Charm++ - Bug #2059
Inconsistent CPU affinity options for running SMP programs
03/15/2019 11:27 AM - Nitin Bhat

<table>
<thead>
<tr>
<th>Status:</th>
<th>New</th>
<th>Start date:</th>
<th>03/15/2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assignee:</td>
<td>Evan Ramos</td>
<td>% Done:</td>
<td>0%</td>
</tr>
<tr>
<td>Category:</td>
<td></td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Target version:</td>
<td>6.10.0</td>
<td>Spent time:</td>
<td>0.00 hour</td>
</tr>
</tbody>
</table>

**Description**

It seems to me that on different machines (and different allocations/partitions), to get Charm++ SMP threads to be pinned to cores, one needs to provide different options.

And in some cases, it is not possible to get the desired pinning. (As in the case of Golub)

I was attempting to run a simple pingpong benchmark on 2 nodes. (2 cores on each node, 1 core for the worker thread and another for the comm thread).

Following are the examples of my run commands on different machines.

**Example 1 - Golubs - verbs-linux-x86_64-smp build:**

Allocated a job with 2 nodes, 2 cores each

```
[nbhat4@golub213 p2pPingpong]$ ./charmrun +p2 ./megaZCPingpong ++ppn 1 +pemap 0 +commap 1
```

Running on 2 processors: ./megaZCPingpong ++ppn 1 +pemap 0 +commap 1

Charm++ Running on MPI version: 3.1
Charm++ level of thread support used: MPI_THREAD_SINGLE (desired: MPI_THREAD_FUNNELED)
Charm++ Running in SMP mode: 2 processes, 1 worker threads (PEs) + 1 comm threads per process, 2 PEs total
Charm++ The comm. thread both sends and receives messages
Converse/Charm++ Commit ID: v6.9.0-136-g37f84a4ac
Warning> Randomization of virtual memory (ASLR) is turned on in the kernel, thread migration may not work! Run 'echo 0 > /proc/sys/kernel/randomize_va_space' as root to disable it, or try running with '+isomalloc_sync'.
CharmLB> Load balancer assumes all CPUs are same.
Charm++ cpu affinity enabled.
Charm++ cpuaffinity PE-core map : 0
Charm++ Running on 1 hosts (1 sockets x 2 cores x 1 PUs = 20-way SMP)
Charm++ Warning: Internally-determined PU count does not match hwloc's result!
Charm++ cpu topology info is gathered in 0.044 seconds.

*WARNING: Multiple PEs assigned to same core, recommend adjusting processor affinity or passing +CmsSleepOnIdle to reduce interference.*

Size (bytes),Iterations,Regular Send,ZC EM Send1,ZC EM Send 2,ZC EM Send 3,Regular Recv with Copy,ZC EM Send with Copy,ZC Direct1,ZC Direct2,ZC Direct3,ZC Direct (Reg/Dereg),ZC EM Recv1,ZC EM Recv 2,ZC EM Recv3,Reg time,Dereg time,Reg+Dereg 1,Reg+Dereg 2

Job runs very slowly indicating multiple PEs assigned to the same core as suggested by the warning

**Example 2 - Golubs - verbs-linux-x86_64-smp build:**

Allocated a job with 2 nodes, 2 cores each

```
[nbhat4@golub213 p2pPingpong]$ ./charmrun +p2 ./megaZCPingpong ++ppn 1 +pemap 0,2 +commap 1,3
```

Running on 2 processors: ./megaZCPingpong ++ppn 1 +pemap 0,2 +commap 1,3 ./charmrun: line 99: mvapich2-start-mpd: command not found

Charm++ Running on MPI version: 3.1
Charm++> level of thread support used: MPI_THREAD_SINGLE (desired: MPI_THREAD_FUNNELED)
Charm++> Running in SMP mode: 2 processes, 1 worker threads (PEs) + 1 comm threads per process, 2 PEs total
Charm++> The comm. thread both sends and receives messages
Converse/Charm++ Commit ID: v6.9.0-136-g37f84a4ac
Warning> Randomization of virtual memory (ASLR) is turned on in the kernel, thread migration may not work! Run 'echo 0 > /proc/sys/kernel/randomize_va_space' as root to disable it, or try running with '-isomalloc_sync'.
CharmLB> Load balancer assumes all CPUs are same.
Charm++> cpu affinity enabled.
Charm++> cpuaffinity PE-core map : 0,2
HWLOC> Couldn't bind to cpuset 0x00000004: Invalid argument
HWLOC> Couldn't bind to cpuset 0x00000008: Invalid argument
Charm++> Running on 1 hosts (1 sockets x 2 cores x 1 PUs = 20-way SMP)
Charm++> Warning: Internally-determined PU count does not match hwloc's result!
Charm++> cpu topology info is gathered in 0.085 seconds.
Size (bytes),Iterations,Regular Send,ZC EM Send1,ZC EM Send 2,ZC EM Send 3,Regular Recv with Copy,ZC EM Send with Copy,ZC Direct1,ZC Direct2,ZC Direct3,ZC Direct (Reg/Dereg),ZC EM Recv1,ZC EM Recv 2,ZC EM Recv3,Reg time,Dereg time,Reg+Dereg 1,Reg+Dereg 2

Job runs very slowly probably indicating multiple PEs assigned to the same core or PEs not being pinned to cores at all

Example 3 - Iforge - mpi-linux-x86_64-smp build:
Allocated a job with 2 nodes, 2 cores each

Run command: mpirun -n 2 ./megaZCPingpong ++ppn 1 +pemap 0 +commap 1
Charm++> Running on MPI version: 3.0
Charm++> level of thread support used: MPI_THREAD_FUNNELED (desired: MPI_THREAD_FUNNELED)
Charm++> Running in SMP mode: 2 processes, 1 worker threads (PEs) + 1 comm threads per process, 2 PEs total
Charm++> The comm. thread both sends and receives messages
Converse/Charm++ Commit ID: v6.9.0-136-g4bb2ac6
Warning> Randomization of virtual memory (ASLR) is turned on in the kernel, thread migration may not work! Run 'echo 0 > /proc/sys/kernel/randomize_va_space' as root to disable it, or try running with '-isomalloc_sync'.
CharmLB> Load balancer assumes all CPUs are same.
Charm++> cpu affinity enabled.
Charm++> cpuaffinity PE-core map : 0
Charm++> Running on 1 hosts (2 sockets x 12 cores x 1 PUs = 24-way SMP)
Charm++> cpu topology info is gathered in 0.023 seconds.
WARNING: Multiple PEs assigned to same core, recommend adjusting processor affinity or passing +Cm iSleepOnIdle to reduce interference.
Size (Bytes),Iterations,Regular Send(us),ZC EM Send UNREG mode(us),ZC EM Send REG Mode (us),ZC EM Send PREREG Mode(us),Regular Recv with Copy(us),ZC EM Send with Copy(us),ZC Direct UNREG(us),ZC Direct REG(us),ZC Direct PREREG(us),ZC Direct (Reg/Dereg),ZC Post Recv UNREG(us),ZC Post Recv REG(us),ZC Post Recv PREREG(us),Reg Time(us),Dereg Time(us),Reg + Dereg Time 1(us),Reg + Dereg Time 2(us)

Job runs very slowly indicating multiple PEs assigned to the same core as suggested by the warning. However, the mpi-linux-x86_64-smp build with other options ++ppn 1 +pemap 0,2 +commap 1,3 works well. And the verbs-linux-x86_64-smp build with same options ++ppn 1 +pemap 0 +commap 1 also works well.

Example 4 - Bridges - RM-small partition ofi-linux-x86_64-smp build:
Allocated a job with 2 nodes, 2 cores each

[nbhat4@r001 p2pPingpong]$ ./charmrun +p2 ./megaZCPingpong ++ppn 1 +pemap 0 +commap 1
Running on 2 processors: ./megaZCPingpong +ppn 1 +pemap 0 +commap 1
Job runs very slowly indicating no pinning at all

Example 5 - Bridges - RM-small partition ofi-linux-x86_64-smp build:
Allocated a job with 2 nodes, 2 cores each

[nbhat48@r001 p2pPingpong]$ ./charmrun +p2 ./megaZCPingpong +ppn 1 +pemap 0,2 +commap 1,3

Running on 2 processors: ./megaZCPingpong +ppn 1 +pemap 0,2 +commap 1,3
Work! Run `echo 0 > /proc/sys/kernel/randomize_va_space` as root to disable it, or try running with `+isomalloc_sync`.

CharmLB> Load balancer assumes all CPUs are same.

HWLOC> Couldn’t bind to cpuset 0x00000004: Invalid argument

Charm+++ cpu affinity enabled.

Charm+++ cpuaaffinity PE-core map : 0,2

HWLOC> Couldn’t bind to cpuset 0x00000008: Invalid argument

HWLOC> Couldn’t bind to cpuset 0x000000002: Invalid argument

Charm+++ Running on 2 hosts (1 sockets x 1 cores x 1 PUs = 28-way SMP)

Charm+++ Warning: Internally-determined PU count does not match hwloc’s result!

Charm+++ cpu topology info is gathered in 0.150 seconds.

<table>
<thead>
<tr>
<th>Size (Bytes),</th>
<th>Iterations,</th>
<th>Regular Send(us),</th>
<th>ZC EM Send UNREG mode(us),</th>
<th>ZC EM Send REG Mode (us),</th>
<th>ZC EM Send PREREG Mode(us),</th>
<th>Regular Recv with Copy(us),</th>
<th>ZC EM Send with Copy(us),</th>
<th>ZC Direct UNREG(us),</th>
<th>ZC Direct REG(us),</th>
<th>ZC Direct PREREG(us),</th>
<th>ZC Direct (Reg/Dereg),</th>
<th>ZC Post Recv UNREG(us),</th>
<th>ZC Post Recv REG(us),</th>
<th>ZC Post Recv PREREG(us),</th>
<th>Reg Time(us),</th>
<th>Dereg Time(us),</th>
<th>Reg + Dereg Time 1(us),</th>
<th>Reg + Dereg Time 2(us)</th>
</tr>
</thead>
</table>

Job runs very slowly indicating no pinning at all. However, with another partition (RM), the job runs successfully with `++ppn 1 +pemap 0,2 +commap 1,3`.

I think the preferred options for pinning for the example I described are `"++ppn 1 +pemap 0 +commap 1"`. However, they do not work consistently across all systems. We need to investigate and make them work consistently.

**History**

#1 - 03/15/2019 03:45 PM - Evan Ramos

I'm wondering if this is a regression from the introduction of hwloc.

#2 - 03/15/2019 09:04 PM - Jim Phillips

You need to at least show the options with which you submitted the job, since mpirun often picks up environment variables. The verbs builds should now report running mpi, and I don't trust the charmrun script that tries to call mpirun.

#3 - 03/15/2019 09:15 PM - Jim Phillips

The verbs builds should not report running mpi.

#4 - 03/28/2019 11:54 AM - Eric Bohm

- Assignee set to Evan Ramos