Bandwidth Enhancement between Graphics Processing Units on the Peripheral Component Interconnect Bus

ANTON Alin

Politehnica University of Timisoara, Romania,
Department of Computing, Faculty of Automatic Control and Computing,
2nd Vasile Parvan Ave., 300223 Timisoara, Romania,
alin.anton@cs.upt.ro

***Abstract* – General purpose computing on graphics processing units is a new trend in high performance computing. Present day applications require office and personal supercomputers which are mostly based on many core hardware accelerators communicating with the host system through the Peripheral Component Interconnect (PCI) bus. Parallel data compression is a difficult topic but compression has been used successfully to improve the communication between parallel message passing interface (MPI) processes on high performance computing clusters. In this paper we show that special purpose compression algorithms designed for scientific floating point data can be used to enhance the bandwidth between 2 graphics processing unit (GPU) devices on the PCI Express (PCIe) 3.0 x16 bus in a homebuilt personal supercomputer (PSC).**

***Keywords:* bandwidth; compression; GPU; PCI.**

I. INTRODUCTION

Modern day computer applications require powerful hardware accelerators. The GPU is the most affordable accelerator device. Not only that GPUs are already present in most of the commodity including the portable computing platforms, but high performance equivalents are being sold on the market as development versions of their cluster counterparts.

Applications vary from electric design and automation, computational fluid dynamics and machine learning, to computational finance, bioinformatics and defense [1].

In spite of the popular claims and the approaches based on the Top500 High Performance LINPACK (HPL) benchmark [2], the overall performance of a general-purpose graphics card processor is hard to estimate outside the scope of the software application. The HPL source code is customized for the target hardware platform and only parts of the actual computing are performed on the GPU devices [3].

Besides that, the programming paradigm is also important. Starting with CUDA 4.0 it is possible to transfer chunks of memory directly from one GPU device to another [4].

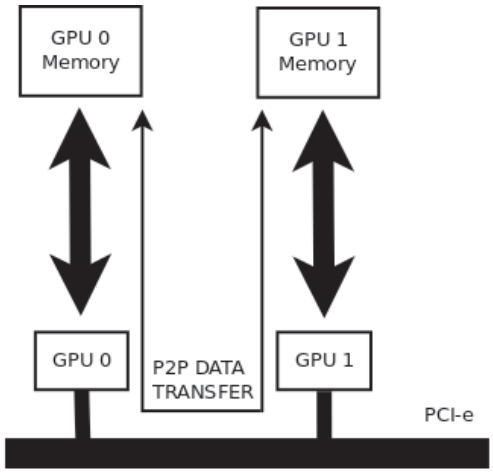

Fig.1 Peer to peer communication between two GPU devices on the PCI bus

The OpenACC standard also allows for the #pragma acc update peer directive [5].

Fig.1 shows the dataflow in a computer system with multiple GPUs.

Data compression is not a good candidate for parallelization. Parallel implementations of the Burrows-Wheeler (BW) [6] and Lemper-Ziv-Storer-Szymanski (LZSS) [7] algorithms exist. However, compression is most efficient on large blocks of uncompressed data or when designed in terms of stream computing – where GPUs excel.

"GFC" is a floating point data compression algorithm implemented on GPU devices [8]. It works for IEEE-754 double precision floating point numbers used in scientific numerical computing [9].

The compression of MPI messages between nodes has been successfully demonstrated in [10]. In [11] the authors present an MPI traffic replay compression technique.

The objective of the present work is to compress the Peer to Peer (P2P) data exchanged between two identical GPU devices in order to enhance the bandwidth between the cards.

"GFC" is applied in order to compress the data being transferred between two identical GPU devices sharing the same PCIe bus.

The paper is organized as follows: Section II describes the experimental setup, Section III presents the results and discusses their impact and Section IV concludes the paper.

II. EXPERIMENTAL SETUP

Fig. 2 shows the two identical GPU devices in their PCIe v3.0 [13] lanes. Both lanes have 16 wires multiplexed by a controller in the motherboard.

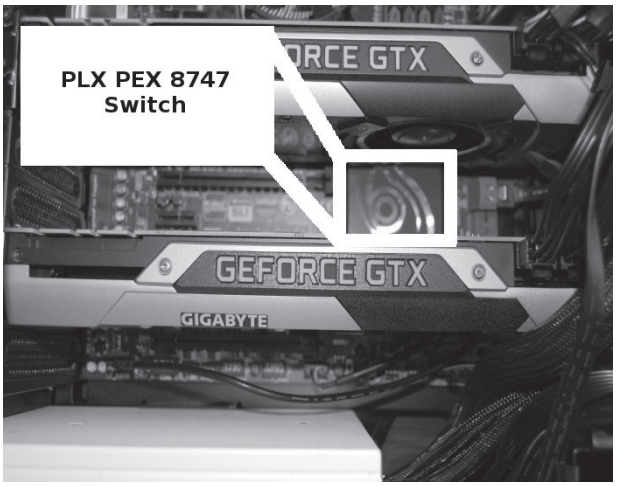

Fig.2 The GPU devices and the interceding PLX PEX 8747 switch "hidden" in the motherboard

The TITAN Black cards each have 2880 cores clocked at 890 Mhz with 6 GB GDDR5 memory clocked at 7 Gbps.

Each of the card operates in a range of 70—95 Celsius degrees. This requires for special cooling provided by 6 powerful fans and a very stable dual power source.

Compared to the regular video cards, the TITAN Black is designed as a development card for the TESLA equivalents. The most important difference is the lack of error correcting codes for the internal memory, making it unfeasible to group many of them together. All cores

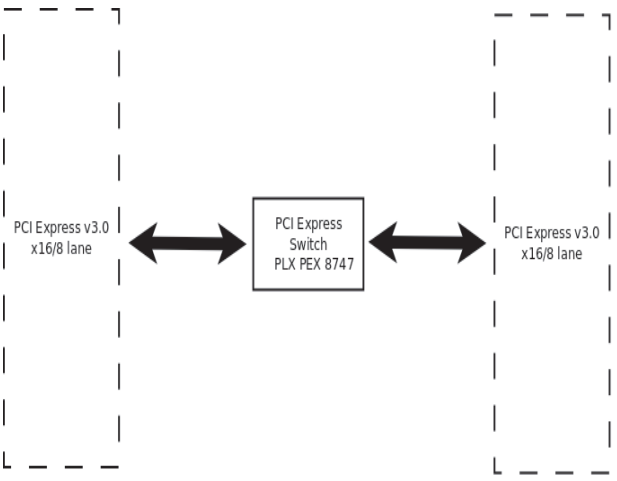

Fig. 3 PCI Express Switch multiplexer for x16 lanes

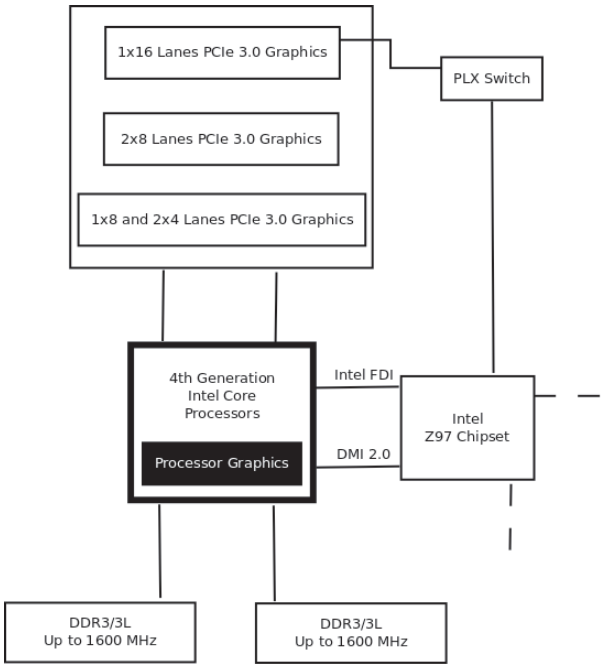

Fig.4 PLX Switch controlled by Z97 Chipset helps 4th Generation Processors to provide 2x16 PCIe 3.0 Lanes

are capable of double precision floating point arithmetic which is only relevant for scientific computing. Single precision is 4 times as fast as double precision when 32 bits arithmetic is enough.

Fig.3 is a schematic of Fig.2 showing how the Z97 Gigabyte motherboard provides PCIe v3.0 lanes using a multiplexer. The board is only capable of handling 1 x16 lane on the PCIe v3.0 bus. When two cards are in place, the PLX PEX 8747 multiplexer is used to switch between the lanes as necessary. At least in theory the speed of the PLX switch causes the multiplexing to have a negligible impact on the bandwidth of the lanes, thus providing for 2 genuine x16 slots when nothing else uses the 16 wire bus.

Fig.4 shows all the possible primary PCIe configurations supported by the Z97 chipset. The main lanes are connected directly to the LGA1150 processor socket balls.

Fig.5 shows the application of the "GFC" algorithm on each GPU device for compression. Data being sent from GPU_0 is compressed with "GFC" installed on GPU_0. Afterwards the block of data is decompressed using "GFC" installed on the second device GPU_1.

"GFC" uses CUDA kernels for compression and decompression.

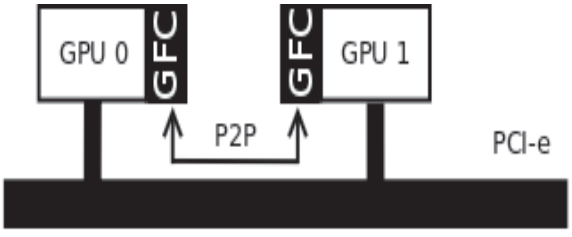

Fig. 5 Peer to Peer (P2P) "GFC" compression

The cudaMemcpy() call is used to transfer data from one device to another using CUDA Unified Virtual Addressing (UVA):

```
cudaSetDevice(gpu0);
Compress(blocks, warpsperblock, dimensionality);
cudaMemcpy(gpu0_compressed, gpu1_compressed,
        MAX*sizeof(double), cudaMemcpyDefault);
cudaSetDevice(gpu1);
Decompress(blocks, warpsperblock, dimensionality);
```

Two datasets are used, one provided by [8] consisting of intercepted MPI LU factorization traffic with a size of 185 Mb, and a random generated archive of 512 Mb.

The LU factorization data is of real practical interest since dense matrix factorizations can be performed in parallel on two GPU devices, the upper part on one unit and the lower part on the other.

The LGA1150 microprocessor socket supports PCIe v3.0 having occupied lanes as 1x16, 2x8, 1x8 + 2x4. This is why the PLX multiplexer switches the 16 wires from one lane to another as necessary. It allows the producer to claim full 2x16 PCIe v3.0 support.

III. RESULTS AND DISCUSSION

The measured P2P bandwidth between the two GPU devices without compression is shown in Fig.6. All measurements are performed 100 times in both directions, first from GPU0 to GPU1 then from GPU1 to GPU0.

Time is measured in milliseconds using the CUDA directives. The theoretical maximum bandwidth for a PCIe v3.0 x16 lane is that of 32 Gb/s.

Fig.6 shows that barely half of the bandwidth is used in spite of the multiplexer and that the data rate increases along with the size of the transfer buffer.

The highest throughput is at 12.6 Gb/s when copying 4Gb from one device to another.

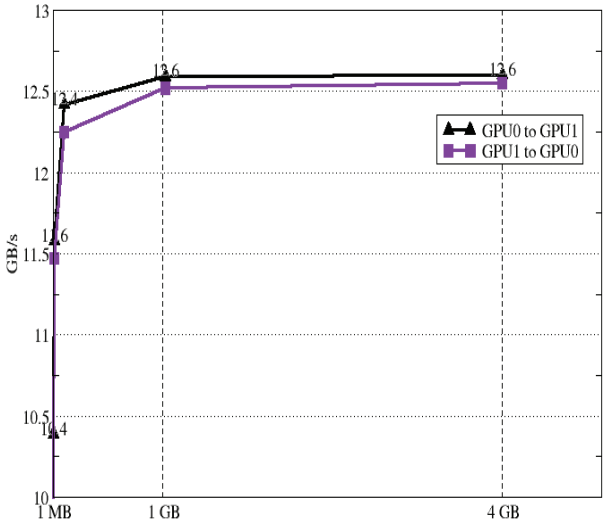

Fig.6 P2P data rate without "GFC" depending on the amount of traffic between the devices

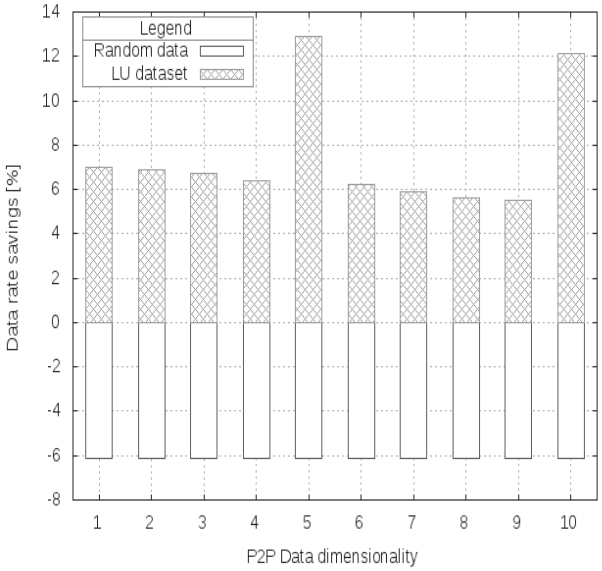

Fig.7 Data rate savings related to dataset dimensionality

The data rate saving (DRS) is given by equation (1).

$$DRS = 1 - \frac{\text{compressed data rate}}{\text{uncompressed data rate}} \quad (1)$$

This shows that the devices are optimized for heavy loads. Another observation is that data copied from GPU1 to GPU0 moves a little bit slower than from GPU0 to GPU1. This is a slot proximity issue because the GPU devices are identical.

Given the best data rate from Fig.7 of DRS=1.29 for dimensionality=5, the improvement over the 12.6 GB/s from Fig.6 is of an additional 3.65 GB/s.

When "GFC" is applied with LU traffic between the cards, the best data rate is at 16.25 GB/s which is higher than what PCIe v2.0 can produce for a single card.

One can conclude that when the multiplexer is challenged with 2 active devices at the same time, the chip is splitting the lanes between the devices. This is not really helpful in terms of performance improvements but during direct peer to peer transfers both devices are active at the same time.

Also, the data rate when compressing random data is heavily impacted by a negative improvement of -6.15%. That is, the data rate decreases with 793.6 MB/s for dimensionality=5.

"GFC" uses the dimensionality parameter to take into account the dimensionality of the input data when predicting the next values from the stream. This is why in Fig.7 it can be shown that "GFC" gets really good when the predictor based compressor uses a dimensionality parameter that corresponds to the dataset dimensionality or is a multiple of it.

In [12] the authors used "FPC", the original implementation of "GFC" for the compression of MPI messages between processor nodes. The impact of using "GFC"-like algorithms for bandwidth enhancement on GPU devices is very high, because they can be easily synthesized with field programmable gate arrays

(FPGAs) or embedded in the design of future GPU cards.

IV. CONCLUSION

It has been shown that scientific floating point compression can be used to improve the bandwidth between PCIe GPU accelerators. While scientific floating point compression is not new the use of scientific floating point compression to improve the P2P PCIe bandwidth between GPU accelerators was introduced by the presented work. The "GFC" implementation is used to demonstrate P2P traffic compression between 2 GPU accelerators.

The PCIe bus is still one of the main interconnect links within a computer system, connecting the central processor units with peripheral extension cards, such as graphics processors. The bandwidth of PCIe has evolved from a few hundred MB/s up to the 32 GB/s of the v3.0 standard.

The presence of the lane PLX PEX 8747 switch interceding between the two identical GPU cards has been shown to produce a serious negative impact. The circuit evidently holds back the cards from communicating with each other at data rates far less than the 32 GB/s lane capacity. However more measurements are necessary in order to fully assess the impact on P2P performance.

This work is a proof of concept demonstrating an additional 3.65 GB/s for peer to peer LU decomposition traffic, between identical GPU devices. For an application that takes one hour to complete an additional 3.65 GB/s is significant. Compression should only be enabled when it is efficient for the dataset. Random data for instance, in this case, decreases the data rate savings by 6.15%.

Future work should implement PCIe scientific floating point compression in FPGA and use the FPGA to intercede between the GPU processors with the help of riser cards and PCIe extension cables. Different scientific floating point compression algorithms can be tested and combined.

Special-purpose accelerator compilers and libraries should be used to automate the process of integrating scientific floating point and integer compression with device to host and device to device communication.

An adaptive solution should be used to only activate compression when it would benefit the data rate savings and in order to choose the best algorithm.

ACKNOWLEDGEMENTS

The author acknowledges and thanks Dr. Martin Burtscher for providing the "GFC" implementation and the LU dataset [8].

This work was partially supported by the strategic grant POSDRU/159/1.5/S/137070 (2014) of the Ministry of National Education, Romania, co-financed by the European Social Fund – Investing in People, within the Sectorial Operational Programme Human Resources Development 2007-2013.

REFERENCES

[1] NVIDIA GPU-Accelerated Applications Catalog, March, 2015.

[2] J. Dongarra. 1987. The LINPACK Benchmark: An Explanation. In Proceedings of the 1st International Conference on Supercomputing, Elias N. Houstis, Theodore S. Papatheodorou, and Constantine D. Polychronopoulos (Eds.). Springer-Verlag, London, UK, UK, 456-474.

[3] D. Rohr, M. Bach, M. Kretz, and V. Lindenstruth. 2011. "Multi-GPU DGEMM and High Performance Linpack on Highly Energy-Efficient Clusters." *IEEE Micro* 31, 5 (September 2011), 18-27. DOI=10.1109/MM.2011.66.

[4] NVIDIA CUDA C Programming Guide v7.0, March, 2015.

[5] The OpenACC™ Application Programming Interface v2.0, June, 2013.

[6] J. A. Edwards, U. Vishkin, Parallel algorithms for Burrows–Wheeler compression and decompression, Theoretical Computer Science, Volume 525, 13 March 2014, Pages 10-22, ISSN 0304-3975.

[7] B. Zhou, H. Jin, and R. Zheng, "A Parallel High Speed Lossless Data Compression Algorithm in Large-Scale Wireless Sensor Network," International Journal of Distributed Sensor Networks, vol. 2015, Article ID 795353, 12 pages, 2015. doi:10.1155/2015/795353.

[8] M. A. O'Neil and M. Burtscher. "Floating-Point Data Compression at 75 Gb/s on a GPU." Fourth Workshop on General Purpose Processing on Graphics Processing Units. March 2011.

[9] D. Goldberg. 1991. What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. 23, 1 (March 1991), 5-48. DOI=10.1145/103162.103163.

[10] R. Filgueira, D. E. Singh, J. Carretero, A. Calderon, and F. Garcia. 2011. Adaptive-CoMPI: Enhancing Mpi-Based Applications - Performance and Scalability by using Adaptive Compression. Int. J. High Perform. Comput. Appl. 25, 1 (February 2011), 93-114.DOI=10.1177/1094342010373486.

[11] A. Anton, V. Cretu, A. Ruprecht, S. Muntean, "Traffic Replay Compression (TRC): a highly efficient method for handling parallel numerical simulation data", Proceedings of the Romanian Academy, Series A: Mathematics, Physics, Technical Sciences, Information Science, Vol. 14(4), pp. 385-392, 2013, ISSN 1454-9069.

[12] R. Filgueira, M. Atkinson, A. Nunez, and J. Fernandez. 2012. An adaptive, scalable, and portable technique for speeding up MPI-based applications. In Proceedings of the 18th international conference on Parallel Processing (Euro-Par'12), Christos Kaklamanis, Theodore Papatheodorou, and Paul G. Spirakis (Eds.). Springer-Verlag, Berlin, Heidelberg, 729-740.DOI=10.1007/978-3-642-32820-6_72.

[13] PCI Express® Base Specification Revision 3.0 November 10, 2010.