Hybrid programming

- Goal: to see how one can mix openMP, MPI, Threads, ...
- MPI and openMP + Threads/MPI
- Cuda and openMP
- CUDA and MPI

Mixing programming models

- Many models of programming can be mixed
- Most important ones:
  - openMP + MPI
  - Cuda + MPI
  - Cuda + openMP
- How does this work: a matter of finding the right combination of libraries to link togeher.

openMP + MPI

- Goal: mixing fine grain parallelism with coarse grain parallelism
- Example: Domain Decomposition-type application. Local computations involve adding vectors, local matvecs, etc. These can benefit from openMP
- Most compilers are 'aware' of openMP.. so often all you need is add

See next example `hybrid.c` == Compile with

mpicc -fopenmp -lgomp -o hybrid.ex hybrid.c

```
#include <stdio.h>
#include <mpi.h>
#include <omp.h>
int main(int argc, char *argv[]){
  int numprocs, rank, namelen;
  char proc_nam[MPI_MAX_PROC_NAM];
  int iam = 0, np = 1;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Get_proc_nam(proc_nam, &namelen);
  #pragma omp parallel default(shared) private(
iam, np)
  { np = omp_get_num_threads();
    iam = omp_get_thread_num();
    printf("Thread %d out of %d from proc. %d out
 of %d on %s\n",
    iam, np, rank, numprocs, proc_nam);
  }
  MPI_Finalize();
  return 0;
}
```

openMP + Cuda

- Goal: mixing SIMT type calculations with loop-level parallelism
- Works by adding `-fopenmp -lgomp` in compilation and
- Also add `#include <omp.h>` to your .cu programs where omp pragmas are used.
- A simple example to be shown.

MPI + Cuda

- Goals: to enable GPU processing in MPI codes when there are GPUs available
- This is a very common situation now.
- Need to add some include files and cuda libraries in makefile
- Cuda-aware MPI. Versions 1.7 and later of openMPI are 'Cuda-aware' In other words they allow to send/ access Cuda buffers directly (avoiding *cuda memory copies*).