MPICH vs OpenMPI

Asked 13 years, 9 months ago Modified 5 years, 4 months ago Viewed 106k times

162

Can someone elaborate the differences between the OpenMPI and MPICH implementations of MPI ? Which of the two is a better implementation ?

mpi hpc openmpi

Share Follow

edited Mar 13, 2010 at 19:05

asked Mar 11, 2010 at 17:58

lava
1,965 2 14 15

2 See this: stackoverflow.com/questions/144309/... – Taylor Leese Mar 11, 2010 at 18:03

2 We personally chose OpenMPI as our MPI implementation. For us, it benchmarked better and portability wasn't as much of an issue. See the question link Taylor L posted. – Xorlev Mar 11, 2010 at 21:34

1 you may also consider that in Google trends OpenMPI is 2-3 times more searched than MPICH/MPICH2. – Foad S. Farimani Jan 28, 2018 at 14:19

I think MPICH is no longer supported in recent versions of Linux (e.g., Ubuntu 18 can't run it), IIRC it only works in certain kernel versions – jrh May 5, 2020 at 14:06

@jrh mpich can easily be compiled from source. – Victor Eijkhout Aug 20, 2021 at 12:47

5 Answers

Sorted by: Highest score (default)

195

Purpose

First, it is important to recognize how MPICH and Open-MPI are different, i.e. that they are designed to meet different needs. MPICH is supposed to be high-quality reference implementation of the latest MPI standard and the basis for derivative implementations to meet special purpose needs. Open-MPI targets the common case, both in terms of usage and network conduits.

Support for Network Technology

Open-MPI documents their network support here. MPICH lists this information in the README distributed with each version (e.g. this is for 3.2.1). Note that because both Open-MPI and MPICH support the OFI (aka libfabric) networking layer, they support many of the same networks. However, libfabric is a multi-faceted API, so not every network may be supported the same in both (e.g. MPICH has an OFI-based IBM Blue Gene/Q implementation, but I'm not aware of equivalent support in Open-MPI). However, the OFI-based implementations of both MPICH and Open-MPI are working on shared-memory, Ethernet (via TCP/IP), Mellanox InfiniBand, Intel Omni Path, and likely other networks. Open-MPI also supports both of these networks and others natively (i.e. without OFI in the middle).

In the past, a common complaint about MPICH is that it does not support InfiniBand, whereas Open-MPI does. However, MVAPICH and Intel MPI (among others) - both of which are MPICH derivatives - support InfiniBand, so if one is willing to define MPICH as "MPICH and its derivatives", then MPICH has extremely broad network support, including both InfiniBand and proprietary interconnects like Cray Seastar, Gemini and Aries as well as IBM Blue Gene (/L, /P and /Q). Open-MPI also supports the Cray Gemini interconnect, but its usage is not supported by Cray. More recently, MPICH supported InfiniBand through a netmod (now deprecated), but MVAPICH2 has extensive optimizations that make it the preferred implementation in nearly all cases.

Feature Support from the Latest MPI Standard

An orthogonal axis to hardware/platform support is coverage of the MPI standard. Here MPICH is usually far and away superior. MPICH has been the first implementation of every single release of the MPI standard, from MPI-1 to MPI-3. Open-MPI has only recently supported MPI-3 and I find that some MPI-3 features are buggy on some platforms (MPICH is not bug-free, of course, but bugs in MPI-3 features have been far less common).

Historically, Open-MPI has not had holistic support for `MPI_THREAD_MULTIPLE`, which is critical for some applications. It might be supported on some platforms but cannot generally be assumed to work. On the other hand, MPICH has had holistic support for `MPI_THREAD_MULTIPLE` for

many years, although the implementation is not always high-performance (see "Locking Aspects in Multithreaded MPI Implementations" for one analysis).

Another feature that was broken in Open-MPI 1.x was one-sided communication, aka RMA. This has more recently been fixed and I find, as a very heavy user of these features, that they are generally working well in Open-MPI 3.x (see e.g. the ARMCI-MPI test matrix in Travis CI for results showing RMA working with both implementations, at least in shared-memory. I've seen similar positive results on Intel Omni Path, but have not tested Mellanox InfiniBand.

Process Management

One area where Open-MPI used to be significantly superior was the process manager. The old MPICH launch (MPD) was brittle and hard to use. Fortunately, it has been deprecated for many years (see the MPICH FAQ entry for details). Thus, criticism of MPICH because of MPD is spurious.

The Hydra process manager is quite good and has the similar usability and feature set as ORTE (in Open-MPI), e.g. both support HWLOC for control over process topology. There are reports of Open-MPI process launching being faster than MPICH-derivatives for larger jobs (1000+ processes), but since I don't have firsthand experience here, I am not comfortable stating any conclusions. Such performance issues are usually network-specific and sometimes even machine-specific.

I have found Open-MPI to be more robust when using MacOS with a VPN, i.e. MPICH may hang in startup due to hostname resolution issues. As this is a bug, this issue may disappear in the future.

Binary Portability

While both MPICH and Open-MPI are open-source software that can be compiled on a wide range of platforms, the portability of MPI libraries in binary form, or programs linked against them, is often important.

MPICH and many of its derivatives support ABI compatibility (website), which means that the binary interface to the library is constant and therefore one can compile with `mpi.h` from one implementation and then run with another. This is true even across multiple versions of the libraries. For example, I frequently compile Intel MPI but `LD_PRELOAD` a development version of MPICH at runtime. One of the big advantages of ABI compatibility is that ISVs (Independent Software Vendors) can release binaries compiled against only one member of the MPICH family.

ABI is not the only type of binary compatibility. The scenarios described above assume that users employ the same version of the MPI launcher (usually `mpirun` or `mpiexec`, along with its compute-node daemons) and MPI library everywhere. This is not necessarily the case for containers.

While Open-MPI does not promise ABI compatibility, they have invested heavily in supporting containers (docs, slides). This requires great care in maintaining compatibility across different versions of the MPI launcher, launcher daemons, and MPI Library, because a user may launch jobs using a newer version of the MPI launcher than the launcher daemons in the container support. Without careful attention to launcher interface stability, container jobs will not launch unless the versions of each component of the launcher are compatible. This is not an insurmountable problem:

> The workaround used by the Docker world, for example, is to containerize the infrastructure along with the application. In other words, you include the MPI daemon in the container with the application itself, and then require that all containers (mpiexec included) be of the same version. This avoids the issue as you no longer have cross-version infrastructure operations.

I acknowledge Ralph Castain of the Open-MPI team for explaining the container issues to me. The immediately preceding quote is his.

Platform-Specific Comparison

Here is my evaluation on a platform-by-platform basis:

- Mac OS: both Open-MPI and MPICH should work just fine. To get the latest features of the MPI-3 standard, you need to use a recent version of Open-MPI, which is available from Homebrew. There is no reason to think about MPI performance if you're running on a Mac laptop.
- Linux with shared-memory: both Open-MPI and MPICH should work just fine. If you want a release version that supports all of MPI-3 or MPI_THREAD_MULTIPLE, you probably need MPICH though, unless you build Open-MPI yourself, because e.g. Ubuntu 16.04 only provides the ancient version 1.10 via APT. I am not aware of any significant performance differences between the two implementations. Both support single-copy optimizations if the OS allows them.
- Linux with Mellanox InfiniBand: use Open-MPI or MVAPICH2. If you want a release version that supports all of MPI-3 or `MPI_THREAD_MULTIPLE`, you likely need MVAPICH2 though. I find that MVAPICH2 performs very well but haven't done a direct comparison with OpenMPI on InfiniBand, in part because the features for which performance matters most to me (RMA aka one-sided) have been broken in Open-MPI in the past.

- Linux with Intel Omni Path (or its predecessor, True Scale): I have use MVAPICH2, Intel MPI, MPICH and Open-MPI on such systems, and all are working. Intel MPI tends to the most optimized while Open-MPI delivered the best performance of the open-source implementations because they have a well-optimized PSM2-based back-end. I have some notes on GitHub on how to build different open-source implementations, but such information goes stale rather quickly.
- Cray or IBM supercomputers: MPI comes installed on these machines automatically and it is based upon MPICH in both cases. There have been demonstrations of MPICH on Cray XC40 (here) using OFI, Intel MPI on Cray XC40 (here) using OFI, MPICH on Blue Gene/Q using OFI (here), and Open-MPI on Cray XC40 using both OFI and uGNI (here), but none of these are vendor supported.
- Windows: I see no point in running MPI on Windows except through a Linux VM, but both Microsoft MPI and Intel MPI support Windows and are MPICH-based. I have heard reports of successful builds of MPICH or Open-MPI using Windows Subsystem for Linux but have no personal experience.

Notes

In full disclosure, I currently work for Intel in a research/pathfinding capacity (i.e. I do not work on any Intel software products) and formerly worked for Argonne National Lab for five years, where I collaborated extensively with the MPICH team.

Share Follow

edited Aug 16, 2018 at 17:49

answered Aug 25, 2014 at 19:46

Jeff Hammond
5,434 3 28 47

It is possible that OpenMPI has superior support for shared-memory in collectives, but I need to investigate thoroughly before updating my answer. – Jeff Hammond Oct 6, 2014 at 8:50

2 Can you elaborate why you see no point in running MPI on Windows? – Dmitri Nesteruk Feb 20, 2015 at 8:54

3 No, but feel to ask a new question on StackOverflow about HPC on Windows. – Jeff Hammond Feb 22, 2015 at 2:38

@Jeff, you highlighted the `MPI_THREAD_MULTIPLE` in the answer, but I don't have real experience to use it before. Could you give some user cases/ examples where the `MPI_THREAD_MULTIPLE` is useful and efficient compared with other modes such as `MPI THREAD FUNNELED` ? My first impression is this function make the program more complex and hard to debug between thread and process. Thanks. – Patric Jan 14, 2016 at 3:54

1 Just a note that Open-MPI now compiles and runs fine on the Windows Subsytem for Linux - I would guess mpich too. – jawknee Dec 21, 2017 at 13:33

18

If you do development rather than production system, go with MPICH. MPICH has built-in debugger, while Open-MPI does not last time I checked.

In production, Open-MPI most likely will be faster. But then you may want to research other alternatives, such as Intel MPI.

Share Follow

edited Sep 1, 2016 at 18:37

Jeff Hammond
5,434 3 28 47

answered Mar 11, 2010 at 21:47

Anycorn
50.5k 43 169 263

5 I'm not sure what you mean by built-in debugger, but I find that Open-MPI has good integration with e.g. gdb: open-mpi.org/faq/?category=debugging. – Jeff Hammond Sep 1, 2016 at 18:36

For production, are there any thoughts on using MPICH with TAO? – namu Mar 8, 2017 at 21:51

What is the built-in debugger? How do I use it? – Nanashi No Gombe Sep 11, 2020 at 9:10

12

I concur with the previous poster. Try both to see which one your application runs faster on then use it for production. They are both standards compliant. If it is your desktop either is fine. OpenMPI comes out of the box on Macbooks, and MPICH seems to be more Linux/Valgrind friendly. It is between you and your toolchain.

If it is a production cluster you need to do more extensive benchmarking to make sure it is optimized to your network topology. Configuring it on a production cluster will be the main difference in terms of your time as you will have to RTFM.

Share Follow

answered Mar 18, 2010 at 15:29

Chad Brewbaker
2,551 2 19 26

16 If everyone RTFMed, we wouldn't need StackOverflow :-) – Jeff Hammond Oct 24, 2014 at 18:08

1 FWIW, Open-MPI has an FAQ entry on Valgrind-cleanliness: open-mpi.org/faq/?category=debugging#valgrind_clean – Jeff Hammond Sep 1, 2016 at 18:38

@Jeff Um what about bugs? Out of date docs? That's behind a plurality of my (hundreds of ..) questions here ;) – WestCoastProjects Nov 26, 2016 at 1:32

1 @JeffHammond if only someone had always already WTFM, that would definitely be the solution! – William Gallafent Dec 13, 2020 at 12:00

9

Both are standards-compliant, so it shouldn't matter which you use from a correctness point of view. Unless there is some feature, such as specific debug extensions, that you need, then benchmark both and pick whichever is faster for your apps on your hardware. Also consider that there are other MPI implementations that might give better performance or compatibility, such as MVAPICH (can have the best InfiniBand performance) or Intel MPI (widely supported ISVs). HP worked hard to get their MPI qualified with lots of ISV codes too, but I'm not sure how it is faring after being sold on to Platform...

Share Follow

edited Sep 1, 2016 at 18:38

Jeff Hammond
5,434 3 28 47

answered Mar 18, 2010 at 16:07

Funcan
151 2

2

From my experience one good feature that OpenMPI supports but MPICH does not is **process affinity**. For example, in OpenMPI, using `-npersocket` you can set the number of ranks launched on each socket. Also, OpenMPI's `rankfile` is quite handy when you want to pinpoint ranks to cores or oversubscribe them.

Last, if you need to control the mapping of ranks to cores, I would definitely suggest writing and compiling your code using OpenMPI.

Share Follow

answered Jun 8, 2018 at 18:10

hmofrad
1,824 2 24 28

2 MPICH supports affinity. wiki.mpich.org/mpich/index.php/... – Jeff Hammond May 26, 2019 at 20:23