LBL Updates

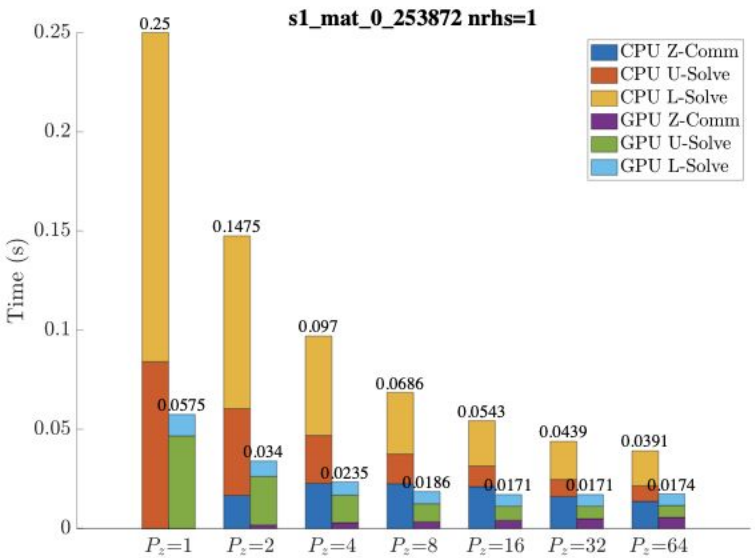
April 2023

Topics

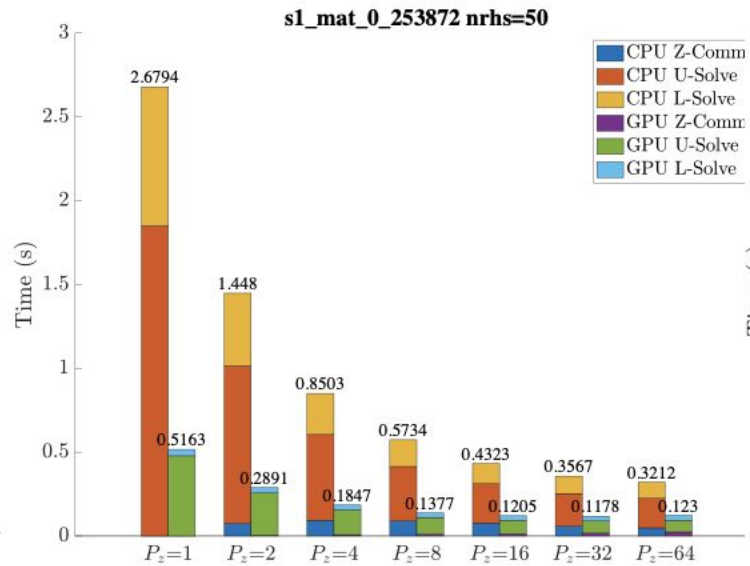
- 3D Solvers (Yang)
- Batch 1D Solves for Toroidal Preconditioning (Hans)
- One-sided Solvers on Crusher/Spock/Frontier GPUs (Nan)
- Reduced Precision (Sherry)
- Q&A

3D GPU solves on Perlmutter

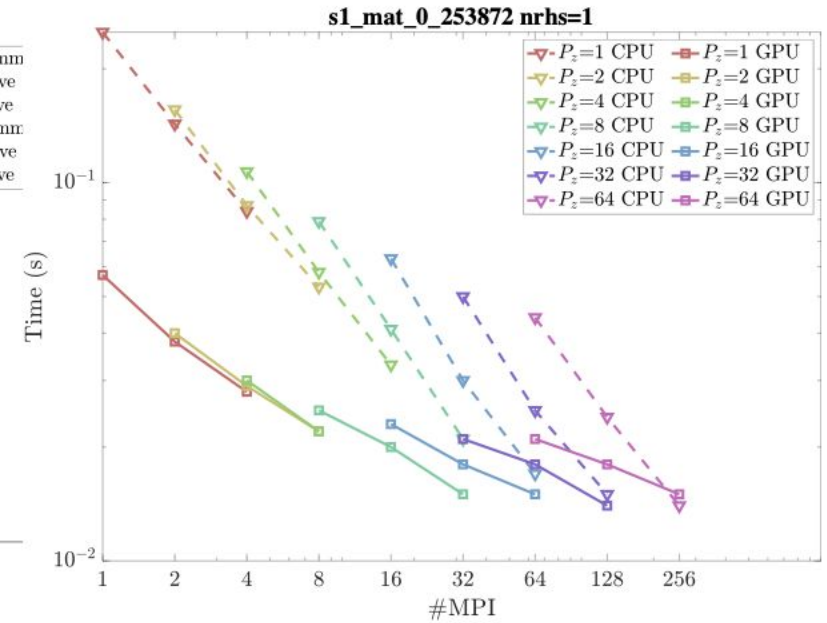
- 1 GPU per 2D grid: L solve has better GPU performance than U solve
- Multi-GPUs (NVSHMEME) per 2D grid: scales to larger number of GPUs



1 GPU per 2D grid, 1 RHS



1 GPU per 2D grid, 50 RHSs



Multi-GPU per 2D grid, 1 RHS

Batch 1D Solves for Toroidal Preconditioning (Hans)

- Got access to M3D-C1 repo Thu 3/30 & input file from Jin
  - Instructions for building on Perlmutter? Anything in mp248 project dir?
- Current (stand-alone) prototype:
  - Uses MPI Isend/Irecv/Waitall to gather data
  - MPI total comm time across ranks on 4 AMD Perlmutter nodes:
    - 0.5-2 ms for 32, 64, 128, 256 MPI ranks, little variation between ranks (async is good)
  - Solve time is negligible, all data copy and comm:
    - Library batch interface is “free” compared to comm/pack, GPU kernel launch.
    - Depends on batch / system size (use 64 poloidal planes? 2x100 DoF each plane?)
    - Have a test harness to compare SJ’s F90 answer to libraries for correctness
    - Will try cusparseSgpsvInterleavedBatch() next (pentadiagonal systems, needs 1 elim)

Batch 1D Solves for Toroidal Preconditioning (cont)

Questions:

- What matrix does the “ja2_FY23_2.f90” correspond to in M3D-C1?
  - Do we just solve once across batch and communicate back?
  - Or do we take multiple time steps before communicating (no updates to matrix)?
  - Or ?

  Affects whether we overwrite diagonal values, create temps, etc.
- Will this become a “matrix free” preconditioner (via PETSc)?
  - Jin’s BMG interface seems to build a matrix and precondition?
  - Where does this batch line solve connect in?
- How do we test if this is successful at improving convergence?
  - Could use someone to help me with this

One-Sided Solvers (Nan)

- comments/discussion/questions

Memory Usage (Sherry)

- symbolic factorization pushes up maximum memory utilization, but doesn’t need to be preserved after factorization
  - Staging/pipelining can reduce the peak
  - big engineering lift necessitating future
- SuperLU currently supports single precision factorization but need a PETSc interface
  - provides 7 digits
  - need to understand net memory benefit and convergence penalty

Feedback for LBL

- Steve will provide larger matrices for weak scaling 3D GPU studies
- Hans/Steve will communicate questions via email
- Perlmutter issues