symmetry

MDPI

Article

Enhancing Data Transfer Performance Utilizing a DTN between Cloud Service Providers

Wontaek Hong [1,2,*], Jeonghoon Moon [1], Woojin Seok [1] and Jinwook Chung [2]

[1] Division of National Supercomputing, Korea Institute of Science and Technology Information, Daejeon 34141, Korea; jhmoon@kisti.re.kr (J.M.); wjseok@kisti.re.kr (W.S.)
[2] Department of Electrical and Computer Engineering, SungKyunKwan University, Suwon, Gyeonggi-do 16419, Korea; jwchung@skku.edu
* Correspondence: wthong@kisti.re.kr; Tel.: +82-42-869-0643

Received: 30 March 2018; Accepted: 12 April 2018; Published: 16 April 2018

Abstract: The rapid transfer of massive data in the cloud environment is required to prepare for unexpected situations like disaster recovery. With regard to this requirement, we propose a new approach to transferring cloud virtual machine images rapidly in the cloud environment utilizing dedicated Data Transfer Nodes (DTNs). The overall procedure is composed of local/remote copy processes and a DTN-to-DTN transfer process. These processes are coordinated and executed based on a fork system call in the proposed algorithm. In addition, we especially focus on the local copy process between a cloud controller and DTNs and improve data transfer performance through the well-tuned mount techniques in Network File System (NFS)-based connections. Several experiments have been performed considering the combination of synchronous/asynchronous modes and the network buffer size. We show the results of throughput in all the experiment cases and compare them. Consequently, the best throughput in write operations has been obtained in the case of an NFS server in a DTN and an NFS client in a cloud controller running entirely in the asynchronous mode.

Keywords: data transfer nodes (DTN); cloud controller; network file system (NFS); data transfer; OpenStack

1. Introduction

As scientific research is performed on the basis of collaboration with other organizations, the storing, transferring, and sharing of massive data are considered to be very important components of efficient collaborative research [1,2]. The OptIPuter—which is the abbreviation for Optical networking, Internet Protocol, and computer technologies for scientific visualization—is a challenging distributed cyberinfrastructure that enables advanced data-intensive applications and collaboration with other research groups based on lightpaths. The DYNES (DYnamic NEtwork Systems) project is also aimed at providing novel cyberinfrastructure to support data-intensive science communities such as high-energy physics and astronomy, utilizing dynamic circuit provisioning services. In these challenges, because science big data should be transferred rapidly and reliably, advanced research based on global collaboration requires high-bandwidth networks instead of normal internet for business applications. In reality, many R&E (Research and Education) Network providers all over the world such as Internet2, ESnet (Energy Sciences Network), GEANT (pan-European data network), and KREONET (Korea Research Environment Open NETwork) have provided application scientists with dedicated high-performance networks in order to satisfy their network requirements.

In addition, ESnet introduced the term Science DMZ (Demilitarized Zone) in 2010 which refers to specialized network architecture to support high-end scientific applications [3,4]. It is a conceptual network design that includes system optimization, dedicated transmission servers, and software

optimization techniques for transmission, sharing, and storing of science big data. In particular, it offers an optimized data transmission environment based on DTNs (Data Transfer Nodes), which are high-performance Linux servers with an optimized system kernel and well-tuned TCP (Transmission Control Protocol) stack for high-volume data transfer services. DTNs accelerate the transmission throughput using well-tuned parallel data transfer protocols such as GridFTP and bbcp.

Similar to scientific application domains, cloud applications require the fast transfer of massive data in unexpected situations [5,6]. For example, cloud virtual machine (VM) images should be moved to the other cloud domains rapidly in the case of an emergency such as disaster recovery or mitigation. We expect that the principal role of DTNs could be applied to enhance the VM migration speed in the cloud environment. In this context, we present an approach to transfer cloud VM images rapidly between cloud service providers with the help of dedicated DTNs. The overall procedure is composed of local/remote copy processes and a DTN-to-DTN transfer process. It performs all tasks together based on the fork system call automatically and without additional administrative interruption. In particular, we focus on the local copy process between a cloud controller and DTNs and improve read/write operation performance with the well-tuned mount techniques in the Network File System (NFS) protocol. As a result, our experiment evaluation demonstrates that we can get the best write operation throughput with the combination of NFS server (async) and client (async) when a cloud controller runs an NFS client process and a DTN executes a server process.

The rest of the paper is organized as follows. Section 2 provides an overview of related research. In Section 3, we present our proposal designed for transferring VM images rapidly with DTNs in the cloud environment. Section 4 shows the organization of the experiment testbed and scenarios. We also analyze and discuss the results of experiments. Finally, we conclude the paper in Section 5.

2. Related Work

There have been several approaches to provide a more advanced research environment based on Science DMZ. Pho, N. et al. [7] designed the USP (Univ. of Sao Paulo, Sao Paulo, Brazil) Science DMZ environment in order to transfer Genome data securely and efficiently using GridFTP. In particular, they estimated and compared average throughputs in various kinds of clouds and supercomputing sites with and without an SDN (Software Defined Networking)-enabled setup. Similarly, the medical Science DMZ [8] focuses on preserving the privacy of sensitive data as well as transferring high-volume data rapidly between collaborative research institutes over R&E networks. In terms of the implementation of that concept, SciPass [9] proposes a way to transfer data securely over a 100 Gbps SDN environment. It offers two kinds of operation modes for normal traffic flows and scientific high-volume flows. As for high-volume flows, SciPass enhances data transfer throughput by bypassing traditional firewalls with the setting of bypass rules in the OpenFlow switches. The CoordiNetZ system [10] was proposed to address the shortcomings of the security mechanism in the contemporary Science DMZ model. The current mechanism depends on only coarse-grained flow access control which is provided in router ACLs (Access Control Lists). CoordiNetZ is mainly composed of SciMon, SciFlow, Controller, and Coordinator. It provides improved security mechanisms such as a security policy framework and fine-grained flow control based on the contextual information collected from host DTNs. OFFN (OneOklahoma Friction Free Network) [11] provides an advanced networking environment based on a dedicated optical network that aggregates cyberinfrastructure resources including computing and storage nodes across that state. As a result, it realizes the concept of the Science DMZ in that state and enables all OFFN members to share virtualized resources efficiently. Nguyen, V. and Kim, Y. [12] pointed out the complexity of the VLAN (Virtual Local Area Network) configuration in traditional networks. In order to address that weakness, a new SDN application for efficient VLAN management was designed and implemented through leveraging major features of SDN such as programmable flow management and centralized controller architecture. It enables operators to configure VLAN dynamically and troubleshoot misconfigurations easily. The approaches mentioned above are focused on their ScienceDMZ experiences in the view of institutional reference architectures, secure transfer,

and software-defined networking. Compared with their approaches, we concentrate instead on utilizing DTNs as a kind of performance accelerator to enhance data transfer performance when VM migration is required in cloud environments.

Regarding VM migration and placement in the cloud environment, Kaur, P. et al. [13] surveyed the difference between cold VM migration and live VM migration. In cold VM migration, VMs are moved to target sites after being powered off. On the other hand, the process of live VM migration transfers running virtual machines to the destination without shutting down. As a result, it enables an almost seamless transfer service between cloud service providers. Voorsluys, W. et al. [14] paid attention to the negative effect on performance during a live VM migration. In order to learn the cost of that migration, some experiments are performed based on Xen VMs and Web 2.0 applications. They also analyzed the results of that evaluation in terms of SLA (Service Level Agreements) violations. As for SLA in SaaS (Software as a Service) providers, a knowledge-based adaptable scheduler [15] was proposed in order to satisfy customers' requirements and maximize providers' profit. That scheduler consists of admission control, knowledge process, and a scheduler. The results of simulation using CloudSim [16,17] were presented in view of total profit, average response time, and so on. Rathod, S. and Reddy, V. [18] proposed a secure virtual migration framework based on decentralized architecture. That framework also offers a dynamic VM selection function that returns proper VMs depending on the user workload. A CIC (Composed Image Cloning) technique [19] was proposed in order to reduce the consumption of network bandwidth when VMs are migrated in federated cloud environments. That approach divides a VM image into two blocks, namely, a composable block with modular parts and a user data block. When a composed VM is requested for migration, the cloud gets only the required modular parts, not the entire image. As for VM migration, the several studies mentioned above try to improve overall performance during live VM migration and consider trade-off issues in complying with SLA. Unlike their approaches, we focus on improving transfer throughput based on DTNs when multiple VM images are transferred in cold VM migration.

In order to transfer massive data efficiently in cold migration, we consider a store-and-forward technique utilizing the dedicated DTNs. There have been similar studies on moving data efficiently using the store-and-forward transmission method, such as Phoebus [20] and I-TCP (Indirect-TCP) [21]. Phoebus is a kind of WAN (Wide Area Network) accelerator which intercepts certain traffic, stores it in intermediate nodes, and forwards it rapidly to the backbone networks utilizing high bandwidth. It splits an end-to-end connection into several small connections based on intermediate nodes. As a result, it overcomes the decrease in throughput in access networks by maximizing transmission performance in backbone networks. I-TCP is designed to improve wireless network throughput between mobile devices. It splits a TCP connection into two TCP connections—one connection from a mobile device to the wireless AP (Access Point) in the edge network, and the other one from that wireless AP to the other mobile device. This approach is also intended to overcome the overall reduced performance caused by the wireless network used by a sender. However, those approaches have drawbacks such as the significant system load caused by managing several additional TCP connections in intermediate nodes and recovering those connections in case of error. Compared with previous approaches, our approach focuses on enhancing data transfer performance only in application protocol layers such as FTP and NFS protocol utilizing the dedicated DTNs. Therefore, we do not need to manage TCP connections directly, monitor certain traffic, and intercept it.

3. Proposed Solution

Generally, a DTN provides a data transfer service with a large amount of internal storage using a storage controller like a RAID (Redundant Array of Independent Disks) controller or with a connection to external storage systems utilizing high-speed mounting of filesystems such as Lustre, GPFS (General Parallel File System), or NFS (Network File System), as shown in Figure 1. As for the latter case, the connections between a DTN and external file systems are usually set up with high-speed network technology like Gigabit Ethernet, Fiber channel, InfiniBand, and so on. Because DTNs and external

storage systems are distributed and loosely coupled, that use case increases the flexibility when the actual services are designed. Based on these two use cases of the DTN, we propose a way in which to deploy DTNs in the cloud environment and enhance data transfer performance between a cloud controller and DTNs.

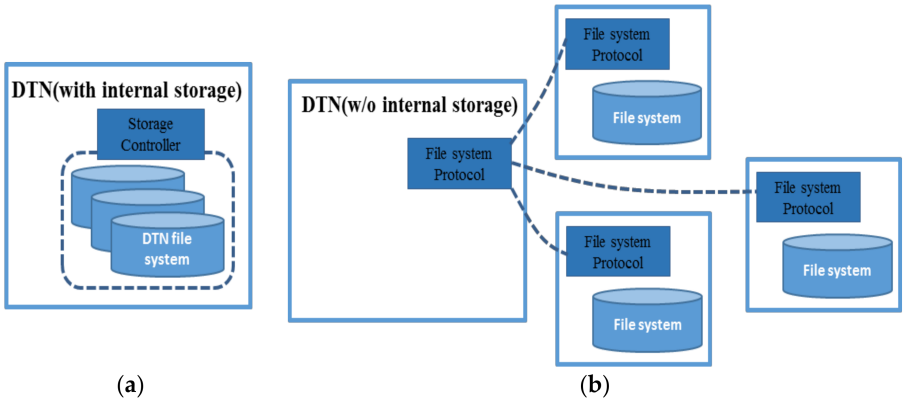

Figure 1. (**a**) A Data Transfer Node (DTN) with internal storage; (**b**) A DTN with external storage.

The DTN has normally been used when massive data in science application domains should be transferred rapidly over a high-performance network. Similar to the intention of this concept, it could be applied to connect between cloud providers. In detail, when two cloud providers want to send and receive their VM images, the connection between cloud controllers in their respective domains could be established with DTNs as shown in Figure 2. Among the two normal use cases in DTN deployments, we focus on the external storage use case and propose a way in which to establish the connection between a DTN and a cloud controller based on NFS. When a DTN works together with an external storage server, that storage system usually plays a role in the mount server of the file systems.

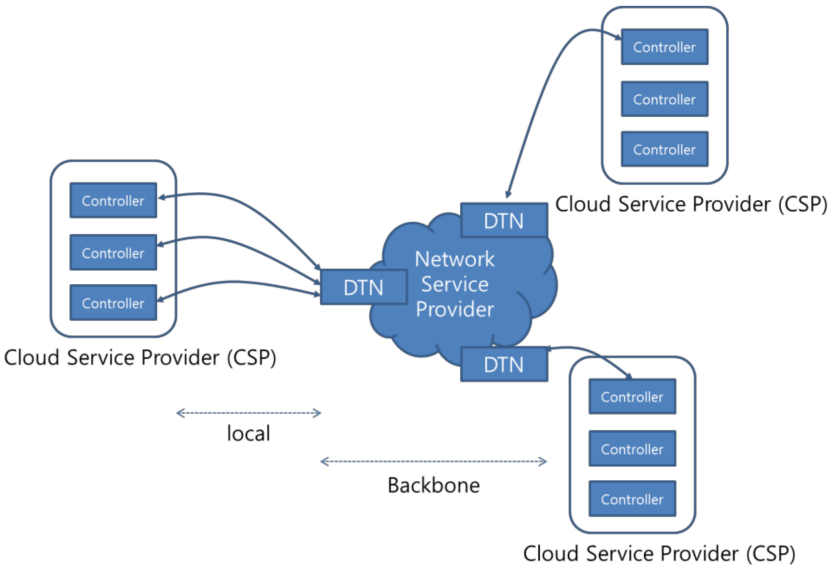

Figure 2. The connection between cloud service providers utilizing DTNs.

In the external storage use case, a cloud controller could be considered as a kind of external storage system. In reality, the cloud controller provides a VM image store like the Glance service. However, we suggest that an NFS server process is deployed in a DTN and an NFS client process is run on a cloud controller. The advantage of this approach is as follows. First, we can reduce the load of a cloud controller in the cloud environment. Even though it is possible to run cloud services composed

of many microservices in several physical nodes, the cloud controller usually accommodates all of them together. In addition, if we consider that there are additional controllers that want to join our cloud environment, when an NFS server process is run on a DTN, multiple cloud controller nodes will configure and run their NFS client functions, not server functions, relatively easier.

The general procedures of our proposal are illustrated in Figure 3. First, a cloud service provider who wants to send VM images compresses them to a tar ball file. Second, an NFS service between cloud controllers and DTNs is established considering the role of NFS server and client. Additionally, NFS performance tuning and optimization should be done to improve the overall performance. Next, a local copy process is run to send the tarred VM images from a cloud controller to a DTN via the network service provider. As soon as that local copy process is over, the large file between the DTNs is transferred via a long-distance network. Finally, a destination cloud service provider performs a remote copy and restores the tar ball file to uncompressed VM files in reverse order.

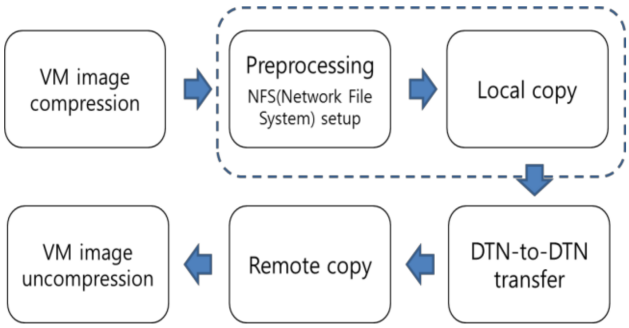

Figure 3. Proposed procedure for transferring virtual machine (VM) images with DTNs.

In order to improve the performance between NFS server and client, several NFS tuning parameters could be considered [22]. In particular, because the connection between a cloud controller and a DTN should be established in a friction-free network, we can concentrate on a few major performance tuning parameters. As for the optimization of them, we focus on some of mount options such as network buffer sizes and synchronous/asynchronous options in the NFS client and server. For reading and writing to DTNs, the overall performance of a cloud controller can be improved by tuning read/write buffer size options (rsize, wsize) based on Linux kernels and hardware specifications. Additionally, the read/write policies for a DTN should be determined understanding the trade-off between reliability and throughput when selecting synchronous and asynchronous options.

In addition to a local copy between a cloud controller and DTNs, a DTN-to-DTN transfer process including data transfer protocol like GridFTP and a remote copy are required in order to support seamless end-to-end transmission between cloud controllers in different domains. This means that a local copy process should be completed before a DTN-to-DTN transfer process starts. Additionally, it means that a remote copy process should start after a DTN-to-DTN transfer service is finished.

Taking that requirement into consideration, we propose how to harmonize the three processes—a local copy process, a DTN-to-DTN transfer process, and a remote copy process—based on a fork system call. As shown in Figure 4, the proposed procedure includes the first child process that deals with a local copy based on NFS protocol, the second child process that performs the DTN-to-DTN transfer service utilizing data transfer protocol, and a parent process that finishes all tasks including a remote copy. As a result, this proposed procedure ensures that all three processes do not get mixed up while the tarred VM images files are transferred completely.

```
algorithm transfervms
input: vm_image_dir, nfsclient_dir, dtn_list, compressed_image_file
output: none
{
   first_process_id = fork();
   if (first_process_id is less than 0) {
      return(error);
   }
   else if (first_process_id is 0) {
      copy compressed_image_file from vm_image_dir to nfsclient_dir
   }
   else {
      wait until first process ends
      second_process_id = fork();
      if (second_process_id is less than 0) {
         return(error);
      }
      else if (second_process_id is 0) {
         transfer compressed_image_file with dtn_list
      }
      else {
         wait until second process ends
         copy compressed_image_file from nfsclient_dir to vm_image_dir
      }
   }
}
```

Figure 4. Algorithm for harmonizing local/remote copy and DTN-to-DTN transfer processes.

4. Experiments and Analysis

In this section, we show the results of the NFS mount and tuning experiments, where we sought to enhance the performance of transferring a local copy to a DTN. Several experiments were performed to seek the optimized read/write buffer size and combination of synchronous/asynchronous modes in the NFS server and client.

With regard to building an experiment testbed, as shown in Figure 5, we used the OpenStack Mitaka version which was released in April 2016 [23] and NFS version 4. The OpenStack controller server has four Intel Xeon E5620 cores running at 2.40 GHz, 54 GB of main memory, and a 500 GB disk drive. The DTN version with FIONA (Flash I/O Network Appliance) has eight Intel Xeon E5-2630 cores running at 2.40 GHz, 32 GB of main memory, and a 4TB SSD/32TB SATA disk drive. The network speed between them is 1 Gbps.

In order to estimate the performance of read/write operations over an NFS connection, we utilized the dd command, a kind of utility for Linux operating systems that copies a file, converting according to the operands [24]. It can be used to create binary image files instantly with a given size. For example, when we want to estimate the performance based on the variation of the write buffer size, the operands "if = /dev/zero" and "of = /home/nfsClient/testfile" are chosen. In the case of the read operation, the operands "if = /home/nfsClient/testfile" and "of = /dev/null" are chosen. It also provides various configuration parameters like bs (read and write bytes at a time) and count (the number of input blocks) so that we can create different experiment scenarios.

Generally, there is a trade-off between the reliability and the throughput when synchronous and asynchronous modes are selected in an NFS server and client. In our experiment, the DTN and the cloud controller node run the NFS server and client processes, respectively, in order to provide a local copy function based on a high-performance NFS mount. We anticipated that would be a performance difference according to the combination of operation modes in the NFS server and client process. Therefore, we performed different experiments with three scenarios in order to look into the performance influence based on the combination of synchronous/asynchronous modes in the

NFS server/client and read/write buffer size. In these experiments, we considered three cases, as shown in Table 1: NFS server (sync)/client (async), NFS server (async)/client (async), and NFS server (sync)/client (sync). Instead of a real compressed VM image file, the file created by dd command was used to be read and written to a destination node. The size of that file is about 256 MB. Each experiment case was performed varying the read/write buffer size from 1 Kb to 1024 Kb and all stages in each buffer size were repeated three times. After that, we calculated the average throughput for each read/write buffer size.

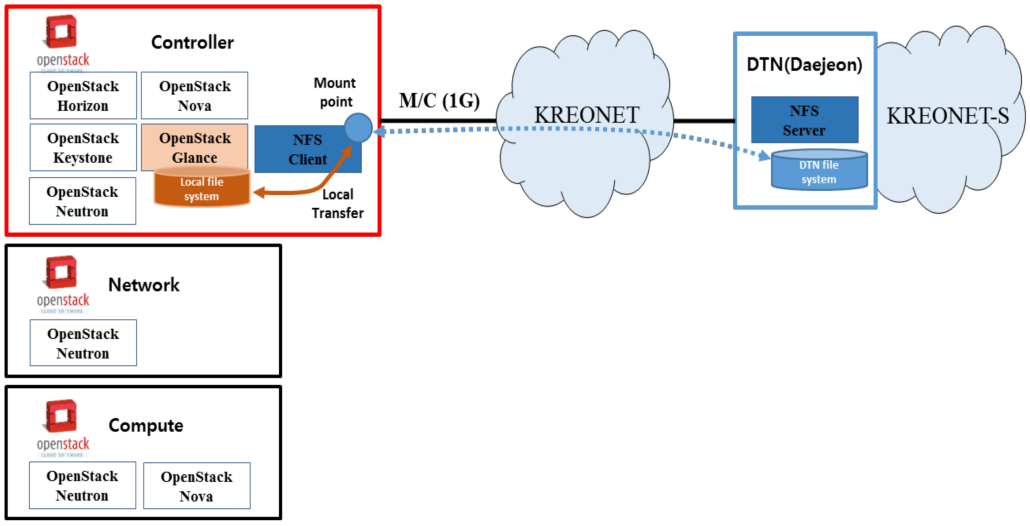

Figure 5. Experiment testbed topology.

Table 1. Experiment cases.

	DTN (NFS Server)	Controller (NFS Client)
Case I	synchronous	asynchronous
Case II	asynchronous	asynchronous
Case III	synchronous	synchronous

The average throughput of read operations was almost that same over all three cases and was about 118 Mbps. This means that the combination of synchronous/asynchronous modes in the NFS server/client and the variation in the read buffer size are almost insignificant in our environment. We omit the graph of those results.

On the other hand, the average throughput of the write operation was quite different in all cases, as shown in Figures 6 and 7. First, for the NFS server (sync) and client (async), the throughput converged to about 87 MB/s from the write buffer with size 8 Kb. Second, for NFS server (async) and client (async), the throughput converged to about 106 MB/s from the write buffer with size 8 Kb. Finally, in the case of the NFS server (sync) and client (sync), the throughput increased with an S-shaped growth curve and converged to about 85 MB/s from the write buffer with size 64 Mb. In Case III, the throughput in the write buffer with size 1024 Kb was about 48 MB/s and did not yet arrive at a converged throughput. Therefore, additional experiments were performed varying the write buffer size to about 64 Mb. Consequently, in our testbed environment, the best throughput was estimated with the write buffer size of 8 Kb in the case of the NFS server (async) and client (async). Compared with the case of the NFS server (sync) and client (async), the throughput of the write operation was improved by about 22%.

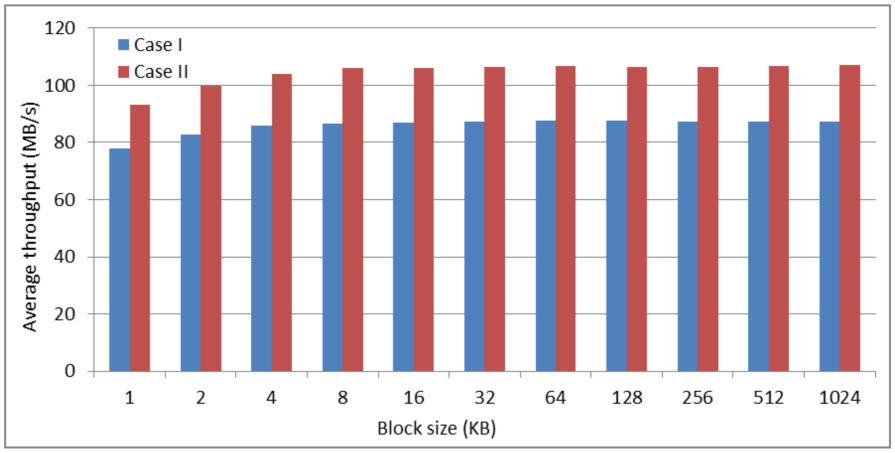

Figure 6. Throughput of the write operation in Cases I and II.

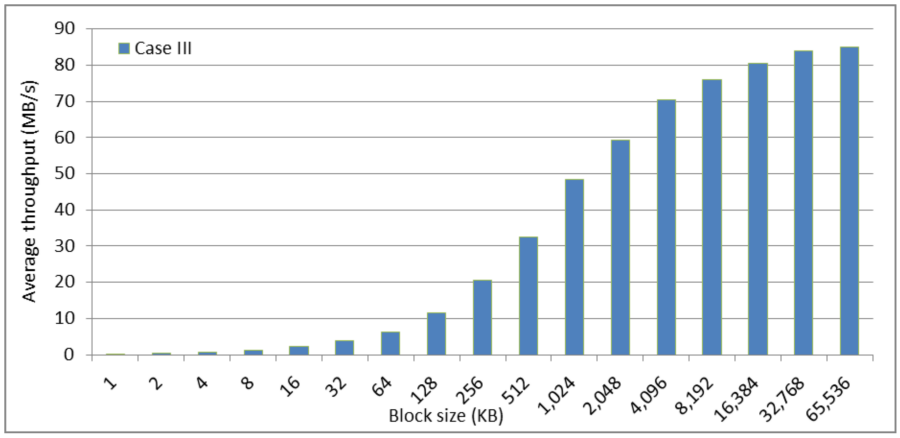

Figure 7. Throughput of the write operation in Case III.

5. Conclusions

The paper presents our approach to transfer cloud VM images with high speed utilizing a DTN between cloud service providers. The proposed procedure for transferring VM images to a destination controller includes local/remote copy processes and a DTN-to-DTN transfer process. Multiple processes are kept separate while VM images are transferred to a destination owing to the proposed algorithm based on a fork system call. In particular, we have contributed to improving the local copy performance between a cloud controller and DTNs by way of NFS mount and performance tuning. In order to perform that experiment, we built an experiment testbed consisting of a cloud controller implemented using OpenStack software and a DTN with FIONA (Flash I/O Network Appliance). When considering synchronous and asynchronous options in NFS server/client, there is generally a trade-off issue between the reliability and the throughput. In order to explore that issue in our cloud environment, we performed several experiments to select a proper operation mode and network buffer size for read/write operations. As a result, in the case of NFS server (async) and client (async), we gained the best throughput which converged to about 106 MB/s during write operations. That throughput is improved by about 22% compared with the case of NFS server (sync) and client (async). This means that our approach can be used to perform performance-critical data transfer based on an NFS mount in cloud environments where the connection between a cloud service provider and a network service provider is established stably. As future work, we plan to accommodate additional experiment scenarios to reflect on various kinds of VM images used in a real-world cloud service.

Acknowledgments: This work was partially supported by Korea Institute of Science and Technology Information (KISTI), and by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. R7117-16-0218, Development of Automated SaaS Compatibility Techniques over Hybrid/Multisite Clouds).

Author Contributions: Wontaek Hong conceived the original idea of this research and wrote the paper; Jeonghoon Moon helped perform the experiments; Woojin Seok surveyed related work; Jinwook Chung supervised this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Smarr, L.; Brown, M.; Laat, C. Special section: OptIPlanet—The OptIPuter global collaboratory. *Future Gener. Comput. Syst.* **2009**, *25*, 109–113. [CrossRef]
2. Zurawski, J.; Boyd, E.; Lehman, T.; McKee, S.; Mughal, A.; Newman, H.; Sheldon, P.; Wolff, S.; Yang, X. Scientific data movement enabled by the DYNES instrument. In Proceedings of the First International Workshop on Network-Aware Data Management, Seattle, WA, USA, 14 November 2011.
3. Dart, E.; Rotman, L.; Tierney, B.; Hester, M.; Zurawski, J. The Science DMZ: A Network Design Pattern for Data-Intensive Science. In Proceedings of the SC'13 International Conference on High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 17–21 November 2013.
4. Science DMZ Network Architecture. Available online: http://fasterdata.es.net/science-dmz/ (accessed on 27 March 2018).
5. Mansouri, Y.; Toosi, A.; Buyya, R. Data Storage Management in Cloud Environments: Taxonomy, Survey, and Future Directions. *ACM Comput. Surv.* **2017**, *50*, 91. [CrossRef]
6. Kokkinos, P.; Kalogeras, D.; Levin, A.; Varvarigos, E. Survey: Live Migration and Disaster Recovery over Long-Distance Networks. *ACM Comput. Surv.* **2016**, *49*, 26. [CrossRef]
7. Pho, N.; Magri, D.; Redigolo, F.; Kim, B.; Feeney, T.; Morgan, H.; Patel, C.; Botka, C.; Carvalho, T. Data Transfer in a Science DMZ using SDN with Applications for Precision Medicine in Cloud and High-performance Computing. In Proceedings of the SC'15 International Conference on High Performance Computing, Networking, Storage and Analysis, Austin, TX, USA, 15–20 November 2015.
8. Peisert, S.; Bamett, W.; Dart, E.; Cuff, J.; Phil, D.; Grossman, R.; Balas, E.; Berman, A.; Shankar, A.; Tiemey, B. The Medical Science DMZ. *J. Am. Med. Inform. Assoc.* **2016**, *23*, 1199–1201. [CrossRef] [PubMed]
9. Balas, E.; Ragusa, A. SciPass: A 100 Gbps capable secure Science DMZ using OpenFlow and Bro. In Proceedings of the SC'14 International Conference on High Performance Computing, Networking, Storage and Analysis, New Orleans, LA, USA, 16–21 November 2014.
10. Nagendra, V.; Yegneswaran, V.; Porras, P. Securing Ultra-High-Bandwidth Science DMZ Networks with Coordinated Situational Awareness. In Proceedings of the 16th ACM Workshop on Hot Topics in Networks, Palo Alto, CA, USA, 30 November–1 December 2017.
11. Neeman, H.; Akin, D.; Alexander, J.; Brunson, D.; Calhoun, S.; Deaton, J.; Fotou, F.; George, B.; Gentis, D.; Gray, Z.; et al. The OneOklahoma Friction Free Network: Towards a Multi-Institutional Science DMZ in an EPSCoR State. In Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment, Atlanta, GA, USA, 13–18 July 2014.
12. Nguyen, V.; Kim, Y. SDN-based Enterprise and Campus Networks: A Case of VLAN Management. *J. Inf. Process. Syst.* **2016**, *12*, 511–524.
13. Kaur, P.; Rani, A. Virtual Machine Migration in Cloud Computing. *J. Grid Distr. Comput.* **2015**, *8*, 337–342. [CrossRef]
14. Voorsluys, W.; Broberg, J.; Venugopal, S.; Buyya, R. Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation. In Proceedings of the 1st International Conference on Cloud Computing, Beijing, China, 1–4 December 2009.
15. Motavaselalhagh, F.; Esfahani, F.; Arabnia, H. Knowledge-based adaptable scheduler for SaaS providers in cloud computing. *Hum.-Centric Comput. Inf. Sci.* **2015**, *5*, 16. [CrossRef]
16. CloudSim. Available online: https://github.com/Cloudslab/cloudsim (accessed on 27 March 2018).
17. Calheiros, R.; Ranjan, R.; Beloglazov, A.; Rose, C.; Buyya, R. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Softw. Pract. Exper.* **2011**, *41*, 23–50. [CrossRef]

18. Rathod, S.; Reddy, V. NDynamic Framework for Secure VM Migration over Cloud Computing. *J. Inf. Process. Syst.* **2017**, *13*, 476–490.
19. Celesti, A.; Tusa, F.; Villari, M.; Puliafito, A. Improving virtual machine migration in federated cloud environments. In Proceedings of the 2010 Second International Conference on Evolving Internet, Valencia, Spain, 20–25 September 2010.
20. Kissel, E.; Swany, M.; Brown, A. Phoebus: A system for high throughput data movement. *J. Parallel Distrib. Comput.* **2011**, *71*, 266–279. [CrossRef]
21. Bakre, A.; Badrinath, B. I-TCP: Indirect TCP for mobile hosts. In Proceedings of the 15th International Conference on Distributed Computing Systems, Vancouver, BC, Canada, 30 May–2 June 1995.
22. NFS. Available online: http://nfs.sourceforge.net/ (accessed on 27 March 2018).
23. OpenStack. Available online: http://www.openstack.org/ (accessed on 27 March 2018).
24. Dd Tool. Available online: https://en.wikipedia.org/wiki/Dd(Unix) (accessed on 27 March 2018).

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).