Experience in building virtual private supercomputer

Vladimir Korkhov, Ivan Gankevich,
Alexander Degtyarev, Alexander Bogdanov
St. Petersburg State University, Russia
e-mail: vladimir@csa.ru, igankevich@ya.ru,
deg@csa.ru, bogdanov@csa.ru

Vladimir Gaiduchok, Nabil Ahmed, Amissi Cubahiro
St. Petersburg Electrotechnical University LETI,
Russia
e-mail: gvladimiru@gmail.com

ABSTRACT

Efficient distribution of high performance computing resources according to actual application needs has been a major research topic since high-performance computing (HPC) technologies became widely introduced. At the same time, comfortable and transparent access to these resources was a key user requirement. In this paper we discuss approaches to build a virtual private supercomuputer (VPS), a virtual computing environment tailored specifically for a target user with a particular target application. Virtualization is one of the cornerstone technologies that helps shaping resources to what is needed by actual users by providing as much as needed when it is needed. However, new issues arise when working with large-scale applications that require large amounts of resources working together. We describe and evaluate possibilities to create the VPS based on light-weight virtualization technologies, and analyze the efficiency of our approach compared to traditional methods of HPC resource management.

Keywords

Virtual cluster, application container, job scheduling, virtual network, high-performance computing.

1. INTRODUCTION

Virtualization refers to the act of creating a virtual version of an object, including but not limited to a virtual computer hardware platform, operating system (OS), storage device, or computer network resources [3]. It can be divided onto several types, and each one has its own pros and cons. Generally, hardware virtualization refers to abstraction of functionalities from physical devices. Nowadays, on modern multicore systems with powerful hardware it is possible to run several virtual guest operating systems on a single physical node. In a usual computer system, a single operating system uses all available hardware resources (CPU, RAM, etc), whilst virtualized system can use a special layer that spreads low-level resources to several systems or applications; this layer looks like a real machine for launched applications.

Virtualization technologies facilitate creation of a virtual supercomputer or virtual clusters that are adapted to problem being solved and help to manage processes running on these clusters (see Figure 1). The work described in this paper continues and summarises our earlier research presented in [4, 6, 7, 8, 9].

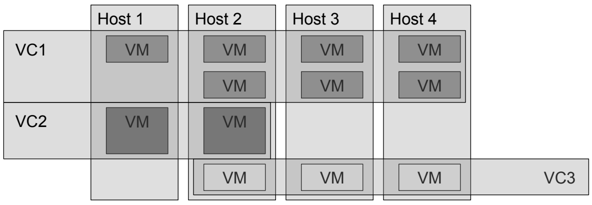

Figure 1: A testbed example with a set of virtual clusters over several physical resources.

In our experience, the main benefit of virtualization for high-performance computing is structural decomposition of a distributed system into unified entities – virtual machines or application containers – which simplifies maintenance of the system. A new entity can be created for each new version of the application with optimal configuration and set of libraries, so that multiple versions of the same software may co-exist and run on the same physical cluster. Entities can be copied or efficiently shared between different physical machines to create private cluster for each application run.

In our experience, virtualization sometimes gives increase in application performance, however, it is not easy achievable. Allocating a separate container for each application allows compiling it for hybrid GPGPU systems which may or may not improve performance of an application. However, such optimisations are possible even without application containers. Full virtualization gives an option of choosing the right operating system for an application, but gives constant decrease in performance due to overheads, which is not tolerable for large-scale parallel applications.

Thus, for high-performance computing virtualization is a tool that helps manage parallel and distributed applications running on physical cluster. It allows different versions of the same libraries and operating systems to co-exist and to be used as environments for running applications that depend on them.

In this work we evaluate the capabilities given by different approcahes and virtualization technologies to build a computational environment with configurable computation (CPU, memory) and network (latency, bandwidth) characteristics, which we call Virtual Private Supercomputer (VPS) [7]. Such configuration enables flexible partitioning of available physical resources between a number of concurrent applications utilizing a single infrastructure. Depending on application requirements and priorities of execution each application can get a customized virtual environment with as much resources as it needs or is allowed to use.

Section 2 gives an overview of related work in the area of virtualization applied to high-perofmance computing. Section 3 presents the approaches to use light-weight virtualization to build the virtual computing environment along with some results of its experimental evaluation. Section 4 discusses the experience and observed experimental results; and Section 5 concludes the paper.

2. RELATED WORK

Research works on the subject of virtual clusters can be divided into two broad groups: works dealing with provisioning and deploying virtual clusters in high performance environment or GRID and works dealing with overheads of virtualization. Works from the first group typically assume that virtualization overheads are low and acceptable in high performance computing, and works from the second group in general assume that virtualization has some benefits for high performance computing.

In [5] authors evaluate overheads of the system for online virtual cluster provisioning (based on QEMU/KVM) and different resource mapping strategies used in this system and show that the main source of deploying overhead is network transfer of virtual machine images. To reduce it they use different caching techniques to reuse already transferred images as well as multicast file transfer to increase network throughput. Simultaneous use of caching and multicasting is concluded to be an efficient way to reduce overhead of virtual machine provisioning.

In [10] authors evaluate general overheads of Xen paravirtualization compared to fully virtualized and physical machines using HPCC benchmarking suite. They conclude that an acceptable level of overheads can be achieved only with paravirtualization due to its efficient inter domain communication (bypassing dom0 kernel) and absence of high L2 cache miss rate when running MPI programs which is common to fully virtualized guest machines.

In contrast to these two works the main principles of our approach can be summarised as follows. Do not use full or paravirtualization of the whole machine but use virtualization of selected components so that overheads occur only when they are unavoidable (i.e. do not virtualize processor). Do not transfer opaque file system images but mount standard file systems over the network so that only minimal transfer overhead can occur. Finally, amend standard task schedulers to work with virtual clusters so that no programming is needed to distribute the load efficiently. These principles are paramount to make virtualization light-weight and fast.

3. EXPERIENCE WITH APPLICATION CONTAINERS

Only light-weight virtualization technologies can be used to build efficient virtual clusters for large-scale problems. This stems from the fact that on large scale no service overhead is acceptable if it scales with the number of nodes. In case of virtual clusters, scalable overhead comes from processor virtualization which means that no para- and fully-virtualized machines are suitable for large virtual clusters. This leaves only application container technologies for investigation. The other challenge is to make dynamic creation and deletion of virtual clusters take constant time.

3.1 System configuration

Test system comprises many standard components which are common in high performance computing, these are distributed parallel file system which stores home directories with experiment's input and output data, cluster resource scheduler which allocates resources for jobs and client programs to pre- and post-process data; the non-standard component is network-attached storage exporting container's root files systems as directories. Linux Container technology (LXC) is used to provide containerisation, GlusterFS is used to provide parallel file system and TORQUE to provide task scheduling. The most recent CentOS Linux 7 is chosen to provide stable version of LXC (greater than 1.0) and version of kernel which supports all container features. Due to limited number of nodes each of them is chosen to be both compute and storage node and every file in parallel file system is stored on exactly two nodes. Detailed hardware characteristics and software version numbers are listed in Table 1.

Table 1. Hardware and software components of the system.

Component	Details
CPU model	Intel Xeon E5440
CPU clock rate (GHz)	2.83
No. of cores per CPU	4
No. of CPUs per node	2
RAM size (GB)	4
Disk model	ST3250310NS
Disk speed (rpm)	7200
No. of nodes	12
Interconnect speed (Gbps)	1
Operating system	CentOS 7
Kernel version	3.10
LXC version	1.0.5
GlusterFS version	3.5.1
TORQUE version	5.0.0
OpenMPI version	1.6.4
IMB version	4.0
OpenFOAM version	2.3.0

To summarise, only standard Linux tools are used to build the system: there are no opaque virtual machines images, no sophisticated full virtualization appliances and no heavy-weight cloud computing stacks in this configuration.

3.2 Evaluation

To test the resulting configuration OpenMPI and Intel MPI Benchmarks (IMB) were used to measure network throughput and OpenFOAM was used to measure overall performance on a real-world application.

The first experiment was to create virtual cluster, launch an empty (with */bin/true* as an executable file) MPI program in it and compare execution time to ordinary physical cluster. To set this experiment up in the container the same operating system and version of OpenMPI as in the host machine was installed. No network virtualization was used, each run was repeated several times and the average was displayed on the graph (Figure 2). The results show that a constant overhead of 1.5 second

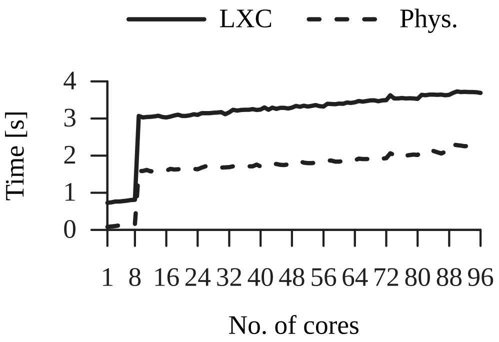

Figure 2: Comparison of LXC and physical cluster performance running empty MPI program.

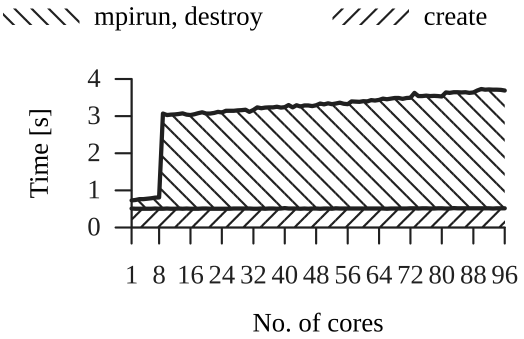

Figure 3. Breakdown of LXC empty MPI program run.

is added to every LXC run after the 8th core: one second is attributed to the absence of cache inside container with SSH configuration files, key files and libraries in it and other half of the second is attributed to the creation of containers as shown in Figure 3. The jump after the 8th core marks bounds of a single machine which means using network for communication rather than shared memory. The creation of containers is fully parallel task and takes approximately the same time to complete for different number of nodes. Overhead of destroying containers was found to be negligible and was combined with *mpirun* time. So, usage of Linux containers adds some constant overhead to the launching of parallel task depending on system's configuration which is split between creation of containers and filling the file cache.

Another experiment dealt with real-world application performance and for this role the OpenFOAM was chosen as the complex parallel task involving large amount of network communication, disk I/O and high CPU load. The dam break RAS case was run with different number of cores (total number of cores is the square of number of cores per node) and different LXC network types and the average of multiple runs was displayed on the graph (Figure 4). Measurements for 4 and 9 cores were discarded because there is a considerable variation of execution time for these numbers on physical machines. From the graph it can be seen that low performance of virtual ethernet decreased final performance of OpenFOAM by approximately 5-10% whereas *macvlan* and *none* performance is close to the performance of physical cluster (Figure 5). Thus, the choice of network type is the main factor affecting performance of parallel applications running on virtual clusters and its overhead can be eliminated by using *macvlan* network type or by not using network virtualization at all. More experimental results are presented in [6].

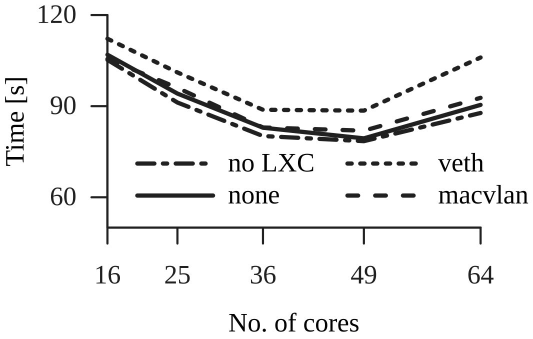

Figure 4: Average performance of OpenFOAM with different LXC network types.

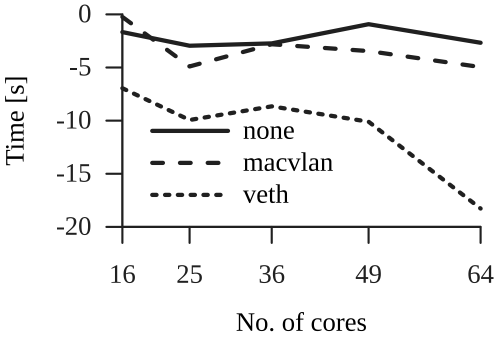

Figure 5: Difference of OpenFOAM performance on physical and virtual clusters. Negative numbers show slowdown of virtual cluster.

To summarise, there are two main types of overheads when using virtual cluster: creation overhead which is constant and small compared to average time of a typical parallel job and network overhead which can be eliminated by not using network virtualization at all.

3.3 Application containers with Docker

The next step in using containers for building virtual cluster is applying various automation and management tools that help to ease deployment and handling of virtual clusters. We investigated capabilities provided by several modern tools (Docker, Mesos, Mininet) to model and build virtualised computational infrastructure, investigated configuration management in the integrated environment and evaluated performance of the infrastructure tuned to a particular test application. Docker – a lightweight and powerful open source container virtualization technology which we use to manage containers – has a resource management system available so it is possible to test different configurations: from "slow network and slow CPUs" to "fast network and fast CPUs".

Even though container-based virtualization is easy to run and use, it's not often easy and user-friendly to scale configuration or to limit resources. This is where Apache Mesos [1] and Mesosphere Marathon [2] were used. Apache Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively. At a high level Mesos is a cluster management platform

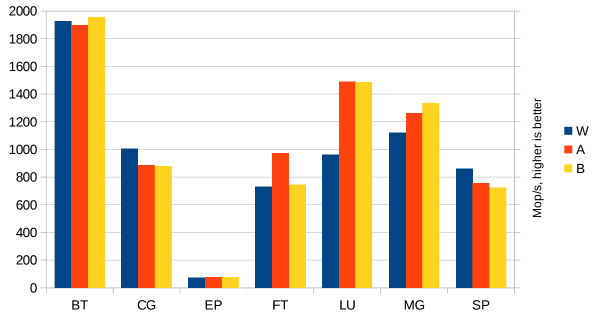

Figure 6: Performance of different tests from NAS Parallel Benchmarks suite on different configurations

that combines servers into a shared pool from which applications or frameworks like Hadoop, Jenkins, Cassandra, ElasticSearch, and others can draw. Marathon is a Mesos framework for long-running services such as web applications, long computations and so on.

Figure 6 shows the experimental results for execution of NAS Parallel Benchmarks (NPB) suite on different configurations of virtual testbed (Class W: workstation size; Classes A, B: standard test problems, 4X size increase going from one class to the next). With NPB results are also very different, everything depends on benchmark type. For example, for SP test smaller size system of nonlinear PDEs had better Mop/s than for bigger size. However, for lower matrices sizes in LU test results are worse than for bigger matrices.

4. DISCUSSION

Light-weight container-based virtualization is the most promising technology for using as an enabling part of the virtual supercomputer concept [7, 8] to ensure proper and efficient distribution of resources between several applications. Knowing the application demands in advance we can create appropriate infrastructure configuration giving just as much resources as needed to each particular instance of a virtual supercomputer running a particular application. In such a way, free resources can be controlled and granted to other applications without negative effect on current executions with minimal overhead.

5. CONCLUSION

Presented approach for creating virtual clusters from Linux containers was found to be efficient and its performance comparable to ordinary physical cluster. From the point of view of system administrator, storing each HPC application in its own container makes versioning and dependencies control easy manageable and their configuration does not interfere with the configuration of host machines and other containers. Usage of standard virtualization technologies 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 tasks.

Acknowledgement

The research presented in this paper was carried out using computational resources of Resource Center "Computer Centre of Saint-Petersburg State University" within frameworks of grants of Russian Foundation for Basic Research (project no. 13-07-00747) and Saint Petersburg State University (projects no. 9.38.674.2013, 0.37.155.2014).

REFERENCES

[1] Apache Mesos. Available online: `http://mesos.apache.org/`. Retrieved: 2015-07-04.

[2] Mesosphere Marathon. Available online: `https://mesosphere.github.io/marathon/`. Retrieved: 2015-07-04.

[3] Virtualisation in Wikipedia. Available online: `http://en.wikipedia.org/wiki/Virtualization`. Retrieved: 2015-07-04.

[4] A.V. Bogdanov, A.B. Degtyarev, I.G. Gankevich, V.Yu. Gayduchok, and V. I. Zolotarev. Virtual workspace as a basis of supercomputer center. In *Proceedings of 5th International Conference on Distributed Computing and Grid-Technologies in Science and Education*, pages 60–66, 2012.

[5] Yang Chen, Tianyu Wo, and Jianxin Li. An efficient resource management system for on-line virtual cluster provision. In *Proc. of International Conference on Cloud Computing (CLOUD)*, pages 72–79. IEEE, 2009.

[6] I. Gankevich, S. Balyan, S. Abrahamyan, and V. Korkhov. Applications of on-demand virtual clusters to high performance computing. *Computer Research and Modelling*, 7(3):504–509, 2015.

[7] I. Gankevich, V. Gaiduchok, D. Gushchanskiy, Y. Tipikin, V. Korkhov, A. Degtyarev, A. Bogdanov, and V. Zolotarev. Virtual private supercomputer: Design and evaluation. In *CSIT 2013 - 9th International Conference on Computer Science and Information Technologies (CSIT), Revised Selected Papers, DOI: 10.1109/CSITechnol.2013.6710358*, 2013.

[8] I. Gankevich, V. Korkhov, S. Balyan, V. Gaiduchok, D. Gushchanskiy, Y. Tipikin, A. Degtyarev, and A. Bogdanov. Constructing virtual private supercomputer using virtualization and cloud technologies. In *Proceedings of International Conference on Computational Science and Its Applications (ICCSA 2014). Lecture Notes in Computer Science*, volume 8584, pages 341–354, 2014.

[9] V. Korkhov, S. Kobyshev, and A. Krosheninnikov. Flexible configuration of application-centric virtualized computing infrastructure. In *Proceedings of International Conference on Computational Science and Its Applications (ICCSA 2015). Lecture Notes in Computer Science*, volume 9158, pages 342–353, 2015.

[10] Kejiang Ye, Xiaohong Jiang, Siding Chen, Dawei Huang, and Bei Wang. Analyzing and modeling the performance in Xen-based virtual cluster environment. In *Proc. of the 12th International Conference on High Performance Computing and Communications (HPCC)*, pages 273–280. IEEE, 2010.