Argonne NATIONAL LABORATORY

MPI AND MPICH USE IN PETSC

Junchao Zhang
(jczhang@anl.gov)

Mathematics and Computer Science Division
Argonne National Laboratory

Nov. 15, 2023

What is PETSc?

- The *Portable, Extensible Toolkit for Scientific Computation* (https://petsc.org) is a popular math library for scalable solution of scientific applications modeled by partial differential equations (PDEs)
  - Matrix, vectors, preconditioners, linear solvers, non-linear solvers, optimizers, etc
- Written in C, but has C, Fortran, Python and Rust (WIP) bindings
  - Python and Rust MPI bindings are driven by PETSc contributors
- Runs on Linux, Mac and Windows (Intel or MS MPI) from laptops to exascale machines
- Supports Nvidia, AMD and Intel GPUs with GPU-aware MPI or not
  - `-use_gpu_aware_mpi <bool>`

Overall MPI Use in PETSc

- Users do not need MPI if they only use PETSc sequentially
  - `./configure --with-mpi=0`
  - petsc will use its fake single-process MPI (mpiuni) to provide MPI APIs
- `./configure --with-cc=mpicc --with-cxx=mpicxx …`
- `./configure --with-cc=gcc --with-cxx=g++ --download-mpich …`
- Requires minimal MPI-2.1 support, and could lower it to MPI-2.0 (1997) if users really can not make it
- Supports MPI-4.0 large count (`--with-64-bit-indices`)
- Does not use MPI derived data types much, for mainly dealing with sparse data

(Key) MPI Use in PETSc (cont.)

- *Repeated, split-phased* sparse neighborhood communication in Krylov solvers
  - Default uses *persistent* MPI_Send/Recv (`-sf_type basic`)
  - Support MPI nonblocking or *persistent* neighborhood (neighbor_alltoallv)
    - `-sf_type neighbor –sf_neighbor_persistent <bool>`
  - Support MPI one-sided with various window flavors and sync mechanisms (but yet show an advantage over two-sided)
    - `-sf_type window -sf_window_flavor <create|dynamic|allocate> -sf_window_sync <fence|active|lock>`
- MPI_Allreduce() in VecNorm / VecDot (*O(1)*) or in building two-sided information from one-sided (*O(P)*)
- MPI_Iallreduce() in pipelined CG solver (`-ksp_type pipecg`)
- MPI_Ibarrier() with `-buildtwosided ibarrier`*
  - Less reliable than allreduce, always run into error at large scale

*Hoefler, Siebert and Lumsdaine, The MPI_Ibarrier implementation uses the algorithm in Scalable communication protocols for dynamic sparse data exchange, 2010

MPICH Use in PETSc

- MPICH is recommended by PETSc for users needing valgrind
- `./configure --with-cc=gcc --with-cxx=g++ --with-cuda --download-mpich`
  - Lastet MPICH will be automatically downloaded and configured with GPU support
  - MPICH extensions will be auto-detected and macros will be set up for use in PETSc code
    - PETSC_HAVE_MPIX_STREAM (for petsc to use stream-aware MPI)

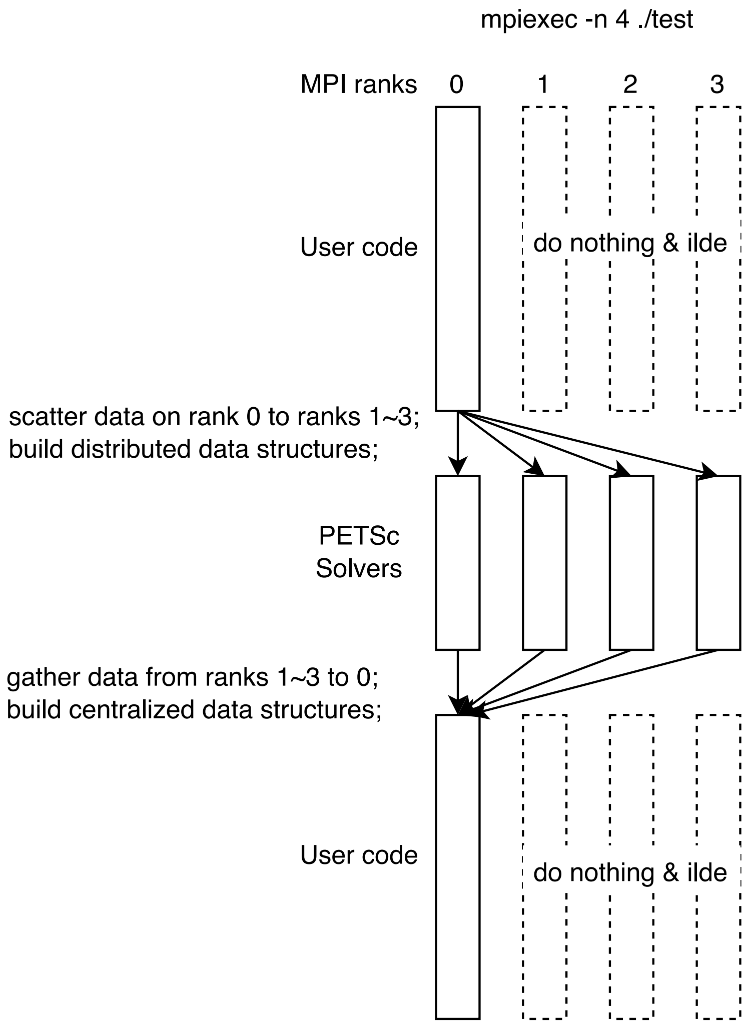
      `–-sf_use_stream_aware_mpi <bool>` (experimental)
    - PETSC_HAVE_MPIX_THREADCOMM

The “PETSc + OpenMP” Problem

- PETSc uses the flat-MPI model (i.e., no OpenMP for multicore parallelism)
  - After failed attempt to adopt OpenMP in PETSc a decade ago
- The approach works well except when some OpenMP-only codes want to use PETSc
  - To leverage the tons of solvers and algorithms within PETSc
  - It would be formidable for one to re-implement those solvers in OpenMP

The PCMPI Solution

- Run user omp code (with calls to petsc) with mpiexec
  - `mpiexec -n 4 ./test -mpi_linear_solver_server`
- Deactivate all but rank 0 in PetscInitialize() and let them wait for rank 0's commands
- The outermost KSP solver's pre-conditioner (PC) is secretly changed to type of `PCMPI`
- When user calls KSPSolve(), re-activate the idle ranks
  - Scatter data from rank 0, do petsc MPI parallel solve, then gather data to rank 0
- See https://petsc.org/release/manualpages/PC/PCMPI


mpiexec -n 4 ./test
MPI ranks
0
1
2
3
User code
do nothing & ilde
scatter data on rank 0 to ranks 1~3;
build distributed data structures;
PETSc
Solvers
gather data from ranks 1~3 to 0;
build centralized data structures;
User code
do nothing & ilde

The MPICH MPIX_Threadcomm Solution

```
Mat       A;
Vec       x, b;
int       nthreads = 4;
MPI_Comm comm;
PetscInitialize(&argc, &argv, NULL, NULL);
// user code building A, x, b etc
…
MPIX_Threadcomm_init(MPI_COMM_WORLD, nthreads, &comm);
#pragma omp parallel num_threads(nthreads)
{   Mat A2;
    Vec x2, b2;
    KSP ksp;
    MPIX_Threadcomm_start(comm); // comm's size is 4
    MatCreate(comm, &A2);
    MatCreateVecs(A2, &x2, &b2);
    // Assemble A2, b2 from the shared A, b
    KSPSolve(ksp, b2, x2);
    // Transfer the solution x2 to x
    MatDestroy(&A2);
    MPIX_Threadcomm_finish(comm)
  }
MPIX_Threadcomm_free(&comm);
PetscFinalize();
```

- Run the test as a regular OMP code:
  `OMP_NUM_THREADS=8 ./test -args`

- User's sequential code (might use OpenMP)
- PETSc is initialized on a single process
- Build sequential petsc objects such as matrices and vectors

- Build parallel petsc objects on the threadcomm *comm*
- *Somehow* transfer data from the shared sequential A, b to parallel A2, b2
- Other parts of the petsc code work as if they were run by `mpiexec -n 4 ./test`
- Caveats: petsc needs to be thread safe, e.g., in logging
- Future work: provide a new preconditioner type `PCOMP` to wrap around this stuff

Conclusion & Thanks to MPICH Developers

- PETSc is an excellent testbed and inspiring application for MPI and MPICH research
- Looking forward to greater integration between PETSc and MPICH
- Q & A