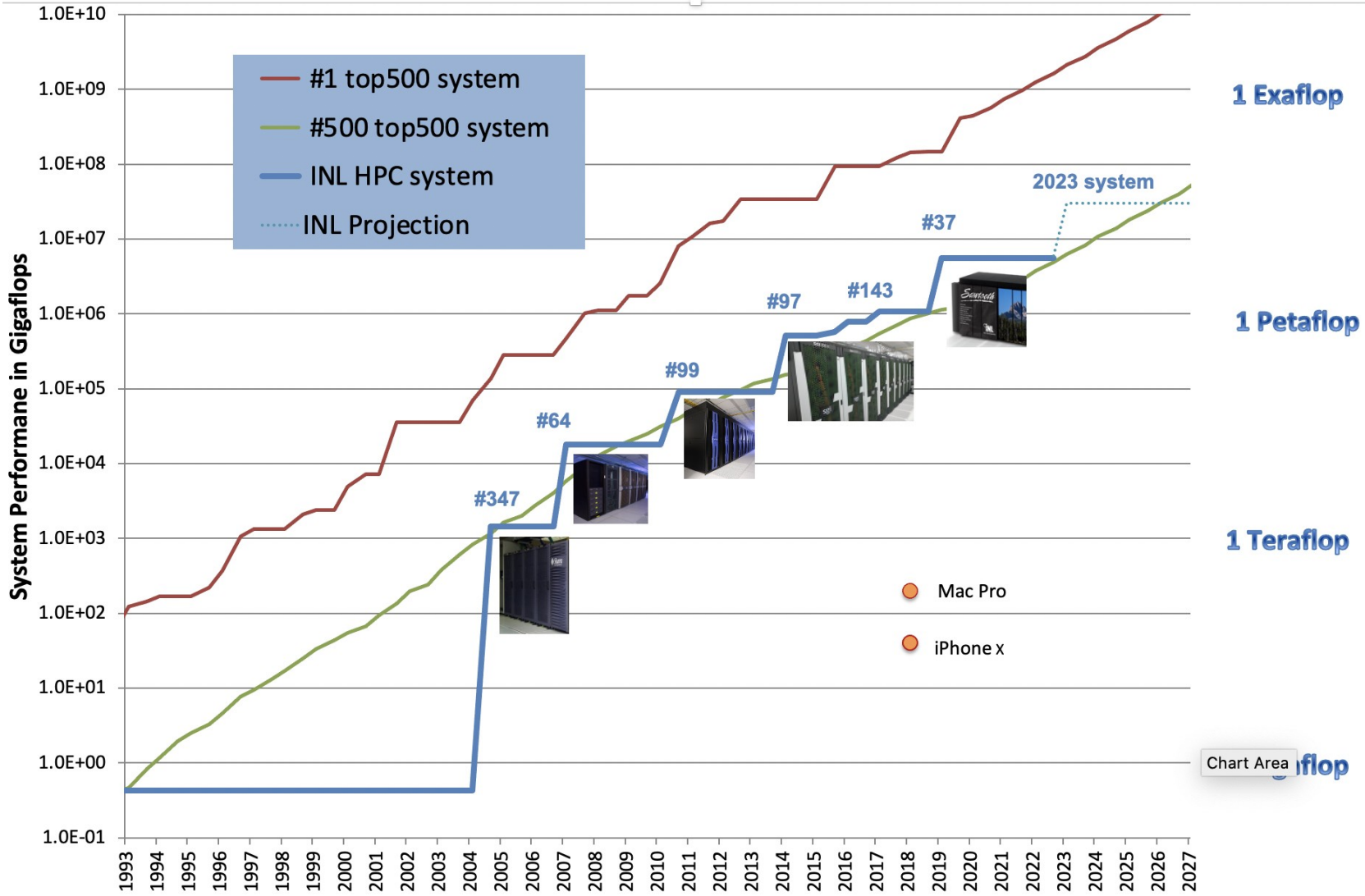
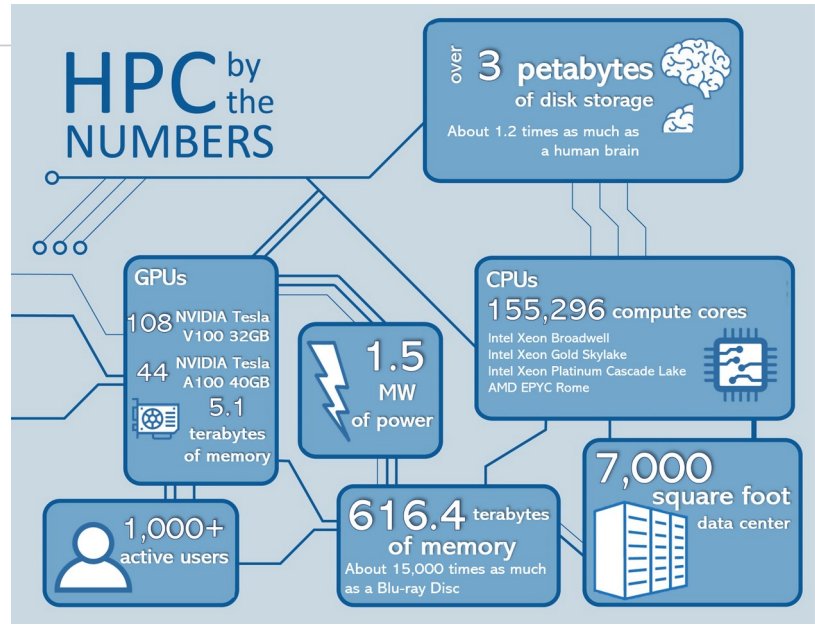
Aggressive Asynchronous Communication in the MOOSE framework using MVAPICH2

Matthew Anderson

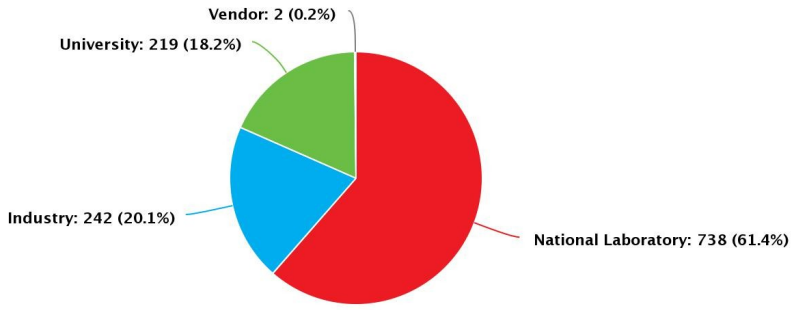
INL/CON-22-68721

Idaho National Laboratory High Performance Computing

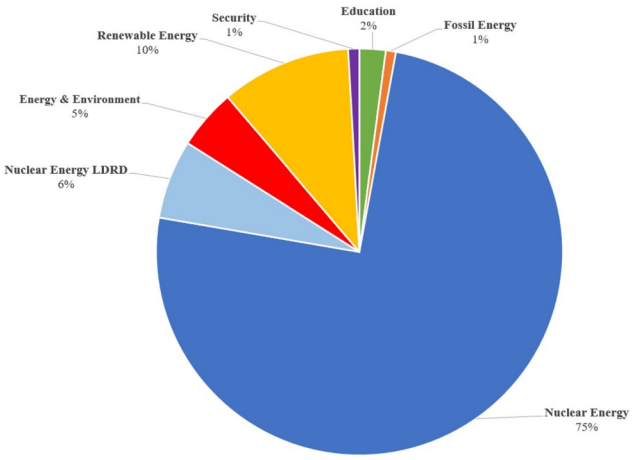




HPC User Affiliations

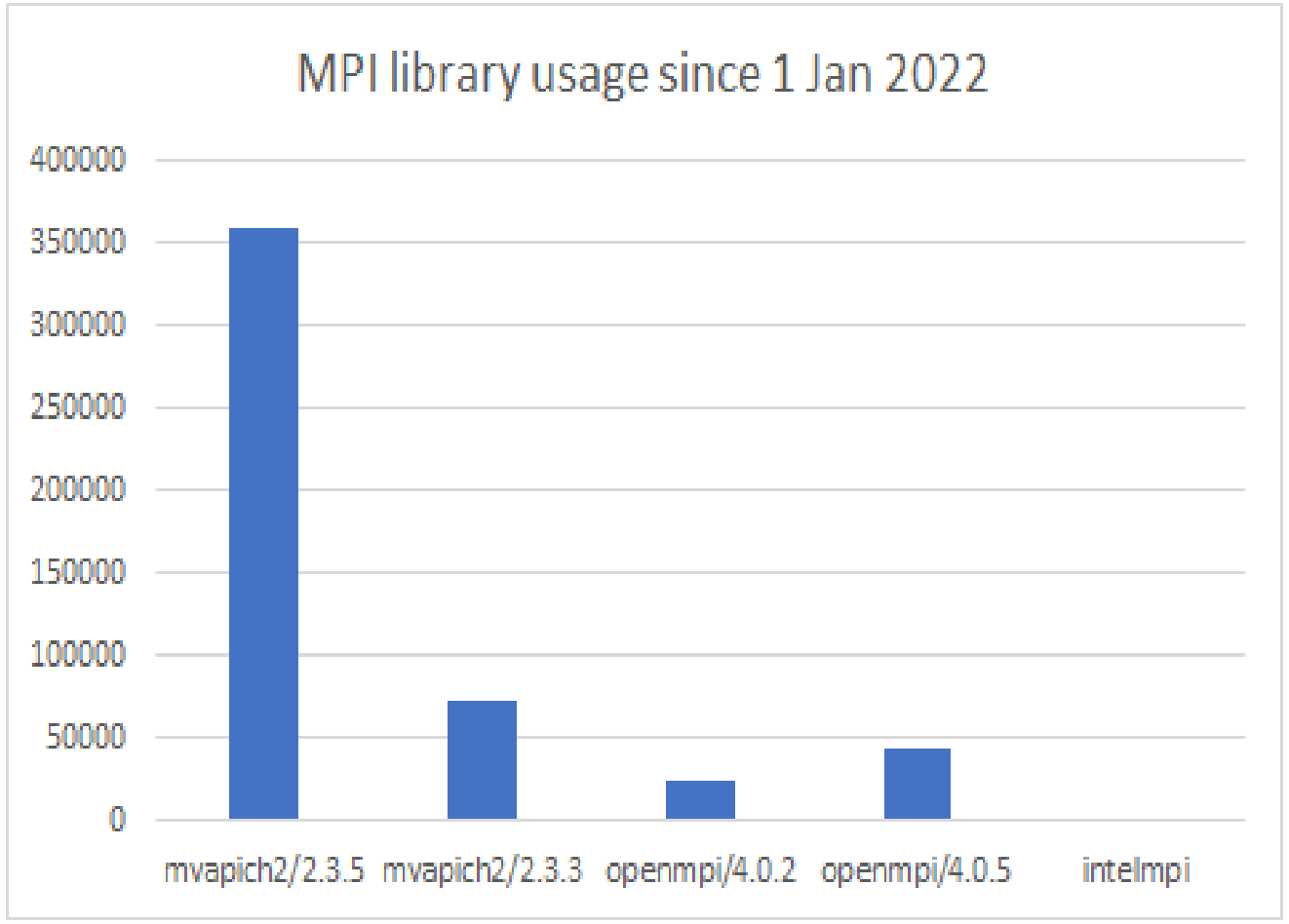


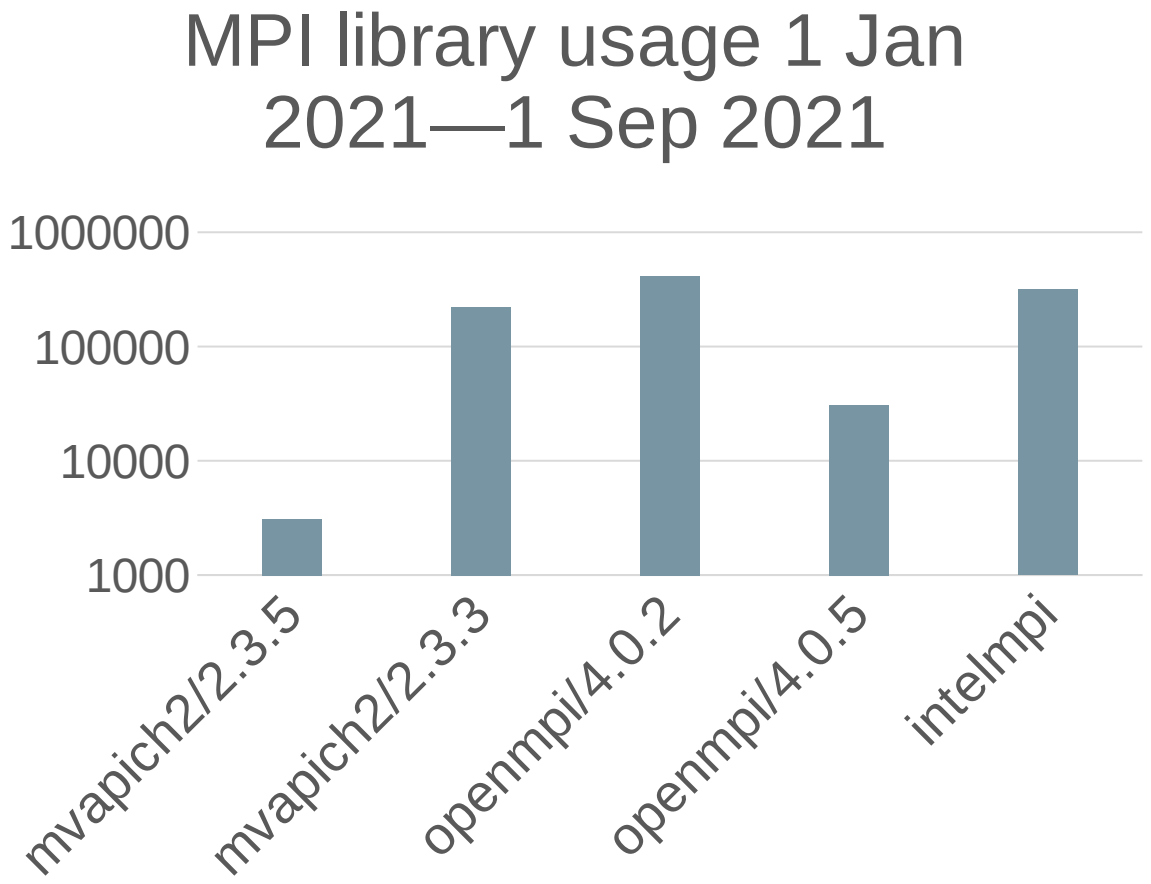
INL HPC Systems



System Name Year	Core Count (Memory)	Processor	Network	GPUs	Performance (TOP 500 rank)	FY22 Core Hours	FY22 Average queue length per core
Sawtooth 2019	99,792 (394TB)	Intel Cascade Lake	Mellanox Infiniband EDR	108 NVIDIA V100	5.6 PFLOPS (#37 2019)	714 million	5.7
Lemhi 2018	20,160 (94TB)	Intel Skylake	Omnipath	NA	1.0 PFLOPS (#427 2018)	127 million	2.7
Falcon 2014 (2017 refresh)	34,992 (121TB)	Intel Broadwell	Mellanox Infiniband FDR	NA	1.1 PFLOPS (#92 2014)	200 million	1.5
Hoodoo 2021	352 (5.6TB)	AMD EPYC 7302	Mellanox Infiniband HDR	44 NVIDIA A100	0.858 PFLOPS (Never on TOP500 list)	10.2 million	NA

Rapid adoption of MVAPICH2 2.3.5 on INL HPC systems



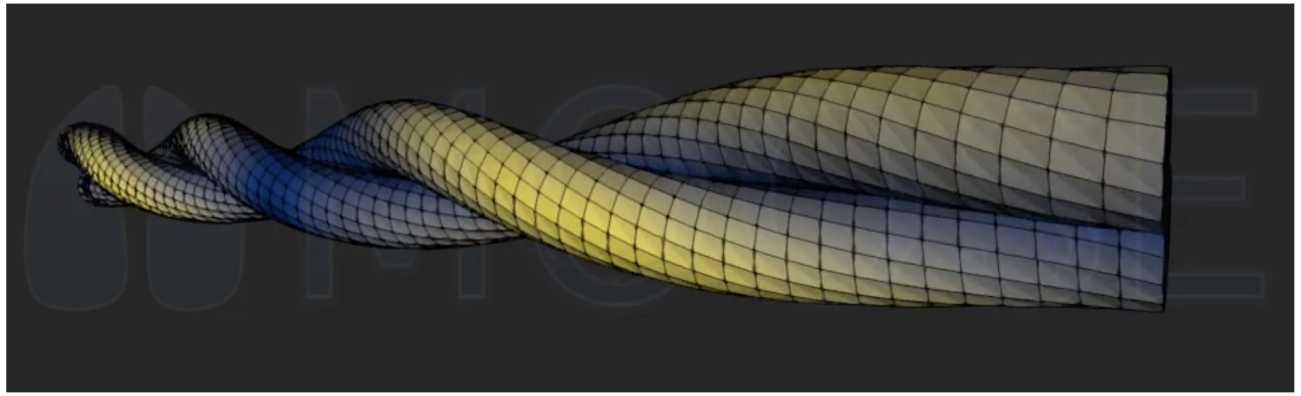


What drives MVAPICH2 usage on INL HPC systems?

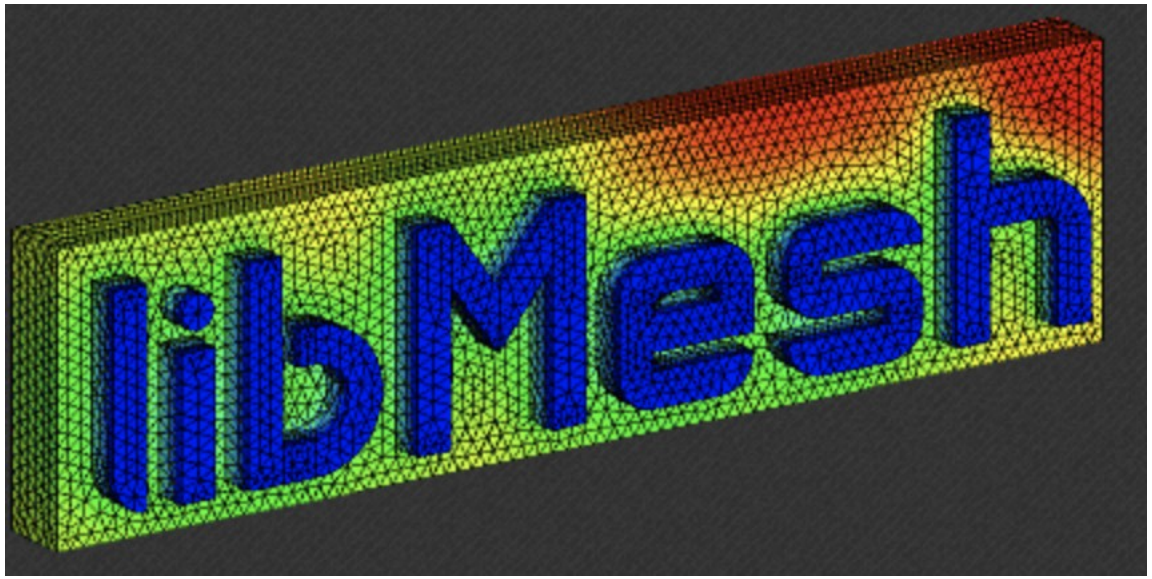
MOOSE

Multiphysics Object-Oriented Simulation Environment

An open-source, parallel finite element framework

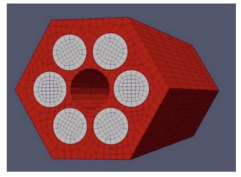


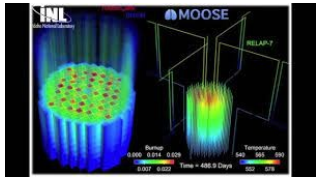
PETSc

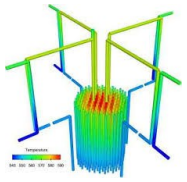


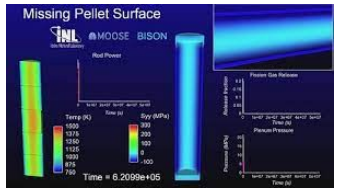
MOOSE herd apps

Application	Domain	More information
Bison	Nuclear fuel performance	https://mooseframework.inl.gov/bison/index.html
Blue crab	Nuclear plant systems analysis	https://www.osti.gov/servlets/purl/1766199
Dire Wolf	Heat-pipe microreactor analysis	https://gain.inl.gov/MicroreactorProgramTechnicalReports/Document-INL-EXT-20-59691.pdf
Griffin	Neutron diffusion solver	https://doi.org/10.1016/j.anucene.2021.108546
Marmot	Mesoscale fuel performance	https://mooseframework.inl.gov/magpie/getting_started/Marmot.html
Mastodon	Multiscale Hazard Analysis	https://mooseframework.inl.gov/mastodon/
Pronghorn	Advanced reactor thermal hydraulics	https://doi.org/10.1080/00295450.2020.1825307
Sabertooth	Fuel performance and thermal hydraulics with neutronics	https://inldigitallibrary.inl.gov/sites/sti/sti/Sort_41824.pdf
Sockeye	Heat-pipe analysis	https://doi.org/10.1080/00295450.2020.1861879









Building MVAPICH2 For PETSc and Libmesh usage

- ./configure MAKE=/usr/bin/gmake CC=gcc CFLAGS='-fPIC -g -O -fno-omit-frame-pointer' AR=/usr/bin/ar ARFLAGS=cr CXX=g++ CXXFLAGS='-g -O -fPIC -std=gnu++11 -fno-omit-frame-pointer' FFLAGS='-fPIC -g -O' FC=gfortran F77=gfortran FCFLAGS='-fPIC -g -O' --enable-shared --with-device=ch3:mrail --with-pbs=/opt/pbs --enable-g=meminit --with-rdma=gen2 --disable-ibv-dlopen

MV2_THREADS_PER_PROCESS 1

MV2_HOMOGENEOUS_CLUSTER 1

MV2_USE_LAZY_MEM_UNREGISTER=0

The ”new” addition:

MV2_USE_SHARED_MEM=0

Transition to aggressive asynchronous communication

```cpp
for (unsigned int message=0; message < NUM_MESSAGES; message++)
{
  // Can't wait until C++17
  sends.emplace_back();
  auto & request = sends.back();

  MPI_Issend(send_buffer.data(), MESSAGE_SIZE, MPI_DOUBLE, pid, TAG, MPI_COMM_WORLD, &request);

  num_sends_started++;

  // See if we need to do some receives and use that same moment to expire some sends
  if (!(num_sends_started % CHUNK_SIZE))
  {
    num_messages_received += check_and_receive();

    sends.remove_if(send_complete);
  }
}
```

```cpp
unsigned int check_and_receive()
{
  unsigned int num_received = 0;

  std::array<double, MESSAGE_SIZE> receive_buffer;

  while (true)
  {
    MPI_Status status;

    int flag = false;

    MPI_Iprobe(MPI_ANY_SOURCE, TAG, MPI_COMM_WORLD, &flag, &status);

    if (flag) // Receive the waiting message
    {
      MPI_Recv(receive_buffer.data(), MESSAGE_SIZE, MPI_DOUBLE, MPI_ANY_SOURCE, TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

      num_received++;
    }
    else // No waiting messages
      break;
  }

  return num_received;
}
```

In MOOSE applications, we see this tends to run the VBUF pool out of memory.
Initially tried adjusting MV2_VBUF_TOTAL_SIZE and MV2_IBA_EAGER_THRESHOLD to be smaller (thereby only really tiny messages use the vbuf pool)

Working with Hari Subramoni and Nick Contini (thank you!), moved to using MV2_USE_SHARED_MEM=0

Conclusions

- The Moose framework heavily relies on MVAPICH2
  - Involves Libmesh and PETSc
- MVAPICH 2.3.5 now version of choice at INL for almost all nuclear energy codes
- Aggressive eager communication in MOOSE/libmesh/PETSc/Hypre has created some issues where the VBUF pool runs out of memory.