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

Griffin Dube, Cavender Holt, John Hollowell, Sarah Placke, Sansriti Ranjan, Nikolas Heitzig, Jon Calhoun

Abstract—This paper reproduces the results of the Supercomputing 2019 paper, *MemXCT: Memory-Centric X-ray CT Reconstruction with Massive Parallelization* by Hidayetoğlu et al. as part of the Supercomputing 2020 Student Cluster Competition Reproducibility Challenge. We reproduce the single CPU-GPU performance experiments and the strong scaling experiments and compare the results to the original paper. Although we are not able to use the exact HPC system from the original paper, we use similar hardware and configurations to recreate the original environment in Microsoft Azure Cloud services. We were not able to demonstrate exact performance characteristics from the original paper due to hardware limitations in our environment, but we are able to reproduce similar performance trends for both single-node and scaling experiments.

Index Terms—Reproducible computation, Student Cluster Competition, Replication, High-performance Computing, X-ray CT, Microsoft Azure

1 Introduction

EACH year for the Reproducibility Challenge at the Student Cluster Competition (SCC), teams reproduce results from a paper accepted to the previous year's Supercomputing (SC) Technical Program [1]. For the 2020 reproducibility challenge, teams are tasked to reproduce results from Hidayetoğlu et al. and their SC'19 paper, *MemXCT: Memory-Centric X-ray CT Reconstruction with Massive Parallelization* [2] as part of the SC'20 Student Cluster Competition Reproducibility Challenge. Their paper presents MemXCT, a system that uses a memory-centric approach to iterative reconstruction of X-ray computed tomography while avoiding redundant computation. MemXCT also optimizes parallel performance of the x-ray CT reconstruction process by using a pseudo-Hilbert ordering for more effective domain decomposition, and multi-stage buffering of inputs in order to improve caching performance by partitioning input into multiple buffers such that they each fit within L1 cache. The MemXCT paper demonstrates parallelism using Intel KNL 7230 processors as well as GPUs for intra-process parallelism, and implements MPI for parallelism across nodes. These optimizations are evaluated across multiple accelerators, including Intel KNL as well as three generations of NIVIDA GPUs.

In this critique, we:

- reproduce optimized single node GPU performance of MemXCT;
- perform strong scaling tests on two real world datasets; and
- compare performance of MemXCT on Microsoft Azure to the results found in the original paper.

Our critique is structured as follows: Section 2 outlines the software and hardware configuration we use to obtain our results. Section 3 covers the scripts we use to reproduce results. In Sections 4 and 5, we present our results for single node and scaling tests, respectively. In Section 6, we present visualization results, and lastly, we conclude in Section 7.

Manuscript received March 16, 2021; revised XXX XX, 20XX.

2 Experimental Setup

Due to the COVID-19 pandemic, the SC'20 Student Cluster Competition is hosted in the cloud on Microsoft Azure. Thus, all experiments are run using Microsoft Azure cloud instances, deployed using the CycleCloud tool for managing high-performance resources on Azure [3]. One restriction informing our hardware selection during the competition is the budget of $3,700 which normalizes the competition such that one team does not have an unfair advantage. This budget is divided across multiple challenges throughout the competition, so we are able to dedicate only a portion of it to running these experiments. As the budget is divided across running multiple challenges for the competition, our ability to conduct extensive scaling experiments is limited due to the cost of adding nodes to a virtual cluster. Additionally, none of the node architectures in Azure perfectly match the hardware resources available in the original paper.

2.1 Hardware

For scaling tests, we use a virtual cluster, while for the single GPU performance tests we use single a Azure virtual machine (VM) to simplify setup. We list the VMs we use as well as their the respective hardware specifications in Table 1.

In the hardware configuration in Table 1, each node contains 12 CPU cores and 2 GPUs. For scaling tests, we choose to use the NC12 VM in order to provide the most similar GPU performance to Blue Waters [5] which is used as a test system in the MemXCT paper. Blue Waters is a heterogeneous Cray supercomputer with over 22,000 nodes

	NC12	NC12_v2	NC12_v3
CPU	Intel Xeon E5-2690 v3	Intel Xeon E5-2960 v4	
CPU Mem.	112 GiB DDR4	224 GiB DDR4	
GPU	K80	P100	V100
GPU Mem.	24 GiB	32 GiB	32 GiB
Interconnect	Ethernet	Ethernet	Ethernet

TABLE 1
VM Hardware Specifications [4].

housed at the University of Illinois at Urbana-Champaign. Each node contains two AMD 6276 Interlagos CPUs or one AMD 5276 CPU and one NVIDA K20x GPU. The NC12's K80 GPUs are closest in performance and architecture to Blue Waters' K20x that are available on Azure. Some variation in performance is attributed to performance differences between Blue Waters' AMD 6276 Interlagos processors and our previously described hardware configuration. Hardware differences between those used in the original paper have potential to result in slightly better scaling properties due to differences in the GPU generations; however, this does not affect the single CPU-GPU performance trends.

2.2 Software

We use Azure CycleCloud's default CentOS 7 image as our operating system. In order to compile MemXCT, we employ gcc 9.2.0, OpenMPI 4.0.3, and CUDA 11.1, and we submit jobs using PBSPro. In reproduction of results, we use two provided competition datasets – similar to the Shale Sample RDS1 [2] – as our testing input.

We replicate, as closely as possible, the experiments in Section 4 of the MemXCT paper. Because the Intel processors we use on the NC12, NC12_v2, and NC12_v3 instances only support vector registers up to 256 bits [6], we perform vectorization using AVX2, as opposed to AVX-512 which is used in the MemXCT paper. Further, memory bandwidth measurements are recorded for GPU nodes as opposed to KNL nodes due to the lack of KNL VMs on Azure.

3 Description of Experimental Runs

We plan our experimental runs such that an average of the performance and timing results is taken, producing an average of our results to account for any variability across multiple runs in the system. We perform five runs for each of our experiments and for each input dataset. We choose this experimental design in order to provide a high-quality result that is substantial enough to provide a critique of the MemXCT paper without excessive use of Azure cloud resources.

We compile and run MemXCT using a variety of test scripts described below:

- **Initialization** - We use this set of scripts to simplify setup of all software including MemXCT on our Azure cloud instances. We first install a version of azcopy, a utility used for downloading the datasets used for testing. Next, we install CUDA 11.1 and OpenMPI 4.0.3. Finally we clone the MemXCT repository, download the test datasets using azcopy, and compile MemXCT.
- **Azure Run Script** - We set up the environment variables required by MemXCT, including the domain information, tile size, block size, buffer size, input filenames, and the number of OpenMP threads to use. After this environment setup is complete, we execute the application using `mpirun`. There is a unique script for each input dataset due to differences in domain.
- **Performance and Scaling** - In these scripts, we first call the initialization script to perform all setup. We then run the application five times using the *Azure Run Script* in order to generate our results.

Initialization files are located in our digital artifact under the `/compile` directory while the Azure Run scripts and Performance and scaling scripts are in the `/run/scripts` directory. The use of these scripts ensures that our tests remain consistent across different VM types to provide an accurate performance analysis.

4 Single CPU-GPU Performance Comparison

We measure single node GPU performance on Azure's NC12, NC12_v2, and NC12_v3 VMs we describe in Table 1. We use the K80, P100, and V100 GPUs available on these VMs to reproduce the GPU performance optimization and memory bandwidth utilization graphs from Figure 9 in the MemXCT paper [2]. Figure 1 and Figure 2 show these results.

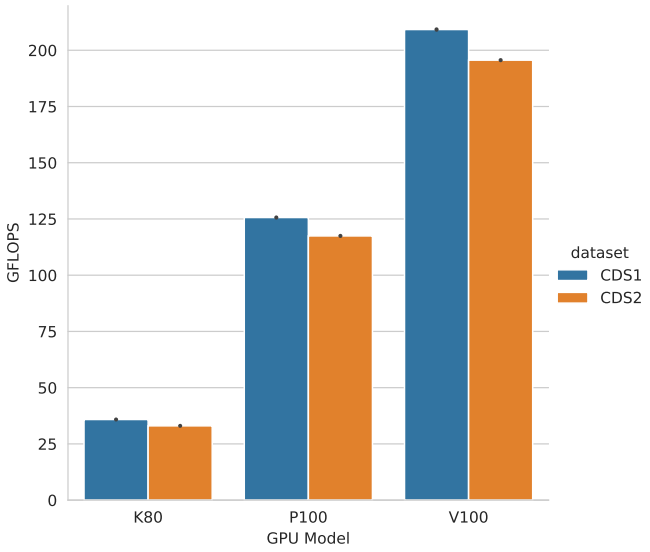

Fig. 1. Reproduced GPU Performance in GFLOPS

One main difference in our performance tests compared to those run in the MemXCT paper is the use of two real-world datasets specific to the Student Cluster Competition. We refer to these datasets as CDS1 and CDS2. Our use of these competition datasets as opposed to artificial datasets used in the original paper is due to requirements set forth by the Student Cluster Competition. CDS1 and CDS2 are of size 750×512 and 375×1024 respectively, similar to the size of ADS2 described in Table 3 of the original paper.

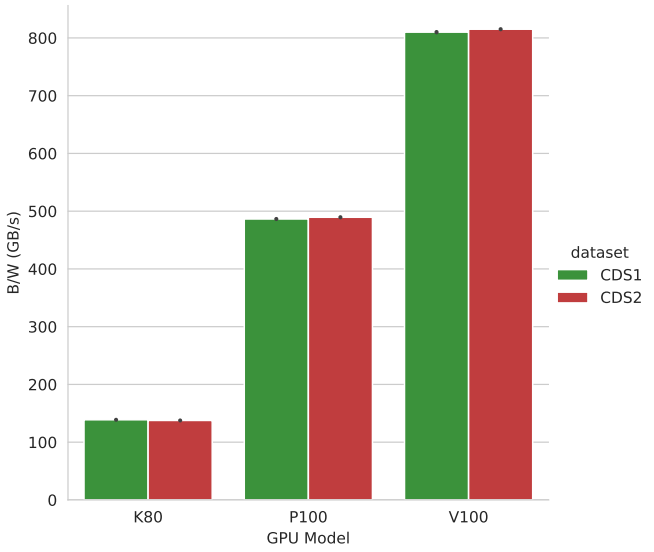

Fig. 2. Reproduced GPU Memory Bandwidth Utilization in GB/s

The GPU performance in Figure 1 corresponds to the fully optimized version of MemXCT, including pseudo-Hilbert ordering and multi-stage input buffering. Our results demonstrate similar performance to those presented in Figure 9 (d-f) of the original paper. Our results reproduce the expected GPU performance with only slight variation in GFLOPS performance. For K80 GPUs, we produce results within 5% of those presented in the MemXCT paper. For both P100 and V100 GPUs, our results are within 15% and 23% respectively of the original results. These slight variations are consistent with differences due to change in input datasets displayed Figure 9 (d-f) of the original paper.

Results from our tests in Figure 2 of memory bandwidth utilization differ from the results described in Figure 9(c) of the MemXCT paper because of limitations in the available hardware on Microsoft Azure. While the original results describe application memory bandwidth utilization on KNL processors, we employ the same NC-series GPU VMs we use for measuring GFLOPS performance. For K80 GPU nodes, memory bandwidth utilization performs poorest, while the memory bandwidth utilization on P100 and V100 VMs outperforms the KNL performance results from the original paper. These results, while not directly comparable to the results of the original paper, provide insight to the memory bandwidth utilization of MemXCT on various generations of GPU. The memory bandwidth value is calculated internally to the MemXCT application by determining the size and number of each projection and back-projection buffers used, then dividing it by the sum of the projection and back-projection kernels. We find that using V100 GPU acceleration as opposed to KNL processors results in 2.5× improvement in memory bandwidth utilization while maintaining similar GFLOPS performance. In both KNL and GPU nodes, we see that trends in memory bandwidth utilization correspond to those in performance. From examining our single-node performance results combined with results from Figure 9 of the MemXCT paper, we conclude performance is dependent on memory bandwidth utilization.

5 Strong Scaling on GPUs and CPUs

We perform strong scaling experiments on a much smaller set of nodes when compared with those presented in Figure 11 of the MemXCT paper, which scales from 1-4096 nodes. We perform strong scaling experiments using between one and four NC12 VMs in Azure. We display the results of these runs in Figures 3 and 4. This is largely due to a lack of compute resources and funds available to us on Azure during the Student Cluster Competition; however, the trends shown are comparable to those investigated in the paper.

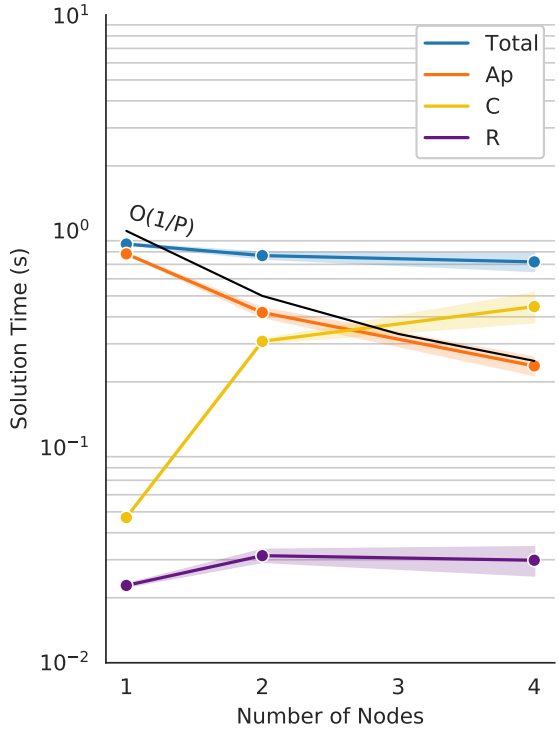

Fig. 3. Strong Scaling, with CDS1 on Microsoft Azure

Our strong scaling results are shown for an average of five runs for each of the competition datasets, CDS1 and CDS2. These results show the Total, Partial Forward Projection (A_p), Communication (C) and Reconstruction (R) times for each dataset. These operations, described in Section 3.4.3 of the MemXCT paper, make up the computation of the Forward Projection Matrix, A, given by the equation $A = RCA_p$. A_p corresponds to the actual computation of the output while R and C is the overhead we incur because of MPI parallelization of the MemXCT application. We also provide a reference trend representing a complexity of $\mathcal{O}(1/P)$, representing linear strong scaling as number of processors increases.

We produce timing results used in strong scaling tests as output of the application using MPI timing functions. Although the number of nodes we use is not identical to the number of nodes used in scaling tests for Blue Waters in the MemXCT paper, we see similar trends in our scaling

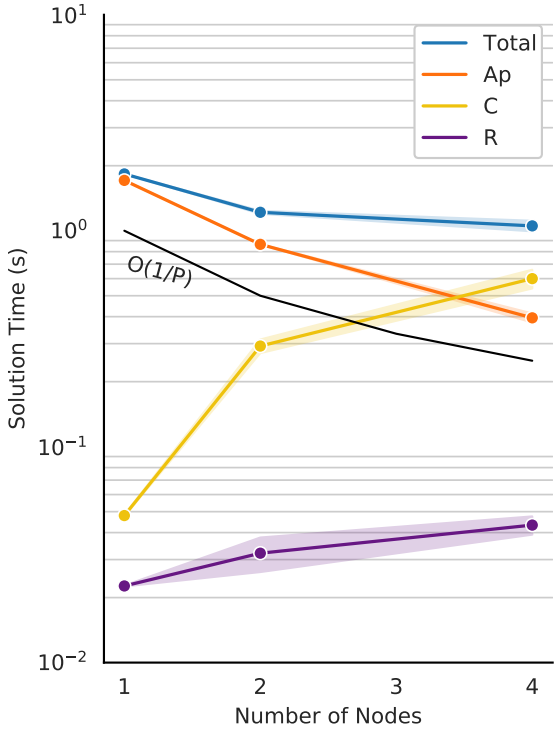

Fig. 4. Strong Scaling, with CDS2 on Microsoft Azure

results. For small numbers of nodes, Partial Forward Projection follows the $\mathcal{O}(1/P)$ reference curve relatively closely. Communication slowly increases as we introduce more nodes, requiring more communication between processes. Reconstruction time remains relatively the same, slightly increasing as we add nodes, and lastly, total time decreases slightly.

Of note is the fact that communication overhead dominates partial forward projection computation for a number of nodes larger than 3 in our tests, while this does not occur until reaching a number of nodes larger than 128 on Blue Waters. It is possible that this disparity is due to the interconnects used for our virtual cluster. As mentioned in Section 2, we use ethernet for our scaling tests as opposed to the Gemini interconnects used by Blue Waters which leads to higher latency communication. Further, Azure's NC12 VMs do not support accelerated networking, meaning that inter-node communication must pass through an additional level of virtual switches before reaching other nodes [7]. Aside from this, our strong scaling results follow similar trends and produce solution times on the same order of magnitude to those presented in the original MemXCT paper.

6 Visualization

Figures 5 and 6 show the reconstructed outputs for CDS1 and CDS2 we generate with the Fiji tool [8]. Reconstruction of these shale samples is done following the steps in the `README.txt` located in the directory `/figures/scripts` of our digital artifact. Visualization of these datasets results in reconstructed images similar to the shale sample presented in Figure 8 of the MemXCT paper as RDS1.

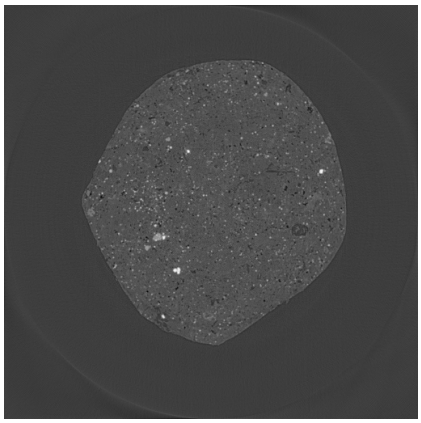

Fig. 5. CDS1 Reconstruction Image (512x512)

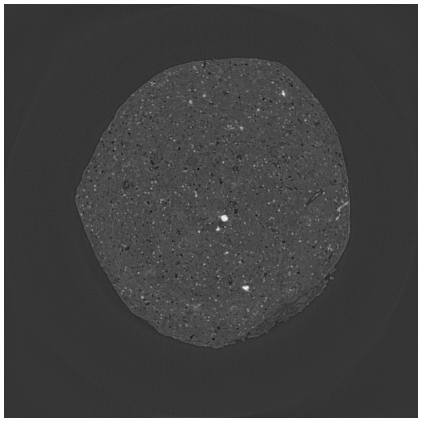

Fig. 6. CDS2 Reconstruction Image(1024x1024)

7 Conclusion

During the Student Cluster Competition, we reproduce single GPU performance, memory bandwidth utilization, and strong scaling results for the MemXCT paper. Although all results were not identical to those produced in the paper, many similar trends exist between the two sets of results. Single GPU performance tests show similar results to the paper in terms of performance. Differences in memory bandwidth utilization due to a lack of similar hardware on Azure allow for an investigation of memory bandwidth utilization on K80, P100 and V100 GPUs.

Our strong scaling results present very similar trends to those seen in the MemXCT paper on a small number of nodes. Increasing this scale would provide similar results to those discussed in the paper.

Reproducing these results confirm replicability of the results presented by the MemXCT paper and show that this application is capable of running on a variety of different hardware with consistent performance.

Acknowledgments

We would also like to acknowledge the support of Clemson Computing & Information Technology and Dell Technologies.

REFERENCES

[1] [Online]. Available: https://studentclustercompetition.us/index.html

[2] 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

[3] Microsoft. (2021) Azure cyclecloud documentation: What is azure cyclecloud? https://docs.microsoft.com/en-us/azure/cyclecloud/?view=cyclecloud-8.

[4] Microsoft. (2020) Linux virtual machines pricing. https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/.

[5] B. Bode, M. Butler, T. Dunning, T. Hoefler, W. Kramer, W. Gropp, and W.-m. Hwu, "The Blue Waters super-system for super-science," in *Contemporary High Performance Computing*, ser. Chapman & Hall/CRC Computational Science. Chapman and Hall/CRC, April 2013, pp. 339–366. [Online]. Available: https://www.taylorfrancis.com/books/e/9781466568358

[6] Intel, "Intel® xeon® processor e5-2690 v3," https://ark.intel.com/content/www/us/en/ark/products/81713/intel-xeon-processor-e5-2690-v3-30m-cache-2-60-ghz.html, 2021.

[7] Microsoft, "Create a linux virtual machine with accelerated networking using azure cli," https://docs.microsoft.com/en-us/azure/virtual-network/create-vm-accelerated-networking-cli, 2021.

[8] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona, "Fiji: an open-source platform for biological-image analysis," *Nature Methods*, vol. 9, no. 7, pp. 676–682, Jul 2012. [Online]. Available: https://doi.org/10.1038/nmeth.2019