A Framework for Providing a Hybrid Fault Tolerance in Cloud Computing

Mohammed Amoon[1, 2]
[1]Dept. of Computer Science, RCC, King Saud University, P. O. Box: 28095-11437 Riyadh-Saudi Arabia
[2]Computer Science and Eng. Dept., Faculty of Electronic Eng., Menofia University, Egypt
mamoon@ksu.edu.sa, m_amoon74@yahoo.com

***Abstract*—Fault tolerance is a major challenge that should be considered to ensure good performance of cloud computing systems. In this paper, the problem of tolerating faults in cloud computing systems is addressed so that failures can be avoided in the presence of faults and the monetary profit of the cloud is maintained. A framework is proposed in order to achieve reliable platform of cloud applications. An algorithm for selecting the most suitable fault tolerance technique is presented. Another algorithm for selecting the most reliable virtual machines for performing customers' requests is presented.**

Keywords—Fault tolerance; Cloud Computing; Replication; Check pointing

I. Introduction

Cloud computing enables great changes to take place in today's IT with the global popularity of hybrid computing environments [1]. There are many public cloud platforms such as Amazon EC2/S3, Microsoft Azure, Google App Engine and IBM SmartCloud, while a large number of enterprises and institutions have established their own private cloud for internal IT services. The main types of cloud service offerings are: (1) Infrastructure-as-a-Service (IaaS), such as provided by Amazon s3, (2) Platform-as-a-Service (PaaS), such as provided by Microsoft Azure and (3) Software-as-a-Service (SaaS), such as provided by Gmail [2].

By utilising cloud computing services, the high cost in building and maintaining a computing environment for accomplishing these services is effectively reduced [3]. Also, clouds provide unlimited data and object storage and computation for companies. Users can have computation tasks accomplished on the cloud in a pay-as-you-go mode and will not need setting up and maintaining their own computing environment, which is particularly a cost-saving solution in data- and computation-intensive applications such as applications in scientific research [4], [5].

Although the main aim of the cloud computing systems is providing the pervasive computing they are not free of failures [6]. Failures of the cloud include hardware failures such as resource crash, link down, etc. and software failures such as extra load, programs removal, etc.

Failures are a common phenomenon in large scale distributed computing such as cloud computing. In [7] and [8], the worst cloud outages of 2013 and 2014 are listed, respectively. Famous cloud providers in the market such as Amazon, Dropbox, Facebook and Google Drive suffered service interruptions in some areas. These outages cause clouds to lose data, money and customer's trust because some customers' services were unavailable totally or partially for some time. For example, Amazon could have potentially suffered close to \$5 million in missed revenue for a single hour of offline time [7]. Thus, an effective fault-tolerant technique is mandatory and reliable cloud applications are required to be deployed in such a way that cloud can automatically recover from failures without affecting the Quality of Service (QoS) needed and profit expected.

In this paper, both proactive and reactive techniques are considered in order to provide a hybrid fault-tolerant strategy for cloud computing systems. To be proactive, the strategy assumes both the usage time of virtual machines and their failure probability. To be reactive, the strategy applies both task replication and checkpointing techniques. The paper proposes an algorithm for selecting the suitable technique according to the OoS requirements and the resources availability information.

This paper is organized as follows: Section 2 presents fault tolerance in cloud computing. Problem formulation is introduced in Section 3. Section 4 provides the cloud architecture and the details of the proposed algorithms. In Section 5, mentions the related work. Section 6 presents the conclusions.

II. Fault Tolerance in Cloud Computing

In general, fault tolerance techniques used in cloud computing are proactive, reactive and task resubmission [9]. Proactive techniques require more information about cloud resources and works in a probabilistic fashion. Decisions of how to address possible failures in the cloud are made before starting execution of an application. They avoid recovery from faults by predicting the failure and proactively replace the suspected components from other working components. This potentially reduces the failure rates within cloud, and also increases the capacity and throughput.

Reactive fault tolerance techniques reduce the effect of failures on application execution when the failure effectively occurs. Many reactive fault tolerance techniques are available to be deployed in cloud computing systems. The most popular techniques are checkpointing and replication. Checkpointing enable the cloud to recover from failure and resume the execution of the application starting from a point near at which failure is occurred. This can be done be saving the state of the application periodically to a stable storage. In case of fault, this saved state can be used to resume execution of the application where the last check-point was registered instead of restarting the application from its very beginning. This can reduce the execution time and tolerate faults to a large extent. The resumption of the application may be done on the same VM after recovery or may be done on any other available VM. This strategy involves more wasted time. This time is due to the recovery of the failed VM in case of only one VM is available for executing the application or it is due to the rescheduling in case of multiple VMs are available. Nevertheless, this technique is the most suitable if there is only one copy of the required VM in the cloud.

Replication technique is based on the assumption that the probability of a single VM failure is much higher than of a simultaneous failure of multiple VMs. It avoids recomputation by simultaneously starting several copies of the same application on different VMs. With redundant copies, the cloud can continue to provide the service in spite of failure of some VMs carrying out application copies without affecting the QoS requirements. There are two well-known mechanisms following the replication strategy: multiversion and parallel. In multiversion, the application is executed by multiple VMs in parallel. Results from all VMs are introduced to a voting mechanism which selects the best one according to the value. In parallel mechanism, the application is executed by multiple VMs in parallel but the first produced result will be taken into account and all the other results are discarded. Parallel mechanism is more concerned with the time of getting results while multiversion is more concerned with the validation of results[10].

The response time of parallel mechanism outperforms both checkpointing and multiversion strategies. So, it can be selected in case of critical-time applications. Checkpointing is better than the other two strategies in terms of additional VMs required. So, checkpointing is suitable when cloud has a limited number of VMs. The multiversion strategy is better than the other two strategies when validation of result is required [10].

Task resubmission is the most widely used fault tolerance technique in current scientific workflow systems. Whenever a failed task is detected, it is resubmitted either to the same or to a different resource at a runtime [8].

III. Problem Formulation

Most of the existing clouds provide services to their customers in the form of storage such as iCloud, Dropbox and Google or in the form of computing such as Amazon Elastic Compute Cloud (EC2)[2]. Services are provided by the cloud as follows: customers submit their requests to the cloud with their QoS requirements. The cloud determines the level of the service and the required VMs after a negotiation with the customer about the price. Then, VMs start providing the service.

The Economy based objective of the clouds allows resources to be used for some purposes for which they were not originally designed. As a result, a large number of failures may be occurred. These failures will have a great impact on the availability, credibility and economy of the cloud [11]. This is because the system will search for another suitable resource or VM to perform the customer's service. This affects the time needed to serve customers' applications and then degrades the performance of the cloud. Thus, there is a need to minimize the effect of these failures on performance, when occurred.

Most of the existing cloud systems provide fault tolerance by replicating data and applications to be served by more than one VM, simultaneously. For example, Amazon S3 stores each object at multiple servers and Apple's iCloud service, similarly, rents infrastructure from both Amazon's EC2 and Microsoft's Azure. Nevertheless, as increasingly common reports of cloud outages attest, reliability remains imperfect [12]. The replication approach has the following major challenges:

1) In the cloud environment, there may be many VMs that can fulfill customers' QoS requirements, but they have a high tendency to fail. In such a scenario, if the broker neglects the failure history of VMs and its replicas when selected, the likelihood of failures will be high. This eventually results in compromising the user's QoS parameters in order to complete cloud applications.

2) It is too expensive to perform replication for all cloud applications and VMs. This is because there will be a profit charge lost when using extra VMs in serving the same application, however these extra VMs can be exploited to serve other applications. Thus, we only need to replicate applications executed on the most valuable VMs that will have a great impact on the performance of the cloud if they failed. Determining the most valuable VMs is a great challenge.

3) There are some fault tolerance techniques rather than replication such as checkpointing and parallel techniques. Selecting the most suitable technique for each service is another challenge in this work.

In this paper, in order to address the first challenge, we use a heuristic that finds a list of the VMs that can fulfill customers' QoS requirements and sorts them in an ascending order according to their failure probability.

In order to address the second challenge, we use a VM classification heuristic to determine the most valuable VMs. This heuristic depends on both the usage service time of VMs and the reliability level introduced by the customer.

For the third challenge, an algorithm is proposed to select the most suitable fault tolerance technique for the selected VM. The algorithm depends on customers' requirements such as cost and deadline time of applications.

IV. Cloud Architecture

Figure 1 shows the level architecture of a cloud computing system. Basically, it contains three main entities: Allocator, Virtual Machines (VMs) and Physical Resources. The Allocator acts as the interface between the cloud services providers and customers. So, it requires the following modules to be involved in its structure:

1) Quality of Service (QoS) Controller: The main function of this module is to ensure that the cloud can perform requests of customers within the limits of the QoS requirements. The most important QoS requirements include response time, cost, reliability and security. The QoS controller receives a request from customer along with his QoS requirements. It sends a query request for suitable VMs to the VMs database and receives a reply. If there are suitable VMs that can perform the request within the needed QoS requirements, the QoS controller will accept the request and delivers it to the Broker who can make allocation decisions for binding user requests to VMs. If there are no suitable VMs, the QoS controller will reject the request.

2) VMs Database: It contains all information about VMs in the cloud such as the speed, memory size, number of processors, and bandwidth. Also, it contains information about the usage history of each VM such as the usage time, the failure time and the failure rate. This database is updated according to the information received from the VMs Monitor.

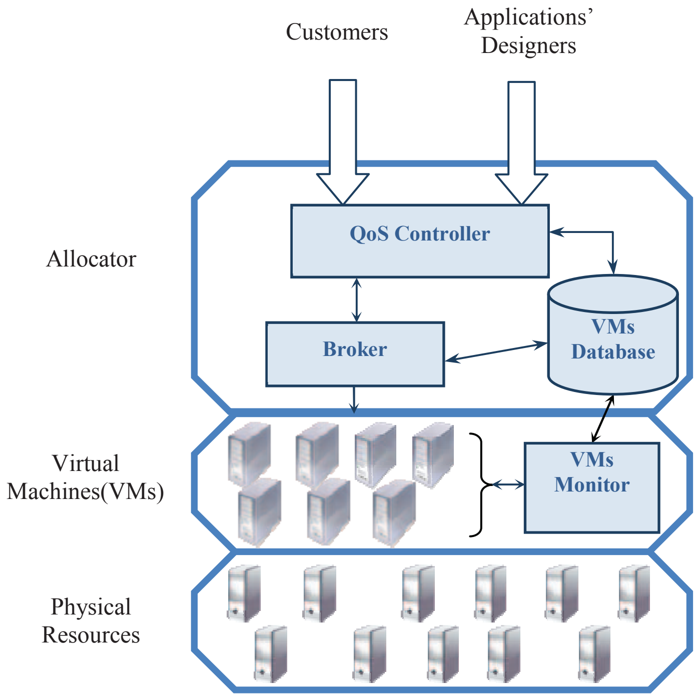

Fig. 1. The level architecture of a cloud computing system

3) Broker: The Broker determines the VMs that can perform customer requests and satisfy his/her QoS requirements. It receives accepted customer requests along with its QoS requirements from the QoS Controller. It needs the latest status information regarding availability and reliability of VMs in order to perform the binding process between requests and VMs. It can get this information from the VMs database. Also, the Broker has the main role in determining the price for the services needed according to the pricing mechanism used by the cloud. The pricing mechanism determines how service requests are charged. For instance, requests can be charged based on submission time, pricing rates or availability of resources. Pricing is the basis of managing the supply and demand of computing resources within the cloud and facilitates in prioritizing resource allocations effectively.

Figure 2 shows the internal structure of the Broker and interactions between its components. It contains the following components:

- **VMFT Selection**: The main function of this module is to select the suitable fault tolerance technique for customers' applications. This module contains a software agent called VM agent. This agent receives the application along with QoS requirements from the QoS Controller. It consults the VMs classifier module and gets a classified list of VMs that can perform the customer's application and fulfill the QoS requirements. Using Fault Tolerance Strategy Selection (FTSS) algorithm (see subsection 4.1), the agent selects the most suitable fault tolerance strategy. This selection depends on customers' requirements and the available VMs. Then, it delivers the application to the dispatcher along with a list of the identifiers of VMs selected.

- **VMs Classifier**: The main purpose of this module is to determine the VMs that will perform customers' applications. The module contains a software agent called VMC agent. This agent receives the QoS requirements of the application from the VMFT selection module. Then, it sends a query request to the VMs database to get the latest status information of VMs that can achieve the QoS requirements. It classifies the VMs that can perform customer's application according to both the usage time and the failure probability of the VMs. The details of the VMC algorithm are presented in the next subsection 4.2.

- **Dispatcher**: Once VMs are determined, the Dispatcher delivers customer applications to start execution on the selected VMs.

The second entity in the cloud computing system contains VMs. VMs are built up of numerous physical or actual resources. Each of these resources can be shared through multiple virtual machines, which is what are offered to the cloud customers. A VM typically emulates the physical environment of the cloud and virtualization software is used to map customer requests to the underlying physical resources. The virtualization software can be used to create many individual, isolated VM environments using the physical resources of the cloud. Multiple VMs can be applied on a single physical computer to meet customer requests by partitioning the various resources on the same physical computer to different VMs and then to different specific requirements of customer requests. The VMs Monitor is responsible for handling failures of VMs. It monitors VMs and notifies the scheduler to update the VM database records in case of a VM failure or recovery.

Physical Resources represents the third entity of the cloud computing system. They are distributed in the cloud and are employed by application designers to implement and deploy

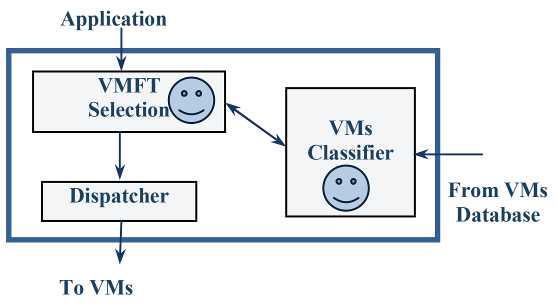

Fig. 2. Broker components and their interactions

cloud applications. To execute their applications, customers use physical resources of the cloud. Customers have no direct access to these resources. They can access cloud resources through VMs.

A. Fault Tolerance Strategy Selection (FTSS)

In this section, an algorithm for selecting the suitable fault tolerance strategy is presented and explained. Since collaborative solutions are much promising for handling fault tolerance [13], the handling of our algorithm to this issue is exclusively done by the cooperation provider and the customer.

The algorithm uses the QoS requirements, submitted by customers along with applications, and resources availability information registered in the cloud to select the suitable strategy. The steps of the algorithm will be as follows:

Input: cu is the application cost required by the customer,

τu is the application deadline time required by the customer,

ci the estimated cost if the application will be executed by Virtual Machine i,

τi the estimated time if the application will be executed by Virtual Machine i,

m the list of most valuable VMs in the cloud,

```
j = 1;
    While (there are applications not served)
        {
        For (each application $A_j$)
            {
            Find a list of all VMs that can execute $A_j$;
            For each Virtual Machine $i$ in the list do
                If ($c_u < c_i$ && $\tau_u < \tau_i$)
                    remove virtual machine $i$ from the list;
        If(list is empty)/*The application cannot be served */
                {
                Send "Application cannot be served" to the QoS controller;
                    End the algorithm for $A_j$;
                }
            Else /* The application can be served */
                {
            Sort the VMs list ascending based on $c_i$ x $\tau_i$;
            If (there is more than one VM in the list) /*replication is selected*/
            {
            If (first VM in the list is in the $m$ list)/*valuable VM*/
            {
                Determine the number of replications;
                If (validation is required)
                    Multiversion strategy is selected;
                Else
                    Parallel strategy is selected;
            }
            Else
                Checkpointing strategy is selected;
            }
            j++;
            }
    }/* while end*/
```

B. VMs Classification Algorithm

The main purpose of the VMC algorithm is to classify cloud virtual machines according to the available information about them. The information used in classification include the time of virtual machine usage and the failure probability of the virtual machine. The time of virtual machine usage,T_i, is the total time spent by the virtual machine i in executing cloud applications. This time indicates the great value of the virtual machine to the cloud system since more usage time of a VM means that it is more valuable and profitable to the cloud.

The failure probability of a virtual machine provides an indication about its failure history when invoked to execute cloud applications and whether it needs applying fault tolerant techniques or not. The higher the failure probability of a virtual machine, the higher the need to be a fault tolerant one.

It is assumed that the number of virtual machine failures in any two disjoint time intervals is independent over time.

Hence, it can be assumed resource failure probability to follow a Poisson distribution. Thus, the failure probability distribution of a virtual machine i is given by the formula:

$$fp_i = \frac{e^{-\mu} \mu^{x_i}}{x_i!}$$

Where x_i is the number of failures occurring in a given time interval and μ is the mean number of failures in the given time interval for a virtual machine i. The value of the failure probability is in the range of (0, 1).

The virtual machine classification module in the VMs is responsible for classifying virtual machines of the cloud. This module obtains the virtual machine usage time and the virtual machine failure probability from the virtual machine monitoring module. Then, it constructs a list of the most valuable virtual machines based on the formula:

$$S_i = fp_i \times T_i$$

Where S_i is the selection parameter used to determine virtual machines to be replicated when an application is assigned to be executed by virtual machine i.

In this paper, the virtual machine with a higher value of S_i is considered more valuable than a virtual machine with a lower value of S_i. More valuable virtual machines have higher priority of replication than other virtual machines because the failure of it will have great impact on the performance of the cloud. Thus, these virtual machines have higher fault-tolerance needs than other less valuable virtual machines.

Based on the value of S_i, virtual machines can be sorted in descending order and the top m virtual machines can be selected as the most valuable ones. Only applications that will be executed on these m virtual machines will be replicated. The value of m is determined according to the QoS requirements determined by customers when submitting their applications.

V. Related Work

A large number of research efforts have already been devoted to fault tolerance in the area of grid computing. However, a little work has been done for fault tolerance in cloud environments. Aspects that have been explored include the design and implementation of fault detection services, as well as the development of failure prediction, and recovery strategies. The recovery strategies are often implemented through task replication reactive techniques.

Alhosban et al. [14] have proposed a fault occurrence likelihood estimation and exception handling technique. Their technique is divided into two phases: Phase I, service reliability and utility and Phase II, runtime planning and evaluation. In Phase I, the fault likelihood of the service is assessed. In Phase II, a recovery plan to execute in case of fault(s) is built. They have calculated the overall system reliability based on the fault occurrence likelihoods assessed for all the services. Their technique support dynamic management based on the changes in user requirements and QoS levels.

In [15], P. Das and P. M. Khilar have proposed a reactive fault tolerance technique to reduce the service time and to increase the system availability. They have integrated the fault tolerance with virtualization. The basic fault tolerance mechanism used in their model is replication. They have performed replication in form of software variants running on multiple virtual machines. Their model not only tolerate faults but also reduce the chance of future faults by not assigning tasks to virtual nodes of physical servers whose success rates are very low.

Z. Zheng et al. [10] have proposed a component ranking framework for building fault-tolerant cloud applications. Their proposal includes two phases. The first phase identifies the significant components in a cloud application. They have employed component invocation frequencies for making component ranking. The second phase employs an algorithm to automatically determine an optimal fault-tolerance strategy for the significant cloud components.

K. Ganga and S. Karthik[16] have classified the fault tolerance techniques as proactive and reactive techniques. In proactive techniques, the failure is predict and the infected resources are replaced by uninfected ones. Reactive techniques use checkpointing, replication and resubmission in a try to reduce the effect of failures when occurred. The main target in [16] was to apply job replication on scientific workflow systems.

An approach for realizing generic fault tolerance mechanisms is presented by Jhawar, Piuri and Santambrogio [12]. They have presented their approach as independent modules. The approach validates fault tolerance properties of each mechanism and matches between user's requirements and the available fault tolerance modules to obtain a comprehensive solution with desired properties. Also, a framework is designed that allows the integration between provider's system and the existing cloud infrastructure.

Reviewing literatures reveals that most of the previous works done are mainly based on using the response time and number of failures as the main criteria for selecting VMs for customers' applications. There is no work done that considers the usage time or the failure probability of VMs.

Also, most of the previous work considers only on technique for fault tolerance, mostly replication with a static or fixed number of replicas. Thus, extra VMs will be used in executing user applications. As a result, cloud will lose the monetary benefit of these VMs. So, a way is required to provide a dynamic number of replicas to maintain the monetary profit of the cloud.

VI. Conclusion

In this paper, we have proposed a fault tolerance framework for an economy based cloud computing systems. The proposed framework enables a service provider to offer fault tolerance support to customers' applications. We presented our framework as independent modules. Two algorithms that represent the main work of the framework are presented. The first algorithm is used to select VMs for customers' applications depending on both usage time and failure probability of VMs. The second algorithm is for selecting a suitable fault tolerance technique. The algorithm uses a dynamic number of replicas if replication is selected as the fault tolerance technique. Our future work will mainly be driven toward the implementation of the framework to measure the strength of fault tolerance service and to make an in-depth analysis of the cost benefits among common service providers.

References

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, Vol. 25, 2009, pp. 599–616.

[2] S. O.Kuyoro, F.Ibikunle and O. Awodele, "Cloud Computing Security Issues and Challenges," International Journal of Computer Networks (IJCN), Vol. 3, Issue 5, 2011, pp. 247-255.

[3] E. Deelman, G. Singh, M. Livny, B. Berriman, J. Good, "The cost of doing science on the cloud: the montage example," in: ACM/IEEE Conference on Supercomputing, SC'08, Austin, Texas, 2008, pp. 1–12.

[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: a Berkeley view of cloud computing, Technical Report No. UCB/EECS-2009-28, University of California at Berkeley. Available: http://www.eecs.berkeley.edu/ Pubs/TechRpts/2009/EECS-2009-28.pdf (accessed on: 24.10.2011).

[5] L. Wang, M. Kunze, J. Tao, G.v. Laszewski, "Towards building a cloud for scientific applications", Advances in Engineering Software, Vol. 42, 2011,pp. 714–722.

[6] A. Gómeza, L.M. Carril, R. Valin, J.C. Mouriñoa, C. Cotelo," Fault-tolerant virtual cluster experiments on federated sites using BonFIRE," Future Generation Computer Systems 34 (2014) 17–25.

[7] The worst cloud outages of 2013, Online, cited 01.07.2013. URL: http://www.infoworld.com/slideshow/107783/the-worst-cloud-outages-of-2013-so-far-221831.

[8] The worst cloud outages of 2014, Online, cited 01.01.2015. URL: http://www.infoworld.com/article/2606209/cloud-computing/162288-The-worst-cloud-outages-of-2014-so-far.html.

[9] K.Ganga and Dr S.Karthik, "A Fault Tolerant Approach in Scientific Workflow Systems based on Cloud Computing," Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), February 21-22, 2013, pp 378-390.

[10] Z. Zheng et al.," Component Ranking for Fault-Tolerant Cloud Applications," IEEE Transactions on Services Computing, Vol. 5, No. 4, Oct.-Dec. 2012, pp. 540-550.

[11] R. Jhawar, V. Piuri and M. Santambrogio, "Fault Tolerance Management in Cloud Computing: A System-Level Perspective," IEEE Systems Journal, Vol. 7, No. 2, June 2013, pp. 288-297.

[12] Ennan Zhai, David Isaac Wolinsky, Hongda Xiao, Hongqiang Liu, Xueyuan Su, and Bryan Ford, "Auditing the structural reliability of the clouds," Technical Report YALEU/DCS/TR-1479, Department of Computer Science, Yale University, 2014. Available at http://cpsc.yale.edu/ sites/default/files/files/tr1479.pdf, accessed on March 9, 2014.

[13] A. Tchana, L. Broto and D. Hagimont, "Approaches to Cloud Computing Fault Tolerance," Proceedings of International Conference on Computer, Information and Telecommunication Systems (CITS), May 14-16, 2012, pp. 1-6.

[14] A. Alhosban et al, "Self-healing Framework for Cloud-based Services," Proc. of 2013 Int'l Conf. on Computer Systems and Applications, May 27-30.

[15] P. Das and P. M. Khilar, "VFT: A Virtualization and Fault Tolerance Approach for Cloud Computing," Proc. of 2013 IEEE Conference on Information and Communication Technologies, Thuckalay, Tamil Nadu, India, April 11-12, pp. 473-478.

[16] K. Ganga and S.Karthik, "A Fault Tolerent Approach in Scientific Workflow Systems based on Cloud Computing," Proc. of the IEEE International Conference on Pattern Recognition, Informatics and Mobile Engineering, Feb. 21-22, 2013, pp.117-122