Multivariate Analysis of Variance for High Energy Physics Software in Virtualized Environments

Víctor Fernández Albor[1], Marcos Seco[1], Víctor Méndez Muñoz[2], Tomás Fernández Pena[3], Juán Saborido Silva[1] and Ricardo Graciani Diaz[4]

[1] Physics department, Santiago de Compostela University
Av Ciencias sn, Santiago de Compostela, Spain
E-mail: {victormanuel.fernandez,marcos.seco,juan.saborido}@usc.es
[2] Computer Architecture and Operating Systems (CAOS),Universidad Autónoma de Barcelona
E-mail: victor.mendez@uab.es
[3] Research Center in Information Technologies (CiTIUS), Santiago de Compostela University
Av Ciencias sn, Santiago de Compostela, Spain
E-mail: tf.pena@usc.es
[4] Departamento de Estructura y Constituyentes de la Materia,Barcelona University
E-mail: Ricardo.graciani@ecm.ub.es

Cloud Computing is emerging today as the new approach followed by computing centres, since the flexibility the Cloud provides is a powerful component to manage their resources. Through the use of virtualization, cloud promise to address with the same shared set of physical resources a large user base with different needs. However, virtualization may induce significant performance penalties for the demanding scientific computing workloads. This work presents an evaluation of the usefulness of the current cloud computing services for scientific applications. The performance of a sample of High Energy Physics (HEP) software running in a Kernel-based Virtual Machine (KVM) under different set-ups is analyzed with the use of a multivariate analysis of variance (MANOVA). While clouds are still changing, our results indicate that the current cloud services have to take into account the different setups of Cores and Memory in order to get reasonable performances in HEP-Software.

The International Symposium on Grids and Clouds (ISGC) 2015
March 15-20, 2015
Academia Sinica, Taipei, Taiwan

1. Introduction

Scientific computing requires an ever-increasing number of resources to deliver results for growing problem sizes in a reasonable time frame. Cloud computing proposes an alternative in which resources are no longer hosted by the researchers' computational facilities, but leased from big data centers only when needed. The integration of third-party infrastructures on pay-per-use basis can provide flexibility to the computing model. This feature introduces the hybrid concept, because it could be composed by community and/or commercial infrastructures. The adoption of such a federated hybrid cloud can provide some benefits to the eScience. The first one is the use of a virtualized running environment. However, the virtualization could be an issue if the performance of the applications is being decreased excessively.

There is no consensus on how to define the Cloud [1] in the world of distributed computing, one possible definition is that of Foster et al. [2], *"A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet."*

The cloud computing paradigm holds good promise for the performance hungry scientific community [3]. Clouds are a cheaper alternative to supercomputers and specialized clusters, a much more reliable platform than grids, and a much more scalable platform than the largest of commodity clusters or resource pools. However, clouds also raise important challenges in many areas connected to scientific computing, including performance, which is the focus of this work.

An important question arises: *is the virtualized performance of clouds sufficient for HEP software and which are the suitable set-ups in that case?* Albeit early attempts to characterize clouds and other virtualized services [4, 5], this question remains largely unexplored. This paper tries to analyze statistically the performances in virtualized platforms, of different set-ups and several types of software used in the HEP field.

Three computer programs have been selected to make the test: (i) GAUSS which is an specific software developed for the Large Hadron Collider beauty experiment (LHCb); (ii) AIRES developed for Auger, an experiment to study cosmic rays; and (iii) a program developed by one of the groups at the Instituto Galego de Física de Altas Enerxías (IGFAE), which are belonging to A Large Ion Collider Experiment experiment (ALICE) and makes use of FastJet. FastJet is software that contains jet finding algorithms used by major experiments like A Toroidal Large Hadron Collider Apparatus experiment (ATLAS), Compact Muon Solenoid experiment (CMS), DØ, CDF, and others. In the following section, more details of these computer programs will be given.

The paper is organized as follows: the related work is summarized in Section 2. Section 3 includes the description of the HEP software used. Section 4 describes the method and the experimental set-up. Section 5 describes the test-bed. Section 6 gives test results. The paper finishes with a conclusion section.

2. Related Work

It is considered an established fact that the largest computing centers are migrating to the Cloud because they consider its benefits [6]. Their resources are being moved from dedicated hosts to virtualized machines, more flexible and reliable than traditional systems. EGI Federated Cloud [9] is currently using an infrastructure since 2013. EGI Federated Cloud, driven by use cases originated in the development and execution of scientific applications, is not developing any software, just deploying, testing and integrating third party solutions. The task force has identified federated cloud services to be integrated and deployed [11], and one of these services will be the Virtual Machine (VM) sharing and Virtual Networking. Nowadays, those services have been deployed in a production level [10]. Most of those services use well-known technologies from the Grid computing, to support the particularities of the scientific computing in the context of a distributed infrastructure.

There are important projects such as VENUS-C [12], which is providing a development stack as PaaS focused on some Life Sciences use cases and it is based on Windows Azure [13]. The VENUS-C provides cloud solutions for individual and small research groups. Other related projects are combining Grid and Cloud technologies, like StratusLab middleware [14, 15], the architecture described in [16], integrating Grid and Cloud on the next generation network, or DIRAC integration of Grid, Cloud and local clusters in [17, 22]. These complementary services are mainly related to the management of a multiple provider IaaS in a scientific community environment, in the way of the identified by EGI Federated Cloud, mentioned above, or similar requirements like the WLCG [21].

The majority of Cloud projects are using for virtualization one of both of the two main open-source hypervisors Kernel based Virtual Machine (KVM hereinafter) [18] and Xen [19]. In addition, most of virtualized SITES are using KVM with their cloud managers configuration. In the case of KVM, it is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM also requires a modified QEMU although work is underway to get the required changes upstream [20].

There have been studies that characterized the effectiveness of cloud technologies and virtualization over competing technologies. In this vein L. Youseff et al. [4] did a study of the performance of paravirtualized systems based on Xen with benchmarks from HPC Challenge, LLNL ASCI and NAS parallel suits. An application for simulation of oceanographic and climatological phenomena was also used as a benchmark. For most of the test they found no overhead when using a virtualization environment. Meanwhile, E. Deelman et al. [5] did a study of the costs of using a commercial cloud services versus the cost of having a dedicated infrastructure. They estimated the cost by running an astrophysics application to compute mosaics of input images, obtaining the amount of CPU used by the runs, the size of the input and output files and multiplying these numbers by the costs of a CPU-hour, GB of storage, input transfer and output transfer. Their results show that commercial clouds could be an interesting alternative for certain class of scientific jobs. Finally, A. Chierci et al. [7] did a qualitative study performance of XEN versus KVM by changing a computing element and a monitoring machine from XEN, the technology used for all the machines LHCb tier2 they are hosting at CNAF, by two KVM machines. Although the users were not informed, they did not notice the change in the more than 3 weeks that the test was run. A second part of their study consisted on comparing two XEN, paravirtualized and a hardware VM, and one KVM virtual machines. The HEP-Spec06 [8] test, based on a subset of the SPEC benchmark is a standard benchmark suite in HEP community used to measure CPU performance, iperf and bonnie++ tests were used to do the quantitative comparisons. The intent of this article is obtain a quantitative evaluation of the overhead of using a KVM VM to run HEP software applications instead of a bare metal machine which has not been addressed by these previous works.

3. Hep-Software description

The software used in HEP experiments can be classified in three categories: simulation, event reconstruction and physics analysis. Simulation software is used in all stages of an experiment, from the designing to the commissioned phases. In the designing phase, it is used to obtain some estimates needed for making detector designing choices, also in the development of the reconstruction and analysis programs and, finally, helps to evaluate the physics possibilities of the detector. Once the detector is commissioned, these programs are used to look for unexpected effects and backgrounds or, when a theory is not well known, to compare the results with various models. They are also used to make studies of the detector performance and to provide efficiency values for the data analysis. The second group is the software that reconstructs the trajectories followed by the different particles produced in a collision. This is not a trivial task because, even in the case that there were just one initial collision, the high energies involved imply a high numbers of particles and, therefore, a high number of simultaneous active sensors that have to be matched to the different particle tracks. The third group is the software that performs the physical analysis, this is the most heterogeneous group of the three since apart from the software used to select the interesting events for a given study it includes customized software developed to do the actual physical analysis of the reactions under study.

The tests ran for this paper belongs to the first and third categories. Thereby GAUSS and AIRES belong to the simulation class software whereas FastJet software belongs to the physical analysis category.

The AIRES package [24] was developed for the Auger experiment with the objective of simulating the production of particle showers created by the collision of ultra high energy particles present in cosmic rays, some of them with energies well over a million times the highest energies attainable at LHC, with the Earth's atmosphere. Because this code generates most the needed data via algorithms, the IO load is low compared with the CPU load. This IO load is related to the input of the simulation parameters, the output of the results, logs and some summary files, and the update of the internal dump file (IDF), which contains the necessary internal data to restart a broken simulation from the last update of this file.

The second program used for the tests is GAUSS [28] which is a Montecarlo simulation program developed by the LHCb collaboration. The program uses the GAUDI framework, also developed by LHCb that simplifies its interface with other programs. GAUSS make use of programs like Pythia [31], EvtGen [32], etc. The simulation is done by GEANT4 [29] a program developed for the simulation of the passage of particles through matter and which is currently used in HEP experiments, astroparticle physics, nuclear physics, space and medical applications, to name a few fields. These simulations are the most CPU consuming events in HEP experiments. For example, a typical GAUSS simulation takes 50000 SPECInt2000 sec/event while reconstruction takes of the order of 20 times less [30]. Just as with AIRES, the IO load of GAUSS comes, mainly, from the reading of the configuration files, and the output of the results, logs and summary files. Unlike AIRES, that does not need to be supplied with information about the distribution of matter in the volume of simulation because it uses a model of the atmosphere that generates this information, GAUSS needs to read this information because the complex distribution of matter inside the detector cannot be modeled by a simple algorithm. Nevertheless, this information needs to be read just once, independently of the number of collisions that a given GAUSS run will simulate. Another difference with AIRES is that GAUSS does not need checkpoints because it only takes about a couple of minutes to generate a single event.

The last program used [25, 26] corresponds to the analysis category. It was developed by the phenomenology group of the Instituto Galego de Física de Altas Enerxías (IGFAE) to study the contamination of jets by background particles in heavy ion collisions. Jets are groups of particles, produced in the disintegration of high-energy particles, whose trajectories lie within a small angle cone. In [26], two programs were considered, but we have just used the one based in the FastJet package. This package [27], via a plug-in architecture, has access to a plethora of algorithms ranging from those taking $O(N^3)$ steps, where N is the number of particles to analyze, to those taking just $O(N \log N)$. In this case the IO load is also small compared to the CPU load of the job despite the fact that the amount of input data that has to be read is considerably higher than in the previous cases.

4. Method and Experimental Setup

This section deals with the experimental design that is in general, the process of defining a method to gather and analyze data to study variable phenomena that can be under the full control of the experimenter or not. This method establishes a series of steps that will increase the possibility of obtaining meaningful information from an experiment. We will briefly describe these steps. The first step is the statement of the problem, that is, what is the question the experimenter wants to be answered. The second is the subject matter model or what are the factors that will affect the outcome of the experiment. The third step has three parts: treatment design (deciding which combinations of the values of the identified factors the experimenter is going to explore), error-control (how the treatments are assigned to the experimental units to minimize errors) and, the last part, sampling and observation (design how the data is taken). These steps define the experiment and the data to be taken. To analyze the data we first have to model the response, fourth step, in terms of the treatment and errorcontrol factors. The fifth step is to choose the variable the experimenter is going to measure, or response, this choice is not always obvious but the variable should be chosen in a way that inferences and results from the experiment can be clearly stated an communicated. The last step is the principles of analysis or the application of statistical analysis to the data obtained in concordance with the experimental design and its associated model. Experimental design is, arguably, the most important part of an experiment because a faulty design could lead to the failure of said experiment.

As mentioned above, at the treatment design phase it is decided how many factors should be used and how many levels per each factor. Often the treatments are chosen to have a factorial structure. In this case, we have several factors, each with a set of qualitative or quantitative levels. Suppose our factors denoted by F_i with $I = 1,...,N$ being N the number of factors, and their respective levels by f_{ij}, where $j= 1,...,\#levels(F_i)$.

A factorial treatment is then a set of combinations of levels, one per factor, which are denoted by $(f_{1i1}\, f_{2i2}.\ \dots\ f_{NiN})$. This type of treatment can applied to any error-control design and it answers questions about the effects of single factors or the interference of several of the factors. Depending on whether all the factors have the same number of levels or not we distinguish between symmetrical or pure and asymmetrical or mixed factorial structures.

5. HEP-Software Test-bed

Following the steps outlined in the last section we can describe our experiment as follows:

(i) Statement of the problem: our main objective is to study the performance of HEP scientific software on virtualized and non-virtualized environments. As aside questions we also want to find the behavior of the applications in terms of the amount of memory/CPU resources available or the length of time the job is running.

(ii) Subject matter: Considering the posed questions some of the factors to consider are obvious: Virtualization, amount of memory, number of CPU cores, or length of the job. We also consider another factor, which we will name Benchmark that will consider the type of software being executed.

(iii) Three aspects of design:

a. Treatment design: we are going to use a mixed factorial design of the type $2^3 \times 3^2$. The 2 level factors are virtualization whose values are baremetal (BM) and KVM, memory 2 and 4GB and job length for which we will consider short and long jobs, the number of events will define the job length. The 3 level factors are the number of cores either 1, 2 or 3 and the benchmark.

b. Error-control design: each combination is going to run four times with different random seeds to provide a reasonable amount of randomization.

c. Sampling and data taking: the experiment will be run in a way that we maximize the use of the CPU resources. For example, for the configuration with two cores, there should be always two jobs running. In this case the number of jobs per combination ran will be 8: 4 repetitions times 2 jobs running simultaneously.

(iv) Modeling the response: we have four treatment factors: virtualization, memory, number of cores and length of the job; and one intrinsic factor: the benchmark used.

(v) Response choice: the variable we are going to measure is the walltime of the job.

(vi) Principles of analysis: to perform the statistical analysis we are going to use the multivariate analysis of variance or MANOVA for short. We are using statistical analysis because we want to know the behaviour of different types high energy physics applications and if there are some patterns which could share each other using different setups which could affect applications in some way that could not be just explained with the comparison of the walltimes, and after have observed that virtualization overhead could affect to some scientific applications [34], and some other factors as will be explained in section 5.1.

The hardware setup used to run this tests consisted on two DELL Poweredge 1950 machines both with two Intel Xeon 5160 CPU's and 16GB RAM. These processors have two cores each and run at 3GHz. The operating system where the tests were run was Scientific Linux 6.3 with a gcc version 4.4.6. The test software was served via a CVMFS [23] client, version 2.0.19-1.el6. Finally, with respect to the virtualization host, the KVM framework version 0.12.3+noroms-0ubuntu9.21 with a libvirt version 0.7.5-5ubuntu27.23 was used with Ubuntu 10.04 as the host operating system. It was intended to run the test in an environment as realistic as possible. Therefore, the machines were keep fully loaded during the test, running as many concurrent jobs as cores were available. The number of cores was controlled through the */sys/devices/system/cpu/cpu#/online* file, where # is the processor number in the system. The amount of memory accessible to the jobs was controlled via the cgroups service. A Python script was in charge of running the test, set-up the amount of memory at the job's disposal and, in the case of non-virtualized environment, configure the number of active cores.

5.1 HEP-Software Hardware Architecture Performance Problems

As was mentioned in the previous subsection, our test bed was formed by machines with Intel processors. Nevertheless, those same tests were also run in AMD machines, and they were discarded due to performance problemes. In particular, two Supermicro AS 2022TG-HTRF equipped with two eight core *AMD 6128 @2GHz CPUs* and 16GB for the bare metal tests and 20GB for the virtualized ones were used. A performance loss of around 5% was observed for FastJet and AIRES and for both, Intel and AMD architectures when comparing the performances in virtualized and non-virtualized environments. In contrast the observed performance loss for GAUSS test was of 20-30% for AMD but of the order 5% for Intel. With the help of perf, a profiling program for Linux, we traced the performance loss to a specific function in GEANT4 software. We suspect that these differences were, in part, due to the fact that, whilst the version of GAUSS we used was precompiled by LHCb Computing group with some optimization flags, FastJet and AIRES were compiled by us with no special optimization flags. The difference could be pin on KVM, since the versions shipped with Ubuntu 10.04 do not support all AMD CPU characteristics notably 3dnow and 3dnowext.

Because of incompatibilities with systems libraries, it was not possible to use a version of KVM more up to date. This same problem appears on versions of other distributions like Scientific Linux, SUSE, etc. To confirm this last point the host operating system was updated to Ubuntu 12.04 which allowed the installation of a more recent version of KVM. The new version, along with NUMA, admits more AMD CPU characteristics, although still not all of them.

6. Test Results

In this section the results of MANOVA analysis of the tests are presented. Around 800 simulations were executed with 28 different setups and hardware platforms. The KVM hypervisors were launched by hand with the different setups, and the data was extracted with a specific format. In the case of non virtualized nodes, all setups were run with the same Python executable, which changed the different configuration of memory and cores, with a high isolation level.

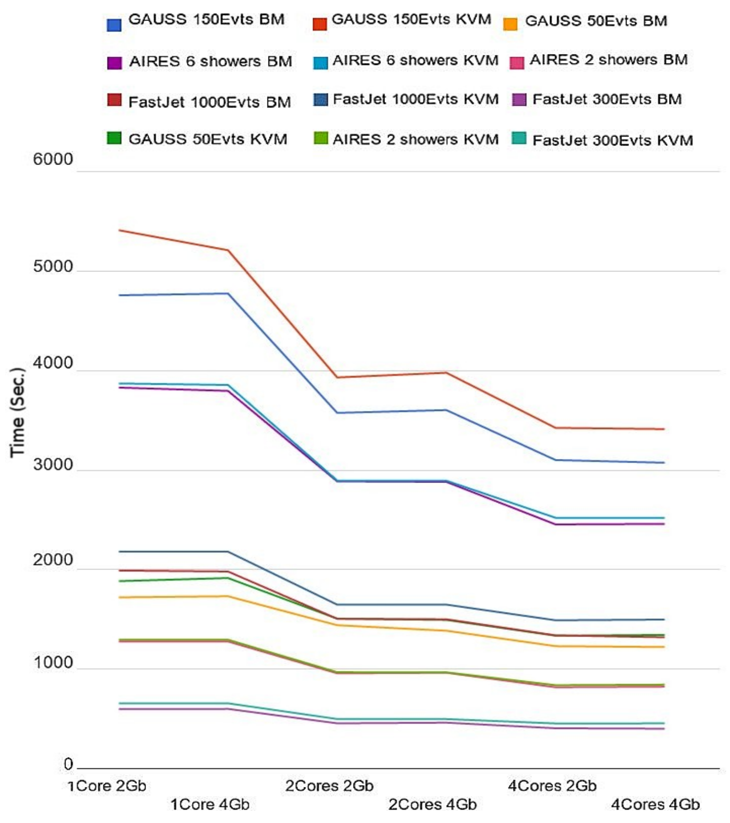


Figure 1: Walltime mean values of all HEP-Software in KVM and BM nodes

6.1 Trend of Means and Median Setups

Once the jobs were executed in all the different setups, the resulting walltimes were selected from the data depending on the platforms where they were executed. Fig. 1 shows the means of the walltimes of the different setups with virtualized (KVM) and non-virtualized or bare metal nodes. Two types of jobs are shown, large and short, with different configurations depending on the HEP-software.

The trend shows that the job walltime decreases when the number of cores increases. However, the amount of memory is not an important factor to run this type of jobs.

Fig. 2 shows the walltime medians of each experiment. In the GAUSS medians, the interquartile ranges are bigger than for the Alice and Auger software, and the distance of medians between large and short jobs in FastJet and AIRES are shorter than the GAUSS, which shows a worse performance in GAUSS jobs with setups of 1 Core and 2-4Gb. Fig. 2 also shows that in the case of GAUSS large jobs, the lower and upper quartile are more separated than in the case of short jobs, and it gives clear information about this type of jobs, which have variable performances depending on the number of events. In this sense, the setups of 2Cores-2Gb and 1Core-4Gb seem to have more stable performances.

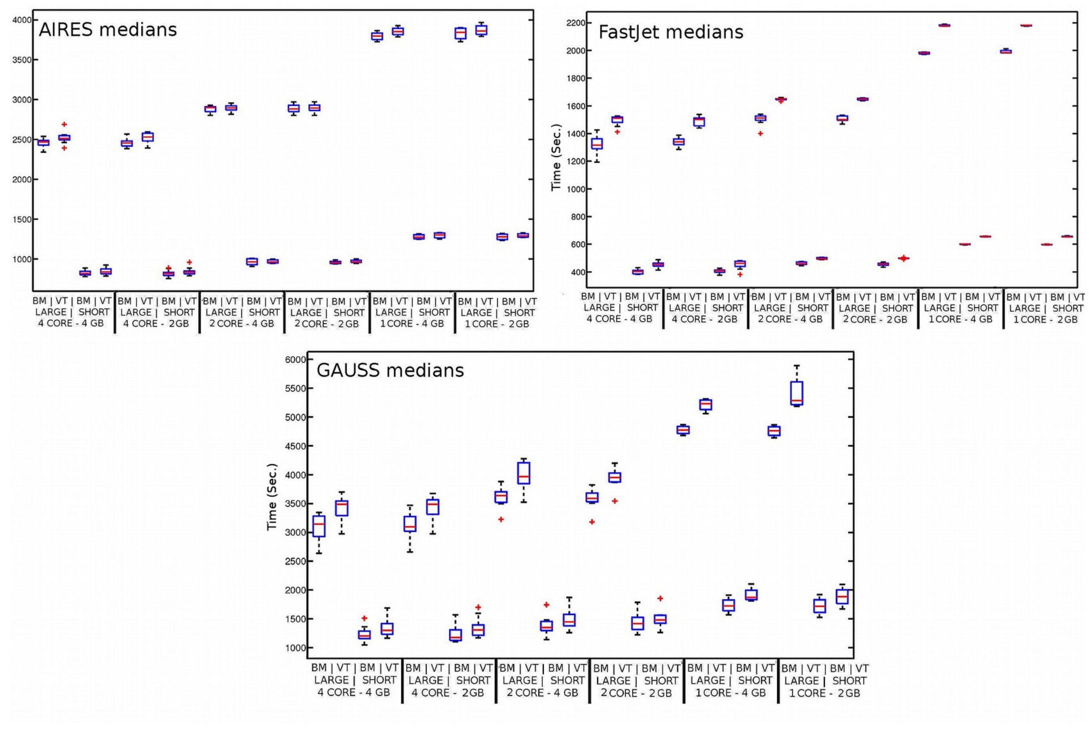


Figure 2: Medians of HEP-Software by experiment

6.2 BM and KVM Performance Analysis

This subsection shows the results of Multivariate Analysis of Variance. The statistical design is a MANOVA 1-way, in which the dependent variables are the different HEP Software with several setups of memory and cores, and with large and short jobs.

The survey of separate analysis of variance is shown on Table 1. In this case, the null hypothesis (*H0*) is the walltime of the executed jobs. All the dependent variables have statistic results that show a normal distribution, and they are significant with $P < 0.01$ and $F > 1$. The independent variable is the job walltime. The MANOVA with all the setups without means is shown in Fig. 3.

	GAUSS BM	GAUSS KVM
Snedecor F	157.9286	171.4093
Probability P	0.0000	0.0000
	AIRES BM	AIRES KVM
Snedecor F	2082.0955	1559.9788
Probability P	0.0000	0.0000
	FastJet BM	FastJet KVM
Snedecor F	1233.9208	2925.2098
Probability P	0.0000	0.0000

Table 1: Separate Analysis of variance survey

The range of variables depending of discriminations setups are: FastJet KVM, FastJet BM, AIRES KVM, AIRES.

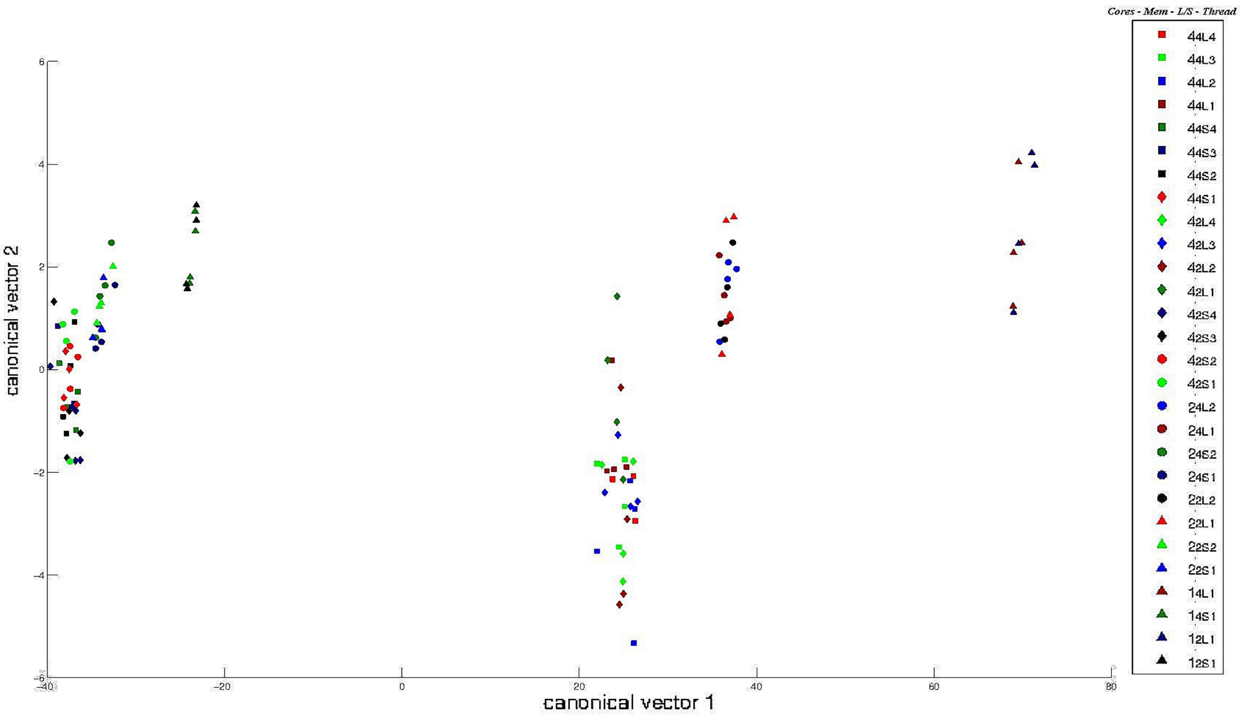


Figure 3: MANOVA centroids. Legend indicates the number of cores, amount memory, large or short jobs and the thread number.

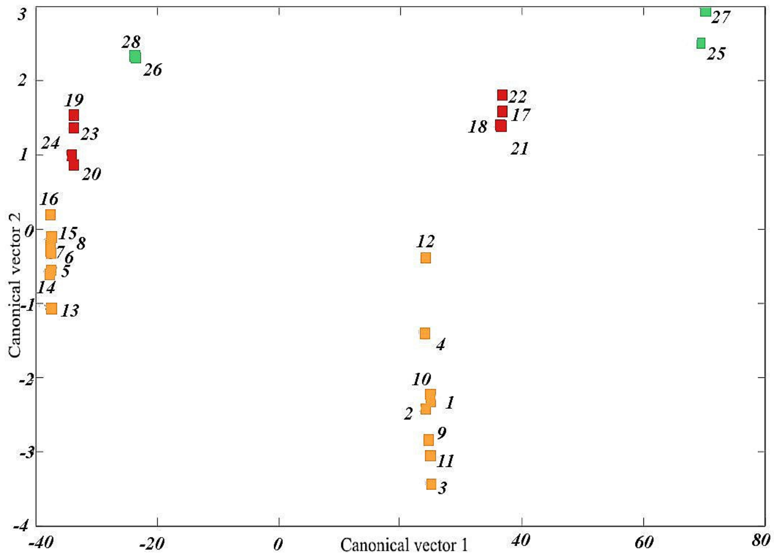
Table 2 is a MANOVA survey, to consider the possible correlations between the six dependent variables. The multiple analysis of variance offers results of all variables instead of the individual results of Table 1. The Canonical Vectors (CV) [35] are projections of the original data in a lower dimensional space which allow a better discrimination between categories.

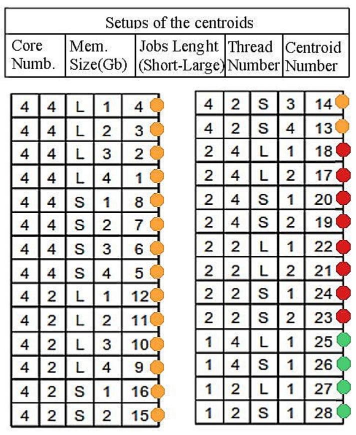
CV 1 and CV 2 correspond to GAUSS, CV 3 and CV 4 to AIRES and, finally, CV 5 and CV 6 to FasJet. On each of the three groups the first and second canonical vectors correspond to BM and KVM respectively. In this case, the survey shows that the Wilk's Lambda and the Pillai's Trace, are both significant in terms of probability *P* and *Snedecor F*, but only for CV 1 and CV 2 of LHCb software. In these Wilk's Lambda is close to 0 (< *0.08*) and the Pillai's Trace is higher than 1 with 2.53 CV 1 and 1.53 CV 2. The conclusion is that in the MANOVA of all dependent variables the significant ones are the GAUSS performances in BM and KVM.

Exact Signif.	Wilk's Lambda	Prob. P	Snedecor F	Pillai's Trace
CV 1	5.143E-005	0	2.3368	2.5391
CV 2	0.08621	0	1.4718	1.5397
CV 3	0.4338	0.9324	0.7788	0.7384
CV 4	0.6143	0.9913	0.6235	0.4446
CV 5	0.7893	0.9985	0.4691	0.2229
CV 6	0.8979	0.9833	0.4443	0.1020

Table 2: MANOVA survey of all canonical vectors

In this way, simple ANOVA is showing significance for each separated dependent variable and, as conclusion, the performances change depending on the setup. Furthermore, in the set of dependent variables in MANOVA is observed that there are correlations between the performances obtained for FastJet and AIRES and those obtained for GAUSS jobs, and this fact was expected as a response of the different software.





Setups of the centroids				
Core Numb.	Mem. Size(Gb)	Jobs Lenght (Short-Large)	Thread Number	Centroid Number
4	4	L	1	4 🟠
4	4	L	2	3 🟠
4	4	L	3	2 🟠
4	4	L	4	1 🟠
4	4	S	1	8 🟠
4	4	S	2	7 🟠
4	4	S	3	6 🟠
4	4	S	4	5 🟠
4	2	L	1	12 🟠
4	2	L	2	11 🟠
4	2	L	3	10 🟠
4	2	L	4	9 🟠
4	2	S	1	16 🟠
4	2	S	2	15 🟠
4	2	S	3	14 🟠
4	2	S	4	13 🟠
2	4	L	1	18 🔴
2	4	L	2	17 🔴
2	4	S	1	20 🔴
2	4	S	2	19 🔴
2	2	L	1	22 🔴
2	2	L	2	21 🔴
2	2	S	1	24 🔴
2	2	S	2	23 🔴
1	4	L	1	25 🟢
1	4	S	1	26 🟢
1	2	L	1	27 🟢
1	2	S	1	28 🟢

Figure 4: Correlation means of HEP-Software Wall-Time

Further on the correlation analysis, we show a biplot in Fig. 4. The biplot is the shadow in 2-dimensions of the normalized data, which maximizes variation. The measurements are shown as one unit of each axis, to give the idea of observations plotted correlations, which are corresponding to the distances between the points of the plot. The setup of the centroids is defined by core number, memory, large and short jobs, the thread number and finally the Centroid number. This configuration is going to be the same for all the biplots in the remainder of the paper.

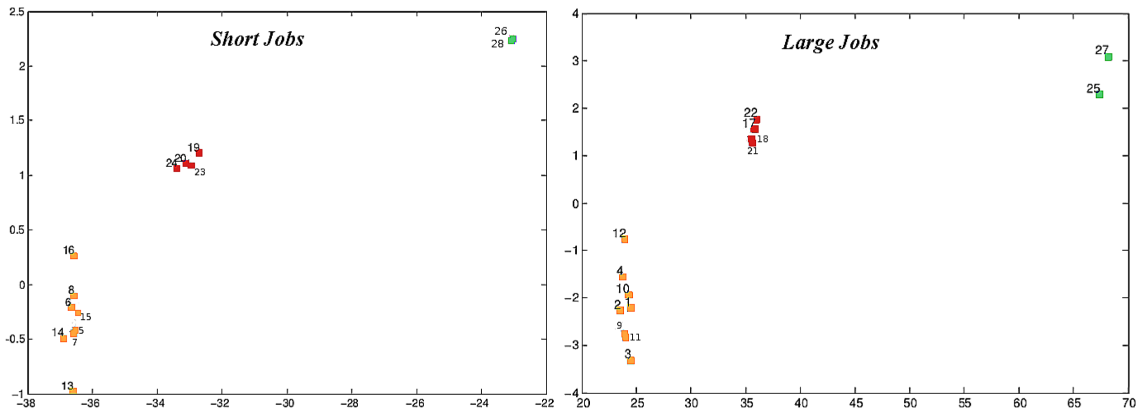


Figure 5: Centroids of short and large jobs.

Fig. 4 plots the occurrences of the performances of the six dependent variables. The first analysis of the biplot considers the different setup discriminations. The main discrimination of correlated occurrences is the type, either large (right side) or short (left side) jobs. These discriminations are shown more in detail in Fig.5.

As shown in Fig. 5, the occurrences of single cores (green color) are on similar areas with the setups of 2 and 4 GB, but the distance of the other setups along with the trend of the means, are suggesting that the single core setups are providing bad performances of HEP-Software. Another clear categorization is the occurrences of 2 Core setups (red color) in large and short jobs, independently of the amount of memory, there are correlations between them. These correlations allow us to conclude that an increase in memory does not improve performances. The occurrences of 4 Core setups (yellow color) are correlated in short and large jobs, this categorization shows that in large jobs the occurrences are closest with 4 GB of memory, therefore this type of jobs are more stable.

6.3 Individually performance Analysis of BM and KVM

Finally, in order to complete the Statistical Analysis, a study of BM and KVM performances was done.

Exact Signif. BM	Wilk's Lambda	Prob. P	Pillai's Trace
CV 1	0.00035287	9.3296e-111	1.6816
CV 2	0.41196	0.002807	0.68244
CV 3	0.80721	0.72265	0.19279
Exact Signif. KVM	**Wilk's Lambda**	**Prob. P**	**Pillai's Trace**
CV 1	0.00023424	2.1501e-118	1.8774
CV 2	0.24353	2.7345e-09	0.8783
CV 3	0.8273	0.83787	0.1727

Table 3: MANOVA Survey of BM and KVM

Table 3 shows the MANOVA survey that considers the possible correlations between three dependent variables in BM and KVM. And in this way, the set of dependent variables in MANOVA BM and KVM show a 2 dimensional significance with *$P < 0.01$*, *(9.3296e-111, 0.002)* in the case of BM and *(2.1501e-118, 2.7345e-09)* in the case of KVM. The Pillai's Trace is higher than 1 in both cases and the Wilk's Lambda is close to 0.

The MANOVA analysis of all the dependent variables for the current one indicates, as in the previous one, that the significant variables are the performances of GAUSS in the BM and KVM environments. The rest of dependent variables are probably correlated with the ones that characterize GAUSS.

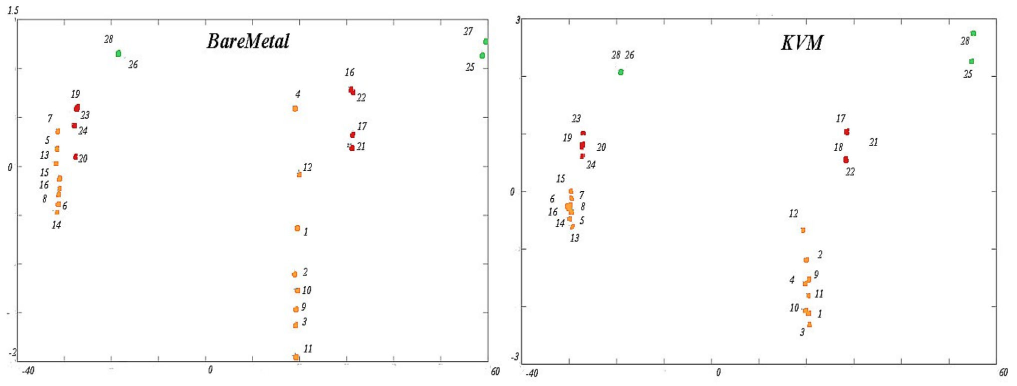


Figure 6: Bare Metal and Kernel-based Virtual Machine Centroids

Fig. 6 shows the centroids correlation of BM-MANOVA and KVM-MANOVA, they are similar to the sets of Fig. 4. Notice that there are some differences such as the distance between large jobs, with setups of 4Cores and 2-4 GB, in the case of KVM are closer than in the case of BM. This distance shows that the analyzed HEPSoftware seems to have stable performances in this type of setups for Virtualized environments.

An additional comment can be made about the distance between the centroids in the KVM and BM separated analysis. These distances are higher in KVM for the different setups, which show that in this HEP-Software the virtualized setup is an important factor to be considered when looking for a better performance.

7. Conclusions

This paper has presented a statistical analysis of performance over virtualization with different HEP-Software. The main challenge of this work is to know whether current cloud technologies provide enough performance for specific machine configurations. In this sense, the Kernelbased Virtual Machine (KVM) performance was analyzed with the use of multivariate analysis of variance methods. Some performance problems were found in relation with hardware architectures, in specific software called GAUSS of the LHCb experiment.

The analysis of variance, including the MANOVA statistics in Tables 1 and 2 and the corresponding biplots, demonstrates the dependent variable walltime have significant performance, for each software, of BM together with KVM, and there are significant differences depending on the length of the job. Furthermore, the amount of memory, when it is above 2Gb, does not seem to be an important factor in the performance of this job unlike the case of the number of cores where it is observed an increase in performance as the number of cores increases.

The separated MANOVA of BM and KVM in table 3 and the biplot of Fig 6, present similar results to the ones obtained in the previous case when MANOVA was applied to the six dependent variables, but the proximity of the centroids in the configurations of more core-memory show more stability in Virtualized machines than in Baremetal.

As cloud services gain maturity in HEP experiments, the correct configuration and setup of KVM hypervisor will be an important decision in order to reach an acceptable performance for the specific HEP-Software.

Acknowledgements

This work was supported in part by grants of the Ministerio de Educacion y Ciencia, Spain: FPA2007- 66437-C02-01/02 assigned to UB and FPA2010-21885-C02- 01/02 to USC. Additional support was provided by the Ministerio de Educacion y Ciencia, Spain and the EU ERDF program under contract TIN 2010-17541 assigned to USC and EU 7th Framework Programme INFRA- 2007-1.2.3: e-Science Grid infrastructures Grant Agreement Number 222667, Enabling Grids for e-Science (EGEE) project and INFRA-2010-1.2.1: Distributed computing infrastructure Contract Number RI-261323 (EGIInSPIRE).

References

[1] IIan Foster, Yong Zhao, Ioan Raicu, Shiyong Lu, Cloud Computing and Grid Computing 360-Degree Compared A.: Grid Computing Environments Workshop, 2008. GCE ’08

[2] Web page: Cloud computing *http://cloudcomputing.syscon.com/node/612375/print*

[3] Simon Ostermann, Alexandru Iosup, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, Dick Epema A.: A Performance Analysis of EC2 Cloud Computing Services for Scientific Computting

[4] Youse, L.,Wolski, R., Gorda, B.C., Krintz, C., Paravirtualization for HPC systems, In Frontiers of High Perf. Comput. and Networking - ISPA 2006 Workshops, Lect. Notes Comput. Sc., 4431 (2006)

[5] Deelman, E., Singh, G., Livny, M., Berriman, J.B., Good, J. The cost of doing science on the cloud: the Montage example. In: SC, p. 50. IEEE/ACM (2008)

[6] Armbrust M., Fox A., and etc Griffith R. A view of cloud computing. Communications of the ACM, 2010.

[7] Andrea Chierici, Riccardo Veraldi, A quantitative comparison between XEN and KVM A.: 2010 J. Phys.: Conf. Ser. 219 042005

[8] Web page: HEP-SPEC06 *https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiBenchHEPSPEC*

[9] Web page: EGI Fedcloud task force *https://wiki.egi.eu/wiki/Fedcloud-tf:Blueprint*

[10] Web page: EGI Fedcloud Testbed *https://wiki.egi.eu/wiki/Fedcloudtf:Testbed*

[11] Simon, A., Freire, E., Rosende, R., Diaz, I., Rey, P., LopezCacheiro, J., C.Fernandez: Egi fedcloud task force. In: IBERGRID, 6th Iberian Grid Infrastructure Conference Proceedings (2012)

[12] Web page: Venus-c *http://www.venus-c.eu*

[13] Web page: Windows Azure *http://www.windowsazure.com*

[14] Web page: StratusLab *http://www.stratuslab.org/doku.php/start*

[15] Konstantinou, I., Floros, E., Koziris, N.: Public vs private cloud usage costs: the StratusLab case. In: 2nd International WorkShop on Cloud Computing Platforms (2012)

[16] Rings, T., Caryer, G., Gallop, J., Grabowski, J., Kovacikova, T., Schultz, S., Stokes-Rees, I.: Grid and cloud computing: Opportunities for integration with the Next Generation Network. Journal of Grid Computing (2009). URL doi:10.1007/s10723-0009- 9132-y

[17] Fifield, T., Carmona, A., Casajus, A., Graciani, R., Sevior, M.: Integration of cloud, grid and local cluster resources with DIRAC. Journal of Physics: Conference Series 331(6), 062,009 (2011). URL *http://stacks.iop.org/1742-6596/331/i=6/a=062009*

[18] Web page: linux-kvm *http://www.linux-kvm.org/page*

[19] Web page: Xen *http://www.xen.org*

[20] Web page: QEMU *http://wiki.qemu.org/KVM*

[21] Web page: WLCG *http://wlcg.web.cern.c*

[22] Mendez, V., Fernandez, V., Graciani, R., Casajus, A., Pena, T. F., Merino, G., Saborido, J.J.: The integration of cloudstack and occi/opennebula with DIRAC. J. Phys.: Conf. Ser. 396 032075 (2012)

[23] Web page: CernVM-FS *http://cernvm.cern.ch/portal/*

[24] S.J. Sciutto. AIRES. A system for air shower simulations. *http://www2.fisica.unlp.edu.ar/auger/aires/doc/-UsersManual.pdf*

[25] L. Apolonario, N. Armesto and L. Cunqueiro. An analysis of the influence of background subtraction and quenching on jet observables in heavy-ion collisions. http://arxiv.org/abs/1211.1161

[26] L. Apolonario, N. Armesto and L. Cunqueiro. Background subtraction and jet quenching on jet reconstruction. Talk presented at the 5th International Conference on Hard and Electromagnetic Probes of High Energy Nuclear Collisions (Hard Probes 2012), Cagliari, Italy *http://arxiv.org/abs/1207.6587*

[27] M. Cacciari, G.P. Salam and G. Soyed. FastJet user manual, *http://arxiv.org/abs/1111.6097*

[28] I. Belyaev, G. Corti, S. Easo, W. Pokorski, F. Ranjard and P. Robbe. Gauss. LHCb Simulation Program. http://cern.ch/lhcbcomp /Simulation/Gauss.pdf. http://lhcbrelease-area.web.cern.ch/LHCbrelease- area/DOC/gauss/

[29] GEANT4 Collaboration. Geant4–a simulation toolkit *http://dx.doi.org/10.1016/S0168-9002(03)01368-8*

[30] G. Corti. Monte Carlo simulation(s) in LHCb and introduction to Gauss. https://indico.cern.ch/getFile.py/access?contribId=0&- resId=1&materialId=slides&confId=94046 (LHCb internal seminar).

[31] Pythia Collaboration. http://home.thep.lu.se/~torbjorn/- Pythia.html

[32] A. Ryd, D. Lange, N. Kuznetsova, S. Versille, M. Rotondo, D. Kirby, F.Wuerthewein and A. Ishikawa. EvtGen: A Montecarlo Generator for BPhysics. *http://robbep.home.cern.ch/robbep/EvtGen/GuideEvt-Gen.pdf*

[33] K. Hinkelmann and R.A. Fisher. The Design of experiments. Vol. 1. Jonh Wiley & Sons Inc. (2007)

[34] J. Delgado, A. S. Eddin, M. Adjouadi, S.M. Sadjadi, Paravirtualization for Scientific Computing: Performance Analysis and Prediction, High Performance Computing and Communications (HPCC), IEEE International Conference, Banff AB, 2-4 Sep 2011

[35] Web page: http://ibgwww.colorado.edu/%7Ecarey/p7291dir/handouts/manova1.pdf