LBL Updates

October, 2021

LBL Updates

- Continued work on multi-GPU U-Solve
  - works with NVSHMEM on Summit
- NVSHMEM/ROCSHMEM installation/runtime issues on Perlmutter(NVIDIA) and Spock(AMD)
  - Under investigation; necessary prerequisites for multi-GPU matsolves
- 3D Code
  - Now allow for uneven block row distribution for matrices (A,B)

Q&A

1. *I wonder whether there is a plan to do MatSolve on GPU. In my tests, I found that when setting the matrix type to mpiaijcusparse, then superlu_dist can do the factorization using GPU, but the iterative solver is still running on CPU, which can involves CPU-GPU communications and is slow.*
   - GPU SpTRSV (w/NVSHMEM) in SuperLU development branch (not the release version)
   - Will document compilation process / PETSc options
   - However, there are some requisite vector transformations that haven't been ported to the GPU (functional but not necessarily performant)
2. *Related to the previous question, I wonder if it is possible to offload the matrix completely after the factorization, so that communication during the iterative solving is not needed.*
   - Assumes factored matrix fits on the GPU (no GPU-GPU communication)
   - Assumes PETSc support/configured properly (no CPU-GPU communication)
   - Assumes vector transformations have been ported to GPU
   - Small CPU-GPU communication for convergence checks is unavoidable

Q&A

3. *I tried to use superlu_dist on GPU with MPI. I found that if the mpi ranks is equal to the number of GPUs, then each rank will occupy a GPU. For more MPI ranks, multiple ranks will share a GPU. I wonder what is the best strategy then, should I set the rank to be equal to # of GPUs?*
   - For single-GPU matsolves, #MPI>=#GPU (Jacobi blocks per MPI>=1)
   - For multi-GPU matsolve (w/NVSHMEM), #MPI == #GPUs
   - See caveats in Q1,2
4. *How does superlu_dist interact with ptscotch? What would happen without it?*
   - *SuperLU/PETSc currently uses either METIS, min degree*
   - *You could link with ptscotch instead of METIS (same interface/config option)*

Q&A

5. OpenMP implementation… lack of support in nvidia compiler for OMP5 features… taskloop

6. Need multi-GPU support for iterative solver…

    a. PETSc support for multi-GPU

    b. NVSHMEM support on Perlmutter

    c. blas1 vector operations (to obviate CPU-GPU communication after L/U-solve)

    d. try single-GPU branch (known PETSc support, but may have blas1 performance issues in c.)

(multi)GPU-accelerated iterative solvers

```
while(!converged){
  <<<?>>>Matvec();              // CPU has to launch kernel
  <<<?>>>Dot();                 // CPU has to launch kernel
  result=<<<?>>>l2norm();       // CPU has to launch kernel
  if(result<tol)converged=1;
}
```

Control flow runs on CPU (CPU launches kernels and checks convergence)

Computations run on GPUs

(small) batched iterative solvers

- CPU launches a GPU kernel of independent solvers.
- GPU Kernel includes a thread block for each independent system
- As thread blocks (systems) are self-contained, convergence checks can be inside the thread block (CPU doesn’t need to be involved)