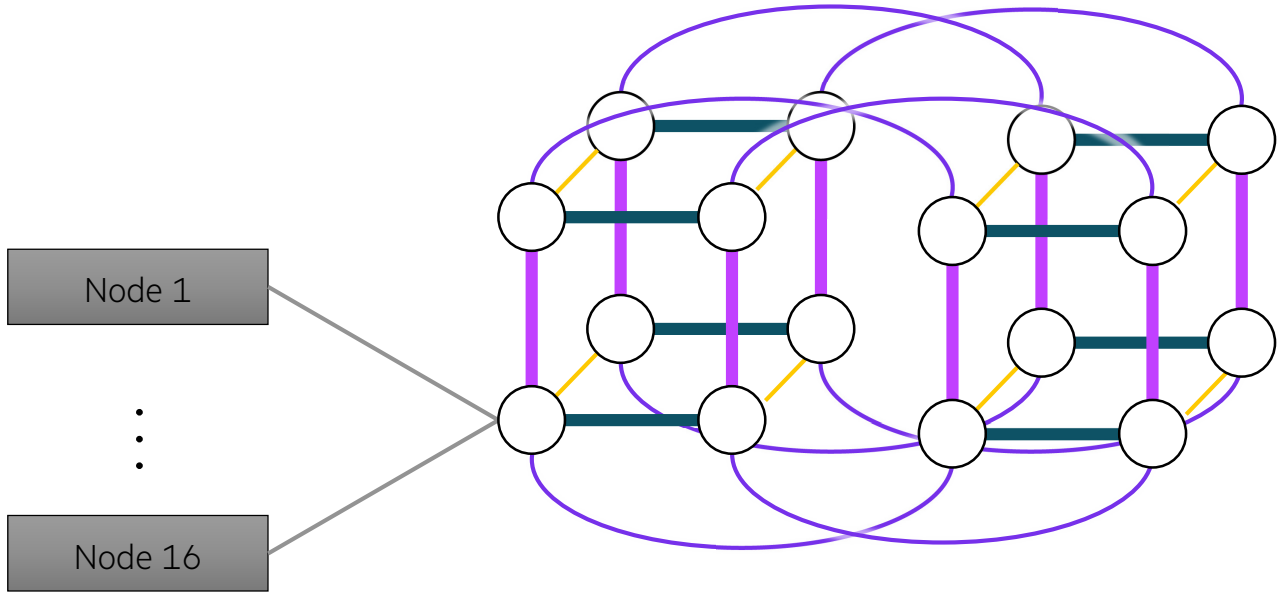
Hewlett Packard
Enterprise

NETWORK TOPOLOGY

B. Dick, HLRS

INTERCONNECT TOPOLOGY



1D (line)
2D (square)
3D (cube)
4D (hypercube)

...

9D (hypercube)

- 16 nodes connected to a common switch
- switches arranged as a **9D hypercube**
  - HLRS plans to deploy topology aware scheduling (blocks of 64 nodes) and MPI placement
- enhanced B/W on 1st two dimensions (thicker lines)

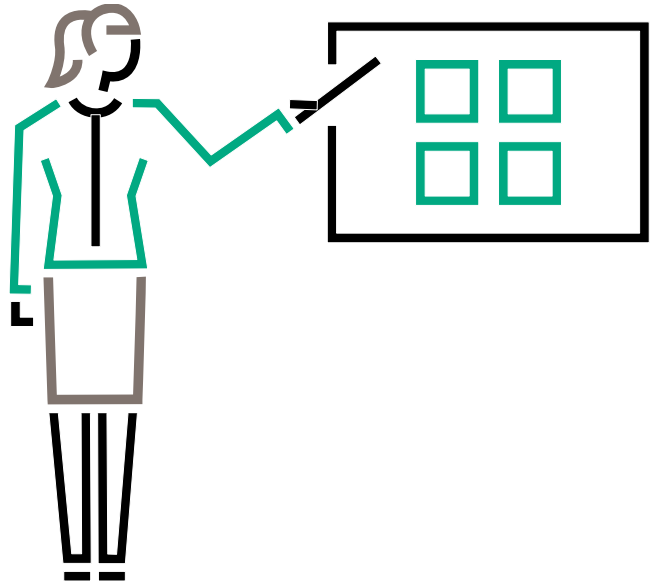
Hewlett Packard
Enterprise

HPE MPI INTRODUCTION

Michael Raymond, Christian Simmendinger

SCHEDULE

- Introduction
- Software Modules
- Building Applications
- Launching jobs
- Environment Variables & Tuning



CAPABILITIES

- MPI 3.1
- OpenSHMEM 1.4
- Scales to over 200k ranks and over 15k hosts
- Compatible with all popular 3rd party profiling & debugging tools

REFERENCES

- HPE Performance Software – Message Passing Interface Guide
- MPI(1)
- mpirun(1)
- intro_shmem(1)
- HPE Customer Service

DIFFERENCES

- HPE MPI is not based on MPICH or Open MPI
- Has its own ABI
  - HPE MPT has its own unique ABI
  - HPE HMPT is ABI compliant with the MPICH ABI
  - They are otherwise the same software
  - It is safe to install them both
- Has a different way of launching worker processes
- No static libraries

MODULES

- HPE MPI uses module files for setting paths
- The default installation location varies by the Linux distribution
  - RHEL: /usr/share/Modules/modulefiles/
  - SLES: /usr/share/modules/modulefiles/
  - HLRS: /opt/..

- E.g.:

```
$ source /etc/profile.d/modules.sh
$ module use /usr/shared/modules/modulefiles/
$ module load mpt
```

If you relocate the MPT files, just update BASEPATH in the mpt module file

Hewlett Packard
Enterprise

HPE MPI
Building Applications

MAKING THINGS SIMPLE

```
$ module load mpt
$ mpicc -c bar.c
$ mpicc -o bar bar.c
$ ldd ./bar
        linux-vdso.so.1 =>  (0x00007ffff7ffd000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007ffff7dc9000)
        libcpuset.so.1 => /lib64/libcpuset.so.1 (0x00007ffff7bbb000)
        libbitmask.so.1 => /lib64/libbitmask.so.1 (0x00007ffff79b6000)
        libmpi.so => /opt/hpe/hpc/mpt/mpt-2.21/lib/libmpi.so (0x00007ffff75eb000)
        libc.so.6 => /lib64/libc.so.6 (0x00007ffff7229000)
        /lib64/ld-linux-x86-64.so.2 (0x000055555554000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007ffff7025000)
        librt.so.1 => /lib64/librt.so.1 (0x00007ffff6e1d000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007ffff6c06000)
```

SPECIFYING THE ACTUAL COMPILER

- Defaults to GNU for C/C++ and Intel for Fortran
- Can specify with an environment variable or on the command line
- E.g. with C
  - export MPICC_CC=icc
  - mpicc -cc=icc ….
- See the mpicc(1) man page

FULL CONTROL

- MPI_ROOT points to the base directory

$ module load mpt

$ cc -I ${MPI_ROOT}/include -L ${MPI_ROOT}/lib -o bar bar.c -lmpi

$ cc -I ${MPI_ROOT}/include -L ${MPI_ROOT}/lib -o foo foo.c -lmpi –lsma

- The software module will also set:
  - $CPATH
  - $LIBRARY_PATH
  - $FPATH
  - $PATH
  - $LD_LIBRARY_PATH

THE -RPATH-LINK OPTION

- The -rpath-link option lets you specify a fixed location of where to find a shared library
- This may speed up launch time on systems with libraries on network file systems
- Downsides:
  - Prevents the installation of multiple versions of HPE MPI
  - Prevents some collaboration options

LIBRARY INFORMATION

LIBMPI_MT.SO VS LIBMPI_ST.SO

- _st.so
  - Best performance
  - No MPI_THREAD_MULTIPLE support
- _mt.so
  - Fully thread safe
- libmpi.so is symlinked to libmpi_mt.so
- Suggestion: Just link against -lmpi
- Key point: If an application is linked against both, bad things will happen

Hewlett Packard
Enterprise

HPE MPI
JOB LAUNCHING AND CPU PINNING

MPIRUN - BASIC OPTIONS

- Typical Usage

  $ mpirun ./a.out
- Specific number of ranks / PEs

  $ mpirun -np 4096 ./a.out
- Verbose output

  $ mpirun -v ./a.out

BASIC OPTIONS OUTSIDE OF A RESOURCE SCHEDULER

- Run on the local host

  $ mpirun -np 32 ./a.out

- Simple multi-host

  $ mpirun r1i0n0,r1i0n1 -np 16 ./a.out

ADVANCED OPTIONS

- Specify ranks / PEs per cluster host

  $ mpirun -ppn 16 ./a.out
- Enable spawning with reserved hosts

  $ mpirun -spawn ./a.out
- Use multiple binaries

  $ mpirun -np 32 ./a.out –c –d : -np 64 ./b.out : -np 16 ./a.out -e
- Use TotalView or DDT

  $ mpirun -tv ./a.out

  $ mpirun -ddt ./a.out

MORE ADVANCED OPTIONS

- Prefix output with MPI_COMM_WORLD rank

  $ mpirun -p “%w:” -np 32 ./a.out

- Take environment variables from a file

  $ mpirun -configfile envs.txt -np 32 ./a.out

- Take more command-line options from a file

  $ mpirun -f hostlist.txt –f ranks.txt ./a.out

MPI_DSM_DISTRIBUTE

- Enables CPU pinning
- Pins ranks on cores 0…N-1
  - On IB/OPA systems, local rank 0 is pinned on the socket closest to the primary HCA/HFI. Pinning proceeds linearly from there.
- Core #s are cpuset relative
- This feature is enabled by default

MPI_DSM_CPULIST

- Enables specifying specific core pinning
- Need to include ":allhosts" to affect all hosts
- Typical example:
  - MPI_DSM_CPULIST="0,1,8,9,16,17:allhosts"
- Supports striding:
  - MPI_DSM_CPULIST="0-7/2:allhosts"
  - Gives cores 0, 2, 4, 6
- See MPI(1) for more details

MPI_DSM_VERBOSE

- When enabled, prints pinning info at launch
- E.g.

MPT DSM information

MPT MPI_DSM_DISTRIBUTE enabled

wrank	grank	lrank	pinning	node name	cpuid
0	0	0	yes	r2i1n9	0
1	1	1	yes	r2i1n9	1
2	2	2	yes	r2i1n9	2
3	3	3	yes	r2i1n9	3

DMPLACE & OMPLACE

DPLACE

- CPU pinning tool that intercepts all forks, pthread creates, and execs
- Supports explicit placement and load-balancing pinning
- Typical usage:

$ mpiexec_mpt dplace -s 1 -e -c 0,2,4,6 ./a.out

- The -s 1 skips the shepherd
- The -e exactly places in the specified order
- Core #s are cpuset-relative

- Pin to each (C)ore before moving to the next (S)ocket
  - $ dplace -c SC ./a.out
- Pin to first core on each socket, then the 2nd core…
  - $ dplace -c CS ./a.out
- See dplace(1) for more details

OMPLACE

- Makes dplace easier to use with OpenMP / pthreads
- Parses OMP_NUM_THREADS and spreads the processes out appropriately
- Alternately, use -nt to specify the number of pthreads to assume

  $ mpirun omplace –nt 4 ./omp_app

  - It will set OMP_NUM_THREADS for you
- Use -v to have the placement plan printed out

MPI_TASK_GEOMETRY

- Enables putting specific MPI ranks on specific hosts
- You might use this for reasons of performance or resource consumption
- Example:

  layout.txt

  {(0,3)(1,2,5)(4)}

  $ export MPI_TASK_GEOMETRY=layout.txt

  $ mpirun …

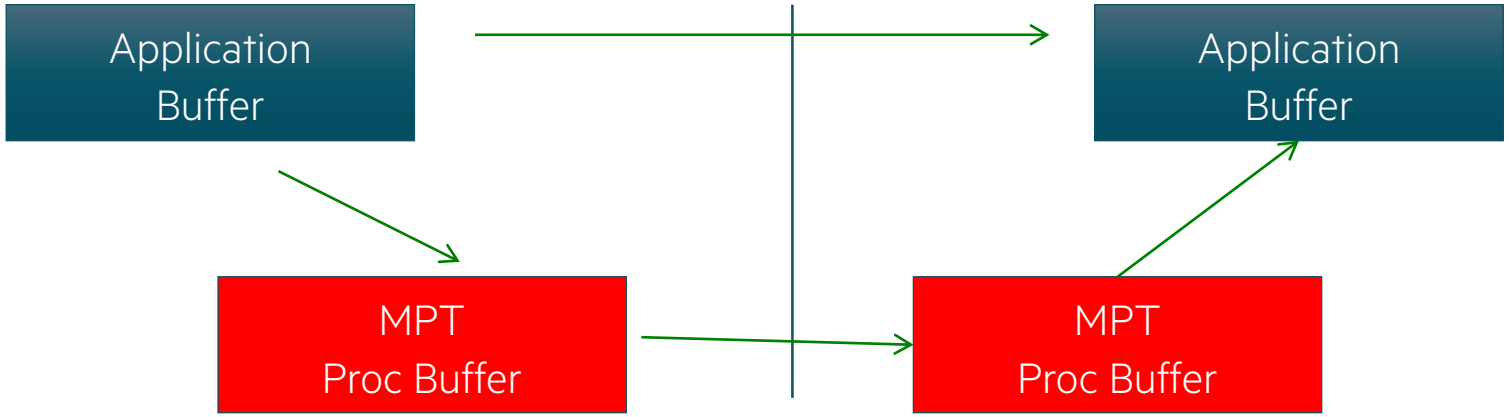
  - This puts ranks 0 & 3 on the first host, 1,2,5 on the second host, and 4 on the third host
  - The file starts and ends with “{}”. Hosts are within “()”
  - The first entry must be 0 because of how MPT routes stdin.

Hewlett Packard
Enterprise

HPE MPI
TUNING OF MPT

BUFFERING VS ZERO-COPY

- Buffering: Sender can keep going
- Zero-copy: Less memory and faster movement
- Control the threshold for blocking messages with MPI_BUFFER_MAX
  - If set, above this value messages are zero-copy
  - MPI_BUFFER_MAX=0 gives the best performance for the osu_latency microbenchmark



AMOUNT OF INTERNAL BUFFERS

- MPT keeps some number of internal 16k buffers
- Used for buffering and (un)packing non-contiguous datatypes
- MPI_BUFS_PER_PROC=128 by default
- Some users set to 2048
- Flexibility vs memory consumption and caching effects

BUFFERS WARNING MESSAGE

- You may see this warning:

MPT Warning: Could not allocate an internal send buffer in the last 30 seconds on rank 31 at r1i3n2. Try increasing MPI_BUFS_PER_PROC.
Alternatively, destination rank 175 on host r1i2n13 may be running slowly.

- Try increasing MPI_BUFS_PER_PROC next time
- It may indicate fabric / hardware problems though

COLLECTIVES

HIGHLY OPTIMIZED COLLECTIVES

- MPT uses heuristics to determine which optimized version of a collective to use
- Sometimes the heuristics choose poorly
- Set MPI_COLL_OPT=false to disable all optimizations

CONTROLLING THE HEURISTICS

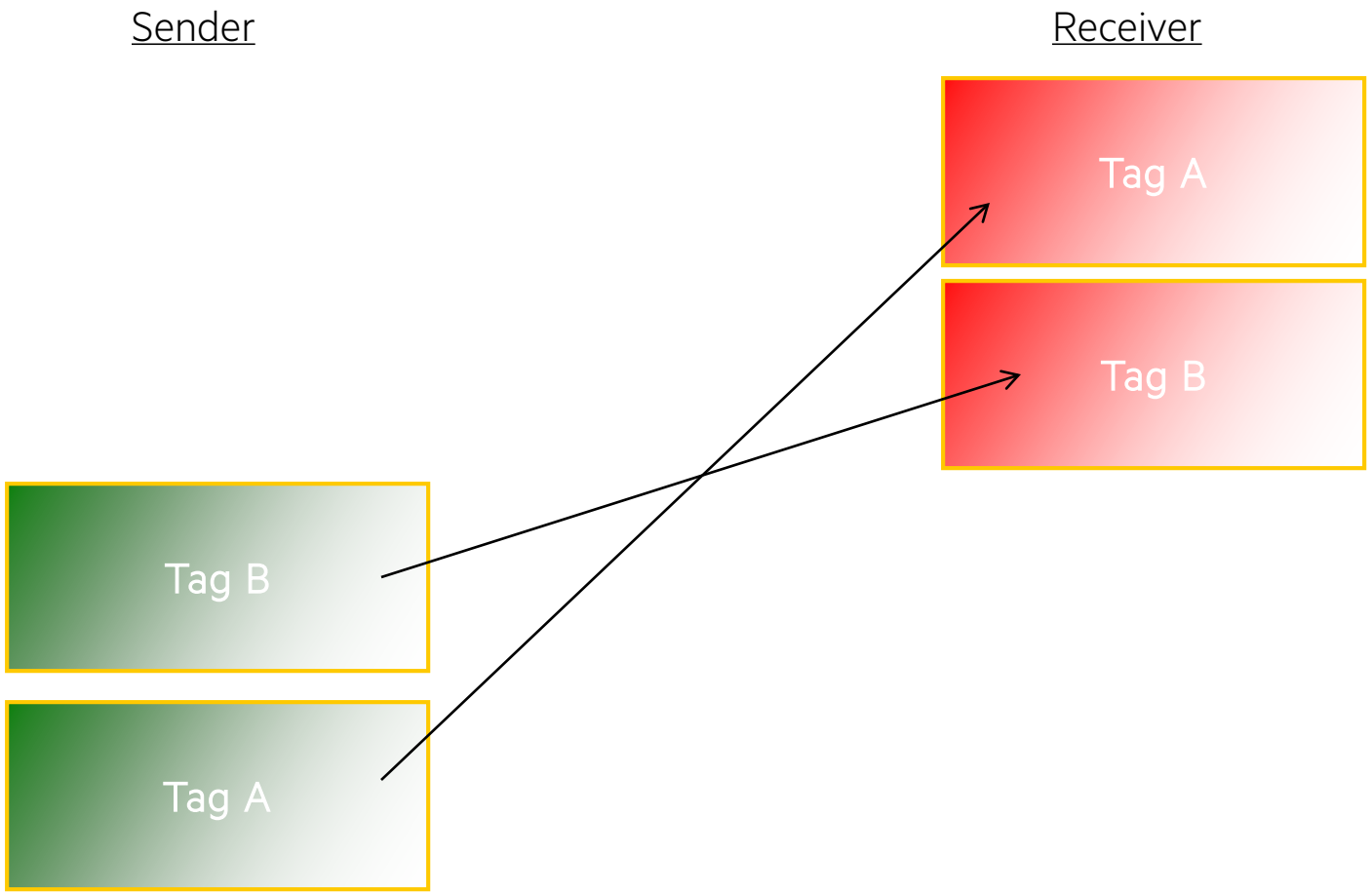
- You can force MPT to use specific implementations of each collective
- The MPI_ADJUST_* variables specify a specific optimization for all data and rank sizes
- E.g. MPI_ADJUST_ALLREDUCE=1 forces the use of recursive doubling
- See MPI(1) for more details

FLOATING POINT REPRODUCIBILITY

- The order of floating point operations can affect the results
- Don’t count on the same internal actions for reductions across different versions or even node allocations
- Set MPI_COLL_REPRODUCIBLE=true to force the same method to be used every time
- A small performance hit

INFINIBAND

TAG MATCHING IN HARDWARE



MELLANOX TAG MATCHING

- Implemented mostly in hardware whereas tag matching on OPA is all in software
- Has limits on the number of receive buffers that can be posted to the hardware

- Enabled with MPI_IB_TM=true
- Performance
  - Slightly slows down some microbenchmarks
  - Shows slightly better overlap with IMB-NBC Ialltoall

MELLANOX OFFLOAD - CONFIGURATION

$ export HPCX_HOME=/store/hpcx/latest/hpcx-v1.4.355-gcc-MLNX_OFED_LINUX-3.1-1.0.5-suse11.3-x86_64

- Or whatever your path is

$ module use $HPCX_HOME/modulefiles

$ module load hpcx

$ module load mpt

- After loading hpcx

$ export MPI_COLL_HCOLL=true

INFINIBAND - LARGE RUNS

- IB has several static settings that should be tweaked for large jobs
- MPT has individual variables for them
- The MPI_IB_CONGESTED toggle changes all of them to the best known values for large jobs
- Some sites are setting this for all their jobs
- You may also want to route your IB traffic on different QoS service levels
  - Configure this with MPI_IB_SERVICE_LEVEL
  - Put your interactive/shell traffic at the highest level
  - Put your Lustre traffic on the middle level
  - Put your MPI traffic on the lowest level so it does not starve Lustre

DEFAULT @ HLRS

DEFAULT ENVIRONMENT

- MPI_COLL_HCOLL=true
- MPI_CHECK_ARGS=true
- MPI_BUFFER_MAX=2048
- MPI_REQUEST_DEBUG=true
- MPI_IB_TM=true
- MPI_XPMEM_ENABLED=true

- MPI_SYSLOG_COPY=2
- MPI_IB_TIMEOUT=22
- MPI_IB_SERVICE_LEVEL=1


Hewlett Packard
Enterprise