The ABCs of Open MPI

Decoding the Alphabet Soup of the Modern HPC Ecosystem (Part 3)

Ralph H. Castain, Jeffrey M. Squyres

Presented in conjunction
with the EasyBuild community

Webex Logistics

- This session is being recorded
- Ask questions in the Q&A panel

Overview

- Background
- PMIx: What is it?
- Building Open MPI

$\left.\begin{array}{l}\\ \\ \\ \end{array}\right\}$ Covered in part 1

- PMIx (cont.)
- A breakdown of Open MPI:
  - The run-time stuff
  - The MPI stuff

$\left.\begin{array}{l}\\ \\ \\ \\ \end{array}\right\}$ Covered in part 2

- PRRTE
- Configuration / debugging tips
- The upcoming Open MPI v4.1.x series
- The upcoming Open MPI v5.0.x series

Recap

- Too much to cover for a real “recap”
- Part 1
  - YouTube video
  - PDF slides
- Part 2
  - YouTube video
  - PDF slides

PRRTE

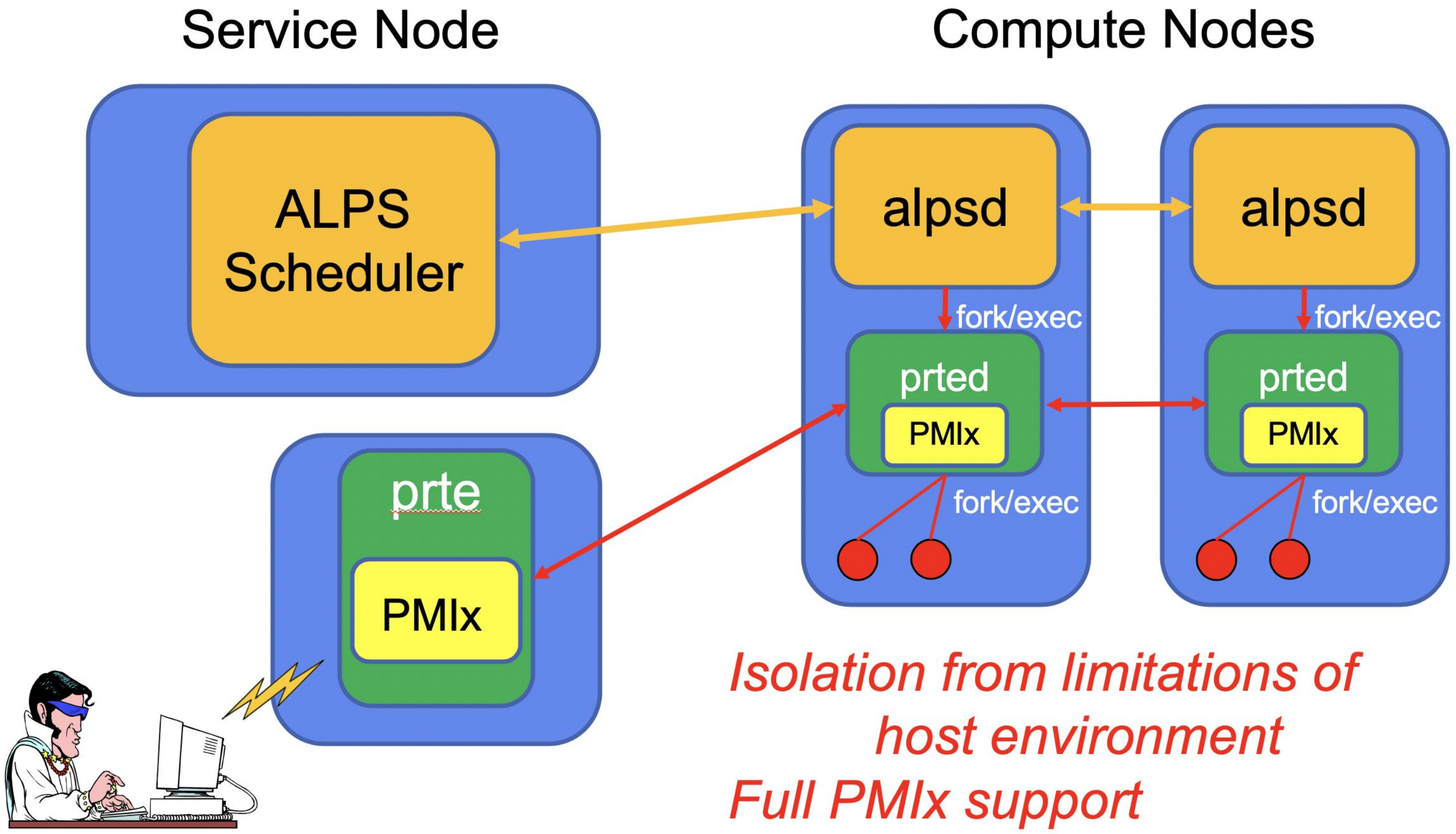
Open PMIx Questions from Session 2

- Examples of applications using async/cross-model stuff?
  - https://eurompi2018.bsc.es/sites/default/files/uploaded/eurompi2018-paper-vallee.pdf
- Pros/cons of srun vs mpirun
  - Mpirun
    - Offers more options, larger range of PMIx support
      - Dynamics, job control, monitoring
    - Historically was MPI implementation specific
      - Changes with OMPI v5’s use of PRRTE
  - Srun
    - Works the same (placement, binding) regardless of MPI implementation
- Separate talk by Ralph on PMIx Launch Orchestration?
  - Happy to do so - will schedule it

What is PRRTE?

- PMIx Reference RunTime Environment
    - Supports full range of PMIx
    - Per-user development environment for PMIx-based tools and apps
    - Provides a “shim” to environments that don’t have full PMIx support
- Persistent Distributed Virtual Machine (DVM)
    - Launches daemons on all allocated nodes at beginning of session
    - User launches applications against the DVM
    - Tear down DVM when user session is done (user commands)
- Derived from OpenRTE (ORTE)
    - Forked from OMPI several years ago
    - Standalone project within PMIx community
    - Moving towards distribution with PMIx (Spack, OpenHPC)

How Is It Used?



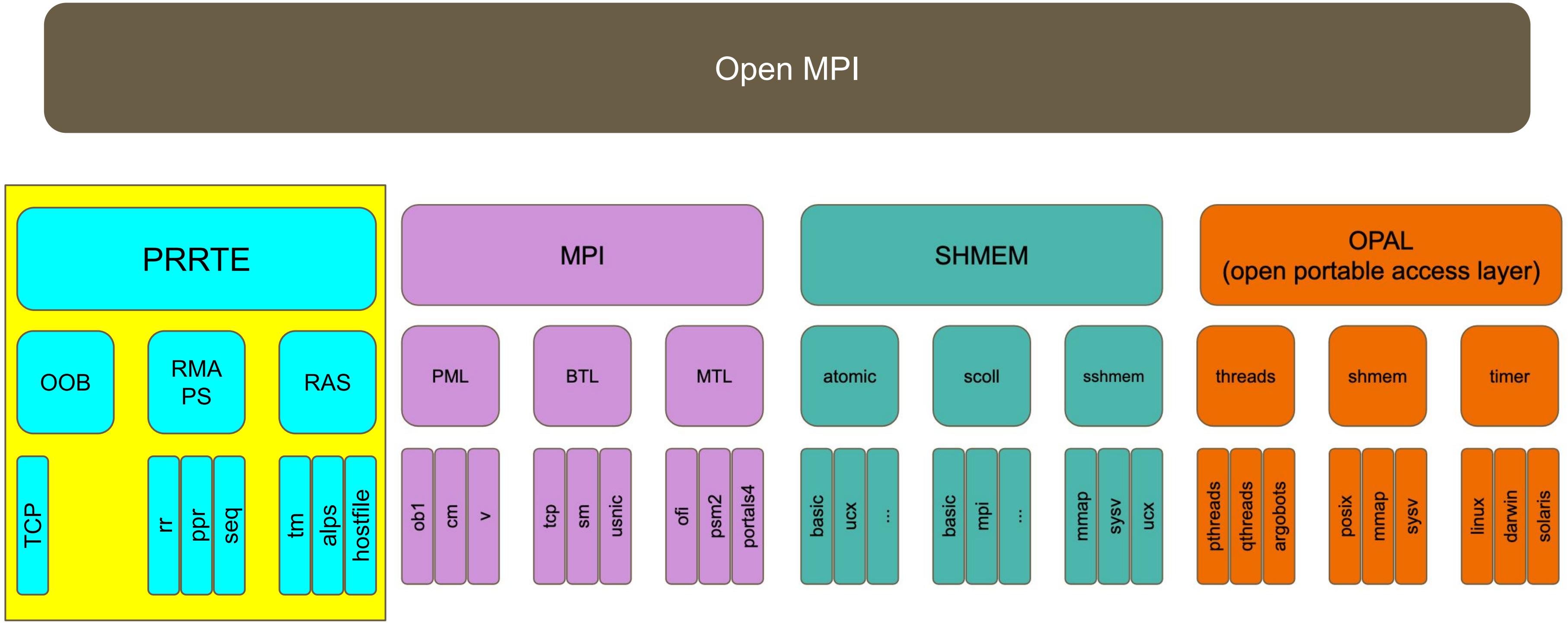
Isolation from limitations of host environment

Full PMIx support

Where Is It Used?

- As a shim in non-full-featured environments
    - Cray/ALPS - adds support for dynamic operations
    - Slurm - extends range of PMIx support beyond wireup
- User-level development environment
    - Tool, apps - develop PMIx-based code
    - Each user gets isolated environment
- Support for workflow managers
    - Full dynamic operations
    - Multi-app/tenant
    - Fast launch as daemons persist across apps
- Base runtime for Open MPI
    - Replacing ORTE in v5.0

Where Does It Fit?



PRRTE Architecture

- MCA Component Architecture
  - “Borrowed” from Open MPI
  - Same build system
- Dependencies
  - Required: libevent, HWLOC, PMIx (3.1+)
  - Optional: torque/pbs, ALPS, LSF, Gridengine
  - Autodetected: Slurm, Singularity, zlib
- Key frameworks
  - Rmaps - process placement
  - Oob - inter-daemon communication
  - Plm - daemon and application launch
  - State - proc/job state machine

No Embedded Libraries!

https://openpmix.github.io/code/getting-the-pmix-reference-server

Adaptive Command Line (schizo Framework)

- Supports multiple libraries
  - OMPI, various OSHMEM flavors, MPICH
  - Command line options fully configurable
  - Detect and utilize based on absolute path of argv[0]
- Runtime selection of "personality"
  - Obtain absolute path of argv[0]
    - Use PATH if no path information provided
  - Search PRRTE install `<prefix>/etc` for configuration files
    - `ompi.ini`, `oshmem-stb.ini`, `mpich.ini`
    - Compare absolute argv[0] to entries
  - Example ompi.ini
    - `/opt/local/openmpi/v5.0.0/bin/mpirun`
    - `/opt/local/openmpi/v5.0.0/bin/mpiexec`
    - `/opt/local/openmpi/v5.0.0/bin/oshrun`

→ Use OMPI schizo component for cmd line processing

MCA Parameter Usage

- Major difference from Open MPI in how these are handled
    - But you will see it in OMPI v5 as mpirun ⇒ prte
- Some apply only to start of DVM (e.g., oob, rml, routed, state)
    - Cannot be changed without restarting DVM
- Others only set default behavior (e.g., rmaps, hwloc)
    - Per-job behavior controlled via cmd line option
    - MCA param on cmd line is ignored for these values!
- Many ORTE params have been removed
    - Only applied to per-job behaviors
    - No longer supported

MCA Parameters

- Standardized way for querying / setting run-time parameters
- Multiple ways to set MCA parameters:

Command line:	`mpirun --mca <PARAM> <VALUE> …`
Environment variable:	`OMPI_MCA_<PARAM>=<VALUE>`
User-level file:	`$HOME/.openmpi/mca-params.conf`
System-level file:	`$prefix/etc/openmpi-mca-params.conf`

- Similar pattern for PMIx…

Environment variable:	`PMIX_MCA_<PARAM>=<VALUE>`
User-level file:	`$HOME/.pmix/mca-params.conf`
System-level file:	`$prefix/etc/pmix-mca-params.conf`

- …and for PRRTE

Environment variable:	`PRTE_MCA_<PARAM>=<VALUE>`
User-level file:	`$HOME/.prte/mca-params.conf`
System-level file:	`$prefix/etc/prte-mca-params.conf`

Some Differences For PRRTE

- PRRTE command lines
  - Uses “`prte`” to start the DVM
  - Uses “`prun`” to launch jobs
  - Open MPI params: prefix with “`--omca`” instead of “`--mca`”
  - PMIx params: prefix with “`--pmixmca`”
  - PRRTE params: prefix with “`--prtemca`”
  - Generic “`--mca`” picks best match based on framework and param name
    - Checks against list of known frameworks by project
- Envars and param files remain the same
  - PMIx will pickup and automatically forward the system and user default param values for Open MPI and PMIx
  - PRRTE will do the same for its default values

Build Tips

- No public APIs!
  - Applications never link against PRRTE
  - No need to worry about mix/match of PMIx, libevent, HWLOC with apps being launched
- Ensure symlinks setup to mpirun, mpiexec, etc.
  - All need to point to "`prte`" executable
  - Appropriate symlinks for each supported library
- Ensure `.ini` files created
  - Open MPI will automatically install its `ompi.ini` for embedded PRRTE
  - Need to manually create all others
- Setup any system-level default params
  - `<prefix>/etc/prte-mca-params.conf`

PRRTE vs PRTE

- “PRRTE” is the *project* name
  - Historical acronym
  - Package and libraries use it
- “PRTE” is the *operational* name
  - Avoids the “stutter” problem when typing
  - Covers all tool, MCA parameter names

PRRTE Tools

- `pcc` : wrapper compiler
  - Ensures build against same PMIx, libevent, HWLOC as PRRTE
  - Convenience only - not required (apps do not link against PRRTE)
- `prte` : start DVM
- `prte_info` : reports build information (ala `ompi_info`)
- `prted` : PRRTE daemon for remote nodes
- `prun` : PRRTE launcher
  - Used to start applications
- `pterm` : stop DVM

Debugging Tips

- Simulate large clusters on small allocation
  - Set "`--prtemca routed_radix 1`" to create linear chain of daemons
    - Tests scalable communication
  - Can use "`--prtemca ras_base_multiplier N`" to launch multiple daemons/node
    - Cannot run MPI jobs this way, useful for testing runtime scalability
- Use PMIx tools
  - Useful system info for help with diagnosis
- Verbosity is your friend (PRRTE MCA params)
  - Starting points: `plm_base_verbose`, `state_base_verbose` => set to 5
    - See `prted` cmd line, error output from remote daemons, state machine progress
  - Next: `oob_base_verbose`, `errmgr_base_verbose` => set to 5
  - If daemons are starting but procs aren't working:
    - `pmix_server_verbose` => set to 5

Ralph's Concluding Remarks

- Thank you…
  - for your attention!
  - EasyBuild (and especially Kenneth Hoste) for your hospitality!
- PMIx vs. OpenPMIx vs. PRRTE - a reminder
  - PMIx is the Standard (i.e., a document)
  - OpenPMIx is the library (i.e., a reference implementation of the PMIx Standard)
    - Someone, someday *might* implement their own version…but nothing so far
  - PRRTE is a full-featured PMIx environment
- If PMIx has things that interest you…
  - Include PMIx (at desired feature level) in your RFPs
  - Push your vendor to integrate with OpenPMIx for the desired feature level
  - Meantime, consider using PRRTE as a shim

Overall Open MPI Configuration / Debugging Tips

Start with the basics

1. Start trying to run a simple, non-MPI program locally
   This tests the basic Open MPI runtime system (without the MPI layer)
   ```
   mpirun -np 1 hostname
   ```

2. Then run MPI “hello world” locally (in Open MPI `examples` directory)
   This actually starts up / shuts down the MPI layer
   ```
   mpirun -np 1 hello_c
   ```

3. Then run MPI “ring” locally (in the Open MPI `examples` directory)
   This actually uses MPI communications (must have >=2 processes)
   ```
   mpirun -np 2 ring_c
   ```

Then add complexity

4. Then try to run a remote simple non-MPI program

   ```
   mpirun -np 3 --host host1:1,host2:1,host3:1 hostname
   ```

   (or run through your batch scheduler)

5. Then do the same with hello world

   ```
   mpirun -np 3 --host host1:1,host2:1,host3:1 hello_c
   ```

6. Then do the same with ring

   ```
   mpirun -np 3 --host host1:1,host2:1,host3:1 ring_c
   ```

Standard troubleshooting (things we hear often)

1. Check your PATH and LD_LIBRARY_PATH
   - Both locally and remote (for non-interactive logins and/or batch scripts)
   - Make sure they are both pointing to where you think they are pointing to
   - Bonus points: run "`ldd my_mpi_program`" to check where the linker will find libmpi

2. MacOS has a very lengthy default temporary directory (in $TMPDIR)
   - This can cause problems with shared memory files in Open MPI (they can exceed the max filename size)
   - Suggestion (on MacOS):

     ```
     mkdir $HOME/tmp
     export TMPDIR=$HOME/tmp
     ```

Standard troubleshooting (things we hear often)

3. Isolate the real error message
    a. Open MPI (and/or PMIx) is sometimes blamed for app issues
    b. E.g., one application process crashes, which subsequently triggers MPI and/or PMIx error messages
    c. Scroll back up and make sure you find the initial error

How can I tell which (MPI) network I'm using?

Refer back to part 2 of this seminar:

- Force the use of a given network via the PML, MTL, and BTL MCA params
- Use in conjunction with MPI benchmarks to see performance differences

- ...also, stay tuned for v5.0.0 😊

Not-uncommon question (mostly from ISVs)

"I want to re-locate an Open MPI installation"

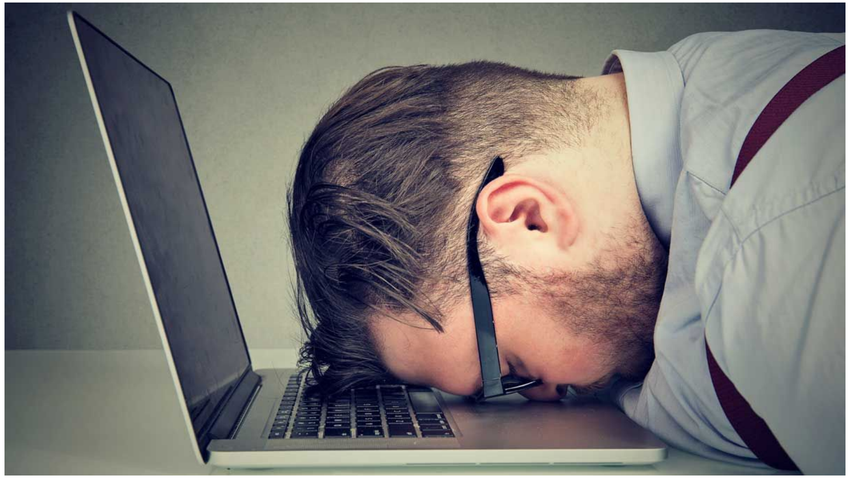
You must set the following three environment variables:

1. OPAL_PREFIX
2. PMIX_PREFIX
3. PRTE_PREFIX

More subtle issue: duplicated libraries

- If multiple -- potentially [slightly] different -- copies of a shared library are loaded in to an MPI process space, Bad Things can (will) happen
  - Symptoms can be Random Bad Things happening (segv, etc.)

- Check for implicit linker loads of the following (especially if the application itself is using these libraries):
  - Hwloc
  - Libevent
  - PMIx



How to get help

General help page: https://www.open-mpi.org/community/help/

- Supply as much detail as possible
  - Do not assume that we know what you know
  - Describe your environment
  - Describe what your program is supposed to do
  - Describe what your program is actually doing

- Provide a (small!) reproducer program

- Check the baseline MPI performance in your environment
  - Run the Ohio State MPI micro benchmarks

The Open MPI v4.1.x Series (upcoming)

v4.1.0: Expected ~August 2020

- General performance improvements
- Libfabric / OFI improvements
    - Support multi-device environments
    - One-sided performance improvements
- OMPIO improvements
    - Support IME, GPFS

- Backwards compatible with the v4.0.x series
    - Including ABI compatibility

v4.1.0: Collective performance improvements

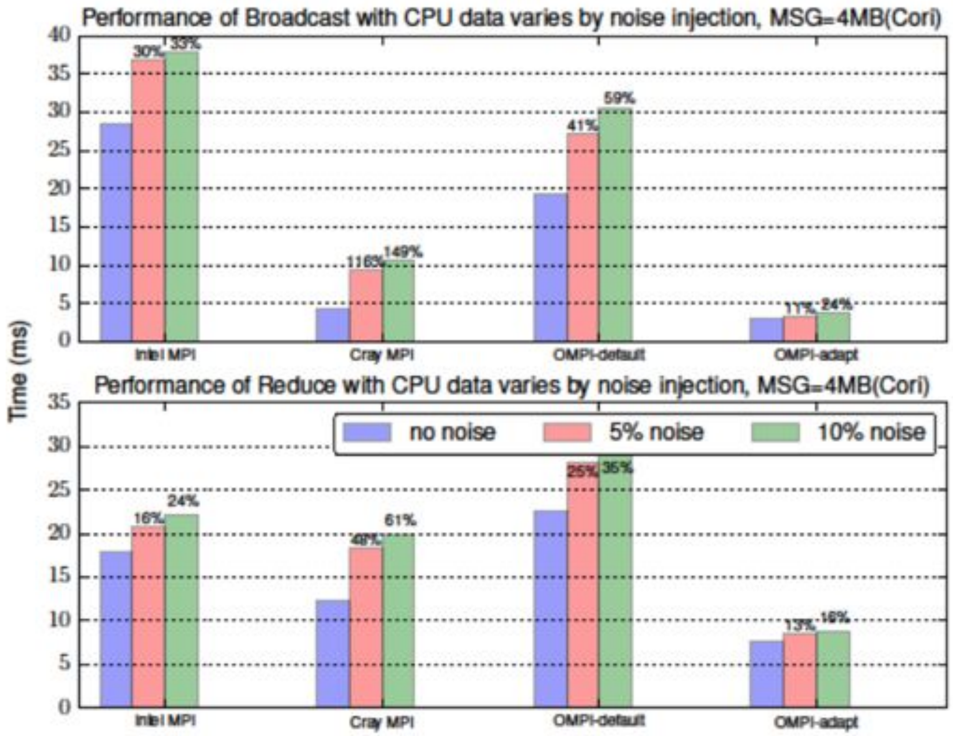
Two levels of improvements:

1. General algorithm tuning selection improvements
2. New ADAPT and HAN collective modules
    a. All-new code base from research at U. Tennessee
    b. These modules do not activate by default -- must be manually enabled
    c. Shows significant performance improvements compared to prior generation
    d. We need real-world testing!

v4.1.0: ADAPT collectives

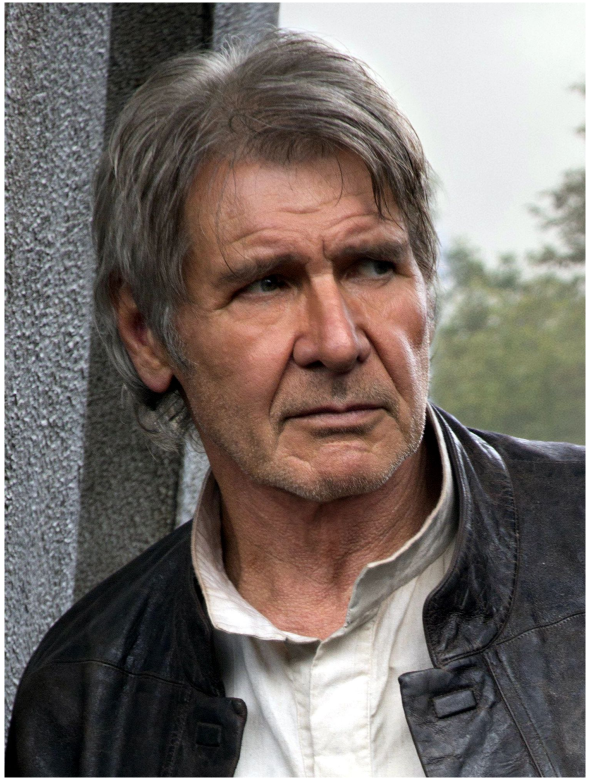
- Main idea: tolerate scheduling noise
  - Processes that are de-scheduled
  - Processes that are “late”
- Relaxes unnecessary synchronizations

- Performance on Cori (US LBNL), 1K cores
  - Every 0.1s, each process randomly inject 0-10ms noise (average 5%) and 0-20ms noise (average 10%)
  - Top: MPI_BCAST, bottom: MPI_REDUCE



v4.1.0: HAN collectives

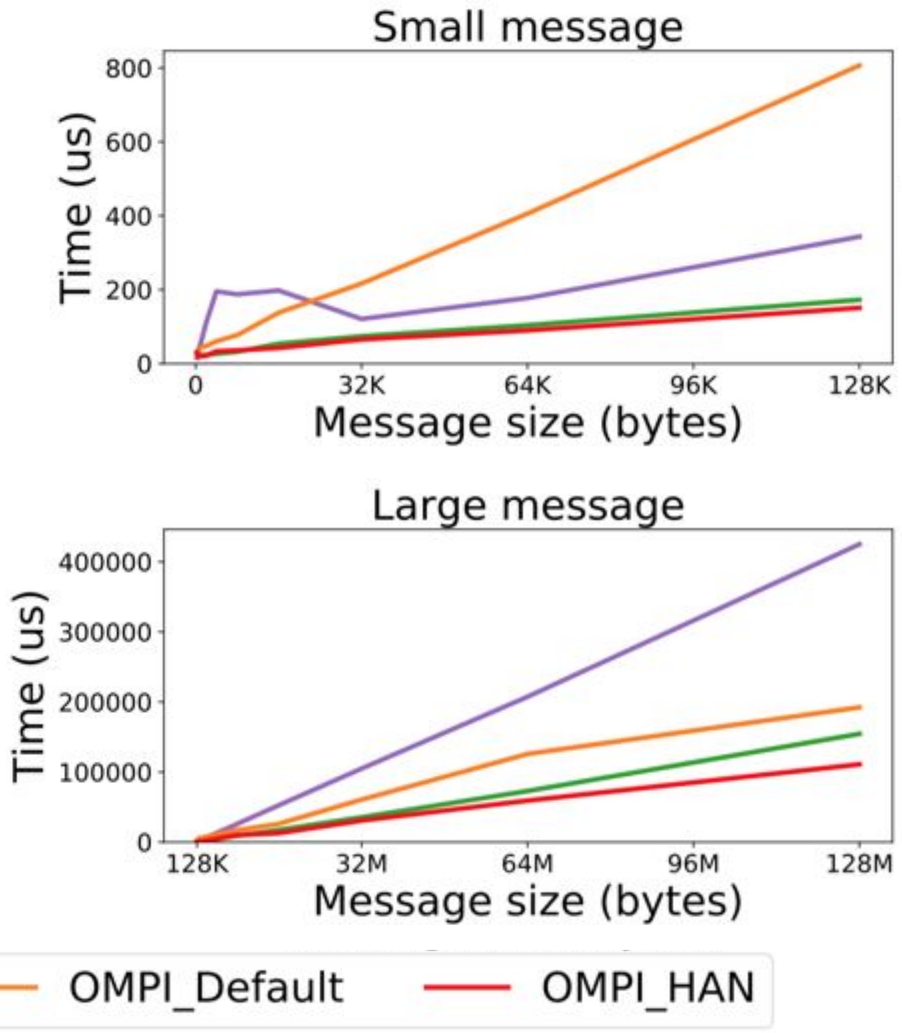
- Hierarchical-Aware Networking (HAN)
- Support two-level hierarchies:
  - Intra-node
  - Inter-node
- Reshape the collective to minimize the amount of data transferred over the slowest link
  - Selects / orchestrates the base set of collective algorithms in Open MPI



v4.1.0: HAN collectives

- Hierarchical-Aware Networking (HAN)
- Support two-level hierarchies:
  - Intra-node
  - Inter-node
- Reshape the collective to minimize the amount of data transferred over the slowest link
  - Selects / orchestrates the base set of collective algorithms in Open MPI

- Performance on Stampede 2
  - MPI_BCAST on 1,536 processes (24ppn)



How do I enable ADAPT and/or HAN?

- Either of two different ways:
  a. Set the MCA priority of `adapt` and/or `han` to 100. For example:
     ```
     $ mpirun --mca coll_adapt_priority 100 --mca coll_han_priority 100 …
     ```
  b. Include `adapt` and/or `han` in the `coll` MCA parameter. For example:
     ```
     $ mpirun --mca coll han,adapt,tuned,sm,basic …
     ```

- Do I have to enable *both* ADAPT and HAN?
  a. No.
  b. Specifically: you can use them idependently <u>*or*</u> together.

PLEASE TEST WITH REAL APPS!

The Open MPI v5.0.x Series (upcoming)

v5.0.0: Expected ~2020

- Originally expected 1H2020
  - Has been delayed (COVID, development complications, etc)
  - Hopefully will still release in 2020… (!)
  - Some backward compatible pieces moved to v4.1
- Many, many minor improvements

- Breaks backwards compatibility with v4.x series
  - ABI, `mpirun` command line arguments, etc.
- Requires new debuggers and tools!
  - MPIR no longer supported
  - TotalView, DDT releasing updated support
  - Shim to ease transition from MPIR
    - https://github.com/openpmix/mpir-to-pmix-guide

v5.0.0: Expected ~2020

Some ***Big Changes*** are coming in the runtime

- No support for PMI-1, PMI-2 from Slurm and Cray
  - Only PMIx is supported
- ORTE replaced by PRRTE
  - No longer support (most) `mpirun` single-dash, multi-char options (small exceptions)
  - Adaptive command line requires additional setup
  - Different syntax for MCA parameters on command line
- PMIx (v4.0.x) as first-class citizen
  - Can configure/build just MPI layer with no runtime for direct-launch only environments
  - PMIx symbols exposed for use by application
  - All non-standardized Open MPI-defined MPI info key names replaced by PMIx attributes!
    - PMIx equivalents available for standardized keys

v5.0.0: more features

- Support for User-Level Fault Mitigation (ULFM)
- Support for AVX instructions in MPI_Op operations
- Support for user-level threads packages
  - Qthreads
  - Argobots
- ADAPT and HAN will be the default for MPI collectives (hopefully!)
- Openib BTL fully replaced by UCX PML
- `vader` BTL was renamed to `sm` (but still has a `vader` alias)
- At least some elements of MPI-4
  - Sidenote: MPI-4.0 document due by end of 2020
  - Still working on which specific MPI-4 features will be included

v5.0.0: Connectivity map

- A long-asked-for feature: show which networks are used at run time
- Not a perfect system (e.g., it does not show inside Libfabric or UCX)

```
$ mpirun -np 4 --host mpi002:2,mpi004:2 --mca hook_comm_method_enable_mpi_init 1 hello_c
Host 0 [mpi002] ranks 0 - 1
Host 1 [mpi004] ranks 2 - 3

 host | 0     1
======|=============
    0 : sm    usnic
    1 : usnic sm

Connection summary: (btl)
  on-host:  all connections are sm
  off-host: all connections are usnic
...
```

Subject to change before v5.0.0

Questions?

easybuild

Presented in conjunction
with the EasyBuild community

Thank you!