Fall 2022: ME759 Final Project Proposal

Project Title: Optimizing the LU Factorization Algorithm for CPU-GPU Heterogenous Systems

Link to git repo for project: https://git.doit.wisc.edu/VRAMADAS/repo759.git

Problem statement: LU Factorization is a popular algorithm for solving systems of linear equations in many modern-day applications. Tan et al [1] optimize LU Factorization in LINPACK to introduce pipelining of various stages in the algorithm for customized AMD GPU clusters. Wu et al [2] propose a two-stage column block parallelization instead. They however use LAPACK and BLAS routines in their algorithm. I propose to implement LU Factorization algorithm by distributing the computation across both multiple CPU cores and Nvidia GPUs without using either package. Further, I plan to minimize data movement overheads by pipelining data reads and algorithm execution by scheduling them concurrently. Since parts of this optimization will be in matrix multiplication, column swapping etc., the benefits can be carried over to other algorithms that use these basic operations.

Motivation/Rationale: Implementing LU Factorization and squeezing out some extra performance will be a good introduction to software for heterogenous computing. This will focus not only on the parallel programming models in CPUs and GPUs but also on aspects of scheduling tasks between the two. Additionally, given the rich literature that already exists regarding LU Factorization, I will get exposure to how experts approach the problem and try to incorporate that into my solution. Finally, it will help me get a sense of the larger picture of the work I do in CPU-GPU system architectures at Heterogenous Architectures Lab at UW Madison. My work is largely limited to architectural aspects of GPUs and seeing the benefits of heterogenous computing will put my work in perspective and possible guide future work.

Explain how you contemplate going about it: I plan to use OpenMP to parallelize portions of the algorithm on the CPU. I expect to use loop parallelization to speed up arithmetic instruction execution and the task parallelization to break down memory fetches into smaller modules. This would enable data fetches and algorithmic execution to be scheduled in parallel with each other. Additionally, I plan to offload larger blocks of data on to the GPU to accelerate the algorithm. The execution will gain a good degree of concurrency using scratchpad memory to reduce memory access times and CUDA streams to make data movement. Deciding the size of chunks to be run on CPU vs GPU will be determined like how Huang et al [3] do it. Finally, I will compare the performance of different chunk sizes between CPU-GPU. This comparison will also include pure CPU and pure GPU performances.

ME759 aspects the proposed work draws on:

- OpenMP
    - Loop Parallelization
    - Task Parallelization
- CUDA
    - Scratchpad memory
    - CUDA Streams

Deliverables:

- Project Code
- Input data files, golden output data files
- Report detailing the methodology and results

How you will demonstrate what you accomplished: The results of this study will be a comparison between the performance of various schedulers. The comparison will present the ideal chunk size and degree of concurrency to use when splitting data blocks between CPU and GPU for LU Factorization to achieve best utilization of resources. This will demonstrate the benefits of splitting execution between CPU and GPU and follow on to different algorithms in the future. Transferring the optimizations to new algorithms is straightforward since operations such as matrix multiplication or column swapping are present in LU Factorization and ubiquitous in linear algebra routines. Finally, the performances of various chunk sizes will present a window into a possible architectural exploration of GPU techniques to minimize memory transfer bottlenecks.

Other remarks: Wu et al [2] implement their solution using MPI to communicate between cores. I would like to try a similar approach and then compare with my primary implementation of OpenMP + CUDA if I have time at the end.

References:

1. G. Tan, C. Shui, Y. Wang, X. Yu and Y. Yan, "Optimizing the LINPACK Algorithm for Large-Scale PCIe-Based CPU-GPU Heterogeneous Systems," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 9, pp. 2367-2380, 1 Sept. 2021, doi: 10.1109/TPDS.2021.3067731.
2. R. Wu and X. Xie, "Two-Stage Column Block Parallel LU Factorization Algorithm," in IEEE Access, vol. 8, pp. 2645-2655, 2020, doi: 10.1109/ACCESS.2019.2962355.
3. W. Huang, L. Yu, M. Ye, T. Chen and T. Hu, "A CPU-GPGPU Scheduler Based on Data Transmission Bandwidth of Workload," *2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2012, pp. 610-613, doi: 10.1109/PDCAT.2012.15.