The Tersoff many-body potential: Sustainable performance through vectorization

Markus Höhnerbach Ahmed E. Ismail Paolo Bientinesi

High Performance and Automatic Computing Group
Aachen Institute for Advanced Study in Computational Engineering Science
RWTH Aachen University

Supercomputing 2015
Producing High Performance and Sustainable Software for Molecular Simulation

Vectorization

- Basically mandatory on the Xeon Phi
- Ubiquitous: Gromacs (intrinsics), NAMD (intrinsics), LAMMPS (USER-INTEL, pragma based approach)
- Typically pair potentials and neighbor list build

Gains in complex cases?

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

[a] Brown et al, An Evaluation of Molecular Dynamics Performance on the Hybrid Cray XK6 Supercomputer, Procedia Computer Science, 2012.

[b] Brown at al, Implementing molecular dynamics on hybrid high performance computers – Three-body potentials, Computer Physics Communications, 2013.

[c] Hou et al, Efficient GPU-accelerated molecular dynamics simulation of solid covalent crystals, Computer Physics Communications, 2013.

The Tersoff potential

$$V = \sum_i \sum_{j:r_{ij}<r_C} \overbrace{f_C(r_{ij})\left[f_R(r_{ij}) + b_{ij} f_A(r_{ij})\right]}^{V(i,j,\zeta_{ij})} \tag{1}$$

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

$$\zeta_{ij} = \sum_{k:r_{ik}<r_C} \underbrace{f_C(r_{ik}) g(\theta_{ijk}) \exp(\lambda_3(r_{ij} - r_{ik}))}_{\zeta(i,j,k)} \tag{3}$$

- 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

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**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$

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**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$

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**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$

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

Challenges

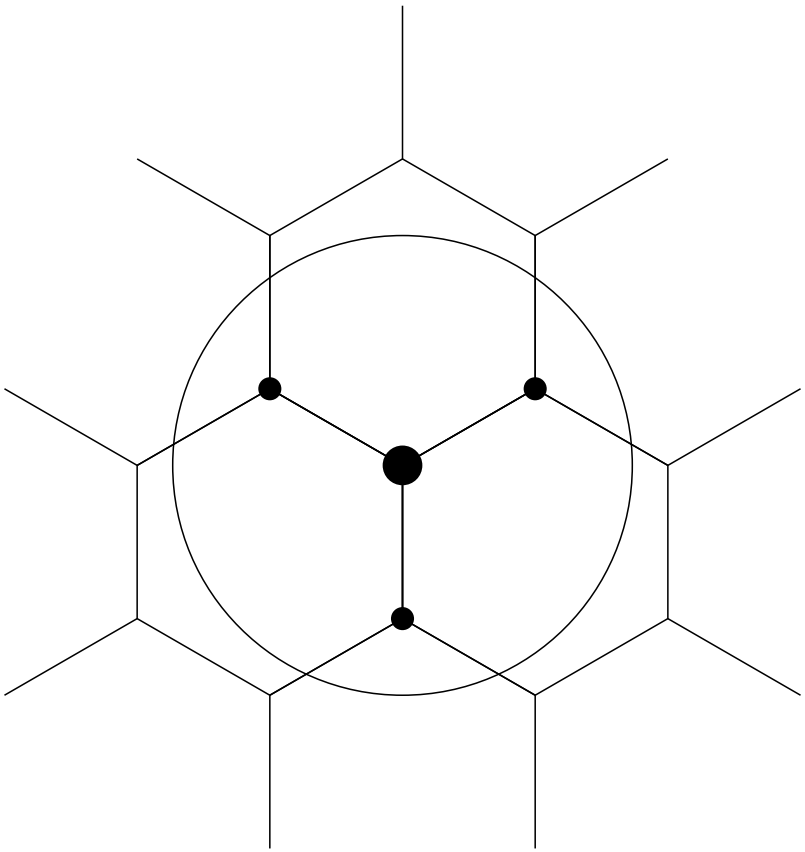

Figure: Graphene

- Few neighbors
- Fewer interactions

Model Problem

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

[a] Thanks to Marcus Schmidt

Vectorization

"J" algorithm

```
for ... do
    for ... do
        ...;
        for ... do
            ...;
        ...;
        for ... do
            ...;
```

"I" algorithm

```
for ... do
    for ... do
        ...;
        for ... do
            ...;
        ...;
        for ... do
            ...;
```

Implementation

About LAMMPS

- Already Xeon Phi support via USER-INTEL package
- Usage model: Offloading

First attempt: Double precision intrinsics

- Would it not be nice to have different precisions?
- Would it not be nice to support different instruction sets?
- Is this sustainable?

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, SSE, AVX, AVX2, IMCI, AVX-512, array notation (Cilk)

Advantages

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

Effect of Vectorization

Experiment: Xeon Phi 5110P

"I" algorithm	Sequential	Native

Timings (in seconds)

Precision	LAMMPS	I-Scalar	I-Vec
double	88.72	58.04	14.18
single	–	45.59	8.56

Speedup

Precision	$\frac{\text{LAMMPS}}{\text{I-Scalar}}$	$\frac{\text{LAMMPS}}{\text{I-Vec}}$	$\frac{\text{I-Scalar}}{\text{I-Vec}}$
double	1.53	6.26	4.09
single	1.95	10.36	5.32

Effect of Vectorization

Experiment: Xeon E5-2680 v3

Sequential Haswell Double precision

Timings (in seconds) & Speedups

Arch.	"I"	"J"	$\frac{\text{LAMMPS}}{\text{"I"}}$	$\frac{\text{LAMMPS}}{\text{"J"}}$
LAMMPS	28.23		1	
Scalar	18.63	14.7	1.52	1.91
SSE	37.15	21.3	0.76	1.32
AVX	23.92	12.5	1.18	2.25
AVX2	16.59	10.9	1.70	2.59

Full System Comparison

Experiment

Arch.	Model	Year	Cores
Haswell	2x Xeon E5-2680 v3	2014	24
Sandy Bridge	2x Xeon E5-2450	2012	16
Phi	1x Xeon Phi 5110P	2012	$8 \cdot 29$
	Offload via Sandy Bridge		

Timings (in seconds)

System		double	single
Sandy Bridge	LAMMPS	395.89	
	Vec	250.02	229.65
Phi	Vec	170.88	125.14
Haswell	LAMMPS	182.43	
	Vec	136.99	103.16
KNL	Vec	?	?

Outlook and conclusion

- Complicated potentials benefit from vectorization
- Hiding behind abstraction works

- Unification? There's VCL, Vc, UME, various math libraries
- What about accuracy?

- Integration into LAMMPS/USER-INTEL
- Continue work on Xeon Phi and LAMMPS
- OpenMP 4.1
- AVX-512

Dicussion & Questions

The Group	`hpac.rwth-aachen.de`
The IPCC	`hpac.rwth-aachen.de/ipcc`
The Code	`github.com/v0i0/lammps-tersoff-vector`
E-Mail	`hoehnerbach@aices.rwth-aachen.de`

Backup

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**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$;

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**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$;

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**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$;

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

The Tersoff Algorithm – Transformation

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**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$;

The Tersoff Algorithm – Transformation

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**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$;

The Tersoff Algorithm – Transformation

for *i in local atoms of the current thread* **do**

  for *j in atoms neighboring i* **do**

    $\zeta_{ij} \leftarrow 0,\ F_i^{ij} \leftarrow 0,\ F_j^{ij} \leftarrow 0,\ F_k^{ij} \leftarrow 0;$

    for *k in atoms neighboring i* **do**

      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i,j,k)\ ;$

      $F_i^{ij} \leftarrow F_i^{ij} + \partial_{x_i}\zeta(i,j,k);$

      $F_j^{ij} \leftarrow F_j^{ij} + \partial_{x_j}\zeta(i,j,k);$

      $F_k^{ij} \leftarrow F_k^{ij} + \partial_{x_k}\zeta(i,j,k)\ ;$

    $E \leftarrow E + V(i,j,\zeta_{ij})\ ;$

    $F_i \leftarrow F_i - \partial_{x_i} V(i,j,\zeta_{ij});$

    $F_j \leftarrow F_j - \partial_{x_j} V(i,j,\zeta_{ij});$

    $\delta\zeta \leftarrow \partial_\zeta V(i,j,\zeta_{ij});$

    for *k in atoms neighboring i* **do**

      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k);$

      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k);$

      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)\ \ ;$

The Tersoff Algorithm – Transformation

for *i in local atoms of the current thread* **do**

  for *j in atoms neighboring i* **do**

    $\zeta_{ij} \leftarrow 0,\ F_i^{ij} \leftarrow 0,\ F_j^{ij} \leftarrow 0,\ F_k^{ij} \leftarrow 0;$

    for *k in atoms neighboring i* **do**

      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i,j,k)\ ;$

      $F_i^{ij} \leftarrow F_i^{ij} + \partial_{x_i}\zeta(i,j,k);$

      $F_j^{ij} \leftarrow F_j^{ij} + \partial_{x_j}\zeta(i,j,k);$

      $F_k^{ij} \leftarrow F_k^{ij} + \partial_{x_k}\zeta(i,j,k)\ ;$

    $E \leftarrow E + V(i,j,\zeta_{ij})\ ;$

    $F_i \leftarrow F_i - \partial_{x_i} V(i,j,\zeta_{ij});$

    $F_j \leftarrow F_j - \partial_{x_i} V(i,j,\zeta_{ij});$

    $\delta\zeta \leftarrow \partial_\zeta V(i,j,\zeta_{ij});$

    $F_i \leftarrow F_i - \delta\zeta F_i^{ij};$

    $F_j \leftarrow F_j - \delta\zeta F_j^{ij};$

    for *k in atoms neighboring i* **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot F_k^{ij}\ ;$

The Tersoff Algorithm – Transformation

for *i in local atoms of the current thread* **do**

  for *j in atoms neighboring i* **do**

    $\zeta_{ij} \leftarrow 0,\ F_i^{ij} \leftarrow 0,\ F_j^{ij} \leftarrow 0,\ F_k^{ij} \leftarrow 0;$

    for *k in atoms neighboring i* **do**

      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i,j,k)\ ;$

      $F_i^{ij} \leftarrow F_i^{ij} + \partial_{x_i}\zeta(i,j,k);$

      $F_j^{ij} \leftarrow F_j^{ij} + \partial_{x_j}\zeta(i,j,k);$

      $F_k^{ij} \leftarrow F_k^{ij} + \partial_{x_k}\zeta(i,j,k)\ ;$

    $E \leftarrow E + V(i,j,\zeta_{ij})\ ;$

    $\delta\zeta \leftarrow \partial_\zeta V(i,j,\zeta_{ij});$

    $F_i \leftarrow F_i - \partial_{x_i} V(i,j,\zeta_{ij}) - \delta\zeta \cdot F_i^{ij};$

    $F_j \leftarrow F_j - \partial_{x_j} V(i,j,\zeta_{ij}) - \delta\zeta \cdot F_j^{ij}\ ;$

    for *k in atoms neighboring i* **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot F_k^{ij}\ ;$