Optimizing Molecular Dynamics

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Departmentf of Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu

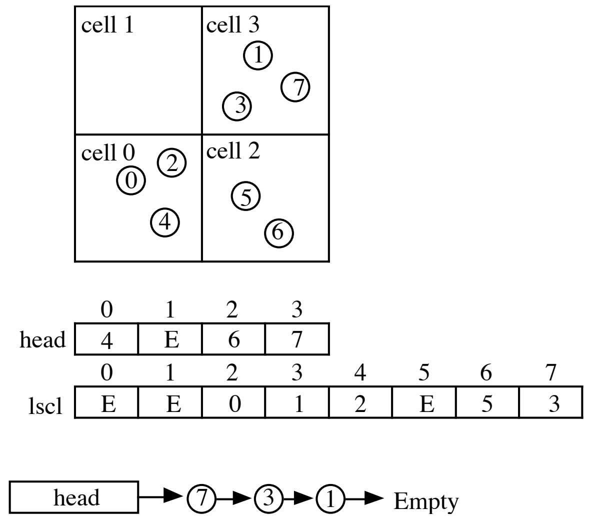
- **Intranode optimization: CPU & memory access**
- **Internode optimization: Communication**

Data/computation locality!

Intranode: Memory Access

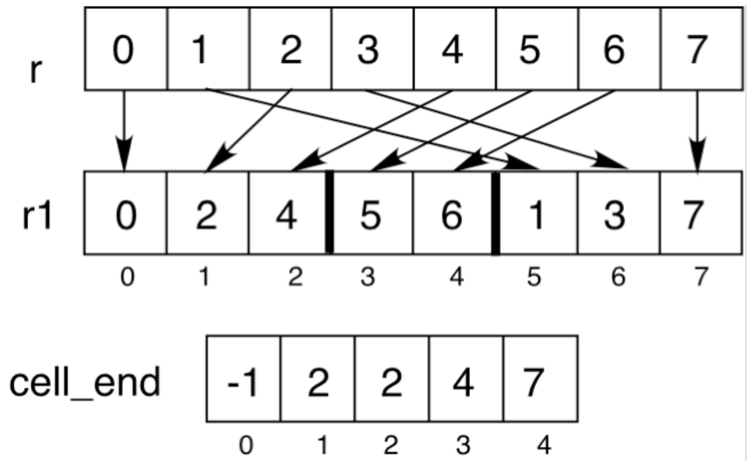
Data re-ordering

- **Linked-list cells—irregular memory access pattern**



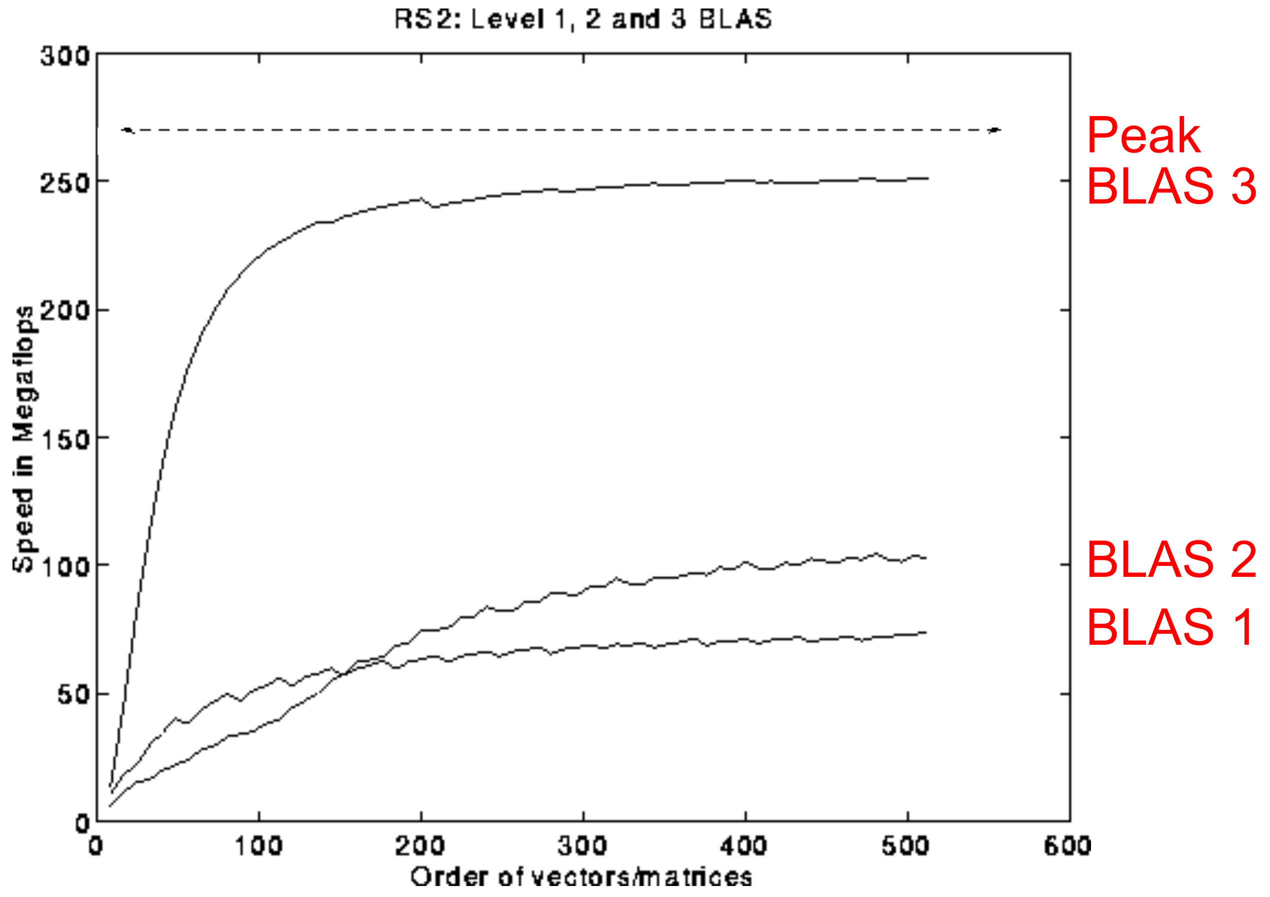
- **Data locality: Regular data layout**

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



BLAS3-Performance Molecular Dynamics?

- **BLAS3: q = flop/memory access = (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 multicore (AVX), GPU**

BLAS 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)

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

$$\boldsymbol{\Psi} = \left[|\psi_1^\alpha\rangle, \ldots, |\psi_{N_{\mathrm{band}}}^\alpha\rangle\right] \ \widetilde{\mathbf{B}}(i) = \left[|\beta_{i,1}\rangle, \ldots, |\beta_{i,N_{\mathrm{atom}}}\rangle\right] \ \left[\widetilde{\mathbf{D}}(i,j)\right]_{I,J} = D_{ij,I}\delta_{IJ}$$

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

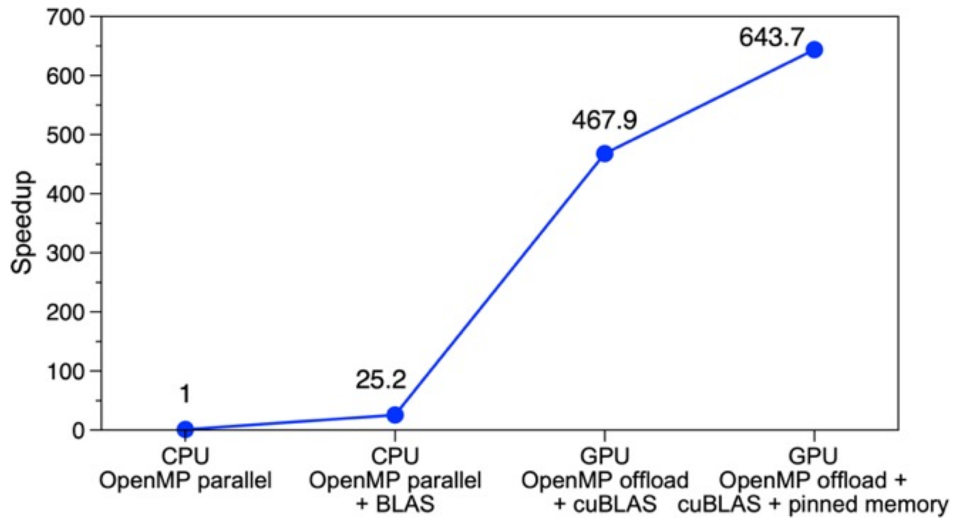
- **50.5% of the theoretical peak FLOP/s performance on 786,432 Blue Gene/Q cores (entire Mira at the Argonne Leadership Computing Facility)**
- **55% of the theoretical peak FLOP/s on Intel Xeon E5-2665**

K. Nomura *et al.*, *IEEE/ACM Supercomputing, SC14* ('14)

More BLAS-ification

- **BLASified nonlocal electron dynamics on graphics processing unit (GPU): Operation of nonlocal potential is projected onto *a vector space spanned by Kohn-Sham orbitals* at time 0 within the real-time scissor approximation** [Wang *et al*., *J. Phys. Condens. Mat*. **31**, 214002 ('19)]**, making it dense matrix operations implemented with highly optimized level3 (or matrix-matrix) BLAS (basic linear algebra subprogram) library on GPU**

$$\hat{v}_{\mathrm{nl}}|\psi_n(t)\rangle \cong \Delta_{\mathrm{sci}} \sum_{m \geq \mathrm{LUMO}} |\psi_m\rangle\langle\psi_m|\psi_n(t)\rangle$$



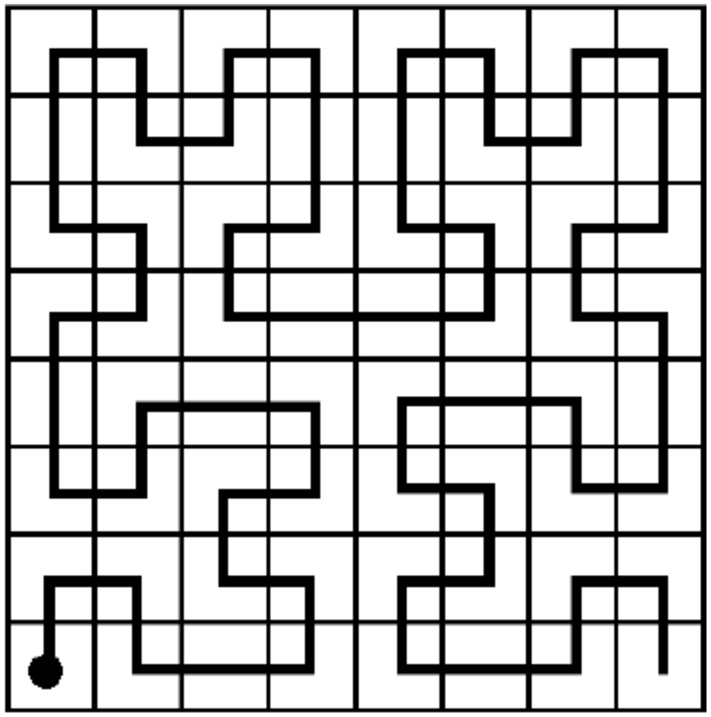
- **See lecture on** divide-&conquer Maxwell-Ehrenfest-surface hopping simulation

Razakh *et al*., *PDSEC* (IEEE, '24); Piroozan *et al*., *PMBS* (IEEE, '24)

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**



J. Mellor-Crummey *et al*., *Int'l J. Parallel Prog*. **29**, 217 ('01)

Hilbert-Peano Curve

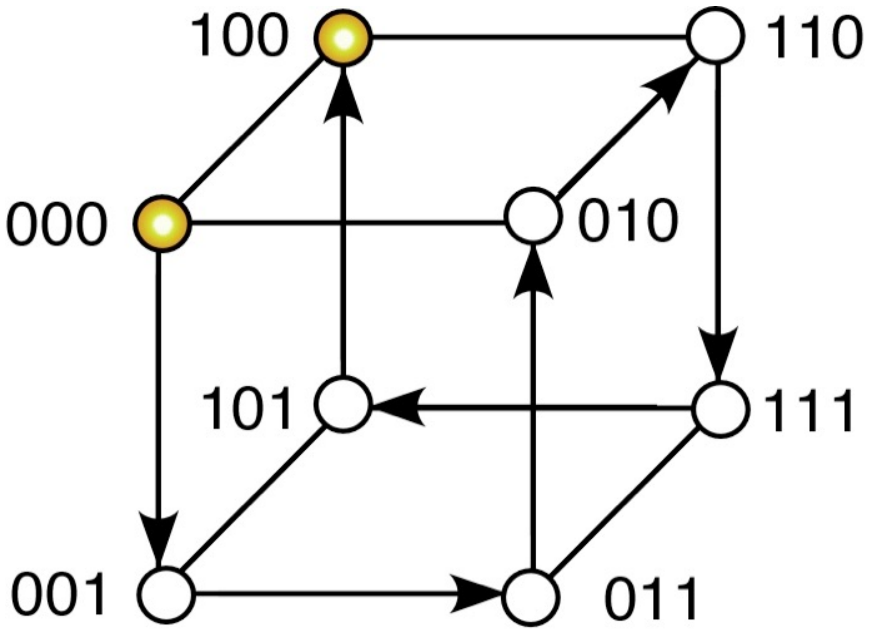
- **Gray code: a sequence of numbers such that successive numbers have Hamming distance 1**

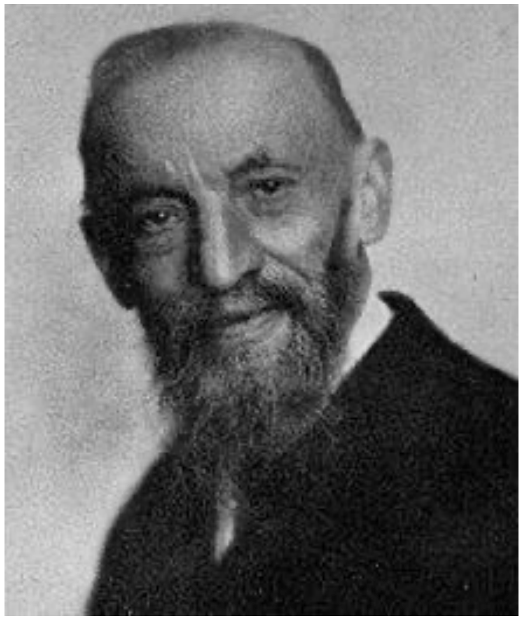
# of bits where two binary numbers differ

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

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

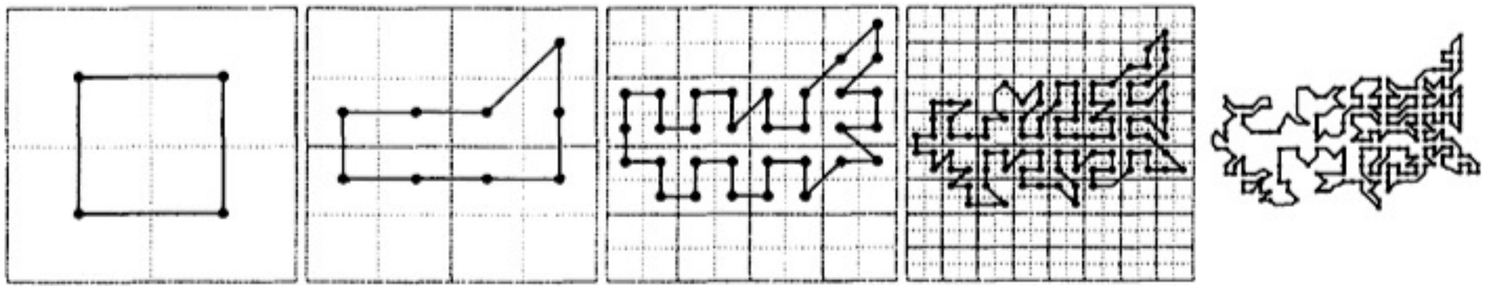
David Hilbert (1862–1943)




Giuseppe Peano (1858–1932)

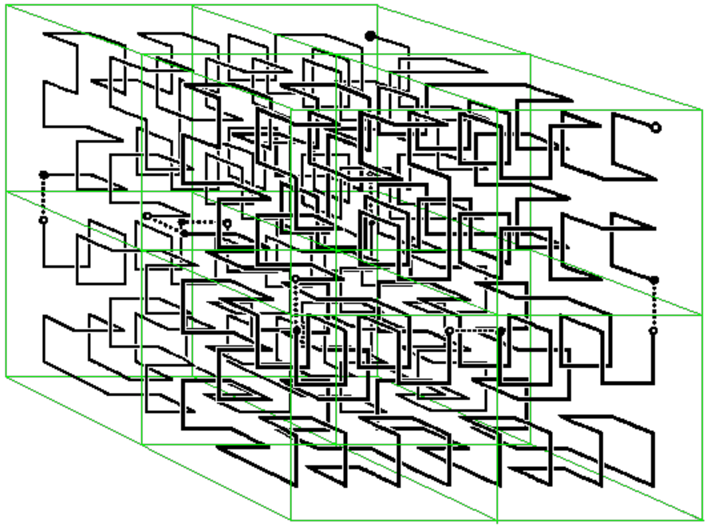
Hilbert-Peano Curve

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

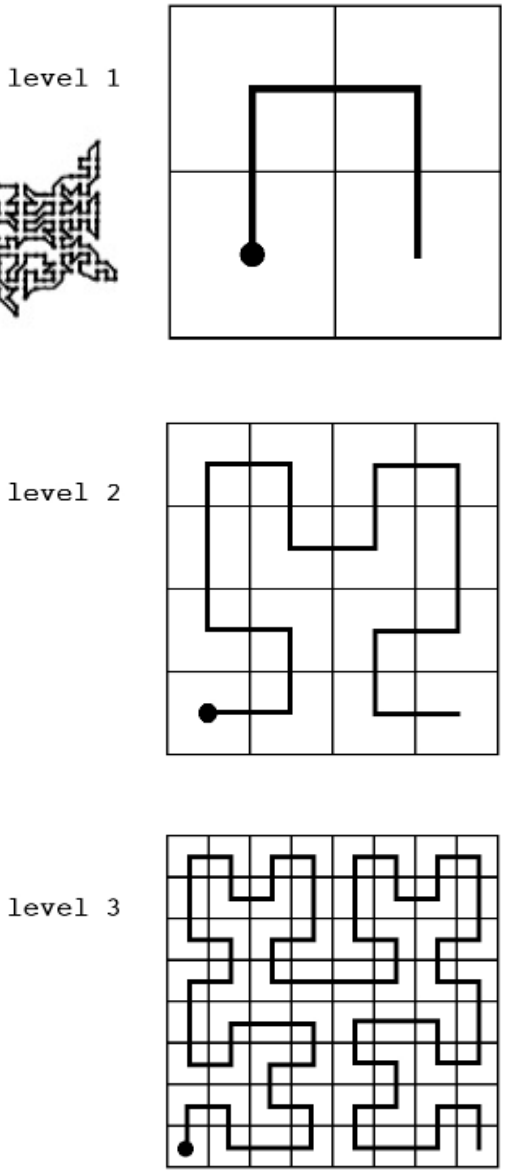


NP-complete traveling salesman problem

- **3-dimensional Hilbert curve**



See Corcoran *et al.*, *SC17*

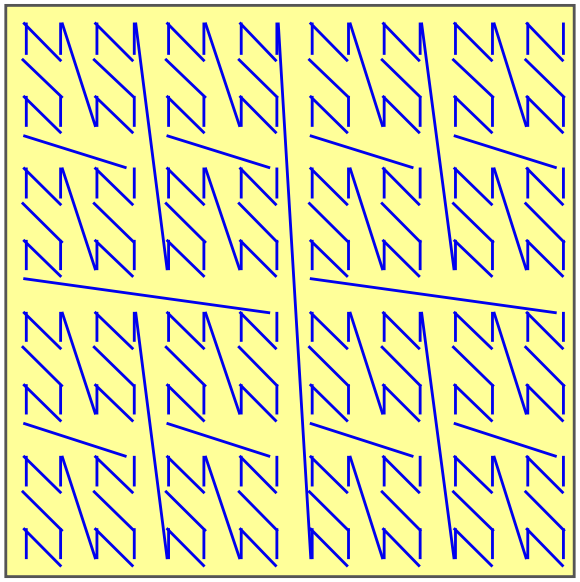


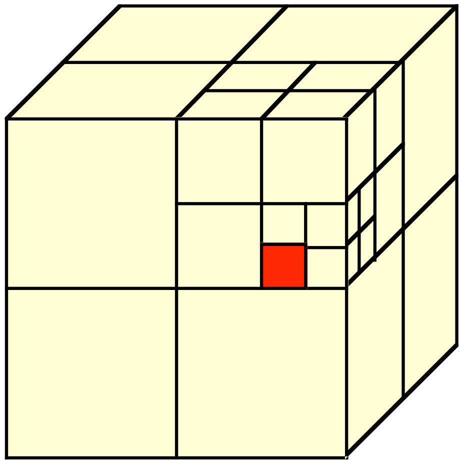
Morton (Z) Curve

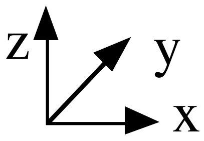
- **Spacefilling curve based on octree index**

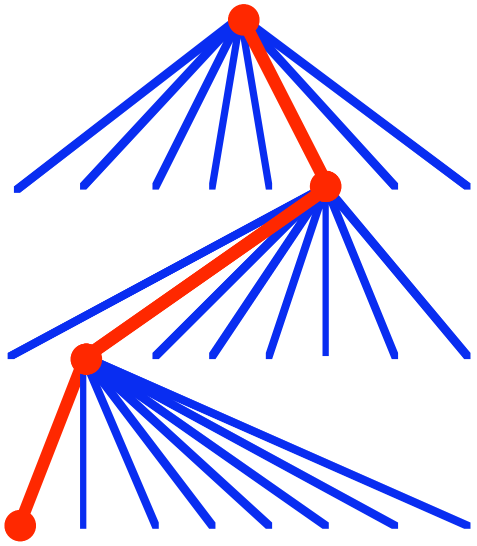
```
x =    1    1    0
y =   0    0    0
z =  1    0    0
R = 101  001  000
```

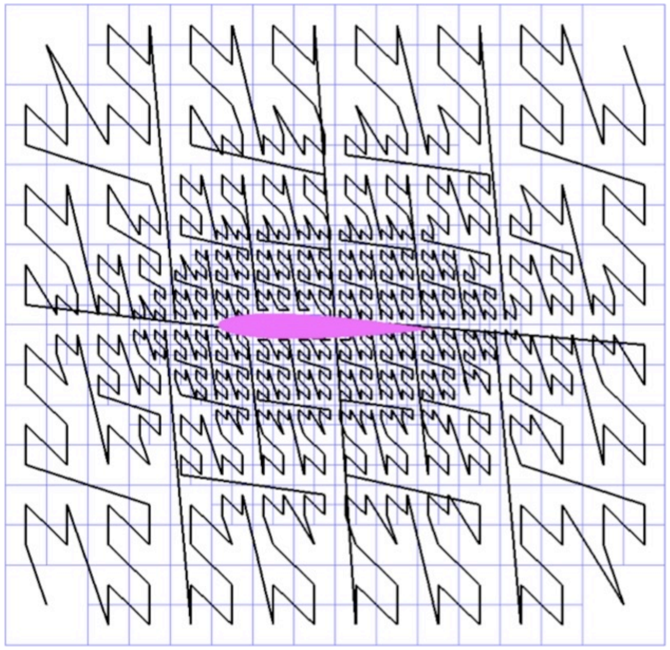
- **3D → list map preserves spatial proximity**
- **Multiresolution analysis made easy**







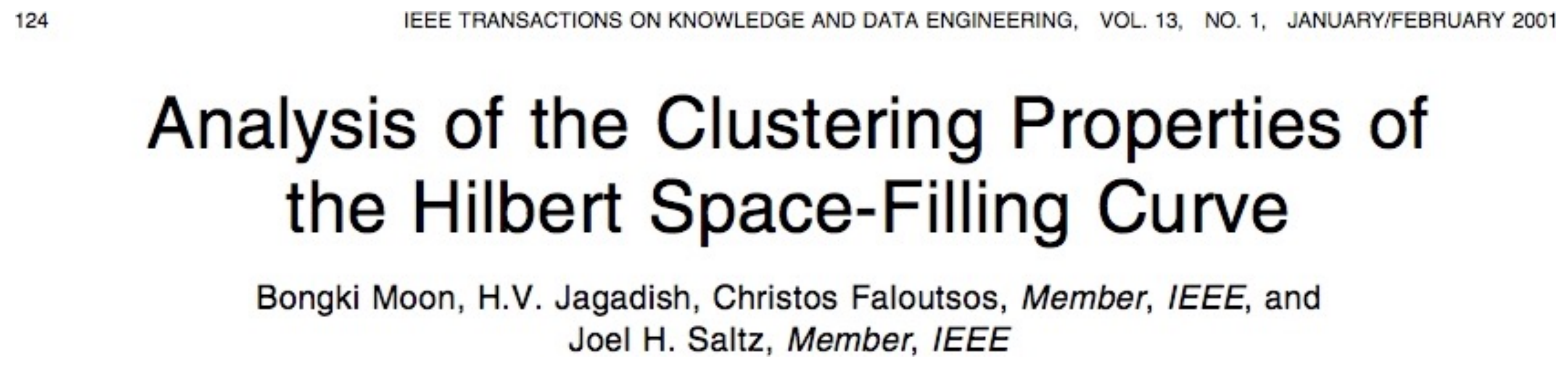




A. Omeltchenko *et al*., *Comput. Phys. Commun*. **131**, 78 ('00)

Multiresolution analysis

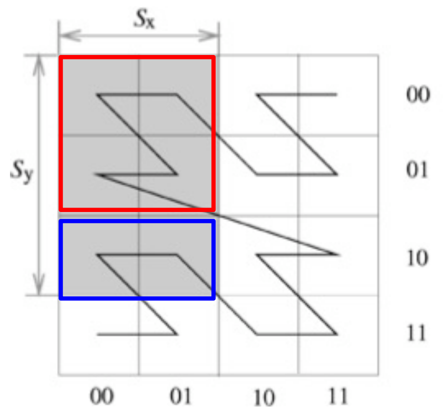
Analysis of Data Locaility


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

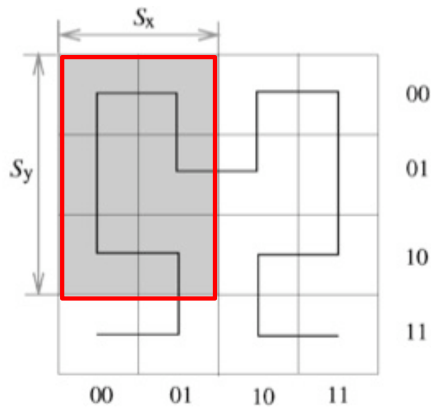
Analysis of the Clustering Properties of the Hilbert Space-Filling Curve

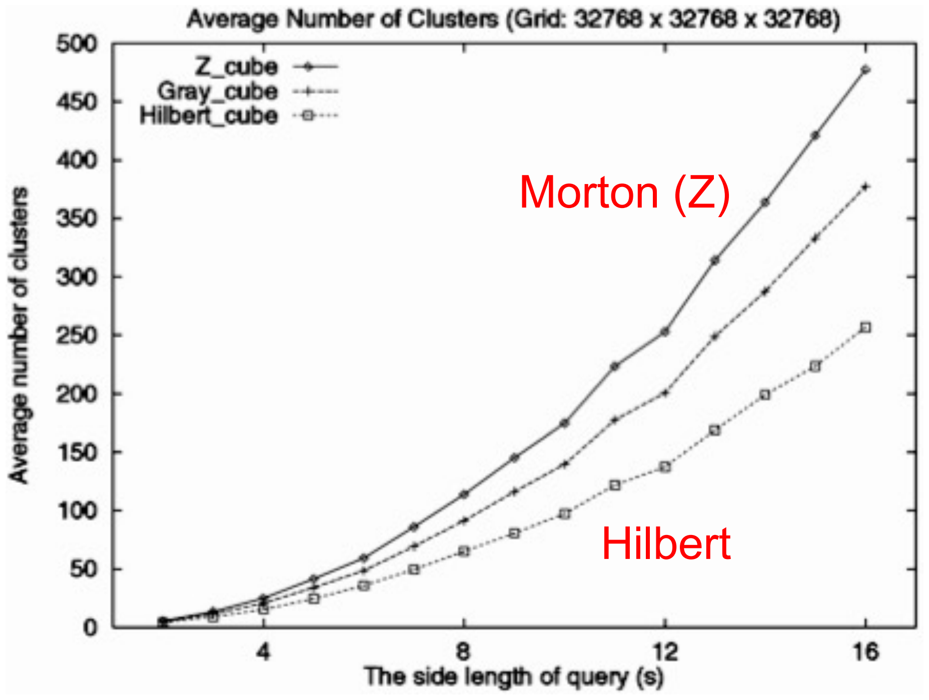
Bongki Moon, H.V. Jagadish, Christos Faloutsos, *Member, IEEE*, and Joel H. Saltz, *Member, IEEE*

Morton



Hilbert





2 clusters

1 cluster

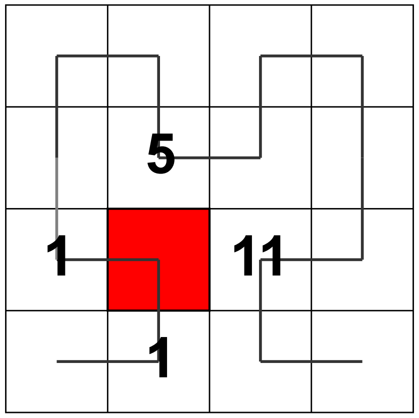
Cluster ~ cache line ~ latency cost

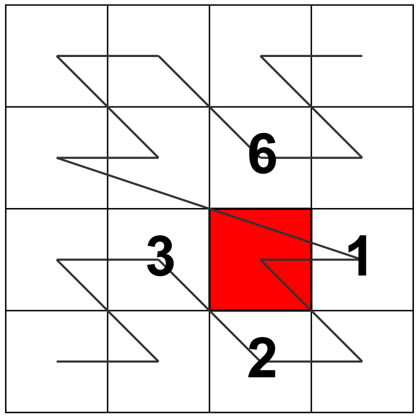
Hilbert curve is better than Morton curve for spatial range query

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 $2 < k_c < 13$

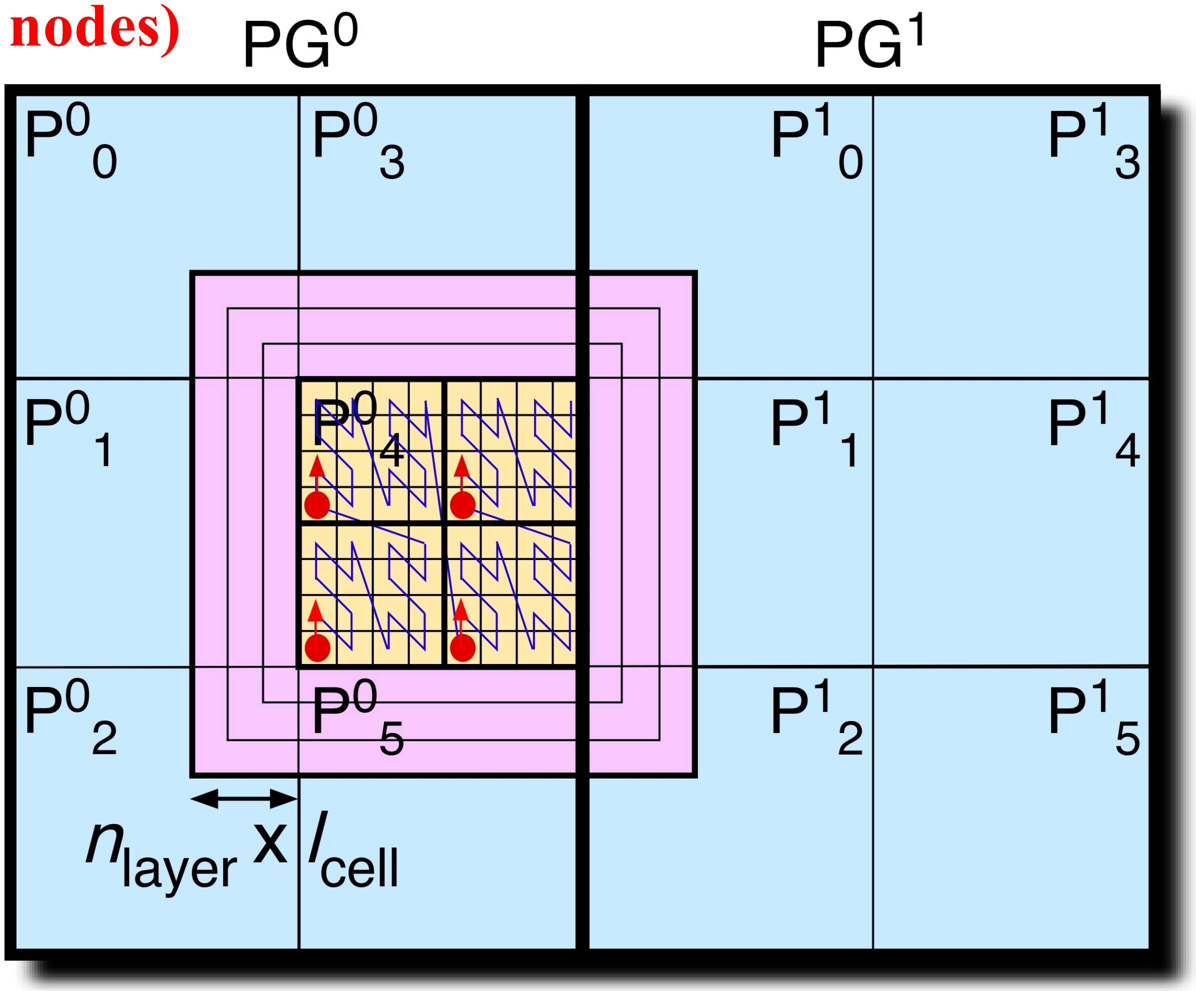
Scott Calaghan (CSCI 596 final project)

Tunable Hierarchical Cellular Decomposition

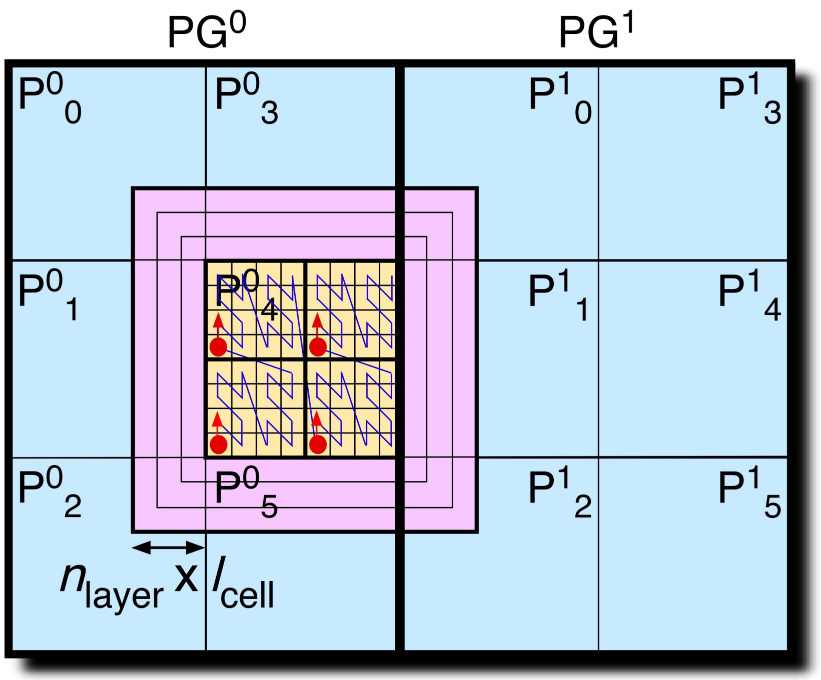
Mapping $O(N)$ divide-&-conquer algorithms onto memory hierarchies

- **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 = 2–6)**
- **Tunable cell data & computation structures: Data/computation re-ordering & granularity parameterized at each decomposition level**
- **Tunable hybrid MPI + OpenMP + SIMD implementation**

Nomura *et al.*, *IPDPS 2009*

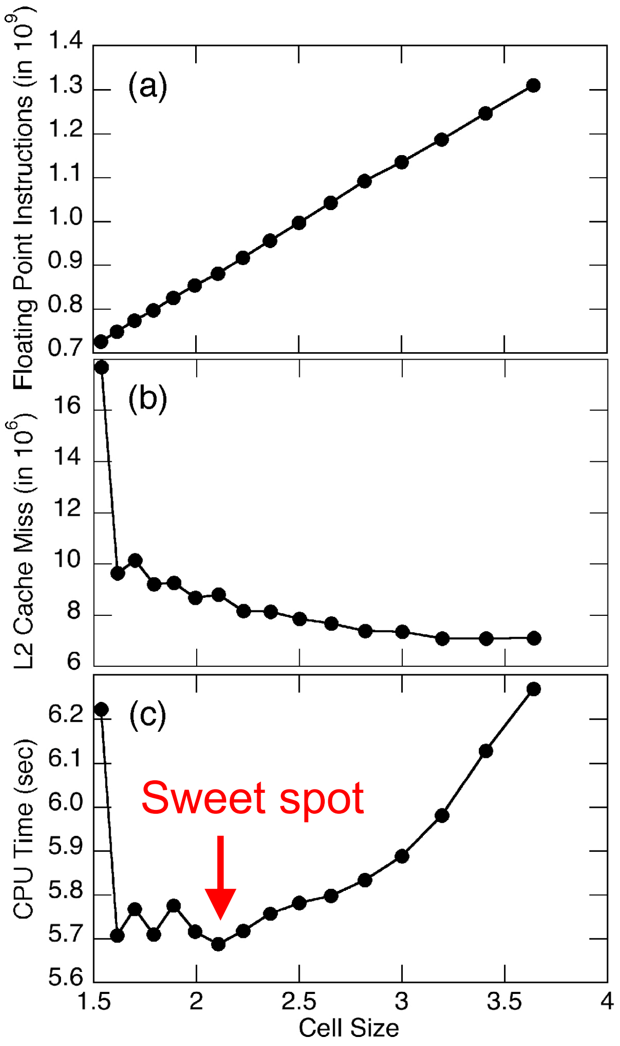


Performance Tunability



Floating-point operation/L2 cache miss trade-off:

331,776-atom silica MRMD on 1.4 GHz Pentium III



MPI/OpenMP parallelism trade-off:

8,232,000-atom silica MRMD & 290,304-atom RDX F-ReaxFF on 8-way 1.5 GHz Power4

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

SIMD Vectorization

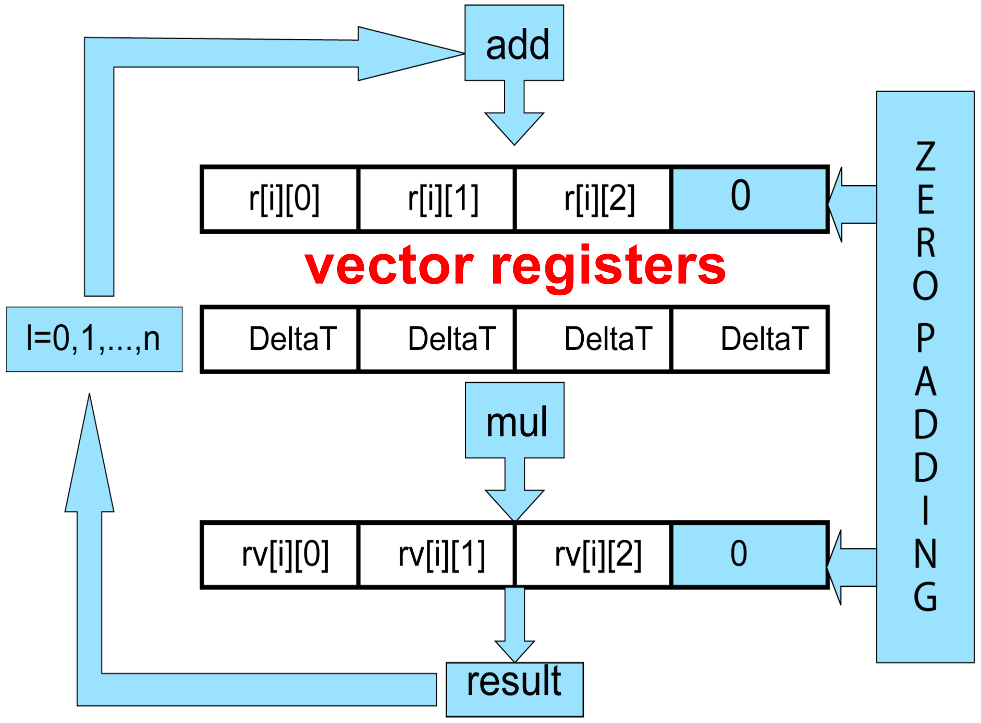
- **Single-instruction multiple-data (SIMD) parallelism**

(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

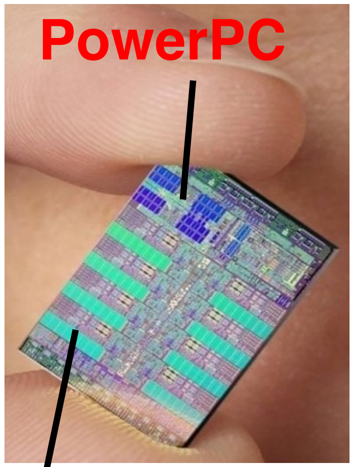


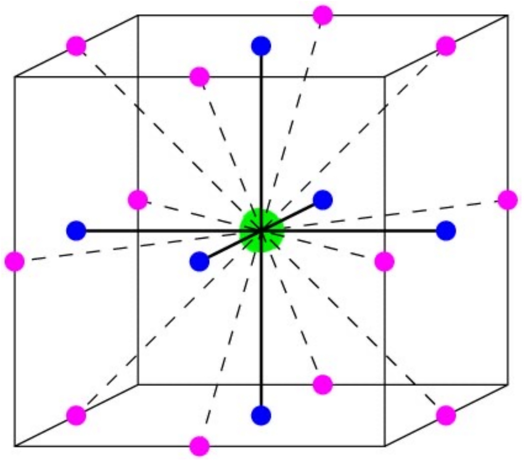
***cf.* False-sharing avoidance**

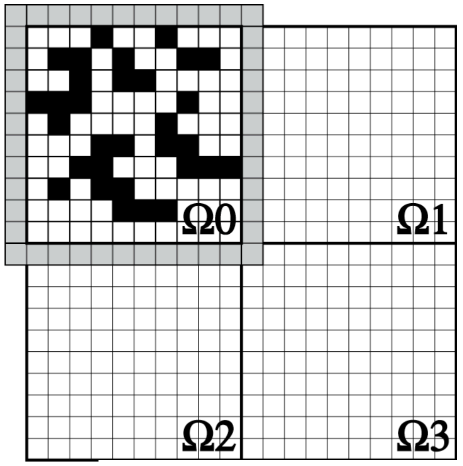
Peng *et al.*, *PDPTA 2009*; *UCHPC 2010; J. Supercomputing* **57**, 20 ('11)

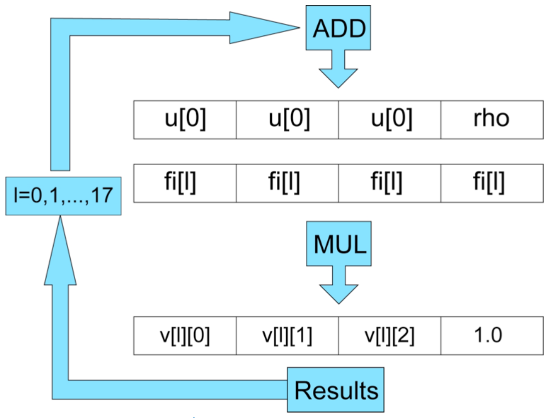
Hierarchical Parallelization

- **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**

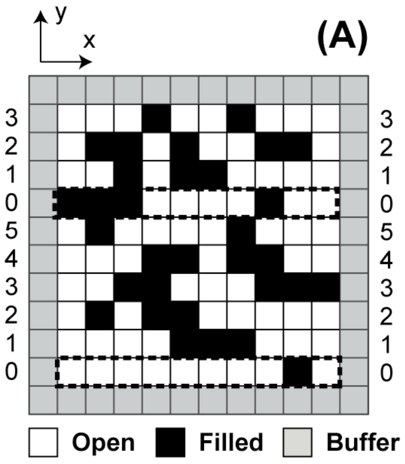


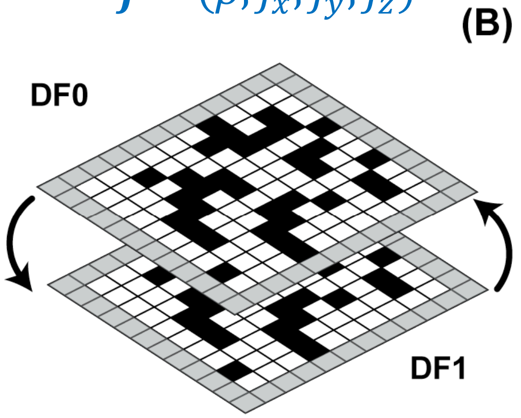






$\vec{j} = (\rho, j_x, j_y, j_z)$





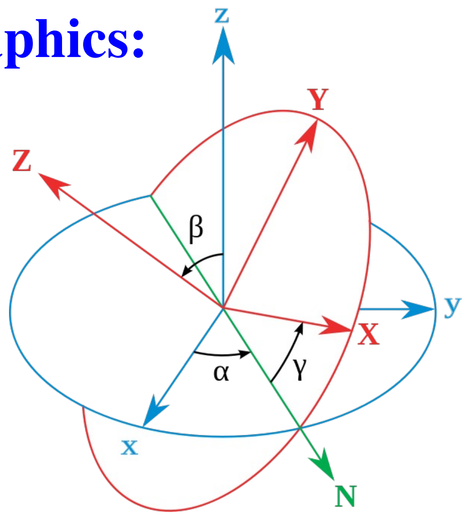
Peng *et al.*, *IJCS 08*; *Euro-Par 08*; *IPDPS 09*; *cf.* https://en.wikipedia.org/wiki/Four-vector

More Four-Vectors for SIMD

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

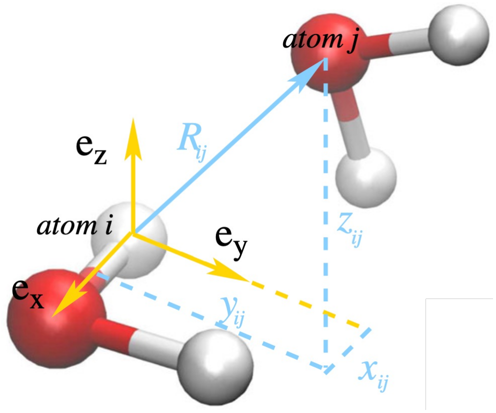
- **Special relativity in physics: space (*x*, *y*, *z*)-time (*t*) four-vector**
  $X^\mu = (ct, x, y, z)$; *c*: light speed
- **Quaternion representation of rotation in computer graphics:**

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



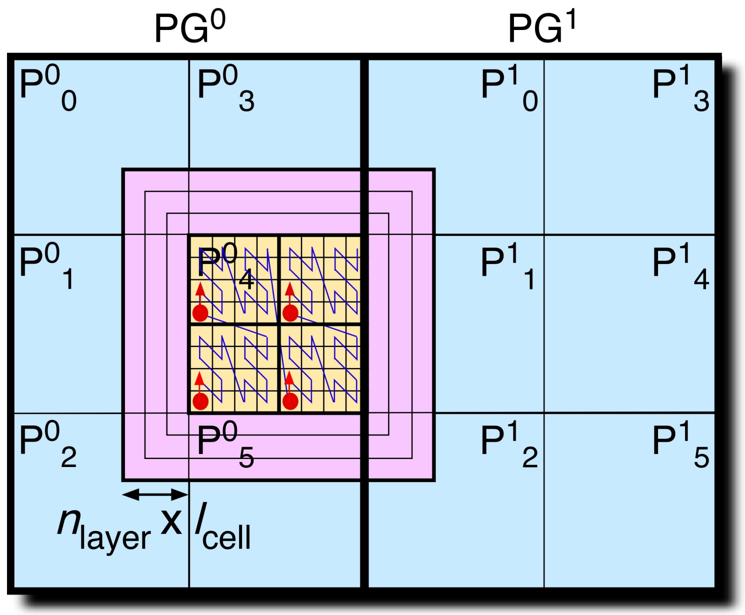
- **Feature vector in deep-learning molecular dynamics:**
  $D_{ij} = \left(1/R_{ij}, x_{ij}/R_{ij}, y_{ij}/R_{ij}, z_{ij}/R_{ij}\right)$

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



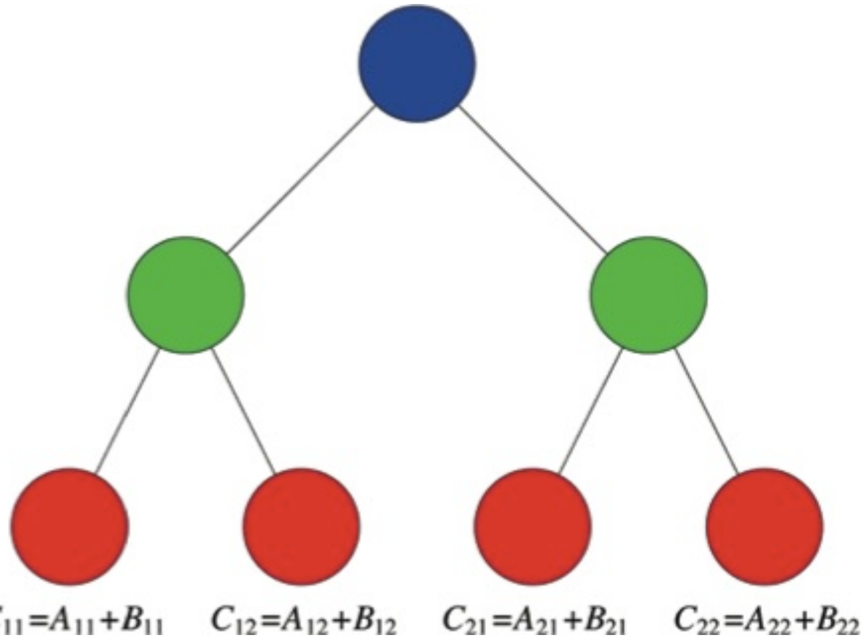
Recap: Intranode Performance Optimization

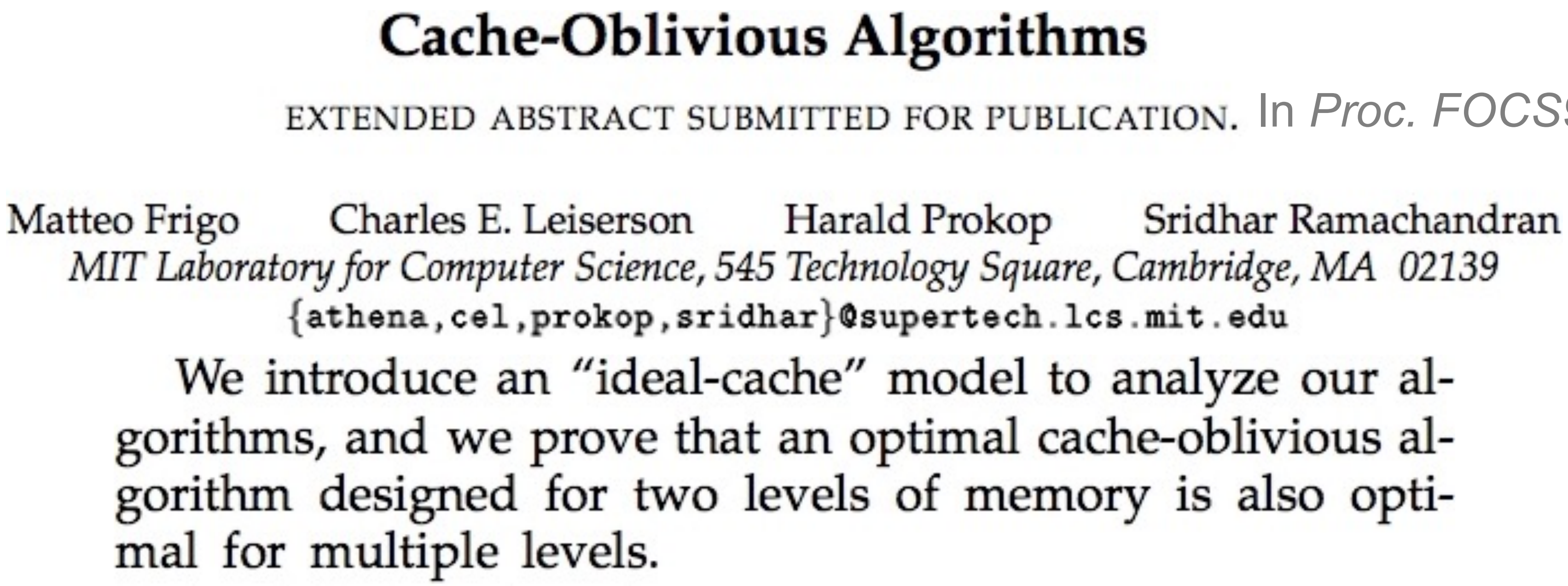
- **Key hardware feature:** **Memory hierarchy; latency *vs.* bandwidth (*cf.* Internet speed test)**
- **Key strategy:** **Feed the fast processor by (1) enhancing data/computation locality to achieve high cache-hit rate (*e.g.*, space-filling curve) & (2) increasing operational intensity, q = fast computation/slow memory I/O (*e.g.*, BLAS-ify)**
- **Do it yourself:** **Hierarchical decomposition *via* divide-&-conquer implemented with hybrid MPI + OpenMP + SIMD (*e.g.*, CUDA)**



Cache-Oblivious Linked-List Cell MD?

- **Recursive blocking for cells?**





Cache-Oblivious Algorithms

EXTENDED ABSTRACT SUBMITTED FOR PUBLICATION. In *Proc. FOCS99*

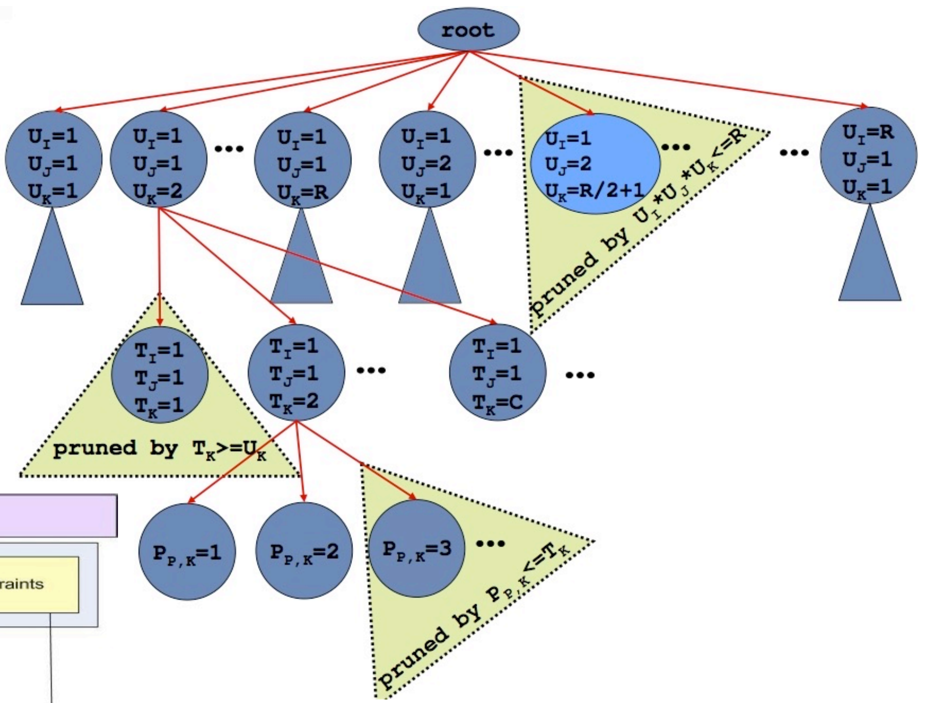
Matteo Frigo Charles E. Leiserson Harald Prokop Sridhar Ramachandran
MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139
`{athena,cel,prokop,sridhar}@supertech.lcs.mit.edu`

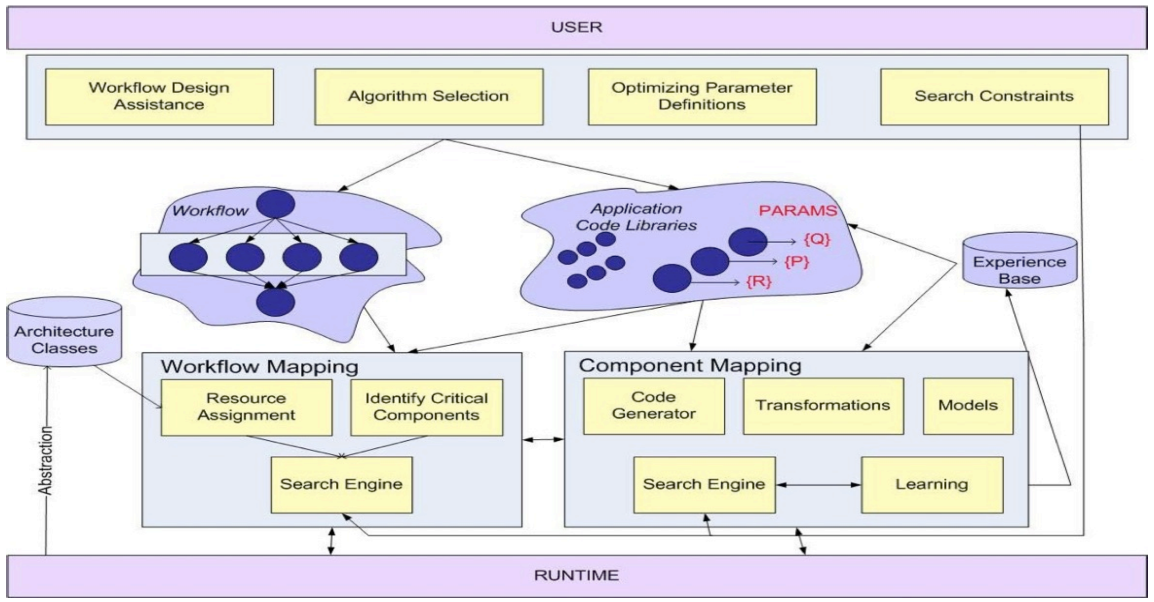
We introduce an "ideal-cache" model to analyze our algorithms, and we prove that an optimal cache-oblivious algorithm designed for two levels of memory is also optimal for multiple levels.

Intelligent Performance Optimization

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

***e.g.*, Tunable hierarchical cellular decomposition exposes maximal data locality**



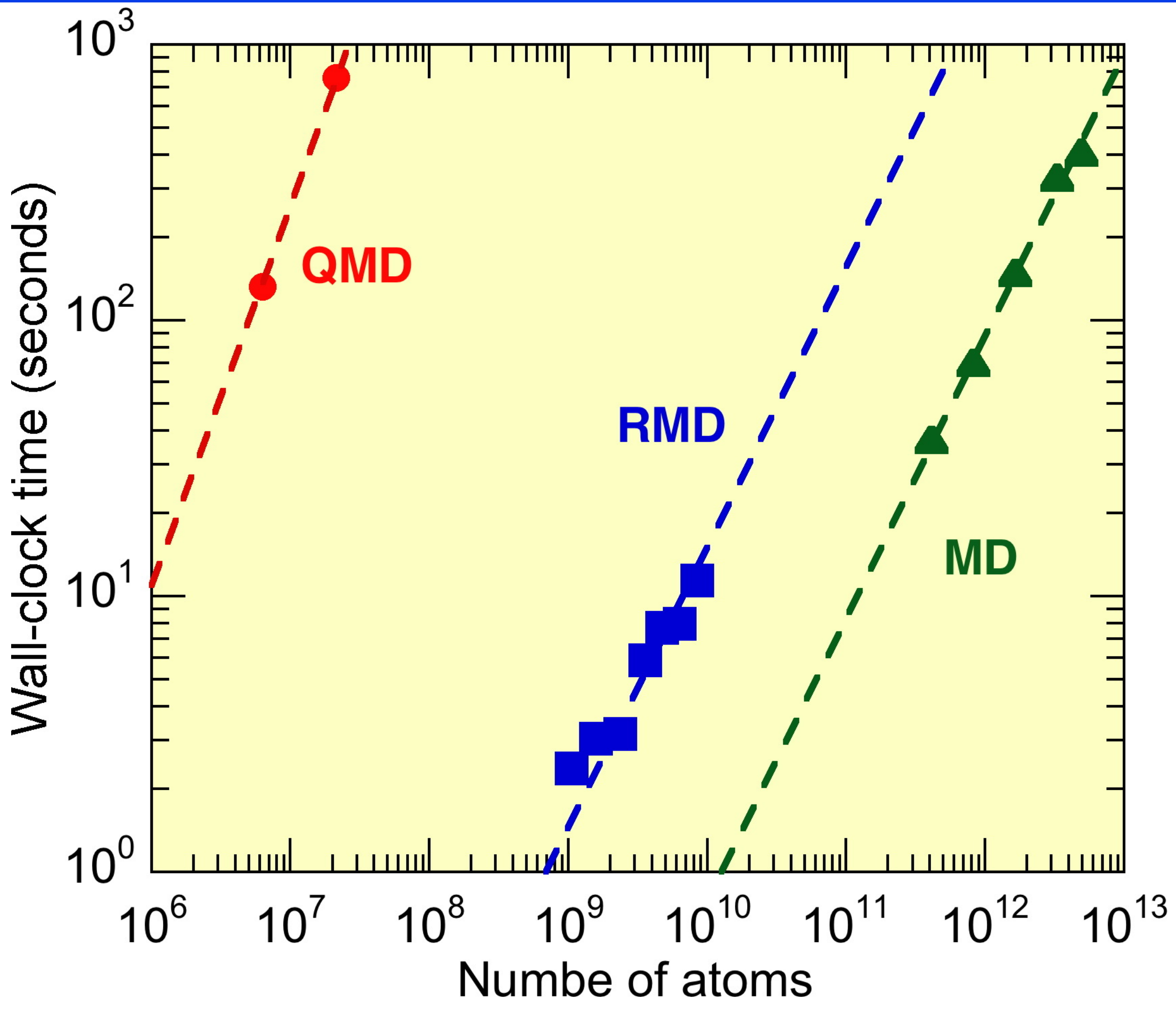


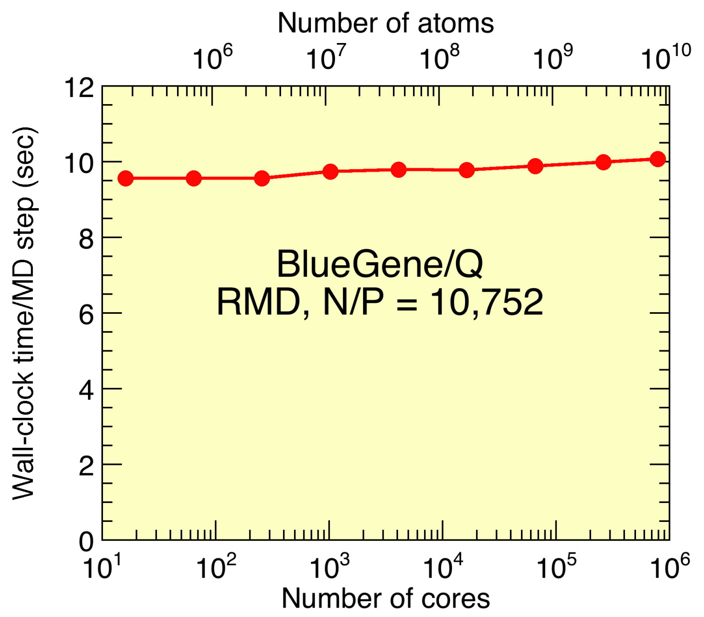
Pruned decision tree
C. Chen, *Ph.D. Thesis* (Computer Science, USC, '07)

Use AI!

"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*

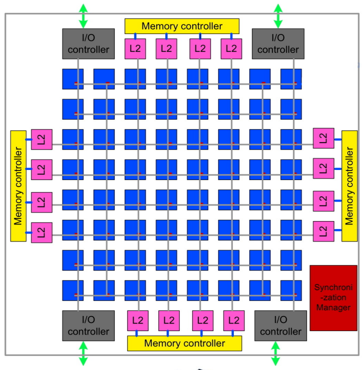
- **4.9 trillion-atom space-time multiresolution MD (MRMD) of SiO_2**
- **8.5 billion-atom fast reactive force-field (F-ReaxFF) RMD of RDX**
- **39.8 trillion grid points (50.3 million-atom) DC-DFT QMD of SiC**

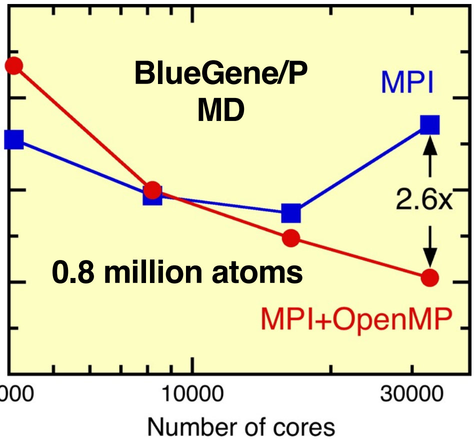
parallel efficiency 0.984 on 786,432 Blue Gene/Q cores

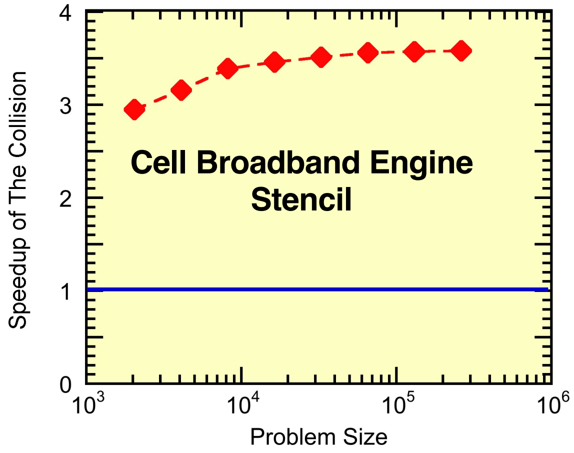
Scalability on Multicore Clusters

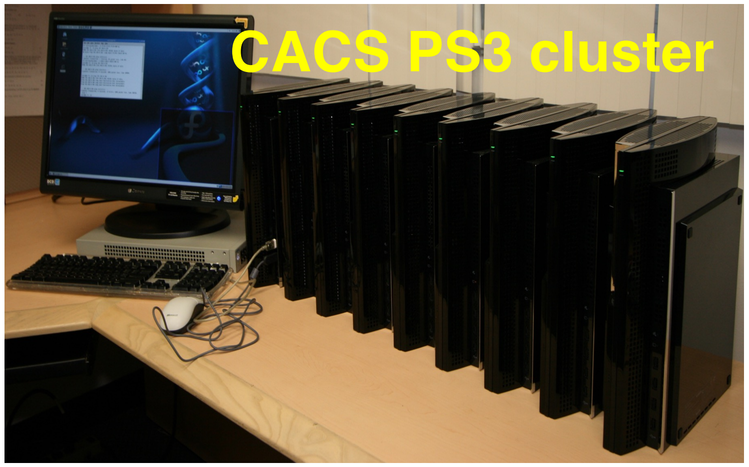
Hybrid message-passing (MPI) + multithreading (OpenMP) + single-instruction multiple-data (SIMD) programming

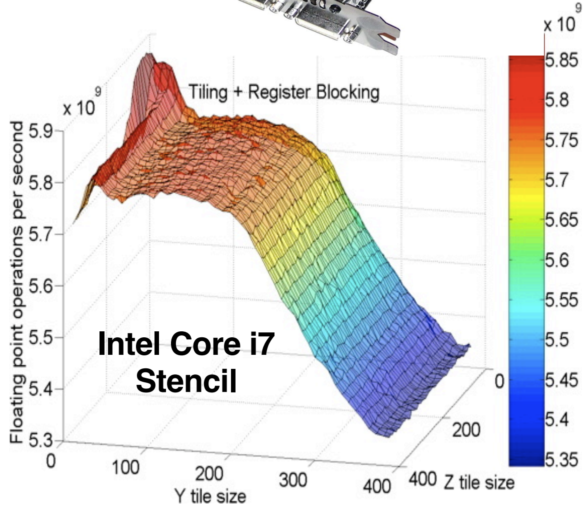
- 2.6× speedup over MPI by hybrid MPI+OpenMP on 32,768 IBM BlueGene/P cores
- Multithreading parallel efficiency 0.99 for MD on 64-core Godson-T processor
- SIMD efficiency 0.93 on PlayStation3
- 8.8× speedup on an NVIDIA GeForce 8800 GTS graphics processing unit (GPU) over an AMD Sempron CPU
- 55% of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

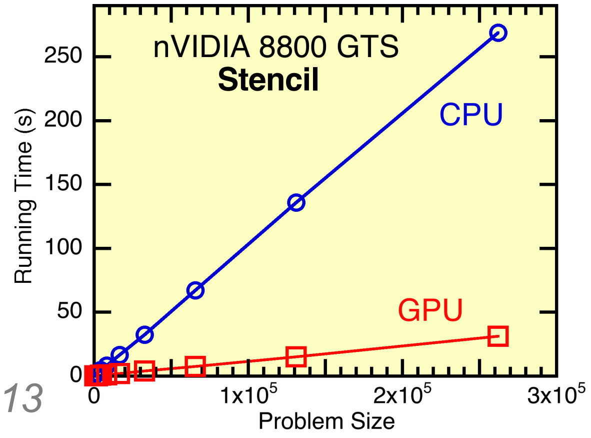








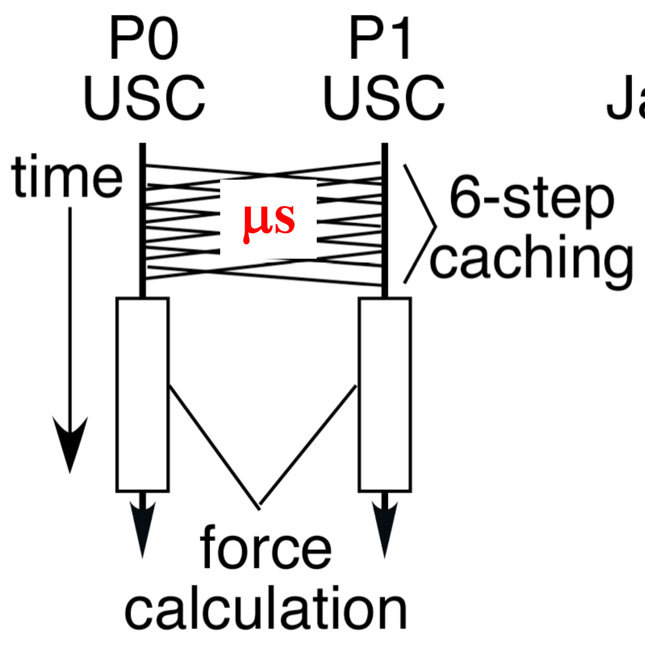


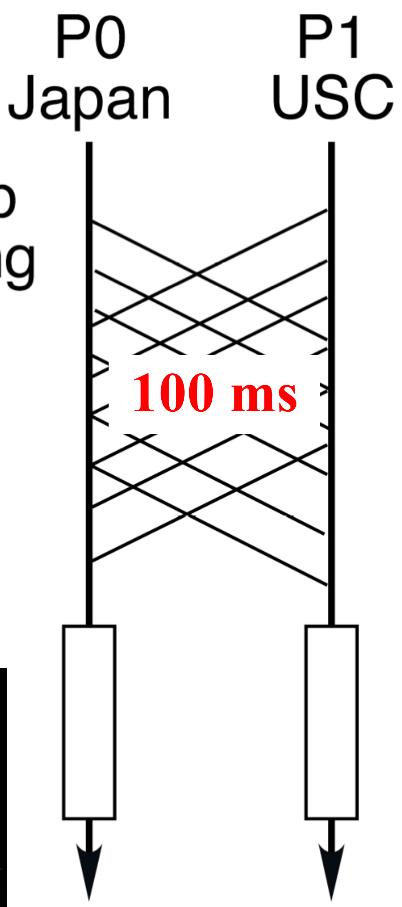


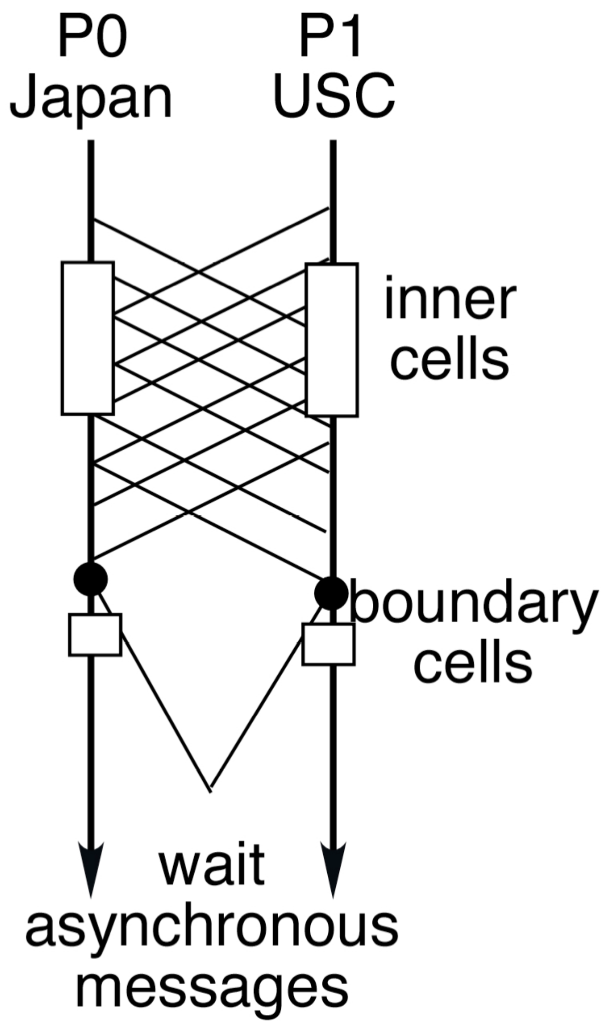
IJCS08; IPDPS09; PPL09; ICS10; CF11; Euro-Par12; JSC12; JPDC13; SC13

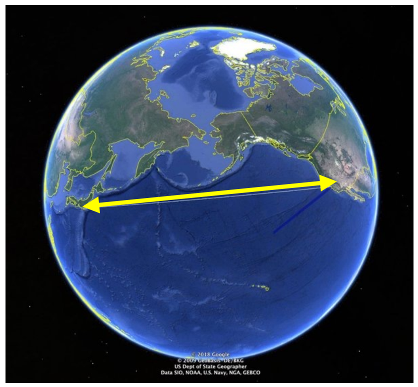
Internode Optimization

- **Communication bottleneck in metacomputing on a Grid**





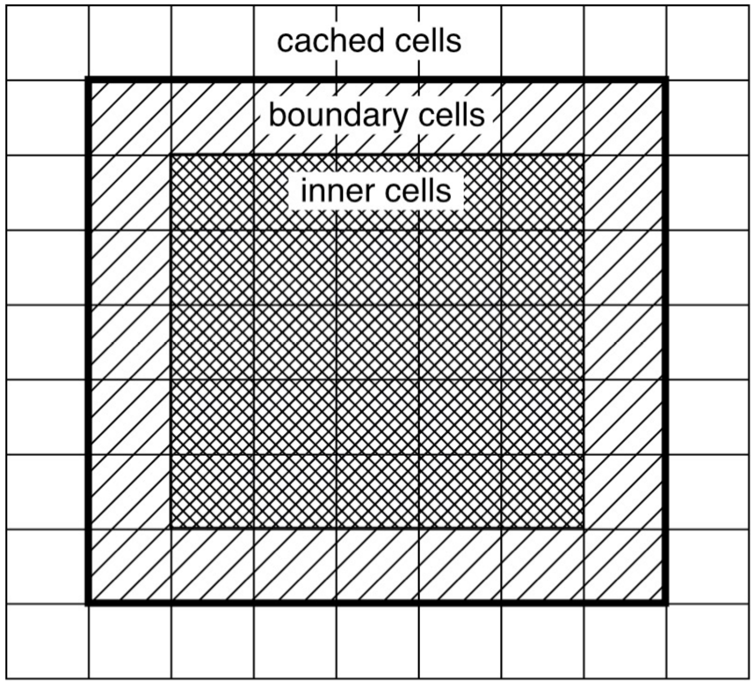




Grid-Enabled MD Algorithm

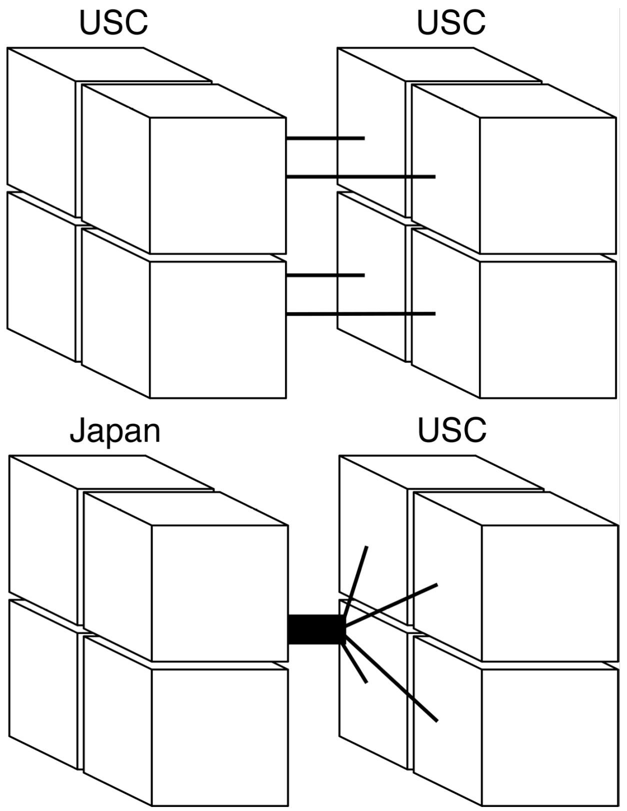
Grid MD algorithm:

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



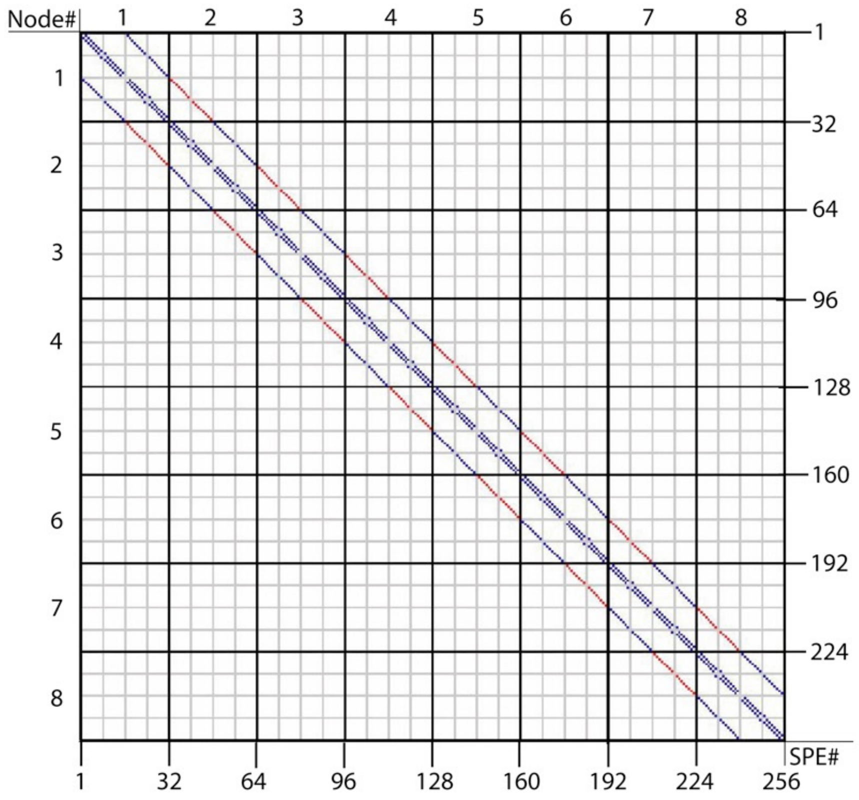
Renormalized Messages:

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

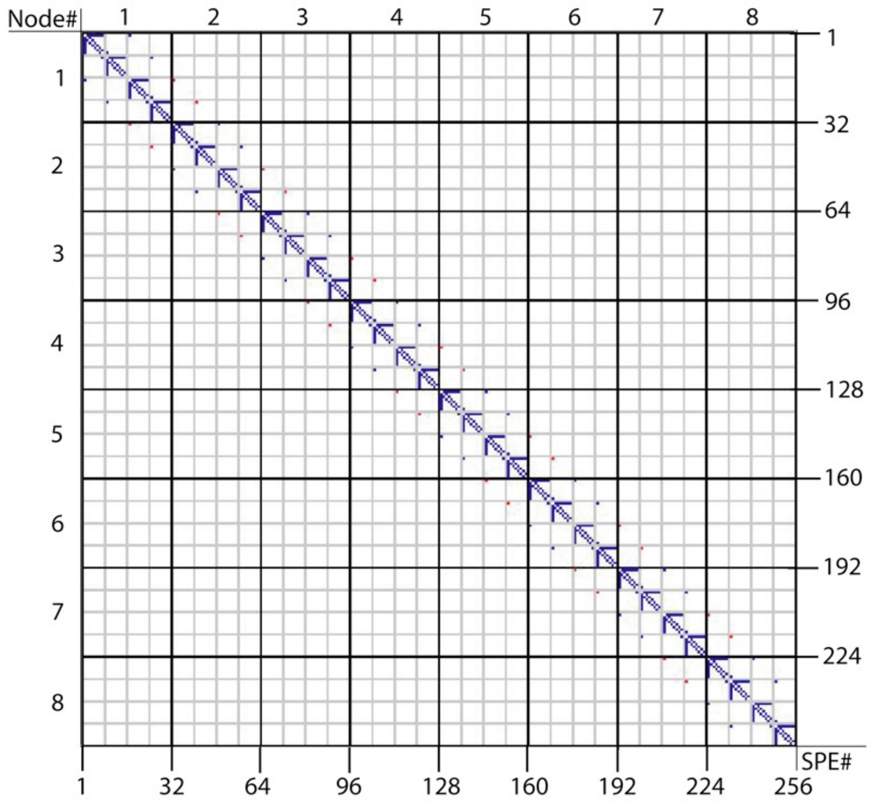


Renormalized Messages

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



Original



Renormalized

*LANL Roadrunner: first petaflop/s computer

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

Where to Go from Here

- **Performance profiling: First thing to find is how well/badly your program is performing in terms of flop/s performance, vectorization, cache miss, *etc*.**
- **Use professional tools like Intel VTune & Advisor if available on your computer:**

https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html
https://www.intel.com/content/www/us/en/developer/tools/oneapi/advisor.html
https://www.youtube.com/watch?reload=9&v=ymy139CuAx8

Advisor 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**

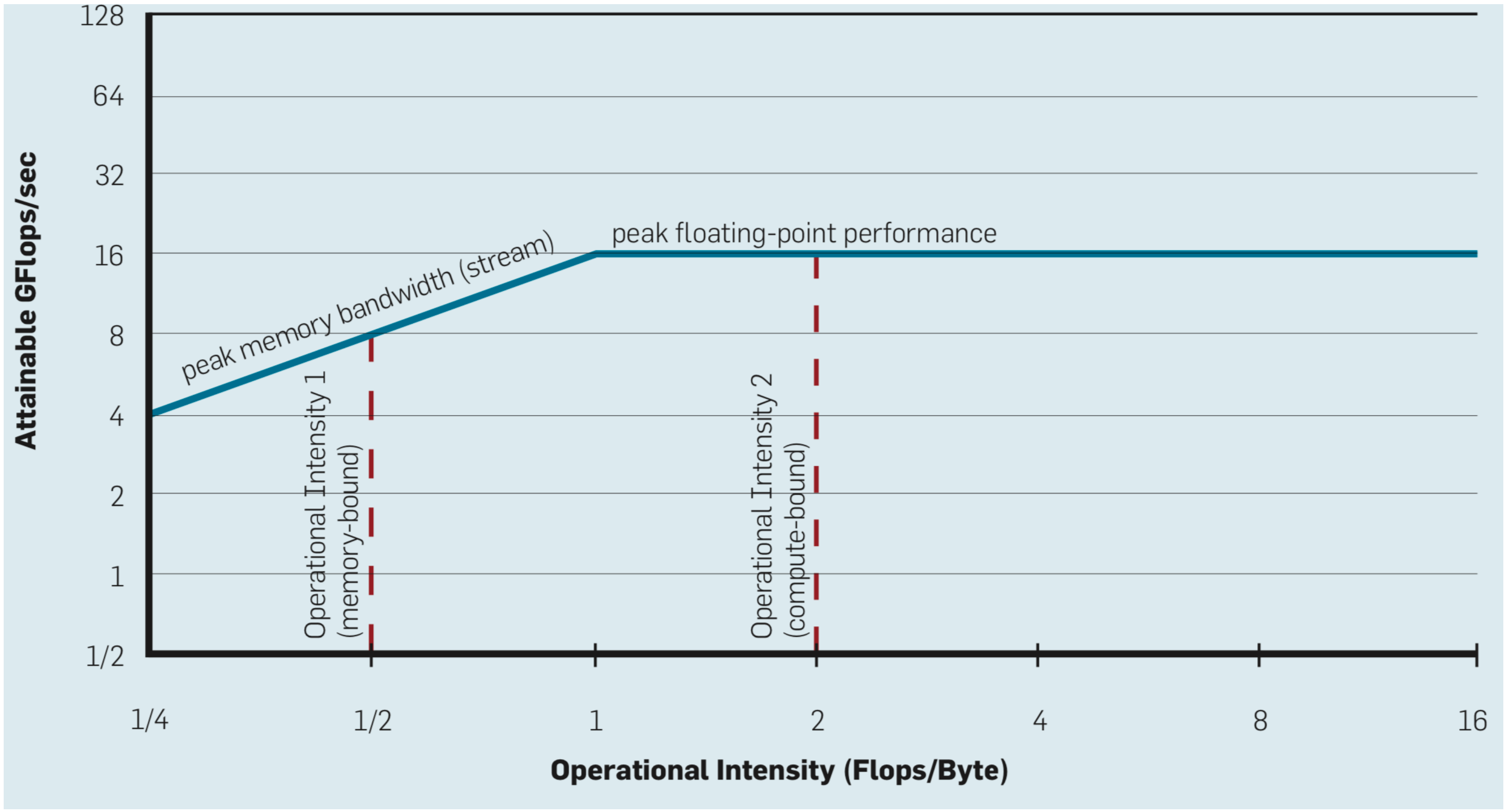
$$Attainable fp\ performance \left[\frac{\mathrm{Gflop}}{\mathrm{sec}}\right] = \min\left(\begin{array}{r} Peak\ fp\ performance \left[\frac{\mathrm{Gflop}}{\mathrm{sec}}\right], \\ Peak\ memory\ bandwidth \left[\frac{\mathrm{GByte}}{\mathrm{sec}}\right] \times Operational\ intensity \left[\frac{\mathrm{flop}}{\mathrm{Byte}}\right]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

S. Williams *et al*., *Commun. ACM* **52(4)**, 65 ('09)
V. Elango *et al*., *ACM. T. Arch. Code Opt.* **11**, 67 ('15)

Roofline Model of Performance

$$\mathit{Attainable\ fp}\left[\frac{\text{Gflop}}{\text{sec}}\right] =$$

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



Key: Data/computation locality

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