Reproducibility Report for ACM SIGMOD 2020 Paper: "Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects"

WOLFGANG LEHNER, ALEXANDER KRAUSE, ANNETT UNGETHÜM, Technische Universität Dresden, Germany

We ran the experimental suite of the authors on their original hardware. Some teamwork and issue resolving was necessary, but nevertheless, we could reproduce all but two experiments of the submitted paper. The two outliers deviate from the original, yet do not invalidate the key claims of the authors.

1 INTRODUCTION

We were invited to review the paper titled "Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects" by Clemens Lutz[1], Sebastian Breß[2], Steffen Zeuch[1] , Tilmann Rabl[3] and Volker Markl[1] [1] regarding reproducibility. With the helpful support of the authors we were able to reproduce most results of the original paper, some with slight but tolerable variations.

2 SUBMISSION

The submission was supplemented by a reproducibility guide. This document contained information about the necessary hard- and software, as well as a seemingly step-by-step guide (see Section 4.1 for clarification) for the overall reproducibility process and contact information of two persons, who could support in case help was needed. The required code was hosted on the authors' webspace and easily obtainable. Table 1 gives an overview about the relevant provided files.

Table 1. Important Provided Files (selection)

File	Purpose
`prepareSoftware.py`	Install necessary software locally and remotely.
`checkEnvironment.py`	Assert existence of relevant software on your local and remote machines.
`runExperiments.py`	Execute either a single or all predefined experiments.
`buildPaper.py`	Generate the whole or parts of the paper from the obtained experimental results.

3 HARDWARE AND SOFTWARE ENVIRONMENT

The authors provided several scripts to automatically execute the required experiments "locally" and "remotely". However, the scripts were only designed to run on the actual hardware of the authors ("remotely") and required a login for those servers. We tried to setup the experimental environment on our comparably newer hard- and software platform, yet running the experiments on machines different from the original servers was not possible with the given software infrastructure. Modified scripts to facilitate the experiments in our environment were not provided upon request. We therefore proceeded by using the original three servers of the authors, which allowed us to perform the experiments on the very same hardware and software environment as the original paper.

[1]DFKI GmbH, Berlin, Germany
[2]TU Berlin, Berlin, Germany
[3]HPI, University of Potsdam, Potsdam, Germany

4 REPRODUCIBILITY EVALUATION

4.1 Process

We identified the infrastructural limitations early in the process and contacted the authors in order to discuss necessary steps to be taken. For gaining access to the three servers we had to apply for a temporary account of the authors' university. In addition, a separate user was created on the machines. The authors took care of installing/setting up all the necessary software on those servers. Connecting to the servers was realized through the university's ssh gate server. There, the access is provided on the basis of a ticket that is only valid for one day. Application has to be done through a command line prompt once again every day. Unfortunately, the expiration problem was only discovered days later, when the `runExperiments.py` had been killed through connection loss, which in turn led to a delay in the result generation process. The scripts generated csv result files for their respective experimental run. However, once the experiment failed, we could only rerun the whole experiment instead of only the failed configurations, which invalidated all older result files and voided previous time investments. Despite using command line arguments for a longer ticket duration, the ssh session expired randomly before the expected expiration date, thus we could still not use the `runAll` switch to execute all experiments simultaneously.

The authors estimated an accumulated runtime of 2 days. Due to the necessary ssh ticket renewal, we could not run all experiments automatically. Furthermore, some experiments took much longer than expected, with the strongest deviation being 26 hours and 30 minutes as opposed to estimated 2 hours and 30 minutes. Since the `runExperiments.py` does not support running a user defined selection of experiments but either a single one or all together, we ran all experiments manually and individually in a serial manner.

Throughout the whole process, the authors were very responsive and eager to help debug and resolve the various issues with their scripts or crashing experiments. Every occurring problem was resolved very fast.

4.2 Results

The provided scripts allow to generally reproduce the overall outcome of the paper. However, there are some differences which are hard to explain from an external point of view without the possibility to inspect hardware internals.

Detailed results.

(1) **Introduction (Fig. 1)** Difference between measured memory and NVLink bandwidth is larger than in the original paper. This is not problematic at all, it is rather helpful to underline the statement of the paper.
(2) **Analysis of a fast interconnect (Fig. 3)** Differences to the original paper are mostly insignificant, with one exception: In Fig. 2(b), the measured random memory access bandwidth on the Xeon is increased by more than 18% when trying to reproduce the results. This changes the ranking of NVLink and CPU memory random access. The results are not completely inexplicable, because the distribution of the data (e.g. how many pages have to be touched) influences the efficient bandwidth of random memory access.
(3) **Experiments: Join Throughput w/ different transfer methods (Fig. 12)** Unified migration and unified prefetch show an even lower throughput in our reproduction. Again, this is rather profitable for the overall story. The remaining differences are insignificant.
(4) **Experiments: Join Throughput w/ base data on different processors (Fig. 13)** An almost perfect match.
(5) **Experiments: Join Throughput w/ hash table on different processors (Fig. 14)** An almost perfect match.

(6) **Experiments: Scaling of data size for TPC-H query 6 (Fig. 15)** In the original paper, the branching variant on the CPU shows a higher throughput than branching on the GPU with NVLink. This is reversed in our reproduction. Additionally, for scale factor 1.000 our throughput for branching on the GPU with NVLink decreases significantly. Since we only have the resulting csv files, we have no explanation for this behavior.
(7) **Experiments: Scaling the probe-side relation (Fig. 16)** The ranking of the lower performing variants differs from the results mentioned in the paper. However, the difference was small to begin with. Thus, this behavior is not problematic. Additionally, the difference between the two higher performing configurations is larger than in the original paper. Again, we also see a decrease for an NVLink variant for the largest size, which is not present in the original paper. This is not problematic, because the observation "Overall, we are able to process data volumes larger than the GPU's memory capacity at a faster rate than the CPU" still holds true.
(8) **Experiments: Scaling the build-size relation (Fig. 17)** CPU throughput is lower for small sizes in the original paper. The remaining trend for larger sizes stays the same as in the original paper. Not critical.
(9) **Experiments: Join throughput for Zipf distribution of probe relation (Fig. 19)** Insignificant deviations from the original paper, just a slightly lower performance in our reproduction.
(10) **Experiments: Different build-to-probe ratios on NVLink (Fig. 18)** Insignificant differences.
(11) **Experiments: Different join selectivities (Fig. 20)** An almost perfect match.
(12) **Experiments: Throughput of workloads and time per join phase (Fig. 21)** CPU performance of workload B is way higher in our reproduction (3.6x). GPU still outperforms the CPU, but CPU is now outperforming Het.

5 SUMMARY

With the exception of (6) and (12) from Section 4.2, all experiments have either almost the same results as our measurements or are in a tolerable spectrum. In conclusion, our reproducibility experiments can support the main claims of the original paper.

REFERENCES

[1] Clemens Lutz, Sebastian Breß, Steffen Zeuch, Tilmann Rabl, Volker Markl: Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects. SIGMOD Conference 2020: 1633-1649