POSTER: Boosting the Performance of Remote GPU Virtualization Using InfiniBand Connect-IB and PCIe 3.0

C. Reaño*, F. Silla*, A. J. Peña†, G. Shainer‡, S. Schultz‡, A. Castelló§, E. S. Quintana-Ortí§, and J. Duato*
* Universitat Politècnica de València / València, Spain 46022 / carregon@gap.upv.es, {fsilla, jduato}@disca.upv.es
† Argonne National Laboratory / Argonne, IL 60439, USA / apenya@mcs.anl.gov
‡ Mellanox Technologies / Sunnyvale, CA 94085, USA / {shainer, scots}@mellanox.com
§ Universitat Jaume I / Castellón, Spain 12071 / {adcastel, quintana}@icc.uji.es

***Abstract*—A clear trend has emerged involving the acceleration of scientific applications by using GPUs. However, the capabilities of these devices are still generally underutilized. Remote GPU virtualization techniques can help increase GPU utilization rates, while reducing acquisition and maintenance costs. The overhead of using a remote GPU instead of a local one is introduced mainly by the difference in performance between the internode network and the intranode PCIe link. In this paper we show how using the new InfiniBand Connect-IB network adapters (attaining similar throughput to that of the most recently emerged GPUs) boosts the performance of remote GPU virtualization, reducing the overhead to a mere 0.19% in the application tested.**

I. Introduction

A clear trend has emerged involving the acceleration of scientific applications by using graphics processing units (GPUs) in domains as diverse as chemical physics [1], finance [2], and image analysis [3]. However, the GPU computing resources are still generally underutilized. Remote GPU virtualization techniques can help increase GPU utilization rates, while reducing acquisition and maintenance costs. For these reasons, many different virtualization solutions are available today, such as rCUDA [4], [5], SnuCL [6], GVirtuS [7], DS-CUDA [8], and VOCL [9].

The overhead of using a remote GPU instead of a local one is introduced mainly by the difference in performance between the internode network and the intranode PCIe link. In this paper we show how using the new InfiniBand Connect-IB network adapters, making use of PCIe 3.0 x16 connections like the most recent GPUs (see Figure 1), by means of dual InfiniBand FDR ports, boosts the performance of remote GPU virtualization, reducing its overhead to negligible values.

The rest of the paper is organized as follows. Section II introduces rCUDA, the solution for remote GPU virtualization used in this study. In Section III we study the impact of the new Connect-IB network adapters on the performance of rCUDA. Finally, Section IV summarizes the main conclusions of our work.

II. rCUDA: Remote CUDA

Several solutions exist for using GPUs in the scope of general-purpose computing on GPUs (GPGPU), being the most popular CUDA [10] and OpenCL [11]. In this work, we use CUDA because it is a more mature and widely used

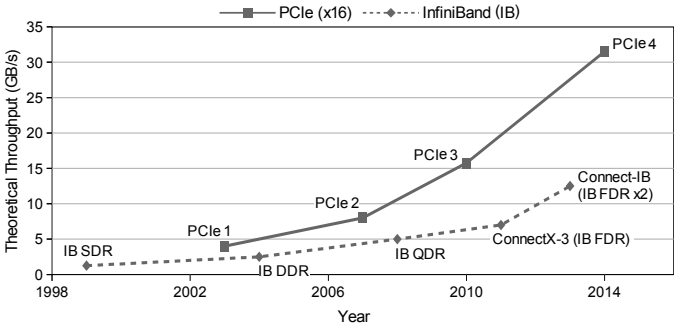

Fig. 1: Comparison between the theoretical bandwidth of different versions of PCI Express x16 and those of commercialized InfiniBand fabrics and network adapters.

technology. Among the remote GPU virtualization frameworks mentioned in the introduction, we use rCUDA [4], [5] because it offers the most extensive features.

Remote CUDA, or rCUDA, is compatible with CUDA 5.5 and supports different communication technologies, such as TCP/IP and InfiniBand. It is organized following a client-server distributed architecture. When an application running on the client side requests the use of a remote GPU, this is intercepted by the rCUDA client and transparently forwarded to the rCUDA server by means of the communication layer. The request is then issued to the physical GPU; upon completion, the result is sent back to the rCUDA client and, finally, to the initial application. More detailed information about rCUDA can be found at www.rCUDA.net and [12].

III. Impact of Connect-IB on rCUDA

For all the experiments shown in this section, we have used two servers, each equipped with two Intel Xeon hexa-core E5-2680 v2 processors (2.8 GHz, 32 GB RAM) and Mellanox ConnectX-3 and Connect-IB InfiniBand FDR adapters. Additionally, one of the nodes had an NVIDIA Tesla K40c GPU.

First, we use the *bandwidthTest* benchmark available in NVIDIA CUDA Samples [13] for comparing the bandwidth when using a local GPU and a remote one.

Figure 2 presents the bandwidth for copies from host pinned memory to device memory, using native CUDA on a local GPU and the rCUDA framework over InfiniBand employing different cards: ConnectX-3 and Connect-IB (with

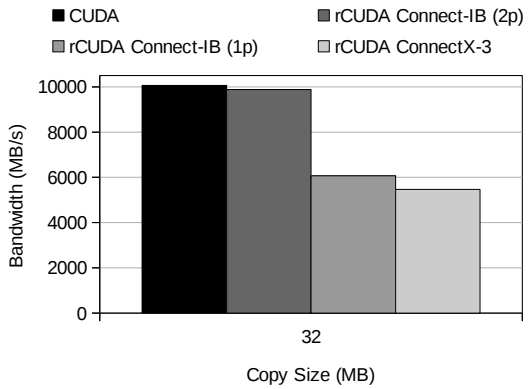

Fig. 2: Bandwidth test for copies from host pinned memory to device memory, using CUDA and the rCUDA framework over InfiniBand employing different cards: ConnectX-3 and Connect-IB (with 1 and 2 ports).

1 and 2 ports). As can be seen, the bandwidth obtained when using a remote GPU with the new dual-port Connect-IB card is 98.23% of the total bandwidth achieved when using a local GPU. This is a huge improvement over the bandwidth attained by the previous ConnectX-3 adapter (only 54.38% of the local GPU bandwidth) and enables the wide use of remote accelerators.

Second, we study how the new characteristics of Connect-IB influence the performance of real applications, using the CUDA-MEME [14] application as an example.

To present the results in this section in their context, we first show in Figure 3(a) the time spent in transfers (i.e., time spent in memory copies between host memory and the device memory) and the time employed by computations (i.e., time employed by CUDA kernels) of CUDA-MEME. As can be seen, the behavior of this application is representative of most CUDA applications: moving data to the GPU and performing intensive computations there to compensate for the overhead of having transferred the data across the PCIe bus. Obviously, when using remote GPUs, more computations are needed to compensate for the additional overhead of also transferring the data through the network. In this case, the time spent in computations (261.36 seconds) seems large enough to compensate for the time employed by transfers (5.18 seconds).

In Figure 3(b) we depict the CUDA-MEME execution time for a dataset containing 1,000 sequences using the DNA alphabet, choosing the OOPS model for motif distribution and with a maximum of 1 million sites for each motif. In this figure, we expose the execution time using CUDA and rCUDA with different network adapters. The overhead of using rCUDA in these experiments is 0.19%, 0.56%, and 0.61%, when using Connect-IB 2 ports, Connect-IB 1 port, and ConnectX-3, respectively. These results are aligned with those obtained in the previous experiment and confirm that the overhead of remote GPU virtualization is negligible when using the new InfiniBand Connect-IB dual-port adapters.

IV. Conclusions

In this paper we have shown how using the new InfiniBand Connect-IB network adapters (with similar performance to that of the most recent PCIe 3.0 GPUs) boosts the performance of remote GPU virtualization. The result is a reduction of the overhead to a mere 0.19% in the application tested.

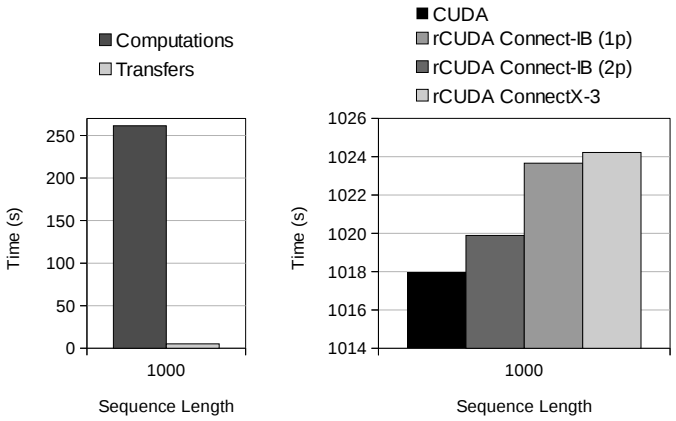

Fig. 3: **(a)** NVIDIA profiling results for CUDA-MEME, in particular, time employed by computations (i.e., CUDA kernels) and memory transfers (i.e., CUDA memory copies). **(b)** CUDA-MEME execution time using CUDA and the rCUDA framework over InfiniBand employing different cards: ConnectX-3 and Connect-IB (with 1 and 2 ports).

Acknowledgments

This work was funded by the Generalitat Valenciana under Grant PROMETEOII/2013/009 of the PROMETEO program phase II. This material is based upon work supported by the U. S. Department of Energy, Office of Science, Advanced Scientific Computing Research (SC-21), under Contract No. DE-AC02-06CH11357. Authors from the Universitat Politècnica de València and Universitat Jaume I are grateful for the generous support provided by Mellanox Technologies.

References

[1] D. P. Playne *et al.*, "Data parallel three-dimensional Cahn-Hilliard field equation simulation on GPUs with CUDA," in *PDPTA*, 2009.

[2] A. Gaikwad *et al.*, "GPU based sparse grid technique for solving multidimensional options pricing PDEs," in *WHPCF*, 2009.

[3] Y. C. Luo and R. Duraiswami, "Canny edge detection on NVIDIA CUDA," in *CVPRW*, 2008.

[4] A. J. Peña *et al.*, "Performance of CUDA virtualized remote GPUs in high performance clusters," in *ICPP*, 2011.

[5] C. Reaño *et al.*, "Influence of InfiniBand FDR on the performance of remote GPU virtualization," in *Cluster*, 2013.

[6] J. Kim *et al.*, "SnuCL: an OpenCL framework for heterogeneous CPU/GPU clusters," in *ICS*, 2012.

[7] G. Giunta *et al.*, "A GPGPU transparent virtualization component for high performance computing clouds," in *Euro-Par*, 2010.

[8] M. Oikawa *et al.*, "DS-CUDA: a middleware to use many GPUs in the cloud environment," in *SC*, 2012.

[9] S. Xiao *et al.*, "VOCL: an optimized environment for transparent virtualization of graphics processing units," in *InPar*, 2012.

[10] NVIDIA, *CUDA C Programming Guide 5.5*, 2013.

[11] Khronos OpenCL Working Group, *OpenCL 1.2 Specification*, 2011.

[12] A. J. Peña, "Virtualization of accelerators in high performance clusters," Ph.D. dissertation, Universitat Jaume I de Castellón (Spain), 2013.

[13] NVIDIA, *CUDA Samples Reference Manual 5.5*, 2013.

[14] Y. Liu *et al.*, "CUDA-MEME: accelerating motif discovery in biological sequences using CUDA-enabled graphics processing units," *Pattern Recognition Letters*, 2010.