Reproducing Performance of Memory-Centric X-ray CT Reconstruction on Azure CycleCloud

Runxin Zhong, Jiajie Chen, Chen Zhang, Mingshu Zhai, Zeyu Song, Yutian Wang, Wentao Han, Lin Gan, Jidong Zhai[1]

Department of Computer Science and Technology, Tsinghua University, Beijing, China

Abstract

Hidayetoğlu et al. propose a novel memory-centric algorithm for X-ray CT image reconstruction. They use SpMV as kernels for reconstruction, and propose two-level pseudo-Hilbert ordering and multi-stage input buffering to improve cache locality and memory bandwidth utilization.

In this paper, we reproduce the results of single-device experiments and strong-scaling tests for both CPU and GPU, and visualize the output image based on three new datasets. We use various cloud instances on Azure CycleCloud and STREAM benchmark to analyze performance and its bottleneck under different scenarios, *i.e.*, Intel CPUs, AMD CPUs, and NVIDIA GPUs. We compile the code with GCC/ICC/NVCC as well as OpenMPI/Intel MPI. Our results show that the MemXCT algorithm is able to obtain good strong scalability, which is consistent with the conclusion in SC'19 paper.

Keywords: Reproducible computation, Scalability, Cloud, Student Cluster Competition

1. Introduction

X-ray computed tomography (XCT) is widely used to study the internal morphology of materials. But traditional X-ray reconstruction approaches fail to reach the goals of high computational efficiency, high-quality output, and high utilization of large scale resources together. Hidayetoğlu et al. propose MemXCT [1], a novel memory-centric iterative method to solve this problem. In this approach, both forward- and backward-projection in iterative reconstruction are converted into Sparse Matrix-Vector Multiplication (SpMV) operations. Two important optimization techniques are proposed to improve cache locality and memory bandwidth utilization, *i.e.*, applying pseudo-Hilbert ordering [2] for tiled domain and data within each tile; and buffering data for irregular access with appropriate multi-stage buffer size to improve L1 cache hit rate. With these optimizations applied on SpMV, MemXCT achieves great utilization for different levels of hierarchical memory and is able to reconstruct a large CT tomogram in only 10 seconds using 4096 Intel KNL nodes.

In this paper, we first reproduce the performance results, **single device performance** and **strong scaling performance** in the MemXCT paper[1] under different architectures. A standard benchmark called STREAM [3] is used to measure the memory bandwidth for CPU. Furthermore, we perform **visualization** to verify our results.

The remaining sections are organized as follows. Section 2 describes experimental setup configuration including hardware, software, etc. Section 3 introduces compilation, execution, and strategies for our experiments. We present our results and analysis from Section 4 to Section 6. The visualization of using new datasets and the final conclusion are provided in Section 7 and Section 8, respectively.

2. Experimental Setup

2.1. Hardware and Software Configurations

Azure SKUs, hardware and software configuration used in our experiments can be found in Table 1 and Table 2.

[1]Team 05, Tsinghua University

[2]HB120rs_v2 is only used to generate tomogram of CDS3 dataset for visualization. It has 2 AMD EPYC 7V12 processors per node and 60 cores per socket.

Azure SKU	Cores / Sockets	Accel.	Net.
NC24r_Promo	24 / 2	4×K80	IB
NC24rs_v2	24 / 2	4×P100	IB
NC24rs_v3	24 / 2	4×V100	IB
F16s_v2	16 / 1	N/A	Eth.
HB60rs	60 / 2	N/A	IB
HB120rs_v2 [2]	120 / 2	N/A	IB

Table 1: Hardware Configuration on Azure CycleCloud

OS	CycleCloud Ubuntu 18.04 202010140
Compiler	ICC 19.1.2.254 and GCC 7.5.0
MPI	Intel MPI 2019.6 Build 20200624 OpenMPI 3.0.1
NVIDIA Driver	455.32.00
CUDA	10.0.130

Table 2: Software Configuration

2.2. Datasets

Three new datasets CDS1 to CDS3 provided in the beginning of the competition are used. The details of these datasets are listed in Table 3, along with the artificial datasets used in the original paper for comparison.

Name	Sinogram	Tomogram
ADS1	360×256	256×256
ADS2	750×512	512×512
ADS3	1500×1024	1024×1024
ADS4	2400×2048	2048×2048
CDS1	750×512	512×512
CDS2	375×1024	1024×1024
CDS3	1501×2048	2048×2048

Table 3: Datasets in Evaluation

2.3. Modifications

Besides the basic requirements of the Reproducibility Challenge, we enhance our experiments with two features as follows.

Various CPUs. We select Intel Xeon Platinum 8168 CPU for lack of Intel KNL 7230 used in the MemXCT paper. Also, AMD EPYC 7551 is available in Azure CycleCloud. Therefore, we conduct our experiment on both Intel and AMD CPUs to analyze and compare their performance difference.

STREAM Benchmark. Due to the nature of virtualization, CPU metrics are different from the ones taken from the official sites. Therefore, we use STREAM [3] Benchmark to measure the actual memory performance of underlying hardware.

3. Description of Experimental Run

Compilation. The source code of MemXCT and libraries are stored in a NFS server and managed by Spack. We carefully follow the instructions from the paper and write scripts to easily compile the code.

Vectorization. We use Intel C++ Compiler to generate the vectorization report. After analyzing the report, we find that only the inner most loop of the key kernel and many inner loops in the 'main.cpp' file are vectorized.

Run. We write shell scripts to automatically run different experiments. The number of OpenMP threads is set to be the same of the core number per CPU, for the performance experiment on a single CPU. MPI process number is set to be the number of used devices (CPU socket or GPU) for all experiments.

Experiment Strategies. The main experiments consist of four parts: Single CPU and Single GPU Performance, as well as Strong Scaling on CPUs and GPUs. For CPU experiments, we collect the results from both Intel CPUs and AMD CPUs. For GPU, we apply single device experiments on three models, namely NVIDIA K80, P100, and V100, and evaluate the strong scaling on P100 as a trade-off between efficiency and cost. We first finished GPU experiments because of the stable and easily-tuned performance according to the results in MemXCT paper. On the contrary, due to the inaccessibility to Intel KNL CPU, we have to tune the parameters in CPU experiments manually. Therefore, we do these experiments later.

Comparison with MemXCT paper. Our experiments mostly follow the design of MemXCT paper, except that we use Intel Skylake and AMD CPUs instead of Intel KNL CPU for CPU experiments.

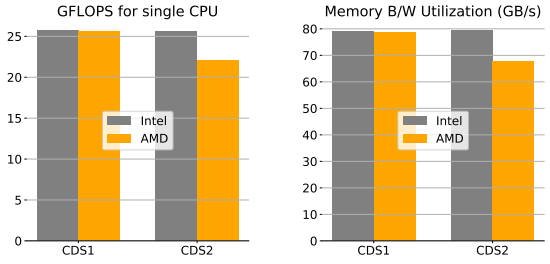
4. Single CPU / GPU Performance

In these two experiments, we compare our performance to the original paper based on the benchmark results and hardware official data.

4.1. CPU Performance

	Xeon Platinum 8168 (F16s_v2 Intel)	EPYC 7551 (HB60rs AMD)
Copy	80.14 GB/s	108.77 GB/s
Scale	80.82 GB/s	77.42 GB/s
Add	84.26 GB/s	84.85 GB/s
Triad	84.06 GB/s	84.45 GB/s
VM Cores	16	30
Cores	24	32
Mem B/W	128.0 GB/s	170.6 GB/s

Table 4: STREAM result (Copy, Scale, Add, Triad) and other details for Intel and AMD CPU. VM cores means the number of CPU cores exposed to virtual machine. Actual core number and memory B/W come from Intel[4] and AMD[5] websites.



(a) GFLOPS (b) Memory B/W

Figure 1: GFLOPS and Memory B/W for single Intel and AMD CPU

Results of STREAM benchmark and other details are shown in Table 4. For AMD CPU, because these basic operations all appear in MemXCT kernels, the memory bandwidth should be limited to around 80 GB/s despite large bandwidth for Copy. In the figure, the core number and memory bandwidth of each CPU socket are lower than the actual ones due to virtualization on cloud.

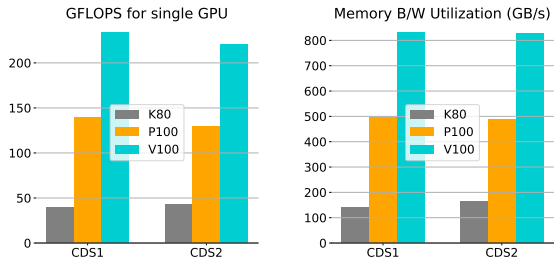
The single device results in CDS1 and CDS2 datasets are given in Figure 1. For CDS1, Intel and AMD CPU achieve similar results on both GFLOPS and Memory Bandwidth. However, the performance of CDS1 is far less than ADS2 in the Figure 9(a) of MemXCT paper. The reasons are listed as follows.

First, the performance is bounded by memory bandwidth. In Figure 1b, the memory bandwidth of CDS1 and CDS2 reaches the maximum memory bandwidth in Table 4. Therefore, we conclude that it is Memory Bandwidth that limits the performance of a single CPU for both Intel and AMD.

On the other hand, the Theta system used by MemXCT paper has 16GB MCDRAM providing a large memory bandwidth of 400 GB/s. As analyzed in the paper, when dataset is small enough to fit into MCDRAM, the performance is very high but it drops to about 40 and 20 GFLOPS when datasets are large. The performance numbers of ADS3 and ADS4 of the paper are similar to our CDS1 and CDS2 numbers, leading to the same conclusion of bottleneck in memory bandwidth.

For CDS2, there is a drop in performance and memory bandwidth utilization of AMD experiment. The reason is that AMD CPU has 15 NUMA nodes and the Intel CPU has only one. When memory footprint increases, cross NUMA memory access latency can become a new bottleneck.

4.2. GPU Performance



(a) GFLOPS (b) Memory B/W

Figure 2: GFLOPS and Memory B/W for single GPU

The performance results for single GPU are presented in Figure 2. We can see a similar performance pattern for GFLOPS compared to Figure 9(d)(e)(f)

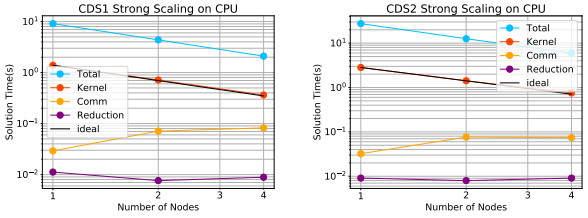
GPU	Mem B/W	Theoretical Mem B/W	Util.
K80	162 GB/s	480 GB/s	33.75%
P100	498 GB/s	732 GB/s	68.03%
V100	829 GB/s	900 GB/s	92.11%

Table 5: GPU Memory Bandwidth Utilization. Actual memory bandwidth is the highest value in Figure 2b. Theoretical memory bandwidth comes from NVIDIA websites[6][7][8].

in MemXCT paper because GPUs are used exclusively in the cloud.

We make comparison between bandwidth utilization and official data, as shown in Table 5. We can see that the memory bandwidth utilization increases with newer GPU. It is reasonable because newer GPU has much higher computing ability. For old GPUs such as K80, computation is the bottleneck, while memory bandwidth utilization is higher and is becoming a new bottleneck for more powerful V100.

5. Strong Scaling on GPUs and CPUs



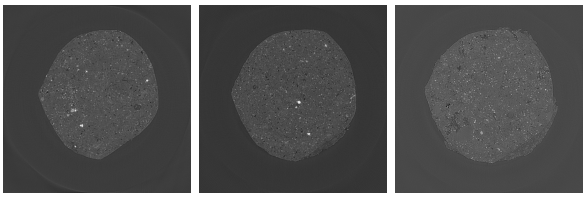
(a) CDS1 on CPU (b) CDS2 on CPU

(c) CDS1 on GPU (P100) (d) CDS2 on GPU (P100)

Figure 3: Results of Strong Scaling experiments

We test the strong scaling on CPU (Intel Xeon Platinum 8168) and GPU (P100) nodes respectively. We use one, two, and four nodes to run experiments instead of a large number of nodes in the MemXCT paper. The results are given in Figure 3. We can see the performance is similar for both CDS1 and CDS2, and is also consistent with the result in MemXCT paper provided that CDS1 has the same size as ADS2. The kernel exhibits linear speedup while the reduction time and communication grow with the node number. It proves the effectiveness of the optimizations in MemXCT including partitioning and utilization for memory hierarchy, even if the results come from different architectures.

6. Visualization



(a) CDS1 (b) CDS2 (c) CDS3

Figure 4: Reconstructed tomograms for CDS1 to CDS3

We save the reconstructed tomogram for CDS1 to CDS3 [3] from previous experiments and use Fiji to visualize the result as shown in Figure 4. The reconstructed tomograms all look reasonable and clear.

7. Conclusion

In this paper, we reproduce the **Single Device Performance** and **Strong Scaling** experiments of MemXCT on Azure CycleCloud. We use Intel, AMD CPU, and NVIDIA GPUs including K80, P100 and V100 to evaluate MemXCT's performance under different architecture. Due to the virtualization in cloud, STREAM benchmark is applied to help us to identify the bottleneck. We find memory bandwidth is the main bottleneck in our experiments. Besides, The result of strong scaling shows MemXCT has great linear speedup for its fully-optimized kernel, which is similar to the MemXCT paper.

[3] CDS3 is run on 4 HB120rs_v2 nodes due to the requirement for large memory

References

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

[2] J. Zhang, S.-i. Kamata, Y. Ueshige, A pseudo-hilbert scan algorithm for arbitrarily-sized rectangle region, in: International Workshop on Intelligent Computing in Pattern Analysis and Synthesis, Springer, 2006, pp. 290–299.

[3] J. D. McCalpin, Stream: Sustainable memory bandwidth in high performance computers, Tech. rep., University of Virginia, Charlottesville, Virginia, a continually updated technical report. http://www.cs.virginia.edu/stream/ (1991-2007).
URL http://www.cs.virginia.edu/stream/

[4] Intel Xeon Platinum 8168 Specifications.
URL https://ark.intel.com/content/www/us/en/ark/products/120504/intel-xeon-platinum-8168-processor-33m-cache-2-70-ghz.html

[5] AMD EPYC 7551 Specifications.
URL https://www.amd.com/en/products/cpu/amd-epyc-7551

[6] NVIDIA K80 Datasheet.
URL https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-product-literature/TeslaK80-datasheet.pdf

[7] NVIDIA P100 Datasheet.
URL https://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-PCIe-datasheet.pdf

[8] NVIDIA V100 Datasheet.
URL https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf