Optimization of the AIREBO Many-Body Potential for KNL

Markus Höhnerbach; Advisors: Ahmed E. Ismail, Paolo Bientinesi

Introduction

The AIREBO [1] potential provides forces and energies for molecular dynamics (MD) simulations of carbon and carbohydrate structures. As opposed to more classical approaches, it also captures the bonding behaviour and allows bonds to break and appear. We develop an optimized AIREBO implementation for Intel’s Xeon and Xeon Phi (co)processors [2] and integrate it into the open-source LAMMPS [3] molecular dynamics code. AIREBO consists of short and longer-ranged contributions that both depend on the bond-order.

Short-Ranged Contributions

\[ E_{\text{REBO}} = \sum_{i,j} V_{ij}^p(r_{ij}) + b_i V_{ij}^l(r_{ij}). \]

Bond-Order Calculation

\[
\beta_i = \left[ 1 + \sum_k N_k N_k^\alpha \right]^{-1/2} \rho^p(i, \alpha, \omega, \beta) \]

\[
\rho^p = \frac{1}{2} \left( \sum_k N_k N_k^\alpha \right) + \rho^p(i, \alpha, \omega), \]

\[ N_\alpha = \text{No. neighbors of } \alpha \text{ besides } \beta \]

\[ N^\text{rev} = 1 + \sum_k \rho^p(i, \alpha, \beta) \]

Intermediate Neighbor List

\[ \ell \rightarrow i \rightarrow j \]

Best neighbor chain for \( C_l \)

\[ E_{\text{LJ}} = \sum_{i,j} \left[ \epsilon \left( \sigma \left( |\mathbf{r}_{ij}| - \sigma \right) \right)^{12} \right]^{1/6} \]

\[ C_l = 1 - \max_k \left( w_u(r_{ik}) w_u(r_{jk}) w_l(r_{ij}) \right) \]

Optimization Challenges

- Need 2 neighbor lists: Short-ranged & longer-ranged.
- Due to the short-ranged nature of \( E_{\text{REBO}}, E_{\text{TORSION}}, \) and \( C_l \), the vast majority of loops are “too short” to vectorize. Need to use the outermost loop (over all atoms in the simulation).
- While most time consumed in \( E_{\text{LJ}} \), remaining routines become significant after its optimization.

Techniques

- Integrating TORSION into REBO calculation.
- Lennard-Jones search using hashmap.
- AVX/AVX2: Optimize gather to transpose.
- Add lower precision modes.
- Automatic translation from intrinsics to library.

Runtime Performance before Optimization

<table>
<thead>
<tr>
<th>N-list</th>
<th>N-lists</th>
<th>( E_{\text{REBO}} )</th>
<th>( E_{\text{LJ}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.35</td>
<td>23.55</td>
<td>17.77</td>
<td>175.06</td>
</tr>
</tbody>
</table>

Runtime Performance after Optimization

<table>
<thead>
<tr>
<th>N-list</th>
<th>N-lists</th>
<th>( E_{\text{REBO}} )</th>
<th>( E_{\text{LJ}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.63</td>
<td>10.64</td>
<td>10.83</td>
<td>32.65</td>
</tr>
</tbody>
</table>

Longer-Ranged Contributions

\[ E_{\text{LJ}} = \sum_{i,j} \left[ \epsilon \left( \sigma \left( |\mathbf{r}_{ij}| - \sigma \right) \right)^{12} \right]^{1/6} \]

\[ C_l = 1 - \max_k \left( w_u(r_{ik}) w_u(r_{jk}) w_l(r_{ij}) \right) \]

Bond-Order Vectorization

For \( i \), \( j \):
- Search neighbors.
- Insert candidates into hashmap.
- Reuse longer neighbor list, short neighbor list.
- distance check for \( C_l \)
- Hashmap lookup

Results

- Performance Comparison
- Speedups (Third-party, [4])

References