Hands On #1 (C)

- Download and install MPICH on laptop and run cpi example in examples directory
- Copy cpi.c over to Mira, compile, and run
- Compile and run MLife examples from wgropp.cs.illinois.edu/advmpi-16.tgz on Mira
  - Untar, run configure, cd to code-examples-c and make to build executable

Hands On #1 (Fortran)

- Download and install MPICH on laptop and run pi3f90 example in examples/f90 directory
- Copy pi3f90.f90 over to Mira, compile, and run
- Compile and run MLife examples from wgropp.cs.illinois.edu/advmpi-16.tgz on Mira
  - Untar, run configure, cd to code-examples-f9x and make to build executable

Hands On #2 (C)

1. Build and run mesh2d on Vesta or your favorite system
   – Run with large enough meshes; e.g., mlife2d –x 1000 –y 1000
2. Take the file mlife2d-pt2pt.c and modify it to try one of the following (we recommend testing on your laptop first):
   - Persistent sends
   - Ready sends
   - Sendrecv
     – Advanced: Restructure to allow computation during communication
3. Measure the performance (modify mlife2d.c to include your new routines)

Hands On #2 (Fortran)

1. Build and run mesh2d on Vesta or your favorite system
   - Run with large enough meshes; e.g., mlife2d –x 1000 –y 1000
2. Take the file mlife2d-pt2pt.f90 and modify it to try one of the following (we recommend testing on your laptop first):
   - Persistent sends
   - Ready sends
   - Sendrecv
     - Advanced: Restructure to allow computation during communication
3. Measure the performance (modify mlife2d.f90 to include your new routines)

Hands On #3 (C)

- Take the file mlife2d-pt2ptuv.c and
  1. Create a new version that that uses MPI_Type_vector.
  2. Compare with mlife2d-pt2pt.c . How does your version compare?
     - I.e., use the version in mlife2d-pt2pt.c as the solution to this exercise
  3. Advanced: Replace Type_vector with Type_create_resized with stride to produce a type that can be used for any length vector with the same stride
  4. Advanced: Compare performance with manual packing and with an estimate of performance (how fast should it be)?

Hands On #3 (Fortran)

- Take the file mlife2d-pt2ptuv.f90 and
  1. Create a new version that that uses MPI_Type_vector.
  2. Compare with mlife2d-pt2pt.f90 . How does your version compare?
     - I.e., use the version in mlife2d-pt2pt.c as the solution to this exercise
  3. Advanced: Replace Type_vector with Type_create_resized with stride to produce a type that can be used for any length vector with the same stride
  4. Advanced: Compare performance with manual packing and with an estimate of performance (how fast should it be)?

Hands On #4 (C)

- Run mlife2d with different sizes of meshes:
  - Mlife2d –x 2000 –y 2000 –i 100
  - Mlife2d –x 4000 –y 4000 –i 100
- Observe the RMA performance using Fence synchronization
- Create a new version, starting from mlife2d-fence.c, that uses MPI_Win_lock/unlock synchronization. How does your version perform?
  - Hint: How do you know when you can safely proceed to the next iteration? Or what do you need to do in the code to ensure you can move to the next iteration?

Hands On #4 (Fortran)

- Run mlife2d with different sizes of meshes:
  - Mlife2d –x 2000 –y 2000 –i 100
  - Mlife2d –x 4000 –y 4000 –i 100
- Observe the RMA performance using Fence synchronization
- Create a new version, starting from mlife2d-fence.f90, that uses MPI_Win_lock/unlock synchronization. How does your version perform?
  - Hint: How do you know when you can safely proceed to the next iteration? Or what do you need to do in the code to ensure you can move to the next iteration?

Hands On #5 (C)

- Measure performance of halo exchange with different process mappings, using mlife2d
- Advanced (extra credit): compare with expected performance, using a simple communication performance model

Hands On #5 (Fortran)

- Measure performance of halo exchange with different process mappings, using mlife2d
- Advanced (extra credit): compare with expected performance, using a simple communication performance model