Accessing GPUs from Containers in HPC

Lucas Benedicic
Systems Integration Group
CSCS, Swiss National Supercomputing Centre
Lugano, Switzerland
Email: lucas.benedicic@cscs.ch

Miguel Gila
HPC Operations Group
CSCS, Swiss National Supercomputing Centre
Lugano, Switzerland
Email: miguel.gila@cscs.ch

I. Introduction

Compared with the now long known hypervisor-based virtualization technologies, containers provide a level of virtualization which allows running multiple isolated user-space instances on top of a common host kernel [1]. Containers are light, flexible and easy to deploy.

On the other hand, graphics processing units (GPUs) are becoming increasingly important in providing power-efficient and massively-parallel computational power to the scientific community in general and HPC in particular.

However, having these two valuable technologies work together is not a trivial task. The difficulty lies in that they are essentially incompatible.

Containers have been designed to be both hardware- and platform-agnostic, whereas GPU-accelerated applications require specific hardware and software in order to benefit from increased performance.

Therefore, in order for containers to be a viable option even in hybrid HPC systems like Piz Daint, access to GPU hardware should be seamless in terms of application deployment (e.g., no modifications are required) and not impose any performance limitations. Piz Daint, currently the fastest supercomputer in Europe, is a hybrid Cray XC30 in production at the Swiss National Supercomputing Centre (CSCS) in Lugano, Switzerland. The system features a total of 5,272 compute nodes, each of which is equipped with an Intel® Xeon® E5-2670, an Nvidia® Tesla® K20X with 6 gigabytes of GDDR5 memory and 32 gigabytes of host memory.

Working in close collaboration with the National Energy Research Scientific Computing Center (NERSC), CSCS is extending Shifter [2] to enable scalable GPU access from containers.

II. Motivation

There are several reasons for choosing Shifter over other container technologies [3], e.g., Docker, for using GPUs in an HPC environment. In particular, Docker is impractical for deployment in supercomputers like Piz Daint due to the barriers it imposes [2], including security, kernel and architectural constraints, scalability issues and integration with resource managers.

The goal is to provide the container users with the ability to access the compute resources of the K20X GPUs available in the compute nodes of Piz Daint using Nvidia CUDA.

III. Implementation

With a continously increasing number of contributions to the Shifter codebase, CSCS has implemented the required customizations to support these use cases. The changes include new configuration entry points, custom base images exposing the required software components and a number of typical use cases readily available to any user to modify.

The proposed design attempts to accomplish several goals:

- **Functionality** The provided GPU support through Shifter should be compatible with different programming models like CUDA or OpenCL. Also, the level of compatibility provided by the kernel driver should be respected, e.g., for running applications based on different runtime versions like CUDA 6.5 or 7.0, and OpenCL 1.0 or 1.1.
- **Performance** Accessing the resources of the compute nodes from containers should not introduce any performance penalty and should allow GPU-enabled applications to run as fast as in the native case.
- **Transparency** The provided GPU support through Shifter should be intuitive and useful in a variety of use cases. Regarding complex scientific workflows, there is a clear need for seamlessly supporting such applications, regardless of the operating system or kernel version on the compute node.

A. CUDA

Because of the way Nvidia GPGPU solution for Linux is engineered, the container needs to have direct access to its main components on the host namely the kernel driver and the Nvidia user-space libraries. Since containers share the kernel and its drivers with the host system, only the Nvidia user-space libraries are left to be handled separately in the container. It is important that the version of the user-space libraries exactly matches the version of the Nvidia driver loaded into the host kernel for the GPU access to work correcly.

One of the first working prototypes featured a complete installation of both the Nvidia driver and a supported CUDA toolkit inside a container. However, such container images could not be deployed on systems with a version mismatch and had to be built specifically for each host system.

The current solution for Shifter makes container images portable while still leveraging GPUs, since they are Nvidia-driver agnostic. The required character devices and driver

files are mounted at deployment time, when the container is executed on the target machine.

Before mounting the driver files, the prototype scans the target system looking for the driver libraries that match the currently loaded kernel driver. Two methods can be used to make these libraries available inside the container. The first one involves the use of environment variables, e.g., `LD_LIBRARY_PATH` or `CUDA_ROOT`. The second method modifies the configuration of the `ldconfig` cache in order for the dynamic linking engine to find the libraries. This set of user-space libraries works as an interface, allowing the CUDA applications access to the GPU device via the kernel driver itself. Since Shifter does not currently support mounting individual files, the so-called *discovered* driver libraries are placed in a common directory before launching the container. This directory is then mounted inside the container and its dynamic-linking configuration is altered in order for the run-time to access these libraries. By using this procedure, it is possible to successfully execute different CUDA and OpenCL applications directly from the container without changing the application itself.

IV. Performance

Since containers are meant to package complete applications, we feel the most representative benchmarks are the performance tests of common parallel algorithms. Within this group, we have selected serial versions that execute on a single node and use only one device. With these measurements we would like to confirm that there is no additional overhead when accessing the GPU from a Shifter container rather than from the native Cray runtime. To accomplish this, we have selected and executed four benchmarks of the scalable heterogeneous computing (SHOC) benchmark suite [4], namely:

- **FFT** measures the performance of a two dimensional Fast Fourier Transform. The benchmark computes multiple FFTs in parallel;
- **MD** measures the speed of a simple pairwise calculation of the Lennard-Jones potential from molecular dynamics;
- **Stencil2D** measures the performance of a standard two-dimensional nine point stencil calculation;
- **S3D** measures performance for S3D's getrates kernel which computes the rate of various chemical reactions across a regular 3D grid.

These benchmarks are part of the SHOC version 1.1.5. The selected problem size was class 4, i.e., HPC-Focused or Large Memory GPUs, and the parameters for the algorithms were left to their default values.

The plots show the arithmetic mean after executing ten repetitions. Fig. 1 shows the performance comparison between CUDA implementations of the four benchmarks, whereas Fig. 2 shows the performance comparison between OpenCL implementations of the same benchmarks.

We can clearly see that there is no performance drop whether the benchmarks were executed natively or from a container launched with Shifter on Piz Daint. The error lines on top of each column, which are too small to be seen on the plots apart from the native case of FFT in CUDA, confirm a negligible standard deviation emerging from the ten independent runs of the tests.

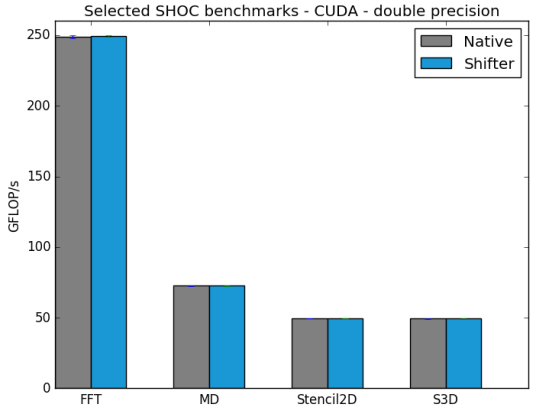

Fig. 1. Performance comparison between native and Shifter execution of some CUDA benchmarks.

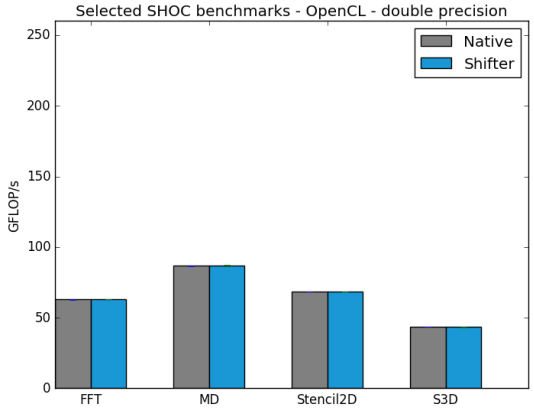

Fig. 2. Performance comparison between native and Shifter execution of some OpenCL benchmarks.

V. Conclusion

Container environments opened up opportunities to enable workloads that were typically constrained to specialized or hypervisor-based operating system. The presented work allows CSCS users to seamlessly leverage HPC resources directly from containers, including access to GPU devices. Moreover, a container workflow is easier to deploy and also equally performant as natively compiled applications on Piz Daint.

References

[1] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*. IEEE, 2015, pp. 171–172.

[2] D. M. Jacobsen and R. S. Canon, "Shifter: Containers for HPC," in *Cray Users Group Conference (CUG'16)*, 2016.

[3] L. Benedicic, M. Gila, S. Alam, and T. Schulthess, "Opportunities for container environments on Cray XC30 with GPU devices," in *Cray Users Group Conference (CUG'16)*, 2016.

[4] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (SHOC) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. ACM, 2010, pp. 63–74.