Performance Evaluation of Multi-Threaded Granular Force Kernels in LIGGGHTS

Richard Berger
Why add threading optimizations?

• Domain decomposition not enough for load-balancing
Transfer chute example

high particle density

low particle density
Why add threading optimizations?

• Domain decomposition not enough for load-balancing

• Shared memory programming gives you more control

• With MPI you have to rely on individual implementations (OpenMPI, MPICH2)

• More optimization potential with shared memory programming (e.g. cache efficiency)

• A hybrid approach would give us the best of both worlds.
Starting Point: MiniMD

• **LIGGGHTS**
  – Based on LAMMPS
  – ~280,000 LOC
  – Optimizing this code base is hard

• **MiniMD-granular**
  – Based on MiniMD, which is a light-weight benchmark of LAMMPS
  – ~3,800 LOC
  – Makes it much easier to test new ideas and optimize critical parts

• **What was done in OpenMP:**
  o Pair Styles (pair_gran_hooke)
  o Neighbor List
  o Integration
  o Primitive Walls
## Atom decomposition

### OpenMP static schedule

### Force array

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
</tr>
<tr>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
</tr>
<tr>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
</tr>
<tr>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
</tr>
</tbody>
</table>

Each box represents the force calculated for one particle.

- **Thread 0**: Yellow
- **Thread 1**: Green
- **Thread 2**: Orange
- **Thread 3**: Blue
**Data Race:**
Access the same memory location, at least one thread writes

**Write Conflict:**
Two threads try to update force of the same particle
Sources of Data Races

- **Newton’s 3rd Law (Actio = Reactio, always used in LIGGGHTS):**
  - Pair Forces of local particles only computed once, applied to both contact partners

- **Ghost Particles**
  - Pair Forces are only computed once at Process Boundaries
  - Multiple threads could try adding contributions to a single ghost particle

- **Global Accumulators:**
  - Compute (Energy, Virial)
Boxfill example
Load balancing

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>

- **Thread 0**
- **Thread 1**
- **Thread 2**
- **Thread 3**
Load balancing
Visualization of the workload (serial run)

13,000 particles  67,000 particles
Load balancing
Visualization of the workload (OpenMP run)

13,000 particles
67,000 particles
Load balancing
Optimized Access Pattern

13,000 particles

67,000 particles
OpenMP Results (miniMD-granular)
Newton 3rd law not used

13k Particles, OpenMP 2 threads vs. MPI 2 procs, , Newton OFF

Runtime in seconds

Improvements with 2 OMP threads

2 MPI procs
OpenMP Results (miniMD-granular)
Newton 3rd law not used

13k Particles, OpenMP 2 threads vs. MPI 2 procs, Newton OFF

Runtime in seconds

OMP+Load balancing of Pair Forces
OpenMP Results (miniMD-granular)
Newton 3rd law not used

13k Particles, OpenMP 2 threads vs. MPI 2 procs, Newton OFF

Runtime in seconds

- **OMP+Load balancing of Wall-Particle Forces**

- **Other**: Other processes
- **Integ**: Integration
- **Comm**: Communication
- **Neigh**: Neighbors
- **Force Wall**: Forces against wall
- **Force Pair**: Pair forces

Christian Doppler Laboratory on Particulate Flow Modelling | www.particulate-flow.at
OpenMP Results (miniMD-granular)
Newton 3rd law not used

13k Particles, OpenMP 2 threads vs. MPI 2 procs, , Newton OFF

Runtime in seconds

OMP+Load balancing of Neighbor Lists
OpenMP Results (miniMD-granular)
Newton 3rd law not used

13k Particles, OpenMP 4 threads vs. MPI 4 procs, Newton OFF

Runtime in seconds
0 0,5 1 1,5 2 2,5 3 3,5 4 4,5 5

- Other
- Integ
- Comm
- Neigh
- Force Wall
- Force Pair

OMP 1  OMP 4  OMP+LB 4  OMP+LB+WOpt 4  OMP+LB+WOpt+NOpt 4  OMP+LB+WOpt+NOpt+Sort 4  MPI 4
OpenMP Results (miniMD-granular)
Newton 3rd law not used

13k Particles, OpenMP 4 threads vs. MPI 4 procs, Newton OFF

Runtime in seconds

OMP 1  |  OMP 4  |  OMP+LB 4  |  OMP+LB+WOpt 4  |  OMP+LB+WOpt+NOpt 4  |  OMP+LB+WOpt+NOpt

Other  |  Integ  |  Comm  |  Neigh  |  Force Wall  |  Force Pair

MPI Communication Penalty
MiniMD -> LIGGGHTS

- MiniMD was a good start

- But threading optimizations in LIGGGHTS require more effort

- LAMMPS has OpenMP support (by Axel Kohlmeyer), uses Array Reduction

- In its current form the only way to add OpenMP support to LIGGGHTS is by code duplication

- Custom Locks instead of Array Reduction

- New features were added to allow detailed timings

- Load balancing
LIGGGHTS Results
Testcase 1 – 13k particles, MPI 4 vs OpenMP 4

The diagram illustrates the runtime in seconds for different parallelization methods: serial, MPI 4, and OpenMP 4. The bar chart breaks down the runtime into various components: Other, Output, Comm, Neigh, Modify, and Force Pair. The chart shows that the serial method has a significantly higher runtime compared to the parallel methods, indicating potential inefficiencies or overhead in the serial implementation.
LIGGGHTS Results
Testcase 1 – 67k particles, MPI 4 vs OpenMP 4

saves 2 out of 21 minutes

~ 10% improvement

runtime in seconds

serial
mpi 4
omp 4

Other
Output
Comm
Neigh
Modify
Force Pair
Outlook

• Currently working on LIGGGHTS 3.x

• OpenMP support should be much simpler

• Bringing OpenMP to more code paths (e.g. insertion of particles)

• Reaching feature parity

• Performance evaluation on bigger testcases from industrial partners
Thank you for your attention!