Beyond Pair Potential: A CUDA implementation of REBO Potential

Przemysław Trędak

Faculty of Physics, University of Warsaw

GTC 2015, March 19, 2015

Naïve approach to parallelizing MD potentials

```
for all $i$ in atoms do in parallel
    for all $(j, k, \ldots)$ in atoms interacting with $i$ do
        compute forces acting on atom $i$
    end for;
end parallel for;
```

- Very simple approach
- 1 thread per atom

Naïve approach to parallelizing MD potentials

- For 2-body potentials it works reasonably well!

```
for all $i$ in atoms do in parallel
    for all $j$ in atoms interacting with $i$ do //2-body

                compute forces acting on atom $i$

    end for;
end parallel for;
```

Naïve approach to parallelizing MD potentials

- For 2-body potentials it works reasonably well!
- For 3-body and more complicated potentials not so much:

```
for all $i$ in atoms do in parallel
    for all $j$ in atoms interacting with $i$ do //2-body
        for all $k$ in atoms interacting with $i$ do //3-body
            ⋱
                compute forces acting on atom $i$
        end for;
    end for;
end parallel for;
```

Different many-body potentials

$$E = \sum_{i}^{M} V_N(\ldots)$$

- Bonded interactions: N, M - constant
- Nonbonded N-body interactions: N - constant, M - variable
- "Real" many-body potentials: N, M - variable $\leftarrow$ focus of this talk

REBO potential

- 2nd generation Brenner potential
- Used for simulation of hydrocarbons
- Many-body potential

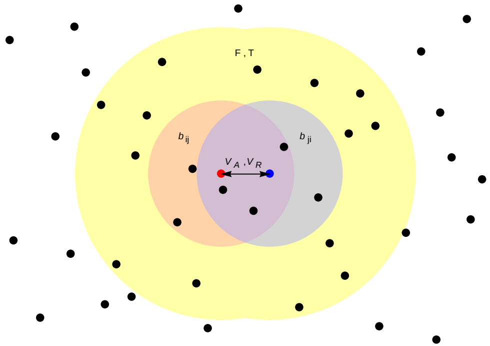
Form of REBO potential

$$E = \sum_i \sum_{j>i} \left[V_R \left(r_{ij} \right) - \bar{b}_{ij} V_A \left(r_{ij} \right) \right]$$

- V_R and V_A are simple two body terms
- Difficulty hidden in $\bar{b}_{ij}$ term

Challenges in parallel implementation

- Effective impact of a single interaction



- Complexity of the computation of interaction (3D cubic splines)

Design decisions and assumptions

- During one kernel write only to nearest neighbors - need to split work into several steps
- Use neighbor lists for nearest neighbors
- No atomic operations during force computation - better to use more memory
- Small number of nearest neighbors - during normal simulation no more than 16

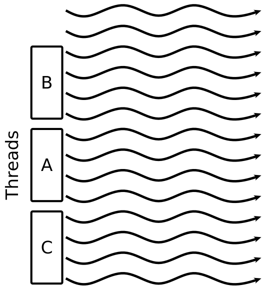
Impact of GPU architecture

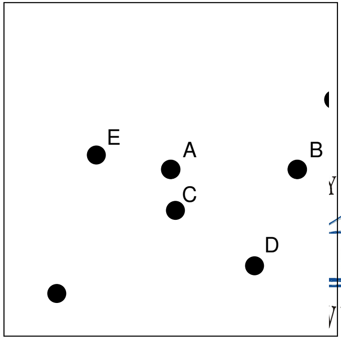
- CUDA GPUs employ SIMT (Single Instruction Multiple Threads) architecture
- 1 warp of threads executes in lockstep
- Starting with Kepler (SM 3.0) - instructions available (`__shfl`) to share data inside a warp
- Easy to logically split a single warp into several pieces of size 2^n

Proposed algorithm

Let N - maximum number of nearest neighbors of any atom rounded up to nearest power of 2.

- Every N threads are grouped to work on interactions of a single atom i

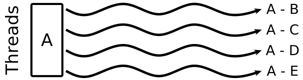


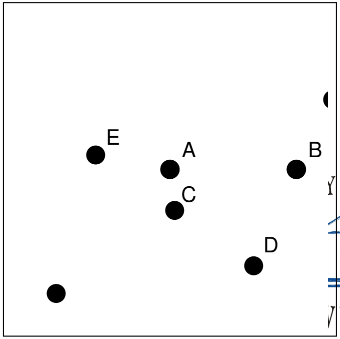


Proposed algorithm

Let N - maximum number of nearest neighbors of any atom rounded up to nearest power of 2.

- Every thread j from a group in parallel computes interaction between i and j

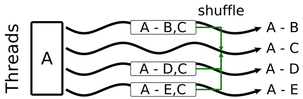


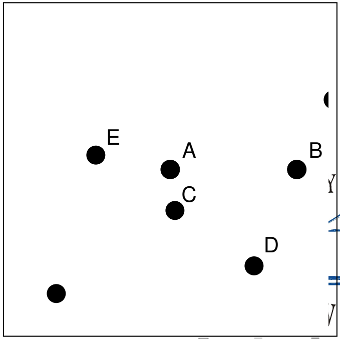


Proposed algorithm

Let N - maximum number of nearest neighbors of any atom rounded up to nearest power of 2.

- During this computation all of the forces acting on atom $k \neq i, j$ are being sent using shuffle instructions to appropriate thread from the group





Challenges

High divergence of threads if number of neighbors is less than N

- When real number of neighbors is less than N, some threads in a group are idle

Challenges

High divergence of threads if number of neighbors is less than N

- When real number of neighbors is less than N, some threads in a group are idle

Solution

- During neighbor list creation atoms are divided into groups with the same nearest neighbor count
- Kernels are templated, so that for every group the lowest N is used
- Nearest neighbors count for most atoms is ≤ 4 - minimum efficiency is 75%

Challenges

High amount of GPU memory used to avoid atomic operations

- Maximum number of atoms per K20 GPU (5 GB of RAM) - 2.5M atoms

Challenges

High amount of GPU memory used to avoid atomic operations

- Maximum number of atoms per K20 GPU (5 GB of RAM) - 2.5M atoms

Analysis

- With this many atoms, achieved performance would be 0.5 ns/day
- For real simulations, desired performance is higher - size of the system achievable on 1 GPU is not limiting
- Other GPUs have much more RAM

Challenges

Very high register pressure and local memory spilling

- Due to complexity of the main kernel, even 128 registers per thread is not enough to avoid spilling
- Limited occupancy with 256 registers per thread hurts performance

Challenges

Very high register pressure and local memory spilling

- Due to complexity of the main kernel, even 128 registers per thread is not enough to avoid spilling
- Limited occupancy with 256 registers per thread hurts performance

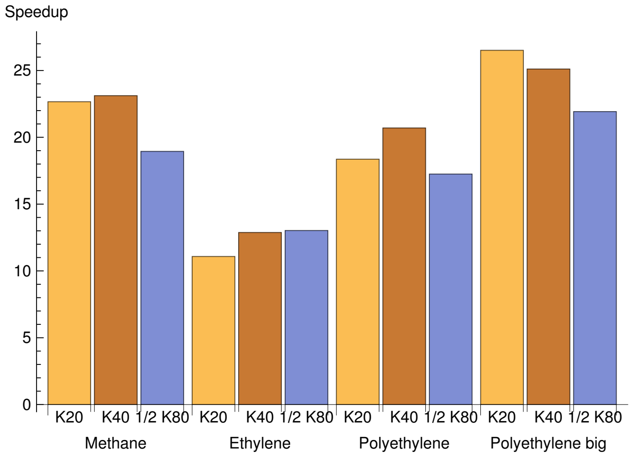
Solution

- Careful optimizations to reduce register pressure
- Spline computation in separate kernels
- Tesla K80

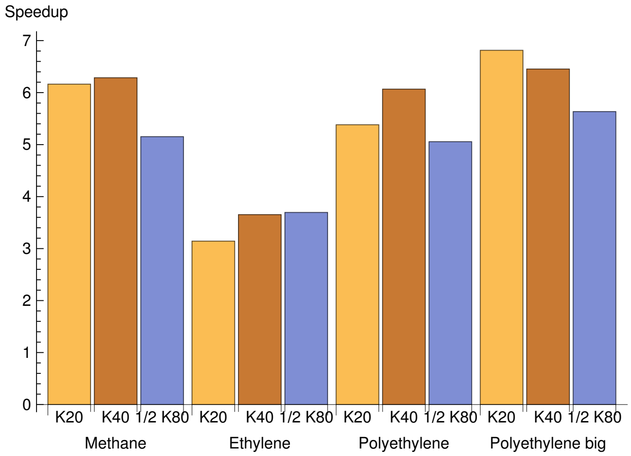
Performance tests

- CPU version - OpenMP implementation of REBO in LAMMPS, Intel Core i7-4930K 3.40 GHz (Ivy Bridge-E)
- GPU version - custom code,
  - NVIDIA Tesla K20 GPU, Intel Xeon E5620 2.4 GHz (Westmere)
  - NVIDIA Tesla K40 GPU, default clocks, Intel Xeon E5-2690 v2 3.0 GHz (Ivy Bridge-EP)
  - $\frac{1}{2}$ NVIDIA Tesla K80 GPU, default clocks, Intel Xeon E5-2650 v3 2.3 GHz (Haswell-EP)
- Tests:
  - Methane gas (625000 atoms)
  - Ethylene gas (768000 atoms)
  - Polyethylene (32640 atoms)
  - Polyethylene (587520 atoms)

Speedup over 1 CPU core



Speedup over full node



Conclusions and future work

Conclusions

- Getting advantage of SIMT architecture enables efficient algorithm for many-body REBO potential
- GPU version of REBO potential achieves great speedup over optimized CPU code

Future work

- Reducing performance impact of data movement between CPU and GPU
- Open source the code

Thank you

Questions?

You can contact me at przemyslaw.tredak@fuw.edu.pl

Please complete the Presenter Evaluation sent to you by email or through the GTC Mobile App. Your feedback is important!