
Process Placement with MPICH2 tuned for vSMP Foundation

Document Date : May 20, 2009

Overview

With MPICH tuned for vSMP Foundation, process placement is controlled through the environment variable - VSMP_PLACEMENT.

Ranks Placement

Unset, or set to NONE or NO (all settings are case-insensitive), would result in no placement on the MPI ranks. You should use some other method to place the ranks if you want them placed.

NODES

Set to NODES, would results in the behavior where each process (rank) gets CPU affinity such that it can run on any CPU in a given node. The next rank uses the next node. The nodes are assigned in a round-robin order starting from the last.

PACKED

Set to PACKED (or PACK), each node in turn is filled up with processes before the next node is used. The ranks are given affinity to exactly one CPU on the node - starting from the last CPU of the last node and progressing "downwards" to the first CPU on the first node.

SPREAD

Set to SPREAD, each rank is assigned the next node in sequence (similar to NODES), but with affinity to just one CPU in that node. The affinity setting starts from the last CPU in the last node and after all nodes have been used, progresses to the next to last CPU on the nodes and so on.

SPECIFIC

One can also use one of these forms:

rank:cpu [,rank:cpu...]

Nrank:node [,rank:node...]

Crank:cpu [,rank:cpu...]

The first form sets ranks to have affinity for the indicated CPU. The third form does the same. The second form sets the affinity to any CPU on the mentioned node.

WARNING: If explicit selection is used, any rank not mentioned will not have CPU affinity set for it.

NUMA Topology

The NUMA topology of the system where the MPI processes are to run is used to indicate how many CPUs each node has. In all cases, you may append a suffix of the form "`^nodes^cpus-per-node`" to override the default topology and define your own number of nodes and cpus per node.

The "override" count of CPUs should not exceed the actual number of CPUs in the system.

Examples

```
export VSMP_PLACEMENT=none
export VSMP_PLACEMENT=PACKED
export VSMP_PLACEMENT=NODES^4^2
export VSMP_PLACEMENT=0:0,1:1,2:2,3:3
export VSMP_PLACEMENT=N0:0,1:1,2:2,3:3
```

Detailed Examples

On a system that has 4 nodes with 8 CPUs each, an MPI job with 16 ranks is run. Here are the CPU affinities used based on the setting of the `VSMP_PLACEMENT` environment variable:

```
VSMP_PLACEMENT=PACKED
CPUs used: 31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16
```

```
VSMP_PLACEMENT=SPREAD
CPUs used: 31,23,15,7,30,22,14,6,29,21,13,5,28,20,12,4
```

```
VSMP_PLACEMENT=PACKED^2^8
CPUs used: 15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
```

```
VSMP_PLACEMENT=SPREAD^16^2
CPUs used: 31,29,27,25,23,21,19,17,15,13,11,9,7,5,3,1
```

NODE based Affinities

And now for some node based affinities (masks) when NODES are used:

VSMP_PLACEMENT=NODES

Masks: ff000000,ff0000,ff00,ff,ff000000,ff0000,ff00,ff,
ff000000,ff0000,ff00,ff,ff000000,ff0000,ff00,ff

VSMP_PLACEMENT=NODES^4^4

Masks: f0000000,f000000,f00000,f0000,f000,f00,f0,f,
f0000000,f000000,f00000,f0000,f000,f00,f0,f

PLACING MULTIPLE JOBS

For example, in order to run 2 MPI jobs, each with 32 ranks, you can use:

'setenv VSMP_PLACEMENT PACKED^8^8' for the first job

and 'VSMP_PLACEMENT PACKED^4^8' for the second job