Virtual Clusters as a Way to Experiment Software

A. S. Krosheninnikov[a], V. V. Korkhov[b], S. S. Kobyshev[c], A. B. Degtyarev[d], A. V. Bogdanov[e]

St. Petersburg State University, 7-9 Universitetskaya nab., St. Petersburg, 199034, Russia

E-mail: [a] feeblehamster@gmail.com, [b] v.korkhov@spbu.ru, [c] kobyshev.sergey@gmail.com, [d] a.degtyarev@spbu.ru, [e] a.v.bogdanov@spbu.ru

Modern scientific applications often require provisioning of computing and networking infra-structures tailored to particular application needs which might not be clearly defined. In addition, allo-cated resources of an actual physical infrastructure might be underutilized by applications (e.g., a pro-cess does not utilize provided CPU or network connection fully). Traditional virtualization technolo-gies can solve the problem partially, however, lead to noticeable overheads. In this paper we consider creation of virtual topologies to test distributed software behaviour without the need to construct real network topologies. We evaluate an approach to create dedicated computing environments configured for particular applications based on light-weight virtualization also known as containers. We investi-gate available capabilities to model and create dynamic container-based virtual infrastructures sharing a common set of physical resources, and evaluate their performance on a set of test applications with different requirements.

Keywords: virtualization, containers, virtual cluster

This work was supported by Russian Foundation for Basic Research (projects N 16-07-01111, 16-07-00886, 16-07-01113) and St. Petersburg State University (project N 0.37.155.2014).

Introduction

Continuous development of computer hardware along with the development of computational methods and algorithms encourages new ways of combining software and hardware, matching application requirements and resources, thus mapping programs to computing infrastructures. Virtualization technologies started a new age of tailoring computing environment to the needs of users and applications. However, flexibility of full- and para-virtualization approaches is hold back by some limitations causing extra overheads, resource consumption and lack of dynamics. Light-weight or container-based virtualization, a new generation of virtualization techniques, can give better answers to create a flexible and dynamic distributed computing infrastructure with small overheads.

Containers as a way to create a dedicated environment for running applications have been around for years. However, the boost of new interest to them started when new technologies and tools to orchestrate their operations appeared. One of the most commonly used tool to manage container infrastructures is Docker [The Docker Platform, 2016].

Traditional hypervisor-based virtualization is still widely used to deploy and run applications on a wide range of platforms, however, it suffers from a number of restrictions:

- Significant overheads while running fully-virtualized guest operating systems, in particular, overheads to boot up virtual machine instances
- Lack of flexibility to allocate resources to particular processes
- Downfalls of application performance due to virtualization overheads, hypervisor mediation etc.

In this paper, we evaluate the capabilities obtained while using the OS-level virtualization technology to build a computational environment with configurable computation (CPU, memory) and network (latency, bandwidth) characteristics. Such configuration enables flexible partitioning of available physical resources between a number of concurrent applications utilizing a single physical 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.

Conducting offline simulations is often needed to preliminary assess the correctness and adequacy of the applications that interact with the network, in particular to evaluate performance of network-related algorithms. The simulation tools like NS-3 [NS-3 Project Homepage, 2016] and OMNet++ [OMNet++ project homepage, 2016] allow to perform simulations and evaluate the efficiency and scalability of algorithms or protocols without running them in real networks.

Our main interest is to use container-based computing infrastructures for parallel high-performance computing applications: parallel programs that consist of a number of processes running on computing nodes and communicating during the execution. Using containers as computing nodes can help us to control and share available computing and networking resources between concurrent parallel applications. Thus, applications with complementary requirements (e.g. fast CPU-slow network + slow CPU-fast network) can co-exist on a single physical node or a VM without affecting each other much.

The approach that we propose is complementary to the traditional queue-based batch processing used in HPC systems. Applications would not have to wait in the queue until worker nodes become fully free and available. Instead, the scheduler can control fraction of resources allocated for each application thus enabling immediate execution for applications with requirements fitting still available fraction of resources. In addition, flexible quality of service (QoS) and service-level agreement (SLA) policies can be built on top of such infrastructure: applications might be ready to get smaller amount of resources right away rather than wait in line to acquire more resources.

Building and evaluating container-based distributed computing environment

We have implemented a prototype of container-based distributed computing infrastructure on the cloud resources provided by Microsoft Azure. To build the infrastructure we used a set of 8 virtual machines residing in different regions (5 machines in East US; 3 machines in North Europe) with the following characteristics: Instance type: A1; Cores: 1; Memory: 1.75GB.

Current prototype implementation does not use any specific container management tools and relies only on Docker and a custom python-based toolkit developed to configure and execute containers for parallel applications (with OpenMPI deployed and configured) and control resource usage with help of a separately maintained database.

We used several programs from NAS Parallel Benchmarks (NPB) suite as the applications with various requirements to the underlying infrastructure. NPB is a small set of programs designed to help evaluate the performance of parallel computers and clusters. The benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications. Moreover, the benchmark suite contains benchmarks for unstructured adaptive mesh, parallel I/O, multi-zone applications, and computational grids [NAS Parallel Benchmarks, 2016]. In our experiments we used MG, FT, and CG kernels:

- MG - Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive
- FT - discrete 3D fast Fourier Transform, all-to-all communication
- CG - Conjugate Gradient, irregular memory access and communication

The aims of the experiments were the following:

- Investigate performance of parallel applications on the prototype of distributed container-based infrastructure
- Check how performance of applications with different requirements on cpu/memory/network varies depending on infrastructure configuration
- Evaluate possibilities of concurrent execution of parallel applications, minimizing their influence on each other

The first set of experiments was performed to evaluate the resource saturation point for an application: the point when adding more memory (or available cpu, or network bandwith) would not increase application performance anymore. Sample results are presented in figure 1. We can see that after some point the performance of applications does not increase with increasing the amount of allocated resources per application. Namely, the left plot demonstrates that for the application (FT class S) the performance stops increasing after increasing available bandwidth between the nodes more than 900 Kbit/s; the right plot shows that the amount of available memory is crucial for the application to start (FT class A, the application does not start with less than 70MB of memory available) but does not influence the performance when amount of memory is increased.

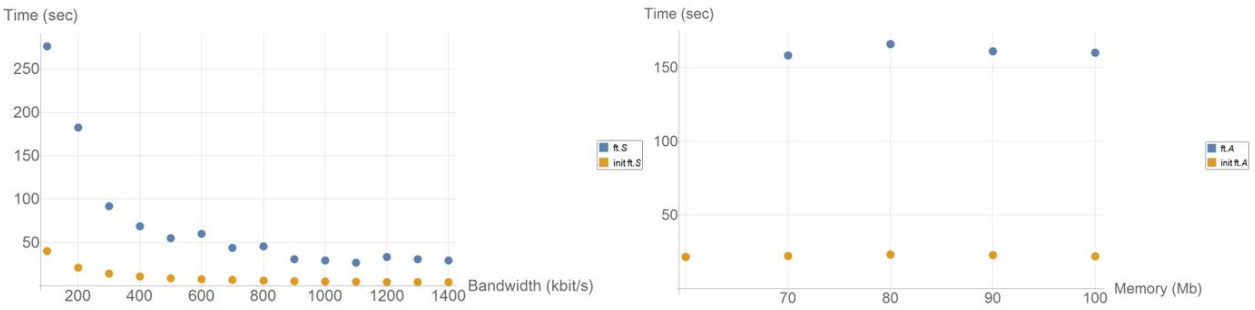

Figure 1. Experimental results: saturation of resource requirements for FT kernel

The results presented in Figure 2 illustrate details of application performance for a particular configuration of computing infrastructure.

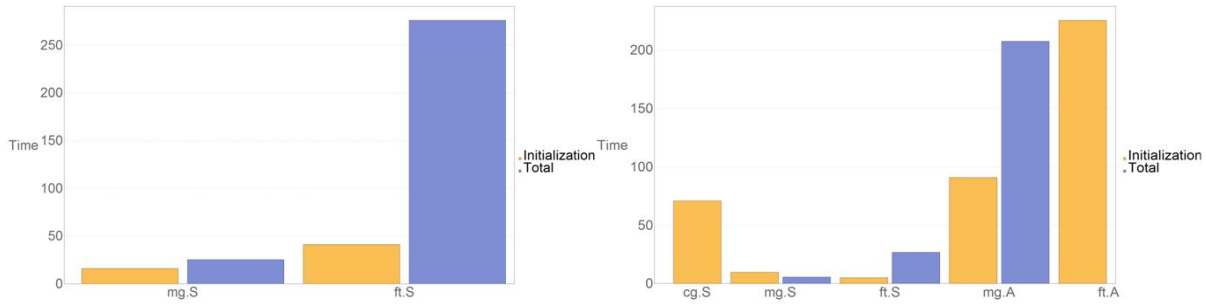

Figure 2. Experimental results. Left: mem 256MB, net 100 Kb/s; Right: mem 512 MB, net 1024 Kb/s

Next, we have evaluated sequential and concurrent execution of two different application kernels, MG and FT, to ensure that concurrent execution of both applications will not affect their performance in case container clusters are configured to meet the individual requirement of the applications. Figure 3 illustrates observed differences in initialization and benchmark time of MG and FT kernels in a particular virtual hardware configuration (512 MB memory, 120 Kbit/s network) for sequential and concurrent execution. Here sequential execution means allocation of the whole set of resources to each of the applications and executing them one by one; concurrent execution means execution of both kernels simultaneously in separate containers with given limitations. Figure 4 shows experimental comparison of shared and concurrent execution, where shared execution means running both MG and FT kernels in a single container simultaneously. We can observe that in this case kernels can compete for shared resources which results in overall performance degradation.

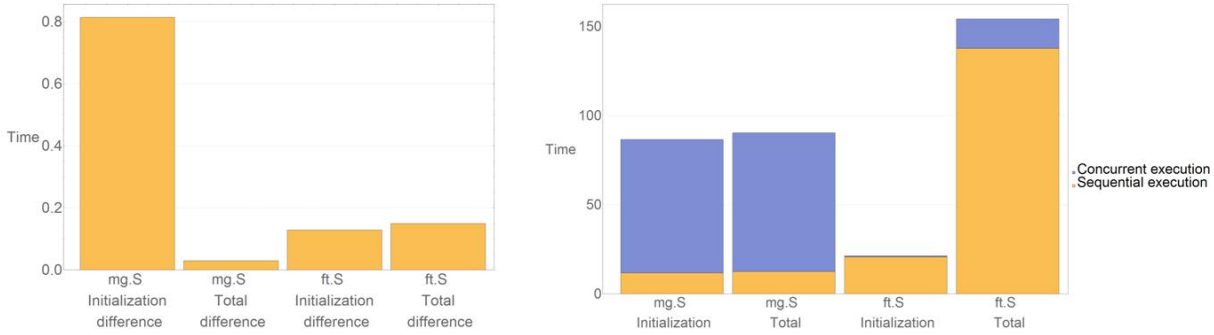

Figure 3. Experimental evaluation of separate and concurrent execution: difference in init and benchmark time; mem 512MB, net 120 Kb/s

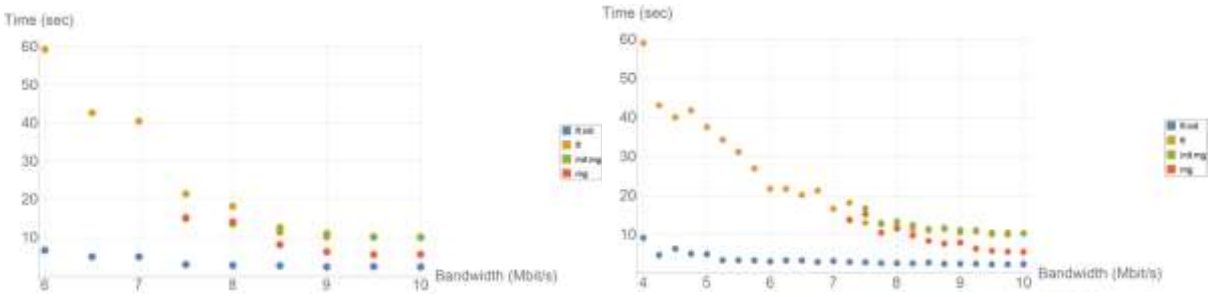

Figure 4. Experimental comparison of shared (left) and concurrent (right) execution: mem 100MB, lan 120 Kb/s

Conclusions and future work

The presented work continues the developments presented in [Korkhov, Kobyshev, …, 2015]; this approach can be used as an enabling part of the virtual supercomputer concept [Gankevich, Gaiduchok, …, 2013; Gankevich, Korkhov, …, 2014] 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. Here we use containers as an enabling part of the computing infrastructure. In such a way, free resources can be controlled and granted to concurrent applications without negative effect on other executions.

In this paper we proposed and evaluated the usage of concurrently running container clusters that are created based on application requirements and have minimal effect on each other by resource allocation control. We demonstrated a proof-of-concept prototype running on Microsoft Azure cloud resources and showed experimental evaluation of its performance on a set of NAS benchmarks based on real application kernels.

Our future work will be to look more closely into container cluster management and orchestration software (e.g. Docker Swarm, Kubernetes, or Mesos) to delegate the functionality of maintaining the cluster to these tools so that we could concentrate on mechanisms of application requirements evaluation and concurrent execution of applications on distributed container resources.

References

Gankevich I., Gaiduchok V., Gushchanskiy D., Tipikin Y., Korkhov V., Degtyarev A., Bogdanov A., Zolotarev V. Virtual private supercomputer: Design and evaluation // CSIT 2013 // 9th International Conference on Computer Science and Information Technologies (CSIT), Revised Selected Papers. P. 1-6. DOI: 10.1109/CSITechnol.2013.6710358

Gankevich I., Korkhov V., Balyan S., Gaiduchok V., Gushchanskiy D., Tipikin Y., Degtyarev A., Bogdanov A. Constructing Virtual Private Supercomputer Using Virtualization and Cloud Technologies // Proceedings of International Conference on Computational Science and Its Applications (ICCSA 2014). Lecture Notes in Computer Science. 2014. Vol. 8584. P. 341-354.

Korkhov V., Kobyshev S., Krosheninnikov A. Flexible configuration of application-centric virtualized computing infrastructure, Computational Science and Its Applications (ICCSA 2015). Lecture Notes in Computer Science. 2015. Vol. 9158. P. 342-353.

NAS Parallel Benchmarks [Electronic resource]: http://www.nas.nasa.gov/publications/npb.html (accessed 20.11.2016).

NS-3 Project Homepage [Electronic resource]: http://www.nsnam.org/ (accessed 20.11.2016).

OMNeT++ project homepage [Electronic resource]: https://omnetpp.org/ (accessed 20.11.2016).

The Docker platform [Electronic resource]: https://www.docker.com/ (accessed 20.11.2016).