High Performance Computing in a Cloud Using OpenStack

Roman Ledyayev and Harald Richter
Institute for Informatics, Clausthal University of Technology
Clausthal, Germany
Email: {roman.ledyayev, hri}@tu-clausthal.de

***Abstract*—Cloud computing is a rapidly gaining popularity computing paradigm, prompted by much research efforts. However, not much work is done in the area of joining cloud computing with high performance computing in an efficient way, e.g., for scientific simulation purposes. Moreover, there is even less research effort in making cloud computing for scientific simulations more efficient and suitable for specific simulation codes. This paper presents an ongoing "SimPaaS" project – our research efforts in building a cloud based platform for scientific simulations. It deals with some challenging features in cloud computing, such as performance. The concepts and methods proposed in this paper allow customizing and optimizing cloud infrastructure to increase its performance and to meet certain requirements, drawn from the analysis of case study simulation codes. Even though clouds are not the most suitable environments for high performance computing, the conclusion was drawn that there are ways to increase cloud performance and effectively combine the two paradigms.**

Keywords-simulation; cloud computing; High Performance Computing; OpenStack; VM

I. INTRODUCTION

In a project, entitled Simulation Platform as a Service (SimPaaS) [34], we attempt to marry two technology domains: cloud computing and scientific simulations, sometimes referred to as scientific computing or High Performance Computing (HPC). In this introduction, a brief definition of these two technologies in context of the ongoing research project is presented. Note, that in this paper, the terms scientific simulations and High Performance Computing are used interchangeably.

There are multiple definitions out there of the cloud [1]. In the project under consideration, the definition provided by the National Institute of Standards and Technology (NIST) [2] was adopted and relied on.

Cloud computing is a relatively new computing paradigm, composed of a combination of grid computing and utility computing concepts. Cloud promises high scalability, flexibility, cost-effectiveness, power savings and various other benefits to satisfy ever emerging computing requirements of modern applications.

The scalability, flexibility, cost-effectiveness and relative user-friendliness of various cloud services make it also an attractive model to address computational challenges in the scientific community. Individual research groups, who decide not to build their own cloud environments, do not need to provide and maintain IT-infrastructure on their own, but instead rely on cloud-computing services to satisfy their needs. However, they can also build their own specialized cloud services, which can be implemented on-site, and which could enable them to customize and optimize their cloud utilization specifically for scientific simulations.

High Performance Computing is an important field with two branches. These are: numerical simulations and big data analysis. The latter is well suited for clouds, because of the distributed file systems available in clouds. Massive amounts of data can be stored in a distributed file system and subsequently processed by individual cloud nodes. However, up to now it is an unsolved problem for numerical simulations to run efficiently on a cloud, because clouds, by nature, are distributed systems and thus based on TCP/IP communication. TCP/IP, in turn, has no quality of service with respect to bandwidth and latency, thereby creating much variance in both key parameters. As a result, exchanging data with high bandwidth and low latency becomes dependant on the traffic in the Internet. On the other hand, HPC is a numerical intensive task, based on highly efficient Inter-Process Communication (IPC). It is difficult to join both worlds, clouds and HPC, because in parallel and multicore computers, which are the hardware basis for HPC, interprocess, interprocessor and intercore communications are highly optimized. Additionally, clouds are intensively using the concept of virtualization, which results in computing overheads, as seen from the HPC's point of view. As a consequence, a lot of CPU power is not used for executing HPC codes, but to run the cloud operating system, such as OpenStack [3].

In this paper, a set of methods that can transform OpenStack into a middleware, able to accommodate HPC, is presented. The proposed method set is based on a mixture of hardware and software changes.

The ultimate goal of the project is to provide a cloud-based software platform for scientific simulations (Simulation as a Service). Figure 1 shows how such service would fit in the cloud stack. SimPaaS project prototype cloud will implement a platform on top of Infrastructure as a Service (IaaS), which provides virtualized resources for automatic distributed and scalable deployment. The specific simulation applications can be implemented as Software as a Service (SaaS) on top of the simulation cloud.

 ISBN: 978-1-61208-338-4

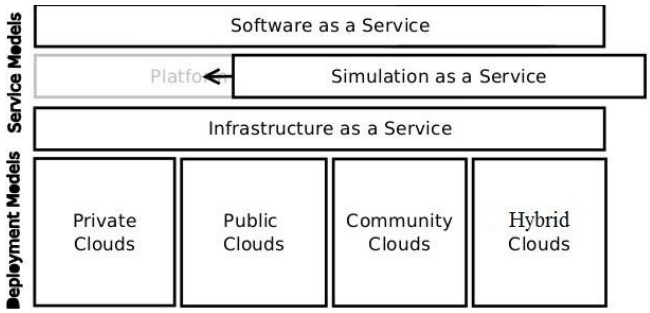

Figure 1. Simulation as a Service in the cloud stack

The rest of the paper is organized as follows. Section II presents related work. In Section III, various types of simulation codes and their impact on Inter-Process Communication and cloud efficiency are discussed and more details about the current test bed cloud setup are presented. Next, Section IV presents the proposed method set of making the current cloud suitable for High Performance Computing. Finally, some preliminary conclusions and directions for future research are outlined in Section V.

II. Related Work

There are projects and research articles that have made fairly successful attempts to introduce scientific simulation into the area of cloud computing on various levels and to various degrees.

First, we must admit that significant research efforts [4] have been made to migrate scientific computing simulations to the cloud. Several science clouds [5] have been established. There have also been attempts to design distributed computing frameworks, which would fully support scientific computing algorithms and take advantage of those characteristics of a cloud that have made it such a convenient and popular source for utilizing computing resources [6]. Several simulations and applications have been executed on these hybrid clouds. Research teams have been able to measure the performance of running those simulations in the clouds, thus evaluating the efficiency of scientific computing on the cloud in general [33]. Jakovits et al. [6] drew a conclusion that clouds are perfect environments for scientific simulations. It was observed that the communication, interference and other latencies added by the virtualization technology are the major hindrances for executing scientific computing applications on the cloud [4][7][9]. The cloud computing community is trying to address these issues and a number of interesting solutions have been proposed over the last few years [7][8].

Even though virtualization is to be taken into account when using clouds for HPC, some studies show, that for running scientific codes in parallel, performance is comparable, indicating that the virtual machine hosting environment introduced little overhead, even when all of the available cores were running at full capacity [34].

There have also been projects that used OpenStack to build and manage a scientific cloud. One such project worth mentioning was called Applied Computational Instrument for Scientific Synthesis (ACISS) [10]. Some objectives of the ACISS project overlap with our own goals for SimPaaS project [10].

Second, even though there has been much research in cloud computing and related technologies, comparatively little work has focused on their use in simulation, especially parallel and distributed simulation. Execution of parallel and distributed simulations over clouds not only represents an interesting opportunity, but also presents certain technical challenges, as discussed by Fujimoto et al. [9].

However, it is clear that significant further developments are needed to create a platform for materials simulations that meets all the particular needs of HPC without requiring further configuration and is accessible not only to system administrators but also to general public users. Furthermore, the questions of cost-effectiveness and performance have not been conclusively answered and need to be addressed for each type of scientific cloud application. In particular, concerns about cloud computing performance are strong in the materials science community. Jorissen et al. [11] shows that Scientific Cloud Computing (SCC) is especially appropriate for materials science and quantum-chemistry simulations, which tend to be dominated by computational performance rather than data transfer and storage.

Despite the number of cloud computing research projects mentioned above, which deal with cloud computing and scientific simulation, there have not been, to the best of our knowledge, very many efforts to design and implement techniques to address specific performance requirements and optimize the overall resource utilization of simulation applications run in a cloud.

III. Current Setup and Simulation Applications

This section describes the current setup of the cloud test bed based on OpenStack, the choice of tools and gives brief information about the applications used in the experimental research (simulation case studies).

A. Scientific Cloud Based on OpenStack

OpenStack, co-founded by Rackspace and NASA in 2010, is quickly becoming one of the most popular open source cloud computing platforms. According to its mission statement, the OpenStack developers strive to produce the platform that will be simple to implement and massively scalable [12]. Admittedly, there are a number of alternatives to OpenStack, both in the open source and commercial arena. Eucalyptus [13], CloudStack [14], Joyent [15], OpenNebula [16] and proprietary-powered pioneers like Amazon Web Services [17] and VMware [18]. Our choice of OpenStack over these other platforms was motivated by a few factors. One of them is active development, which keeps it up to date with new releases every half a year. Other reasons are: less overhead, better scalability, and its open source nature. It has also been actively used by our partners – GWDG [19].

Figure 2 depicts a high level diagram of the current prototypical cloud setup. Grizzly 1.3 release of OpenStack was used and deployed on Ubuntu 12.04.3 LTS Precise 64 bit (kernel version 3.2.0-52-generic) as the operating system. Ubuntu with KVM are used as Hypervisor on one of the machines, which serves as cloud controller.

 ISBN: 978-1-61208-338-4

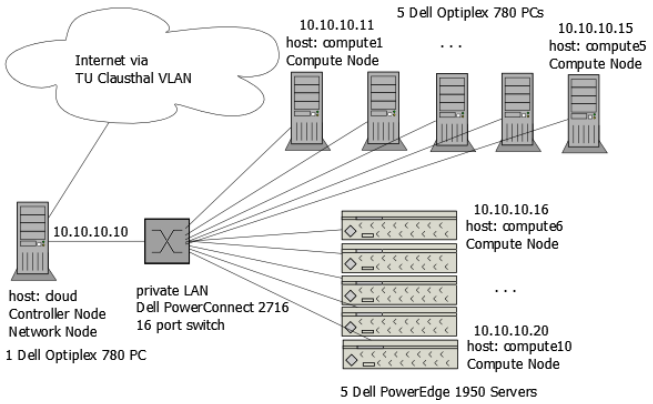

Figure 2. Current setup of the prototypical cloud

To measure the performance of the cloud, Ganglia [20] monitoring system (Ganglia Web 3.5.10) was installed on each node. It measures CPU, memory and network performance and loads. At the moment, we also consider automation tools like Puppet modules [21] and attempt to integrate the prototypical cloud with other research projects, such as Cloud4E project [22].

B. Simulation Applications

The primary focus was made on simulation applications in materials science, high energy physics and public transport networks.

1) Modelling and Optimization of Public Transport Networks

These are mathematical solvers for planning and optimization of public networks, which are running on multi-core platforms. The software used: Xpress and Mosel [23], LinTim [24].

2) High Energy Physics Monte Carlo Simulation and Data Analysis

This is both computational intensive and data-intensive simulation and data analysis. These applications are using data-parallelism and there is no communication between the processes. The software used: ROOT/PROOF [25].

3) Material Simulation

Open Source Field Operation and Manipulation (OpenFOAM [26]) is used to perform simulations in computational fluid dynamics (CFD). It covers the whole spectrum of the machines on which it can be compiled and run, from single-core and multi-core (e.g., Intel) to parallel computer (e.g., HLRN supercomputer [27]) and Graphical Processing Units (GPUs) which are currently in progress. The default, however, is parallel computer. Currently, only MPI library (OpenMPI) is used for message-passing.

C. Initial Simulation Test Results

In Figure 3, some preliminary simulation test results are presented. OpenFoam's "breaking of a dam" use case was selected for test runs, using the same configuration files (mesh size, decomposition). The simulation was executed on both, dedicated physical machines and VMs in the cloud.

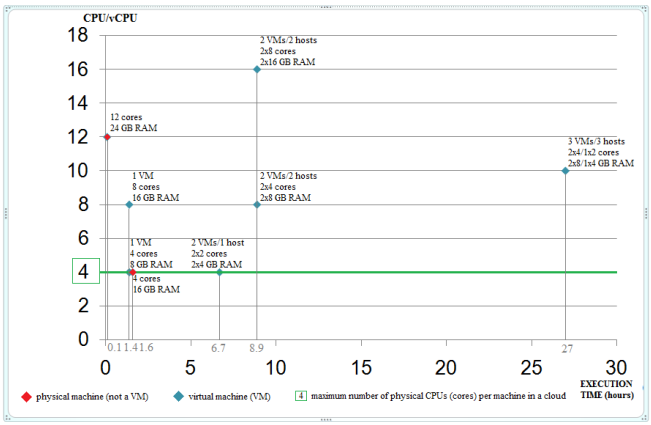

Figure 3. Initial OpenFOAM simulation test results

It was observed, that increasing the number of VMs and/or the number of hosts on which those VMs are run, drastically increases the time necessary to complete the simulation. It is also worth noting, that increasing the number of virtual cores per VM, even if that number is above the number of the physical cores available, had no visible impact on performance and provided no speedup in execution time.

Thus, some preliminary conclusions could be drawn from these test results. First of all, virtualization does not seem to have any significant impact on performance when 1 VM is used and there is no overcommitting in the number of CPUs (the number of virtual CPUs is no larger than the number of physically available CPUs). However, increasing the number of VMs significantly increases simulation execution time. This is especially noticeable if those VMs are run on two or more separate hosts. In such cases, the simulation is executed in parallel and in distributed mode, as opposed to cases when it is run on one VM (non-distributed). The fastest execution time was achieved when simulations were run on 1 VM with the number of virtual cores equal to the number of physical cores of the physical computer on which it was run. In this case, the execution time was almost equal to the time it takes to run the same simulation on a physical computer outside of the cloud.

Independent of the used programming paradigm and library, simulation codes can be categorized with respect to their interprocessor communication. These categories are important for understanding the measures proposed.

D. Types of HPC Simulation Codes

1) Category 1: Multiple Runs of a Sequential Code

In this case, a sequential code has to be executed multiple times but with different input values respectively, which is called parameter sweep. In theory, a cloud can achieve high throughput here, because all sweeps can be executed in parallel by different virtual cores on the same Virtual Machine (VM) or by different VMs. Inter-Process Communication is not needed, with the exception of the beginning and the end of the runs, where some master node

 ISBN: 978-1-61208-338-4

must distribute the set of input values to computing nodes and, subsequently, collect their results.

2) Category 2: Single Run of a Parallel Code

The decisive question in this case is how much IPC is needed in the considered code, because IPC is the bottleneck in a cloud compared to a parallel computer or a multi-core CPU. Fortunately, if the code is data parallel and not function parallel then IPC is mostly moderate. Most solvers for partial differential equations, for example, belong to data parallel class. However, if the code is function parallel then standard clouds are not a good choice, because a high fraction of IPC and an intricate pattern of communication are normally present. Only a significant improvement of the cloud's inter-core and/or inter-VM bandwidth and latency could help, which is suggested in the next section.

3) Category 3: Multiple Runs of a Parallel Code

This is the combination of the two cases discussed above, which means that respective problems and prerequisites are also combined. To avoid slowdowns, either cloud nodes must be upgraded as described below or the scheduler must be modified and the cloud's IPC must be made more effective. A code written for a parallel computer with specific features should be executed on a machine in a cloud with the same properties, if possible. However, OpenStack does not guarantee that a specific real machine is indeed chosen for a given VM. Therefore, changes must be made to address this shortcoming as well.

IV. METHODS OF INCREASING CLOUD PERFORMANCE

The method set proposed in this paper can be divided into two categories of hardware and software changes.

A. Hardware Changes

In OpenStack, it is a common practice that heterogeneous computers are configured to be computing or storage nodes, together with a controller node. These nodes and the controller are coupled on ISO layers 1 and 2 by a Gigabit Ethernet switch. Optionally, a VLAN can also be established. The hardware proposed here uses a 10 Gigabit switch and proper network cards in 10 Gigabit Ethernet or Infiniband technology in a way that a Beowulf cluster comes into existence. Beowulf cluster implies that homogenous computers are coupled with identical network interfaces and with high-speed switches, such that deterministic latency and bandwidth are achieved on ISO layers 1 and 2. Additionally, the TCP/IP protocol set has to be abandoned, because it is slow and non-deterministic and because a Beowulf cluster is not world-wide and distributed but localized in a computing center. As a consequence, most of the TCP/IP features are not needed. However, in order to maintain compatibility with existing MPI and OpenMP implementations, the Berkeley socket API must be preserved. This is possible by employing the commercially available "VMA Messaging Accelerator Protocol" [28] from Mellanox. It provides the Berkeley API but bypasses the TCP/IP protocol stack and writes user data directly into the Mellanox network cards and reads input from them without much protocol overhead. So, changes in the user codes are not needed.

1) Unnecessary TCP/IP functions in Beowulf cluster

Data transmission errors are practically excluded in the described cluster, because of the relatively short distances between switches and nodes. As a consequence, the automatic packet retransmission of TCP is not needed. Additionally, the TCP sliding window protocol is unnecessary, because all network interfaces are identical and have the same speed and buffer sizes. Furthermore, the packet reorder function of TCP is also not needed, because only point-to-point connections without routers exist. No packet that is sent later can arrive earlier. Additionally, IP packet segmentation and reassembly is also not necessary, because there is the same LAN technology in usage everywhere, without differences in maximum frame size. Finally, IP routing is not useful here, because there is no router but switches in the cluster. As a consequence, nearly all functions of the TCP/IP stack can be dismissed, as it is done by the VMA messaging accelerator, which boosts bandwidth and drastically reduces latency.

2) Bandwidth and latency improvements

Mellanox claims that their accelerator reduces latency on a 10 Gigabit Ethernet to 1.6 µs between two sending and receiving Berkeley socket APIs in case of 12 bytes payload per frame. This is significantly faster than via the Internet, but it means that the compute and storage nodes of the cloud are no longer part of the Internet. Only their controller node can stay connected. However, this is fully compatible with concepts used in OpenStack, which allows using floating IP addresses for compute and storage nodes that must overlap with publicly used addresses. Additionally, the network service of OpenStack can be exclusively localized in the controller node which has Internet connection.

Suggestions could be made to fix bandwidth and latency issues with hardware devices such as fiber optic networking, ramdisks/SSDs, etc. The substitution of copper cables as computer interconnects by glass fiber optics can fix the bandwidth problem only if fiber speeds are significantly higher than 10 GB/s, which is the limit for copper. However, it will not fix the latency problem, because the electric/optic converters introduce additional delays and because the speed of light is similar in glass and in copper (about 0.75c). The replacement of hard drives in the cloud by SSDs could accelerate the throughput of some OpenStack services. But, as with the glass fiber solution, the cloud costs would significantly increase, which is not desirable in this case.

3) Hardware scalability

The hardware scalability, necessary to engage thousands of CPUs in one cluster, is achieved by a hierarchical cascade of layer 2 switches and by employing VLAN technology. This allows enlarging the spatial circumference of the cloud to several kilometers, which is sufficient to accommodate thousands of computers. There should be no problem with scalability, since Mellannox 10 Gigabit Ethernet switches are supported by OpenStack via using Mellanox plugins.

B. Software Changes

Software changes have to be made in the underlying operating system (Linux Ubuntu), in the OpenStack network service (Neutron) and its scheduler.

 ISBN: 978-1-61208-338-4

1) New operating system for OpenStack

The first software change is to replace Linux host OS, on which OpenStack runs, by a Real-Time Linux such as the RT Preempt Patch [29] or Xenomai [30]. The reason for that is that standard Linux, such as Ubuntu, uses the Completely Fair Scheduler from Ingo Molnar, which does not allow prioritizing processes as real-time operating systems do. The selected RT OS is used not only as host OS for OpenStack, but also as guest OS for every VM. This gives users a precise control over which task is scheduled at which time. Such a feature is important when two user tasks want to exchange data at the same time, because of Inter-Process Communication.

2) New OpenStack scheduler

The Nova scheduler of OpenStack determines which physical compute node is allocated to which VM as soon as the VM is provisioned. This reflects a scheduling in space, but not in time, and is of static nature. An automatic live migration of VMs that periodically balances out the load between physical nodes does not exist in the Grizzly release. This is not favorable for HPC, because resources may become extremely overloaded.

a) Periodic live migration

Because of the fact that an RT OS was chosen for both, host and guest OS, it is possible to re-allocate VMs for load balancing, because Nova can be prioritized before user tasks and executed periodically. To achieve this, scheduling in host and guest OS must be synchronized, so that all schedulers act together.

b) Gang scheduling

Schedulers must schedule all sets of communicating VMs simultaneously, as soon as they are starting data exchange, so that they can send and receive information efficiently (Gang Scheduling). Otherwise, unnecessary waiting times would be the consequence, because rendezvous between senders and receivers cannot take place. In that case, sent data must be buffered until receivers are ready to proceed and receivers must wait until senders can send data, which is unfavorable for HPC.

c) Advanced reservation

Accessing a hard drive requires up to 10 µs until the searched record is found. Such I/O delays are unbearable for HPC, especially if they occur one after another in physical nodes that must communicate with each other. Advanced reservation, in this case, means that user code can be instrumented with an OpenStack call that is directed to the storage services, Cinder and Swift, and that read and cache or cache and write a whole data object in advance before the hard drive is ready. This allows hiding I/O latencies and thus improves communication latency.

d) High priority scheduling

Some system processes, such as Nova itself or the Mellanox messaging accelerator, must be scheduled before user tasks. Consequently, user tasks must be rescheduled for high priority system tasks. Priority scheduling is a standard feature of all RT OS. However, for the synchronous cooperation of host and guest schedulers, a software framework must be created that instructs all schedulers via a common API, that should also be available in Open Cloud Computing Interface (OCCI) [31], which is both, a protocol and an API for all kinds of management tasks.

3) New OpenStack networking service

VLANs are needed for a scalable Beowulf cluster to extend cable lengths and cascade switches. To make this possible, the OpenStack networking service must be enhanced with the following functions:

- Generation of VLAN-IDs.
- Creation of a mapping table "VLAN-ID – Port number" for every switch, according to the cluster topology used, so that each switch can forward a frame to correct destination.
- Generation of IEEE 802.1Q Ethernet header tags at every switch input port and removal of these tags at every switch output port. This is needed for Ethernet interfaces that do not support VLAN tags.
- Defining which method is used for assigning a specific frame to its VLAN.
- Automatic setting of frame priorities. This is needed in cases when multiple frames collide at the same time at the same switch output, and a decision must be made by the switch which frame gets a passage first. This allows resolving conflicts in the transport system of the cloud under full control of the user code, thus avoiding speed up degradations.

V. CONCLUSION AND FUTURE WORK

The contribution of this paper is two-fold. In the first place, the possibility of effectively combining cloud computing, which by its very nature is not a very suitable environment for applications designed for HPC platforms, with High Performance Computing was examined. In the second place, a set of methods which, if followed and implemented, could make clouds more suitable for running HPC codes was proposed.

By running OpenFoam as a benchmark for OpenStack, we found out that this cloud operating system is not well-suited for OpenFoam, because it degrades performance with respect to a reference computer of the same capabilities that is outside of OpenStack. The results can be generalized to any cloud operating system, to any computational fluid dynamics codes and to any HPC codes in general.

We plan to investigate why this happens. At the moment, we believe this happens because of potential overcommitting of physical resources by virtual ones, and by data exchanges needed between VMs running on two computers and between virtual cores running in the same VM, as soon as it takes place via TCP/IP. With TCP/IP, parallel computing mutates into distributed computing which results in code slow down. However, three use cases with different slow down factors could be identified, and measures could be given to repair this behavior. These measures are: 1) Abandon the internal functions of TCP/IP, but preserve its Berkeley socket API, because of the often used MPI library. 2) Replace the distributed cloud by a Beowulf cluster and install OpenStack on it. 3) Replace guest and host operating

 ISBN: 978-1-61208-338-4

systems by real-time Linux. 4) Add a frame-work that schedules communicating VMs and cores synchronously. 5) Introduce disk I/O in advance for data objects to avoid unpredictable message latencies. 6) Add periodic live-migration of VMs for load balancing between physical CPUs.

Future work will be to investigate how much these measures can change the cloud's HPC efficiency under the boundary conditions of given user codes and cloud hardware.

Our future efforts will concentrate on further analysis of the issues mentioned above and the propositions described in the previous section, experimenting with the results of running the simulation codes inside the cloud, and, finally, designing and implementing one or more of the proposed solutions from the method set.

References

[1] A. Isherwood (Hewlett-Packard's Vice President of European Software Sales), quoted in ZDnet News, December 11, 2008.

[2] P. Mell and T. Grance, "The NIST definition of cloud computing", NIST Special Publication 800-145, September, 2011.

[3] http://www.openstack.org/, [accessed: January, 2014].

[4] S. Srirama, O. Batrashev, P. Jakovits, and E. Vainikko, "Scalability of Parallel Scientific Applications on the Cloud," Scientific Programming, vol. 19, Apr. 2011, pp. 91–105, 2011, doi:10.3233/SPR-2011-0320.

[5] S. N. Srirama, O. Batrashev, and E. Vainikko, "SciCloud: Scientific Computing on the Cloud," Proc. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010, pp. 579–580, doi:10.1109/CCGRID.2010.56.

[6] P. Jakovits, S. N. Srirama, and I. Kromonov, "Stratus: A Distributed Computing Framework for Scientific Simulations on the Cloud," Proc. IEEE 14th International Conference on High Performance Computing and Communications (HPCC 2012), IEEE Press, 2012, pp. 1053–1059, doi: 10.1109/HPCC.2012.154.

[7] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," Software: Practice and Experience, vol. 41, 2011, pp. 23–50, doi:10.1002/spe.995.

[8] Q. Li and Y. Guo, "Optimization of Resource Scheduling in Cloud Computing," Proc. 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2010), pp. 315–320, doi: 10.1109/SYNASC.2010.8.

[9] R. M. Fujimoto, A. W. Malik, and A. J. Park, "Parallel and distributed simulation in the cloud," SCS M&S Magazine, issue 3, July 2010, [Online]. Available: http://www.scs.org/magazines/2010-07/index_file/Articles.htm, [accessed: January, 2014].

[10] C. Hoge, "Building a scientific cloud computer with OpenStack," OpenStack Day, Portland, July 16-20, 2012. [Online]. Available: http://www.oscon.com/oscon2012/public/schedule/detail/24261, [accessed: January, 2014].

[11] K. Jorissen, F.D. Vila, and J.J. Rehr, "A High Performance Scientific Cloud Computing Environment for Materials Simulations," Computer Physics Communications, vol.183, issue 9, pp. 1911-1919, doi: 10.1016/j.cpc.2012.04.010.

[12] https://wiki.openstack.org/wiki/Main_Page, [accessed: January, 2014].

[13] http://www.eucalyptus.com/, [accessed: January, 2014].

[14] http://cloudstack.apache.org/, [accessed: January, 2014].

[15] http://www.joyent.com/, [accessed: January, 2014].

[16] http://opennebula.org/start, [accessed: January, 2014].

[17] http://aws.amazon.com/, [accessed: January, 2014].

[18] http://www.vmware.com/, [accessed: January, 2014].

[19] http://www.gwdg.de/index.php, [accessed: January, 2014].

[20] http://ganglia.sourceforge.net/, [accessed: January, 2014].

[21] https://wiki.openstack.org/wiki/Puppet-openstack, [accessed: January, 2014].

[22] http://www.cloud4e.de/, [accessed: January, 2014].

[23] http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Mosel.aspx, [accessed: January, 2014].

[24] http://lintim.math.uni-goettingen.de/index.php?lang=en, [accessed: January, 2014].

[25] http://root.cern.ch/drupal/content/proof, [accessed: January, 2014].

[26] http://www.openfoam.com/, [accessed: January, 2014].

[27] https://www.hlrn.de/home/view, [accessed: January, 2014].

[28] http://www.mellanox.com/related-docs/prod_acceleration_software/VMA.pdf, [accessed: January, 2014].

[29] https://rt.wiki.kernel.org/index.php/Main_Page, [accessed: January, 2014].

[30] http://www.xenomai.org/, [accessed: January, 2014].

[31] http://occi-wg.org/, [accessed: January, 2014].

[32] J.J. Rehr, F.D. Vila, J.P. Gardner, L. Svec, and M. Prange, "Scientific Computing in the Cloud," Computing in Science & Engineering, vol. 12, issue 3, pp. 34-43, 2010, doi: 10.1109/MCSE.2010.70.

[33] P. Saripalli, C. Oldenburg, B. Walters, and N. Radheshyam, "Implementation and Usability Evaluation of a Cloud Platform for Scientific Computing as a Service (SCaaS)," 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC), 2011, pp. 345-354, doi: 10.1109/UCC.2011.58.

[34] http://www.simzentrum.de/en/education/cloud-basierte-software-infrastruktur-fuer-verteilte-simulation/, [accessed: May, 2014].

 ISBN: 978-1-61208-338-4