Constructing Virtual Private Supercomputer Using Virtualization and Cloud Technologies

Ivan Gankevich, Vladimir Korkhov, Serob Balyan, Vladimir Gaiduchok, Dmitry Gushchanskiy, Yuri Tipikin, Alexander Degtyarev, and Alexander Bogdanov

St. Petersburg State University,
Universitetskii 35, Petergof, 198504, St. Petersburg, Russia
igankevich@ya.ru, vkorkhov@apmath.spbu.ru, serob.balyan@gmail.com,
{gajduchok,dgushchanskii,iutipikin}@cc.spbu.ru,
{deg,bogdanov}@csa.ru

Abstract. One of efficient ways to conduct experiments on HPC platforms is to create custom virtual computing environments tailored to the requirements of users and their applications. In this paper we investigate virtual private supercomputer, an approach based on virtualization, data consolidation, and cloud technologies. Virtualization is used to abstract applications from underlying hardware and operating system while data consolidation is applied to store data in a distributed storage system. Both virtualization and data consolidation layers offer APIs for distributed computations and data processing. Combined, these APIs shift the focus from supercomputing technologies to problems being solved. Based on these concepts, we propose an approach to construct virtual clusters with help of cloud computing technologies to be used as on-demand private supercomputers and evaluate performance of this solution.

Keywords: virtualization, supercomputer, virtual cluster, cloud computing.

1 Introduction

Virtual supercomputer can be seen as a collection of virtual machines working together to solve a computational problem much like a team of people working together on a single task. There is a known definition of personal supercomputer as a kind of metacomputer which was given in [1], however, in our approach virtual supercomputer is not only a personal supercomputer but it also offers a way of creating virtual clusters that are adapted to problem being solved and to manage processes running on these clusters (figure 1). This is the case where virtual shared memory cannot be used directly because of high latency and low performance caused by complex data transfer patterns. Migration of processes to data as well as methods of workload balancing [2] can solve this problem and form a basis of load balancing technique for a virtual supercomputer.

B. Murgante et al. (Eds.): ICCSA 2014, Part VI, LNCS 8584, pp. 341–354, 2014.
© Springer International Publishing Switzerland 2014

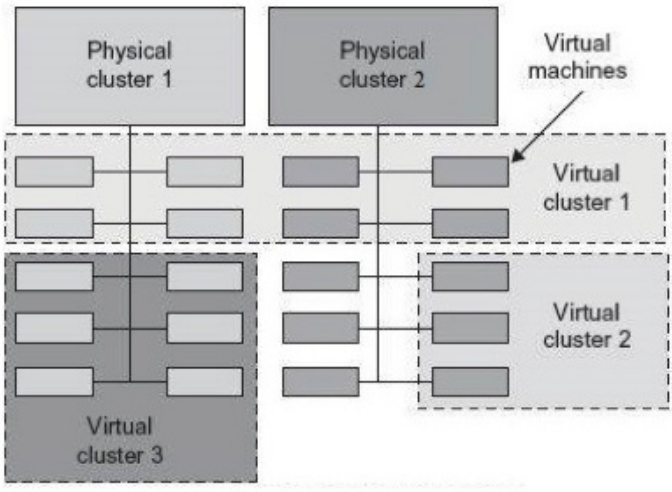

Fig. 1. A cloud platform example with three virtual clusters over two physical clusters

Computers like people need some sort of collective board to share results of their work and advance problem solution one step further. In a distributed computing environment distributed file systems and distributed databases act as such a board, storing intermediate and final results of computation. Apart from a shared desk people in a team need some sort of management to solve a problem in time and computers need a way of combining them into hierarchy helping efficiently distribute tasks among available computing nodes. Finally, from a technical point of view, problem solution should be decoupled from actual execution of tasks by a virtualization layer as not every problem has efficient mapping on physical architecture of a distributed system. So, virtual supercomputer is not only a cluster of machines but also virtualization and middleware layers on top of it.

There are many ways to construct such a supercomputer and it is time-consuming to compare and assess benefits of all technology combinations, however, it is convenient to tailor technologies to the needs of chosen problems and to show advantages of virtual supercomputer approach in these particular cases. Section 2 gives an overview of related work in this area. The chosen problems should be general enough to cover a wide area of potential applications. We selected two examples of such generic problems to be solved in personal supercomputer environment: ontology storage, retrieval and analysis involving use of a distributed database; fluid dynamics simulations involving execution of highly parallel code. These problems are discussed in Section 3. Corresponding virtual supercomputer configuration, its key principles are discussed in Section 4. Experimental evaluation of our solution is presented in Section 5. Section 6 concludes the paper and shows the directions of future work.

2 Related Work

One of the first approaches to construct clusters from virtual machines was proposed in [3] and partially realized in In-VIGO [4], VMPlants [5], and Virtual Clusters on the Fly [6] projects. In-VIGO focused on the end-to-end design of a Web service that could employ VMs as part of a cluster computing system, while VMPlants and Virtual Clusters on the Fly focused on rapid construction of virtual clusters. All three systems were particularly concerned with the issue of specifying and adapting to requirements and constraints imposed by the user.

Dynamic Virtual Clustering (DVC) [7] implemented the scheduling of VMs on existing physical cluster nodes within a campus setting. The motivation for this work was to improve the usage of disparate cluster computing systems located within a single entity.

The idea of an adaptive virtual cluster changing its size based on the workload was presented in [8] describing a cluster management software called COD (Cluster On Demand), which dynamically allocates servers from a common pool to multiple virtual clusters.

Grid architecture that allows to dynamically adapt the underlying hardware infrastructure to changing Virtual Organization (VO) demands is presented in [27]. The backend of the system is able to provide on-demand virtual worker nodes to existing clusters and integrate them in any Globus-based Grid.

The goal of current work is to investigate possibilities provided by modern cloud and virtualization technologies to enable personal supercomputing meaning creation of dedicated virtual clusters based on user's and application's requirements. Unlike many of other projects in this area, our intention is to use created clusters of VM as a single resource provided to a single parallel application but not generating sets of worker nodes provided to different applications separately.

3 Large-Scale Supercomputer Problems

3.1 Ontology Storage and Retrieval

One way of applying virtual supercomputer is graphs storage and processing. Transport logistics, articles citation or social networks are common examples of such tasks. In some cases graphs may have thousands or millions vertices and edges, as in Web graph related tasks [9]. That kind of structures can be handled by various types of algorithms such as shortest path computations, a special subgraphs allocation, different varieties of clustering etc. It is challenging to efficiently process large graphs since some graph's properties work poorly with high performance techniques [10].

- Parallelism based on partitioning of computation can be difficult to express because the structure of computations in the algorithm is not known a priori.
- The irregular structure of graph data makes it difficult to extract parallelism by partitioning the problem data. Scalability can be quite limited by unbalanced computational loads resulting from poorly partitioned data.

- Graphs can represent complex irregular relationships between entities thus it may provide the lack of locality for computations and data access patterns.
- Runtime can be dominated by the wait for memory fetches because usually graph algorithms are based on exploring the structure of a graph in preference to performing large numbers of computations on the data.

For the sake of effective handling large graphs require particular storage and processing tools: graph databases such as Pregel [11] or hypergraph oriented HyperGraphDB (http://www.hypergraphdb.org) [12]. They permit direct operation with a graph without any intermediate relational data representations. The tools support replication and distributed transactions hence make work with a graph size independent.

A special case of graphs is semantic network, which is considered a widespread method for knowledge representation. Due to this fact creation and processing of knowledge bases and ontologies constructed upon them may be seen as a another resource consuming task [13,14]. Such networks may not be as big as graphs related to Web graph problems in terms of numbers of elements, but they often have complex hierarchical relations between vertices and compound nodes and edges structure. This complicates the methods of their processing as the graph structure and graph data are interconnected. Knowledge extraction and ontology-based reasoning can be used as examples of such complex tasks.

Growing interest in ontologies development and processing generates demand for tools which are capable of handling complex operational problems on their own. Such tools have been created and already mentioned HyperGraphDB is one of them. HyperGraphDB implements OWL 2.0 standard of ontology representation with operating multiple ontologies in one database as subgraphs. Usage of subgraphs as the base allows representations of ontologies to use all benefits of distributive graph database. HyperGraphDB has an integration with Protege Editor, the most popular ontology editor, and permits using popular reasoners such as Hermit, Fact++ and Pellet. Thereby the database hides all the internal work and allows users to work with familiar tools.

3.2 Fluid Dynamics Simulations

Another way of applying virtual supercomputer is fluid dynamics simulations. This class of applications demands highly scalable architecture. In particular, experiments in a virtual testbed can be carried out on a single multiprocessor machine [15] only in the most simple cases involving small simulation region and time interval. However, large-scale simulations with multiple atmospheric and ship motion models involved require use of multiple machines comprising distributed computing system. Moreover, hierarchy of mathematical models and high number of dimensions of these models demand a way of organizing computations into a single distributed workflow [16], for example, WRF, Wavewatch3 and wind wave model. So, a capability of a virtual supercomputer to dynamically compose distributed pipelines can accelerate execution of experiments in a virtual testbed.

4 Principles of Virtual Supercomputer

Although virtual supercomputer can be implemented in many ways and using different combinations of technologies, there are some principles that such implementation is considered to obey. On one hand these principles arise from similarity of different technologies and their implementations, on the other hand the purpose of some principles is to solve problems inherent to existing general-purpose distributed systems. In any case, the principles are useful for solving large-scale problems on virtual supercomputer and some of them can be neglected for problems of small sizes. So, the principles follow.

- Virtual supercomputer is completely determined by its application programming interface (API) and this API should be platform-independent. The use of API as the only interface in distributed processing systems is common, but its dependency on operating system or programming language leads to problems in the long run. For example, the first API for portable batch systems (PBS) was implemented in low-level C language and only for UNIX-like platforms which led to inability or inefficiency of its usage in other programming languages and in exposing it as a web service. Moreover, the API does not cover all the functions of underlying PBS [17]. So, using platform-independent API is one of the ways to avoid such integration and connectivity problems. In other words, API is a programming language of a virtual supercomputer and the only way of interacting with it.
- Virtual supercomputer API provides functions to connect with other virtual supercomputers and such interaction is seamless. Interaction of different distributed systems is the way of solving large-scale problems [18] and seamless interaction helps compose hybrid distributed systems dynamically: to extend capacity when needed [21]. So it is the way of scaling virtual supercomputer to solve problems that are too complex for one virtual supercomputer.
- Virtual supercomputer processes data stored in a single distributed database and this processing is done using virtual shared memory. Efficient data processing is achieved by distributing data among available nodes and by running small programs (queries) on each host where corresponding data resides; this approach helps not only run query concurrently on each host but also minimizes data transfers [11,19]. However, in existing implementations these programs are not general-purpose: they are parts of algorithm and they are specific to data model this algorithm was developed for. For example, in MapReduce framework programs represent map and reduce functions that are run on each row of table (or line of file) and it is difficult to compose general-purpose program to process any data within this framework [19]. On the other hand, virtual shared memory interface allows processing of data located on any host [20] and does it in efficient way. So, distributed database is a way of storing large data sets and virtual shared memory is a way of writing general-purpose program to process it.
- Experiments show that using paravirtualization instead of full virtualization is advantageous in terms of performance [22] and virtual computing nodes

should be created using paravirtualization technologies only. However, not every operating system can be paravirtualized and it should be possible to access virtual supercomputer facilities through fully-virtualized hosts. So, paravirtualization is inevitable in achieving balance between good performance and ease of system administration in distributed environment and as a consequence operating system should be UNIX-like for paravirtualization to work.
- Load balance is achieved using virtual processors with controlled clock rate and process migration. The first technique allows balancing coarse-granularity tasks and the second is suitable for fine-grained parallelism.
- Virtual supercomputer uses complex gridlike security mechanisms. One of the cloud problems is security issues [23] but we feel that proper combination of GRID security tools with cloud computing technologies is possible.

To summarize, virtual supercomputer is an API offering functions to run programs, to work with data stored in a distributed database and to work with virtual shared memory and this API is the only programming language of a virtual supercomputer.

5 Experimental Evaluation

5.1 Experimental Setup and Evaluation of Virtualization Impact

Implementation of a virtual supercomputer will not be possible without use of server virtualization technologies: virtual machine migration provides load-balancing and fault-tolerance capabilities and it is necessary to evaluate their performance relative to physical machines.

We conducted most of our experiments at Resource Centre Computer Centre of SPbSU [25]. This centre offers a specific approach to manage resources. Each user is given a virtual machine with necessary characteristics. Such a machine can be flexibly customized since user is granted administrative rights. When resources of a single virtual machine become insufficient to meet all user requirements, they can be easily extended, or even additional VMs can be created in order to form a virtual cluster. This is how dynamic allocation of computational resources is carried out. Further on, access to HPC resources is provided via this personal virtual machine.

In the same way virtual clusters can be created automatically when more resources are needed and processes are dynamically migrated to newly constructed virtual machines. Automated creation of virtual machines can be achieved by rewriting PBS prologue script where desired properties of virtual machines can be described in the same way as they are described when submitting PBS job to cluster of physical machines, however in case of virtual machines operating system can also be specified. So, there is no difference from a user point of view whether a job is submitted to real or virtual cluster.

Alternatively, user can run jobs on dedicated HPC clusters. In case of our resource centre they are T-Platforms cluster and HP cluster. User home directory

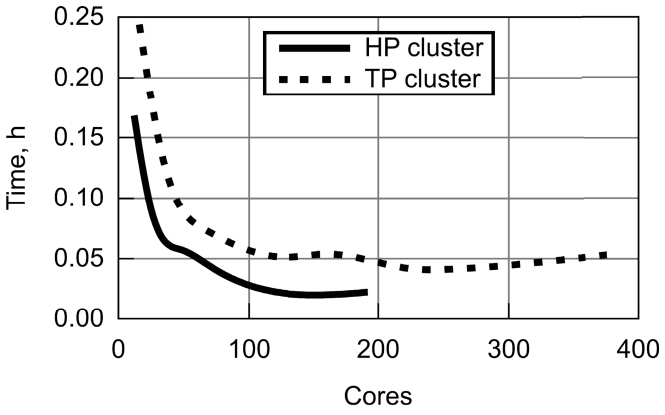

Fig. 2. Performance of clusters with different interconnect bandwidth based on GROMACS workload

is mounted via NFS on clusters. It provides universal access to computational data: raw data and results are stored in a single place.

We chose GROMACS as an example of real application running on clusters as it is commonly used by the users of St.Petersburg University Computer Center. GROMACS is used for efficient molecular simulations [26]. Figure 2 illustrates the GROMACS runs (2 different tasks) on T-Platforms (maximum 376 CPU cores were used) and HP (maximum 192 cores were used) clusters. Picture shows that these tasks have different scalability on different clusters. Without going into details, we can say that the root causes of this behavior is network bandwidth (HP has twice as much better network), memory size (swapping to disk substantially increase run time; HP cluster has 96 GB RAM per node while T-Platforms has only 16 GB) and intensive communication between worker processes.

But what can user do when network communication prevents scalability? The right way is using multicore SMP machine with large amount of memory. Computer centre has 3 machines of this type. In usual case in order to harness such a machine user has to migrate his applications, environment and data. In our case virtual machine is migrated to SMP node. It can be done with ease and it solves many problems: user does not need to do any actions, even get accustomed to new environment because his tuned virtual machine is completely migrated to powerful physical machine, and all applications, libraries and user settings remain unchanged. So, virtual machine migration is another way of extending dynamic computational resource pool.

Moreover, the resource pool can contain different accelerators. Contemporary hypervisors have means to harness available GPUs in a system. One can dedicate up-to-date GPUs supporting GPGPU to virtual machines, that is, to provide users with GPGPU computations. Physical GPUs can be shared between several virtual machines. Several GPUs can be utilized by hypervisor.

Graphics processors supporting GPGPU can be seen as vector co-processors. If a task can be parallelized using SIMD approach, it can be computed fast on GPU. Of course, there are some limitations, and the most important one is data transfer between GPU and CPU (it becomes a bottleneck). That is why some tasks with intensive data transfer will not reach estimated speedup. But one can test necessary application by adding GPU to virtual machine and running the application. If test shows good results, GPU remains in the VM configuration. Otherwise this task should be computed on CPU and GPU can be removed from the VM. So, virtual supercomputer can be upgraded by including modern GPUs. That is how new promising GPGPU technology can be used within virtual systems.

Virtualization leads to substantial benefits when using it in a big computing center [28], but what is about conventional PC? We used such a computer for additional tests. It has 2 Intel Xeon E5410 CPU (total 8 cores), 8 GB RAM, 250 GB HDD. Such systems become ubiquitous today. Xen technology was used for virtualization. We created paravirtualized guests. Both the host and the guest systems run Debian 7.0. We were interested in testing such a PC with practical workloads. We tested a GROMACS workload that puts heavy load on CPUs. We tried to run this job on the host system without virtualization and on the guest paravirtualized OS. After series of tests we can say that in our case (paravirtualized guest using Xen) virtualization led to 5% time overheads only (Figure 3), so virtualization offers benefits even on conventional PC.

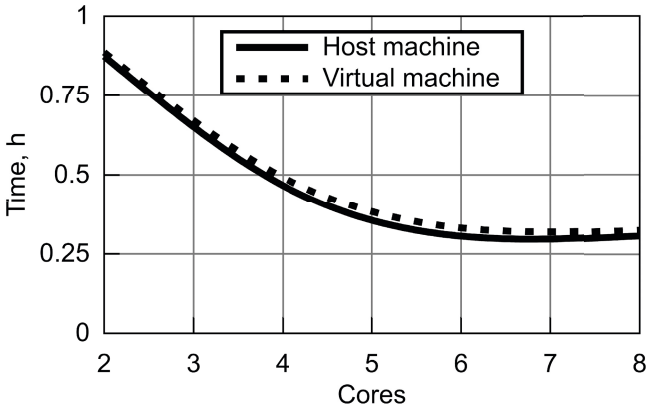

Fig. 3. Performance comparison for host and virtual machine based on GROMACS workload

5.2 Cloud Management Tools for Virtual Clusters

For creation and management of virtual clusters we tested the FishDirector software system developed by Sardina Systems [24]. FishDirector software supports heterogeneous platforms and is hypervisor agnostic. It is architected to scale from a few physical servers to well beyond 100,000, on-premise and in-Cloud.

It is based on OpenStack and allows to create, monitor and manage virtual clusters. Software lets users to create and manage clusters by using web interface or by command line scripts.

FishDirector's decision engines provide power-down/power-up automation which can massively help reduce the carbon emissions of physical servers across the data centre and lowers total power draw. When running HPC jobs on the cluster there is a possibility of conflicts between big and small jobs within a computing estate. To avoid the small job and big job biases FishDirector's Operations Manager predetermines the policies upon how big and small HPC jobs are to be run and adds those constraints to the system, jobs are squeezed into the resource pool, freeing up the remaining resources for other jobs to run and resources are returned to the overall pool once any job is completed. In addition, HPC users with jobs that are set to run for lengthy periods of time will benefit from not having to create time consuming job snapshots for checkpoint restarts.

FishDirector monitors the stability and performance of HPC nodes in the system. Should one of the nodes become unstable, FishDirector can transfer the VM to a different node with no loss of data and without having to roll-back to a checkpoint restart. This reduces the overall time taken to process the HPC job without having to take regular time-consuming job snapshots with the added confidence of full data integrity. Overall structure of FishDirector architecture is shown in Figure 4.

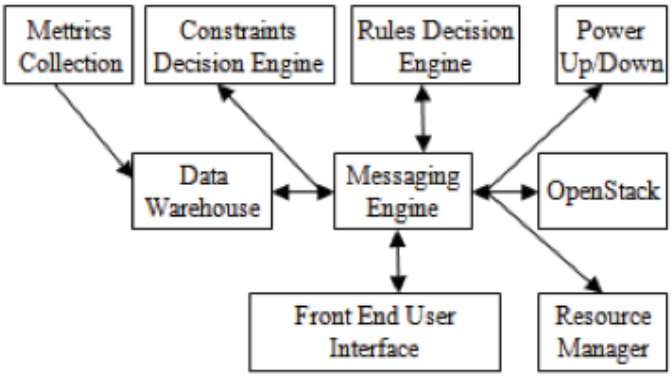

Fig. 4. FishDirector reference architecture

As it was mentioned above, not all hardware and operating systems support paravirtualization, and in that case full virtualization should be used. But not every machine can be fully virtualized considering underlying hardware capabilities. To show the importance of efficient full virtualization we conducted tests on two different machines: the first one supported full virtualization and the second one did not. For this purpose we created virtual guests using FishDirector package on each host and then ran the same test script on physical and virtual machines. The script found an inverse of a matrix with given sizes (100x100,

500x500 and 1000x1000) and then checked whether these matrices are really inverse of each other or not. Besides showing the significance of full virtualization, this test could show efficiency of FishDirector and mathematical correctness of virtual guests (created by FishDirector) work. We ran this script twenty times for each size of matrix on every machine using only RAM and twenty times with writing data on HDD and collected average values of the test time (fig. 5).

	Memory	Size / Machine	100x100	500x500	1000x1000
Without full virtualization	HDD	VM	0.18	4.8	20.16
		Physical	0.06	1.02	3.7
	RAM	VM	0.1	2.2	10.5
		Physical	0.025	0.32	1.5
Supports full virtualization	HDD	VM	0.11	0.92	5.11
		Physical	0.083	0.7	3.93
	RAM	VM	0.09	0.35	1.25
		Physical	0.067	0.3	1.05

Fig. 5. Average values of test results

As it can be seen the difference between the performance of the host and the guest (using only RAM) are much bigger when the used machine did not support full virtualization (around 7 times) (fig. 6), than when fully-virtualized machine was used (around 1.15 times) (fig. 7).

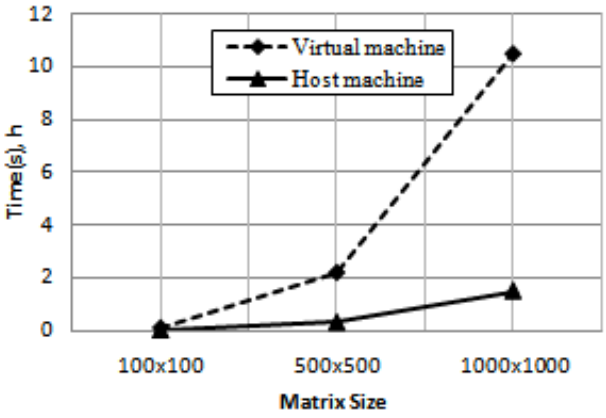

Fig. 6. Performance of host and virtual machines using only RAM, when full virtualization is not supported

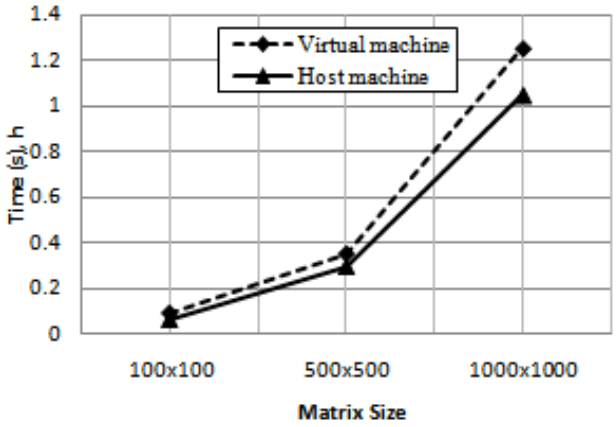

Fig. 7. Performance of host and virtual machines using only RAM, when full virtualization is supported

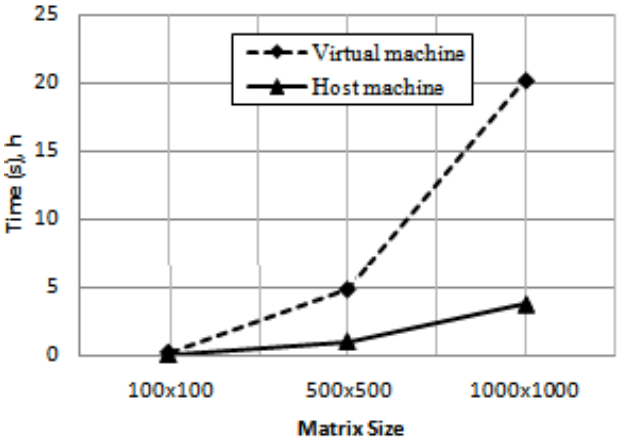

Fig. 8. Performance of host and virtual machines using HDD, when full virtualization is not supported

The situation was almost the same when HDD was used: the performance of virtual machine, which worked on a host with support of full virtualization, was lower of physical host's performance for about 7 times (fig. 8), and in case of full virtualization the difference was about 1.3 times (fig. 9).

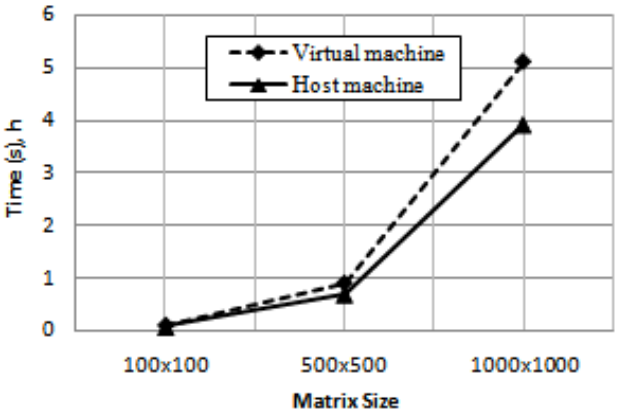

Fig. 9. Performance of host and virtual machines using HDD, when full virtualization is supported

6 Conclusions and Future Work

It is known that virtualization improves security, resilience to failures, substantially eases administration due to dynamic load balancing [28] while does not introduce substantial overheads. Moreover, a proper choice of virtualization package can improve CPU utilization.

Usage of standard cloud technologies as well as process migration techniques can improve overall throughput of a distributed system and adapt it to problems being solved. In that way virtual supercomputer can help people efficiently run applications and focus on domain-specific problems rather than on underlying computer architecture and placement of parallel tasks. Moreover, described approach can be beneficial in utilizing stream processors and GPU accelerators dynamically assigning them to virtual machines.

The key idea of a virtual supercomputer is to harness all available HPC resources and provide user with convenient access to them. Such a challenge can be effectively solved only using contemporary virtualization technologies. They can materialize the long-term dream of having virtual supercomputer at your desk.

In this paper we presented an approach to create and manage virtual clusters based on virtualization and cloud technologies. We presented generic experimental evaluation of the virtualized hardware used as building blocks for the virtual cluster. In the future we plan to perform specific experiments with popular software packages to estimate performance and usability of our solution in real-life problems provided by end-users.

Acknowledgment. The research was carried out using computational resources of Resource Centre Computer Centre of Saint Petersburg State University (T-EDGE96 HPC-0011828-001) and supported by Russian Foundation for Basic Research (project N 13-07-00747) and Saint Petersburg State University (projects N 9.38.674.2013, 0.37.155.2014).

References

1. Smarr, L., Catlett, C.E.: Metacomputing. Communications of the ACM 35(6), 44–52 (1992)
2. Korkhov, V.V., Moscicki, J.T., Krzhizhanovskaya, V.V.: The user-level scheduling of divisible load parallel applications with resource selection and adaptive workload balancing on the grid. IEEE Systems Journal 3(1), 121–130 (2009)
3. Figueiredo, R.J., Dinda, P.A., Fortes, J.A.B.: A case for grid computing on virtual machines. In: Proceedings of the 23rd International Conference on Distributed Computing Systems (2003)
4. Matsunaga, A.M., Tsugawa, M.O., Adabala, S., Figueiredo, R.J., Lam, H., Fortes, J.A.B.: Science gate- ways made easy: the In-VIGO approach. Concurrency and Computation: Practice and Experience 19(6), 905–919 (2007)
5. Krsul, I., Ganguly, A., Zhang, J., Fortes, J.A.B., Figueiredo, R.J.: VMPlants: Providing and manag- ing virtual machine execution environments for grid computing. In: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing (2004)
6. Nishimura, H., Maruyama, N., Matsuoka, S.: Virtual clusters on the fly - fast, scalable, and flexible installation. In: CCGRID 2007: Seventh IEEE International Symposium on Cluster Computing and the Grid (May 2007)
7. Emeneker, W., Stanzione, D.: Dynamic virtual clustering. In: IEEE Cluster 2007, Austin, TX (September 2007)
8. Chase, J.S., Irwin, D.E., Grit, L.E., Moore, J.D., Sprenkle, S.E.: Dynamic virtual clusters in a grid site manager. In: HPDC 2003: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, p. 90. IEEE Computer Society, Washington, DC (2003)
9. Shuming, S., Zhang, H., Yuan, X., Wen, J.-R.: Corpus-based semantic class mining: distributional vs. pattern-based approaches. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 993–1001 (2010)
10. Andrew, L., Gregor, D., Hendrickson, B., Berry, J.: Challenges in parallel graph processing, Parallel Processing Letters, vol. Parallel Processing Letters 17(01), 5–20 (2007)
11. Grzegorz, M., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 135–146 (2010)
12. Iordanov, B.: HyperGraphDB: A generalized graph database. In: Shen, H.T., Pei, J., Özsu, M.T., Zou, L., Lu, J., Ling, T.-W., Yu, G., Zhuang, Y., Shao, J. (eds.) WAIM 2010. LNCS, vol. 6185, pp. 25–36. Springer, Heidelberg (2010)
13. Kevin, K., Luk, S.K.: Building a large-scale knowledge base for machine translation. In: Proceedings of the National Conference on Artificial Intelligence, p. 773 (1994)
14. Ravi, K., Raghavan, P., Rajagopalan, S., Tomkins, A.: Extracting large-scale knowledge bases from the web. In: Proceeding of the International Conference on Very Large Data Bases, pp. 639–650 (1990)
15. Degtyarev, A., Gankevich, I.: Efficiency comparison of wave surface generation using OpenCL, OpenMP and MPI. In: Proceedings of 8th International Conference Computer Science & Information Technologies, Yerevan, Armenia, pp. 248–251 (2011)
16. Wibisono, A., Vasyunin, D., Korkhov, V.V., Zhao, Z., Belloum, A., de Laat, C., Adriaans, P.W., Hertzberger, B.: WS-VLAM: A GT4 based workflow management system. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007, Part III. LNCS, vol. 4489, pp. 191–198. Springer, Heidelberg (2007)

17. Peter, T., et al.: Standardization of an API for distributed resource management systems. In: Seventh IEEE International Symposium on Cluster Computing and the Grid, CCGRID 2007. IEEE (2007)
18. Douglas, T., Tannenbaum, T., Livny, M.: Distributed computing in practice: The Condor experience. Concurrency and Computation: Practice and Experience 17(2-4), 323–356 (2005)
19. Jeffrey, D., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
20. Ping, A., et al.: STAPL: An adaptive, generic parallel C++ library. In: Dietz, H.G. (ed.) LCPC 2001. LNCS, vol. 2624, pp. 193–208. Springer, Heidelberg (2003)
21. Bogdanov, A., Dmitriev, M.: Creation of hybrid clouds. In: Proceedings of 8th International Conference Computer Science & Information Technologies, Yerevan, Armenia, pp. 235–237 (2011)
22. Paul, B., et al.: Xen and the art of virtualization. ACM SIGOPS Operating Systems Review 37(5), 164–177 (2003)
23. Hamlen, K., Kantarcioglu, M., Khan, L., Thuraisingham, B.: Security Issues for Cloud Computing
24. Sardina Systems, `http://www.sardinasystems.com`
25. Resource Center Computer Center of St.Petersburg State University, `http://cc.spbu.ru`
26. Berendsen, H.J.C., van der Spoel, D., van Drunen, R.: GROMACS: A message-passing parallel molecular dynamics implementation. Computer Physics Communications 91(1-3), 43–56 (1995) ISSN 0010-4655
27. Rodríguez, M., Tapiador, D., Fontán, J., Huedo, E., Montero, R.S., Llorente, I.M.: Dynamic Provisioning of Virtual Clusters for Grid Computing. In: César, E., Alexander, M., Streit, A., Träff, J.L., Cérin, C., Knüpfer, A., Kranzlmüller, D., Jha, S. (eds.) Euro-Par 2008. LNCS, vol. 5415, pp. 23–32. Springer, Heidelberg (2009)
28. Bogdanov, A.V., Degtyarev, A.B., Gankevich, I.G., Gayduchok, V.Y., Zolotarev, V.I.: Virtual workspace as a basis of supercomputer center. In: Proceedings of the 5th International Conference on Distributed Computing and Grid-Technologies in Science and Education, Dubna, Russia, pp. 60–66 (2012)