2016 IEEE 9th International Conference on Cloud Computing

Amazon Elastic Compute Cloud (EC2) vs. in-House HPC Platform: a Cost Analysis

Joseph Emeras*, Sébastien Varrette† and Pascal Bouvry†
*Interdisciplinary Centre for Security Reliability and Trust
†Computer Science and Communications (CSC) Research Unit
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg
Firstname.Name@uni.lu

***Abstract*—Since its advent in the middle of the 2000's, the Cloud Computing (CC) paradigm is increasingly advertised as THE solution to most IT problems. While High Performance Computing (HPC) centers continuously evolve to provide more computing power to their users, several voices (most probably commercial ones) emit the wish that CC platforms could also serve HPC needs and eventually replace in-house HPC platforms. If we exclude the pure performance point of view where many previous studies highlight a non-negligible overhead induced by the virtualization layer at the heart of every Cloud middleware when submitted to an High Performance Computing (HPC) workload, the question of the real cost-effectiveness is often left aside with the intuition that, most probably, the instances offered by the Cloud providers are competitive from a cost point of view. In this article, we wanted to assert (or infirm) this intuition by evaluating the Total Cost of Ownership (TCO) of the in-house HPC facility we operate since 2007 within the University of Luxembourg (UL), and compare it with the investment that would have been required to run the same platform (and the same workload) over a competitive Cloud IaaS offer. Our approach to address this price comparison is two-fold. First we propose a theoretical price - performance model based on the study of the actual Cloud instances proposed by one of the major Cloud IaaS actors: Amazon Elastic Compute Cloud (EC2). Then, based on our own cluster TCO and taking into account all the Operating Expense (OPEX), we propose a hourly price comparison between our in-house cluster and the equivalent EC2 instances. The results obtained advocate in general for the acquisition of an in-house HPC facility, which balance the common intuition in favor of Cloud Computing (CC) platforms, would they be provided by the reference Cloud provider worldwide.**

I. Introduction

Many public or private organizations have departments and workgroups that could benefit from High Performance Computing (HPC) resources to analyze, model, and visualize the growing volumes of data they need to conduct business. From a general perspective, HPC is essential for increased competitiveness and stronger innovation. There are two realistic scenarios today to access High Performance Computing (HPC) capacities beyond what is available from the desktop systems. One option is to acquire and operate an HPC system. However, for many companies and especially SMEs, this is seen as a non-viable solution since the Total Cost of Ownership (TCO) is perceived as too high, and additional skills and manpower are needed to operate and maintain such a system. With the rapidly growing enthusiasm around the Cloud Computing (CC) paradigm, and more particularly of the Infrastructure-as-a-Service (IaaS) model which is best suited for HPC workload, a second viable option is foreseen and attracts more and more attention due to the massive advertisement toward the cost-effectiveness of this approach. In this model, users rent to providers a *resource* that may be computing, storage and network or higher level services. At present, many large actors of the Cloud Computing market propose an IaaS service to their users. The cost model of IaaS is based on the actual resource usage of the user, who is thus billed depending on his activity. The computing resources are operated upon virtual machines and ran on a multi-tenant mode on the physical hardware. Characterized by their scalability and high-availability, IaaS platforms tend to be more and more efficient and are now sliding towards the territory of the traditional HPC facilities. Because of these interesting characteristics, many IaaS implementations have been largely studied and benchmarked in the context of an HPC usage. While it is now widely established that running an HPC workload on top of IaaS resources induces a non-negligible performance overhead due to the hypervisor at the heart of every Cloud middleware, many people assume that this performance impact is counter-balanced by the massive cost savings brought by the Cloud approach. *But is it true?* Since 2007, we operate within the University of Luxembourg (UL) a medium size HPC facility [1] ($\simeq$ 4300 cores, 50 TFlops and 4 PB of shared storage over parallel and distributed File Systems as of Jan. 2015) addressing the needs of academic researchers coming from many different fields (physics, material sciences, economy, life-science etc.). While developing our own expertise in the management of such a platform, it also gave us the opportunity to scrutinize on a daily basis and over a significant period of time its associated operating costs.

In this context, we propose in this paper a TCO analysis for our in-house HPC facility. Also, although the comparative performance of Cloud vs. HPC systems received a wide audience in the recent literature, the analogous analysis covering the costs remains at an early stage, with very few contribution from the academic community. This is also due to the constant and rapid evolution of the Cloud instances offered by the different providers and the frequent price changes, making it hard to establish a fair comparison of the Cloud usage price with regards the equivalent HPC infrastructure operational costs. Furthermore the direct comparison of a given Cloud instance price with an HPC operating cost is biased and should be taken with precautions as it omits the performance aspect where the Cloud instance performance does not match with the one of an HPC node. It is to feed this gap that we propose in this article a fair cost analysis. Our approach to address this

price comparison is twofold. First we propose a theoretical price - performance model based on the study of the actual Cloud instances proposed by one of the major Cloud IaaS actors according to Gartner's 2015 Quadrant: Amazon Elastic Compute Cloud (EC2). Then, based on our own cluster TCO and taking into account all the Operating Expense (OPEX) we propose a hourly price comparison between our in-house cluster and the EC2 instances.
This article is organized as follows: Section II reviews the related work as regards the cost analysis of Cloud platforms when compared to HPC facilities. Then, Section III summarize the characteristics of our in-house UL HPC, with a focus on the operating costs required for the TCO estimation. A first contribution *i.e.* this TCO analysis for our platform (reported in an average hourly node rate) is reported in this section. The Amazon EC2 instances, especially the ones relevant for HPC workloads and thus suited for a comparison with in-house HPC facilities, are depicted in Section IV. Then we proposed a novel insight over Amazon EC2 instance prices through a new cost model that matches almost perfectly the corresponding computing performance. This new approach and its theoretical validation is depicted in Section V. Furthermore, we apply this novel cost model to propose a fair comparison with the TCO analysis performed over our in-house HPC facility in Section VI. This permits to conclude on the opportunity to privilege one or the other solution when considering the acquisition (or the renting) of an HPC platform. Finally, Section VII concludes the paper and provides some future directions and perspectives opened by this study.

II. Related Work

A. Cloud Performance Evaluation for HPC Applications

Evaluating the adequateness and the performance of cloud platforms for HPC applications has been largely studied in various ways. Many works addressed this problem using either synthetic benchmarks, real applications or even both to quantify the performance impact when running over an hypervisor (such as Xen, KVM, VirtualBox or VMWare), a Cloud middleware (such as OpenStack) or a commercial Cloud IaaS [2], [3], [4], [5], [6], [7], [5], [8], [9], [10], [11], [12]. Depending on the experimental procedures, benchmarks and input data chosen, the loss due to the cloud varies among these experiments, but all agree that there exist a performance drop at scale. This performance drop has been explained in several works as due to poor networking capabilities of Clouds. Other elements are impacting the performances. For instance in [13], the performance evaluation of EC2 *cc1.4xlarge* instances based on real world climate and cardiac simulation application along with the NPB benchmark suite showed that not only the interconnect is determining but also the underlying file system, even on application that are classified as not strongly I/O intensive. In this work it was also observed that jitter due to virtualization has a minor effect on the applications performance. However as proposed in [14], some applications may still benefit from the cloud if their workload does not involve latency bound or communication bound patterns. But another important concern is the variability of performance coming from the intrinsic property of the cloud being location independent and multi-tenant [15]. This aspect could be tackled by software techniques such as the one described in [16], where the authors proposed a dynamic load balancer to reduce this variability for tightly-coupled iterative HPC applications based upon Charm++ and to be ran on the Cloud.

B. HPC vs. Cloud Cost Analysis and Modeling

If it is currently established that Cloud platforms performance does not fully match the requirements of HPC applications as would do a pure HPC facility, the acclaimed cost effectiveness of the CC paradigm might counter-balance this performance overhead. Actually, several works already attempted to describe or model the cost of using a public cloud infrastructure instead of an HPC facility. In [17] the EC2 *cc2.8xlarge* instance is used as reference with *on-demand* pricing to compare the price and performance of the Cloud versus five in-house clusters. To be able to do this comparison, they propose a cost model for HPC cluster nodes. As the authors could not get or distribute data describing the cost of ownership, they propose an alternative model. Based upon the execution times of four NPB Benchmarks on their EC2 reference instance and local clusters, the alternative model proposes by the means of linear regression to attribute a node hourly price to each of their clusters. This approach has the advantage to be able to price cluster nodes, however it is now outdated as EC2 offers several instances that would fit an HPC workload and not only the sole *cc2.8xlarge* instance that was available at this time. Thus this model should be updated taking into consideration the instances evolution. Another lack of this study is that only *on-demand* pricing has been study although it may not be the most adequate pricing type when the workload is considered on a long period (months). In that case, the Cloud price considered is artificially too high due to the pricing type, only adapted for short workloads. Thus the reserved pricing types should also be taken into account in the study. In [14], the authors propose a pricing model based on the study of cost and prices collected from several supercomputers installations and cloud offerings. The results of this comparison was that the cost ratio between an on-premise supercomputing and a cloud deployment is between 2 and 3 (meaning that one cluster core-hour cost is 2 to 3 times more expensive than a cloud's core-hour). Finally to better fit what may happen in reality due to price fluctuations they select a ratio between 1 to 5. Then, based on this model they compare the cost of running three applications on the cloud or on a cluster. According to this comparison, there is little sensitivity in the choice of the ratio and it is straightforward to determine the price break-even point. However, because of how the price model is computed the authors warn that this economic comparison should be taken conservatively. This motivates our approach of using real cost data for operating a local cluster. In [18] the authors analyzed the cost of the Amazon EC2 cloud from different point of view: Memory size or CPU performance. They propose rankings of the instances based on their price regarding a given metric (i.e. memory size and bandwidth or cpu performance). What was observable was that instances were designed to be cost effective for one of the metrics only. This observation motivates our approach to include several instance types in the price model instead of a single one. The work proposed in [19] approximates the cost of the local cluster nodes based on the EC2 *cc1* hourly price. To compute this node hourly price, they scaled the *cc1* price in order to reflect the performance differences between their cluster and the cloud instance. The authors warn that the price determined in this way should not be taken as a literal per node-hour cost. Despite the lack of representativeness of this approach it enabled the authors to determine a node hourly price for both *on-demand* and the different *reserved* pricing modes of EC2. In particular this showed that the users with the largest utilization rates could benefit from the reservation pricing mode on the *cc1* instance. Compared to the cluster operating cost it could lead to savings up to 3.75% for a 1-year contract.

Finally in [9], the comparison between the local cluster and the EC2 *cc1* instance was more straightforward as both had the same performance. The data taken into account for cluster pricing estimation was acquisition and operational expenses. However the authors approach diverge from ours and propose a price model of their in-house cluster based on the actual utilization of the cluster by the users. The claim here is that increased cluster utilization lowers the cluster cost. This is true if the point of view is job centric, however on a pure owner expense and operational vision it is wrong. The actual OPEX increases with the utilization as more energy is consumed by the nodes and thus the TCO is higher than for lower utilization rates. We propose first to compute the actual node hourly price based on the TCO with full utilization of the cluster to get the maximum cost. As the cost of renting an instance on the cloud is currently not dependent on the energy used (whether the instance is used at full load or not the price remains the same), it is thus fairer to compare the cloud instance price with the cost of operating the cluster at full load. With this information we can directly compare the cluster and cloud hourly costs. Then, based on the observation of the workload of the cluster we can easily determine the price per job and thus take into account the utilization in the final cost evaluation. In [20], the authors compare the cost of running the same workload both on a cloud and on an existing campus cluster of non-dedicated resources. They take as reference instance the EC2 *c1.medium* as its performance is close to their local nodes and use the *on-demand* pricing mode. In the local cluster cost model, the costs taken into account are manpower, hardware costs (including switches, servers and computing nodes) and energy costs. The conclusion was that operating their workload on the cloud would be more expensive but would give a better throughput as there are no queue wait time in the cloud. The other interesting conclusion was that the cost of the cloud could be easily dominated by data management costs; about a third of the cost incurred by their workload, if ran on the cloud, would be data transfer only. This, added to data storage prices should be taken into account to ensure that costs on the cloud are kept under control.

III. The UL HPC Platform: Presentation and TCO Estimation per Computing Node

The University of Luxembourg (UL) operates since 2007 an HPC facility [1] and the related storage by a very small team of system administrators. The aspect of bridging computing and storage is a requirement of UL service – the reasons are both legal (certain data may not move) and performance related. Nowadays, people from the three faculties and/or the two Interdisciplinary centers within the UL, are users of this facility. As of Jan. 2015, it consists of 4 clusters spread across 2 geographic sites featuring a total of 400 computing nodes (for 4284 cores) and a cumulative shared storage capacity of 4270

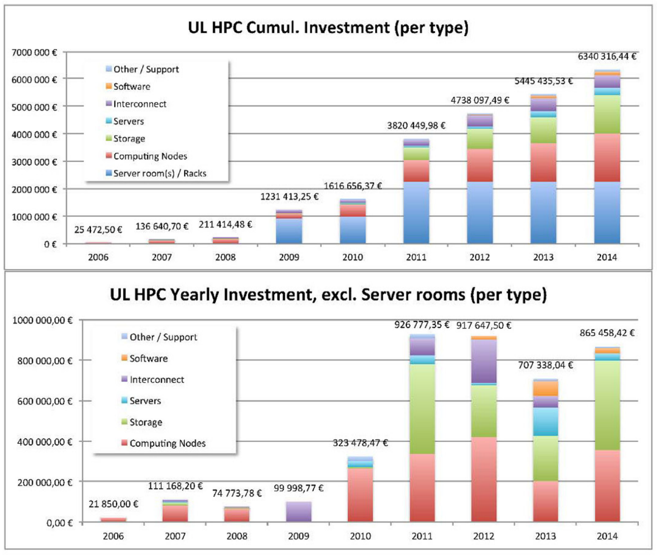


Figure 1. Investment in the UL HPC Facility for the time period 2007–2014. top: cumulative, incl. server rooms; bottom: yearly, excl. server rooms.

TB. In this study, we restrict the data and metrics collected to the two main clusters (namely *chaos* and *gaia*) that compose UL HPC facility – they contribute up to 80% (resp. 98%) of the total platform computing (resp. storage) capacity. Within these clusters, the internal interconnect is based on an Infiniband QDR (40Gb/s) network built on top of and adapted topology ("fat tree" and/or "star" depending on the situation) to ensure the best performances for the process communications. It is worth to note that such a technology out-performs by far all Ethernet-based network in use among major Cloud providers (including Amazon). In all cases, the cumulative Hardware investment has reached 6.34 M€ by the end of 2014 and its evolution is depicted in the figure 1.

Table I. UL HPC Manpower

Year	PM effort	Total Annual Cost
2006	1	11 553,63 €
2007	4	32 118,40 €
2008	19	64 200,24 €
2009	13	60 450,24 €
2010	9	57 864,00 €
2011	13	87 379,20 €
2012	25	136 947,68 €
2013	49	194 280,10 €
2014	40	173 026,90 €

The manpower is also a piece of information often missing in many cost analysis. In the table I, these costs are highlighted in the case of our facility. They reflect a reality where despite all our efforts, we were (and are still) outnumbered. While the TCO analysis proposed in this article relies on these numbers (more precisely, on the normalized prices for the year 2014), we estimate to 7.35 FTEs (thus $\simeq$ 88 Person Months (PMs)) the ideal manpower that would be required to maintain the platform at its current sizing, in the repartition reported as appendix, in the table VI. Another crucial question when performing a TCO analysis is the hypothesis linked to the usage of the platform. In our case, we have the chance to ensure a consistent and precise monitoring of the HPC resource usage (CPU, memory etc.) with regards the submitted jobs.

Table II. CHAOS AND GAIA COMPUTING NODES CHARACTERISTICS AND ESTIMATED TCO EVALUATION PER COMPUTING NODE.

	Node	CPUs	Memory GB	GPUs	Nb. Nodes	CPU Family	CPU Type	CPU Clock	Disk GB	GFLOPS	**Hourly Cost ($)**
CHAOS	d-cluster1	12	24	0	16	westmere	L5640	2.26	250	108.48	0.386
	e-cluster1	16	32	0	16	sandybridge	E5-2660	2.20	250	281.60	0.380
	h-cluster1	12	24	0	32	westmere	L5640	2.26	250	108.48	0.375
	r-cluster1	32	1024	0	1	nehalem	X7560	2.26	250	289.28	1.760
	s-cluster1	16	32	0	16	sandybridge	E5-2660	2.20	250	281.60	0.380
GAIA	gaia-[1-60]	12	48	0	60	westmere	L5640	2.26	256	108.48	0.400
	gaia-[123-154]	12	48	0	32	westmere	X5675	3.07	256	147.36	0.291
	gaia-[61-62]	12	24	1792	2	westmere	L5640	2.26	256	108.48	0.588
	gaia-[63-72]	12	24	10240	10	westmere	L5640	2.26	256	108.48	0.545
	gaia-[75-79]	16	64	12480	5	sandybridge	E5-2660	2.20	256	281.60	0.524
	gaia-[83-122]	12	48	0	40	westmere	X5670	2.93	256	140.64	0.291
	gaia-73	160	1024	0	1	sandybridge	E7-4850	2.00	256	2560.00	2.596
	gaia-74	32	1024	0	1	sandybridge	E5-4640	2.40	256	614.40	1.463

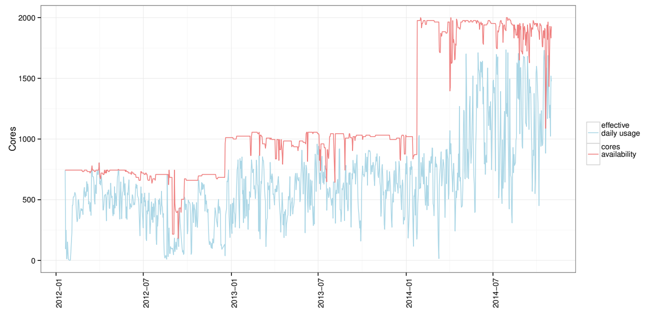


Figure 2. Example of used vs. available computing resources within the gaia cluster.

For instance, a typical usage of the *gaia* cluster resources is illustrated in the figure 2. It permits to derive an accurate ratio in terms of *used resources* with regards to the available computing components. From this information, we are able to estimate the TCO of our local HPC facility, more precisely of the two clusters chaos and gaia. These clusters being heterogeneous and composed of different node classes as reported in Table II, we compute the hourly cost of a node belonging to each node class. In all cases for the calculation of the TCO, we take into account the amortized node purchase price, network and server equipment price, room equipment, manpower and energy (power and cooling) costs. Computing nodes are considered amortized on a four year basis, network equipment on eight years and server equipment on three years. Server rooms on their side are supposed to be amortized in 15 years. Unfortunately the only thing we could not integrate into the TCO is the building cost itself as the HPC center is hosted within the UL which, as a research institutions, benefit from governmental funding for its buildings. For obvious privacy reasons, we cannot disclose the detailed buying price of the machines. Yet the final amortized node hourly costs are reported in the last column of the Table II. To facilitate the comparison with Amazon EC2 instances prices, we reported this TCO in US Dollar rather than in Euro.

IV. AMAZON EC2 INSTANCES FOR HPC

Since its launch in 2006 with the *m1.small* instance, Amazon has largely expanded the variety of instances and features it provides through its Elastic Compute Cloud (EC2) offer. They are now proposed in several families, each corresponding to a different computing need. Along the years and at new instance type releases, Amazon decreased the prices of the already existing ones. To ensure backward compatibility, most older instances are still distributed, however they do not provide a good price/performance ratio anymore regarding the most recent ones. Table III presents all the available instance types and their release dates as of early 2015 (except for the *cc1.4xlarge* which is not available anymore). An instance type e.g. *m1* or *c1* belongs to an instance family, e.g. *General Purpose* or *Compute Optimized*. For each instance type, there exist one or several models available that are not presented in this table. Instance models are described by an extension of the instance type (e.g. *m1.small*), possible values are: micro, small, medium, large, xlarge, 2xlarge, 4xlarge, 8xlarge. A given instance model corresponds to a particular performance of the instance in terms of vCPU, Memory, Storage, Network or such other performance characteristics.

Table III. EC2 INSTANCE TYPES – GROUPED BY FAMILY.

Instance Family	**Instance Type**	**Processor Microarchitecture**	**Introduction Date**
General Purpose	m1	Xeon Family	2006-08-23
	m3	Ivy Bridge-EP	2012-10-31
	t2	Xeon Family	2014-07-01
	m4	Haswell-EP	2015-06-11
Memory Optimized	m2	Xeon Family	2010-02-22
	cr1	Sandy Bridge-EP	2013-01-21
	r3	Ivy Bridge-EP	2014-04-10
Compute Optimized	c1	Xeon Family	2008-08-08
	cc1	Nehalem-EP	2010-07-13
	cc2	Sandy Bridge-EP	2011-11-14
	c3	Ivy Bridge-EP	2013-11-14
	c4	Haswell-EP	2014-11-13
Storage Optimized	hi1	Xeon Family	2012-07-18
	hs1	Sandy Bridge-EP	2012-12-21
	i2	Ivy Bridge-EP	2013-12-20
Dense Storage	d2	Haswell-EP	2015-03-30
GPU	cg1	Nehalem-EP	2010-11-14
	g2	Sandy Bridge-EP	2013-11-05
Micro	t1	Xeon Family	2009-10-26

To summarize, the cloud characteristics (which are also valid for EC2) that may impact HPC applications are the following: (1) clouds are run in *virtualized environments*. Although the virtualization overhead is not so important for CPU bound applications, the virtualized networking is still an important drawback for applications that need communications (i.e a large part of HPC applications). Even though the networking uses virtualization improvements such as Single Root I/O Virtualization (SR-IOV), there exists a performance drop for inter-node communications. (2) there is generally no high performance network such as Infiniband yet available. This is still a problem for many applications whose performance is highly linked with the network performance. (3) cloud by default uses multi-tenancy. This is a real problem as it does not ensure a reliable and stable behavior for applications. In particular, I/O operations in multi-tenancy platforms can face high I/O jitter. However EC2 also provides a charged *Dedicated Instances* service to ensure the user will have the exclusive access to the nodes reserved. (4) spatial and link proximity of the reserved Virtual Machines (VMs) is not guaranteed by default. Because of this intrinsic characteristic, inter-node bandwidth and latency can vary a lot. As for multi-tenancy, EC2 also provides for some instances the *placement group* feature, a logical grouping of instances within the same Availability Zone (isolated location within the same region) ensuring a low-latency, 10 Gb network between the nodes.

If we now aim at the execution of an HPC workload, one of the key constraint to address is the need of a reliable and efficient underlying network. Not all EC2 instances offer such a guarantee, however there exists a mechanism called *placement group* available for some of them and that allows the user to group a set of instances in the same cluster. This is an important requirement for HPC applications that use several nodes. Here we consider that most *classical* HPC applications fall into that category and thus need to be placed within the same cluster. In the Table IV, we detail the EC2 instances that allow placement groups and thus may be suitable for such an HPC workload. It is also important to say on an HPC point of view that the nodes that host the EC2 instances have HyperThreading activated, thus each vCPU on an instance is actually a HyperThread. In all cases, Amazon's EC2 provides several mechanisms to target High Performance on its cloud instances. (1) Enhanced Networking with SR-IOV. This hardware feature, proposed on the most efficient instances, provides higher network performance for the VMs: higher bandwidth and lower network latency and jitter. The instances were this feature is available are *c3*, *c4*, *r3*, *i2* and *d2*. Unfortunately GPU instances do not provide the Enhanced Networking feature but we have to consider them anyway for an HPC environment. (2) Placement Groups (or cluster networking). This feature provides high bandwidth performance with a full bisection Ethernet network. (3) Dedicated Instances. By default on EC2, cloud instances are multi-tenant VMs hosted on the same physical node. By opting for the dedicated instance option (paying) at launch time, the instance will have exclusive access to the underlying physical machine. (4) EBS-Optimized. Elastic Bloc Store (EBS) is Amazon's persistent storage for EC2 instances (as opposed to instances local storage which is destroyed after the instance termination). It can be purchased with a provisioned IO option to increase and guarantee data transfer rates. EBS volumes have a maximum throughput of 128 MiB/s that can be attained only with instances proposing the EBS-optimized option. One of the last instance type released by Amazon at this time, the *c4* instance (end of 2014), proposes EBS storage with no additional costs but does not provide local storage.

V. A Novel Price Model binding EC2 Instances with their HPC Performance

EC2 instance performances are provided in EC2 Compute Units (ECUs). This metric represents the relative measure of the integer processing power of a given instance. Amazon does not give precisely how they measure ECUs, the only information available is that one ECU is equivalent to one Intel Xeon core running at 1.0-1.2 GHz. It is also clearly stated by Amazon that this metric might be revised in the future by adding or substituting measures that go into the definition of an ECU. According to Amazon, the goal of providing instance performance as ECU is to make it easier to compare CPU capacity between different instance types and thus to "*provide a consistent amount of CPU capacity no matter what the actual underlying hardware*". Although the claim of using several benchmarks and tests to manage the consistency and predictability of the performance from an EC2 Compute Unit, it seems that this information might be sometimes misleading. In [21], the authors showed that for several instance types the performance measured experimentally was unreliable, unpredictable and not reflecting the ECU provided by EC2. This was due to the fact that the instances that were tested in this study were running indifferently on several types of Central Process Units (CPUs) and it seems that for instances whose CPU model is described as being a *Xeon Family*, there is no guarantee of the actual CPU model on which the instance will be ran. In another study driven by Iosup et al. in [8], the authors experimentally compared the actual GFLOPS regarding the announced ECUs for *m1* and *c1* instances and observed a real performance comprised between 39 to 44% of the advertised performance for *m1* instances and most of *c1* instances with an exception of 58.6% for the *c1.xlarge* instance.
As ECU is a non standard way to measure a processor performance and it actually makes it difficult to fairly compare the computing performance of an EC2 instance with an HPC node. Moreover, as this metric is not provided with a strict definition and as it was observed performance mismatch between the announced ECU and actual performance we need another metric such as FLOPS which is the traditionally used theoretical processor computing performance metric. As according to its definition, an ECU is equal to one Xeon core at 1.0-1.2 GHz, the ECU should reflect the processor FLOPS value.

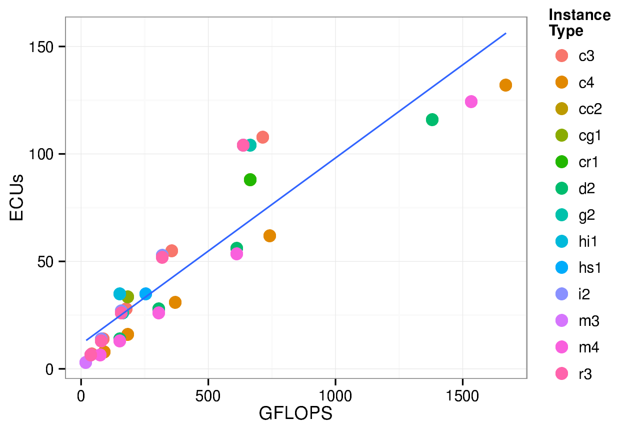


Figure 3. ECUs vs. theoretical peak performance GFLOPS.

In Figure 3, we compare the instances ECU provided by Amazon regarding the theoretical GFLOPS peak from the corresponding instance processor. In this figure we take into account only the instances where the CPU model is precisely defined. We can observe that there is indeed a linear relationship between GFLOPS and ECUs. The linear model describing this gives an adjusted R^2 value of 0.9 and a p-value of $2.2e-16$. However, we also observe that while ECU has a

Table IV. EC2 INSTANCES SUITABLE FOR HPC WORKLOAD AND THEIR KEY CHARACTERISTICS.

Family	Type	ECUs	GFLOPS	vCPUs	Processor	Clock	Memory	Storage	GPUs	EBS-Opt	Network	SR-IOV
Compute optimized	c3.2xlarge	28.0	179.2	8	Xeon E5-2680v2	2.8	15.00	2 x 80 (SSD)	0	TRUE	1 Gib	TRUE
Compute optimized	c3.4xlarge	55.0	358.4	16	Xeon E5-2680v2	2.8	30.00	2 x 160 (SSD)	0	TRUE	1 Gib	TRUE
Compute optimized	c3.8xlarge	108.0	716.8	32	Xeon E5-2680v2	2.8	60.00	2 x 320 (SSD)	0	FALSE	10 Gib	TRUE
Compute optimized	c3.large	7.0	44.8	2	Xeon E5-2680v2	2.8	3.75	2 x 16 (SSD)	0	FALSE	100 Mib	TRUE
Compute optimized	c3.xlarge	14.0	89.6	4	Xeon E5-2680v2	2.8	7.50	2 x 40 (SSD)	0	TRUE	100 Mib	TRUE
Compute optimized	c4.2xlarge	31.0	371.2	8	Xeon E5-2666v3	2.9	15.00	EBS only	0	TRUE	1 Gib	TRUE
Compute optimized	c4.4xlarge	62.0	742.4	16	Xeon E5-2666v3	2.9	30.00	EBS only	0	TRUE	1 Gib	TRUE
Compute optimized	c4.8xlarge	132.0	1670.4	36	Xeon E5-2666v3	2.9	60.00	EBS only	0	TRUE	10 Gib	TRUE
Compute optimized	c4.large	8.0	92.8	2	Xeon E5-2666v3	2.9	3.75	EBS only	0	TRUE	100 Mib	TRUE
Compute optimized	c4.xlarge	16.0	185.6	4	Xeon E5-2666v3	2.9	7.50	EBS only	0	TRUE	100 Mib	TRUE
Compute optimized	cc2.8xlarge	88.0	665.6	32	Xeon E5-2670	2.6	60.50	4 x 840	0	FALSE	10 Gib	FALSE
Dense storage	d2.2xlarge	28.0	307.2	8	Xeon E5-2676v3	2.4	61.00	6 x 2000	0	TRUE	1 Gib	TRUE
Dense storage	d2.4xlarge	56.0	614.4	16	Xeon E5-2676v3	2.4	122.00	12 x 2000	0	TRUE	1 Gib	TRUE
Dense storage	d2.8xlarge	116.0	1382.4	36	Xeon E5-2676v3	2.4	244.00	24 x 2000	0	TRUE	10 Gib	TRUE
Dense storage	d2.xlarge	14.0	153.6	4	Xeon E5-2676v3	2.4	30.50	3 x 2000	0	TRUE	100 Mib	TRUE
GPU instances	cg1.4xlarge	33.5	185.6	16	Xeon x5570	2.9	22.00	2 x 840	896	FALSE	10 Gib	FALSE
GPU instances	g2.2xlarge	26.0	166.4	8	Xeon E5-2670	2.6	15.00	1 x 60 (SSD)	1536	TRUE	1 Gib	FALSE
GPU instances	g2.8xlarge	104.0	665.6	32	Xeon E5-2670	2.6	60.00	2 x 120 (SSD)	6144	FALSE	10 Gib	FALSE
Memory optimized	cr1.8xlarge	88.0	665.6	32	Xeon E5-2670	2.6	244.00	2 x 120 (SSD)	0	FALSE	10 Gib	FALSE
Memory optimized	r3.2xlarge	26.0	160.0	8	Xeon E5-2670v2	2.5	61.00	1 x 160 (SSD)	0	TRUE	1 Gib	TRUE
Memory optimized	r3.4xlarge	52.0	320.0	16	Xeon E5-2670v2	2.5	122.00	1 x 320 (SSD)	0	TRUE	1 Gib	TRUE
Memory optimized	r3.8xlarge	104.0	640.0	32	Xeon E5-2670v2	2.5	244.00	2 x 320 (SSD)	0	FALSE	10 Gib	TRUE
Memory optimized	r3.large	6.5	40.0	2	Xeon E5-2670v2	2.5	15.00	1 x 32 (SSD)	0	FALSE	100 Mib	TRUE
Memory optimized	r3.xlarge	13.0	80.0	4	Xeon E5-2670v2	2.5	30.50	1 x 80 (SSD)	0	TRUE	100 Mib	TRUE
Storage optimized	hi1.4xlarge	35.0	153.6	16	Xeon Family	2.4	60.50	2 x 1024 (SSD)	0	FALSE	10 Gib	FALSE
Storage optimized	hs1.8xlarge	35.0	256.0	16	Xeon E5-2650	2.0	117.00	24 x 2048	0	FALSE	10 Gib	FALSE
Storage optimized	i2.2xlarge	27.0	160.0	8	Xeon E5-2670v2	2.5	61.00	2 x 800 (SSD)	0	TRUE	1 Gib	TRUE
Storage optimized	i2.4xlarge	53.0	320.0	16	Xeon E5-2670v2	2.5	122.00	4 x 800 (SSD)	0	TRUE	1 Gib	TRUE
Storage optimized	i2.8xlarge	104.0	640.0	32	Xeon E5-2670v2	2.5	244.00	8 x 800 (SSD)	0	FALSE	10 Gib	TRUE
Storage optimized	i2.xlarge	14.0	80.0	4	Xeon E5-2670v2	2.5	30.50	1 x 800 (SSD)	0	TRUE	100 Mib	TRUE

strong linear relationship with GFLOPS within a same instance type, the more we add instance types in that comparison, the more it deviates from the linearity. The relationship between the ECU and GFLOPS is thus more complex than that and if we also consider the processor generations (i.e. Xeon Core, Nehalem, Sandy Bridge, Ivy Bridge or Haswell) we obtain a better linear fitting (adjusted R^2: 0.96). This is remarkable as it should be a redundant information with GFLOPS. It seems thus that ECU is more complex than just a relationship with the sole GFLOPS metric as claimed by Amazon, ECU seems also dependent on the processor generation and probably some other characteristics.
Because of this lack of information on how is really calculated an ECU and the fact that there exists a linear relationship between ECU and GFLOPS, we propose to use GFLOPS instead of ECU in the sequel. This will enable us to compare the processing performance of EC2 instances and HPC computing nodes with a single metric that is well known and that can actually be measured.

Toward a new Amazon EC2 Price Model linked to their respective Computing Performance

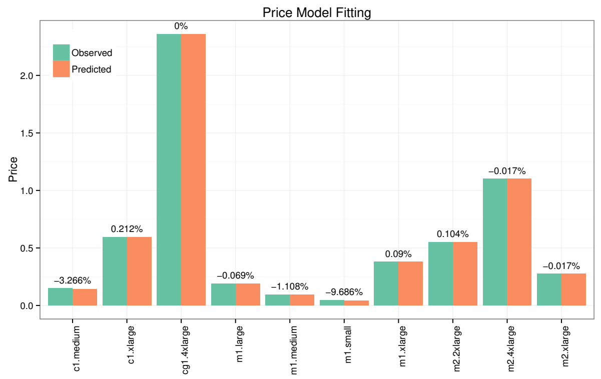
Only two regions are available in Europe at this time, one in Ireland and one in Frankfurt (Germany), we choose to focus on the Ireland region as its instance prices are cheaper and Frankfurt region was launched in the end of 2014 and does not offer all the instance types yet. In order to have a coherent instance price that enables to compare with the cluster costs and taking into account the hardware performance, we propose to use a multiple linear regression approach in order to compute an Amazon EC2 price model. Based on this model we will be able to determine for our local machines a fair EC2 equivalent price that will reflect their hardware performance, as proposed later in the section VI. Note that Amazon offers different purchasing modes for its instances. The "*On-Demand*" mode is the most common and allows to pay a fixed rate by the hour with no commitment. In the rest of this section, we present our computed results for our price model using the proposed rate for this mode. So to establish an accurate price model, we have to take into consideration additional characteristics than the pure peak performance of the considered resource (in GFLOPS). Indeed, we have observed that instances that belong to the same type have prices that tend to be linearly dependent on their performance. *i.e.* instance *c3.4xlarge* is two times more expensive than *c3.2xlarge* but its number of cores, available memory and disk size are two times larger. Thus for each instance type we compute a pricing linear model based on linear regression analysis. We select the model parameters in two steps. (1) First we perform an automated stepwise selection, then (2) from the meaningful parameters detected we manually assess the ones that are the most representative in the model via R^2 shrinkage.
Among many parameters evaluated, we established that the significant ones are: Processor speed (GFLOPS), Memory size (GB), Disk Size (GB) and number of GPU cores. It follows that the new price model for EC2 instances prices can be described through the Equation 1:

$$\begin{aligned} Instance\_Price = {} & \alpha * GFLOPS \\ & + \beta * Memory\_GB \\ & + \gamma * Disk\_GB \\ & + \delta * Nb\_GPUs \end{aligned} \tag{1}$$

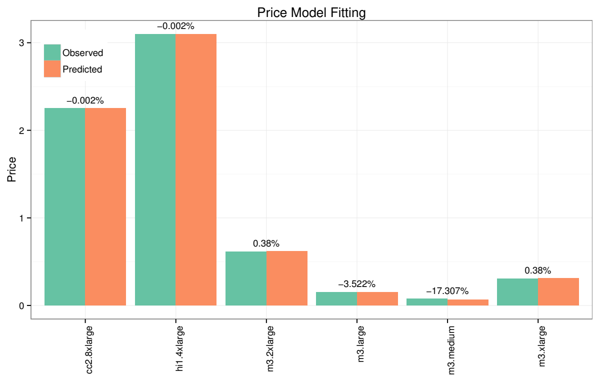
Furthermore, we have detected that the instances that were released in the same time period could be grouped in the same price model without changing too much its fitting. We thus end up with five pricing models, depending on the instance *generation*, *i.e.* release period of the instance. We labeled these

Table V. COEFFICIENTS OF OUR NEW EC2 PRICE MODEL FOR ON DEMAND INSTANCES, WITH FITTING PERFORMANCE EVALUATION.

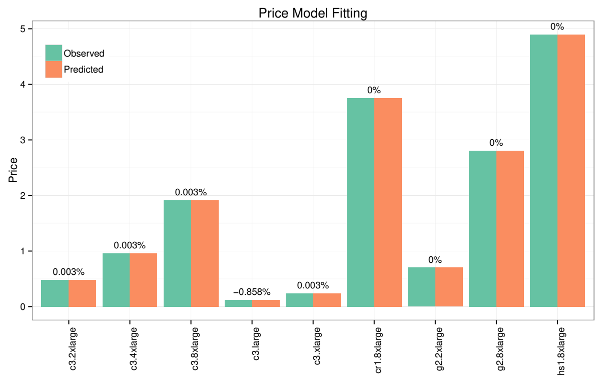
Model	**Types**	**GFLOPS** (α)	**MemGiB** (β)	**DiskGiB** (γ)	**GPUs** (δ)	**Adj.** R^2	**P-Value**
1st Gen.	m1, c1, m2, cg1	0.0039522	0.0061130	0.0000670	0.0015395	0.9999909	0e+00
2nd Gen.	cc2, m3, hi1	-0.0035266	0.0355353	0.0007284	0.0000000	0.9999785	1e-07
3rd Gen.	hs1, cr1, g2, c3	0.0017209	0.0106101	0.0000655	0.0001644	1.0000000	0e+00
4th Gen.	i2, r3, c4	0.0009952	0.0081883	0.0007605	0.0000000	0.9998832	0e+00
5th Gen.	m4, d2	0.0000000	0.0173750	0.0000342	0.0000000	1.0000000	0e+00



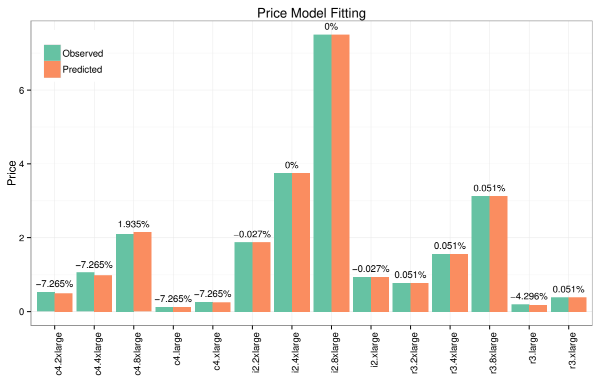
(a) 1st Gen.



(b) 2nd Gen.



(c) 3rd Gen.



(d) 4th Gen.

Figure 4. Evaluation of Proposed EC2 Price Models Fittings (with error rates) against actual On Demand instance prices.

models "1st Gen", "2nd Gen", "3rd Gen", "4th Gen" and "5th Gen" to reflect this matching instance generation. Refining this process several times gives us five models, whose parameters for the On Demand pricing model are described in Table V. We have evaluated the five models individually and we provide these results in Figure 4, the error rate in percentage is also given for each instance. Due to space limitation and because it present a perfect linear fitting, the 5th model evaluation figure is not presented here. We can see that the fitting is very good for all the models and **the error rate is low**.

VI. APPLICATION OF THE NEW EC2 PRICE MODEL AGAINST THE IN-HOUSE UL HPC FACILITY

A. *Hourly Price Model*

With the new price model computed in V, we are now able to give an EC2 equivalent price for each class of node of our local clusters. This means that even though not all the cluster nodes have a perfect cloud instance match, we are still able to determine what would be its equivalent price on EC2 if that matching instance was available. This information is later used to assess the interest of operating a given node class regarding renting an On Demand instance with the same performance on the cloud. Such an estimation is proposed in Figure 5 which presents the ratio, for each node class in the cluster, of the *EC2 Equivalent Price* based on the closest possible existing EC2 instance, versus the operating cost of the considered node as evaluated in the section III (see Table II). We insist on reminding that the operating cost provided in Table II represents the maximum cost scenario where the HPC infrastructure is used at 100%. For a lower utilization of the cluster, this operating cost will be decreased due to lower energy usage. What we observe is that for some node classes, the ratio is close to 1, meaning that operating in-house these HPC nodes costs about the same price as renting the corresponding instance on Amazon. Thus it may be interesting

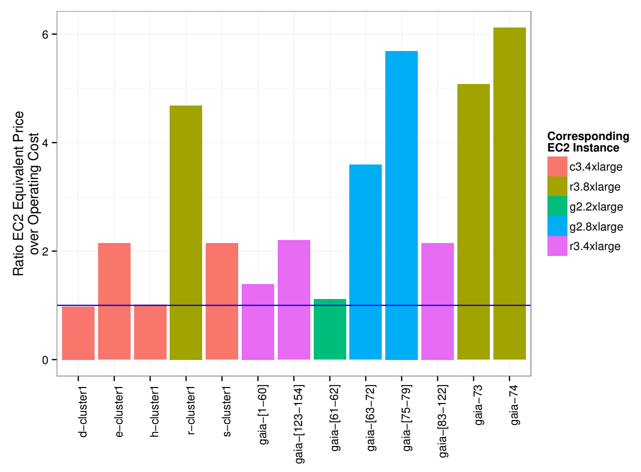


Figure 5. Ratio between the EC2 Equivalent Price of our in-house HPC Resources with regards their respective TCO estimation.

to rent these kind of resources on EC2 instead of operating them locally. However for **all the other ones** the results are more mitigated. Some node classes also have low ratio (around 2) but some others have very high ratio (more than 4). For these the renting of such kind of instance would be simply too costly.

The above comparison is based on "On Demand" instance prices. In this purchase model, the customer pays a fixed rate by the hour with no commitment and we just showed that in this scenario, the acquisition of an in-house HPC facility is probably a more cost-effective solution. Actually, Amazon proposes cheaper options for the instance prices:

- *Reserved* instances provide a capacity reservation, and offer a significant discount on the hourly charge for an instance
- *Spot* instances enable to bid a price for a given instance, and thus may lead to even greater saving

This last renting mode remains too hard to model accurately. However we extended our pricing model against the purchased rate offered in the "Reserved" mode. There are three payment options in this case: (1) *No Upfront*: this option provides access to a Reserved Instance without requiring an upfront payment, (2) *Partial Upfront*: this option requires a part of the Reserved Instance to be paid upfront and the remaining hours in the term are billed at a discounted hourly rate, regardless of usage. (3) *All Upfront*: a full payment is made at the start of the term, with no other costs incurred for the remainder of the term regardless of the number of hours used. In all cases, for reserved purchase option, the important thing to know is that the option is attached to one instance only and the reservation price has to be paid even though the instance is used or not. Thus the corresponding upfront price has to be paid for each EC2 instance needed. Moreover, for the reserved purchase it is possible to subscribe a contract based on 1 year or 3 years. However due to the frequent releases of new EC2 instances and the price drops of the older ones, it seems too risky to subscribe to a 3 years contract even though the savings are more important than for the 1 year basis. Because of this we will consider only the 1 year reserved option.

While we also achieved a very good fitting for all the models with a low error rate, it seemed more pertinent to us to propose from these results a general overview of the in-house HPC nodes costs when compared to all the considered EC2 purchasing mode (*i.e.* On Demand, All Upfront, Partial Upfront and No Upfront) and prices for the different renting modes. This complete cost analysis is depicted in the Figure 6. It can be seen that there exist a few cases (for the `d-cluster1-*`, `h-cluster1-*` and `gaia-[61,62]` nodes) where the EC2 Reserved instances are more cost-effective than the in-house resources. Again, for all the other cases, and thus even when a cheaper purchasing alternative is chosen for EC2 instances, it remains more efficient, from a pure cost point of view, to consider the acquisition of the corresponding HPC resources rather than renting them on the Cloud.

B. Yearly Price Model Applied to Real Cluster Usage

In the previous section we compared the costs of operating a local cluster node versus its equivalent price on EC2 based on our cost model. We saw that depending on what the user specifically asks to the system in terms of resources to run his job, the associated price largely varies. Now we investigate how this price difference impacts the costs on a yearly basis. For that purpose we analyzed all job requests on the reference period of year 2014. Using the logs form the batch scheduler we can extract which nodes were allocated to the jobs. Based on the prices computed in Figure 6 and the knowledge of nodes associated to jobs we are capable of calculating the yearly cost for the year 2014 to operate the in-house cluster and the EC2 equivalent price for the following pricing modes: On Demand, Reservation with all upfront, partial upfront and no upfront.

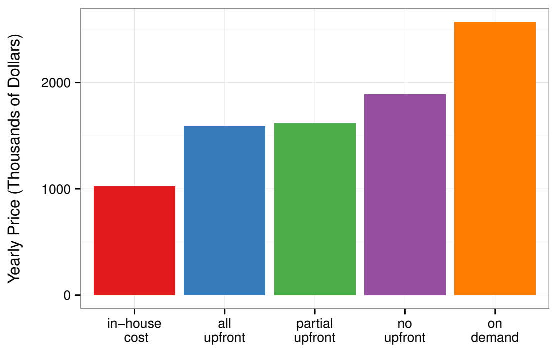


Figure 7. Annual cost for year 2014 of UL workload if operated locally or rented on EC2.

We present these results in Figure 7. It may seem surprising at first that the cost of operating the cluster locally is 40% cheaper than renting on EC2 with the all upfront reserved mode and 2.5 times less expensive for on demand pricing. Actually, a large part of the workload from *chaos* and *gaia* use nodes with a low local operating cost compared to their *EC2 relative price*. This result particularly shows that the migration of HPC workloads to the cloud is not only a problem of adaptability of the cloud performance to HPC needs but also a problem of correctly determining which types of job are good candidates to be run on the cloud in order to prevent such cost overheads.

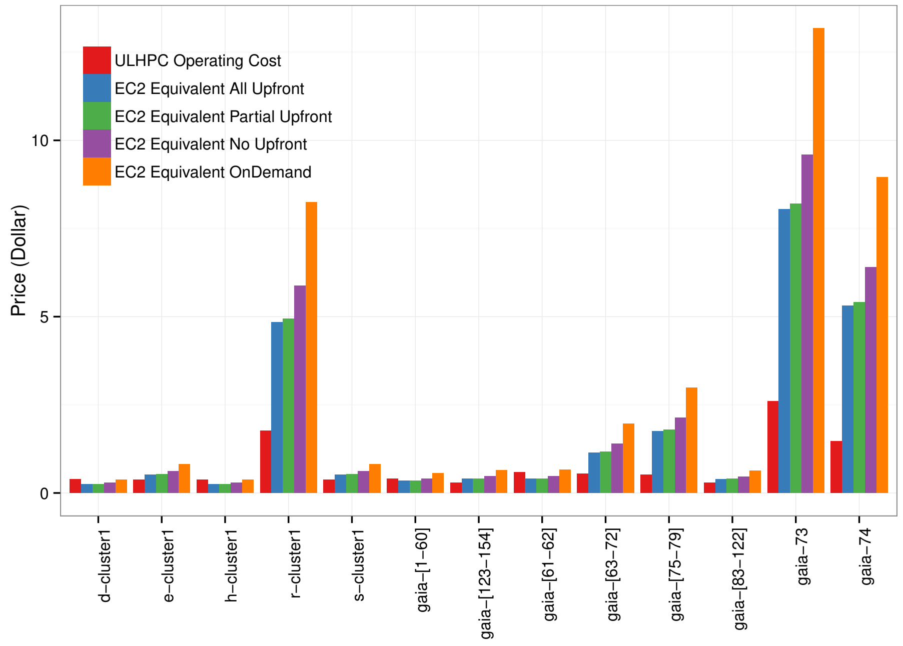


Figure 6. Comparison of the in-house HPC resources operating cost with regards an equivalent Amazon EC2 instance for the different purchasing options (On demand, All upfront, Partial upfront and No upfront).

VII. Conclusion

The objective of the present study was to evaluate the real cost-effectiveness of Cloud Computing (CC) platform when compared to an in-house HPC facility. To answer this question, we propose in this article three main contributions:

1) A representative Total Cost of Ownership (TCO) analysis of a medium-size academic HPC facility, which is based on our own experience while managing this platform within our research institution since 2007.
2) A novel price model for the instances offered by the main Cloud IaaS actors (Amazon EC2) able to compete with the performances of such an HPC platform. Our proposed cost model is flexible and rely on the inherent performance metrics (from an HPC perspective) of the considered instance resources.
3) The application of this new model against the resources of our local clusters to offer, when confronted to the operating cost evaluated in the first part of this study, a fair comparison of the prices and thus of the investments linked to each of the solution (local own infrastructure vs. rented resource on the Cloud).

Our proposed cost analysis is not only **accurate** (the fitting of our novel cost model exhibits a low error rate), it also **advocates in general in favor of the acquisition of an in-house HPC facility**.

The perspective opened by the work are manifold. Firstly, we wish to extend our analysis over the Spot instance that permit to bid for the price of the resources. This offers a chance for better cost-savings on the instance rate prices, and thus might indicate other classes of HPC resources for which the renting option makes sense. Also, we hope to benefit from the planned move to a new campus, and thus the possibility to monitor the building cost when implementing the brand new HPC server rooms within this new site, to integrate the building cost in our TCO analysis. This will also give us the opportunity to update our model with regards the cost of cutting-edge HPC technologies (Direct Liquid Cooling equipment, Infiniband EDR interconnect etc.)

Acknowledgments: The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [1] – see http://hpc.uni.lu.

References

[1] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos, "Management of an academic hpc cluster: The ul experience," in *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*. Bologna, Italy: IEEE, July 2014, pp. 959–967.

[2] X. Besseron, V. Plugaru, A. H. Mahmoudi, S. Varrette, B. Peters, and P. Bouvry, "Performance Evaluation of the XDEM framework on the OpenStack Cloud Computing Middleware," in *PARENG 2015*. Dubrovnik, Croatia: Civil-Comp Press, Mar. 24–28 2015.

[3] M. Guzek, S. Varrette, V. Plugaru, J. E. Pecero, and P. Bouvry, "A Holistic Model of the Performance and the Energy-Efficiency of Hypervisors in an HPC Environment," *Intl. J. on Concurrency and Computation: Practice and Experience (CCPE)*, vol. 26, no. 15, pp. 2569–2590, Oct. 2014.

[4] S. Varrette, V. Plugaru, M. Guzek, X. Besseron, and P. Bouvry, "HPC Performance and Energy-Efficiency of the OpenStack Cloud Middleware," in *Proc. of the 43rd Intl. Conf. on Parallel Processing (ICPP-2014), Heterogeneous and Unconventional Cluster Architectures and Applications Workshop (HUCAA'14)*. Minneapolis, MN, US: IEEE, Sept. 2014.

[5] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Nov 2010, pp. 159–168.

[6] Z. Hill and M. Humphrey, "A quantitative analysis of high performance computing with amazon's ec2 infrastructure: The death of the local cluster?" in *Grid Computing, 2009 10th IEEE/ACM International Conference on*, Oct 2009, pp. 26–33.

[7] S. Akioka and Y. Muraoka, "Hpc benchmarks on amazon ec2," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, April 2010, pp. 1029–1034.

[8] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of ec2 cloud computing services for scientific computing," in *Cloud Computing*, D. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Springer Berlin Heidelberg, 2010, vol. 34, pp. 115–131.

[9] 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," ser. SC '11. New York, NY, USA: ACM, 2011, pp. 11:1–11:10. [Online]. Available: http://doi.acm.org/10.1145/2063348.2063363

[10] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, "Case study for running hpc applications in public clouds," ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 395–401. [Online]. Available: http://doi.acm.org.proxy.bnl.lu/10.1145/1851476.1851535

[11] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini, and R. Biswas, "Performance evaluation of amazon ec2 for nasa hpc applications," ser. ScienceCloud '12. New York, NY, USA: ACM, 2012, pp. 41–50. [Online]. Available: http://doi.acm.org.proxy.bnl.lu/10.1145/2287036.2287045

[12] R. R. Exposito, G. L. Taboada, S. Ramos, J. TouriÃśo, and R. Doallo, "Performance analysis of {HPC} applications in the cloud," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 218 – 229, 2013, including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X12001458

[13] P. Strazdins, J. Cai, M. Atif, and J. Antony, "Scientific application performance on hpc, private and public cloud resources: A case study using climate, cardiac model codes and the npb benchmark suite," in *IPDPSW 2012*, May 2012, pp. 1416–1424.

[14] A. Gupta, P. Faraboschi, F. Gioachin, L. Kale, R. Kaufmann, B.-S. Lee, V. March, D. Milojicic, and C. Suen, "Evaluating and improving the performance and scheduling of hpc applications in cloud," *Cloud Computing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[15] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *ICPPW 2010*, Sept 2010, pp. 275–279.

[16] A. Gupta, O. Sarood, L. Kale, and D. Milojicic, "Improving hpc application performance in cloud through dynamic load balancing," in *CCGrid 2013 13th IEEE/ACM*, May 2013, pp. 402–409.

[17] A. Marathe, R. Harris, D. K. Lowenthal, B. R. de Supinski, B. Rountree, M. Schulz, and X. Yuan, "A comparative study of high-performance computing on the cloud," in *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*. ACM, 2013, pp. 239–250.

[18] I. Sadooghi, J. Hernandez Martin, T. Li, K. Brandstatter, Y. Zhao, K. Maheshwari, T. Pais Pitta de Lacerda Ruivo, S. Timm, G. Garzoglio, and I. Raicu, "Understanding the performance and potential of cloud computing for scientific applications," *Cloud Computing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[19] A. Carlyle, S. Harrell, and P. Smith, "Cost-effective hpc: The community or the cloud?" in *CloudCom 2010 IEEE*, Nov 2010, pp. 169–176.

[20] A. S. McGough, M. Forshaw, C. Gerrard, S. Wheater, B. Allen, and P. Robinson, "Comparison of a cost-effective virtual cloud cluster with an existing campus cluster," *Future Generation Computer Systems*, vol. 41, no. 0, pp. 65 – 78, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X14001344

[21] J. O'Loughlin and L. Gillam, "Towards performance prediction for public infrastructure clouds: An ec2 case study," in *CloudCom 2013 IEEE*, vol. 1, Dec 2013, pp. 475–480.

APPENDIX

Table VI. IDEAL MANPOWER REPARTITION TO OPERATE A MEDIUM-SIZE HPC FACILITY

SMP Services	**1 Full Time Engineer (FTE)**
Operating Systems	10%
Compilers	10%
COTS Applications	15%
Open Source Applications	30%
Storage	20%
Change Management	10%
Hardware	5%
SAN Storage and Parallel File System	**1 FTE**
Hardware	30%
Software (GPFS,Lustre,OneFS...)	50%
Allocation and Provisioning	10%
Capacity Planning	10%
Parallel Cluster	**1.6FTE**
Operating Systems	10%
Compilers	10%
COTS Applications	20%
Open Source Applications	30%
Storage	20%
Change Management	10%
Hardware	5%
Queueing systems	10%
Internal Cluster Network	5%
Monitoring	20%
Database	**1 FTE**
Installation and Configuration	10%
Tuning and Optimization	20%
Backups and DR	10%
Database Design	40%
Database Administration	20%
Web Services	**40% FTE**
Operating Systems	10%
Apache	10%
Application Server	10%
Server-based security	10%
Backup / Recovery	**35% FTE**
Server Admin	20%
Client Admin	10%
Capacity Planning	5%
Scientific Programming	**2 FTE**
Development	100%
Debugging/Testing	60%
Profiling/Optimization	40%