GROMACS - Bug #274

Crashes in mdrun with DD code, not sequential

12/26/2008 09:33 PM - David van der Spoel

Status:	Closed		
Priority:	Normal		
Assignee:	Berk Hess		
Category:	mdrun		
Target version:	4.0_rc1		
Affected version - extra info:		**Difficulty:**	uncategorized
Affected version:			

Description

Created an attachment (id=333)
Input files for mdrun.

Hi,

the attached system crashes within 500 steps when run using

mpirun -c 2 mdrun -cpi back -v -dlb no -reprod

but not when running on single processor. This is after 500 ps equilibration.
It was tested with latest 4.0.release CVS and compiled with -g. It is reproducible on Apple and x86_64 linux (Centos, gcc 4.1.2). It does not seem to crash when run using gdb on two processors.

History

#1 - 12/26/2008 09:34 PM - David van der Spoel

The code also crashes on 4 cpus.

#2 - 01/05/2009 05:00 PM - Berk Hess

After 300 steps valgrind writes the message below.

I assume this means that the data being communicated
(which is the array atc->f that at the malloc call that valgrind gives),
is not initialized.
But I checked the array sizes and the data is not reallocated
at any point during the run. Also the array indices are not out of range.
I have no clue what could cause this problem.

Berk

29334
29334 Syscall param writev(vector[...]) points to uninitialised byte(s)
29334 at 0x66ED686: (within /lib64/libc-2.6.1.so)
29334 by 0x760FC1: swritev (in /people/pckr59/hessb/gmx4.0/obj/g_x86_64/src/kernel/mdrun)
29334 by 0x76133C: lam_ssi_rpi_tcp_low_fastsend (in /people/pckr59/hessb/gmx4.0/obj/g_x86_64/src/kernel/mdrun)
29334 by 0x75F003: lam_ssi_rpi_tcp_fastsend (in /people/pckr59/hessb/gmx4.0/obj/g_x86_64/src/kernel/mdrun)
29334 by 0x74D83C: lam_send (in /people/pckr59/hessb/gmx4.0/obj/g_x86_64/src/kernel/mdrun)
29334 by 0x743FF5: MPI_Send (in /people/pckr59/hessb/gmx4.0/obj/g_x86_64/src/kernel/mdrun)
29334 by 0x7438F5: MPI_Sendrecv (in /people/pckr59/hessb/gmx4.0/obj/g_x86_64/src/kernel/mdrun)
29334 by 0x478FEE: pme_dd_sendrecv (pme.c:482)
29334 by 0x479773: dd_pmeredist_f (pme.c:604)
29334 by 0x47EE03: gmx_pme_do (pme.c:1950)
29334 by 0x459C44: do_force_lowlevel (force.c:1731)
29334 by 0x4926CD: do_force (sim_util.c:557)
29334 Address 0x7791b50 is 78,552 bytes inside a block of size 95,676 alloc'd
29334 at 0x4C2202B: malloc (vg_replace_malloc.c:207)
29334 by 0x5168F4: save_realloc (smalloc.c:175)
29334 by 0x47851F: pme_realloc_atomcomm_things (pme.c:357)
29334 by 0x479336: dd_pmeredist_x_q (pme.c:540)
29334 by 0x47E879: gmx_pme_do (pme.c:1842)
29334 by 0x459C44: do_force_lowlevel (force.c:1731)

29334 by 0x4926CD: do_force (sim_util.c:557)
29334 by 0x417321: do_md (md.c:1111)
29334 by 0x41494A: mdrunner (md.c:426)
29334 by 0x41A473: main (mdrun.c:488)

#3 - 01/05/2009 05:27 PM - Berk Hess

I think the valgrind pme message is probably unimportant.

I ran the system on 1 and 2 nodes with log output at every step.
What happens is that at some point the Coulomb SR becomes nan,
while all other energy terms are very similar between 1 and 2 nodes.
This means, as I mailed before, that a Coulomb interaction
goes beyond the tab extension, which is the default 1 nm.
I suspect that a charge group somehow ends up on the wrong node
or in the wrong nblist.
It happens 1 step after an ns step.
I don't have a clue where the problem could be.
Most scenarios I can think of would crash exactly at an ns step.
Unless the pair slowly moves out of the table buffer...

This is difficult to debug.
We need a routine that checks the distances of all atom pairs
in the nb lists.

Berk

#4 - 01/05/2009 07:29 PM - David van der Spoel

If a pair is within the ns cut-off at the ns step then it is hard to imagine that it is outside rcut+table_extension the next step, unless the table_extension is 0. Setting table extension at the beginning of the run might give a clue. If we fill the tables beyond table extesion with NaN, can we generate a floating point exception?

Maybe we should make a debugging version of the generic C loop?

#5 - 01/05/2009 11:20 PM - Berk Hess

Generating an exception does not really help,
since the error probably occurred before this happens.

Also making a debug version of the generic loop is not
very helpful, since this changes the results of the code.

I will think about writing a distance checking routine.

Berk

#6 - 01/06/2009 02:30 PM - Berk Hess

I think the problem is with the pbc and cgcm of vsites.
You have cg's that consist of a single vsite,
I think there is a bug related to that situation.

Berk

#7 - 01/06/2009 02:42 PM - Erik Marklund

Right. I experimented with the charge groups a bit. I tried to keep it to one charge group per atom, with the exception for CH3 and such, as recommended in earlier communications about amber and vsites. Splitting it so that vsites were distributed over several charge grous didn't help the situation.

#8 - 01/06/2009 03:35 PM - Berk Hess

I found the bug.
I thought that in all cases the vsites were constructed
with the correct pbc.
But actually a single charge group vsite is constructed
with the pbc of its first contructing atom.
This gives errors when these two cg's do not have the same pbc.

I committed the fix for the release branch.
I also committed the pbc.c fixes.

Berk

Files

dd.tgz	559 KB	12/26/2008	David van der Spoel