README

Yun Ding

December 30, 2013

Contents

1 Preface

GPU ligand docking project.

- The implementation of ligand docking on GPU architecture
- Replica-exchange Monte Carlo method ussed as the searching method

2 Developers' log

2.1 DONE Data structure from array of structure (AoS) to structure of array (SoA)

2.2 DONE Energy calculation and Metropolis Monte Carlo

2.2.1 DONE perturbation of ligand

1. **DONE** c++ code

   - ☒ in loading, the coord of original ligand should be guaranteed to be in its center reference
   - ☒ after loading, the coord of original ligand should be guaranteed to be in its center reference stored in

     ```
     LigCoord *coord_orig = &mylig->coord_orig;
     ```

   - ☒ InitLigCoord added, to initialize the original coord of the ligand

     ```
     void InitLigCoord (Ligand *, const ComplexSize);
     ```

   - ⊟ in perturbation
     - c++ code

     ```
     double boltzmann     = 1.0;
     double step_t        = 1.0;
     double step_r        = 5.0;
     ```

```
for ( int i = 0; i < 3; i++ )
{
mc_conf3[i] = mc_conf1[i];

mc_conf1[i] += mc_t * unirand( -1, 1 );
}

for ( int i = 3; i < 6; i++ )
{
mc_conf3[i] = mc_conf1[i];

mc_conf1[i] += ( ( mc_r * unirand( -1, 1 ) ) * PI ) / 180.0;
}
```

```
if ( unirand( 0, 1 ) > exp( ( -1.0 * ( mc_energy - energy_old ) ) / ( mc_b *
```

2. **DONE** cuda code

- cuda code
- ☒ same scale as in the c++ code

  ```
  const float t = 1.0f;
  const float r = 5.0f;
  ```

- ☒ same translation vector applied

  ```
  // pocket center coord should be added here rathre than the ligand center
  coord_new->x[l] = rot[0][0] * x + rot[0][1] * y + rot[0][2] * z + r[0] + cx;
  coord_new->y[l] = rot[1][0] * x + rot[1][1] * y + rot[1][2] * z + r[1] + cy;
  coord_new->z[l] = rot[2][0] * x + rot[2][1] * y + rot[2][2] * z + r[2] + cz;
  ```

- rotation matrix applied
- calculating the ligand center
- ☒ set extremely high temperature to force the acceptance of each perturbation
  - ☒ check the translation part
    - the new center is just the old center vecotor plus the translation vector
    - if only tanslation applied
  - ☒ check the rotation part
- ☒ check by implementing a regular pattern of perturbation #+BEGIN$_{\text{SRC}}$

#if COMPILE // perturbation depends on the step number if (step%2 == 0){ if (bidx < 6) { r[bidx] = 1.0f; } if (bidx > 2 && bidx < 6) { r[bidx] = -1.0f; } if (bidx < 6) { r[bidx] += mylig->movematrix$_{\text{old[bidx]}}$; mylig->movematrix$_{\text{new[bidx]}}$ = r[bidx]; } } else { if (bidx < 6) { r[bidx] = -1.0f; } if (bidx > 2 && bidx < 6) { r[bidx] = 1.0f; } if (bidx < 6) { r[bidx] += mylig->movematrix$_{\text{old[bidx]}}$; mylig->movematrix$_{\text{new[bidx]}}$ = r[bidx]; } } #endif #+END$_{\text{SRC}}$

- ☒ the edst is not correct
  - ☒ correct the way to calculate the new center #+BEGIN$_{\text{SRC}}$

for (int i = 0; i < 3; ++i) { / *coord$_{new}$->center[i] += r[i];* / incorrect codes coord$_{\text{new}}$->center[i] = coord$_{\text{orig}}$->center[i] + mylig->movematrix$_{\text{new[i]}}$; // correct } #+END$_{\text{SRC}}$

2.2.2 DONE correct the pmf energy calculation

- energy of pmf slightly different compared with c++ codes result until r190 in svn log
- the result is different in the third digit after the desimal

1. **DONE** check the pmf calculation
   - correct c++ code #+BEGIN$_{\text{SRC}}$

   _epmf += _complex$_{\text{pmf[}}$(*ip1).getPointType()][(*il1).getAtomType()][1]

   - 1.0 / (1.0 + exp((-0.5 * dst + 6.0) * (dst - _complex$_{\text{pmf[}}$(*ip1).getPointType()][(*il1).getAt))); #+END$_{\text{SRC}}$
     - cuda code

```
const float dst_minus_pmf0 = dst - enepara_dc->pmf0[lig_t][prt_t];

epmf[bidx] +=
  enepara_dc->pmf1[lig_t][prt_t] /
  (1.0f + expf ((-0.5f * dst + 6.0f) * dst_minus_pmf0));
```

2. **DONE** correct the pmf calculation pmf calculated to be correct after using the movematrix to record the trail of the system

[1] DEFINITION NOT FOUND.
[2] DEFINITION NOT FOUND.

2.2.3 DONE figure out the coordinate ref system of the ligand, ligand center and protein center

1. **DONE** how upgraded in the serial C++ code

   (a) both ligand and prt center initialized at (0, 0, 0) coordinate

   ```
   for ( int ai = 0; ai < 3; ai++ )
   {
     _pocket_center[ai] = 0.0;
     _ligand_center[ai] = 0.0;
   }
   ```

   (b) both updated simultaneously, equal to the center of the coords loaded from the .sdf file

   ```
   for ( int i5 = 0; i5 < 3; i5++ )
    {
     _ligand_center[i5] /= (double) _lna;
     _pocket_center[i5] /= (double) _lna;
    }
   ```

   (c) ligand moved to the center-of-mass frame

   ```
   for ( int i1 = 0; i1 < _lna; i1++ )
     for ( int i5 = 0; i5 < 3; i5++ )
      tmp8[i1][i5] -= _ligand_center[i5];

    vector<CoordsLigand>::iterator i4;

    for ( i4 = _ligand_xyz.begin(); i4 < _ligand_xyz.end(); i4++ )
     (*i4).setCoords( tmp8[(*i4).getAtomNumber()][0], tmp8[(*i4).getAtomNumber()]
   ```

   (d) ligand center initialized at (0,0,0) in calculating energy

   ```
   for ( int il5 = 0; il5 < 3; il5++ )
    _ligand_center[il5] = 0.0;
   ```

   (e) rotation matrix directly applied to the ligand coord with respective to the lab

   ```
   for ( int il4 = 0; il4 < 3; il4++)
   {
    for ( int il2 = 0; il2 < 3; il2++)
    {
   ```

```
  t_xyz[il2] = 0.0;

  for ( int il3 = 0; il3 < 3; il3++)
   t_xyz[il2] += b_xyz[il3] * r_mat[il4][il3][il2];
 }

 for ( int il2 = 0; il2 < 3; il2++)
  b_xyz[il2] = t_xyz[il2];
}
```

2.2.4 DONE pocket$_{\text{center}}$ accompany the protein structure

- ☒ pocket$_{\text{center}}$ the same for all ligand conformations loaded from the same .sdf file

- ☒ pocket$_{\text{center}}$ is a proporty of each replica and be used in every energy calculation

1. **TODO** how invloved in the energy calculation and Monte Carlo

2.3 DONE move the load weight function to load.h and load.C

2.4 DONE calculation of acceptance ratio added

3 testing

3.1 TODO Replica-exchange Monte Carlo mode

mode describes the exchanging pattern of the ligand and temperature across all the replicas

3.1.1 TODO temperature exchange mode

1. **TODO** testing the parallel tempering

3.1.2 TODO which mode supposts complete information exchange

- mode0 and mode1 combined together provides a mechanism that can do a complete information

3.2 TODO search the lowest energy in the track

to find the lowest energy and the corresponding configuration in each replica

3.2.1 TODO more functions in analysis.C

3.2.2 TODO in production version, every step has to be recorded, which generate redunancy

1. because memory allocated for recording would be left with some unused space if only the accepted configuration information is recorded

2. about 9.0% performace would be lost due to recording redundancy information

1. **TODO** record

   - □ total energy
   - □ movematrix
   - □ ligand conformation and protein conformation

2. **TODO** estimate hard disk requirement

   - □ set total steps and total temperature from cmd

3.3 TODO why care about the # mcs ??

```
complexsize.n_pos = inputfiles->lhm_file.n_pos; // number of MCS positions
```

3.4 DONE simplist monte carlo implementation

3.4.1 to diagnose one replica

```
const int myreplica = 0; // the # of replica chosen to print
```

3.4.2 DONE testing single temperature Monte Carlo

- ☒ subscript in bounds in $\mathrm{accept_d}$.cu solved

- ☒ in perturbing the ligand, $\mathrm{MyRand_d}$() is always positive solved

- initialize the ligand away from the native pose, run single temperature Monte Carlo

   1. track the dst energy dst energy decreases through the process, see gpudocksm-rem-1.2/src/edst$_{\mathrm{singletempMCawaycenter}}$.pdf
   2. track the vdw energy vdw energy fluctuats, see gpudocksm-rem-1.2/src/evdw$_{\mathrm{singletempMCawaycenter}}$.pdf

- initialize the ligand at the native pose

   1. track the dst energy dst energy fluctuats at a low level, indicating the ligand moveing aournd the native pose, see gpudocksm-rem-1.2/src/edst$_{\mathrm{singletempMCatcenter}}$.pdf

3.5 DONE argument parsing

```
void
ParseArguments (int argc, char **argv, McPara * mcpara, InputFiles * inputfiles);
```

3.6 DONE modify energy calculation if needed

- weight abtained from using /home/jaydy/work/dat/output/output/FF$_{\mathrm{opt}}$/0.8.ff
- applying the linear transformation normalized$_{\mathrm{df}}$ = a*df + b

```
inputfiles->norpara_file.path_a = "../dat/linear_a";
inputfiles->norpara_file.path_b = "../dat/linear_b";
```

a:	
_evdw	0.746595
_eele	18.289225
_epmf	0.282088
_ehpc	0.427256
_ehdb	2.147791
_edst	0.497450
_epsp	0.572314
_ekde	233.329020
_elhm	0.726683
b:	
_evdw	1.036550
_eele	-0.028357
_epmf	0.256679
_ehpc	-1.023866
_ehdb	1.000000
_edst	-1.000000
_epsp	0.001993
_ekde	-1.000000
_elhm	-0.294676

- ☒ 18 more float number from normalization parameter in the device constant
- ☒ abort to optimize calculat combination due to its low cost

3.7 DONE introduce the toggle of random walk

```
mcpara->if_random = 1; // random walk by default
```

3.8 DONE load the weight from file

- old

```
mylig->etotal[mylig->track] =
  enepara_dc->w[0] * evdw[0] +
  enepara_dc->w[1] * eele[0] +
  enepara_dc->w[2] * epmf[0] +
  enepara_dc->w[3] * epsp[0] +
  enepara_dc->w[4] * ehdb[0] +
  enepara_dc->w[5] * ehpc[0] +
  enepara_dc->w[6] * ekde[0] +
  enepara_dc->w[7] * elhm[0] +
  enepara_dc->w[8] * edst;
```

- new

```
mylig->etotal[mylig->track] =
  enepara_dc->w[0] * evdw[0] +
  enepara_dc->w[1] * eele[0] +
  enepara_dc->w[2] * epmf[0] +
  enepara_dc->w[3] * ehpc[0] +
  enepara_dc->w[4] * ehdb[0] +
  enepara_dc->w[5] * edst +
  enepara_dc->w[6] * epsp[0] +
  enepara_dc->w[7] * ekde[0] +
  enepara_dc->w[8] * elhm[0];

std::string ifn = path;

list < string > data;
list < string >::iterator data_i;

string line1; // tmp string for each line
ifstream data_file(ifn.c_str()); // open the data_file as the buffer

if (!data_file.is_open()) {
cout << "cannot open " << ifn << endl;
exit(EXIT_FAILURE);
}

while (getline(data_file, line1))
data.push_back(line1); // push each line to the list

data_file.close(); // close

int total_weight_item = data.size();
int weight_iter = 0;

for (weight_iter = 0, data_i = data.begin(); weight_iter < total_weight_item && data_i
// interate the list
string s = (*data_i).substr(0, 30);
istringstream os(s);
double tmp = 0.0;
os >> tmp; // this tmp is what you need. do whatever you want with it
enepara->w[weight_iter] = tmp;
}
```

3.9 DONE What does output$_{20131205105456}$/a$_{xxxx}$.h5's xxxx stand for ???

to leave 4 digits

3.10 DONE check the temperature settings