Runtime System for GPU-based Hierarchical LU Factorization

Qianxiang Ma, Tokyo Institute of Technology. ma@rio.gsic.titech.ac.jp
Rio Yokota, Global Scientific Information and Computing Center. rioyokota@gsic.titech.ac.jp

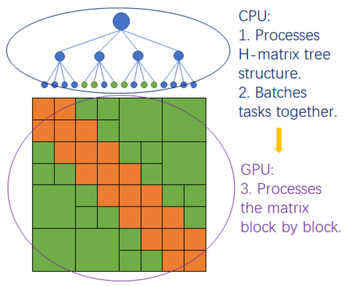
INTRODUCTION

Hierarchical Low-Rank Approximation of Matrices reduces not only the storage requirement from O (n^2) to O (n log n), but also the number of floating point operations (FLOPS) of matrix calculations.

GPUs have considerably more cores inside and greater calculation potentials than CPUs, but fully utilizing such potential is a also challenging task.

DESIGN

When CPU and GPU are collaborating, they have different strong points: CPU can handle branches, tree structures and recursions effectively, but GPU is faster doing repetitive tasks. To best utilize their strong points, we have a highly modularized and pipelined design.



. CPU: Stores the hierarchy. Generate tasks and create the scheduling of the tasks, and pass them to the GPU.

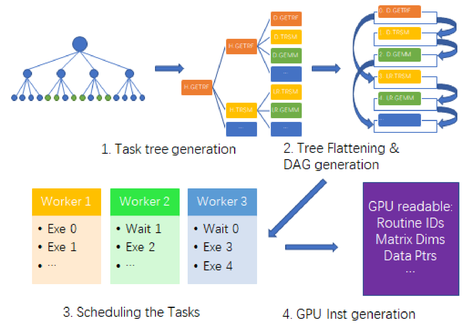
. GPU: Stores the matrix-entries. Use the task and schedule information to process the matrix in a single, highly-batched kernel.

Parallelization techniques also applies.

HOST (CPU)

Different from Matrix multiplications, In-place Hierarchical LU factorization has dependencies amongst the tasks, that fetching premature data leads to incorrect factorizations results and race conditions.

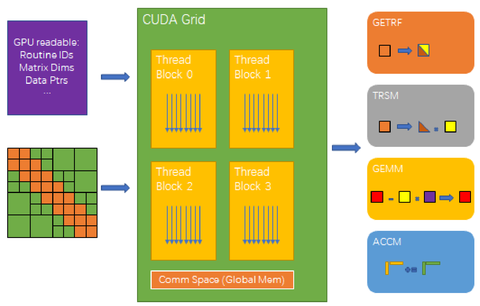
An Illustration of the Host:



DEVICE (GPU)

GPU uses a unified kernel with multiple device functions that implements the BLAS routines. All threads in one thread block execute a device function together.

Kernel Level & Thread Block Level:



Warp level: Vectorized mem I/O & Warp shuffling.

RESULTS

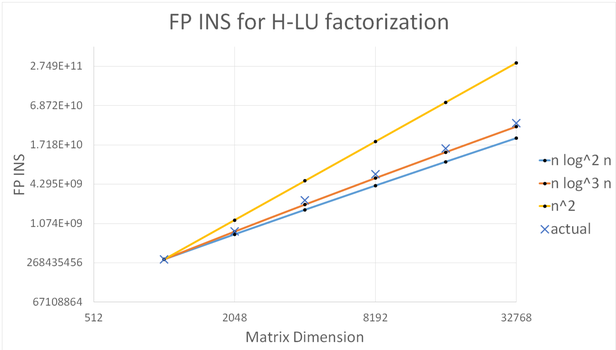


Figure 1: FLOPS for H-LU factorization.

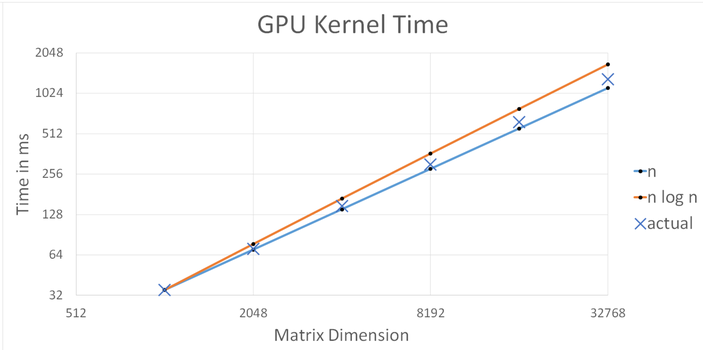


Figure 2: Kernel running time in milliseconds.

. Hierarchical Low-rank LU factorization compresses the FLOPS required very noticeably (0.158% of the dense LU for n = 32768).

. An efficient kernel (28.5 GFLOPS/s on single NVIDIA Tesla V100, n = 32768).

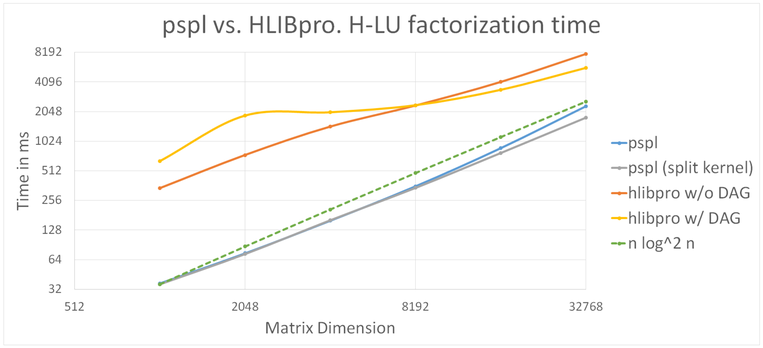


Figure 3: Performance Comparison with HLIBpro.

Comparing with existing H-LU implementations:

HLIBpro (Dr. R. Kriemann 2014, Max-Planck-Institut für).

Configured with nmin = 256, eps = 0, fixed rank 16.

. DAG + Task-based approach.

. Host + Device generally faster than CPU only.

. Controllable runtime overheads through kernel splitting.

CONCLUSION & ACKOWLEDGEMENT

. GPUs are typically considered ineffective handling tree structures and recursions, but with enough preprocessing from the CPU, trees can be transformed into batched tasks.

. Hierarchical Low-rank Approximations of matrices compresses the FLOPS required for matrix calculations very significantly, which is not only LU factorization. We believe that our approach could be developed further to accommodate even more kinds of H-matrix calculations.

. This work was supported by JST CREST Grant Number JPMJCR19F5.

Visit our repository on Github!