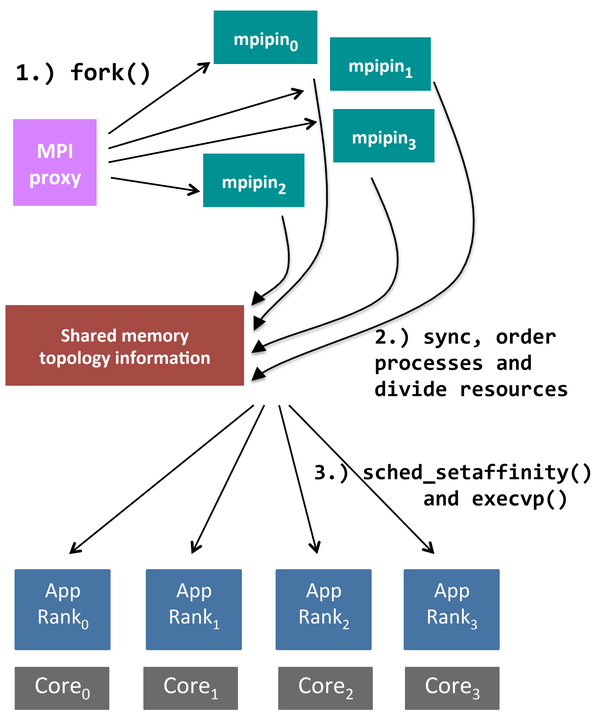
Making the Case for Portable MPI Process Pinning

Balazs Gerofi <bgerofi@riken.jp>, Rolf Riesen <rolf.riesen@intel.com> and
Yutaka Ishikawa <yutaka.ishikawa@riken.jp>

Motivation and Background

- Architectural complexity of node resources in high-performance computing (HPC) keeps increasing
- Many-core CPUs, non-uniform memory architectures (NUMA), deep memory hierarchies are becoming common-place (e.g., Intel Xeon Phi, AMD EPYC, etc.)
- Topology aware MPI process pinning is utmost important for efficiently utilizing the underlying hardware
- **Issues**:
  - Existing process pinning APIs are MPI implementation specific, *non-standard* and often *overly intricate*
  - *Non-portable job scripts* across different MPI environments
  - Difficulty of comparing MPI implementations due to pinning differences

Design and Implementation



- Processes synchronize
- Leader process elected
- Leader creates shared memory region and collects topology information
- Resources are divided
- Each rank pins itself to its corresponding partition (i.e., `sched_setaffinity()`)
- Execute application binary (i.e., `exevp()`)

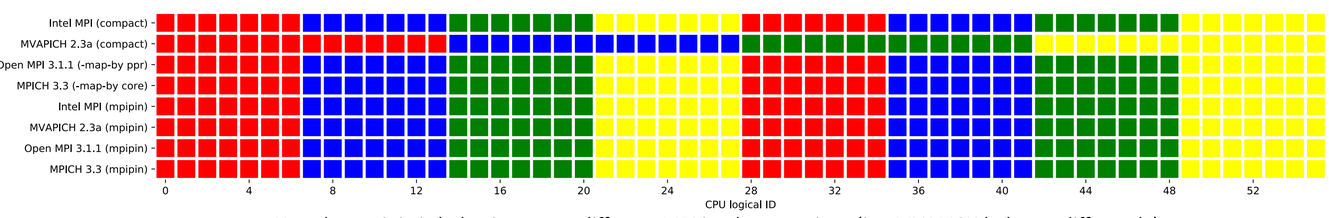
Evaluation Environment

- Oakforest-PACS Supercomputer at JCAHPC (hosted by The University of Tokyo)
- Intel Xeon Phi (KNL) CPU 7250 (1 socket, 68 cores, 4 HW threads / core)
- Intel Xeon CPU E5-2690 v4 (2 sockets, 14 cores / socket, 2 HW threads / core)

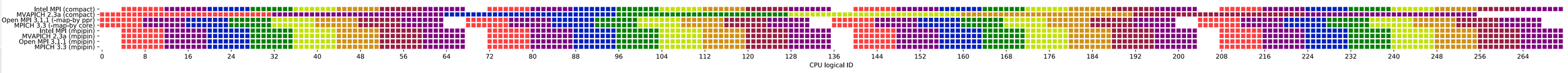
Demonstration and Experiments

Intel MPI, Version 2018 Update 3 Build 20180411 (id: 18329):

```
$ mpirun -env I_MPI_PIN_DOMAIN=14 -env I_MPI_PIN_ORDER=compact \
      -n 4 -ppn 4 -host <host> app
```



Non-deterministic behavior across different MPI implementations (i.e, MVAPICH behaves differently)



Demonstration of CPU exclusion on a 272 logical CPUs KNL (--exclude-cpus=0-3,68-71,136-139,204-207)

MVAPICH2-2.3a with hwloc (HYDRA 3.2):

```
$ mpirun -env MV2_ENABLE_AFFINITY=1 -env MV2_CPU_BINDING_POLICY=hybrid -env MV2_THREADS_PER_PROCESS=14 \
      -env MV2_HYBRID_BINDING_POLICY=compact -n 4 -ppn 4 -host <host> app
```

Open MPI 3.1.1:

```
$ mpirun -map-by ppr:2:socket:pe=7 -np 4 -host <host>:4 app
```

MPICH 3.3:

```
$ mpirun -bind-to core:7 -n 4 -ppn 4 -host <host> app
```

mpipin (regardless the MPI implementation):

```
$ mpirun -n 4 -ppn 4 -host <host> mpipin --threads-per-process 14 --ranks-per-node 4 app
```

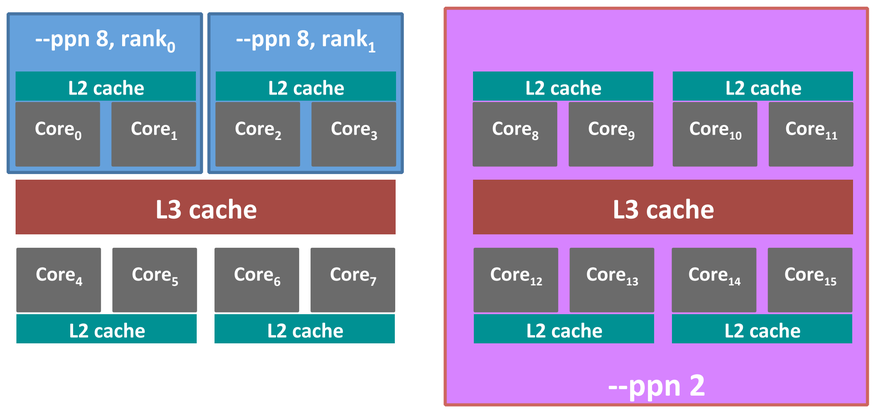
`mpipin`: *an MPI Implementation Agnostic Process Pinning Tool*

```
$ mpirun -hostfile ~/hosts -n <N> -ppn <PPN> \
  mpipin --ranks-per-node <PPN> app
```

Command Line Options

- **--processes-per-node, --ranks-per-node, --ppn**:
  - Specifies the number of MPI processes per node
- **--threads-per-process, --cores-per-process, --tpp**:
  - Specifies the number of threads (logical CPUs) per MPI process
- **--compact**:
  - Follow compact process layout (default)
- **--scatter**:
  - Follow scattered process layout
- **--exclude-cpus, --exclude-cores**:
  - Specifies a list of logical CPUs to be excluded from resource partitioning

Default Pinning Policy



Threads of a process are pinned to logical CPU ids that are close to each other according to the underlying hardware topology (i.e., in the order of shared L1, L2 caches, sockets, NUMA nodes)

Summary

mpipin:

- An MPI implementation agnostic process pinning tool
  - Simple and intuitive interface
  - Transparent, topology aware process placement
  - Not an API extension to mpirun!

シミュレーションが未来をひらく
理化学研究所 計算科学研究機構
RIKEN Advanced Institute for Computational Science