Xen-Based HPC: A Parallel I/O Perspective

Weikuan Yu, Jeffrey S. Vetter

Computer Science and Mathematics
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6173
{wyu,vetter}@ornl.gov

Abstract

Virtualization using Xen-based virtual machine environment has yet to permeate the field of high performance computing (HPC). One major requirement for HPC is the availability of scalable and high performance I/O. Conventional wisdom suggests that virtualization of system services must lead to degraded performance. In this presentation, we take on a parallel I/O perspective to study the viability of Xen-based HPC for data-intensive programs. We have analyzed the overheads and migration costs for parallel I/O programs in a Xen-based virtual machine cluster. Our analysis covers PVFS-based parallel I/O over two different networking protocols: TCP-based Gigabit Ethernet and VMM-bypass InfiniBand. Our experimental results suggest that network processing in Xen-based virtualization can significantly impact the performance of Parallel I/O. By carefully tuning the networking layers, we have demonstrated the following for Xen-based HPC I/O: (1) TCP offloading can help achieve low overhead parallel I/O; (2) parallel reads and writes require different network tuning to achieve good I/O bandwidth; and (3) Xen-based HPC environment can support high performance parallel I/O with both negligible overhead and little migration cost.

1 Introduction

Recent years have witnessed a resurgence in the use of various virtual machine environments, such as VMware [29], UML [8] and Xen [6]. The widespread adoption of multi-core processors also paves the way, and, in a sense, encourages the revival of virtualization in order to feed the multiple cores with more independent parallelized executions, achieve better resource utilization, and provide service consolidation. One technique known as para-virtualization [33], popularized by the Xen system, offers virtualization with low overhead. Xen has attracted a lot of attention from both the academic domain and the enterprise market.

Previous studies [12, 36] have also shown that virtualization could provide benefits to high performance computing (HPC) applications. But the HPC community has been slow to adopt virtualization due, in part, to the belief that virtualization will lead to the performance degradation of HPC applications. One of the major requirements of HPC applications is the need of scalable and high performance parallel I/O. For example, the Gyrokinetic Toroidal Code (GTC [17])–an application for fusion–can require a throughput of several 10s of gigabytes per second, so even slight performance degradations in I/O can hinder performance significantly. Much remains to be answered on how viable the Xen-based HPC environment could be for such data-intensive parallel applications.

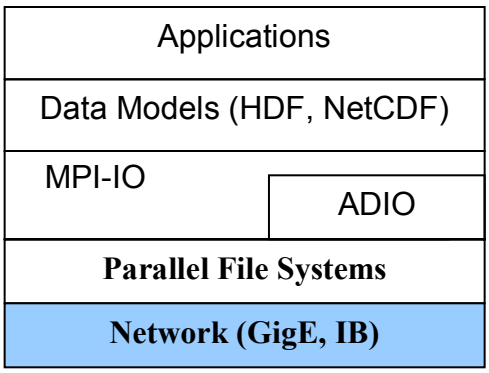


Figure 1 I/O Software Stacks for HPC Applications

Figure 1 shows a diagram of software layers in typical HPC platforms that support data-intensive HPC applications. Collectively, these layers form the portable abstractions for I/O accesses. At the top end, scientific applications perform I/O through middleware libraries such as Parallel netCDF [16], HDF5 [32] and MPI-IO [31], often cooperatively among many processes. The mid-level libraries, represented by MPI-IO, are directly implemented on top of file systems. The internal ADIO interface [30] of MPI-IO allows specialized optimizations that are tuned for various file systems. Towards the bottom of the stack, parallel file systems transfer I/O requests and file data across networks and storage devices. Xen-based virtualization introduces dramatic changes (and associated impacts) to two of the performance critical layers in this stack, i.e. I/O (file system) and networking. Thus, to answer the question of whether parallel I/O is viable through these software layers in a Xen-based HPC environment, it requires an in-depth re-examination of the entire software stack, especially the underlying networking subsystems and the parallel file systems.

In this paper, we take on a parallel I/O perspective to study the viability issue of Xen-based HPC, hereafter referred to as *Xen-HPC I/O*. We carried out a detailed analysis of parallel I/O programs' virtualization overhead and migration cost. Our analysis includes studies on the virtualization overhead of parallel I/O over two different network stacks, Gigabit Ethernet and InfiniBand, as well as the cost to parallel I/O programs during migration. In the analysis of

performance overhead, we have evaluated multiple scenarios including parallel I/O with separate files per process and a shared file. Our performance results indicated that, over GigaBit Ethernet, parallel I/O could achieve low overhead (about 10 to 29% bandwidth losses) for write operations, but not for read. Detail profiling studies revealed that network tuning could improve parallel read bandwidth by as much as 4 times. Moreover, InfiniBand [14] (IB) can achieve negligible overhead for both reads and writes in Xen virtual machines. This is due to the zero-copy communication capability offered by the IB RDMA and the VMM-bypass techniques adopted in Xen/IB [18] implementation.

We have also analyzed the migration cost of parallel programs. Our results suggest that application-transparent parallel I/O migration can be achieved on the fly (live migration), with very little cost. This is a unique strength of Xen-based environment for data-intensive parallel I/O applications, because the earlier migration approaches, such as TICK [11] and BLCR [9], have to close all the open file descriptors before a migration or restart. In summary, this paper makes the following contributions regarding Xen-HPC I/O.

- Xen virtual machines can achieve low overhead parallel I/O over TCP through offloading.
- IB can help Xen virtual machines achieve near-to-native parallel I/O performance.
- Transparent and live migration of Xen-HPC I/O applications is achievable at little cost.

The rest of the paper is organized as follows. In the next section, we provide an overview of PVFS [3] and our experimental virtual cluster for Xen-HPC I/O evaluation. Then, in Section 3, we present a performance characterization of Xen-HPC I/O over TCP, as well as the performance profiling and tuning of Xen-HPC I/O over TCP. In Section 4, we describe the performance of Xen-HPC I/O over IB [13]. In Section 5, we describes our analysis of the migration cost. Finally, we conclude the paper in Section 7 after an overview of related work in Section 6.

2 PVFS-based Parallel I/O in Virtual Clusters

2.1 An Overview of Parallel Virtual File System, (PVFS)

PVFS [3] is a lockless, user-level parallel file system. It is currently at its second generation. Figure 2 shows a diagram of PVFS system components: clients, metadata servers and I/O servers. Both the servers and client can reside completely in the user space. Moreover, PVFS enables distributed metadata server for scalable metadata operations. As also shown in Figure 2, a client opens and creates a file through a *metadata* server (step 1). It may choose another metadata server as shown by the dashed line. As a result of Step 1, a *metafile* is created for the new file and a handle is returned to the client. With the file attributes from the metafile handle, this client then creates *datafile*s in a striped manner through I/O servers (steps 2 and 3). The same metafile handle can be propagated to other clients and used to perform parallel I/O accesses. Through striped and parallel I/O, PVFS enables scalable aggregated bandwidth from many I/O servers.

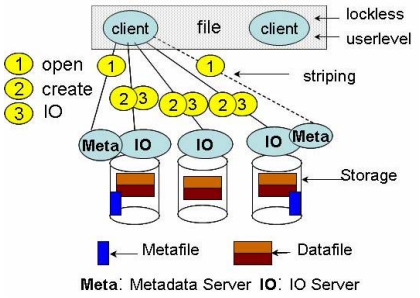


Figure 2 PVFS System Architecture

2.2 A Xen-HPC I/O Environment for Virtualized Parallel I/O

To gain insights into the effectiveness of virtual environments, such as Xen [6], for data-intensive HPC applications, one needs to understand their impacts to parallel I/O. We have configured a cluster of 24 Linux blades as a testing virtual environment. Each node in this cluster is equipped with an Intel Xeon 2.6 GHz processor, 512KB cache, 1GB physical memory, and a 7200RPM, ATA/100 Western Digital hard disk WD800JB. All nodes are connected to a 48-port Gigabit Ethernet network, and a 144-port IB [13] network using PCI-X [28] SDR host channel adaptors.

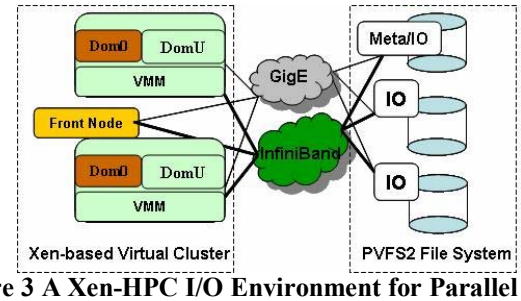


Figure 3 A Xen-HPC I/O Environment for Parallel I/O Applications

Figure 3 shows a diagram of our experimental testbed for Xen-HPC I/O applications. A PVFS file system is configured with five I/O servers and a metadata server. They are running over native Linux, version 2.6.16. In this paper, in contrast to enterprise deployments with virtualized servers, we focus on evaluating the virtualized HPC environment where only computing nodes are virtualized, so native Linux is used here for the servers to provide non-virtualized disk I/O performance [10]. A varying number of Xen Virtual Machines (VM) is created over the virtual machine monitors (VMM, a.k.a hypervisor) on other nodes. Xen VM is available for user accesses, so called domain U (DomU). All

DomUs go through a privilege domain called domain 0 (Dom0) to gain accesses to hardware resources. Each of the physical nodes has only a single CPU. The theme of Xen para-virtualization is the combination of a thin hypervisor and a privilege domain (Dom0) for efficient virtualization. As shown in Figure 3, DomUs in the virtual cluster perform I/O accesses to the PVFS file system via Gigabit Ethernet or IB. Parallel I/O programs are running on the Xen-based virtual cluster. PVFS-2.6.1 is used in this configuration. So we create only one VM per physical node to avoid performance disturbance between two DomUs on a single CPU. MPICH2-1.0.3 [5] and MVAPICH2-0.9.3 [26] are used for TCP and IB [14] experiments, respectively.

3 Xen-HPC I/O with TCP

We conducted a performance analysis of parallel I/O using the IOR benchmark [1], version 2.9.1. IOR is a benchmark that tests the performance of various parallel I/O patterns including independent I/O to separate files, independent I/O to a shared file, as well as collective I/O. Each process in IOR writes to or reads from a contiguous block of buffer (up to the entire memory available) to the parallel file system. Our goals were to find out the best IOR performance over Xen virtual machines (DomU), as well as the virtualization overhead compared to native Linux and Xen Dom0. To this purpose, we tested the performance of IOR with parallel I/O to both separated files and a shared file. We considered that collective I/O is highly dependent on data contiguity and compositions, so an evaluation of collective I/O performance would require a different study using applications with various patterns, thus was not included in this presentation.

3.1 Baseline Performance for Xen-HPC I/O

We tested the performance of parallel I/O over Xen/TCP using the default networking configuration of Xen-3.0.3. 128MB was chosen as the block size of each process in IOR. 5 iterations were performed for both writes and reads. IOR also provides a parameter for specifying the transfer size, which is the size of data in each I/O call. With a conventional understanding that large request sizes can improve I/O performance, we experimented with a series of request sizes from the default, 256KB, to 2MB. Then we chose 1MB as the fixed transfer unit size. Figure 4 shows the performance of parallel writes over different execution environment using separated files and a shared file, respectively. Parallel writes using both separated files and a shared file scale well with the increasing number of processes. Parallel writes from Xen DomUs have about 10-29% and 5-32% bandwidth losses compared to native Linux and Xen Dom0, respectively. Figure 5 shows the performance of parallel reads using separated files and a shared file, respectively. Parallel reads using both separated files and a shared file perform quite poorly compared the performance of parallel write. Parallel reads in DomU perform slightly worse than the same in Dom0.

3.2 Impact of Memory-based Storage

In measuring the baseline performance measurement, the PVFS I/O servers were configured with the backend ext3 disk file system. Data were directly served from the servers' buffer cache. So it is unlikely that the low performance of under 100MB/sec was related to disk performance. However, to make sure that it is also not due to the processing of ext3 disk file systems, we repeated the same parallel read experiments using a memory-based file system, tmpfs, for PVFS storage. Figure 6 shows the results with memory-based storage. The performance of parallel reads is improved slightly. However, it remains low for small number of processes. This suggests that the cause of low read performance is neither due to the server's local file system nor due to the server's disks.

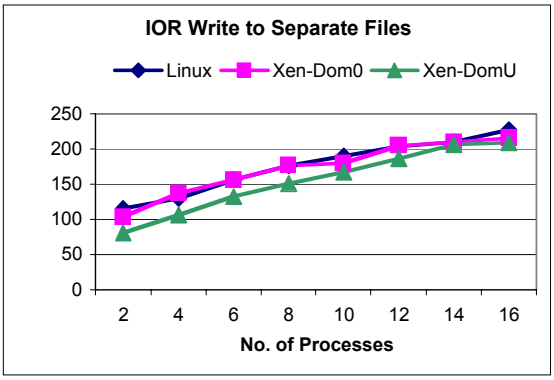
3.3 Xen-HPC I/O with TCP Optimizations

The release of Xen-3.0.4 provided a generic support for TCP offloading including checksum offloading, TCP segmentation, as well as scatter/gather support. We took this release and evaluated the performance of IOR reads and writes with the offloading options turned on and off. Note that, the PVFS servers were configured over native Linux 2.6.16, in which segmentation and checksum offloading were enabled by default.

Figure 7 shows the performance of IOR reads and writes with and without TCP segmentation and checksum offloading. While offloading TCP segmentation and checksum is beneficial to parallel writes, but parallel reads remain indifferent to this tuning. Taken into the fact that TCP segment offloading functions only on the sender side, it is understandable that such tuning has no effects on parallel reads, which is largely dependent on the receiver side TCP processing. Based on these results, we believed that the low read performance was not due to the file system processing or the disk storage, and the offloaded TCP protocol did not help the problem either.

3.4 Tuning Parallel Read with Oprofile

We considered to pursue a performance profiling with Oprofile [2] to further examine the performance of parallel read. Oprofile is a profiling tool that takes advantage of hardware counters provided from different CPU architectures and provides detailed statistics on programs and operating system instruction counts, cache misses, TLB and memory accesses, as well as the timing of individual functions and libraries. Oprofile support for Xen was developed by HP and available since Xen-3.0.3. We profiled Xen Dom0 using two hardware counters, *GLOBAL_POWER_EVENTS* and *BSQ_CACHE_REFERENCE*. The former provides a counter on



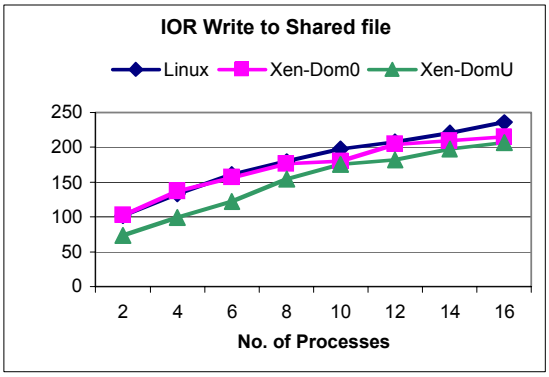
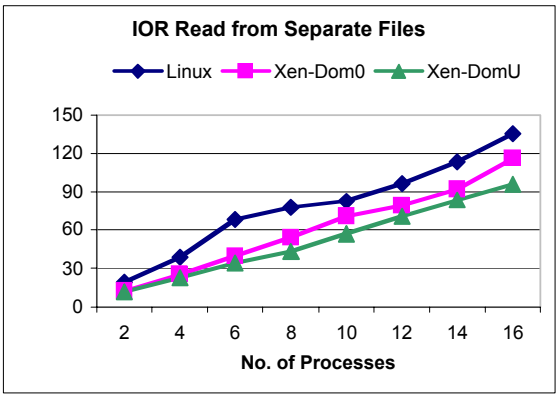


Figure 4 TCP-based Parallel Write over Linux, Xen Dom0 and DomU



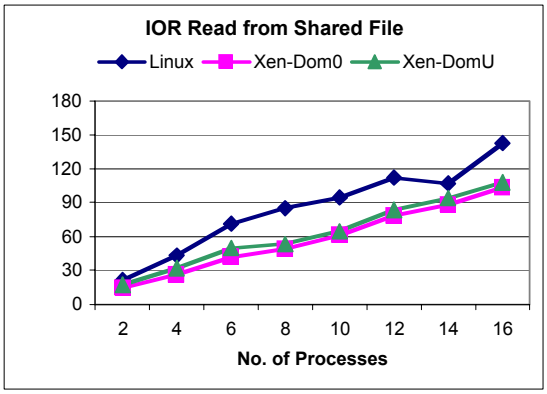


Figure 5 TCP-based Parallel Read over Linux, Xen Dom0 and DomU

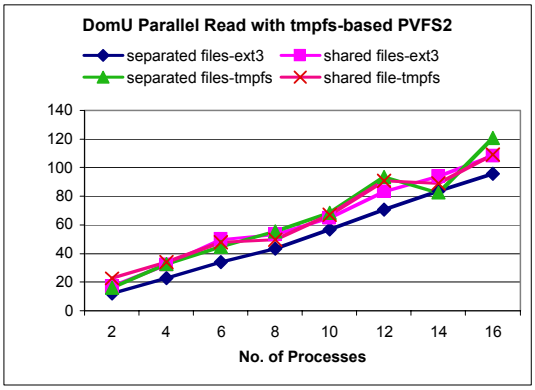


Figure 6 DomU Parallel Read over TCP with Memory-Based Storage

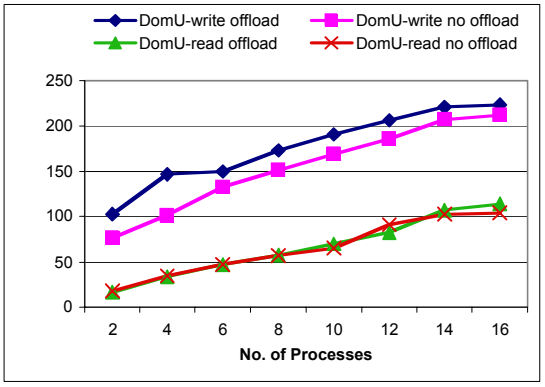


Figure 7 DomU IOR Performance with TCP Segmentation and Checksum Offloading

the time when the process is not stopped, and was used with a mask 0x1 to profile the execution time; the latter measures the cache references as seen by the bus unit, and was used with a mask 0x7 for L1 misses and 0x100 for L2 misses.

Table 1 Cache Misses for IOR

Total Count	Read	Write
L1 Cache Misses	88933	53991
L2 Cache Misses	4261	3860

The profiling results with *BSQ_CACHE_REFERENCE* identified some differences in the cache behaviors of IOR reads and writes. Table 1 shows the total L1 and L2 cache misses for IOR with 16 processes. Based on this phenomenon, we tested a variety of IOR transfer sizes from 1KB to 1MB, to check whether reduced memory footprint would lead to better cache behavior and improve the performance. Figure 8 shows the performance of IOR read when the transfer size varies between 16KB and 1MB. The default curve with 256KB socket buffer size and window scaling (*Sock=256KB, w/ WS*) indicates that the best performance is achieved at 64KB, which is 4 times more than the performance with transfer sizes of 128KB and bigger.

However, we could not observe significant differences in cache behaviors between IOR reads with these transfer sizes. On the other hand, the timing profile with the hardware counter *GLOBAL_POWER_EVENTS* indicated that a larger portion of the time was spent in kernel-level network processing with 1MB IOR transfer size, compared to 64KB. Then, we further tuned the performance of parallel reads with various TCP options. Other curves in Figure 8 show the results. We found that, besides the conventional option of using large socket buffers, it was also beneficial to disable TCP window scaling for parallel reads. Both these options significantly improve the IOR read performance with a transfer size of 128KB. However, neither can improve the performance for even bigger transfer sizes. We speculate that this be constrained by the internal PVFS configuration. In addition, Figure 9 shows that the performance of parallel reads with the 64KB transfer size. The read bandwidth scales up linearly to more than 400MB/sec at 8 processes, and then reaches its plateau with more processes.

4 Xen-HPC I/O over IB

IB [13] is a high performance, low latency interconnect technology for computer clusters. It has rapidly gained its presence in various leading supercomputing sites in the world [4]. IB [13] support for Xen [6] is recently developed at IBM by Liu *et al.*[18]. The Xen release 3.0.2-2 patched with VMM-bypass IB [18] support was used in these experiments. Xen/IB implementation delegates user-accessible memory regions to DomU in order to support RDMA communication without going through Xen VMM, so-called VMM-bypass. It was also demonstrated as a viable approach to offer high performance parallel communication for scientific applications [12]. However, it is yet to be examined whether IB is ready for data-intensive applications in Xen-HPC I/O environments.

In this section, we describe the performance evaluation of Xen/IB virtual machines in terms of their parallel I/O performance. Our experiments were conducted using the same IOR parameters as described in Section 3.1. Figure 10 shows the performance of parallel writes over different execution environment using separated files and a shared file, respectively. For both cases, parallel writes scale well till reaching a plateau around 260 MB/sec. The performance comparisons between DomU and native Linux showed that virtualization overhead is negligible, by and large within 3%. Notably, through repeated testing, the performance of parallel writes from Dom0 is consistently higher than that of native Linux by a little margin, which may be an indication of light-weight implementation of Xen Dom0 and VMM. Figure 11 shows the performance of parallel reads using separated files and a shared file, respectively. Parallel reads scale similarly to parallel write. Again, there is little virtualization overhead when comparing DomU and native Linux.

These results suggest that the VMM-bypass Xen/IB implementation is effective in exposing native parallel I/O to Xen-based virtual machines. The strengths of Xen/IB are ultimately due to the IB hardware's bandwidth capacity and its RDMA mechanism. RDMA eliminates multiple data copies at the networking layer and relieves processors from costly network processing and data handling. Note that we were not able to enhance the performance of Xen-HPC I/O over Xen/IB by tuning the IOR parameters. This is reasonable because the bulk of network processing over IB occurs on the network interface, unlike the TCP case where (for parallel read) the bulk of processing for message reception happens in the kernel of the operating system. Nonetheless, this does suggest that Xen-HPC I/O over IB may have room for improvement on the performance of parallel reads. However, this awaits a further examination of the implementation of PVFS on the IB network.

5 Live Migration for Parallel I/O Programs

Nowadays many HPC applications face the need of migration either to avoid the exponentially decreasing mean-time-between-failure (MTBF) or to achieve good balancing of applications among different partitions of HPC platforms [4]. Due to the difficulty of maintaining open file descriptors, many checkpointing/migration tools require the open files to be closed before checkpointing/migration can take place [9]. This makes it more difficult for data-intensive HPC applications to achieve run-time migration. The migration capability of virtual machines came as a possible remedy to this situation because the entire OS is migrating along with the application. No extra effort is needed for saving open file descriptors. Thus the migration capability of virtual machines

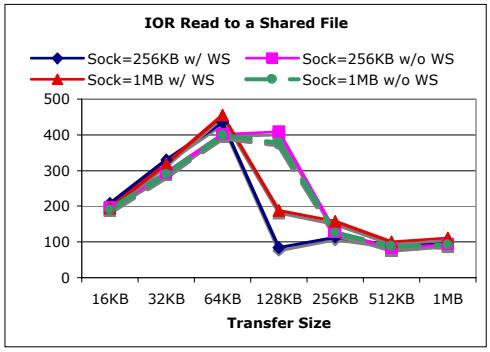


Figure 8 Tuning IOR over TCP

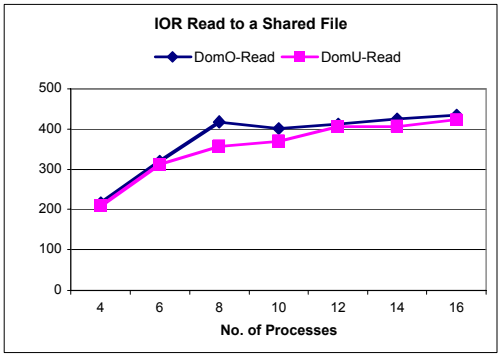
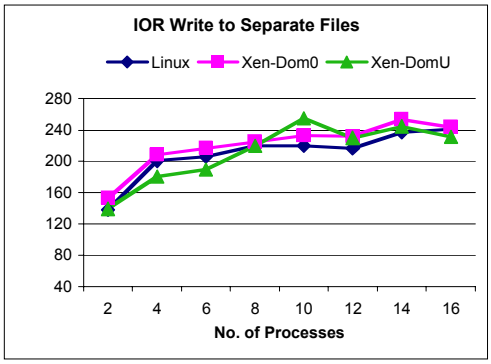


Figure 9 IOR Read with Increasing Processes



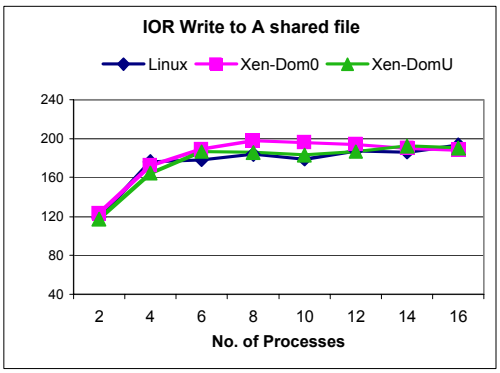
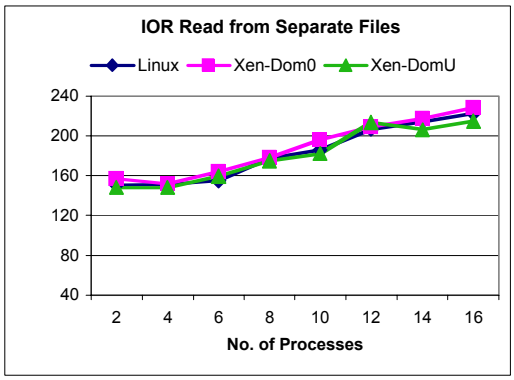


Figure 10 Parallel Write over InfiniBand on Linux, Xen-Dom0 and Xen-DomU



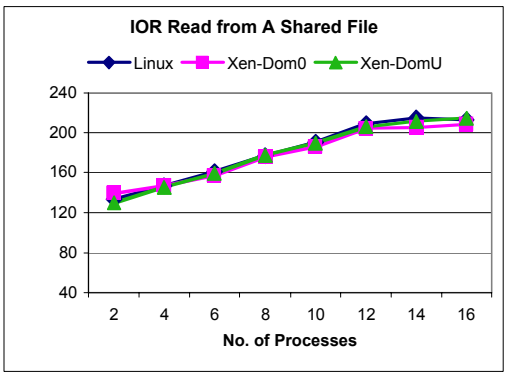


Figure 11 Parallel Read over InfiniBand on Linux, Xen-Dom0 and Xen-DomU

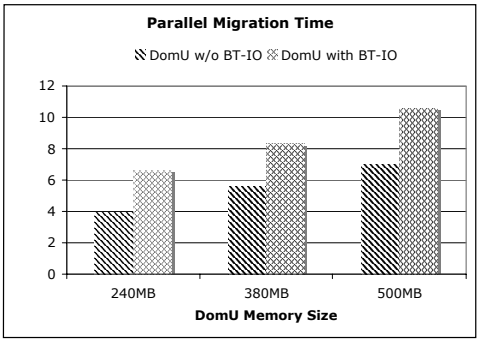


Figure 12 DomU with Different Memory Sizes

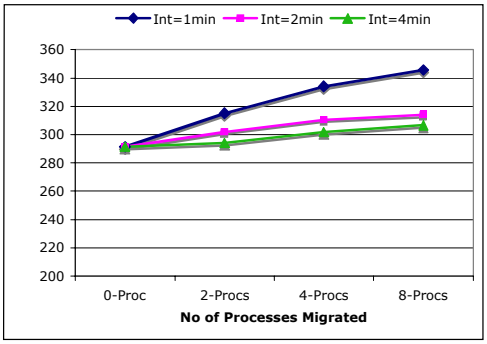


Figure 13 BT-IO with Different Migration Intervals

can be a useful alternative to other fault tolerance tools such as TICK [11] and BLCR [9]. Using NAS I/O benchmark, we have analyzed and quantified the cost and scalability trend for the migration of parallel I/O programs.

NAS BT-IO [35] is an I/O benchmark that tests the I/O capability of NAS BT (Block-Tridiagonal) parallel benchmark. It is developed at NASA Ames Research Center. The data structures are represented as structured MPI datatypes and written to a file periodically. We run BT-IO, class A, both on 9 processes (one per virtual machine per physical node). A varying number of BT-IO processes were migrated at the run time and the total execution time was collected. The migration experiments were conducted over Gigabit Ethernet. The parallel live migration was carried out using a migration script via MPI. The total migration time and the BT-IO execution time were collected.

Figure 12 shows the migration time for DomU with different memory sizes. DomU with more memory leads to longer migration time. The migration time for DomU with a running BT-IO program is also longer compared to that without BT-IO. Note that the migration time actually remains the same when a different number of DomUs are migrated (variations not shown). Nonetheless, the migration of many more virtual machines is likely to present a challenge to the bandwidth capacity of Gigabit Ethernet. We believe that future migration support of Xen VM over IB will relieve this concern because of its high bandwidth. Figure 13 shows the execution time of BT-IO when an increasing number of processes are being migrated at different frequencies, once every 1 to 4 minutes. At 4 minutes interval, the cost of migration, i.e. the increased BT-IO execution time due to migration, stays within 5% of the total execution time when no processes are migrated.

6 Related Work

Reducing the performance overhead associated with virtualization has been investigated in various studies. For example, Xen [6] and Denali [33, 34] have attempted the idea of para-virtualization to achieve light-weight virtualization. Virtualization has led to an extended path for hardware accesses from virtual machines. For example, Sugerman *et al*. [29] have described the network virtualization overhead in the VMware architecture, and strategies to reduce it by way of reducing the cost of emulated I/O access to virtual devices. Menon *et al*. [21] has documented the networking access overhead. Menon *et al*. [20] have further proposed optimizations, such as TCP segmentation offloading [19, 22], checksum offloading and scatter-gather DMA, for improved network virtualization. High speed interconnects, such as IB [13] and Myrinet [23], adopt an OS-bypass mechanism for direct device access from user-applications. However, user applications in virtual environment are segregated further from hardware devices with the addition of VMM. Liu *et al.* [18] have attempted the idea of VMM-bypass I/O to enable direct hardware device accesses, i.e. RDMA, from applications in Xen virtual machines.

Research in [12, 36] studied the feasibility of high performance computing in virtualized environment. Huang *et al*. [13] proposed a case for high performance computing platform composed of Xen-based virtual machines and IB support. Their work showed the cost of virtualization is rather light for scientific kernel benchmarks such as NAS Parallel Benchmarks [25]. However, it is not yet shown how parallel I/O would scale in that environment, nor can it be applicable to parallel I/O over legacy network protocols such as TCP. Youseff *et al.* [36] studied the applicability of Xen for high performance computing. It remains open what overhead the Xen virtual environment would bring to the parallel I/O programs. Neither of the studies has addressed the cost of migration and its scalability.

Checkpointing/migration support was developed for the purpose of load balancing and fault tolerance. Several recent efforts exploited such support for high performance applications, for example, TICK [11], BLCR [9] and MC3 [15]. These implementations support application migration/restart by virtual of system-level checkpointing. It remains to be demonstrated how effective they can support migration of parallel I/O programs. Because file descriptors are contained within operating system, which cannot be migrated along with the applications, BLCR specifically states that checkpointing for open files is not supported [11]. Work in [24] have presented an IPMI-based [14] system health monitoring framework to provide proactive fault tolerance for HPC applications. Our work is complementary to [24] in addressing the migration cost and scalability of HPC programs with intensive parallel I/O.

7 Conclusions

In this paper, we have presented a parallel I/O perspective on the issue of how viable Xen-based HPC environment would be to data-intensive parallel programs. We have shown that parallel I/O in Xen DomU can achieve good performance over both IB and TCP (with tuning). We have also highlighted a process for performance tuning to achieve good parallel read bandwidth. Moreover, we have demonstrated that the Xen-based virtual environment can provide a scalable and light-weight live migration support for parallel I/O programs.

In the future, we also intend to carry out a comprehensive study on collective I/O performance using a set of collective I/O programs that stress the parallel file system with both non-contiguous I/O [7] and structured data models such as HDF5 [32] and Parallel netCDF[16]. We also intend to investigate how large-scale data-intensive applications can take advantage of high performance I/O and light-weight migration in Xen-based virtual environment.

References

[1] "IOR Benchmark." http://www.llnl.gov/asci/purple/benchmarks/limited/ior.

[2] "Oprofile, A System Profiler for Linux ". http://oprofile.sourceforge.net/news/.

[3] "The Parallel Virtual File System, Version 2." http://www.pvfs.org/PVFS.

[4] "TOP 500 Supercomputers." http://www.top500.org/.

[5] Argonne National Laboratory, "MPICH2." http://www-unix.mcs.anl.gov/mpi/mpich2/.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A.Ho, R. Neugebauer, I. Pratt, and A.Warfield, "Xen and the Art of Virtualization.," 19th ACM Symposium on Operating Systems Principles, 2003.

[7] A. Ching, A. Choudhary, W.-k. Liao, R. B. Ross, and W. Gropp, "Noncontiguous I/O through PVFS," IEEE Cluster Computing, Chicago, IL, 2002.

[8] J. Dike, "User Mode Linux," http://user-mode-linux.sourceforge.net.

[9] J. Duell, P. Hargrove, and E. Roman, "The Design and Implementation of Berkeley Lab's Linux Checkpoint/Restart," Technical Report Berkeley Lab 2002.

[10] L. C. a. R. Gardner, "Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor," The USENIX Annual Technical Conference, Short Paper, 2005.

[11] R. Gioiosa, J. C. Sancho, S. Jiang, and F. Petrini, "Transparent Incremental Checkpointing at Kernel Level: A Foundation for Fault Tolerance for Parallel Computers," Proceedings of ACM/IEEE SC'2005, Seattle, WA, 2005.

[12] W. Huang, J. Liu, B. Abali, and D. K. Panda, "A Case of High Performance Computing with Virtual Machines," The 20th ACM International Conference on Supercomputing (ICS '06), 2006.

[13] InfiniBand Trade Association, "InfiniBand Architecture Specification, Release 1.2.."

[14] Intel, "Intelligent Platform Management Interface." http://www.intel.com/design/servers/ipmi/.

[15] H. Jung, D. Shin, H. Han, J. W. Kim, H. Y. Yeom, and J. Lee, "Design and Implementation of Multiple Fault-Tolerant MPI over Myrinet (M3)," ACM/IEEE Supercomputing, Seattle, WA, 2005.

[16] J. Li, W. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, and R. Latham, "Parallel netCDF: A High Performance Scientific I/O Interface," Proceedings of the Supercomputing '03, 2003.

[17] Z. Lin, S. Ethier, T. S. Hahm, and W. M. Tang, "Size Scaling of Turbulent Transport in Magnetically Confined Plasmas," *Phys. Rev. Lett.*, vol. 88, pp. 195004, 2002.

[18] J. Liu, W. Huang, B. Abali, and D. K. Panda, "High Performance VMM-Bypass I/O in Virtual Machines," The USENIX Annual Technical Conference, 2006.

[19] S. Makineni and R. Iyer, "Architectural characterization of TCP/IP packet processing on the Pentium M microprocessor," the 10th International Symposium on High Performance Computer Architecture, 2004.

[20] A. Menon, A. L. Cox, and W. Zwaenepoel, "Optimizing Network Virtualization in Xen," The USENIX Annual Technical Conference, 2006.

[21] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel, "Diagnosing Performance Overheads in the Xen Virtual Machine Environment," First ACM/USENIX Conference on Virtual Execution Environments (VEE'05), 2005.

[22] D. Minturn, G. Regnier, J. Krueger, R. Iyer, and S. Makineni, "Addressing TCP/IP Processing Challenges Using the IA and IXP Processors," *Intel Technology Journal*, 2003.

[23] I. Myricom, "Myrinet Express (MX): A high-performance, low-level, message passing interface for Myrinet."

[24] A. Nagarajan and F. Mueller, "Proactive Fault Tolerance for HPC with Xen Virtualization," Technical Report, January, 2007.

[25] NASA, "NAS Parallel Benchmarks." http://www.nas.nasa.gov-/Software/NPB/.

[26] Network-Based Computing Laboratory, "MVAPICH: MPI for InfiniBand on VAPI Layer." http://nowlab.cse.ohio-state.edu/projects/mpi-iba/.

[27] P. H. Carns and W. B. Ligon III and R. B. Ross and R. Thakur, "PVFS: A Parallel File System For Linux Clusters," Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, GA, 2000.

[28] PCI-SIG, "PCI I/O Virtualization." http://www.pcisig.com/.

[29] J. Sugerman, G. Venkitachalam, and B. Lim, "Virtualizing I/O devices on VMware Workstation's Hosted Virtual Machine Monitor," The USENIX Annual Technical Conference, 2001.

[30] R. Thakur, W. Gropp, and E. Lusk, "An Abstract-Device Interface for Implementing Portable Parallel-I/O Interfaces," Proceedings of Frontiers '96: The Sixth Symposium on the Frontiers of Massively Parallel Computation, 1996.

[31] R. Thakur, W. Gropp, and E. Lusk, "On Implementing MPI-IO Portably and with High Performance," Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems, 1999.

[32] The National Center for SuperComputing, "HDF5 Home Page."

[33] A. Whitaker, M. Shaw, and S. Gribble, "Denali: Lightweight virtual machines for distributed and networked applications," the USENIX Annual Technical Conference, Monterey, CA, 2002.

[34] A. Whitaker, M. Shaw, and S. Gribble, "Scale and Performance in the Denali Isolation Kernel," Operating Systems Design and Implementation, 2002.

[35] P. Wong and R. F. Van der Wijngaart, "NAS Parallel Benchmarks I/O Version 2.4," Technical Report NAS-03-002, NASA Advanced Supercomputing (NAS) Division 2002.

[36] L. Youseff, R. Wolski, B. Gorda, and C. Krintz, "Paravirtualization for HPC Systems.," Workshop on Xen in High-Performance Cluster and Grid Computing, 2006.