GPU Replica Exchange Monte Carlo

Lane College, Jackson TN
Department of Mathematics
Elijah MacCarthy, Ph.D
emaccarthy@lanecollege.edu

September 1, 2020
GPUs have become very popular in scientific computing in the past few decades. Despite being originally developed for the gaming industry, it has spread its influence to numerous areas. Thus, GPGPUs are being used in astronomy, medicine, finance, mathematics and bioinformatics. GPUs are massively parallel and outperform even the best parallelized algorithms from CPUs.
CUDA is one API used mostly with GPUs.

However, we use OpenACC here because:

- OpenACC allows for preprocessor directives included in program.
- Program does not have to be completely modified as in CUDA.
- Allows for program to be implemented even without GPUs unlike CUDA programs which are solely for GPUs.
- OpenACC supports all accelerators.
There are three levels in OpenACC execution model.
- The gangs, workers and vector.
- This is supposed to map to any architecture.
Monte Carlo simulations use random numbers to model populations.

REMC is a Monte Carlo Method that involves swapping of replicas at different temperatures.

In REMC, simulations of several replicas are implemented at different temperatures, $T$.

After some Monte Carlo time step, updates are performed.
Replica Exchange Monte Carlo (REMC)

- Updates are accepted based on Metropolis criterion with probability
  - \( p \) given by:

\[
p = \min(1, \exp[-\beta(E_{\text{new}} - E_{\text{old}})]),
\]

(1)

- Attempt to exchange neighboring replicas is initiated following a number of Monte Carlo updates

- Replica exchanges are accepted based on given probabilities

\[
p = \min(1, \exp[(\beta_i - \beta_{i+1})(E_{\text{new}} - E_{\text{old}})]),
\]

(2)

- With Parallel REMC, several replicas are simulated in Parallel to reduce computational time
Schema of REMC

\(N_{\text{replica}} = 5\)

\(T_5 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2 \rightarrow T_1 = 1\)

Monte Carlo Step
Parallelization of Replica Exchanges

- REMC perform concurrent simulations of $n$ different replicas of the Monte Carlo system, each running under different temperatures.

- Systems at high temperatures are able to explore a larger volume of the phase space than at low temperatures.

- During the swapping phase, replicas are exchanged between temperatures by a stochastic process that uses Eqn. (1).

- Replica Exchange simulation by itself requires a relatively small communication between replicas, thus, each replica can run on a single processor in multi-core settings.
Parallelization of Energy Computation of Conformations

- Though this works well with small to modestly sized Monte Carlo systems, it becomes a problem with many replicas and longer simulation times.
- This necessitates high-performance computing approach and GPUs.
- Energy computations are the most expensive of the REMC process.
- Replicas are moved from state i to j and the improvement of these movements are guided by the energy functions of the system.
The energy function has two main categories:
- those based on replicas ability to satisfy distance and contact restraints,
- and the other based on physical and statistical energy scores.
- The physical energy scores include van der Waals and electrostatic potentials.
- Whereas the statistical energy scores are derived from structural databases.
Parallelization of Energy Computation of Conformations

- For each replica in the system, we parallelize the operations by invoking several OpenACC gangs for the computations.
- For each gang, several OpenACC threads are launched.
- A maximum of 1024 threads are launched per gang.
- These launched threads partition the tasks in a parallel region among themselves.
- Similarly for the energy scores, we assign each to its compute region.
The energy functions are summed for the potential energy of the system, thus, there is the need for communication of updated energy scores.

These communication of updated energy scores and the data required for the computations introduces an enormous data transfer challenge with this optimization.
Parallelization of Energy Computation of Conformations

- EHB, ESHORT, Energy$_{tot}$ calculate statistical energy component, energy based on contact and distance restraints and total energy.
Parallelization of Monte Carlo Moves/Updates

- Several Monte Carlo moves/updates are attempted in a Monte Carlo process.
- Updates at high temperatures that change the energy of the system have a higher probability of being accepted based on Eqn (1).
- A move from state $i$ to state $j$ is denoted by the transition matrix:
  \[ M_{ij} = \beta_{ij} p_{ij} \]  
  \[ (3) \]
- $\beta_{ij}$ is the probability of attempting a move between the two states and $p_{ij}$ is the probability of accepting the move, which is Eqn 1.
Parallelization of Monte Carlo Moves/Updates

- The moves change the torsional angle and any bond angles of the atom.
- We consider 2, 3, 4, 5 and 6 bond moves for the system.
- Each move is assigned a compute region, meaning several thread blocks are used.
- Some of these threads are responsible for the calculation of the change in energy, used to determine whether a move/update should be accepted or rejected.
- There is therefore a communication of energy scores from the energy regions to the moves/updates.
- After several of Monte Carlo moves/updates, replica swaps are initiated which are also accepted based on energy changes.
Specifications of the Hardware used

<table>
<thead>
<tr>
<th></th>
<th>reference CPU</th>
<th>GPU1</th>
<th>GPU2</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Intel Xeon E5-2680v3</td>
<td>Pascal P100</td>
<td>Kepler K80</td>
</tr>
<tr>
<td>Node counts</td>
<td>1944</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Cores per socket</td>
<td>12</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>RAM</td>
<td>128GB</td>
<td>128GB</td>
<td>128GB</td>
</tr>
<tr>
<td>clock speed</td>
<td>2.5GHz</td>
<td>2.4GHz</td>
<td>2.5GHz</td>
</tr>
</tbody>
</table>

- Speed-up is given by:

\[ S_p = \frac{t_{CPU}}{t_{GPU}}, \] (4)

- The runtime on the GPU and CPU are \( t_{GPU} \) and \( t_{CPU} \) respectively
Case Study of REMC Method in I-TASSER

- We consider a performance comparison of just the REMC of top performing protein structure prediction method from our comprehensive review.
- I-TASSER is top performing protein structure predictor based on our review, hence we use it for our case study.
- Used sequence of length 146, obtained Average speedup of 3.6x.

Table: Performance Comparison of Energy Computations

<table>
<thead>
<tr>
<th>Energy Computations</th>
<th>$t_{CPU}/s$</th>
<th>$t_{GPU}/s$</th>
<th>$S_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EHB</td>
<td>4458.28</td>
<td>1177.57</td>
<td>3.8</td>
</tr>
<tr>
<td>ESHORT</td>
<td>2427.56</td>
<td>709.71</td>
<td>3.4</td>
</tr>
<tr>
<td>All Energies</td>
<td>6885.84</td>
<td>1887.28</td>
<td>3.6</td>
</tr>
</tbody>
</table>
Case Study of REMC Method in I-TASSER

- We compare the moves in serial and GPU REMC

**Table: Comparing Serial and GPU Moves**

<table>
<thead>
<tr>
<th>MC Moves</th>
<th>$t_{CPU}/s$</th>
<th>$t_{GPU}/s$</th>
<th>$S_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move2</td>
<td>262.86</td>
<td>35.09</td>
<td>7.5</td>
</tr>
<tr>
<td>Move3d</td>
<td>107.56</td>
<td>11.83</td>
<td>9.1</td>
</tr>
<tr>
<td>Move3s</td>
<td>68.86</td>
<td>8.40</td>
<td>8.2</td>
</tr>
<tr>
<td>Move4d</td>
<td>60.10</td>
<td>7.09</td>
<td>8.5</td>
</tr>
<tr>
<td>Move4s</td>
<td>42.16</td>
<td>4.99</td>
<td>8.4</td>
</tr>
<tr>
<td>Move5d</td>
<td>52.97</td>
<td>6.98</td>
<td>7.6</td>
</tr>
<tr>
<td>Move5s</td>
<td>37.82</td>
<td>4.75</td>
<td>8.0</td>
</tr>
<tr>
<td>Move6</td>
<td>16.77</td>
<td>3.35</td>
<td>5.0</td>
</tr>
<tr>
<td>All Moves</td>
<td>649.1</td>
<td>79.75</td>
<td>8.1</td>
</tr>
</tbody>
</table>

- We observe a peak speedup of 9.1x with average 8.1x
We compare the time from serial and GPU versions for the energy computations.

Figure: SpeedUp.
Serial REMC Vs. GPU REMC for MC Moves

- We see execution time for the serial rising steadily while that for the GPU is contained.
We have successfully parallelized REMC method on GPUs.
We observed a peak speedup of 9.1x from the Monte Carlo moves.
An average speedup of 8.1 across the moves.
Peak speedup of 3.8 over the energy computations which is the most expensive.
Kyoung-Su and Jung, Keechul. "GPU implementation of neural networks." *Pattern Recognition* 37, no. 6 (2004): 1311-1314


Charalambous, Maria and Trancoso, Pedro and Stamatakis, Alexandros. "Initial experiences porting a bioinformatics application to a graphics processor." *Advances in Informatics* 2005: 415–425


Thank You!
Questions?