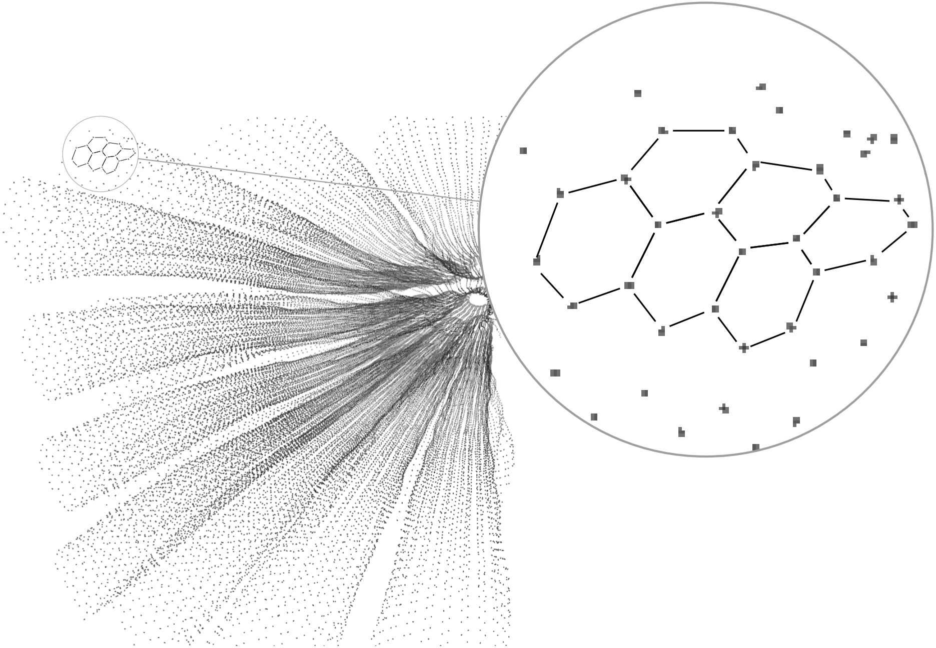
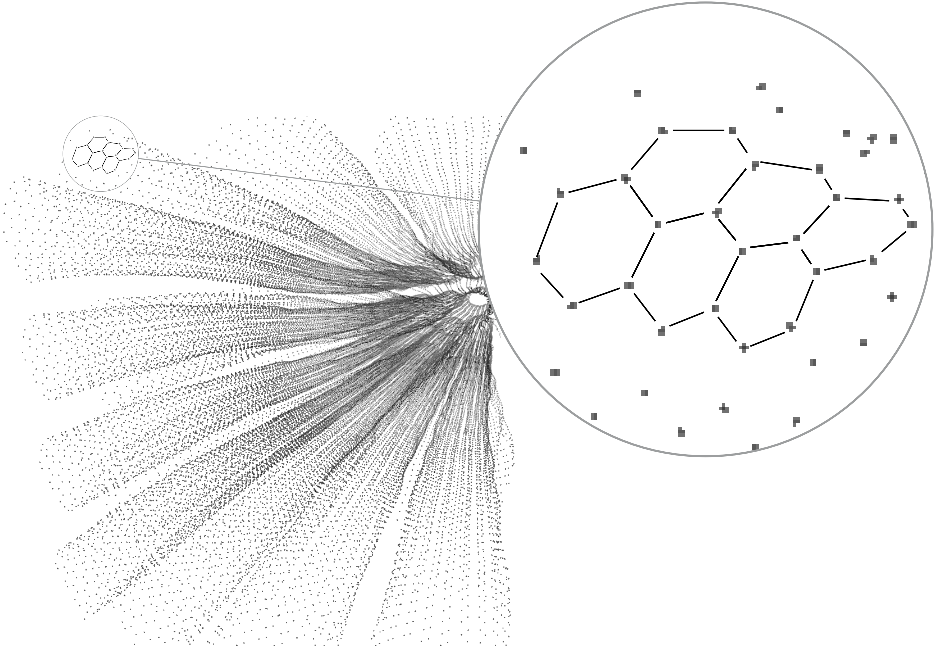
Vectorization of Multi-Body Potentials: Performance and Portability

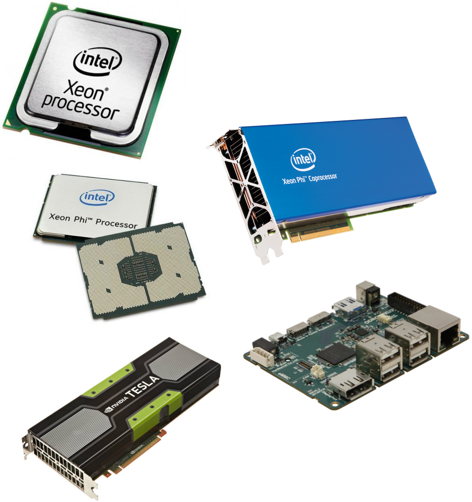
Markus Höhnerbach   Ahmed E. Ismail   **Paolo Bientinesi**

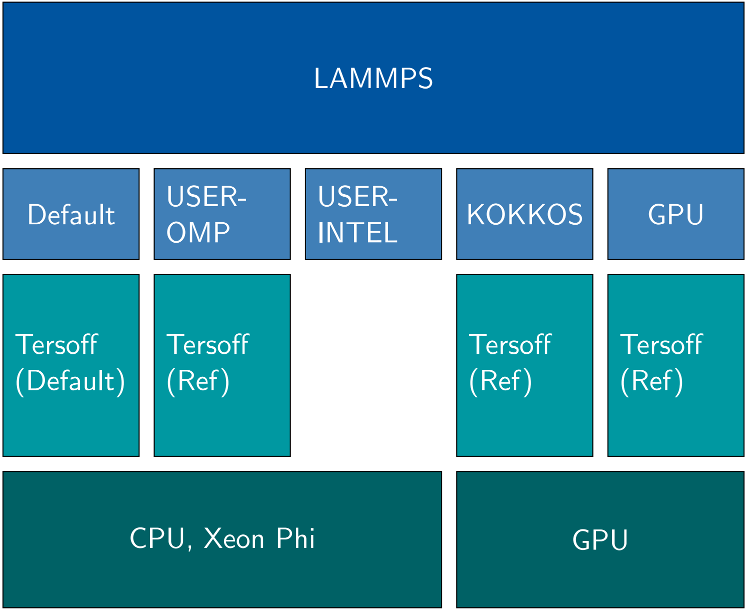
SIAM CSE '17

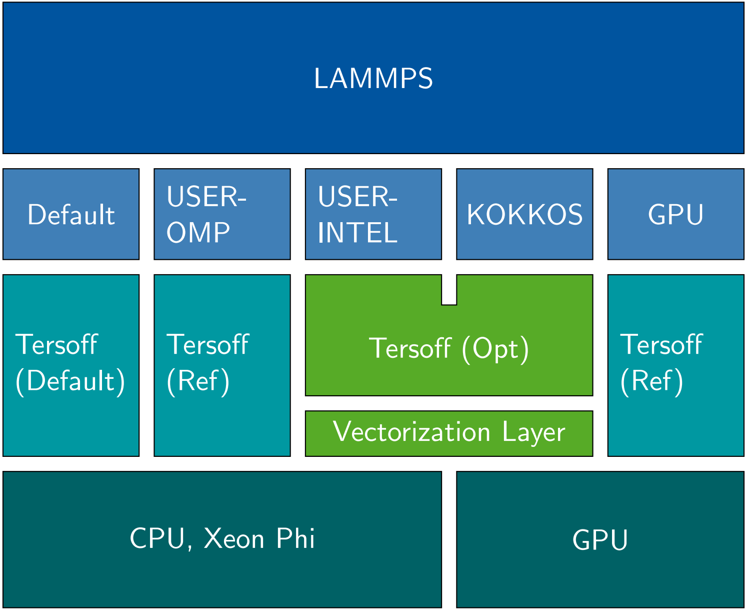
HPAC | RWTH AACHEN UNIVERSITY

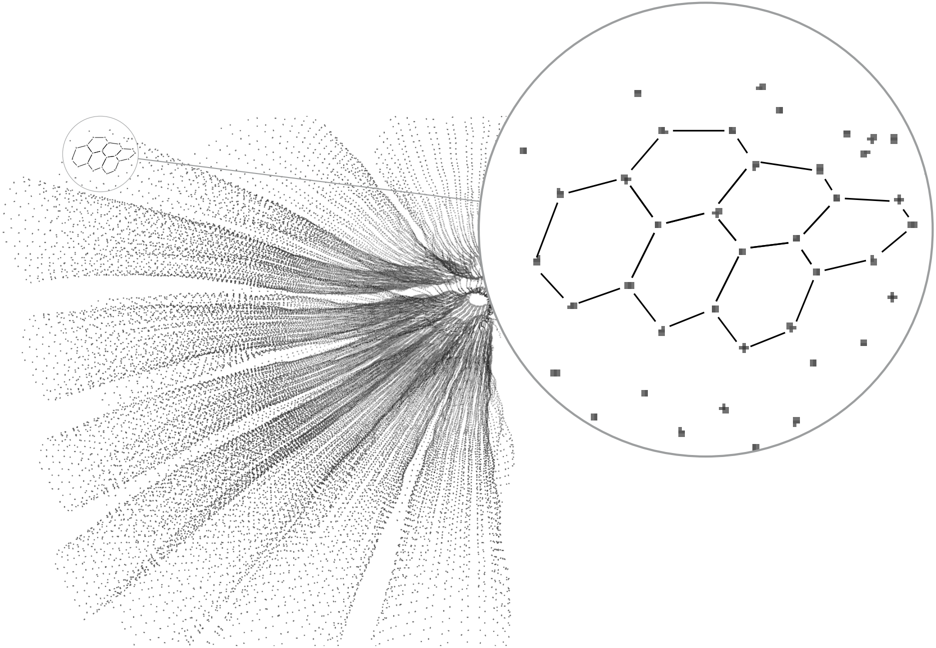


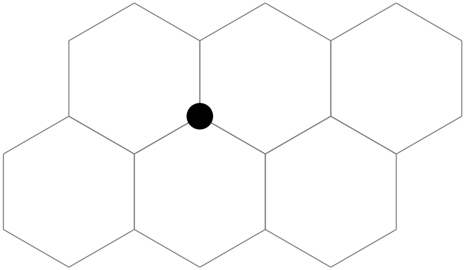



intel
Xeon processor
intel
Xeon Phi™ Processor
intel
Xeon Phi™ Coprocessor
NVIDIA TESLA

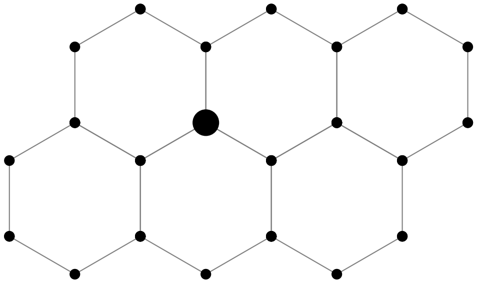

LAMMPS
Default
USER-OMP
USER-INTEL
KOKKOS
GPU
Tersoff (Default)
Tersoff (Ref)
Tersoff (Ref)
Tersoff (Ref)
CPU, Xeon Phi
GPU


LAMMPS
Default
USER-OMP
USER-INTEL
KOKKOS
GPU
Tersoff (Default)
Tersoff (Ref)
Tersoff (Opt)
Vectorization Layer
Tersoff (Ref)
CPU, Xeon Phi
GPU

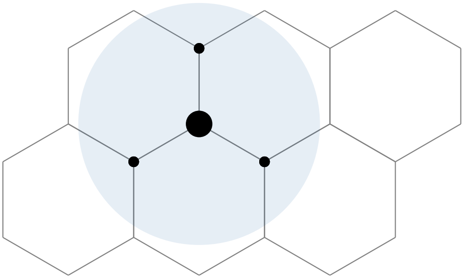




$$V = \sum_{i} \sum_{j:??} V(i,j)$$

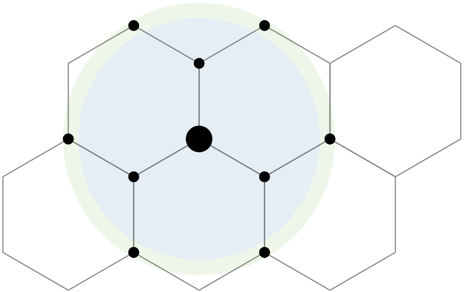


$$V = \sum_i \sum_j V(i,j)$$



$$V = \sum_i \sum_{j \in \mathcal{N}_i} V(i,j)$$

$$j \in \mathcal{N}_i \Leftrightarrow: r_{ij} < r_{ij}^{\mathrm{cutoff}}; \qquad V(i,j) = 0 \ \forall j \notin \mathcal{N}_i$$



$$V = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j)$$

$$j \in \mathcal{S}_i \Leftrightarrow: r_{ij} < r_{ij}^{\mathrm{cutoff}} + r^{\mathrm{skin}}, \mathcal{N}_i \subset \mathcal{S}_i$$

$$V^{\text{two-body}} = \sum_i \sum_{j \in \mathcal{S}_i} V(i,j)$$

$$V^{\text{Tersoff}} = \sum_i \sum_{j \in \mathcal{S}_i} V(i,j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i,j,k))$$

$$V(i,j,\zeta) = f_C(r_{ij}) \left[f_R(r_{ij}) + (1 + \beta^\nu \zeta^\nu)^{-\frac{1}{2\nu}} f_A(r_{ij}) \right]$$

$$\cdots$$

$$V^{\mathrm{two-body}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j)$$

$$V^{\mathrm{Tersoff}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i,j,k))$$

$$V(i,j,\zeta) = f_C(r_{ij}) \left[f_R(r_{ij}) + (1 + \beta^\nu \zeta^\nu)^{-\frac{1}{2\nu}} f_A(r_{ij}) \right]$$

...

$$V^{\mathrm{two-body}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j)$$

$$V^{\mathrm{Tersoff}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i,j,k))$$

$$V(i,j,\zeta) = f_C(r_{ij}) \left[f_R(r_{ij}) + (1 + \beta^{\nu} \zeta^{\nu})^{-\frac{1}{2\nu}} f_A(r_{ij}) \right]$$

$$\cdots$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

$$V^{\mathrm{Tersoff}} = \sum_{i} \sum_{j \in \mathcal{S}_j} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

for i **do**
    for $j \in \mathcal{S}_i$ **do**
        $\zeta_{ij} \leftarrow 0;$
        for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
            $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i, j, k);$
        $V \leftarrow V + V(i, j, \zeta_{ij});$
        $F_i \leftarrow F_i - \partial_{x_i} V(i, j, \zeta_{ij});$
        $F_j \leftarrow F_j - \partial_{x_j} V(i, j, \zeta_{ij});$
        $\delta\zeta \leftarrow \partial_{\zeta} V(i, j, \zeta_{ij});$
        for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
            $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i} \zeta(i, j, k);$
            $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j} \zeta(i, j, k);$
            $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k} \zeta(i, j, k);$

$$V^{\text{Tersoff}} = \sum_i \sum_{j \in \mathcal{S}_j} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

for i **do** | Loop over all atoms

  for $j \in \mathcal{S}_i$ **do**

    $\zeta_{ij} \leftarrow 0$;

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

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

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

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

$$V^{\text{Tersoff}} = \sum_{i} \sum_{j \in \mathcal{S}_j} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

for i **do** | Loop over all atoms

  for $j \in \mathcal{S}_i$ **do** | Loop over atoms "closeby" to i

    $\zeta_{ij} \leftarrow 0$;

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

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

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

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

$$V^{\text{Tersoff}} = \sum_i \sum_{j\in\mathcal{S}_j} V(i,j,\sum_{k\in\mathcal{S}_i\setminus\{j\}} \zeta(i,j,k))$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

for i **do**
  for $j \in \mathcal{S}_i$ **do**
    $\zeta_{ij} \leftarrow 0$;
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i,j,k)$;
    $V \leftarrow V + V(i,j,\zeta_{ij})$;
    $F_i \leftarrow F_i - \partial_{x_i} V(i,j,\zeta_{ij})$;
    $F_j \leftarrow F_j - \partial_{x_j} V(i,j,\zeta_{ij})$;
    $\delta\zeta \leftarrow \partial_\zeta V(i,j,\zeta_{ij})$;
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i}\zeta(i,j,k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j}\zeta(i,j,k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k}\zeta(i,j,k)$;

Loop over all atoms

Loop over atoms "closeby" to i

Compute ζ

$$V^{\text{Tersoff}} = \sum_i \sum_{j \in \mathcal{S}_j} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

for i **do** — Loop over all atoms

  for $j \in \mathcal{S}_i$ **do** — Loop over atoms "closeby" to i

    $\zeta_{ij} \leftarrow 0$; — Compute ζ

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

    $V \leftarrow V + V(i, j, \zeta_{ij})$; — Compute potential

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

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

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

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

$$V^{\mathrm{Tersoff}} = \sum_{i} \sum_{j \in \mathcal{S}_j} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

for i **do** — Loop over all atoms
  for $j \in \mathcal{S}_i$ **do** — Loop over atoms “closeby” to i
    $\zeta_{ij} \leftarrow 0$; — Compute ζ
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i, j, k)$;
    $V \leftarrow V + V(i, j, \zeta_{ij})$; — Compute potential
    $F_i \leftarrow F_i - \partial_{x_i} V(i, j, \zeta_{ij})$;
    $F_j \leftarrow F_j - \partial_{x_j} V(i, j, \zeta_{ij})$;
    $\delta\zeta \leftarrow \partial_{\zeta} V(i, j, \zeta_{ij})$;
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i} \zeta(i, j, k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j} \zeta(i, j, k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k} \zeta(i, j, k)$;

$$V^{\mathrm{Tersoff}} = \sum_i \sum_{j \in \mathcal{S}_j} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

for i **do**
  for $j \in \mathcal{S}_i$ **do**
    $\zeta_{ij} \leftarrow 0$;
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i, j, k)$;
    $V \leftarrow V + V(i, j, \zeta_{ij})$;
    $F_i \leftarrow F_i - \partial_{x_i} V(i, j, \zeta_{ij})$;
    $F_j \leftarrow F_j - \partial_{x_j} V(i, j, \zeta_{ij})$;
    $\delta\zeta \leftarrow \partial_\zeta V(i, j, \zeta_{ij})$;
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i} \zeta(i, j, k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j} \zeta(i, j, k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k} \zeta(i, j, k)$;

Loop over all atoms
Loop over atoms "closeby" to i
Compute ζ
Compute potential
Forces directly due to V

$$V^{\text{Tersoff}} = \sum_i \sum_{j \in \mathcal{S}_j} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} V$$

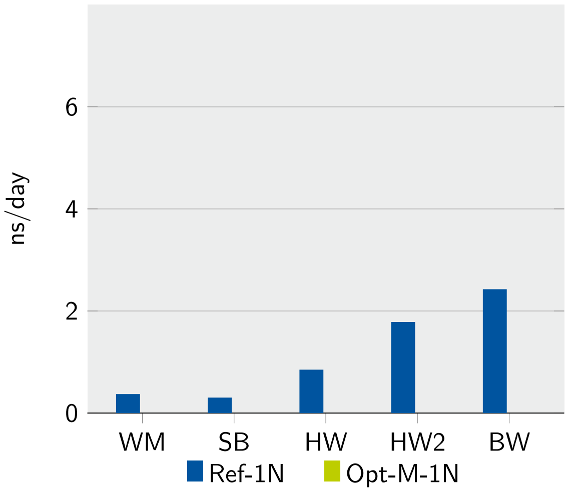
for i **do**	Loop over all atoms
for $j \in \mathcal{S}_i$ **do**	Loop over atoms "closeby" to i

$\zeta_{ij} \leftarrow 0$;
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i, j, k)$;
	Compute ζ

$V \leftarrow V + V(i, j, \zeta_{ij})$;	Compute potential
$F_i \leftarrow F_i - \partial_{x_i} V(i, j, \zeta_{ij})$; $F_j \leftarrow F_j - \partial_{x_j} V(i, j, \zeta_{ij})$;	Forces directly due to V

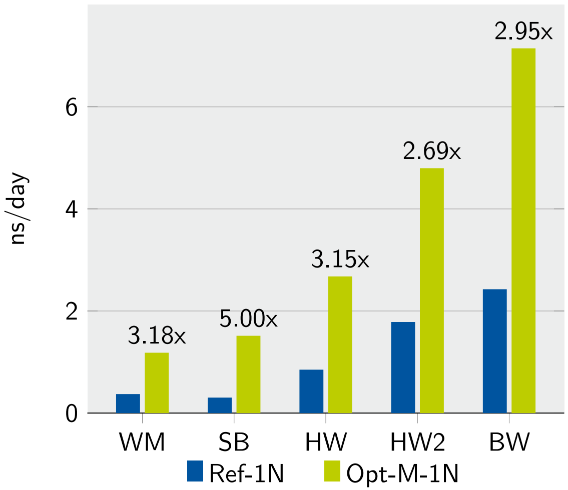
$\delta\zeta \leftarrow \partial_\zeta V(i, j, \zeta_{ij})$;
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $F_i \leftarrow F_i - \delta\zeta \cdot \partial_{x_i} \zeta(i, j, k)$;
      $F_j \leftarrow F_j - \delta\zeta \cdot \partial_{x_j} \zeta(i, j, k)$;
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_{x_k} \zeta(i, j, k)$;
	Chain Rule for ζ

CPU: Single Node Execution (512 000 atoms)



Name	Processor	Cores	Vector ISA
WM	Intel Xeon X5675	2 × 6	SSE4.2
SB	Intel Xeon E5-2450	2 × 8	AVX
HW	Intel Xeon E5-2680v3	2 × 12	AVX2
HW2	Intel Xeon E5-2697v3	2 × 14	AVX2
BW	Intel Xeon E5-2697v4	2 × 18	AVX2

CPU: Single Node Execution (512 000 atoms)



Name	Processor	Cores	Vector ISA
WM	Intel Xeon X5675	2×6	SSE4.2
SB	Intel Xeon E5-2450	2×8	AVX
HW	Intel Xeon E5-2680v3	2×12	AVX2
HW2	Intel Xeon E5-2697v3	2×14	AVX2
BW	Intel Xeon E5-2697v4	2×18	AVX2

for i **do**
  for $j \in \mathcal{S}_i$ **do**
    $\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$
    $\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i,j,k);$
      $\partial_i \zeta \leftarrow \partial_i \zeta + \partial_{x_i} \zeta(i,j,k);$
      $\partial_j \zeta \leftarrow \partial_j \zeta + \partial_{x_j} \zeta(i,j,k);$
      $\partial_k \zeta \leftarrow \partial_{x_k} \zeta(i,j,k);$
    $V \leftarrow V + V(i,j,\zeta_{ij});$
    $F_i \leftarrow F_i - \partial_{x_i} V(i,j,\zeta_{ij});$
    $F_j \leftarrow F_j - \partial_{x_j} V(i,j,\zeta_{ij});$
    $\delta\zeta \leftarrow \partial_\zeta V(i,j,\zeta_{ij});$
    $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$
    $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**
      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$

for i **do** — Loop over all atoms

- **for** $j \in \mathcal{S}_i$ **do**
  - $\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$
  - $\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$
  - **for** $k \in \mathcal{S}_i \setminus \{j\}$ **do**
    - $\zeta_{ij} \leftarrow \zeta_{ij} + \zeta(i, j, k);$
    - $\partial_i \zeta \leftarrow \partial_i \zeta + \partial_{x_i} \zeta(i, j, k);$
    - $\partial_j \zeta \leftarrow \partial_j \zeta + \partial_{x_j} \zeta(i, j, k);$
    - $\partial_k \zeta \leftarrow \partial_{x_k} \zeta(i, j, k);$
  - $V \leftarrow V + V(i, j, \zeta_{ij});$
  - $F_i \leftarrow F_i - \partial_{x_i} V(i, j, \zeta_{ij});$
  - $F_j \leftarrow F_j - \partial_{x_j} V(i, j, \zeta_{ij});$
  - $\delta\zeta \leftarrow \partial_\zeta V(i, j, \zeta_{ij});$
  - $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$
  - $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$
  - **for** $k \in \mathcal{S}_i \setminus \{j\}$ **do**
    - $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$

for i **do** | Loop over all atoms

  for $j \in \mathcal{S}_i$ **do** | Loop over atoms "closeby" to i

    $\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$

    $\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

      $\partial_k \zeta \leftarrow \partial_{x_k} \zeta(i,j,k);$

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

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

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

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

    $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$

    $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$

for i **do** — Loop over all atoms

  for $j \in \mathcal{S}_i$ **do** — Loop over atoms “closeby” to i

    $\zeta_{ij} \leftarrow 0; \quad \partial_k\zeta \leftarrow 0 \quad \forall k;$ — Compute ζ

    $\partial_i\zeta \leftarrow 0; \quad \partial_j\zeta \leftarrow 0;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

      $\partial_k\zeta \leftarrow \partial_{x_k}\zeta(i,j,k);$

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

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

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

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

    $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i\zeta;$

    $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j\zeta;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k\zeta;$

```
for i do                                              | Loop over all atoms
    for j ∈ S_i do                                    | Loop over atoms "closeby" to i
        ζ_ij ← 0;   ∂_k ζ ← 0   ∀k;                   | Compute ζ
        ∂_i ζ ← 0;   ∂_j ζ ← 0;
        for k ∈ S_i \ {j} do
            ζ_ij ← ζ_ij + ζ(i, j, k);
            ∂_i ζ ← ∂_i ζ + ∂_{x_i} ζ(i, j, k);
            ∂_j ζ ← ∂_j ζ + ∂_{x_j} ζ(i, j, k);       | And derivates of ζ
            ∂_k ζ ← ∂_{x_k} ζ(i, j, k);
        V ← V + V(i, j, ζ_ij);
        F_i ← F_i − ∂_{x_i} V(i, j, ζ_ij);
        F_j ← F_j − ∂_{x_j} V(i, j, ζ_ij);
        δζ ← ∂_ζ V(i, j, ζ_ij);
        F_i ← F_i − δζ · ∂_i ζ;
        F_j ← F_j − δζ · ∂_j ζ;
        for k ∈ S_i \ {j} do
            F_k ← F_k − δζ · ∂_k ζ;
```

for i **do** — Loop over all atoms

  for $j \in \mathcal{S}_i$ **do** — Loop over atoms "closeby" to i

    $\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$ — Compute ζ

    $\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

      $\partial_j \zeta \leftarrow \partial_j \zeta + \partial_{x_j} \zeta(i, j, k);$ — And derivates of ζ

      $\partial_k \zeta \leftarrow \partial_{x_k} \zeta(i, j, k);$

    $V \leftarrow V + V(i, j, \zeta_{ij});$ — Compute potential

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

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

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

    $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$

    $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$

for i **do** — Loop over all atoms

  for $j \in \mathcal{S}_i$ **do** — Loop over atoms "closeby" to i

Compute ζ:

$$\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$$

$$\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

And derivates of ζ:

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

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

$$\partial_k \zeta \leftarrow \partial_{x_k} \zeta(i,j,k);$$

Compute potential:

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

Forces directly due to V:

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

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

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

$$F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$$

$$F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

$$F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$$

for i **do** — Loop over all atoms

  for $j \in \mathcal{S}_i$ **do** — Loop over atoms "closeby" to i

    $\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$ — Compute ζ

    $\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

      $\partial_j \zeta \leftarrow \partial_j \zeta + \partial_{x_j} \zeta(i,j,k);$ — And derivates of ζ

      $\partial_k \zeta \leftarrow \partial_{x_k} \zeta(i,j,k);$

    $V \leftarrow V + V(i,j,\zeta_{ij});$ — Compute potential

    $F_i \leftarrow F_i - \partial_{x_i} V(i,j,\zeta_{ij});$ — Forces directly due to V

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

    $\delta\zeta \leftarrow \partial_\zeta V(i,j,\zeta_{ij});$ — Chain Rule for ζ

    $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$

    $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$

for i **do** — Loop over all atoms

  for $j \in \mathcal{S}_i$ **do** — Loop over atoms "closeby" to i

    $\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$ — Compute ζ

    $\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

      $\partial_j \zeta \leftarrow \partial_j \zeta + \partial_{x_j} \zeta(i,j,k);$ — And derivates of ζ

      $\partial_k \zeta \leftarrow \partial_{x_k} \zeta(i,j,k);$

    $V \leftarrow V + V(i,j,\zeta_{ij});$ — Compute potential

    $F_i \leftarrow F_i - \partial_{x_i} V(i,j,\zeta_{ij});$ — Forces directly due to V

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

    $\delta\zeta \leftarrow \partial_\zeta V(i,j,\zeta_{ij});$ — Chain Rule for ζ

    $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$ — Updates for F_i and F_j

    $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$

```
for i do                                              | Loop over all atoms
  for j ∈ S_i do                                      | Loop over atoms "closeby" to i
    ζ_ij ← 0;   ∂_k ζ ← 0   ∀k;                       | Compute ζ
    ∂_i ζ ← 0;   ∂_j ζ ← 0;                           |
    for k ∈ S_i \ {j} do                              |
      ζ_ij  ← ζ_ij  + ζ(i, j, k);                     |
      ∂_i ζ ← ∂_i ζ + ∂_{x_i} ζ(i, j, k);             |
      ∂_j ζ ← ∂_j ζ + ∂_{x_j} ζ(i, j, k);             | And derivates of ζ
      ∂_k ζ ←         ∂_{x_k} ζ(i, j, k);             |
    V ← V + V(i, j, ζ_ij);                            | Compute potential
    F_i ← F_i − ∂_{x_i} V(i, j, ζ_ij);                | Forces directly due to V
    F_j ← F_j − ∂_{x_j} V(i, j, ζ_ij);                |
    δζ ← ∂_ζ V(i, j, ζ_ij);                           | Chain Rule for ζ
    F_i ← F_i − δζ · ∂_i ζ;                           | Updates for F_i and F_j
    F_j ← F_j − δζ · ∂_j ζ;                           |
    for k ∈ S_i \ {j} do                              | Updates for F_k
      F_k ← F_k − δζ · ∂_k ζ;                         |
```

for i **do**

  for $j \in \mathcal{S}_i$ **do**

    $\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$

    $\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

      $\partial_k \zeta \leftarrow \partial_{x_k} \zeta(i,j,k);$

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

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

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

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

    $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$

    $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$

Loop over all atoms

Loop over atoms “closeby” to i

Compute ζ

And derivates of ζ

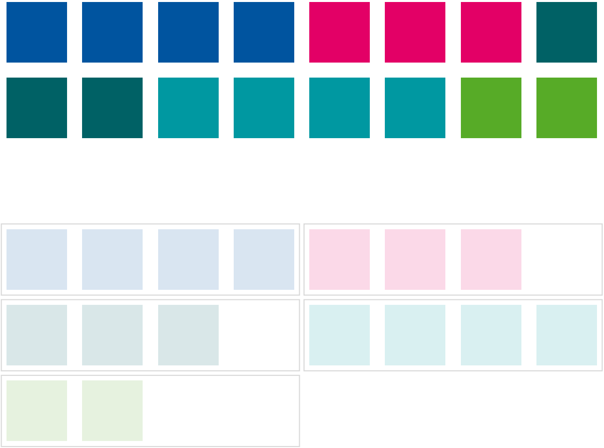
Compute potential

Forces directly due to V

Chain Rule for ζ

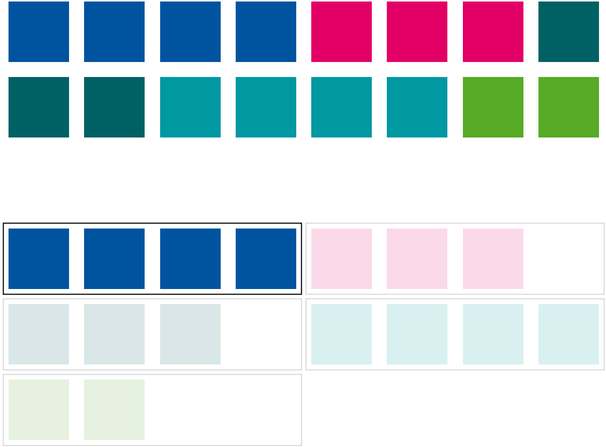
Updates for F_i and F_j

Updates for F_k



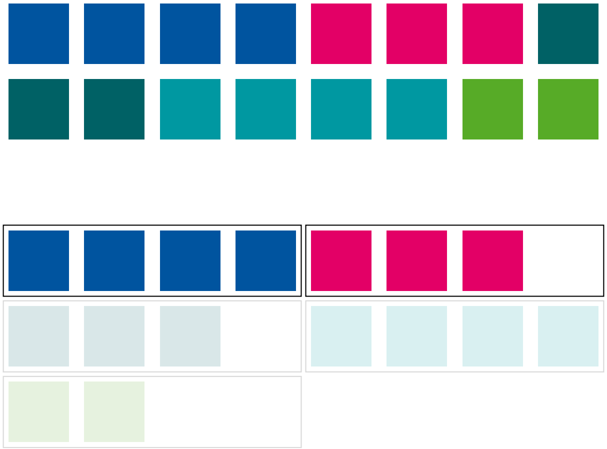
Option A

```
#pragma omp parallel for
for (int i = ...
  #pragma omp simd
  for (int j = ...
    ...
```



Option A

```
#pragma omp parallel for
for (int i = ...
  #pragma omp simd
  for (int j = ...
    ...
```

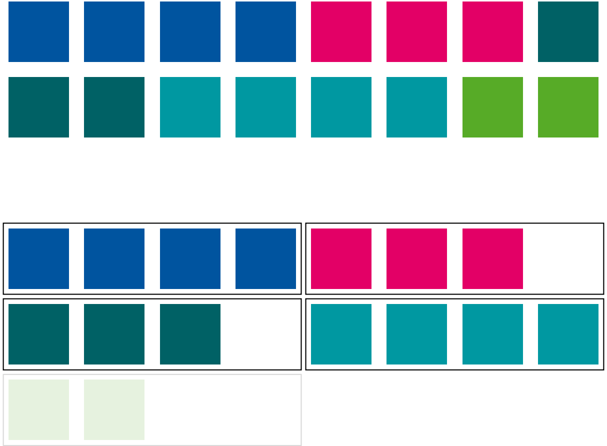


Option A

```
#pragma omp parallel for
for (int i = ...
  #pragma omp simd
  for (int j = ...
    ...
```

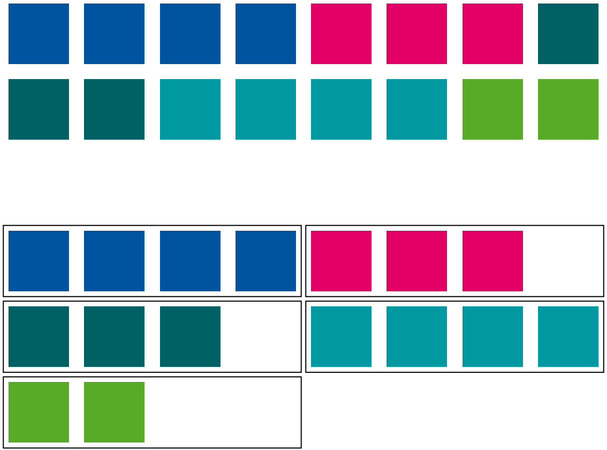
Option A

```
#pragma omp parallel for
for (int i = ...
  #pragma omp simd
  for (int j = ...
    ...
```



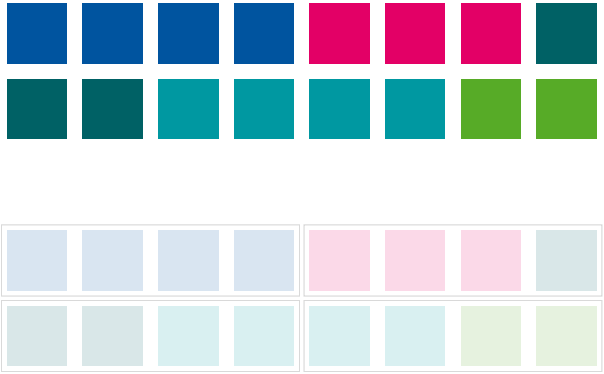
Option A

```
#pragma omp parallel for
for (int i = ...
  #pragma omp simd
  for (int j = ...
    ...
```



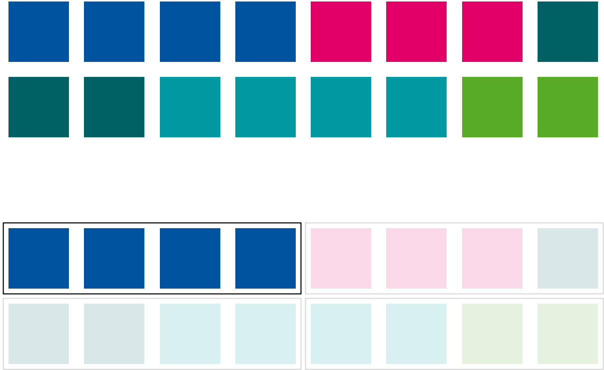
Option A

```
#pragma omp parallel for
for (int i = ...
  #pragma omp simd
  for (int j = ...
    ...
```



Option B

```
#pragma omp parallel for simd collapse(2)
for (int i = ...

  for (int j = ...
    ...
```

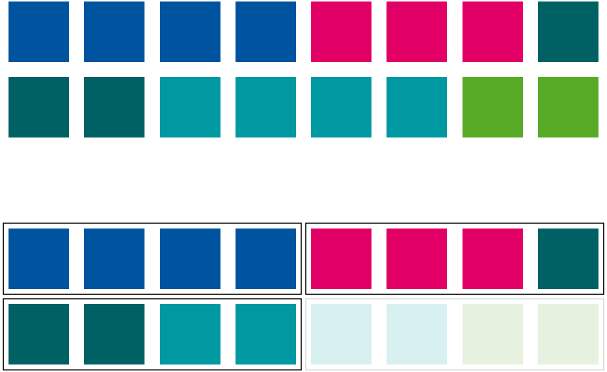


Option B

```
#pragma omp parallel for simd collapse(2)
for (int i = ...

  for (int j = ...
    ...
```

Option B

```
#pragma omp parallel for simd collapse(2)
for (int i = ...

  for (int j = ...
    ...
```

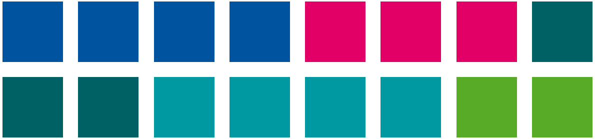


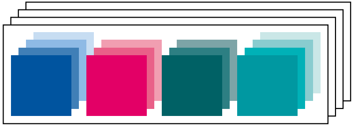
Option B

```
#pragma omp parallel for simd collapse(2)
for (int i = ...

  for (int j = ...
    ...
```

Option B

```
#pragma omp parallel for simd collapse(2)
for (int i = ...

  for (int j = ...
    ...
```





Option C

```
#pragma omp parallel for simd
for (int i = ...

  for (int j = ...
    ...
```

for i **do**
  for $j \in \mathcal{S}_i$ **do**

$$\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$$

$$\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$$

for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

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

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

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

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

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

$$F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$$

$$F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$$

for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

$$F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$$

for i **do**

  for $j \in \mathcal{S}_i$ **do**

    $\zeta_{ij} \leftarrow 0; \quad \partial_k \zeta \leftarrow 0 \quad \forall k;$

    $\partial_i \zeta \leftarrow 0; \quad \partial_j \zeta \leftarrow 0;$

    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

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

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

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

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

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

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

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

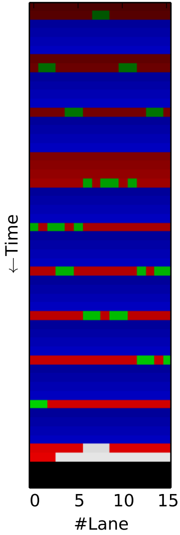
    $\delta\zeta \leftarrow \partial_\zeta V(i,j,\zeta_{ij});$

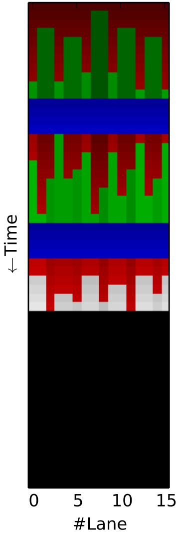
    $F_i \leftarrow F_i - \delta\zeta \cdot \partial_i \zeta;$

    $F_j \leftarrow F_j - \delta\zeta \cdot \partial_j \zeta;$

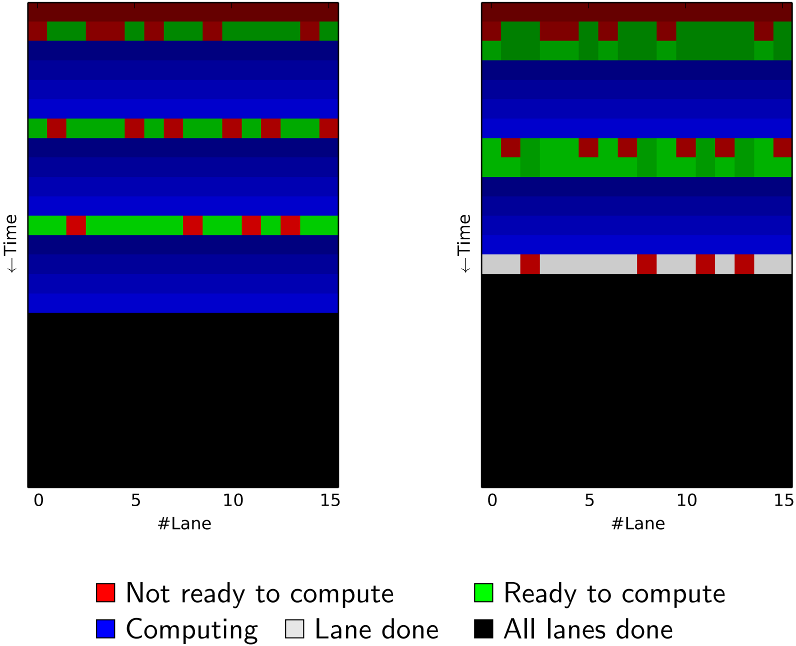
    for $k \in \mathcal{S}_i \setminus \{j\}$ **do**

      $F_k \leftarrow F_k - \delta\zeta \cdot \partial_k \zeta;$





Not ready to compute
Ready to compute
Computing
Lane done
All lanes done


←Time
0
5
10
15
#Lane
←Time
0
5
10
15
#Lane
Not ready to compute
Ready to compute
Computing
Lane done
All lanes done

Implementation

- **Vector-Wide conditionals**: Do all elements in a vector satisfy a condition?

```
int o = 1;
for (int i = 0; i < VL; i++) if (! a[i]) o = 0;
```

- **Reductions**: Sum up all elements in a vector.

```
double o = 0;
for (int i = 0; i < VL; i++) o += a[i];
```

- **Conflict write handling**: Accumulate values from vector into memory with indices from vector. `for (int i = 0; i < VL; i++) mem[b[i]] += a[i];`

- **Adjacent gather optimizations**: Write data from indices $i_1, i_2, \ldots$ into vector a, and data from $i_1 + 1, i_2 + 1, \ldots$ into b.

```
for (i = 0; i < VL; i++) { j = idx[i];
  a[i] = mem[j]; b[i] = mem[j+1]; }
```

- **Vector-Wide conditionals**: Do all elements in a vector satisfy a condition?

```
int o = 1;
for (int i = 0; i < VL; i++) if (! a[i]) o = 0;
```

- **Reductions**: Sum up all elements in a vector.

```
double o = 0;
for (int i = 0; i < VL; i++) o += a[i];
```

- **Conflict write handling**: Accumulate values from vector into memory with indices from vector. `for (int i = 0; i < VL; i++) mem[b[i]] += a[i];`

- **Adjacent gather optimizations**: Write data from indices $i_1, i_2, \ldots$ into vector a, and data from $i_1 + 1, i_2 + 1, \ldots$ into b.

```
for (i = 0; i < VL; i++) { j = idx[i];
  a[i] = mem[j]; b[i] = mem[j+1]; }
```

- **Vector-Wide conditionals**: Do all elements in a vector satisfy a condition?

```
int o = 1;
for (int i = 0; i < VL; i++) if (! a[i]) o = 0;
```

- **Reductions**: Sum up all elements in a vector.

```
double o = 0;
for (int i = 0; i < VL; i++) o += a[i];
```

- **Conflict write handling**: Accumulate values from vector into memory with indices from vector. `for (int i = 0; i < VL; i++) mem[b[i]] += a[i];`

- **Adjacent gather optimizations**: Write data from indices $i_1, i_2, \ldots$ into vector a, and data from $i_1 + 1, i_2 + 1, \ldots$ into b.

```
for (i = 0; i < VL; i++) { j = idx[i];
  a[i] = mem[j]; b[i] = mem[j+1]; }
```

- **Vector-Wide conditionals**: Do all elements in a vector satisfy a condition?

```
int o = 1;
for (int i = 0; i < VL; i++) if (! a[i]) o = 0;
```

- **Reductions**: Sum up all elements in a vector.

```
double o = 0;
for (int i = 0; i < VL; i++) o += a[i];
```

- **Conflict write handling**: Accumulate values from vector into memory with indices from vector. `for (int i = 0; i < VL; i++) mem[b[i]] += a[i];`

- **Adjacent gather optimizations**: Write data from indices $i_1, i_2, \ldots$ into vector a, and data from $i_1 + 1, i_2 + 1, \ldots$ into b.

```
for (i = 0; i < VL; i++) { j = idx[i];
  a[i] = mem[j]; b[i] = mem[j+1]; }
```

- **Vector-Wide conditionals**: Do all elements in a vector satisfy a condition?

```
int o = 1;
for (int i = 0; i < VL; i++) if (! a[i]) o = 0;
```

- **Reductions**: Sum up all elements in a vector.

```
double o = 0;
for (int i = 0; i < VL; i++) o += a[i];
```

- **Conflict write handling**: Accumulate values from vector into memory with indices from vector. `for (int i = 0; i < VL; i++) mem[b[i]] += a[i];`

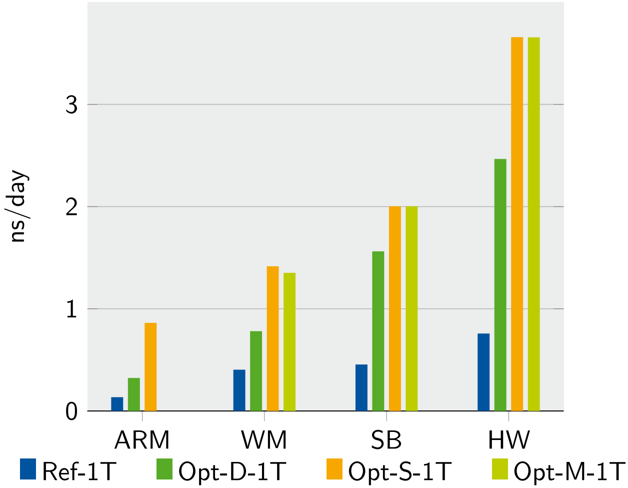
- **Adjacent gather optimizations**: Write data from indices $i_1, i_2, \ldots$ into vector a, and data from $i_1 + 1, i_2 + 1, \ldots$ into b.

```
for (i = 0; i < VL; i++) { j = idx[i];
  a[i] = mem[j]; b[i] = mem[j+1]; }
```

```
for (; kk < numneigh_i; kk++) {
  int k = firstneigh[kk + cnumneigh_i] & NEIGHMASK;
  fvec vx_k(x[k].x);
  fvec vy_k(x[k].y);
  fvec vz_k(x[k].z);
  int w_k = x[k].w;

  fvec vdx_ik = vx_k - vx_i;
  fvec vdy_ik = vy_k - vy_i;
  fvec vdz_ik = vz_k - vz_i;
  fvec vrsq = vdx_ik * vdx_ik + vdy_ik * vdy_ik
                              + vdz_ik * vdz_ik;
  ...
  if (! v::mask_testz(veff_mask)) {
    fvec vzeta_contrib = ...;
    vzeta = v::acc_mask_add(vzeta, veff_mask,
                            vzeta, vzeta_contrib);
  }
}
```

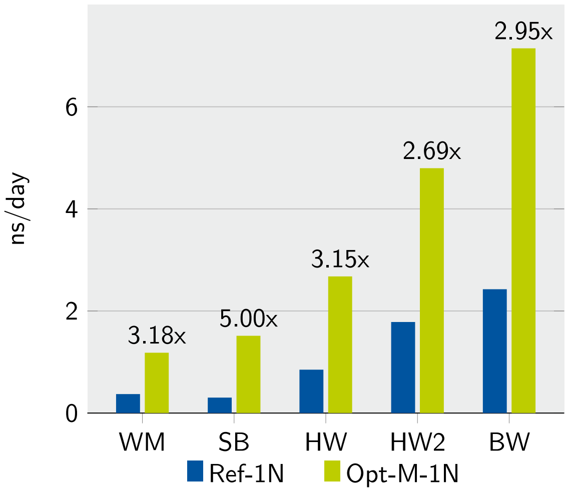
Results

CPU: Single-Threaded Execution (32 000 atoms)



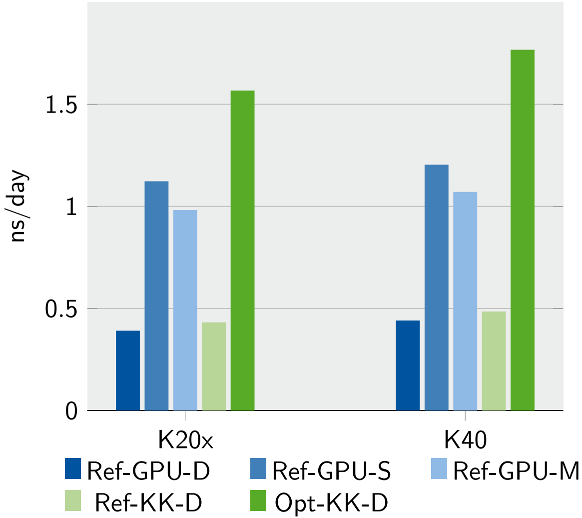
Name	Processor	Cores	Vector ISA
ARM	ARM Cortex-A15	2×4[1]	NEON
WM	Intel Xeon X5675	2×6	SSE4.2
SB	Intel Xeon E5-2450	2×8	AVX
HW	Intel Xeon E5-2680v3	2×12	AVX2

CPU: Single Node Execution (512 000 atoms)



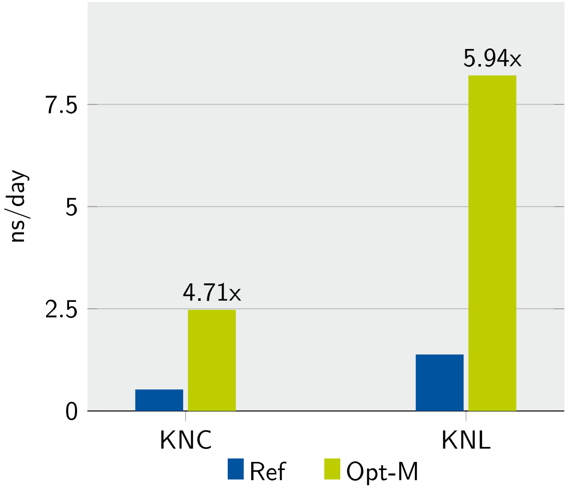
Name	Processor	Cores	Vector ISA
WM	Intel Xeon X5675	2×6	SSE4.2
SB	Intel Xeon E5-2450	2×8	AVX
HW	Intel Xeon E5-2680v3	2×12	AVX2
HW2	Intel Xeon E5-2697v3	2×14	AVX2
BW	Intel Xeon E5-2697v4	2×18	AVX2

GPU (256 000 atoms)



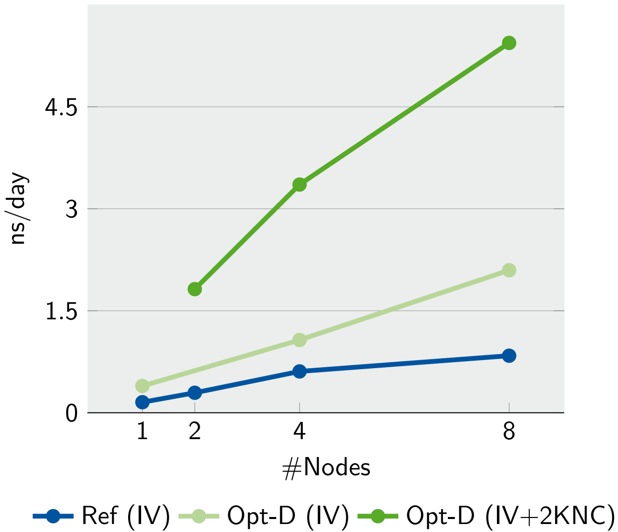
Name	CPU	Cores	ISA	Accelerator
K20X	Intel Xeon E5-2650	2 × 8	AVX	Nvidia Tesla K20x
K40	Intel Xeon E5-2650	2 × 8	AVX	Nvidia Tesla K40

Native Execution on Xeon Phi Systems (512 000 atoms)



Name	CPU	Cores	ISA	Accelerator	Cores	ISA
IV+2KNC	Intel Xeon E5-2650v2	2 × 8	AVX	Intel Xeon Phi 5110P	2 × 60	IMCI
KNL	–	–	–	Intel Xeon Phi 7250	68	AVX-512

SuperMIC: Strong Scalability (2 million atoms)



Name	CPU	Cores	ISA	Accelerator	Cores	ISA
IV+2KNC	Intel Xeon E5-2650v2	2 × 8	AVX	Intel Xeon Phi 5110P	2 × 60	IMCI

Conclusion

- Optimized Tersoff on a range of architectures
- Handling short loops (neighbor list)
- Identified necessary primitives
- Implemented using C++ abstraction
- Experimented on many architectures
- Achieved performance
- Next up: REBO/AIREBO

Preliminary Results for REBO & AIREBO

$$V^{\text{two-body}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i, j)$$

$$V^{\text{Tersoff}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$V^{\text{REBO}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k), \sum_{l \in \mathcal{S}_j \setminus \{i\}} \zeta(j, i, l), \sum_{\substack{k \in \mathcal{S}_i \setminus \{j\}, \\ l \in \mathcal{S}_j \setminus \{i, k\}}} \omega(k, i, j, l))$$

$$V^{\text{AIREBO}} = V^{\text{REBO}} + V^{\text{TORSION}} + V^{\text{LJ}}$$

V^{LJ}: Lennard-Jones augmented with many switching functions and bond-order exactly like REBO, evaluated as-if atom pair was sufficiently close. V^{TORSION}: A dihedral term.

Preliminary Results for REBO & AIREBO

$$V^{\text{two-body}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j)$$

$$V^{\text{Tersoff}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i,j,k))$$

$$V^{\text{REBO}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i,j,k), \sum_{l \in \mathcal{S}_j \setminus \{i\}} \zeta(j,i,l), \sum_{\substack{k \in \mathcal{S}_i \setminus \{j\}, \\ l \in \mathcal{S}_j \setminus \{i,k\}}} \omega(k,i,j,l))$$

$$V^{\text{AIREBO}} = V^{\text{REBO}} + V^{\text{TORSION}} + V^{\text{LJ}}$$

V^{LJ}: Lennard-Jones augmented with many switching functions and bond-order exactly like REBO, evaluated as-if atom pair was sufficiently close. V^{TORSION}: A dihedral term.

Preliminary Results for REBO & AIREBO

$$V^{\text{two-body}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j)$$

$$V^{\text{Tersoff}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i,j,k))$$

$$V^{\text{REBO}} = \sum_{i} \sum_{j \in \mathcal{S}_i} V(i,j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i,j,k), \sum_{l \in \mathcal{S}_j \setminus \{i\}} \zeta(j,i,l), \sum_{\substack{k \in \mathcal{S}_i \setminus \{j\}, \\ l \in \mathcal{S}_j \setminus \{i,k\}}} \omega(k,i,j,l))$$

$$V^{\text{AIREBO}} = V^{\text{REBO}} + V^{\text{TORSION}} + V^{\text{LJ}}$$

V^{LJ}: Lennard-Jones augmented with many switching functions and bond-order exactly like REBO, evaluated as-if atom pair was sufficiently close. V^{TORSION}: A dihedral term.

Preliminary Results for REBO & AIREBO

$$V^{\text{two-body}} = \sum_i \sum_{j \in \mathcal{S}_i} V(i, j)$$

$$V^{\text{Tersoff}} = \sum_i \sum_{j \in \mathcal{S}_i} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k))$$

$$V^{\text{REBO}} = \sum_i \sum_{j \in \mathcal{S}_i} V(i, j, \sum_{k \in \mathcal{S}_i \setminus \{j\}} \zeta(i, j, k), \sum_{l \in \mathcal{S}_j \setminus \{i\}} \zeta(j, i, l), \sum_{\substack{k \in \mathcal{S}_i \setminus \{j\}, \\ l \in \mathcal{S}_j \setminus \{i, k\}}} \omega(k, i, j, l))$$

$$V^{\text{AIREBO}} = V^{\text{REBO}} + V^{\text{TORSION}} + V^{\text{LJ}}$$

V^{LJ}: Lennard-Jones augmented with many switching functions and bond-order exactly like REBO, evaluated as-if atom pair was sufficiently close. V^{TORSION}: A dihedral term.

Preliminary Results for REBO & AIREBO

Both the optimization of REBO and AIREBO is currently WIP.

By now, we achieve a 2.5x total speedup for REBO on KNL for a representative sequential simulation run.

Due to the need for ghost neighbors and a large distance of possbile interactions, the simulation is dominated by communication and neighbor list construction, which fundamentally limit the speedups achieveable with vectorization.

Done. Time for your questions...

Markus Höhnerbach Ahmed E. Ismail **Paolo Bientinesi**

`http://github.com/hpac/lammps-tersoff-vector`