Integrating High Performance File Systems in a Cloud Computing Environment

Abhisek Pan[1,2], John Paul Walters[1], Vijay S. Pai[1,2], Dong-In D. Kang[1], Stephen P. Crago[1]
[1] Information Sciences Institute, University of Southern California
3811 North Fairfax Drive, Suite 200, Arlington, VA 22203 U.S.A.
[2]Purdue University, West Lafayette, IN 47906, U.S.A.

***Abstract*—This paper outlines our ongoing efforts to effectively integrate a parallel file system in a cloud environment. We investigate how a parallel file system (PFS) can be effectively integrated and provided as a service to cloud users running High Performance Computing (HPC) applications, and what would be the performance and security implications of such a service. A critical requirement of running today's data-intensive HPC applications is the availability of a parallel high performance storage system. The objectives of our work are two-fold: first, to identify and evaluate the possible ways to deploy a PFS in a cloud infrastructure, and second, to design a framework to provide the PFS as a service inside a cloud-management framework which would allow users to provision and configure PFS storage dynamically and securely, in much the same way they can provision instances and volumes. Initial results indicate that the best file-system performance can be obtained by providing a native parallel file-system installation as a service and using file-system passthrough from the virtual machine instances to access the files directly.**

I. Introduction

The flexibility, scalability, and dynamic provisioning capabilities provided by a cloud computing infrastructure make it an attractive platform for running High Performance Computing (HPC) applications. Additionally the pay-as-you-use pricing model used by cloud providers allows the HPC community to run their applications without first worrying about incurring huge capital investments to set up expensive clusters, and then dealing with variabilities in utilization of those clusters. On the flip-side, performance and scalability concerns have meant that the HPC community is yet to fully embrace the idea of running their HPC applications in a cloud environment [1], [2].

Researchers have performed a considerable number of studies to gain a better understanding of the cost-performance trade offs in using cloud infrastructures instead of traditional HPC clusters to run scientific applications [1]–[8]. In the public cloud domain, Amazon Web Services has introduced the quadruple-extra-large and eight-extra-large cluster compute instances, both of which are specifically geared towards running I/O intensive and tightly coupled HPC applications [9].

The consensus among researchers is that HPC applications with significant communication and I/O requirements tend to perform poorly in a cloud infrastructure because of the limited bandwidth of network and I/O service available in a cloud environment. In terms of I/O, an important requirement of today's data-intensive HPC applications is the availability of a parallel high performance file system. These applications, with high levels of parallelism and I/O activity, require high speed concurrent storage access in order to process input-output data, store periodic checkpoints, and execute out-of-core algorithms. HPC clusters are usually equipped with massively parallel file system installations such as Lustre [10], GPFS [11], PVFS [12], and PanFS [13]. In order to attain acceptable HPC performance in a cloud environment, the applications running in virtual machine instances must have access to a high-performance file system. Although there has been some work in measuring I/O overheads in a cloud environment [14], and in using a high performance file system as a backend for storage volumes [15] or web-based storage [16], there has been fewer studies on how to provide a high-performance and scalable storage service to HPC applications running in a cloud environment.

The absence of a dedicated file system service in the cloud means that *virtual* parallel file systems have to be deployed inside virtual machine (VM) instances served by virtual disks. Since parallel file systems contain multiple storage targets which are accessed over the network, such virtual PFS deployments incur overheads of multiple virtualization layers, both in network and storage device levels, which seriously hamper their performance.

This paper outlines our ongoing efforts toward a more effective integration of parallel file systems into cloud environments. The objectives of our work are two-fold: first, to determine the most efficient way to access a PFS from virtual-machine instances in the cloud, and second, to design a framework to provide the PFS as a service through a cloud management platform.

In this work, we have implemented and evaluated the following possible methods of incorporating a high-performance file system in a HPC-cloud environment:

1) Building a virtual PFS with I/O servers and clients running in virtual machine instances,
2) PFS as a back-end to persistent volume storage, and
3) Accessing the PFS directly from the virtual machine instances
    - through file system passthrough from the instances

to the instance-hosts, and
- through file system clients running inside the virtual machine instances

We are also incorporating a file system service inside a cloud-management framework that would allow users to provision and configure PFS storage dynamically and securely, in much the same way they can provision instances and volumes. The challenges involve ensuring security and isolation in allocating storage to the cloud users, and determining the file system configuration options that should be exposed to the users through the file system service APIs. In this work we are using the Lustre file system [10] as the PFS backend and the OpenStack cloud management software [17] as the cloud-management framework.

The rest of this paper is organized as follows: background about the current state-of-the-art in cloud storage, LustreFS, and OpenStack in Section II, PFS access methods in Section III, PFS as a service in Section IV, framework and preliminary results in Section V, related work in Section VI, and conclusion and future work in Section VII.

II. Background

Here we review the current state-of-the-art in cloud storage, and provide brief overviews of the Lustre file system and the OpenStack cloud management framework.

A. Storage Options in Cloud

Infrastructure-as-a-Service (IaaS) cloud platforms provide three types of storage options - the ephemeral block storage, the persistent block storage, and the key-value based object storage. Both ephemeral and persistent storage services are implemented through virtual disk devices. The ephemeral storage is provisioned as part of the VM instance, and exists as long as the VM instance exists. It is allocated locally on the VM host. The persistent block storage, such as the Amazon Elastic Block Storage [18], is provisioned separately from instances and can be attached to and detached from an instance. In the open-source cloud platforms such as OpenStack and Eucalyptus, the persistent volumes are allocated on a single volume server host, usually separate from the instance hosts, and are exposed through a storage networking protocol such as iSCSI [19] or AoE [20]. The key-value based object storage, such as the Amazon Simple Storage Service (S3) [21] allows users to store and retrieve their data using web interfaces. This service lacks a file system interface and is not particularly suitable for HPC applications.

To deploy a PFS in a IaaS platform, we need to launch VM instances as storage servers with ephemeral and persistent virtual block devices as the backend storage targets, and access the servers from PFS clients running in other VM instances. This scheme of PFS deployment suffers from several layers of indirection, degrading the performance of the PFS. Any client-to-server request passes through the client-side virtual network device, the physical network, and the server-side virtual network device. In the storage server instance, the guest VM file system requests are converted to block device operations and then to the host file system requests. The actual underlying block devices that these host file system requests access are again likely to be on a remote volume server, which is accessible over a storage network. The schemes we discuss in Section III attempt to eliminate some of these overheads.

B. Lustre File System

Lustre is a highly scalable parallel distributed file system capable of handling petabytes of storage with extremely high aggregate throughput. A Lustre installation comprises three types of systems - the clients that request file system services, the Object Storage Servers (OSS) which manage the file data stored in the Object Storage Targets (OST), and the Meta-Data Servers, which manage the file namespace meta-data stored in the Meta-Data Targets (MDT). A typical Lustre installation contains one MDS and multiple OSSes. A file can be striped over all OSSes; hence, the aggregate bandwidth of the system is the sum of bandwidths provided by all the OSSes. The MDS does not present a bottleneck in file I/O scalability as it is used only for meta-data operations and not file I/O operations. Lustre supports multiple underlying networking protocols like TCP/IP and Infiniband.

C. OpenStack

OpenStack is an open-source cloud computing platform for managing public and private cloud infrastructures. It is implemented as a set of services that communicate with each other via message queues and databases. Currently OpenStack consists of the following services:

- Compute: Service responsible for provisioning and managing VM instances. It comprises of the front-end API service, with support for the Amazon EC2 API along with the native OpenStack API, and the backend scheduler and image services.
- Storage: Service to manage persistent block-level storage for VM instances and object storage for static data.
- Network: Service for managing networking models and IP addresses.
- Dashboard: A web-based user interface for managing OpenStack services.
- Authentication: Underlying identity, token, catalog, and policy service which supports all the other services.

III. Parallel File System Access Methods

In this work we have identified multiple ways to incorporate a parallel file system in the cloud environment.

A. As a Virtual PFS

The most straightforward way of incorporating a PFS in a cloud environment is to use virtual machine instances as storage servers with the underlying virtual block storage — ephemeral or persistent — as the backend storage targets (Figure 1). However such a setup is likely to be the most inefficient because of the multiple levels of network and storage emulation layers required for file access and the potential for the single volume server being a bottleneck.

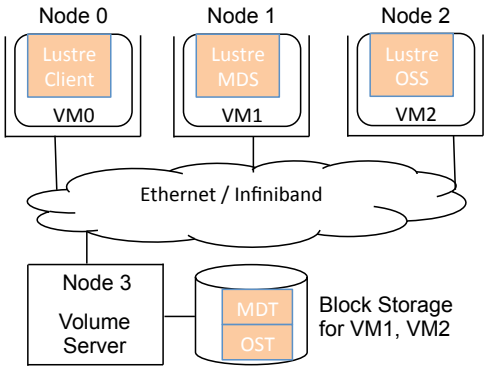


Fig. 1: Virtual PFS

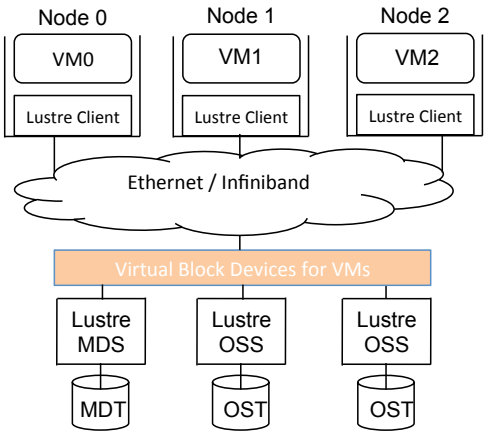


Fig. 2: PFS as Backend to EBS

B. As Backend to Emulated Block Storage

The PFS installation is used as a backend to the persistent block storage service already present in our existing setup (Figure 2).

In OpenStack, the nova-volume service uses iSCSI-exposed ([19]) LVM volumes to provide persistent block-level storage to the OpenStack Compute instances. A request for a new block-storage volume leads to the creation of a new logical volume in the 'nova-volume' volume-group in the volume server. This volume is exposed to the compute node - the node hosting the compute instance - as an iSCSI target. The libvirt daemon running on the compute host then uses this target to create and attach a block storage device to the running instance. We replace the iSCSI-LVM implementation with a Lustre-based one through the following modifications:

1) The requested volumes are created as a file in the LustreFS installation instead of logical volumes.
2) The Lustre client daemon running on the compute host makes the volume file accessible to libvirt as a storage option. Libvirt uses the directory-type storage pool to manage the volume files.
3) Libvirt uses the file to create and attach a block device to the instance running on the compute host.

These modifications improve the quality of the volume service by incorporating into the existing set-up the reliability and performance advantages of a PFS over a remote block device. This method is similar to the distributed block storage system implemented in [15], where the authors used Lustre as a backend to their implementation of a virtual block storage (VBS) service. However in this scenario, a file access from an application running in a VM instance will be first emulated as a block storage access inside the hypervisor (unless we are using a para-virtualized driver interface), and then the hypervisor would invoke the file system interface to make the actual access to block storage objects. An alternative would be to enable direct file system access from inside the VM instances, either over the network or through file system passthrough.

C. Exporting the PFS directly

The PFS installation is made directly available to the programs running inside a VM, thus avoiding the intermediate block emulation layer. Direct file system access also allows the applications inside the VMs to configure the PFS parameters such as stripe width and stripe size and stripe width according to the data access pattern.

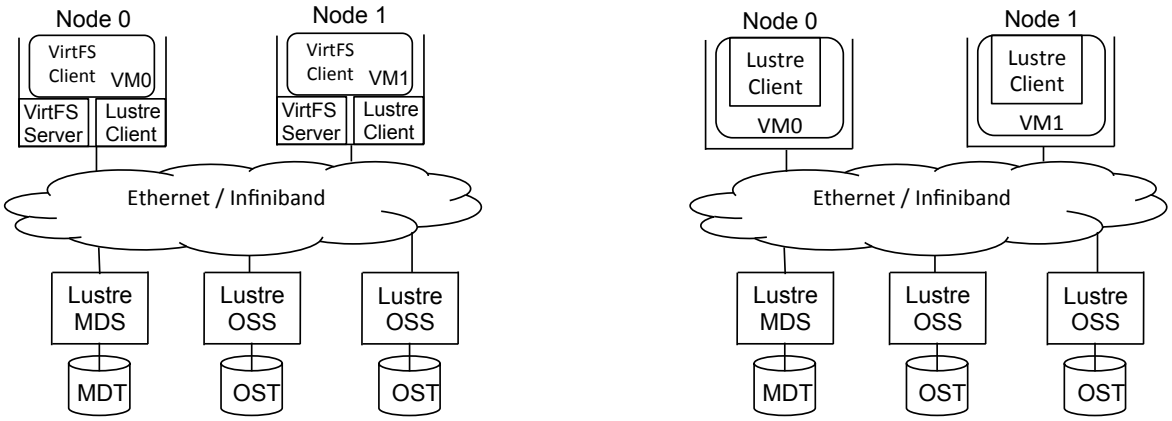
1) Using File System Passthrough: VirtFS [22] is a virtualization aware file system passthrough framework which uses the underlying VirtIO [23] transport to enable applications inside VM instances to gain direct accesses to file system on the instance hosts. These files can be on direct or network attached volumes or even on a parallel file system like Lustre. Using VirtFS with Lustre enables the applications inside the VMs to access files on a native Lustre installation directly, without having to go through the emulated block storage abstraction (Figure 3a).

2) Using Lustre clients running inside VMs: In this method we can run Lustre clients inside the VMs, and the clients can communicate with the Lustre servers though the network (Figure 3b). This is different from the previous case in the sense that in the previous case file access happened through the applications interfacing with the Lustre client daemons running on the host system through the VirtFS/VirtIO framework, whereas in this case the Lustre client daemons running in the VMs access the files on the Lustre servers through the emulated network interface on the VMs.

IV. Parallel File System as a Service

To access PFS most directly and efficiently from the cloud environment, we aim to provide the PFS installation as an integrated service to cloud users as part of our cloud management framework. The following capabilities need to be incorporated in the current OpenStack framework to set-up PFS-as-a-Service successfully.

1) Ensure that space in the installed file system can be provisioned to each cloud user in a secure manner by extending the user management framework in OpenStack to provide authentication and authorization for direct file system access.
2) Establish file system parameters which should be controllable by the cloud users when they provision a file



(a) Using File System Passthrough

(b) Lustre Client inside VM

Fig. 3: Direct PFS Access

system service, such as the total space allocated, number of I/O servers, number of storage devices per I/O server, and the choice of dedicated or shared I/O servers.

V. Framework and Preliminary Results

We used the MPI-IO interface build of the IOR benchmark to evaluate the performance of the various PFS access mechanisms. The experimental set-up consists of two clients each running one IOR task performing read-write access on a Lustre installation. The installation comprises 1 MDS and 4 OSSes. Each OSS is served by 4 OSTs and the files are striped over all the OSTs. Each node writes to and reads from different files in order to avoid the effects of client-side caching. We evaluated the three direct file access methods dicussed in section III along with a native file-system access run:

1) Virtual PFS: Lustre servers and clients running within VM instances; IOR tasks running within the same VM instances as the clients.
2) Client-in-VM: Lustre servers running natively on hardware; Lustre clients and IOR tasks running within VM instances.
3) FS Passthrough: Lustre servers running natively on hardware; Lustre clients and VirtFS servers running on VM hosts, and IOR tasks and VirtFS clients running in VM instances.
4) Native: Lustre servers, clients, and IOR tasks running natively.

Figure 4 shows the normalized aggregate bandwidth obtained from each of the configurations for sequential read and write accesses to 8 GB files in block size of 4 GB and transfer size of 1 MB. The results are normalized in terms of the aggregate bandwidth of the Virtual PFS configuration, which is the typical method for setting up a parallel file system within a cloud environment. The X-axis lists the configuration types, and the Y-axis shows the normalized performance as percentage of the baseline.

We see substantial improvement in write performance over the base configuration. The write performance improves by

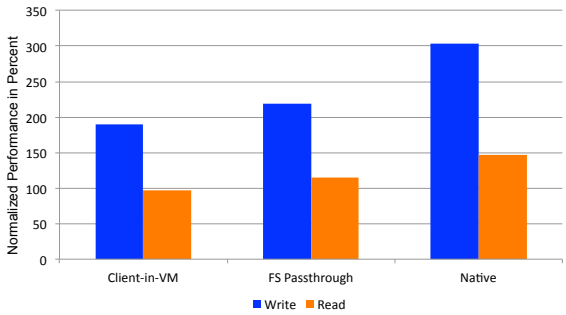


Fig. 4: Normalized Performance of PFS Access Methods

90% when we use the client-in-vm configuration and by 120% when we use the filesystem passthrough. The native performance shows over 200% improvement. The file system passthrough provides a 15% better performance over running the Lustre client inside the VM instances. This means that accessing files over the VirtIO transport layer is more efficient than access over the virtualized network stack.

The read performance shows similar trends but the improvements are comparatively lower. The client-in-vm configuration provides similar performance as the base configuration, while using the file-system passthrough improves performance by about 15% and the native performance show around 50% improvement.

The results indicate that using file system passthrough inside the VM instances to connect to the Lustre client running on the instance host should be the preferred method to access a the Lustre installation. The baseline performance is substantially lower than that obtained in the alternative configurations, because of the multiple layers of emulation overheads that the baseline incurs.

VI. Related Work

Although it has been fairly well-established that I/O intensive HPC applications tend to perform poorly in a cloud

environment, there has been relatively little work about measuring and improving the quality of I/O service available to the HPC applications running in a cloud. Ghoshal et al. did a comparative study of the I/O performance available in the Amazon cloud, in a private cloud running on a HPC cluster, and on a GPFS file system installed in the same cluster [14]. For multi-mode MPI benchmarks, native parallel file system performance was many orders of magnitude better than that obtained by sharing an EBS device among Amazon EC2 cluster compute instances through NFS.

Gao et al. used Lustre as the back-end to convert the single-volume-server architecture of their implementation of a virtual block storage service (VBS) to a distributed storage system, in order to achieve higher levels of reliability, scalability, and aggregate throughput [15].

Liu et al. discuss the idea that HPC applications can benefit from the flexibility of storage options in a cloud environment, in terms of file systems (NFS, PFS), underlying storage (Ephemeral, Persistent, S3-type), and file system parameters (number and placement of I/O servers, striping configuration) [24]. They show that the choice of shared file systems such as NFS or PVFS can have considerable impact on the performance of I/O intensive applications, depending on their specific I/O requirements.

Abe and Gibson proposed pWalrus, a hybrid storage service [16]. pWalrus uses PVFS as the back end to a Amazon S3-type storage system [21], and allows the users to access their files on the PVFS installation either through the S3 API or directly after logging into the cluster hosting the file system.

VII. Conclusion and Future Work

In this paper, we present our ongoing work toward effectively integrating PFS within a cloud environment running HPC applications. We have identified and implemented possible methods of PFS access from a VM instance, and our preliminary results show that the typical way of deploying a PFS in a cloud environment is substantially less efficient than using file system passthrough to access a native PFS installation from inside VM instances.

We are investigating the causes of the degradation seen in the baseline VPFS configuration. We are also working towards providing a native PFS installation as a service within the OpenStack cloud management framework so that users can provision high-performance storage and access their allocated space in a secure, efficient, and isolated manner.

References

[1] A. Gupta and D. Milojicic, "Evaluation of hpc applications on cloud," HP Laboratories, Tech. Rep. HPL-2011-132, August 2011.

[2] J. Napper and P. Bientinesi, "Can cloud computing reach the top500?" in *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, ser. UCHPC-MAW '09. New York, NY, USA: ACM, 2009, pp. 17–20. [Online]. Available: http://doi.acm.org/10.1145/1531666.1531671

[3] R. R. Expósito, G. L. Taboada, S. Ramos, J. Touriño, and R. Doallo, "Performance analysis of hpc applications in the cloud," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 218–229, 2013.

[4] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2011.66

[5] Y. Zhai, M. Liu, J. Zhai, X. Ma, and W. Chen, "Cloud versus in-house cluster: evaluating amazon cluster compute instances for running mpi applications," in *State of the Practice Reports*, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 11:1–11:10. [Online]. Available: http://doi.acm.org/10.1145/2063348.2063363

[6] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. Wasserman, and N. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 30 2010-dec. 3 2010, pp. 159 –168.

[7] J. Ekanayake, X. Qiu, T. Gunarathne, S. Beason, and G. Fox, "High performance parallel computing with clouds and cloud technologies," in *Cloud Computing and Software Services: Theory and Techniques*. CRC Press, July 2010.

[8] Z. Hill and M. Humphrey, "A quantitative analysis of high performance computing with amazon's ec2 infrastructure: The death of the local cluster?" in *Grid Computing, 2009 10th IEEE/ACM International Conference on*, oct. 2009, pp. 26 –33.

[9] Amazon Web Services LLC, "http://aws.amazon.com/hpc-applications/."

[10] P. J. Braam and P. Schwan, "Lustre: The intergalactic file system," in *Ottawa Linux Symposium*, June 2002.

[11] F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, ser. FAST '02. Berkeley, CA, USA: USENIX Association, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1083323.1083349

[12] I. F. Haddad, "Pvfs: A parallel virtual file system for linux clusters," *Linux J.*, vol. 2000, no. 80es, Nov. 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=364352.364654

[13] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou, "Scalable performance of the panasas parallel file system," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, ser. FAST'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 2:1–2:17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1364813.1364815

[14] D. Ghoshal, R. S. Canon, and L. Ramakrishnan, "I/o performance of virtualized cloud environments," in *Proceedings of the second international workshop on Data intensive computing in the clouds*, ser. DataCloud-SC '11. New York, NY, USA: ACM, 2011, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/2087522.2087535

[15] X. Gao, Y. Ma, M. Pierce, M. Lowe, and G. Fox, "Building a distributed block storage system for cloud infrastructure," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 30 2010-dec. 3 2010, pp. 312 –318.

[16] Y. Abe and G. Gibson, "pWalrus: Towards Better Integration of Parallel File Systems into Cloud Storage," in *Workshop on Interfaces and Abstractions for Scientific Data Storage (IASDS10)*, September 2010.

[17] "http://www.openstack.org/," 2012. [Online]. Available: http://www.openstack.org/

[18] Amazon Web Services LLC, "http://aws.amazon.com/ebs/."

[19] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)," RFC 3720 (Proposed Standard), apr 2004.

[20] S. Hopkins and B. Coile, "AoE (ATA over Ethernet)," The Brantley Coile Company, Inc., Tech. Rep. AoEr11, February 2009.

[21] Amazon Web Services LLC, "http://aws.amazon.com/s3/."

[22] V. Jujjuri, E. Van Hensbergen, and A. Liguori, "Virtfs — a virtualization aware file system pass-through," *Proceedings of the Ottawa Linux Symposium*, 2010.

[23] R. Russell, "virtio: towards a de-facto standard for virtual i/o devices," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 95–103, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1400097.1400108

[24] M. Liu, J. Zhai, Y. Zhai, X. Ma, and W. Chen, "One optimized i/o configuration per hpc application: leveraging the configurability of cloud," in *Proceedings of the Second Asia-Pacific Workshop on Systems*, ser. APSys '11. New York, NY, USA: ACM, 2011, pp. 15:1–15:5. [Online]. Available: http://doi.acm.org/10.1145/2103799.2103818