portal for user login. It also enhances the system security and elasticity by applying the centralized control of applications and files. While implementing our cloud platform for workflow, we came across several issues that have previously not been addressed. For example, for accessing the computing resource, due to the virtual machine may execute on different physical machines every time. This can be troublesome if we provide a fixed public IP address and port for connecting to the client's desktop. So, we use iptables and thus setup port forwarding connections to the virtual machine that user launched in the workflow. Our system will allocate a mapping port dynamically which allows users' workflow can connect to the backend physical machine with the dedicated IP address of and the port which will be forwarded to the appropriate physical machine which is currently hosting the user's virtual machine.

Fig. 6 illustrates the web-based workflow interface. Users can drag/drop components, input data parameters, set the timer conditions, specify the actions, etc. And then connect them to build the relation of workflow.

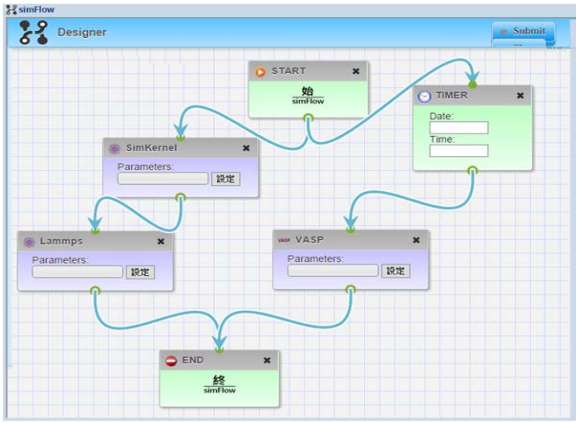

Fig. 6 Web-based workflow interface

V. Conclusion

In this paper, we aim at the development of workflow management system for cloud computing platforms based on our previous research on the dynamic allocation of the cloud computing resources and its workflow process. We developed an execution engine for workflow management which allows an entire computation is partitioned by user and distributed over several computing nodes with a result of being able to process many computing tasks. We implemented an abstraction and designed mechanisms for workload scheduling in which independent jobs gets scheduled among various available processors of distributed computing for optimization. We also developed a sketch of clientless web-based interface that seeks to enable manageability among tasks from workflow, which is efficient, resilience and independent of the operating system.

In the future, we will adopt the new task scheduling and data placement for execution of applications including a sequence of tasks in distributed platforms. We plan to design an approach to mapping workflow's data and tasks between different cloud platforms for decreasing the network traffic and improvement in quality of providing service and reliability.

References

[1] Fox, Armando, et al. "Above the clouds: A Berkeley view of cloud computing." Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28 (2009): 13.
[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu. "Cloud Computing and Grid Computing 360-Degree Compared," IEEE Grid Computing Environments Workshop, pp. 1-10, 2008.
[3] A. Tsalgatidou, et al., "Developing scientific workflows from heterogeneous services," SIGMOD Record, vol. 35, no. 2, pp. 22-28, 2006.
[4] C. Hoffa, et al., "On the use of cloud computing for scientific workflows," eScience, pp. 640-645, 2008.
[5] Buyya, Rajkumar, James Broberg, and Andrzej M. Goscinski, eds. Cloud computing: Principles and paradigms. Vol. 87. John Wiley & Sons, 2010.
[6] Kaya, Kamer, and Cevdet Aykanat. "Iterative-improvement-based heuristics for adaptive scheduling of tasks sharing files on heterogeneous master-slave environments." Parallel and Distributed Systems, IEEE Transactions on 17.8 (2006): 883-896.
[7] Buyya, Rajkumar, Suraj Pandey, and Christian Vecchiola. "Cloudbus toolkit for market-oriented cloud computing." Cloud Computing. Springer Berlin Heidelberg, 2009. 24-44.
[8] Wickremasinghe, Bhathiya, Rodrigo N. Calheiros, and Rajkumar Buyya. "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications." Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE, 2010.
[9] Calheiros, Rodrigo N., et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and Experience 41.1 (2011): 23-50.
[10] Liu, Xiao, et al. "SwinDeW-C: a peer-to-peer based cloud workflow system." Handbook of Cloud Computing. Springer US, 2010. 309-332.
[11] P. Chaganti , "Google Web Toolkit: GWT Java AJAX Programming," Packt Publishing, 2007.
[12] Goth, Greg. "Virtualization: Old technology offers huge new potential." IEEE Distributed Systems Online 8.2 (2007): 3-3.
[13] R. A. Meyer and L. H. Seawright, "A Virtual Machine Time-Sharing System," IBM Systems Journal, vol. 9, no. 3, 1970.
[14] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. "kvm: the Linux Virtual Machine Monitor," In Proceedings of the Linux Symposium, vol. 1, pp. 225-230, 2007.
[15] Libvirt - The virtualization API. http://libvirt.org/
[16] NCHC Formosa 3 Cloud Cluster. http://formosa3.nchc.org.tw/
[17] NCHC, National Center for High-performance Computing. http://www.nchc.org.tw/