Design and Evaluation of Intercloud Systems and Mobile Intercloud Applications

Yik Him Ho, Chung Kwan Law, Yi Dou, and Henry C. B. Chan

***Abstract*—With the advent of cloud computing, there is a potential need for the interconnection of clouds (i.e., Intercloud). Intercloud seeks to connect heterogeneous clouds together to form a network of clouds. As an extension to our previous work, this paper gives an overview of an Intercloud system with a focus on the Intercloud Gateways and Intercloud communications protocol. To analyze the performance of the Intercloud systems, we have conducted experiments under different settings and scenarios. Furthermore, we have evaluated the Intercloud system to support a mobile Intercloud application.**

***Index Terms*—cloud computing, Intercloud, mobile cloud computing, mobile Intercloud**

I. Introduction

CLOUD computing has become an important field in recent years. With cloud computing, a variety of computing resources (e.g., hardware, software, and storage) can be provided as services over the Internet [1] [2]. In general, there are three service models for cloud computing: Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS), and Platform-as-a-Service (PaaS) [3] [4]. By using cloud computing, operating and maintenance costs can be significantly reduced. For example, a company can set up virtual servers and use computing resources on an on-demand basis. This allows the provision of computing resources (e.g., storage and processing power) as utility services similar to electricity services. Indeed, computing resources situated in different servers across the Internet can work in a collaborative manner. In other words, they become shared distributed resources, which can be employed collectively to serve user requirements in a dynamic fashion [5]. While these services are typically provided through public clouds, other clouds, including private clouds, community clouds, and hybrid clouds, can also be set up depending on user requirements.

When many clouds are set up, there is a need for them to collaborate and communicate (e.g., to enable heterogeneous cloud service providers to interact with each other) [6]. Here, cloud-to-cloud communications and collaboration refers to the interaction between clouds of the same cloud provider as well as between clouds of different cloud providers. For instance, one potential application would be the collaboration of cloud-based data storage systems. However, despite the popularity of cloud computing, the development of cloud-to-cloud service support is still in its infancy. In particular, there are currently limited well-defined and standardized methods (and implementations) for achieving inter-cloud operability. The IEEE has been developing the Intercloud Standard (P2302), and has made a draft standard available [7]. The standard defines an Intercloud architecture that resembles the architecture of the Internet. Various system components, such as Intercloud roots, Intercloud Gateways, and Intercloud exchanges, have been proposed to facilitate Intercloud operations. In general, this paper is based on the Intercloud framework defined by the IEEE P2302 framework. This framework facilitates the development of Intercloud systems for different Intercloud applications [8] [9] [10].

Manuscript received January 8, 2019; revised February 11, 2019. This work was supported in part by the Department of Computing, The Hong Kong Polytechnic University, and by the Innovation and Technology Fund of the Hong Kong Special Administrative Region Government and CITIC Telecom International CPC Limited.

Yik Him Ho, Chung Kwan Law, Yi Dou and Henry C. B. Chan are with the Department of Computing, The Hong Kong Polytechnic University (e-mail: csyhho@comp.polyu.edu.hk).

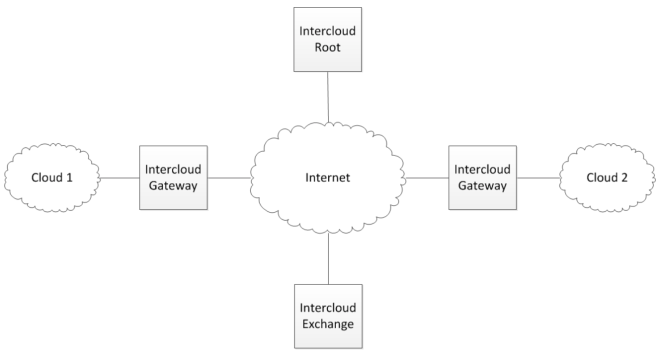

Fig. 1. Basic Intercloud architecture

As shown in our previous work [11], we have designed an Intercloud Gateway as well as an extensible Intercloud Communications Protocol (ICCP) for supporting Intercloud operations. Using the Intercloud Gateway, either a cloud user or cloud provider can securely and effectively move their files or virtual machines from one cloud to another cloud without client involvement. As a continuation of our previous work on designing an extensible ICCP and developing an Intercloud Gateway application to support Intercloud communications [8], we analyze the performance of the Intercloud Gateway in this paper.

The rest of the paper is organized as follows. Section II presents a brief review of ICCP. Section III presents the design and functions of the Intercloud Gateway. Section IV presents and discusses the experimental results. Section VI presents the conclusion.

II. Intercloud Communications Protocol (ICCP)

As shown in Fig. 1, the system architecture consists of the following key components: Intercloud Roots, Intercloud Exchanges, and an Intercloud Gateway to support Intercloud operations:

- Intercloud Roots: These servers function like DNS root servers on the Internet. They provide a directory service and facilitate queries for cloud resources. Furthermore, they can also function as a root certificate authority.
- Intercloud Exchanges: They function like Internet Exchanges. Each cloud should be associated with an

Intercloud Exchange. The Intercloud Exchanges facilitate communications and resource matching among the clouds.
- Intercloud Gateways: They function like Internet routers, allowing Intercloud communications based on XML messages.

ICCP specifies how two clouds can communicate or interact with each other. It is inspired by HTTP and designed based on XML. Each Intercloud operation is defined as a command (e.g., put/get data object) in ICCP. As XML is highly extensible, new Intercloud operations can easily be added by defining new commands. Similar to HTTP, ICCP comprises request and response messages, and each message is defined by XML tags (such as HTTP headers).

To facilitate Intercloud operations, Intercloud Gateways communicate with each other using XML-based request/response messages. In each request/response message, the protocol version and session are specified by the “Version” and “ID” attributes, respectively. Various information tags (e.g., GeneralInfo, RequestInfo, ResponseInfo etc.) can be defined in a request or response message in order to provide specific information for the requested command. An example of the XML messages can be found in the next section.

III. Intercloud Gateway

As presented in our previous paper [11], based on ICCP, we have developed an Intercloud Gateway to support Intercloud communications. In other words, an Intercloud Gateway is designed to handle and process ICCP requests/responses. An Intercloud Gateway connects to a cloud’s object storage and/or virtual machine (VM) hypervisor to perform various Intercloud operations by using the corresponding cloud service provider’s API methods.

A. Basic design

The Gateways communicate with each other through sockets. When a Gateway receives a request/response message, the message will be stored in a message queue first. Based on the command in a message, the message will be forwarded to the corresponding command module for processing. Upon processing, a response message will be sent to the requesting cloud’s Gateway.

B. Pre-defined API classes

The Gateway includes a number of pre-defined API classes for common Intercloud operations, such as transferring data objects and controlling VMs. The aforementioned API classes allow developers to develop tailor-made Intercloud applications for individual needs. A set of JavaDoc descriptions of the API classes and methods can be found at http://iccp.cf.

C. Commands and functions for object storage

Currently, the Gateway supports Amazon S3, Google Cloud Storage, Microsoft Azure and Minio object storage service. Referring to the specification of ICCP, the following is a list of the common commands for object storage, along with the corresponding Object Storage API methods and a brief description:

- ListObject: generated by API’s list() method, which queries the target cloud for a list of data object stored.
- PutObject: generated by API’s put() method, which requests the target cloud to receive a data object and store it.
- GetObject: generated by API’s get() method, which requests the target cloud to return a previously put data object.
- ForwardObject: generated by API’s forward() method, which requests the target cloud to forward a data object to another cloud.
- DeleteObject: generated by API’s delete() method, which requests the target cloud to delete a previously put data object.

The following shows an example:

```
<Request Version="1.0" ID="647352553">
  <GeneralInformation From="c1.e1.r1.iccp.us"
      To="c2.e1.r1.iccp.us" Date="2019-01-12
     " Time="16:36:00"/>
  <RequestInformation Service="ObjectStorage"
      Command="PutObject">
    <ObjectName>bigfile1</ObjectName>
    <TransferMethod>UDT</TransferMethod>
  </RequestInformation>
  <AdditionalInformation>
    <IP>123.123.123.123</IP>
    <Port>9000</Port>
    <Path>path/to/file</Path>
    <Encoding>Base64</Encoding>
    <DataSecurity>Shared</DataSecurity>
    <SharedKey>2efs2esfsghtjdrus</SharedKey>
    <DataDigestAlgorithm>SHA256</
        DataDigestAlgorithm>
    <DataDigest>asdikj234mnd12</DataDigest>
    <Overwrite>True</Overwrite>
  </AdditionalInformation>
</Request>
```

In this example, ‘c1.e1.r1.iccp.us’ requests ‘c2.e1.r1.iccp.us’ to receive data via UDT, decrypt it with the shared key and store it, and overwrite all previous versions. The data are presented as a file downloadable from the UDT server at 123.123.123.123:9000 with the path ‘path/to/file’. Such data are presented using Base64 encoding and are named ‘bigfile1’.

D. Commands and functions for virtual machine

Currently, the Gateway supports VMware and Hyper-V hypervisors. Referring to the specification of ICCP, the following is a list of the common commands for VM along with the corresponding VM API methods and a brief description:

- CreateVM: generated by API’s createVM() method, which requests the target cloud to create a new VM.
- PutVM: generated by API’s putVM() method, which requests the target cloud to receive a VM image and host it.
- GetObject: generated by API’s getVM() method, which requests the target cloud to export a previously put VM and send the exported image to it.
- VMPowerControl: generated by API’s powerControl() method, which requests the target cloud to power on, power off, suspend or reset a VM.
- GetVMDetails: generated by API’s getVMDetails() method, which queries the target cloud for the details of a VM being hosted.

- ListVM: generated by API's listVM() method, which queries the target cloud for a list of VMs being hosted.

IV. Performance Analysis

In this section, we present the performance analysis of the Intercloud system focusing on the transfer of data objects and VMs between clouds. In particular, we evaluate the sub-processes and study different ways to improve efficiencies, such as using multi-threading and data compression. For the VM transfer, we studied three different cloud service providers (CSPs) for the performance analysis:

- International cloud provider 1, located in the U.S.-West
- International cloud provider 2, located in Europe
- Local cloud provider, located in Hong Kong

A. *Processing time for individual sub-processes*

The purpose of this experiment is to evaluate the processing time of each sub-process in an Intercloud transaction. Since transferring a VM can be considered to be transferring a large data object, we only accessed the data object transfer time in this experiment.

1) Experimental settings: The operation of object transfer can be divided into individual sub-processes. For example, we divided the operation of 'PutObject' into the following sub-processes: (1) the owner Gateway downloads the object from its storage, (2) the owner Gateway encrypts the object, (3) the owner Gateway computes the digest of the object, (4) cross-cloud data transfer, (5) the target Gateway verifies the object, and (6) the target Gateway uploads the object to its storage. Similarly, we divided the operation of 'GetObject' into the following sub-processes: (1) the target Gateway downloads the object from its storage, (2) the target Gateway computes the digest of the object, (3) cross-cloud data transfer, (4) the owner Gateway verifies the object, (5) the owner Gateway decrypts the object, and (6) the owner Gateway uploads the object to its storage.

We used three cloud VMs provided by each CSP to perform the experiments. All VMs had the same machine specifications of 2 CPUs, 4GB RAM, 2GB Java default heap and 2GB Java max heap. Different sizes of data objects were tested, including 5MB, 10MB, 20MB, 50MB, 100MB, 200MB, 512MB and 1024MB.

2) Experimental results: Fig. 2, 3 and 4 illustrate the mean processing time, the percentage of time spent and the mean processing speed of individual sub-processes of a 'PutObject' operation. Fig. 5, 6 and 7 illustrate the results for a 'GetObject' operation. The results show that when transferring a data object through the Intercloud Gateways, cross-cloud data transfer accounts for the largest amount of time. However, when the data object size is larger, the processing time for uploading a data object to storage becomes more significant, while the time percentage for cross-cloud data transfer gradually decreases. Furthermore, when the data object size is larger, there is a steady increase in the processing speed of cross-cloud data transfer.

B. *Effectiveness of multi-threading*

Many applications today employ multi-threading to enhance overall performance. In this experiment, we investigate the performance of the following operations processed by the Gateway: (1) downloading an object from storage, (2) uploading an object to storage, (3) data encryption, (4) data decryption, and (5) digest calculation. Our aim is to evaluate the effectiveness of concurrency with different thread limits and machine specifications. In particular, we examine the limit of parallel transfer operations among CSPs with various machine specifications.

1) Experimental settings: In each experiment, 100 copies of a 1GB object were processed. Each process was tested for 5 times, with different limits of executing threads including 1, 10, 25, 50, and 100.

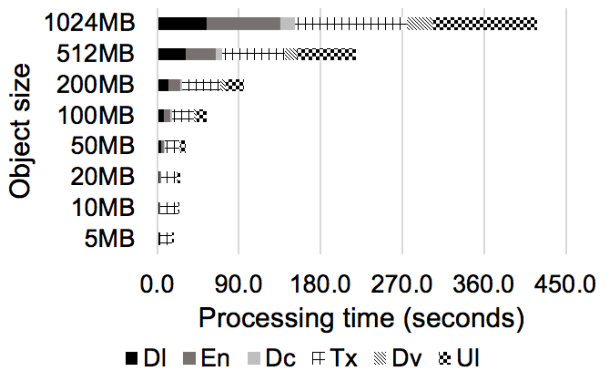

Fig. 2. Mean processing time of PutObject sub-processes. Dl = downloading the object by the owner Gateway, En = encrypting the object by the owner Gateway, Dc = computing the digest by the owner Gateway, Tx = Intercloud data transfer, Dv = object verification by the target Gateway, Ul = uploading the object to storage by the target Gateway

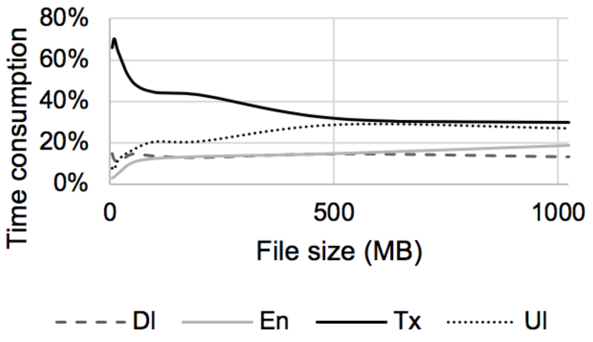

Fig. 3. Mean % of time consumption of PutObject sub-processes. Dl = downloading the object by the owner Gateway, En = encrypting the object by the owner Gateway, Tx = Intercloud data transfer, Ul = uploading the object to storage by the target Gateway

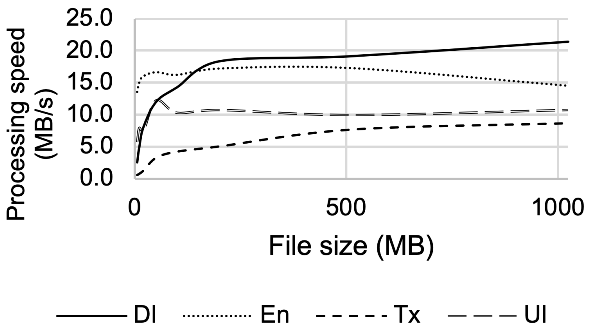

Fig. 4. Mean processing speed of PutObject sub-processes (MB/s). Dl = downloading the object by the owner Gateway, En = encrypting the object by the owner Gateway, Tx = Intercloud data transfer, Ul = uploading the object to storage by the target Gateway

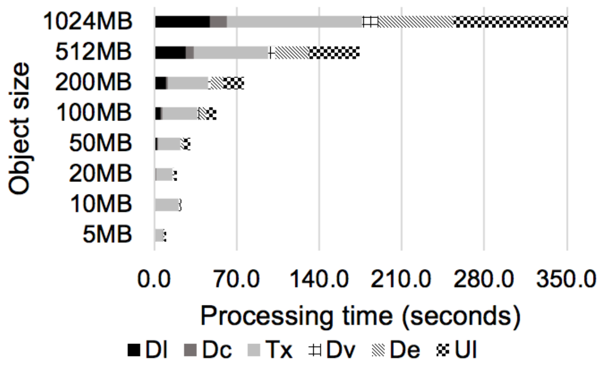

Fig. 5. Mean processing time of GetObject sub-processes. Dl = downloading the object by the target Gateway, Dc = computing the digest by the target Gateway, Tx = Intercloud data transfer, Dv = decrypting the object by the owner Gateway, De = object verification by the owner Gateway, Ul = uploading the object to storage by the owner Gateway

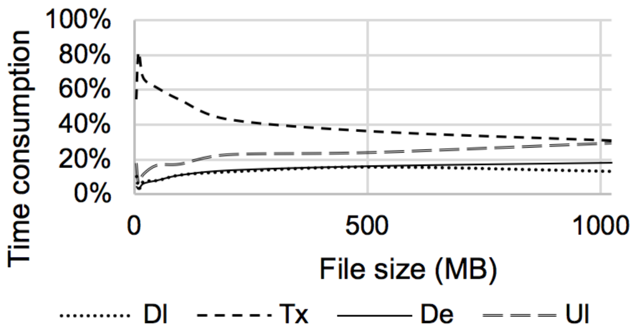

Fig. 6. Mean % of time consumption of GetObject sub-processes. Dl = downloading the object by the target Gateway, Tx = Intercloud data transfer, De = object verification by the owner Gateway, Ul = uploading the object to storage by the owner Gateway

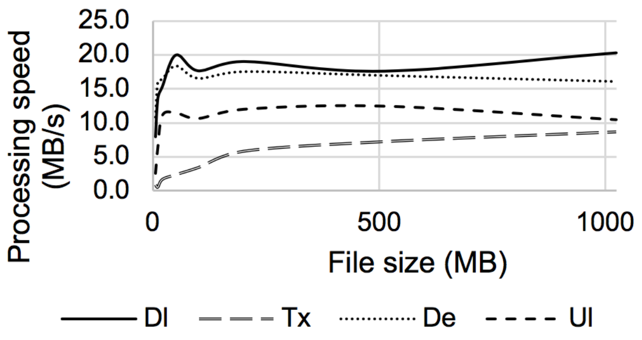

Fig. 7. Mean processing speed of GetObject sub-processes (MB/s). Dl = downloading the object by the target Gateway, Tx = Intercloud data transfer, De = object verification by the owner Gateway, Ul = uploading the object to storage by the owner Gateway

2) Experimental results: For all sub-processes, the results indicate that when the number of executing threads exceeds 10, no obvious performance improvement can be found, i.e., no significant reduction in total processing time, as shown in Fig. 8 using 'data decryption' as an example. For the sub-processes of uploading or downloading, setting the thread limit to more than 10 might result in exceptions. Such exceptions were thrown by the CSPs' API and were likely caused by the network connection between the object storage server and the Gateway. In other words, there will be a 100% successful rate for uploading or downloading when the thread limit is 10 or below.

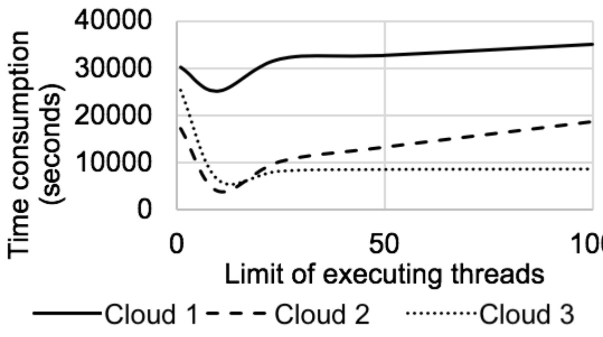

Fig. 8. Mean time consumption of decryption on difference cases

C. Effectiveness of data compression

In theory, the performance of transferring a large size of data object can be enhanced if data compression is used. In order to investigate how well data compression can enhance Intercloud data object transfer, we conducted experiments to analyze its efficiency and effectiveness for cross-region, cross-cloud data transfer. In the analysis, we tested the process of normal and compressed, object transfer and VM migration, separately. Note that our analysis focused on the data transfer. Hence, we only recorded the time usage of the concerned sub-processes.

1) Experimental settings: For the data object transfer, we tested with the following sample objects:

- Obj. 1: a text file filled with the character 'A' (4104MB)
- Obj. 2: a text file filled with random characters (4104MB)
- Obj. 3: a Windows 2012 setup DVD ISO (3523MB)
- Obj. 4: an MP4 HD video file (4462MB)

For VM migration, we tested with the following sample VM disk images. These images were 'freshly-installed' with operating systems:

- Win8.vmdk: a Windows 8 image hosted on VMware
- Win12.vmdk: a Windows 2012 image hosted on VMware
- Cent6.vmdk: a CentOS 6 Linux image hosted on VMware
- Win8.vhdx: a Windows 8 image hosted on HyperV
- Win12.vhdx: a Windows 2012 image hosted on HyperV
- Cent6.vhdx: a CentOS 6 Linux image hosted on HyperV

2) Experimental results - object transfer: In a normal 'PutObject' operation, we recorded the time used for (1) downloading the object from the storage, (2) transferring the object to the target cloud, and (3) uploading the object to the target storage. In a compressed 'PutObject' operation, we recorded the time used for (1) downloading the object from the storage, (2) compressing the object, (3) transferring the compressed object to the target cloud, and (4) uploading the compressed object to the target storage.

Similarly, in a normal 'GetObject' operation, we recorded the time used for (1) downloading the object from the target storage, (2) transferring the object from the target cloud, and (3) uploading the object to the storage. In a compressed 'GetObject' operation, we recorded the time used for (1) downloading the compressed object from the target storage, (2) transferring the compressed object from the target cloud,

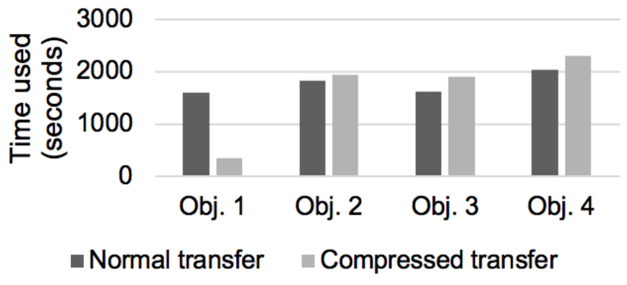

Fig. 9. PutObject: mean time used

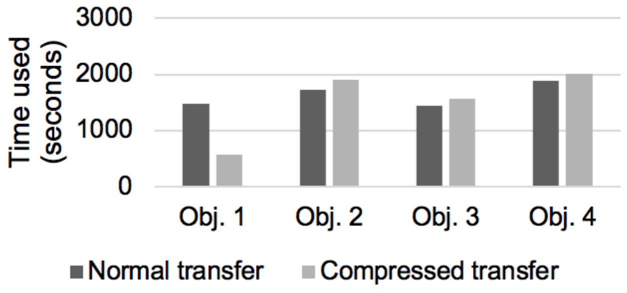

Fig. 10. GetObject: mean time used

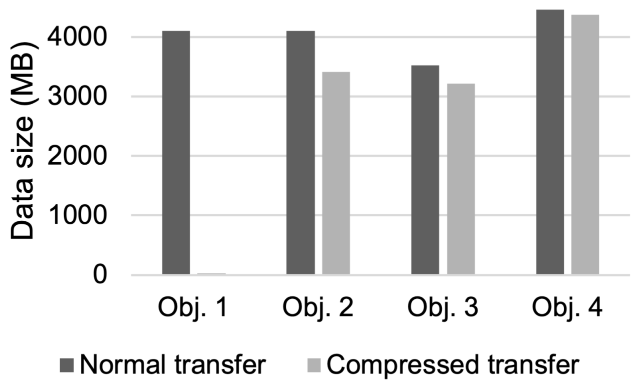

Fig. 11. Object transfer: object size

(3) extracting the object, and (4) uploading the object to the target storage.

Fig. 9 and 10 show the mean time used for PutObject and GetObject, respectively. Fig. 11 compares the sizes of normal objects and compressed objects. Note that the compression ratio of Obj. 1 is very large, therefore it is too small (4MB) to be read in the figure. The results show that the time required for PutObject and GetObject depends on the compression ratio. If the compression ratio for an object is low, compression is not beneficial because it takes more time to compress the object. That means, the time required (i.e., compression time plus transmission time) is even longer (see objects 2, 3 and 4 in the example).

3) Experimental results - VM migration: For normal transfer, we recorded the time usage for transferring the VM disk images. For compressed transfer, we recorded the time used for (1) compressing the VM disk image, (2) transferring the compressed image to the opposite cloud, and (3) extracting the image. Fig. 12 shows the mean time used to transfer the VMs (i.e., including the aforementioned steps). Fig. 13 compares the size of the VMs both without compression and with compression. The results show that

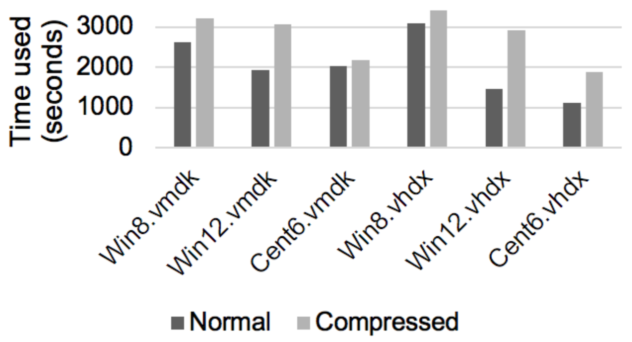

Fig. 12. VM migration: mean time used

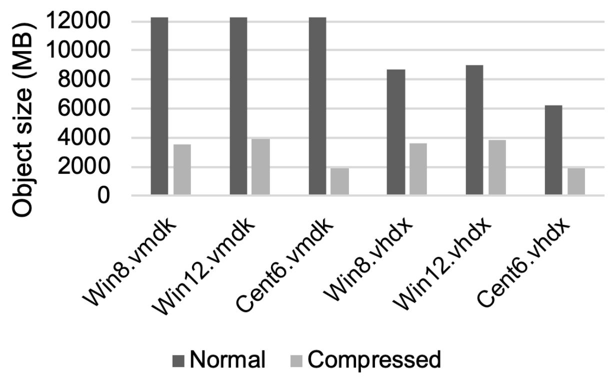

Fig. 13. VM migration: object size

transferring a VM without compression is actually faster, mainly because it requires significant time to compress the VM.

4) Conclusion: In object transfer, performance enhancement (i.e., reduction in processing time) by data compression is mostly related to the content of the object. Furthermore, performance enhancement by data compression in VM migration is in general not effective. Nevertheless, data compression is still preferred in many cases, because the network usage cost can be saved by transferring smaller objects.

V. Mobile Intercloud Applications Data Transfer Scenario

The Intercloud system can also facilitate the development of mobile Intercloud applications [12]. Mobile Intercloud system is an extension of the Intercloud system to a mobile environment with the aim of providing cloud computing services through heterogeneous clouds. In particular, it seeks to allow mobile devices to manage data and virtual machines across different or heterogeneous cloud platforms. Basically, a mobile device is associated with a home cloud. The corresponding virtual mobile terminal, data and applications can be stored in the home cloud. Whenever required, data and applications can be transferred between clouds based on ICCP. For example, to enhance system performance (i.e., access time), it is desirable to move data closer to the mobile device [13].

This section studies a mobile Intercloud scenario using the Intercloud Gateway to illustrate the advantages of the mobile Intercloud system. Assume that a mobile user stores his/her data using a cloud-based data object storage system (e.g.,

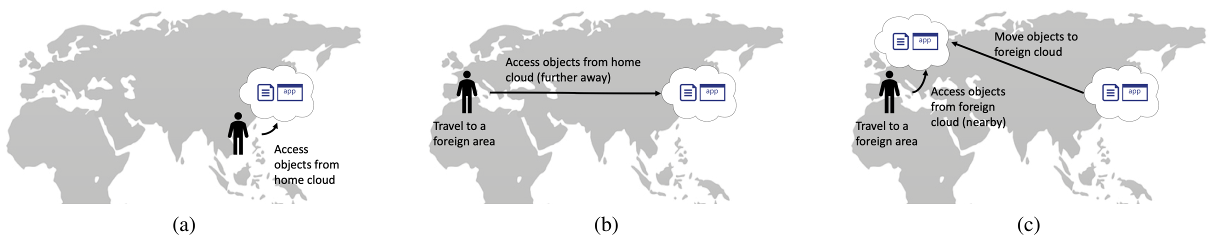
app
Access objects from home cloud
(a)
Access objects from home cloud (further away)
Travel to a foreign area
(b)
Move objects to foreign cloud
Access objects from foreign cloud (nearby)
Travel to a foreign area
(c)

Fig. 14. Mobile Intercloud scenario. (a) A user accesses data objects from the home cloud. (b) When the user travels to a foreign area, the home cloud is further away. (c) To enhance access time, data objects can be transferred to a nearby foreign cloud.

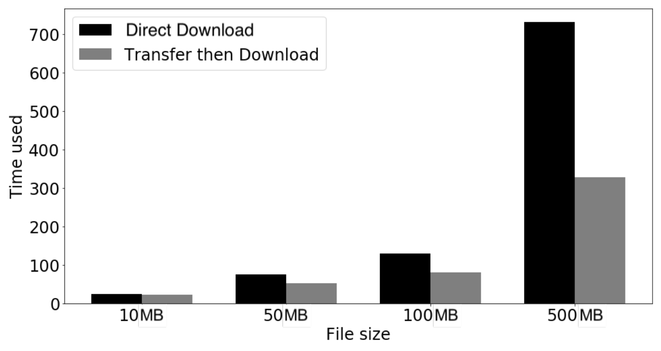

Fig. 15. Comparison between the two approaches

Minio) at the home cloud. In the home cloud, he/she can of course access data efficiently. However, when the mobile user travels to a foreign city for example, the time required to access the data object storage system from the home cloud (e.g., downloading a file) is greatly increased because of latency and bandwidth. The scenario is illustrated in Fig. 14.

At the foreign cloud, the user can initiate another data object storage (namely foreign data object storage) nearby, and utilize the Intercloud Gateway to transfer the data/files from the home data object storage to the foreign data object storage. He/she can then access the data/files from the foreign cloud more directly and efficiently. We have conducted experiments to evaluate this approach. Basically, we compared the time required between (1) directly downloading data from the home cloud and (2) transferring data to the foreign cloud first and then accessing data from the foreign cloud. Fig. 15 shows the comparison results for different data sizes. We can see that it is better to transfer data from the home cloud to the foreign cloud first (i.e., Intercloud data transfer) so that the mobile device can access the data more efficiently. As shown in the figure, this is particularly beneficial for large data objects. One of the reasons is that Intercloud data transfer is faster, and it is more efficient for a mobile device to access data from a nearby cloud. This experiment illustrates the key benefit of using an Intercloud system to support mobile Intercloud applications.

VI. Conclusion

In conclusion, we have presented an overview of an Intercloud system. By using Intercloud Gateways, data and virtual machines can be transferred using ICCP. The Gateways not only support data transfer between clouds from the same cloud providers, but also between heterogeneous clouds running on different platforms. We have presented experimental results to analyze the system performance. Furthermore, we have presented experimental results to illustrate the advantage of a mobile Intercloud system.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496100

[3] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, July 2013.

[4] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: a study of infrastructure as a service (iaas)," *International Journal of Engineering and Information Technology*, vol. 2, pp. 60–63, 01 2010.

[5] M. Singhal, S. Chandrasekhar, T. Ge, R. Sandhu, R. Krishnan, G. Ahn, and E. Bertino, "Collaboration in multicloud computing environments: Framework and security issues," *Computer*, vol. 46, no. 2, pp. 76–84, Feb 2013.

[6] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud - protocols and formats for cloud computing interoperability," in *2009 Fourth International Conference on Internet and Web Applications and Services*, May 2009, pp. 328–336.

[7] IEEE P2302/D0.2 - Draft Standard for Intercloud Interoperability and Federation (SIIF).

[8] Q. H. Vu, T. Pham, H. Truong, S. Dustdar, and R. Asal, "Demods: A description model for data-as-a-service," in *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, March 2012, pp. 605–612.

[9] D. Bernstein and D. Vij, "Intercloud directory and exchange protocol detail using xmpp and rdf," in *2010 6th World Congress on Services*, July 2010, pp. 431–438.

[10] D. Bernstein, D. Vij, and S. Diamond, "An intercloud cloud computing economy - technology, governance, and market blueprints," in *2011 Annual SRII Global Conference*, March 2011, pp. 293–299.

[11] C. K. Law, W. Xie, Z. Xu, Y. Dou, C. T. Yu, H. C. B. Chan, and D. W. K. Kwong, "System and protocols for secure intercloud communications," in *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec 2016, pp. 399–404.

[12] Y. H. Ho, P. M. F. Ho, and H. C. B. Chan, "Mobile intercloud system and objects transfer mechanism," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[13] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.