Practical Considerations in Cloud Utilization for the Science Gateway nanoHUB.org

Lynn K. Zentner[1], Steven M. Clark[2], Preston M. Smith[2],
Swaroop Shivarajapura[1], Victoria Farnsworth[1], Krishna P.C. Madhavan[1,3], Gerhard Klimeck[1,4]
[1]Network for Computational Nanotechnology, Birck Nanotechnology Center,
[2]Rosen Center for Advanced Computing, ITaP,
[3]School of Engineering Education,
[4]School of Electrical and Computer Engineering,
Purdue University, West Lafayette, Indiana, USA
{lzentner, clarks, psmith, swaroop, vfarnsworth, cm, gekco}@purdue.edu

***Abstract*—nanoHUB.org is arguably the largest online nanotechnology user facility in the world. Just between July 2010 and June 2011 it served 177,823 users. 10,477 users ran 393,648 simulation jobs on a variety of computational resources ranging from HUBzero-based virtual execution hosts for rapid, interactive runs as well as grid-based resources for computationally-intense runs. We believe that as such our users experience a fully operational scientific "cloud"-based infrastructure even though it is not using "standard" computational cloud infrastructures such as EC2. In this paper we explore the use of standard computational cloud-based resources to determine whether they can deliver satisfactory outcomes for our users without requiring high personnel costs for configuration. In a science gateway environment, the assignment of jobs to the appropriate computational resource is not trivial. Resource availability, wait time, time to completion, and likelihood of job success must all be considered in order to transparently deliver an acceptable level of service to our users. In this paper, we present preliminary results examining the benefits and drawbacks of utilizing standard computational cloud resources as one potential venue for nanoHUB computational runs. In summary we find that cloud resources performed competitively with other grid resources in terms of wait time, CPU usage, and success in a multiple job submission strategy.**

Keywords-science gateway; nanotechnology; nanoHUB; grid computing; cloud computing; performance monitoring; HUBzero; cyber-environments

I. INTRODUCTION

nanoHUB.org is recognized as the largest Internet-based nanotechnology user facility in the world, serving over 177,000 users annually in 172 countries [1]. The infrastructure that underlies nanoHUB is called HUBzero® [2]. nanoHUB hosts a wide and growing variety of resources used heavily for education and research [3,4]. As of July 2011, these resources include 55 full courses, 1657 online lectures, and 212 simulation tools. Analysis of 719 citations in the scientific literature of nanoHUB.org demonstrates use of our simulation programs and resources by educators, researchers, and experimentalists outside the nanoHUB community [5]. For our users, nanoHUB itself aims to act as a cloud of tools, compute services, and compute hardware resources focused primarily on enhancing scientific inquiry. Over 10,000 users performed simulation runs on nanoHUB.org in the last year. Many of these runs are fast, interactive simulations that execute well in our HUBzero-based virtual execution hosts. However, an increasing number of simulations run on nanoHUB.org are computationally intense, requiring multiple cores in a true parallel environment or, for more modest parallel runs, the ability to execute in a distributed environment on many serial machines. Like the user base represented in many engineering and science cyber-environments (also referred to interchangeably as science gateways), nanoHUB.org users are not computational experts. They usually prefer to focus on performing domain-specific computations and require a transparent environment for job submissions that returns results quickly and reliably [6]. Most of these users are unaware of computational details such as MPI, OpenMP, or batch queues. At nanoHUB, we continue to explore and refine methods for optimizing the job submission process and use of available computational resources to best serve the interests of our users.

In this paper, we attempt to characterize the benefits and drawbacks of using standard computational cloud resources as part of the nanoHUB job submissions process. Our ultimate goal is to understand how cloud-based computational resources could affect the user experience in the context of simulation use within engineering and science cyber-environments such as nanoHUB.org.

II. BACKGROUND

There is much discussion in the literature concerning the definition and evolution of cloud computing and how it relates to cluster and grid computing. One view is that cloud computing has evolved from cluster and grid computing and yet possesses several notable differences [7]. A typical grid job in the science gateway context will be deployed to a specific physical resource, possibly remote from submission site, via a community account with specific instructions and through a scheduling engine. The job may wait in a queue until the designated host is available for execution. Access and configuration parameters are controlled at the execution host.

Cloud computing, in contrast, consists of a large, centralized, underlying hardware system over which virtual machines (VMs) can be spawned as necessary and dynamically configured to provide run-time environments

for a variety of jobs [8]. In this context, the cloud provides Infrastructure as a Service (IaaS) [9]. Commercial systems such as Amazon's Elastic Compute Cloud (EC2), provide access on demand under a "pay as you use" model [10].

A recent study [11] compared the cost to an institution to utilize a local community cluster in comparison to accessing commercial cloud services. This study concluded that while for most use cases, the cost of using a local community cluster would be less than costs associated with utilizing cloud services, there were cases when the cloud would be the better choice. For example, for research groups with low utilization of their queues or in situations where time to completion was a high priority, cloud services would offer advantages over traditional grid resources. When security and logistics issues are addressed, utilizing cloud services in conjunction with other resources can be an effective approach to managing fluctuations in demand [12]. However, a careful approach to scheduling is necessary in order to balance costs against user requirements [13].

Engineering and science cyber-environments such as nanoHUB.org attract users with the promise to provide a high quality user experience when using simulation tools and services. One of the primary factors affecting user experience when using cloud services is the performance of the applications themselves. Performance of many applications in a cloud is generally comparable in terms of CPU, memory, and MPI performance to typical commodity Gigabit Ethernet-based clusters [14]. One of the most notable shortcomings for high-performance computing in cloud computing is network performance [15], as observed in benchmarks run on Amazon EC2. Also, I/O intensive applications performed substantially worse under EC2's Gigabit Ethernet inter-connect and comparatively slower file system [16]. Typically, applications that work well in a cloud environment are loosely-coupled and CPU-intensive, without heavy communication or I/O requirements.

Science gateways such as nanoHUB have traditionally deployed computationally intense jobs to local cluster resources as well as to grid resources such as the TeraGrid [17], Open Science Grid (OSG) [18], and DiaGrid [19]. An initial study on grid submissions from nanoHUB.org to a variety of grid resources indicated better success rates with jobs submitted to local clusters than to remote grid sites [20].

In November 2008, a joint nanoHUB-OSG Task Force [21] began to address issues related to effectively utilizing grid resources through a science gateway such as nanoHUB. One of the outcomes from this task force was the development of a grid probe system used to test grid site health and direct jobs based on the results. The grid probe is a simple program sent at regular intervals to various grid sites. The resulting response times of the probes are actively utilized to direct production job submissions from nanoHUB. In essence, the grid probe system provided us with some intelligence on where to guide user jobs where there was no guidance before. Over 1 million probe results have been collected since this project began.

Success rates for production runs from nanoHUB.org submitted based on grid probe results indicate that there is still a significant failure rate for real jobs sent to TeraGrid, OSG, and others, with as many as half of them failing for reasons ranging from user error to grid errors (see Table 1). Further work is planned in order to develop more complex probes that are more representative of actual jobs.

The fundamental question that is being posed here is if standard cloud computing infrastructures can raise the level of service in comparison to standard grid computing infrastructures. This paper documents some of our first results that address this overall complex issue.

Table 1. Success Rates for Grid Submissions from nanoHUB.org

	TeraGrid (778 runs)	OSG (29,448 runs)	Local Cluster (61,124 runs)
Successful	49%	52%	84%
Failed	51%	48%	16%

III. THE TESTING PROCESS

A. Job submission process to the cloud

As an extension to the work described in the previous section, we propose to determine whether the cloud presents an alternate or supplemental venue for job submissions and how the success rates for production jobs sent to the cloud compare with that for grid jobs and jobs submitted to local clusters. Our work is focused on job submissions from environments such as nanoHUB.org that are aimed to serve end users without any grid or cloud programming knowledge or access. We aim to determine both the benefits and drawbacks of utilizing the cloud as part of our computational resources. It is expected that the virtualization in the cloud environment will offer the benefit of jobs not having to wait in a queue, as well as the ability to provide a known execution environment in the virtual containers.

We used Purdue's cloud computing test bed, "Wispy" [22], an IaaS type cloud service, as our venue for cloud submission. Wispy operates the Nimbus v2.7 software produced by the Science Clouds Project [23]. Within Wispy, there are 32 nodes with 4 cores at 2.3GHz (96 cores in total), and 16GB of memory on each node. Each node has 250GB of local disk. Each physical node in Wispy is connected with gigabit Ethernet, with IP addresses directly connected to Purdue's public Internet. For virtualization, Wispy uses the KVM [24] hypervisor, which is included by default in Red Hat Enterprise Linux distributions. Figure 1 illustrates the components and protocols utilized by the Wispy cloud service.

For this study, sixteen virtual machines were configured with the ability to run a maximum of two jobs per machine, with a machine being terminated after 30 minutes of idle time. If no available slots were open on an already running machine, a new machine was opened. The virtual machines have a maximum lifetime of 24 hours. Jobs are not scheduled if they cannot complete in the remaining time.

The selection of the number of virtual machines and jobs per machine was somewhat arbitrary. For this initial set of tests, our intent was to explore the behavior of the cloud

without unduly stressing it. Future work may be considered which will vary these parameters and compare results. It should be noted that during these tests, the load on Wispy was essentially nanoHUB jobs. In most cases, we had the entire cloud to ourselves. We recognize this as a limitation of the current study. This issue will be addressed in the future. While there is no traditional queue, there is a 2-3 minute start up time for each virtual machine, but as long as the machines are kept busy, this is generally a one-time cost.

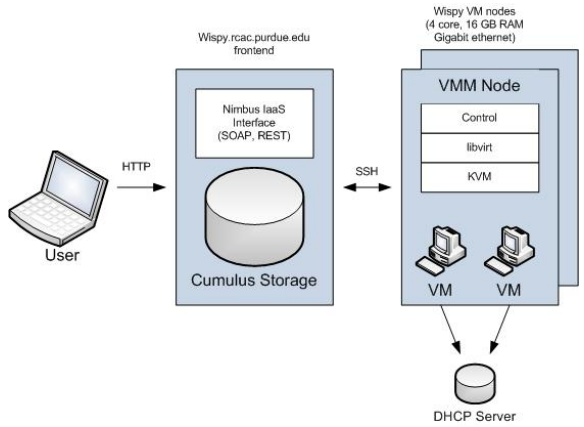

Figure 1. Components and Protocols used in Wispy Cloud Service

The testing approach was based on real user experience and desired outcomes and was structured to simulate real interactive sessions in which time to completion counts and where a variety of venues are available for job submission. To begin our tests, a set of three representative applications that are regularly utilized by nanoHUB users were selected: Nanowire [25], NanoFET [26], and CNTFET [27]. All three applications run on a single core and generally take more than two minutes to run. They were selected for these tests because they are some of the more common applications run on nanHUB.org and are generally sent to grid resources. In addition to Wispy, we utilized other venues available for nanoHUB jobs, including BoilerGrid [28], DiaGrid, OSG Factory [29], OSG, a dedicated local queue available to nanoHUB (rcac-ncn-hub), as well as standard local queues (rcac-standby). Table 2 provides details on the capabilities of the hardware used.

Table 2. Hardware specifications

Site Groups	CPU	RAM
BoilerGrid /DiaGrid	2.5 GHz Quad-Core AMD 2380 2.1 GHz 12-Core AMD 6172 2.33 GHz Quad-Core Intel E5410	32G 48G 16G
RCAC: ncn-hub/ standby	2.5 GHz Quad-Core AMD 2380 2.1 GHz 12-Core AMD 6172	32G 48G
Wispy	2.3 GHz Intel Xeon 5140	16G
OSG	Various Intel/AMD – 2.5GHz avg	12G
OSG Factory	Various Intel/AMD – 2.4GHz avg	18G

Each of the three applications were randomly submitted to three venues at the same time with data being collected in log files in order to compare success rates as well as wait time, wall time, and CPU time across the venues. These data are discussed in the next section.

IV. RESULTS AND DISCUSSION

Initial testing resulted in a total of 17,871 runs submitted across the venues. A run was considered to be successful only if the following two conditions were met: (i) it completed successfully; and (ii) it was the first to finish in the set of parallel runs that were submitted to various venues. Once one run in the set completed successfully, the redundant runs were terminated. This approach follows a common multiple submission strategy shown to improve job completion time, and used by both nanoHUB.org and other sites [30]. A small percentage of jobs terminated due to errors, accounting for about 4% of all the runs.

Table 3 and Figure 2 show the resulting data. It should be noted that due to scheduled hardware maintenance issues at Purdue University, there were reduced job submissions to the local RCAC queues and to Wispy, compared to the number of job submitted to other grid sites.

Table 3. Detailed Numerical Results of the Test Runs

Venue	Success-ful	Errors	Lost Race	Total
BoilerGrid	1198	356	1760	3314
DiaGrid	1635	0	1429	3064
OSG	667	95	2527	3289
OSG Factory	1108	33	2140	3281
rcac-ncn-hub	681	139	591	1411
Rcac-standby	344	118	489	951
Wispy	980	53	1528	2561
Total	6613	794	10464	17871

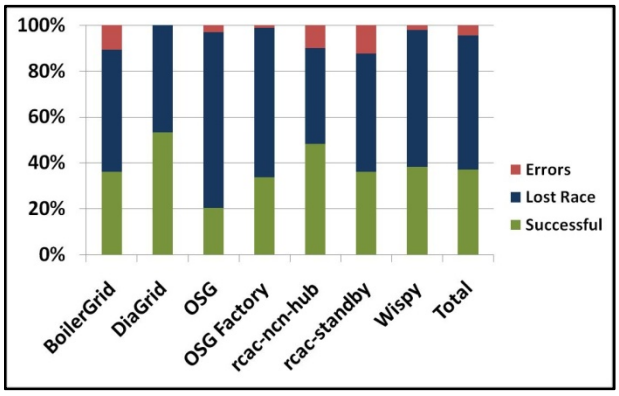

Figure 2. Visual representation of the numerical results of Table 3 on a relative scale.

In analyzing these data, DiaGrid experienced the highest success rate, with 53% of jobs submitted winning the race. However, this number is slightly skewed by the fact that there were times when the local queues and Wispy were not available, thereby reducing the number of resources available

to compete. It is not surprising to see that the dedicated rcac-ncn-hub queue, even with downtime, returned a high percentage of successful runs, with 48%. Not only is there essentially a zero wait time, but it is a faster, newer machine than Wispy (see Table 2). After the dedicated queue, Wispy had the next highest success rate at 38%. It was followed closely by BoilerGrid and our local rcac-standy queue, both at 36% success and finally OSG Factory at 34% success. It was expected that using OSG Factory Glidein would result in faster completion time than OSG, since, once a machine is available at one of their sites, the hold on the resource continues. This expectation was verified by the data, with OSG Factory having a significantly higher success rate than regular OSG sites, which had a 20% success rate.

Figure 3 shows the average wait times plotted together with the CPU and wall times for jobs executed at each venue. The shortest wait times are found on Wispy, rcac-standby, and the dedicated rcac-ncn-hub queue. One would expect the dedicated queue as well as the cloud test bed would have short wait times. For the cloud, wait times should typically only consist of the time needed to start up new virtual machines. As long as the cloud is kept busy, that time is kept to a minimum. The standby queue also had very low wait times. This could be attributed to low levels of outside job submissions. OSG had significantly longer wait times than the local resources, but lower than the other grids, perhaps due to the wide variety of sites available for submission. OSG Factory had an average wait time nearly double of OSG. BoilerGrid and DiaGrid also had wait times significantly higher than the Wispy and the local queues.

OSG shows the longest CPU time and wall time, which may account for its low success rate in the context of our tests. The local standby queue had the shortest average CPU and wall times, with all other sites exhibiting times in a similar range.

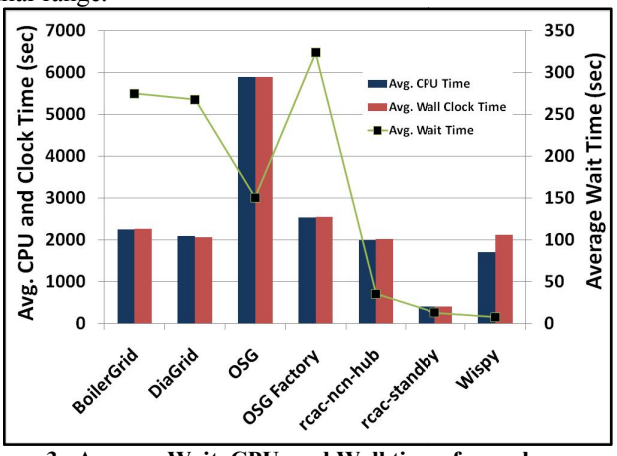

Figure 3. Average Wait, CPU, and Wall times for each venue

Figure 4 illustrates the utilization of the sixteen virtual machines on Wispy over the time of this study. The figure shows a blue line each time a machine is created, but this time is fairly short. Yellow represents an active but idle machine not currently running any jobs. A machine can remain idle up to 30 minutes before it is terminated. Magenta represents a machine running a single job and green represents a machine running the maximum of two jobs.

Zero usage between hours 56-214 and 234-312 corresponds to scheduled outages due to mechanical system upgrades at Purdue computing facilities. At both the beginning of the tests and the restart after the suspension, new VMs can be seen being activated when all existing VMs are busy with two jobs.

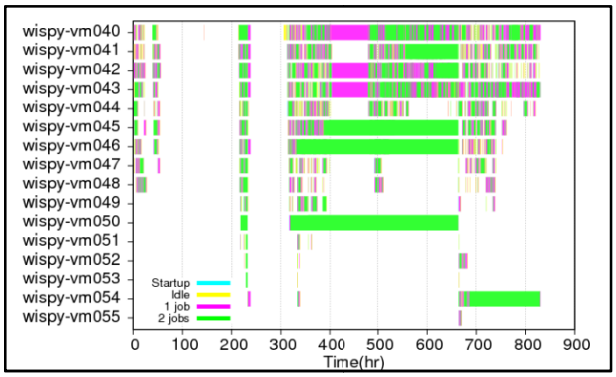

Figure 4. Loads on virtual machines for this experiment

Long stretches of green on VMs 041, 042, 045, 046, 050, and 054 are not indicative of a VM running longer than the 24 hour maximum lifetime, but rather an issue with the monitoring code. In some cases, the monitoring system did not recognize that a VM had finished and registered it as still in use. This problem has been corrected in the code. The only noticeable effect would be the spawning of new machines rather than reusing these existing machines if they were available. The three long patches of magenta near the top of the chart for VM040, VM042, and VM043 result from a problem with the OSG Factory software which caused job submission threads to essentially be put on hold and resulted in VMs left open due to subsequent submissions being unable to proceed until the problem was resolved.

For the most part, machine use appears to be fairly efficient, with each machine only being terminated a few times over the course of the tests and new VMs needing to be added to the full capacity of sixteen only a few times.

The results of these preliminary tests indicate that a cloud provider such as Wispy is competitive, for serial jobs such as the ones used in this experiment, with grid resources and even a dedicated machine. Wispy exhibited the lowest average wait time and had the second lowest average CPU time for successful runs out of all the venues.

V. FUTURE WORK

Expanding the use of computational cloud technology for a science gateway that has learned to build reliability into the user experience is a multi-faceted project. Any such effort involves testing, evaluating, building, documenting, and then making the transition to the cloud transparent to a user-base that is made up of scientists whose primary focus is their science, not the detailed process of submitting jobs, and certainly not the development of infrastructure.

The first step in this project was to create applications that could begin to test and to document the reliability and feasibility of this back-end infrastructure. The next phase of our work will evaluate different scheduling methods that enable access to the cloud and allow instant access from HUBzero-based environments. The next step involves the movement to more complex, multi-core job executions across multiple virtual machines. The long term goal will be to offer standard container sets for different combinations of compilers, operating systems, and OS versions, along with the appropriate methods to schedule these job runs. Another end goal is to provide a simple-enough configuration definition to allow the developer to declare the properties of each deployable VM. As we expand our understanding of the intricacies involved with sending nanoHUB jobs to the cloud, we will begin to explore the use of commercial large cloud vendor services such as Amazon's EC2 and evaluate real job performance in commercial clouds, just as we have evaluated real jobs sent to grid computing venues.

A review of our long-term goals for cloud access shows there are several concerns that will have to be monitored throughout the project. One of the main concerns is being able to automate enough of the standard computational cloud access container so as not to overload an over-allocated cyber infrastructure staff. In order to maintain this service for a large customer base, there must be a balance between developer autonomy for standard requests and being able to satisfy requests for custom builds. Another balancing act involves allowing scientific tool developers to request custom virtual machines for their individual requirements and being able to also immediately assess the security of the VM. Coinciding with these issues is the continual evaluation of related resource provisioning policies and the monitoring of the robustness of the job submissions. A historical view of this robustness will be maintained to continue to strengthen the selection of remote backend resources.

As nanoHUB.org delves into the evolving world of cloud computing, our highest goal is to increase the service offering and job flexibility while still maintaining the reliability and high user experience that the customer-base has grown to expect. Careful planning and execution of this project will be essential to meeting that goal.

VI. Conclusions

This paper presents preliminary results from our tests of using the "standard" computational cloud to supplement our use of grid resources for computationally intense simulation runs. Our initial results validate that one of the strengths of the cloud is for urgent, time-sensitive runs, whether or not other resources are available. However, dedicated or high priority queues can also provide similar performance. Grid resources where priority may be beyond the control of the submitter, were consistently outperformed by the cloud and by more dedicated resources.

Performance is only part of the equation. Cost and ease of submission must also be considered. For many research groups with existing infrastructures, costs for computing venues may have already been absorbed in terms of purchase and support costs paid. In those cases, the extra expenditure for cloud resources must be considered carefully. In addition, the amount of support time that is required for submissions to the various venues will play a big part in determining where to submit jobs. The winning venue may not be the fastest, but rather the venue that allows the Cyberinfrastructure Operation Team to most easily connect to and submit jobs on behalf of users, who are not computational scientists.

We have just begun to explore the potential of clouds for a science gateway site and the results are promising. There is clearly further work to be done in this area. The applications selected for this study were typical serial jobs. To truly test the capabilities of the cloud, more computationally intense runs need to be tested. Additionally, it might be informative to repeat these runs with a cloud venue under "stress." During our tests, the cloud was not highly utilized by other computations. We would like to stress the cloud to try to identify whether limits in its elasticity could potentially affect our runs.

The fields of utility, cluster, cloud, and grid computing continue to evolve quickly. It is to the benefit of a science gateway site such as nanoHUB.org to continually evolve our use of computational resources to best benefit our users, and this research clearly shows that a cloud strategy should be part of the equation.

We emphasize here that many of our users and supporters think of nanoHUB.org as a completely operational computational cloud computing environment that serves end users, rather than computational scientists who are interested in the computational cloud itself. "Standard" cloud computing infrastructures as they are discussed by the cloud computing community may turn out to be a very strong contributor to the 10,000+ end-user experiences on nanoHUB.org.

Acknowledgment

Mark S. Lundstrom founded nanoHUB.org in 1998. In 2005, Michael McLennan created the Rappture Toolkit and Rick Kennel wrote the scalable middle ware of HUBzero that, respectively, enable and power interactive nanoHUB simulations. The Network for Computational Nanotechnology (NCN) manages nanoHUB.org and is funded by NSF Award # EEC-0228390. The Wispy test bed is supported in part by NSF Award #1007115: ExTENCI: Extending Science through Enhanced National Cyberinfrastructure. We also gratefully acknowledge collaborations and support from the OSG VO team and the TeraGrid Science Gateway group.

References

[1] nanoHUB.org usage metrics: http://nanohub.org/usage

[2] M. McLennan and R. Kennell. "HUBzero: A platform for dissemination and collaboration in computational science and engineering," Computing in Science and Engineering, vol. 12, no. 2, pp. 48-52, 2010.

[3] Alejandro Strachan, Gerhard Klimeck, Mark S. Lundstrom, "Cyber-Enabled Simulations in Nanoscale Science and Engineering," Computing in Science and Engineering, Vol. 12, pp. 12-17, 2010, doi:10.1109/MCSE.2010.38.

[4] Gerhard Klimeck, Michael McLennan, Sean B. Brophy, George B. Adams III, Mark S. Lundstrom, "nanoHUB.org: Advancing Education and Research in Nanotechnology," IEEE Computers in Engineering and Science (CISE), Vol. 10, pp. 17-23, 2008, doi:10.1109/MCSE.2008.120.

[5] Gerhard Klimeck, George B. Adams III, Krishna P. C. Madhavan, Nathan Denny, Michael G. Zentner, Swaroop Shivarajapura, Lynn K. Zentner, and Diane L. Beaudoin, "Social Networks of Researchers and Educators on nanoHUB.org", proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2011, Newport Beach, CA, USA, May 23-26, 2011. DOI: 10.1109/CCGrid.2011.33.

[6] D. Gannon, "Programming E-Science Gateways," Making Grids Work, vol. IV, 2008, pp. 191-200, DOI: 10.1007/978-0-387-78448-9_15.

[7] I. Foster, Yong Zhao, I. Raicu, S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop (GCE '08), 12-16 Nov. 2008, pp. 1-10, doi: 10.1109/GCE.2008.4738445.

[8] S. Jha, D. S. Katz, A. Luckow, A. Merzky, and K. Stamou, "Understanding Scientific Applications for Cloud Environments," Cloud Computing: Principles and Paradigms, 2011, John Wiley & Sons, Inc., pp.345-371.

[9] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. 2008. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.* 39, 1 (December 2008), 50-55. DOI=10.1145/1496091.1496100 http://doi.acm.org/10.1145/1496091.1496100

[10] Amazon Elastic Compute Cloud (Amazon EC2): http://aws.amazon.com/ec2/

[11] A.G. Carlyle, S.L. Harrell, P.M. Smith, "Cost-Effective HPC: The Community or the Cloud?," 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), Nov. 30-Dec. 3 2010, pp.169-176, doi: 10.1109/CloudCom.2010.115.

[12] P. Marshall, K. Keahey, T. Freeman, "Elastic Site: Using Clouds to Elastically Extend Site Resources," 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 17-20 May 2010, pp.43-52, doi: 10.1109/CCGRID.2010.80.

[13] Marcos Dias de Assunção, Alexandre di Costanzo and Rajkumar Buyya, "A cost-benefit analysis of using cloud computing to extend the capacity of clusters," Cluster Computing, vol. 13, no. 3, April 2010, pp. 335-347, DOI: 10.1007/s10586-010-0131-x.

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

[15] S. Akioka and Y. Muraoka, "Hpc benchmarks on amazon ec2," International Conference on Advanced Information Networking and Applications Workshops, 2010, pp. 1029–1034.

[16] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, "Scientific workflow applications on amazon ec2," Workshop on Cloud-based Services and Applications in conjunction with 5th IEEE Internation Conference on e-Science (e-Science 2009), December 2009.

[17] TeraGrid: https://www.teragrid.org/

[18] Open Science Grid (OSG): http://www.opensciencegrid.org/.

[19] DiaGrid: http://www.dia-grid.org/.

[20] L. K. Zentner, S. M. Clark, K. P. C. Madhavan, S. Shivarajapura, V. Farnsworth, and G. Klimeck, "Automated Grid Probe System to Improve End-To-End Grid Reliability for a Science Gateway," TeraGrid 2011, Salt Lake City, UT, July 18-21, 2011.

[21] OSG Joint Task Forces: https://twiki.grid.iu.edu/bin/view/VirtualOrganizations/JointTaskForces.

[22] Purdue Cloud Computing Testbed "Wispy": http://www.rcac.purdue.edu/teragrid/resources/.

[23] Science Clouds: http://scienceclouds.org/

[24] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux virtual machine monitor," Ottawa Linux Symposium, 2007, pp. 225-230, http://www.kernel.org/doc/ols/2007/ols2007v1-pages-225-230.pdf.

[25] Hong-Hyun Park; Lang Zeng; Matthew Buresh; Siqi Wang; Gerhard Klimeck; Saumitra Raj Mehrotra; Clemens Heitzinger; Benjamin P Haley (2011), "Nanowire," DOI: 10254/nanohub-r1307.8. .

[26] M. P. Anantram; Shaikh S. Ahmed; Alexei Svizhenko; Derrick Kearney; Gerhard Klimeck (2011), "NanoFET," DOI: 10254/nanohub-r1090.6.

[27] Neophytos Neophytou; Shaikh S. Ahmed; Eric Polizzi; Gerhard Klimeck; Mark Lundstrom (2011), "CNTFET Lab," DOI: 10254/nanohub-r1091.7.

[28] BoilerGrid: http://www.rcac.purdue.edu/userinfo/resources/boilergrid/.

[29] OSG Glidein Factory: http://hepuser.ucsd.edu/twiki2/bin/view/UCSDTier2/OSGgfactory.

[30] Diane Lingrand, Johan Montagnat, and Tristan Glatard. "Modeling user submission strategies on production grids," *Proceedings of the 18th ACM international symposium on High performance distributed computing* (HPDC '09), 2009, pp. 121-130, http://doi.acm.org/10.1145/1551609.1551633.