Shift-Collapse Algorithm

Manaschai Kunaseth

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
University of Southern California
&
National naotechnology Center (NANOTEC), Thiland

Email: manaschai@nanotec.or.th

M. Kunaseth *et al*., *ACM/IEEE supercomputing, SC13*

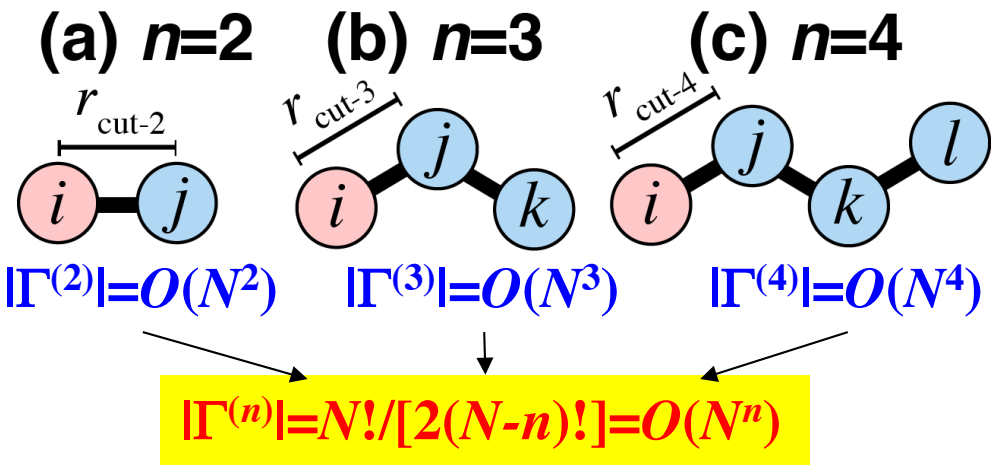
Dynamic *n*-Tuple Computation

Force computation of *n*-body potential term requires *n*-tuple of atomic positions: $(\mathbf{r}_0,\mathbf{r}_1,\ldots,\mathbf{r}_{n-1})$

$$\mathbf{f}_i^{(n)} = -\sum_{\forall(\mathbf{r}_0,\ldots,\mathbf{r}_{n-1})\in\Gamma^{(n)}} \frac{\partial}{\partial \mathbf{x}_i}\Phi_n(\mathbf{x}_0,\ldots,\mathbf{x}_{n-1})\Big|_{(\mathbf{x}_0,\ldots,\mathbf{x}_{n-1})=(\mathbf{r}_0,\ldots,\mathbf{r}_{n-1})}$$

MD problem statement: Given a set on *N* atoms, find a particular set of *n*-tuples:

- ***n*-tuple space Γ: exponential in *n*, $O(N^n)$**
- ***n*-tuple lists are dynamically constructed every MD step**
- **Many MD problems (*e.g.* bio-MD) only consider dynamic pairs $n = 2$ and static lists of $n > 2$.**



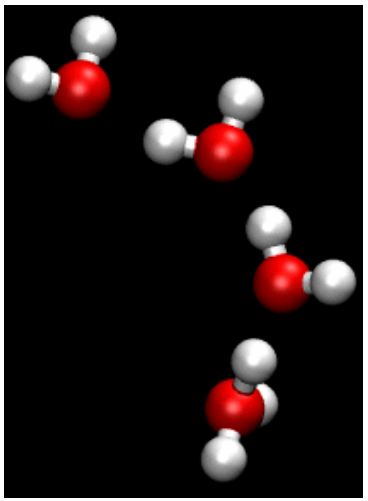
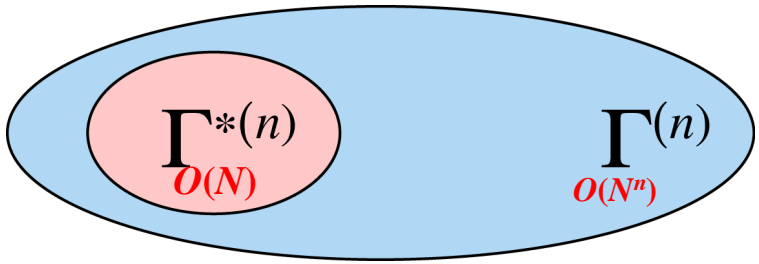


Illustration of dynamic *n*-tuple (*n*=3)

Ranged-Limited *n*-Tuples

Atom interaction in many system are short-ranged: $\Gamma^{*(n)} = \left\{ (\mathbf{r}_0, \ldots, \mathbf{r}_{n-1}) \,\middle|\, r_{k,k+1} < r_{\text{cut}-n} \right\}$

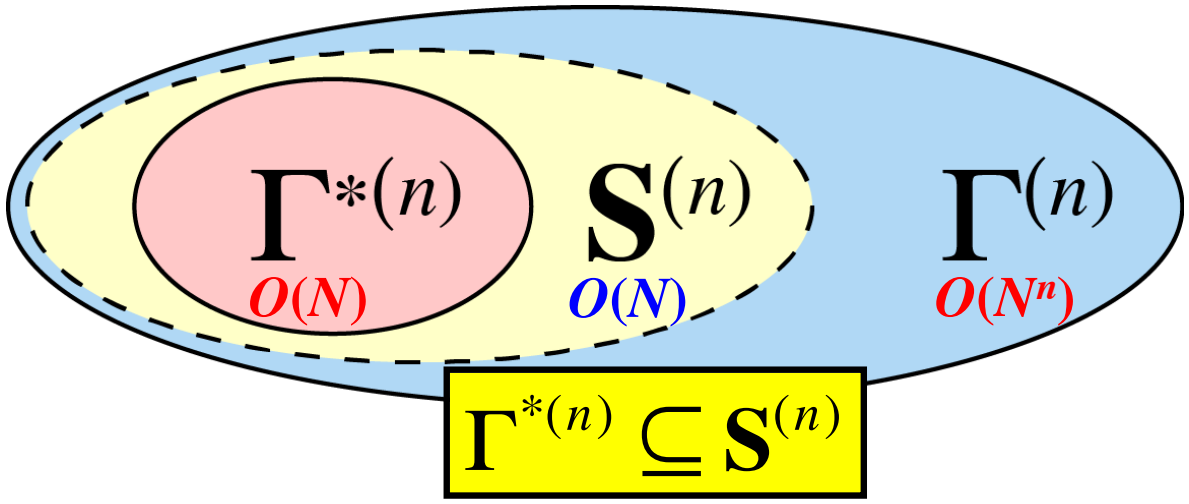
- **Only atoms within a certain cutoff distance are considered for force computation**
- **The range-limited *n*-tuple set $\Gamma^{*(n)}$: $|\Gamma^{*(n)}| = O(N)$**
- **Exhaustive search $\Gamma^{*(n)}$ is intractable**



Cell method: find a super set $S^{(n)}$ (*i.e. force set*) that tightly bounds $\Gamma^{*(n)}$:

- **Prune $\Gamma^{(n)}$ to obtain $S^{(n)}$, then exhaustive search $\Gamma^{*(n)}$**

How to obtain $S^{(n)}$ efficiently?

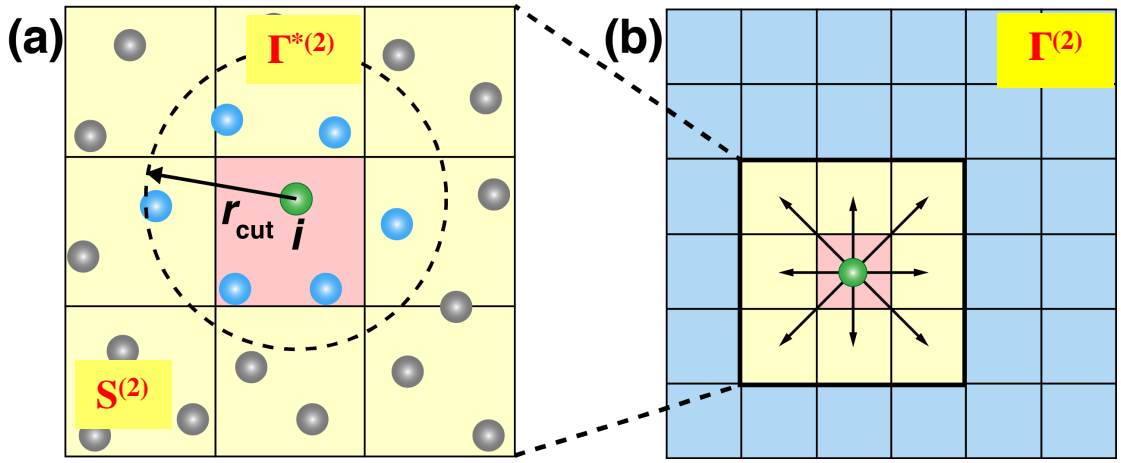


Pair-Space Pruning: Cell Method

Cell method: Divide system into small non-overlapping cells of size ≥ $r_{cut\text{-}n}$:

- **For each atom in a cell, all of its range-limited pairs are guaranteed be within the nearest-neighbor cells**
- **By looping over all cells in the system, all range-limited pairs are enumerated**
- **Reduce search complexity from $O(N^2)$ ➔ $O(N)$**

Conventional cell methods



Cell data structure:
$\mathbf{c}(\mathbf{q}) = \{\mathbf{r}_i \,|\mathbf{r}_i$ is in the volume$\}$

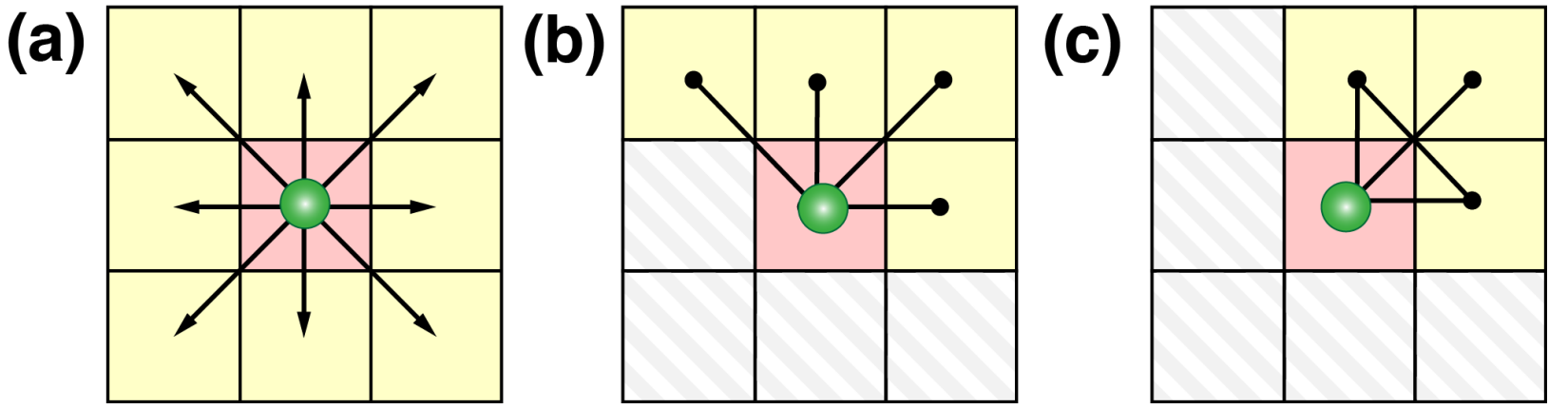
Loop over all cells in the system Ω

USC Viterbi School of Engineering

Related Works: Pair Computation Case

Cell methods for pair computation has been used extensively:

- **FS and HS: "Owner-compute" rule for pair computation** (Rappaport, 1988)
- **Force decomposition: Non "owner-compute" rule approach** (Plimpton, 1995)
- **Hybrid spatial & force decomposition** (Kale *et al.*, 2002), (Snir, 2004)
- **Neutral-territory method: optimal for low latency networks** (Shaw, 2005)
- **Eighth-shell (ES) method: Best available cell-based** (Bower *et al.*, 2006),(Hess *et al.*, 2008)



Full-Shell (FS) Method
(Rappaport, 1988)

Half-Shell (HS) Method
(Rappaport, 1988)

Eighth-Shell (ES) Method
(Bower *et al.*, 2006)

Redundancy removal

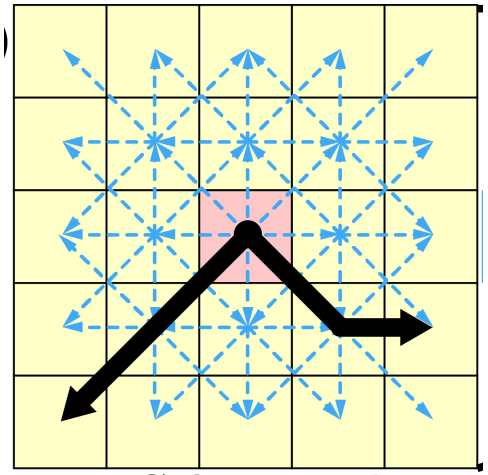
Footprint reduction

Issue: Limited Study on *n*-Tuple Computation

Interaction for $n > 2$ is complicated. Only simple FS is trivial in term of correctness:

- **Redundant searches**
- **Large cell footprint ➔ large import volume in parallel runs**
- **Very low performance: not feasible for large-scale and/or long-time simulation**

Range-limited property of MD



FS for $n = 3$

Research question:

How can we generalize the computation-redundancy removal in the HS scheme and the footprint reduction in the ES scheme developed for pair computation into arbitrary dynamic *n*-tuple computations?

Computation-Pattern Algebraic Framework

A general *n*-tuple computation formulation:

- **Formulate computation in terms of vector algebra**
- **Allow mathematically rigorous proofs of correctness and optimality**

Unified description of *n*-tuple computation within uniform cell pattern (UCP) framework:

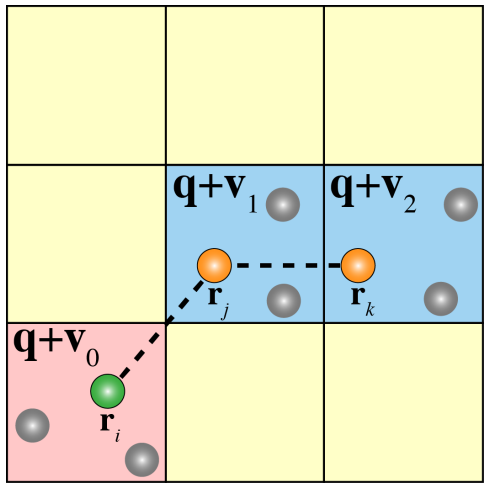
- **Generalize FS, HS, ES in the case of $n = 2$ in to general n**

Uniform Cell-Pattern (UCP) Framework

Framework to enumerate set of *n*-tuples $S^{(n)}$ (*i.e.* force set) :

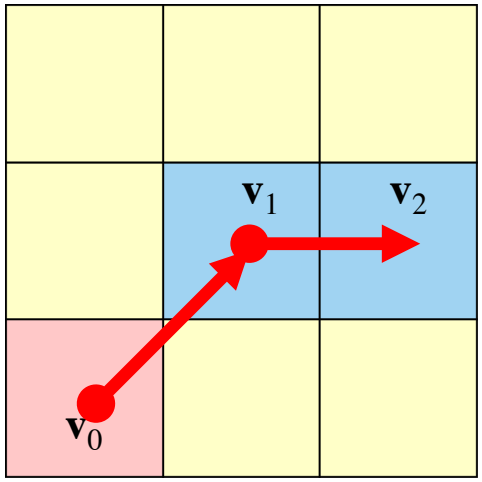
- **Abstraction of *n*-tuples in terms of cell list**
- **Generate *n*-tuples from "pattern" and "cell data"**

Set of $(\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2)$



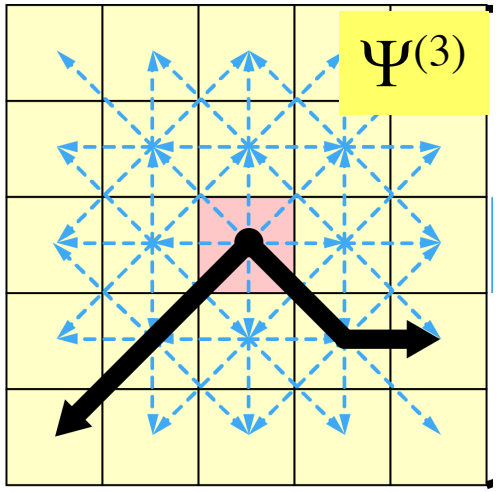
Triplets contained in 3 cells

$\mathbf{p}^{(3)} = (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$



Computation path for $n = 3$

$\Psi^{(n)} = \{\mathbf{p}^{(n)}\}$



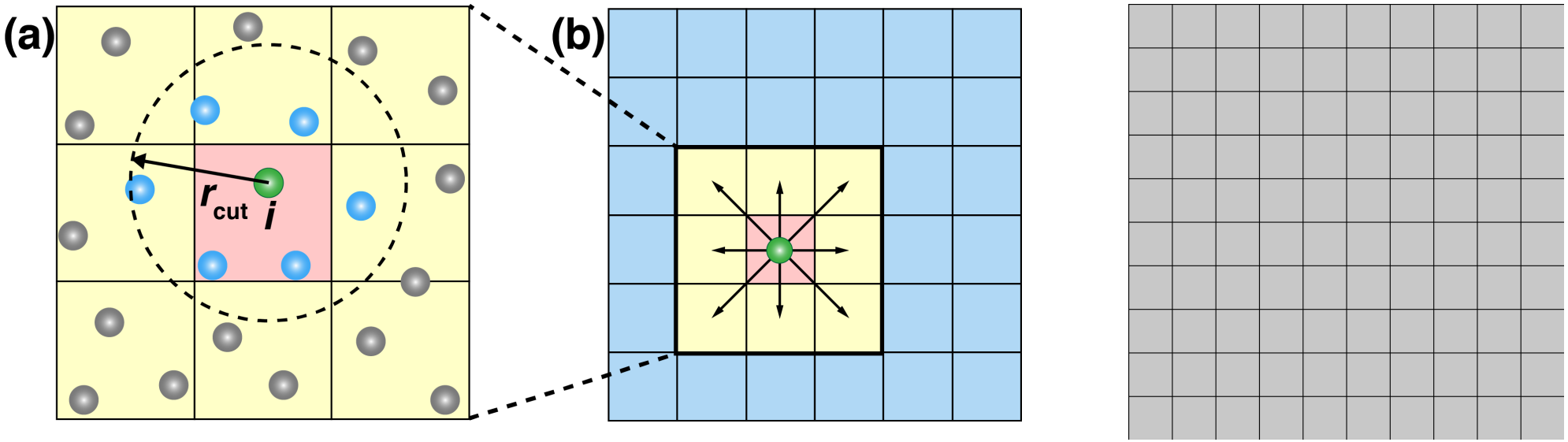
Computation pattern for $n = 3$

$$\mathbf{S}_{\text{cell}}(\mathbf{c}(\mathbf{q}), \Psi^{(n)}) = \left\{ (\mathbf{r}_0, \ldots, \mathbf{r}_{n-1}) \;\middle|\; \begin{array}{c} \forall \mathbf{p} = (\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}) \in \Psi^{(n)} \\ \forall k \in \{0, \ldots, n-1\} : \forall \mathbf{r}_k \in \mathbf{c}(\mathbf{q} + \mathbf{v}_k) \end{array} \right\}$$

$$\mathbf{S}^{(n)} = \text{UCP}(\Omega, \Psi^{(n)}) = \bigcup_{\forall \mathbf{c}(\mathbf{q}) \in \Omega} \mathbf{S}_{\text{cell}}(\mathbf{c}(\mathbf{q}), \Psi^{(n)})$$

UCP Force-Set Generation

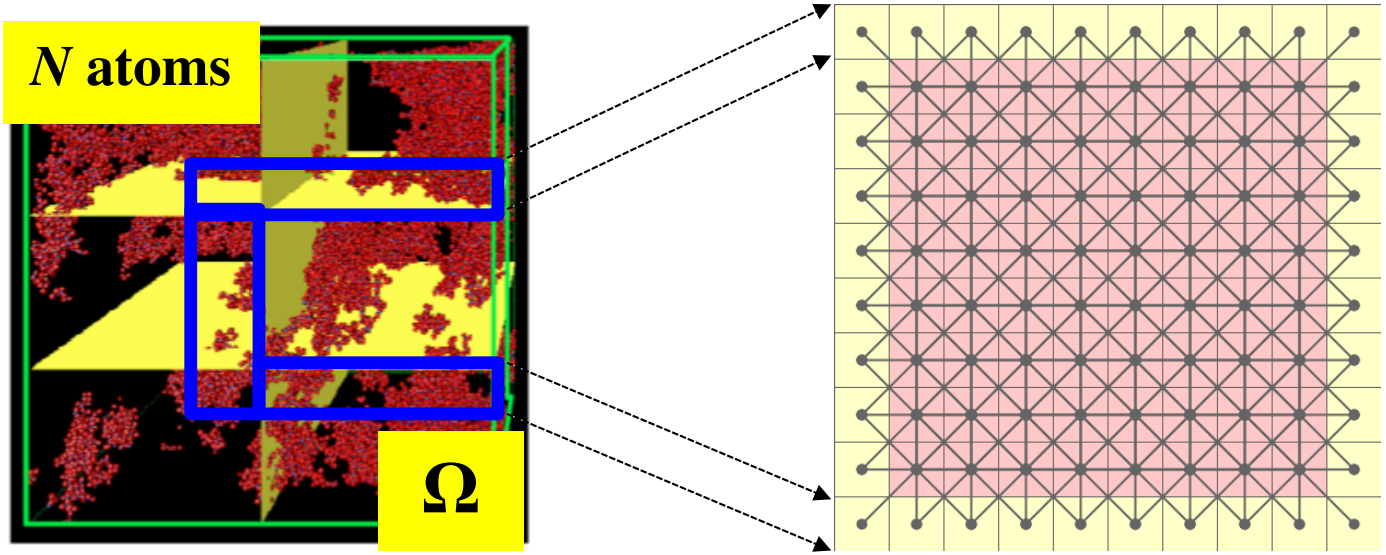
- Apply a pattern to all cells in the system to obtain force set $S^{(n)}$
- Correctness of UCP: $S^{(n)}$ must contain all range-limited tuples: $\Gamma^{*(n)} \subseteq S^{(n)}$
- Unnecessary tuples must be filtered out
- Goal: find a pattern that minimizes the filtering process



Parallel MD

Spatial decomposition:

- **Partition *N* atoms in the system into *P* equal volumes.**
- **Each volume is assigned to a different processor**
- **Import volume: data from nearest-neighbor process (blue regions) need to be imported.**



Atoms are assigned to different processors

Data of required cells (yellow cells) are imported from the other processors

UCP-MD Optimization Problem

UCP allow us to formulate *n*-tuple MD problem mathematically:

PROBLEM (OPTIMAL UCP-MD). ***Given a cell domain Ω, find a set of n-complete computation patterns $\{\Psi^{*(n)}\}$ for all $n \in \{2,\ldots,n_{max}\}$ such that each $\Psi^{*(n)}$ satisfies the following:***

Search-space minimization:

$$\Psi^{*(n)} = \underset{\Psi^{(n)}}{\operatorname{argmin}} \left(\sum_{\forall \mathbf{c} \in \Omega} \left| \mathbf{S}_{\text{cell}}(\mathbf{c}, \Psi^{(n)}) \right| \right)$$

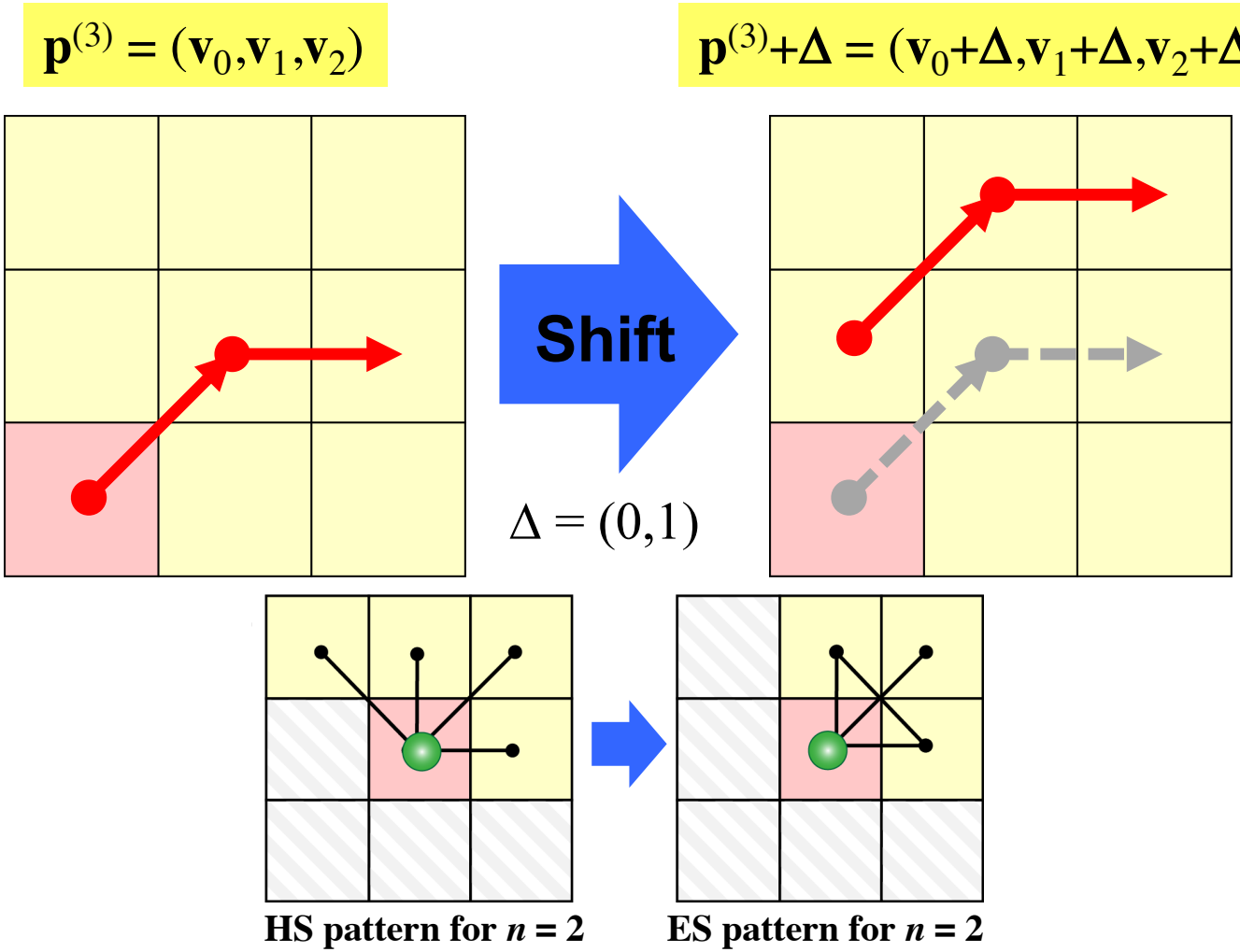
Import-volume minimization:

$$\Psi^{*(n)} = \underset{\Psi^{(n)}}{\operatorname{argmin}} \left(\left| \bigcup_{\forall \mathbf{c} \in \Omega} \Pi(\mathbf{c}, \Psi^{(n)}) - \Omega \right| \right)$$

Translation Invariance

We have proved that translation preserves the resulting force set:

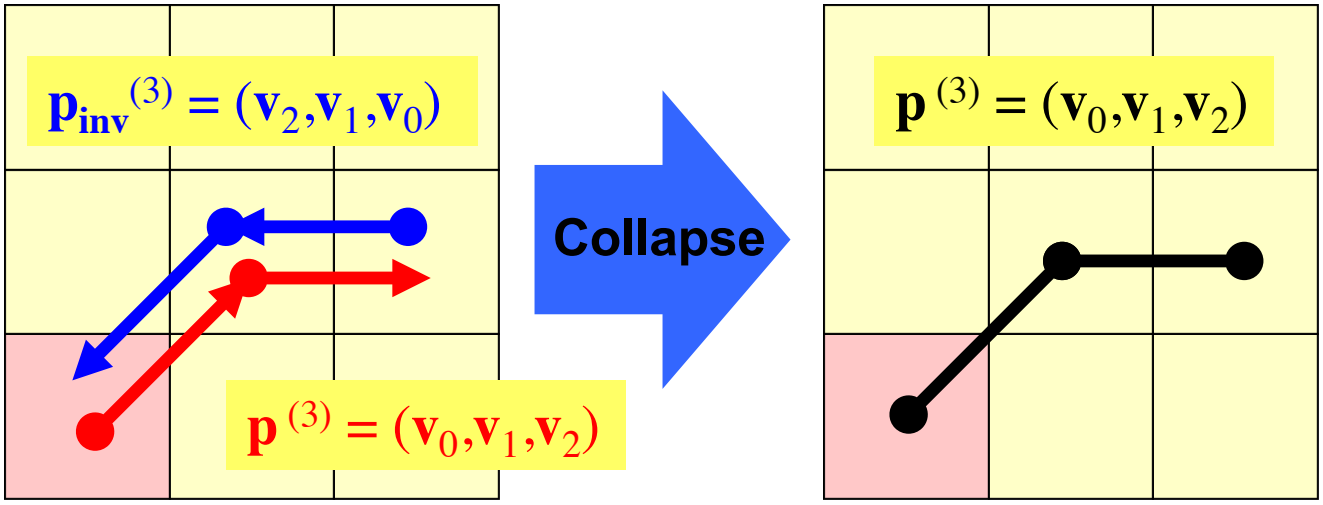
- **Translation of the origin of the computation path**
- **Manipulate paths to reduce footprint**

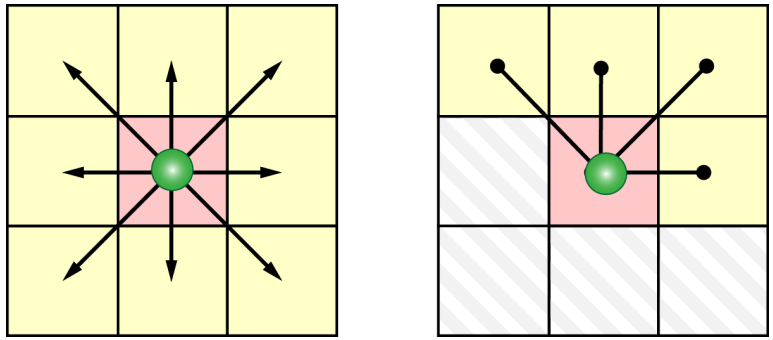


Reflection Invariance

Reflecting the Newton's third law:

- **Forces from *n*-tuple is undirectional: $(\mathbf{r}_0,\mathbf{r}_1,\ldots,\mathbf{r}_{n-1})$ produces the same forces as $(\mathbf{r}_{n-1},\mathbf{r}_{n-2},\ldots,\mathbf{r}_0)$**
- **Paths can be "collapsed" if they are in the opposite direction of each other**



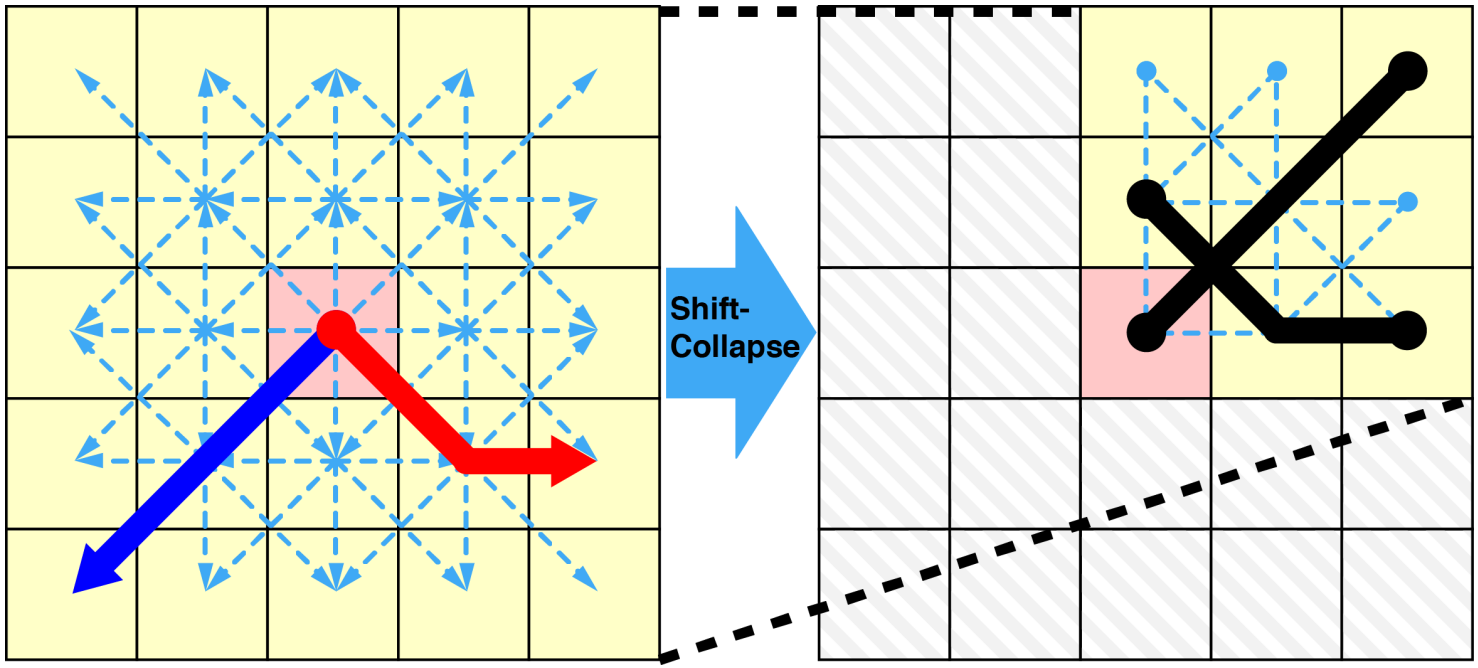


FS pattern for $n = 2$ HS pattern for $n = 2$

Shift-Collapse (SC) Algorithm

Optimal computation pattern created by the following 3 phase:

1. **Generate FS pattern**
2. **Perform octant-compression (OC) shift**
3. **Collapse all equivalent paths**



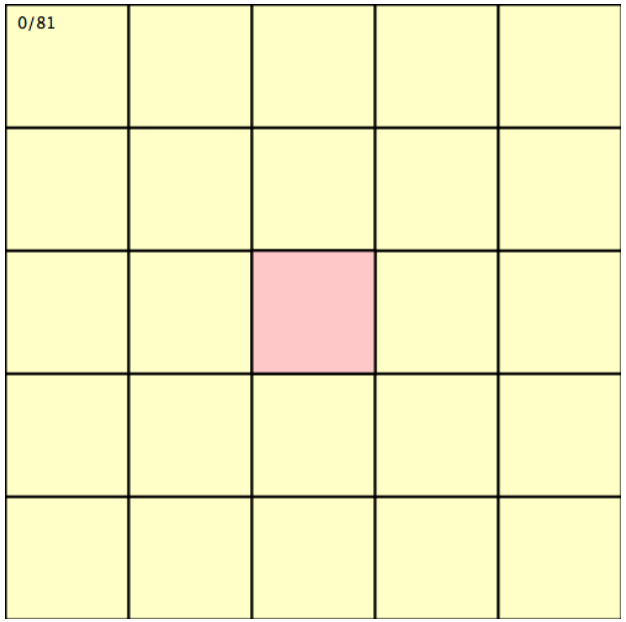
THEOREM 2. *Given an arbitrary cell domain Ω, a computation pattern $\Psi^{(n)}_{SC}$ generated from the shift-collapse algorithm is n-complete.*

Full-Shell Pattern Generation

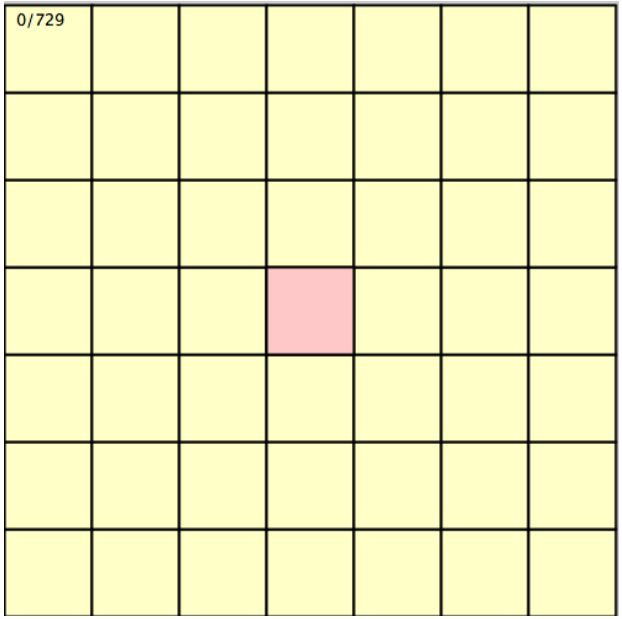
Full-shell generation algorithm:

1. **(n-1)-fold nested loop**
2. **$v_{k+1} = v_k \pm 1$; for all $k = [0, n\text{-}1]$ and $v_0 = (0,0)$**
3. **Add $(v_0, v_1, \ldots, v_{n-1})$ to the output pattern**

Generation of Ψ_{FS}: $n = 3$



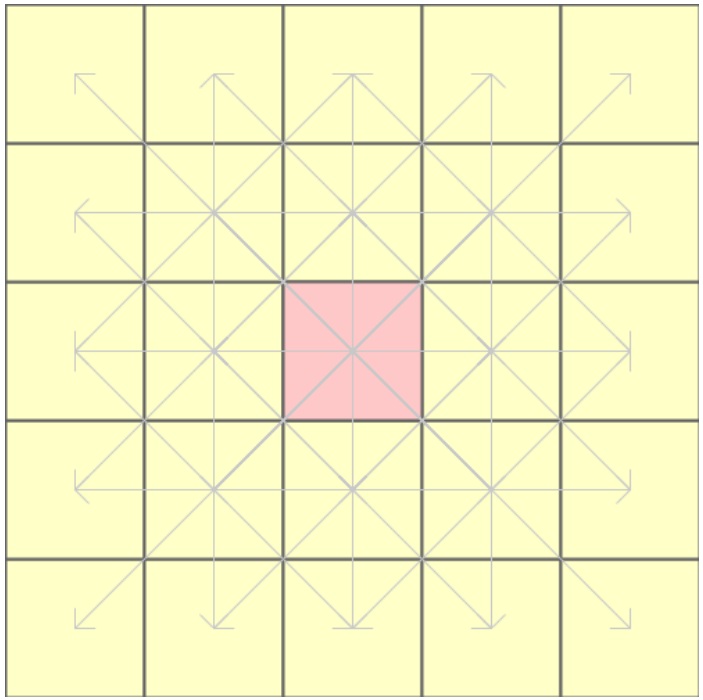
Generation of Ψ_{FS}: $n = 4$

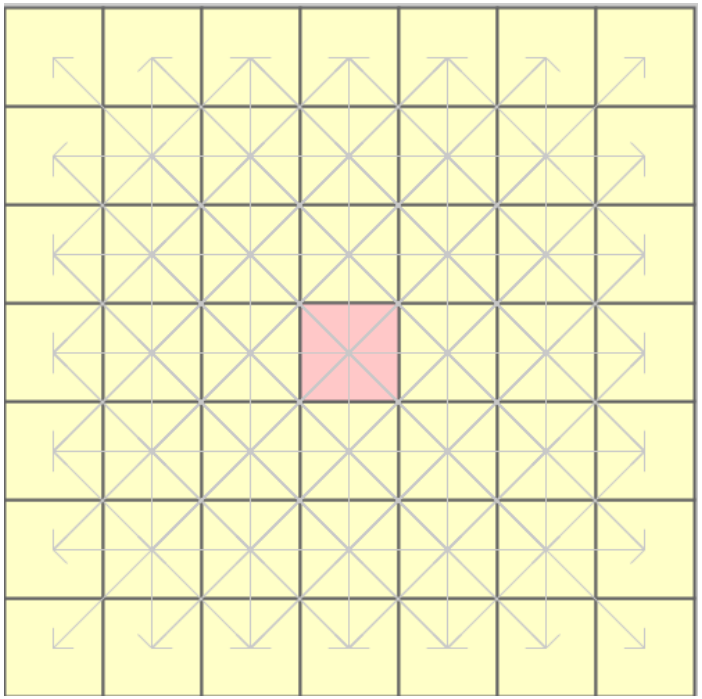


Octant-Compression Shift and Reflective Collapse

OC-shift algorithm:

1. For each path $p = (v_0, v_1, \ldots, v_{n-1})$ in a pattern $\Psi^{(n)}$
2. find $\Delta_{min} = \min(v^{\beta}_{k})$; $\forall k = \{0,\ldots,n\text{-}1\}: v_k \in p$; $\beta = \{x, y, z\}$
3. $p_{new} = p - \Delta_{min}$





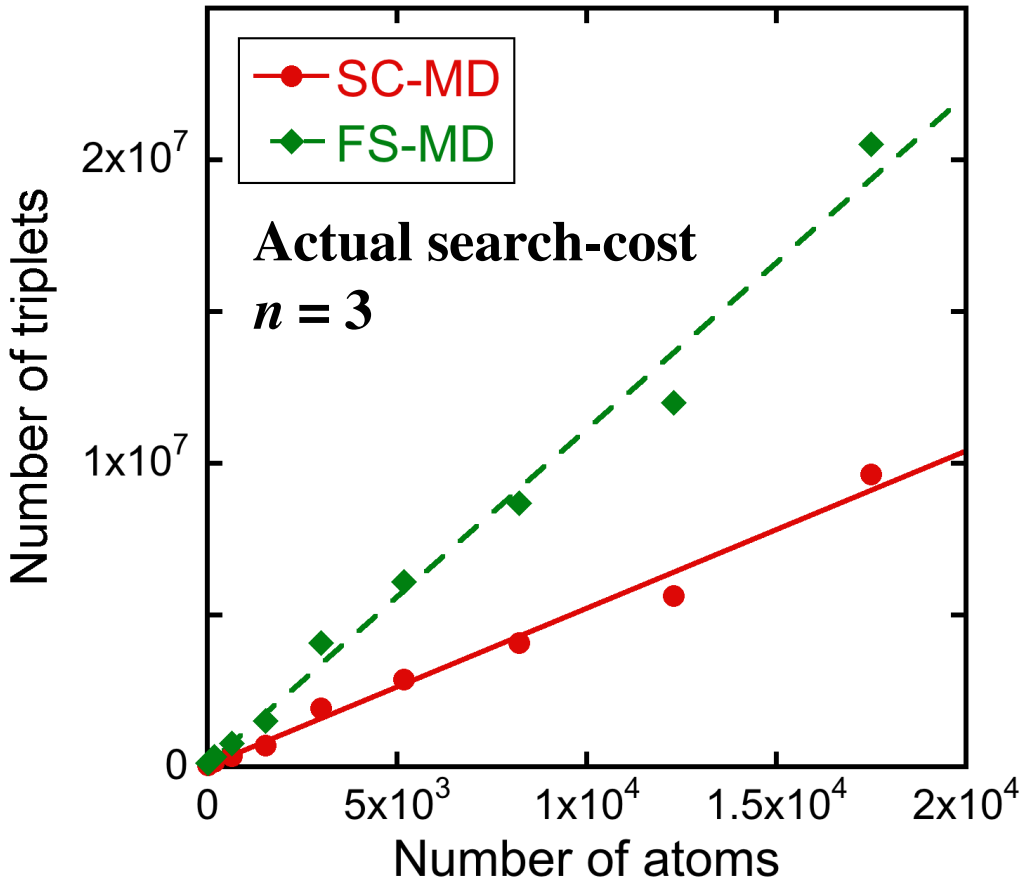
R-Collapse algorithm:

1. For each pair of paths, remove one if they are reflective to each other

Search-Cost Optimization

Search cost of UCP is the cost of filtering necessary from $S^{(n)}$

- **We have proved that search-cost is proportional to size of pattern $|\Psi^{(n)}|$**
- **Search cost of SC is about half that of FS**
- **for $n = 2$, search cost of SC is the same as search cost of HS and ES**



Analytical search cost:

- **FS pattern**

$$\left|\Psi_{\mathrm{FS}}^{(n)}\right| = 27^{n-1}$$

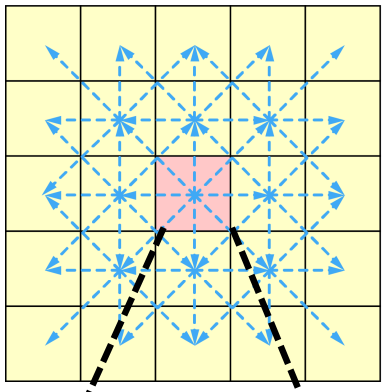
- **SC pattern**

$$\left|\Psi_{\mathrm{SC}}^{(n)}\right| = \frac{1}{2}\left(27^{n-1} - 27^{\left\lfloor \frac{n+1}{2} \right\rfloor - 1}\right) + 27^{\left\lfloor \frac{n+1}{2} \right\rfloor - 1}$$

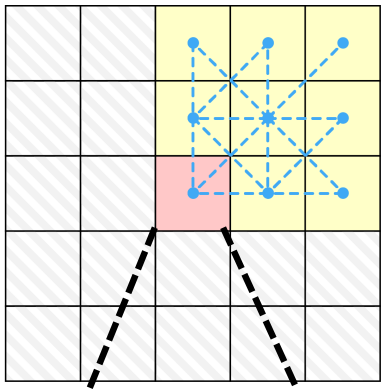
$$= \frac{1}{2}27^{n-1} + O(27^{n/2})$$

Import-Volume Optimization

- **OC shift reduces cell footprint to the most compact configuration**

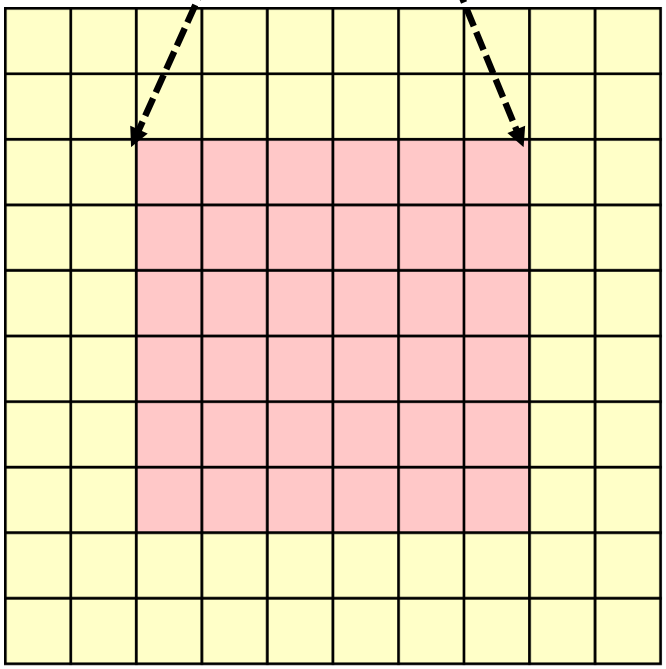


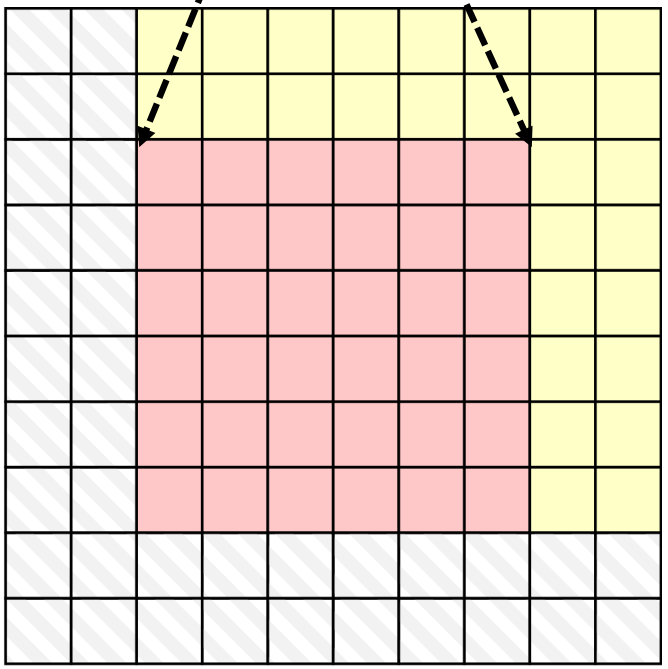
$$\frac{1}{2}V(\omega_{\mathrm{FS}}) > V(\omega_{\mathrm{SC}})$$

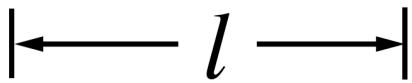


$(l+(2n-1))^3$

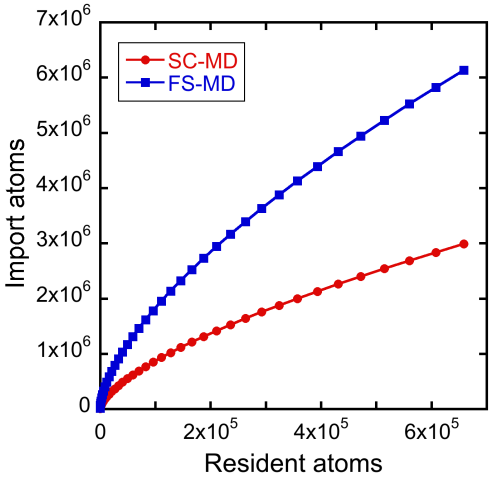
$(l+(n-1))^3$



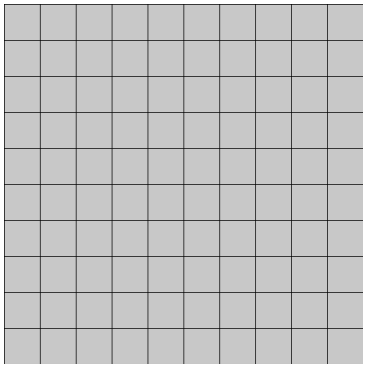




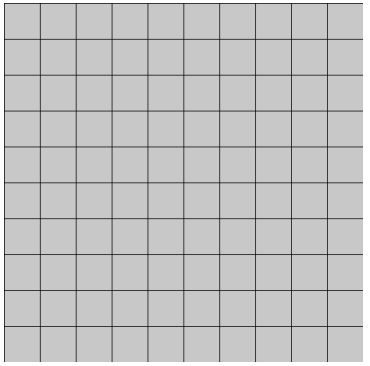
SC generalizes ES



Pair FS



ES



Performance Benchmark

Compare actual runtime of three codes for dynamic n-tuples, (n = 2, 3):

- **SC-MD: shift-collapse MD code**
- **FS-MD: Simple full-shell MD code**
- **Hybrid-MD: production code using full-shell (n = 2) + dynamic Verlet-neighbor list (n = 3) only if $r_{cut\text{-}2} > r_{cut\text{-}3}$** [SC01 Best paper]

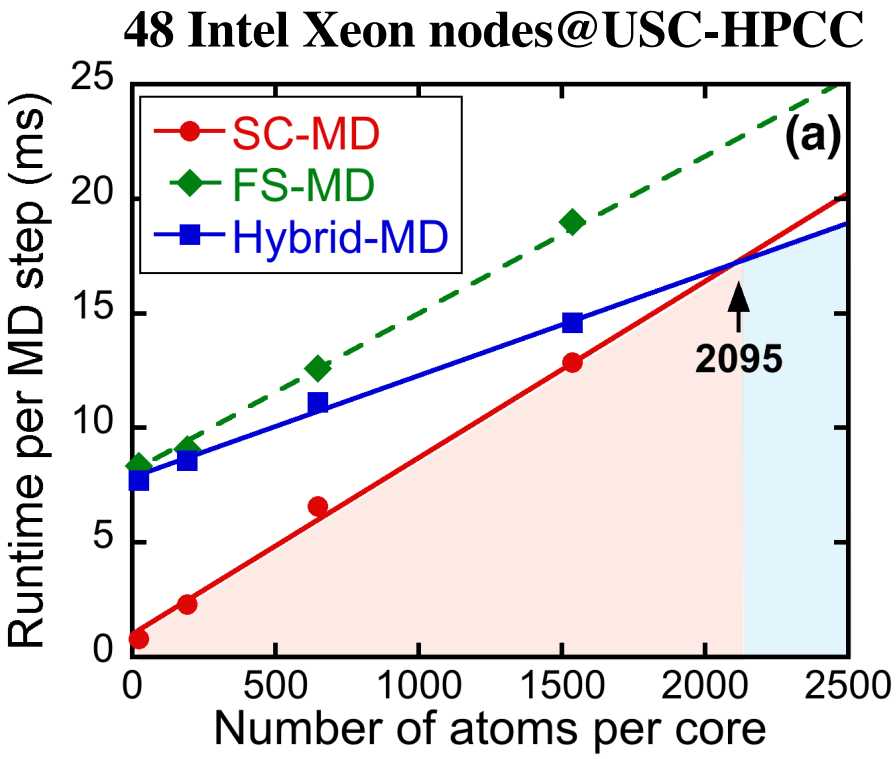
	BlueGene/Q-ANL	USC-HPCC
Processor	IBM PowerPC A2	Intel Xeon X5650
Clock Speed	1.6 GHz	2.66 GHz
Number of nodes	36,864	256
Cores per node	16	12
Memory per node	32 GB	24 GB
Network	3D torus	Myrinet 10G

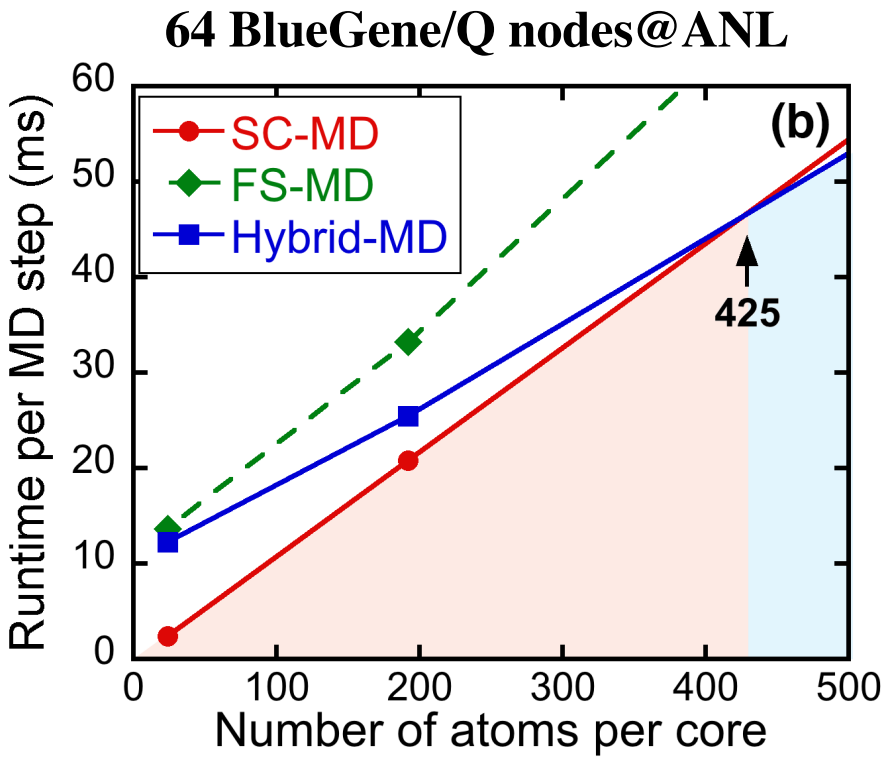
Specification of test platforms

Fine-Grain Parallelism

Runtime comparison on 48 Intel-Xeon nodes and 64 BlueGene/Q nodes

- **SC-MD is always faster than FS-MD**
- **At smallest grain, SC-MD is 9.7- and 5.1-fold speedups over hybrid code**
- **Crossover of optimal algorithm from SC-MD to Hybrid MD at larger granularity (i.e. $N/P > 2{,}095$ on Intel Xeon and $N/P > 425$)**



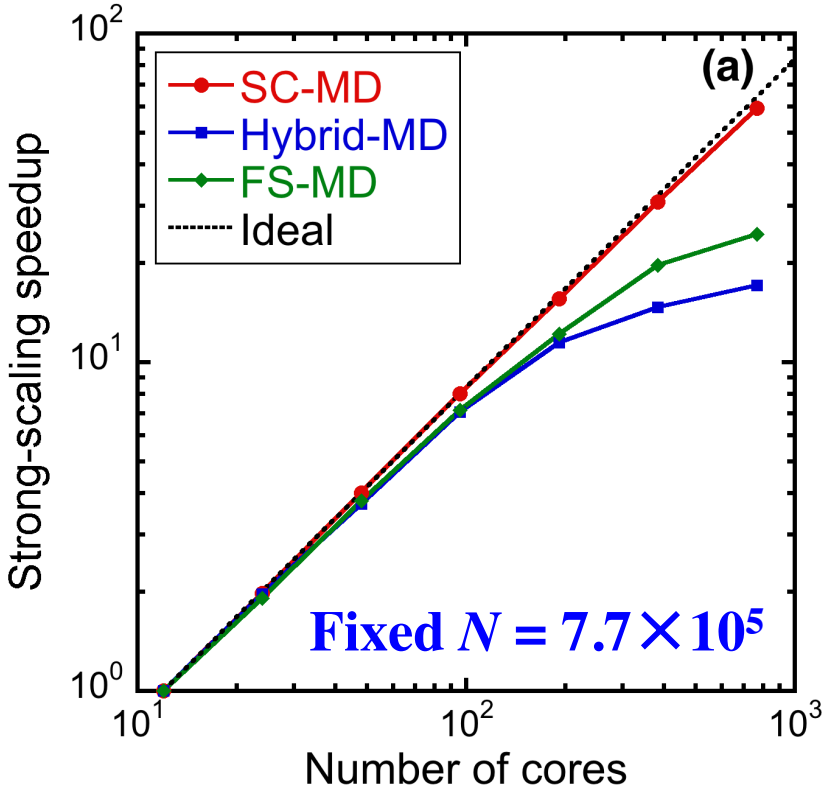


Strong Scalability

Strong-scaling benchmarks on 64 Intel-Xeon and 512 BlueGene/Q nodes:

- **SC-MD shows excellent strong scalability on both platforms: 0.90 and 0.92 parallel efficiencies**
- **Scalability of FS-MD and Hybrid MD degrade severely after 8 nodes on Xeon cluster and 16 nodes on BlueGene/Q**

1-64 Intel Xeon nodes@USC-HPCC



1-512 BlueGene/Q nodes@ANL