IPCC @ RWTH Aachen University

Optimization of multibody and long-range solvers in LAMMPS

Paolo Bientinesi Rodrigo Canales
Markus Höhnerbach Ahmed E. Ismail

First year showcase
February 23rd, 2016

High Performance and Automatic Computing

RWTH AACHEN UNIVERSITY

Team

RWTH

Prof. Paolo Bientinesi

Rodrigo Canales

Markus Höhnerbach

Prof. Ahmed Ismail

Intel

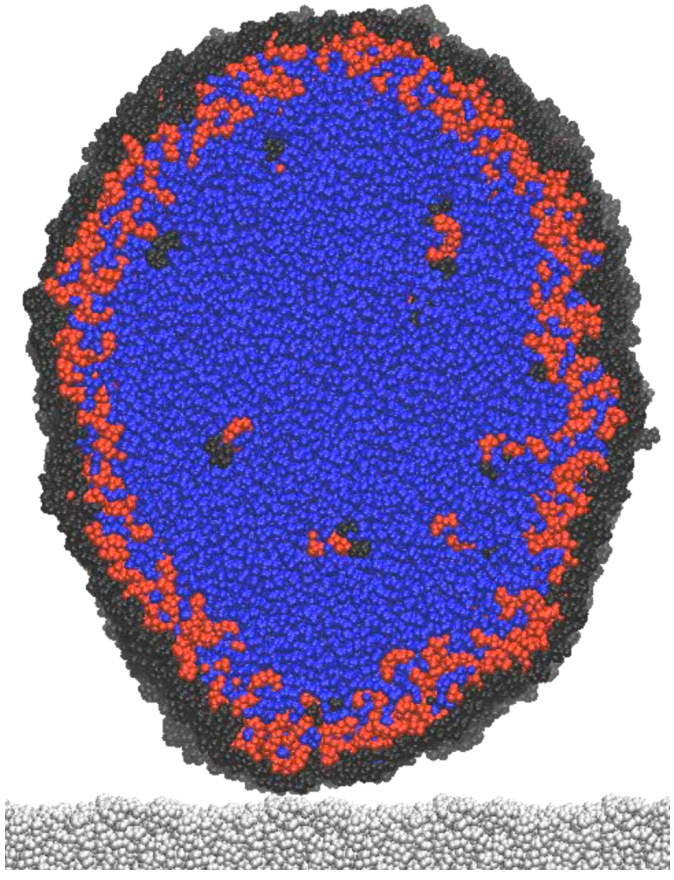
Georg Zitzlsberger

Klaus-Dieter Örtel

Michael W. Brown

LAMMPS

Large-scale **A**tomic–**M**olecular **M**assively **P**arallel **S**imulator



- Sandia National Labs
  `http://lammps.sandia.gov`
- Wide collection of potentials
- Open source
  Support for OpenMP, Xeon Phi, and GPU (CUDA and OpenCL)

Goals

- Optimize core kernels within LAMMPS
  - Multi-threading and **vectorization**
  - Intel® Xeon Phi™

- **Buckingham** potential, (P3M solver) ⇒ Rodrigo

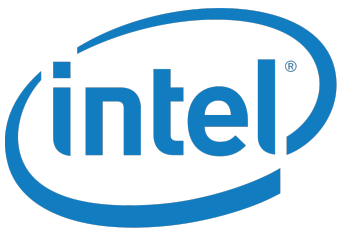
- **Tersoff** potential, (AIREBO potential) ⇒ Markus

Intel® Parallel Computing Center @ RWTH Aachen University

Buckingham Potential Optimization

Rodrigo Canales
RWTH Aachen University

February 23, 2016



PARALLEL PACKAGES FOR LAMMPS

- LAMMPS was designed to be run on a computer cluster. By default it divides the problem among processes using MPI.
- Additional parallelization packages have been created for different architectures
  - USER OpenMP
  - GPU
  - Kokkos
  - USER Intel

USER INTEL PACKAGE

- Developed by Michael Brown (Intel®)
- Adds offloading support for Xeon Phi coprocessors
- Gives the option of using multiple precision
- Includes several potentials optimized for Intel®architectures.

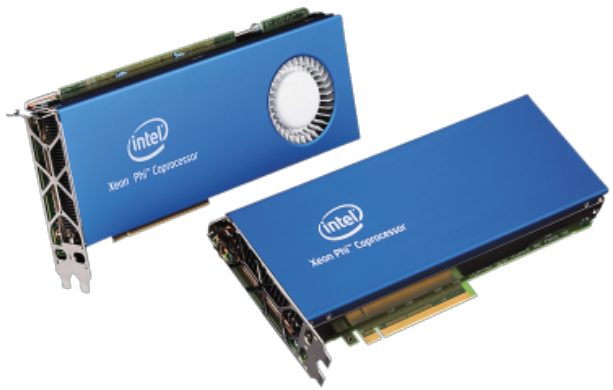


Figure : Xeon Phi coprocessors (source: Intel)

PAIR POTENTIALS

Lennard Jones

Available in USER-INTEL

$$\Phi_{lj} = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right]$$

Buckingham

$$\Phi_{buck} = Ae^{-r/\rho} - \frac{C}{r^6}$$

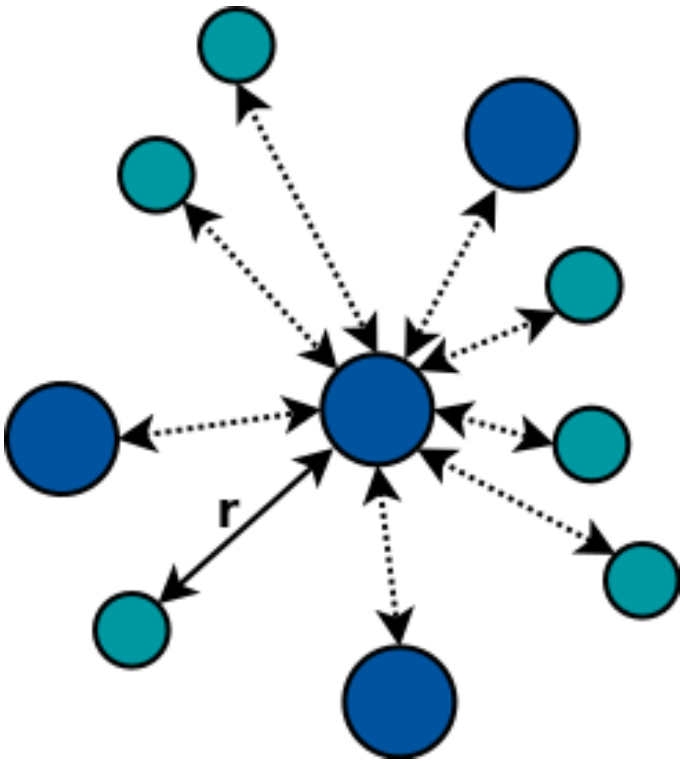
- buck/cut
- buck/coul/cut
- buck/coul/long

$$\Phi_{buck/coul} = \Phi_{buck} + \frac{Cq_iq_j}{\epsilon r_{ij}}$$

BUCK POTENTIAL OPTIMIZATION

- USER-INTEL package as base of the development
  - Data Packing for parameters
  - Alignment of force and position arrays
  - Multiple precision support
  - Enable Xeon Phi Offloading
  - Vectorization

VECTORIZATION



- For each atom calculate the distance (and forces) to each of the neighbor atoms (2 loops)
- Goal: Calculate more than one force simultaneously
- Vectorization:
  - Compiler assisted: SIMD pragmas.
  - Intrinsics

SIMD PRAGMA VECTORIZATION

- First vectorization attempt: SIMD pragmas
- Inner loop vectorization
- Multi-precision Force and energy accumulators.

TEST PLATFORM

- The Baseline comparison is the USER_OMP package.
- Three types of Buckingham potentials.
- Tested on Xeon Phi (native) and on Xeon Processor for both single and double precision.
- Speedup calculated for 1 thread
- Computing node setup:

Processor (host):	Intel Xeon E-2650 *Sandy Bridge*
Coprocessor:	Intel Xeon Phi 5110p

SPEEDUP XEON PHI

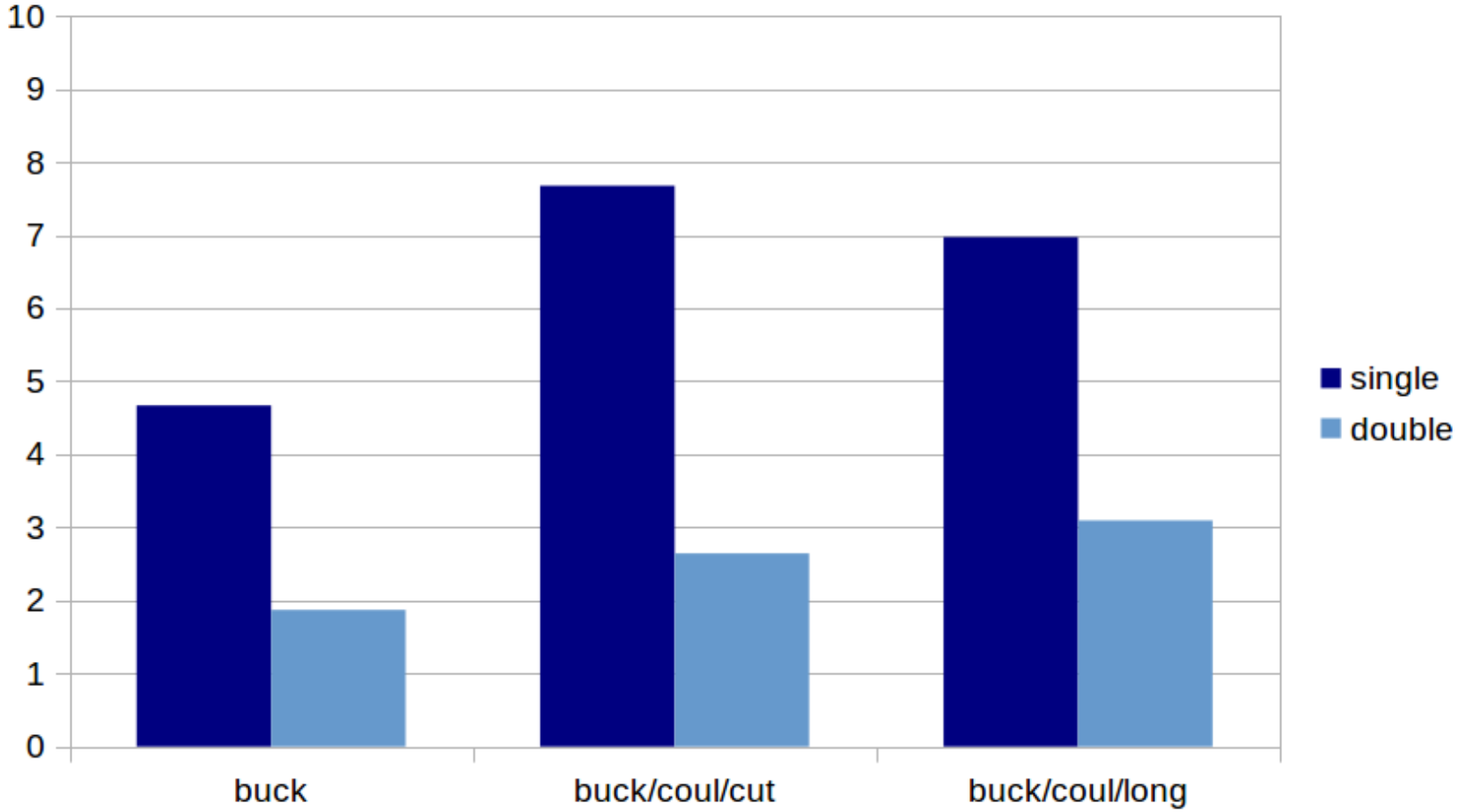


Figure : Speedup on Xeon Phi 5110p

SPEEDUP XEON

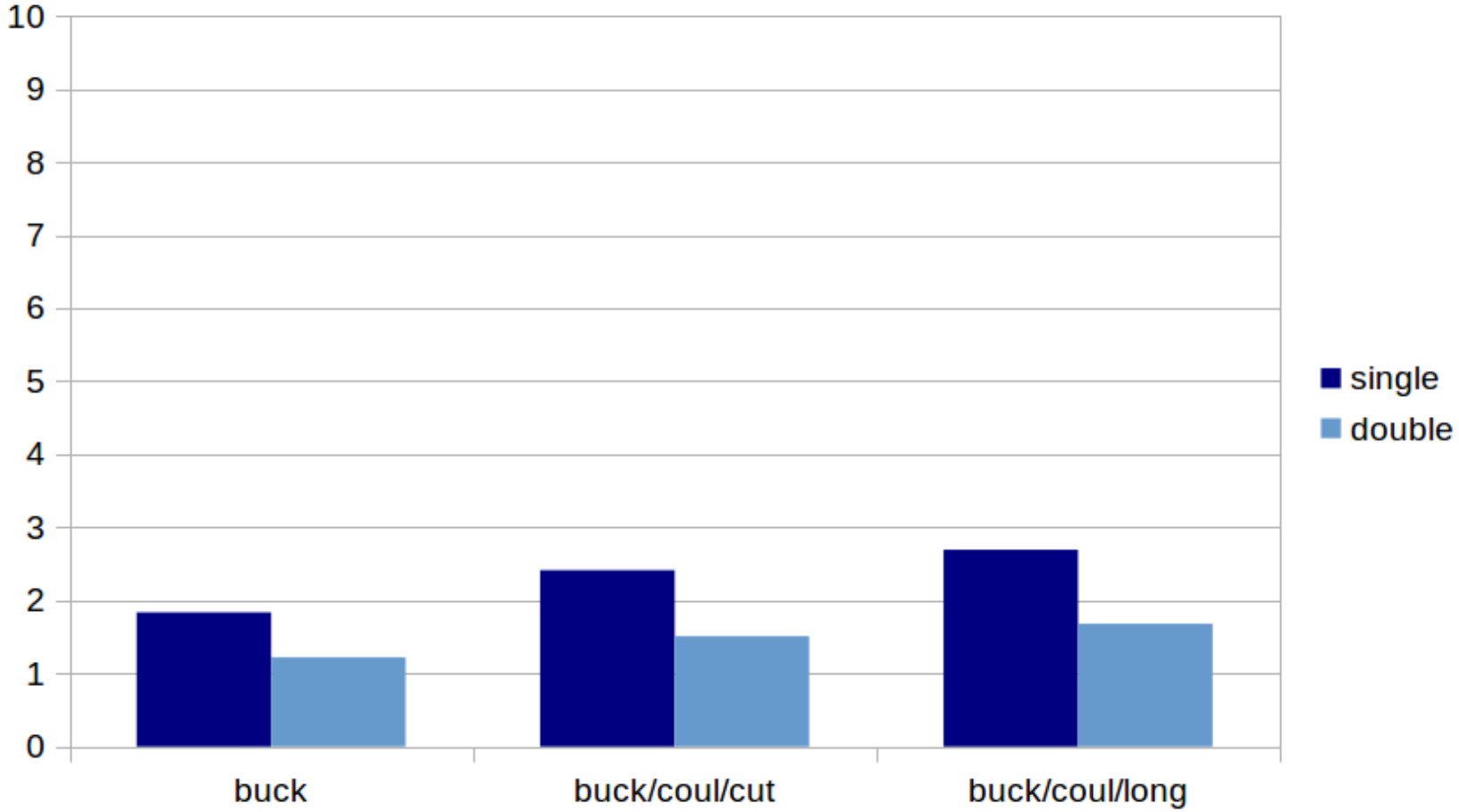


Figure : Speedup on the Xeon E5-2650 (Sandy Bridge)

KNC INTRINSICS VECTORIZATION

- Motivation: Optimize neighbor loading
- Manually Gather - Swizzle
- C++ Templates to allow multiple precision

Summary of changes

Templating intrinsics:	760 lines
Implementation pair/buck:	330 lines

KNC INTRINSICS VS SIMD PRAGMAS

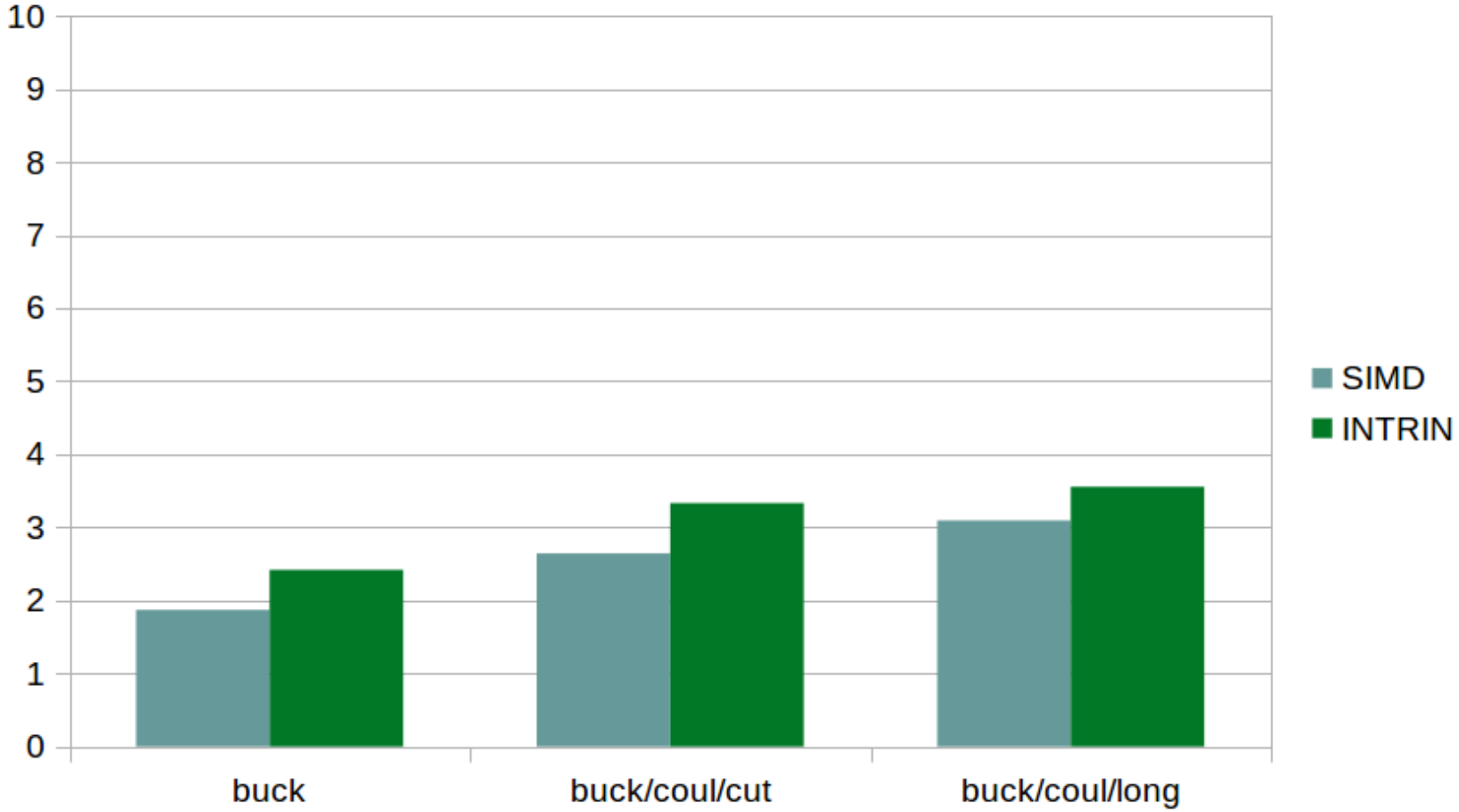


Figure : Speedup comparison on the Xeon Phi (Double)

KNC INTRINSICS VS SIMD PRAGMAS

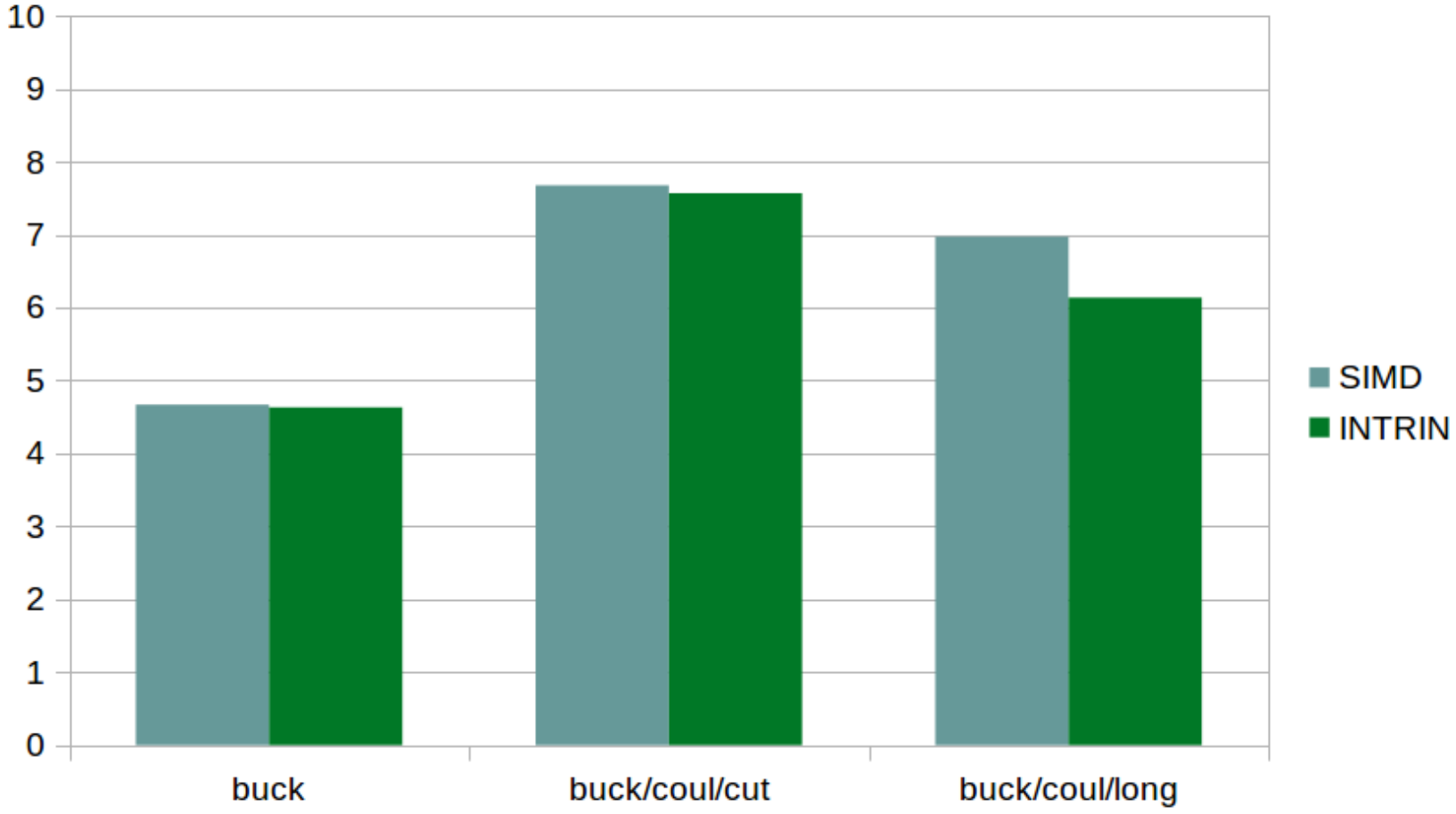


Figure : Speedup comparison on the Xeon Phi (Single)

Runtime on Full System

Buckingham potential for 576000 atoms, double precision

Xeon Phi Native	Base	252.5s
(240 Threads)	Simd	201.5s
	KNC Intrinsics	203.1s
Xeon (×2) + Xeon Phi	Base	119.7s
(32 + 240 Threads)	Simd	76.7s
	KNC Intrinsics	74.4s

CONCLUSIONS

- We optimized the Buckingham potential for the KNC and the Xeon architectures.
- We proved that the compiler does a good job on assisted vectorization.
- Source code is already integrated into LAMMPS development branch.

Current and Future Work

- Optimizing the P3M electrostatics long range solver.
  - Prototyping using SIMD pragmas and vector functions.
  - Developing towards native mode execution.
- Next goal: Optimization of the P3M dispersion solver.

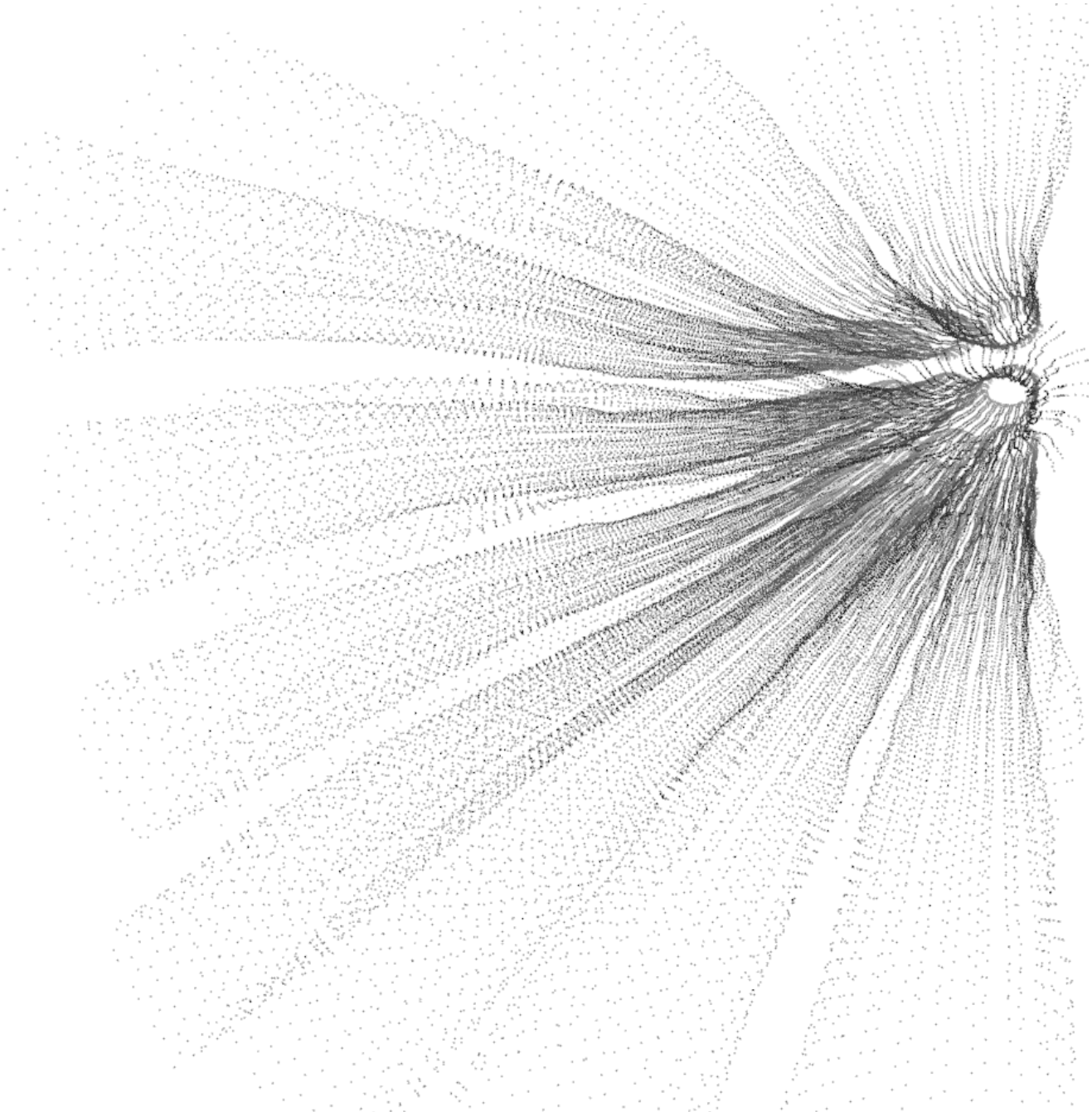
IPCC Workitem: Multibody potentials Modernization of the Tersoff potential and the current status of the AIREBO potential

Markus Höhnerbach

Aachen Institute for Advanced Study in Computational Engineering Science
RWTH Aachen University

February 23, 2016

Molecular Dynamics



The Tersoff potential

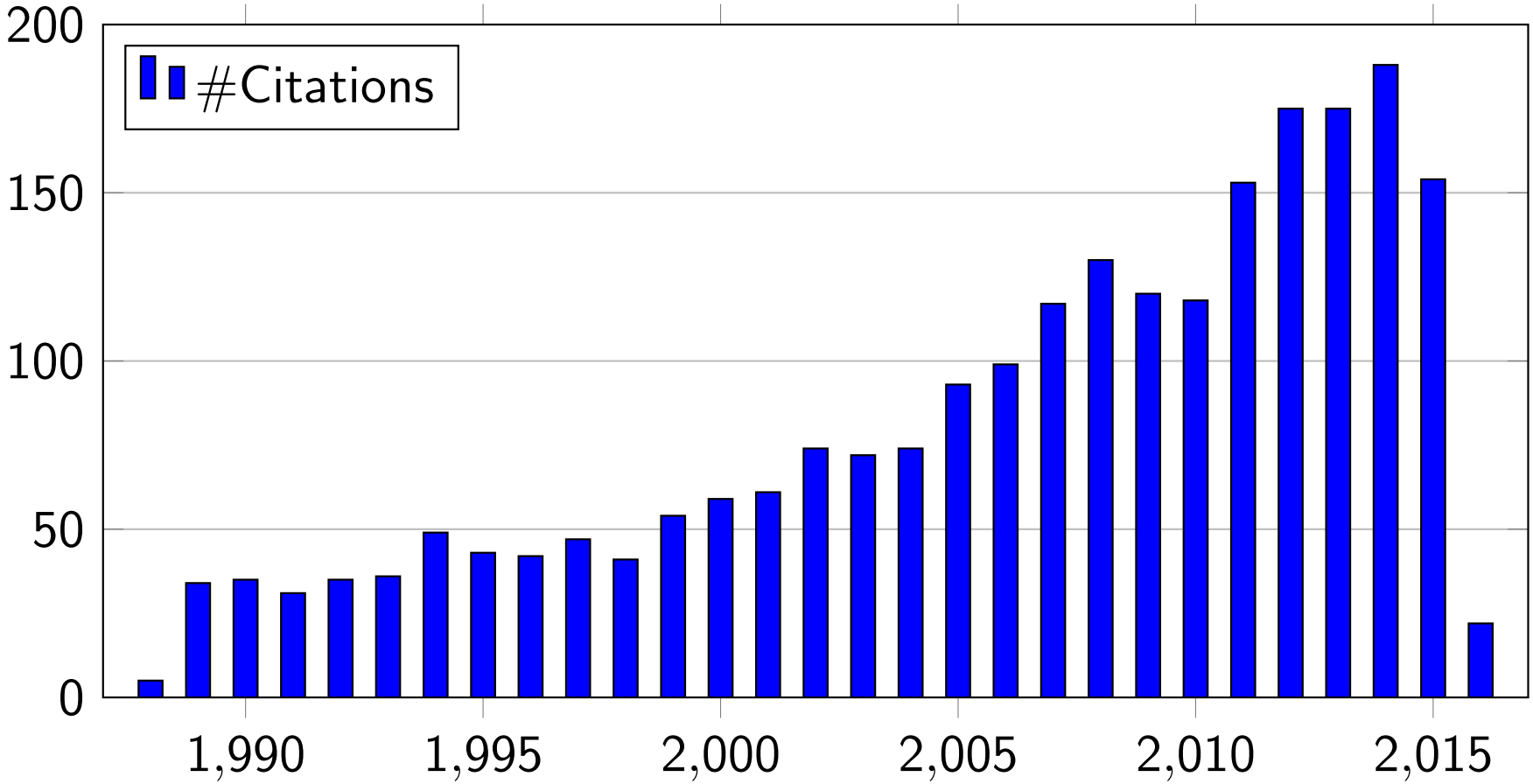
$$V = \sum_i \sum_{j \in \mathcal{N}_i} \overbrace{f_C(r_{ij}) \left[f_R(r_{ij}) + \mathbf{b_{ij}} f_A(r_{ij}) \right]}^{V(i,j,\zeta_{ij})} \tag{1}$$

$$b_{ij} = (1 + \beta^\eta \zeta_{ij}^\eta)^{-\frac{1}{2\eta}} \tag{2}$$

$$\zeta_{ij} = \sum_{k \in \mathcal{N}_i \setminus \{j\}} \underbrace{f_C(r_{ik}) g(\theta_{ijk}) \exp(\lambda_3 (r_{ij} - r_{ik}))}_{\zeta(i,j,k)} \tag{3}$$

- Terms in V and b_{ij} depend on the type of i and j
- Terms in ζ_{ij} depend on the type of i, j and k

Popularity

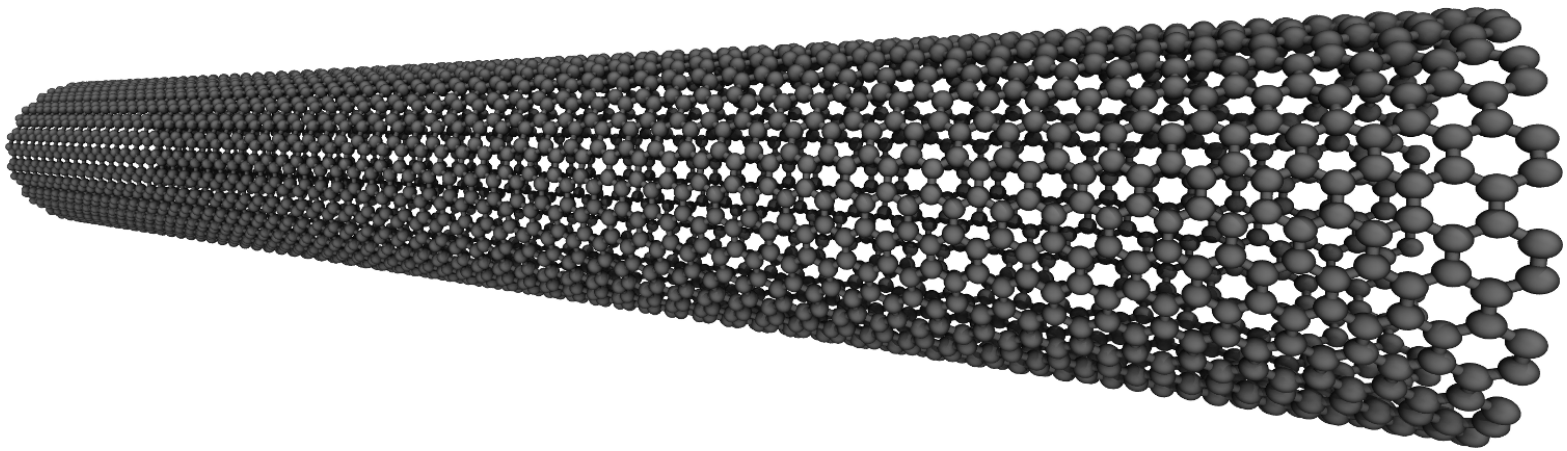


- Tersoff potential: Widely used, fairly simple (~700 LOC)
- Previous work for GPU: EAM[a], Stillinger-Weber[b] and Tersoff [c]

The Tersoff Algorithm

for *i in local atoms of the current thread* **do**
  for *j in atoms neighboring i* **do**
    $\zeta_{ij} \leftarrow 0;$
    for *k in atoms neighboring i* **do**
      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i,j,k);$
    $E \leftarrow E + V(i,j,\zeta_{ij});$
    $F_i \leftarrow F_i - \partial_{x_i} V(i,j,\zeta_{ij});$
    $F_j \leftarrow F_j - \partial_{x_j} V(i,j,\zeta_{ij});$
    $\delta\zeta \leftarrow \partial_{\zeta} V(i,j,\zeta_{ij});$
    for *k in atoms neighboring i* **do**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k);$
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k);$
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$

Close-Up



Challenges

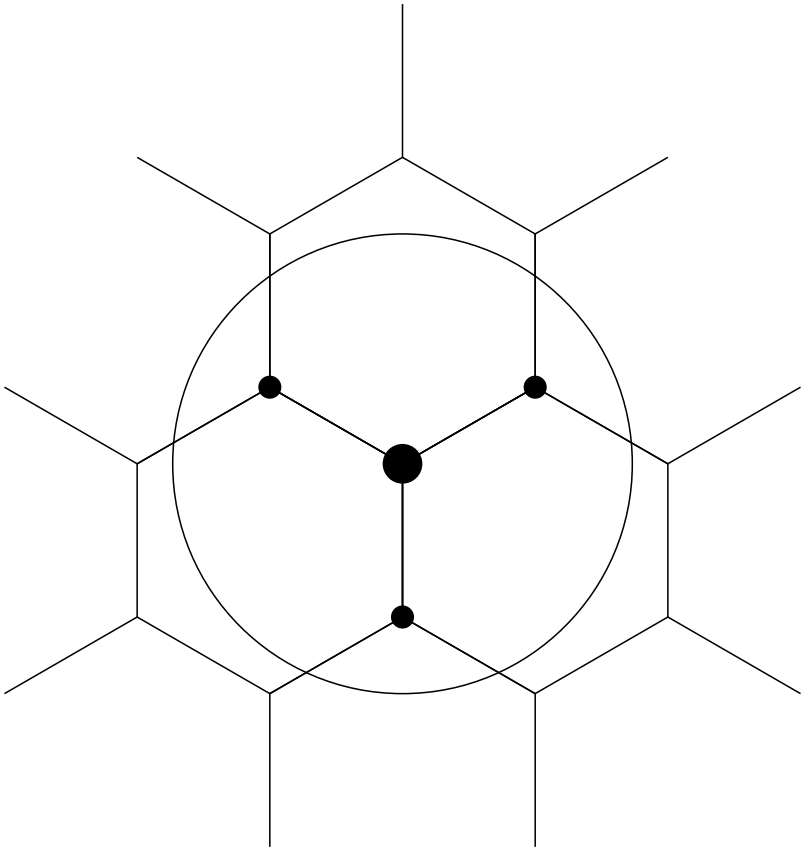


Figure : Graphene

- Few neighbors
- Fewer interactions

Model Problem: CNT

Stress in Carbon Nanotubes[a]
For single core measurements:
Scaled down 100x and simplified

Model Problem: Si

Bulk silicon
Shipped with LAMMPS

[a] Thanks to Marcus Schmidt

Vectorization

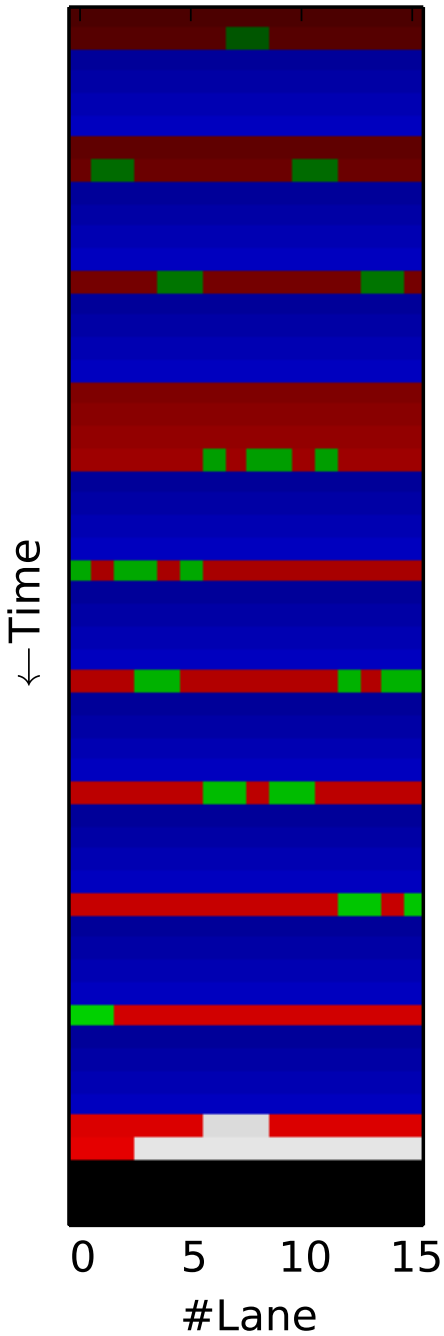
"J" algorithm

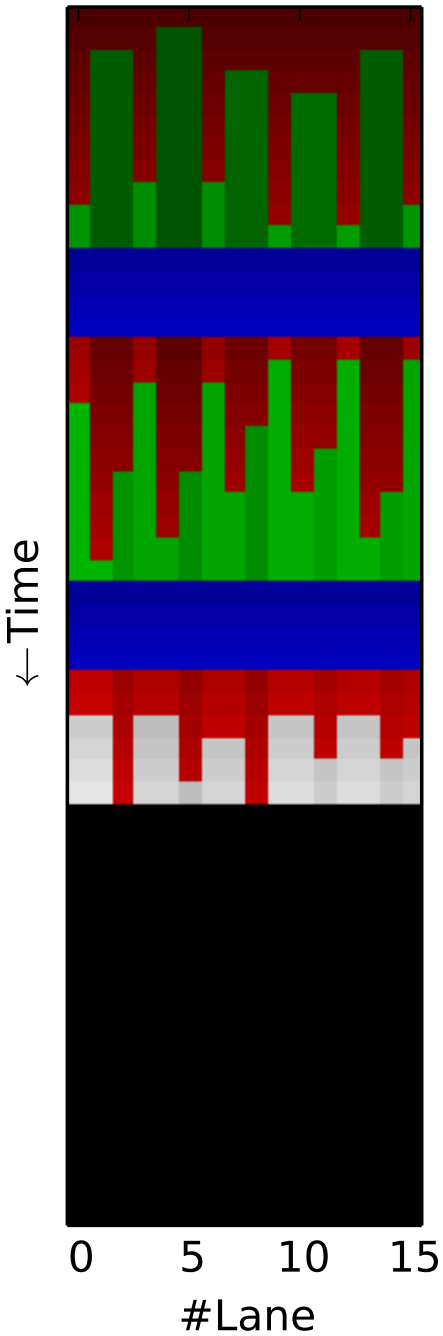
for i **do**
  for $j \in \mathcal{N}_i$ **do**
    skip cutoff;
    ...;
    for $k \in \mathcal{N}_i \setminus \{j\}$ **do**
      skip cutoff;
      ...;
    ...;
    for $k \in \mathcal{N}_i \setminus \{j\}$ **do**
      skip cutoff;
      ...;

"I" algorithm

for i **do**
  for $j \in \mathcal{N}_i$ **do**
    skip cutoff;
    ...;
    for $k \in \mathcal{N}_i \setminus \{j\}$ **do**
      skip cutoff;
      ...;
    ...;
    for $k \in \mathcal{N}_i \setminus \{j\}$ **do**
      skip cutoff;
      ...;

K Loop





Abstraction

```
typedef vector_routines<double, double, AVX> v;
typedef v::fvec fvec;
fvec a(1);
fvec b(2);
fvec c = v::recip(a + b);
```

Features

- Supports single, double and mixed precision
- Supports scalar, SSE4.2, AVX, AVX2, IMCI, AVX-512, array notation (Cilk)

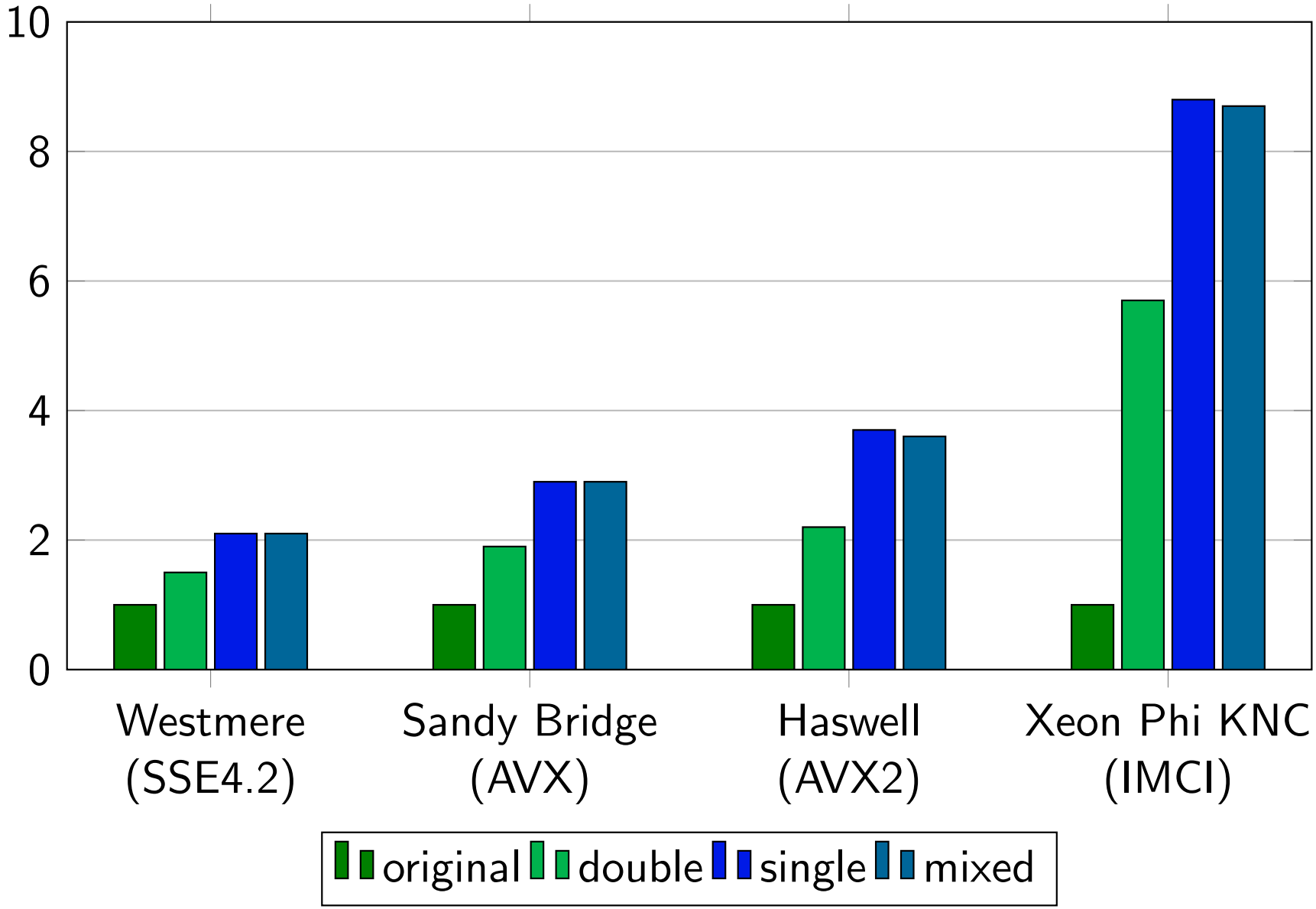
Advantages

- Maintainability
- Testing (through AN)
- Portability
- Thin wrapper

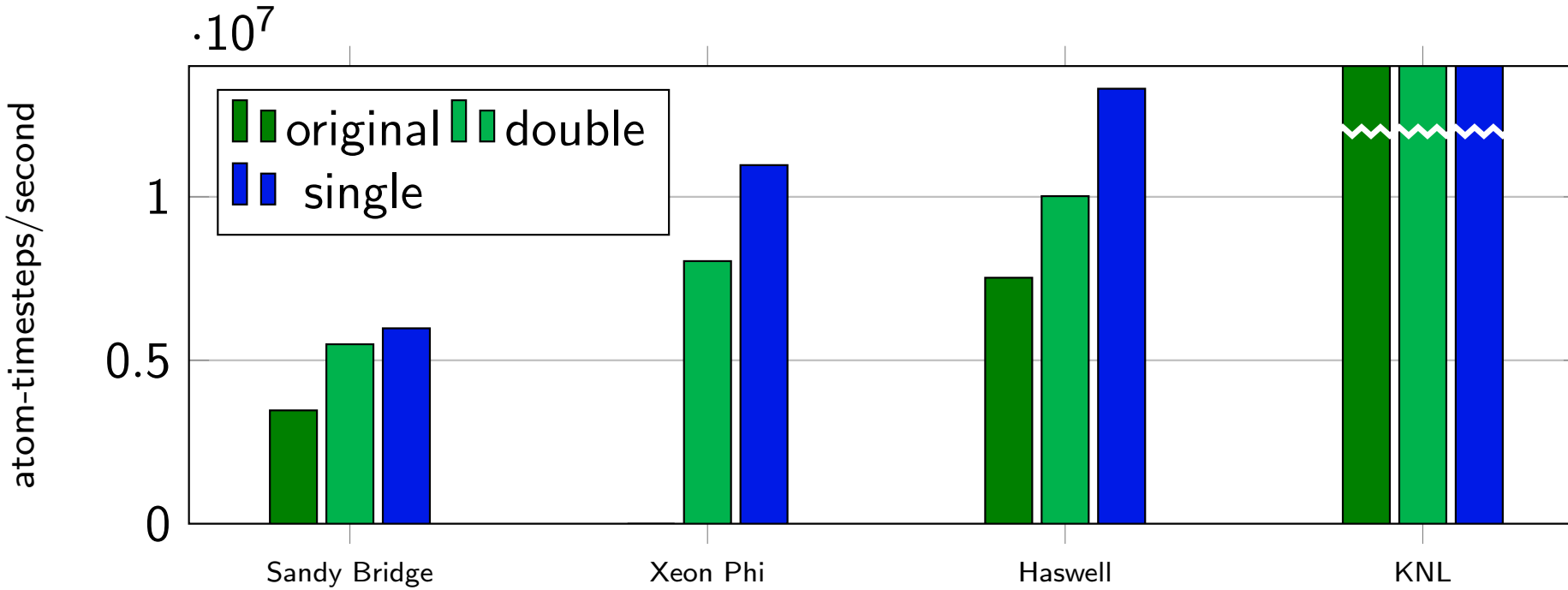
KNL Readiness

- Intrinsics abstraction already supports AVX-512
- Compilation possible for `-xMIC-AVX512`
- Running under Intel SDE `sde -knl -- ...`
- Has been tested on KNL prototypes by Intel employees
- We already have benchmarks prepared for the point when performance data can be shared

Portable Speedups (single-threaded, native)



Individual Node Performance (Multi-Threaded, Offloaded, Realistic Simulation)



Configuration

Arch.	Model	Year	Cores
Haswell	2x Xeon E5-2680 v3	2014	24
Sandy Bridge	2x Xeon E5-2450	2012	16
	1x Xeon Phi 5110P	2012	60

AIREBO

Compared to Tersoff

- Symmetrical bond order (REBO)
- Additional bond order terms (REBO)
- Lennard-Jones (longer) ranged force
- Torsion force

Challenges

- Software Engineering: 4200 lines vs 800 lines
- Again very short loops (1, 2, 3 iterations)
- Ill-suited patterns: Searches, branching on values

Conclusion (1/2)

Timeline

May'15	IPCC @ RWTH – kickoff	✓
Q1–Q2	Buckingham potential	✓
Q1–Q2	Tersoff potential	✓
Q3–Q4	AIREBO potential	*in progress*
Q3–	PPPM electrostatics solver	*in progress*
Year 2	PPPM dispersion solver	

Conclusion (2/2)

Dissemination

Oct'15	Code dungeon EMEA IPCC meeting, Munich	✓
Nov'15	`github.com/HPAC` Code, tests and benchmarks	 *in progress*
Nov'15	Talk + paper at SC'15	✓
Dec'15	Code release LAMMPS' USER–Intel package	✓
Dec'15	IXPUG Vectorization WG (Markus); PC (Paolo)	*in progress*
Feb'16	Intel paper on chemistry codes	*in progress*