Understanding the Performance and Potential of Cloud Computing for Scientific Applications

Iman Sadooghi, *Member, IEEE*, Jesús Hernández Martin, Tonglin Li, *Member, IEEE*,
Kevin Brandstatter, Ketan Maheshwari, Tiago Pais Pitta de Lacerda Ruivo,
Gabriele Garzoglio, Steven Timm, Yong Zhao, and Ioan Raicu

Abstract—Commercial clouds bring a great opportunity to the scientific computing area. Scientific applications usually require significant resources, however not all scientists have access to sufficient high-end computing systems. Cloud computing has gained the attention of scientists as a competitive resource to run HPC applications at a potentially lower cost. But as a different infrastructure, it is unclear whether clouds are capable of running scientific applications with a reasonable performance per money spent. This work provides a comprehensive evaluation of EC2 cloud in different aspects. We first analyze the potentials of the cloud by evaluating the raw performance of different services of AWS such as compute, memory, network and I/O. Based on the findings on the raw performance, we then evaluate the performance of the scientific applications running in the cloud. Finally, we compare the performance of AWS with a private cloud, in order to find the root cause of its limitations while running scientific applications. This paper aims to assess the ability of the cloud to perform well, as well as to evaluate the cost of the cloud in terms of both raw performance and scientific applications performance. Furthermore, we evaluate other services including S3, EBS and DynamoDB among many AWS services in order to assess the abilities of those to be used by scientific applications and frameworks. We also evaluate a real scientific computing application through the Swift parallel scripting system at scale. Armed with both detailed benchmarks to gauge expected performance and a detailed monetary cost analysis, we expect this paper will be a recipe cookbook for scientists to help them decide where to deploy and run their scientific applications between public clouds, private clouds, or hybrid clouds.

Index Terms—Cloud computing, Amazon AWS, performance, cloud costs, scientific computing

✦

1 INTRODUCTION

THE idea of using clouds for scientific applications has been around for several years, but it has not gained traction primarily due to many issues such as lower network bandwidth or poor and unstable performance. Scientific applications often rely on access to large legacy data sets and pre-tuned application software libraries. These applications today run in HPC environments with low latency interconnect and rely on parallel file systems. They often require high performance systems that have high I/O and network bandwidth. Using commercial clouds gives scientists opportunity to use the larger resources on-demand. However, there is an uncertainty about the capability and performance of clouds to run scientific applications because of their different nature. Clouds have a heterogeneous infrastructure compared with homogenous high-end computing

- *I. Sadooghi, J. H. Martin, T. P. P. de Lacerda Ruivo, T. Li, K. Brandstatter, and I. Raicu are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. E-mail: {isadoogh, tonglin, kbrandst}@iit.edu, {Jherna22, tpaispit}@hawk.iit.edu, , iraicu@cs.iit.edu.*
- *Y. Zhao is with School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. E-mail: yongzh04@gmail.com.*
- *K. Maheshwari is with the Argonne National Laboratory Lemont, IL 60439. E-mail: ketan@mcs.anl.gov.*
- *G. Garzoglio and S. Timm are with the Fermi National Accelerator Laboratory, Batavia, IL 60510. E-mail: {garzogli, timm}@fnal.gov.*

systems (e.g. supercomputers). The design goal of the clouds was to provide shared resources to multi-tenants and optimize the cost and efficiency. On the other hand, supercomputers are designed to optimize the performance and minimize latency.

However, clouds have some benefits over supercomputers. They offer more flexibility in their environment. Scientific applications often have dependencies on unique libraries and platforms. It is difficult to run these applications on supercomputers that have shared resources with pre-determined software stack and platform, while cloud environments also have the ability to set up a customized virtual machine image with specific platform and user libraries. This makes it very easy for legacy applications that require certain specifications to be able to run. Setting up cloud environments is significantly easier compared to supercomputers, as users often only need to set up a virtual machine once and deploy it on multiple instances. Furthermore, with virtual machines, users have no issues with custom kernels and root permissions (within the virtual machine), both significant issues in non-virtualized high-end computing systems.

There are some other issues with clouds that make them challenging to be used for scientific computing. The network bandwidth in commercial clouds is significantly lower (and less predictable) than what is available in supercomputers. Network bandwidth and latency are two of the major issues that cloud environments have for high-performance computing. Most of the cloud resources use commodity network with significantly lower bandwidth than supercomputers [13].

The virtualization overhead is also another issue that leads to variable compute and memory performance. I/O is yet another factor that has been one of the main issues on application performance. Over the last decade the compute performance of cutting edge systems has improved in much faster speed than their storage and I/O performance. I/O on parallel computers has always been slow compared with computation and communication. This remains to be an issue for the cloud environment as well.

Finally, the performance of parallel systems including networked storage systems such as Amazon S3 needs to be evaluated in order to verify if they are capable of running scientific applications [3]. All of the above mentioned issues raise uncertainty for the ability of clouds to effectively support HPC applications. Thus it is important to study the capability and performance of clouds in support of scientific applications. Although there have been early endeavors in this aspect [10], [14], [16], [20], [23], we develop a more comprehensive set of evaluation. In some of these works, the experiments were mostly run on limited types and number of instances [14], [16], [17]. Only a few of the researches have used the new Amazon EC2 cluster instances that we have tested [10], [20], [24]. However the performance metrics in those papers are very limited. This paper covers a thorough evaluation covering major performance metrics and compares a much larger set of EC2 instance types and the commonly used Amazon Cloud Services. Most of the aforementioned above mentioned works lack the cost evaluation and analysis of the cloud. Our work analyses the cost of the cloud on different instance types.

The main goal of this research is to evaluate the performance of the Amazon public cloud as the most popular commercial cloud available, as well as to *offer some context for comparison against a private cloud solution.* We run micro benchmarks and real applications on Amazon AWS to evaluate its performance on critical metrics including throughput, bandwidth and latency of processor, network, memory and storage [2]. Then, we evaluate the performance of HPC applications on EC2 and compare it with a private cloud solution [27]. This way we will be able to better identify the advantages and limitations of AWS on the scientific computing area.

Over the past few years, some of the scientific frameworks and applications have approached using cloud services as their building blocks to alleviate their computation processes [12], [31]. We evaluate the performance of some of the AWS services such as S3 and DynamoDB to investigate their abilities on scientific computing area.

Finally, this work performs a detailed price/cost analysis of cloud instances to better understand the upper and lower bounds of cloud costs. Armed with both detailed benchmarks to gauge expected performance and a detailed monetary cost analysis, we expect *this paper will be a recipe cookbook for scientists to help them decide where to deploy and run their scientific applications between public clouds, private clouds, or hybrid clouds.*

This paper is organized as follows: Section 2 provides the evaluation of the EC2, S3 and DynamoDB performance on different service alternatives of Amazon AWS. We provide an evaluation methodology. Then we present the benchmarking tools and the environment settings of the testbed in this project. Section 2.4 presents the benchmarking results and analyzes the performance. On 2.5 we compare the performance of EC2 with FermiCloud on HPL application. Section 3 analyzes the cost of the EC2 cloud based on its performance on different aspects. In Section 4, we review the related work in this area. Section 5 draws conclusion and discusses future work.

2 PERFORMANCE EVALUATION

In this section we provide a comprehensive evaluation of the Amazon AWS technologies. We evaluate the performance of Amazon EC2 and storage services such as S3 and EBS. We also compare the Amazon AWS public cloud to the FermiCloud private cloud.

2.1 Methodology

We design a performance evaluation method to measure the capability of different instance types of Amazon EC2 cloud and to evaluate the cost of cloud computing for scientific computing. As mentioned, the goal is to evaluate the performance of the EC2 on scientific applications. To achieve this goal, we first measure the raw performance of EC2. We run micro benchmarks to measure the raw performance of different instance types, compared with the theoretical performance peak claimed by the resource provider. We also compare the actual performance with a typical non-virtualized system to better understand the effect of virtualization. Having the raw performance we will be able to predict the performance of different applications based on their requirements on different metrics. Then we compare the performance of a virtual cluster of multiple instances running HPL application on both Amazon EC2 and the Fermi-Cloud. Comparing the performance of EC2, which we do not have much information about its underlying resources with the FermiCloud, which we know the details about, we will be able to come up with a better conclusion about the weaknesses of the EC2. On the following sections we try to evaluate the performance of the other popular services of Amazon AWS by comparing them to the similar open source services.

Finally, we analyze the cost of the cloud computing based on different performance metrics from the previous part. Using the actual performance results provides more accurate analysis of the cost of cloud computing while being used in different scenarios and for different purposes.

The performance metrics for the experiments are based on the critical requirements of scientific applications. Different scientific applications have different priorities. We need to know about the compute performance of the instances in case of running compute intensive applications. We also need to measure the memory performance, as memory is usually being heavily used by scientific applications. We also measure the network performance which is an important factor on the performance of scientific applications.

2.2 Benchmarking Tools and Applications

It is important for us to use wide-spread benchmarking tools that are used by the scientific community. Specifically in Cloud Computing area, the benchmarks should have the

ability to run over multiple machines and provide accurate aggregate results.

For memory we use CacheBench. We perform read and write benchmarks on single instances. For network bandwidth, we use Iperf [4]. For network latency and hop distance between the instances, we use ping and traceroute. For CPU benchmarking we have chosen HPL benchmark [5]. It provides the results in floating-point operations per second (FLOPS).

In order to benchmark S3, we had to develop our own benchmark suite, since none of the widespread benchmarking tools can be used to test storage like this. We have also developed a tool for configuring a fully working virtual cluster with support for some specific file systems.

2.3   Parameter Space and Testbed

In order to better show the capability of Amazon EC2 on running scientific applications we have used two different cloud infrastructures: (1) Amazon AWS Cloud, and (2) FermiCloud. Amazon AWS is a public cloud with many datacenters all around the world. FermiCloud is a private Cloud which is used for internal use in Fermi National Laboratory.

In order to compare the virtualization effect on the performance we have also included two local systems on our tests: (1) A 6-core CPU and 16 Gigabytes of memory system (DataSys), and (2) a 48-cores and 256 Gigabytes memory system (Fusion).

2.3.1   Amazon EC2

The experiments were executed on three Amazon cloud data centers: US East (Northern Virginia), US West (Oregon) and US West (Northern California). We cover all of the different instance types in our evaluations.

The operating system on all of the US West instances and the local systems is a 64 bits distribution of Ubuntu. The US East instances use 64 bits CentOS operating system. The US West instances use Para-virtualization technique on their hypervisor. But the HPC instances on the US East cloud center use Hardware-Assisted Virtualization (HVM) [7]. HVM techniques use the features of the new hardware to avoid handling all of the virtualization tasks like context switching or providing direct access to different devices at the software level. Using HVM, Virtual Machines can have direct access to hardware with the minimal overhead.

We have included different instances as well as a non-virtualized machine. The m1.small instance is a single core instance with low compute and network performance. M1.medium is a single core system with 3.75 GB of memory. C1.xlarge instance is a compute optimized with 8 cores and 7 GB of memory. M2.4xlarge is a memory optimized instances and is supposed to have high memory performance. Hi1.4xlarge is a storage optimized instace with two SSD drives. Finally cc1.4xlarge and cc2.8xlarge as cluster compute instances, and c3.8xlarge as the new generation of HPC instances have 16 and 32 cores and more than 40 GB memory. These instances are optimized for HPC workloads.

2.3.2   FermiCloud

FermiCloud is a private cloud providing Infrastructure-as-a-Service services internal use. It manages dynamically
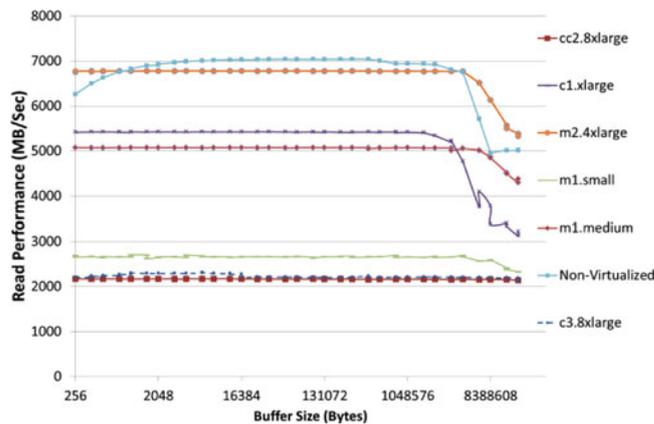

Fig. 1. CacheBench Read benchmark results, one benchmark process per instance.

allocated services for both interactive and batch processing. As part of a national laboratory, one of the main goals FermiCloud is being able to run scientific applications and models. FermiCloud uses OpenNebula Cloud Manager for the purpose of managing and launching the Virtual Machines [27]. It uses KVM hypervisor that uses both para-virtualization and full virtualization techniques [33]. The FermiCloud Infrastructure is enabled with 4X DDR Infiniband network adapters. The main challenge to overcome in the deployment of the network is introduced when virtualizing the hardware of a machine to be used (and shared) by the VMs. This overhead slows drastically the data rate reducing the efficiency of using a faster technology like Infiniband. To overcome the virtualization overhead they use a technique called Single Root Input/output Virtualization (SRIOV) that achieves device virtualization without using device emulation by enabling a device to be shared by multiple virtual machines. The technique involves with modifications to the Linux's Hypervisor as well as the OpenNebula manager [32].

Each server is enabled with a 4x (four links) Infiniband card with a DDR data rate for a total theoretical speed of up to 20 Gb/s and after the 8b/10b codification 16 Gb/s. Network latency is 1 μs when used with MPI [6]. Each card has eight virtual lanes that can create one physical function and seven virtual functions via SR-IOV. The servers are enabled with 2 quad core 2.66 GHz Intel processors, 48 Gb of RAM and 600 Gb of SAS Disk, 12 TB of SATA, and eight port RAID Controller [32].

2.4   Performance Evaluation of AWS

2.4.1   Memory Hierarchy Performance

This section presents the memory benchmark results. We sufficed to run read and write benchmarks. The experiments for each instance were repeated three times.

Memory bandwidth is a critical factor in scientific applications performance. Many Scientific applications like GAMESS, IMPACT-T and MILC are very sensitive to memory bandwidth [8]. Amazon has not included the memory bandwidth of the instances. It has only listed their memory size. We also measure the memory bandwidth of each instance.

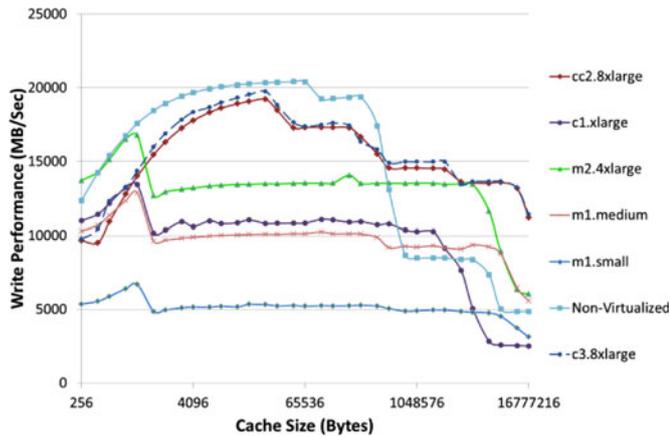Fig. 1 shows the system memory read bandwidth in different memory hierarchy levels. The horizontal axis shows

Fig. 2. CacheBench write benchmark results, one benchmark process per instance.

Fig. 3. iPerf benchmark results. Network bandwidth in a single client and server connection, internal network.

the cache size. The bandwidth is very stable up to a certain cache size. The bandwidth starts to drop after a certain size. The reason for the drop off is surpassing the memory cache size at a certain hierarchy level.

Memory performance of the m1.small instance is significantly lower than other instances. The low memory bandwidth cannot be only attributed to the virtualization overhead. We believe the main reason is memory throttling imposed based on the SLA of those instances.

Another noticeable point is the low bandwidth of the cc2.8xlarge and c3.8xlarge. These instances have similar performance that is much lower than other instances. A reason for that can be the result of the different virtual memory allocation on the VMs by HVM virtualization on these instances. We have however observed an effect in large hardware-assisted virtual machines such as those on Fermi-Cloud. In such machines, it will take a while for the system to balance the memory out to its full size at the first launch of the VM.

After all, the results show that the memory bandwidth for read operation in the larger instances is close to the local non-virtualized system. *We can conclude that the virtualization effect on the memory is low, which is a good sign for scientific applications that are mostly sensitive to the memory performance.*

Fig. 2 shows the write performance of different cloud instances and the local system. The write performance shows different results from the read benchmark. As in write, the c3.8xlarge instance has the best performance next to the non-virtualized local system.

For each instance we can notice two or three major drop-offs in bandwidth. These drop-offs show different memory hierarchies. For example on the c3.8xlarge instance we can notice that the memory bandwidth drops at 24 Kbytes. We can also observe that the write throughputs for different memory hierarchies are different. These data points likely represent the different caches on the processor (e.g. L1, L2, L3 caches).

Comparing the cluster instance with the local system, we observe that on smaller buffer sizes, the local system performs better. But cloud instance outperforms the local system on larger cache sizes. The reason for that could be the cloud instances residing on more powerful physical nodes with higher bandwidths. We can observe that the write bandwidth on the cloud instances drops off at certain
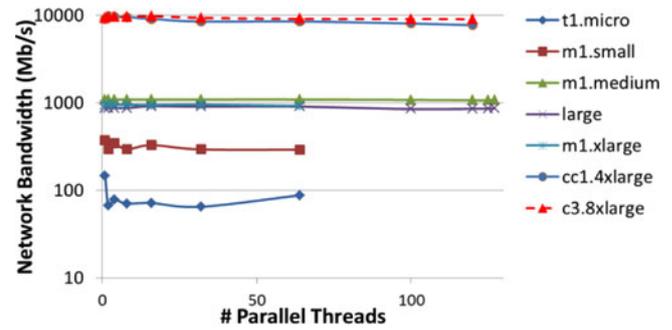
buffer sizes. That shows the memory hierarchy effects on the write operation.

Users can choose the best transfer size for write operation based on the performance peaks of each instance type to get the best performance. This would optimize a scientific application write bandwidth.

2.4.2 Network Performance

We have run many experiments on network performance of Amazon cloud. The experiments test the network performance including bandwidth and latency.

We first test the local network bandwidth between the same types of instances. Fig. 3 shows the network performance of different types of nodes. In each case both of the instances were inside the same datacenter. The network bandwidth for most of the instances were as expected except for two instances.

The lowest performance belongs to the t1.micro and m1.small instances. These two instances use the same 1 Gb/s network cards used by other instances. But they have much lower bandwidth. We believe that the reason is sharing the CPU cores and not having a dedicated core. This can affect network performance significantly as the CPU is shared and many network requests cannot be handled while the instance is on its idle time. During the idle time of the instance, the virtual system calls to the VMM will not be processed and will be saved in the queue until the idle time is over. The network performance is highly affected by processor sharing techniques. Other works had the same observations and conclusions about the network performance in these two instance types [9]. Another reason for the low performance of the m1.small and t1.micro instances could be throttling the network bandwidth by EC2. The Xen hypervisor has the ability of network throttling if needed.

Among the instances that use the slower network cards the m1.medium instance has the best performance. We did not find a technical reason for that. The m1.medium instances use the same network card as other instances and does not have any advantage on system configuration over other instance types. We assume the reason for that is the administrative decision on hypervisor level due to their popularity among different instance types.

Another odd result is for m1.medium instance. The bandwidth in medium instance exceeds 1 Gb/Sec, which is the specified network bandwidth of these. m1.medium instance bandwidth achieves up to 1.09 Gb/sec. That is theoretically not possible for a connection between two
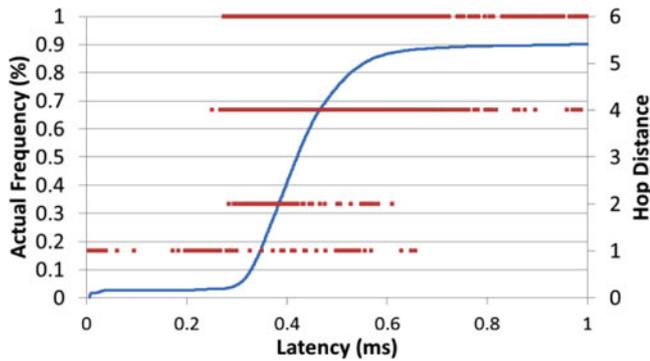
Fig. 4. Cumulative Distribution Function and Hop distance of connection latency between instances inside a datacenter.

Fig. 5. HPL benchmark results: compute performance of single instances comparing with their ideal performance.

physical nodes with 1 Gb/s network cards. We believe the reason is that both of the VMs reside in the same physical node or the same cluster. In case of residing on the same node, the packets stay in the memory. Therefore the connection bandwidth is not limited to the network bandwidth. We can also assume that not necessarily the instances have 1 Gb/s network cards. In fact the nodes that run medium instances may have more powerful network cards in order to provide better network performance for these popular instances.

The HPC instances have the best network bandwidth among the instances. They use 10 Gb/sec network switches. The results show that the network virtualization overhead in these instances is very low. The performance gets as high as 97 percent of ideal performance.

We also measure the network connection latency and the hop distance between instances inside the Oregon datacenter of Amazon EC2. We run this experiment to find out about the correlation of connection latency and the hop distance. We also want to find the connection latency range inside a datacenter. We measure the latency and the hop distance on 1,225 combinations of m1.small instances. Fig. 4 shows the network latency distribution of EC2 m1.small instances. It also plots the hop distance of two instances. The network latency in this experiment varies between 0.006 ms and 394 ms, an arguably very large variation.

We can observe from the results that: (1) 99 percent of the instances which have the transmission latency of 0.24 to 0.99 ms are 4 or 6 hops far from each other. So we can claim that if the latency is between 0.24 to 0.99 ms the distance between the instances is 4 to 6 hops with the probability of 99 percent. (2) More than 94 percent of the allocated instances to a user are 4-6 percent far from each other. In other words the hop distance is 4-6 instances with the probability of more than 94 percent.

We can predict the connection latency based on the hop distance of instances. We have run the latency test for other instance types. The results do not seem to be dependent on instance type for the instances with the same network interconnect. *The latency variance of Amazon instances is much higher than the variance in a HPC system. The high latency variance is not desirable for scientific applications. In case of HPC instances which have the 10 Gigabit Ethernet cards, the latency ranges from 0.19 to 0.255 ms which shows a smaller variance and more stable network performance.*
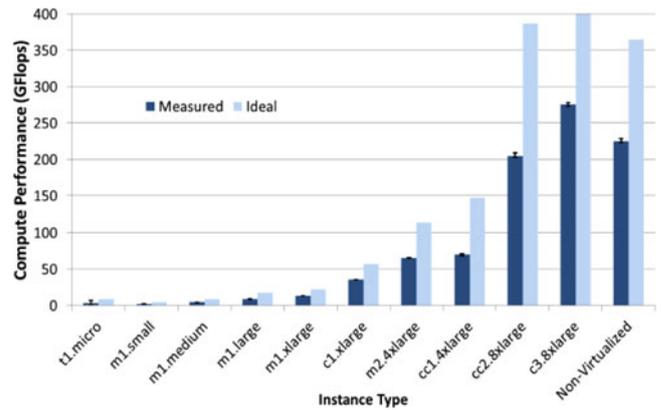
Other researches have compared the latency of EC2 HPC instances with HPC systems. The latency of the HPC instance on EC2 is reported to be 3 to 40 times higher than a HPC machine with a 23 Gb/s network card [10]. The latency variance is also much higher.

2.4.3 Compute Performance

In this section we evaluate the compute performance of EC2 instances. Fig. 5 shows the compute performance of each instance using HPL as well as the ideal performance claimed by Amazon. It also shows the performance variance of instances.

Among the Amazon instances, the c3.8xlarge has the best compute performance. The t1.micro instance shows the lowest performance. The figure also shows the performance variance for each instance. The performance variance of the instances is low in most of the instance types. Providing a consistent performance is an advantage for cloud instances.

Among all of the instances, the c3.8xlarge and the non-virtualized node achieve the best efficiency. Overall we can observe that the efficiency of non-HPC instances is relatively low. Other papers have suggested the low performance of HPL application while running on virtualized environments [11], [14]. However, noticing the fact that the HPC instances were as efficient as the non-virtualized node, and the fact that there is no other factor (e.g. network latency) affecting the benchmark, can imply that the virtualization overhead has no major effect on this program on a single node scale.

2.4.4 I/O Performance

In this section we evaluate the I/O performance of the EBS volume and local storage of each instance. The following charts show the results obtained after running IOR on the local storage and EBS volume storage of each of the instances with different transfer sizes and storage devices. Fig. 6 shows the performance of POSIX read operation on different instances. Except for the hi1.4xlarge, which is equipped with SSDs, the throughput among other instances does not vary greatly from one another. For most of the instances the throughput is close to a non-virtualized system with a normal spinning HDD.

Fig. 7 shows the maximum write and read throughput on each instance on both EBS volumes and local storage
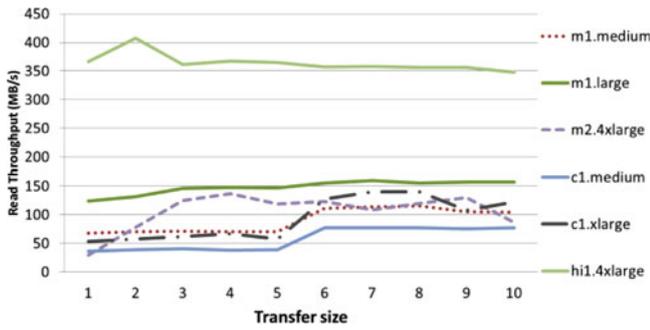
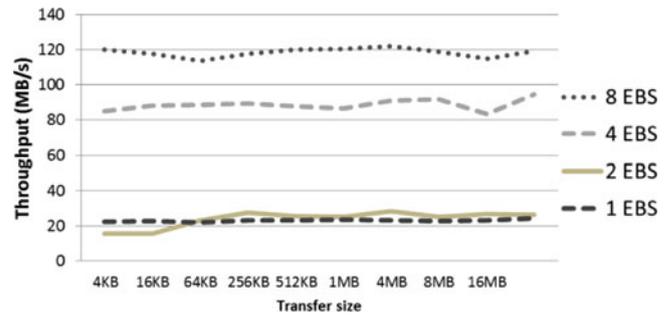Fig. 6. Local POSIX read benchmark results on all instances.

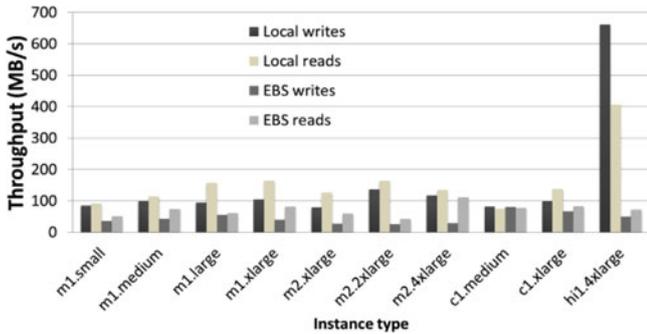Fig. 8. RAID0 Setup benchmark for different transfer sizes—write

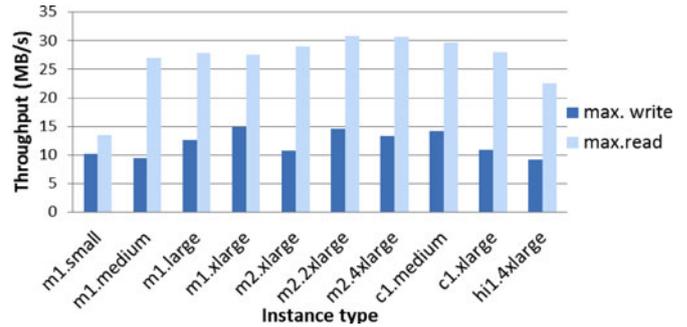Fig. 7. Maximum write/read throughput on different instances.

Fig. 9. S3 performance, maximum read and write throughput.

Fig. 10. Comparing the read throughput of S3 and PVFS on different scales.

devices. Comparing with local storage, EBS volumes show a very poor performance, which is the result of the remote access delay over the network.

Finally, to complete these micro-benchmarks, we set up a software RAID-0 with EBS volumes, varying the number of volumes from 1 to 8. We ran the same benchmark on a c1.medium instance. Fig. 8 shows the write performance on RAID-0 on different number of EBS volumes. Looking at the write throughput, we can observe that the through-put does not vary a lot and is almost constant as the trans-fer size increases. That shows a stable write throughput on EBS drives. The write throughput on the RAID-0 increases with the number of drives. The reason for that is that the data will be spread among the drives and is writ-ten in parallel to all of the drives. That increases the write throughput because of having parallel write instead of serial write. Oddly, the performance does not improve as the number of drives increases from one to two drives. The reason for that is moving from the local writes to net-work. Therefore the throughput stays the same. For 4 EBS volumes, we can observe a 4x increase on the throughput. In case of 8 EBS volumes we expect a 2x speed up com-paring with the 4 EBS experiment. However the write throughput can not scale better because of the limitation of the network bandwith. The maximum achievable throughput is around 120 MB/s, which is bound to the network bandwidth of the instances that is 1 Gb/s. so we can conclude that the RAID throughput will not exceed 120 MB/s if we add more EBS volumes.

2.4.5 S3 and PVFS Performance

In this section we evaluate and compare the performance of S3 and PVFS. S3 is a highly scalable storage service from Amazon that could be used on multinode applications. Also, a very important requirement for most of the scientific
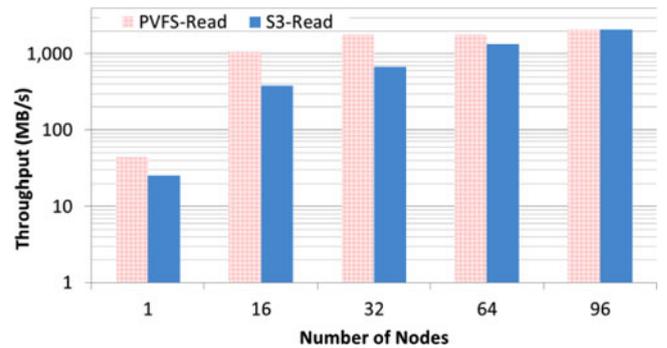
applications is a parallel file system shared among all of the computing nodes. We have also included the NFS as a centralized file system to show how it performs on smaller scales.

First we evaluate the S3 performance on read and write operations. Fig. 9 shows the maximum read and write throughput on S3 accessed by different instance types. Leaving aside the small instances, there is not much differ-ence between the maximum read/write throughput across instances. The reason is that these values are implicitly lim-ited by either the network capabilities or S3 itself.

Next, We compare the performance of the S3 and PVFS as two possible options to use for scientific applications. PVFS is commonly used in scientific applications on HPC environments. On the other hand, S3 is commonly used on the multinode applications that run on cloud environment. We have only included the read performance in this paper. The experiment runs on m1.medium instances. Fig. 10 shows that the read throughput of the S3 is much lower compared to PVFS on small scales. This results from the fact that the S3 is a remote network storage while PVFS is
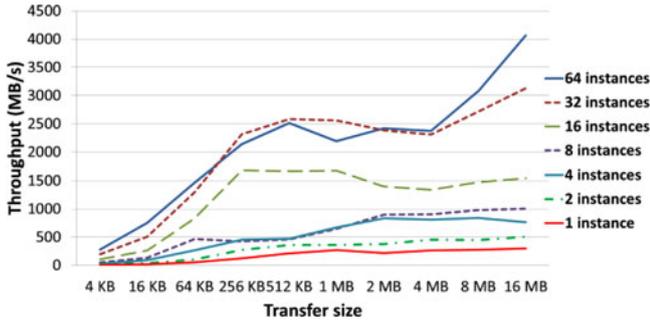
Fig. 11. PVFS read on different transfer sizes over instance storage.

Fig. 13. CDF plot for insert and look up latency on 96 8xxl instances.

installed and is spread over each instance. As The number of the instances increase, PVFS cannot scale as well as the S3 and the performance of the two systems get closer to each other up to a scale that S3 slightly performs better than the PVFS. Therefore it is better to choose S3 if we are using more than 96 instances for the application.

Next, we evaluate the performance of PVFS2 for the scales of 1 to 64 as we found out that it performs better than S3 in smaller scales. To benchmark PVFS2 for the following experiments we use the MPIIO interface instead of POSIX. In the configuration that we used, every node in the cluster serves both as an I/O and metadata server. Fig. 11 shows the read operation throughput of PVFS2 on local storage with different number of instances and variable transfer size. The effect of having a small transfer size is significant, where we see that the throughput inceases as we make the transfer size bigger. Again, this fact is due to the overhead added by the I/O transaction.

Finally, Fig. 12, shows the performance of PVFS2 and NFS on memory through the POSIX interface. The results show that the NFS cluster does not scale very well and the throughput does not increase as we increase the number of nodes. It basically bottlenecks at the 1Gb/s which is the network bandwidth of a single instance. PVFS2 performs better as it can scale very well on 64 nodes on memory. But as we have shown above, it will not scale on larger scales.

2.4.6   DynamoDB performance

In this section we are evaluating the performance of Amazon DynamoDB. DynamoDB is a commonly used NoSql database used by commercial and scientific applications [30]. We conduct micro benchmarks to measure the throughput and latency of insert and look up calls scaling from 1 to 96 instances with total number of calls scaling
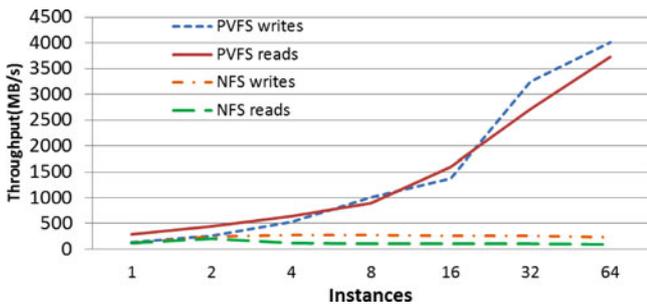
from 10,000 to 960,000 calls. We conduct the benchmarks on both m1.medium and cc2.8xlarge instances. The provision capacity for the benchmarks is 10 K operations/s which is the maximum default capacity available. There is no information released about how many nodes are used to offer a specific throughput. We have observed that the latency of DynamoDB doesn't change much with scales, and the value is around 10 ms. This shows that DynamoDB is highly scalable. Fig. 13 shows the latency of look up and insert calls made from 96 cc2.8xxlarge instances. The average latency for insert and look up are respectively 10 and 8.7 ms. Ninety percent of the calls had a latency of less than 12 ms for insert and 10.5 ms for look up.

We compare the throughput of DynamoDB with ZHT on EC2 [26]. ZHT is an open source consistent NoSql database providing a service which is comparable to DynamoDB in functionality. We conduct this experiment to better understand the available options for having a scalable key-value store. We use both m1.medium and cc2.8xlarge instances to run ZHT. On 96 nodes scale with 2cc.8xlarge instance type, ZHT offers 1,215.0 K ops/s while DynamoDB failed the test since it saturated the capacity. The maximum measured throughput of DynamoDB was 11.5 K ops/s which is found at 64 cc2.8xlarge instance scale. For a fair comparison, both DynamoDB and ZHT have eight clients per node.

Fig. 14 shows that the throughput of ZHT on m1. medium and cc2.8xlarge instances are respectively 59x and 559x higher than DynamoDB on 1 instance scale. On the 96 instance scale they are 20x and 134x higher than the DynamoDB.

Fig. 12. Scalability of PVFS2 and NFS in read/write throughput using memory as storage.

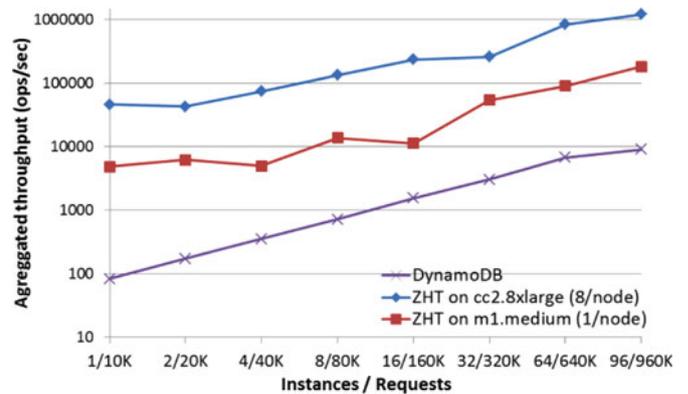Fig. 14. Throughput comparison of DynamoDB with ZHT running on m1. medium and cc2.8xlarge instances on different scales.
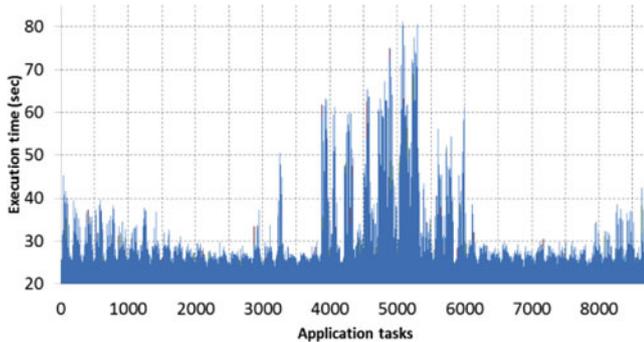
Fig. 15. The LMPS application tasks time distributions.

2.4.7 Workflow Application Performance

In this section we analyze the performance of a complex scientific computing application on the Amazon EC2 cloud. The application investigated is Power Locational Marginal Price Simulation (LMPS), and it is coordinated and run through the Swift parallel programming system [12]. Optimal power flow studies are crucial in understanding the flow and price patterns in electricity under different demand and network conditions. A big computational challenge arising in power grid analysis is that simulations need to be run at high time resolutions in order to capture effect occurring at multiple time scales. For instance, power flows tend to be more constrained at certain times of the day and of the year, and these need to be identified.

The power flow simulation application under study analyzes historical conditions in the Illinois grid to simulate instant power prices on an hourly basis. The application runs linear programming solvers invoked via an AMPL (A Mathematical Programming Language) representation and collects flow, generation, and price data with attached geographical coordinates [25]. A typical application consists of running the model in 8,760 independent executions corresponding to each hour of the year. Each application task execution spans in the range between 25 and 80 seconds as shown in the application tasks time distribution graph in Fig. 15. A snapshot of one such result prices plotted over the map of Illinois is shown in Fig. 16. The prices are in US dollars per megaWatt-hour shown as interpolated contour plots across the areas connected by transmission lines and generation stations shown as lines and circles respectively. A series of such plots could be post processed to give an animated visualization for further analysis in trends etc.

The execution of the application was performed on an increasing number of m1.large instances (see Fig. 17).

For data storage, we use S3. Given that the application scales well to 80 instances, but not beyond that. The performance saturation is a salient point that comes out of Fig. 17. With S3 object store being remote, at 100 VMs it takes long enough to fetch the data that it is dominating execution time. More scalable distributed storage subsystem should be investigated that is geared towards scientific computing, such as PVFS, Lustre, or GPFS.

2.5 Performance Comparison of EC2 versus FermiCloud

In this section we compare the performance of the EC2 as a public cloud with FermiCloud as a private cloud on HPL
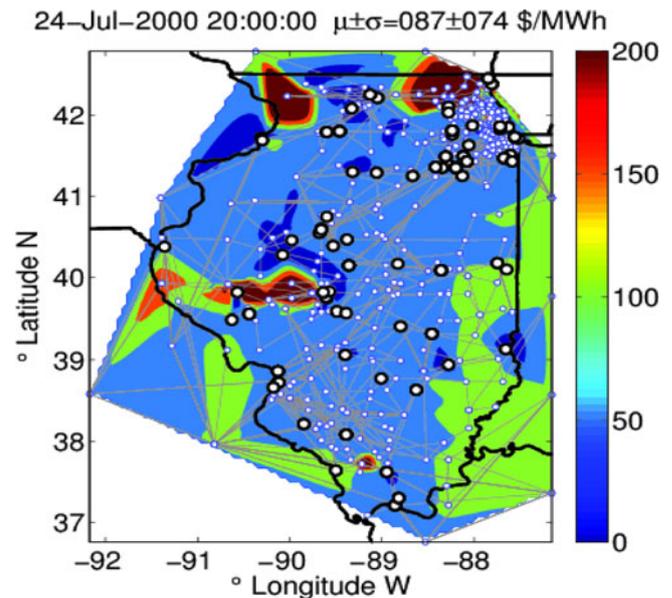

Fig. 16. A contour plot snapshot of the power prices in $/MWh across the state of Illinois for an instance in July 2000

benchmark which is a real HPC application. Before comparing the performance of Amazon on real Applications, we need to compare the raw performance of the two resources.

2.5.1 Raw Performance Comparison

Before comparing the performance of the two infrastructures on real applications like HPL, we need to compare their raw performance on the essential metrics in order to find the root causes of their performance differences. The most effective factors on HPL performance are compute power, and Network latency and bandwidth. We need to compare these factors on the instances with similar functionalities.

On both of the Clouds, we chose the instances that can achieve the highest performance on HPL applications. On EC2, we use c3.8xlarge instances that are enabled with Intel Xeon E5-2,680 v2 (Ivy Bridge) Processors and a 10 Gigabits network adapter with SRIOV technology. On FermiCloud, each server machine is enabled with 2 quad core 2.66 GHz Intel processors, and eight port RAID Controller. On FermiCloud machines are backed by (16 Gigabits effective) Infiniband network adapters.
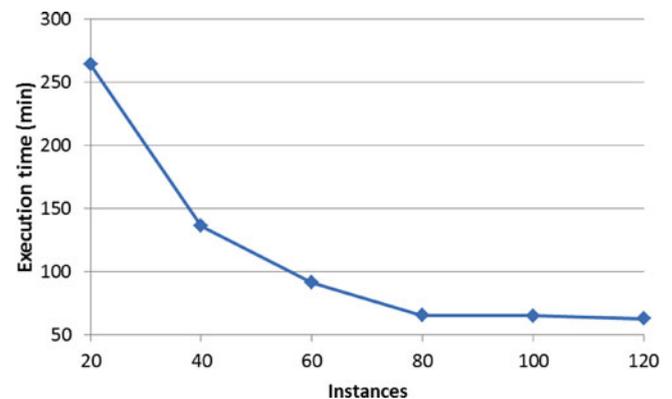

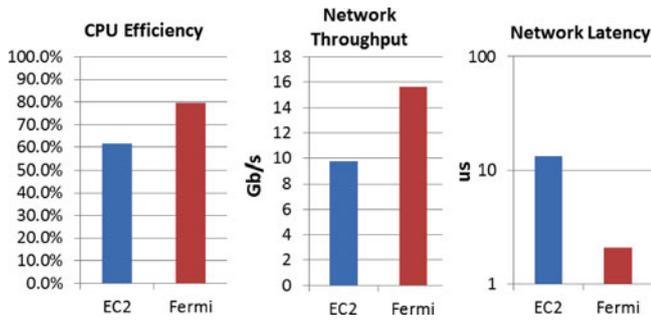Fig. 17. The runtime of LMPS on m1.large instances in different scales.

Fig. 18. Raw performance comparison overview of EC2 versus FermiCloud.

Fig. 19. Efficiency comparison of EC2 and FermiCloud running HPL application on a virtual cluster.

The CPU efficiency is defined as the performance of the VM running HPL on a single VM with no network connectivity, divided by the theoretical peak performance of the CPU. Fig. 18 compares the raw performance of the Amazon EC2 with FermiCloud on CPU and network performance. The results show that the virtualization overhead on Fermi-Cloud instances are slightly lower than the EC2 instances.

The significant difference of the two infrastructures is on the network adapters. The FermiCloud instances are enabled with InfiniBand network adapters and are able to provide higher performance compared to the EC2 instances that have 10 Gigabit network cards. The efficiency of both of the systems on network throughput is high. The network throughput efficiency is defined as the VM network performance divided by the theoretical peak of the device. Fermi-Cloud and EC2 network adapters respectively achieve 97.9 and 97.4 percent efficiency. We used MPIbench to calculate the network latency. There is a 6x difference between the network latency of the two clouds. The latency of the Fermi-Cloud instance is 2.2 us as compared to the latency of EC2 instance which is 13 us. Another important factor is the latency variance. The latency variance on both systems is within 20 percent which is stable. HPL application uses MPI for communication among the nodes. The network latency can decrease the performance of the application by affecting the MPI performance.

2.5.2   HPL Performance Comparison

In this section we evaluate the performance of HPL application on both on a virtual cluster on both FermiCloud and EC2. The main difference on the two infrastructures is on their virtualization layer and the network performance. FermiCloud uses KVM and is enabled with InfiniBand network adapters. EC2 uses its own type of virtualization which is based on Xen hypervisor and has 10 Gigabit network adapters.

The best way to measure the efficiency of a virtual cluster on a cloud environment is defining it as the performance of the VM which include the virtualization overhead divided by the host performance that doesn't include virtualization overhead. We can measure the efficiency as defined for Fer-miCloud since we have access to the host machines. But that is not possible for EC2 since we do not have access to the physical host machines. Therefore we compare the scalability efficiency of the two clouds which is defined as the overhead of the application performance as we scale up the number of cloud instances.
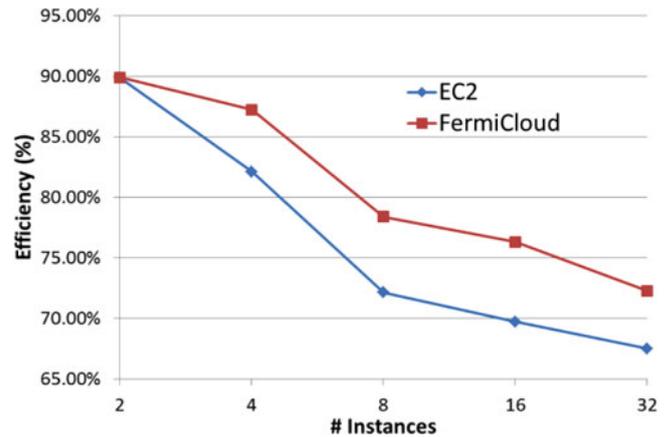
Fig. 19 compares the efficiency of EC2 and FermiCloud running HPL application on a virtual cluster. Due to budget limitations we run the experiment up to 32 instances scale.

The results show that the efficiency is dependent on the network latency. On the two instances scale, both clouds show good efficiency. They only lose 10 percent efficiency that is due to the MPI communications latency added between the instances. Since both of the clouds have relatively powerful network adapters, the communication overhead is still not a bottleneck on two instances scale. As the number of instances increase, the applications processes make more MPI calls to each other and start saturating the network bandwidth. Having InfiniBand network, the Fermi-Cloud loses less efficiency than the EC2. The efficiency of EC2 drops to 82 percent and the efficiency of the Fermi-Cloud drops to 87 percent. The only major difference between the instances of private and public cloud is on their network latency. As a result, we can see that they provide similar efficiency with the private cloud instance being roughly about 5-8 percent more efficient on different scales.

3   COST ANALYSIS

In this section we analyze the cost of the Amazon EC2 cloud from different aspects. We analyze the cost of instances for compute intensive applications as well as for data intensive applications. Our analysis provides suggestions to different cloud users to find the instance type that fits best for certain application with specific requirements. Next section compares the instances based on their memory capacity and performance.

3.1   Memory Cost

This section compares the cost of the memory on Amazon EC2 instances. Fig. 20 compares the cost of instances based on their memory capacity and bandwidth.

The GB/Dollar metric on the left hand side shows the capacity cost effectiveness of the instances. The most cost effective instances for memory capacity are the high memory (m2.2xlarge & m2.4xlarge) instances. But looking at the cost of the memory bandwidth, we can observe that these instances do not have the best memory bandwidth efficiency. The most cost effective instances based on the
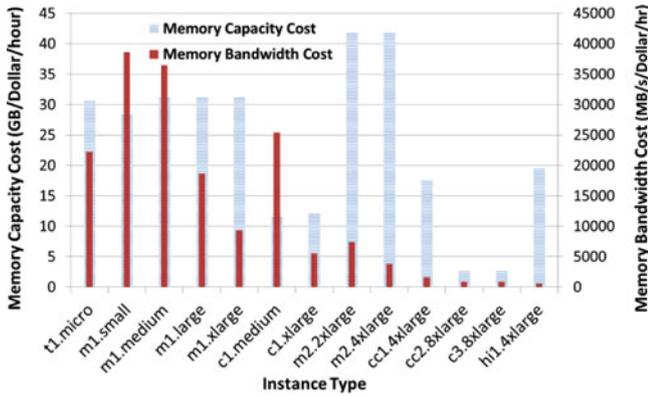
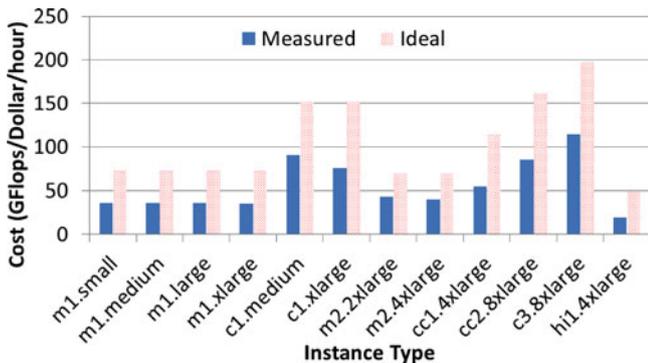Fig. 20. Memory capacity and memory bandwidth cost.
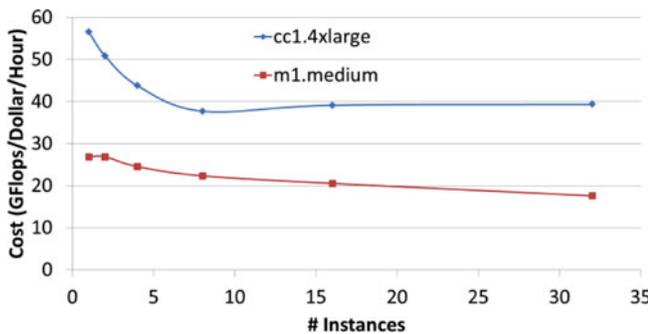

Fig. 21. CPU performance cost of instances.

Fig. 22. Cost of virtual cluster of m1.medium and cc1.4xlarge.

memory bandwidth efficiency are the m1.small and m1.medium instances.

3.2 CPU Cost

In this section we analyze the cost-effectiveness of instances based on the performance of the instances while running compute intensive applications. The metric for our analysis is GFLOPS/Dollar.

Fig. 21 compares the ideal performance cost of the instances based on Amazon claims with their actual performance while running HPL benchmark. The results show that the most cost-effective instance is c3.8xlarge.

3.3 Cluster Cost

We analyze the cost of the virtual clusters set up by m1.medium and cc1.4xlarge instances in different sizes. Fig. 22 compares the cost of the virtual clusters based on their compute performance.
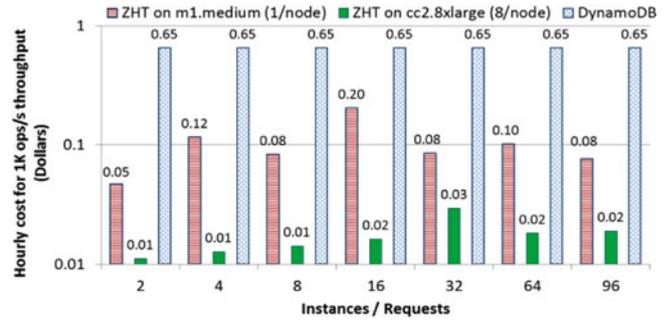

Fig. 23. Cost comparison of dynamoDB with ZHT.

TABLE 1
Performance Summary of EC2 Instances

	CPU bw	Mem. bw	Net. bw	Disk I/O
m1.small	Low	Low	Low	Low
m1.med	Low	Avg	Avg	Low
m1.lrg	Avg	Avg	Avg	Avg
m1.xlrg	Avg	Avg	Avg	Avg
c1.med	Avg	Avg	Avg	Low
c1.xlrg	Avg	High	Avg	Avg
m2.2xlrg	High	High	Avg	Avg
cc1.4xlrg	High	High	High	Avg
cc2.8xlrg	High	High	High	Avg
c3.8xlrg	High	High	High	High
hi1.lrg	High	Avg	High	High

3.4 DynamoDB Cost

Finally in this section we evaluate the cost of DynamoDB. In order to better understand the value of offered service, we compare the cost with the cost of running ZHT on EC2 on different instance types.

Fig. 23 shows the hourly cost of 1,000 ops/s capacity offered by DynamoDB compared to the equal capacity provided by ZHT from the user point of view.

We are comparing the two different scenarios of cost of using a free application on rented EC2 instances versus getting the service from DynamoDB. In case of DynamoDB, since the users pays for the capacity that they get, the number of instances doesn't affect the cost. That's why the cost of DynamoDB is always constant. For ZHT, the system efficiency and performance varies on different scales hence the variation in costs for ZHT at different scales. Since the cc2.8xlarge instances provide much better performance per money spent, the cost per operation is as good as 65X lower than DynamoDB. However, the better costs come at the complexity of managing a virtual cluster of machines to operate ZHT. It is likely that for low loads including sporadic requirements for DynamoDB, it makes financial sense to run on Amazon AWS services, but for higher performance requirements it is much more beneficial to simply operate a dedicated ZHT system over EC2 resources.

3.5 Performance and Cost Summary

This section summarizes the performance and the cost efficiency of Amazon EC2 and other services of AWS. Table 1 shows the performance overview of the different instance types on EC2. The performance results of the instances mostly match with the prediction based on the claims of

TABLE 2
Cost-Efficiency Summary of EC2 Instances

	CPU bw	Mem. Cap.	Mem. bw	Net. bw
m1.small	Avg	Avg	High	High
m1.med	Avg	Avg	High	High
m1.lrg	Avg	Avg	Avg	Avg
m1.xlrg	Avg	Avg	Low	Low
c1.med	High	Low	High	Low
c1.xlrg	High	Low	Low	Low
m2.2xlrg	Avg	High	Low	Low
cc1.4xlrg	Avg	Avg	Low	Low
cc2.8xlrg	High	Avg	Low	Avg
c3.8xlrg	High	Avg	Low	Avg
hi1.lrg	Low	Low	Low	Low

TABLE 3
Performance and Cost-Efficiency Summary of AWS Services

	Scalability	Cost-efficiency	Data Granularity
S3	High	High	Large data
DynamoDB	High	Low	Small data

Amazon. There have been anomalies in some of the specific instance types. Instances like m1.xlarge have average performance while m1.medium instance has shown a performance that was higher than expected.

Table 2 summarizes the cost-efficiency of instance types of EC2. The compute optimized instances show better cost efficiency. Finally Table 3 summarizes the performance of S3 and DynamoDB.

4 RELATED WORK

There have been many researches that have tried to evaluate the performance of Amazon EC2 cloud [14], [16], [17]. However the experiments were mostly run on limited types and number of instances. Therefore they lack the generality in their results and conclusions, as they have not covered all instance types.

Ostermann et al. have evaluated Amazon EC2 using micro-benchmarks in different performance metrics. However their experiments do not include the more high-end instances that are more competitive to HPC systems. Moreover, the Amazon performance has improved since then and more features have been added to make it useful for HPC applications [14]. In addition to the experiments scope of that paper, our work provides the evaluations of the raw performance of a variety of the instances including the high-end instances, as well as the performance of the real applications.

He et al. have deployed a NASA climate prediction application into major public clouds, and compared the results with dedicated HPC systems results. They have run micro-benchmarks and real applications [15]. However they only run their experiments on small number of VMs. We have evaluated the performance of EC2 on larger scales.

Jackson et al. have deployed a full application that performs massive file operations and data transfer on Amazon EC2 [18]. The research mostly focuses on different storage options on Amazon.

Walker evaluates the performance of EC2 on NPB benchmarks and compares their performance on EC2 versus NCSA ABE supercomputer on limited scale of 1 and 4 instances [36]. The paper suffices to bring the results without detailed analysis and does not identify what this gap contributes to. Other papers have run the same benchmark on different infrastructures and provided better analysis of the results [15], [35].

Only a few of the researches that measure the applicability of clouds for scientific applications have used the new Amazon EC2 cluster instances that we have tested [10], [20], [24]. Mehrotra compares the performances of Amazon EC2 HPC instances to that of NASA's Pleiades supercomputer [10]. However the performance metrics in that paper is very limited. They have not evaluated different performance metrics of the HPC instances. Ramakrishnan et al. have measured the performance of the HPCC benchmarks [20]. They have also applied two real applications of PARATEC and MILC.

Juve et al. investigate different options of data management of the workflows on EC2 [24]. The paper evaluates the runtime of different workflows with different underlying storage options. The aforementioned works have not provided a comprehensive evaluation of the HPC instances. Their experiments are limited to a few metrics. Among the works that have looked at the new HPC instances, our work is the only one that has evaluated all of the critical performance metrics such as memory, compute, and network performance.

Jackson compares the conventional HPC platforms to EC2 using real applications on small scales. The evaluation results show poor performance from EC2 virtual cluster running scientific applications. However they haven't used HPC instances, and have used instances with slower interconnects. Apart from the virtualization overhead, the instances are not quite comparable to highly tuned nodes on the super computers [21].

Many works have covered the performance of public clouds without having an idea about the host performance of the nodes without virtualization overhead [14], [15], [16]. Younge et al. have evaluated the performance of different virtualization techniques on FutureGrid private cloud [11]. The focus of that work is on the virtualization layer rather than the cloud infrastructure.

Gupta et al. in identify the best fit for the cloud among the HPC applications [35]. He investigates the co-existence of the cloud with super computers and suggests a hybrid infrastructure run for HPC applications that fit into the cloud environment. The paper also provides the cost analysis of running cloud on different HPC applications and shows where it is beneficial to use cloud.

Many papers have analyzed the cost of the cloud as an alternative resource to dedicated HPC resources [18], [19], [25]. Our work covers the storage services performance both on micro-benchmarks as well as the performance while being used by data-intensive applications.

Our work is unique in a sense that it provides comprehensive evaluation of EC2 cloud in different aspects. We first evaluate the performance of all instance types in order to better identify their potentials and enable users to choose the best instances for different use case scenarios. After

identifying the potentials, we compare the performance of the public cloud and a private cloud on different aspects, running both microbenchmarks and real scientific applications. Being able to measure the virtualization overhead on the FermiCloud as a private cloud, we could provide a more realistic evaluation of EC2 by comparing it to the FermiCloud.

Another important feature of the Cloud is having different services. We provide a broader view of EC2 by analyzing the performance of cloud services that could be used in modern scientific applications. More scientific frameworks and applications have turned into using cloud services to better utilize the potential of Cloud [12], [31]. We evaluate the performance of the services such Amazon S3 and DynamoDB as well as their open source alternatives running on cloud. Finally, this work is unique in comparing the cost of different instances based on major performance factors in order to find the best use case for different instances of Amazon EC2.

5 CONCLUSION

In this paper, we present a comprehensive, quantitative study to evaluate the performance of the Amazon EC2 for the goal of running scientific applications. We first evaluate the performance of various instance types by running micro benchmarks on memory, compute, network and storage. In most of the cases, the actual performance of the instances is lower than the expected performance that is claimed by Amazon. The network bandwidth is relatively stable. The network latency is higher and less stable than what is available on the supercomputers. Next, based on the performance of instances on micro-benchmarks, we run scientific applications on certain instances. We finally compare the performance of EC2 as a commonly used public cloud with FermiCloud, which is a higher-end private cloud that is tailored for scientific for scientific computing.

We compare the raw performance as well as the performance of the real applications on virtual clusters with multiple HPC instances. The performance and efficiency of the two infrastructures is quite similar. Their only difference that affects their efficiency on scientific applications is the network bandwidth and latency which is higher on FermiCloud. FermiCloud achieves higher performance and efficiency due to having InfiniBand network cards. We can conclude that there is need for cloud infrastructures with more powerful network capacity that are more suitable to run scientific applications.

We evaluated the I/O performance of Amazon instances and storage services like EBS and S3. The I/O performance of the instances is lower than performance of dedicated resources. The only instance type that shows promising results is the high-IO instances that have SSD drives on them. The performance of different parallel file systems is lower than performance of them on dedicated clusters. The read and write throughput of S3 is lower than a local storage. Therefore it could not be a suitable option for scientific applications. However it shows promising scalability that makes it a better option on larger scale computations. The performance of PVFS2 over EC2 is convincible for using in scientific applications that require a parallel file system.

Amazon EC2 provides powerful instances that are capable of running HPC applications. However, the performance a major portion of the HPC applications are heavily dependent on network bandwidth, and the network performance of Amazon EC2 instances cannot keep up with their compute performance while running HPC applications and become a major bottleneck. Moreover, having the TCP network protocol as the main network protocol, all of the MPI calls on HPC applications are made on top of TCP protocol. That would add a significant overhead to the network performance. Although the new HPC instances have higher network bandwidth, they are still not on par with the non-virtualized HPC systems with high-end network topologies. The cloud instances have shown to be performing very well, while running embarrassingly parallel programs that have minimal interaction between the nodes [10]. The performance of embarrassingly parallel application with minimal communication on Amazon EC2 instances is reported to be comparable with non-virtualized environments [21], [22].

Armed with both detailed benchmarks to gauge expected performance and a detailed price/cost analysis, we expect that this paper will be a recipe cookbook for scientists to help them decide between dedicated resources, cloud resources, or some combination, for their particular scientific computing workload.

REFERENCES

[1] Amazon EC2 Instance Types, Amazon Web Services. (2013). [Online]. Available: http://aws.amazon.com/ec2/instance-types/
[2] Amazon Elastic Compute Cloud (Amazon EC2), Amazon Web Services. (2013). [Online]. Available: http://aws.amazon.com/ec2/
[3] Amazon Simple Storage Service (Amazon S3), Amazon Web Services. (2013). [Online]. Available: http://aws.amazon.com/s3/
[4] Iperf, Souceforge. (2011, Jun.). [Online]. Available: http://sourceforge.net/projects/iperf/
[5] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary HPL, (netlib.org). (2008, Sep.). [Online]. Available: http://www.netlib.org/benchmark/hpl/
[6] J. J. Dongarra, S. W. Otto, M. Snir, and D. Walker, "An introduction to the MPI standard," Univ. of Tennessee, Knoxville, TN, USA, Tech. Rep. CS-95–274, Jan. 1995.
[7] Release: Amazon EC2 on 2007–07–12, Amazon Web Services. (2013). [Online]. Available: http://aws.amazon.com/releasenotes/Amazon-EC2/3964
[8] K. Yelick, S. Coghlan, B. Draney, and R. S. Canon, "The magellan report on cloud computing for science," US Dept. of Energy, Washington DC, USA, 2011, http://www.nersc.gov/assets/StaffPublications/2012/MagellanFinalReport.pdf
[9] L. Ramakrishnan, R. S. Canon, K. Muriki, I. Sakrejda, and N. J. Wright, "Evaluating Interconnect and virtualization performance for high performance computing," *ACM Perform. Eval. Rev.*, pp. 55–60,vol. 40, 2012.
[10] P. Mehrotra J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Laza-noff, S. Saini, and R. Biswas, "Performance evaluation of Amazon EC2 for NASA HPC applications," in *Proc. 3rd Workshop Scientific Cloud Comput.*, 2012, pp. 41–50.

[11] A. J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, J. Qiu, and G. C. Fox, "Analysis of virtualization technologies for high performance computing environments," in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, 2011, pp. 9–16.

[12] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, I. Raicu, T. Stef-Praun, and M. Wilde, "Swift: Fast, reliable, loosely coupled parallel computation," in *Proc. IEEE Int. Workshop Scientific Workflows*, 2007, pp. 199–206.

[13] I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, B. Clifford, "Towards loosely-coupled programming on petascale systems," *IEEE/ACM Supercomput.*, pp. 1–12, 2008.

[14] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of EC2 cloud computing services for scienti c computing," in *Proc. 1st Int. Conf. Cloud Comput*, 2009, pp. 115–131.

[15] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, "Case study for running HPC applications in public clouds," in *Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput.*, 2010, pp. 395--401.

[16] G. Wang and T. S. Eugene Ng., "The Impact of virtualization on network performance of amazon EC2 data center," in *Proc. IEEE 29th Conf. Inform. Commun.*, 2010, pp. 1163–1171.

[17] S. L. Garfinkel, "An evaluation of amazon's grid computing services: Ec2, s3 and sqs," Comput. Sci. Group, Harvard Univ, Cambridge, MA, USA, Tech. Rep. tR-08–07, 2007.

[18] K. R. Jackson, K. Muriki, L. Ramakrishnan, K. J. Runge, and R. C. Thomas, "Performance and cost analysis of the supernova factory on the amazon aws cloud," *Scientific Programm.*, vol. 19, no. 2–3, pp. 107–119, 2011.

[19] J.-S. Vockler, G. Juve, E. Deelman, M. Rynge, and G. B. Berriman, "Experiences using cloud computing for a scientific workflow application," in *Proc. 2nd Workshop Scientific Cloud Comput.*, 2011, pp. 15–24.

[20] L. Ramakrishnan, P. T. Zbiegel, S. Campbell, R. Bradshaw, R. S. Canon, S. Coghlan, I. Sakrejda, N. Desai, T. Declerck, and A. Liu, "Magellan: Experiences from a science cloud," in *Proc. 2nd Int. Workshop Scientific Cloud Comput.*, 2011, pp. 49–58.

[21] 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 *Proc. 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci.*, 2010, pp. 159–168.

[22] J.-S. Vockler, G. Juve, E. Deelman, M. Rynge, and G. B. Berriman, "Experiences using cloud computing for a scientific workflow application," in *Proc. 2nd Workshop Scientific Cloud Comput.*, 2011, pp. 15–24.

[23] J. Lange, K. Pedretti, P. Dinda, P. Bridges, C. Bae, P. Soltero, A. Merritt, "Minimal overhead virtualization of a large scale supercomputer," in *Proc. 7th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ*, 2011, pp. 169–180.

[24] G. Juve, E. Deelman, G. B. Berriman, B. P. Berman, and P. Maechling, "An evaluation of the cost and performance of scientific workflows on amazon EC2," *J. Grid Comput.*, vol. 10, no. 1, p. 5–21, 2012.

[25] R. Fourer, D. M. Gay, and B. Kernighan, *Algorithms and Model Formulations in Mathematical Programming*. New York, NY, USA: Springer-Verlag , 1989, ch. *AMPL: A mathematical programming language*, pp. 150–151.

[26] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, "ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table," in *Proc. IEEE 27th Int. Parallel Distrib. Process. Symp.*, 2013, pp. 775–787.

[27] FermiCloud, Fermilab Computing Sector. (Apr. 2014). [Online]. Available: http://fclweb.fnal.gov/

[28] R. Moreno-vozmendiano, S. Montero, and I. Llorente, "IaaS cloud architecture: From virtualized datacenters to federated cloud infrastructures: Digital Forensics," *IEEE Comput.*, vol. 45, no. 12, pp. 65–72, Dec. 2012.

[29] K. Hwang, J. Dongarra, and G. C. Fox, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. San Mateo, CA, USA: Morgan Kaufmann, 2011

[30] W. Voegels. (2012, Jan.). Amazon DynamoDB—a fast andscalable NoSQL database service designed for Internet-scale applications. [Online]. Available: http://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html

[31] I. Sadooghi S. Palur, A. Anthony, I. Kapur, K. Ramamurty, K. Wang, and I. Raicu, "Achieving efficient distributed scheduling with message queues in the cloud for many-task computing and high-performance computing," in *Proc. 14th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2014, pp. 404 –413.

[32] T. L. Ruivo, G. B. Altayo, G. Garzoglio, S. Timm, K. Hyun, N. Seo-Young, and I. Raicu, "Exploring infiniband hardware virtualization in opennebula towards efficient high-performance computing," in *Proc. 14th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2014, pp. 943–948.

[33] D. Macleod, (2013). OpenNebula KVM SR-IOV driver.

[34] K. Maheshwari, K. Birman, J. Wozniak, and D. V. Zandt, "Evaluating cloud computing techniques for smart power grid design using parallel scripting," in *Proc. 14th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.* 2013, pp. 319–326.

[35] A. Gupta, L. Kale, F. Gioachin, V. March, C. Suen, P. Faraboschi, R. Kaufmann, and D. Milojicic, "The who, what, why and how of high performance computing applications in the cloud," HP Labs, Tech. Rep. HPL-2013-49, Jul. 2013.

[36] E. Walker, "Benchmarking amazon EC2 for high-performance scientific computing," *USENIX; Login: Mag.*, vol. 33, no. 5, pp. 18–23, Oct. 2008.

Iman Sadooghi received the MS degree in software engineering from San Jose State University. He is a 4th year PhD student in the Department of the Computer Science of the Illinois Institute of Technology and a member of DataSys laboratory. His research interests include distributed systems and cloud computing. He has two publications in the area of cloud computing. He is a member of the IEEE and the IEEE Computer Society.

Jesus Hernandez Martin received the MS degree in computer science from the Illinois Institute of Technology in 2012. He also received the BS + MS degree in computer engineering from the Technical University of Madrid during the same year. He is a data engineer at Lyst. His interests range from distributed systems, data processing, and analytics to high-performance client-server architectures.
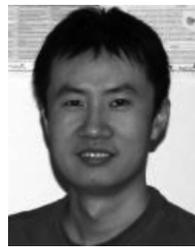
Tonglin Li received the BE and ME degrees in 2003 and 2009, respectively, from Xi'an Shiyou University, China. He is a 5th year PhD student working in Datasys Lab, the Computer Science Department at the Illinois Institute of Technology, Chicago. His research interests are in distributed systems, cloud computing, data-intensive computing, supercomputing, and data management. He is a member of the IEEE and ACM.

Kevin Brandstatter is a 4th year undergraduate student in the Department of Computer Science at the Illinois Institute of Technology. He has worked as a research assistant in the Datasys lab under Dr. Raicu for the past three years. His research interests primarily include distributed storage systems such as distributed file systems and distributed key value storage.

Ketan Maheshwari is a postdoctoral researcher in the Mathematics and Computer Science Division at the Argonne National Laboratory. His research interests include applications for parallel scripting on distributed and high-performance resources. His main activities in recent years involve design, implementation and execution of parallel applications on clouds, XSEDE, Cray XE6, and IBM supercomputers at the University of Chicago. The applications include massive protein docking, weather and soil modeling, earthquake simulations, and power grid modeling funded by DOE and NIH.

Yong Zhao is a professor at the University of Electronic Science and Technology of China. His research interests include big data, cloud computing, grid computing, data intensive computing, extreme large-scale computing, and cloud workflow. He has published more than 40 papers in international computer books, journals, and conferences, which are referenced more than 4,000 times; He has chaired/co-chaired ACM/IEEE Workshop on Many Task Computing on Grids Clouds and Supercomputers, Workshop on Data Intensive Computing in the Clouds, and IEEE International Workshop on CloudFlow 2012, 2013.

Tiago Pais Pitta de Lacerda Ruivo received the MS graduate degree in computer science from the Illinois Institute of Technology and former Graduate Research Intern in the Fermi National Accelerator Laboratory. His research interests include network virtualization and mobile technologies.

Gabriele Garzoglio received the Laura degree in physics from the University of Genova, Italy, and the PhD degree in computer science from DePaul University. He is the head of the Grid and Cloud Services Department of the Scientific Computing Division at Fermilab and he is deeply involved in the project management of the Open Science Grid. He oversees the operations of the Grid services at Fermilab..

Steven Timm received the PhD degree in physics from Carnegie Mellon. He is an associate department head for the Cloud Computing in the Grid and Cloud Services Department of the Scientific Computing Division at Fermi National Accelerator Laboratory. Since 2000, he has held various positions on the staff of Fermilab relating to Grid and Cloud Computing. He has been the lead of the FermiCloud project since its inception in 2009.

Ioan Raicu is an assistant professor in the Department of Computer Science at the Illinois Institute of Technology, as well as a guest research faculty in the Math and Computer Science Division at the Argonne National Laboratory. He is also the founder and director of the Data-Intensive Distributed Systems Laboratory at IIT. His research work and interests include the general area of distributed systems. His work focuses on a relatively new paradigm of Many-Task Computing (MTC), which aims to bridge the gap between two predominant paradigms from distributed systems, HTC, and HPC. His work has focused on defining and exploring both the theory and practical aspects of realizing MTC across a wide range of large-scale distributed systems. He is particularly interested in resource management in large-scale distributed systems with a focus on many-task computing, data intensive computing, cloud computing, grid computing, and many-core computing.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.