International Journal of Multimedia and Ubiquitous Engineering
Vol.8, No.5 (2013), pp.19-30
http://dx.doi.org/10.14257/ijmue.2013.8.5.03

# DIVE-C: Distributed-parallel Virtual Environment on Cloud Computing Platform

In-Yong Jung[1], Byong-John Han[1], Hanku Lee[2] and Chang-Sung Jeong[1]

*[1]Department of Electrical Engineering, Korea University, Seoul, Korea*
*[2]Division of Internet & Multimedia Engineering, Konkuk University, Seoul, Korea*

*[1]{dekarno, guru1013, csjeong}@korea.ac.kr, [2]hklee@konkuk.ac.kr*

***Abstract***

*In social media services and social network services, it is necessary to collect, analyze and process their big data with low maintenance cost. Therefore, distributed-parallel data processing on cloud platform is getting spotlight as useful solution for them. In this paper, we present a new architecture of DIVE-C: DIstributed-parallel Virtual Environment on Cloud computing platform for distributed parallel data processing applications which offers a transparent virtual computing environment in order to provide a way easy to launch user's distributed parallel applications. It hides the complexity of the cloud, and helps users to focus on their new applications and core services. DIVE-C uses agent-based resource management scheme to configure VM resources and application deployment for offering various distributed-parallel application models. VM resources are automatically provided by unified cloud management layer. Furthermore, an easy-to-use web interface of DIVE-C offers convenience to users. We implemented a prototype of DIVE-C, and its experiment results show the competitive performance of DIVE-C for dynamic resource and virtual computing environment provisioning for various data processing models.*

**Keywords:** *Cloud computing, Cloud platform, PaaS, Virtual computing environment, Distributed- parallel computing*

## 1. Introduction

Cloud computing as the main IT key words today is changing many traditional paradigms of IT industry. Cloud computing integrates on-demand IT technologies that supply adaptive resources, platforms and applications. Therefore, it is getting spotlight as useful solution for cost reduction.

Today, various social cloud services such as social media services, social network services (SNS) and content delivery systems are becoming key businesses in the cloud computing. They are deployed and launched on the cloud, so internet and cloud computing ecosystem are overflowed with big data generated by these social cloud services. They need more sophisticated way to reduce their cost to develop or maintain their applications which handle big data such as media contents, messages and user records. Therefore, it is necessary to do research for cloud based software technologies which provide scalable distributed parallel software execution services for processing big data on cloud.

Copyright ⓒ 2013 SERSC

Cloud platform technologies are suggesting an easier way for developing and launching their new IT services through abilities of supplying resources and services against dynamic demand of clients. Cloud platform for distributed parallel computing offers convenience and cost reduction of developing, testing, deploying, launching and maintaining new social media cloud applications.

In this paper, we present a new architecture of DIVE-C (DIstributed-parallel Virtual Environment on Cloud computing platform) for distributed parallel data processing applications. DIVE-C has three layered architecture, and offers a transparent virtual environment which provides a way easy to launch distributed parallel applications on various models through automated software environment creation and adaptive virtual resource provisioning. It hides the complexity of the cloud system, and helps users to focus on their new applications and core services.

This paper is organized as follows: Section 2 introduces the related works in cloud computing and distributed parallel application technologies. In Section 3, we describe about the features and detailed architecture of DIVE-C. In Section 4, we show the results of experiments for the performance evaluation of our system. Finally in Section 5, we summarize our work and conclude this paper.

## 2. Related Works

Cloud platform technologies (PaaS) offer convenience and cost reduction to users for developing, deploying, launching and maintaining their new application service. There are many commercially released or open-source cloud PaaS solutions. We can classify them into a number of groups as follows [1-3]:

1) Application Platform as a Service (APaaS): This group is focused on developing, deploying and lauching applications for the public web in automated way such as Google App Engine, Microsoft Azure Services Platform, WSO2 Stratos and Appscale.

2) Software Infrastructure as a Service (SIaaS): This group provides a number of useful software infrastructures which can be integrated with user applications such as Amazon Simple Queue Service, SimpleDB and Google Apps BigTable.

3) Distributed-parallel application platform: These solutions offer unique services such as spinning up a mapreduce platform or providing API library to launch distributed parallel applications. Amazon Elastic Mapreduce belongs to this group.

4) Other notable platforms: This type includes various platform services with unique functions such as offering testing and billing services such as SOASTA and Zuora.

Today, several use cases that create value through analyzing big data using mapreduce framework have been reported [4]. Apache Hadoop [5] is getting spotlight as a solution for big data analysis platform, because it offers distributed storage and mapreduce based massively distributed processing framework. Therefore, most of distributed parallel application platforms are focusing on hosting Hadoop on cloud. Amazon Elastic Mapreduce [6] and MS Windows Azure Platform [7] offer services for spinning up a Hadoop cluster on public cloud. There are a number of open projects to support automated deployment of mapreduce application on VMs such as Appscale [8] and Apache Wirr [9]. GAE [10] supports API library for mapreduce to handle big data on google’s cloud. The research for an effective infrastructure management to exploit the cloud infrastructure is other important issue. Cloud infrastructures interacting with cloud platforms should provide on-demand virtual servers and storages for applications. There are various solutions for building public, private and hybrid cloud such as Amazon EC2/S3 [11], Eucalyptus [12], and OpenStack [13]. These solutions can offer VM servers and extra block level storages, and support their networking, authorization and web service APIs for users. Cloud infrastructure manager (CIM) technology supports sophisticated way and easy-to-use interface to manage, control and integrate these infrastructures with other software systems. There are several solutions for CIM such as Rightscale [14], Kaavo [15] and KOALA [16] project which support automated software deployment, monitoring, and web-based unified interfaces for heterogeneous clouds.

## 3. Architecture of DIVE-C

In this section, we describe several features of DIVE-C, and then its architecture in detail. As shown in Figure 1, DIVE-C consists of three layers; Cloud PaaS Portal (CPP), Cloud Distributed-parallel Data Processing Platform (CDDP) and Cloud Infrastructure Management Platform (CIMP). It delivers a transparent virtual computing environment with on-demand processing model on VM resources to user. Since the complexity of using cloud is hidden, user can focus on their applications via DIVE-C.

### 3.1. Features

In the following, we describe a number of key features of our system.

1) Transparent virtual cloud computing environment: PaaS user needs to focus on the core functions and development of their new application [17, 18]. DIVE-C provides a transparent virtual computing environment by hiding the complexity via automated configuration of VM resources and software environment. Users can focus on development of their new cloud applications and core functions via DIVE-C.

2) Agent-based processing platform for supporting various distributed-parallel application models: DIVE-C offers an agent-based resource management scheme for configuring its VM resources to support various distributed-parallel processing models. Each VM resource on cloud infrastructure has its Resource Agent launched on booting stage automatically to interact with Resource Agent Controller in CDDP. DIVE-C can control multiple VM resources via these Resource Agent Controllers to customize and deliver optimized virtual software environment with on-demand distributed parallel processing model to user.

3) Automatic VM resource provisioning: DIVE-C offers adequate on-demand VM resources optimized for user application deployment on cloud infrastructure. Lifecycle of each VM resource is managed automatically, so users need not handle a specific node where their application should be deployed and executed.

4) Unified cloud infrastructure management: CIMP supports unified management through the abstraction layer for API which interacts with several cloud infrastructures. It means that user applications can be deployed on any cloud infrastructure. If a managed infrastructure cannot offer competitiveness or stability, it can be replaced with the other proper infrastructure easily.

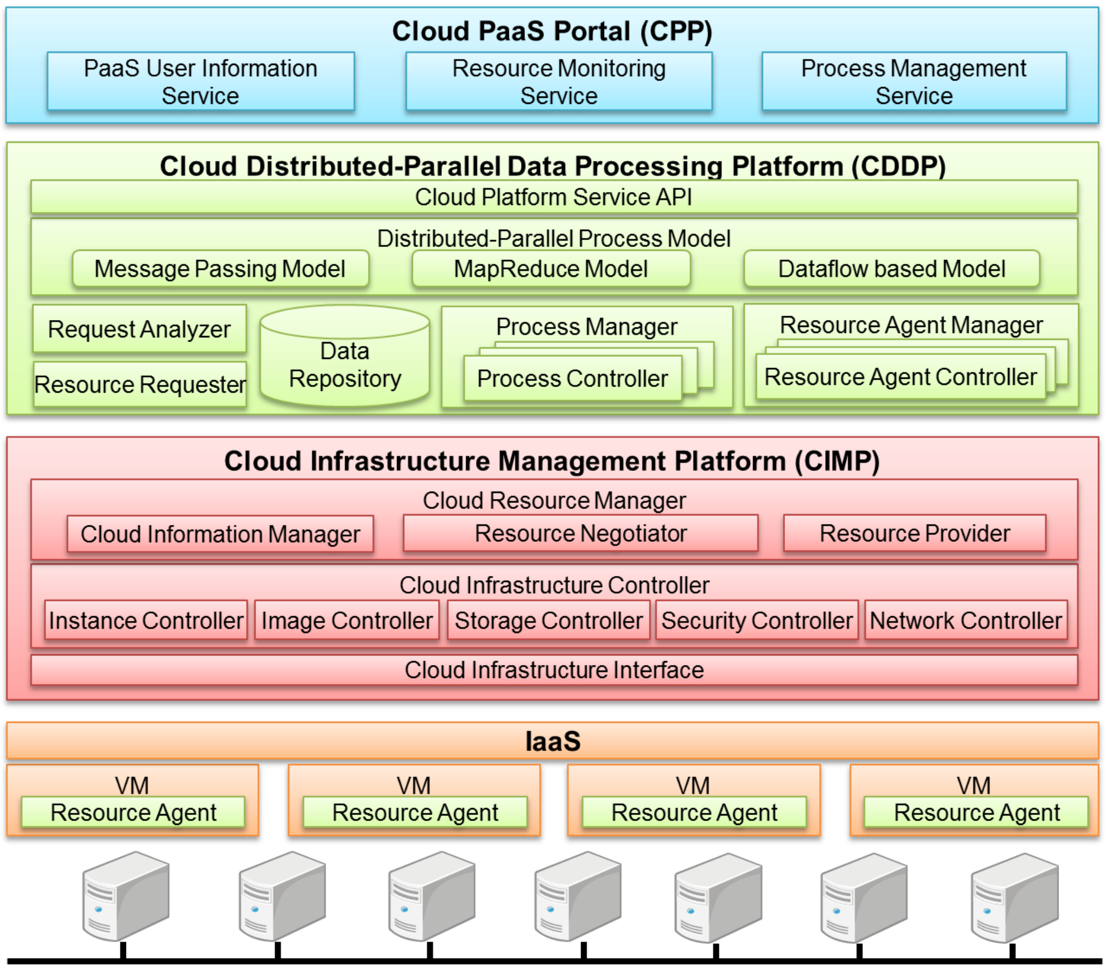

Cloud PaaS Portal (CPP)
PaaS User Information Service
Resource Monitoring Service
Process Management Service
Cloud Distributed-Parallel Data Processing Platform (CDDP)
Cloud Platform Service API
Distributed-Parallel Process Model
Message Passing Model
MapReduce Model
Dataflow based Model
Request Analyzer
Resource Requester
Data Repository
Process Manager
Process Controller
Resource Agent Manager
Resource Agent Controller
Cloud Infrastructure Management Platform (CIMP)
Cloud Resource Manager
Cloud Information Manager
Resource Negotiator
Resource Provider
Cloud Infrastructure Controller
Instance Controller
Image Controller
Storage Controller
Security Controller
Network Controller
Cloud Infrastructure Interface
IaaS
VM
Resource Agent

**Figure 1. Three-layered architecture of DIVE-C with cloud infrastructure**

5) Easy-to-use web interface: Cloud PaaS Portal provides automated system and convenient graphical interface to develop, deploy and launch their applications by offering an easy-to-use web interface for CDDP.

### 3.2. Cloud PaaS Portal (CPP)

CPP provides a web interface for platform users to support application deployment and execution management. It provides graphical interfaces to organize user requests for automated application deployment and launching. Users choose appropriate data processing model and requirements which describe the proper software environment for user application deployment via CPP. The information is filled by user, and documented as XML-like request form named SLD (Service Level Description). Then, user uploads his applications and input data via CPP to request the execution of their applications. Then, user data including SLD and applications are submitted to the CDDP. During the execution of applications, user can receive notifications about information such as error or job completion via CPP.

### 3.3. Cloud Distributed-parallel Data Processing Platform (CDDP)

This layer composes a software execution environment, and manages the execution of user application. It requests a new resource creation to CIMP in response to a user request specified by SLD received from the CPP, and configures an actual software environment for application.

CDDP supports a number of distributed parallel process models: Message passing model for grid applications, Mapreduce model based on Hadoop mapreduce and Dataflow based model such as abstraction of Hadoop framework or workflow processes which consist of various mapreduce applications.

The unit of each application execution is a process, and managed by the instance of Process Controller. Process Manager creates a new Process Controller instance when a user SLD is received. The Process Controller requests the creation of new resources through Resource Requester, based on SLD information analyzed by Request Analyzer. Submitted user data and applications are stored in Data Repository. After resource creation, Process Controller manages Resource Agent Controllers each of which is created and connected to a Resource Agent of each VM.

Our resource control scheme is influenced by agent-based middleware architectures [1, 19]. Resource Agent is a demon process which offers a connection between CDDP and each VM instance. A new VM instance containing a Resource Agent is created, and each Resource Agent is connected to the Resource Agent Controller created by Resource Agent Manager one by one. Pre-installed Resource Agent is automatically started after booting up of each VM instance, and tries to access Resource Agent Manager to build a new connection with its Resource Agent Controller instance. Then, Process Controller can start server customizing. Process Controller lists up controllable Resource Agent Controllers for its process. Then, it determines a role of each VM resources, and configures application environment for each of them via the connection between Resource Agent Controller and Resource Agent. Process Controller locates

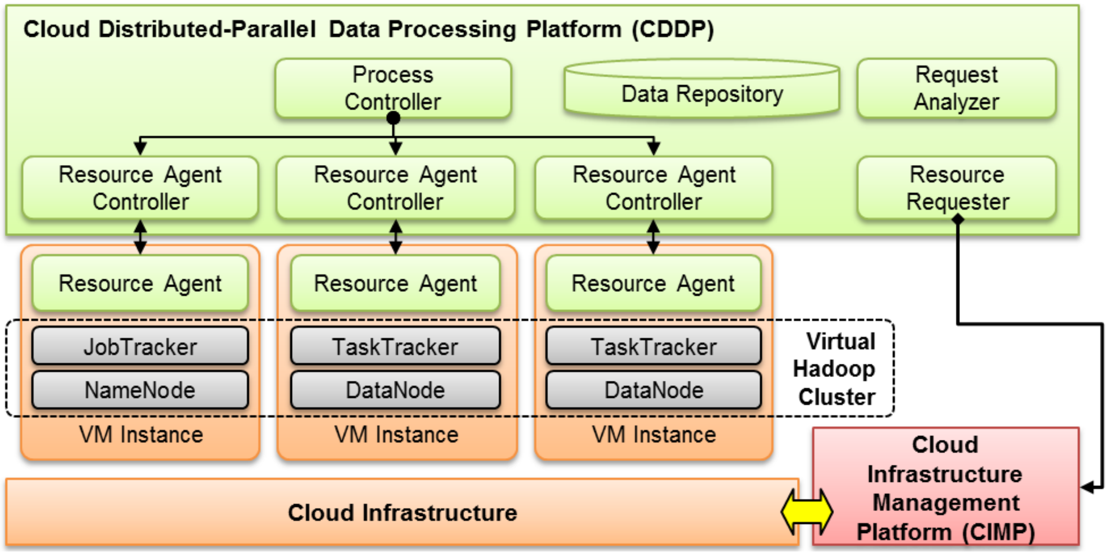


**Figure 2. Interactions between Process Controllers, Resource Agent Controllers and Resource Agents on each VM for provisioning of virtual Hadoop cluster**

user data and applications from Data Repository, and sends them to the VM instances. Each Resource Agent on VM instance receives them from the CDDP, stores them onto its VM instance, and sets up a proper software environment for the execution of the applications. After all VM instances are configured, Process Controller sends commands to each Resource Agent for running user applications and collecting their results. The utilization of VM resource and running status of application are monitored by Resource Agent. All of these procedures are hidden, so users need not know about its details. Figure 2 shows the interactions between Process Controllers, Resource Agent Controllers and Resource Agents on each VM for provisioning of virtualized Hadoop cluster with 4 VMs (1 Master and 3 Slaves) as an example.

### 3.4. Cloud Infrastructure Management Platform (CIMP)

This layer offers an abstraction of cloud infrastructure interface. It consists of three main components: Cloud Manager, Cloud Infrastructure Controller and Cloud Infrastructure Interface as shown in Figure 1.

Resource Negotiator of Cloud Resource Manager receives the detailed VM resource requirement from Resource Requester, and finds an optimized fabric of VM resources based on pre-defined VM types available in the cloud infrastructure. VM types of cloud infrastructure comprise the attributes of VM including the number of processor, memory size and disk size. Each combination of these attributes determines whether the VM is compute-intensive (multi-core VCPU) or storage-intensive (large ephemeral disk), high-performing or economical. Then, Resource Provider launches the creation of a bundle of VM instances by calling a number of internal APIs supported by Cloud Infrastructure Controller. After the creation of VM instances, each VM is controlled by its Resource Agent. After all of these steps, a list of the VM resources created for a use request is returned to CDDP.

Cloud Infrastructure Controller offers an abstracted client-side API library of cloud infrastructure web services API. Various cloud management operations including creation and termination of VMs are executed through this layer. Cloud Information Manager gathers, and monitors the information of cloud infrastructure such as the status of cloud/VMs and utilization factor of cloud infrastructure.

### 3.5. Implementation

We implemented a prototype of DIVE-C using java 6. Components such as Process Controller and Resource Agent Controller are implemented as java threads. Cloud Infrastructure Controller of CIMP supports an abstracted java library of Amazon EC2 API [20] compatible with various cloud infrastructures such as Eucalyptus and OpenStack.

## 4. Experiments

In this section, we shall show the results of experiments for the performance evaluation of our system.

### 4.1. Experimental Setup

Our DIVE-C prototype is hosted on a server with 3.33GHz 4 core processor and 4GB memory. The server is connected to a cloud infrastructure test bed built on Ubuntu Enterprise Cloud [21] based on Eucalyptus 1.6.2. Our cloud infrastructure test bed consists of 1 front-end node (Cloud controller, Walrus, Cluster controller and Storage controller) and 4 worker nodes (Node controller and KVM hypervisor). Each node has 2 Xeon E5606 2.13GHz processors and 24GB memory, and is connected to each other in Ethernet. Resource Agent is pre-installed in the customized VM bundle image and started automatically after booting of VM instance. The custom VM image is based on Ubuntu 10.04 LTS including java 6 and cloud-init tool to collect meta-data of each VM instance from cloud infrastructure.

## 4.2. Functionality Tests

We performed two kinds of experiments for the evaluation of our system. The first experiment is on measuring provisioning time for VM resources and software environment. Figure 3 shows the average time for provisioning virtual computing environments including (a) creation time of VMs, (b) customizing time of VMs, (c) startup delay for fully distributed Hadoop 0.20.2 cluster as an example scenario and (d)

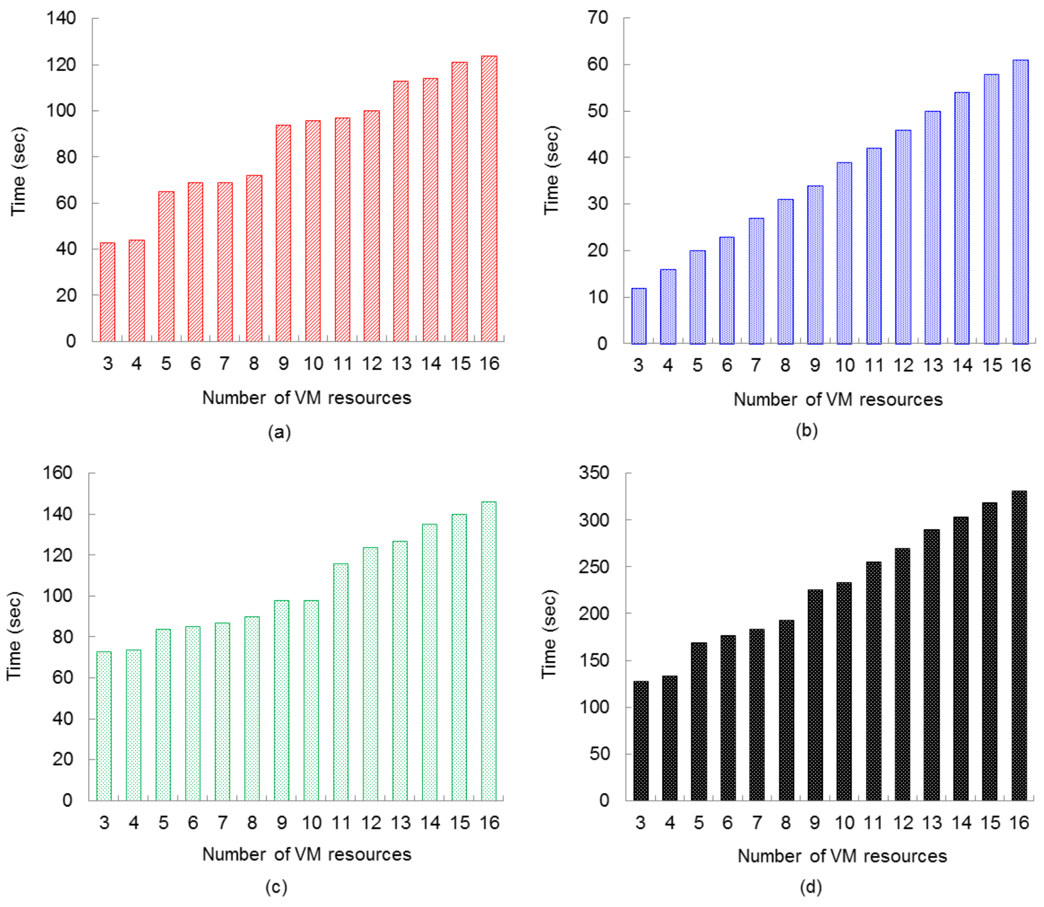


**Figure 3. Resource provisioning time: (a) VM creation, (b) VM customization, (c) Hadoop startup delay, (d) total provisioning time. (d) = (a) + (b) + (c)**

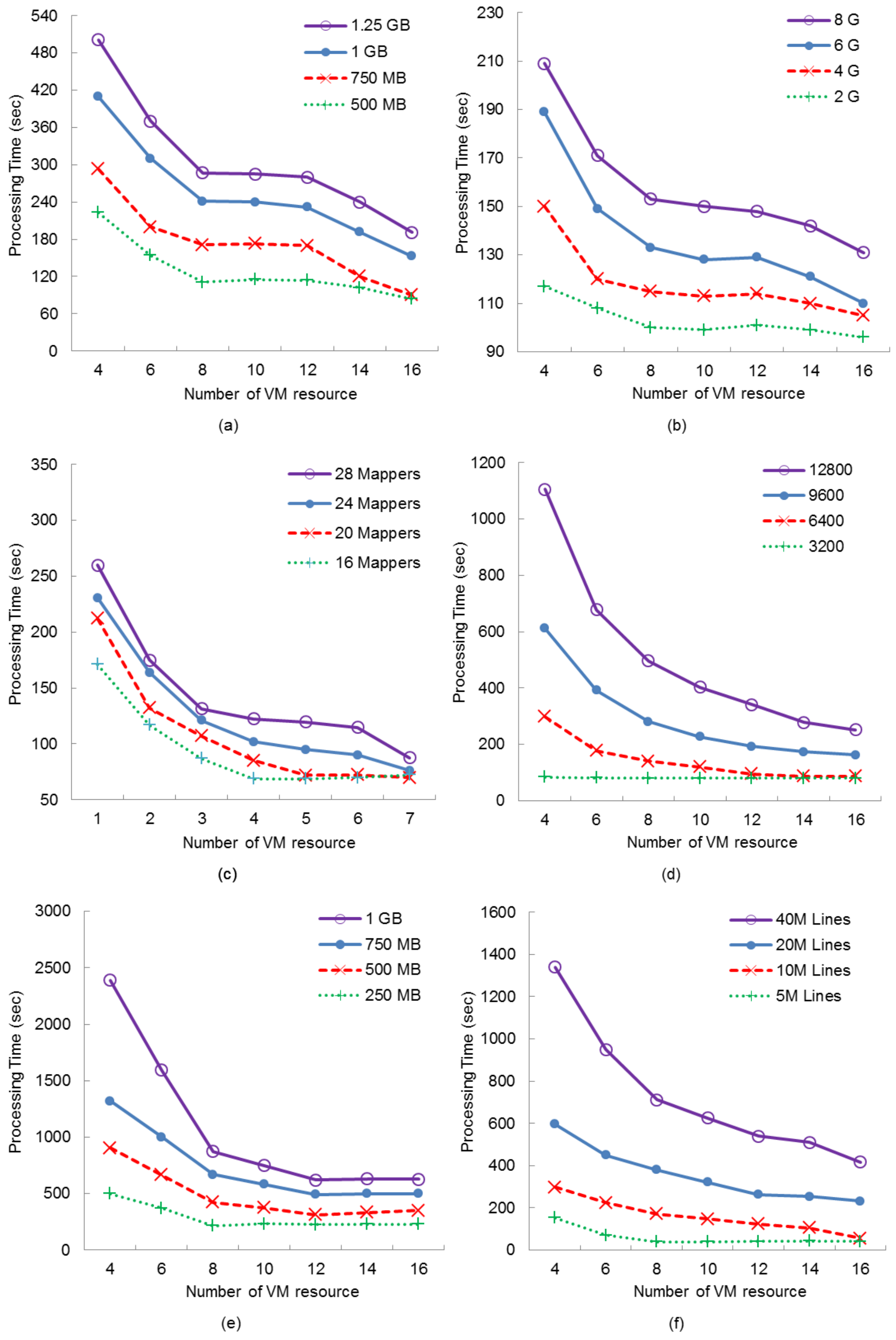


**Figure 4. Performance comparison of virtual clusters on UEC testbed with various distributed-parallel application models: (a) Terasort, (b) Distributed grep, (c) Pi estimator, (d) MPI mandelbrot, (e) Wordcount, (f) Apache log parser**

Copyright ⓒ 2013 SERSC

total provisioning time ((a)+(b)+(c)). Tested VM instance type has 2 core virtual CPU and 2GB memory. According to Figure 3(a), creation time is related to the number of worker nodes. Since our test bed has 4 worker nodes and resource scheduling policy is round-robin, the creation time is almost identical for each multiple of 4 VMs. It is significantly increased when the number of requested VMs is just over each multiple of four. If the cloud infrastructure supports more worker nodes, the increase of creation time can be slow down. Figure 3(b) and (c) show that as the number of VM resources increases, customizing time and Hadoop startup delay for VMs are linearly increased respectively. Because the startup delay composes a great proportion, the total provisioning time is actually linear as shown in Figure 3(d). The second experiment is about measuring performance of benchmarking applications with various processing models on virtual cluster of our test bed: mapreduce, MPI and dataflow. Figure 4(a), (b) and (c) show the average processing time with regard to various input size for (a) Terasort [22] (b) Distributed grep and (c) Pi estimator as benchmark workloads for mapreduce applications. They are launched on the virtual Hadoop 0.20.2 clusters built on VM instances. Input data for Terasort and Distributed grep is generated by using Teragen [22], and Pi estimator is launched with 1 billion samples. Figure 4(d) shows the average processing time for various size of mandelbrot set with 10000 iterations as MPI application. The performance of dataflow model applications is shown in Figure 4 (e) and (f). Figure 4(e) shows the average processing time of Wordcount written in Pig latin, the SQL-like distributed-parallel script language for Hadoop environment. Figure 4(f) shows the average processing time of Apache log parser using Cascading library. Cascading is the useful abstraction API set of basic Hadoop mapreduce operations and datamodels to build an application by connecting each multiple mapreduce stages. Tested VM instance type is identical to that of the first experiment above. Our results show the typical improvement of performance via workload distribution. The execution time decreases when the number of VM resources increases.

## 5. Conclusion and Future Works

We have presented the cloud platform architecture named DIVE-C for distributed-parallel data processing and infrastructure management, and shown how it can be used to provide a virtual computing environment on cloud and process data. Our system has three layers and offers a transparent virtual environment easy to launch distributed-parallel applications on various models. On-demand virtual computing environment for user's application is supported by agent-based resource management layer, and launched on the VM resources through the unified cloud infrastructure management layer. Furthermore, DIVE-C offers an easy-to-use web interface to support convenience for users. It hides these features and the complexity of the cloud system, and helps users to focus on their new applications and core services. Experiment results show the competitive performance of DIVE-C for virtual computing environment provisioning. We are planning to add more key features to achieve more valuable and fault-tolerant system design, and advanced performance evaluation with practical distributed parallel applications on heterogeneous clouds.

## Acknowledgements

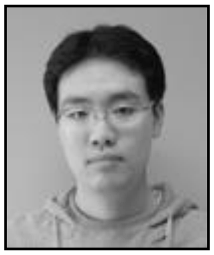
This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2013-H0301-13-3006), Next-

Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0006425), and Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program (R2012030096).

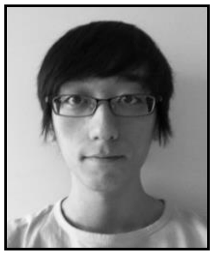
## References

[1] I. Y. Jung, D. K. Lee, B. J. Han, K. H. Kim and C. S. Jeong, Proceedings of the 3rd International Conference on Internet, **(2011)** December 15-18; Sepang, Malaysia.
[2] G. Raines and L. Pizette, “Platform as a Service: A 2010 Marketplace Analysis”, MITRE **(2010)**.
[3] Y. V. Natis, B. J. Lheureux, M. Pezzini, D. W. Cearley, E. Knipp and D. C. Plummer, “PaaS Road Map: A Continent Emerging”, Technical Report: G00209751, Gartner Inc., **(2011)**.
[4] D. Borthakur, K. Muthukkaruppan, K. Ranganathan, S. Rash, J. S. Sarma, N. Spiegelberg, D. Molkov, R. Schmidt, J. Gray, H. Kuang, A. Menon and A. Aiyer, Proceedings of the 2011 international conference on Management of data, **(2011)** June 12-16; Athens, Greece.
[5] Apache Hadoop, http://hadoop.apache.org.
[6] Amazon Elastic Mapreduce, http://aws.amazon.com/elasticmapreduce.
[7] Microsoft Windows Azure, http://www.windowsazure.com.
[8] N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman and R. Wolski, First International Conference on Cloud Computing, **(2009)** October 19-21; Munich, Germany.
[9] Apache Whirr, http://whirr.apache.org.
[10] Google App Engine, https://appengine.google.com.
[11] Amazon Web Services, http://aws.amazon.com.
[12] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff and D. Zagorodnov, Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, **(2009)** May 18-21; Shanghai, China.
[13] OpenStack, http://www.openstack.org.
[14] RightScale, http://www.rightscale.com.
[15] Kaavo, http://www.kaavo.com/.
[16] C. Baun, M. Kunze and V. Mauch, 2011 IEEE 4th International Conference on Cloud Computing, **(2011)** July 4-9; Washington D.C., USA.
[17] M. Boniface, B. Nasser, J. Papay, S. C. Pillips, A. Servin, X. Yang, Z. Zlatev, S. V. Gogouvitis, G. Katsaros, K. Konstateli, G. Kousiouris, A. Menychtas and D. Kyriazis, Fifth International Conference on Internet and Web Applications and Services, **(2010)** May 9-15; Barcelona, Spain.
[18] S. Kächele, J. Domaschka and F. J. Hauck, Proceedings of the First International Workshop on Cloud Computing Platforms, **(2011)** April 10; Salzburg, Austria.
[19] H. L. Kim, B. J. Han, I. Y. Jung and C. S. Jeong, KSII. T. INTERNET. INF., vol. 4, **(2010)**, pp. 1080.
[20] Amazon EC2 API Tools, http://aws.amazon.com/developertools/Amazon-EC2/351.
[21] S. Wardley, E. Goyer and N. Barcet, “Ubuntu Enterprise Cloud Architecture”, Technical White Paper, Canonical **(2009)**.
[22] O. O’Malley and A. C. Murthy, “Winning a 60 Second Dash with a Yellow Elephant”, Technical Report, Yahoo! **(2009)**.

## Authors



**In-Yong Jung** received a B.S. degree in Electrical Engineering from Korea University, Seoul, South Korea, in 2008. He is currently working toward the Ph.D. degree in Electronic and Computer Engineering at the Korea University. His research interests include distributed and parallel computing, cloud computing and GPGPU.



**Byong-John Han** received a B.S degree in Electrical Engineering from Korea University, Seoul, South Korea, in 2007. He has involved in some projects such as Intellectual Unmanned Vehicle and Semantic Information System in Grid middleware system. He is in the doctoral course in Electrical Engineering from Korea University, Seoul, South Korea. His current research is Intellectual management in Grid middleware system.

**Hanku Lee** is the director of the Social Media Cloud Computing Research Center and an associate professor of the division of Internet and Multimedia Engineering at Konkuk University, Seoul, Korea. He received his Ph.D. degree in computer science at the Florida State University, USA. His recent research interests are in cloud computing, distributed real-time systems, distributed and compilers.

**Chang-Sung Jeong** is a professor at the Department of Electrical Engineering at Korea University. Before joining Korea University, he was a professor at POSTECH during 1982-1992. He was on editorial board for Journal of Parallel Algorithms and Application in 1992-2002. Also, he has been working as a chairman of Computer Chapter at Seoul Section of IEEE region 10. His research interests include distributed concurrent computing, grid computing, and collaborative ubiquitous computing.

Copyright ⓒ 2013 SERSC