Critique of "MemXCT: memory-centric X-ray CT reconstruction with massive parallelization" by SCC Team from ETH Zürich

Jan Kleine, Rahul Steiger, Simon Wachter, Emir İşman, Simon Jacob, Dario Romaniello
Department of Computer Science, ETH Zürich, Switzerland
{*jkleine, rasteiger, siwachte, eisman, jacobsi, darioro*}*@ethz.ch*

Abstract—This report analyzes the reproducibility of the paper "MemXCT: Memory-Centric X-ray CT Reconstruction with Massive Parallelization" by Hidayetoğlu et al. in the cloud as part of the SC20 Virtual Student Cluster Competition (VSCC). To reproduce the results from the original work, the ETH Zürich SC20 VSCC team performed a series of CT reconstructions and performed a scaling study using three provided sinograms. All experimental runs were performed during the SC20 VSCC on an HPC cluster hosted in the Microsoft Azure CycleCloud. In this paper, we describe discrepancies in results as a factor of the differences in experiment environments and insufficient parameter tuning. We successfully reproduce the single device performance and partially reproduce the strong scaling behavior. Digital artifacts from these experiments are available at: 10.5281/zenodo.5598108.

Index Terms—Reproducible computation, Student Cluster Competition, MemXCT.

1 Introduction

This report analyzes the reproducibility of the paper "MemXCT: Memory-Centric X-ray CT Reconstruction with Massive Parallelization" by Mert Hidayetoğlu et al. [1].

X-Ray computed tomography (XCT) is a nondestructive 3D imaging technique for observing the internal structure of samples and materials. The MemXCT application reconstructs the internal morphology of materials (as visualized in Fig. 1) from a series of X-ray images using a novel, memory centric approach that reduces redundant computations at the expense of additional memory complexity. The resulting implementation achieves a speedup of between $6.9\times$ and $49\times$, compared to the compute-centric reconstruction library Trace [2].

As emphasized by Hoefler and Belli [3] it is important for the proper exchange of knowledge in the scientific community to present reproducible and interpretable results. This review of best practices in reproducibility provided the framework for deploying and reproducing the work demonstrated in the MemXCT paper.

The computations for the analysis were performed as part of the reproducibility challenge during the Virtual Student Cluster Competition (VSCC) at the SC20 conference. The Student Cluster Competition is an annual competition hosted during the SC conference, where teams of six undergraduate students compete by building their own HPC cluster and optimizing various HPC workloads on it. The VSCC was hosted in the Microsoft Azure CycleCloud, where teams could provision HPC clusters in the cloud using Azure's vast assortment of virtual machines (VMs). Each team had a budget of $3,700, which could be used in a 72 hour time frame. During this time the teams had to produce outputs and results for this study and various other HPC applications and benchmarks, which are unrelated to this work. These constraints affected the design of the experiment and hardware used for the analysis.

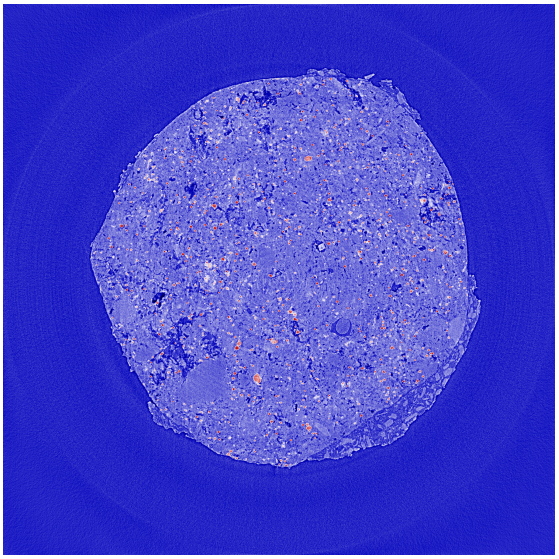


Fig. 1: Reconstruction of Data set CDS3.

In this report we reproduce the single device performance on CPUs and GPUs, as well as the strong scaling behaviour presented in the MemXCT paper. In Section 2 we describe the hardware and software setup of the system used, as well as the input data we used to reproduce the results. In Section 2.4 we give an overview of all performed reconstructions, followed by an analysis with respect to the two aforementioned performance and scaling behaviours in Sections 3.1 and 3.2. Section 4 concludes the report.

SKU	#CPUs	Processor	Memory	InfiniBand Network	Accelerator
HC44rs	44	2× Intel Xeon Platinum 8168 (24 core)	352 GiB	EDR (100 Gbit/s)	-
HB120rs_v2	120	2× AMD EPYC 7742 (64 core)	480 GiB	HDR (200 Gbit/s)	-
NC24rs_v2	24	2× Intel Xeon E5-2690 v4 (14 cores)	448 GiB	QDR (40 Gbit/s)	4x NVIDIA P100
NC24rs_v3	24	2× Intel Xeon E5-2690 v4 (14 cores)	448 GiB	QDR (40 Gbit/s)	4x NVIDIA V100

TABLE 1: List of the Microsoft VM SKUs used for the experimental runs.

2 METHODS

This section describes the methods used for this study, starting with the hardware configuration of the cluster, followed by a description of the deployed software stack and the models used to reproduce the results.

2.1 Cluster Configuration

The Azure CycleCloud allows us to build very adaptive systems, as we can choose from a large variety of compute nodes. To reproduce the results of the MemXCT paper, we choose four different VM types (SKUs). This way we were able to perform the experiments on two different CPU types (for the CPU code) and two different GPU types (for the GPU code). The hardware details for the SKUs used are detailed in Table 1. Note that the number of processor cores do not match the actual number of cores on the listed processors. This is due to the fact that some cores are reserved for Azures hypervisor that manage the VMs.

The operating system on the compute nodes was CentOS 7.7, specifically the generation 2 OpenLogic CentOS 7.7 HPC image. Workload managing and node provisioning was handled by Slurm and the autoscale integration provided by Microsoft.

2.2 Software Configuration

The MemXCT applications were compiled with GCC 9.2.0 and the following software dependencies:

- MVAPICH2 3.2.1 (shipped with aforementioned OS images)
- Cuda 11.1 (V11.1.74)

On the CPU side the compilation flags `-O3` and `-march=<arch>` were used, with `<arch>` being the appropriate CPU architecture. The GPU code additionally had the appropriate `-gencode arch=compute_XX,code=sm_XX` flags added to ensure the code was generated directly for the correct type of GPU.

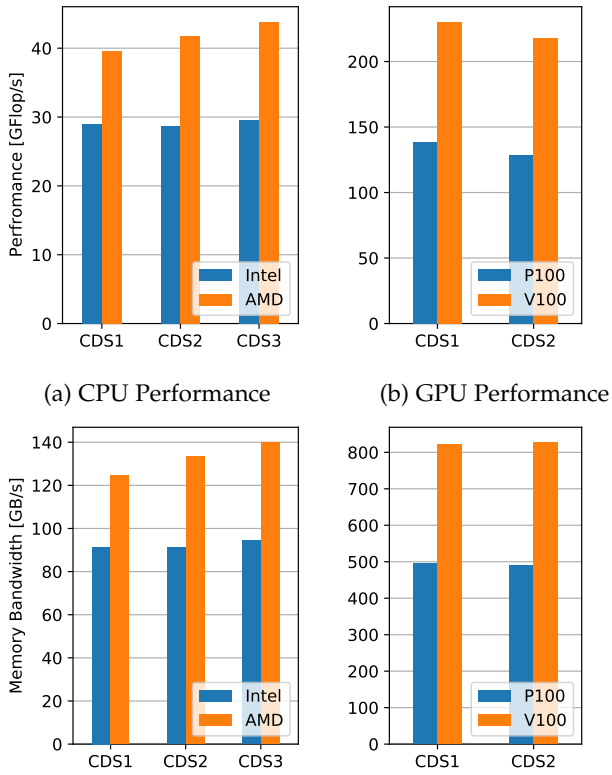
We used the public MemXCT CPU [4] and GPU [5] codes. A software bug in the GPU version of the code, discovered during the competition, prevented the code from utilizing more than one GPU per node while running distributed across multiple nodes. To allow the program to properly run distributed on multiple GPUs per node, we needed to modify line 93 of the `kernels.cu` file. The original implementation miscalculated device IDs. In our case, fixing this meant replacing the line by "`int device = myid % 4;`" as our GPU nodes feature 4 accelerators each.

2.3 Input Data

The data sets used for the reproducibility analysis were three CT Sonograms. The corresponding dimensions are listed in Table 2. The data sets were provided by the competition committee at the time of the competition.

Data set	θ	ρ
CDS1	750	512
CDS2	375	1024
CDS3	1501	2048

TABLE 2: Data sets used for the reproducibility challange with number of angles (θ), number of channels (ρ)



(a) CPU Performance (b) GPU Performance

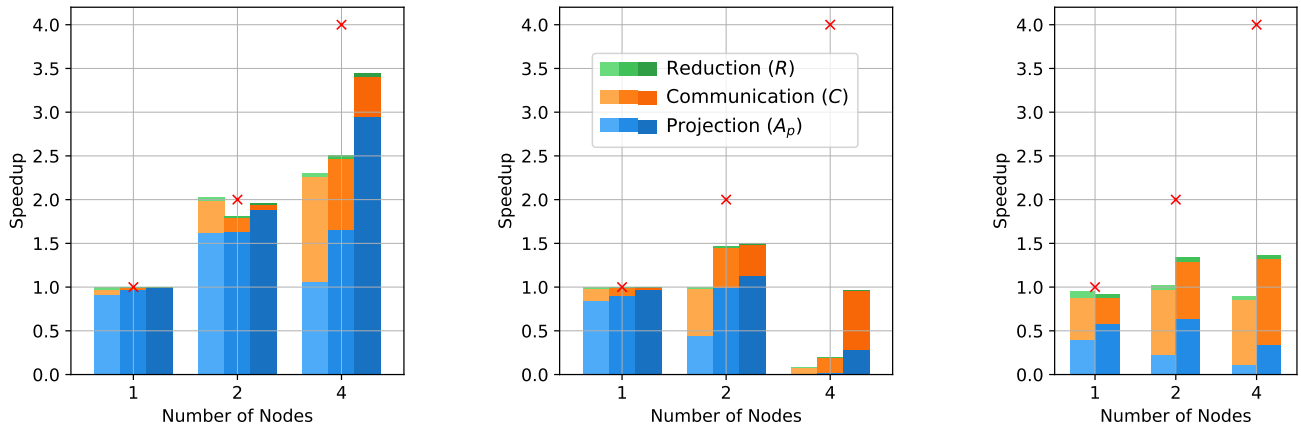
(c) CPU Memory Bandwidth (d) GPU Memory Bandwidth

Fig. 2: Performance and Bandwidth of single device/socket reconstructions.

2.4 Description of Experimental Runs

For the reproducibility analysis we performed a total of 19 runs.

10 runs were performed for the single device comparison. On the CPU-only SKU we ran the reconstruction of models CDS1, CDS2, and CDS3 on both the Intel and AMD



(a) Intel Xeon Platinum 8168 (b) AMD EPYC 7742 (c) NVIDIA V100

Fig. 3: Strong scaling results of inputs CDS1, CDS2, and CDS3 (light to dark) across one, two, and four nodes. The timings include forward and backprojection and are divided to show the relative time spent on projection (A_p), communication (C) and reduction (R). The ideal speedup for each case is marked in red.

nodes, using only one of the two installed CPU sockets (i.e. 22 cores on the Intel based HC SKU and 60 cores on the AMD based HBv2 SKU). This setup was decided on by the competition committee to measure the actual single device, i.e., single CPU performance. On the NVIDIA based GPU SKUs (namely NCv2 and NCv3) a single MPI rank was launched, which utilizes a single GPU. Due to the larger memory footprint of CDS3, it did not fit in device memory and only CDS1 & CDS2 were reconstructed on GPUs. The resulting performance and bandwidth numbers are visualized in Fig. 2 and discussed in Section 3.1.

For the strong scaling part of the reproducibility analysis we performed three sets of runs on the HC, HBv2, and NCv3 SKUs. Each set consisted of the reconstructions of the inputs on one, two, and four nodes. Due to the time and budget constraints of the competition we did not reconstruct input CDS3 in the NVIDIA V100 nodes. The kernel- (A_p), communication- (C), and reduction (R) runtimes of the forward and back projection were recorded. The resulting timings are shown in Fig. 3 and discussed in Section 3.2.

3 Results

In this section we discuss the results of the single device performance study (Section 3.1) and the strong scaling study (Section 3.2).

3.1 Single CPU & GPU Performance

As shown in Fig. 2a the Intel Xeon Platinum 8168 processor achieved a floating point performance between 28.6 and 29 GFlop/s. While these numbers are lower than the 80-100 Gflop/s displayed in Figure 9(a) of the MemXCT paper, they are plausible when looking at the system difference. The KNL cards used in the paper have 64 cores, approximately three times as many cores as a single socket on our Intel system. Though, because the KNL cores are simpler than normal CPU cores, we don't think this accounts for the performance difference. The more important difference are the 16 GB MCDRAM featured on the KNL cards, a $400\,\mathrm{GB/s}$ high speed, on-chip memory. Especially on the smaller models that consume less than 16 GB, this memory greatly benefits performance. In the paper this can be seen with the larger models. When the memory footprint exceeds 16 GB the MemXCT paper reports a much more comparable 20-40 GFlop/s. These differences most likely account for the $3.1\times$ performance difference.

The AMD CPU performs between 39.5 to 43.8 GFlop/s. This is lower than expected as we have close to the same number of cores as a KNL card and more memory bandwidth than on the Intel CPUs, but in some cases only achieve half of the performance presented in the paper. Comparing this to our previous results we would have expected close to twice the performance of out Intel system, based on memory bandwidth. We attribute this to insufficient parameter tuning on our part. This becomes more apparent when looking at the memory bandwidth utilization. In the case of the Intel Xeon 8168 CPU we achieved up to $94.4\,\mathrm{GB/s}$, which is very close to the theoretical peak bandwidth of $95.5\,\mathrm{GB/s}$[1]. However, on the AMD node we achieved $139.9\,\mathrm{GB/s}$, which is only 80% of the theoretical limit of $175\,\mathrm{GB/s}$, further suggesting that our runtime parameters were not optimal.

Noteworthy is also the fact that our CPU performance stayed rather constant across different sized inputs, which is not the case with the results shown in the paper. Hidayetoğlu et al. report a drop in performance that is caused by the switch from high bandwidth MCDRAM to lower bandwidth system memory. In our case the model data always resided in system memory, hence we never see such a decrease in performance.

The GPUs performed as expected with 129.0 - 138.9 GFlop/s and 217.9 - 230.3 GFlop/s on a single P100 and V100 respectively (see Fig. 2b). This very closely matches the results presented in Figure 9(c) of the MemXCT paper and is unsurprising as we use the same GPU models used in the paper and the majority of the computation is performed on the GPU. Since the data is loaded onto the GPU at

1. Microsoft lists a maximum memory bandwidth of $191\,\mathrm{GB/s}$ for the entire node, hence a single socket is limited by $95.5\,\mathrm{GB/s}$

the beginning and resides there for the remainder of the computation, the host configuration does not have a large impact on single device performance and we expected very similar results.

3.2 Strong Scaling on GPUs and CPUs

The strong scaling results on the Intel Xeon Platinum 8168 (Fig. 3a) platform follow a similar pattern as seen in the MemXCT paper Figure 11(c), which shows the strong scaling behaviour on the KNL based Theta super computer. When going from one to four nodes the MemXCT application achieves a speedup of nearly $3.5\times$ on the CDS3 input in our setup. The paper reports a similar speedup of approximately $3.3\times$ on the Theta super computer when quadrupling the node count from 128 to 512. Further, like in the paper, we observed that the time spent on reduction only accounts for a very small fraction of the overall computation time and stays rather constant across all runs. We also observe that the larger models scale better, which is expected behaviour and explained in the paper.

While the speedups match, we observed that the relative time spend on communication grows more rapidly than what is reported in the paper. Figure 11(c) from the paper shows a communication increase of approximately $2\times$ across the entire scaling study. In our case the time spend on communication increase by as much as $5.7\times$ when going from one to four nodes.

This is easily explained by the fact, that our network environment and, more importantly, our network usage is different. While it is hard to compare the environments (as we don't have details on the network topology employed in Microsoft's cloud infrastructure), we had one important difference regarding the usage and scale. In the paper the smallest run is performed on 128 nodes, meaning a significant amount of network communication is already occurring needed in the first run. We, on the other hand, started out scaling study with a single node. Hence, our first run does not contain any network communication, while the second and third run do. We suspect the transition from intra-node to inter-node communication to be the main reason for the difference in communication time behaviour.

Our scaling results on the V100 GPUs shown in Fig. 3c show a seemingly bad scaling behaviour. Though, these numbers are difficult to compare to the other scaling studies and what is presented in the paper. We note that from the beginning a substantial part of the overall computation time is spend on communication. This is due to the fact the V100 GPUs are substantially more powerful in terms of GFlop/s ($7.5\times$, as seen in Section 3.1 and Fig. 2) than the CPUs we used. Since the models we used are relatively small the time spend on the compute heavy projection is very short. Increasing the number of GPUs does reduce the time spend on projection, but at the same time more communication is needed, negating most of the gained performance. Based on this, we expect larger models (such as CDS3 or larger) to scale a lot better. This behaviour is also described in the paper and we can see the same in the Intel results discussed above.

We attribute the bad scaling behaviour on the AMD nodes to the insufficient parameter tuning already discussed in Section 3.1. Based on the hardware we expect a properly tuned MemXCT application to achieve similar or better performance than what we found in our Intel based study.

4 Conclusion

Overall, we were able to reproduce the performance behavior of the MemXCT application, especially in the deployment to single devices. The single GPU results are a very close match to the numbers presented in the MemXCT paper, while the single socket performance is comparable as soon as the KNL cards used in the paper cannot rely on high-bandwidth, on-chip memory.

There are some discrepancies in our observed strong scaling behaviour when compared to the results presented in the paper. We can attribute this difference in strong scaling to a vastly different experimental setup, which was not optimized for strengths of MemXCT strong scaling as part of the competition design. Taking these differences into account, we still recognize similarities as discussed in Section 3.2 and showed that the application scales well given sufficient input size.

Acknowledgments

We would like to thank our advisors Prof. Torsten Hoefler from ETH Zürich, Hussein Harake from CSCS, and Timo Schneider from ETH Zürich for their constant support together with the Swiss National Supercomputing Centre (CSCS) and the Scalable Parallel Computing Laboratory (SPCL) from ETH Zürich. Thanks also to our sponsors Dalco, Gigabyte, NVIDIA, and the Hasler foundation.

References

[1] M. Hidayetoğlu, T. Biçer, S. G. de Gonzalo, B. Ren, D. Gürsoy, R. Kettimuthu, I. T. Foster, and W.-m. W. Hwu, "Memxct: Memory-centric x-ray ct reconstruction with massive parallelization," ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356220

[2] T. Bicer, D. Gürsoy, V. D. Andrade, R. Kettimuthu, W. Scullin, F. D. Carlo, and I. T. Foster, "Trace: a high-throughput tomographic reconstruction engine for large-scale datasets," *Advanced Structural and Chemical Imaging*, vol. 3, no. 1, jan 2017.

[3] T. Hoefler and R. Belli, "Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results," in *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 2015, pp. 1–12.

[4] M. Hidayetoğlu, "Memxct cpu code," Sep 2020. [Online]. Available: https://github.com/merthidayetoglu/MemXCT-CPU

[5] ——, "Memxct gpu code," Sep 2020. [Online]. Available: https://github.com/merthidayetoglu/MemXCT-GPU