Run MPI with containers

Simple test

Simple container with ucx and openmpi.

```
Bootstrap: docker
From: debian:trixie

%environment
    export LD_LIBRARY_PATH=/usr/local/lib

%post
 apt-get update && apt-get install -y build-essential wget rdma-core libibverbs-dev
 wget https://github.com/openucx/ucx/releases/download/v1.18.1/ucx-1.18.1.tar.gz
 tar xzf ucx-1.18.1.tar.gz
 cd ucx-1.18.1
 mkdir build
 cd build
 ../configure --prefix=/opt/
 make -j4
 make install
 cd ..
 export OPENMPI_VERSION="4.1.6"
 export OPENMPI_MAJOR_VERSION="v4.1"
 export OPENMPI_MAKE_OPTIONS="-j4"
 mkdir -p /openmpi-src
 cd /openmpi-src
 wget https://download.open-mpi.org/release/open-mpi/${OPENMPI_MAJOR_VERSION}/openmpi-
${OPENMPI_VERSION}.tar.gz \
       && tar xfz openmpi-${OPENMPI_VERSION}.tar.gz
 cd openmpi-${OPENMPI_VERSION} && ./configure --with-ucx=/opt --without-verbs
 make all ${OPENMPI_MAKE_OPTIONS}
 make install
```

```
cd /
rm -rf /openmpi-src
```

To build it:

```
singularity build -f openmpitest.sif openmpi.def
```

Then we compile an MPI application inside the container. For example osu-benchmarks.

```
wget https://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-7.5-1.tar.gz
tar -xvf osu-micro-benchmarks-7.5-1.tar.gz
```

```
singularity shell openmpitest.sif
```

```
cd osu-micro-benchmarks-7.5-1
./configure CC=/usr/local/bin/mpicc CXX=/usr/local/bin/mpicxx --
prefix=/scratch/$USER/osu_install
make install
```

Then you can use the following job:

```
#!/bin/bash

#SBATCH -N 2
#SBATCH -n 2
#SBATCH -o mpi-%j.out
#SBATCH -e mpi-%j.err

module purge
module load singularityce
module load openmpi
export PMIX_MCA_psec=native
export PMIX_MCA_gds=^ds12

export SINGULARITY_BINDPATH=/scratch

srun --mpi=pmix singularity run openmpitest.sif /scratch/$user/osu_install/libexec/osu-micro-
benchmarks/mpi/collective/osu_alltoall
```

Some possible errors

if the option `--mpi=mpix` is not used, you will have the following error:

```
[dna067:2560172] OPAL ERROR: Unreachable in file pmix3x_client.c at line 111
--------------------------------------------------------------------------
The application appears to have been direct launched using "srun",
but OMPI was not built with SLURM's PMI support and therefore cannot
execute. There are several options for building PMI support under
SLURM, depending upon the SLURM version you are using:

  version 16.05 or later: you can use SLURM's PMIx support. This
  requires that you configure and build SLURM --with-pmix.

  Versions earlier than 16.05: you must use either SLURM's PMI-1 or
  PMI-2 support. SLURM builds PMI-1 by default, or you can manually
  install PMI-2. You must then build Open MPI using --with-pmi pointing
  to the SLURM PMI library location.

Please configure as appropriate and try again.
```

By default OpenMPI 4.x will try to use PMIx client v3. The intialisation does not success bacuase there is no PMIx server initialized. mpirun takes care of initializing an embedded PMIx server.

Psec error

You can also have this error:

```
A requested component was not found, or was unable to be opened.  This
means that this component is either not installed or is unable to be
used on your system (e.g., sometimes this means that shared libraries
that the component requires are unable to be found/loaded).  Note that
PMIX stopped checking at the first component that it did not find.

Host:      dna075
Framework: psec
Component: munge
```

Here, the application will run. This is related to the PMIX_SECURITY_MODE. When `srun` is executed, it will setup previous variable to: `munge,native`. To verify it:

```
srun --mpi=pmix env | grep PMIX_SECURITY
PMIX_SECURITY_MODE=munge,native
```

Which means that munge protocol will be used for authentication. As the PMIx library on the container (client side) does not have that component, it will failed but it will then use the `native` component. You can read here for more explanations. You have to use `export PMIX_MCA_psec=native` to avoid this message.

gds error

You can also see this error:

```
[dna075:373342] PMIX ERROR: ERROR in file gds_ds12_lock_pthread.c at line 168
```

This is an OpenPMIx bug related to the 'Generalized DataStore for storing job-level and other data' component. You can blacklist it by setting: `export PMIX_MCA_gds=^ds12`.

> This is fixed in OpenMPI 5

Running OpenMPI 5.0

This works, there is no any compatibilty problem with the host version. If you want to test, set the version of OpenMPI to `5.0.7`. Other versions have problems to compile.

Running a container from dockerhub

This section explains how to run a thirdparty container that you cannot edit and it is based on another MPI distribution.

Let's take for example the openfoam container, which is based on MPICH. To build the container:

```
singularity build openfoam.sif docker://quay.io/pawsey/openfoamlibrary:v2312-rocm5.4-gcc
```

This container provides: `mpich 3.4.3` and the `osu benchmarks`. The osu benchmarks will help us to measure the performance of the MPI library and compare it with the native MPI library installed on the cluster.

If we try to execute the `osu benchmarks` from the container, we get:

```
srun -n 2 singularity exec openfoam.sif osu_alltoall
This test requires at least two processes
This test requires at least two processes
srun: error: dna066: tasks 0-1: Exited with exit code 1
```

This means that the binary is not able to initialize the MPI layer and just one instance is launched. Let's try to launch it with `PMIx`:

```
srun --mpi=pmix -n 2 singularity exec openfoam.sif osu_alltoall
This test requires at least two processes
This test requires at least two processes
srun: error: dna066: tasks 0-1: Exited with exit code 1
```

We get the same error, probably because MPICH does not support PMIx protocol. Let's try with pmi2:

```
srun --mpi=pmi2 -n 2 singularity exec openfoam.sif osu_alltoall

# OSU MPI All-to-All Personalized Exchange Latency Test v7.3
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
1                       2.52
2
2.44

4                       2.56
8                       2.45
16                      2.45
32                      2.52
```

This works but it has some overhead compare to the MPI installed natively. The other problem is that in a multinode configuration the overhead is high:

```
#!/bin/bash

#SBATCH -N 2
#SBATCH -n 2

module load singularityce

srun --mpi=pmi2 -n 2 singularity exec openfoam.sif osu_alltoall
```

```
# OSU MPI All-to-All Personalized Exchange Latency Test v7.3
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
1                     48.89
2                     44.16
4                     44.73
8                     49.08
16                    50.07
32                    48.33
```

Using wi4mpi

Wi4mpi is a tool that allows us to translate calls between different MPI implementations. The idea here is to be able to use the OpenMPI installed natively on the cluster. The following job slurm is used:

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -n 2

module load singularityce
module load openmpi wi4mpi
export SINGULARITY_BINDPATH=/dcsrsoft
export WI4MPI_FROM=MPICH
export WI4MPI_TO=OMPI
export WI4MPI_RUN_MPI_C_LIB=${OPENMPI_ROOT}/lib/libmpi.so
export WI4MPI_RUN_MPI_F_LIB=${OPENMPI_ROOT}/lib/libmpi_mpifh.so
export
WI4MPI_RUN_MPIIO_C_LIB=${WI4MPI_RUN_MPI_C_LIB}
```

```
export WI4MPI_RUN_MPIIO_F_LIB=${WI4MPI_RUN_MPI_F_LIB}
export
SINGULARITYENV_LD_PRELOAD=${WI4MPI_ROOT}/libexec/wi4mpi/libwi4mpi_${WI4MPI_FROM}_${WI4MPI_TO}.
so:${WI4MPI_RUN_MPI_C_LIB}

srun --mpi=pmix -n 2 singularity exec openfoam.sif osu_alltoall
```

Result:

```
You are using Wi4MPI-3.6.4 with the mode preload From MPICH To OMPI

# OSU MPI All-to-All Personalized Exchange Latency Test v7.3
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
1                     0.77
2                     1.00
4                     0.81
8                     0.93
16                    0.98
```

First, you can notice that now it works with `PMIx` and that the peformance is much better than before. We have still some errors/warnings:

```
A requested component was not found, or was unable to be opened.  This
means that this component is either not installed or is unable to be
used on your system (e.g., sometimes this means that shared libraries
that the component requires are unable to be found/loaded).  Note that
PMIx stopped checking at the first component that it did not find.

Host:      dna065
Framework: psec
Component: munge
--------------------------------------------------------------------------
[1752250146.039806] [dna065:2110525:0]     ucp_context.c:1177 UCX  WARN  network device
'mlx5_2:1' is not available, please use one or more of: 'ens1f1'(tcp), 'ib0'(tcp), '
lo'(tcp)
[1752250146.049910] [dna065:2110525:0]     ucp_context.c:1177 UCX  WARN  network device
'mlx5_2:1' is not available, please use one or more of: 'ens1f1'(tcp), 'ib0'(tcp), '
lo'(tcp)
```

```
[1752250146.039795] [dna065:2110526:0]     ucp_context.c:1177 UCX  WARN  network device
'mlx5_2:1' is not available, please use one or more of: 'ens1f1'(tcp), 'ib0'(tcp), '
:
```

As we have seen before, the first error can be fixed using the following variable:

```
export PMIX_MCA_psec=native
```

The second error is related with the infiniband dectection. The UCX libray is linked to some libraries that are not available in the container. We can try to mount those libraries on the container:

```
export SINGULARITY_BINDPATH=/dcsrsoft,/lib64/libibverbs.so.1:/lib/x86_64-linux-
gnu/libibverbs.so.1,/lib64/libmlx5.so.1:/lib/x86_6
4-linux-gnu/libmlx5.so.1,/lib64/librdmacm.so.1:/lib/x86_64-linux-
gnu/librdmacm.so.1,/lib64/libnl-route-3.so.200:/lib/x86_64-linux-gnu/libnl-route-3.so.200
```

If we try again, we should see this:

```
[dna066:1443708] mca_base_component_repository_open: unable to open mca_pmix_s1: libpmi.so.0:
cannot open shared object file: No such file or directory (ignored)
[dna066:1443711] mca_base_component_repository_open: unable to open mca_pmix_s1: libpmi.so.0:
cannot open shared object file: No such file or directory (ignored)
[dna066:1443708] mca_base_component_repository_open: unable to open mca_pmix_s2: libpmi2.so.0:
cannot open shared object file: No such file or directory (ignored)
[dna066:1443711] mca_base_component_repository_open: unable to open mca_pmix_s2: libpmi2.so.0:
cannot open shared object file: No such file or directory (ignored)
You are using Wi4MPI-3.6.4 with the mode preload From MPICH To OMPI

# OSU MPI All-to-All Personalized Exchange Latency Test v7.3
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
1                       0.67
2                       0.67
4                       0.66
8                       0.68
16                      0.66
32                      0.74
64                      0.76
128                     1.05
256                     1.03
```

The peformance was improved. If we try multinode now:

```
# OSU MPI All-to-All Personalized Exchange Latency Test v7.3
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
1                       1.64
2                       2.15
4                       1.63
8                       1.61
16                      1.71
32                      1.70
64                      2.25
128                     2.14
```

We have around 40x of improvement.

Créé 2025-05-19 11:09:14 UTC par Cristian Ruiz
Mis à jour 2025-09-08 08:21:30 UTC par Cristian Ruiz