Virtual Private Supercomputer: Design and Evaluation

I. Gankevich[1], V. Gaiduchok[1,2], D. Gushchanskiy[1], Yu. Tipikin[1],
V. Korkhov[1], A. Degtyarev[1], A. Bogdanov[1,2]

[1]Saint-Petersburg State University
[2]Saint-Petersburg Electrotechnical University "LETI"

Saint-Petersburg, Russia

ABSTRACT
Virtual private supercomputer is an efficient way of conducting experiments on high-performance computational environment and the main role in this approach is played by virtualization and data consolidation. During experiment virtualization is used to abstract application from underlying hardware and also from operating system offering consistent API for distributed computations. In between experiments data consolidation is used to store initial data and results in a distributed storage system and offers API for distributed data processing. Combined, these APIs form solid basis of a distributed system shifting user focus from supercomputing technologies to problem being solved.

Keywords
Virtual supercomputer, distributed computing, virtualization.

1. INTRODUCTION
Virtual supercomputer can be seen as a collection of machines working together to compute the problem solution much like a team of people working together to solve a problem. Computers like people need some sort of collective board (or desk) 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 needs of chosen problems and to show advantages of virtual supercomputer approach in these particular cases. The chosen problems should be general enough to make other problems special cases of them, so the two problems were chosen: one of them being ontology storage, retrieval and analysis involving use of a distributed database and another one being fluid dynamics simulations involving execution of highly parallel code. Both problems are suitable for solving in a distributed environment and are discussed in Section 2. Corresponding virtual supercomputer configuration, its key principles and performance evaluation are discussed in Section 3. Based on this evaluation conclusions are made.

2. LARGE-SCALE SUPERCOMPUTER PROBLEMS

2.1. 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 [1]. That kind of structures can be handled by various types of algorithms such as shortest path computations, a special subgraphs allocation, different vari-eties of clustering, etc. It's challenging to efficient process large graphs since some graph's properties work poor with high performance techniques [2]:

- 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 [5] or hypergraph oriented HyperGraphDB (http://www.hypergraphdb.org) [6]. They permit directly operating 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 a creation and processing of knowledge bases and ontologies constructed upon them may be called as another recourse consuming task [3,4]. 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. It complicates methods of their processing because a structure of a graph and graph data are interconnected. Knowledge extraction and ontology-based reasoning can be called examples of such complex tasks.

Growing interest in ontologies development and processing produces a demand for tools which are capable of handling complex operational problems on their own. Such tools have been created and alredy 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, Fect++ and Pellet. Thereby the database hides all the internal work and allows users to work with familiar tools.

2.3. Fluid dynamics simulations

Another way of applying virtual supercomputer is fluid dynamics simulations and this application demands a highly scalable architecture. In particular, experiments in virtual testbed can be carried out on single multiprocessor machine [13] 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 (a pipeline), 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.

3. VIRTUAL SUPERCOMPUTER SOLUTION

3.1. Principles

Although virtual supercomputer can be implemented in many ways and using different combinations of technologies, there are some principles that 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 are as follows.

- 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 a 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 [7]. Moreover, the API do not cover all the functions of underlying PBS [7]. 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 [8] and seamless interaction helps compose hybrid distributed systems dynamically: to extend capacity when needed [11]. 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 [5,9]. 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 [9]. On the other hand, virtual shared memory interface allows processing of data located on any host [10] 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.

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.

3.2. Evaluation

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 our researches at Resource Center Compute Center of SPbSU. This center offers interesting 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 with 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.

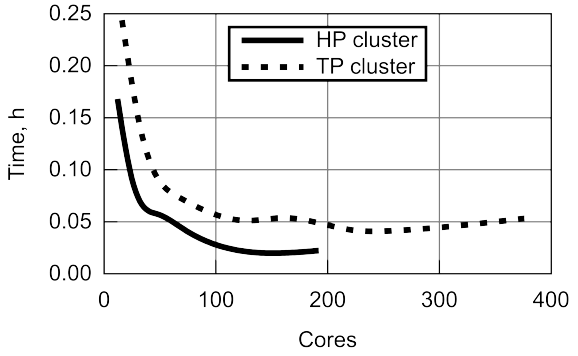

Figure 1. Performance of clusters with different interconnect bandwidth based on GROMACS workload.

Alternatively, user can run jobs on dedicated HPC clusters. In case of our resource center they are T-Platforms cluster and HP cluster. User home directory 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 run-ning on clusters. GROMACS is used for efficient molecular simulations [14]. Figure 1 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 center has 3 machines of this type. In a 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

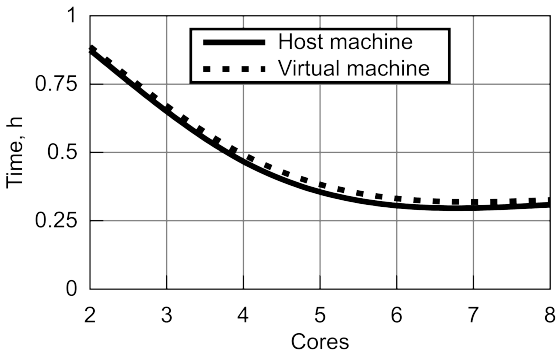

Figure 2. Performance comparison for host and virtual machine based on GROMACS workload.

Virtualization leads to substantial benefits when using it in a big computing center [12]. But what about conventional user PCs? 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 were installed with Debian 7.0. We were interested in testing such a PC with practical workloads. We stopped on GROMACS. GROMACS is a package for molecular simulations. The task chosen put 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, so benefits of virtualization can be used even on conventional PC (Figure 2).

CONCLUSION

It is known that virtualization improves security, resilience to failures, substantially eases administration due to dynamic load balancing [12] while doesn't introduce substantial overheads as it was shown. Moreover, proper choice of virtualization package can improve CPU utilization.

The key idea of 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.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Shi, Shuming, Huibin Zhang, Xiaojie Yuan and Ji-Rong Wen. "Corpus-based semantic class mining: distributional vs. pattern-based approaches.", Proceedings of the 23rd International Conference on Computational Linguistics, pp. 993-1001, 2010.

[2] Lumsdaine, Andrew, Douglas Gregor, Bruce Hendrickson and Jonathan Berry. "Challenges in parallel graph processing.", Parallel Processing Letters 17, no. 01, pp 5-20, 2007.

[3] Knight, Kevin and Steve K. Luk. "Building a large-scale knowledge base for machine translation.", Proceedings of the National Conference on Artificial Intelligence, pp. 773-773, 1994.

[4] Kumar, Ravi, Prabhakar Raghavan, Sridhar Rajagopalan and Andrew Tomkins. "Extracting large-scale knowledge bases from the web." Proceeding of the International Conference on Very Large Data Bases, pp. 639-650, 1999.

[5] Malewicz, Grzegorz, Matthew H. Austern, Aart JC Bik, James C. Dehnert, Ilan Horn, Naty Leiser and Grzegorz Czajkowski. "Pregel: a system for large-scale graph processing." Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 135-146, 2010.

[6] Iordanov, Borislav. "HyperGraphDB: a generalized graph database.", Web-Age Information Management, pp. 25-36, 2010.

[7] Troger, Peter, et al. "Standardization of an API for distributed resource management systems." Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on. IEEE, 2007.

[8] Thain, Douglas, Todd Tannenbaum, and Miron Livny. "Distributed computing in practice: The Condor experience." Concurrency and Computation: Practice and Experience 17.2-4: 323-356, 2005.

[9] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1: 107-113, 2008.

[10] An, Ping, et al. "STAPL: An adaptive, generic parallel C++ library." Languages and Compilers for Parallel Computing. Springer Berlin Heidelberg. 193-208, 2003.

[11] Alexander Bogdanov, Michael Dmitriev. Creation of Hybrid Clouds // Proceedings of 8th International Conference «Computer Science & Information Technologies» — Yerevan, Armenia. pp. 235-237, 2011.

[12] A.V. Bogdanov, A.B. Degtyarev, I.G. Gankevich, V.Yu. Gayduchok, V.I. Zolotarev. Virtual Workspace as a Basis of Supercomputer Center // Proceedings of the 5th International Conference «Distributed Computing and Grid--Technologies in Science and Education» — Dubna, Russia. pp. 60-66, 2012.

[13] Degtyarev A., Gankevich I. Efficiency Comparison of Wave Surface Generation Using OpenCL, OpenMP and MPI // Proceedings of 8th International Conference «Computer Science & Information Technologies» — Yerevan, Armenia, pp. 248-251, 2011.