One Optimized I/O Configuration per HPC Application: Leveraging the Configurability of Cloud

Mingliang Liu, Jidong Zhai and Yan Zhai
Department of Computer Science and Technology, Tsinghua University
{liuml07,zhaijidong,zhaiyan920}@gmail.com

Xiaosong Ma
Department of Computer Science, North Carolina State University
Computer Science and Mathematics Department, Oak Ridge National Laboratory
ma@csc.ncsu.edu

Wenguang Chen
Department of Computer Science and Technology, Tsinghua University
cwg@tsinghua.edu.cn

ABSTRACT

There is a trend to migrate HPC (High Performance Computing) applications to cloud platforms, such as the Amazon EC2 Cluster Compute Instances (CCIs). While existing research has mainly focused on the performance impact of virtualized environments and interconnect technologies on parallel programs, we suggest that the configurability enabled by clouds is another important dimension to explore.

Unlike on traditional HPC platforms, on a cloud-resident virtual cluster it is easy to change the I/O configurations, such as the choice of file systems, the number of I/O nodes, and the types of virtual disks, to fit the I/O requirements of different applications. In this paper, we discuss how cloud platforms can be employed to form customized and balanced I/O subsystems for **individual** I/O-intensive MPI applications. Through our preliminary evaluation, we demonstrate that different applications will benefit from individually tailored I/O system configurations. For a given I/O-intensive application, different I/O settings may lead to significant overall application performance or cost difference (up to 2.5-fold). Our exploration indicates that customized system configuration for HPC applications in the cloud is important and non-trivial.

1. INTRODUCTION

Cloud computing is gaining popularity in many areas, including High Performance Computing (HPC). Beside being CPU-intensive, HPC applications are sensitive to network bandwidth/latency as well as the performance of I/O storage systems, making them particularly challenging to run efficiently in clouds. Amazon EC2, the leading IaaS provider, released *cluster compute instances (CCIs)* in July 2010 [3,6], to explicitly target HPC applications by offering dedicated physical node allocation, powerful CPUs, and improved interconnection.

Even with such new cloud platforms provided for HPC workloads, the efficiency and cost-effectiveness of cloud computing for HPC applications are still to be determined [7]. For example, our recent evaluation indicates that although the EC2 CCIs are able to provide significantly better performance compared to earlier instance classes, small message dominated applications suffer from the long communication latency on cloud platforms [1]. Apparently, the feasibility of cloud computing is highly workload-dependent and heavily depends on individual applications' characteristics. Therefore, an important question is: ***What kind of applications are suitable to execute in cloud rather than using traditional resources (i.e., supercomputers or in-house small clusters)?***

In assessing HPC applications' suitability for cloud execution, most existing studies have focused on the the performance impact of virtualized environments and interconnect technologies on cloud platforms. In this paper, we suggest that the unique configurability provided by cloud is another important dimension to consider. To make our discussion succinct and clear, in this paper we limit our discussion to I/O configurability on I/O-intensive HPC applications, though the topic can be extended to other application domains and beyond I/O.

Besides the well-known elasticity in resource acquisition, usage, and releasing, cloud platforms provide a level of configurability impossible with traditional HPC platforms. Unlike supercomputers or in-house clusters, which only offer fixed, one-size-fits-all configurations, clouds allow users to customize their own virtual cluster and perform reconfiguration at the user, application, or even job level. For example, on traditional parallel computers the users will not be able to choose or configure the shared or parallel file system, but on EC2 one can easily set up a personalized I/O subsystem. For example, users can install and select from multiple file systems such as NFS, PVFS [4], and Lustre [5]. For a selected parallel file system, they can configure the number of I/O servers and their placement. Further, users can

APSys'11 July 11-12th, 2011 Shanghai, China

choose from different types of low-level storage devices, the ephemeral disk and Elastic Block Store (EBS), or use a combination of them. Changes in configuration can typically be deployed quickly, with simple modifications made to scripts. Therefore, the cloud enables users to setup optimized I/O configurations for **each** application upon its execution, instead of configuring an I/O sub-system for **all** applications that may run in the system. Intuitively, the better configurability of cloud should give it competitive advantage against in-house clusters and supercomputers.

To demonstrate this, we perform a few experiments and conduct related analysis on Amazon EC2 CCIs to see how different I/O sub-system configurations could affect the performance of I/O-intensive MPI applications. We also use several micro-benchmarks to evaluate the cloud storage devices and different file systems, which help us understand the low level I/O behavior in cloud systems and prune the experimental space.

This work distinguishes itself from past investigations on HPC in the cloud in the following aspects:

- We identify the configurability enabled by cloud platforms as an important factor to consider, especially for I/O-intensive HPC applications.
- We conduct a qualitative discussion on the storage system configuration options on cloud platforms and their potential impact on HPC application performance as well as user experience.
- We evaluate two applications with different I/O configurations and observe large performance gaps between these configurations. The result suggests that per-application optimization on cloud I/O configuration deserves further investigation.

The rest of the paper is organized as follows. Section 2 introduces the Amazon EC2 CCI storage system and evaluation of multiple storage options on it. Section 3 shows our preliminary result of two applications with different I/O sub-system configurations. Section 4 discusses related work and conclusion is given in Section 5.

2. STORAGE SYSTEM CUSTOMIZING OPTIONS IN THE CLOUD

In this section, we discuss several key parameters and related issues in setting up customized storage sub-system on cloud-based virtual clusters.

2.1 File System Selection

A parallel or shared file system is indispensable for parallel program execution, which provides a unified persistent storage space to facilitate application launching, input and output file accesses, and parallel I/O support. Typically, supercomputers or large clusters are equipped with parallel file systems such as Lustre [5], GPFS [11], and PVFS [4], while smaller clusters tend to choose shared/distributed file systems such as NFS. However, end users seldom have the option of choosing or configuring file systems on traditional HPC facilities. One outstanding advantage of cloud HPC execution is that users can choose a parallel or shared file system based on individual applications' demands, and can switch between different selections quite easily and quickly.

In choosing an appropriate file system, users need to consider their applications' I/O intensity and access pattern. For example, a simple NFS installation may suffice if the application has little I/O demand, or a low I/O concurrency (seen in parallel codes where one process aggregates and writes all output data, a rather common behavior in simulations). In contrast, applications that write large, shared files usually benefit from parallel file systems, especially those heavily optimized for MPI-IO operations. Especially, parallel file systems will allow users to scale up the aggregate I/O throughput by adding more I/O servers, while the single NFS server may easily become a bottleneck under heavy I/O loads. Depending on the variability among per-file access patterns, users may elect to use file systems that give extra flexibility in performance optimization, such as the per-file striping setting allowed by PVFS.

In our preliminary evaluation presented in Section 3, we demonstrate the performance difference of an I/O intensive parallel program running on NFS and PVFS, two popular file systems on clusters. Due to the space limit, we confine our discussion here to these two file systems. However, we believe that the potential to gain performance and/or cost benefits via storage option tuning apply to other file system choices as well.

2.2 Storage Device Selection

Another important storage parameter is the type of underlying storage devices. Cloud platforms typically provide multiple storage choices, with different levels of abstraction and access interfaces. For example, with EC2 CCIs, each instance can access three forms of storage: (1) the local block storage ("ephemeral") with 1690GB capacity, where user data are not saved once the instances are released, (2) off-instance Elastic Block Store (EBS), where volumes can be attached to an EC2 instance as block storage devices, whose content persist across computation sessions, and (3) Simple Storage Service (S3), Amazon's key-value based object storage, accessed through a web services interface.

For HPC applications, the ephemeral and EBS are more apparent choices as storage devices, as they do not require modifications to applications. S3, on the other hand, is designed more for Internet or database applications and lacks general file system interfaces needed in HPC programs.

Beside data consistency, the ephemeral and EBS devices possess different performance characteristics and usage constraints. One instance can only mount up to two ephemeral disks, while the number of EBS disks attached to it can be almost unlimited. In our preliminary benchmarking, we found the performance gap between the EBS and ephemeral disks small, especially for writes (the major I/O operation in numerical parallel programs), as to be demonstrated in Figure 1. However, we observed that the ephemeral disk has better availability and lower performance variance. This may be the result of different disk affinity and different virtualization techniques. Finally, the ephemeral disks are free, while EBS disks are charged by the storage capacity consumed.

Therefore, the choice between these two storage devices again depends on the needs of individual applications or users. For instance, production runs that generate results to be migrated to and visualized on users' local platforms may get the best benefit from using ephemeral disks, as persistent storage is not needed. On the other hand, repeated

processing of the same dataset (such as sequence database searches) will benefit from using the EBS. As another example, bandwidth-thirsty applications relying on parallel I/O with file striping may suffer from the load imbalance introduced by the high performance variability of EBS disks.

2.3 File System Internal Parameters

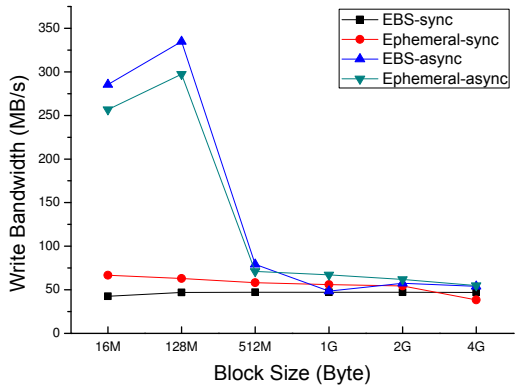

Figure 1: NFS write bandwidth, measured with IOR

For NFS, one sample internal parameter is the choice of write modes: *sync* vs. *async*. Figure 1 shows the results from running IOR [12], a parallel I/O benchmark, with these two modes. We measured the NFS write bandwidth, using two client instances, each running 8 IOR processes. The 16 processes write a shared file collectively, each dumping a contiguous partition (block). The block size is varied from 16MB to 4GB. As shown in Figure 1, with small data blocks, the *async* mode is able to offer a significant performance advantage, due to the buffering effect at server side. It will be a sound choice for applications that output a moderate amount of data and do not require that such data are flushed to secondary storage, such as periodic checkpoints (whose loss can be compensated by re-executing a certain number of computation timesteps in most numerical simulations). We use the *async* mode for NFS server by default in the following experiments.

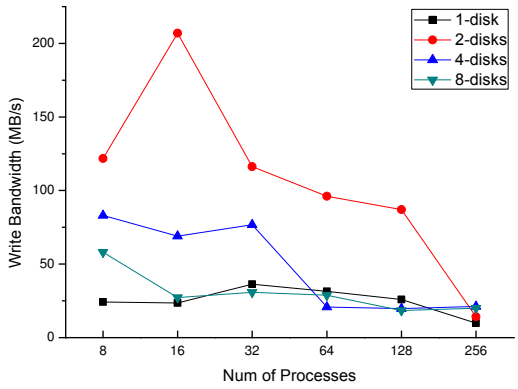

Figure 2: NFS write bandwidth with EBS disks combining into a software RAID0

On cloud platforms, a user can have tremendous flexibility in scaling up the aggregate I/O capacity and bandwidth, according to the I/O demands of their applications. For parallel file systems, this can be done by increasing the number of I/O servers, while the NFS could not scale because of the single server bottleneck. For all file systems, we can also improve each I/O server's capability by increasing the number of disks attached to it. One simple way of doing this is to combine multiple disks into a software RAID0 partition. However, one may not obtain expected scaling effect when adding (and paying for) more disks to an I/O server. Figure 2 illustrates this with another set of IOR collective write experiments, where we fixed the size of data block per process (2GB) and varied the number of processes. We evaluated different numbers (1, 2, 4, 8) of EBS disks forming a single RAID0 partition on the NFS server, and found that the system hits its peak bandwidth when only two disks are mounted. There are many possible reasons for this, such as the virtualized layer between storage volumes and multiple users, the network connection between the physical node and the physical storage devices, and the performance variance of EBS devices. In our future work we plan to investigate the disk scaling problem, as a part of automatic cloud storage configuration.

In addition, each file system has many parameters (such as striping factor, unit size, metadata server placement and disk synchronization), most of which can be configured the same way on both the cloud and the in-house cluster platforms. However, there are exceptions due to the difference in usage model brought by the virtualized cluster environment on clouds. For example, the compute resource of an I/O server is not fully utilized if the I/O server process occupies an 8-core virtual machine. Unlike on a static cluster, where an I/O server is possibly shared among multiple applications and/or users, a cloud-based virtual cluster is typically used for a dedicated run. Therefore, it is an interesting problem to explore the I/O server and metadata server placement on cloud instances, to achieve better resource utilization and overall cost-effectiveness. In Section 3 we will report empirical results with sample placement settings.

3. PRELIMINARY APPLICATIONS RESULTS

3.1 Platform Description

Our tests use the new HPC-oriented Amazon EC2 Cluster Compute Instances (CCIs) available since July 2010. Each CCI has 2 quad-core Intel Xeon X5570 "Nehalem" processors, with 23GB of memory. Each instance is allocated to users in a dedicated manner, unlike those in most other EC2 instance classes [3]. The CCIs are interconnected with 10 Gigabit Ethernet. Due to the unstable availability of EBS storage that we recently encountered, all our experiments in this section use the ephemeral disks.

Regarding software settings, we use the Cluster Instances Amazon Linux AMI 2011.02.1 associated with CCIs, the Intel compiler 11.1.072 and Intel MPI 4.0.1. The default compiler optimization level is $-O3$.

3.2 Selected HPC Applications

Our experiments evaluated two parallel applications. BTIO is the I/O-enabled version of the BT benchmark in NPB suite [13]. The program solves Navier-Stokes equations in three spatial dimensions, which are discretized, unsteady and compressible. Each of the processor will be in charge of multiple Cartesian subsets of entire data set. The I/O

strategy employs collective buffering and writing via MPI-IO interface, to avoid heavy fragmentation caused by writing per-process output files. The default I/O frequency is used, appending to the shared output file every 5 computation time steps, resulting in an output file sized around 6.4GB. The BT problem size was class C for all the tests, built with `FULL` subtype. POP is an ocean circulation model which solves the three-dimension primitive equations for fluid motions on the sphere [9] and exhibits a read-once-write-many I/O pattern. It reads a setup file at the initialization phase and performs write operation in native Fortran I/O interface at each period time step. We used the POP grid dimensions of 320x384x20 and the total output size is about 6GB, aggregated and written by process 0.

3.3 BTIO Results

In this section, we present the performance and total computation cost of running BTIO on EC2 CCIs, using NFS (*async* mode) and PVFS respectively. For each storage option, we also present the performance difference between two I/O server placement modes: I/O servers occupy separate compute instances under the *dedicated* mode, and co-reside instances with compute processes under the *part-time* mode.

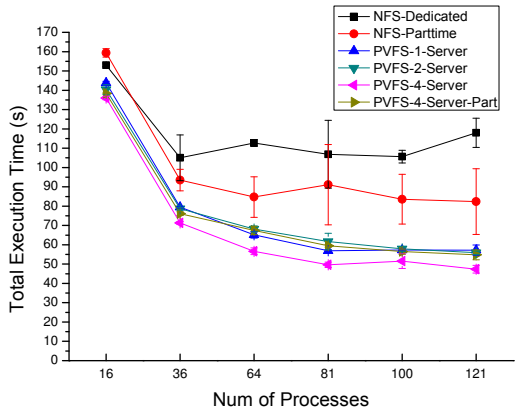

Figure 3: BTIO total execution time

Figure 3 shows the total execution time of BTIO under different file system configurations. There are up to 4 I/O servers, each mounting two ephemeral disks with software RAID0. For PVFS, one of the I/O servers acts as the metadata server. From the results, we can see that for BTIO, PVFS beats NFS in all test cases. For example, there is an up to 60% performance improvement from using NFS (with one dedicated server) to using PVFS (with 4 dedicated servers). The performance gain is more likely due to the collective MPI-IO optimizations contained in the PVFS design, as even with just one I/O servers, PVFS significantly outperforms NFS. Also, the NFS performance appears to have a much higher variance compared to that of PVFS. Interestingly, NFS performs better when the I/O server runs in the part-time mode, which can be partly attributed to the enhanced data locality and reduced network contention. However, we doubt that this factor alone causes the large performance difference observed and are carrying out more detailed investigation.

Judging by the I/O bandwidth measured from BTIO, PVFS performance scales well with increased number of I/O servers (results not plotted due to space limit). However, the scalability does not reflect well in Figure 3, the overall performance chart. This is due to the total portion of execution time devoted to I/O. For this strong-scaling experiment we expect to see that I/O gains more weight in the execution time, but the computation component of BT does not scale well on Amazon EC2, due to the unsatisfactory interconnection. Therefore as the number of processes increases, the total percentage of time spent on I/O is maintained at around 10%.

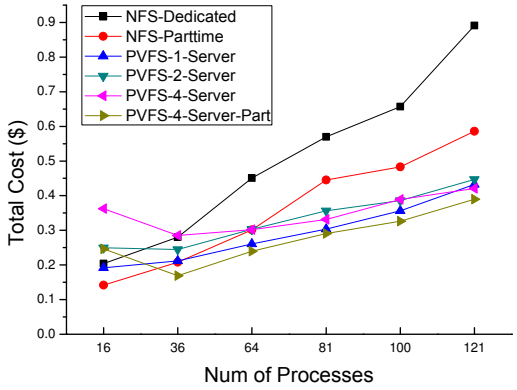

Figure 4: The total cost of BTIO executions

A related issue is the total cost of execution, when we factor in the total number of instances needed. Figure 4 shows the cost comparison derived from the BTIO results, using the Amazon charging rate of $1.6 per instance per hour. Here we can see the appeal of using part-time I/O servers. In addition, at a small execution scale, NFS with one part-time server actually appears to be the most cost-effective option.

3.4 POP Results

We run POP with similar I/O configurations (Figure 5). Unlike with BTIO, here NFS (*async* mode at server side) outperforms PVFS across all configurations. There are several possible reasons. First, POP has process 0 carry out all I/O tasks, making the parallel I/O support that PVFS was designed for less useful and the network communication more bottleneck-prone in I/O. Second, POP does I/O via POSIX interfaces, which is not optimized in PVFS. Due to its heavy communication with small messages, POP does not scale on EC2, as can be seen from Figure 5. However, its I/O behavior is still representative in parallel applications.

4. RELATED WORK

Recently, there have been a series of research efforts focusing on the performance of using public clouds for high performance computing. Most of previous work are focusing on computation and communication behavior running on clouds. Very little work has investigated I/O and storage issues in virtualized or cloud environments. Some of them focused on long-term storage or inter-job or inter-task data flow. For example, Juve et al. studied the performance and cost of different storage options for scientific workflows running on Amazon EC2 [8]. Palankar et al. assessed the feasibility of using Amazon S3 for scientific grid computing [10]. Abe et al. constructed pWalrus, which provides

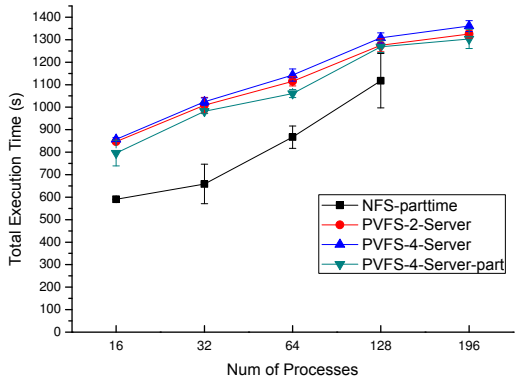

Figure 5: POP total execution time

S3-like storage access on top of the PVFS parallel file system on an open-source cloud [2]. Yu and Vetter studied parallel I/O in Xen-based HPC, but the environment used only have virtualized compute nodes and the authors only tested at most 16 processes [14]. To our best knowledge, our work is the first to evaluate different parallel/shared file system settings for parallel I/O on public clouds.

5. CONCLUSION

In this paper, we demonstrate that the unique configurability advantage offered by public clouds may bring opportunities for HPC applications to achieve significant performance or cost improvement. In particular, we illustrate the impact of virtual cluster configurability by assessing the impact of I/O system customization on the Amazon EC2 system. Our results indicate that for I/O-intensive applications, cloud-based clusters enable users to build per-application parallel file systems, where no one-size-fits-all parallel I/O solution can satisfy the needs of all applications. In the future, we will try to explore how the storage options interact with the characteristics of HPC applications and finally to automate the process of choosing I/O storage options for an individual application.

6. ACKNOWLEDGEMENT

We thank the anonymous reviewers for their valuable feedbacks. We'd like to thank Scott Mcmillan, Bob Kuhn and Nan Qiao from Intel for their interest, insights and support. This research was supported by National High-Tech Research and Development Plan (863 plan) 2006AA01A105, National Natural Science Foundation of China under Grant No. 61073175. It was also sponsored in part by NSF grants 0546301 (CAREER), 0915861, 0937908, and 0958311, in addition to a joint faculty appointment between Oak Ridge National Laboratory and NC State University, as well as a senior visiting scholarship at Tsinghua University.

7. REFERENCES

[1] Technical report r2011.4.10. Technical report, Tsinghua University. `http://www.hpctest.org.cn/resources/cloud.pdf`.

[2] 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*, 2010.

[3] Amazon Inc. High Performance Computing (HPC). http://aws.amazon.com/ec2/hpc-applications/, 2011.

[4] P. Carns, W. Ligon III, R. Ross, and R. Thakur. PVFS: A parallel file system for Linux clusters. In *Proceedings of the 4th annual Linux Showcase & Conference-Volume 4*, pages 28–28. USENIX Association, 2000.

[5] Cluster File Systems, Inc. Lustre: A scalable, high-performance file system. http://www.lustre.org/docs/whitepaper.pdf, 2002.

[6] N. Hemsoth. Amazon adds hpc capability to ec2. HPC in the Cloud, July 2010.

[7] 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 *2nd IEEE International Conference on Cloud Computing Technology and Science*, pages 159–168. IEEE, 2010.

[8] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling. Data sharing options for scientific workflows on amazon ec2. In *SC'10*, pages 1–9, 2010.

[9] LANL. Parallel ocean program (pop). `http://climate.lanl.gov/Models/POP`, April 2011.

[10] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel. Amazon S3 for Science Grids: A Viable Solution? In *Proceedings of the International Workshop on Data-Aware Distributed Computing*. ACM, 2008.

[11] F. Schmuck and R. Haskin. GPFS: a shared-disk file system for large computing clusters. In *Proceedings of the First Conference on File and Storage Technologies*, 2002.

[12] H. Shan, K. Antypas, and J. Shalf. Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark. In *SC'08*, page 42. IEEE Press, 2008.

[13] P. Wong and R. der Wijngaart. Nas parallel benchmarks i/o version 2.4. *NASA Ames Research Center Tech. Rep. NAS-03-002*, 2003.

[14] W. Yu and J. S. Vetter. Xen-Based HPC: A Parallel I/O Perspective. In *IEEE International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2008.