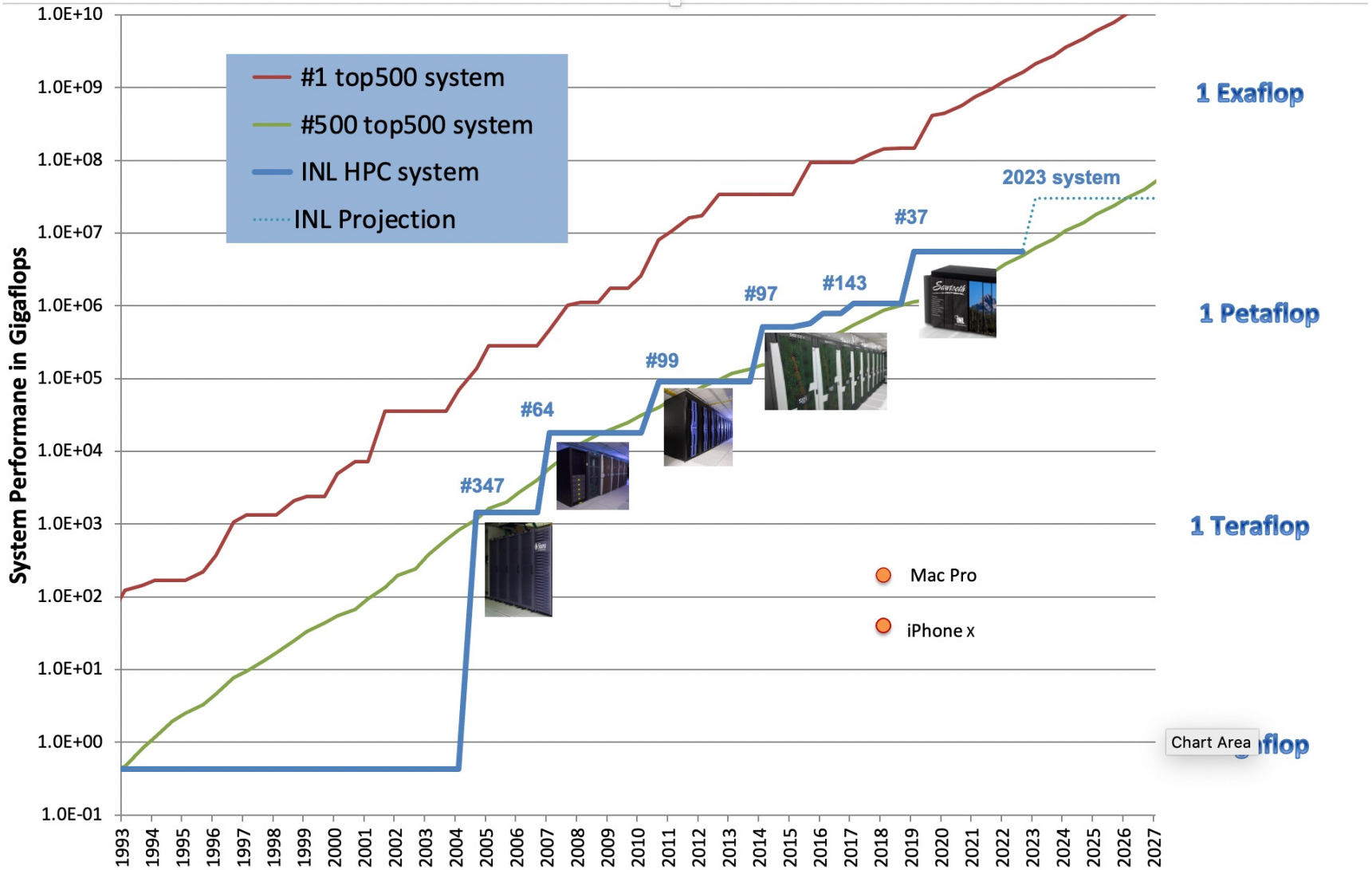
MVAPICH Integration for PBS Pro

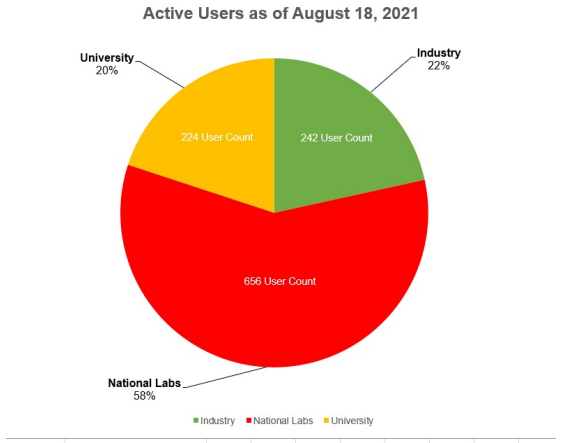
Matthew Anderson

INL/CON-21-64080

Idaho National Laboratory High Performance Computing

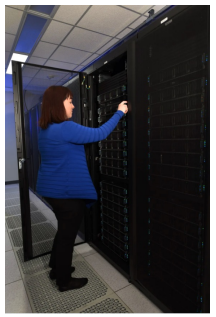


INL HPC System Summary



INL HPC supports three petaflops scale systems:

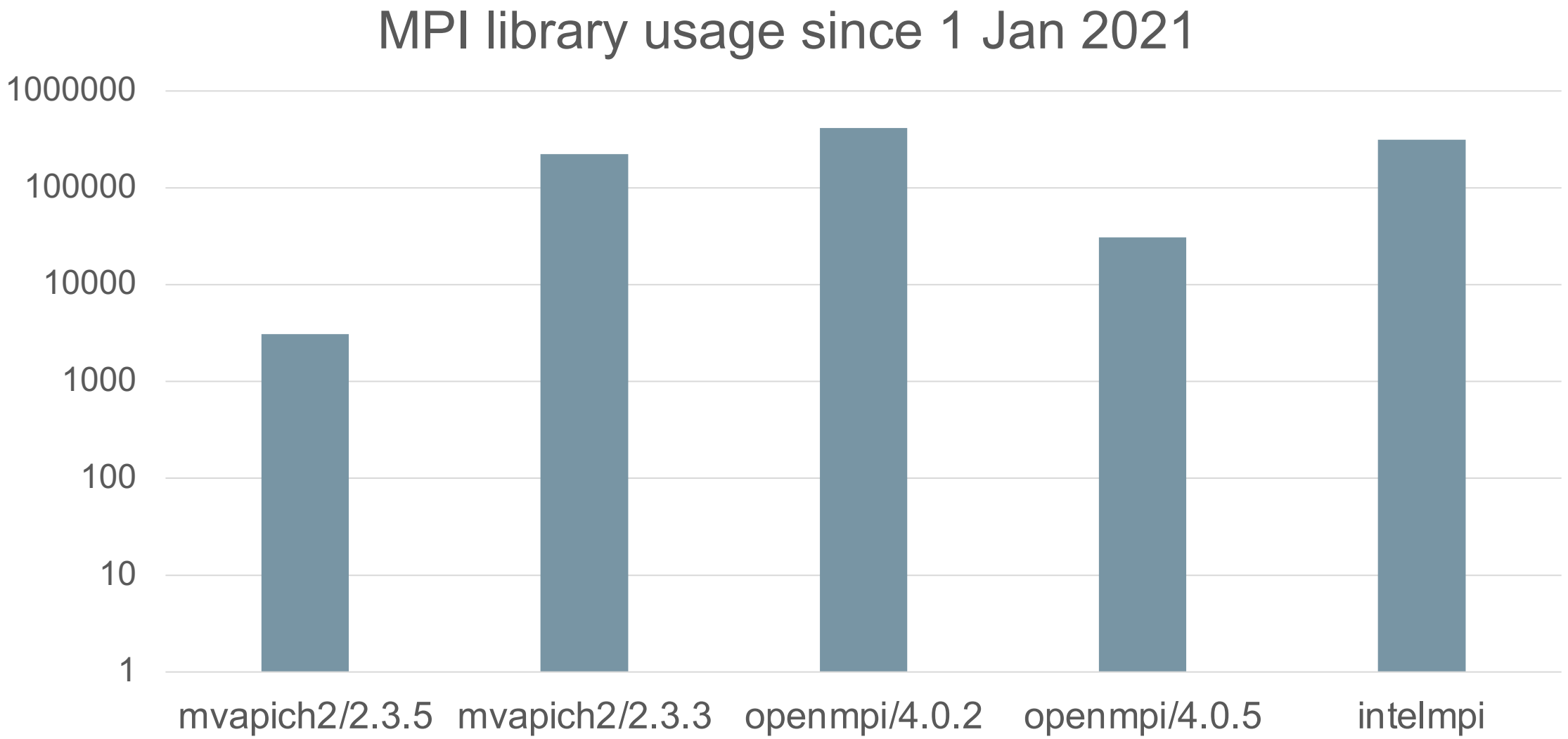
Sawtooth, Lemhi, and Falcon



All with PBS Pro

System Name, Year	Core Count (Memory)	Processor	Network	GPUs	Peak Performance	FY21 Core Hours	FY21 Oversub-scription Factor
Falcon 2014 (refreshed in 2017)	34,992 (121TB)	Intel Broadwell	Mellanox Infiniband FDR	No	1.087 PFLOPS	202 million	1.39
Lemhi 2018	20,160 (94TB)	Intel Skylake	Omnipath	No	1.0 PFLOPS	122 million	1.59
Sawtooth 2019	99,792 (394TB)	Intel Cascade Lake Platinum	Mellanox Infiniband EDR	108 NVIDIA V100 GPUs	5.6 PFLOPS	561 million	2.39

MVAPICH usage on INL HPC systems



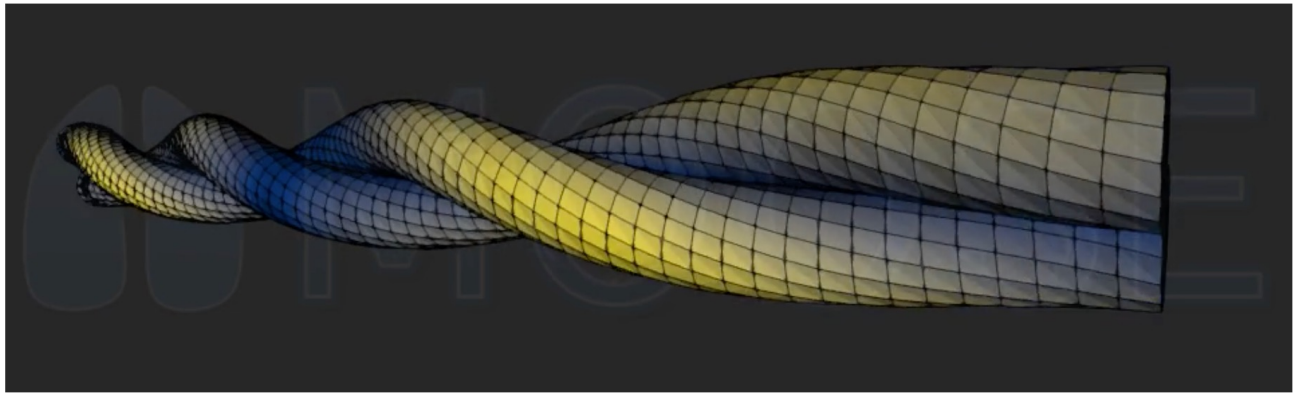
What drives MVAPICH usage on INL HPC systems?

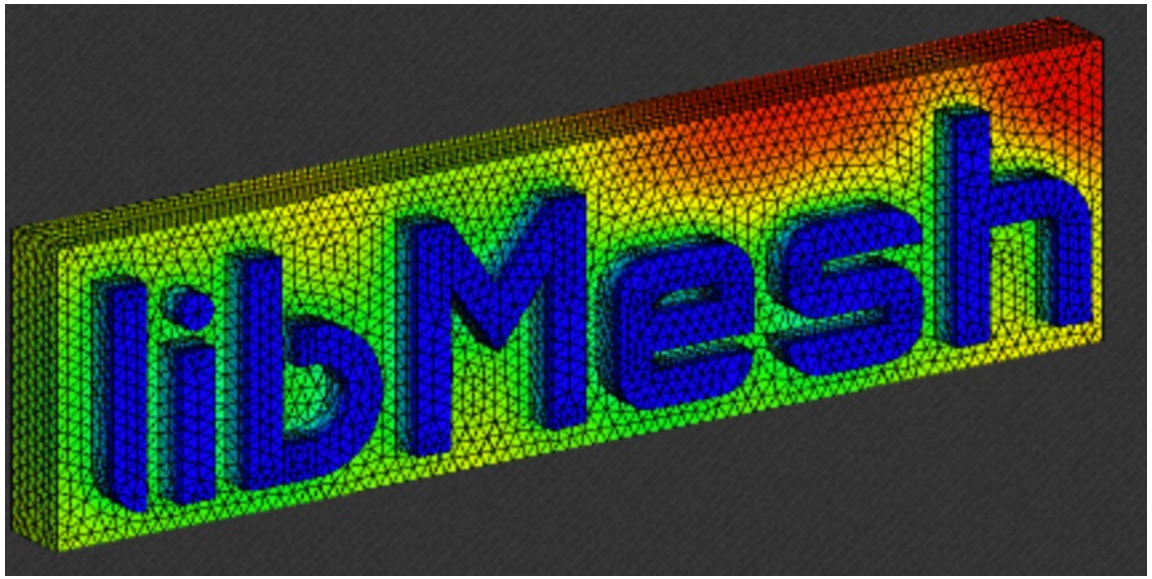
MOOSE

Multiphysics Object-Oriented Simulation Environment

PETSc

An open-source, parallel finite element framework





MVAPICH and tight integration with PBS Pro

Motivation: Track memory usage for each process inside PBS logs, e.g. PBS logs look like this…

update_job_usage: CPU percent: 3587

update_job_usage: CPU usage: 644911.266 secs

update_job_usage: CPU percent: 3575

update_job_usage: CPU usage: 649281.617 secs

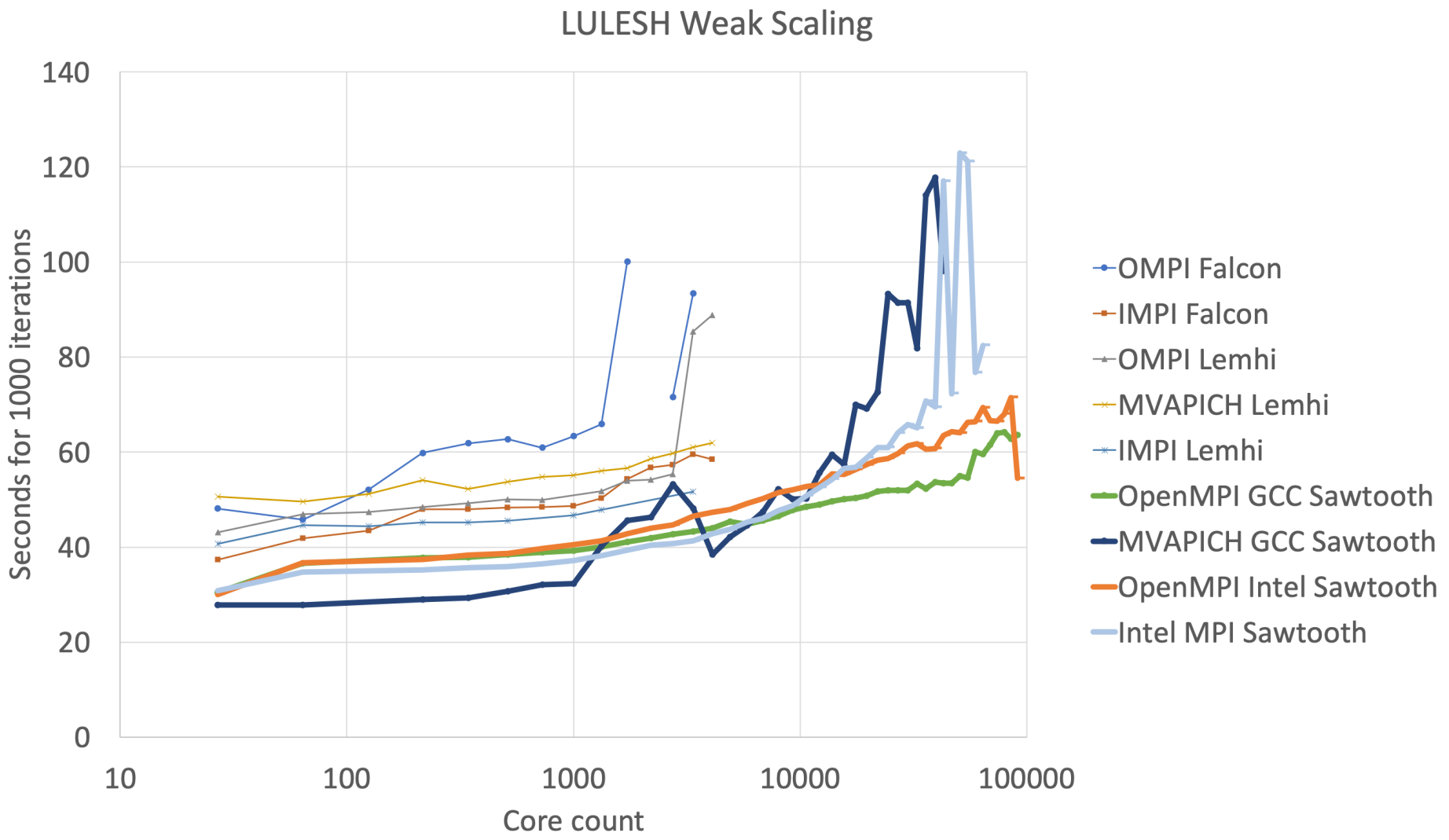
update_job_usage: Memory usage: mem=116356180kb

Logs are typically in /var/spool/pbs/mom_logs. The logs show that tasks are attached to the PBS jobID and will report cpu and memory usage by the job's tasks. It's easy for us to check if we know the PBS jobID; PBS has a script (pbs_dtj) to go out and parse the appropriate logs for only the job in question, so it's easier to find the relevant log messages.

We go for tighter integration by linking directly to the PBS task manager (libpbs) instead of just using pbs_attach.

Implemented in mvapich2 2.3.5 (Thanks Hari Subramoni!)

LULESH Performance on INL HPC systems



OSU MPI Latency Test v5.7

# Size	Latency (us)
0	1.38
1	1.50
2	1.49
4	1.49
8	1.48
16	1.52
32	1.52
64	1.54
128	1.59
256	1.98
512	2.08
1024	2.28
2048	2.65
4096	3.60
8192	5.27
16384	7.67
32768	9.85
65536	12.55
131072	18.03
262144	29.04
524288	50.70
1048576	94.24
2097152	181.47
4194304	355.46

Building MVAPICH 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

Valgrind MPI Allreduce release2.3.5 with tight integration

Using host libthread_db library "/lib64/libthread_db.so.1".
Core was generated by `../../../moose_test-opt -i diffusion_hmg.i Mesh/dmg/nx=100 Mesh/dmg/ny=100 Debu'.
Program terminated with signal 11, Segmentation fault.
#0 0x00002aaab4408b2e in MPL_trdump () from /apps/local/mvapich2/2.3.5-gcc-8.4.0/lib/libmpi.so.12

- ==98001== 88 (24 direct, 64 indirect) bytes in 1 blocks are definitely lost in loss record 31 of 54
- ==98001== at 0x4C29F73: malloc (vg_replace_malloc.c:307)
- ==98001== by 0xDAE1D5E: hwloc_bitmap_alloc (bitmap.c:74)
- ==98001== by 0xDA7523F: get_socket_bound_info (mv2_arch_detect.c:898)
- ==98001== by 0xD93C87A: create_intra_sock_comm (create_2level_comm.c:593)
- ==98001== by 0xD93BEBA: create_2level_comm (create_2level_comm.c:1762)
- ==98001== by 0xD59A894: mv2_increment_shmem_coll_counter (ch3_shmem_coll.c:2183)
- ==98001== by 0xD4E4CBB: PMPI_Allreduce (allreduce.c:912)
- ==98001== by 0x99F1766: PetscAllreduceBarrierCheck (pbarrier.c:26)
- ==98001== by 0x99F70BE: PetscSplitOwnership (psplit.c:84)
- ==98001== by 0x9C5C26B: PetscLayoutSetUp (pmap.c:262)
- ==98001== by 0xA08C66B: MatMPIAdjSetPreallocation_MPIAdj (mpiadj.c:630)
- ==98001== by 0xA08EB9A: MatMPIAdjSetPreallocation (mpiadj.c:856)
- ==98001== by 0xA08F6D3: MatCreateMPIAdj (mpiadj.c:904)

hwloc_bitmap_free(cpubind_set);

Missing in the end of

get_socket_bound_info().

Fixed in 2.3.6

still many other valgrind issues in mvapich, but they seem harmless

Conclusions

- The Moose framework heavily relies on MVAPICH
  - Involves Libmesh and PETSc
- MVAPICH 2.3.5 and later have tight PBS Pro integration!
- MPI_Allreduce valgrind clean in 2.3.6
- Still many other valgrind issues in MVAPICH but they appear harmless. If these can be cleaned up, it would make it easier to catch problems.