Gromacs - Feature #1122
Allow to force pinning
01/17/2013 10:09 AM - Roland Schulz

<table>
<thead>
<tr>
<th>Status:</th>
<th>Blocked, need info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>Szilárd Páll</td>
</tr>
<tr>
<td>Category:</td>
<td>mdrun</td>
</tr>
<tr>
<td>Target version:</td>
<td>5.x</td>
</tr>
</tbody>
</table>

**Description**

We try to be nice and don't pin if affinity is already set. This seems in general good, but sometime it is very non-obvious why the affinity is non-default. An example is the QLogic Infiniband OpenMPI back-end, whose developers thought it is a good idea to set the affinity if no-one else has set it. Thus even if no OpenMPI affinity options are set it still sets the affinity. And it can only be deactivated by an environment-variable (IPATH_NO_CPUAFFINITY). Needless to say it took me forever to find that. Adding an option "-pin force" (or renaming the current option to "auto" and adding a "yes"), which always sets the affinity even if it is already set, would help users in those cases.

I set it as bug, because even though I'm suggesting to add an option, it is really trying to fix a usability problem.

Off topic: The QLogic documentation points out that ideally the affinity is set as early as possible so that the MPI library affinity is correct too. Not sure how we could improve that.

**Related issues:**

Related to Gromacs - Bug #1633: mdrun -nsteps -1 reports silly numbers

**Associated revisions**

Revision 4bebcd8f - 01/18/2013 09:21 PM - Berk Hess

thread affinity now uses some topology information

The order of logical cores on x86 is hardware and software dependent.
The cpuid topology reports this and this information is now used.
The mdrun -pinht option is generalized for SMT to -pinstride.
The mdrun -pinoffset option is now in logical (iso phyiscal) cores.
Thread-MPI no longer sets affinity, it's now all done in one place.
The option -pin is now an enum, default auto: only on when using all cores and when no external affinity has been set.
A big NOTE is printed with auto when no pinning is used.
Option -pin on can now override thread affinity set outside mdrun.
Fixes #1122
All thread affinity code has been moved from runner.c to gmx_thread_affinity.c.
Updated the mdrun manual for pinning also active without OpenMP.

Change-Id: Ibf0fe5882688de80c223640502c68e6170d4d044

**History**

#1 - 01/17/2013 10:21 PM - Berk Hess
I actually wrote the same thing in a comment in my patch. I suggested to make -pin on (try to) override any already present pinning. This might be as simple as skipping all affinity detection checks when -pin is set to on. But I didn't want to make even more changes just before the release.

---

**#2 - 01/22/2013 01:10 PM - Berk Hess**

Roland, did my commit allow you to force the affinity in your case? We still need to check in which case we can override OpenMP affinity.

---

**#3 - 01/24/2013 07:41 PM - Roland Schulz**

No it isn't fixed. In fact it is so bad that I though it was hanging. One single call to construct_vsites takes 0.1s. So even for DHFR it doesn't get to step 0 in less than 10min. It is possible to override processor affinity in general? Or is something we cannot be fixed, because as soon as the QLogic driver has set the affinity, we cannot override it anymore?

---

**#4 - 02/14/2013 02:08 PM - Berk Hess**

- Category set to mdrun
- Assignee set to Szilárd Páll

This should be simple to fix, but I don't know which OpenMP affinity settings we can override.

---

**#5 - 02/14/2013 09:33 PM - Szilárd Páll**

As far as I know it is/should be always possible to override the affinities. However, I know of one case

I think there are two reasonable solutions:

1. keep -pin on a "soft" option and not let it override the external affinity & add a "hard" option e.g -pin force;
2. make -pin on a "hard" option that does override externally set affinity, but issues a big warning when we are actually overriding.

The reason why I think we should either add "force" option or a warning is that we sometimes default to pinning and other times to no pinning, but not pinning often causes considerable performance loss. Therefore, we advise people (note to self: this advice and explanation on it is missing from the wiki) to do pin manually*. This means that many people will (and should) probably always add -pin on to their command line to avoid realizing the performance loss when it's too late. However, we should avoid encouraging people to always override external affinities - which will piss off sysadmins of machines where node-sharing is set up properly and the job scheduler does sets the process affinities.

*Slightly off-topic: perhaps we should even add an environment variable or a cmake option that turns the default pinning on regardless of whether the full machines is used or not (it has already happened to me numerous times that I realized only after running a bunch of benchmarks that I should have set -pin on).

---

**#6 - 04/29/2013 07:54 PM - Mark Abraham**

- Target version deleted (4.6.1)
- Affected version set to 4.6
Is this resolved?

#7 - 04/29/2013 08:07 PM - Szilárd Páll
No, it is not. None of my above suggestions has been implemented yet - mainly because I got no feedback.

#8 - 06/26/2013 01:44 AM - Mark Abraham
- Target version set to 4.6.x

#9 - 09/19/2013 06:48 PM - Szilárd Páll
- Tracker changed from Bug to Feature

#10 - 09/19/2013 06:49 PM - Szilárd Páll
The more I think of this, the more it seems to be a feature that we have to proceed with caution about. I'm going to ask for feedback on the developers' list.

#11 - 09/23/2013 08:43 PM - Mark Abraham
I'm in favor of the current default behaviour wrt affinities. It seems a reasonable compromise in the era of multicore-oblivious kernels.

I am happy to add

```
mdrun -pin force
```

that will override external affinity settings, with suitable warning that this is actually taking place (ie. Szilard's option 2 from post 7).

#12 - 09/26/2013 02:57 AM - Szilárd Páll
- Status changed from In Progress to Feedback wanted

See: https://gerrit.gromacs.org/#/c/2633/

#13 - 09/27/2013 02:02 AM - Szilárd Páll
Actually, as I suspected back in January, things are not as simple as they seemed. Something does not work very well when trying to override affinities set through the OpenMP interface (GOMP_CPU_AFFINITY/KMP_AFFINITY). Initially I was testing on a small number of cores, and while there seemed to be a small (1-2%) performance difference, I thought it was just fluctuation. However, I've just done some more tests and and measured huge performance degradation.

Running on 8-core Intel Sandy Bridge E, gcc 4.7.

- mdrun -ntmpi 1 -ntomp 8 -pin force: 42.5 ns/day and

```
hwloc-ps -t
23858  PU-0  PU:2  PU:4  PU:6  PU:8  PU:10  PU:12  PU:14 /.../mdrun
23858  PU:0
23863  PU:2
23864  PU:4
23865  PU:6
23866  PU:8
23867  PU:10
23868  PU:12
23869  PU:14
```

- taskset 0x1 $mdrun -ntmpi 1 -ntomp 8 -pin force: 33.6 ns/day and

11/25/2015
GOMP_CPU_AFFINITY=0 $mdrun -ntmpi 1 -ntomp 8 -pin force 33.5 ns/day and

Obviously, the affinity patterns looks OK (and are identical), so I can't really explain the performance difference with anything other than some memory/cache affinity issue or something else related to threads and OpenMP because the same thing is not observable with tMPI-only (e.g. mdrun -ntmpi 8 -ntomp 1).

I've reproduced the same behavior on AMD Piledriver amd Magny Cours as well.

Additionally, what's weird is that with -ntmpi 4 -ntomp 2 I get bizarre affinity patterns reported by hwloc (slightly different when overriding and when not), but the affinity masks queried inside mdrun look fine.

mdrun -ntmpi 4 -ntomp 2 -pin force
Current Process 24432 rank 0 ->
  thread 0 -> 0,  1 CPU(s) in the mask
  thread 1 -> 1,  1 CPU(s) in the mask

Current Process 24432 rank 3 ->
  thread 0 -> 6,  1 CPU(s) in the mask
  thread 1 -> 7,  1 CPU(s) in the mask

Current Process 24432 rank 1 ->
  thread 0 -> 2,  1 CPU(s) in the mask
  thread 1 -> 3,  1 CPU(s) in the mask

Current Process 24432 rank 2 ->
  thread 0 -> 4,  1 CPU(s) in the mask
  thread 1 -> 5,  1 CPU(s) in the mask

hwloc-ps -t
24432  PU:0 PU:2 PU:4 PU:6 PU:8 PU:10 PU:12 PU:14 /.../mdrun
24433  PU:0
24436  PU:4
24437  PU:8
24438  PU:12
24439  PU:6
24440  PU:2
24441  PU:10
24442  PU:14

GOMP_CPU_AFFINITY=0 mdrun -ntmpi 4 -ntomp 2 -pin force

Current Process 24353 rank 2 ->
  thread 0 -> 4,  1 CPU(s) in the mask
  thread 1 -> 5,  1 CPU(s) in the mask

Current Process 24353 rank 1 ->
  thread 0 -> 2,  1 CPU(s) in the mask
  thread 1 -> 3,  1 CPU(s) in the mask

Current Process 24353 rank 3 ->
  thread 0 -> 6,  1 CPU(s) in the mask
  thread 1 -> 7,  1 CPU(s) in the mask

Current Process 24353 rank 0 ->
  thread 0 -> 0,  1 CPU(s) in the mask
  thread 1 -> 1,  1 CPU(s) in the mask

$ hwloc-ps -t
24353  PU:0 PU:2 PU:4 PU:6 PU:8 PU:10 PU:12 PU:14 /.../mdrun
24354  PU:0
24358  PU:4
24359  PU:8
24360  PU:12
24361  PU:2
24362  PU:10
24363  PU:14
24364  PU:6
Szilárd Páll wrote:

Additionally, what's weird is that with -ntmpi 4 -ntomp 2 I get bizarre affinity patterns reported by hwloc (slightly different when overriding and when not), but the affinity masks queried inside mdrun look fine.

It turns out that this is normal, OpenMP seems to create a thread pool but assign thread logical IDs in a different order.

Could the last observation be caused by the bug reported in #1360?

It could. I'll rebase change 2633 and see if it makes a difference.

No, #1360 is not related which is not surprising as nothing indicated that multiple threads were pinned to the same hardware thread.

Is this solved already?

No, the above issues still persist.

- Status changed from Feedback wanted to Blocked, need info

- Description updated

Szilárd Páll wrote:

No, the above issues still persist.

Why issues (plural)? Note 15 says that the problem described in note 14 is normal. Thus I understand your notes, that the only remaining issue is the one described in note 13. Is this correct?

Did you try the issue described in 13 with more than one compiler/OpenMP-runtime? The way you describe it, it could be that some of the OpenMP internal storage might be allocated on the wrong section of the NUMA memory.
#24 - 06/17/2014 11:00 PM - Mark Abraham
- Target version changed from 4.6.x to 5.x

#25 - 12/09/2014 10:36 PM - Szilárd Pál
- Related to Bug #1633: mdrun -nsteps -1 reports silly numbers added