EXASCALE COMPUTING PROJECT

Multi-GPU capabilities in SuperLU and STRUMPACK sparse direct solvers

Sherry Li, xsli@lbl.gov

Wajih Boukaram, Lisa Claus, Pieter Ghysels, Yang Liu (LBNL)

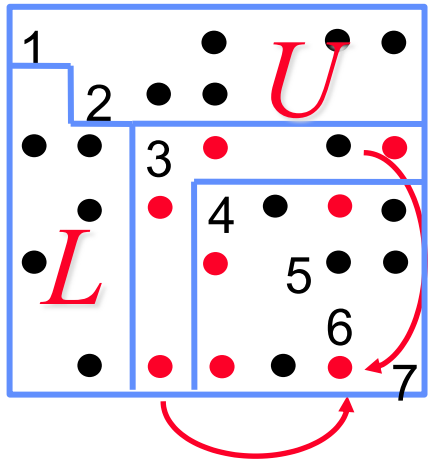
Piyush Sao (ORNL)

ECP Community BOF Days, May 10-12, 2022

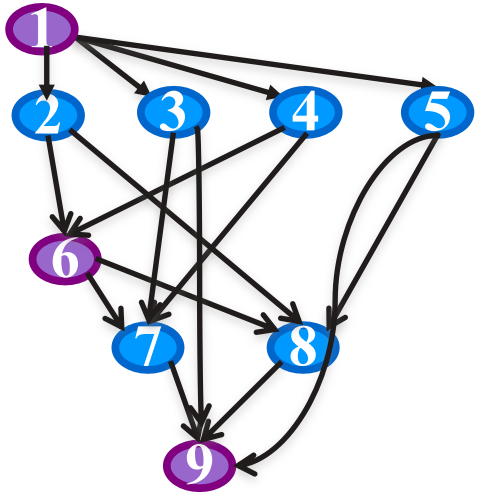
Recent Advances in Selected Sparse Linear Solvers Libraries

Two libraries: SuperLU and STRUMPACK

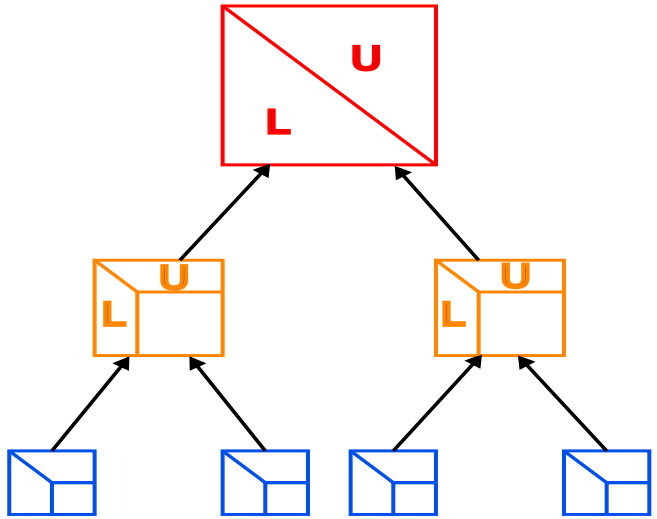
factorization based solvers and preconditioners



SuperLU: supernodal
Designed for nonsymmetric pattern
DAG (directed acyclic graph)



STRUMPACK: multifrontal
Designed for symmetric pattern
Tree



$S^{(j)} \leftarrow ((A^{(j)} - D^{(k1)}) - D^{(k2)}) - \ldots)$

$S^{(j)} \leftarrow A^{(j)} - (..(D^{(k1)} + D^{(k2)}) + \ldots)$

EXASCALE COMPUTING PROJECT

SuperLU_DIST

Communication-avoid 3D algorithm ('pddrive3d')

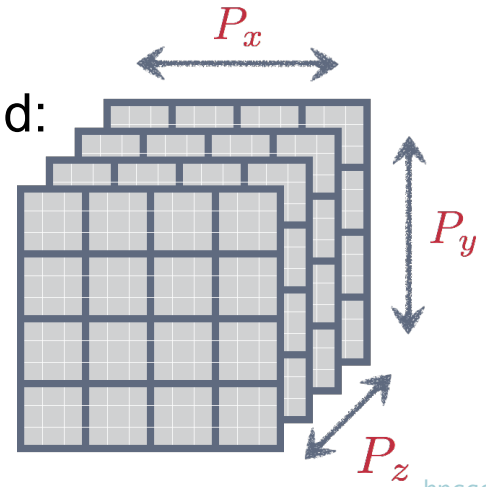
Sao, Li, Vuduc, JPDC 2019

- For matrices from planar graph, provably asymptotic lower communication complexity:
  - Comm. volume reduced by a factor of sqrt(log(n))
  - Latency reduced by a factor of log(n)
- Strong scale to 24,000 cores

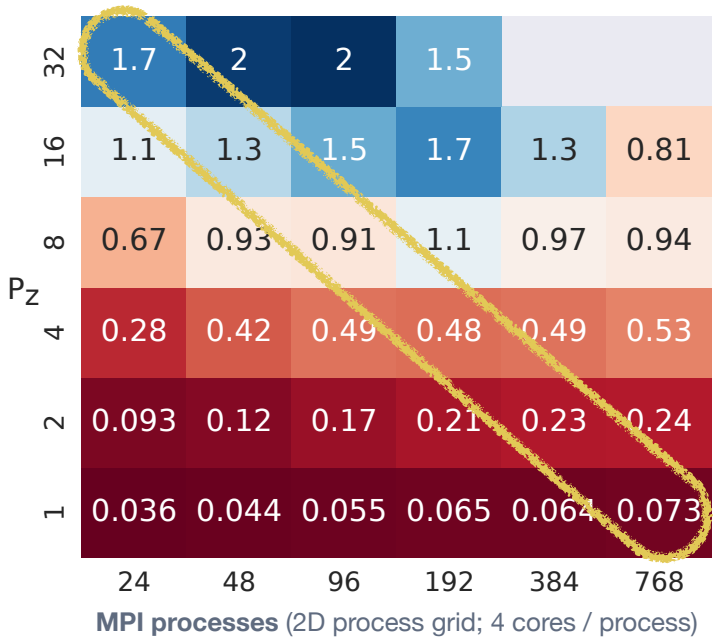
Compared to 2D algorithm:

- Planar graph: up to 27x faster, 30% more memory @ P_z = 16
- Non-planar graph: up to 3.3x faster, 2x more memory @ P_z = 16

3D process grid:
$\{P_{XY}, P_Z\}$



hpcgarage.org/h



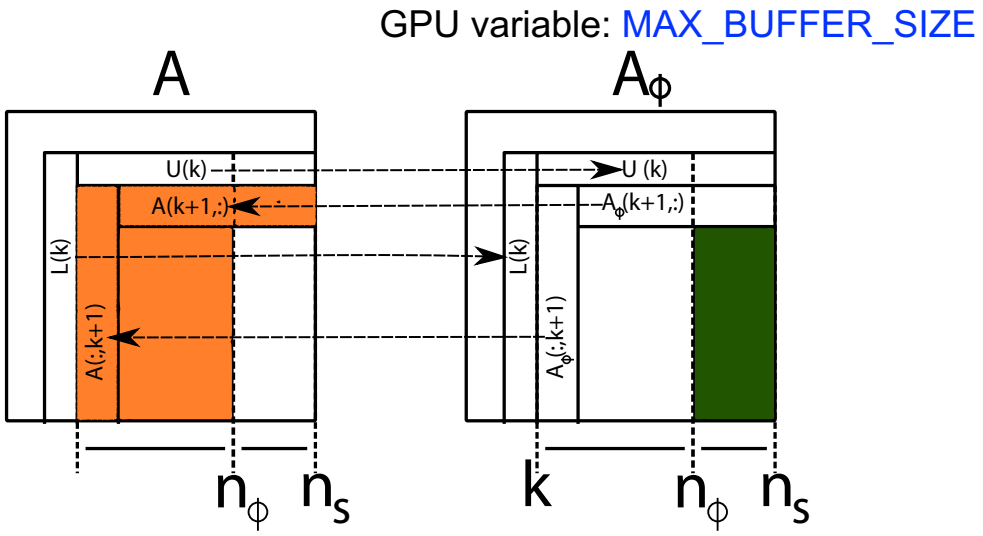
2D to 3D:
→ 23x speedup

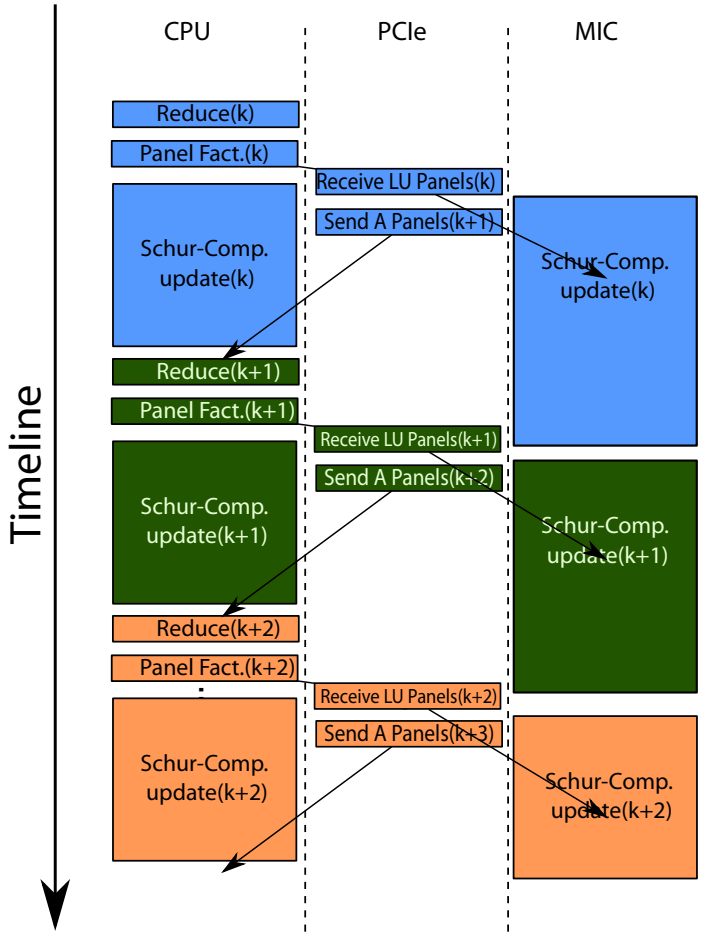
Teraflop/s
(32x procs → 2x speedup)

Offload Schur-complement update to GPU

- Both CPU and GPU perform GEMM and Scatter

  Sao, Liu, Vuduc, Li, IPDPS 2015





Results on NVIDIA and AMD GPUs

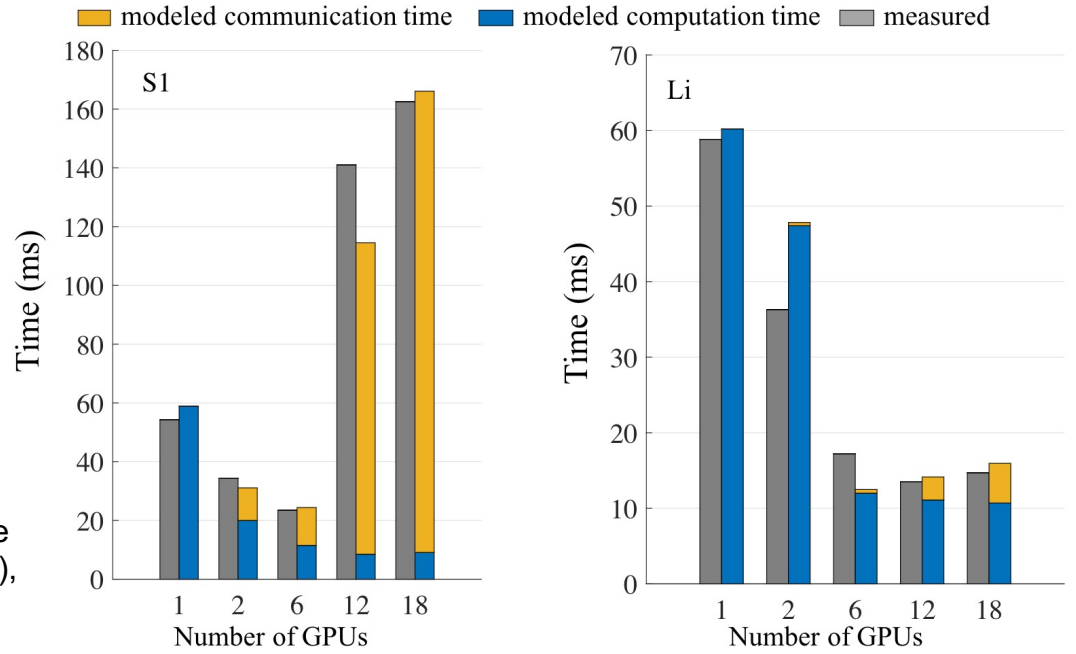
- ”Frank” system at Univ. of Oregon
  - Saturn: Xeon Platinum, NVIDIA A100
  - Instinct Xeon E5, 2 AMD MI100
- 3D algorithm: 1x1x2
  - Offload partial Schur-complement update
  - panel factor still on CPU
- export SUPERLU_ACC_OFFLOAD
  =0 CPU-only
  =1 +GPU

		NVIDIA A100 (up to 5X)	2 AMD MI100 (up to 7X)
torso3	CPU	8.08	16.3
	+GPU	11.2	17.7
Li4244	CPU	83.7	156.7
	+GPU	15.9	30.2
Geo_1438	CPU	496.7	181.6
	+GPU	92.0	25.0

Sparse triangular solve: multi-GPUs

- Created a single-GPU SpTRSV solvers for NVIDIA (CUDA) and AMD (HIP) GPUs
  - Works best if entire L & U can fit on one GPU
- Extended with one-sided GPU libraries (NVSHMEM, ROCSHMEM*)
  - ***Enables scalable, distributed memory, GPU-accelerated solvers***
  - ***With 18 GPUs, up to 6x speedup over Nvidia cusparse_csrsv2()***
  - Performance and scalability are highly dependent on matrix sparsity and inter-node communication performance
- Modeled alternative process mappings for GPUs
  - ***Potential 2x speedup over default 1D block cyclic mapping using 6 GPUs***

**AMD evaluation delayed due to waiting for AMD software updates*



Nan Ding, Yang Liu, Samuel Williams, Xiaoye S. Li, "A Message-Driven, Multi-GPU Parallel Sparse Triangular Solver", SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21), 2021.

SuperLU for multiple types of GPUs

SuperLU

- Programming environments
  - Use macros as unified wrappers to CUDA (NVIDIA) and HIP (AMD)
  - CMake options: TPL_ENABLE_HIPLIB or TPL_ENABLE_CUDALIB

```
1  #ifdef HAVE_CUDA
2  #include "cuda_runtime_api.h"
3  #include "cuda_runtime.h"
4  #include <cublas_v2.h>
5  #define gpuError_t cudaError_t
6  #define gpuSuccess cudaSuccess
7  #define gpuMalloc cudaMalloc
8  #define gpuMemcpyAsync cudaMemcpyAsync
9  #define  threadIdx_x threadIdx.x
10 #define  blockIdx_x blockIdx.x
11 ...
```

```
1  #elif defined(HAVE_HIP)
2  #include "hip/hip_runtime_api.h"
3  #include "hip/hip_runtime.h"
4  #include "hipblas.h"
5  #define gpuError_t hipError_t
6  #define gpuSuccess hipSuccess
7  #define gpuMalloc hipMalloc
8  #define gpuMemcpyAsync hipMemcpyAsync
9  #define  threadIdx_x hipThreadIdx_x
10 #define  blockIdx_x hipBlockIdx_x
11 ...
```

EXASCALE COMPUTING PROJECT

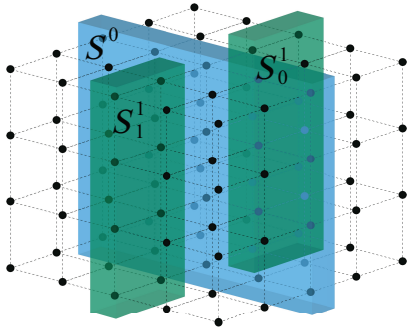
STRUMPACK

STRUMPACK is further enhanced with low-rank (LR) compression

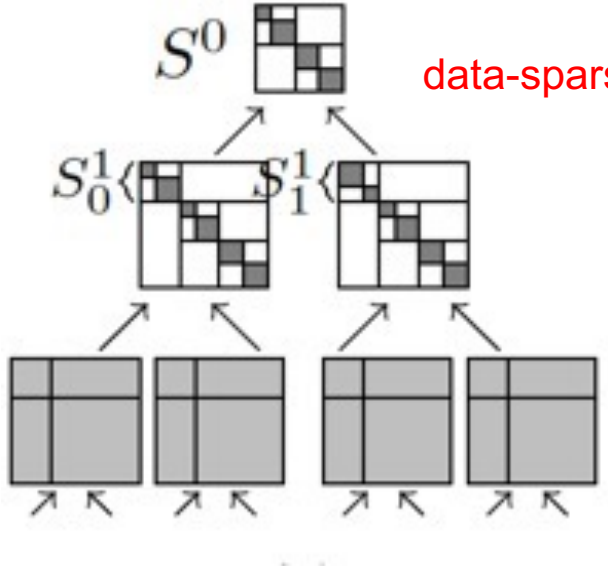
- Globally sparse, locally dense
  - Embed LR data-sparse in sparse multifrontal algorithm
- In addition to structural sparsity, further apply LR data-sparsity to dense frontal matrices
- **Randomized sketching + Nested bases** to achieve linear scaling in sparse case
  - **O(N logN) flops, O(N) memory for 3D elliptic PDEs**

    (as opposed to $O(N^2)$ flops with exact factorization)
- ***Support multiple LR formats:***
  - ***HSS, HODLR, BLR, Butterfly, HODBF***

Nested dissection ordering



Multifrontal
Separator tree

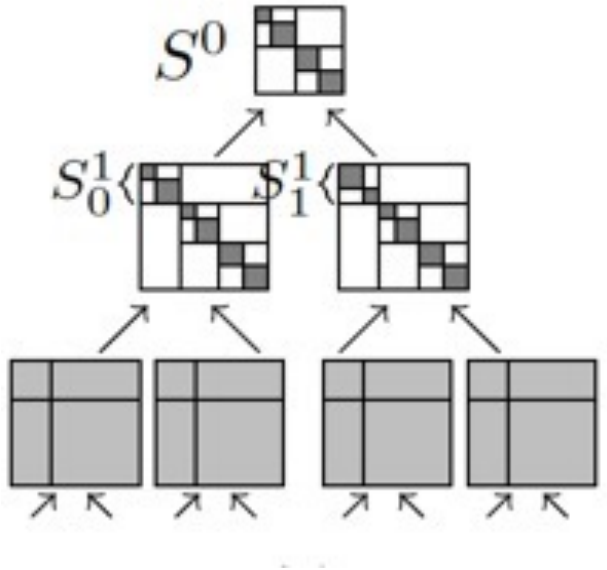


data-sparse

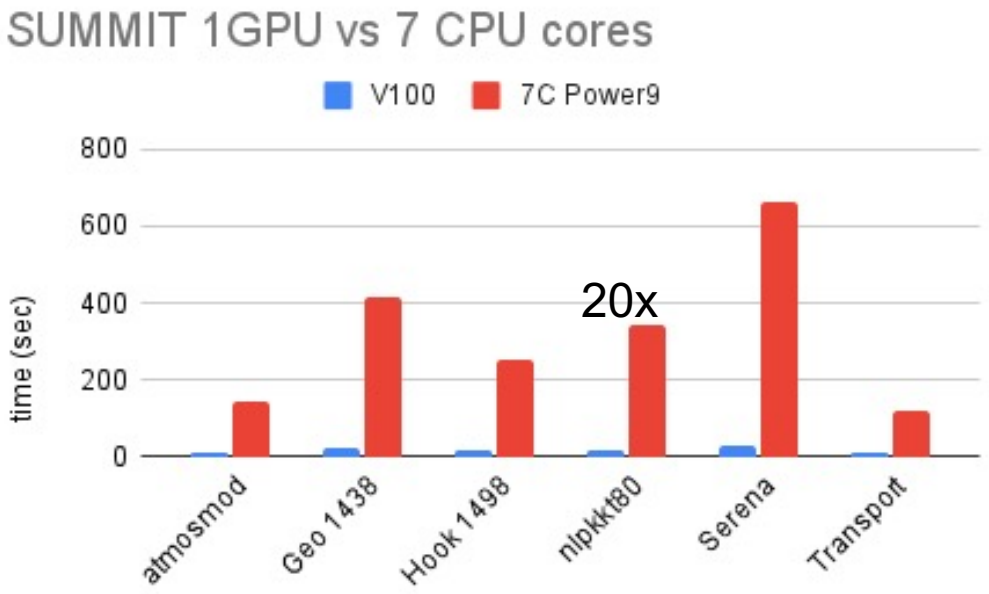
STRUMPACK GPU implementation

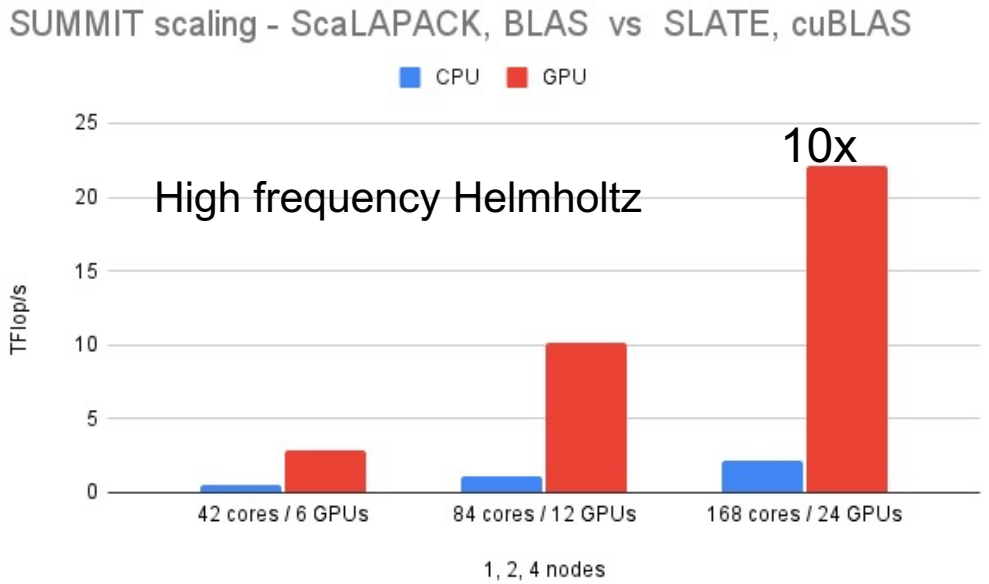
- Level by level traversal of the multifrontal tree
- Batch all nodes on the same level, apply batched dense linear algebra
  - Small matrix variable sized batched dense LU CUDA/HIP kernels
  - cuBLAS/cuSOLVER loop with multiple streams for larger matrices
  - Working with MAGMA to add variable sized batched LU

Multifrontal
Separator tree



STRUMPACK GPU results – Summit V100



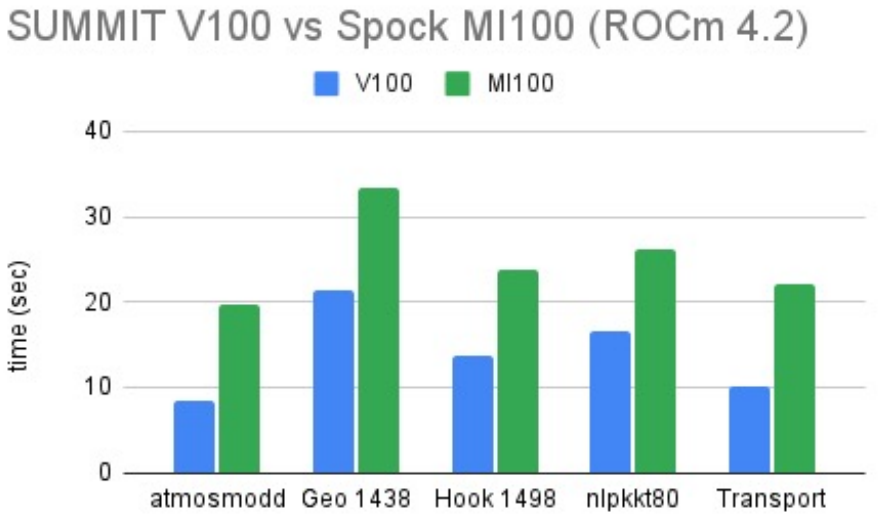


- Up to 41% of peak on single V100
- NERSC Hackathon Dec 2021: various improvements benefitting both small and large problems

P. Ghysels, R. Synk. "High performance sparse multifrontal solvers on modern GPUs", Parallel Computing, 2022

STRUMPACK for multiple types of GPUs

- Accelerator programming environments
  - CUDA, HIP/ROCm, SYCL
  - strumpack::gpu::Stream wraps cudaStream_t / hipStream_t, strumpack::gpu::DeviceMemory wraps cudaMalloc / hipMalloc and cudaFree/hipFree, ...
  - SYCL implementation is functional
    - OneAPI for BLAS/LAPACK (vbatch)
    - Can target CUDA, ROCm, OpenMP
- Single node speedup
  - V100 ~1.9x faster than MI100 (ROCm 4.2)



- Summary
  - On small scale machine, GPU can speed up more than 10x
  - On large scale machine, GPU benefit diminishes due to communication bottleneck

- Ongoing work
  - GPU-resident direct solvers, including batched