Communication Papers of the Federated Conference on Computer Science and Information Systems pp. 105–110
DOI: 10.15439/2018F17
ISSN 2300-5963 ACSIS, Vol. 17

# Benchmarking overlapping communication and computations with multiple streams for modern GPUs

Pawel Czarnul
Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology
Narutowicza 11/12, 80-233 Poland
Email: pczarnul@eti.pg.edu.pl

***Abstract*—The paper presents benchmarking a multi-stream application processing a set of input data arrays. Tests have been performed and execution times measured for various numbers of streams and various compute intensities measured as the ratio of kernel compute time and data transfer time. As such, the application and benchmarking is representative of frequently used operations such as vector weighted sum, matrix multiplication etc. The paper shows benefits of using multiple data streams for various compute intensities compared to one stream, benchmarked for 4 GPUs: professional NVIDIA Tesla V100, Tesla K20m, desktop GTX 1060 and mobile GeForce 940MX. Additionally, relative performances are shown for various numbers of kernel computations for these GPUs.**

## I. Introduction

GENERAL purpose programming on GPUs (GPGPU) has become an effective approach to parallelization of many real world problems such as simulation of phenomena in 2D, 3D spaces, numerical computations, image and video processing etc. Nowadays, NVIDIA CUDA, OpenCL and OpenACC are the three dominant APIs for GPUs. Typically, several techniques are used for optimization of such programs [1]. These include, described in CUDA terms: optimization of memory referencing – global memory coalescing, proper shared memory accesses with consideration of memory banks, minimization of thread divergence, overlapping host to device communication, kernel execution and device to host communication, data prefetching from global memory to shared memory using registers, loop unrolling [2], [3]. Such changes can improve execution times significantly, at least several times and are of high importance consequently. This paper analyzes the impact of communication between the host and the device, kernel execution and device to host communication using various numbers of streams [4], for 4 various GPU models, compared to a single stream implementation for an application that can be regarded as a template useful for processing of sets of input data packets such as operations on vectors or matrices that are building blocks for many applications.

## II. Related work

CUDA application and system models, numerous examples and typical aforementioned optimizations are discussed in the literature [2], [3], also from the point of view of power/performance efficiency of different optimizations [5]. The particular problem addressed in this work can be applied to any GPU application that processes a sequence of independent input data sets for which communication and computations can be overlapped, for example a sequence of matrix multiplications, block-based matrix multiplication, computing similarities among a large number of multidimensional vectors [6] etc. Furthermore, results from this study can also be incorporated into frameworks that can automatically parallelize computations performed in batches. This is the case, for instance, for KernelHive [7] that can schedule computations and manage input and output data and run kernels on compute devices such as GPUs and CPUs. In this case the proposed technique allows overlapping communication and computations. Furthermore, the results for various compute intensities can be embedded into GPU processing models, also incorporating overlapping communication and computations, in modeling and simulation systems such as MERPSYS [8].

The matter considered in this work was found important before in the context of analysis of the optimal number of streams for best execution time in terms of the number of iterations of a loop within a kernel [9]. Analysis and a model were proposed for older cards with CUDA compute capabilities 1.x and 2.x. Practical tests, benchmarking were performed for older GPUs such as GeForce GTX 280 and GTX 480. Finally, an analytical formula was proposed to find the best number of streams. Compared to those results, we focus on current generation cards by benchmarking outcomes experimentally, plot results as relative gains compared to a single stream for various compute intensities showing minima and compare performances of 4 modern GPUs of various types – professional, desktop and mobile and not just desktop series cards.

Using multiple streams can speed up computations on a GPU or GPUs for several applications. For instance, in [10] it was shown how a multithreaded application using multiple streams increases the frame rate performance up to 78 frames per second compared to 36 for a single stream implementation for the application of real-time ultrasound elastography

suitable for diagnosis and treatment of cancer.

The authors of [11] investigated and modeled processing on a GPU using streams in the context of implementing a push based DBMS called G-SDMS. They focused especially on runtime resource scheduling using streams. Tests and verification of scheduling algorithms were performed on NVIDIA GTX680 and Tesla K40.

The authors of [12] showed how using multiple CUDA streams with multiple OpenMP threads allows to improve current state-of-the-art communication performance in an environment with multiple GPUs for a representative 3D stencil example. Tests were performed on Tesla K20, GTX590 and Tesla C2050.

In [13], the authors have presented a HCLOOC library which enables to write data-parallel kernels for accelerators including GPUs and overlapping host-accelerator data transfer with kernel execution.

## III. Model and approach

The application model considered in this work assumes a sequence of independent operations performed on various input data chunks. Specifically, each operation takes two data chunks as input and produces one data chunk as its output. This is depicted in Figure 1. Such a model, with such data sizes, corresponds to frequently performed operations such as vector weighted addition, matrix multiplication, blending images etc. that only differ in the ratio of computations to (host-to-device and device-to-host) communication times. In this paper, this ratio is varied in order to assess benefits of using more than 1 stream (2 and 4 tested) for various GPUs, from professional through desktop up to mobile chips. More streams allow overlapping communication and computations and consequently minimization of application execution time. Results allow to assess benefits of using multi-stream code versions for a given type of card and compute intensity of the analyzed application.

## IV. Experiments

### A. Testbed environments

Within this work, we evaluate the proposed approach for 4 different, modern GPUs that differ in the target market segments:

- server data center oriented Tesla V100 and K20m,
- desktop GeForce GTX 1060,
- mobile GeForce 940MX

as well as CUDA compute capabilities:

- 7.0 – Tesla V100,
- 6.1 – GeForce GTX 1060,
- 5.0 – GeForce 940MX,
- 3.5 – Tesla K20m.

This means that the tests can be considered as representative both in terms of GPU types and compute capabilities. Specific parameters of the GPUs are listed in Table I.

### B. Tests and results

For each of the GPUs, tests were performed for various compute intensities and various numbers of streams: 1, 2 and 4. Additionally, two ways of issue orders were tested for 2 and 4 streams:

- order A – n $\times$ { host to device copy, kernel launch, device to host copy },
- order B – n $\times$ host to device copy, n $\times$ kernel launch, n $\times$ device to host copy.

Compute intensity is defined as the ratio of kernel compute time for each data packet divided by each data packet size, measured for 1 stream. We should note that such an application and various compute intensities are representative of many real applications because:

1) There are many applications that take input data of size $2n$ and produce output of size $n$. Typical examples include matrix multiplication and vector addition, used in numerous codes. Notable applications using such operations nowadays include, for example, machine learning [14], weather pattern analsis [15], shortest path problem etc. [16].
2) Various compute intensities correspond to various operations such as: higher for matrix multiplication, lower for addition of vectors etc.

Figures 2 through 9 present comparison of execution times obtained for various compute intensities for all the GPUs tested, expressed both in terms of actual execution times as well as relative times compared to 1 stream versions.

From the latter we can easily see distinct features that impact potential uses of these results for actual applications:

- Except the 940MX, there is about 20-30% gain in using 2, 4 streams compared to 1 stream for small compute intensities (<1).
- There is an increase of gain from using 2, 4 streams up to around 50%, except the 940MX for which the gain reaches around 35%.
- Gain from using 2, 4 streams diminishes as the compute intensity grows.
- There is a gain from using 4 streams compared to 2 streams for compute intensities up to around 2-4. The average gains are as follows for particular GPUs:
  - Tesla K20m – 4.9%,
  - GTX 1060 – 3.5%,
  - GeForce 940MX – 3.3%,
  - Tesla V100 – 4.5%.

Minima observed are due to the fact that too low and too high compute intensities do not allow considerable overlapping of communication and computations.

In terms of comparisons of isssue orders, Tables II and III present differences between the aforementioned B and A orders in percentages, out of minimums out of 5 runs for each configuration. It can be seen that differences for all the GPU tested do not exceed 1.4% which means that are negligible. Configurations with the same number of computations in the

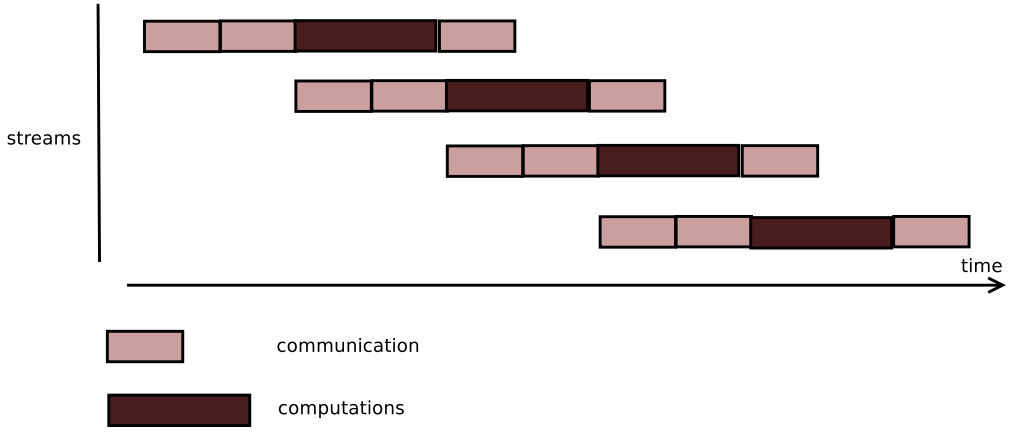


Figure 1. Streams

Table I
GPUs – SPECIFICATIONS

| GPU | Tesla V100 | Tesla K20m | GTX 1060 | GeForce 940MX (GDDR5) |
|---|---|---|---|---|
| **Architecture name** | Volta | Kepler | Pascal | Maxwell |
| **CUDA compute capability** | 7.0 | 3.5 | 6.1 | 5.0 |
| **CUDA cores** | 5120 | 2496 | 1280 | 512 |
| **clock [MHz]** | 1455 | 706 | 1708 | 861 |
| **memory size [GB]** | 16 | 5 | 6 | 4 |
| **memory bus width [bits]** | 4096 | 320 | 192 | 64 |
| **memory bandwidth [GB/s]** | 900 | 208 | 192 | 40.1 |
| **power [W]** | 300 | 225 | 120 | 23 |

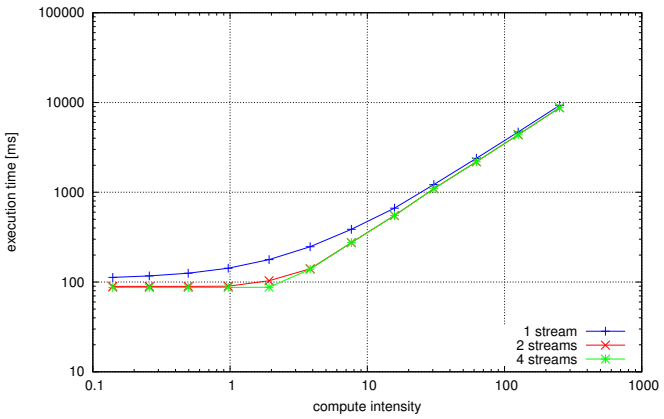


Figure 2. Execution times on Tesla V100

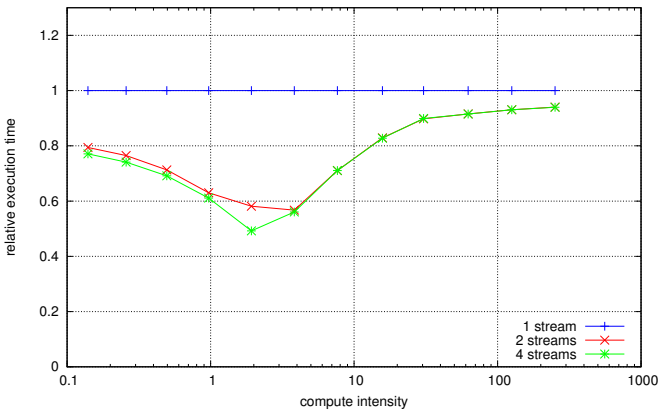


Figure 3. Relative execution times on Tesla V100

kernel (called relative computation count, not to be confused with the assumed definition of compute intensity) were compared against each other for all the GPUs tested.

Furthermore, we have compared relative performance per data size of the GPUs, proportional to the inverse of application execution time, for the same relative computation counts. Performances were scaled against the smallest configuration on the slowest GPU out of the tested ones – GeForce 940MX. Results are shown in Figure 10. It can be seen that performances reach their maximum for certain computation counts and stay such for larger computation counts. Relative order of the GPUs is, from the fastest: V100, GTX 1060, Tesla K20m and 940MX. It should be noted that relative performances may depend on the kernel code, which in this case computes a weighted sum of two input vectors, with the number of computations that can be defined for testing various compute intensities.

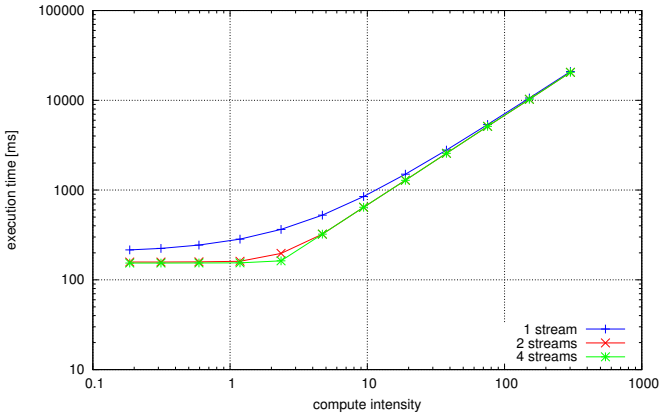


Figure 4. Execution times on Tesla K20m

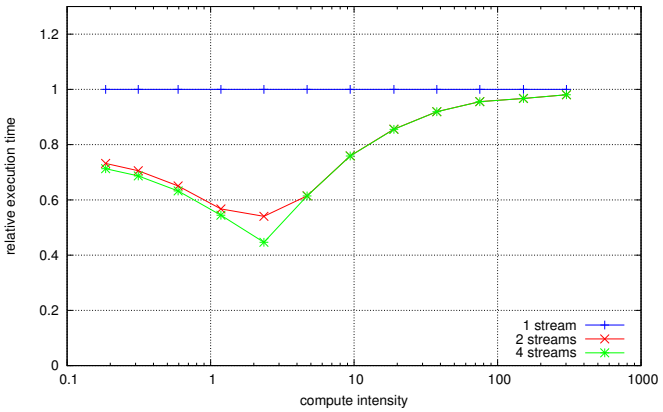


Figure 5. Relative execution times on Tesla K20m

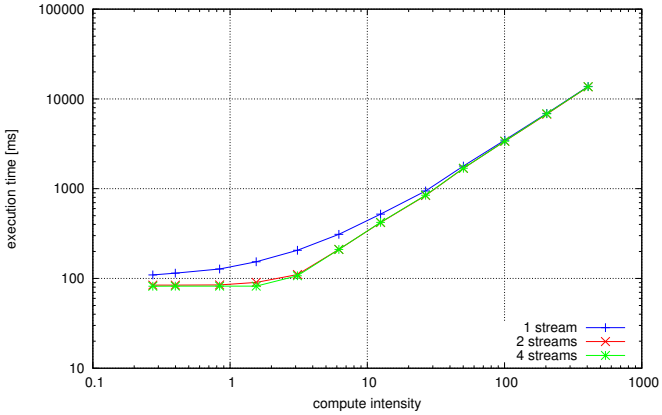


Figure 6. Execution times on GTX 1060

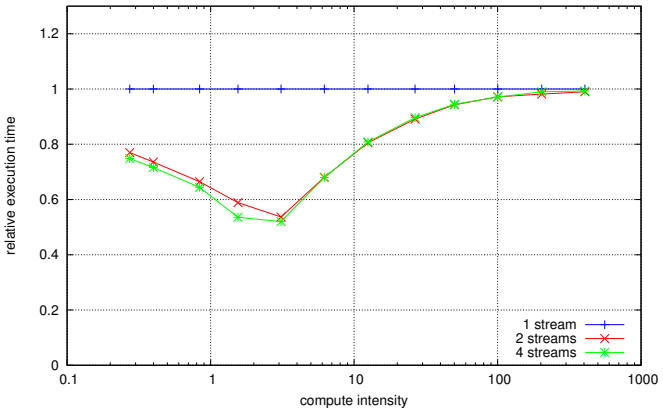


Figure 7. Relative execution times on GTX 1060

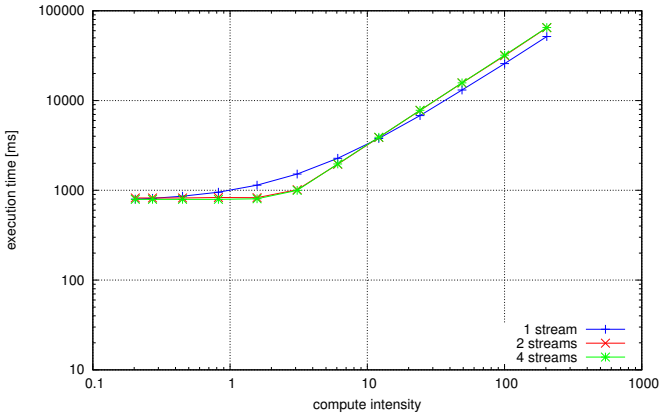


Figure 8. Execution times on GF 940MX

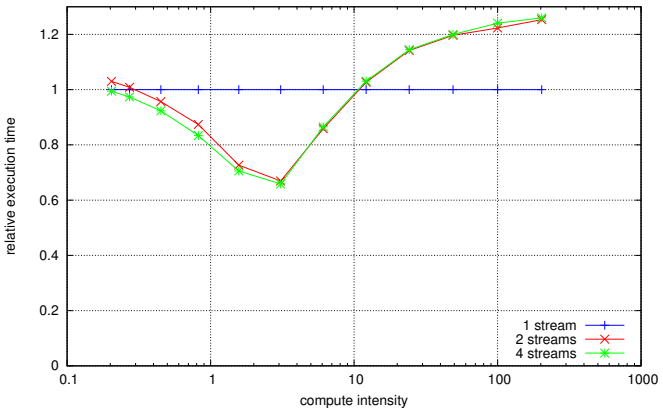


Figure 9. Relative execution times on GF 940MX

## V. Summary and Future Work

In the paper, we have compared execution times of an application processing multiple data chunks on a GPU, in several versions that differ in the number of CUDA streams used. As expected using 2 and 4 streams brings benefits compared to 1 stream. A non-trivial maximum of gains out of using multiple streams has been shown in terms of various compute intensities for 4 GPUs tested including the latest professional NVIDIA V100, desktop GTX 1060, professional Tesla K20m and mobile GeForce 940MX. It has been shown that gain from using 4 streams compared to 2 streams is visible up to compute intensities of around 2-4. Additionally, performances

Table II
B VS A ISSUE ORDER DIFFERENCE [%] – 2 STREAMS

| relative computation count | Tesla V100 | Tesla K20m | GTX 1060 | GeForce 940MX |
|---|---|---|---|---|
| 1.00 | 0.00 | 0.06 | 0.00 | -0.04 |
| 2.00 | -0.45 | 0.06 | 0.00 | 0.00 |
| 4.00 | -0.11 | 0.13 | -0.24 | 0.01 |
| 8.00 | -0.34 | 0.37 | -0.11 | -0.35 |
| 16.00 | -0.22 | 0.36 | 0.09 | 0.00 |
| 32.00 | 0.00 | 0.25 | 0.00 | -0.01 |
| 64.00 | 0.11 | 0.12 | -0.05 | 0.00 |
| 128.00 | -0.10 | 0.05 | -0.23 | 0.03 |
| 256.00 | -0.07 | 0.04 | -0.02 | 0.17 |
| 512.00 | 0.40 | 0.00 | -0.04 | 0.12 |
| 1024.00 | -0.22 | 0.01 | -0.01 | 0.76 |
| 2048.00 | 0.14 | 0.01 | 0.00 | 0.56 |

Table III
B VS A ISSUE ORDER DIFFERENCE [%] – 4 STREAMS

| relative computation count | Tesla V100 | Tesla K20m | GTX 1060 | GeForce 940MX |
|---|---|---|---|---|
| 1.00 | 0.00 | -0.07 | 0.12 | -0.06 |
| 2.00 | 0.00 | -0.06 | -0.12 | -0.04 |
| 4.00 | -0.12 | -0.06 | -0.12 | -0.05 |
| 8.00 | -0.23 | -0.06 | 0.36 | -0.13 |
| 16.00 | 0.12 | 0.49 | -0.56 | -1.39 |
| 32.00 | 0.23 | 0.22 | 0.33 | -1.14 |
| 64.00 | 0.11 | 0.12 | -0.14 | -0.60 |
| 128.00 | -0.11 | 0.05 | -0.22 | -0.15 |
| 256.00 | -0.58 | 0.03 | 0.04 | 0.42 |
| 512.00 | 0.47 | 0.02 | 0.52 | 0.28 |
| 1024.00 | -0.11 | 0.00 | -0.52 | 0.29 |
| 2048.00 | 0.12 | 0.00 | -0.04 | 0.73 |

of the cards were compared showing both relative values as well as how performance per data size grows for each card for growing numbers of computations in the GPU kernel. Tesla V100 outperforms the other cards significantly reaching its top performance for higher relative computation count. It is interesting to see that the current desktop series GTX 1060 outperforms visibly a few years old professional series Tesla K20m. The mobile GPU cannot match the other cards in performance but its performance grows for larger relative computation counts compared to Tesla K20m and GTX 1060.

In the future, the author plans to extend the set of experiments in terms of the following: testing various kernel codes on various devices and how these impact the ratio of computations to communication, testing particular kernels operating on data types of various precision as this impacts the ratio of computations to communication and finally focusing on performance/real power consumption for the tested GPUs.

## ACKNOWLEDGMENTS

Tests partially performed on Tesla V100 within DGX station purchased and installed at the Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, Poland.

Work was supported partially by the Polish Ministry of Science and Higher Education.

## REFERENCES

[1] C. Woolley, "Gpu optimization fundamentals," February 2013, nVIDIA Developer Technology Group, https://www.olcf.ornl.gov/wp-content/uploads/2013/02/GPU_Opt_Fund-CW1.pdf.

[2] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, 1st ed. Addison-Wesley Professional, 2010. ISBN 0131387685, 9780131387683

[3] P. Czarnul, *Parallel Programming for Modern High Performance Computing Systems*. CRC Press, 2018, ISBN 9781138305953.

[4] J. Luitjens, "Cuda streams. best practices and common pitfalls," 2014, nVIDIA, http://on-demand.gputechconf.com/gtc/2014/presentations/S4158-cuda-streams-best-practices-common-pitfalls.pdf.

[5] Y. Ukidave, A. K. Ziabari, P. Mistry, G. Schirner, and D. Kaeli, "Analyzing power efficiency of optimization techniques and algorithm design methods for applications on heterogeneous platforms," *The International Journal of High Performance Computing Applications*, vol. 28, no. 3, pp. 319–334, 2014. doi: 10.1177/1094342014526907. [Online]. Available: https://doi.org/10.1177/1094342014526907

[6] P. Czarnul, "Parallelization of large vector similarity computations in a hybrid cpu+gpu environment," *The Journal of Supercomputing*,

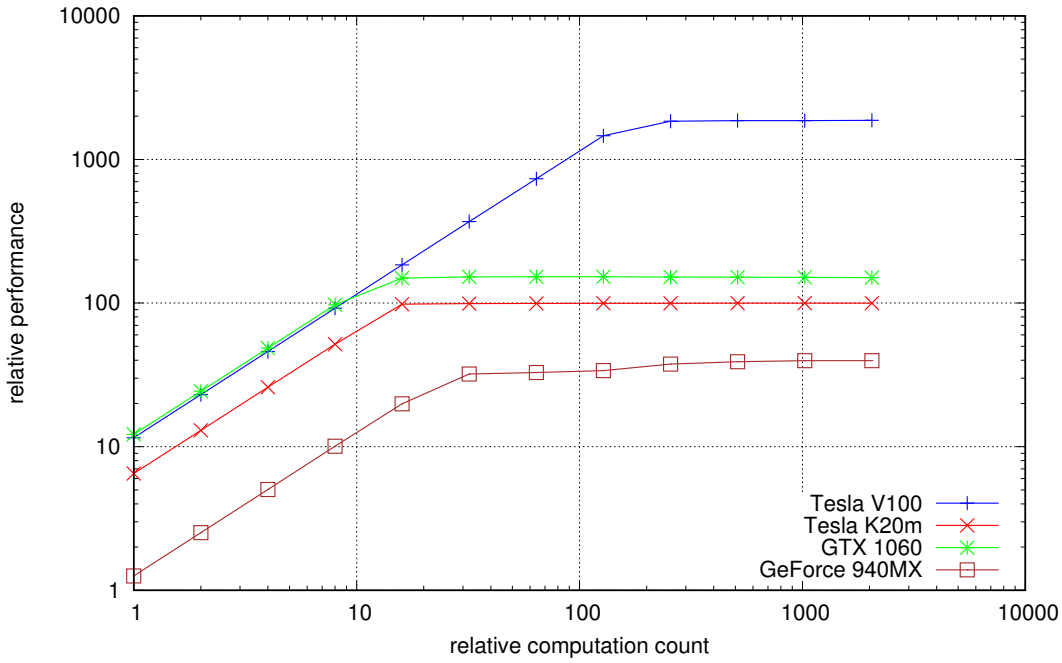


Figure 10. Relative performance vs relative computation count

vol. 74, no. 2, pp. 768–786, Feb 2018. doi: 10.1007/s11227-017-2159-7. [Online]. Available: https://doi.org/10.1007/s11227-017-2159-7

[7] P. Rościszewski, P. Czarnul, R. Lewandowski, and M. Schally-Kacprzak, "Kernelhive: a new workflow-based framework for multilevel high performance computing using clusters and workstations with cpus and gpus," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 9, pp. 2586–2607. doi: 10.1002/cpe.3719. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3719

[8] P. Czarnul, J. Kuchta, M. Matuszek, J. Proficz, P. Rościszewski, M. Wójcik, and J. Szymański, "Merpsys: An environment for simulation of parallel application execution on large scale hpc systems," *Simulation Modelling Practice and Theory*, vol. 77, pp. 124 – 140, 2017. doi: https://doi.org/10.1016/j.simpat.2017.05.009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X17300916

[9] J. Gómez-Luna, J. M. González-Linares, J. I. Benavides, and N. Guil, "Performance models for asynchronous data transfers on consumer graphics processing units," *Journal of Parallel and Distributed Computing*, vol. 72, no. 9, pp. 1117 – 1126, 2012. doi: https://doi.org/10.1016/j.jpdc.2011.07.011 Accelerators for High-Performance Computing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731511001468

[10] N. P. Deshmukh, H. J. Kang, S. D. Billings, R. H. Taylor, G. D. Hager, and E. M. Boctor, "Elastography using multi-stream gpu: an application to online tracked ultrasound elastography, in-vivo and the da vinci surgical system," *PloS one*, vol. 9, no. 12, p. e115881, 2014. doi: 10.1371/journal.pone.0115881. [Online]. Available: http://europepmc.org/articles/PMC4277422

[11] H. Li, D. Yu, A. Kumar, and Y. C. Tu, "Performance modeling in cuda streams – a means for high-throughput data processing," in *2014 IEEE International Conference on Big Data (Big Data)*, Oct 2014. doi: 10.1109/BigData.2014.7004245 pp. 301–310.

[12] M. Sourouri, T. Gillberg, S. B. Baden, and X. Cai, "Effective multi-gpu communication using multiple cuda streams and threads," in *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2014. doi: 10.1109/PADSW.2014.7097919. ISSN 1521-9097 pp. 981–986.

[13] H. Khaleghzadeh, Z. Zhong, R. Reddy, and A. Lastovetsky, "Out-of-core implementation for accelerator kernels on heterogeneous clouds," *The Journal of Supercomputing*, vol. 74, no. 2, pp. 551–568, Feb 2018. doi: 10.1007/s11227-017-2141-4. [Online]. Available: https://doi.org/10.1007/s11227-017-2141-4

[14] K. Osawa, A. Sekiya, H. Naganuma, and R. Yokota, "Accelerating matrix multiplication in deep learning by using low-rank approximation," in *2017 International Conference on High Performance Computing Simulation (HPCS)*, July 2017. doi: 10.1109/HPCS.2017.37 pp. 186–192.

[15] V. Yegnanarayanan, "An application of matrix multiplication," *Resonance*, vol. 18, no. 4, pp. 368–377, Apr 2013. doi: 10.1007/s12045-013-0052-0. [Online]. Available: https://doi.org/10.1007/s12045-013-0052-0

[16] W. Liu and B. Vinter, "A framework for general sparse matrix-matrix multiplication on gpus and heterogeneous processors," *CoRR*, vol. abs/1504.05022, 2015. [Online]. Available: http://arxiv.org/abs/1504.05022