[Article] A cost-effective cloud computing framework for accelerating multimedia communication simulations

A Cost-Effective Cloud Computing Framework for Accelerating Multimedia Communication Simulations

Daniele Angeli, Enrico Masala*

Control and Computer Engineering Dept., Politecnico di Torino, corso Duca degli Abruzzi, 24 — 10129 Torino, Italy.
Phone: +39-011-0907036, Fax: + 39-011-0907099

Abstract

Multimedia communication research and development often requires computationally intensive simulations in order to develop and investigate the performance of new optimization algorithms. Depending on the simulations, they may require even a few days to test an adequate set of conditions due to the complexity of the algorithms. The traditional approach to speed up this type of relatively small simulations, which require several develop-simulate-reconfigure cycles, is indeed to run them in parallel on a few computers and leaving them idle when developing the technique for the next simulation cycle. This work proposes a new cost-effective framework based on cloud computing for accelerating the development process, in which resources are obtained on demand and paid only for their actual usage. Issues are addressed both analytically and practically running actual test cases, i.e., simulations of video communications on a packet lossy network, using a commercial cloud computing service. A software framework has also been developed to simplify the management of the virtual machines in the cloud. Results show that it is economically convenient to use the considered cloud computing service, especially in terms of reduced development time and costs, with respect to a solution using dedicated computers, when the development time is higher than one hour. If more development time is needed between simulations, the economic advantage progressively reduces as the computational complexity of the simulation increases.

Keywords: Multimedia Simulations, Cloud Computing, Video Communication, Amazon EC2, Cloud Cost Comparison

1. Introduction

Nearly all works that propose new algorithms and techniques in the multimedia communication field include simulation results in order to test the performance of the proposed systems.

*Corresponding author. Email: masala@polito.it. NOTICE: this is the author's version of a work that was accepted for publication in Journal of Parallel and Distributed Computing. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version is being published in Journal of Parallel and Distributed Computing. DOI: 10.1016/j.jpdc.2012.06.005

Validation of new algorithms and ideas through simulation is indeed a fundamental part of this type of research due to the complexity of multimedia telecommunication systems.

However, the performance of the proposed systems has to be evaluated in many different network scenarios and for several values of all the key parameters (e.g., available bandwidth, channel noise). Moreover, to achieve statistically significant results, simulations are often repeated many times and then results are averaged. In addition, for research purposes, software and simulators are usually developed only as a prototype, i.e., not optimized for speed. For instance, the video test model software available to researchers is usually one or two order of magnitude slower than commercial software which, however, might not be suitable for research, since it does not come with the source code needed to experiment with new techniques.

As a consequence, the time spent in running such type of simulations and getting results might be significant, sometimes even a few days. Interpreting such results leads to performance improvement and bug fixes that need to be tested again with other simulations. Focusing on the development stage of these simulations and relatively small size simulations implies that there is the potential for several development-simulation-reconfiguration cycles in a single day, and computational resources are idle between simulation runs. Therefore, the time needed to get simulation results play a key role in trying to speed up the research activities.

A commonly used approach to speed up simulations is to run them in parallel on several computers. However, this approach strongly depends on several variables, e.g., computer power and availability. Buying new, dedicated computers might not be affordable in case of small research groups since the number of computers should be high and their usage ratio would be low due to the dead times between the various simulation runs needed to improve the algorithms and fix bugs. Accessing large computer resources could be difficult as well since currently it is not easy to acquire resources to spend in computation costs when the hardware is not owned. Indeed in typical research projects which include funding for multimedia communication research, high-performance computing is not seen as one of the primary goal of the project, and costs are usually dominated by items such as staff and development of testbeds. Therefore, the cost and risk of acquiring a significant number of computers entirely rest on the research group.

An effective technique to speed up simulations could be to rent computing resources in the cloud and run the computations in parallel, however there is a significant lack of works in literature that quantify the advantages or disadvantages of such a solution especially in terms of economic costs in a practical case. This paper addresses this issue by providing quantitative results, including cost comparisons, that can help in taking the most effective decisions.

Note that the type of scientific tasks considered in this work does not fit well into the class of high performance computing (HPC) problems, since in that case requirements are different: the problem is well known, and algorithms to solve it are well tested. The requirement is generally limited to run those type of algorithms in the most cost-efficient way, which typically implies they are batch-scheduled. In the considered scenario, instead, researchers wants to run the simulation code as soon as possible to speed up further improvements.

The type of simulations addressed in this work are better described by the many task computing (MTC) definition, which denotes high-performance computations comprising multiple distinct activities, coupled via file system operations [1]. Multimedia communication simulations considered here are fully parallelizable by nature, making them perfectly suitable for a cloud computing environment. The possibility to parallelize simulations stems from the fact that results are usually averaged on a number of different simulation runs that do not have dependency among them. Moreover, using several values for the parameters as the input of the algorithms adds another dimension to the problem which again increases the possibility to further parallelize

the simulation.

The contribution of this paper is twofold. First, it provides a simple software framework that can be used to automate all the operations involved in setting up and manage the cloud environment for the specific simulation to be run as well as to efficiently and quickly collect the simulation results. Second, it analyzes the cost-performance tradeoff using several actual simulation examples taken from the video communication research area, i.e., H.264/AVC video communications on a packet lossy channel. An analytical approach is employed in order to investigate both the economic costs and the performance of the proposed approach. Moreover, actual prices of a major commercial cloud computing provider are used to quantify, in a practical way, the suitability, economic profitability and development speed up of employing the cloud computing approach for the relatively short scientific simulations, such as the ones faced by many researchers, in a realistic scenario where the typical work pattern of researchers is also considered, i.e., they do not work 24 hours while computers do.

The paper is organized as follows. Section 2 analyzes the related work in the field. Then, Section 3 investigates the requirements of typical simulations in the multimedia communication field and their suitability for cloud computing. Section 4 describes the developed software framework to automate running simulations in the cloud. In Section 5, a brief performance analysis of the various instances in the Amazon cloud computing infrastructure is presented. Section 6 describes the case studies used in this work, followed by Section 7 which analytically investigates the cost performance tradeoffs with practical examples in the case of both a single simulation and a whole research activity comprising several simulations. Conclusions are drawn in Section 8.

2. Related work

Some works have been presented in recent years on the profitability of using cloud computing services in order to improve the performance of running large scientific applications. Cloud-based services indeed claim that they can achieve significant cost savings over owned computational resources, due to the pay-per-use approach and reduced costs in maintenance and administration which are spread on a large user basis [2].

Until recently, most of the scientific tasks were run on clusters and grids, and many works explored how to optimize the performance of scientific applications in such specific contexts. A taxonomy of scientific workflow systems for grid computing is presented in, e.g., [3]. However, cloud is not a completely new concept with respect to grids, it indeed has intricate connection to the grid computing paradigm and other technologies such as utility and cluster computing, as well as with distributed systems in general [4].

Several works investigated several different aspects involved in running scientific workflows in the cloud, for instance focusing on optimal data placement inside the cloud [5], the overall experience and main issues faced when the cloud is used [6] and the suitability of cloud storage systems such as Amazon S3 for the scientific community [7].

Other works addressed the costs of using cloud computing to perform tasks traditionally addressed by means of an HPC approach with dedicated computational resources. Findings indicate that in this scenario profitability is somehow limited, at least with current commercially available cloud computing platforms [2]. Indeed the performance of general purpose cloud computing systems, such as the virtual machines provided by Amazon [8], are generally up to an order of magnitude lower than those of conventional HPC clusters [9] and are comparable to low-performance clusters [10]. Nevertheless, due to the savings achieved by means of the large scale of these cloud

computing systems, they seem to be a good solution for scientific computing workloads that require resources in an instant and temporary way [11], although capacity planning can be quite difficult since traditional capacity planning models do not work well [12]. Many works employ benchmarks aimed at predicting the performance of complex scientific applications. Often, these benchmarks focus on testing the efficiency of the communication between the various computing nodes, which are an important factor in some types of applications. For instance, [13] focuses on establishing theoretical performance bounds for the case of a large number of highly parallel tasks competing for CPU and network resources.

The type of simulations addressed in this work fits into the many task computing (MTC) definition [1]. MTC has been investigated, for instance, in [14], where a data diffusion approach is presented to enable data intensive MTC, in particular dealing with issues such as acquiring computing and storage resources dynamically, replicating data in response to demand, and scheduling computations close to data both under static and dynamic resource provisioning scenarios. Frameworks for task dispatch in such scenarios have also been proposed recently, such as Falcon [15], which simplify the rapid execution of many tasks on architectures such as computer clusters by means of a dispatcher and a multi-level scheduling system to separate resource acquisition from task dispatch. These frameworks are complemented by means of languages suitable for scalable parallel scripting of scientific computing tasks, such as Swift [16].

Some works have been presented to compare the performance achieved by means of the cloud with other approaches based on desktop workstations, local clusters, and HPC shared resources with reference to sample scientific workloads. For instance, in [17] a comparison is performed among all these approaches, mainly focusing on getting reliable estimate of prediction of performance of the various architectures depending on the workflows. However, no economic cost comparisons between the different platforms are shown. Another work [18] consider a practical scientific task traditionally run on a local cluster. The authors study a cloud alternative based on the Amazon infrastructure, first developing a method to create a virtual cluster using EC2 instances to make portability easier, then investigating how the different data storage methods provided by Amazon impact on the performance. While costs of Amazon cloud are considered in details for the proposed architectures, no cost comparisons with the previous cluster-based architecture are presented.

This work helps in quantifying the economic advantage and potential drawbacks in replacing computers dedicated to simulation in a small research lab with a cloud computing solution. To the best of our knowledge, no works have addressed this issue so far with reference to the relatively small size simulations presented in this paper, apart from our short preliminary study presented in [19]. Even though this might seem a quite peculiar simulation scenario, many researchers, at least in the multimedia communication field, share the need to perform simulations of the size discussed here. Note also that the computational requirements of these simulations are constantly increasing due to the tendency to move towards high quality, high resolution images and video, urging researchers to find cost effective ways to deal with these type of simulations.

3. Analysis of Simulation Requirements

Typical simulations in the multimedia communication field involves running the same set of algorithms many times with different random seeds at each iteration. The objective is to evaluate the performance of the system under test by averaging the results achieved in various conditions, e.g., different realizations of a packet lossy channel, so that confidence intervals are minimized. Clearly, such a setup allows many simulations to run in parallel, since no interaction among them

is required except when merging the results at the end of the simulation. Despite the conceptual simplicity, the actual computational load can be high since multimedia codecs might be prototypes only, not optimized for speed, as well as channel models or other robustness techniques might be complex to simulate. Moreover, consider that to achieve statistically significant results many different test signals, e.g., video sequences, should be used in the experiments so that techniques are validated across a range of different input conditions.

Other similarly heavy tasks might include extensive precomputations in order to optimize the performance of algorithms that are supposed to run in real time once deployed in an actual system. As an example, consider a system optimizing the transmission policy of packets in a streaming scenario. Some precomputed values regarding the characteristics of the content, such as the distortion that would be caused by the loss of some parts of the data, can be useful to the optimization algorithms, but values need to be computed in advance (see, e.g., [20, 21]).

Therefore, due to the typical peculiarities of multimedia communication simulations, very little effort is needed to parallelize them. Often, no interaction is needed until collection of results (or not at all when considering different input signals). Even in more complex cases, such as precomputation, usually multimedia signals can be easily split into different independent segments, for instance group of pictures (GOP) in video sequences, and processed almost independently.

Therefore, the simulation types described in this section can take full advantage of the availability of multiple computing units, as in a cloud environment. Parallelism can be exploited both at the CPU level, using more CPUs, and within the CPU taking advantage of multiple cores. As with modern computers, cloud environments offer multicore CPUs in the highest performance tiers, which indeed require parallelism for a cost effective exploitation of the resources.

4. The Cloud Simulation Software Framework

In this work we focus on the Amazon AWS offer as of October 2011, which provides the "Elastic Compute Cloud" service, in brief "Amazon EC2", that includes a number of instance types with different characteristics in terms of CPU power, RAM size and I/O performance. The Amazon AWS platform allows to control the deployment of resources in different ways, for instance by using a web interface or by means of an API, available for different languages. In all cases (web or API), the deployment of virtual systems in a remote environment and their monitoring requires several operations, although conceptually simple. While for simple operations and management of virtual servers this task can be easily accomplished by a human operator through, e.g., the web interface, a more time efficient system is needed to manage at the same time the activation, configuration and deactivation of a number of instances in order to automatically create the requested virtual environment needed by the simulations. A simulation could, in fact, require to create, for instance, ten virtual machines, each one fed with different input parameters so that it operates on the correct set of data, then check that every one of them is correctly running and finally resulting data has to be collected in a single central point.

Since the simulations considered in this work are quite short when carried out in the cloud computing environment, the time spent in setting up the appropriate simulation environment (e.g., activating instances, feeding them with the correct startup files, etc.) must be minimized otherwise the advantage of cloud computing in terms of speed is reduced. To minimize the set up time, an automatic system is needed.

A number of frameworks have been proposed in literature to address the issue of dispatching tasks to a computer system (e.g., a cluster or a grid) where they are usually received by batch

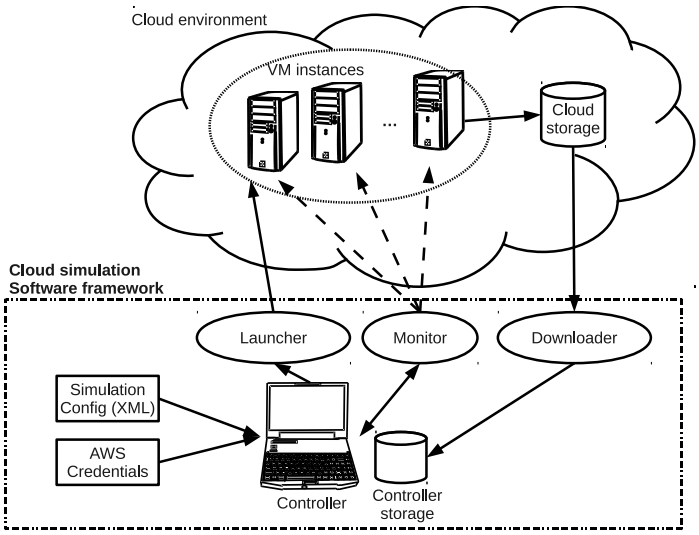

Figure 1: General architecture of the proposed cloud simulation software framework.

schedulers. However, their dispatching time can be high [15] because they usually support rich functionalities such as multiple queues, flexible dispatch policies and accounting. Lightweight approaches have also been proposed, for instance the Falkon framework [15], which reduces the task dispatch time by means of eliminating the support for some of the features.

However, in a relatively small size simulation with almost no dependencies between the tasks the use of such frameworks, aimed at large scale scientific simulations, provides much more features that the ones effectively needed. For these reasons, we designed and implemented our software framework, named Cloud Simulation System (CSS), with the aim to create a very lightweight support for the execution of our simulations. The main design criteria were to be able to automate the execution in the cloud of the simulations of the type considered in this work, and to automatically take care of all the aspects of configuration of the cloud, e.g., starting and terminating instances, uploading the initial data for each instance and downloading the results. Note that these aspects must be adapted to the specific cloud technology used regardless of which framework is employed, therefore also if more complex frameworks were used, the time reduction in setting up the framework would have been limited, also considering the time needed to learn and adapt the features of a new framework for our aims.

Figure 1 shows the general architecture of the CSS. The software has been developed in the Java language in order to be portable on different platforms. First, an offline step is needed, that is, the preparation of a virtual machine image, named AMI in the Amazon terminology, containing the tools needed for the simulation and a few parameterized commands, usually scripts, that can both run a set of simulations (controlled by an input file) and save the simulation results to a storage system, for instance the S3 provided by Amazon.

One of the key component of the architecture is the *controller* computer, which is initially fed with a simulation description, in XML format, of the activities to carry on, including the specific set of input parameters for each single EC2 instance. The controller automatically performs a

```
<?xml version="1.0" encoding="UTF-8"?>
<simul tag="Sim1">
  <options>
    <cloudRegion>EU_Ireland</cloudRegion>
    <amiID>ami-12345678</amiID>
    <vmsKind>c1.medium</vmsKind>
    <cloudKeyPair>ec2-key</cloudKeyPair>
    <s3cmdLoc>/home/ubuntu/s3/s3cmd/s3cmd</s3cmdLoc>
    <s3configLoc>/home/ubuntu/.s3cfg</s3configLoc>
  </options>

  <cloudVM>
    <commandList dataToSave="res*.txt"
          execLoc="/home/ubuntu/simul/h264/">
      <command>./sim.sh list_set_3.txt</command>
    </commandList>
  </cloudVM>
  ...
</simul>
```

Figure 2: Sample simulation description file for the developed cloud simulation software framework.

number of activities needed to ensure the successful execution of the set of simulations specified in the XML file. The main activities include:

1. activating new instances;
2. running the simulation software;
3. periodically monitoring the status of the instances to get early warnings in case some software included in the simulation fails or crashes;
4. checking for the end of the simulation;
5. downloading the results from the remote storage systems.

In more details, the system creates instances using the API provided by the Amazon Java SDK. The software is packaged in a runnable JAR archive and the main options and operations that will be performed are specified in the XML configuration file. Several parameters can be specified, such as the number and type of Amazon instances to use, in which region they will be launched and which commands they will execute at startup. It is also possible to specify a user defined tag to run more than one simulation set at the same time in the cloud. A sample file is shown in Fig. 2. A separate file contains the access credentials to the Amazon AWS platform.

The monitoring of the simulation is performed through a set of scripts that will periodically connect to all the instances involved in the simulation and check their memory and CPU utilization. If any of these metrics show anomalous values an automatic email alert will be sent to a predefined address, including the details of the instances that are having issues.

When all simulations end, files are downloaded from the Amazon S3 storage system, used by all the instances to save their results. The application developed in this framework will automatically detect the end of the simulation and then download the resulting data in the local system.

The described framework can efficiently run simulations in the Amazon AWS platform. In order to optimize the cost performance tradeoff, suitable options must be chosen in the configuration file, for instance the most efficient type of EC2 instance for the given simulation. The

Table 1: Characteristics of the available EC2 instances and costs in the EU region (Oct. 2011).

Name [Symbol]	ECU/core [E_i]	# cores [ν_i]	RAM (GB)	I/O perf.	cost/h ($)	cost/h/ECU ($) [$\varphi_i$]
std.small	1	1	1.7	Moderate	0.095	0.095
std.large	2	2	7.5	High	0.38	0.095
std.xlarge	2	4	15	High	0.76	0.095
hi-cpu.medium	2.5	2	1.7	Moderate	0.19	0.038
hi-cpu.xlarge	2.5	8	7	High	0.76	0.038
hi-mem.xlarge	3.25	2	17.1	Moderate	0.57	0.088
hi-mem.dxlarge	3.25	4	34.2	High	1.14	0.088
hi-mem.qxlarge	3.25	8	68.4	High	2.28	0.088
micro	up to 2	1	0.613	Low	0.025	-

next sections will investigate how to configure the developed framework in order to maximize the performance and minimize the cost of running the simulation in the cloud.

5. Performance Analysis of EC2 Instances

The computational power unit used by Amazon is the EC2 compute unit (ECU), defined as the equivalent to the CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. Table 1 summarizes the characteristics of the Amazon EC2 offer in the EU region as of Oct. 2011 [22]. Note that there is a particular type of instance, named *micro*, whose characteristics cannot be easily defined as the other ones. More details about the *micro* instance are presented later in this work. The key quantities peculiar of each instance are represented using the following symbols: φ_i is the cost/h/ECU for instance type i, ν_i is the number of cores and E_i the number of nominal ECU per core. The meaning of all the symbols used throughout the paper is reported in Table 2.

5.1. Raw Computing Performance

First, the CPU performance of the different instances has been assessed by using a simple CPU-intensive program, i.e., computing the MD5 hash of a randomly-generated 100 MB file. The experiment is repeated 100 times to cache the file into the RAM so that the performance of the storage system does not affect the measurements. Results are reported in Table 3.

The effective computing power $P_i^{(1)}$ of instance i is computed as:

$$P_i^{(1)} = \frac{t_{std.small}^{(1)} \cdot P_{std.small}^{(1)}}{t_i^{(1)}} \tag{1}$$

where $t_i^{(1)}$ is the time, as seen by the user, needed by the instance to compute the MD5 value 100 times using a single process. As a reference, the $P_i^{(1)}$ value for the *std.small* has been set equal to 1.00, so that values can be directly compared with the nominal speed in ECU as declared by Amazon AWS. Superscript $^{(1)}$ indicates that the performance is achieved using only one core.

Note also that the *micro* instance differs from the others since it is not suitable for a continuous computing load. It provides a good alternative for instances that are idle most of the time but they sometimes must deal with some short bursts of loads, such as low-traffic web servers. Moreover, there are no guarantees that a minimum amount of processing power will be available at any time even when the instance is running, making it a sort of "best effort" offer. For the *micro* instance,

Table 2: Symbols used throughout the paper.

i	Instance of type i
E_i	ECU/core for instance type i
ν_i	Number of cores of instance type i
φ_i	Cost (\$)/h/ECU of instance type i
$P_i^{(N)}$	Effective computing power of instance type i using N cores
$p_i^{(N)}$	Effective computing power parameter, i.e., $P_i^{(N)}$ normalized by E_i
$C_i^{(N)}$	Cost (\$) of one hour of instance type i considering its $p_i^{(N)}$
n	Number of processes running in parallel
S	Computing energy (ECU·h) needed to run a simulation
K	Number of instances used to run the simulation in the cloud
T_S	Time (h) required to run a simulation, requiring energy S, in the cloud
C_S	Cost (\$) of running a simulation, requiring energy S, in the cloud
η	Instance usage efficiency in the cloud
T_D	Time (h) spent to study and modify the algorithm in each cycle
T_C	Total time (h) of one cycle
n_{PC}	Number of PCs needed to achieve the same time performance of the cloud
L_{PC}	Lifetime (h) of a PC
C_{PC}	Cost (\$) of a PC including running costs for L_{PC}
N_{cy}	Number of cycles used for the development of a given technique
C_{cloud}	Total cost (\$) of the cloud solution
$C_{n_{PC}}$	Total cost (\$) of the solution based on n PCs
f	Time increase factor due to operators working during daytime only
C_{ratio}	Ratio of the cost of the cloud solution to the cost of the nPC-based solution

Table 3 shows the ECU/core declared by Amazon while time and effective computational power are averaged over several cycles of burst and slow-down periods. For completeness, note that when the *micro* instance performed at maximum computational speed, it reached $P^{(1)} = 3.38$ in our experiments, while it provided only $P^{(1)} = 0.09$ while in the slow phase. Due to this behavior, this type of instance will not be considered further in this work.

When multiple cores are available on a given instance processes can be run in parallel. Figure 3 shows the effective computing power of the various instances while performing the same CPU-intensive task (MD5 hash) using a different number of processes in parallel. The effective computing power parameter $p^{(N)}$ is given by Eq. (2), where the time interval $t_i^{(N)}$ refers to the time elapsed between the start of the first process and the end of the last running process, as

$$p_i^{(N)} = \frac{t_{std.small}^{(1)} \cdot P_{std.small}^{(1)}}{t_i^{(N)} \cdot E_i}. \tag{2}$$

Note that, differently from Eq. (1), the p value is normalized by the nominal ECU/core value E_i, so that values can be easily compared among them. As expected, performance tends to slightly decrease when the number of processes increases. This result confirms that each instance can be loaded with CPU-intensive parallel processes up to the number of cores without incurring in an unreasonable performance reduction.

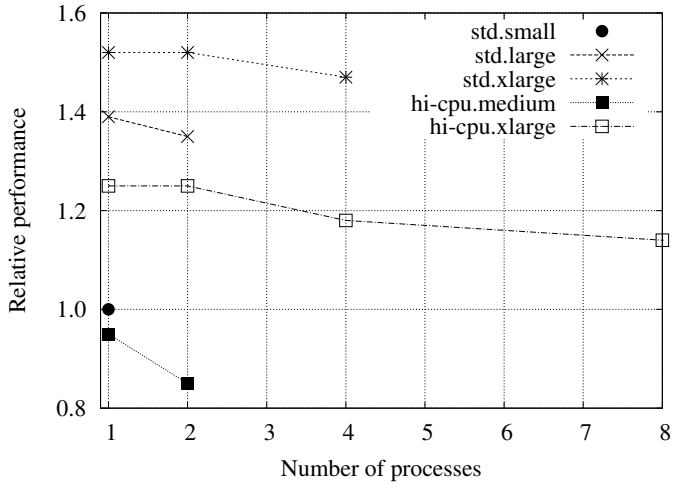

Figure 3: Effective computing power of parallel CPU-intensive tasks on different instances (normalized by the nominal ECU/cores).

Figure 4 shows the cost, per process, for one hour of each instance type for each effective computing power unit p as previously defined. The cost is defined as

$$C_i^{(N)} = \frac{p_i^{(N)}}{\nu_i E_i \varphi_i \cdot n} \tag{3}$$

where $\nu_i E_i \varphi_i$ is the cost of one hour of instance i and n is the number of processes running in parallel. Since the cost of the instance is constant regardless of the number of processes running on it, clearly the cost per process decreases when the number of processes run in parallel in the instance increases, up to the number of available cores. Note that each single point in Fig. 4 considers the effective computing power units p that can be achieved with that specific number of processes. Considering the MD5 hash task, the best performance cost ratio is provided by the *hi-cpu.xlarge* instance type, followed by the *std.xlarge*, *std.large* and *hi-cpu.medium*.

Table 3: Experimental measurements of CPU computing performance of EC2 instances, using only one core. Time refers to the MD5 task.

Name	Nominal ECU/core	Time (s)	Effective comp. power P (std.small=1.00)
std.small	1	100	1.00
std.large	2	36	2.78
std.xlarge	2	33	3.03
hi-cpu.medium	2.5	42	2.38
hi-cpu.xlarge	2.5	32	3.13
micro	up to 2	152	0.66

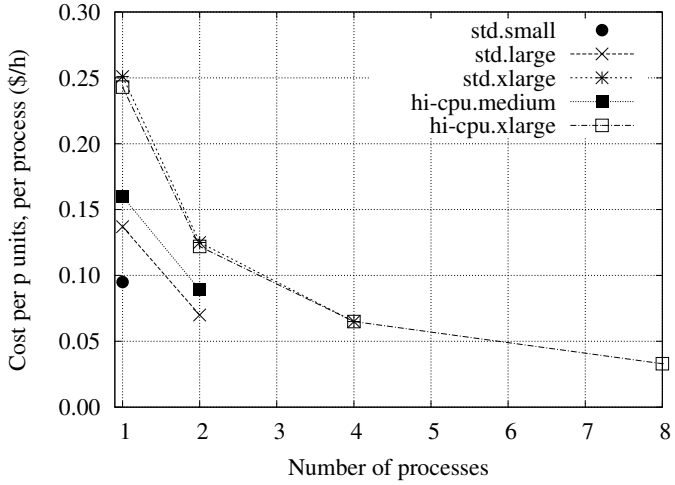

Figure 4: Cost per p units, for each process, depending on the instance type.

5.2. I/O Performance

In addition to CPU-intensive tasks, the I/O performance of the various instances has also been measured. This is important since simulations often require I/O activity, especially when dealing with uncompressed video sequences as it is generally the case with video quality simulations. Table 4 reports the performance of the I/O subsystem, as measured by the *iozone* tool [23], for all the instances considered in this work. The tool has been run with record size equal to 32 KBytes and file size equal to a value larger than the maximum amount of available memory to reduce as much as possible the influence of the operating system disk cache. For convenience, the column named "I/O classification" reports the Amazon classification of the I/O performance, where "M" means Medium and "H" means High. The performance is mostly aligned with the classification, with higher values for the *std* instance types especially for write operations compared to the *hi-cpu* instances. However, note that, as stated by Amazon [22], due to the shared nature of the I/O subsystem across multiple instances, performance is highly variable depending on the time the instance is run.

Table 4: Experimental measurements of I/O performance of the various instances, normalized values where *hi-cpu.medium*=1.00 (last row shows absolute values in KBytes/s, using record size = 32 KBytes.)

Instance type	I/O classif.	Read	Write	Random read	Random write
std.small	M	1.41	2.31	1.11	1.19
std.large	H	1.52	2.32	1.64	1.58
std.xlarge	H	1.46	2.10	1.91	1.52
hi-cpu.medium	M	1.00	1.00	1.00	1.00
hi-cpu.xlarge	H	1.47	1.29	1.50	0.92
hi-cpu.medium	M	71317	14454	4545	8511

Table 5: Parameters of some representative communication simulations.

Parameter	Typical values	*Sim1*	*Sim2*	*Sim3*
Resolution	352×288 to 1920×1080	352×288	704×576	1280×720
Pixels per frame	100K-2000K	101,376	405,504	921,600
Sequence length (frames)	180-300	300	300	300
Uncompressed video sequence (MB)	26-890	43.5	174	395.5
Input files, scripts and executables (MB)	0.5-1	0.61	0.60	0.64
Channel realizations	30-50	50	40	30
# of values for channel parameter (e.g., SNR)	3-5	5	4	4
# of values for algorithm parameter (e.g., maximum packet size)	3-5	5	4	4
# of sequences	4-5	4	4	4
Algorithm for quality measure	PSNR, SSIM, PVQM	PSNR	SSIM	PVQM
Total size of results (compressed) (MB)	200-600	595	305	228

6. Case Study: Sample Simulations

6.1. Simulation Characteristics

In order to assess the performance in realistic cases, we describe the typical requirements of some multimedia communication simulations typically used in the research activities. The most important part of the dataset in multimedia experiments are the video sequences, typically stored in uncompressed format. Generally, sequence length ranges from 6 to 10 s at 30 frames per second, i.e., 180 to 300 frames. The corresponding size in bytes range from about 26 MB up to 890 MB depending on video resolution, from CIF (352×288) to FullHD (1920×1080). Typically, four or five video sequences are generally enough to represent a range of video contents suitable to draw reliable conclusions about the presented techniques. Since results are computed as the average performance over different channel realizations, to achieve statistically significant results from 30 to 50 different random channel realizations are needed. Moreover, often a couple of parameters can be varied, e.g., one in the channel model and one in the algorithms to be tested, thus three (minimum to plot a curve) to five values have to be tested for each parameter. Once each transmission simulation has been performed, the video decoder is run, e.g., the H.264 standard test model software [24] as done in this work, thus obtaining a distorted video sequence whose size in bytes is the same as the original uncompressed video sequence size. Finally, performance can be measured using various video quality metrics, ranging from simple mean squared error (MSE) that can be immediately mapped into PSNR values [25], the most commonly used measure in literature, to much more complex algorithms that tries to account for the characteristics of the human visual system, e.g., SSIM and PVQM [26, 27]. Note that once the previous performance metrics have been computed (typically, a single floating point number for each frame of the decoded sequence), the decoded video sequence can be discarded, therefore the maximum temporary storage occupancy is limited to the size of one video sequence. Table 5 provides actual values for three representative simulations, respectively a low, moderate and high complexity simulation, that will be used as examples in the remaining part of the paper. Note that

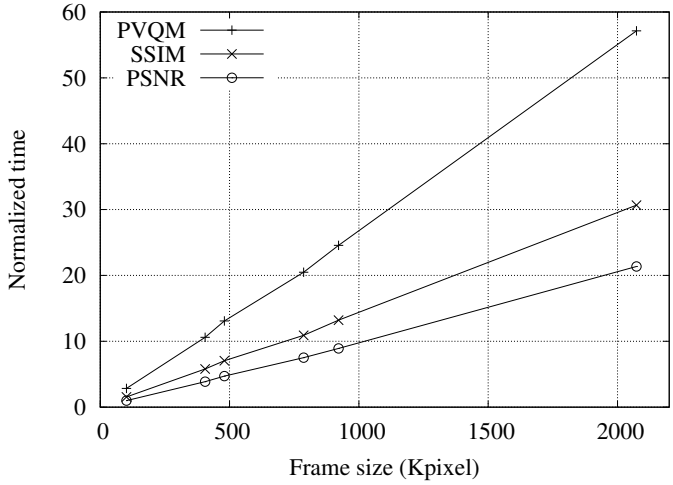

Figure 5: Relative time performance for various frame sizes and quality evaluation algorithms. Time is assumed to be equal to one for CIF frame size (about 100 Kpixel), evaluated with PSNR.

once the number of combinations of parameters has been decided, it is also possible to compute an estimate of the size of the results produced by the simulation, as shown by the last row of the table.

6.2. Simulation Complexity

The time needed to run *Sim1* sequentially on, e.g., an Intel i5 M560 processor at 2.67 GHz with 4 GB RAM is about 29,500 s, i.e., more than 8 hours. It is clear that such a long time might significantly slow down the development of transmission optimization techniques, since every time modifications of the algorithm are made, for any reason, simulations should be run again. Therefore, speeding up simulations is definitely interesting.

Using all the computing power available on the *i5* computer requires 17,250 s to run *Sim1*. By means of a CPU-intensive task such as the MD5 hash described in Section 5.1, it can be seen that the computer performance is equal to about 2.75 ECU per core, i.e., 5.5 ECU total (assuming the performance achieved by the *std.small* instance as the reference, equal to 1 ECU). Using the same symbols introduced at the beginning of Section 5, $\nu_{PC} = 2$ and $E_{PC} = 2.75$.

Thus, we conclude that *Sim1* would require about 26.38 hours on a 1 ECU processor, i.e., the computing energy S needed to run it is 26.38 ECU·h. Considering the nominal computational power stated by Amazon for the various instance types, the total cost would be 2.51 $ when using the *std* family of instances or 1.00 $ using the cheaper *hi-cpu* family of instances. The value is obtained by multiplying the computing energy S by φ_i, i.e., the cost/h/ECU shown in Table 1, which is the same for all instances belonging to the same family. Note that the cost obtained in this way is independent of the number of instances used to perform the simulation, since the code can be significantly parallelized as discussed in Section 3.

Varying the size of the video frame or the algorithm used to measure the video quality performance of the communication changes the computing energy needed for the simulation. Figure 5 shows that the amount of computations increases linearly with the frame size in pixels. Moreover,

Table 6: Computing energy (in brackets the additional energy for compression of final results), execution time and cloud costs on the cheapest family of instances depending on the simulation example.

Sim. ID	Computing energy (ECU·h)	Time on *i5* PC (h)	Cloud instance cost ($)	Download of results ($)
Sim1	26.38 (0.028)	4.80	1.00	0.06
Sim2	101.27 (0.014)	18.41	3.85	0.03
Sim3	316.94 (0.011)	57.63	12.04	0.02

Table 7: I/O performance depending on the simulation example.

Sim. ID	Upload of sequences (only first time) (s)	Upload script, exe and setup data (s)	Store in S3 (s)	Download of results (s)
Sim1	4.4	< 1	30.0	94.4
Sim2	17.4	< 1	4.4	48.3
Sim3	39.6	< 1	1.1	36.3

the computational cost of running more complex quality evaluation algorithms is approximately constant (in percentage) when compared with the PSNR algorithm. The SSIM algorithm incurs in about 50% increase with respect to PSNR, while the PVQM requires about 180% additional computation time.

Table 6 reports the complexity in terms of ECU·h of each sample simulation described in Table 5, as well as the time needed to run them on the *i5* PC using all CPU cores and the cost of running them in the cloud using the cheapest family of instance types. For a high-complexity simulation such as *Sim3*, the time required by the PC is more than two days.

6.3. Simulation Setup, I/O and Memory Requirements

As highlighted in Section 4, during the preparation phase of the AMI, sequences could be preloaded in the AMI itself since it is likely that simulations have to be repeated many times during the development of multimedia optimization techniques, as described in Section 7.2. Storing data in the Amazon cloud (e.g. in the AMI) has a very limited cost, 0.15 $ to store 1 GB for one month (fractions of size and time are charged on an hourly pro rata basis), which would be enough to hold nearly 6 sequences of the type employed in *Sim2*. Data transfer costs to the cloud are zero, while transferring from the cloud costs 0.15 $ per GB. Transfer speed is not an issue since researchers can typically use high speed university network connections. For instance, from the authors' university in Italy the typical transfer speed to and from an active instance in the AWS region in EU (Ireland) is about 11 MB/s and 6 MB/s respectively. In our tests, the most critical part has been transferring from an active instance to the S3 storage within the Amazon infrastructure, at the end of simulations when files have to be stored in S3. This could be done at an average of 3.3 MB/s from each instance. Downloading large files from S3 to a local PC in the university can be done at about 6.3 MB/s.

Table 7 shows the transfer times needed to load uncompressed sequences, transfer scripts, executable and control files needed to run the simulation, and to collect the results, including time to store data in S3 (for each instance). Note that in the provided examples, for simplicity, results are stored in text files that are then compressed and downloaded. The data size is determined by the number of combinations of parameters, which is the highest in *Sim1*, thus this implies more data to download. However, the time is quite limited (about one minute and a half in the

Table 8: Improvement of execution time using a RAM disk instead of the default Amazon storage (EBS) for the instance. *Hi-cpu.medium* instance type.

Sim. ID	Time reduction (%)
Sim1	4.0
Sim2	0.9
Sim3	0.1

worst case) if compared with the duration of the simulation, and could be further reduced by storing data in a more optimized way, e.g., using a binary format to represent numbers instead of text. The computing energy needed to compress result files has already been accounted for in Table 6. In our experiments instance activation times have always been less than about 45 seconds, with typical values around 30, therefore they are comparable or sometimes less than the time needed to download the results and much less than the time needed to run simulations at the most cost-effective tradeoff point, i.e., about one hour, as it will be determined in Section 7.

The types of computations involved in multimedia communications are typically CPU intensive. However, it is often necessary to move large amounts of data within the computer system (e.g., reading and writing large files, that is, the video sequences). The operating system disk cache almost always helps in this regard by using a large amount of memory for this purpose, thus effectively keeping most of the data in memory. An alternative approach to ensure that RAM is used to access these files could be to create, for instance, a RAM disks and keep the most critical files there, such as the video sequence currently tested. We experimented with this solution, moving all files to a RAM disk whose size is about 80% of the available memory, and measuring the values shown in Table 8. Some modest improvements can be achieved in the case of *Sim1* (4%) while they are negligible for *Sim2* and *Sim3*. An additional experiment, not shown in the table, which uses the PSNR quality measure and the video frame resolution employed in *Sim3* shows 9.2% time improvement. We attribute this behavior to the fact that when the PSNR quality measure is used, it is much more important to have fast access to the video sequence files (both the original and the decoded one for the current simulation experiment) since the PSNR only performs very simple computations. For more CPU-intensive algorithms such as SSIM and PVQM there is almost no improvement in faster access to the video files. However, even 9.2% time improvement is not sufficient to justify the use of instances with much more memory, which cost more than twice for unit of computational power.

7. Analytical Analysis

7.1. Single Simulation

In order to mathematically characterize the time and cost needed to perform a given simulation, the following notation is introduced. The amount of workload associated with the given simulation is denoted by S, as already mentioned. Let K be the number of instances used to perform the simulation, and i be the type of the instances, assuming that only one type is used to run the whole simulation. The other variables, corresponding to the characteristics and costs of each instance type, have already been defined in Section 3.

The time T_S needed to perform a given simulation that requires S computing energy using K instances of type i is given by

$$T_S(i, K) = \frac{S}{\nu_i E_i K}. \tag{4}$$

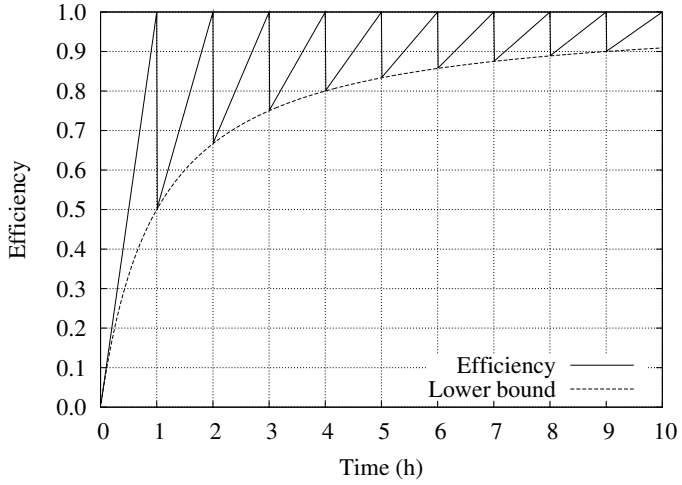

Figure 6: Efficiency as a function of the actual time used to perform the simulation.

The value is inversely proportional to the number of instances K, i.e., the computation speed can be increased as desired, provided that the simulation task can be split in a sufficiently high number of parallel processes, by just increasing the number of instances K.

However, note that Amazon charges every partial hour as one hour, therefore the exact cost of running the given simulation is obtained by rounding up the time used on each instance to the nearest integer hour. Cost is given by

$$C_S(i,K) = \nu_i E_i \varphi_i K \lceil T_S(i,K) \rceil = \nu_i E_i \varphi_i K \left\lceil \frac{S}{\nu_i E_i K} \right\rceil \tag{5}$$

where the $\lceil \cdot \rceil$ function represents the smallest integer greater than or equal to the argument.

We introduce an efficiency value η which represents the ratio of the time interval in which each instance is performing computations to the time interval the instance is paid for, that is,

$$\eta = \frac{T_S(i,K)}{\lceil T_S(i,K) \rceil}. \tag{6}$$

Eq. (6) presents the behavior shown in Figure 6. The function has periodic local maxima (value equal to 1) when T_S is an integer number of hours, and it can be lower bounded as

$$\eta > \frac{T_S}{1 + T_S} \tag{7}$$

implying that efficiency tends to 1 for T_S values much greater than 1 hour.

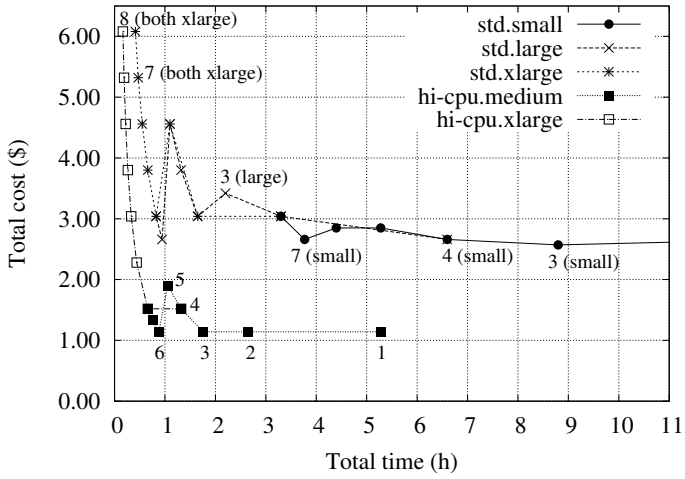

Figure 7: Total cost of *Sim1* as a function of the time that would be needed to complete all the tasks according to the nominal ECU values. Labels within the graph show the number of instances (K) corresponding to the point (with the name of the instance type in case of ambiguity.)

Eq. (5) can be rewritten as

$$C_S(i,K) = \frac{\nu_i E_i K}{S} S\varphi_i \left\lceil \frac{S}{\nu_i E_i K} \right\rceil = S\varphi_i \frac{\left\lceil \frac{S}{\nu_i E_i K} \right\rceil}{\frac{S}{\nu_i E_i K}} = S\varphi_i \frac{\lceil T_S(i,K) \rceil}{T_S(i,K)} = \frac{S\varphi_i}{\eta}. \tag{8}$$

It is clear that there is a lower bound to the cost of simulation S, that is equal to $S\varphi_i$, achieved when the efficiency is one. This happens when the simulation time is exactly a multiple of one hour. In all other cases, $\eta < 1$ and the cost is higher than the minimum value $S\varphi_i$. Efficiency tends to one when simulation time increases. Now consider a simulation time shorter than one hour: $\lceil T_S \rceil = 1$. Substituting $\lceil T_S \rceil$ in Eq. (5) and using Eq. (4), the cost for the specific case $T_S < 1$ can be written as

$$C_S = \frac{S\varphi_i}{T_S}. \tag{9}$$

The previous equation shows that in such a condition, i.e., maximum simulation speed up, due to the Amazon pricing policy on partial hours, the cost is inversely proportional to simulation time.

For the *Sim1* test case the cost that would be needed to complete all the tasks according to the nominal ECU values is shown in Figure 7. Each line represents a different instance type. Each point on the line corresponds to a different number of instances K. For high values of simulation time, curves are approximately flat since the efficiency η is high, therefore cost is almost constant. When the simulation time is decreased by increasing the number of instances K, curves tend to follow an hyperbolic trend, which is due to the trend of the lower bound on the efficiency. However, efficiency oscillates between the lower bound and one, therefore the

Table 9: Experimental measurements of computing performance of EC2 instances using as many processes as the number of available cores (*Sim1*).

Name	Nominal ECU/core	Number of cores	Effective comp. power P (std.small=1.00)
std.small	1	1	1.00
std.large	2	2	1.54
std.xlarge	2	4	1.32
hi-cpu.medium	2.5	2	2.37
hi-cpu.xlarge	2.5	8	1.36

hyperbolic trend is often interrupted. When time is lower than one hour, the trend becomes hyperbolic for all the instance types, as stated by Eq. (9). Moreover, only two hyperbolae are present, which indeed correspond to the two possible φ_i values, set by Amazon for the two considered families of instances, that is, *std* and *hi-cpu*.

Note that the previous analysis do not consider the cost of disk I/O operations on the Amazon cloud computing platform. However, in all our experiments, this has always been negligible compared to the instance costs as detailed in Section 6.3.

The previous analysis assumes that the nominal computing power, in terms of ECU stated by Amazon, allows to compute the running time of a given task on any type of instance. Actually, this is not true, as already shown in Table 3 for the case of a CPU-intensive task. Actual tests on EC2 have been performed by running, using the developed framework, a small set of each of the simulations included in *Sim1*, *Sim2* and *Sim3*. The proposed framework has been configured, each time, to use different types of instances, so that we computed the processing power that can be achieved by using every type of instance. For every run, only one type of instance was tested, i.e., we did not mix instances of different types to make performance comparison easier. Results are reported in Table 9 for *Sim1*, with reference to the performance provided by the *std.small* instance type. Note that these results only apply to the considered simulation, since they are influenced by both CPU and I/O activity. For each instance, a number of processes equal to the number of available cores has been used to maximize the exploitation of the resources of each instance. It is clear that the best performance is provided by the *hi-cpu.medium* instance type. We attribute this behavior to the low number of virtual cores, which seems to perform better in the EC2 infrastructure, and to the fact that the *hi-cpu* family of instances is probably better suited for mainly CPU-bounded tasks such as the ones of our simulations.

To get a clear overview of the actual performance that can be achieved through the Amazon infrastructure, Figure 8 shows the actual times and costs that can be achieved by using our proposed framework in order to run all the tasks included in *Sim1*, with various tradeoffs between time and cost, depending on the number of instances and the instance types. A comparison with Figure 7 shows the same general trend, but there exists some discrepancies with respect to the theoretical behavior expected by the nominal ECU values. From Figure 8 it is clear that the most convenient instance type, in terms of both time and cost, for this particular type of simulation is the *hi-cpu.medium* one. The same applies to *Sim2* and *Sim3*.

As a final remark, note that Figure 8 do not show the point corresponding to running the simulations on the dedicated computer since the price would be more than two orders of magnitude higher than the one shown for the same set of simulations in the cloud, while the time is fixed at 17,250 s, that is, nearly 5 hours. The next section includes a comparison of the time and cost, in a realistic usage case, for the development of a new transmission algorithm based on simulation

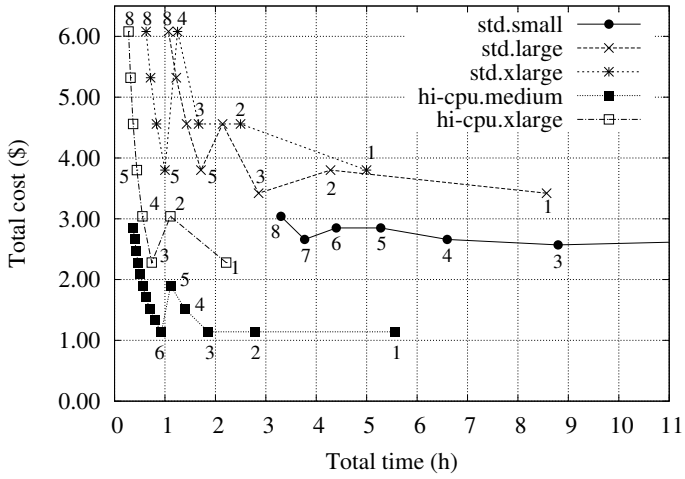

Figure 8: Total cost of *Sim1* as a function of the time needed to complete all the tasks in actual simulations. Different points on the same line correspond to different numbers of deployed instances K. Labels within the graph show the K value corresponding to the point.

results obtained by means of the dedicated computer or the cloud system.

7.2. Multiple Simulations

This section aims at quantifying the costs and speed up of the cloud computing solution with respect to the dedicated computers solution when simulations are part of an actual research activity. In such a scenario, simulation experiments such as *Sim1*, *Sim2* and *Sim3* are run many different times to investigate, improve and refine the performance of the algorithms. In order to investigate the cost and time required by using a cloud simulation system rather than a PC we assume that the research activity that leads to the development of a new algorithm is composed by a number of consecutive simulation cycles. A simulation cycle is the basic unit of the development process. One simulation cycle includes the time needed for investigation and a few modifications of the algorithm as well as the time needed to run the simulations once, i.e.,

$$T_C = T_D + T_S \tag{10}$$

where T_C is the duration of the simulation cycle, T_D is the time spent to study and modify the algorithm, and T_S is the time spent to perform the simulation, either using the cloud computing or the dedicated computer solution. Figure 9 illustrates the situation.

Moreover, in order to better model reality, we consider that operators (e.g., researchers) will work during the daytime only. Therefore, if the T_D implies that the operators' work cannot be terminated within the day, the remaining part of the work will be carried out at the beginning of the next day. When the work is terminated, a new simulation cycle can start. Clearly, if simulations extend past the end of the working day, they can continue since, of course, computers can always work at night. Figure 10 illustrates the situation. In the following, we assume a working day equal to 11 hours and T_D values ranging from one hour up to the duration of the working day.

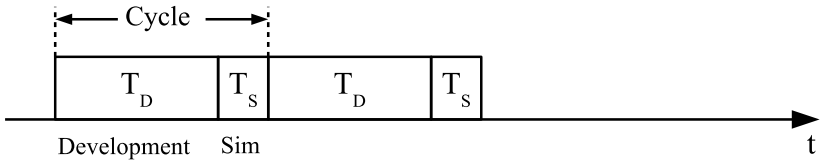

Figure 9: Diagram of the basic structure of a simulation cycle.

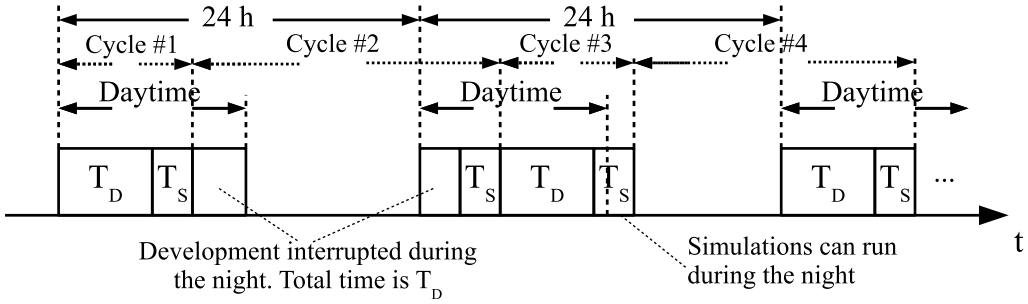

Figure 10: Diagram of the simulation cycle considering that operators do not work at night.

To better understand the implications of working during the daytime only, the total time needed to perform all the simulation cycles in the *Sim1* scenario with PC-based simulations using the *i5* computer is shown in Figure 11. When operators can work at any time, time linearly increases with T_D, as expected. The behavior is more irregular (but still monotonic as a function of T_D) when operators work during daytime only. The irregular behavior is due to the fact that the total time strongly depends on when the simulation in each cycle ends. If the ending time of the simulation does not allow the operators to prepare and start the simulation of the next cycle before the end of the working day, all the night elapses without any activity being performed, thus increasing the total time. In the remainder of the paper we will always compute the total time in the more realistic situation in which operators work during daytime only.

In the following, different values of T_D as well as different simulations (i.e., different T_S values) will be considered to identify which are the conditions that lead to the highest advantage when using the cloud solution instead of the PC-based solution. As shown in Figure 12, the total time required to perform all the simulation cycles (20 in the figure) increases almost linearly with T_D when using the cloud which correspond to $T_S = 1$ h for any simulation set, since this is the best cost-performance tradeoff point as explained in Section 7.1. However, many simulations such as, e.g., *Sim1*, *Sim2* and *Sim3* have much higher T_S when carried out using the *i5* PC. In such conditions, the T_D influence on the total time is more limited in certain intervals. The difference between the curve for a given T_S and the bottom curve corresponding to $T_S = 1$ h can be seen as a measure of convenience (in terms of speedup) of moving the simulation to the cloud. The speedup itself is shown in Figure 13. For simulations such as *Sim1* a significant speedup can be achieved especially when the T_D value is low while it reduces when T_D is close to a whole working day, that is, 11 hours. For simulations with higher T_S values, very large speedups can be achieved for low T_D values while the speedup is more limited when T_D is close to 11 hours.

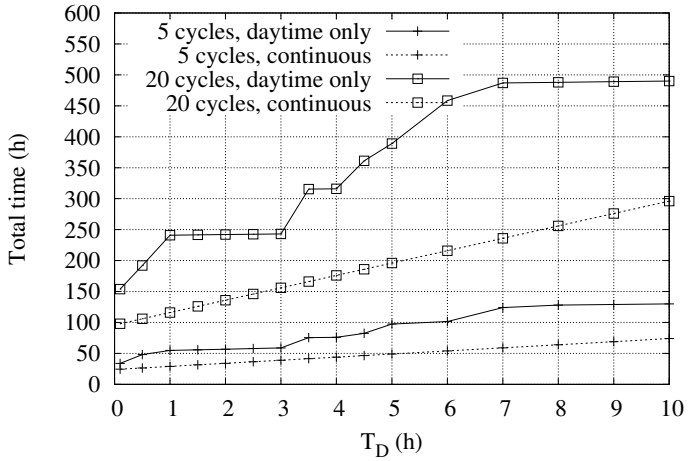

Figure 11: Comparison of the total time needed to perform all the simulation cycles when it is assumed that operators work also during the night as soon as simulation finishes, or during daytime only. *Sim1*, PC-based simulations using the *i5* computer (T_S = 4.8 h).

7.3. Cost Comparison

The best performance in terms of time is always achieved by using the cloud since the T_S value can always be reduced at approximately one hour while the cost remains the same. To perform computations at least as fast as the cloud solution does, n_{PC} PCs are needed, where

$$n_{PC} = \left\lceil \frac{S}{\nu_{PC} E_{PC}} \right\rceil. \tag{11}$$

However, the number of PCs can be reduced if the performance constraint is relaxed. Figure 14 shows that if a moderate increase of the total time is allowed, e.g., 30%, the number of PCs reduces, for instance, from 5 to 3 in for the *Sim1* case when T_D is equal to 3 hours. In general, higher T_D values lead to a faster decrease of the number of PCs, since the importance of the simulation time decrease when compared to the total time, as well as simulations can run at night thus using time that is not used by the operators. For instance, for T_D = 10 h only one PC is needed if a modest 10% total time increase is allowed, because for both the cloud and the PC solution it is necessary to wait for the next day to continue the work (1 hour simulation using the cloud has the same effect, in terms of total time, to 4.8 hours using the PC). Figure 15 shows similar results for the case of *Sim2*.

Once the number of PCs is known, it is possible to compare the cost of the cloud based solution with the amount of money that would be needed to purchase and run the PCs. In the following we assume that the lifetime (L_{PC}) of a PC is 3 years and its cost is about 1,000 \$. Other costs are difficult to quantify, such as management costs, hardware failures, renting rooms and necessity of advance planning for buying/placing the computers. However, some are easy, such as electricity costs. For instance, in Italy the price for a 100 W electricity load used continuously, 24 hours a day, is about 10 \$/month (assuming 1 € = 1.35 \$). Thus the cost C_{PC} of maintaining a PC for its

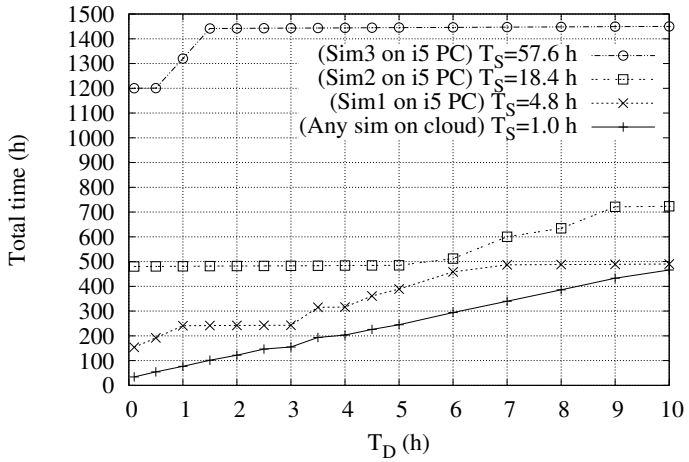

Figure 12: Comparison of the total time needed to perform 20 simulation cycles, as a function of the time T_D needed by the operators to work between simulations and the time needed to run the simulation itself T_S.

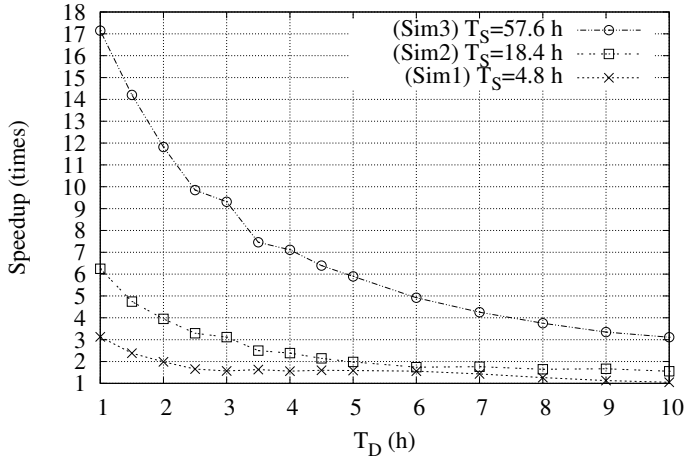

Figure 13: Speedup that can be achieved using the cloud considering different simulations, i.e., T_S values (20 cycles, cloud vs one *i5* PC).

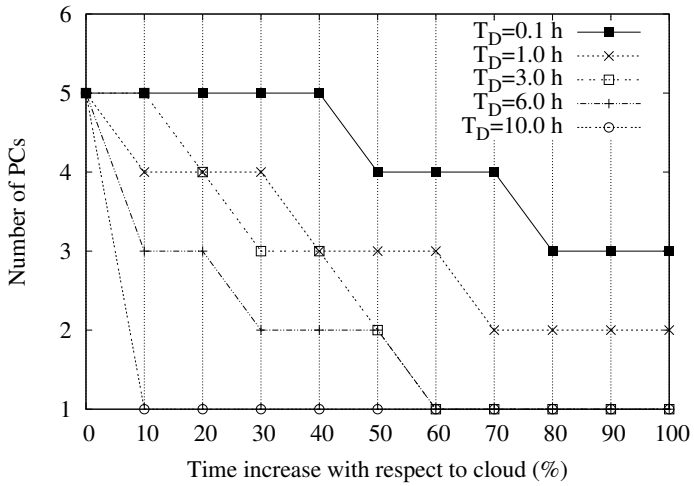

Figure 14: Number of PCs needed to achieve the same performance or performance proportional to that of the cloud solution in the *Sim1* case (20 simulation cycles).

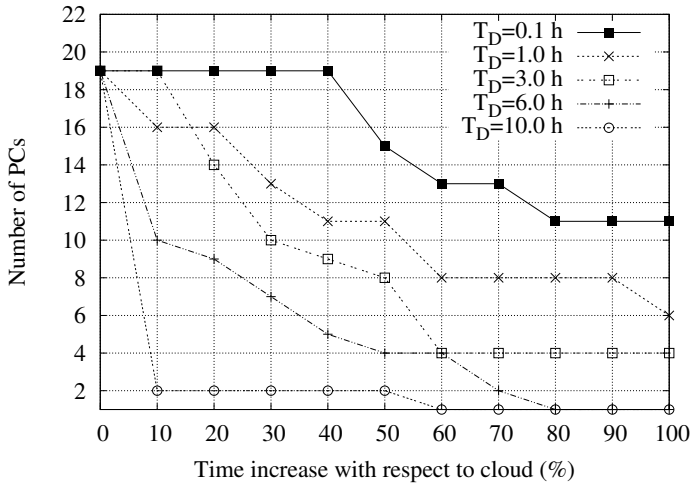

Figure 15: Number of PCs needed to achieve the same performance or performance proportional to that of the cloud solution in the *Sim2* case (20 simulation cycles).

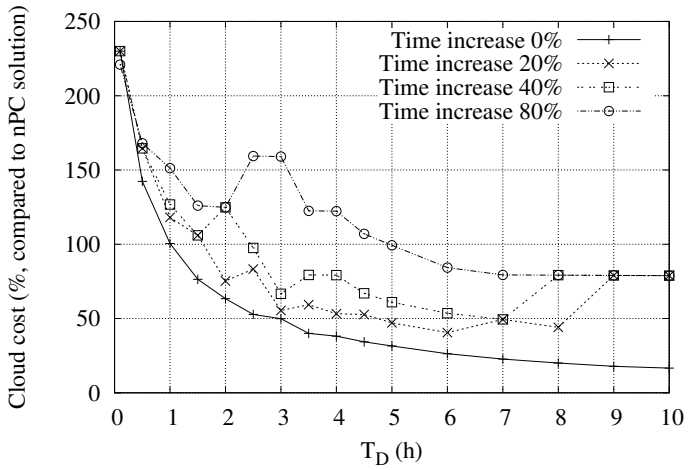

Figure 16: Cloud cost as a percentage of the cost of a same-performance (time increase 0%) or lower performance (time increase greater than 0%) PC-based solution. When the y-value is lower than 100% the cloud solution is cheaper than the PC-based one. *Sim1* case.

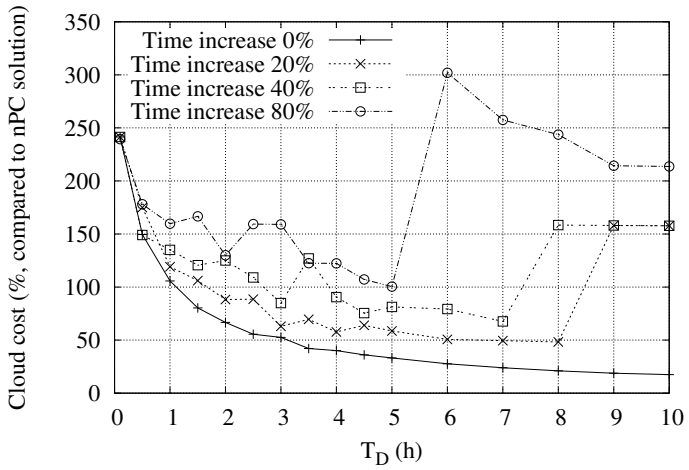

Figure 17: Cloud cost as a percentage of the cost of a same-performance (time increase 0%) or lower performance (time increase greater than 0%) PC-based solution. When the y-value is lower than 100% the cloud solution is cheaper than the PC-based one. *Sim2* case.

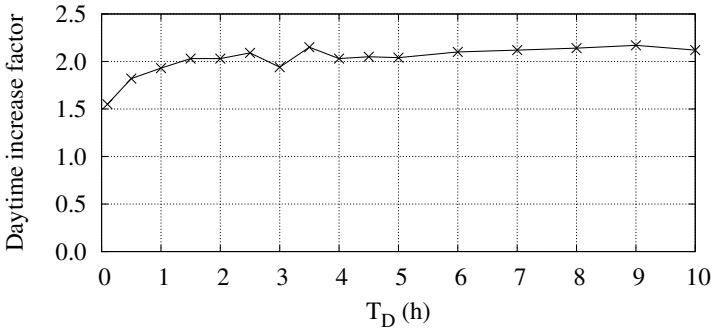

Figure 18: Time increase due to the operators working during daytime only compared to continuous working (20 cycles, cloud based simulations).

lifetime is about 1,360 $. Clearly, all the costs mentioned here, including electricity, are null if the cloud computing solution is used since they are already included in the price set by the cloud provider.

Figure 16 shows the cost of the cloud solution as a percentage of the cost of the same-performance or a lower-performance nPC-based solution (depending on the "time increase" value) for the case of *Sim1*. When the cost value is lower than 100% the cloud solution is cheaper than the nPC-based one. This always happens for T_D greater than 1 hour if the same performance of the cloud solution is sought. If some increase in the total time is allowed (up to 40% "time increase"), the cost raises but it is still well below the cost of the nPC-based solution for T_D values greater than 2.5 hours. Curves tend to increase while moving towards T_D equal to 11 hours, since in that case simulations run for most or all the time at night. In this condition, running times up to 13 hours do not affect the total time, thus the advantage of using the cloud to run computations as fast as possible decreases. Figure 17 shows the same result for the case of *Sim2*. Comparing it with the previous case, it can be seen that when a lower performance in terms of time is considered for the nPC-based solution, the cost of the cloud increases faster if T_S is higher, since higher T_D values imply that an increasing portion of the simulation can run during the night without affecting the total time.

Note that, when the time needed by the nPC-based solution equals the time needed by the cloud solution, the cost curve is the the same regardless of the complexity of the considered simulation. In this condition, the time taken by simulations in both the cloud and the nPC-based solution is 1 hour. Since we consider that operators work during daytime only, the actual time taken by N_{cy} simulation cycles is $N_{cy}(1+T_D)f(T_D)$ where $f(T_D)$ is a factor that takes into account the total time increase due to working during daytime only. For T_D values ranging from 1 to 10 hours it can be approximated as 2, as shown by Figure 18.

However, the cloud is paid only for the actual usage, therefore the respective costs of each solution are:

$$C_{cloud} = \frac{L_{PC}}{T_S} S \varphi_i N_{cy} = \frac{L_{PC}}{N_{cy}(1+T_D)f(T_D)} S \varphi_i N_{cy} = \frac{L_{PC}\varphi_i S}{(1+T_D)f(T_D)} \tag{12}$$

and

$$C_{n_{PC}} = C_{PC} \frac{S}{\nu_{PC} E_{PC}} \tag{13}$$

which is a lower bound since the number of PCs should be an integer value. The upper bound on the cloud / nPC cost ratio is:

$$C_{ratio} = \frac{L_{PC}\varphi_i \nu_{PC} E_{PC}}{C_{PC}} \frac{1}{(1+T_D) f(T_D)}. \tag{14}$$

which matches the behavior shown in the figures. The first fraction does not depend on T_D and it is equal to 4.0386. The condition yielding an economic advantage for the cloud solution is:

$$T_D > \frac{L_{PC}\varphi_i \nu_{PC} E_{PC}}{C_{PC}} \frac{1}{f(T_D)} - 1. \tag{15}$$

Assuming the f factor equal to 2, the cloud solution is cheaper if T_D is greater than approximately 1 hour, and the economic advantage increases with greater T_D values.

8. Conclusions and Future Work

This work focused on investigating the cost performance tradeoff of a cloud computing approach to run simulations frequently encountered during the research and development phase of multimedia communication techniques, characterized by several develop-simulate-reconfigure cycles. The traditional approach to speed up this type of relatively small simulations, i.e., running them in parallel on several computers and leaving them idle when developing for the next simulation cycle, has been compared to a cloud computing solution where resources are obtained on demand and paid only for their actual usage. The comparison has been performed from both an analytical and a practical point of view, with reference to an actual test case, i.e., video communications over a packet lossy network. Performance limits have been established in terms of running time and costs. Moreover, a cloud simulation software framework has been presented to simplify the virtual machines management in the cloud. Actual performance measurements and costs have been reported for a few sample simulations. Results showed that currently, with reference to the commercial offer of Amazon cloud computing services, it is economically convenient to use a cloud computing approach, especially in terms of reduced development time and costs, with respect to a solution using dedicated computers, when the development time is higher than one hour. However, if a high development time is needed between simulations, the economic advantage progressively reduces as the computational complexity of the simulation increases.

Future work includes improving the framework in many aspects. For instance, data could be directly downloaded eliminating the need to store them in the S3 system. Data distribution inside the cloud could be improved so that, for example, some instances work on a subset of data (i.e., only some sequences) reducing the necessity of using the I/O subsystem for re-reading sequences. Associating weights to each task included in the simulation could allow to take into account the different execution time when allocating tasks to the various instances, which would be particularly useful to run simulations in which the sequences have mixed frame sizes and lengths.

Acknowledgments

The authors are grateful to the anonymous reviewers for their insightful comments, that particularly helped in improving the cost analysis part of this work.

References

[1] I. Raicu, I.T. Foster, Y. Zhao, Many-task computing for grids and supercomputers, in: Proc. of Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS), Austin, TX, 2008.

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: A Berkeley view of cloud computing, Technical Report UCB/EECS-2009-28, University of California, Berkeley (Feb. 2009).

[3] J. Yu, R. Buyya, A taxonomy of scientific workflow systems for grid computing, ACM SIGMOD Record 34 (3).

[4] I. Foster, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: Grid Computing Environments Workshop (GCE), Austin, TX, 2008.

[5] X. Liu, A. Datta, Towards intelligent data placement for scientific workflows in collaborative cloud environment, in: IEEE Parallel and Distributed Processing Symposium, Anchorage, AK, 2011, pp. 1052–1061.

[6] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, B. Berriman, Experiences using cloud computing for a scientific workflow application, in: Proc. of the 2nd intl. workshop on Scientific Cloud Computing (ScienceCloud), San Jose, CA, 2011.

[7] M.R. Palankar, A. Iamnitchi, M. Ripeanu, S. Garfinkel, Amazon S3 for science grids: a viable solution?, in: Proc. Intl. Workshop on Data-aware distributed computing (DADC), Boston, MA, 2008, pp. 55–64.

[8] Amazon Web Services, URL: https://aws.amazon.com, last accessed: Nov 24, 2011.

[9] E. Walker, Benchmarking Amazon EC2 for high performance scientific computing, ;login: The USENIX Magazine 33 (5).

[10] C. Evangelinos, C.N. Hill, Cloud computing for parallel scientific HPC applications: Feasibility of running coupled atmosphere-ocean climate models on Amazon's EC2, in: Proceedings of Cloud Computing and Its Applications, 2008.

[11] A. Iosup, S. Ostermann, M. Nezih Yigitbasi, R. Prodan, T. Fahringer, D.H.J. Epema, Performance analysis of cloud computing services for many-tasks scientific computing, IEEE Transactions on Parallel and Distributed Systems 22 (6) (2011) 931–945.

[12] C. Z. Loboz, Cloud resource usage: extreme distributions invalidating traditional capacity planning models, in: Proc. of the 2nd intl. workshop on Scientific Cloud Computing (ScienceCloud), San Jose, CA, 2011.

[13] F.A.B. da Silva, H. Senger, Scalability limits of Bag-of-Tasks applications running on hierarchical platforms, J. Parallel Distrib. Comput. 71 (6) (2011) 788–801.

[14] I. Raicu, I. Foster, Y. Zhao, A. Szalay, P. Little, C.M. Moretti, A. Chaudhary, D. Thain, Towards Data Intensive Many-Task Computing, IGI Global, Hershey, PA, 2012, Ch. 2.

[15] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde, Falkon: a Fast and Light-weight tasK executiON framework, in: Proc. of ACM/IEEE Conference on Supercomputing, Reno, NV, 2007.

[16] M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinoza, M. Hategan, B. Clifford, I. Raicu, Parallel scripting for applications at the petascale and beyond, IEEE Transactions on Computers 42 (11) (2009) 50–60.

[17] Y. Simmhan, L. Ramakrishnan, Comparison of resource platform selection approaches for scientific workflows, in: 19th ACM Intl. Symp. on High Performance Distributed Computing (HPDC), Chicago, IL, 2010, pp. 445–450.

[18] K.R. Jackson, L. Ramakrishnan, K.J. Runge, R.C. Thomas, Seeking supernovae in the clouds: a performance study, in: 19th ACM Intl. Symp. on High Performance Distributed Computing (HPDC), Chicago, IL, 2010, pp. 421–429.

[19] E. Masala, Moving multimedia simulations into the cloud: a cost-effective solution, in: IEEE Intl. Symp. on Parallel and Distributed Processing with Applications, to appear, 2012.

[20] E. Masala, J.C. De Martin, Analysis-by-synthesis distortion computation for rate-distortion optimized multimedia streaming, in: Proc. IEEE Int. Conf. on Multimedia & Expo, Vol. 3, Baltimore, MD, 2003, pp. 345–348.

[21] J. Chakareski, J. G. Apostolopoulos, S. Wee, W.-T. Tan, B. Girod, Rate-distortion hint tracks for adaptive video streaming, IEEE Transactions on Circuits and Systems for Video Technology 15 (10) (2005) 1257–1269.

[22] Amazon Web Services, Amazon EC2 instance types, URL: https://aws.amazon.com/ec2/instance-types, last accessed: Nov 24, 2011.

[23] IOzone filesystem benchmark, URL: http://www.iozone.org, last accessed: Nov 24, 2011.

[24] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Joint Model Number 11.0 (JM-11.0), URL: http://iphome.hhi.de/suehring/tml/download, last accessed: Nov 24, 2011.

[25] S. Yao, W. Lin, S. Rahardja, X. Lin, E.P. Ong, Z.K. Lu, X.K. Yang, Perceived visual quality metric based on error spread and contrast, in: Proc. of IEEE Intl. Symp. on Circuits and Systems (ISCAS), Vol. 4, 2005, pp. 3793–3796.

[26] A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (4) (2004) 600–612.

[27] A. P. Hekstra, J. G. Beerends, D. Ledermann, F. E. De Caluwe, S. Kohler, R. H. Koenen, S. Rihs, M. Ehrsam, D. Schlauss, PVQM—a perceptual video quality measure, Signal Process. Image Commun. 17 (10) (2002) 781–798.