CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2011; **00**:1–18
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

General-Purpose Computation on GPUs for High Performance Cloud Computing

Roberto R. Expósito*,†, Guillermo L. Taboada, Sabela Ramos, Juan Touriño and Ramón Doallo

Computer Architecture Group, Department of Electronics and Systems, University of A Coruña, Spain

SUMMARY

Cloud computing is offering new approaches for High Performance Computing (HPC) as it provides dynamically scalable resources as a service over the Internet. In addition, General-Purpose computation on Graphical Processing Units (GPGPU) has gained much attention from scientific computing in multiple domains, thus becoming an important programming model in HPC. Compute Unified Device Architecture (CUDA) has established as a popular programming model for GPGPUs, removing the need for using the graphics APIs for computing applications. OpenCL (Open Computing Language) is an emerging alternative not only for GPGPU but also for any parallel architecture. GPU clusters, usually programmed with a hybrid parallel paradigm mixing Message Passing Interface (MPI) with CUDA/OpenCL, are currently gaining high popularity. Therefore, cloud providers are deploying clusters with multiple GPUs per node and high-speed network interconnects in order to make them a feasible option for HPC as a Service (HPCaaS). This paper evaluates GPGPU for High Performance Cloud Computing on a public cloud computing infrastructure, Amazon EC2 Cluster GPU Instances (CGI), equipped with NVIDIA Tesla GPUs and a 10 Gigabit Ethernet network. The analysis of the results, obtained using up to 64 GPUs and 256 processor cores, has shown that GPGPU is a viable option for High Performance Cloud Computing despite the significant impact that virtualized environments still have on network overhead, which still hampers the adoption of GPGPU communication-intensive applications. Copyright © 2011 John Wiley & Sons, Ltd.

KEY WORDS: Cloud Computing; General-Purpose computation on GPU (GPGPU); High Performance Computing (HPC); 10 Gigabit Ethernet; CUDA; OpenCL; MPI

1. INTRODUCTION

Cloud computing [1] is an Internet-based computing model that enables convenient, on-demand network access to a shared pool of configurable and virtualized computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort. Public clouds offer access to external users who are typically billed on a pay-as-you-use basis. With the cloud, and the availability of multiple cloud service providers,

*Correspondence to: Roberto R. Expósito, Department of Electronics and Systems, University of A Coruña, Campus de Elviña s/n, 15071, A Coruña, Spain. Tel.: +34 981167000; fax: +34 981167160.
†E-mail: rreye@udc.es

organizations no longer are forced to invest in additional technology infrastructure. They can just leverage the infrastructure provided by the cloud service provider, or move their own applications to this infrastructure. Customers can derive significant economies of use by leveraging the pay-by-use model, instead of upgrading their infrastructure, dimensioned to handle peak requests.

Cloud computing has been driven from the start predominantly by the industry through Amazon, Google and Microsoft, but due to its potential benefits this model has been also adopted by academia, as it is well-suited for handling peak demands in resource-intensive applications in sciences and engineering. Thus, cloud computing is an option for High Performance Computing (HPC), where the use of cloud infrastructures for HPC applications has generated considerable interest in the scientific community [2, 3, 4, 5, 6], which has coined the term HPC as a Service (HPCaaS), an extension of the provision of Infrastructure as a Services (IaaS).

HPC workloads typically require low latency and high bandwidth inter-processor communication to provide scalable performance. However, the widely extended use of commodity interconnect technologies (Ethernet and TCP/IP) and the overhead of the virtualized access to the network limits severely the scalability of HPC applications in public cloud infrastructures.. To overcome these constraints cloud infrastructure providers are increasingly deploying high-speed networks (e.g., 10 Gigabit Ethernet and InfiniBand), widely extended in HPC environments, where message-passing middleware is the preferred choice for communications. MPI [7] is the de facto standard in message-passing interface as it generally provides HPC applications with high scalability on clusters with high-speed networks.

The advent of many-core accelerators, such as Graphical Processing Units (GPU), to HPC is already consolidated thanks to the outbreak of General-Purpose computing on GPUs (GPGPU [8, 9]). The massively parallel GPU architecture, together with its high floating point performance and memory bandwidth is well-suited for many workloads in science and engineering, even outperforming multicore processors, which has motivated the incorporation of GPUs as HPC accelerators [10]. As a result, new parallel programming models like Compute Unified Device Architecture (CUDA) [11] and Open Computing Language (OpenCL) [12] have emerged to expose the parallel capabilities of GPUs to GPGPU programmers in a productive way [13]. These models can be combined with well-established HPC programming models such as MPI [14].

Amazon Elastic Compute Cloud (Amazon EC2) [15] is an IaaS service that provides users with access to on-demand computational resources to run their applications. Amazon EC2 supports the management of the resources through a Web service or an API which is able, among other tasks, to boot straight-forwardly an Amazon Machine Image (AMI) into a custom virtual machine (a VM or "instance"). Amazon EC2 Cluster Compute Instances (CCI) [16]), a resource available since July 2010, provide a significant CPU power (two quad-core processors), together with a high performance 10 Gigabit Ethernet network, thus targeting HPC applications. In November 2010 Amazon EC2 introduced Cluster GPU Instances (CGI), which have the same configuration as CCI plus two NVIDIA Tesla GPUs. HPC applications running on CGI are expected to benefit significantly from the massive parallel processing power the GPUs provide, as well as from their full-bisection high bandwidth network (10 Gigabit Ethernet). This is particularly valuable for applications that rely on messaging middleware like MPI for communications while taking advantage of GPGPU, such as hybrid MPI/CUDA or MPI/OpenCL codes. This paper evaluates GPGPU for High Performance Cloud Computing on Amazon EC2 cloud infrastructure, the largest

DOI: 10.1002/cpe

public cloud in production, using up to 32 CGIs, demonstrating the viability of providing HPC as a service taking advantage of GPGPU in a cluster of CGIs.

The structure of this paper is as follows: Section 2 presents the related work. Section 3 introduces the experimental configuration, both hardware and software, and the methodology of the evaluation conducted in this work. Section 4 analyzes the performance results obtained by representative benchmarks on the experimental testbed on Amazon EC2 public cloud. Section 5 summarizes our concluding remarks.

2. RELATED WORK

There has been a spur of research activity in assessing the performance of virtualized resources in cloud computing environments [17, 18, 19]. The suitability of these resources for HPC increases as their performance improves, which has motivated lately several projects on the adoption of cloud computing for HPC applications. Among them, a popular topic is the feasibility of Amazon EC2 for HPC applications, which has been discussed in several papers since 2008, next presented.

Deelman et al. [20] explored the application of cloud computing for science, examining the tradeoffs of different workflow execution modes and provisioning plans for cloud resources. In [2], Evangelinos and Hill analyze the performance of HPC benchmarks on EC2 cloud infrastructure. These authors reveal an important drawback in network performance, as messaging middleware obtains latencies and bandwidths around one and two orders of magnitude inferior to supercomputers. In [3], Walker evaluated the performance of Amazon EC2 High-CPU instances for high-performance scientific applications, reporting significantly lower performance than traditional HPC clusters. His work also presented the comparative performance evaluation between Amazon EC2 High-CPU instances and an InfiniBand cluster with similar hardware configuration, reporting 40%-1000% runtime increase on EC2 resources.

Buyya et al. [4] discussed the potential opportunities of high-performance scientific applications on public clouds through some practical examples on Amazon EC2 assessing that the trade-offs between cost and performance have to be considered. Ekanayake and Fox [21] compared applications with different communication and computation complexities and observed that latency-sensitive applications experience higher performance degradation than bandwidth-sensitive applications. Ostermann et al. [22] presented an evaluation of the feasibility of Amazon EC2 cloud computing services for scientific computing. They analyzed the performance of the Amazon EC2 standard and High-CPU instances using representative micro-benchmarks and kernels. The main conclusion of their work is that Amazon EC2 required, in 2009, an order of magnitude higher performance in their instances to be feasible its use by the scientific community. Napper and Bientinesi [5] examined the performance of the Linpack benchmark on several EC2 instance types (standard and high-CPU instances). They concluded that clouds can not compete conventional HPC clusters (supercomputers and high-speed clusters) based on the metric GFLOPS/$, since memory and network performance is insufficient to compete existing scalable HPC systems.

Jackson et al. [6] performed a comprehensive evaluation comparing conventional HPC platforms to Amazon EC2 standard instances, using real applications representative of the workload at a typical supercomputing center. Their main conclusion is that the interconnection network

Prepared using cpeauth.cls
DOI: 10.1002/cpe

of the evaluated instances (Gigabit Ethernet at that time) severely limits performance and causes significant runtime variability. Wang and Eugene [23] studied the impact of virtualization on network performance. They presented a quantitative study of the network performance among Amazon EC2 High-CPU instances, detecting unstable TCP/UDP throughput caused by virtualization and processor sharing. Rehr et al. [24] confirmed that Amazon EC2, using standard and High-CPU instance types, is a feasible platform for applications that do not demand high-performance network performance. He et al. [25] extended previous research to three public clouds (including Amazon) running a real application in addition to classical benchmarks to compare the cloud results with those from dedicated HPC systems. Their main conclusion is that public clouds are not designed for running scientific applications primarily due to their poor network performance. Iosup et al. [26] investigated the presence of a Many-Task Computing (MTC) component in existing scientific computing workloads and then they performed an empirical performance evaluation of this MTC component on four public computing clouds, including Amazon EC2 using standard and High-CPU instance types. Their main conclusion is that the computational performance of the evaluated clouds is low, which is insufficient for scientific computing at large, but cloud computing can be a good solution for instant and temporal resource needs.

The main drawback detected by most of these previous works since 2005 up to 2010, is the high network overhead due to the use of commodity interconnection technologies (e.g., Ethernet and TCP/IP) that are not suitable for HPC. In order to overcome this drawback, Amazon EC2 introduce 10 Gigabit Ethernet as interconnection technology together with the launch of its CCI and CGI instance types, in June and November 2010, respectively. These instances target HPC applications thanks to its high computational power and the adoption of a high performance network, 10 Gigabit Ethernet. The CCI instance type has been evaluated in recent related work. Thus, Regola and Ducom [27] evaluated the suitability of several virtualization technologies for HPC, showing that operating system virtualization was the only solution that offers near native CPU and I/O performance. They included in their testbed four Amazon EC2 CCIs, although they focused more on the overall performance of the several evaluated hypervisors instead of the network performance and the scalability of HPC applications. Carlyle et al. [28] reported that it is much more effective in academia operating a community cluster program than providing HPC resources with nodes using Amazon EC2 CCIs. Sun et al. [29] relied on Amazon EC2 CCIs for running the Lattice Optimization and some additional performance benchmarks concluding that these instances suffer from performance degradation for large scale parallel MPI applications, especially network or memory bound applications. Ramakrishnan et al. [30] analyzed the performance of a number of different interconnect technologies, including 10 Gigabit Ethernet network from Amazon EC2 CCIs, showing that virtualization can have a significant impact on performance. Zhai et al. [31] conducted a comprehensive evaluation of MPI applications on Amazon EC2 CCIs, revealing a significant performance increase compared to previous evaluations on standard and High-CPU instances. However, the interconnection network overhead, especially the high startup latency (poor small messages performance), remains as the main MPI scalability limitation.

Additionally, some works have evaluated the performance of other public cloud computing services, such as a seminal study at Amazon S3 by Palankar et al. [32], which also includes an evaluation of file transfer between Amazon EC2 and S3.

DOI: 10.1002/cpe

This exhaustive review of the related works on evaluating Amazon for HPC applications has revealed the lack, to the best of our knowledge in December 2011, of assessments of Amazon EC2 CGIs performance, as well as analyses on the viability of providing HPC as a service in a public cloud taking advantage of message-passing and GPGPU.

3. EXPERIMENTAL CONFIGURATION AND EVALUATION METHODOLOGY

The evaluation of GPGPU for High Performance Cloud Computing has been done on a public cloud computing infrastructure, Amazon EC2, using the Cluster GPU Instances (CGI), which target HPC applications. This section presents the description of the CGI-based platform used in the evaluation (Section 3.1), an overview of the virtualization technologies available in Amazon EC2 (Section 3.2), the description of the GPGPU codes used in the evaluation (Section 3.3) and finally, this section concludes with the methodology followed in this evaluation (Section 3.4).

3.1. Amazon EC2 CGI Platform

The evaluation of GPGPU on Amazon EC2 has been carried out on a cluster of 32 CGIs, whose main characteristics are presented in Table I (CGI) and Table II (cluster of CGIs). A CGI node has 8 cores, each of them capable of executing 4 floating-point operations per clock cycle in double precision (DP), hence 46.88 GFLOPS (Floating-Point Operations per Second) per processor, 93.76 GFLOPS per node and 3000 GFLOPS in the 32 node (256 core) cluster. Moreover, each GPU comes with 3 GB GDDR5 of memory and has a peak performance of 515 GFLOPS in double precision, hence 1030 GFLOPS per node and 32960 GFLOPS in the 32 node (64 GPU) cluster. Aggregating CPU and GPU theoretical peak performances each node provides 1124 GFLOPS, hence the entire cluster provides 35960 GFLOPS.

Table I. Description of the Amazon EC2 Cluster GPU Instance (CGI)

CPU	2 × Intel(R) Xeon quadcore X5570 @2.93 GHz (46.88 GFLOPS DP each CPU)
EC2 Compute Units	33.5
GPU	2 × NVIDIA Tesla “Fermi” M2050 (515 GFLOPS DP each GPU)
Memory	22 GB DDR3
Storage	1690 GB
Virtualization	Xen HVM 64-bit platform (PV drivers for I/O)
API name	cg1.4xlarge

Table II. Characteristics of the CGI-based cluster

Number of CGI nodes	32
Interconnection network	10 Gigabit Ethernet
Total EC2 Compute Units	1072
Total CPU Cores	256 (3 TFLOPS DP)
Total GPUs	64 (32.96 TFLOPS DP)
Total FLOPS	35.96 TFLOPS DP

The instances of this GPU cluster have been allocated simultaneously in a single placement group in order to obtain nearby instances, with full bisection 10 Gbps bandwidth connectivity among them.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

The interconnection technology, 10 Gigabit Ethernet, is a differential characteristic of the Amazon EC2 cluster instances. Additionally, Amazon EC2 guarantees that the hardware infrastructure of the cluster instances, both CGI and Cluster Compute Instance (CCI), is not shared with any other Amazon EC2 instances and at any given time, that each node runs a single virtual machine.

The CentOS 5.5 GPU HVM AMI (ami-aa30c7c3) is provided by AWS for preparing the software configuration of EC2 GPU-based clusters. However, this AMI comes with CUDA toolkit version 3.1, so it is required its upgrading to CUDA version 4.0, mainly since it comes with some extra features such as the ability to share GPUs across multiple threads and the control from a single host thread of all GPUs in the system concurrently. The GNU C/Fortran 4.1.2 compiler was used with -O3 flag to compile all the benchmarks and applications, except NAMD, which is distributed in binary form. The Intel compiler version 12.1 and the Intel MKL library 10.3 have been used for building the HPL benchmark. The message-passing library used for NAMD, MC-GPU y HPL is OpenMPI [33], version 1.4.4.

3.2. Amazon EC2 Virtualization Technologies Overview

The Virtualization Machine Monitor (VMM), also called hypervisor, used by all Amazon EC2 instances is Xen [34], a high performance VMM quite popular among cloud providers. Xen systems have the hypervisor at the lowest and most privileged layer. The hypervisor schedules one or more guest operating systems across the physical CPUs. The first guest operating system, called in Xen terminology domain 0 (dom0), boots automatically when the hypervisor boots and receives special management privileges and direct access to all physical hardware by default. The system administrator can log into dom0 in order to manage any further guest operating systems, known as domain U (domU) in Xen terminology.

Xen supports two virtualization technologies, Full Virtualization (HVM) and Paravirtualization (PV). On the one hand, HVM allows the virtualization of proprietary operating systems, since the guest system's kernel does not require modification, but guests require CPU virtualization extensions from the host CPU (Intel VT [35], AMD-V [36]). In order to boost performance fully virtualized HVM guests can use special paravirtual device drivers to bypass the emulation for disk and network I/O. On the other hand, PV requires changes to the virtualized operating system to be hypervisor-aware. This allows the VM to coordinate with the hypervisor, reducing the use of privileged instructions that are typically responsible for the major performance penalties in full virtualization. This technology does not require virtualization extensions from the host CPU.

Amazon EC2 CCI and CGI use Xen HVM virtualization technology with paravirtual drivers for improving network performance, whereas the rest of Amazon EC2 instance types are Xen PV guests. Therefore, the access to the Network Interface Card (NIC) in Amazon EC2 instances is paravirtualized. However, a direct access to the underlying NIC (and other PCI cards) is also possible in virtualized environments using PCI passthrough [37]. This technique consists of isolating a device in order to be used exclusively by a guest operating system, which eventually achieves near-native performance from the device. Xen supports PCI passthrough [38] for PV and HVM guests, but dom0 OS must support it, typically available as a kernel build-time option. Additionally, the latest processor architectures by Intel and AMD also support PCI passthrough (in addition to new instructions that assist the hypervisor). Intel calls its approach Virtualization Technology for Directed I/O (VT-d [39]), while AMD refers to it as I/O Memory Management Unit (IOMMU [40]).

DOI: 10.1002/cpe

In both cases the CPU is able to map PCI physical addresses to guest virtual addresses and take care of access (and protection) to the mapped device, so that the guest OS can use and take advantage of the device like in a non-virtualized system. In addition to this mapping of virtual guest addresses to physical memory, isolation is provided in such a way that other guests (or even the hypervisor) are precluded from accessing it.

Amazon EC2 CGI relies on Xen PCI passthrough for accessing the GPUs using Intel VT-d technology, so domU OS and applications do direct I/O with the GPUs. Unfortunately, as we mentioned above, the NIC is not available via PCI passthrough so the access to the network is virtualized in Amazon EC2 instances. This lack of efficient virtualized network support in Amazon EC2 explains why previous works using CCIs on Amazon EC2 concluded that the communications performance remains as the main issue for the scalability of MPI applications in the cloud.

3.3. GPGPU Kernels and Applications

The evaluation of GPGPU on Amazon EC2 has been done using representative GPGPU benchmarks and applications, several synthetic kernels, two real-world applications and the widely used High-Performance Linpack (HPL) implementation [41] of the Linpack [42] benchmark.

3.3.1. Synthetic Kernels. They are code snippets which implement either operations that can take full advantage of the hardware (e.g., a bus speed characterization code) or provide with widely extended basic building blocks in HPC applications (e.g., a matrix multiplication kernel). The synthetic kernels used for the evaluation of GPGPU on Amazon EC2 have been selected from two representative benchmark suites, Scalable HeterOgeneus Computing (SHOC) [43] and Rodinia benchmark suite [44]. On the one hand, the SHOC suite assesses low level architectural features through microbenchmarking, as well as determines the computational performance of the system with the aid of application kernels. Table III presents the 10 SHOC synthetic kernels selected. Furthermore, these kernels have OpenCL and CUDA implementations, which allows the comparative analysis of their performance. On the other hand, the Rodinia suite targets the performance analysis of heterogeneous systems, providing application kernels implemented with OpenMP, OpenCL and CUDA for both GPUs and multi-core CPUs. Table III also includes 2 synthetic kernels from the Rodinia suite.

3.3.2. Applications in Science and Engineering. Two distributed real-world applications which support the use of multiple GPUs per node in CUDA, NAMD [45, 46] and MC-GPU [47, 48], have been selected. Both applications can be executed either using only CPUs or mixing CPUs and GPUs. On the one hand, NAMD is a parallel molecular dynamics code, based on Charm++ parallel objects, designed for high-performance simulation of large biomolecular systems. The NAMD suite includes ApoA1, one of the most used data sets for benchmarking NAMD, which models a bloodstream lipoprotein particle with 92K atoms of lipid, protein, and water on 500 steps, with 12A cutoff. NAMD is a communication-intensive iterative application. On the other hand, MC-GPU is an X-ray transport simulation code that can generate clinically-realistic radiographic projection images and computed tomography (CT) scans of the human anatomy. It uses an MPI library to address

Prepared using cpeauth.cls
DOI: 10.1002/cpe

Table III. Selected Synthetic Kernels

Kernel	Suite	Performance Unit	Description
BusSpeedDownload	SHOC	GB/s	PCIe bus bandwidth (host to device)
BusSpeedReadback	SHOC	GB/s	PCIe bus bandwidth (device to host)
MaxFlops	SHOC	GFLOPS	Peak floating-point operations per second
Device Memory	SHOC	GB/s	Device memory bandwidth
SPMV	SHOC	GFLOPS	Multiplication of sparse matrix and vector
GEMM	SHOC	GFLOPS	Matrix multiplication
FFT	SHOC	GFLOPS	Fast Fourier Transform
MD	SHOC	GFLOPS	Molecular dynamics
Stencil2D	SHOC	Time(s)	A two-dimensional nine point stencil calculation
S3D	SHOC	GFLOPS	Computes the rate of various chemical reactions across a regular 3D grid.
CFD	Rodinia	Time(s)	Grid finite volume solver for the three-dimensional Euler equations for compressible flow
Hotspot	Rodinia	Time(s)	Estimate processor temperature based on an architectural floorplan and simulated power measurements

multiple nodes in parallel during the CT simulations. It is a computation-intensive code with little communication.

3.3.3. High-Performance Linpack Benchmark. The High-Performance Linpack (HPL) benchmark [42] solves a random dense system of linear equations. The solution is computed by an LU decomposition with partial pivoting followed by backsubstitution. Dense linear algebra workloads are pervasive in scientific applications, especially in compute-intensive algorithms, so HPL provides a good upper bound on the expected performance of scientific workloads. In addition, TOP500 [49] list is based on HPL. The HPL implementation used in this work is the hybrid MPI/CUDA [50], not publicly available but provided to academia, research centers and registered CUDA developers by NVIDIA.

3.4. Evaluation Methodology

The 15 codes selected for the GPGPU evaluation, 12 synthetic kernels (10 from SHOC and 2 from Rodinia), the 2 applications (NAMD and MC-GPU) and the HPL, have been initially executed both in a single CPU core and a single GPU of an Amazon EC2 CGI VM. Additionally, these codes have been also executed both in a single CPU core and a single GPU of a non-virtualized node with the same CPU and GPU in order to assess the overhead of the virtualization on the CPU and GPU computational power. This physical node from our cluster has been denoted from now on as CAG (Computer Architecture Group) testbed.

Once the performance of a single core and GPU has been measured, the evaluated codes have been executed from 1 up to 32 CGI VMs, using all the available CPU cores (8) and the available GPUs (2) per node. This is the most efficient configuration as the virtualized network imposes a significant overhead con communications performance, which motivates the minimization of the

DOI: 10.1002/cpe

use of the network. Except special notice, all the codes use double precision (DP) floating point arithmetic.

Furthermore, both the CPU and GPU speedups have been computed taking as baseline the performance of a single CPU core. Thus, the performance of the GPUs is significantly higher than the performance of a single CPU core for the evaluated codes. For example, NAMD is able to achieve a speedup of 34 using the 2 GPUs of a VM, whereas the speedup achieved with the 8 CPU cores is 8.

Finally, both the GNU and the Intel compilers have been considered for the CPU code used in this performance evaluation, showing little performance differences between them, never exceeding 4%. As the Intel compiler was generally the best performer, the reported results have been obtained from binaries compiled with this compiler.

4. ASSESSMENT OF GPGPU FOR HIGH PERFORMANCE CLOUD COMPUTING

This section assesses the performance of GPGPU for High Performance Cloud Computing on a public cloud infrastructure, Amazon EC2, using the selected benchmarks/applications presented in the previous section. Two features of the CGI VMs used are key for the analysis of the obtained performance results, the high penalty of the virtualized access to the high-speed 10 Gigabit Ethernet network and the low overhead of the direct access to the GPUs through Xen PCI passthrough using Intel VT-d hardware.

4.1. Synthetic Kernels Performance with CUDA and OpenCL

Figures 1 and 2 present the performance results obtained by the selected synthetic kernels listed in the Table III using their CUDA and OpenCL implementations on a single NVIDIA Tesla "Fermi" M2050, both on Amazon EC2 and CAG testbeds.

Regarding Figure 1, analyzing the results from left to right and top to bottom, the PCIe Bus Bandwidth benchmark shows similar results for CUDA and OpenCL, both from host to device as well as from device to host. Moreover, the results reported on Amazon EC2 are similar to those obtained on CAG.

The Peak Floating Point benchmark presents a similar behavior, with insignificant differences between Amazon and CAG both for single and double precision tests. Here CUDA and OpenCL also achieve a similar peak performance (to be more precise, OpenCL outperforms CUDA slightly), which supports the conclusion that OpenCL has the same potential as CUDA to take full advantage of the underlying hardware. In fact, the observed performance results are very close to the theoretical peak performance on its GPU (NVIDIA Tesla C2050), 1030 GFLOPS for single precision (SP) and 515 GFLOPS for double precision (DP).

Next two graphs in Figure 1 present the Device Memory Bandwidth tests, both for read and write operations. On the one hand, the left graph shows the device global memory bandwidth, measured by accessing global memory in a coalesced manner, where CUDA and OpenCL obtain similar results. However, for these tests the difference between Amazon EC2 and CAG performance is significant, especially for read operations. The main reason for this performance gap is the ECC

Prepared using cpeauth.cls
DOI: 10.1002/cpe

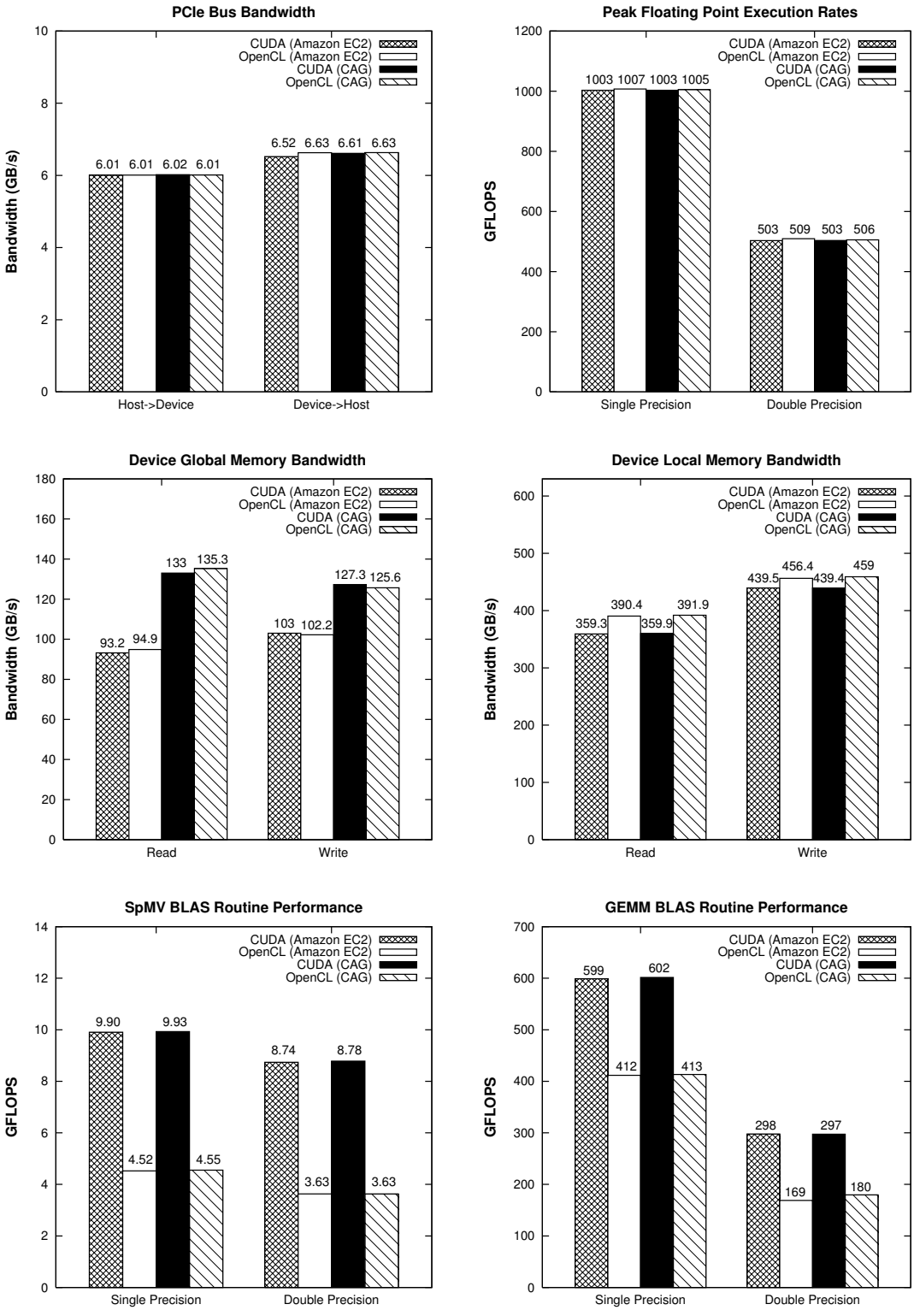

Figure 1. SHOC synthetic benchmarks performance on Amazon EC2 and CAG testbeds

DOI: 10.1002/cpe

memory error protection which is enabled in Amazon EC2, whereas it is disabled in CAG. With ECC memory error protection activated, a portion of the GPU memory is used for ECC bits, so the available user memory is reduced by 12.5% (3 GB total memory yields 2.625 GB of user available memory). The overhead associated with the handling of these ECC bits represents an important performance penalty. On the other hand, the right graph shows the device local memory bandwidth, where OpenCL outperforms CUDA, especially for read operations with up to 8% more bandwidth. In this case, there are no significant performance differences between Amazon EC2 and CAG testbeds.

The last two graphs at the bottom in Figure 1 present the results for two widely used Basic Linear Algebra Subprograms (BLAS) subroutines: the level 2 operation Sparse Matrix-Vector (SpMV) multiplication and the level 3 operation General Matrix Multiplication (GEMM), both for single and double precision. Regarding the testbed, these routines obtain practically the same performance results in Amazon and CAG. However, CUDA significantly outperforms OpenCL, even doubling its performance in some cases (SpMV in single and double precision).

Regarding Figure 2, analyzing the results from left to right and top to bottom, CUDA outperforms significantly OpenCL for the Fast Fourier Transform (FFT) and Molecular Dynamics (MD) kernels. This confirms that the OpenCL SHOC kernels are less optimized for this GPU architecture (Tesla "Fermi") than the CUDA kernels. Moreover, these CUDA codes achieve the highest performance on CAG, mainly due to its superior memory performance thanks to have deactivated the ECC memory correction.

Next graph in Figure 2 presents the Stencil2D kernel, whose OpenCL implementation shows quite poor performance on Amazon EC2, whereas on CAG has a runtime slightly higher than CUDA. Next graph corresponds to S3D kernel, where OpenCL is the best performer in CAG with single precision, whereas in Amazon EC2 OpenCL and CUDA results are similar.

The last two graphs at the bottom in Figure 2 present the results for the two selected kernels from Rodinia suite, the Computational Fluid Dynamics (CFD) and the Hotspot kernels. These kernels have an OpenMP version, in addition to the OpenCL and CUDA versions, which has been executed using the 8 CPU cores available in a Amazon EC2 VM and in the CAG node. The CFD results show that the GPUs are able to speedup around 5.2-8.6 times the CPU performance, thanks to the implementation in CFD of an algorithm that is suitable for its execution on a GPU. The GPU runtime of the Hotspot kernel also outperforms the CPU runtime, achieving around 6.4-9.3 times higher performance. Finally, there are no significant differences in the performance of the CUDA and OpenCL versions.

4.2. Distributed CUDA Applications Scalability

Figure 3 presents the performance of the ApoA1 benchmark executed with NAMD 2.8 and up to 32 Amazon EC2 CGIs. The left graph shows the achieved simulation rate measured in days/ns, whereas the right graph presents their corresponding speedups. This benchmark has been executed using the two implementations of NAMD, the only CPU version and the CPU+GPU (CUDA) implementation. Regarding the results obtained, using a single VM the CPU+GPU version outperforms clearly the only CPU version, achieving around 4 times higher performance. However, the CPU+GPU version can not take advantage of running on two or more VMs, in fact it obtains lower performance, especially using 8 or more VMs, than using a single VM. This behavior contrasts with the only

Prepared using cpeauth.cls
DOI: 10.1002/cpe

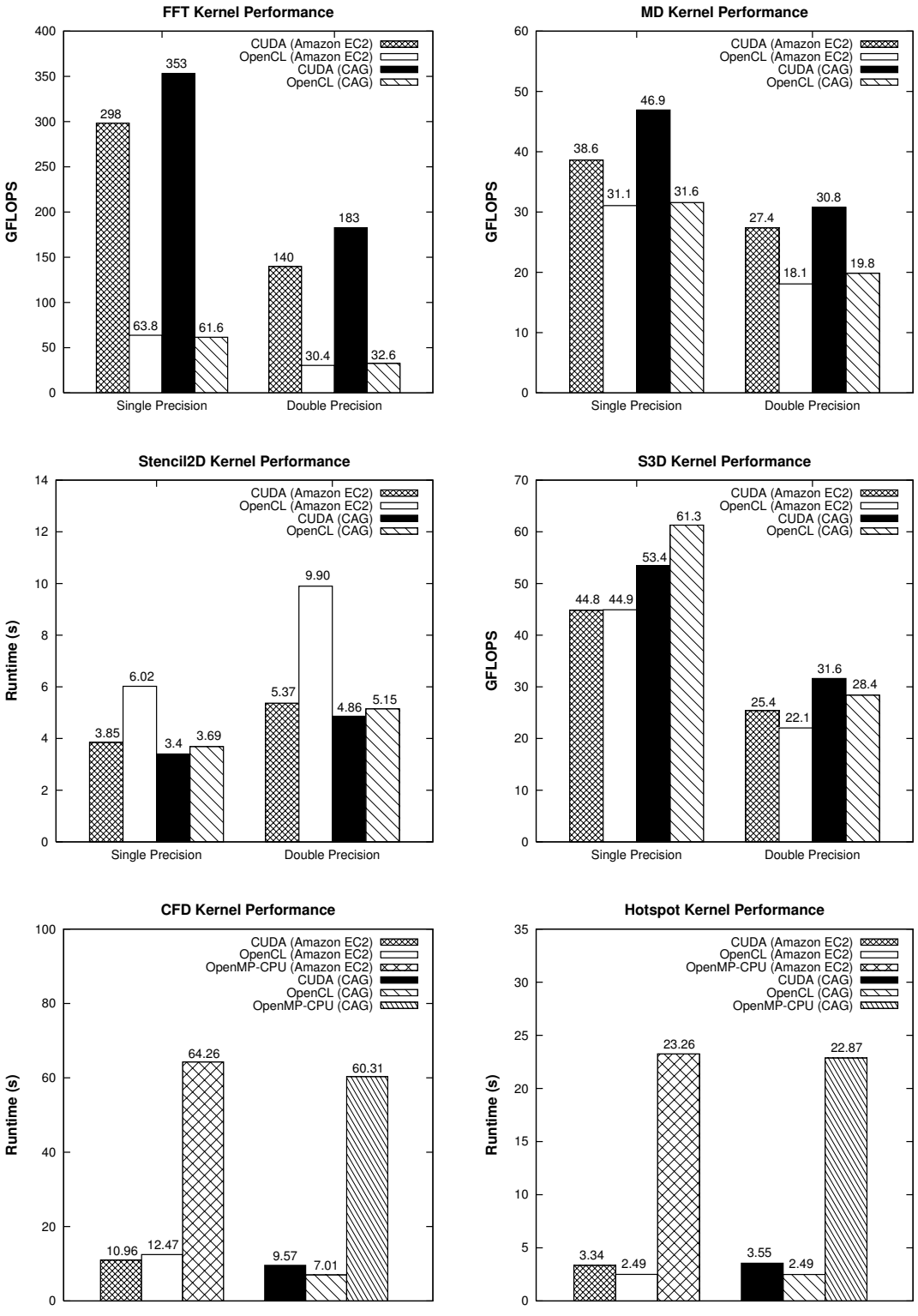

Figure 2. SHOC and Rodinia kernels performance on Amazon EC2 and CAG testbeds

DOI: 10.1002/cpe

CPU version which is able to scale moderately using up to 16 VM, almost reaching the performance results achieved by the CPU+GPU version.

This poor scalability of the NAMD ApoA1 Benchmark on Amazon EC2 is explained by the moderately communication-intensive nature of this benchmark, which suffers a significant performance penalty caused by the overhead of the virtualized access to the network. This performance bottleneck is especially important for the CPU+GPU implementation as it computes faster than the only CPU version and therefore its communication requirements are higher. In fact, NAMD developers recommend the use of low latency interconnects (e.g., InfiniBand) for CUDA-accelerated NAMD executions across multiple nodes [46].

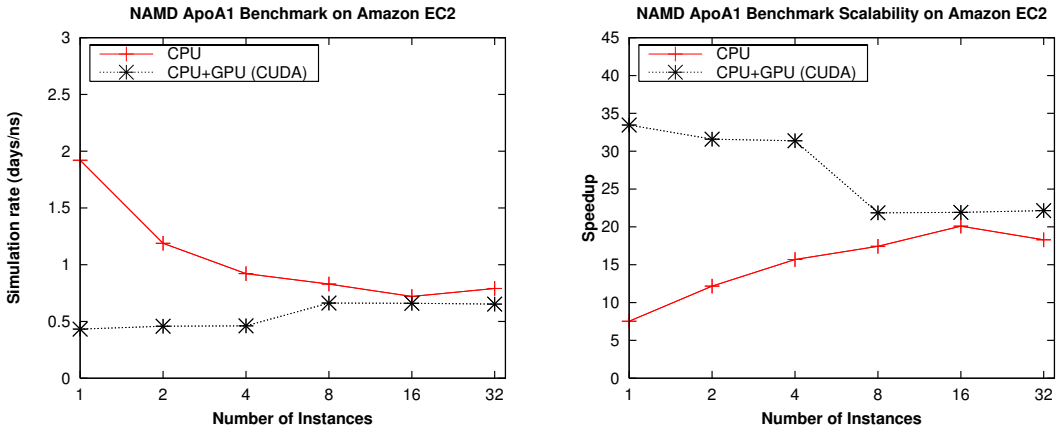

Figure 3. Performance of NAMD ApoA1 benchmark on Amazon EC2 CGIs

Figure 4 presents the performance of MC-GPU using up to 32 Amazon EC2 CGIs. The left graph shows the runtime measured in seconds, whereas the right graph depicts their corresponding speedups. This code is a massively MPI Monte Carlo simulation that has been executed with OpenMPI using only CPU computation as well as CPU plus GPU computation with CUDA. Unlike NAMD, this is a computation-intensive code with little communication, which eventually is able to achieve almost linear speedups using up to 32 instances. Thus, a speedup of almost 220 over 256 cores is achieved by the only CPU version whereas the CPU+GPU version gets a speedup of around 1700 thanks to the use of 64 GPUs, almost 8 times the speedup of the only CPU version.

4.3. HPL Linpack Benchmark Performance

Figure 5 depicts the performance achieved by the HPL benchmark using up to 32 Amazon EC2 CGIs. The left graph shows the GFLOPS obtained by the resolution of the dense system of linear equations, whereas the right graph presents their corresponding speedups. This MPI application has been executed with the OpenMPI implementation using only CPU as well as using CPU plus GPU computation with CUDA. The analysis of the breakdown of the runtime has revealed that most of the HPL runtime is spent in matrix-matrix multiplications in the update of trailing matrices. The bigger the problem size N is, the more time is spent in this routine, so optimization of DGEMM is critical to achieve a high score. Thus, the performance of HPL depends on two main factors, basically on the GEMM subroutine performance in double precision (DGEMM), but

DOI: 10.1002/cpe

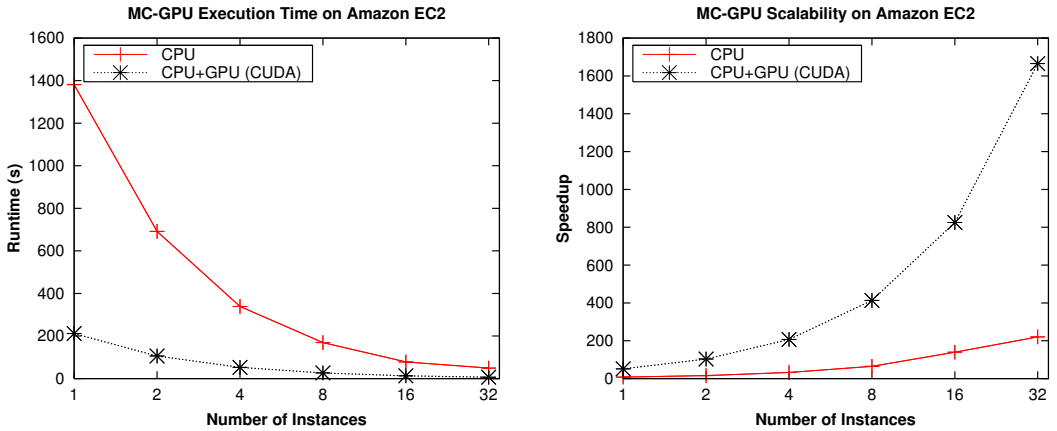

Figure 4. Performance of MC-GPU on Amazon EC2 CGIs

also on network performance, especially as the number of nodes increases. We ran many different configurations to find the best HPL settings (problem size, number of rows and columns, partitioning block size, panel factorization algorithm, and broadcast algorithm among others) and report the peak for each number of instances used.

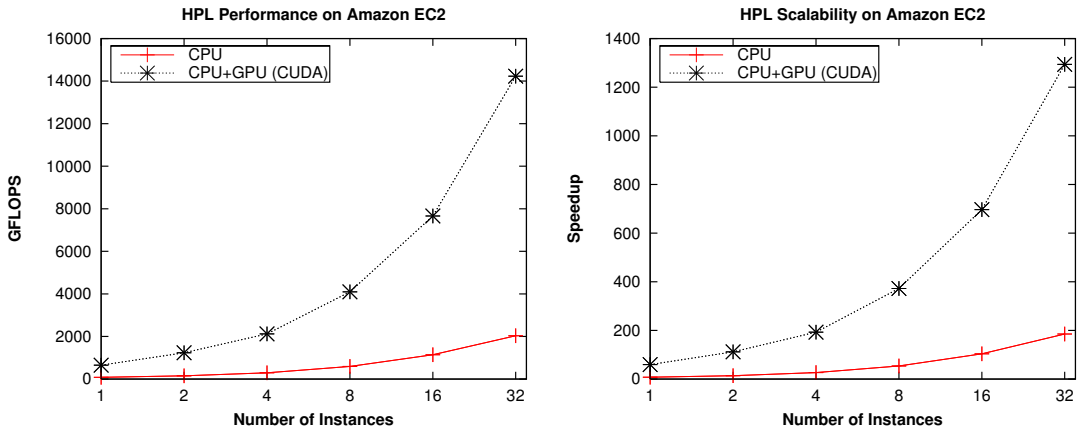

Figure 5. HPL (Linpack) performance on Amazon EC2 CGIs

The measured HPL performance results show a moderate scalability limited severely by the poor virtualized network support in CGI VMs. Thus, the ratio between the achieved HPL performance and the theoretical peak performance decreases significantly as the number of instances increases, limiting the scalability of Amazon EC2 CGIs. Regarding the CPU+GPU (CUDA) implementation of HPL, using one CGI VM a 59% of efficiency is obtained (655 GFLOPS out of 1124 peak GFLOPS are achieved), but when using 32 CGI VMs the efficiency drops below 40% (14.23 TFLOPS out of 35.96 peak TFLOPS are achieved). Although 14.23 TFLOPS represents a speedup of 1290 with respect to the baseline HPL execution on a single CPU core, an efficiency of the CPU+GPU version below 40% with only 32 instances is quite poor.

The only CPU version presents a similar scalability behavior although the efficiencies are higher. Thus, running HPL on one instance the efficiency obtained is relatively high 88% (82.39 GFLOPS

Prepared using cpeauth.cls
DOI: 10.1002/cpe

out of 93.76 peak GFLOPS are achieved), whereas the efficiency on 32 instances drops belows 68% (2.035 TFLOPS out of 3 peak TFLOPS are achieved). The speedup obtained, 185 on 256 cores, represents a moderate scalability.

Figure 6 and Table IV show the achieved efficiency in terms of percentage of the theoretical peak performance in GFLOPS for each number of instances considered. To the best of our knowledge, this is the first time that HPL performance results on Amazon EC2 GPIs are reported, obtaining more than 14 TFLOPS. Amazon EC2 cluster instances have been reported to obtain 240 TFLOPS Rmax (354 TFLOPS Rpeak), ranked #42 in the last TOP500 list (November 2011), with 17024 cores (1064 nodes, each with 16 cores), showing an efficiency of 67.80%, a performance that could only be obtained running HPL on a non virtualized infrastructure. The previous appearance of Amazon EC2 on TOP500 list was on November 2010, with a cluster ranked #233, which obtained 41.8 TFLOPS Rmax (82.5 TFLOPS Rpeak) with 7040 cores (880 nodes, each comprising 8 cores), reporting an efficiency of 51%, in tune with our measured results. As the last system (ranked #500) in the current TOP500 list has 51 TFLOPS Rmax, it would be expected that a system with around 150 CGIs could have entered in the last TOP500 list. The cost of the access to such an infrastructure would be barely 300 USD$ (2.10 USD$) per hour.

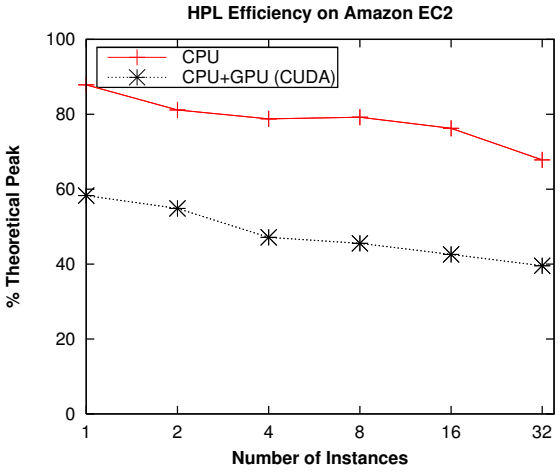

Figure 6. HPL (Linpack) efficiency as percentage of peak GFLOPS on Amazon EC2 CGIs

Table IV. HPL (Linpack) efficiency on Amazon EC2 CGIs

# instances	# cores	# GPUs	CPU GFLOPS (Rpeak)	CPU GFLOPS (Rmax)	CPU+GPU TFLOPS (Rpeak)	CPU+GPU TFLOPS (Rmax)
1	8	2	93.76	82.39 (87.87%)	1.124	0.655 (59.27%)
2	16	4	187.52	152.2 (81.16%)	2.248	1.233 (54.85%)
4	32	8	375.04	295.5 (78.71%)	4.496	2.120 (47.15%)
8	64	16	750.08	594.2 (79.21%)	8.992	4.096 (45.55%)
16	128	32	1500.16	1144 (76.26%)	17.984	7.661 (42.60%)
32	256	64	3000.32	2035 (67.83%)	35.968	14.23 (39.66%)

DOI: 10.1002/cpe

The performance evaluation presented in this section has shown that computationally intensive applications with algorithms which can be efficiently exploited in GPGPU can take full advantage of their execution in CGI VMs from Amazon EC2 cloud. In fact, the applications can benefit from the use of GPUs without any significant performance penalty, except when accessing memory intensively, where a small penalty can be observed as Amazon EC2 CGIs have ECC memory error protection enabled, which slightly limits the memory access bandwidth. Communication intensive applications suffer from the overhead of the virtualized network access, which can reduce scalability significantly.

5. CONCLUSIONS

GPGPU is attracting a considerable interest, especially in the scientific community, due to its massive parallel processing power. Another technology that is receiving an increasing attention is cloud computing, especially in enterprise environments, for which cloud services represent a flexible, reliable, powerful, convenient, and cost-effective alternative to owning and managing their own computing infrastructure. Regarding HPC area, cloud providers, such as Amazon public cloud, are already providing HPC resources, such as high-speed networks and GPUs.

This paper has evaluated GPGPU for High Performance Cloud Computing on a public cloud computing infrastructure, using 32 Amazon EC2 Cluster GPU Instances, equipped with 10 Gigabit Ethernet and two NVIDIA Tesla GPUs each instance. The analysis of the performance results confirms that GPGPU is a feasible option in a public cloud due to the efficient access to the GPU accelerator in virtualized environments. However, our research has also detected that the virtualized network overhead limits severely the scalability of the applications, especially those sensitive to communication start-up latency. Therefore, more efficient communication middleware support is required to get over current cloud network limitations, with appropriate optimizations on communication libraries and OS virtualization layers. Thus, a direct access to the NIC through a device assignment to the virtual machine is the key to reduce the network overhead in cloud environments and make HPC on demand a widespread option for GPGPU.

ACKNOWLEDGEMENT

This work was funded by the Ministry of Science and Innovation of Spain under Project TIN2010-16735 and by an Amazon Web Services (AWS) research grant.

REFERENCES

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 2009; **25**(6):599–616.
2. Evangelinos C, Hill CN. Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2. *Proceedings of 1st Workshop on Cloud Computing and Its Applications (CCA'08)*, Chicago, IL, USA, 2008; 1–6.

Prepared using cpeauth.cls
DOI: 10.1002/cpe

3. Walker E. Benchmarking Amazon EC2 for High-Performance Scientific Computing. *LOGIN: The USENIX Magazine* 2008; **33**(5):18–23.
4. Vecchiola C, Pandey S, Buyya R. High-Performance Cloud Computing: A View of Scientific Applications. *Proceedings of 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN'09)*, Kaoshiung, Taiwan, ROC, 2009; 4–16.
5. Napper J, Bientinesi P. Can cloud computing reach the top500? *Proceedings of Combined Workshops on UnConventional High Performance Computing Workshop Plus Memory Access Workshop (UCHPC-MAW'09)*, Ischia, Italy, 2009; 17–20.
6. Jackson KR, Ramakrishnan L, Muriki K, Canon S, Cholia S, Shalf J, Wasserman HJ, Wright NJ. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. *Proceedings of 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10)*, Indianapolis, USA, 2010; 159–168.
7. MPI: A Message Passing Interface Standard. http://www.mcs.anl.gov/research/projects/mpi/ [Last visited: December 2011].
8. Leist A, Playne DP, Hawick KA. Exploiting graphical processing units for data-parallel scientific applications. *Concurrency and Computation: Practice and Experience* 2009; **21**(18):2400–2437.
9. Pagès G, Wilbertz B. GPGPUs in computational finance: massive parallel computing for American style options. *Concurrency and Computation: Practice and Experience* 2011 (In Press); doi:10.1002/cpe.1774.
10. Fan Z, Qiu F, Kaufman A, Yoakum-Stover S. GPU Cluster for High Performance Computing. *Proceedings of 16th ACM/IEEE Conference on Supercomputing (SC'04)*, Pittsburgh, PA, USA, 2004; 47.
11. Nickolls J, Buck I, Garland M, Skadron K. Scalable Parallel Programming with CUDA. *Queue* 2008; **6**:40–53.
12. Stone JE, Gohara D, Guochun S. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *Computing in Science and Engineering* 2010; **12**(3):66–73.
13. Malik M, Li T, Sharif U, Shahid R, El-Ghazawi T, Newby G. Productivity of GPUs under different programming paradigms. *Concurrency and Computation: Practice and Experience* 2011; **24**(2):179–191.
14. Karunadasa NP, Ranasinghe DN. Accelerating high performance applications with CUDA and MPI. *Proceedings of 4th International Conference on Industrial and Information Systems (ICIIS'09)*, Sri Lanka, India, 2009; 331–336.
15. Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2 [Last visited: December 2011].
16. Amazon Inc. High Performance Computing Using Amazon EC2. http://aws.amazon.com/ec2/hpc-applications/ [Last visited: December 2011].
17. Luszczek P, Meek E, Moore S, Terpstra D, Weaver VM, Dongarra JJ. Evaluation of the HPC Challenge Benchmarks in Virtualized Environments. *Proceedings of 6th Workshop on Virtualization in High-Performance Cloud Computing (VHPC'11)*, Bordeux, France, 2011; 1–10.
18. Younge AJ, Henschel R, Brown JT, von Laszewski G, Qiu J, Fox GC. Analysis of Virtualization Technologies for High Performance Computing Environments. *Proceedings of 4th IEEE International Conference on Cloud Computing (CLOUD'11)*, Washington, DC, USA, 2011; 9–16.
19. Gavrilovska A, Kumar S, Raj H, Schwan K, Gupta V, Nathuji R, Niranjan R, Ranadive A, Saraiya P. High-Performance Hypervisor Architectures: Virtualization in HPC Systems. *Proceedings of 1st Workshop on System-level Virtualization for High Performance Computing (HPCVirt'07)*, Lisbon, Portugal, 2007; 1–8.
20. Deelman E, Singh G, Livny M, Berriman B, Good J. The cost of doing science on the cloud: the Montage example. *Proceedings of 20th ACM/IEEE Conference on Supercomputing (SC'08)*, Austin, TX, USA, 2008; 1–12.
21. Ekanayake J, Fox GC. High Performance Parallel Computing with Clouds and Cloud Technologies. *Proceedings of 1st International Conference on Cloud Computing (CLOUDCOMP'09)*, Munich, Germany, 2009; 20–38.
22. Ostermann S, Iosup A, Yigitbasi N, Prodan R, Fahringer T, Epema D. A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing. *Proceedings of 1st International Conference on Cloud Computing (CLOUDCOMP'09)*, Munich, Germany, 2009; 115–131.
23. Wang G, Eugene Ng TS. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. *Proceedings of 29th IEEE Conference on Computer Communications (INFOCOM'10)*, San Diego, CA, USA, 2010; 1163–1171.
24. Rehr JJ, Vila FD, Gardner JP, Svec L, Prange M. Scientific Computing in the Cloud. *Computing in Science and Engineering* 2010; **12**(3):34–43.
25. He Q, Zhou S, Kobler B, Duffy D, McGlynn T. Case study for running HPC applications in public clouds. *Proceedings of 19th ACM International Symposium on High Performance Distributed Computing (HPDC'10)*, Chicago, IL, USA, 2010; 395–401.
26. Iosup A, Ostermann S, Yigitbasi N, Prodan R, Fahringer T, Epema D. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems* 2011;

22:931–945.
27. Regola N, Ducom JC. Recommendations for Virtualization Technologies in High Performance Computing. *Proceedings of 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10)*, Indianapolis, USA, 2010; 409–416.
28. Carlyle AG, Harrell SL, Smith PM. Cost-Effective HPC: The Community or the Cloud? *Proceedings of 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10)*, Indianapolis, USA, 2010; 169–176.
29. Sun C, Nishimura H, James S, Song K, Muriki K, Qin Y. HPC Cloud Applied to Lattice Optimization. *Proceedings of 2nd International Particle Accelerator Conference (IPAC'11)*, San Sebastian, Spain, 2011; 1767–1769.
30. Ramakrishnan L, Canon RS, Muriki K, Sakrejda I, Wright NJ. Evaluating Interconnect and Virtualization Performance for High Performance Computing. *Proceedings of 2nd International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS'11)*, Seattle, WA, USA, 2011; 1–2.
31. Zhai Y, Liu M, Zhai J, Ma X, Chen W. Cloud versus in-house cluster: evaluating Amazon cluster compute instances for running MPI applications. *Proceedings of 23th ACM/IEEE Conference on Supercomputing (SC'11, State of the Practice Reports)*, Seattle, WA, USA, 2011; 1–10.
32. Palankar MR, Iamnitchi A, Ripeanu M, Garfinkel S. Amazon S3 for science grids: a viable solution? *Proceedings of 1st International Workshop on Data-aware Distributed Computing (DADC'08)*, Boston, MA, USA; 55–64.
33. Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Suyres JM, Sahay V, Kambadur P, Barrett B, Lumsdaine A, *et al.*. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. *Proceedings of 11th European PVM/MPI Users' Group Meeting (EuroPVM/MPI'04)*, Budapest, Hungary, 2004; 97–104.
34. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization. *Proceedings of 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, Bolton Landing, NY, USA, 2003; 164–177.
35. Intel Virtualization Technology (Intel VT). http://www.intel.com/technology/virtualization/technology.htm?iid=tech_vt+tech [Last visited: December 2011].
36. AMD Virtualization Technology (AMD-V). http://sites.amd.com/us/business/it-solutions/virtualization/Pages/amd-v.aspx [Last visited: December 2011].
37. Jones, T. Linux virtualization and PCI passthrough. http://www.ibm.com/developerworks/linux/library/l-pci-passthrough/ [Last visited: December 2011].
38. Xen PCI Passthrough. http://wiki.xensource.com/xenwiki/XenPCIpassthrough [Last visited: December 2011].
39. Abramson D, Jackson J, Muthrasanallur S, Neiger G, Regnier G, Sankaran R, Schoinas I, Uhlig R, Vembu B, Wiegert J. Intel Virtualization Technology for Directed I/O. *Intel Technology Journal* 2006; **10**(3):179–192.
40. AMD I/O Virtualization Technology (IOMMU) Specification. http://support.amd.com/us/Processor_TechDocs/34434-IOMMU-Rev_1.26_2-11-09.pdf [Last visited: December 2011].
41. Petitet, A and Whaley, RC and Dongarra, JJ and Cleary, A. HPL: A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. http://www.netlib.org/benchmark/hpl/ [Last visited: December 2011].
42. Dongarra JJ, Luszczek P, Petitet A. The LINPACK Benchmark: past, present and future. *Concurrency and Computation: Practice and Experience* 2003; **15**(9):803–820.
43. Danalis A, Marin G, McCurdy C, Meredith JS, Roth PC, Spafford K, Tipparaju V, Vetter JS. The Scalable Heterogeneous Computing (SHOC) benchmark suite. *Proceedings of 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU-3)*, Pittsburgh, PA, USA, 2010; 63–74.
44. Che S, Boyer M, Meng J, Tarjan D, Sheaffer JW, Lee SH, Skadron K. Rodinia: A benchmark suite for heterogeneous computing. *Proceedings of IEEE International Symposium on Workload Characterization (IISWC'09)*, Austin, TX, USA, 2009; 44–54.
45. Phillips JC, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel RD, Kalé L, Schulten K. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry* 2005; **26**(16):1781–1802.
46. NAMD: Scalable Molecular Dynamics. http://www.ks.uiuc.edu/Research/namd/ [Last visited: December 2011].
47. Badal A, Badano A. Accelerating monte carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit. *Medical Physics* 2009; **36**(11):4878–4880.
48. MC-GPU: Monte Carlo simulation of x-ray transport in a GPU with CUDA. http://code.google.com/p/mcgpu/ [Last visited: December 2011].
49. TOP500 Org. Top 500 Supercomputer Sites. http://www.top500.org/ [Last visited: December 2011].
50. Fatica M. Accelerating LINPACK with CUDA on Heterogenous Clusters. *Proceedings of 2nd Workshop on General Purpose Computation on Graphics Processing Units (GPGPU-2)*, Washington, DC, USA, 2009; 46–51.

DOI: 10.1002/cpe