use MPI;
A Status Report

Ben Albrecht, Michael Ferguson, Nikhil Padmanabhan
Eliot Ronaghan, Greg Titus
Why Chapel + MPI?

- Co-existence with MPI-based libraries, legacy code.
- Performance for certain communication patterns
  - This should get progressively less important with time as Chapel gets its own libraries for communication patterns.
  - But enables users to not sacrifice performance in this intermediate stage
- A gateway to Chapel
  - MPI+$X$, where $X=$Chapel
Chapel + MPI: Two modes

- **SPMD**
  - MPI + X model
  - Chapel sees only one locale, all communication is through MPI
  - Similar to usual MPI programming

- **Multi-Locale**
  - Both Chapel and MPI coexist
  - Chapel locales <----- MPI ranks
    - We might want more general models
  - Mix legacy codes with Chapel -- use Chapel to simplify codes
Chapel + MPI : Status

• MPI module since v1.14
• All (....well, almost all) of MPI-1
  • Some non-blocking routines from MPI-3 included for convenience
• **NEW**: Now supports qthreads as well as fifo
• **NEW**: Tested on ugni, gasnet+aries as well as gasnet+mpi. (gasnet+infiniband coming soon)
• Chapel+MPI is ready to be used!
  • There may be a few rough edges, but we need experience using it.
Chapel + MPI + qthreads

• “CHPL_TASKS=fifo” used to be the only supported mode.
  • Fret about performance being left behind

• Problem – MPI is not qthreads aware, so can deadlock on blocking MPI calls
  • Indeed, “Hello Chapel” used to deadlock due to an MPI_Barrier call.

• Solution:
  • Use non-blocking calls whenever possible (we are slowly providing safe replacements, so that the user isn’t burdened. WIP!)
  • All blocking MPI calls must be preceded by a “Barrier()” (MPI module wrapped MPI_Ibarrier + MPI_Test)
  • No concurrent blocking calls allowed!
MPI Threading Support

• Used to require MPI_THREAD_MULTIPLE
  • For gasnet+mpi
  • Support in MPI flavors can be spotty (OpenMPI has/had? Issues)

• With gasnet+!mpi support, we can relax to MPI_THREAD_SERIALIZED
  • Of course, then user cannot make concurrent MPI calls.
  • Experimental support already exists
Chapel + MPI : Two examples

- Ring
- FFTW

See https://github.com/npadmana/chiuw2017-chpl-mpi
proc send() {
    coforall loc in Locales do on loc {
        var rank = commRank(CHPL_COMM_WORLD) : c_int,
        size = commSize(CHPL_COMM_WORLD) : c_int;
        var dest : c_int;
        dest = (rank + 1)%size;
        writeln("Rank %i sending to %i
        Send(rank, 1, MPI_INT, dest, 0, CHPL_COMM_WORLD);
        writeln("Rank %i done sending...
        }
use MPI;
use C_MPI;

use SysCTypes;
use FFTW;
use Random;
use PrivateDist;
use BlockDist;
require "fftw3-mpi.h";

forall loc in PrivateSpace {
  fftw_mpi_init();
}

forall loc in PrivateSpace {
  fftw_mpi_cleanup();
}
written("Goodbye, Brad! I hope you enjoyed this distributed FFTW example");
const DSpace={0..#Ng,0..#Ng,0..#Ng2};
var targets: [0..#numLocales,0..0,0..0] locale;
targets[..,0,0]=Locales;
const D: domain(3) dmapped Block(boundingBox=DSpace, targetLocales=targets) = DSpace;

forall loc in PrivateSpace {
    var idx = B.localSubdomain().low;
    Barrier(CHPL_COMM_WORLD);
{
    // MPI calls
    var fwd = fftw_mpi_plan_dft_r2c_3d(Ng, Ng, Ng, B[idx], B[idx], CHPL_COMM_WORLD, FFTW_ESTIMATE);
    execute(fwd);
    destroy_plan(fwd);
}
}

var ksum2 : real;
ksum2 = 2*(+ reduce B[..,..,2..(Ng-1)]**2);
ksum2 += (+ reduce B[..,..,0..1]**2);
ksum2 += (+ reduce B[..,..,Ng..(Ng+1)]**2);
ksum2 *= invNg3;
writef("Total sum B^2 = %er , error= %er\n",ksum2, ksum2/sum2 - 1);