A Performance Comparison of CUDA Remote GPU Virtualization Frameworks

Carlos Reaño and Federico Silla
Universitat Politècnica de València, València, 46022, Spain
carregon@gap.upv.es, fsilla@disca.upv.es

***Abstract*—Using GPUs reduces execution time of many applications but increases acquisition cost and power consumption. Furthermore, GPUs usually attain a relatively low utilization. In this context, remote GPU virtualization solutions were recently created to overcome the drawbacks of using GPUs.**

Currently, many different remote GPU virtualization frameworks exist, all of them presenting very different characteristics. These differences among them may lead to differences in performance. In this work we present a performance comparison among the only three CUDA remote GPU virtualization frameworks publicly available at no cost. Results show that performance greatly depends on the exact framework used, being the rCUDA virtualization solution the one that stands out among them. Furthermore, rCUDA doubles performance over CUDA for pageable memory copies.

***Keywords*-GPGPU; CUDA; HPC; virtualization;**

I. Introduction

The use of CUDA GPUs allows reducing the execution time of parallel applications. However, using GPUs presents several side-effects, such as increased acquisition and maintenance costs. In addition, GPU utilization is usually low. In this context, remote GPU virtualization frameworks have been recently created to overcome these drawbacks, such as VGPU [9], GridCuda [4], DS-CUDA [5], GVirtuS [2], vCUDA [8], GViM [3], rCUDA [6], and Shadowfax II [7].

Different virtualization frameworks provide different features. For example, the vCUDA technology supports the old CUDA 3.2 version and implements an unspecified subset of the CUDA runtime. Moreover, its communication protocol presents a considerable overhead. GViM is based on the old CUDA version 1.1 and does not implement the entire runtime API. The gVirtuS approach is based on the old CUDA version 2.3 and implements only a small portion of the runtime API. VGPU is a recent tool supporting CUDA 4.0 although the information provided by its authors is fuzzy. GridCuda supports the old CUDA 2.3 version. Regarding DS-CUDA, it integrates a more recent version of CUDA (4.1) and includes specific communication support for InfiniBand. Shadowfax II is still under development, not presenting a stable version yet and its public information is not updated to reflect the current code status. In the case of rCUDA, it is binary compatible with CUDA 6.5 and implements specific communication modules for different interconnects. Currently, it supports the general TCP/IP protocol stack and includes specific modules for InfiniBand.

In this work we present a performance comparison among the publicly available CUDA remote GPU virtualization frameworks: gVirtuS, DS-CUDA, and rCUDA. The performance of CUDA is also included for comparison purposes.

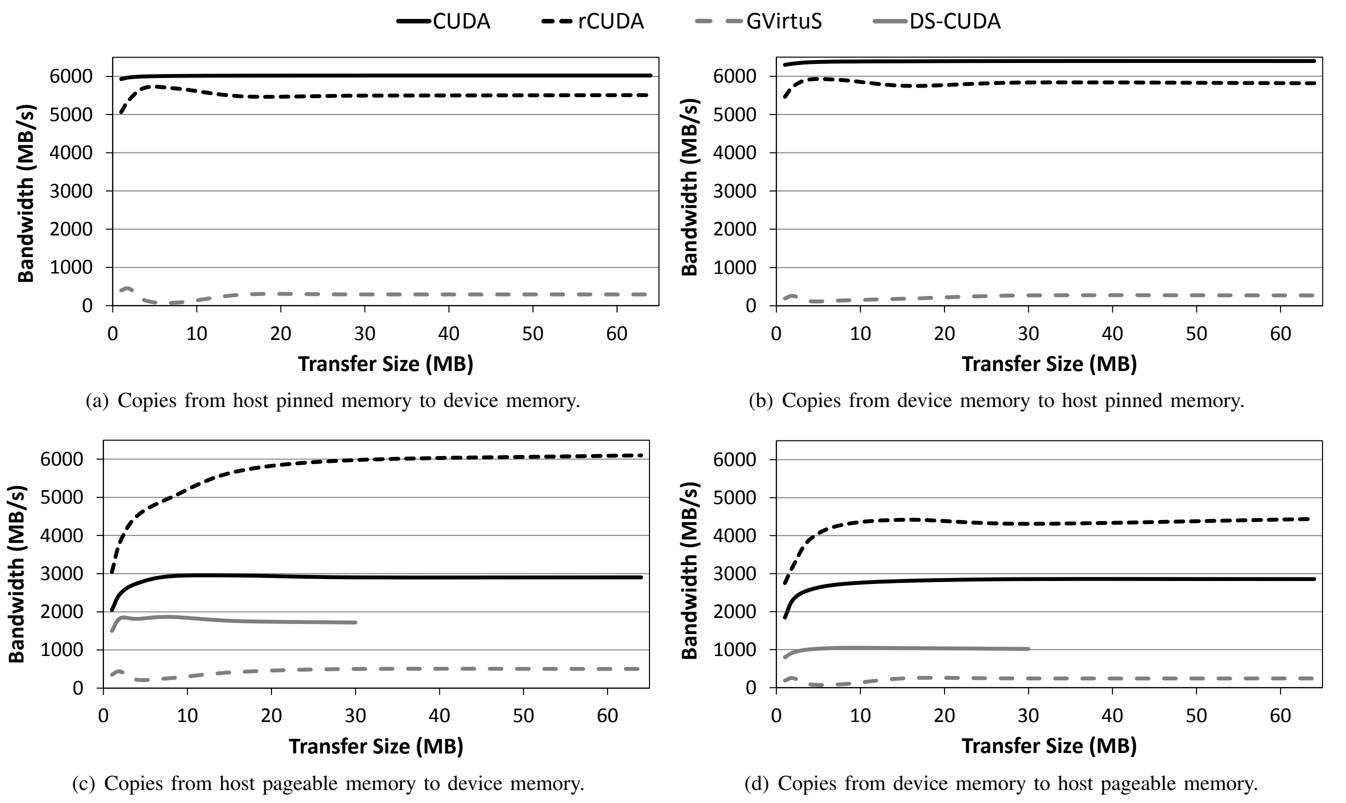
II. Performance Comparison

Figure 1 presents a performance comparison of the three GPU virtualization solutions under study, showing also the performance of CUDA as the baseline reference. The well-known `bandwidthTest` benchmark from the NVIDIA CUDA Samples has been employed. The reason for using bandwidth for comparing performance is that, when transferring data in CUDA applications between main memory and GPU memory, data copy sizes are, in general, large (in the order of MB). These large data transfers are mostly influenced by attained bandwidth, which turns out to be the most limiting factor regarding the performance of these solutions. Consequently, other metrics such as latency are less relevant in this context.

The testbed used in the experiments consists of two 1027GR-TRF Supermicro nodes featuring two Intel Xeon E5-2620v2 processors (Ivy Bridge) operating at 2.1 GHz and 32 GB of DDR3 memory at 1600 MHz. The computers also include an FDR InfiniBand adapter. One of the nodes owns an NVIDIA Tesla K20 GPU. Linux CentOS 6.4 was used along with CUDA 6.5 (NVIDIA driver 340.29) and Mellanox OFED 2.3-2.0.0 (InfiniBand drivers and administrative tools). Given that the InfiniBand FDR network technology was used to connect both computers, both rCUDA and DS-CUDA made use of the InfiniBand Verbs API. In the case of gVirtuS, TCP/IP over InfiniBand was used because it is not able to take advantage of the InfiniBand Verbs API.

Notice that the three GPU virtualization solutions analyzed support different versions of CUDA: DS-CUDA is compatible with CUDA 4.1, gVirtuS supports the old CUDA 2.3 version and rCUDA supports CUDA 6.5. Thus, each of the frameworks has been analyzed with the respective version of CUDA supported. In this regard, it is important to remark that, in order to avoid introducing additional noise in this particular test, we have previously compared the bandwidth achieved by the three versions of CUDA and the result is that differences in performance for the bandwidth test are negligible from one CUDA version to another.

Results in Figure 1 deserve some discussion. First, it can be seen in Figures 1(a) and 1(b) that CUDA achieves the highest performance when pinned memory is used, attaining a bandwidth around 6,000 MB/s. Notice that this bandwidth is noticeably reduced for copies using pageable memory, as show in Figures 1(c) and 1(d). Second, Figure 1 shows that rCUDA outperforms the other two remote GPU virtualization solutions. Actually, for copies using pageable memory rCUDA also performs better than CUDA. This is a well-known effect thoroughly described in previous



(a) Copies from host pinned memory to device memory.

(b) Copies from device memory to host pinned memory.

(c) Copies from host pageable memory to device memory.

(d) Copies from device memory to host pageable memory.

Figure 1. Performance comparison among three publicly available CUDA GPU virtualization solutions: gVirtuS, DS-CUDA, and rCUDA. The comparison is performed in terms of attained bandwidth. The performance of CUDA is also depicted for comparison purposes.

works on rCUDA [6] and is due to the use of an efficient pipelined communication based on the use of internal pre-allocated pinned memory buffers. Nevetherless, the rCUDA performance shown in the figure is a noticeable improvement over other previously published results. On the other hand, notice that both rCUDA and DS-CUDA make use of the InfiniBand Verbs API, thus having access to the large bandwidth available in this interconnect. However, DS-CUDA presents a very poor performance. Also, notice that DS-CUDA supports neither memory copies larger than 32MB nor the use of pinned memory. Regarding gVirtuS, its performance is extremely low. One may think that this is due to the fact that gVirtuS is using TCP/IP over InfiniBand, which clearly achieves lower performance than the InfiniBand Verbs API. However, according to our measurements with the iperf tool, TCP over InfiniBand FDR provides around 1,190 MB/s, which is a noticeably larger bandwidth than the one attained by gVirtuS. Hence, the low performance of this middleware is not due to the use of TCP/IP over InfiniBand but to the way it internally manages communications.

III. Conclusions

In this work we have compared the performance of the publicly available CUDA remote GPU virtualization frameworks. Furthermore, their throughput has been put into the context of the performance of CUDA. Results show that the rCUDA framework outperforms the other two remote GPU virtualization solutions. Moreover, results point out that, for pageable memory copies, rCUDA attains more bandwidth than CUDA, regardless of the copy size or direction. This improvement over CUDA is a novel result, not previously published in prior works on rCUDA.

Acknowledgment

This work was funded by the Generalitat Valenciana under Grant PROMETEOII/2013/009 of the PROMETEO program phase II. Authors are also grateful for the generous support provided by Mellanox Technologies.

References

[1] A. Gaikwad, et al.: GPU based sparse grid technique for solving multidimensional options pricing PDEs. WHPCF'09

[2] G. Giunta, et al.: A GPGPU transparent virtualization component for high performance computing clouds. EuroPar'10

[3] V. Gupta, et al.: GViM: GPU-accelerated virtual machines. HPCVirt'09

[4] T.Y. Liang, et al.: GridCuda: a grid-enabled CUDA programming toolkit. WAINA'11

[5] M. Oikawa, et al.: DS-CUDA: a middleware to use many GPUs in the cloud environment. SC'12

[6] A.J. Peña, et al.: A complete and efficient cuda-sharing solution for HPC clusters. PARCO 2014

[7] Shadowfax II - scalable implementation of GPGPU assemblies. http://keeneland.gatech.edu/ software/keeneland/kidron

[8] L. Shi, et al.: vCUDA: GPU accelerated high performance computing in virtual machines. IPDPS'09

[9] V-GPU: https://github.com/zillians/platform_manifest_vgpu