srun vs mpirun/mpiexec

2020

MPI startup mechanism – *not standardized!*

- The MPI standard does not define a standard startup mechanism. The startup command **`mpiexec -n <numprocs> <program>`** is suggested, but not mandatory.

- Each MPI implementation also has its own startup mechanism. The startup mechanism is linked to the MPI library.

- These startup commands may be called **mpirun**, **mpiexec** or something else. The commands from one MPI implementation cannot be used with the library from another implementation.

srun

- **`srun`** is the standard SLURM command to start an MPI program.
- It is well integrated with SLURM.
- It automatically uses the allocated job resources: nodelist, tasks, logical cores per task.
- It chooses an optimal CPU binding for the tasks on an allocated host.

What if I do not want to use srun?

- You already know mpirun/mpiexec and all the options of Intel MPI

mpirun *from Intel MPI*

- mpirun is a wrapper script that mimics srun and automatically copies settings from the SLURM batch job, if available.
- by default, mpirun takes affinity from SLURM
  - **`export SLURM_CPU_BIND=none`**
  - alternatively, use **`export I_MPI_PIN_RESPECT_CPUSET=no`** to override
- unset I_MPI_PMI_LIBRARY
- do NOT use #SBATCH --export=none, it causes confusing errors.
- Intel MPI 2019 can cause a floating point exception

mpirun *from Intel MPI (further hints)*

- do NOT use **`#SBATCH --export=none`**, it causes confusing errors.

- Intel MPI 2019 can cause a floating point exception
  - **`export I_MPI_HYDRA_TOPOLIB=ipl`** resolves this issue

Example mpirun from Intel MPI
4 nodes, 96 cores, no hyperthreading, 10 mins

```
#!/bin/bash
#SBATCH -t 00:10:00
#SBATCH -N 4
#SBATCH --tasks-per-node 96
#SBATCH -p standard96


module load impi
export SLURM_CPU_BIND=none   # important when using "mpirun" from Intel-MPI!
                             # Do NOT use this with srun!
export I_MPI_HYDRA_TOPOLIB=ipl
export I_MPI_HYDRA_BRANCH_COUNT=-1


mpirun hello_world > hello.output
```

mpiexec *from Intel MPI*

- Stand-alone starter, no connection to SLURM
- Requires either a
  - hostfile: list of hosts that can be used for a job
  - machinefile: control of process placement

- Affinity is set by Intel MPI
  - Affinity can be changed using the **`I_MPI_PIN_*`** environment variables

- mpiexec.hydra is an alias for mpiexec

Example mpiexec with hostfile (Intel MPI)

```
#!/bin/bash
#SBATCH -N 2
#SBATCH -p standard96

hostfile=$(mktemp)
scontrol show hostnames > $hostfile
mpiexec –hostfile $hostfile app
```

<host[:nranks]>

Example mpiexec with machinefile (Intel MPI) 2 nodes, 192 tasks

```
#!/bin/bash
#SBATCH -N 2
#SBATCH -p standard96

# Format is <host[:nranks]>
machinefile=$(mktemp)
scontrol show hostnames | awk ´{print $0 ":96"}' > $machinefile
mpiexec -n 192 -machinefile $machinefile app
```

Precise process distribution using a machinefile

- Example for a job with 7 ranks:

  node4:2
  node6:2
  node4
  node5

  Placement:

  ranks 0,1,4,6 on node4
  ranks 2,3 on node6
  ranks 5 on node5

„Received eager messages“

- This error message can occur when using mpirun/mpiexec with many processes on many nodes.
- This problem is related to a bug in libfabric and it will be fixed in a future version of Intel MPI.
- As a workaround for Intel MPI 2019, you can use:

```
export PSM2_CONNECT_TIMEOUT=60
```

Thanks for your attention

Atos