2012 IEEE 4th International Conference on Cloud Computing Technology and Science

Cloud Monitoring for Optimizing the QoS of Hosted Applications

Khalid Alhamazani[1] (Phd Student), Rajiv Ranjan[2], Fethi Rabhi[1], Lizhe Wang[3], Karan Mitra[2]

[1]School of Computer Science and Engineering, University of New South Wales
{ktal130,Fethir}@cse.unsw.edu.au
[2]Information Engineering Laboratory, CSIRO ICT Centre
{rajiv.ranjan, karan.mitra}@csiro.au
[3]Chinese Academy of Sciences, Beijing, China
{lizhe.wang}@gmail.com

***Abstract*—Cloud monitoring involves dynamically tracking the Quality of Service (QoS) parameters related to virtualized services (e.g., CPU, storage, network, appliances, etc.), the physical resources they share, and the applications running on them or data hosted on them. Monitoring techniques and services can help a cloud provider or application developer in regards to: (i) keeping the cloud services and hosted applications operating at peak efficiency; (ii) detecting variations in service and application performance; (iii) accounting the SLA violations of certain QoS parameters; and (iv) tracking the leave and join operations of cloud services due to failures and other dynamic configuration changes. In this paper, we describe the PhD research motivation, question, and approach and methodology related to developing novel cloud monitoring techniques and services enabling automated application QoS management under uncertainties.**

Keywords-cloud computing; monitoring; QoS; SLA; SNMP

I. INTRODUCTION

Cloud computing paradigm is shifting computing from physical hardware and locally managed environments to virtualized Cloud services. In a nutshell, Cloud computing assembles large networks of virtualized services: hardware services (compute services, storage, and network) and software services (e.g., web server, databases, message queuing service, monitoring service, etc.). Cloud service types can be abstracted into three layers: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [1], [2], [3], [4], and [23]. Hardware and software services form the basis for delivering IaaS and PaaS. The top layer focuses on application services (SaaS) by making use of services provided by the lower layers. PaaS/SaaS services are often developed and provided by third party service providers who are different from the IaaS providers [5]. Cloud providers including Amazon Web Services (AWS) and Rackspace give users the option to deploy their application over a pool of virtually in niter resources with practically no capital investment and with pay-per-use billing. Elasticity, proportional costs and other benefits motivate many organizations to migrate their IT systems to the cloud, so that they can be rapidly provisioned and released through REST/SOAP API with minimal supervision, management and effort or any type of direct interaction from a service provider.

II. MOTIVATION

Performance unpredictability is the biggest obstacle facing the migration of applications (e.g., multi-layered business application, scientific data processing application, multi-media application, etc.) to clouds. While aforementioned applications are often held to strict Quality of Service (QoS) targets in terms of throughput, delay, security, privacy, or availability, little is known about the performance of applications in the cloud [19, 20], the response time variation induced by network latency and cloud location. QoS targets are encoded in legal Service Level Agreement (SLA) documents, which state the nature and scope of QoS parameters. Some of the QoS parameters that may be included in SLA documents include service renting cost, availability, service credit for compensating SLA violations, etc. For example, Amazon S3 (storage service) is offered under a SLA[1] that says Amazon will make commercially reasonable efforts to make S3 available with a monthly uptime percentage of at least 99.999% during any monthly billing cycle. Users are eligible to receive a service credit if Amazon fails to meet the SLA commitment. It is not difficult to note that current SLA model supported by cloud providers are trivial, since they do not cater for other complex QoS parameters which are generally associated with different applications types, including eResearch applications (e.g. data transfer latency, data transfer throughput, data security guarantee, data integrity guarantee etc.).

The recent very high-profile crash of Amazon EC2 cloud [21], which took down the applications of many SMEs, is a salient example of unpredictability in cloud environments. Some applications were down for hours, others for days. Theoretically, the elasticity provided by cloud computing can accommodate even unexpected changes in capacity, adding cloud services (e.g., CPU, storage, network, database, etc.) when needed, and reducing them during periods of low demand, but the decisions to adjust capacity must be made frequently, automatically, and accurately to be cost effective.

The failure or congestion of network links are sometimes inevitable, given the scale, dynamics, and complexity in cloud computing, the crash or malfunction of a hardware resource, changes in workload patterns, or overloading of a hardware resource. Worse still, hardware service status can be changed intentionally through malicious external interference. Determining how to adapt to unpredictable conditions is an inherently complex problem for which several technical implications must be considered.

Our focus in this proposal is to apply dynamically monitored application components' and cloud services QoS parameters for automatic QoS control under uncertainties. At run-time, set of operations takes place in order to meet the QoS specified in SLA document that guarantees the required objectives of the cloud users. The availability, load, and throughput of hardware services can vary in unpredictable

[1] http://aws.amazon.com/s3-sla/

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

ways, so ensuring that applications achieve QoS targets can be difficult. Handling such uncertainties is must for ensuring fulfillment of QoS targets; hence being aware of the system current software and hardware service status is a must. In addition, detecting exceptions and malfunctions while deploying software services on hardware services is a priority e.g. showing QoS delivered by each application component (software service such as web server or database server) hosted on each hardware service. Uncertainties can be tackled through development of efficient, scalable, inter-operable, easy-to-use monitoring tool. In this proposal and as stated in above-mentioned issues, my plan is to conduct the following motivation challenges to solve.

To develop techniques that can dynamically monitor, predict (e.g, estimating QoS parameters in advance, and adapt according to these prediction models.) and capture the relationship between application QoS targets, current cloud service allocation and changes in workload patterns, in order to adjust service configuration remains an open research problem. Overall, the integration of theoretical workload prediction, service performance models and optimization techniques to effect an end-to-end monitoring and automated QoS management over cloud environments is a hitherto neglected research area.

The rest of this paper is organized as follows. Section III discusses the research questions and problem statements. Section IV details the proposed approach and methodology. Discussion on expected contributions to be made by this PhD research is given in Section V, while Section VI summarizes the state-of-the-art. Section VII presents the preliminary research work done. Conclusion and future work is presented at last in Section VIII.

III. Research Questions and Problem Statements

The goal of this PhD research is to develop intelligent and scalable cloud monitoring techniques for automating the QoS management of hosted applications (e.g. video on demand multi-media applications, layered enterprise application, etc.). Monitoring is the process of dynamically tracking the QoS parameters related to virtual cloud services, the physical resources they share, and the applications running on them or data hosted on them. Monitoring techniques can help an administrator or an application developer as regards to: (i) keeping the cloud services and applications operating at peak efficiency; (ii) detecting variations in service and application performance; (iii) accounting the SLA violations of certain QoS parameters; and (iv) tracking the leave and join operations of services due to failures and other dynamic configuration changes. The following are major technical problems in regards to monitoring and automatic QoS management of applications in cloud environments.

RQ1: Scalable Monitoring of Application Components

Although the components (e.g., web server, database server, storage services, etc.) that contribute to cloud application architecture may be distributed, existing techniques usually employ centralized approaches to overall system monitoring and management. We claim that centralized approaches are not an appropriate solution for this purpose, due to concerns of scalability, performance, and reliability arising from the management of multiple application service queues and the expected large volume of application service requests. Monitoring of application components is required for effecting on-line control through a collection of system QoS characteristics. We require that gathered information is supplied as feedback to the prediction and performance models, for their continuous training and refinement. In this part of work, the research question is what type of network model (e.g., hierarchical, unstructured peer-to-peer, structured peer-to-peer, etc.) should be applied in architecting monitoring services such that it scales with increase in application size (number of components), cloud service pool (number of compute services provisioned), and workload patterns (number of application users, request rate, etc.)?

RQ2: Computing QoS parameters and targets in accordance with the application type and nature

There are number of technical challenges involved with designing cloud-based application architecture. Existing in-house application components need to be significantly re-engineered in order to properly operate in the virtualized cloud computing environments. Mainly, the core technical challenges as regards to cloud-enabling applications are cloud hardware and software services. Moreover, techniques required for efficiently overcoming cloud-enabling design differ based on application type, purpose, QoS targets, and SLA. Notably, QoS targets vary across application types. For example, QoS targets for eResearch applications are different from static, single tier web applications (e.g., web site serving static contents) or multi-tier applications (e.g., on demand audio/video streaming). Based on application types, there is always need to negotiate different SLAs. Hence, SLA document will include conditions and constraints that match the nature of those QoS requirements with each application type. For example, a bio-informatician running genome analysis experiment on cloud services will only care of the data transfer (upload and download) network latency and processing latency. On the other hand, with different type of applications like multi-media applications, the quality of the transferred data over network is more important. Hence, other parameters gain priority in this case. In Table 1, we show QoS parameters for streaming system; we present the QoS parameters for different components of cloud-based multi-media application at SaaS, PaaS, and IaaS level.

As shown in figure 1, application components (streaming server, web server, indexing server, compute service, storage service, and network) related to multi-media streaming application is distributed across cloud layers including PaaS and IaaS. Thus in order to guarantee achievement of QoS targets for the application as a whole, it is critical to monitor QoS parameters across the layers [7][22]. However, it is non-trivial for application developers to understand what QoS parameters and targets he or she need to specify and monitor across each layer of cloud stack including PaaS (e.g., web server, streaming server, indexing server, etc. in figure 1) and IaaS (e.g., compute services, storage services, and

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

network). Hence, the research question in this part of work is what type policy language should be developed so that application developers can specify their monitoring policies in a simplified, yet in an effective way? And other research question is how to capture these cross layer QoS parameters in a unified QoS target function that is visible at SaaS layer?

RQ3: Predictive and Automated Application QoS Management

In a cloud environment, it is critical that the application provisioning system is able to predict the demands and behaviors of hosted applications, so that it can manage its application components adaptively. Concrete prediction or forecasting models must be built before the behavior of an application, in terms of number of service components, computing, storage, and network bandwidth requirements, can be predicted accurately. The challenge in devising such models is accurately learning and fitting statistical functions to the observed distributions of application behaviours such as request arrival pattern, service time distributions, I/O system behaviours, user profile, and network usage. This challenge is further aggravated by the existence of statistical correlation (such as stationary, short- and long-range dependence, and pseudo-periodicity) between different behaviors of a service. The process of mapping application components to cloud services (CPU, storage, network, appliances) is a complex undertaking, as it requires the provisioning system, to compute the best PaaS and IaaS configuration (system size and mix of services) to ensure that QoS targets of applications are achieved, while maximizing system efficiency and utilization. This process is further complicated by the uncertain behavior of cloud services and applications. For example, in order to reduce response time, an application may be deployed with an aggressive deployment plan; however this can lead to over-provisioning. Further, as the differences between the cloud service allocations and application QoS targets increase; the wastage of cloud service and the increase in overall cost are magnified. Consequently, there is an immediate need to devise cloud service performance modeling and hosted application workload prediction techniques which ensure efficient system utilization without having an unacceptable impact on QoS targets. The research question in this part of work is can past monitored QoS information of application components and cloud services be helpful for training machine learning models and developing predictive QoS management techniques?

IV. APPROACH AND METHODOLOGY

Task 1. Distributed Hash Table-based Decentralized Monitoring

For solving RQ1, we believe that monitoring service will need to implement scalable network model for collecting and profiling monitored QoS information. This leads us to believe that we should interconnect and monitor application components across the layers (PaaS and IaaS) based on decentralized messaging and information indexing infrastructure, called Distributed Hash Tables (DHTs). However, implementing scalable techniques that monitor the dynamic behaviours related to application components and cloud services is non-trivial. In order to support a scalable application monitoring service over a DHT infrastructure, additional data distribution indexing techniques such as logical multi-dimensional or spatial indices [9, 10] (MX-CIF Quad tree, Hilbert Curves, Z Curves) will be implemented. Since, the choice of data distribution and logical routing technique will govern the manageability and efficiency of the monitoring system. Therefore, the performance of the monitoring system will be evaluated by measuring messaging latency, traffic, routing load balance overhead. These data distribution techniques will be implemented and tested within the FreePastry DHT framework [8].

Task 2. Policy-based QoS Monitoring and Profiling

First, this task will develop a generic service for efficient QoS monitoring and status profiling of cloud services across multiple layers (IaaS and PaaS) and underlying wide-area network infrastructure. One of the main challenges of this task is to identify the most important QoS parameters (e.g., privacy, communication cost, renting cost, etc.) and the mode of interaction (e.g., mobile, laptop, etc.) that application developers would like to monitor and profile for their applications. Towards achieving this goal, this task will identify the application use-cases and their QoS implications when hosted in cloud environments.

Second, this task aims at developing a policy language for simplification of interactions between cloud services and application developers. Policy based techniques using WS-Policy language capabilities and extensions is the widely proposed solution with the use of web services in IT industry [6]. Policy languages could help developers in specifying and monitoring required QoS parameters and profiling respective policies. The main objective of this policy language is to capture sufficient information for application component and cloud service monitoring. This objective leads towards an interesting yet challenging research problem of automatic mapping of user-level preferences and QoS specifications to low, system-level cloud programming interfaces. To address this problem, we will explore possible extensions of policy specification languages including WS-Policy, XACML, and SecPAL. Though applicability of these languages has been successfully exploited in the domain of Web services and secured information systems, their extensions to cloud application monitoring is yet to be explored. The dynamic QoS characteristics related to services and application will be accumulated in the profiling repository and the information in the repository will be continuously updated (Task 1) for building prediction models (Task 3).

Task 3: Novel Statistical Workload Prediction Models

In this task, we will develop novel application workload and cloud service performance prediction models based on the recent advances in computational statistical techniques (e,g. time series clustering, decision tree learning, quadratic

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

response surface models, Kernel Canonical Correlation Analysis etc.). In particular, these models will capture the behaviors of a given application workload in terms of response time, throughput, utilization, as well as other QoS parameters. These behaviors will be captured based on the actual traces of incoming requests for applications. One of the challenges in building prediction models is to deal with noise present in the input data (Task 1). Random effects and outliers will need to be separated to retrieve the true trends and turning points within the time series.

This task will also investigate the most effective ways of dynamically learning new adaptation behaviors based on monitoring and profiled monitoring data (Task 2). It will then decide how these cloud services and application components should react to new situations. This task takes a novel step of fully integrating the prediction models with a monitoring system to continuously learn the state and performance of existing clouds services, application components and underlying wide-area network infrastructures. These prediction models will then be updated accordingly. An online learning technique based on Support Vector Machine (SVM) regression will be developed for training the prediction models based on real-time feedback data. SVM regression has proven to be an effective method at learning non-linear functions by applying linear machine learning techniques. Moreover, Network management techniques (manager/agent) will be applied in order to capture the data being transferred over the network. Simple Network Management Protocol so far is the suggested technique.

V. NOVEL CONTRIBUTIONS

The contribution that we make by solving RQ1 will be novel DHT-based scalable technique for monitoring the behaviour of application components and cloud services in completely decentralized and distributed manner with the aim of providing valid data for online model training (RQ3). Decentralized models for monitoring and managing application components and cloud services are highly scalable and can gracefully adapt to the dynamic system expansion (join) or contraction (leave and failure).

In Task 2, the contribution will be novel user-interfaces and policy language, which will allow non-expert cloud users to express their application specific QoS parameters in a seamless way. The policy language will strive to encapsulate both application-centric and cloud service-centric QoS targets. This will be helpful in achieving important performance goals, achieving application-centric performance targets without over-provisioning the cloud services.

At this stage and having completed Task 1 and 2, the contribution of Task 3 builds upon the contributions made by previous tasks. The monitored QoS parameters will e collected in real-time in profiling repository, which will be continuously updated. Based on this information, we will apply machine learning techniques to develop Intelligent and autonomic application provisioning to allow knowledge-driven, predictive management where the allocations of clouds services are automatically varied in response to the crash or malfunction, changes in workload patterns, or overloading of an application component. Other indirect contributions here would be theoretical application workload and cloud service performance model QoS parameters and functions that will be learned and predicted.

VI. PROGRESS BEYOND STATE OF THE ART

In this section, we analyze existing commercial and academic cloud monitoring techniques and services against the research work proposed in these PhD.

Major commercial players including Amazon Web Services (AWS) [11, 12, 13], Microsoft Azure [17], Google App Engine [14], and GoGrid [16] offer monitoring services. Currently, AWS offers three centralized services that enable monitoring and QoS management of applications hosted on its compute and storage cloud services. These services include CloudWatch [12] for monitoring, Elastic Load Balancer [13] for load-balancing and Auto Scaler [11] for automatic application scaling and de-scaling. The Azure Fabric Controller (FC) [17] is the service, which monitors, maintains and provisions CPU services to host the applications that the developer creates and deploys in the Microsoft Azure Cloud. The behavior of the FC is made redundant by creating multiple replicas (5 to 7) at any given point of time. GoGrid does not explicitly offer a monitoring or auto scale service but it offers a centralized load-balancing service called F5 Load Balancer [16] for distributed application traffic component across components. Unlike aforementioned cloud providers, Google App Engine handles the monitoring and QoS management of cloud services and application components behind the scene. Eucalyptus [15] implements hierarchical network architecture for monitoring the status of CPU, storage, and network services.

However, aforementioned commercial cloud monitoring services have following technical limitations, which require further research and development: (i) they implement centralized of hierarchical network model for engineering monitoring and QoS management services. These network models have been proven to be performance bottleneck with the increase in system size and application request intensity; (ii) they do not support simplified interfaces and policy specification languages that can allow non-technical users in expression QoS and SLA management needs based on their application case-by-case basis (e.g., CRM, SCM, media CDN, and eResearch applications); and (iii) they do not support monitoring of QoS parameters across IaaS and PaaS layers. For example AWS CloudWatch is not capable of monitoring information related to load, availability, and throughput of each core of CPU services and its effect on the QoS delivered by the hosted application component (e.g. Tomcat hosted web application).

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

Further, none of the aforementioned commercial cloud monitoring and auto scaler services implemented predictive and automated QoS management technique. To deal with exceptions in cloud computing, these commercial cloud services implement reactive techniques [18]. These techniques rely on monitoring the state of hardware services (which host instances of software resources) and triggering

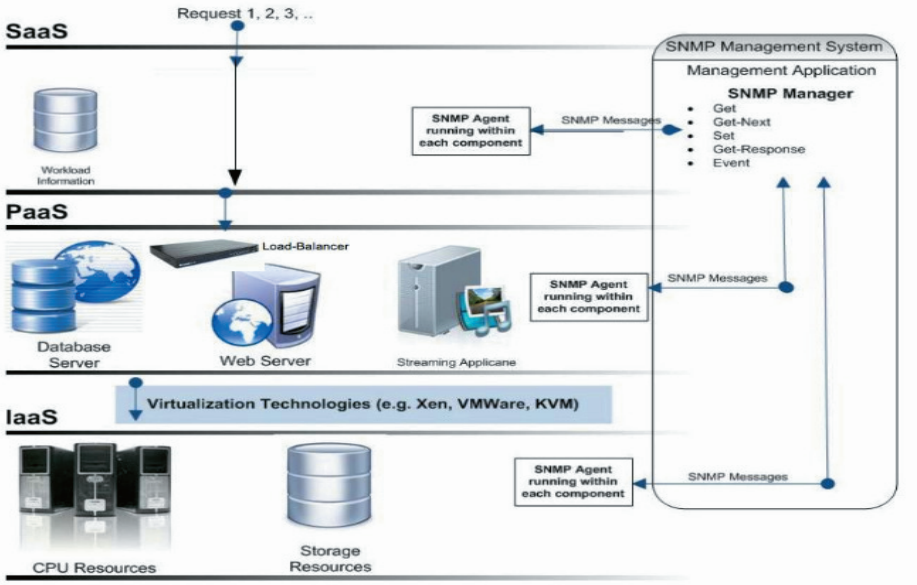

Figure 1. Monitoring Streaming system at different layers architecture

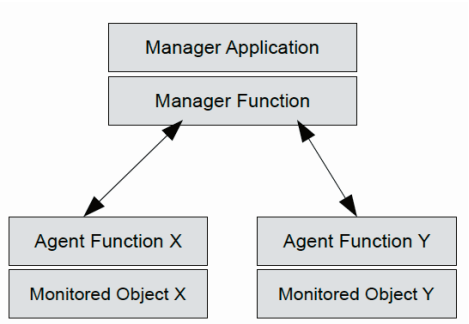

Figure 2. Components remote monitoring architecture.

hard-coded, pre-configured corrective actions (e.g. allocating a new instance of software resource) when some event occurs (e.g. utilization of host CPU resource reaches a certain threshold). Reactive techniques are not sufficient to ensure guaranteed performance in the event of variation in state of cloud services.

VII. Progress Till-Date

We are in the process of developing techniques for monitoring QoS across several layers of the cloud stack Using Simple Network Management Protocol (SNMP). Development in this project will be implemented and tested for managing QoS of multi-media applications as shown in Fig. 1. Having QoS status of application component known across these layers is critical. Network monitoring involves observing and analyzing the status of network devices distributed on a network. In our context, remote monitoring these network endpoints will be performed, and the information collected from these endpoints and then will be sent to a central manager.

SNMP can be used as an enabler for monitoring distributed applications/components. Usually, network elements do have support for SNMP in their operating systems. The SNMP manager will have a one-to-many relationship with the managed/monitored nodes as shown in Fig.2 In SNMP, Management Information Base (MIB) provides means for the manager application to retrieve information from managed services or network elements. MIBs define the structure of the data to be received and it is an extensible. It encapsulates name spaces for users to get required parameters using the Object Identifiers (OID).

To illustrate, the architecture should be composed of (manager application, manager function, agent function, and managed object) as in Fig. 2. The manager application and manager function can be at a different location than agent function and the managed resource. The agent function gathers information and sends it to the manager function. There could be multiple agent functions that sends information to one manager function. That is where aggregation would take place when agent functions gather information for different components at different layers.. The manager function then will present the aggregated value of those monitored QoS parameters as one value readable the end user.

VIII. CONCLUSION AND FUTURE WORK

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

Monitoring in clouds is a key factor, so finding out what are the parameters that manipulate and affect the monitoring process is highly important. Also, it is a prior step and stage to SLA, QoS, and Auto-Scaling. The proposed study is expected to contribute significantly by providing detailed analysis to monitoring tools and defining how monitoring process is correlated to the SLAs, QoS and Auto-Scaling. Besides the academic significance of this research, business sector will vastly gain benefits. Consequently, this will lead to gaining more profits if having more reliable monitoring tools for customers.

Component/Service	Layer	QoS Paramaters
Load-Balancer	PaaS	HrSystemUpTime , SysDescr, SysServices, HrSystemProcesses HrSystemMaxProcesses
Streaming Server	PaaS	TTL, IpDefaultTTL, UdpInDatagrams, UdpOutDatagrams, HrSystemUpTime SysDescr, FilesUpLoadRate, FilesDownLoadRate, BufferLength, SysServices - list of services offered, HrSystemProcesses, HrSystemMaxProcesses
DBMS Server	PaaS	BytesRead, BytesWrite, HrSystemUpTime, SysDescr, SysServices, HrSystemProcesses, HrSystemMaxProcesses
Web Server	PaaS	HrSystemUpTime, SysDescr, SysServices, HrSystemProcesses HrSystemMaxProcesses
Application Server	PaaS	HrSystemUpTime, SysDescr, SysServices, HrSystemProcesses, HrSystemMaxProcesses
CPU	IaaS	Utilization, ClockSpeed, CurrentState, HrSystemMaxProcesses
Network	IaaS	Capacity, Bandwidth, Throughput, ResponseTime, OneWayDelay, RoundTripDelay, TcpConnState, TcpMaxConn
BLOB	IaaS	SysDescr, TcpConnState, TcpMaxConn, Capacity/Size, BufferSize, HrSystemProcesses, HrSystemMaxProcesses

Table1: Monitored QoS Parameters at different cloud layers for streaming system components.

REFERENCES

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," 2008, pp. 1-10.

[2] S. Distefano, A. Puliafito, M. Rak, S. Venticinque, U. Villano, A. Cuomo, G. Di Modica, and O. Tomarchio, "QoS Management in Cloud@ Home Infrastructures," 2011, pp. 190-197.

[3] L. J. Zhang and Q. Zhou, "CCOA: Cloud computing open architecture," 2009, pp. 607-616.

[4] A. M. Hammadi and O. Hussain, "A Framework for SLA Assurance in Cloud Computing," 2012, pp. 393-398.

[5] C. Hoefer and G. Karagiannis, "Taxonomy of cloud computing services," 2010, pp. 1345-1350.

[6] J. Shao, H. Wei, Q. Wang, and H. Mei, "A Runtime Model Based Monitoring Approach for Cloud," 2010, pp. 313-320.

[7] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for QoS-aware clouds," 2010, pp. 237-250.

[8] Freepastry Distributed Hash Table Framework, http://www.freepastry.org/, [accessed September 2012].

[9] P. Ganesan, B. Yang, and H. Garcia-Molina (2004)," One torus to rule them all: multi-dimensional queries in p2p systems," In WebDB '04: proceedings of the 7th international workshop on the web and databases, New York, NY, USA. ACM Press, New York, pp 19–24.

[10] R. Ranjan, A. Harwood, R. Buyya, (2008), "Peer-to-Peer-based Resource Discovery in Global Grids: a tutorial," IEEE Communications Surveys and Tutorials, vol. 10, issue 2, pages 6–33, IEEE Communications Society Press.

[11] Amazon auto scaling service, http://aws.amazon.com/autoscaling/. Accessed: September, 2012.

[12] Amazon cloudwatch service (2012) http://aws.amazon.com/cloudwatch/. Accessed: September, 2012.

[13] Amazon load balancer service, http://aws.amazon.com/elasticloadbalancing/. Accessed: September, 2012.

[14] Google app engine, http://code.google.com/appengine/. Accessed: September, 2012.

[15] Eucalyptus systems, http://www.eucalyptus.com/. Accessed: September, 2012.

[16] GoGrid Cloud Hosting, (f5) load balancer. GoGrid wiki, http://wiki.gogrid.com/wiki/index. php/(F5)-Load-Balancer. Accessed: September, 2012.

[17] Windows azure platform, http://www.microsoft.com/azure/. Accessed: September, 2012.

[18] L. Vaquero et al., "Dynamically Scaling Applications in the Cloud," SIGCOMM Comp. Comm. Rev., Vol. 41, No. 1, pp. 45-52, 2011.

[19] J. Schad et al.,"Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance," Proc. of the VLDB Endow. 3, 1-2 (September 2010), pp. 460-471, Springer.

[20] A. Iosup et al., "On the Performance Variability of Production Cloud Services," Proc. of the IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2011), IEEE Computer Society.

[21] Crash of Amazon EC2 Cloud Services, http://www.businessinsider.com/amazon-lost-data-2011-4, [accessed May 2011].

[22] R. Ranjan et al., "Grid-Federation: A Resource Management Model for Cooperative Federation of Distributed Clusters," Technical Report, GRIDS-TR-2004-10, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2004.

[23] R. Buyya and R. Ranjan, "Special Section: Federated Resource Management in Grid and Cloud Computing Systems," Future Generation Computer Systems, 2010, Vol 26, Issue 8, Pages 1189–1191.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE