

Optimizing Molecular Dynamics

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations

Department of Computer Science

Department of Physics & Astronomy

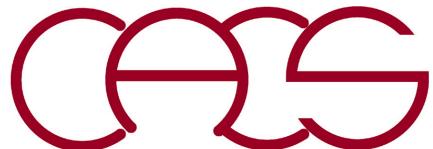
Department of Chemical Engineering & Materials Science

Department of Biological Sciences

University of Southern California

Email: anakano@usc.edu

- Intranode optimization: CPU & memory access
- Internode optimization: Communication



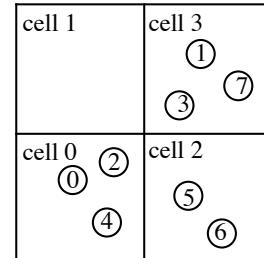
Data/computation locality!



Intranode: Memory Access

Data re-ordering

- Linked-list cells—irregular memory access pattern



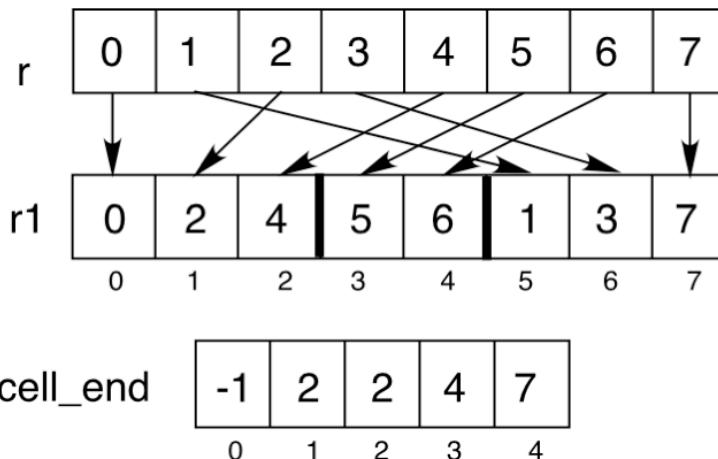
head	0	1	2	3				
	4	E	6	7				
lscl	0	1	2	3	4	5	6	7

head	0	1	2	3	4	5	6	7
	E	E	0	1	2	E	5	3

head → 7 → 3 → 1 → Empty

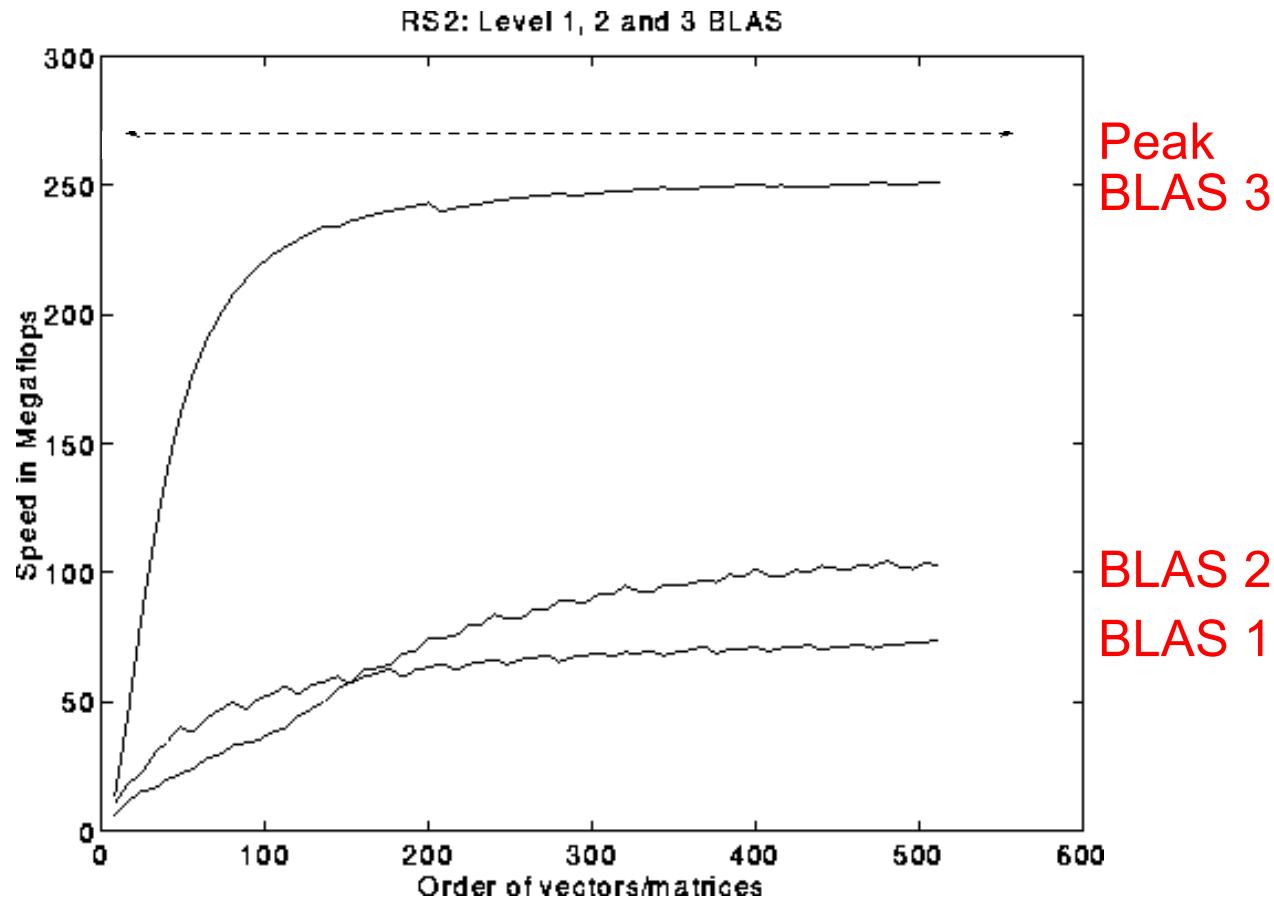
- Data locality: Regular data layout

```
for i = cell_end[c]+1 to cell_end[c+1]
  access r1[i]
endfor
```



BLAS3-Performance Molecular Dynamics?

- BLAS3: $q = \text{flop}/\text{memory access} = (\text{block size})^{1/2}$



- Molecular dynamics: $q = O(n^2)/O(n) = O(n)$: block size
 - > Use of SIMD (single instruction multiple data) instructions on Cell, multicore (SSE)?

Floating Point Performance

- **BLAS-ification:** Transform from band-by-band to all-band computations to utilize a matrix-matrix subroutine (**DGEMM**) in the BLAS3 library for the quantum molecular dynamics application
- Algebraic transformation of computations

Example: Nonlocal pseudopotential operation

D. Vanderbilt, *Phys. Rev. B* **41**, 7892 ('90)

$$\hat{v}_{\text{nl}}|\psi_n^\alpha\rangle = \sum_I^{N_{\text{atom}}} \sum_{ij}^{L_{\max}} |\beta_{i,I}\rangle D_{ij,I} \langle \beta_{j,I}| \psi_n^\alpha \rangle \quad (n = 1, \dots, N_{\text{band}})$$



$$\Psi = [|\psi_1^\alpha\rangle, \dots, |\psi_{N_{\text{band}}}^\alpha\rangle] \tilde{\mathbf{B}}(i) = [|\beta_{i,1}\rangle, \dots, |\beta_{i,N_{\text{atom}}}\rangle] [\tilde{\mathbf{D}}(i,j)]_{I,J} = D_{ij,I} \delta_{IJ}$$

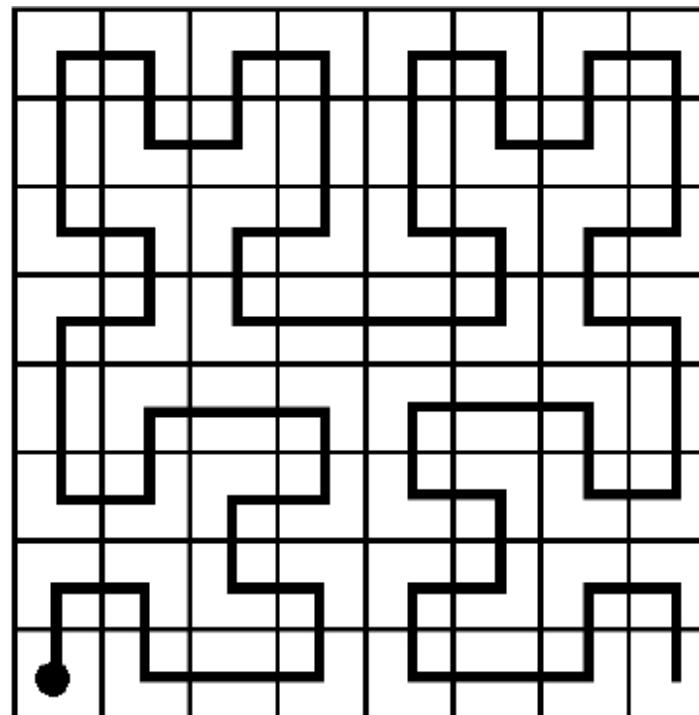
$$\hat{v}_{\text{nl}}\Psi = \sum_{i,j}^L \tilde{\mathbf{B}}(i) \tilde{\mathbf{D}}(i,j) \tilde{\mathbf{B}}(j)^T$$

- **50.5% of the theoretical peak FLOP/s performance on 786,432 Blue Gene/Q cores (entire Mira at the Argonne Leadership Computing Facility)**
- **55% of the theoretical peak FLOP/s on Intel Xeon E5-2665**

Computation Locality

Data to computation re-ordering: How to traverse cells?

- Pair-interaction computation: Preserve nearest-neighbor cells' proximity in memory
- Spacefilling curve: Mapping from the d -dimensional space to one-dimensional list to preserve spatial proximity of consecutive list elements



Hilbert-Peano Curve

- Gray code: a sequence of numbers such that successive numbers have Hamming distance 1

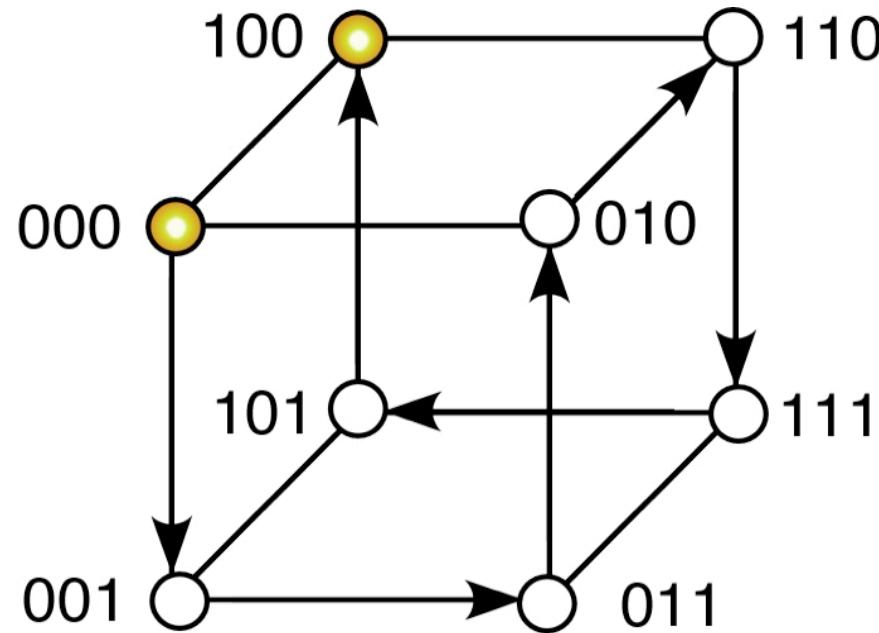
Algorithm: Recursive generation of k-bit Gray code $G(k)$

(1) $G(1)$ is a sequence: 0 1.

(2) $G(k+1)$ is constructed from $G(k)$ as follows:

- Construct a new sequence by appending a 0 to the left of all members of $G(k)$.
- Construct a new sequence by reversing $G(k)$ & then appending a 1 to the left of all members of the sequence.
- $G(k+1)$ is the concatenation of the sequences defined in steps a & b.

- $G(3)$: 000 001 011 010 110 111 101 100



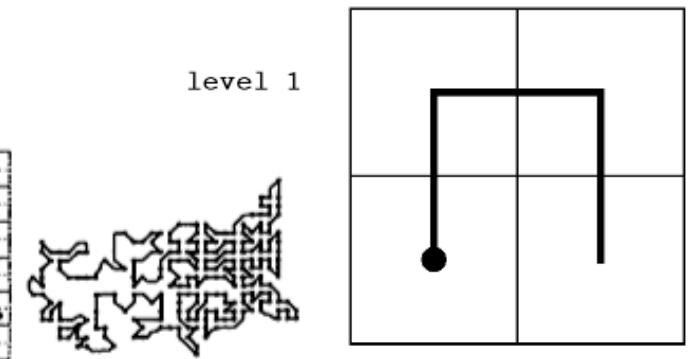
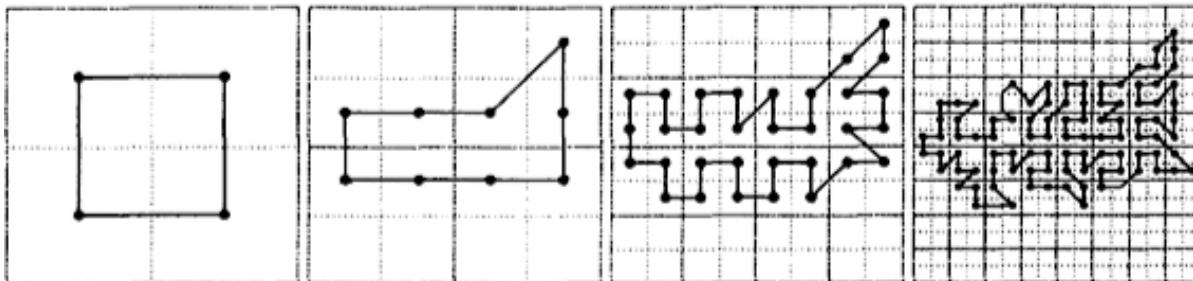
David Hilbert (1862–1943)



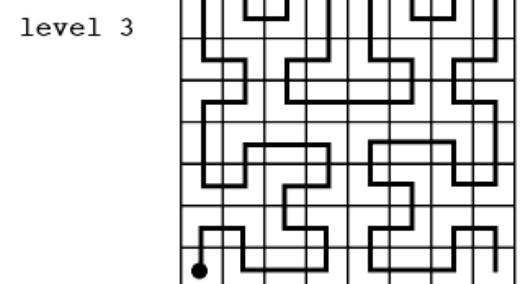
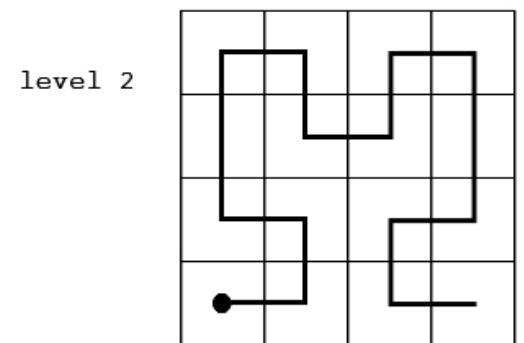
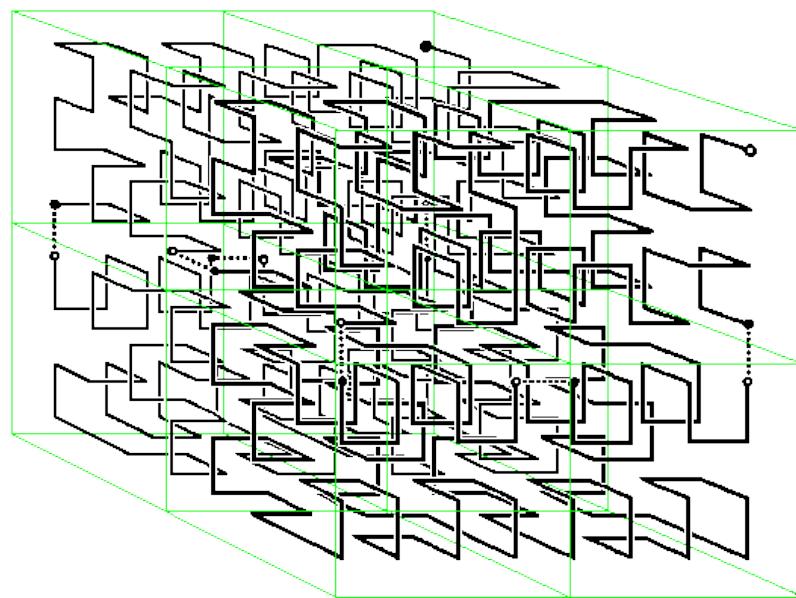
Giuseppe Peano (1858–1932)

Hilbert-Peano Curve

- Hilbert curve: recursive application of the d -dimensional Gray codes
- 2-dimensional Hilbert curve



- 3-dimensional Hilbert curve

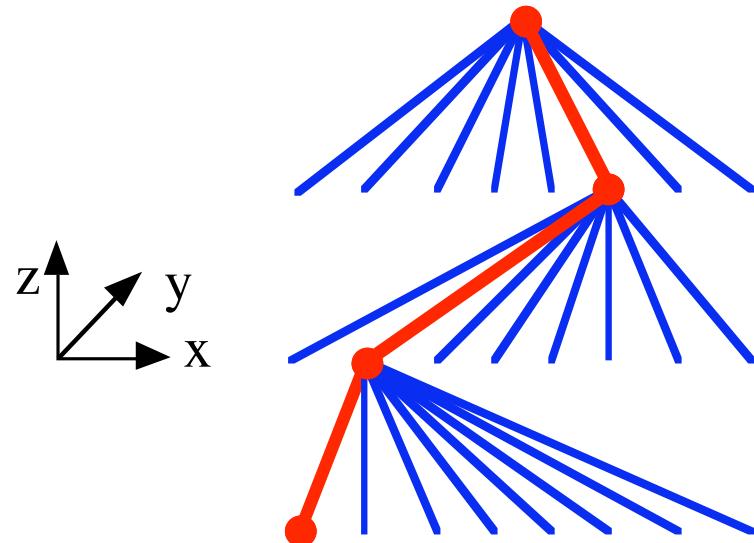
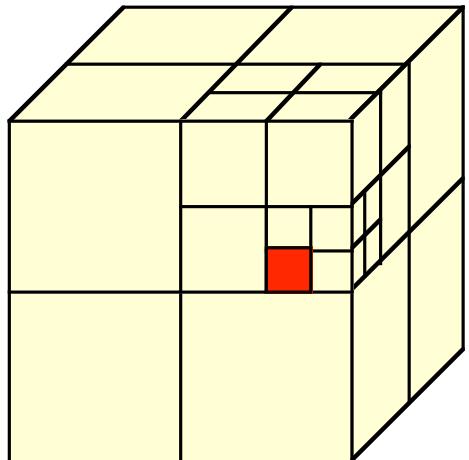


Morton (Z) Curve

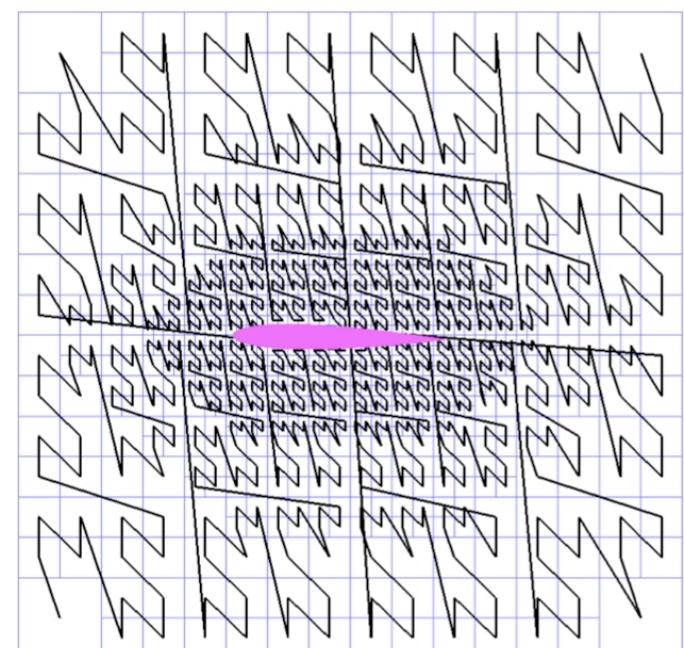
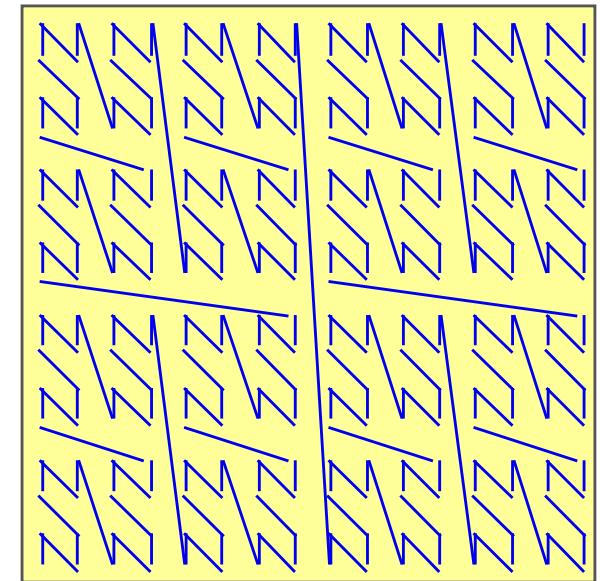
- Spacefilling curve based on octree index

$$\begin{array}{r} \mathbf{x} = \quad 1 \quad 1 \quad 0 \\ \mathbf{y} = \quad 0 \quad 0 \quad 0 \\ \mathbf{z} = 1 \quad 0 \quad 0 \\ \hline \mathbf{R} = \mathbf{101} \; \mathbf{001} \; \mathbf{000} \end{array}$$

- 3D → list map preserves spatial proximity
- Multiresolution analysis made easy



A. Omeltchenko et al., [Comput. Phys. Commun. 131, 78 \('00\)](#)



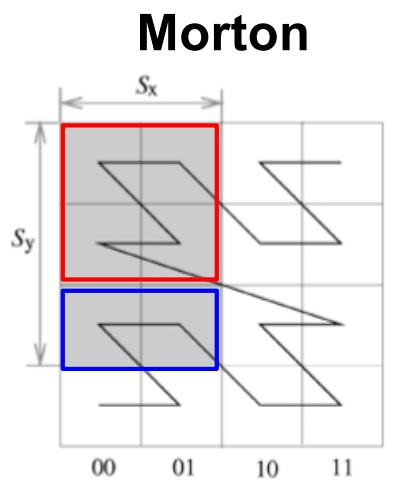
Analysis of Data Locality

124

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 13, NO. 1, JANUARY/FEBRUARY 2001

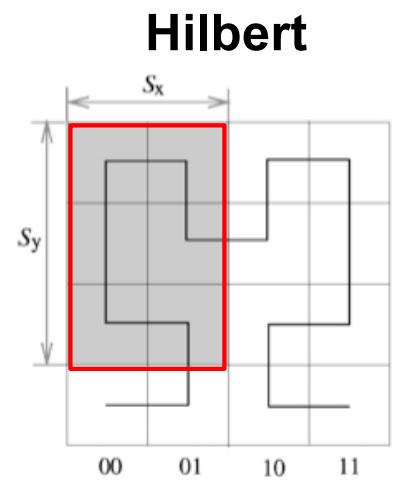
Analysis of the Clustering Properties of the Hilbert Space-Filling Curve

Bongki Moon, H.V. Jagadish, Christos Faloutsos, *Member, IEEE*, and Joel H. Saltz, *Member, IEEE*



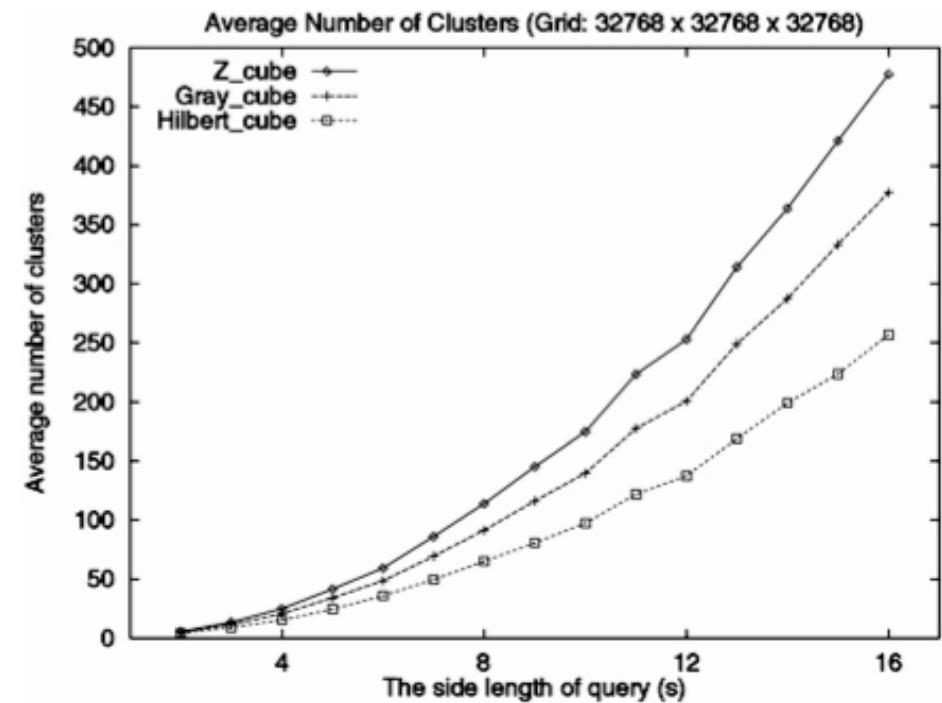
2 clusters

Cluster ~ cache line ~ latency cost



1 cluster

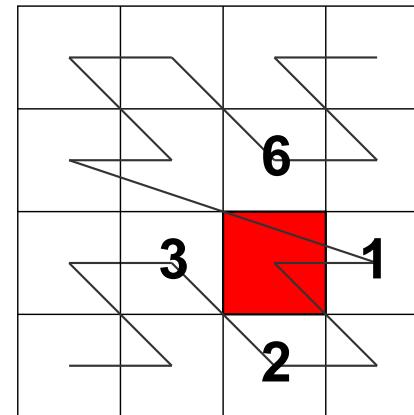
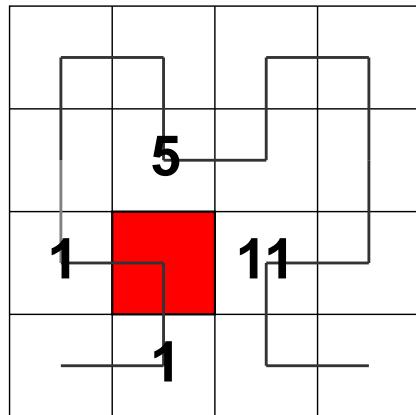
Hilbert curve is better than Morton curve for spatial range query



Alternative Locality Measure for MD

Which curve is better for spatial “pair” query?

- Evaluate curves based on curve distances to neighbors
- Compare number below & above threshold cutoff k_c (like cache)



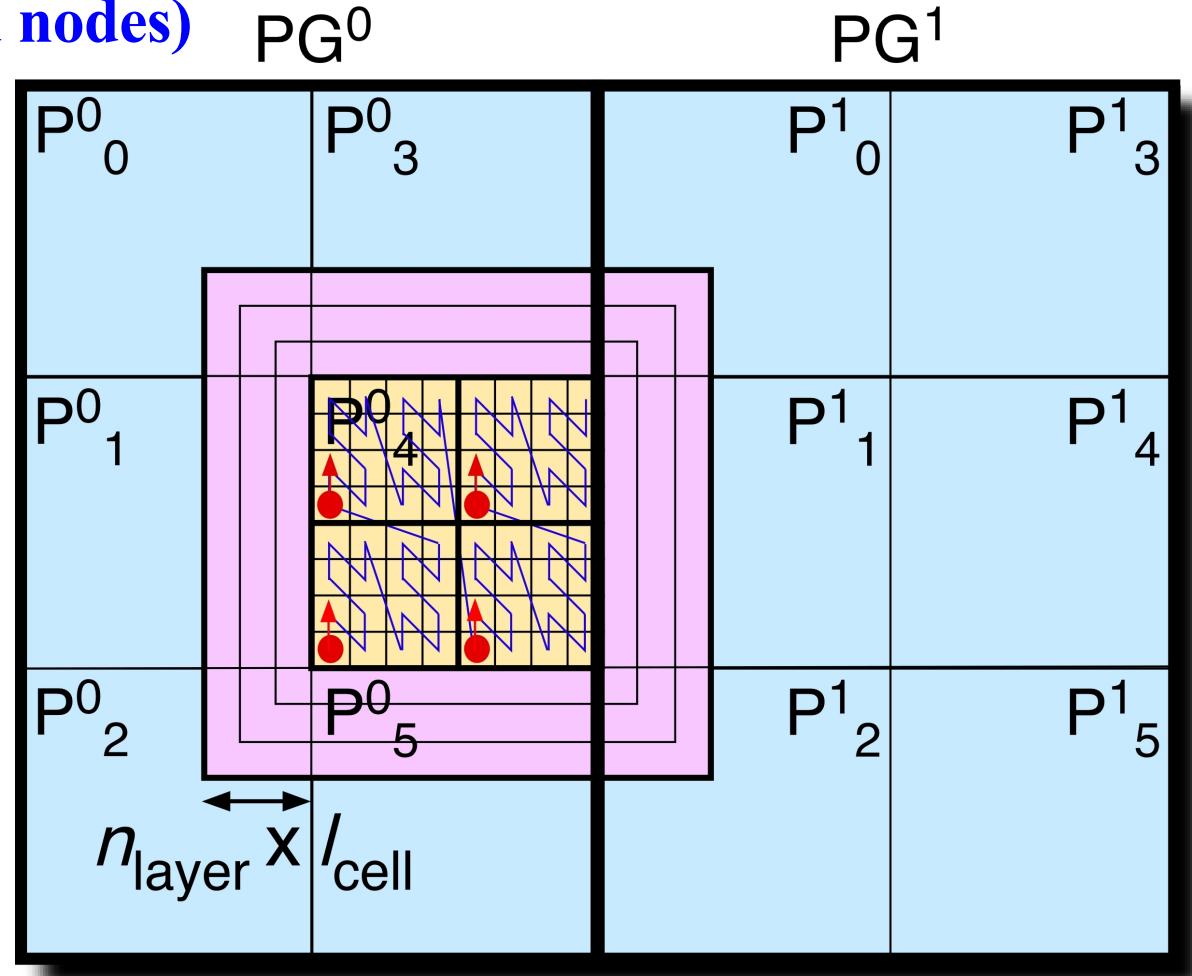
- 4x4 Hilbert:
 - 30 1s
 - 10 3s
 - 4 5s
 - 2 11s
 - 2 13s
- Lower median, higher variance
- Better for $k_c = 1$

- 4x4 Z-curve:
 - 16 1s
 - 16 2s
 - 8 3s
 - 8 6s
- Higher median, lower variance
- Better for $2 < k_c < 13$

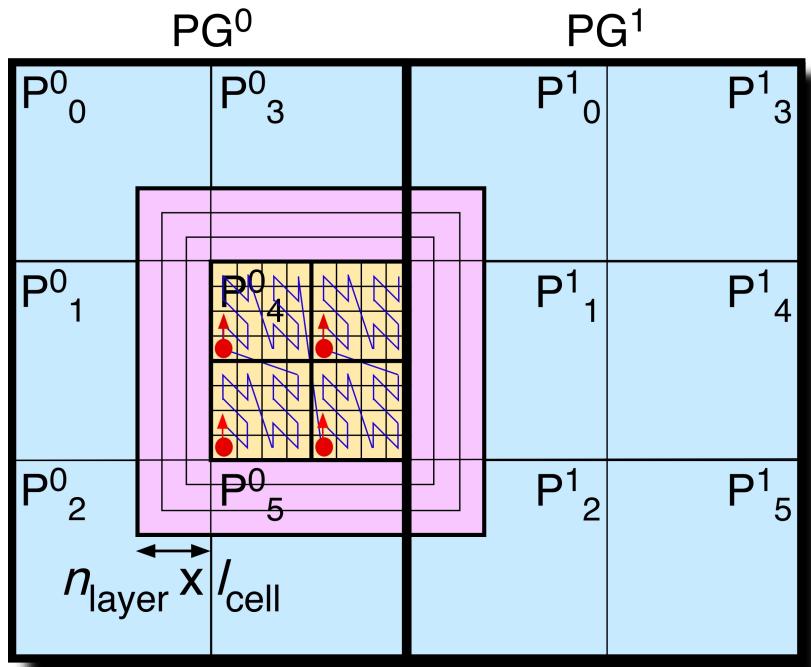
Tunable Hierarchical Cellular Decomposition

Mapping $O(N)$ divide-&-conquer algorithms onto memory hierarchies

- Spatial decomposition with data “caching” & “migration”
- Computational cells (*e.g.* linked-list cells in MD) < cell blocks (threads) < processes (P^{γ}_{π} , spatial decomposition subsystems) < process groups (P^{γ} , Grid nodes)
- Multilayer cellular decomposition (MCD) for n -tuples ($n = 2\text{--}6$)
- Tunable cell data & computation structures: Data/computation re-ordering & granularity parameterized at each decomposition level
- Tunable hybrid MPI + OpenMP + SIMD implementation

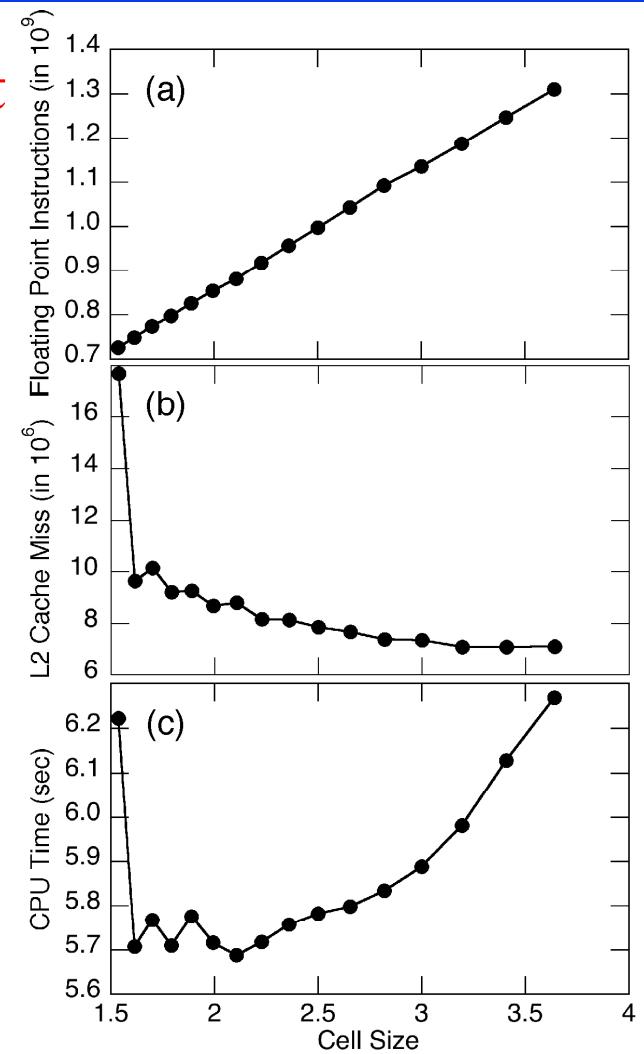


Performance Tunability



MPI/OpenMP parallelism trade-off:
8,232,000-atom silica MRMD & 290,304-atom RDX F-ReaxFF on 8-way 1.5 GHz Power4

**Floating-point operation/L2 cache miss trade-off:
 331,776-atom silica MRMD on 1.4GHz Pentium III**



Number of OpenMP threads, n_{td}	Number of MPI processes, n_p	Execution time/MD time step (sec)	
		MRMD	P-ReaxFF
1	8	4.19	62.5
2	4	5.75	58.9
4	2	8.60	54.9
8	1	12.5	120

SIMD Vectorization

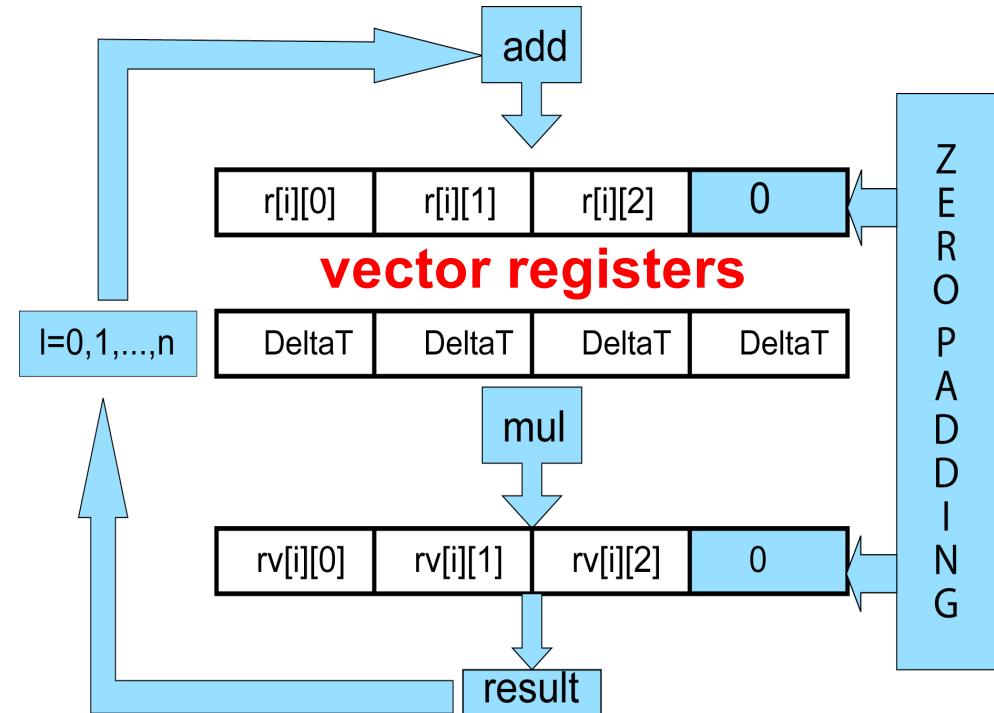
- Single-instruction multiple-data (SIMD) parallelism

(Example) Zero padding to align complex data

Original solution

```
for (i=0; i<N; i++)
    for (a=0; a<3; a++)
        r[i][a] =
        r[i][a] +
        DeltaT*rv[i][a];
```

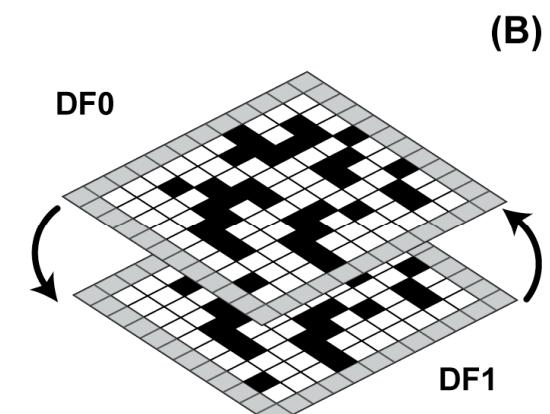
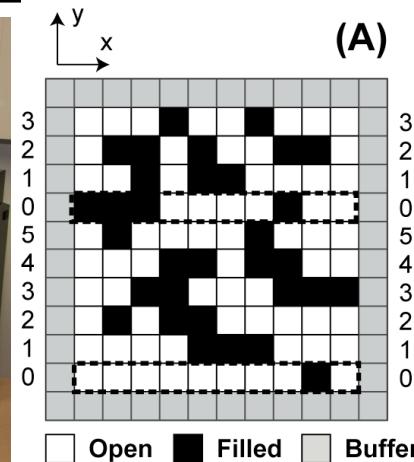
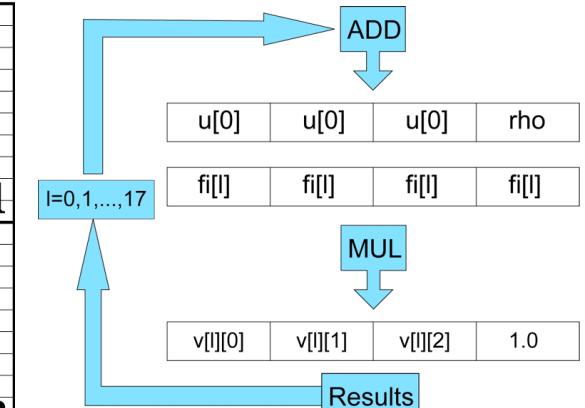
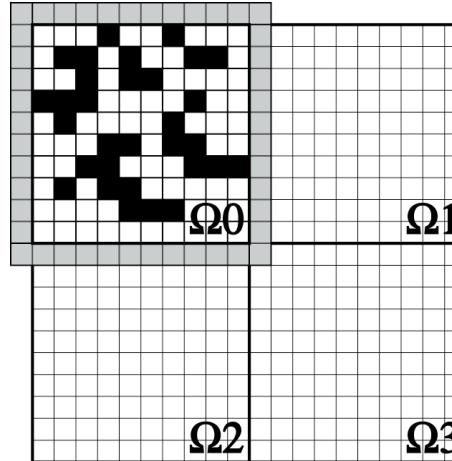
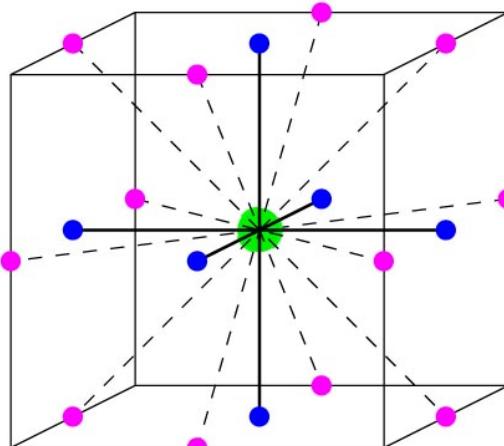
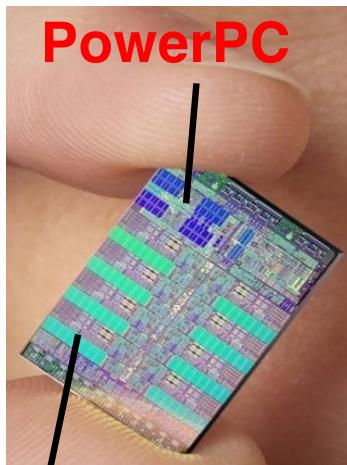
SIMD solution



cf. False-sharing avoidance

Hierarchical Parallelization

- Developed a hierarchical parallel lattice Boltzmann method (pLBM) for flow simulation on a cluster of Cell Broadband Engine-based Playstation3 consoles & IBM BlueGenes
 1. Spatial decomposition *via* message passing
 2. Multithreading through critical section-free, dual representation
 3. Single-instruction multiple data (SIMD) parallelism *via* new vector transforms



More Four-Vectors for SIMD

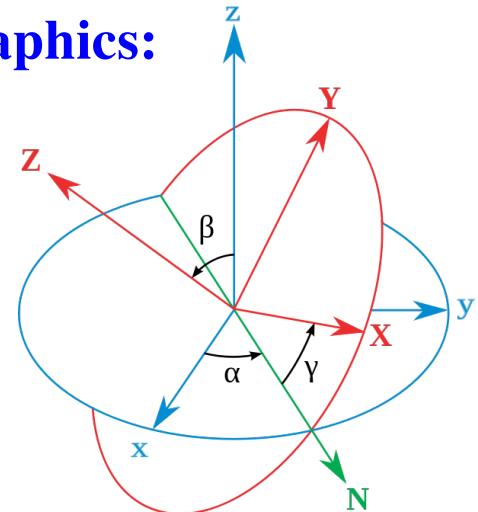
Use SIMD-efficient four-vectors abundant in mathematical physics!

- Special relativity in physics: space (x, y, z)-time (t) four-vector

$$X^\mu = (ct, x, y, z); c: \text{light speed}$$

- Quaternion representation of rotation in computer graphics:

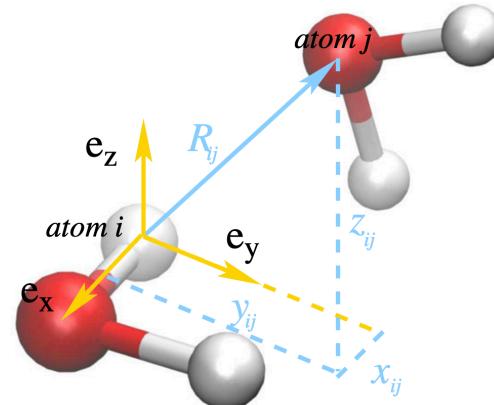
$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos \frac{\theta}{2} \cos \frac{\phi+\psi}{2} \\ \sin \frac{\theta}{2} \cos \frac{\phi-\psi}{2} \\ \sin \frac{\theta}{2} \sin \frac{\phi-\psi}{2} \\ \cos \frac{\theta}{2} \sin \frac{\phi+\psi}{2} \end{bmatrix}; (\theta, \phi, \psi): \text{Euler angles}$$



- Feature vector in deep-learning molecular dynamics:

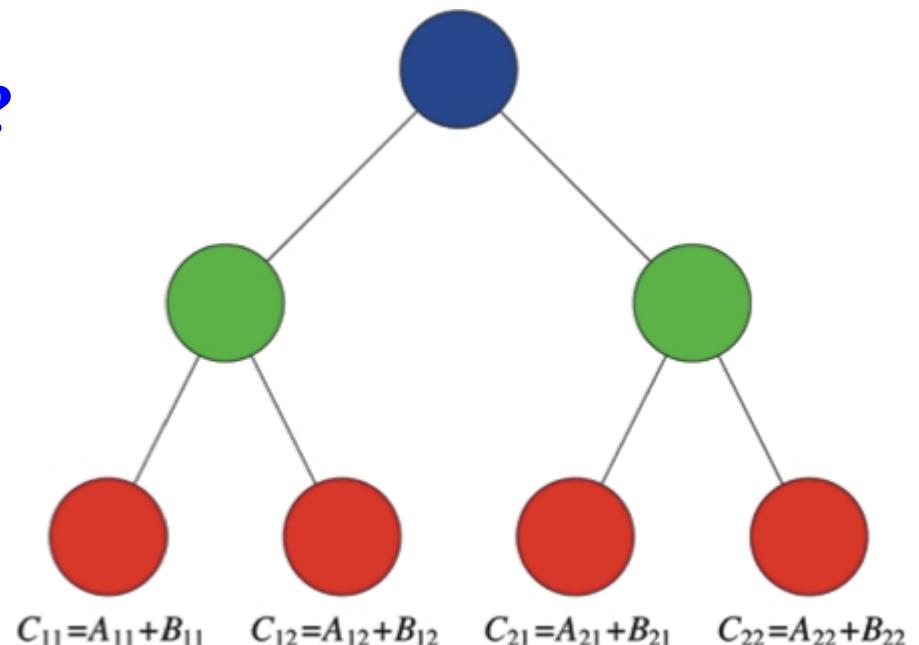
$$D_{ij} = (1/R_{ij}, x_{ij}/R_{ij}, y_{ij}/R_{ij}, z_{ij}/R_{ij})$$

L. Zhang et al., Phys. Rev. Lett. 120, 143001 ('18)



Cache-Oblivious Linked-List Cell MD?

- Recursive blocking for cells?



Cache-Oblivious Algorithms

EXTENDED ABSTRACT SUBMITTED FOR PUBLICATION. In Proc. FOCS99

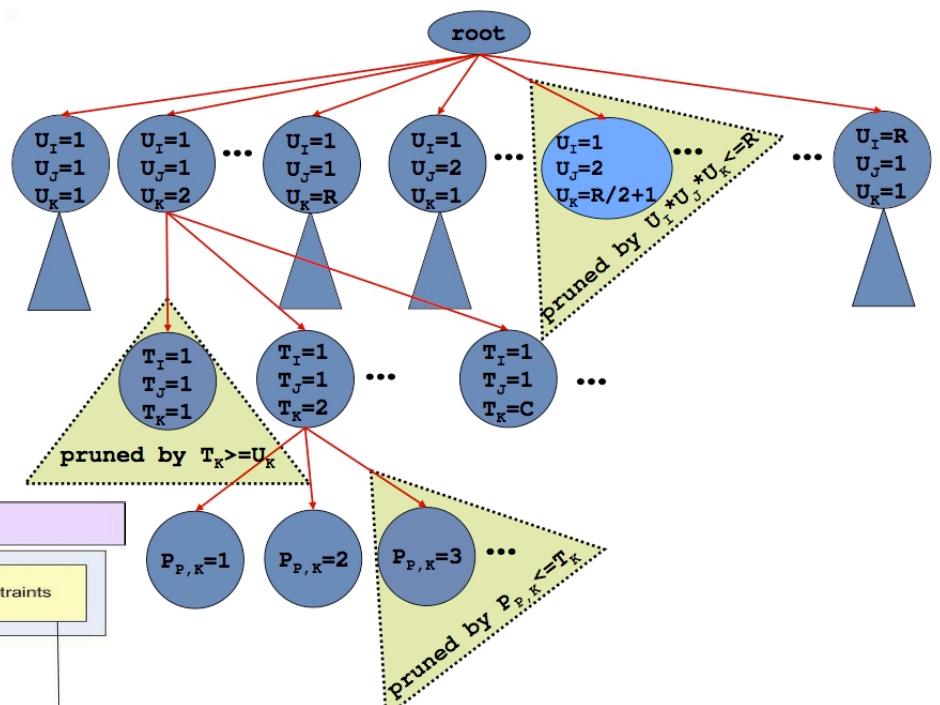
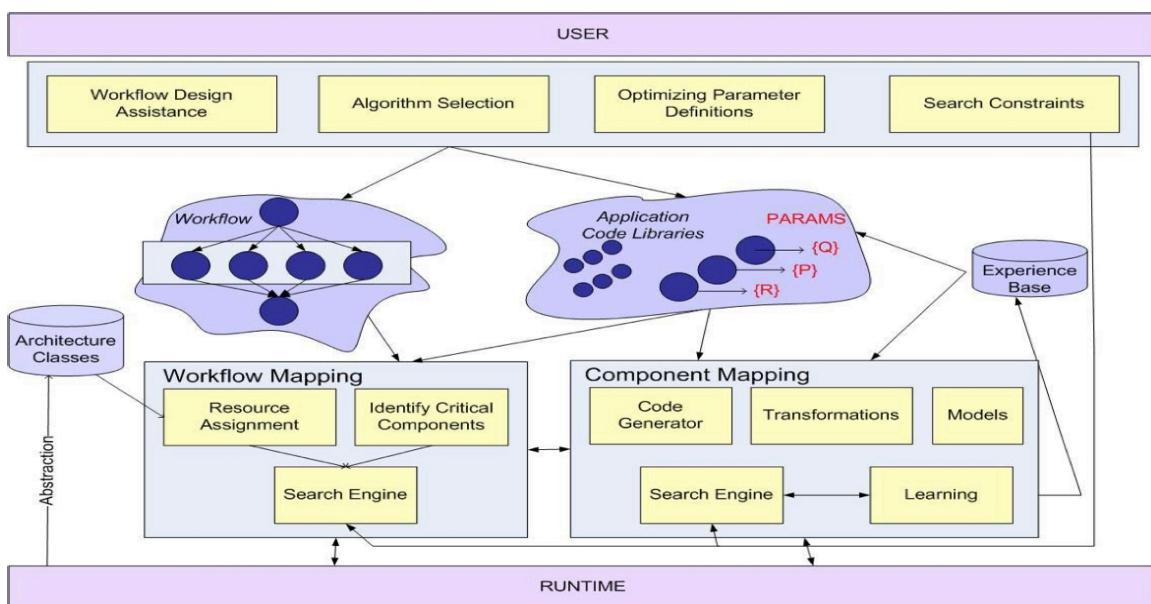
Matteo Frigo Charles E. Leiserson Harald Prokop Sridhar Ramachandran
MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139
`{athena, cel, prokop, sridhar}@supertech.lcs.mit.edu`

We introduce an “ideal-cache” model to analyze our algorithms, and we prove that an optimal cache-oblivious algorithm designed for two levels of memory is also optimal for multiple levels.

Intelligent Performance Optimization

- Knowledge representation to express concurrency/data locality & machine learning to optimally map them to hardware

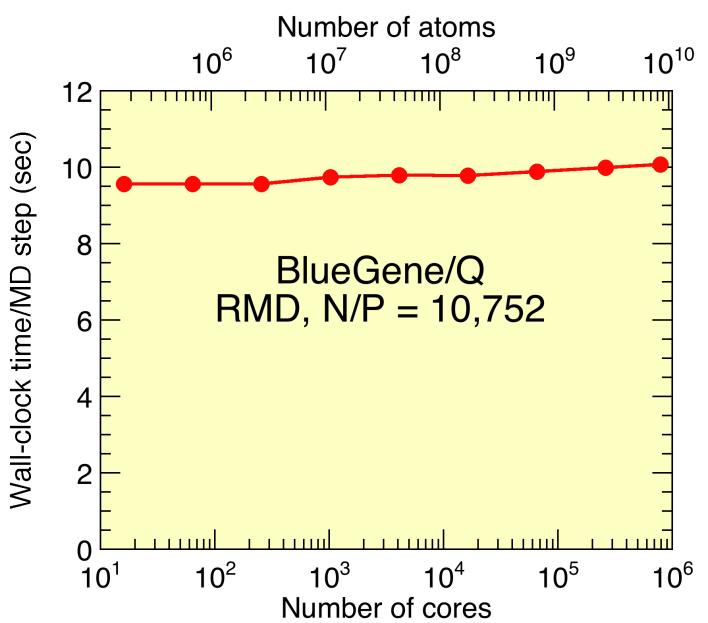
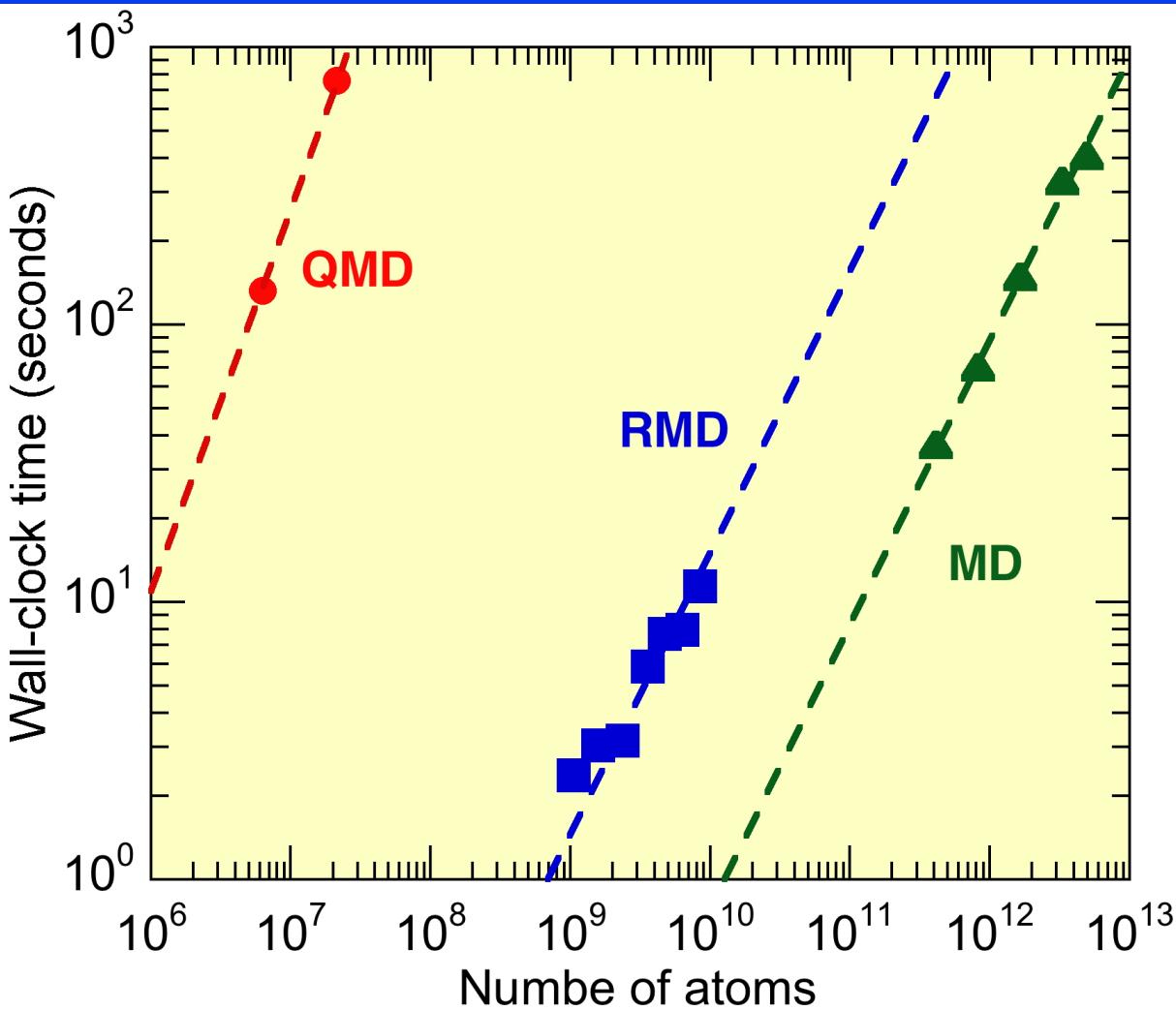
cf. Tunable hierarchical cellular decomposition exposes maximal data locality



Pruned decision tree
C. Chen, Ph.D.
Thesis (Computer
Science, USC, '07)

"Intelligent optimization of parallel & distributed applications," B. Bansal, U. Catalyurek, J. Chame, C. Chen, E. Deelman, Y. Gil, M. Hall, V. Kumar, T. Kurc, K. Lerman, A. Nakano, Y. L. Nelson, J. Saltz, A. Sharma, and P. Vashishta, in *Proc. of Next Generation Software Workshop, Int'l Parallel & Distributed Processing Symp. (IPDPS 07)*

Scalable Simulation Algorithm Suite



QMD (quantum molecular dynamics): DC-DFT

RMD (reactive molecular dynamics): F-ReaxFF

MD (molecular dynamics): MRMD

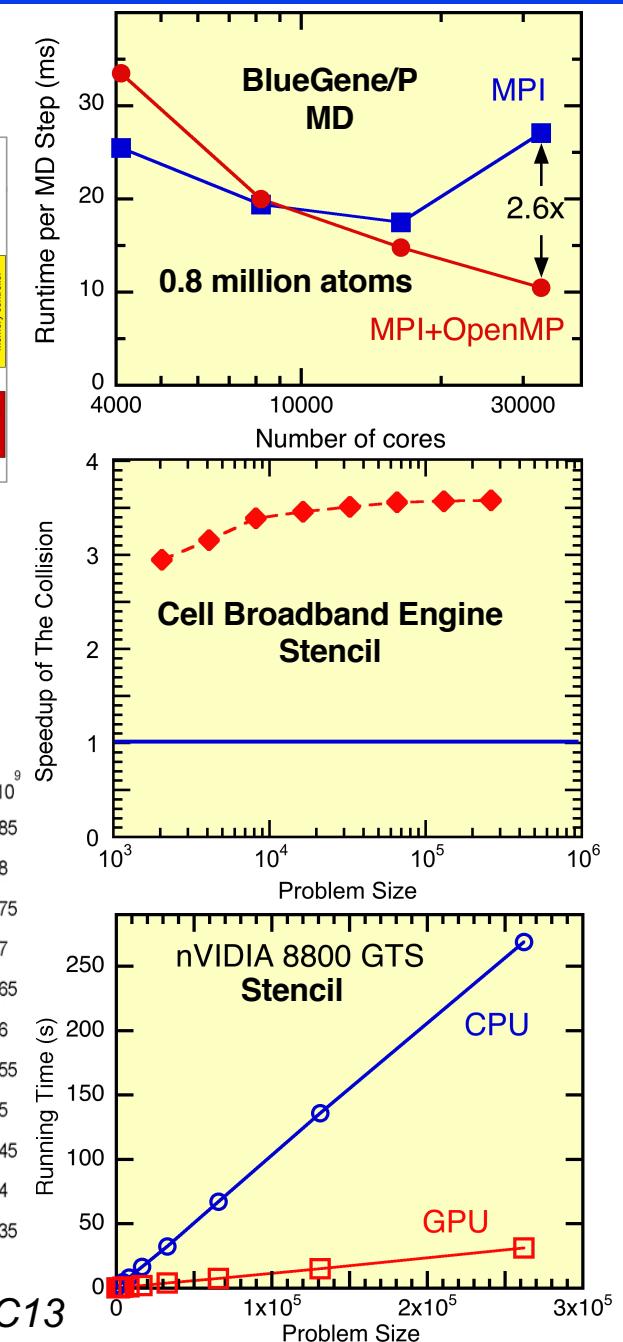
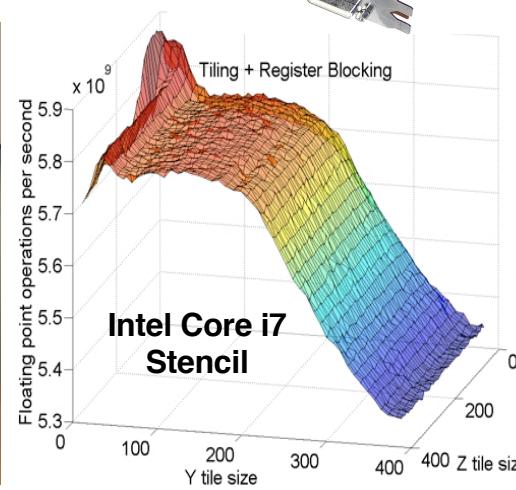
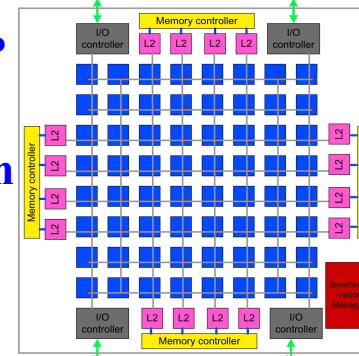
Nomura et al., IEEE/ACM SC14

- 4.9 trillion-atom space-time multiresolution MD (MRMD) of SiO_2
 - 8.5 billion-atom fast reactive force-field (F-ReaxFF) RMD of RDX
 - 39.8 trillion grid points (50.3 million-atom) DC-DFT QMD of SiC
- parallel efficiency 0.984 on 786,432 Blue Gene/Q cores

Scalability on Multicore Clusters

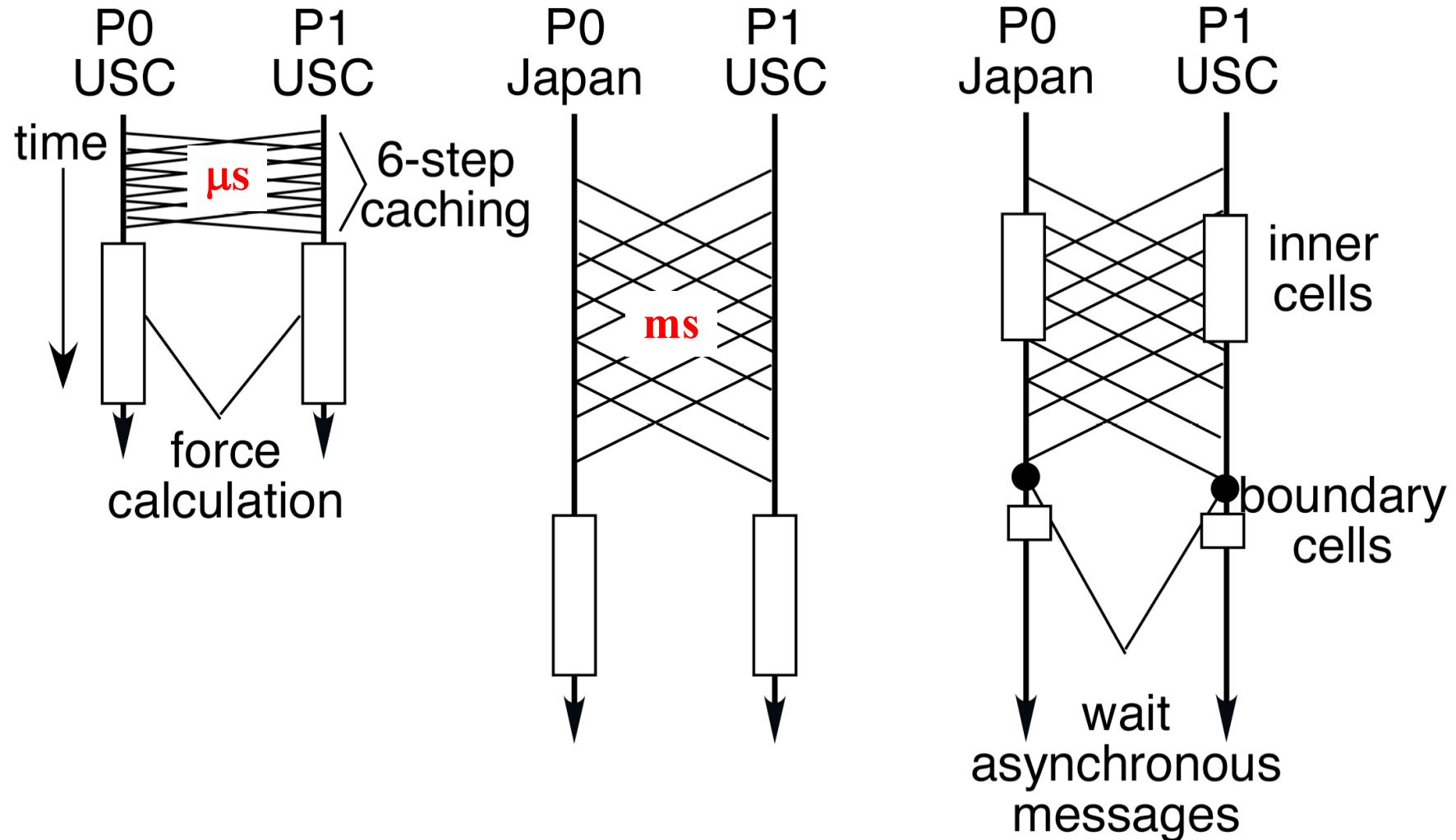
Hybrid message-passing (MPI) + multithreading (OpenMP)
+ single-instruction multiple-data (SIMD) programming

- 2.6× speedup over MPI by hybrid MPI+OpenMP on 32,768 IBM BlueGene/P cores
- Multithreading parallel efficiency 0.99 for MD on 64-core Godson-T processor
- SIMD efficiency 0.93 on PlayStation3
- 8.8× speedup on an NVIDIA GeForce 8800 GTS graphics processing unit (GPU) over an AMD Sempron CPU
- 55% of theoretical peak performance on 2.67 GHz Intel Core i7 920



Internode Optimization

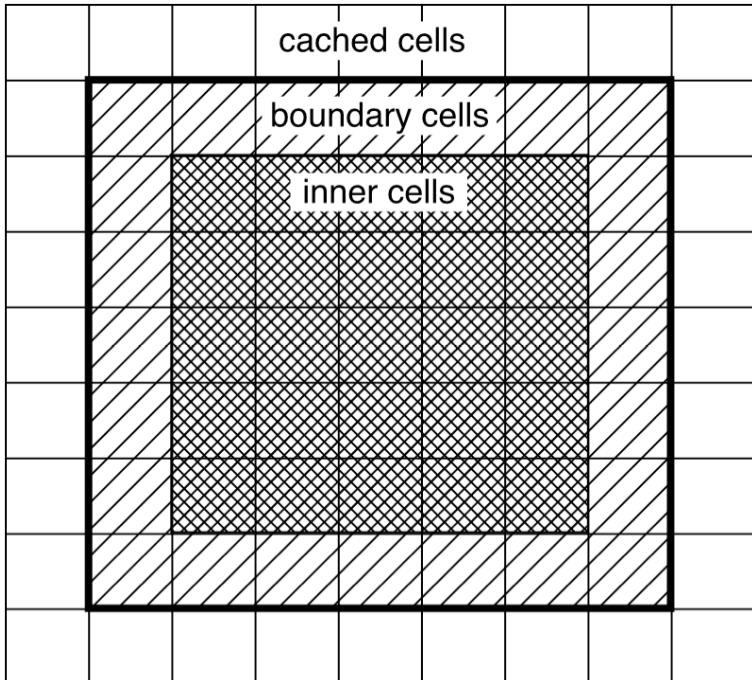
- Communication bottleneck in metacomputing on a Grid



Grid-Enabled MD Algorithm

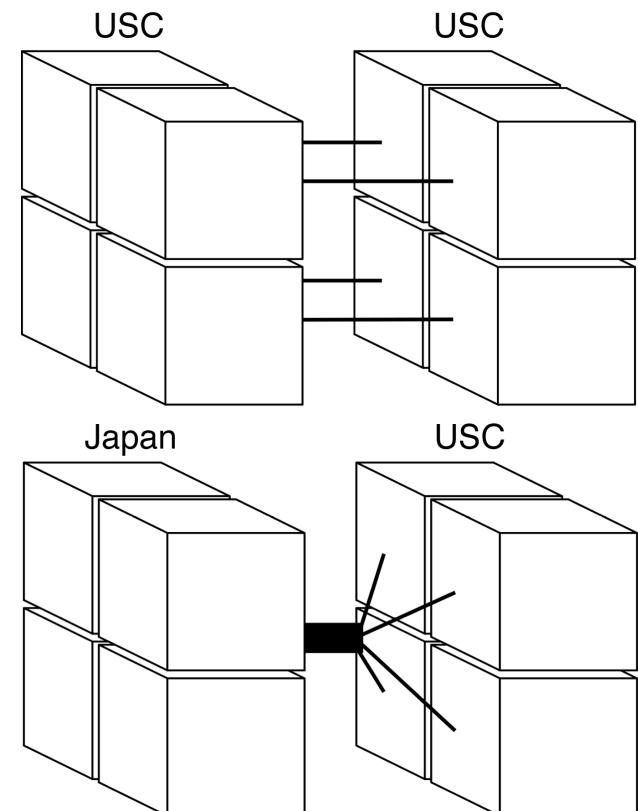
Grid MD algorithm:

1. asynchronous receive of cells to be cached `MPI_Irecv()`
2. send atomic coordinates in the boundary cells
3. compute forces for atoms in the inner cells
4. wait for the completion of the asynchronous receive `MPI_Wait()`
5. compute forces for atoms in the boundary cells



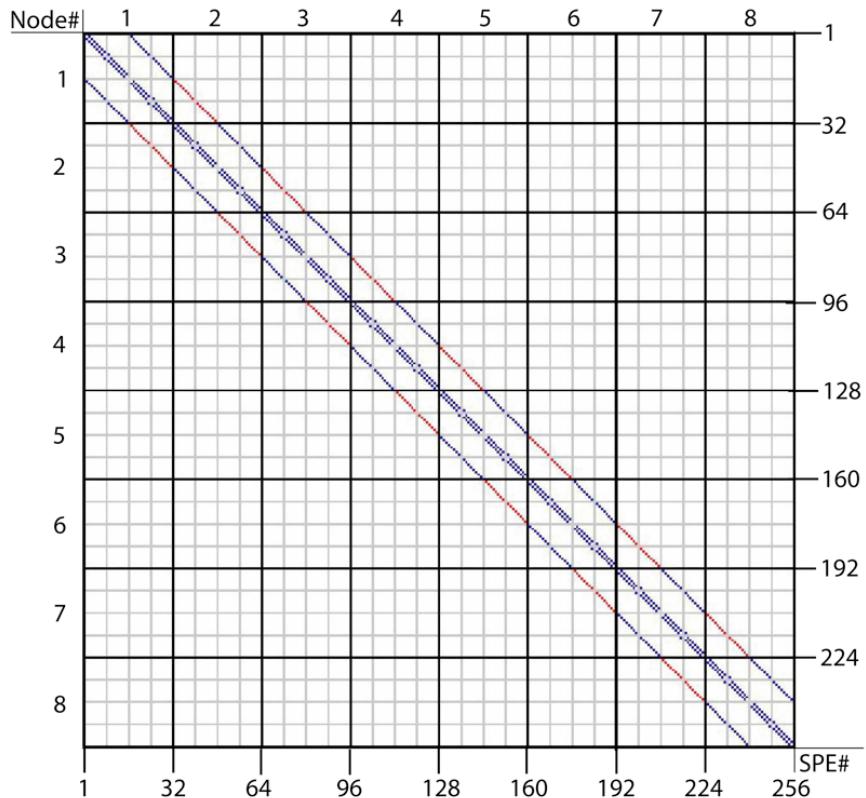
Renormalized Messages:

Latency can be reduced by composing a large cross-site message instead of sending all processor-to-processor messages

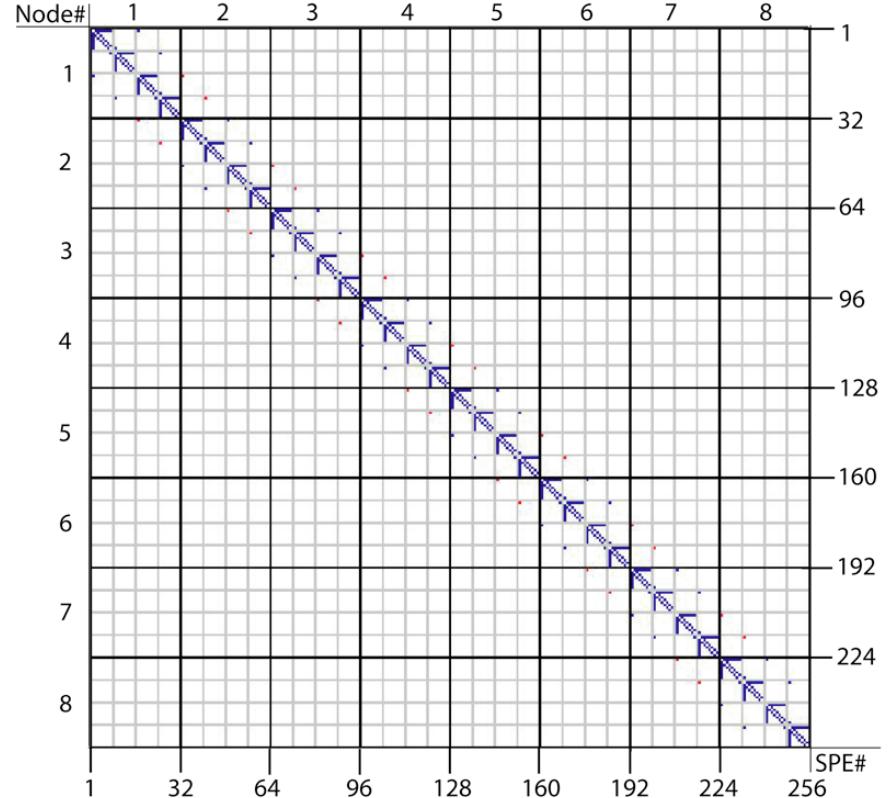


Renormalized Messages

Communication pattern of a 3D particle transport simulation code on a cluster of quad-Cell (32 cores) nodes



Original



Renormalized

H. Dursun et al., Parallel Processing Letters 19, 535 ('09)

Where to Go from Here

- **Performance profiling:** First thing is how well/badly your program is performing in terms of flop/s performance, vectorization, cache miss, etc.

- **Use professional tools like Intel VTune if available on your computer:**

<https://software.intel.com/content/www/us/en/develop/tools/vtune-profiler.html>

<https://www.youtube.com/watch?reload=9&v=ymy139CuAx8>

VTune can show you the “roofline” of your application

- Off-chip memory bandwidth (from DRAM) is critical for performance (to feed enough data to be operated)
- *Operational intensity:* Operations per byte of DRAM traffic
- *Roofline model:* Predicts the floating-point (fp) performance from operation intensity, theoretical peak fp performance & peak memory bandwidth

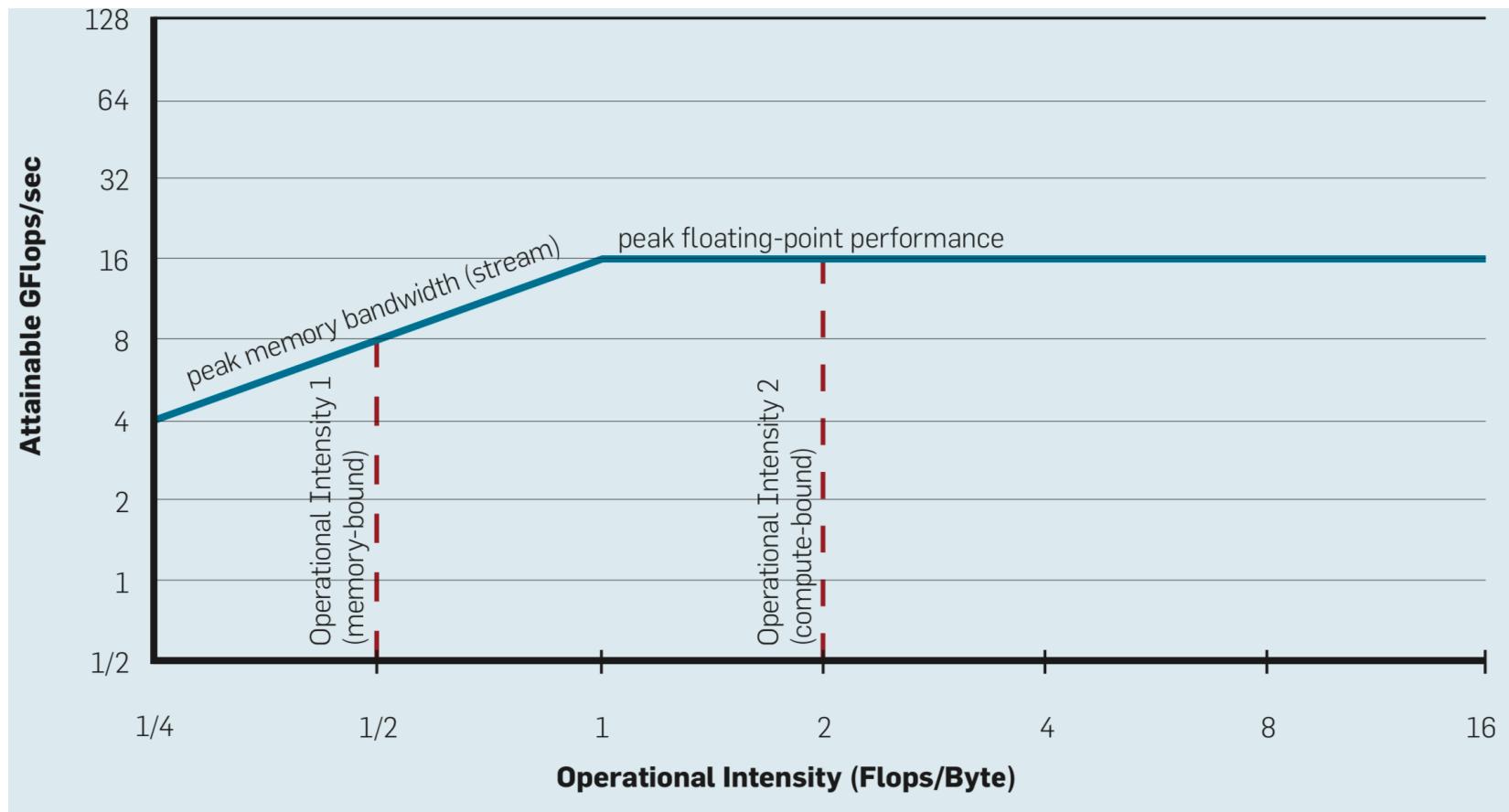
$$\text{Attainable fp performance} \left[\frac{\text{Gflop}}{\text{sec}} \right] = \min \left(\frac{\text{Peak fp performance}}{\left[\frac{\text{Gflop}}{\text{sec}} \right]}, \frac{\text{Peak memory bandwidth}}{\left[\frac{\text{GByte}}{\text{sec}} \right]} \times \text{Operational intensity} \left[\frac{\text{flop}}{\text{Byte}} \right] \right)$$

S. Williams et al., [Commun. ACM 52\(4\), 65 \('09\)](https://doi.org/10.1145/1539303.1539305)

V. Elango et al., [ACM. T. Arch. Code Opt. 11, 67 \('15\)](https://doi.org/10.1145/2783242.2783250)

Roofline Model of Performance

$$\text{Attainable fp} \left[\frac{\text{Gflop}}{\text{sec}} \right] = \min \left(\text{Peak fp} \left[\frac{\text{Gflop}}{\text{sec}} \right], \text{Memory bandwidth} \left[\frac{\text{GByte}}{\text{sec}} \right] \times \text{Operational intensity} \left[\frac{\text{flop}}{\text{Byte}} \right] \right)$$



Key: Data/computation locality

see Berkeley CS267 lecture on “memory hierarchies & matrix multiplication”