Running Scientific Codes on Amazon EC2: a Performance Analysis of Five High-end Instances

Roberto R. Expósito*, Guillermo L. Taboada, Xoán C. Pardo,
Juan Touriño and Ramón Doallo
Computer Architecture Group, Faculty of Informatics, University of A Coruña
Campus de Elviña s/n, 15071 A Coruña, Spain

ABSTRACT

Amazon Web Services (AWS) is a well-known public Infrastructure-as-a-Service (IaaS) provider whose Elastic Computing Cloud (EC2) offering includes some instances, known as cluster instances, aimed at High-Performance Computing (HPC) applications. In previous work, authors have shown that the scalability of HPC communication-intensive applications does not benefit from using higher computational power cluster instances as much as it could be expected. Cost analysis recommends using lower computational power cluster instances unless high memory requirements preclude their use. Moreover, it has been observed that scalability is very poor when more than one instance is used due to network virtualization overhead. Based on those results, this paper gives more insight into the performance of running scientific applications on the Amazon EC2 platform evaluating five (of which two have been recently released) of the higher computational power instances in terms of single instance performance, intra-VM (Virtual Machine) scalability and cost-efficiency. The evaluation has been carried out using both an HPC benchmark suite and a real High-Troughput Computing (HTC) application.

Keywords: Cloud Computing, High Performance Computing, High Throughput Computing, Amazon EC2, OpenMP

1. Introduction

Cloud computing [1] is an Internet-based computing model which has gained significant popularity in the past several years as it provides on-demand network access to a shared pool of configurable and often virtualized computing resources typically billed on a pay-as-you-use basis. Infrastructure-as-a-Service (IaaS) is a service model which easily enables users to provision on-demand virtualized computing resources (i.e. storage, compute capacity). Amazon Web Services (AWS) is an IaaS provider whose Elastic Compute Cloud (EC2) is nowadays among the most used and largest public cloud platforms. With the reduction of the overhead imposed by virtualized environments in the last years, cloud computing is becoming an attractive option for High Performance Computing (HPC). Some early studies [2; 3] have evaluated public clouds for HPC since 2008 and the main conclusion was that clouds at that time were not designed for running tightly coupled HPC applications. Since then Amazon has released instances specifically tailored to HPC applications and other demanding network-bound applications [4], known as cluster instances. These instances provide powerful CPU resources, dedicated physical node allocation and low latency high-bandwith connectivity (10 Gbps Ethernet), allowing users to easily set up a cost-effective virtual cluster for HPC applications.

In recent work [5] authors analyzed the main performance bottlenecks in HPC application scalability on Amazon EC2. A representative body of the NAS Parallel Benchmarks (NPB) [6] was executed using an important number of cores, up to 512, on Amazon EC2 cluster instances. It has been observed that scalability was severely limited by the lack of proper network virtualization support, specially when executing communication-intensive NPB kernels using more than one cluster instance. Motivated by those results, this paper further explores the performance of running scientific applications on the Amazon EC2 platform. Five high-end, both standard and cluster instances, have been selected for this evaluation (see Table 1). As multi-instance executions showed poor scalability, its focus is on evaluating the performance, intra-VM (Virtual Machine) scalability and cost-efficiency of single-instance executions. Moreover, both an HPC benchmark suite (the OpenMP implementation of the NPB kernels) and a representative High-Throughput Computing (HTC) application

* *E-mail addresses:* rreye@udc.es (R.R. Expósito), taboada@udc.es (G.L. Taboada), pardo@udc.es (X.C. Pardo), juan@udc.es (J. Touriño), doallo@udc.es (R. Doallo).

(jModelTest [7]) have been used in the evaluation.

The structure of this paper is as follows: Section 2 presents the related work. Section 3 explains the virtualization support in the Amazon EC2 platform and introduces the main characteristics of the five instances evaluated in this paper. Section 4 describes the experimental configuration, both hardware and software, used in the evaluation and presents the results that have been obtained. Section 5 summarizes our concluding remarks.

2. Related Work

The growing interest in cloud computing has motivated multiple works assessing the feasibility of executing HPC applications on public clouds, among which Amazon EC2 is the most common used option. Many studies [2; 3; 8; 9] have shown that computationally-intensive codes present little overhead when running in virtualized environments, but communication-intensive (i.e. MPI) applications have poor performance in public clouds mainly due to their poor network performance, processor sharing and the use of commodity interconnection technologies (e.g. Gigabit Ethernet) that limit severely the scalability of HPC applications.

After the release of HPC-targeted Amazon EC2 cluster instances, many recent works have evaluated them. Thus, Zhai et al. [10] conducted a comprehensive evaluation of MPI applications on Amazon EC2 cluster instances, revealing a significant performance increase compared with previous evaluations on standard and High-CPU instances. However, the interconnection network overhead, especially the high start-up latency (poor small message performance), remains as the main MPI scalability limitation. Sun et al. [11] relied on Amazon EC2 cluster instances 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-bound or memory-bound applications. Rehr et al. [12] confirmed that Amazon EC2, using standard and High-CPU instance types, is a feasible platform for applications that do not demand high performance network. Finally, our previous work [5] has shown poor scalability when executing communication-intensive applications on multiple EC2 cluster instances due to the network virtualization overhead, but if hybrid shared/distributed memory programming paradigms (i.e. MPI+OpenMP) are used in order to minimize network communications, cluster instances are able to achieve reasonable scalable performance.

3. Overview of the Amazon EC2 Evaluated Instances

Xen [13] is the Virtualization Machine Monitor (VMM) or hypervisor used in the Amazon EC2 platform. Xen architecture has the hypervisor as the lowest and most privileged layer and above it comes one or more guest operating systems (OSs), which the hypervisor schedules across the physical CPUs. The first guest OS, called domain 0 (dom0), boots automatically when the hypervisor boots and receives special management privileges and exclusive direct access to all physical hardware. This dom0 OS is used to manage any further guest OS, called domain U (domU). The virtualization technologies supported for creating these domU guests are full virtualization assisted with hardware support (HVM) and ParaVirtualization (PV). On the one hand, HVM allows the virtualization of proprietary OSs because the guest systems kernel does not require modification, but CPU virtualization extensions in the host CPU are needed. These extensions allow the coordination of the VM and the hypervisor, reducing the use of privileged instructions that are responsible for the major performance penalties in full virtualization. On the other hand, PV requires changes to the virtualized OS to be hypervisor aware but no virtualization extensions in the host CPU are required. Besides that, and in order to boost performance, fully virtualized HVM guests can use special PV device drivers to bypass the emulation for disk and network I/O.

Amazon uses Xen HVM virtualization technology for its cluster instances, that are specifically designed for HPC and other demanding latency-bound applications. They provide powerful CPU resources, dedicated physical node allocation (one VM per physical node) and are interconnected via a high-speed network (full-bisection 10 Gigabit Ethernet using placement groups). Furthermore they have installed paravirtual drivers for improving network and disk performance, instead of using I/O device emulation which is the default in HVM guests. For the rest of EC2 instance types Xen PV or HVM is used (in fact, for some instance types it is possible to select which one to use).

For the purposes of this article five high-end, two standard and three cluster instances, have been evaluated (their main characteristics are summarized in Table 1):

- The High-CPU Extra Large instance (c1.xlarge, abbreviated C1) is well suited for compute-intensive applications if not too memory demanding. It has one Intel Xeon E5506 quad-core Nehalem processor with a total computing power of 20 EC2 Compute

Units (ECUs[1]) and 7 Gbytes of memory. It has been considered the inclusion of this instance in the evaluation because of its low price but still having memory enough as to execute NPB kernels of size Class C without swapping to disk.

- The second generation Double Extra Large instance (m3.2xlarge, abbreviated M3) has been recently released (November 2012) and is the highest processing power standard instance. It has one Intel Xeon E5-2670 octa-core Sandy Bridge processor with a total computing power of 26 ECUs and 30 Gbytes of memory. According to Amazon *"second generation standard instances are ideal for applications that require higher absolute CPU and memory performance"*.
- The Quadruple Extra Large cluster instance (cc1.4xlarge, abbreviated CC1) is the smallest of the HPC-aimed instances. CC1 instances have two Intel Xeon X5570 quad-core Nehalem processors, hence 8 cores per instance with a total computing power of 33.5 ECUs and 23 Gbytes of memory.
- The Eight Extra Large cluster instance (cc2.8xlarge, abbreviated CC2) has two Intel Xeon E5-2670 octa-core Sandy Bridge processors, hence 16 cores per instance with a total computing power of 88 ECUs and 60.5 Gbytes of memory.
- The High Memory Eight Extra Large instance (cr1.8xlarge, abbreviated CR1), a recently released (January 2013) instance that mimics CC2 processor architecture and provides 244 Gbytes of memory and 240 Gbytes of SSD instance storage. Among the characteristics of this instance Amazon indicates that it provides Intel Turbo and NUMA optimizations, but does not give any details.

4. Evaluation of Amazon EC2 Instances

This section describes the configuration and benchmarks used to evaluate the Amazon EC2 instances presented in the previous section and analyzes the evaluation results in terms of performance, intra-VM scalability and cost-efficiency.

Experimental Configuration

The performance evaluation has been conducted on single C1, M3, CC1, CC2 and CR1 instances of the Amazon EC2 platform. These resources have all been allocated in the US East (North Virginia) region and the executions have been carried out mostly during the night, which can be relevant in the results of C1 and M3 instances as they are not dedicated instances. The Amazon Linux 2012.09.1 AMI with kernel 3.2.39 was used in all cases. Finally, the performance results are the mean of several measurements, generally five.

Regarding the software, two different experiments were carried out to evaluate both HPC and HTC codes. For the HPC performance and intra-VM scalability the NAS Parallel Benchmarks (NPB) [6] version 3.3 have been assessed using the official NPB-SER implementation for the serial execution and the NPB-OMP implementation for the OpenMP parallel execution. NPB Class C size problems have been selected because they are the largest workloads that can be executed in all the instances under evaluation without swapping to disk. All the kernels (C/C++ and Fortran) have been compiled with the GNU 4.7.2 compiler and *-O3 -march=native* flags. The metrics considered for the NPB kernels are: (1) MOPS (Millions of Operations Per Second), which measures the operations performed in the benchmark (that differ from the CPU operations issued); and (2) MOPS per US$, which measures the cost-efficiency, that is the number of operations divided by the cost per hour of each instance.

For the HTC evaluation JModelTest [7] version 2.1.3 has been selected. This is a real Java application commonly used in bioinformatics to carry out statistical selection of best-fit models of nucleotide substitution using up to five different model selection strategies. In a shared memory architecture it works by distributing tasks among a pool of Java threads that are initiated and managed by the application. The Java Virtual Machine (JVM) used was the OpenJDK Runtime Environment version 1.7.0_09.

Results of the Evaluation

In Figure 1 the performance and cost-efficiency of the serial execution of all the NPB kernels for each of the evaluated instances is shown. For the EP (Embarrassingly Parallel) and IS (Integer Sort) kernels there are not significative differences on the MOPS obtained. For all the other kernels the C1 instance has obtained the lowest result, not surprisingly as it is the instance with the less powerful CPU, and the M3 instance has the best results, beating the cluster instances in all cases. The reason for this remains unclear for us, as this instance seems to have the same CPU as CC2 and CR1 cluster instances have and it is executed in non-dedicated hardware. Regarding the cluster instances, the newest CR1 instance outperforms slightly CC1 and CC2 instances, which could be explained by the Amazon support to the Intel

[1] According to Amazon one ECU provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

	High-CPU Extra Large	**Double Extra Large**	**Quadruple Extra Large**	**Eight Extra Large**	**High-Memory Eight Extra Large**
Instance type	High-CPU	Standard *(second generation)*	Cluster	Cluster	High-Memory Cluster
CPU	Intel Xeon[a] E5506 @2.13GHz Nehalem	Intel Xeon[a] E5-2670 @2.60GHz Sandy Bridge	2 × Intel Xeon X5570 @2.93GHz Nehalem	2 × Intel Xeon E5-2670 @2.60GHz Sandy Bridge	2 × Intel Xeon E5-2670 @2.60GHz Sandy Bridge Intel Turbo, NUMA
ECUs	20	26	33.5	88	88
#Cores	8[b]	8	8 (16 with HT[c])	16 (32 with HT[c])	16 (32 with HT[c])
Memory	7 GB	30 GB	23 GB	60.5 GB	244 GB
Storage	1690 GB	EBS only	1690 GB	3370 GB	240 GB (SSD)
API name	c1.xlarge	m3.2xlarge	cc1.4xlarge	cc2.8xlarge	cr1.8xlarge
Price (Linux)	$0.58 per hour	$1.00 per hour	$1.30 per hour	$2.40 per hour	$3.50 per hour
Interconnect	1 Gigabit Ethernet	1 Gigabit Ethernet	10 Gigabit Ethernet *(full-bisection bandwidth with placement groups)*		
Virtualization	Xen PV 64-bit	Xen PV/HVM 64-bit (*HVM was used*)	Xen HVM 64-bit *(PV drivers for I/O)*		

[a] As reported by */proc/cpuinfo* since Amazon does not provide that information
[b] For C1 instances the number of physical cores reported by Amazon (8) does not match the information in */proc/cpuinfo* (4)
[c] HT: Hyper-Threading

Table 1: Description of the high-end EC2 instances evaluated in this paper

Turbo technology in this type of instance, as this technology gets its best performance when only one core is used. Having costs per hour into account, results for all kernels show that it is not worth using cluster instances. Having the best performance and lower prices, the M3 instance is the best option in almost all the cases, being C1 also an instance with a great cost-performance ratio.

Results from the execution of some of the NPB kernels are shown in Figure 2. Four representative kernels from the OMP implementation of the NPB have been selected and executed on the evaluated instances: CG (Conjugate Gradient), EP (Embarrassingly Parallel), FT (Fast Fourier Transform) and LU (Lower-Upper Gauss-Seidel solver). Executions have been done varying the number of threads until the maximum number of virtual cores of the instance, as reported by Amazon, was reached. The tendency observed for all the kernels is that, as the number of threads increases, cluster instances get the best performance and high speedup results, being the differences among them significative only for some kernels when the number of threads approaches the maximum. The newest CR1 instance outperforms CC1 and CC2 instances for 8 and 16 threads for all the kernels except LU, where CC2 instance is slightly better. Although Amazon does not give any details, this better behavior of CR1 if compared to CC2, that has the same CPU, could be explained by the NUMA optimizations support included in this type of instance. It is also worth noting that the CC1 instance has the worst results among the cluster instances although they are close to those of CC2 for some of the NPB kernels (e.g. CG and EP). The second generation standard instance M3 gets worst results than cluster instances with poor parallel efficiency (around 50% for 8 threads) but it exhibits a very regular performance behavior for all kernels. Special mention must be made for the results of the C1 instance, although it obtains bad results in some kernels (e.g. FT or LU), for the EP kernel it obtains figures similar to those of M3 and an speedup near to ideal, making it an economical option for Monte Carlo-type simulations when 8 threads or less are used.

With regard to cost-efficiency, results show that M3 and CC1 instances have in general the best cost-performance ratio when using 8 threads or less. For some kernels (i.e. CG end EP) the C1 instance shows even best figures when using 4 threads or less, but the variability of its results does not make it the best choice. The worst results in all cases have been those of CC2 and CR1 cluster instances, being CC2 best than CR1. These are the only instances evaluated using 16 threads and it can be concluded that it is not worth using the newest CR1 instance unless high-memory capacity requirements are a must.

Finally, for the HTC evaluation two of the models that come with the JModelTest distribution has been selected: *aP6.fas* and *wormsCOI.mafft.fas*.

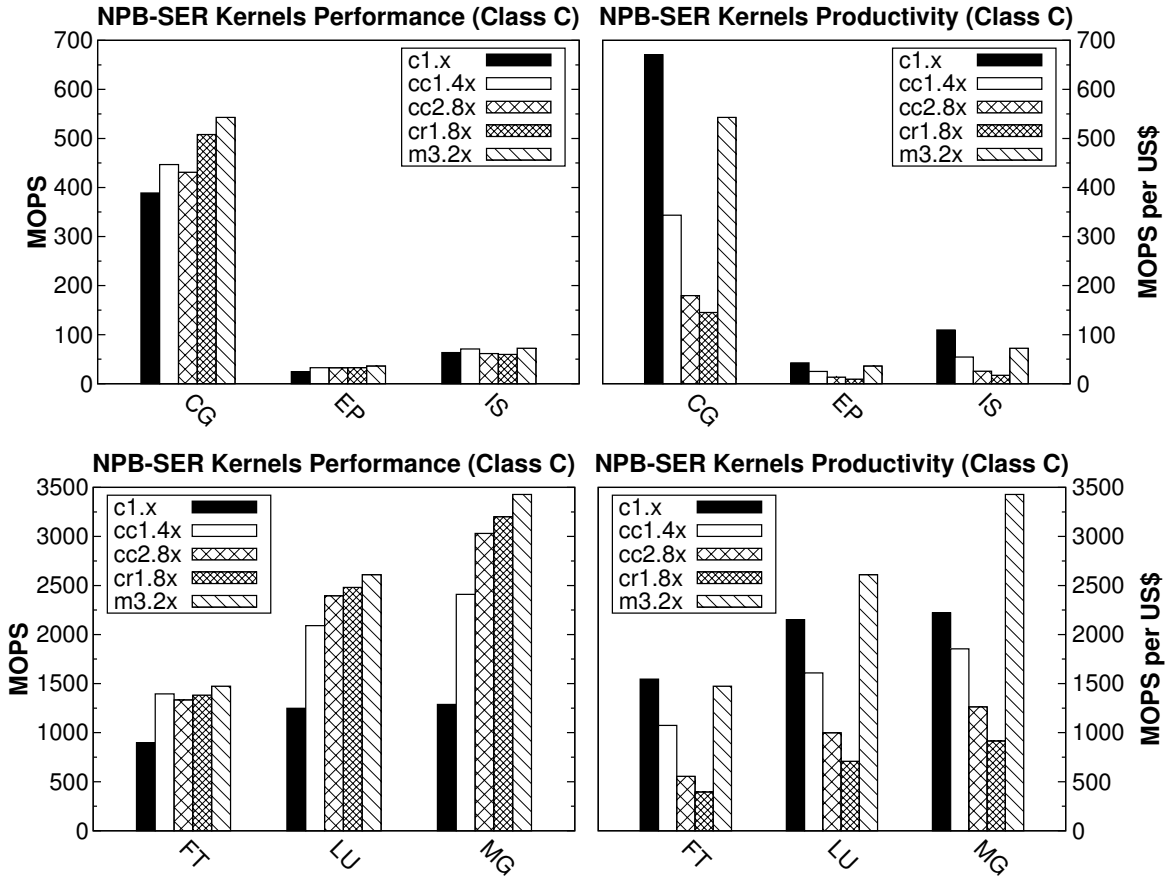

Figure 1: NPB-SER kernels performance and productivity

These are two representative models with very different execution times. They have been executed using the suggested values for the parameters and varying the number of threads until the maximum number of logical cores, as reported by the instance, was reached (i.e. using Hyper-Threading when available). Execution times are shown in Figure 3. Not surprisingly cluster instances get the best results for both models when using 8 threads and M3 slightly outperforms C1. Among the cluster instances CC1 obtains the worst results and there are no significative differences between CC2 and CR1. Note that for the second model when the number of threads is doubled from 8 to 16 the execution time is reduced near to the half, but when Hyper-Threading is used (32 threads instead of 16) the execution time does not improve too much.

5. Conclusions

Motivated by previous results that have shown that the scalability of HPC communication-intensive applications is very poor when using more than one EC2 cluster instance and that, from an economical point of view, is better to use EC2 lower computational power cluster instances, this paper has further evaluated the single-instance execution of scientific codes on some of the most computationally powerful EC2 instances. In the evaluation two high-end standard instances (C1 and M3) and three cluster instances (CC1, CC2 and the newest CR1) have been compared in terms of single instance performance, intra-VM scalability and cost-efficiency using both an HPC benchmark suite (the NPB kernels) and a real HTC application (JModelTest).

From the results the following conclusions can be drawn: (1) in terms of performance and intra-VM scalability, cluster instances obtain the best results both for HPC and HTC codes; (2) in terms of cost-efficiency, M3 and CC1 instances have in general the best cost-performance ratio when using 8 threads or less; (3) due to its higher costs and similar performance it is not worth using the newest CR1 instance as a substitute for the CC2 instance unless high-memory capacity requirements are a must; (4) unless the requirements of the application in number of threads and memory capacity preclude their use, non-cluster instances (M3 and, in some cases, C1) and less powerful cluster instance CC1 are an economical alternative despite their worst performance and speedup results.

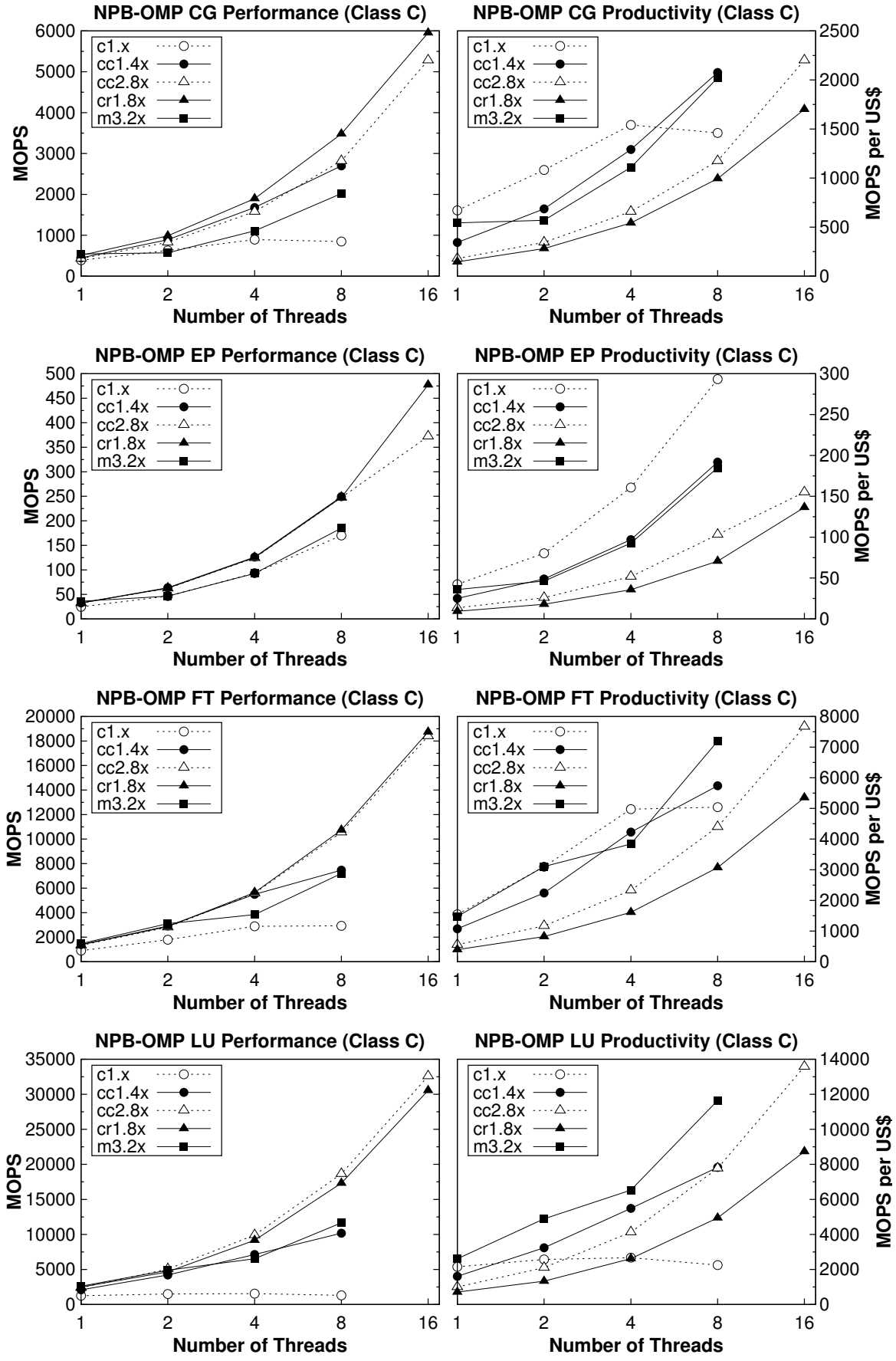

Figure 2: NPB-OMP kernels performance and productivity

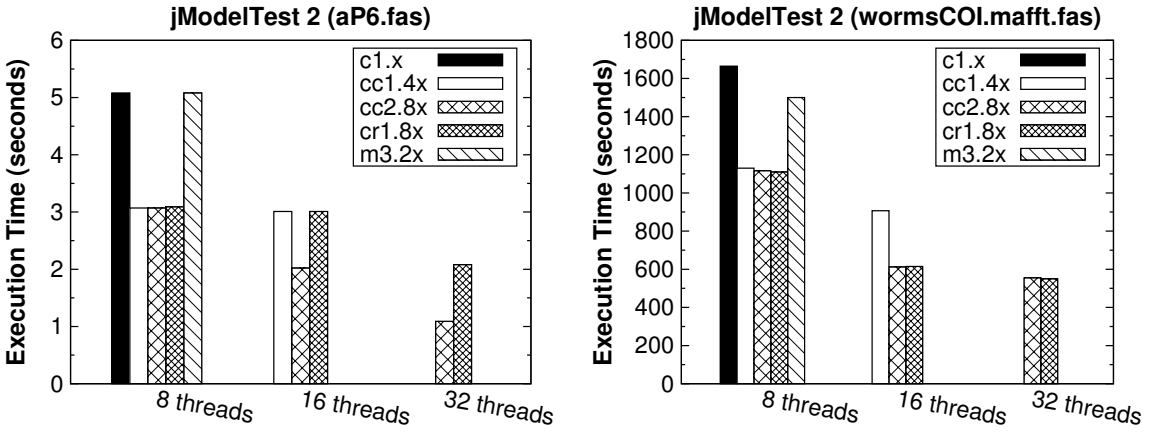

Figure 3: jModelTest 2 execution time

ACKNOWLEDGMENT

This work was funded by the Ministry of Science and Innovation of Spain and FEDER funds of the EU under Project TIN2010-16735, an FPU Grant AP2010-4348, and by an Amazon Web Services (AWS) LLC research grant.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] C. Evangelinos and C. N. Hill, "Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2," in *Proc. 1st Workshop on Cloud Computing and Its Applications (CCA'08)*, (Chicago, IL, USA), pp. 1–6, 2008.

[3] E. Walker, "Benchmarking Amazon EC2 for High-Performance Scientific Computing," *LOGIN: The USENIX Magazine*, vol. 33, no. 5, pp. 18–23, 2008.

[4] Amazon Web Services LLC, "High Performance Computing on AWS." `http://aws.amazon.com/hpc-applications/`. Last visited: Apr 2013.

[5] R. R. Expósito, G. L. Taboada, S. Ramos, J. Touriño, and R. Doallo, "Performance Analysis of HPC Applications in the Cloud," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 218 – 229, 2013.

[6] D. H. Bailey et al., "The NAS Parallel Benchmarks," *International Journal of High Performance Computing Applications*, vol. 5, no. 3, pp. 63–73, 1991.

[7] D. Darriba, G. L. Taboada, R. Doallo, and D. Posada, "jModelTest 2: More Models, New Heuristics and Parallel Computing," *Nat Meth*, vol. 9, p. 772, Aug. 2012.

[8] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud," in *Proc. 2nd IEEE Intl. Conference on Cloud Computing Technology and Science (CloudCom'10)*, (Indianapolis, USA), pp. 159–168, 2010.

[9] P. Luszczek, E. Meek, S. Moore, D. Terpstra, V. M. Weaver, and J. J. Dongarra, "Evaluation of the HPC Challenge Benchmarks in Virtualized Environments," in *Proc. 6th Workshop on Virtualization in High-Performance Cloud Computing (VHPC'11)*, (Bordeux, France), pp. 1–10, 2011.

[10] Y. Zhai, M. Liu, J. Zhai, X. Ma, and W. Chen, "Cloud Versus In-House Cluster: Evaluating Amazon Cluster Compute Instances for Running MPI Applications," in *Proc. 23th ACM/IEEE Conference on Supercomputing (SC'11, State of the Practice Reports)*, (Seattle, WA, USA), pp. 1–10, 2011.

[11] C. Sun, H. Nishimura, S. James, K. Song, K. Muriki, and Y. Qin, "HPC Cloud Applied to Lattice Optimization," in *Proc. 2nd Intl. Particle Accelerator Conference (IPAC'11)*, (San Sebastian, Spain), pp. 1767–1769, 2011.

[12] J. J. Rehr, F. D. Vila, J. P. Gardner, L. Svec, and M. Prange, "Scientific Computing in the Cloud," *Computing in Science and Engineering*, vol. 12, no. 3, pp. 34–43, 2010.

[13] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," in *Proc. 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, (Bolton Landing, NY, USA), pp. 164–177, 2003.