Methodology for trade-off analysis when moving scientific applications to Cloud

Toni Mastelic†, Drazen Lucanin†, Andreas Ipp* and Ivona Brandic†
†Information Systems Institute
Vienna University of Technology, Argentinierstrasse 8/184-1, A-1040 Vienna, Austria
{toni, drazen, ivona}@infosys.tuwien.ac.at
*Institute for Theoretical Physics
Vienna University of Technology, Wiedner Hauptstrasse 8-10/136, A-1040 Vienna, Austria
ipp@hep.itp.tuwien.ac.at

***Abstract*—Scientific applications have always been one of the major driving forces for the development and efficient utilization of large scale distributed systems - computational Grids represent one of the prominent examples. While these infrastructures, such as Grids or Clusters, are widely used for running most of the scientific applications, they still use bare physical machines with fixed configurations and very little customizability. Today, Clouds represent another step forward in advanced utilization of distributed computing. They provide a fully customizable and self-managing infrastructure with scalable on-demand resources. However, true benefits and trade-offs of running scientific applications on a cloud infrastructure are still obscure, due to the lack of decision making support, which would provide a systematic approach for comparing these infrastructures.**

In this paper we introduce a comprehensive methodology for comparing the costs of using both infrastructures based on resource and energy usage, as well as their performance. We also introduce a novel approach for comparing the complexity of setting up and administrating such an infrastructure.

***Index Terms*—computing infrastructure; comparison methodology; cloud computing; trade-off analysis**

I. Introduction

The scientific community has always been one of the major consumers of increasing computational power starting with mainframes, clusters, up to large-scale distributed systems such as the Grid. It utilizes infrastructures such as E-SciencE (EGEE) spreading over 91 institutions, and TeraGRID [1], which is used by 4000 users around 200 universities for research work in molecular bioscience, mathematics, neuroscience, physics etc. Today, the cloud represents another step forward in advanced utilization of distributed computing, which provides a fully customizable and self-managing infrastructure with scalable on-demand resources. It offers several deployment types such as Infrastructure/Platform/Software as a Service (IaaS, PaaS and SaaS), as well as private, hybrid and public models [2]. Unlike Grids that are implemented with predefined environments [3], [4] that provide low or even no customizability, PaaS and specifically IaaS can provide a fully controllable infrastructure with a customizable execution environment [3].

A known drawback of using clouds is the loss in performance due to a virtualization layer [5]–[7], while other aspects such as application portability [8], energy and resource usage [9], reusability of execution environment [10], fixed configurations [3] etc. are usually neglected when it comes to comparing these infrastructures. While the scientific community has leveraged from using distributed computing for several decades, the lack of decision making support for evaluating benefits and trade-offs between two infrastructures inhibits it from harnessing the full power of the cloud.

In this paper we introduce a comprehensive methodology for comparing different computing infrastructures, which provides a systematic approach for calculating costs of using a certain infrastructure. We provide scientists with decision making support for evaluating benefits and trade-offs of choosing an infrastructure type for running their applications on local hardware. Our work focuses on comparing four aspects of these infrastructures: (1) energy and (2) resource usage, as well as (3) performance and (4) setup complexity. We monitor metrics of four basic resources: cpu, memory, disk and network, while performance is measured and compared by monitoring the application itself. Additionally, we introduce a novel approach for comparing the complexity of setting up and administrating an infrastructure based on the combination of a big O notation showing the asymptotic complexity and *story points*, a methodology used by agile development teams for estimating project complexity, which provides more fine-grained information.

The rest of this paper is organized as follows. Section II discusses related work. Section III provides a use case describing the issues when selecting an infrastructure type. Section IV defines a comparison methodology covering all four aspects (1-4) mentioned above. Evaluation setup is described in Section V. Finally, Section VI gives the conclusion and the future work.

II. Related work

There are several examples where the cloud has been utilized for scientific computation. In [3] Aneka cloud platform [11] is used for the classification of gene expression data and the execution of the fMRI brain imaging workflow. The authors compare the execution time and costs for running a scientific application on Amazon's cloud infrastructure. In [12] the authors present the Nebulous framework built on a

cloud middleware layer such as OpenNebula. It is used for simplifying the execution of MPI and OpenMP parallel applications in computational clouds. However, both approaches focus on providing a framework for scientific applications as part of a public PaaS cloud. They do not consider trade-offs and overheads for using cloud related technologies. A virtualized environment for scientific applications is proposed in [4]. They present the virtualization layer for a file system based on *chroot* that provides a generic interface for scientific applications. However, the authors do not consider full virtualization or other cloud related technologies. In [10] the authors compare costs and performance issues when using a cloud versus physical infrastructure. However, their work focuses only on data-intensive applications using Google App Engine, so they do not consider energy and resource consumption as they are using a public cloud. In [6] the authors analyze the performance in a Xen-based virtual cluster environment. They consider resource consumption and introduce a model for measuring the performance overhead for network latency and bandwidth. However, they focus only on virtualization technology without considering complete cloud infrastructure or energy consumption. [13] deals with moving text analysis tools to a cloud. However, the authors focus only on the implementation details and performance improvement. Energy and resource consumption overhead of virtualization technology is evaluated in [9]. The authors compare Xen and KVM hypervisors against bare metal execution by benchmarking the three environments. They present trade-offs and server consolidation guidelines in order to reduce energy usage. However, only virtualization technology is evaluated, while other technologies used in a cloud infrastructure are not considered. Moreover, none of the related work we found considers the setup complexity of each infrastructure, nor tries to compare them.

III. Use case

The scientific community has always required state of the art computing infrastructures in order to benefit from rapidly advancing technologies. Today these infrastructures are based on large-scale distributed systems such as Open Science Grid [14] hosting up to 25000 machines. However, a cloud infrastructure and its related technologies such as the virtualization and the self-management are still not fully utilized, and trade-offs between these two infrastructures are still not clear. Here we provide a use case describing the issues when selecting an infrastructure for running scientific applications. We consider a local hardware infrastructure with 50 physical machines and an application with 120 hours of the execution time.

Physical infrastructures (i.e., Grids) are usually implemented as bag of tasks (BoTs), workflows or Message Passing Interface (MPI) applications. However, some scientific applications do not fit these models and many of them require a complete redesign and reprogramming in order to run in preconfigurated environments [3]. In our use case, scientist A wants to run his application App that requires the execution environment Env on a *physical infrastructure* as shown on Figure 1. Since a *physical infrastructure* is preconfigured with the execution environment Env', scientist A has to *adapt* his application and build App' in order to run it. After the adaptation, he has to *install* and *run* his application on each node separately. Even after successfully adapting his application, due to constant system upgrades, hardware changes etc., further adaptation will still be required [4]. However, reprogramming a modern scientific application represents a complex task [4], which suggests that a traditional execution model - build and run; then adapt, re-build and re-run - no longer satisfies their advanced requirements [8]. Furthermore, in order to boost the performance of his application, scientist A decides to add an *additional node* PM_1 as shown on Figure 1. However, the *additional node* is preconfigured with an environment Env^*, previously used by some other application requiring a specialized execution environment. He has to *reconfigure* the node and rebuild it with the environment Env', install the application and run it. Finally, since his application is now running directly on physical machines, he can expect good *performance*.

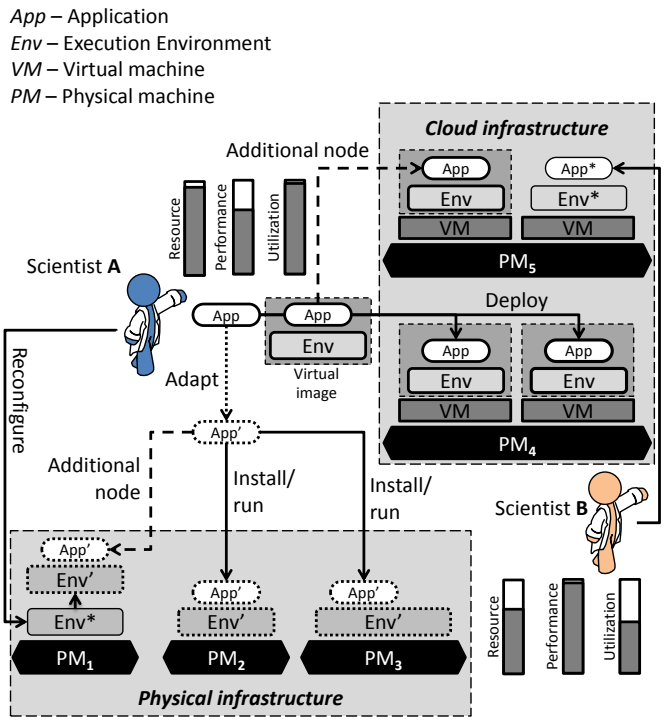

Figure 1. Use case for choosing an infrastructure.

If scientist A decided to run his application on a *cloud infrastructure*, he only has to create a single *virtual image* with a fully customized execution environment Env for his application, and *deploy* it on as much nodes as he likes (Figure 1). Moreover, if he requires an *additional node*, he only has to deploy the same *virtual image* he created in the beginning. Any system upgrades and hardware changes are transparent to his application as it always runs within a customizable virtual environment. However, due to a virtualization layer, he can expect worse *performance* and greater *resource* usage.

On the other hand, if he does not use all the resources of a particular physical machine, i.e., PM_5 on Figure 1, by slicing the physical machine to several virtual machines, scientist B can use those resources. Thus, a resource *utilization* is expected to be better [9] than on a *physical infrastructure* where PM_3 on Figure 1 is not being fully utilized.

In the following section we introduce our methodology for supporting scientists when choosing an infrastructure for running their applications. For terminology, we use a term *node* that represents an operating system running on a machine, where the machine can be physical (PM) or virtual (VM).

IV. Comparison methodology

Comparing infrastructures described in our use case in Section III can be done by looking at many different aspects such as cost, performance, usability etc. In our work we focus on running a scientific application on a private infrastructure, thus we construct our comparison methodology based on several important aspects: setup complexity, resource and energy consumption, as well as the performance. Each of these aspects is described and modeled in the following sections.

A. Complexity model

Setup time could vary depending on the used hardware and software, as well the expertise of a person setting up this infrastructure. Therefore, we introduce a novel approach for comparing infrastructures based on the complexity of a setup procedure using big O notation for expressing asymptotic complexity. We define the complexity as O(N), where N can be 0, 1 or n, where n represents a number of nodes being set up. However, since using only big O notation does not provide a fine grained complexity we require, we expand it by combining it with a *story points* [15], a methodology widely used in agile software development. *Story points* break the procedure (i.e., a story) to a number of points, which we refer to as tasks. Each task is assigned with a complexity grade that abstractly describes the complexity, the effort and the uncertainty for executing a certain task. Grades are defined relatively and form a complexity scale similar to the one in Table I, which shows a modified fibonacci scale widely used by agile software development teams for estimating project complexity combined with our own arbitrary naming convention. However, following the scheme of *story points*, the complexity scale can be fully customized to better reflect infrastructures being compared, as well as the expertise of an administrator team that performs the setup. It is important to notice that a combination of these two methodologies is highly required, since neither of them could solely express the complexity required for setting up computing infrastructure: the big O notation would only show an asymptotic complexity without fine-grained details, while *story points* would show the complexity of each task without considering the number of times this task has to be repeated.

Our complexity model covers all the steps from setting up hardware to the moment when an application is being executed. Four steps are defined: *hardware setup*, *platform setup*, *environment setup* and *run*. The steps are divided into several tasks that are noted using the big O notation and graded using our complexity scale. The defined grades and big O notations are used to calculate complexity of each step using Equation 1, where Cx^{infra} is a complexity of a step x for an infrastructure $infra$, T is a total number of tasks, C_i is the complexity grade c_j and N_i is the big O notation for task t_i. The big O notation for all tasks is shown in Table II and is defined as a *default* notation, which assumes that no helper or deployment tools (e.g., dodai-deploy [16]) is used for their completation. This clearly shows the flexibility of our complexity model in that it can be easily modified by simply adding and/or removing tasks from a certain step, as well as changing the big O notation if a task has been automated.

Table I
Complexity scale for grading tasks.

ID	Task complexity	Grade
c_0	do nothing	0
c_1	one-line commmand	2
c_2	using wizard	3
c_3	advanced setup	5
c_4	manual configuration	13
c_5	recompile and link	40
c_6	to many components	100
c_7	reprogram	1000

$$Cx^{infra} = \sum_{i=1}^{T} C_i \cdot N_i^{infra} \tag{1}$$

We describe and analyze latter three steps in the following sections, while we exclude the *hardware setup* step since it is the same for both infrastructures and is performed only in the initial setup.

Step 1: Platform setup includes installing an operating system and related drivers, as well as any additional piece of software that is required for building the target infrastructure. It consists of the following tasks:

- t_1 Installing an operating system
- t_2 Installing drivers
- t_3 Installing a hypervisor
- t_4 Installing a cloud management software

For a physical infrastructure only tasks t_1 and t_2 apply, while a cloud infrastructure requires all the tasks, i.e., t_1, t_2, t_3 and t_4. Tasks t_1, t_2 and t_3 must be performed on each computing node separately, which makes them O(n). Task t_4 is performed only once on a central node, which makes it O(1). Equations 2 and 3 represent the complexity of the step 1 for a physical and a cloud infrastructure respectively.

$$C1^{physical} = (C_1 + C_2)n \tag{2}$$

$$C1^{cloud} = (C_1 + C_2 + C_3)n + C_4 \tag{3}$$

Step 2: Environment setup considers setting up an execution environment for an application. This includes installing all

Table II
COMPLEXITY GRADES AND BIG O NOTATION FOR EACH TASK

Task	C	$N^{phsyical}$	N^{cloud}
t_1	c_2	n	n
t_2	c_3	n	n
t_3	c_1	0	n
t_4	c_6	0	1
t_5	c_2	n	1
t_6	c_2	n	1
t_7	c_1	0	1
t_8	c_1	1	1

of the dependent libraries and the application itself. It consists of the following tasks:

t_5 Installing libraries
t_6 Installing application

Both tasks t_5 and t_6 apply for both infrastructures. However, for a physical infrastructure this step is performed on every node separately, thus both tasks in this step are O(n) as seen in Equation 4. For a cloud infrastructure this step represents creating a virtual image, thus both tasks are performed only once as shown in Equation 5.

$$C2^{physical} = (C_5 + C_6)n \tag{4}$$

$$C2^{cloud} = C_5 + C_6 \tag{5}$$

Step 3: Run represents the final step where an application is deployed and executed. It consists of the following tasks:

t_7 Deploy
t_8 Run

Setting up a physical infrastructure includes only task t_8 (Equation 6), since the application is already considered as deployed after it has been installed. For a cloud infrastructure the task t_7 is required and refers to the deployment of virtual images. Its complexity is O(1) since images are deployed automatically by a cloud management software, where the complexity of the entire step is represented with the Equation 7.

$$C3^{physical} = C_8 \tag{6}$$

$$C3^{cloud} = C_7 + C_8 \tag{7}$$

Scenarios: For comparing the complexity of setting up both infrastructures using our model described above, we consider three scenarios:

- **Initial setup**, which represents the first setup of an infrastructure for a certain application. Both infrastructures require all three steps for this scenario, where the complexity is calculated by combining the complexities of all included steps. Thus, for a physical infrastructure we combine equations 2, 4 and 6 that gives Equation 8. For a cloud infrastructure we take equations 3, 5 and 7, which gives Equation 9.

$$C3^{physical} = (C_1 + C_2 + \mathbf{C_5} + \mathbf{C_6})n + C_8 \tag{8}$$

$$C3^{cloud} = (C_1 + C_2 + \mathbf{C_3})n + \mathbf{C_4} + \mathbf{C_5} + \mathbf{C_6} + \mathbf{C_7} + C_8 \tag{9}$$

As seen from equations 8 and 9 the initial setup of a physical infrastructure has four tasks depending on a number of nodes n, while a cloud has only three. Since C_1 and C_2 apply for both infrastructures, C_3^{cloud} has to be equal to $C_5^{physical} + C_6^{physical}$ in order for complexities to have the same linear growth - that is, setting up a hypervisor has to have the same complexity as setting up libraries and the application itself. However, as opposed to a physical infrastructure (line L_0 in Figure 2), a cloud infrastructure would still have greater initial complexity because of the one-time tasks t_4, t_5, 6 and t_7 with their complexities C_4, C_5, C_6 and C_7 respectively, as shown by the line L_1 on Figure 2.

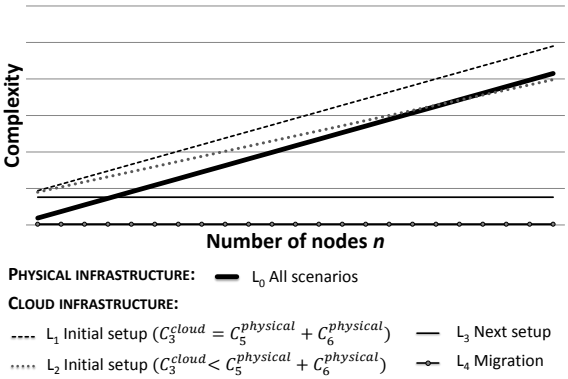

Figure 2. Complexity graph for different scenarios.

However, if $C_3^{cloud} < C_5^{physical} + C_6^{physical}$, then setting up a cloud infrastructure pays of after a certain number of nodes as shown by the line L_2 on Figure 2.

- **Next setup** assumes that the initial setup has been done, and now we want to run a different application requiring a different execution environment. For this scenario a physical infrastructure still requires all three steps (Equation 8), since a new environment might also require a different operating system. On the other hand, a cloud infrastructure requires only steps $S2$ and $S3$ since a new environment (i.e. new virtual image) can be deployed on top of the existing infrastructure (Equation 10).

$$C3^{cloud} = C_4 + C_5 + C_6 + C_7 + C_8 \tag{10}$$

As seen from the Equation 10 a cloud infrastructure has only O(1) tasks (line L_3 in Figure 2), while a physical infrastructure keeps its linear complexity growth depending on a number of nodes as shown by the line L_0 in Figure 2. Even if step $S1$ would not be required, that is if an existing operating system would be sufficient, the physical infrastructure would still have a setup complexity of linear growth because of step $S2$ and its complexity expressed in Equation 4.

- **Migration** considers migrating an application to another infrastructure of the same type. A physical infrastructure

again requires all the three steps defined by Equation 8 and shown with line L_0 in 2, as we cannot assume that a new environment will be fit for the application being migrated. On the other hand, a cloud only requires step $S3$ (Equation 7), since the infrastructure already exists, as well as the virtual image that only needs to be deployed and executed. Thus, the complexity of a cloud infrastructure for the migration scenario is constant as shown by the line L_4 on Figure 2.

B. Performance model

The performance is monitored in order to compare the behavior of an application and the trade-off of using a specific infrastructure. The performance is measured using a performance metric specific to a running application, referred to as a *specific class metric* in [17]. The application is benchmarked until the configuration with the best performance is found, and is later compared using the same approach on a different infrastructure. We consider different *application parameters* that are specific to the application, e.g., number of workers per node in an MPI application. The goal is to avoid influencing the results by selecting a configuration that is more suited for a certain infrastructure type.

However, based on the existing research results from [6], we can see that paravirtualization results in small performance degradation, which makes infrastructures that use paravirtualization a good alternatives to physical infrastructures. However, such alternatives are only acceptable if the time lost due to performance degradation is less than the time saved due to shorter setup time. The same applies for full virtualization as well. However, full virtualization shows worse performance than paravirtualization [9], thus the setup time must be even shorter to compensate for it.

After the performance trade-off is calculated, resource and energy consumption is compared between the two infrastructures in order to measure consumption overheads.

C. Resource model

As described in the use case in Section III, reasons for measuring resource consumption are twofold: (1) calculate the resource consumption overhead by measuring the average resource consumption during an application runtime and (2) resource utilization to see how much resources were wasted. We selected four main resources that we monitor on a system level for each physical machine: cpu, memory, disk and network traffic, inbound and outbound. Total consumption R_{used} of a specific resource is calculated by summing up consumption r over all machines n using Equation 11. As shown in [9] resource consumption will generally be greater in a cloud infrastructure due to the virtualization overhead, as well as the overhead of a cloud management software.

$$R_{used} = \sum_{i=1}^{n} r_i \tag{11}$$

For cpu usage we chose the cpu time metric, which represents the time in milliseconds used by a cpu for executing some task. The resident memory metric is used for a memory resource. It represents the amount of bytes allocated within physical RAM. For a disk resource we take the disk usage metric defined as the amount of bytes stored on a hard-disk. Finally, network bandwidth is used for calculating the network usage. It represents the number of bytes/sec transmitted and received over a network.

Utilization U of a specific resource is calculated using Equation 12, where R_{used} is the amount of used resources and $R_{reserved}$ is the amount of resources that has been reserved for the execution of an application.

$$U = \frac{R_{used}}{R_{reserved}} \tag{12}$$

We consider static resource reservation where each application requires its own execution environment, with R_{total} being the total amount of available resources. Thus for a physical infrastructure $R_{reserved}$ will always be equal to R_{total} since all the resources of a physical machine are reserved, while for a cloud infrastructure $R_{reserved}$ can be less than R_{total} since resources can be sliced to several virtual machines. Due to this fact, utilization is expected to be better in a cloud infrastructure, as shown in [9].

D. Energy model

Power consumption is measured for each physical machine within the infrastructure, after which an average value is calculated during an application runtime. Energy used by the application depends not only on the average power consumption $W_{average}$, but on total execution time T_{total} as well. Thus we formulate Equation 13 in order to calculate a total energy usage E_{total} of the application.

$$E_{total} = W_{average} \cdot T_{total} \tag{13}$$

Due to a generally longer execution time on an infrastructure that uses a virtualization and higher resource usage due to its overhead, the energy consumption is expected to be greater for a cloud infrastructure than for a physical infrastructure [9].

V. Evaluation setup

We plan to evaluate our comparison methodology in a real world scenario on our own local infrastructure by building both cloud and a physical infrastructures. Machines are already connected with an ethernet connection and supplied with power through Dominion PX [18] power unit used also for measuring the energy consumption. Both for physical and virtual nodes we plan to use Ubuntu Server [19] as an operating system. Moreover, for cloud infrastructure we will use OpenStack [20] as a cloud management software and KVM [21] for a virtualization layer.

For performing our evaluation, we intend to use a real world scientific application for calculating Weibel instabilities within quark-gluon plasma during particle collisions [22] such as those in the Large Hardon Collider at CERN [23], all this in collaboration with the Particle Physics Group from the Institute for Theoretical Physics at Vienna University

of Technology. After analyzing preliminary research results, we conclude that this application can be benchmarked for all four selected resources from the Section IV-C, as well as energy consumption. For monitoring resource and energy consumption we will use an improved version of our M4Cloud monitoring tool presented in [17].

Evaluation comprises of four steps: (1) building both infrastructures using our local hardware in order to evaluate complexity of the different setup procedures; (2) performing tests with different application and system parameters as defined in Section IV-B in order to find a best performing configuration; (3) resource and energy consumption will be measured for the best configuration in order to measure a resource and energy consumption overheads; (4) comparing and analyzing the results to see which infrastructure provides more benefits for building and executing the scientific application.

This evaluation will not only show the usability of our comparison methodology, but will also provide some new insights into trade-offs between the two infrastructures.

VI. Conclusion and future work

In this paper we introduced a methodology for comparing different computing infrastructures used by the scientific community. We demonstrated that comparing two infrastructures goes beyond comparing only performance trade-offs. By introducing a novel approach for measuring complexity of setting up and administrating an infrastructure, we added another dimension for comparing the trade-offs between infrastructures. We combined it with performance, energy and resource consumption overheads and provided a straightforward approach for supporting scientists in their decision-making procedure for choosing an infrastructure.

In addition to performing the evaluation described in the Section V, we also consider extending our work by including an application development step in order to define a methodology for agile cloud software development, which would provide clear guidelines for scientists when targeting their applications to a specific infrastructure. Moreover, we plan to improve our complexity model by adding a *learning curve* parameter, as well as task parallelization, and use it for making an in-depth analysis of different cloud setups.

References

[1] C. Catlett, "TeraGrid: A Foundation for US Cyberinfrastructure," in *Network and Parallel Computing* (H. Jin, D. Reed, and W. Jiang, eds.), vol. 3779 of *Lecture Notes in Computer Science*, ch. 1, p. 1, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2005.

[2] P. Louridas, "Up in the air: Moving your applications to the cloud," *Software, IEEE*, vol. 27, pp. 6 –11, july-aug. 2010.

[3] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, pp. 4 –16, dec. 2009.

[4] B. Konning, C. Engelmann, S. Scott, and G. Geist, "Virtualized environments for the harness high performance computing workbench," in *Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on*, pp. 133 –140, feb. 2008.

[5] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and modeling virtualization performance overhead for cloud environments.," in *CLOSER* (F. Leymann, I. Ivanov, M. van Sinderen, and B. Shishkov, eds.), pp. 563–573, SciTePress, 2011.

[6] K. Ye, X. Jiang, S. Chen, D. Huang, and B. Wang, "Analyzing and modeling the performance in xen-based virtual cluster environment," in *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, pp. 273 –280, sept. 2010.

[7] A. Iosup, S. Ostermann, M. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, pp. 931 –945, june 2011.

[8] P. Kang, E. Tilevich, S. Varadarajan, and N. Ramakrishnan, "Maintainable and reusable scientific software adaptation: democratizing scientific software adaptation," in *Proceedings of the tenth international conference on Aspect-oriented software development*, AOSD '11, (New York, NY, USA), pp. 165–176, ACM, 2011.

[9] Y. Jin, Y. Wen, and Q. Chen, "Energy efficiency and server virtualization in data centers: An empirical investigation," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 133 –138, march 2012.

[10] A. Angabini, N. Yazdani, T. Mundt, and F. Hassani, "Suitability of cloud computing for scientific data analyzing applications; an empirical study," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on*, pp. 193 –199, oct. 2011.

[11] C. Vecchiola, X. Chu, and R. Buyya, *Aneka: A Software Platform for .NET-based Cloud Computing*. IOS Press, 2010.

[12] G. Galante and L. de Bona, "Nebulous: A framework for scientific applications execution on cloud environments," in *Sistemas Computacionais (WSCAD-SSC), 2011 Simpasio em*, p. 6, oct. 2011.

[13] H. Vashishtha, M. Smit, and E. Stroulia, "Moving text analysis tools to the cloud," in *Services (SERVICES-1), 2010 6th World Congress on*, pp. 107 –114, july 2010.

[14] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Wrthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, and R. Quick, "The open science grid," *Journal of Physics: Conference Series*, vol. 78, no. 1, p. 012057, 2007.

[15] M. Cohn, *Agile Estimating and Planning*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[16] N. I. of Informatics Cloud Team, "dodai-deploy - software management tool.." https://github.com/nii-cloud/dodai-deploy/wiki, 2012.

[17] T. Mastelic, V. Emeakaroha, M. Maurer, and I. Brandic, "M4cloud - generic application level monitoring for resource-shared cloud environments," in *CLOSER 2012, 2st International Conference on Cloud Computing and Services Science*, 2012.

[18] Raritan, "Dominion PX - inteligent power distribution unit.." http://www.raritan.com/drc/files/products/DominionPX/DominionPX_1.10-UserGuide.pdf, 2008.

[19] Canonical, "Ubuntu - debian based linux distribution." http://www.ubuntu.com/, 2012.

[20] OpenStack, "OpenStack - open source software for building private and public clouds.." http://www.openstack.org/, 2012.

[21] R. H. Inc., "KVM - kernel based virtual machine.." http://www.linux-kvm.org, 2012.

[22] A. Ipp, A. Rebhan, and M. Strickland, "Non-abelian plasma instabilities: Su(3) versus su(2)," *Phys. Rev. D*, vol. 84, p. 056003, Sep 2011.

[23] CERN, "LHC - large hadron collider.." http://public.web.cern.ch/public/, 2012.