Journal of the Korea Convergence Society
Vol. 8. No. 7, pp. 29-37, 2017 https://doi.org/10.15207/JKCS.2017.8.7.029

Design and performance evaluation of a storage cloud service model over KREONET

Wontaek Hong[1*,2], Jinwook Chung[2]
[1]Div. of Supercomputing, KISTI
[2]Dept. of Electrical and Computer Engineering, SungKyunKwan Univ.

KREONET 기반의 스토리지 클라우드 서비스 모델 설계 및 성능평가

홍원택[1*,2], 정진욱[2]
[1]한국과학기술정보연구원 슈퍼컴퓨팅본부, [2]성균관대학교 정보통신공학부

Abstract Compared to the commercial networks, R&E networks have the strength such as flexible network engineering and design. Based on those features of R&E networks, we propose our storage cloud service model which supports general-purpose network users in a central region and experimental network users in distributed regions simultaneously. We prototype our service model utilizing multiple proxy controllers of OpenStack Swift service in order to deploy several regions via experimental backbone networks. Our experiments on the influence of the network latency and the size of data to be transmitted show that the bigger size of data is preferable to the smaller size of data in an experimental backbone network where the network latency increases within 10ms because the rate of throughput decline in the bigger object is comparatively small. It means that our service model is appropriate for experimental network users who directly access the service in order to move intermittently high volume of data as well as normal users in the central region who access the service frequently.

• Key Words : storage cloud, swift, R&E network, multiple proxies, OpenStack

요 약 연구망은 상용망과 비교하여 유연한 네트워크 엔지니어링 및 설계 등의 강점을 갖는다. 본 논문은 이러한 연구망의 특성에 기반하여 일반 망 사용자들과 분산된 지역의 첨단 망 사용자들을 동시에 지원하는 스토리지 클라우드 서비스 모델을 제안한다. 첨단 백본 망에 연결된 다수 지역을 적용하기 위해 오픈스택 Swift 서비스의 복수 프락시 컨트롤러를 활용하여 제안 서비스 모델을 프로토타이핑 한다. 망 지연 및 전송 데이터 크기의 영향과 관련한 실험에서 10ms 범위 내의 망 지연이 발생하는 첨단 백본 망에서는 데이터 크기가 상대적으로 큰 데이터가 작은 데이터보다 선호되는 것을 볼 수 있었고, 이것은 큰 데이터에서의 처리 감소율이 작은 데이터에 비해 상대적으로 작은 것에 기인한다. 이러한 실험 결과는 제안 모델이 중앙 지역에서 서비스의 접근 빈도가 잦은 일반 사용자들뿐만 아니라 간헐적으로 대용량 데이터를 전송하기 위해 서비스에 접근하는 첨단 망 사용자들에게도 적합하다는 것을 보여준다.

• 주제어 : 스토리지 클라우드, 스위프트, 연구망, 다수 프락시, 오픈스택

*Corresponding Author : 홍원택(wthong@kisti.re.kr)
Received May 2, 2017 Revised June 2, 2017
Accepted July 20, 2017 Published July 28, 2017

1. Introduction

Cloud computing, which has been recently paid attention to in ICT filed provides resources for users based on the “Pay as you go” concept by utilizing virtualization technique. As a result, it makes to reduce cost and time required for the construction of IT infrastructure. This concept of cloud computing has also been applied to advanced science application filed that requires high performance computing, storage and network resource together. It has helped to improve research environment for application scientists[1,2]. In reality, it makes experiments in advanced science application field such as LHC(Large Hadron Collider) and STAR(Solenoidal Tracker at RHIC) possible through “On-Demand Science” similar to “Pay as you go” concept, which were impossible to be performed because of limited resource capacity and difficulty of building a collaborative research environment[3,4] In particular, the sharing and transmission of high volume of data through the support of network resources in a collaborative research environment is considered as very important research area in order to provide cloud storage resources efficiently in research network communities[5]. It enables collaborative researches of advanced science application communities by procuring their own cloud storage. Consequently, application science users are able to reduce time and cost needed for building individual data storage and solve the problem of securing skilled operators who deal with their own IT infrastructure.

In accordance with these kinds of research network trends, there have been practical approaches to support a collaborative research environment for advanced science application utilizing cloud services in KREONET(Korea Research Environment Open NETwork), one of Korean research networks. According to the service types and properties, these services are able to be categorized into ICT Software tools and virtualized resource infrastructure services in the view of SaaS(Software as a service) and IaaS (Infrastructure as a service) concept[6]. Especially, as there have been frequent collaborations based on high volume of data between research communities, the research issues concerned with network performance have been considered as important factors to support more stable storage cloud service[7,8]. In that sense, our research is also focused on network issues for storage cloud service in research network domains. We analyze the requirements for data sharing and transmission service model over KREONET. Based on those analyses, we propose an appropriate service model in order to support general-purpose network users in a central region and experimental network users in distributed regions simultaneously. After that, we describe the design of our storage cloud service utilizing OpenStack Swift component. In addition, we build the experimental testbed and validate our model through various experimental scenarios. Finally, we discuss the result of experiments performed.

The rest of the paper is organized as follows. In Section 2, we describe the related work including cloud services in research network communities. Section 3 presents our service model designed for the requirements for a storage cloud service over KREONET. Section 4 shows the organization of experimental testbed and scenarios. In Section 5, we analyze and discuss the result of experiments. We conclude in Section 6.

2. Related work

The international research network communities have recently studied network technologies and structures in order to solve the problems concerned when providing cloud services based on their own networks.

OpenCloud project in Internet2 of U.S. aims at providing“Value-Added” cloud services over the research and education network with the collaboration of Internet2 and membership institutes[9]. Especially, XOS which is a new cloud operating system has been studied on the basis of resource virtualization and

SDN(Software Defined Networking). It defines its service management toolkit which simplifies the process of creation, operation, management and composition of services.

GEANT, the research and education network of EU proposes the gOCX(GEANT Open Cloud eXchange) concept [10] and intends to provide necessary network frameworks and functions for support of high quality cloud services between cloud service providers and research network users.

S. Yokoyama and N. Yoshioka [11] point out the weakness of collaboration framework through the federation of multiple clouds in the previous cloud service standardization and studies. They propose the on-demand cloud architecture for cloud collaboration over a community cloud environment which has been studied in NII and SINET, Japan. In this proposal, multiple private clouds are able to be federated horizontally and the utilization rate of cloud resources is maximized by sharing their resource.

Magellan project[12] analyzes the unique requirements of science applications through the DOE(the Department of Energy) ASCR(Advanced Scientific Computing Research) program in US and lists the expected problems when clouds are applied to scientific applications. Especially, the limitations of cloud soft ware such as Eucalyptus, OpenStack and Hadoop are discussed in detail. The experiences in operating a cloud testbed are also provided and the gaps between resource providers and science applications are discussed additionally.

On the other hand, there has been an effort to indicate a limit to the internal network performance of public cloud and to improve overall performance of science applications through network performance testing[13]. Namely, they propose network health parameters and metrics for network connection between cloud instances and apply their proposed model to pCT(proton computed tomography), a medical imaging modality application of e-Science. Finally, they validate their model through a performance evaluation.

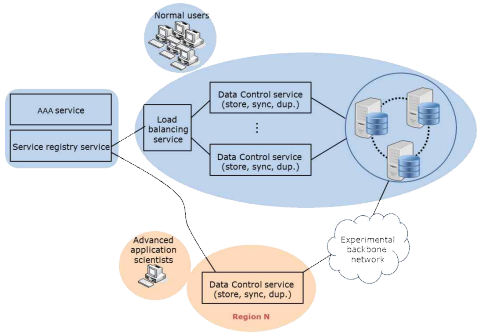
A. Melekhova and V. Vinnikov[14,15] survey the generalized grid models, cloud models and grid- cloud integration paradigm. The study reveals that the virtualization is considered as one of key factors to achieve converged model. Based on the detailed study, the up-to-date efficient management techniques of virtual resources are also provided. Additionally, the algorithmic ideas on memory workload estimation are proposed for nested virtual machines.

N. Nagar and U. Suman[16] propose comparative parameters which are considered when clouds are applied utilizing cloud software such as Eucalyptus, OpenNebula, Nimbus and OpenStack. They provide guidelines for choosing a proper cloud technology according to the application requirements by comparative study based on selected comparative parameters.

3. Proposed service model and use cases

Before we propose our service model, this section describes the requirements for a storage service over R&E network. According to our survey including review of related works in section 2, there are two important requirements. First, it should be highly available, distributed and scalable service. Besides, customizable cloud software is required because we need to consider various user requirements from diverse advanced science user communities. Second, we have to consider R&E network features. Compared to storage cloud services in normal networks, services in R&E networks can distinguishingly utilize the strength of R&E networks such as flexible network engineering and design. Actually, domestic networks in KREONET are composed of a general-purpose network for normal users and an experimental backbone network for advanced application scientists. Regarding general-purpose network users, there is a pattern to frequently access the service in a central region. Compared to them, experimental network users

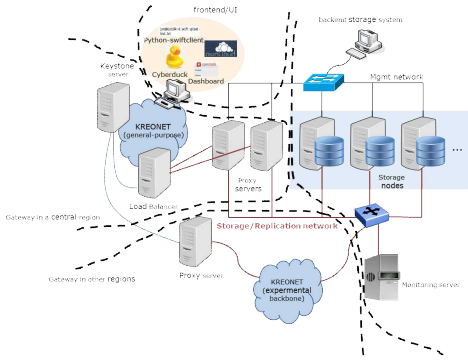
in other regions access the service remotely anticipating high performance networking.



[Fig. 1] Conceptual service model

Figure 1 shows a conceptual service model that reflects the requirements mentioned above. As for the first requirement, it should provide the data control service for a storage cloud including the functionalities such as storing, synchronizing and duplicating of data. OpenStack Swift service can be utilized in order to meet all of these needs. Open source cloud OS, OpenStack software [17] controls pools of compute, storage and networking resources. Major services of OpenStack are Nova computing service and Swift storage service. In particular, Swift service is highly available and scalable storage service for large file sharing and transmission. It also provides metadata which is information about the object and deals with user authentication based on Keystone service. Regarding the second requirement, we support load balancing mechanism in a central region for general-purpose network users. Additionally, we deploy multiple proxy servers of swift service in other regions. It enables high performance guaranteed storage service due to friction-free backbone network.

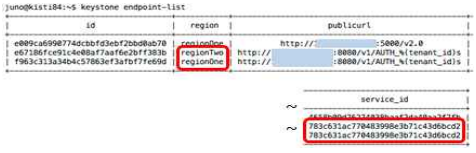
Based on our solutions to main requirements, figure 2 shows the organization of our proposed system which is composed of four main parts. As for backend storage nodes, they keeps at least three replicas for high availability. As more storage capacity is required, additional storage nodes or disks in each node will be easily added and expanded. A proxy server provides a public interface of Swift service. It contacts to a Keystone server to deal with authentication process before entering a Swift service. Each region operates its own proxy server independently and it is registered in the Keystone server that is located in the central region where all storage nodes are deployed. In that case, users of the Swift service in that region do not need to access a load balancer or proxy servers in the central region. The reason why it is possible is that storage and replication networks extended to each region are friction-free and single-hop networks. All storage nodes in figure 2 are located in a single domain and several proxy servers work together with them. On that point, a load balancer that coordinates multiple proxy servers is necessary. In most of the use cases, clients will use one of several proxy servers via a load balancer. Therefore, it is configured to distribute requests from clients according to its own load balancing policy such as round-robin, static-rr and least-connection.



[Fig. 2] Proposed service deployment

Figure 3 shows captured image performed by OpenStack Swift CLI in order to explain the use cases of our proposed system. There is multiple endpoints list of a Keystone service in the central site in figure 2. Two endpoint entries have the same service id and they belong to different regions respectively. In detail, first entry represents a service id of a load balancer in the central site and the other one is a service id of a proxy server in a remote site. Regarding the first

scenario for normal users in the view of service users, they contact a load balancer via a Keystone server in the central site. According to the policy of load balancer, the requests are transmitted to one of proxy servers which were already registered in the Keystone server. After that, that proxy server does CRUD(Create, Read, Update and Delete) actions through backend storage nodes and returns the result. On the other hand, regarding the second scenario for scientific application users in other regions, they access a Swift service via an experimental backbone network and contact a proxy server which is located in their region. They do not suffer from the last one mile problem if only they are able to directly access their nearby regional networks. Even if backend storage nodes are not located in their region, their requests are transmitted within a very short time. In reality, the RTT(Round-Trip Time) between Seoul and Daejeon is less than 3ms.

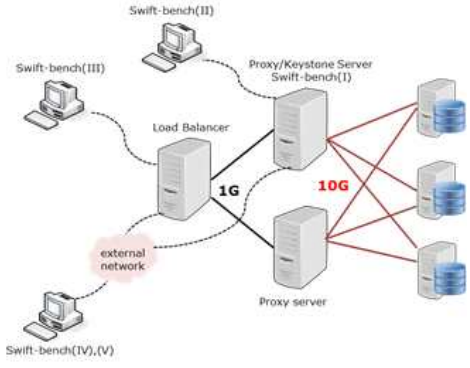


[Fig. 3] Multiple swift service endpoints

4. Experiments

In this section, we perform two kinds of experiments in order to verify our proposal. First, according to the location of a client and whether or not a load balancer exists, we estimate the performance of our proposed system. Second, we examine the performance degradation in accordance with the network latency between a proxy server and storage nodes. The second experiment is related to simulate the situation that the network delay might happen when a proxy server is connected to storage nodes via an experimental backbone network.

With regard to building the experimental testbed, we use OpenStack Juno version which was released October 2014.Different from other sub projects in OpenStack, Swift service is relatively independent of others. In order to authenticate Swift users, Keystone component is usually used together with a Swift service. A proxy server connects between swift service and clients. When uploading and downloading object files, it identifies which drives belong to which Swift partition. As fora storage node server, it keeps account, container and object ring files which relates to information that will be stored for accounts, containers and objects. All object servers which create 1024 partitions have 12 Intel Xeon E5-2620 cores running at 2.40GHz, 96GB of main memory and two 6TB disk drives for three replicas. HAProxy[18] is used to support load balancing function for multiple connections. It is a popular open source program which distributes TCP/HTTP workload among several servers. Swift-bench[19] plays a role in a bench marking tool for swift cluster. It generates workload of PUT, GET and DEL operations and estimates the number of average requests per second per operation. It also provides various configuration parameters such as total client concurrency, the size of objects, the number of objects and the number of GETs that we can create different experimental scenarios.



[Fig. 4] Experiment-I testbed topology

(Experiment-I) Figure 4 shows the testbed topology for the first experiment. There are several experimental scenarios depending on the location of a client, namely a swift-bench process and whether or not a load

balancer is.

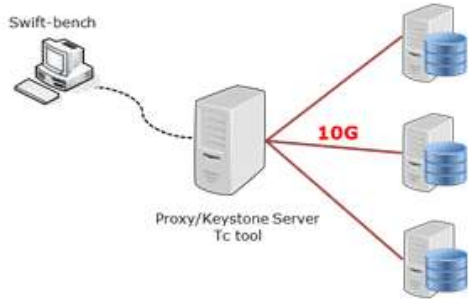
In more detail, we can categorize the experimental cases as shown in table 1.In this experiment, we assume that a keystone server exists with a proxy server machine and there are all OpenStack components including a load balancer are deployed in the same network. Additionally, a client is located in a proxy server, in the same network with a proxy server or in the different network with a proxy server. According to the combination of them, we can get five different experimental cases. As for each case, we generate workload of PUT/GET/DEL operations while changing the object size from 10 bytes to 100Mbytes. The requests from the process of swift-bench are generally handled as follows: First, when the request is executed, a keystone server authenticates that request. After that, a load balancer accepts it and it is continuously sent to one of proxy servers. The selected proxy server propagates operation commands to three storage nodes and returns the result in reverse.

<Table 1> Categorization of experimental cases

	Client in a proxy server	Client in the same network with a proxy server	Client in the different network with a proxy server
No LB exists	(I)	(II)	(IV)
LB exists	n/a	(III)	(V)

(Experiment-II)Figure 5 shows the testbed topology for the second experiment. Compared to the first experiment, the second one is performed in a relatively simple way. This experiment simulates the decrease in performance in proportion to network latency between a proxy server and storage nodes over an experimental backbone network. We need to do these kinds of experiments in a similar environment before real deployment. The important parameters to give influence to the performance of swift cluster if deployed over the experimental backbone network are network latency and the data size of object files to be moved.

We utilize swift-bench and TC tool in order to make the simulation environment. In a real deployment, the network delay occurs in the connection between a proxy controller and storage nodes because that connection is created in a long distance network. As though it is quite long distance between two regions over KREONET, the RTT between them has been known as within 10 milliseconds. TC tool [20] makes network delay artificially and adds rules to netem which is a kernel component for simulating the properties of WAN(Wide Area Network). In our experiment, it is used to simulate the network delay between two regions which will be deployed in experiment al back bone networks. Similar to the first experiment, swift-bench is used to generate PUT/GET/DEL operations over the test be das shown in figure5. Different from the first experiment, we fix the location of swift-bench as where the proxy server is. We can get various experimental results depending on the combination of the object size and network latency.



[Fig. 5] Experiment-II testbed topology

5. Analysis and discussion

In this section, we present the results of experiments mentioned in a previous section and discuss the meaning of them.

(Experiment-I) We performed several kinds of experiments to verify the influence of the location of a swift client and whether or not a load balancer is. Because a swift-bench program is selected as a client tool, we can get the result of PUT/GET/DEL operations sequentially. Based on the categorization in

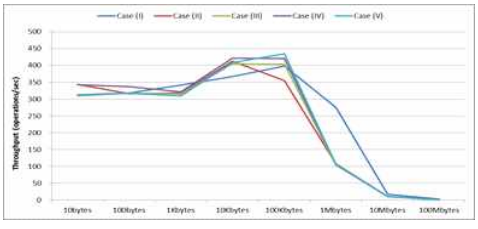
a table 1, each experimental case is repeatedly executed with the different size of objects from 10bytes to about 100Mbytes. The other parameters which are used in a swift-bench are as follows: The number of objects is 100 and the number of GETs is200.

Overall, the performance of all cases is almost identical as shown in from figure 6 to figure 8 as for PUT/GET/DEL operations. If observed more closely, we can see better performance of Case (I)than other cases regarding GET/DEL operations. Intuitively, it is able to be explained by the fact that the client request is created in the nearby location from a proxy server and storage nodes, namely in same network boundary.

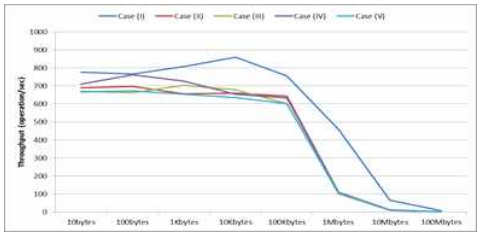
As for the performance degradation of PUT/GET operation in the sixth stage (1Mbytes), it means that the performance of our testbed decreases sharply from about 1Mbytes, the data size of object. On the other hand, regarding DEL operations, there is no influence from the data size of object in all cases.

In PUT operations, the best performance occurs in the fourth and fifth stages (10Kbytes and 100Kbytes). It means that the proper data size which is not too small is rather helpful to improve the throughput of a swift cluster system because the performance is directly related to the frequency of the generation of requests to proxy servers or load balancers.

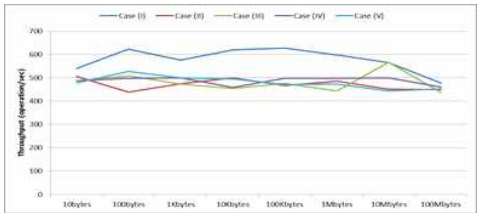
In short, several experimental cases based on where a swift client is located and whether or not a load balancer is show that a load balancer for the distribution of workload is not critical to the performance of our swift cluster testbed and the location of a swift client such as a swift-bench program could have an effect on the throughput of GET/DEL operations.



[Fig. 6] Throughput of PUT operation in Experiment-I



[Fig. 7] Throughput of GET operation in Experiment-I

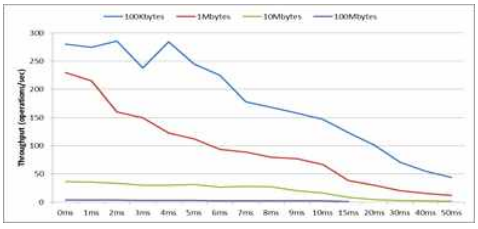


[Fig. 8] Throughput of DEL operation in Experiment-I

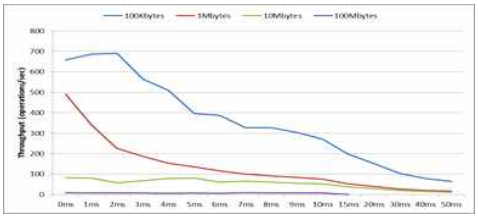
(Experiment-II) In this subsection, we analyze the result of experiment-II. In order to investigate on the influence of network latency and the data size of object files, each experimental case is repeatedly performed with the different network latency per the different size of objects from 100Kbytes to 100Mbytes. The other parameters which are used in a swift-bench are as follows: The number of objects is 100. The number of GETs is 200.

Figure 9,10 and 11 show the throughput of PUT/GET/DEL operations in the different data size of objects while swift-bench is executed with the different network delay values from 0ms to 50ms. As the network latency increases, the throughput of all operations decreases as expected. As for PUT/GET operations, the rate of throughput decline in the first and second cases (100Kbytes and 1Mbytes) is bigger than that in the third and fourth cases (10Mbytes and 100Mbytes).It means that as the network delay in the smaller size of object files increases, the throughput of swift cluster system has more influence than in the bigger size of object files. Based on that observation, we can find out that the bigger size of object data is preferable to the smaller size of object data in an experimental backbone network where the network latency increases. As for the fourth case (100Mbytes),

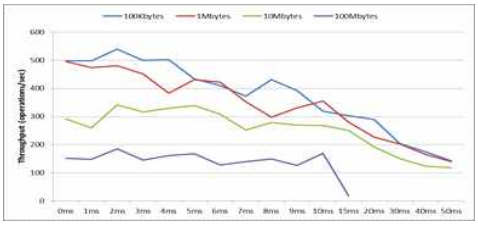
we could not get the meaningful throughput of operations from more than 15ms network latency because the swift-bench was not executed well in that condition. On the other hand, the ratio of throughput decrease of DEL operation is smaller than that of PUT/GET operations in the size of all objects. It means that DEL operation is less sensitive to network latency than PUT/GET operations. It can be inferred that the DEL operation require less traffic volume than PUT/GET operations without regard to the object size.



[Fig. 9] Throughput of PUT operation in Experiment-II



[Fig. 10] Throughput of GET operation in Experiment-II



[Fig. 11] Throughput of DEL operation in Experiment-II

6. Conclusion

As the collaboration between research communities utilizing high volume of data is frequent, to support storage cloud service has been a very important role in R&E networks. In this paper, we propose our storage cloud service model based on the requirements for a storage service over R&E network. Compared to storage cloud services in normal networks, services in R&E networks can distinguishingly utilize the strength of R&E networks such as flexible network engineering and design. Utilizing those features of R&E networks, it is designed to support general-purpose network users in a central region and experimental network users in distributed regions simultaneously. It is based on load balancing mechanism and multiple proxy servers of OpenStack Swift service that enables highly available, distributed and scalable service.

We have built the experimental testbed for our service model and evaluated it with several experimental scenarios. Our evaluation has shown that the location of a swift client and whether or not there is a load balancer is not critical factor to the performance for our storage service. In addition, our experiments on the influence of network latency and the size of data to be transmitted show that the bigger size of object data is preferable to the smaller size of object data in an experimental backbone network where the network latency increases because the rate of throughput decline in the bigger object is comparatively small. It means that our service model is appropriate for experimental network users who directly access the service via their remote backbone network in order to move intermittently high volume of data as well as normal users in the central regionwho access the service frequently.

REFERENCES

[1] J. Balewski et al., Offloading Peak Processing to Virtual Farm by STAR Experiment at RHIC. Journal of Physics: Conference Series, 368(2012):012011, 2012.

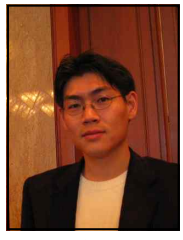
[2] R. P Taylor et al., The Evolution of Cloud Computing in ATLAS.Journal of Physics: Conference Series, 664(2015):022038, 2015.

[3] M. Parashar, M. Abdelbaky, I. Rodero and A. Devarakonda, Cloud Paradigms and Practices for Computational and Data-Enabled Science and

Engineering. Computing in Science & Eng., vol. 15(4), 2013, pp. 10-18.
[4] K. Keahey and M. Parashar, Enabling On-Demand Science via Cloud Computing. IEEE Cloud Computing, May 2014, pp.21-27.
[5] D. Yuan, L. Cui and X. Liu, Cloud Data Management for Scientific Workflows: Research Issues, Methodologies, and State-of-the-Art. IEEE International Conference on Semantics, Knowledge and Grids, Aug. 2014.
[6] NIST Cloud Computing Program, http://www.nist.gov/itl/cloud/.date accessed:24/10/2016.
[7] Y. Liu, V. Vlassov and L. Navarro, Towards a Community Cloud Storage. IEEE International Conference on Advanced Information Networking and Applications, May, 2014, pp.837-844.
[8] Jung-Yul Choi, "A Study on Networking Technology for Cloud Data Centers", Journal of digital Convergence, Vol. 14, No. 2, pp. 235-243, 2016.
[9] OpenCloud, http://www.opencloud.us/. date accessed: 24/10/2016.
[10] GEANT project white paper, Milestone MS101 (MJ1.2.1): Network Architectures for Cloud Services. Mar. 2014.
[11] S. Yokoyama and N. Yoshioka, On-demand Cloud Architecture for Academic Community Cloud - Another Approach to Inter-cloud Collaboration. 4th International Conference on Cloud Computing and Services Science,2014, pp.661-670.
[12] L. Ramakrishnan et al., Magellan: experiences from a science cloud. Proceedings of the 2nd international workshop on scientific cloud computing, Jun. 2011, pp.49-58.
[13] R. Chard, K.Bubendorfer and B. Ng,Network Health and e-Science in Public Clouds. IEEE 10th International Conference on e-Science, Oct. 2014, pp.309-316.
[14] A. Melekhova and V. Vinnikov, Cloud and Grid Part I: Difference and Convergence. Indian Journal of Science and Technology, vol. 8(29), Nov. 2015.
[15] A. Melekhova and V. Vinnikov, Cloud and Grid Part II: Virtualized Resource Balancing. Indian Journal of Science and Technology, vol. 8(29), Nov. 2015.
[16] N. Nagar and U. Suman, Architectural Comparison and Implementation of Cloud Tools and Technologies. International Journal of Future Computer and Communication vol. 3(3), Jun. 2014, pp.153-160.
[17] OpenStack, http://www.openstack.org/. date accessed: 24/10/2016.
[18] HAProxy, http://www.haproxy.org/. date accessed: 24/10/2016.
[19] M. Lanner and D. Bishop, Benchmarking. In: OpenStack Swift, O'Reilly, 2014, pp.273-298.
[20] TC tool, https://wiki.linuxfoundation.org/networking/netem. date accessed: 24/10/2016.

저자소개

홍 원 택(Wontaek Hong) [정회원]



- 2000년 : 성균관대학교 일반대학원 전기전자 및 컴퓨터공학과 (석사)
- 2011년 : 성균관대학교 일반대학원 전자전기컴퓨터공학과 (박사과정수료)
- 2000년 ~ 2002년 : ㈜콤텍시스템 기술연구소 연구원
- 2002년 ~ 현재 : 한국과학기술정보연구원 선임연구원

<관심분야>
망 관리 및 성능 분석, 소프트웨어 정의 네트워킹, 클라우드 관리 플랫폼

정 진 욱(Jinwook Chung) [정회원]

- 1991년 : 서울대학교 대학원 전자계산학과 (박사)
- 1973년 ~ 1985년 : 한국과학기술연구소 실장
- 1992년 ~ 1993년 : 미국 Maryland 대학교 객원교수
- 1985년 ~ 현재 : 성균관대학교 정보통신공학부 교수
- 2007년 ~ 2008년 : 성균관대 일반대학원장

<관심분야>
컴퓨터 네트워크, 네트워크 프로토콜, 인터넷 윤리