# Optimizing Molecular Dynamics

**Aiichiro Nakano**

*Collaboratory for Advanced Computing & Simulations*

*Department of Computer Science*

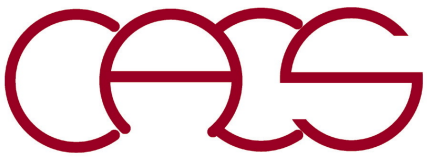
*Department of Physics & Astronomy*

*Department of Quantitative & Computational Biology*

*University of Southern California*

**Email: anakano@usc.edu**

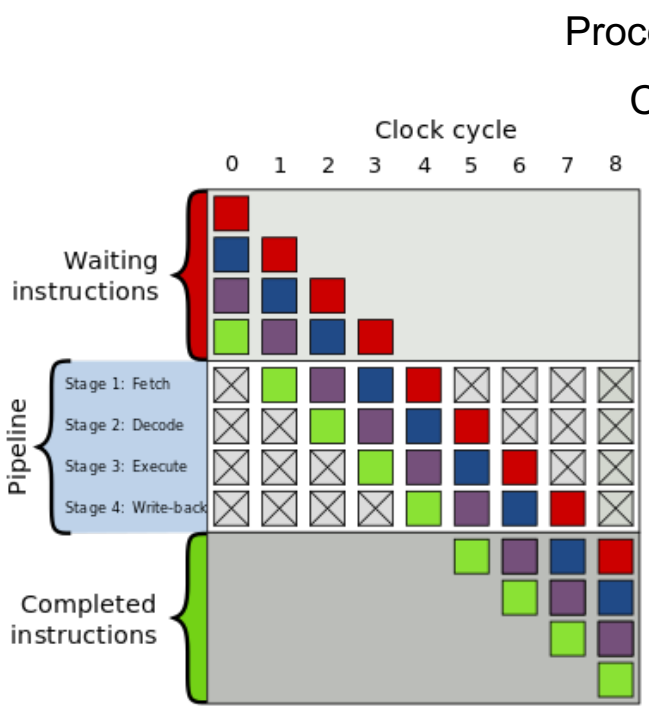
- **Intranode optimization: CPU & memory access**
- **Internode optimization: Communication**

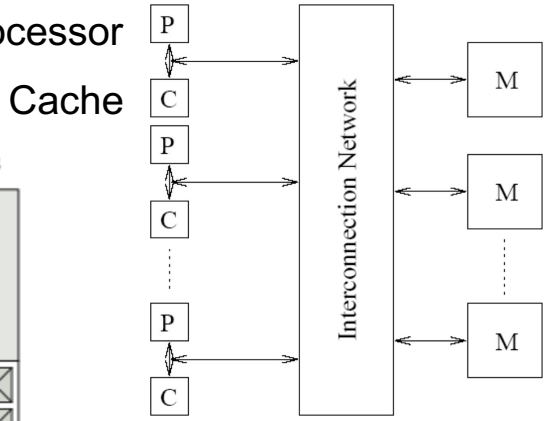


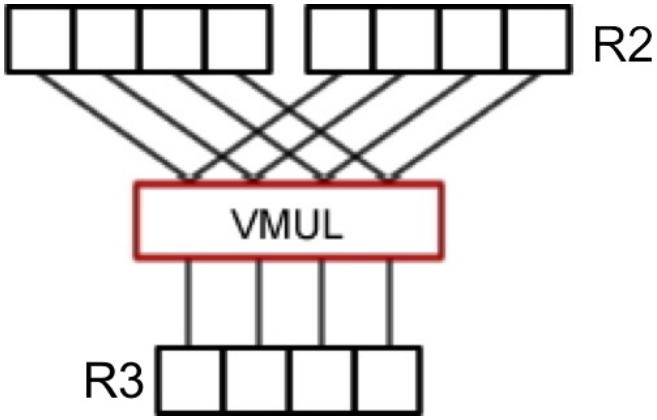
**Data/computation locality!**

# Key Hardware Features

- ***Pipelining*: Multiple stages of computation are executed concurrently for multiple data elements**
- ***Cache*: Fast memory that holds a subset of main memory**
- ***Vectorization*: Single-instruction multiple-data (SIMD) parallelism on vector registers each holding multiple data elements simultaneously hold**




Grama'03, Chap. 2; Berkeley CS267, Lec. 2

# Roofline Model of Performance

- **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**

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

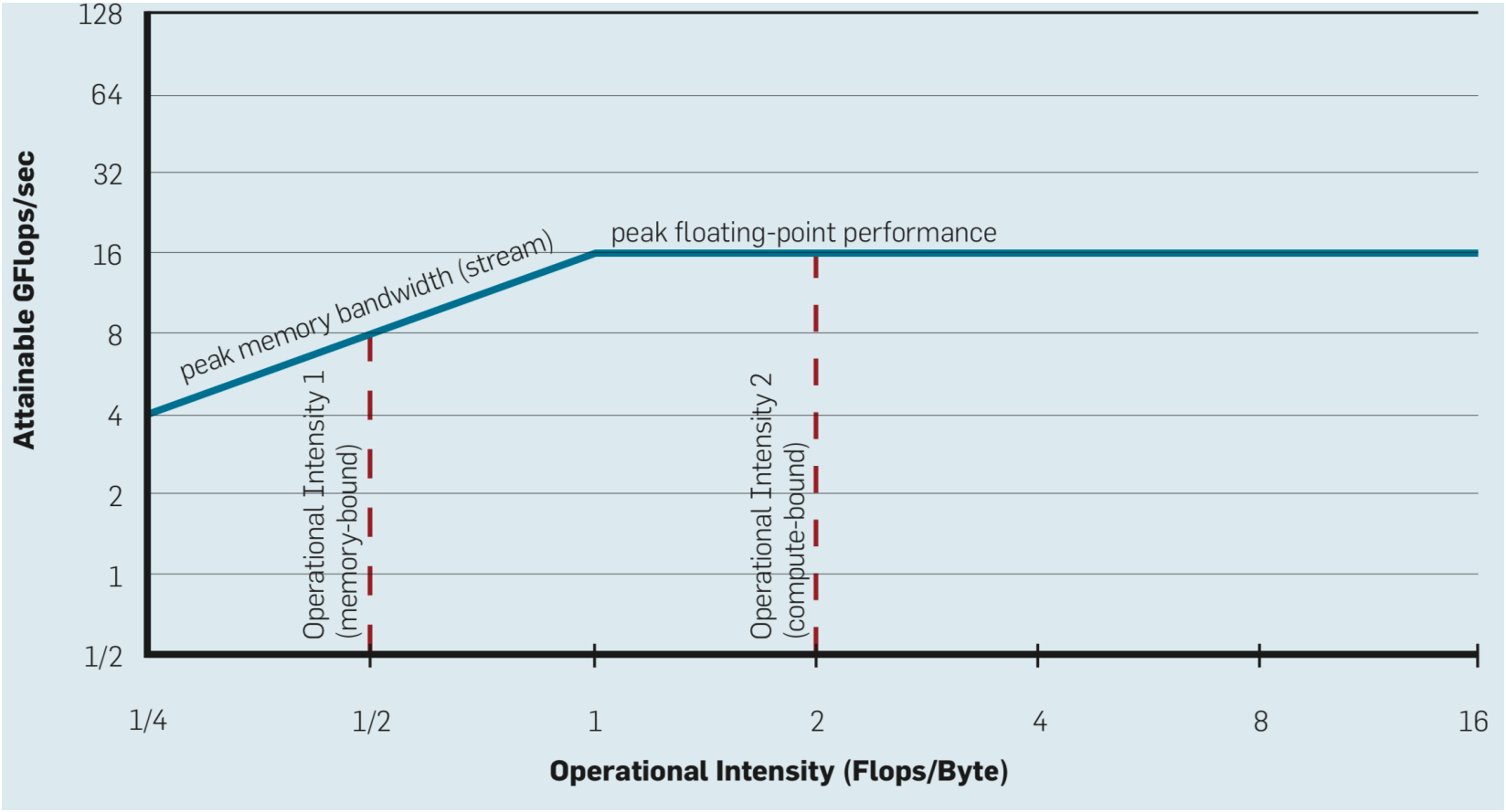
$$Peak\ fp\ \left[\frac{\text{Gflop}}{\text{sec}}\right] = \overbrace{\tilde{f}}^{\text{clock [GHz]}} \times \overbrace{n_{\text{core}}}^{\text{\# of cores}} \times \overbrace{n_{\text{vector}}}^{\text{\# of operands/vector}} \times 2\ \overbrace{n_{\text{FMA}}}^{\text{\# of FMA}}$$

FMA: fused multiply-add unit

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

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



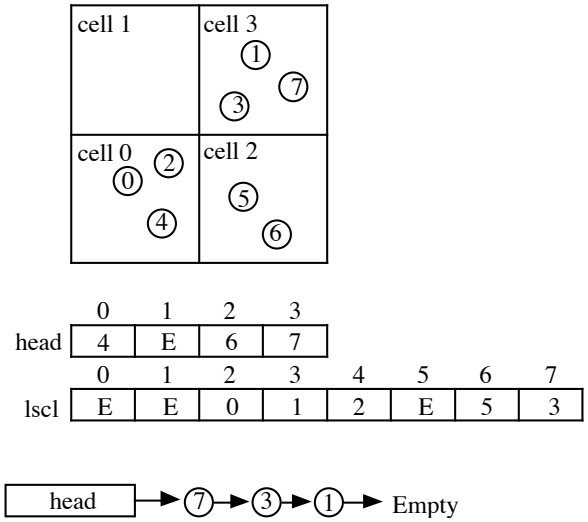

**Key: Data/computation locality:**

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

# Intranode: Memory Access

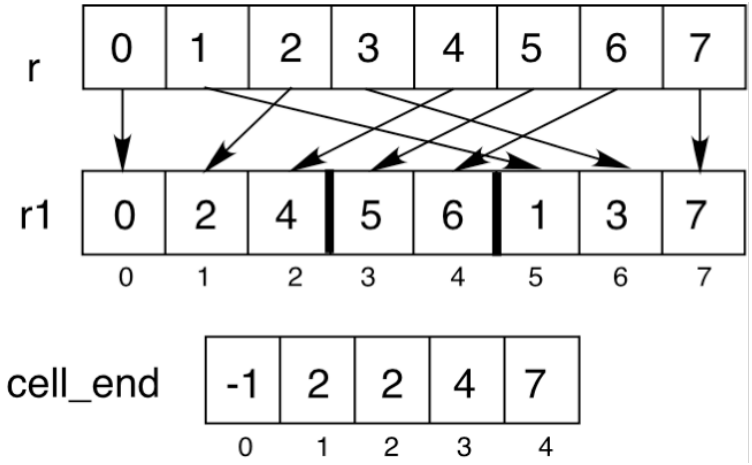
## Data re-ordering

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



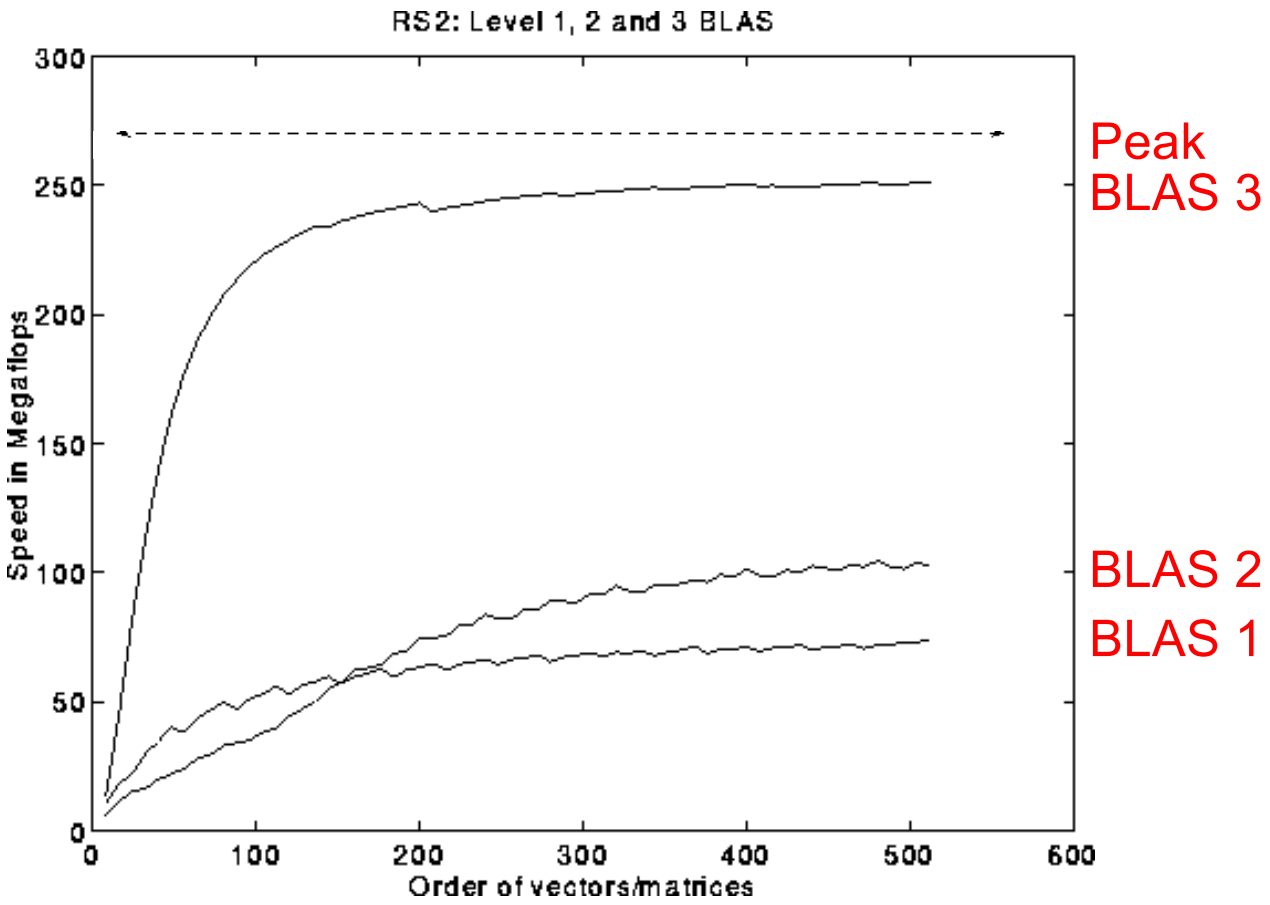

- **Data locality: Regular data layout**

```
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 Cell, multicore (AVX)?**

# 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)

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

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

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

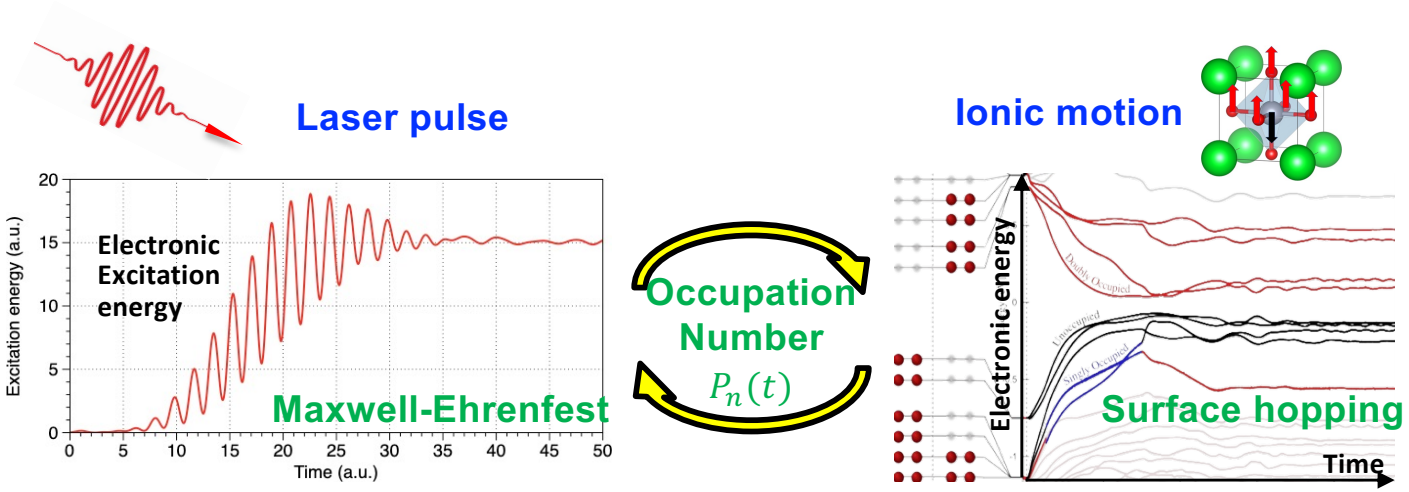
- **50.5% of the theoretical peak FLOP/s performance on 786,432 Blue Gene/Q cores (entire Mira at the Argonne Leadership Computing Facility)**
- **55% of the theoretical peak FLOP/s on Intel Xeon E5-2665**

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

# More BLASification

- **DCMESH (divide-&-conquer Maxwell+ Ehrenfest + surface-hopping) code**
- **Converted the most compute-intensive nonlocal-correction (NLC) computations into a matrix form (BLAS-GEMM)**

Linker *et al.*, *Science Adv.* **8**, eabk2625 ('22)



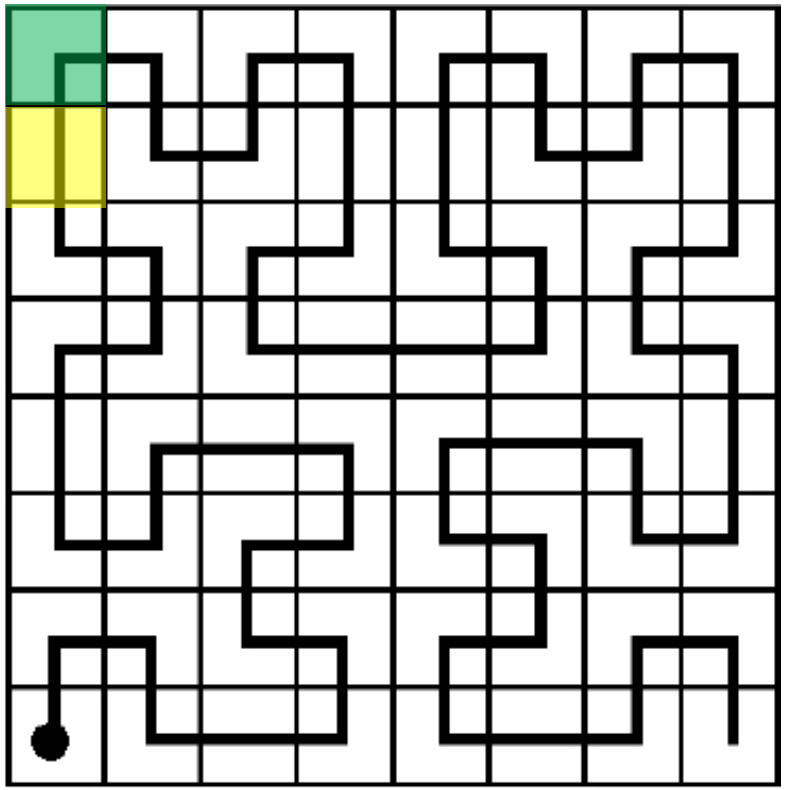

| Code | Wall time (s) | Speed-up |
|---|---|---|
| **Baseline** | 660.17 | 1 |
| **BLAS-GEMM** | 24.52 | 26.92 |

**Speed-up due to Algorithm Improvement for 2×2×2 $PbTiO_3$ on AMD 7513P**

# Computation Locality

**Data-to-computaion locality: How to traverse the linked-list 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

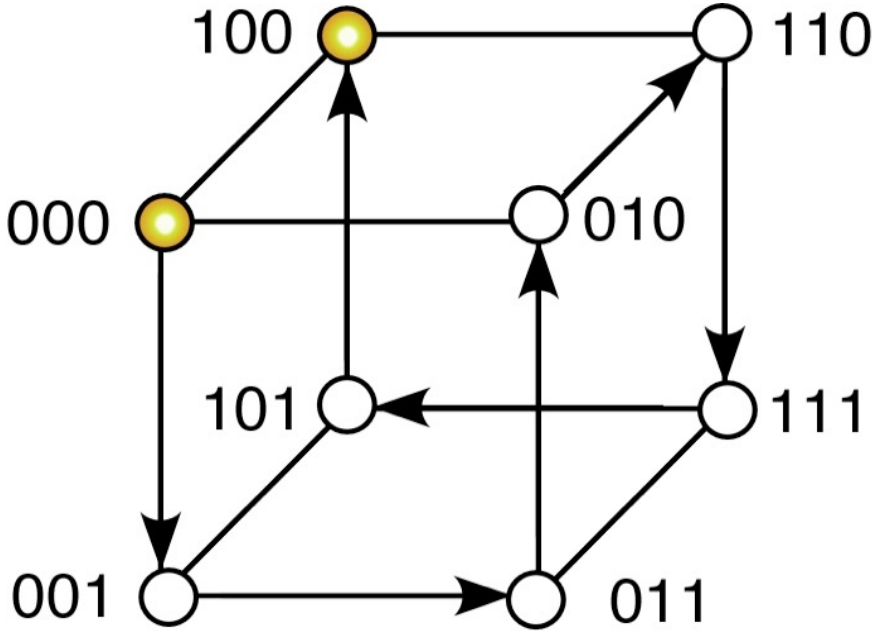
- **Gray code: a sequence of numbers such that successive numbers have Hamming distance 1**

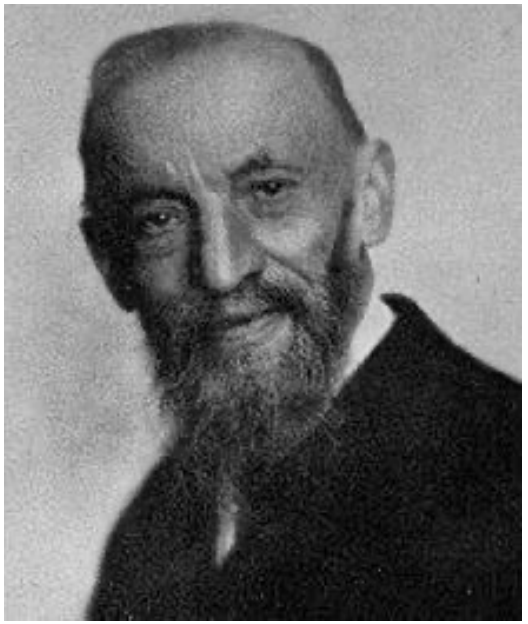
  **# of bits where two binary numbers differ**

```
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)

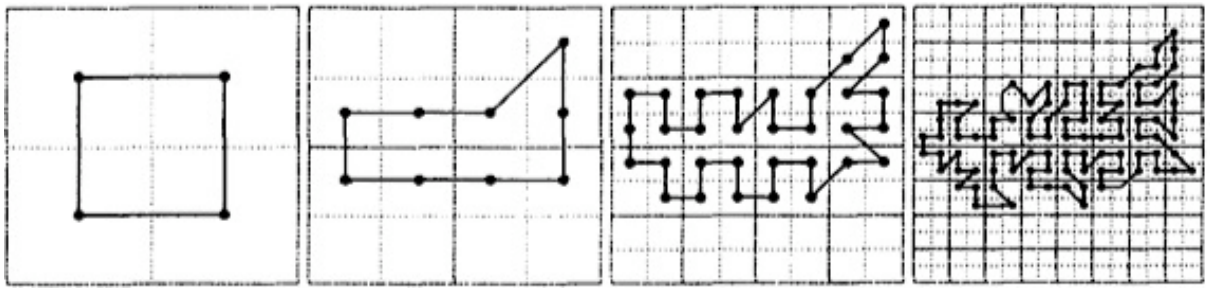


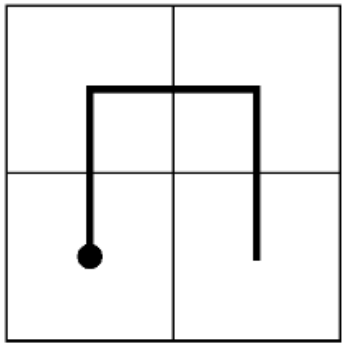
Giuseppe Peano (1858–1932)

# Hilbert-Peano Curve

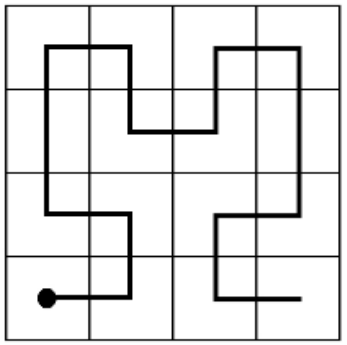
- **Hilbert curve: recursive application of the *d*-dimensional Gray codes**
- **2-dimensional Hilbert curve**



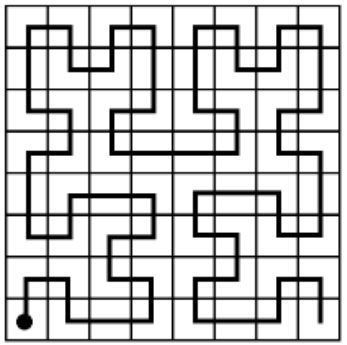
level 1



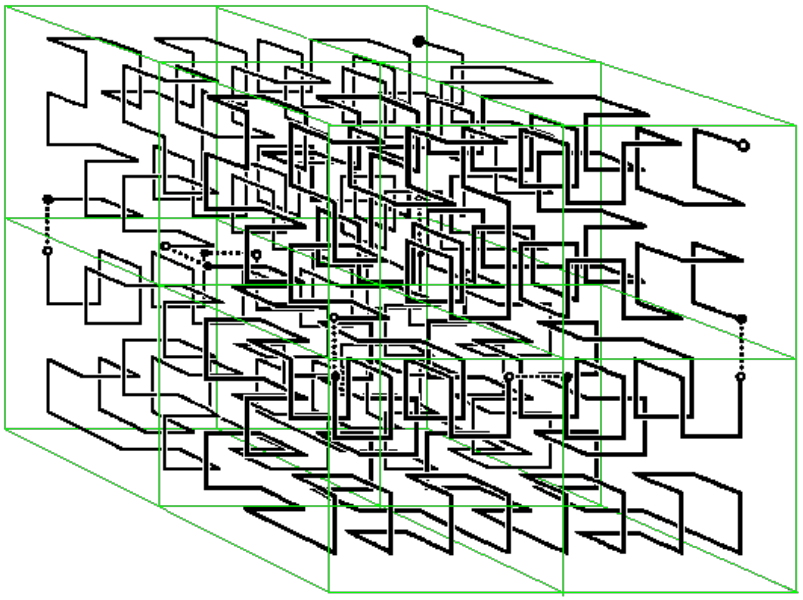
level 2



level 3



- **3-dimensional Hilbert curve**

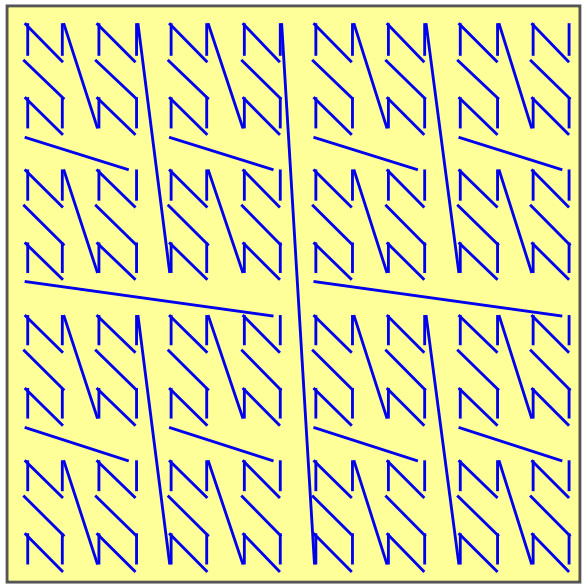


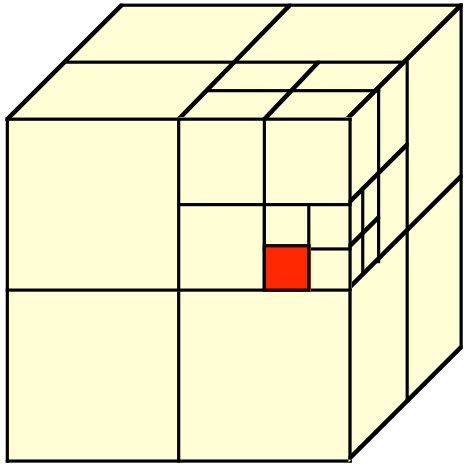
# Morton (Z) Curve

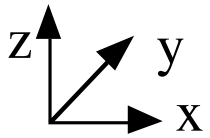
- **Spacefilling curve based on octree index**

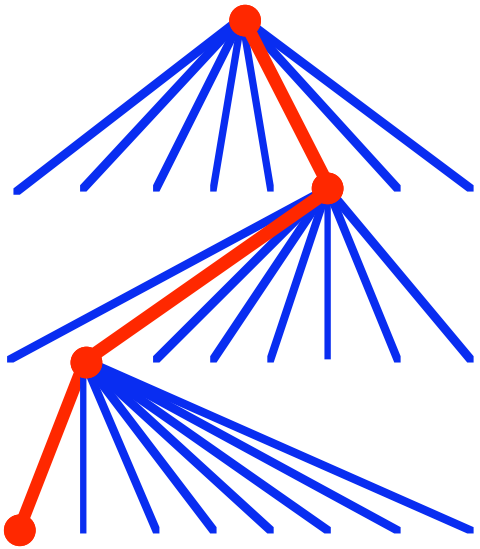
| | | | |
|---|---|---|---|
| x = | 1 | 1 | 0 |
| y = | 0 | 0 | 0 |
| z = | 1 | 0 | 0 |
| R = | 101 | 001 | 000 |

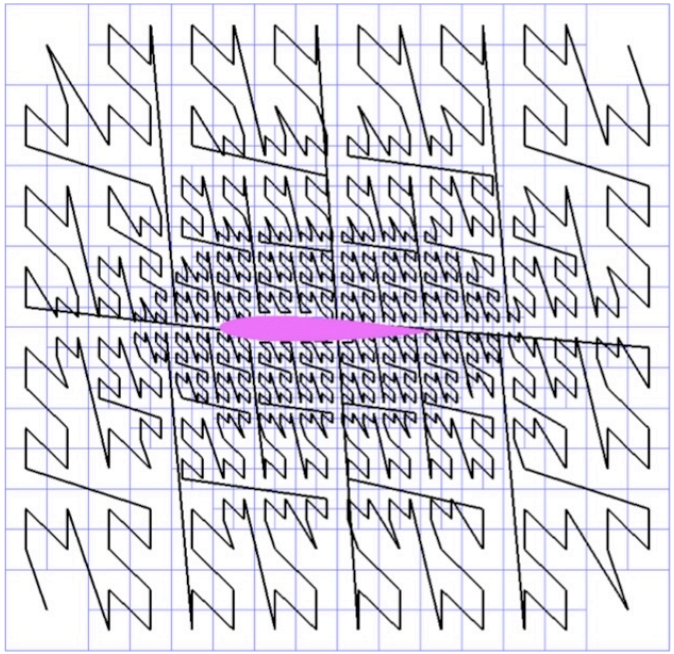
- **3D → list map preserves spatial proximity**
- **Multiresolution analysis made easy**









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

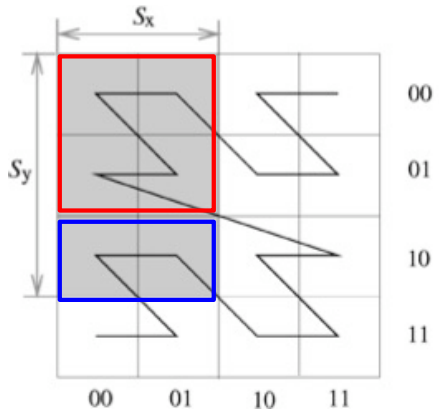
# Analysis of Data Locaility

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

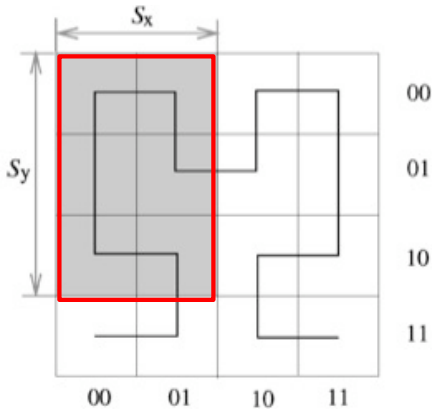
Bongki Moon, H.V. Jagadish, Christos Faloutsos, *Member, IEEE*, and Joel H. Saltz, *Member, IEEE*

**Morton**



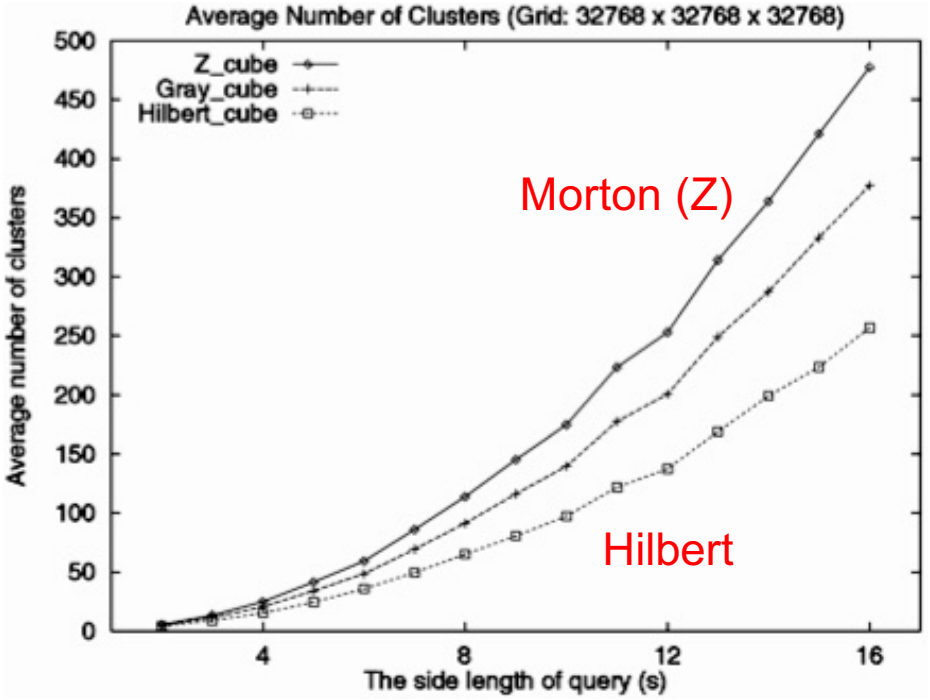

**2 clusters**

**Hilbert**




**1 cluster**

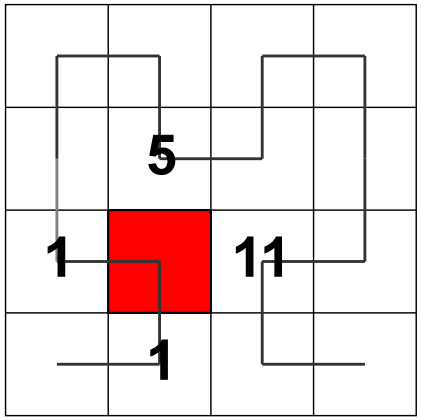
**Cluster ~ cache line ~ latency cost**

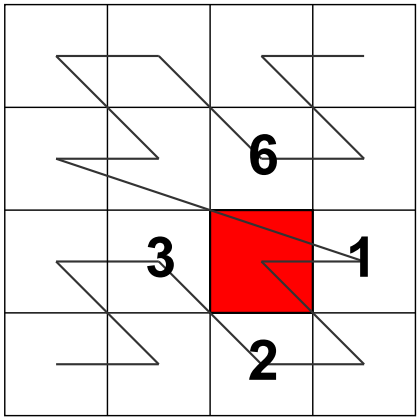



**Hilbert curve is better than Morton curve for spatial range query**

# Alternative Locality Measure for MD

- **Doubly nested loops of cell access**
- **Evaluate curves based on along-curve distances to neighbors**
- **Compare number below and 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 $2<k_c<13$

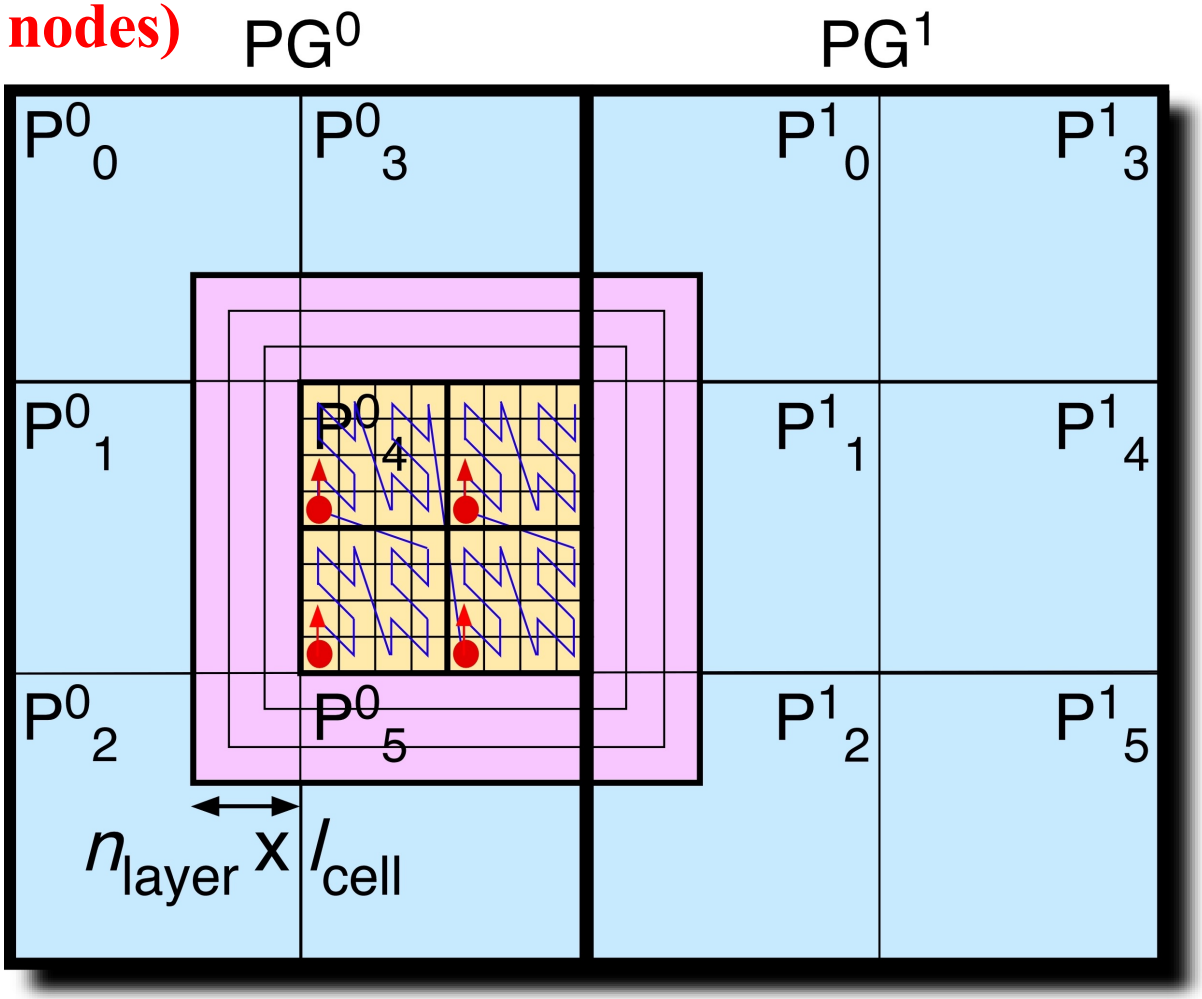
Scott Calaghan (CSCI 653 final project)

# Tunable Hierarchical Cellular Decomposition

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

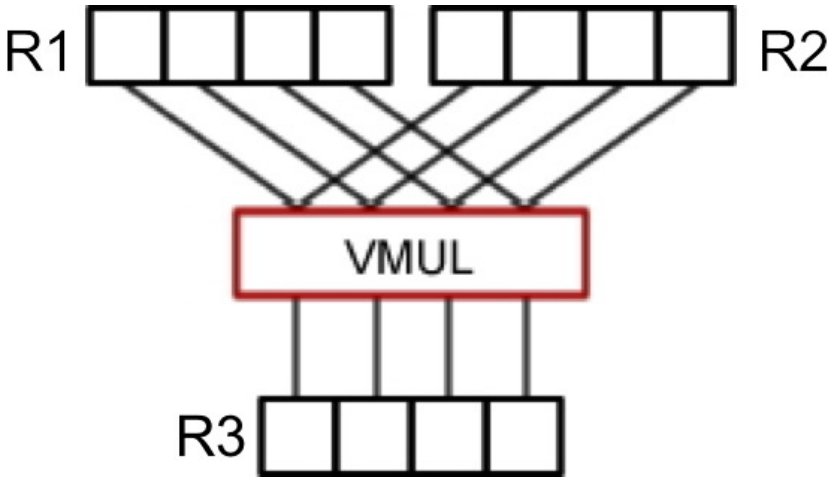
- **Spatial decomposition with data "caching" & "migration"**
- **Computational cells (*e.g.* linked-list cells in MD) < cell blocks (threads) < processes ($P^{\gamma}_{\pi}$, spatial decomposition subsystems) < process groups ($P^{\gamma}$, Grid nodes)**
- **Multilayer cellular decomposition (MCD) for *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*



# SIMD/Vector Operation

- **Single-instruction multiple-data (SIMD) parallelism: An arithmetic operation is operated on multiple operand-pairs stored in vector registers, each of which can hold multiple double-precision numbers.**

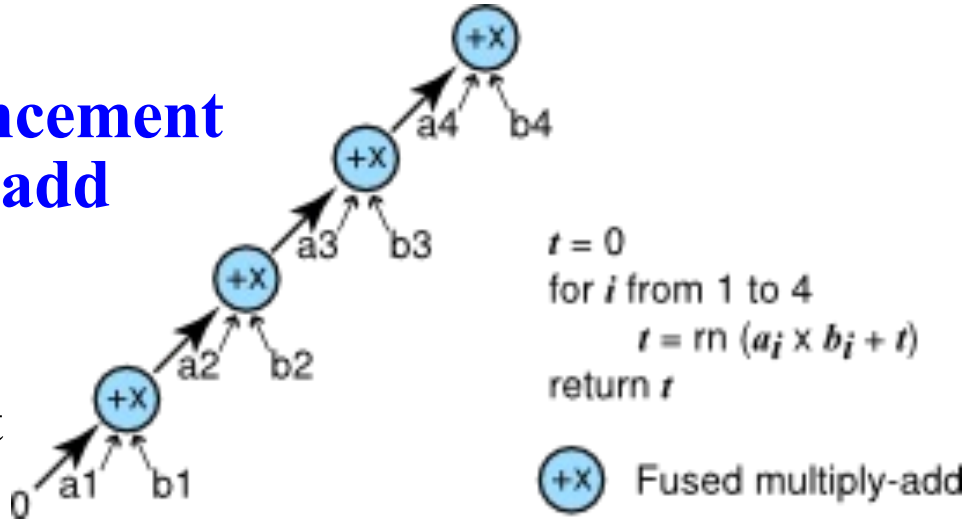



**Example:** Vector multiplier (VMUL) loads data from two vector registers, R1 and R2, each holding 4 double-precision numbers, concurrently performs 4 multiplications, and stores the results on vector register R3.

- **Peak performance enhancement on top of fused multiply-add (FMA) unit.**

$$a \leftarrow a + b \times c$$

with 1-cycle throughput



# Vector Processing at CARC

## Node information

https://www.carc.usc.edu/user-information/user-guides/hpc-basics/discovery-resources

| CPU model | Microarchitecture | Partitions | SSE | SSE2 | SSE3 | SSE4 | AVX | AVX2 | AVX-512 |
|---|---|---|---|---|---|---|---|---|---|
| xeon-2650v2 | ivybridge | oneweek, debug | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| xeon-2640v3 | haswell | main, oneweek, debug | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| xeon-2640v4 | broadwell | main, gpu, debug | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| xeon-4116 | skylake_avx512 | main | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| xeon-6130 | skylake_avx512 | gpu | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| epyc-7542 | zen2 | epyc-64 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| epyc-7513 | zen3 | epyc-64, gpu, largemem | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| epyc-7282 | zen2 | gpu | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |

## Intel & AMD advanced vector extension (AVX):

- AVX2 operates on 4 double-precision floating-point numbers
- AVX512 8

# SIMD Vectorization: MD

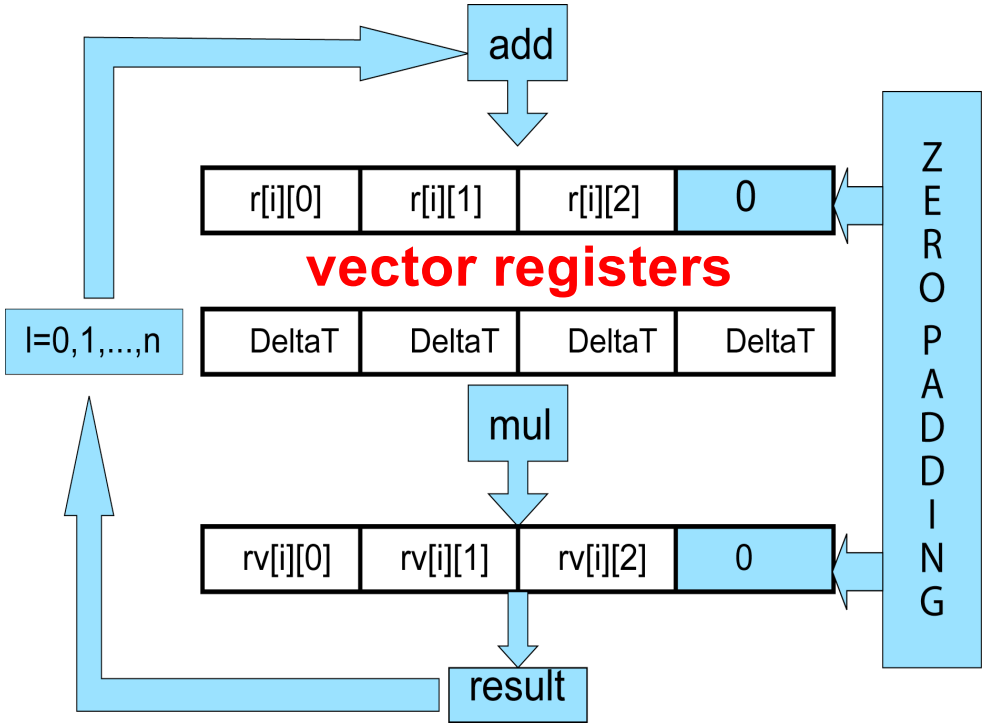
- **Single-instruction multiple-data (SIMD) parallelism**

**(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**

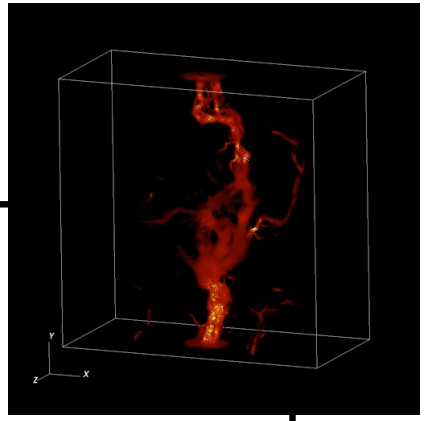



***cf.* False-sharing avoidance**

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

# SIMD Vectorization: LBM

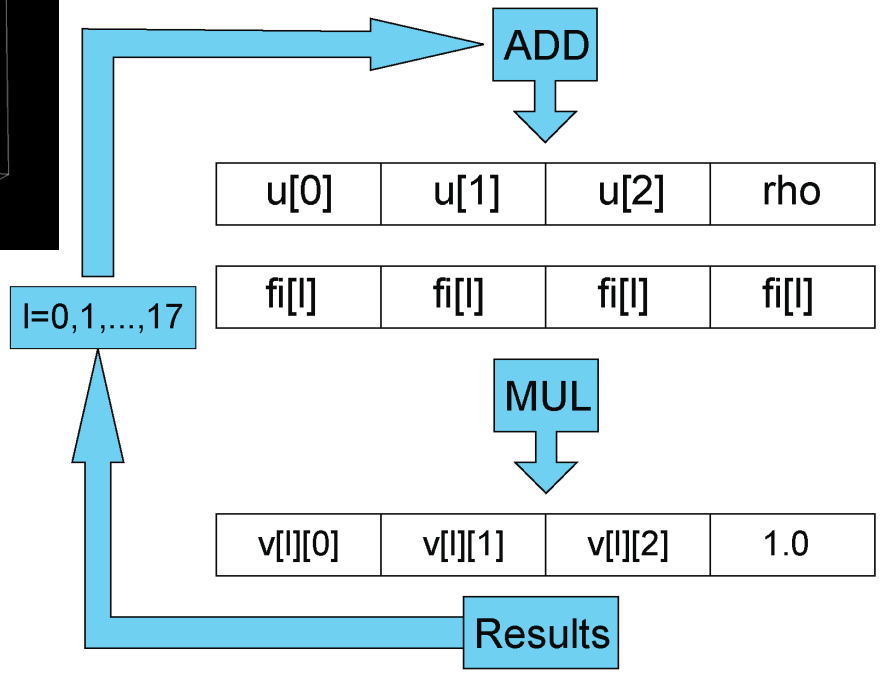
- **Translocated statement fusion in lattice-Boltzmann flow simulation**



**Original solution**

```
for(i=0;i<3;i++){
  u[i]=0.0; rho=0.0;
  for(l=0;l<18;l++){
    fi[l] = f[18*cnz+1];
    u[i] += fi[l]*v[l][i];
    rho += fi[l];
  }
}
```

**SIMD solution**




**3×18×5 = 270 computation**

**18×4 = 72 computation**

**Ideal Speedup 3.5**

**SIMDizable mathematical formulations:**

**Special relativity, quaternion, *etc.***

$$J^{\alpha} = (c\rho, j^1, j^2, j^3)$$

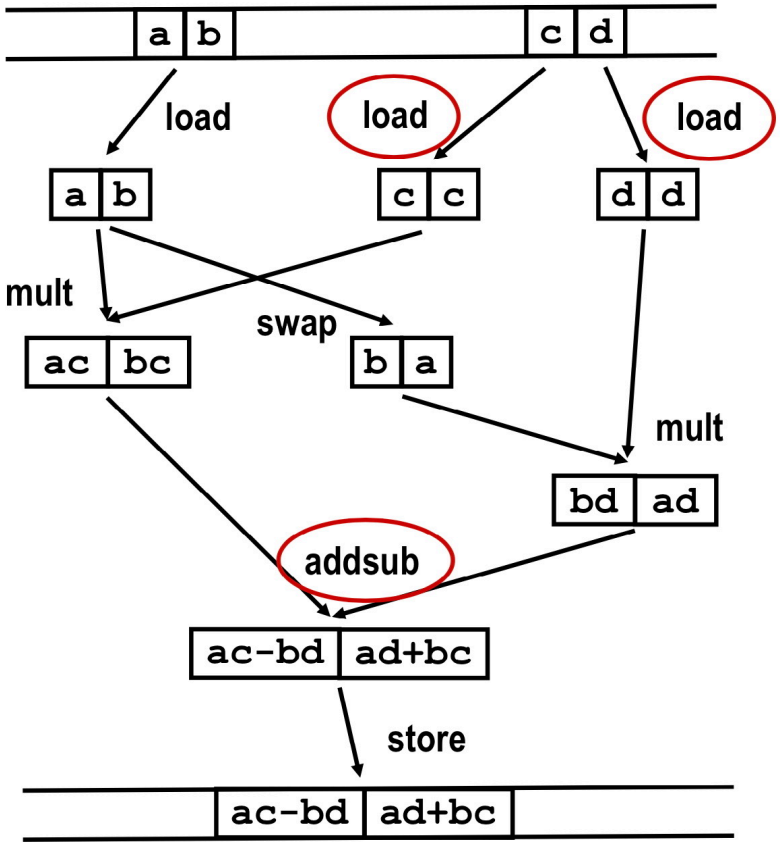
$$A^{\alpha} = (\phi/c, A^1, A^2, A^3)$$

$$\boxdot A^{\alpha} = \left(\frac{1}{c^2}\frac{\partial^2}{\partial t^2} - \nabla^2\right) A^{\alpha} = \frac{4\pi}{c} J^{\alpha}$$

# SIMDized Complex Multiplication

- **SSE (streaming SIMD extension) instruction set**
  - Pre-AVX instruction set
- **For quantum dynamics?**

$$(a + ib)(c + id) = (ac - bd) + i(ad + bc)$$



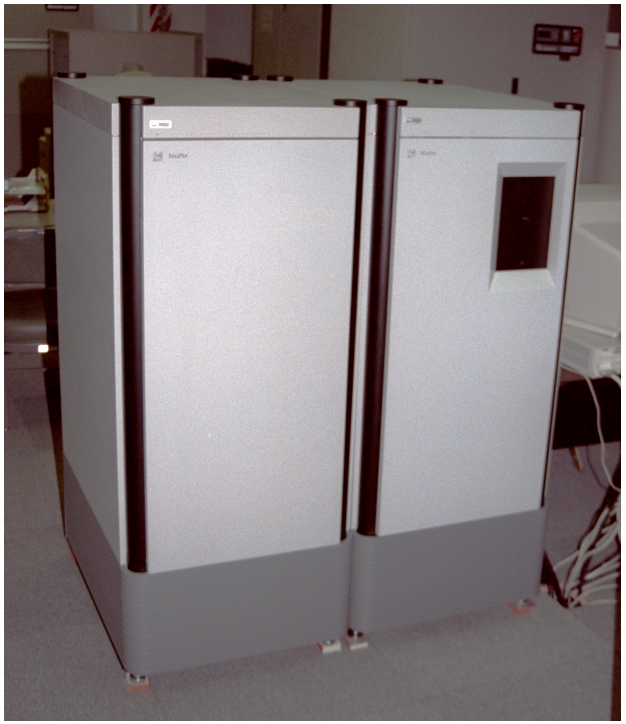

Memory

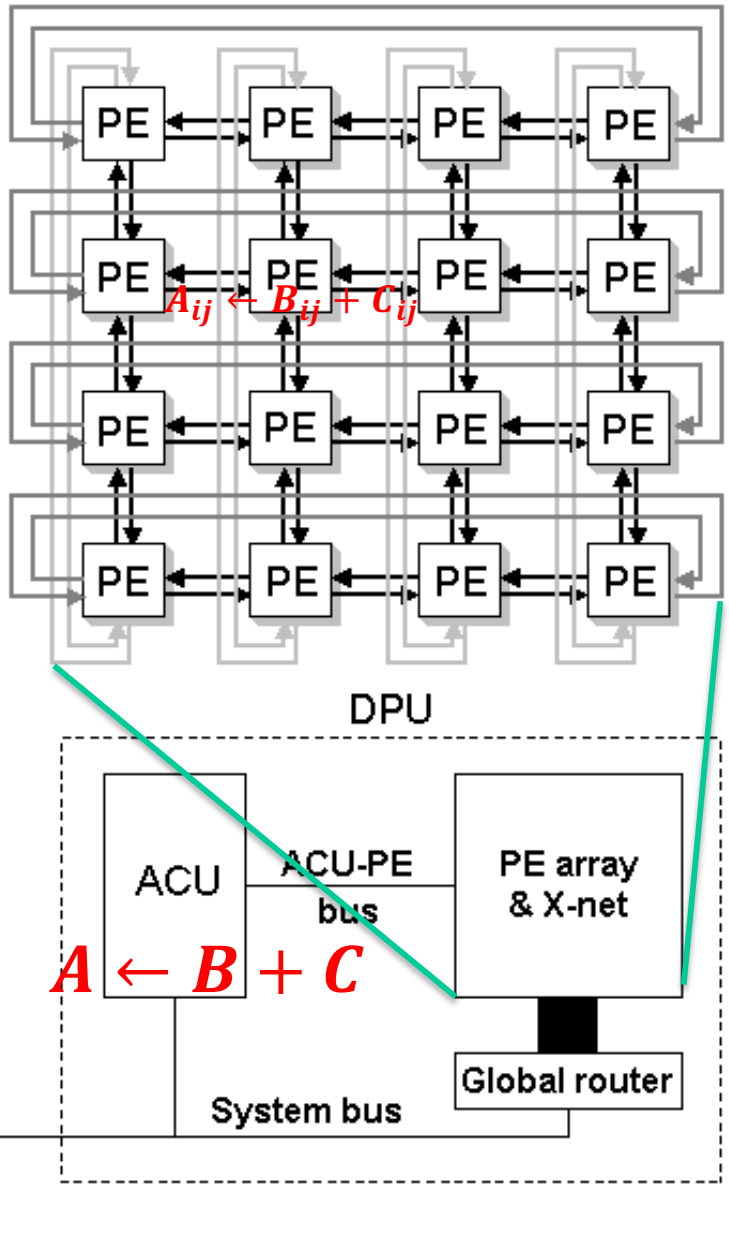
Result:
4 load/stores
3 arithm. ops.
1 reorder op

Not available in SSE2

Memory

# Massive SIMD Data Parallelism



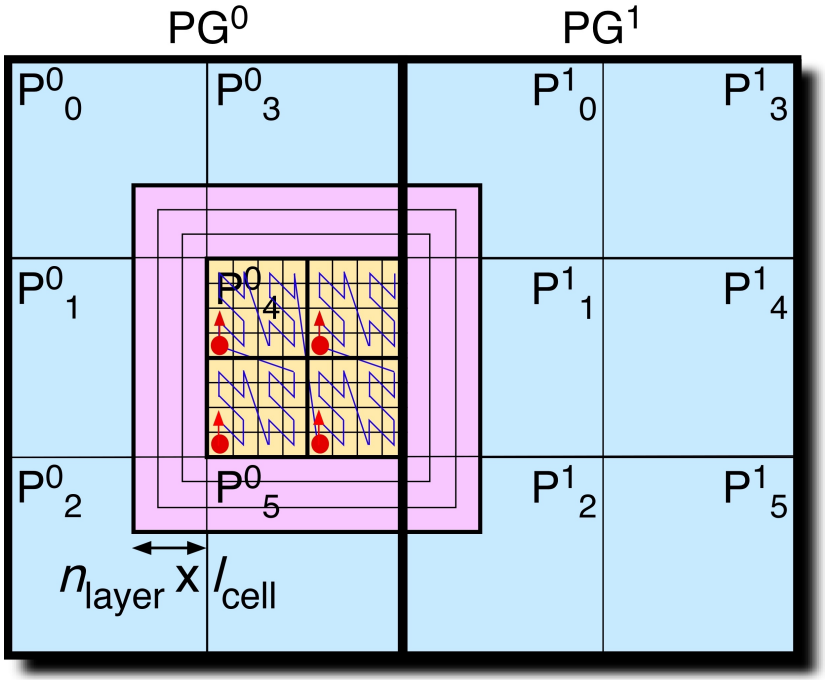



**Quantum dynamics on 8,192-processor (128 × 64) MasPar 1208B**

Nakano,
*Comput. Phys. Commun.*
**83**, 181 ('94)

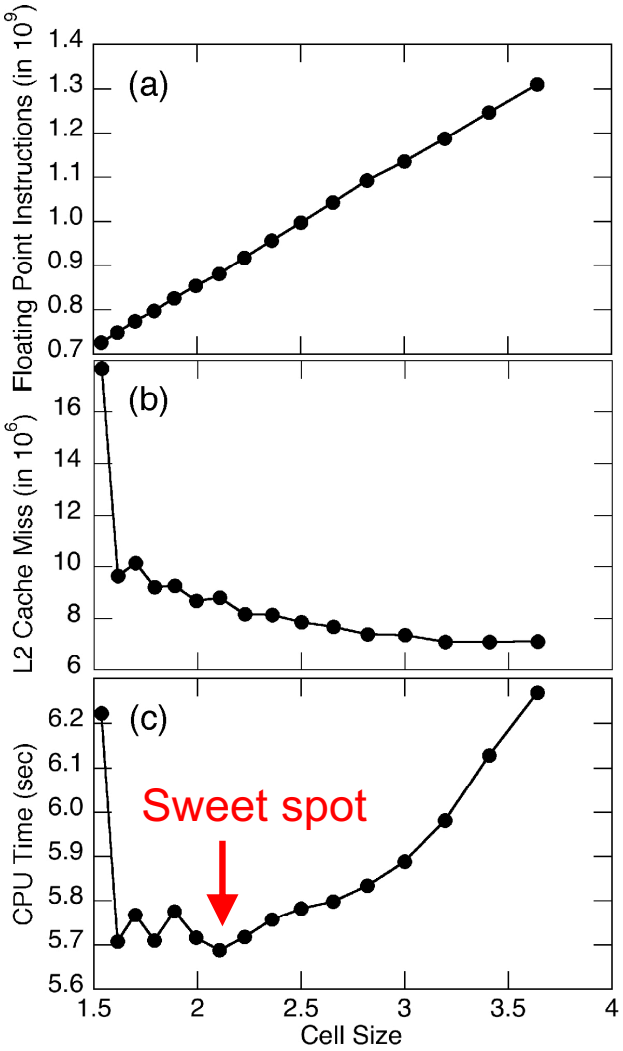
**See lecture on "pre-Beowulf parallel computing"**

# Performance Tunability




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

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



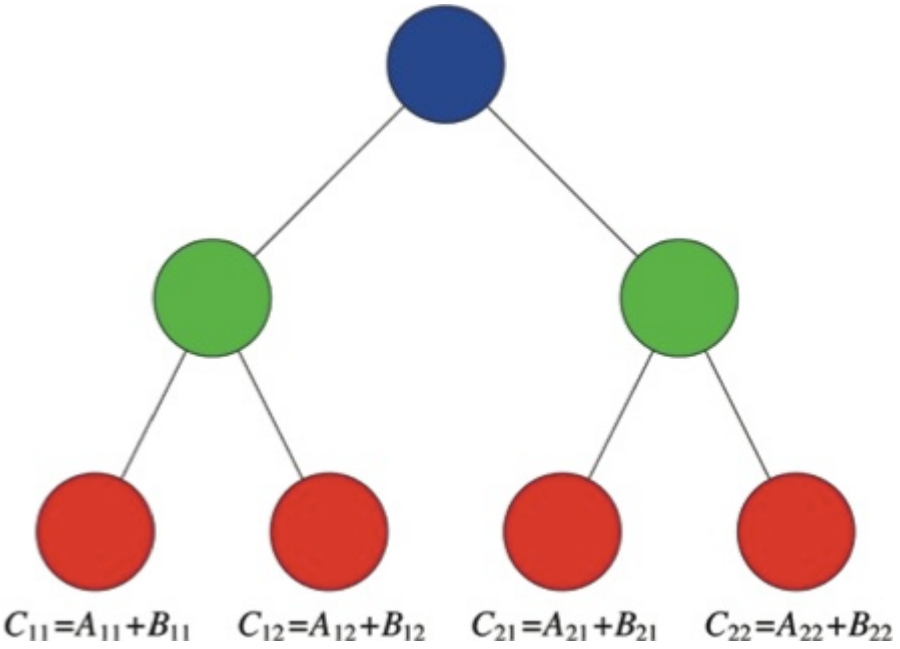

**MPI/OpenMP parallelism trade-off:**

**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 | **4.19** | 62.5 |
| 2 | 4 | 5.75 | 58.9 |
| 4 | 2 | 8.60 | **54.9** |
| 8 | 1 | 12.5 | 120 |

# Cache-Oblivious Linked-List Cell MD?

- **Recursive blocking for cells?**




## Cache-Oblivious Algorithms

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

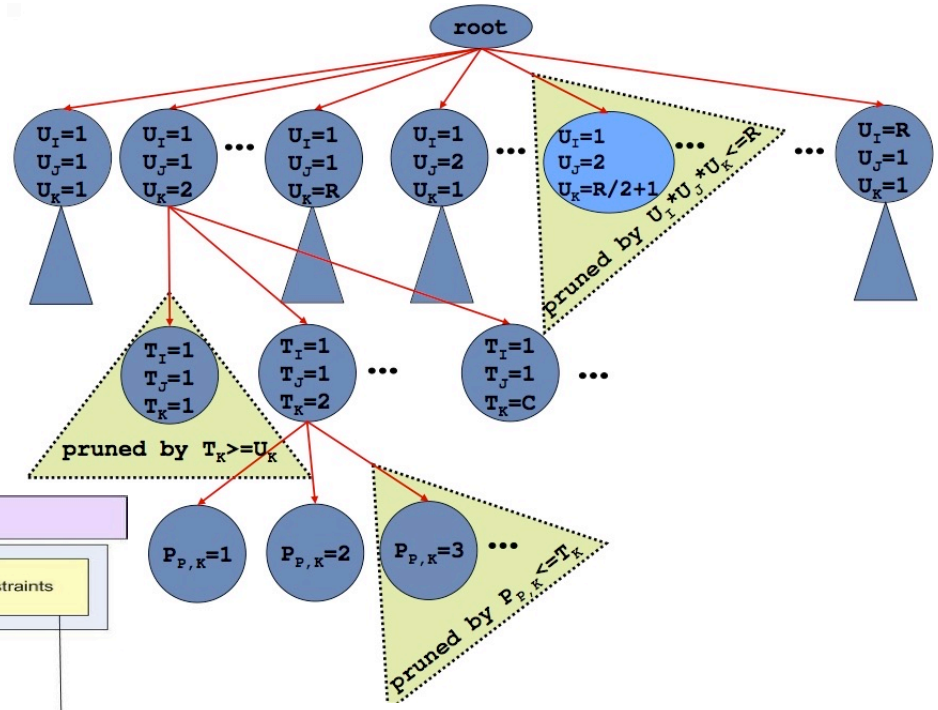
Matteo Frigo Charles E. Leiserson Harald Prokop Sridhar Ramachandran
*MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139*
{athena,cel,prokop,sridhar}@supertech.lcs.mit.edu

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.

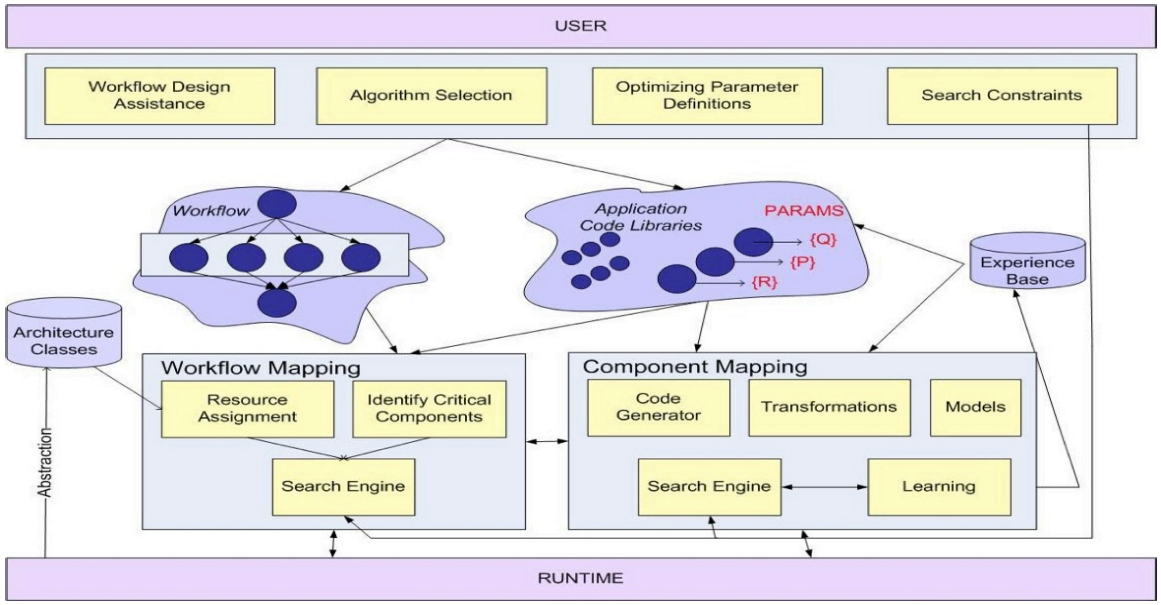
# Intelligent Performance Optimization

- **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)*

# Key Internode Feature

## Simple communication cost model

$$t_{\text{comm}} = t_l + m/b$$

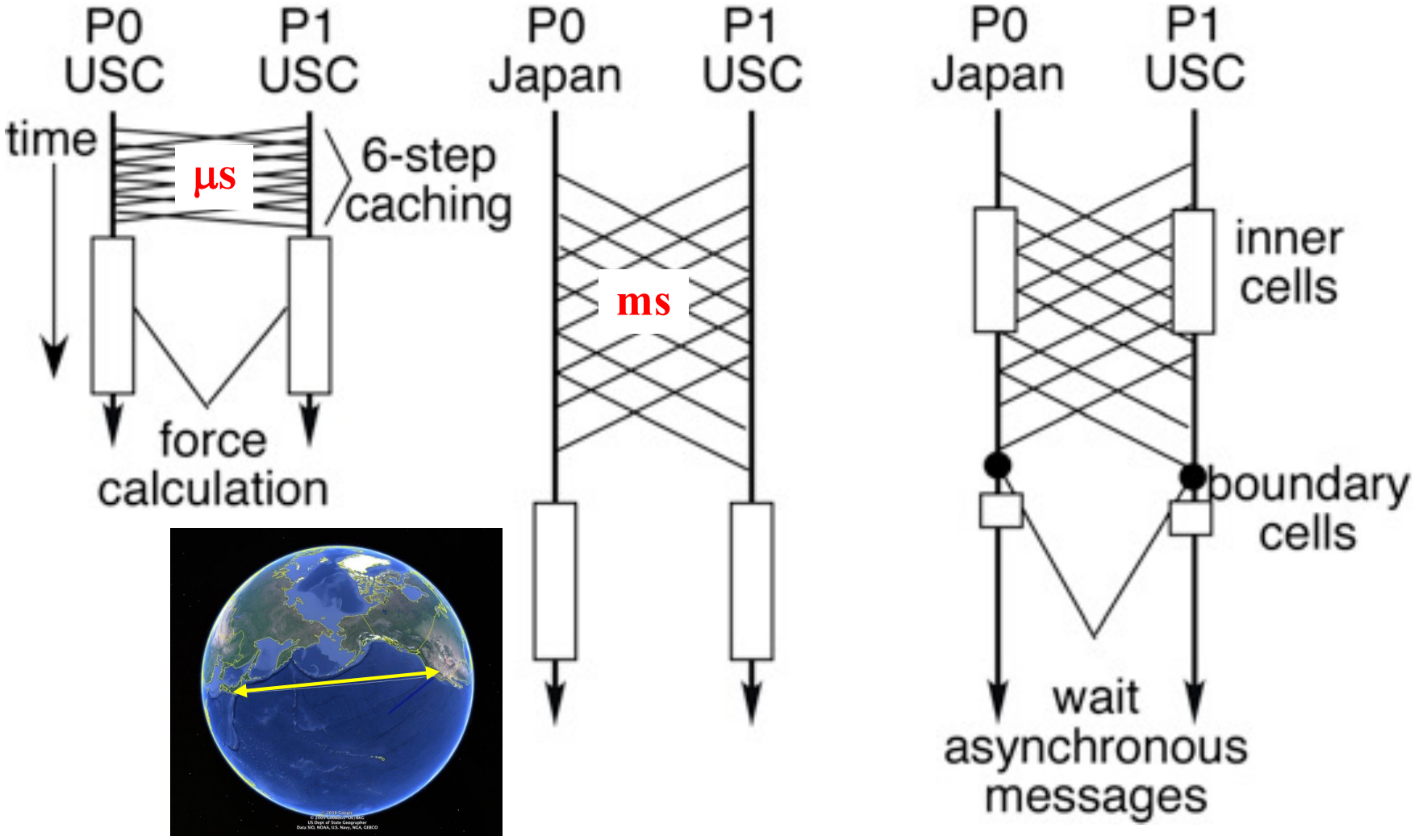
$t_{\text{comm}}$: Total communication time for $m$ Bytes

$t_l$: Latency = time delay for the head of a message to travel between the source & destination nodes

$b$: Bandwidth = number of Bytes per second that can be transmitted through the communication link

# Internode Optimization

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

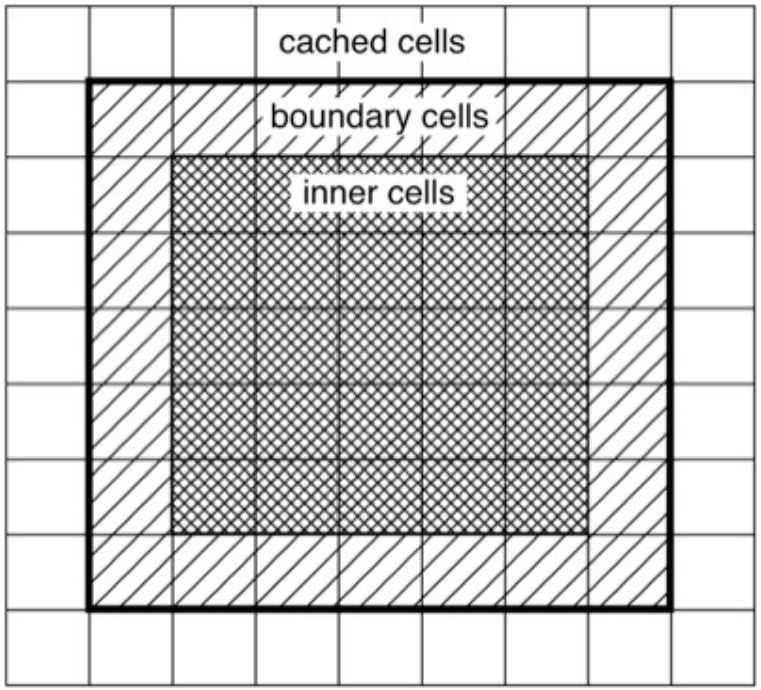



- **Overlap computation & communication to hide the latency**

# Grid-Enabled MD Algorithm

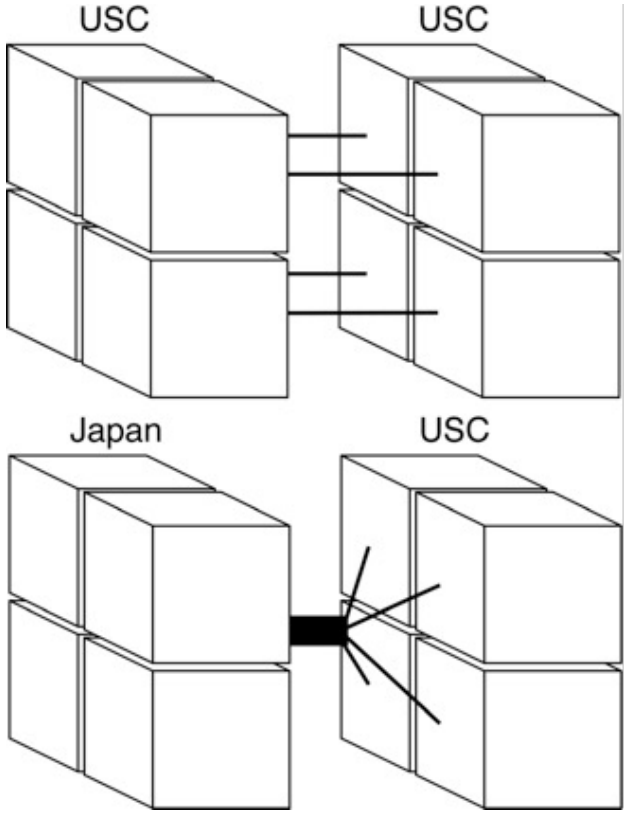
## Grid MD algorithm:

1. asynchronous receive of cells to be cached
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
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**

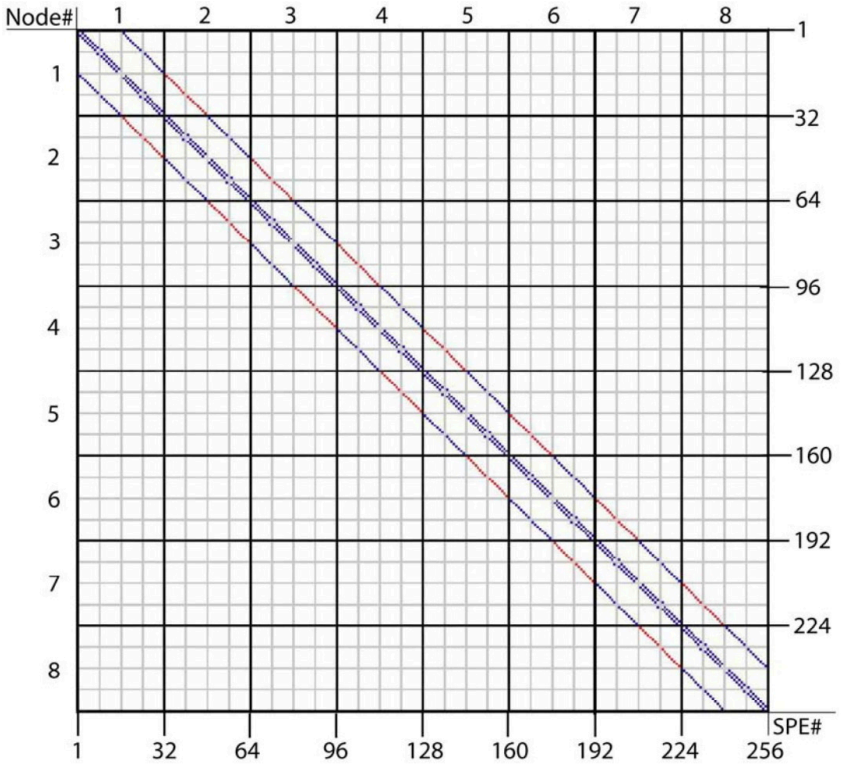



Kikuchi *et al.*, in *Proc. SC02*

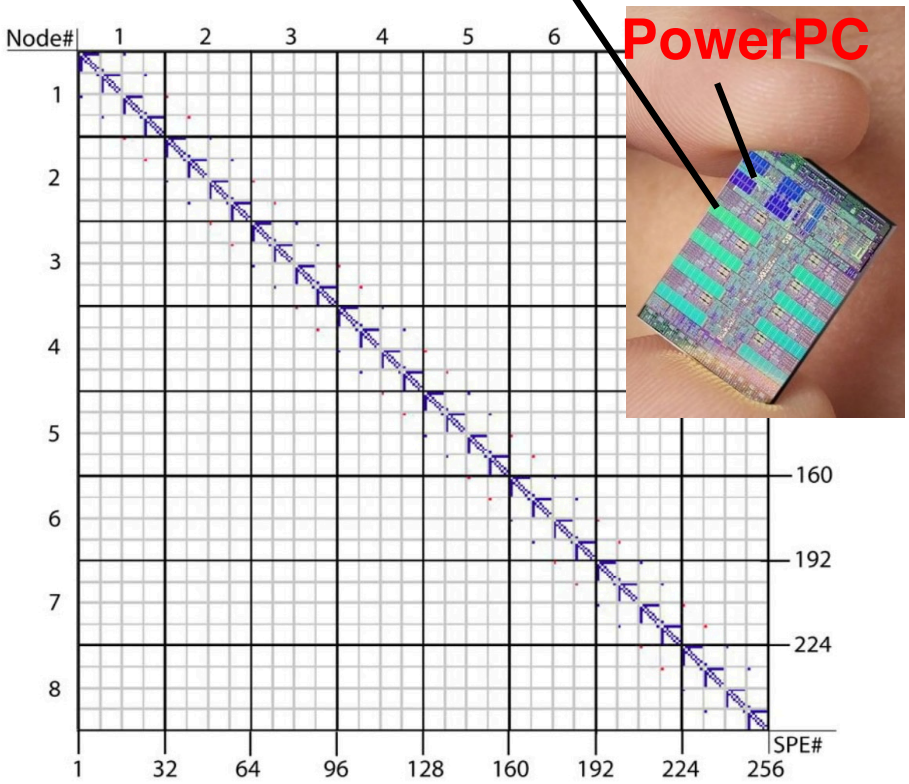
# Renormalized Message Passing

- **Mapping a 3D-lattice problem onto 8 computing nodes each with 4 Cell processors (or 32 cores)**
- **Renormalized message passing reduce the number of internode communications**

H. Dursun *et al*., *Par. Proc. Lett.* **19**, 535 ('09)


(a) Original

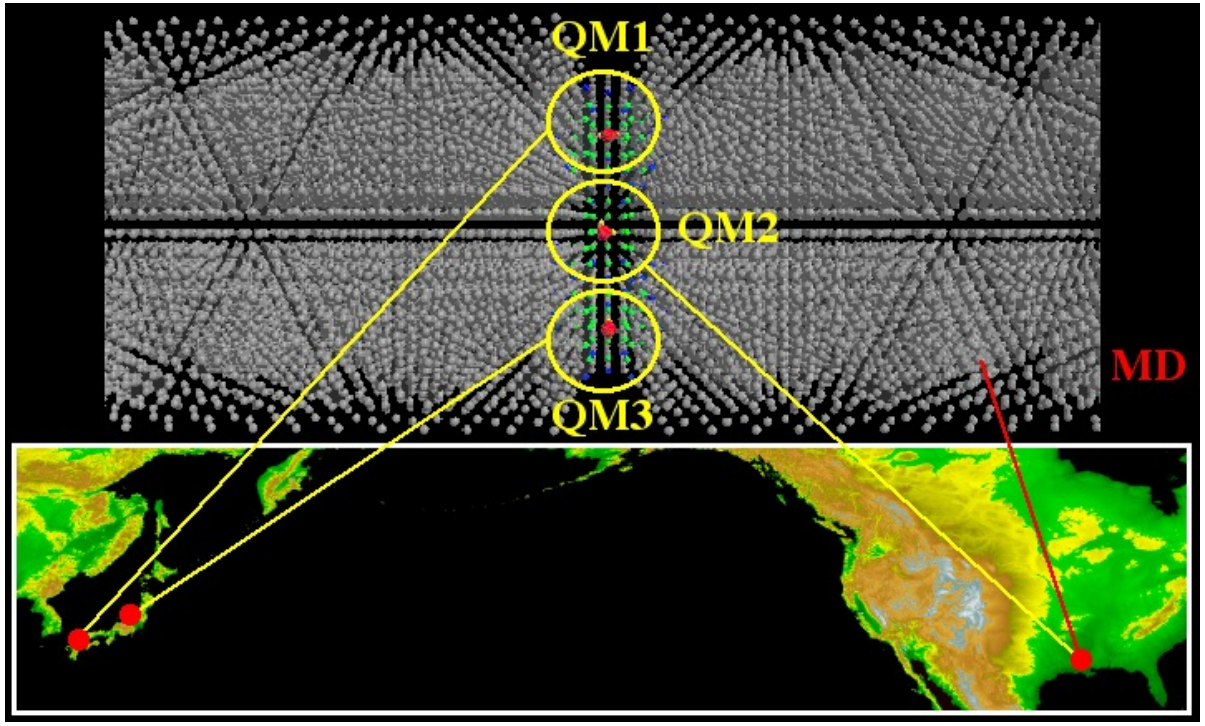


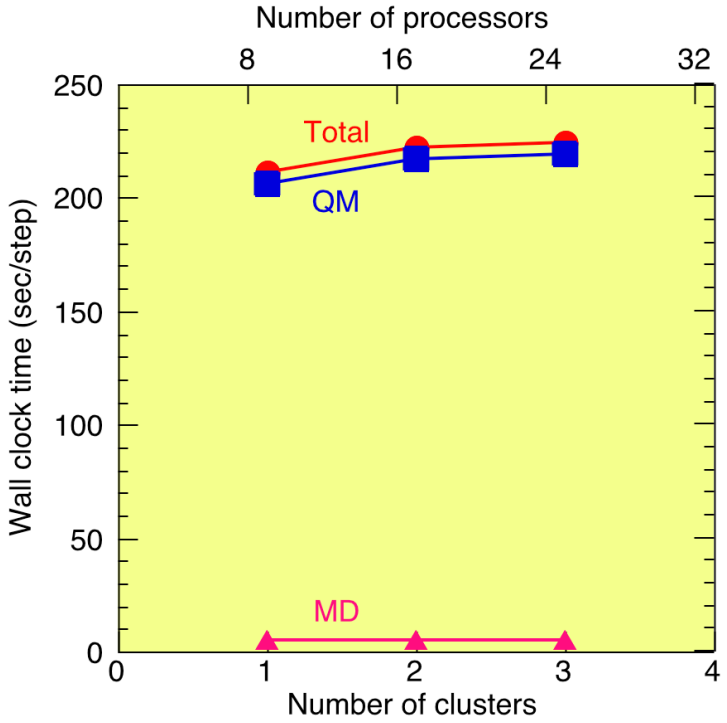

(b) Modified

# Global Collaborative Simulation

## Multiscale divide-&-conquer MD/QM simulation on a Grid of distributed PC clusters in the US & Japan

- **Task decomposition (MPI Communicator) + spatial decomposition**
- **MPICH-G2** (www.niu.edu/mpi)**/Globus** (www.globus.org)



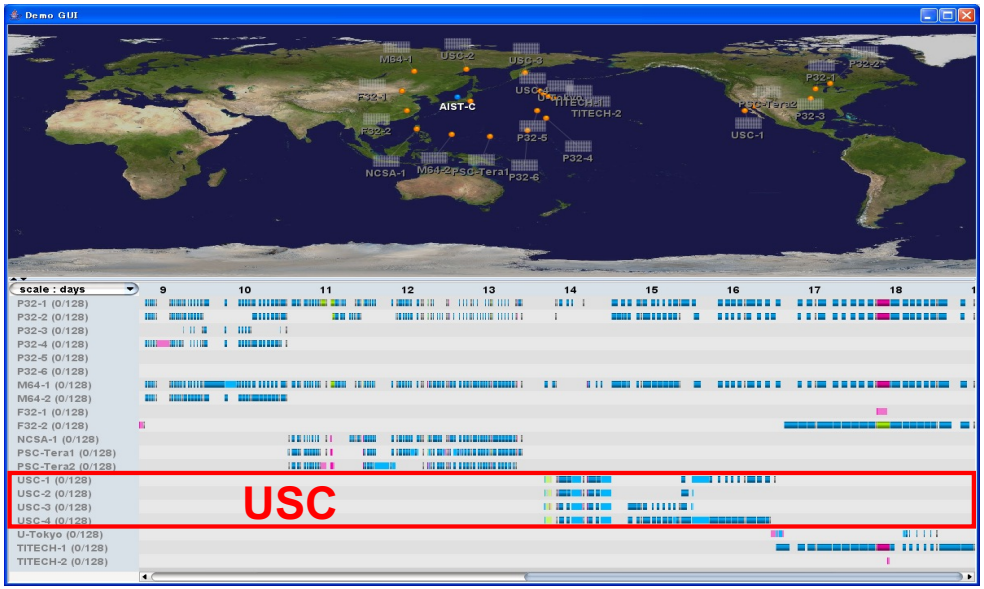




**Japan:** Yamaguchi—65 P4 2.0GHz
Hiroshima, Okayama, Niigata—3×24 P4 1.8GHz
**US:** Louisiana—17 Athlon XP 1900+

**MD — 91,256 atoms**
**QM (DFT) — 76*n* atoms on *n* clusters**

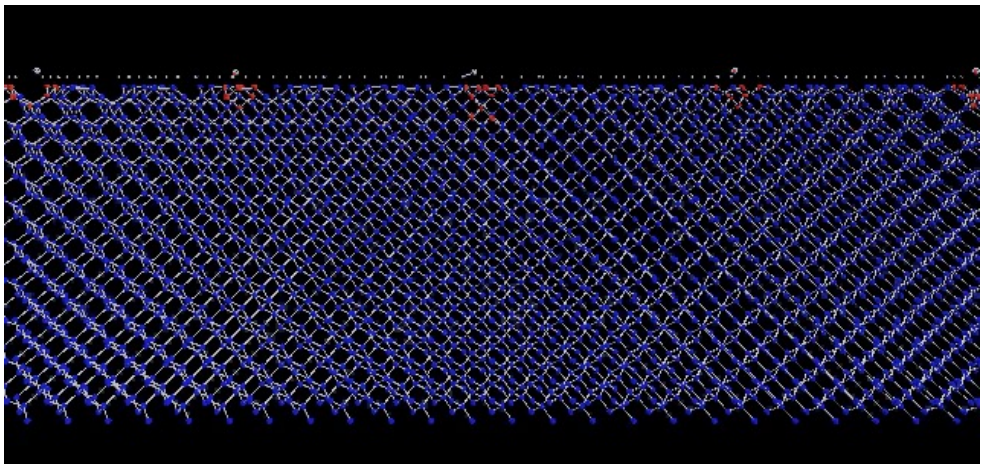
- **Scaled speedup, $P = 1$ (for MD) + $8n$ (for QM)**
- **Efficiency = 94.0% on 25 processors over 3 PC clusters**

# Global Grid QM/MD

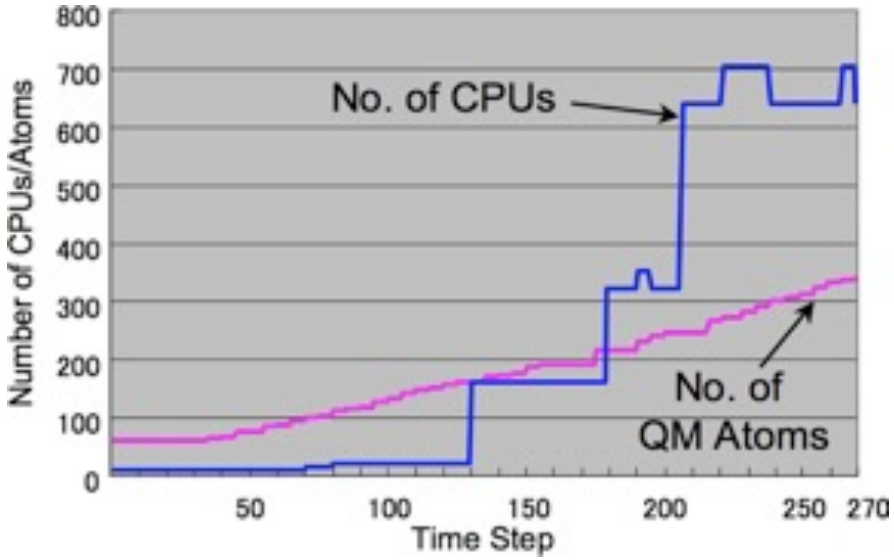



- **Sustained (153,600 cpu-hrs) Grid supercomputing at 6 centers in the US (USC, NCSA, PSC) & Japan (AIST, U Tokyo, TITech)**
- **Dynamic allocation of computing resources on demand & automated migration due to reservation schedule & faults**
- **Hybrid GridRPC** (ninf.apgrid.org) **+ MPI** (www.mcs.anl.gov/mpi) **Grid computing**

**High-energy beam oxidation of Si**



H. Takemiya *et al.*, *Proc. IEEE/ACM SC06*



# Fast TCP

BBC NEWS WORLD EDITION

Last Updated: Thursday, 5 June, 2003, 10:48 GMT 11:48 UK

E-mail this to a friend  Printable version

## Promise of ultra-fast downloads

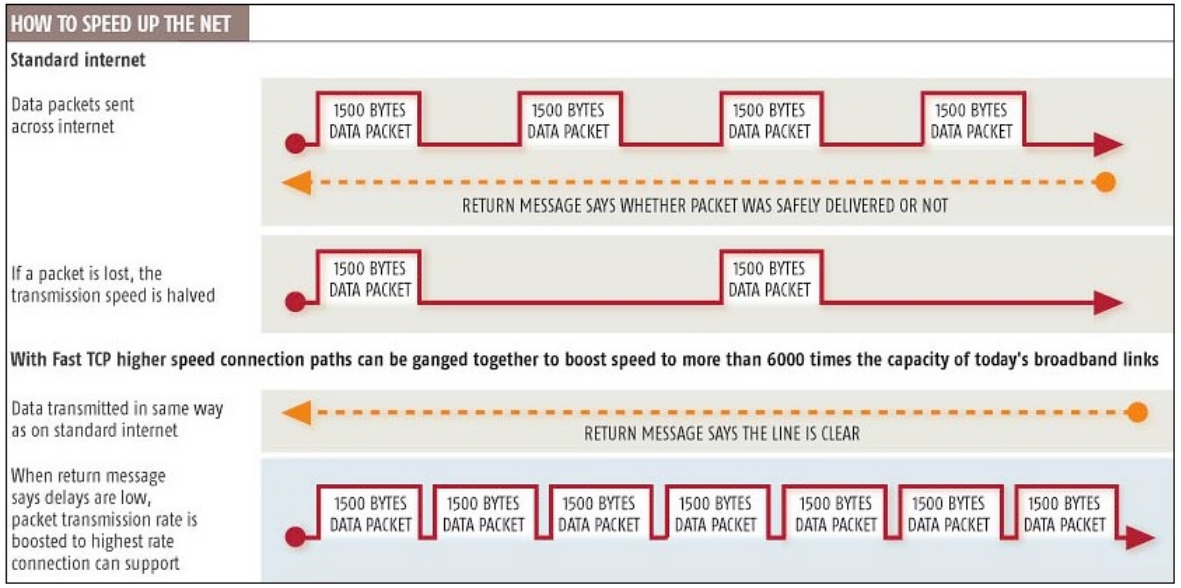
**Soon you could be downloading an entire movie off the net far faster than you do now.**

Fast net tech could soon take off

US researchers are working on ways to improve the way that net protocols decide how quickly data travels around the net.

Early tests of the new system show that it can triple data transmission speeds.

By linking lots of the faster systems together the researchers have produced data transfer speeds many times higher than is possible today.




## Packet tracking promises ultrafast internet

10:54 05 June 03

Exclusive from New Scientist Print Edition. Subscribe and get 4 free issues.

Imagine an internet connection so fast it will let you download a whole movie in just five seconds, or access TV-quality video servers in real time. That is the promise from a team at the California Institute of Technology in Pasadena, who have developed a system called Fast TCP.

**Fast TCP: Achieved 8.6 Gb/s between Sunnyvale, CA & CERN, Switzerland**

**Steven Low (Caltech)**

http://netlab.caltech.edu/FAST