**Gromacs - Bug #1191**

**REMD bug report**

03/14/2013 09:25 AM - daisuke kushibe

<table>
<thead>
<tr>
<th>Status:</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>Affected version - extra info:</td>
<td></td>
</tr>
<tr>
<td>Affected version:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

Hello I'm Daisuke Kushibe.

I report REMD bug.

When we do REMD simulation, GROMACS REMD simulation sometimes has crashed.

error message is about "Lincs warning".

But, this crash is unrelated to Lincs like constraint algorithm.

repl_ex.c has following bug.

In function gmx_bool replica_exchange (1297 line)

When we do not use domain decomposition or particle decomposition, following routine is never curried out. therefore, REMD routine never carried out "dd_collect_state" function.

In dd_collect_state(or pd_collect_state) function is important for REMD simulation. Because "state" variable includes the information of simulation box such as box, boxv etc. As a result of this bug, physical quantity is not conserved.

If I say closely, physical quantity is exchanged collectly. But, state variable is not collect. therefore physical quantity is initialized every exchange.

When we use Parrinello-Rahman algorithm, the size of box changes, but box size is initialized every "replex" steps.

Therefore, the simulation system experiences the abnormal stress, This is a trigger of REMD simulation crash(cause of the Lincs warning).

```c
if (bThisReplicaExchanged) {
    /* Exchange the states */

    if (PAR) {
        /* Collect the global state on the master node */
        if (DOMAINDECOMP) {
            dd_collect_state(cr->dd, state_local, state);
        }
        else {
            pd_collect_state(cr, state);
        }
    }
}
```

It is necessary to change source code to avoid this bug.

```c
if (PAR(cr))
```
"non_par_collect_state" is for 1cpu core "dd_collect_state".

The true problem is as follow.
(1)"Sometimes" REMD crashs, but not "Always".
(2)Physical quantity is not conserved by replica exchange when we do not use "domain decoposition" and "particle decomposition".
In this case, calculation does not crash. but algorithm is wrong.

see also two figures.

that's all

---

**History**

#1 - 03/14/2013 09:55 AM - daisuke kushibe

I'm sorry.
I forgot to write the target gromacs version.
Target version is 4.6.1

#2 - 03/14/2013 12:55 PM - Mark Abraham

I'm not quite sure what circumstances are triggering what perceived problem. So I'll explain a few things and we'll see where we get to.

The implementation of REMD in GROMACS exchanges the coordinates, not the thermodynamic ensemble state variables. (Other MD codes do this differently; GROMACS should do likewise but will not do so any time soon.) Unfortunately, the coordinate variables live in a data structure whose name uses the word "state". The *_collect_state functions are really just collecting the configuration variables so that they're all in one place for the exchange. As such, there's nothing for a simulation with only one node to collect; everything is already there.

Whether a single-node replica should be copying something from state_local to state is not immediately clear to me. You seem to be suggesting that the box is getting reset strangely and that this perturbation of equilibrium induces LINCS errors. An REMD trajectory file written by GROMACS is supposed to be continuous with respect to thermodynamic ensemble. So while an NPT ensemble will vary its volume, that variation should never be step-wise, even at the point of an exchange. Viewed from an individual set of coordinates, the trajectory will be discontinuous with respect to volume, because the coordinates are scaled to fit the volume of the new ensemble upon each exchange. But you'd have to have demuxed your trajectory in order to see that. What are your figures actually showing?

Can you please combine your .tpr files into a tarball and attach them, so I can see the behaviour first hand?
The implementation of REMD in GROMACS exchanges the coordinates, not the thermodynamic ensemble state variables.

I understand this specifications.

(Other MD codes do this differently; GROMACS should do likewise but will not do so any time soon.)

From the view points of parallel calculation, communication cost is very high in this case. The most cheapest way is to exchange ref_t value. Instead, we do not need to rearrange the .edr and .trr by replica_index.xvg information.

Unfortunately, the coordinate variables live in a data structure whose name uses the word "state".

The word "state" means t_state *state in function of gmx_bool replica_exchange. The "state" variable in source code includes matrix box, boxv, box_rel, nosehoover_xi, nosehoover_vxi, nhpres_xi, nhpres_vxi etc. These variables have to be exchanged by "just before" information, not "Initial value".

Therefore, I test REMD simulation using gromacs's test data.

[test information]
Name : methanol(1 molecule)
Directory : tutor/methanol

For simplicity I executed single methanol molecule hamiltonian system in vacuum. My purpose is to show the change history of t_state *state variable.

To do this, I insert debug write fprintf to <void parrinellorahman_pcoupl(couple.c,338 line)> ann <gmx_bool replica_exchange (repl_ex.c)>.

Insertion code are as follow.

============void parrinellorahman_pcouple debug code (couple.c)======================
int    d, n;
tensor winv;
real   vol = box[XX][XX]*box[YY][YY]*box[ZZ][ZZ];
real   atot, arel, change, maxchange, xy_pressure;
tensor invbox, pdiff, t1, t2;
real   maxl;
printf(fplog,"My PR : step=%10d, vol=%10g, box_x=%10g, box_y=%10g, box_z=%10g\n",
(int)step,(float)vol,(float)box[XX][XX],(float)box[YY][YY],(float)box[ZZ][ZZ]);
===========================================================================
==============gmx_bool replica_exchange debug code(repl_ex.c) =========================
fprintf(fplog,"Just before Exchange\n") ;
fprintf(fplog,"My Repl : step=%10d, time=%10g, box_x=%10g, box_y=%10g, box_z=%10g\n",
(int)step,(float)time,
state_local->box[XX][XX],state_local->box[YY][YY],state_local->box[ZZ][ZZ]);
fprintf(fplog,"My Repl : step=%10d, time=%10g, box_rel_x=%10g, box_rel_y=%10g, box_rel_z=%10g\n",
(int)step,(float)time,
state_local->box_rel[XX][XX],state_local->box_rel[YY][YY],state_local->box_rel[ZZ][ZZ]);
fprintf(fplog,"My Repl : step=%10d, time=%10g, boxv_x=%10g, boxv_y=%10g, boxv_z=%10g\n",
(int)step,(float)time,
state_local->boxv[XX][XX],state_local->boxv[YY][YY],state_local->boxv[ZZ][ZZ]);
fprintf(fplog,"My Repl : step=%10d, time=%10g, nosehoover_xi=%10g, nosehoover_vxi=%10g, nhpres_xi=%10g, nhpres_vxi=%10g\n",
(int)step,(float)time,
state_local->nosehoover_xi, state_local->nosehoover_vxi, state_local->nhpres_xi, state_local->nhpres_vxi); 
fprintf(fplog,"My Coord : state->x[XX]=%10g, state->x[YY]=%10g, state->x[ZZ]=%10g\n",
state->x[XX], state->x[YY], state->x[ZZ]);
exchange_state(cr->ms, exchange_partner, state);

fprintf(fplog,"Just after Exchange\n") ;

fprintf(fplog,"My Repl : step=%10d,time=%10g,box_x=%10g,box_y=%10g,box_z=%10g\n",(int)step,(float)time,
state_local->box[XX][XX],state_local->box[YY][YY],state_local->box[ZZ][ZZ]) ;

fprintf(fplog,"My Repl : step=%10d,time=%10g,box_rel_x=%10g,box_rel_y=%10g,box_rel_z=%10g\n",(int)step,(float)time,
state_local->box[XX][XX],state_local->box[YY][YY],state_local->box[ZZ][ZZ]) ;

fprintf(fplog,"My Repl : step=%10d,time=%10g,boxv_x=%10g,boxv_y=%10g,boxv_z=%10g\n",(int)step,(float)time,
state_local->boxv[XX][XX],state_local->boxv[YY][YY],state_local->boxv[ZZ][ZZ]) ;

fprintf(fplog,"My Repl : step=%10d,time=%10g,nosehoover_xi=%10g,nosehoover_vxi=%10g,nhpres_xi=%10g\n",(int)step,(float)time,
state_local->nosehoover_xi,state_local->nosehoover_vxi,state_local->nhpres_xi) ;

printf(fplog,"My Coord : state->x[XX]=%10g,state->x[YY]=%10g,state->x[ZZ]=%10g\n",state->x[XX],state->x[YY],state->x[ZZ]) ;

createFromParrinellorahman_pcoupl function.
For simplicity, I use three replicas and replx=10000.
I do not use domain decomposition and particle decomposition algorithm.
Each MD is carried out by 1 CPU core. Total CPU cores for this REMD simulation are 3 cpu core.

Result is as follow,

(1) Replica exchange occur during replica No0 and replica No1.
Replica exchange at step 1000 time 2
Repl 0 <-> 1
dE_term = -3.078e-02 (kT)
dP = -1.192e-04  d = -3.089e-02
bRepex 0 x 1
bPlx pr 1.0

(2) Debug information of replica No0 (md0.log)
My PR : step= 0, vol= 174876, box_x= 55.9213, box_y= 55.9213, box_z= 55.9213
My PR : step= 0, vol= 174876, box_rel_x= 0, box_rel_y= 1, box_rel_z= 1
My PR : step= 0, vol= 174876, boxv_x= 0, boxv_y= 0, boxv_z= 0
My PR : step= 10000, vol= 172373, box_x= 55.6531, box_y= 55.6531, box_z= 55.6531
My PR : step= 10000, vol= 172373, box_rel_x= 0, box_rel_y= 1, box_rel_z= 1
My PR : step= 10000, vol= 172373, boxv_x= -0.0267922, boxv_y= -0.0267922, boxv_z= -0.0267922

IN bThisReplicaExchanged OK?,PAR(cr)=0,DOMAINDECOMP(cr)=0

Just before Exchange
My Repl : step= 10000, time= 20, box_x= 55.6531, box_y= 55.6531, box_z= 55.6531
My Repl : step= 10000, time= 20, box_rel_x= 0, box_rel_y= 1, box_rel_z= 1
My Repl : step= 10000, time= 20, boxv_x= -0.0267945, boxv_y= -0.0267945, boxv_z= -0.0267945

copy_state_nonatomdata
My PR : step= 10001, vol= 174876, box_x= 55.9212, box_y= 55.9212, box_z= 55.9212
My PR : step= 10001, vol= 174876, box_rel_x= 0, box_rel_y= 1, box_rel_z= 1
My PR : step= 10001, vol= 174876, boxv_x= 0, boxv_y= 0, boxv_z= 0

(3) Debug information of replica No1 (md1.log)
My PR : step= 0, vol= 174876, box_x= 55.9212, box_y= 55.9212, box_z= 55.9212
My PR : step= 0, vol= 174876, box_rel_x= 0, box_rel_y= 1, box_rel_z= 1
My PR : step= 0, vol= 174876, boxv_x= 0, boxv_y= 0, boxv_z= 0
My PR : step= 10000, vol= 172373, box_x= 55.6531, box_y= 55.6531, box_z= 55.6531
My PR : step= 10000, vol= 172373, box_rel_x= 0, box_rel_y= 1, box_rel_z= 1
My PR : step= 10000, vol= 172373, boxv_x= -0.0267919, boxv_y= -0.0267919, boxv_z= -0.0267919

IN bThisReplicaExchanged OK?,PAR(cr)=0,DOMAINDECOMP(cr)=0

Just before Exchange
My Repl : step= 10000, time= 20, box_x= 55.6531, box_y= 55.6531, box_z= 55.6531
My Repl : step= 10000, time= 20, box_rel_x= 0, box_rel_y= 1, box_rel_z= 1
My Repl : step= 10000, time= 20, boxv_x= -0.026794, boxv_y= -0.026794, boxv_z= -0.026794
My Repl : step = 10000, time = 20, nosehoover_xi = -0.0267945, nosehoover_vxi = -0.0267945, nhpres_xi = -0.0267945

Just after Exchange
My Repl : step = 10000, time = 20, box_x = 55.6531, box_y = 55.6531, box_z = 55.6531
My Repl : step = 10000, time = 20, box_rel_x = 1, box_rel_y = 1, box_rel_z = 1
My Repl : step = 10000, time = 20, boxv_x = -0.0267945, boxv_y = -0.0267945, boxv_z = -0.0267945

My Repl : step = 10000, time = 20, nosehoover_xi = -0.0267945, nosehoover_vxi = -0.0267945, nhpres_xi = -0.0267945

copy_state_nonatomdata
My PR : step = 10001, vol = 174876, box_x = 55.9213, box_y = 55.9213, box_z = 55.9213
My PR : step = 10001, vol = 174876, box_rel_x = 1, box_rel_y = 1, box_rel_z = 1
My PR : step = 10001, vol = 174876, boxv_x = 0, boxv_y = 0, boxv_z = 0

[flow chart]
Parrinello-Rahman routine (every step)
Replica exchange (every "replex" step)

?box[ZZ][ZZ] value
Replica No0 : 55.9213 (step 0) --> 55.6531 (step 10000) --> Replica Exchange! --> 55.9212 (10001 step)
Replica No1 : 55.9212 (step 0) --> 55.6531 (step 10000) --> Replica Exchange! --> 55.9213 (10001 step)

?boxv[ZZ][ZZ] value
Replica No0 : 0 (step 0) --> -0.0267922 (step 10000) --> Replica Exchange! --> 0 (10001 step)
Replica No0 : 0 (step 0) --> -0.0267922 (step 10000) --> Replica Exchange! --> 0 (10001 step)

[What's wrong]
(1) We need to solve Parrinello-Rahman type equation of motion. Input parameter includes "matrix box etc".
but, the value of "matrix box" is not correct. Because, when we do not use domain decomposition or particle
decomposition, dd_collect_state function is never carried out.
Because PAR = 0 and DOMAINDECOMP are equal to 0.

(2) Replica Exchange has to been done by "Just before step information".
(2) Why "step 10001's box[ZZ][ZZ] in replica No0" is equal to "step 0's box[ZZ][ZZ] in replica No1"?
(3) Why "step 10001's box[ZZ][ZZ] in replica No0" is not equal to "step 10000's box[ZZ][ZZ] in replica No1"?