Solvers for $O(N)$ Electronic Structure in the Strong Scaling Limit with Charm++

Nicolas Bock

Matt Challacombe

Theoretical Division, Los Alamos National Laboratory

Laxmikant V. Kalé

University of Illinois at Urbana-Champaign

"Linear-Complexity Dense Linear Algebra, Parallelization and Applications"

SIAM CS&E 2015, Salt Lake

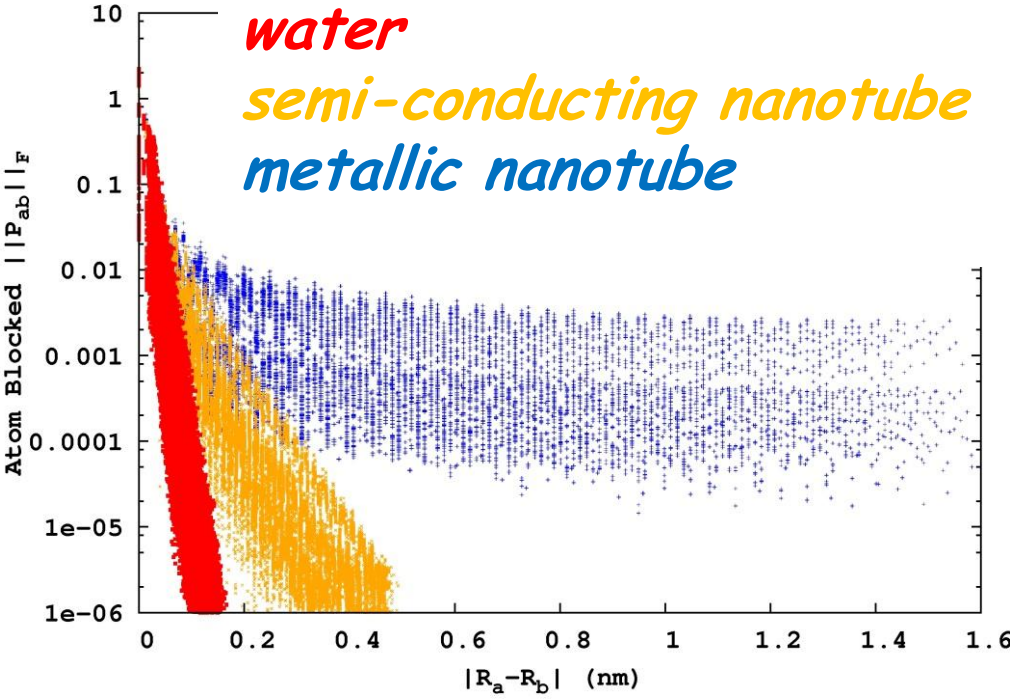
LA-UR-10-07458//LA-UR 11-06091//LA-UR-14-22050//LA-UR-14-20354//

Sponsored by: **DOE LDRD-ER grant 20110230ER** & TenBar Café

Quantum Locality & Kohn's Nearsighted Principle

In a local, atom centered representation, quantum mechanical matrices possess decay properties. For non-metallic systems, matrix elements decay exponentially with atom-atom separation:

$P_{ab} \sim \exp(-\beta|R_a - R_b|)$

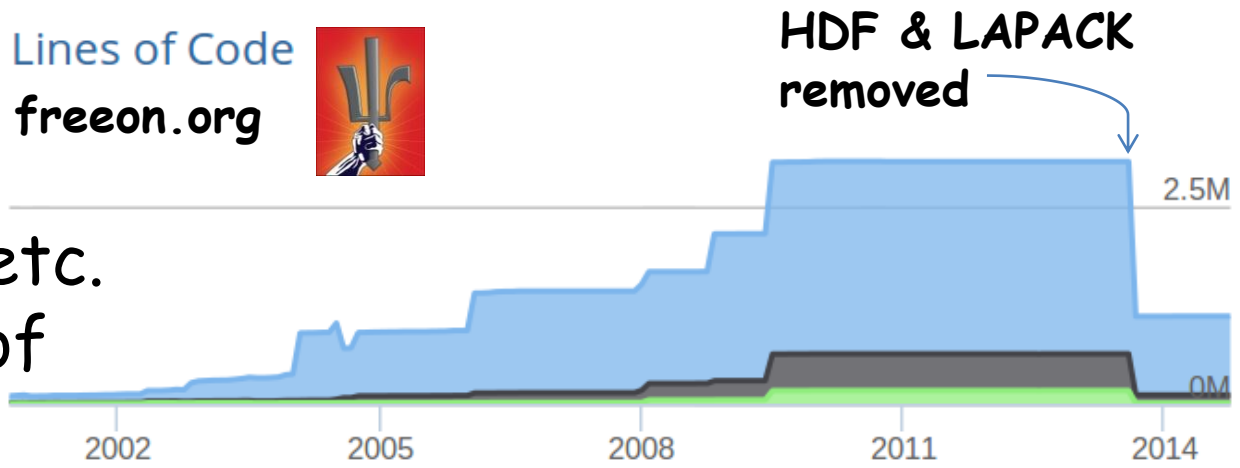


The rate of matrix decay is related to metric and gap ill-conditioning, with spatial extent increasing toward the metalic state (the Mott transition).

There are many ways to exploit this decay. The most conventional approach involves matrix truncation (sparsification). Only good for very rapid decay.

O(N) Codes for HF/DFT are Complicated

Five solvers for basic HF/DFT. More complicated still for gradients, response & etc. Easily runs to millions of lines.

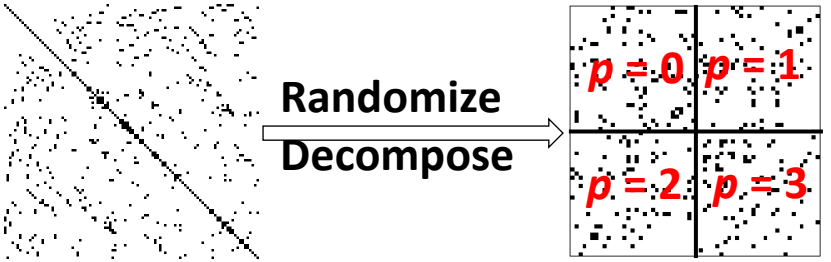


Complexity in Solver Collectives:

- Entangled & incompatible data structures
- Mixed programming models
- Layers of esoteric approximations
- Disparate rates of error accumulation
- **Optimization of individual solvers inhibits evolution of the collective**

Case Study: Parallel Sparse Matrix-Matrix Multiply

Most optimal parallel SpMM is based on Bulloc's method for work homogenization: DBCSR data structure, Hutter & co 2014.

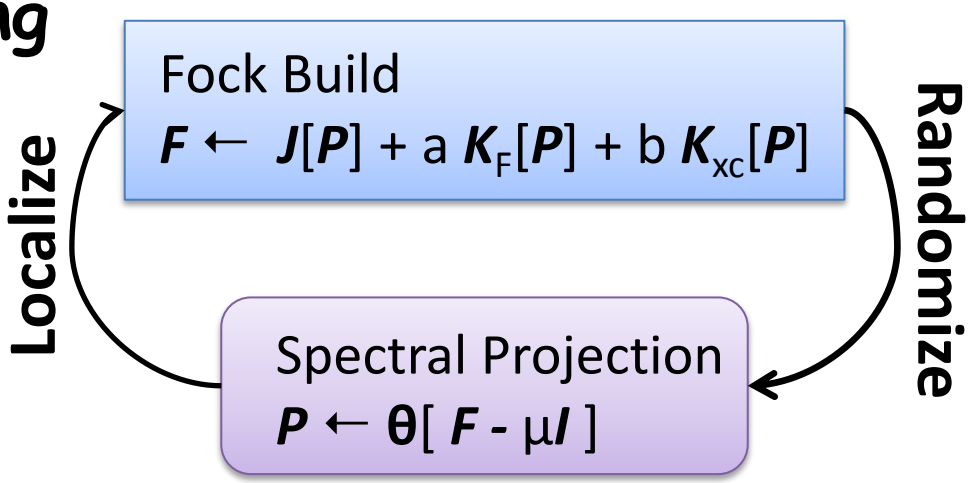


- Randomization leads to *cost homogenization*
- **Randomization ⇨ delocalization & data redistribution**

Cannon's algorithm for distribution throttled as $O(\sqrt{p})$

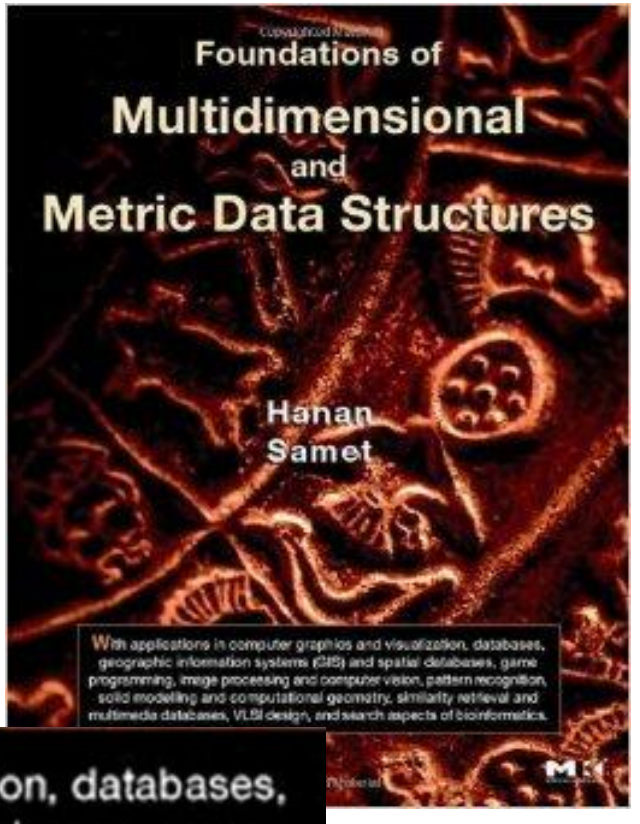
- **Cannot access strong scaling regime, *p* >> *N***

Solvers entangled with DBCSR require locality. Massive data redistribution on each SCF cycle ☹



An N-body Framework for all Solvers

- ***Potential for massive simplification with shared frameworks (stacks) supporting solvers collective***
- Kernel compression & optimal data operations
  - Multilevel approximation
  - Range, metric and overlap queries
- ***Demonstrated for all HF/DFT solvers***
- Communication optimality
- Runtimes for recursive task parallelism
- Overlapping widely w/other disciplines



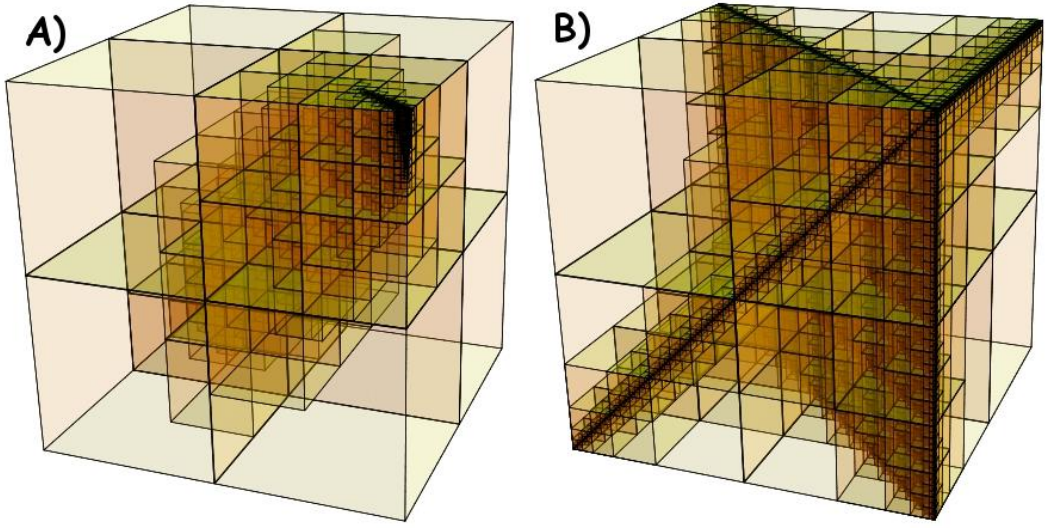
With applications in computer graphics and visualization, databases, geographic information systems (GIS) and spatial databases, game programming, image processing and computer vision, pattern recognition, solid modelling and computational geometry, similarity retrieval and multimedia databases, VLSI design, and search aspects of bioinformatics.

Matrix Multiplication as N-Body Solver

Challacombe & Bock, arXiv:1011.3534

SpAMM is a fast solver for multiplication of matrices with ***decay & locality.*** Employs matrix quadtrees, rigorous sub-multiplicative norms, recursive occlusion and culling of terms in the product space. ***O(N) for dense decay matrices.***

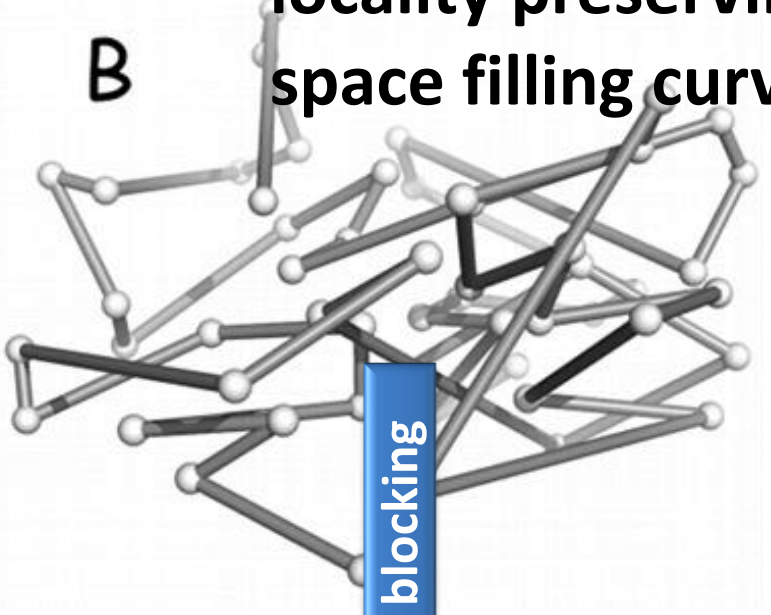
$$A^k \otimes B^k = \begin{cases} 0 & \text{if } \left\|A^k\right\| \left\|B^k\right\| < \tau, \\ A^k \cdot B^k & \text{elseif } k = k_{\max}, \\ \begin{pmatrix} A_{11}^{k+1} \otimes B_{11}^{k+1} + A_{12}^{k+1} \otimes B_{21}^{k+1} & A_{11}^{k+1} \otimes B_{12}^{k+1} + A_{12}^{k+1} \otimes B_{22}^{k+1} \\ A_{21}^{k+1} \otimes B_{11}^{k+1} + A_{22}^{k+1} \otimes B_{21}^{k+1} & A_{21}^{k+1} \otimes B_{21}^{k+1} + A_{22}^{k+1} \otimes B_{22}^{k+1} \end{pmatrix} & \text{else.} \end{cases}$$



Occlusion and culling in the recursive product space (idealized) for matrices with
A) exponential and
B) algebraic decay

locality preserving
space filling curves

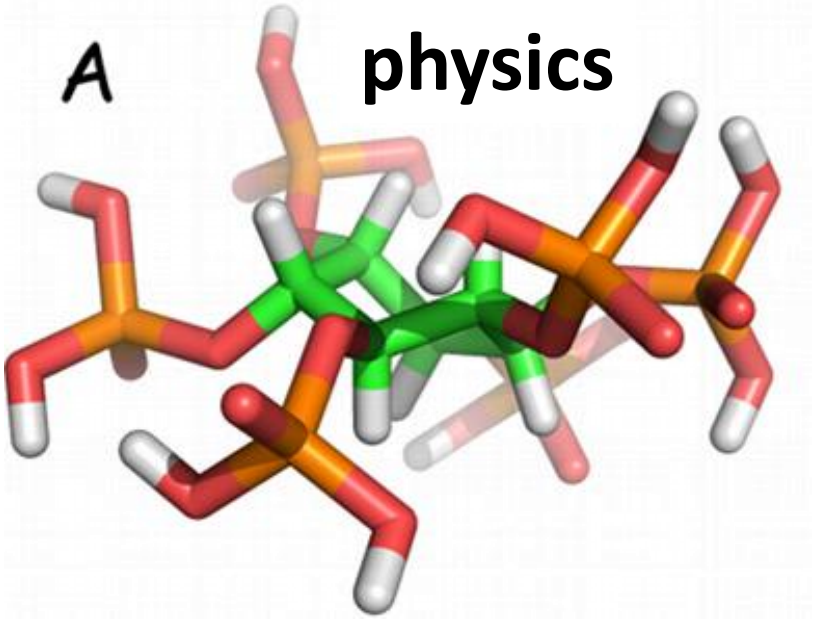
B



blocking

A

physics

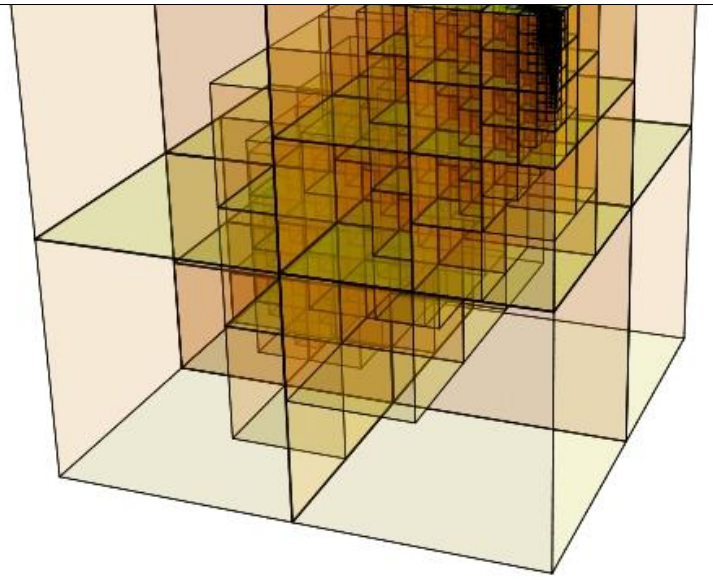


locality

matrix as
database
(quadtree)

convolution

SpAMM: *O*(*N*), fine-grained
recursive task parallelism

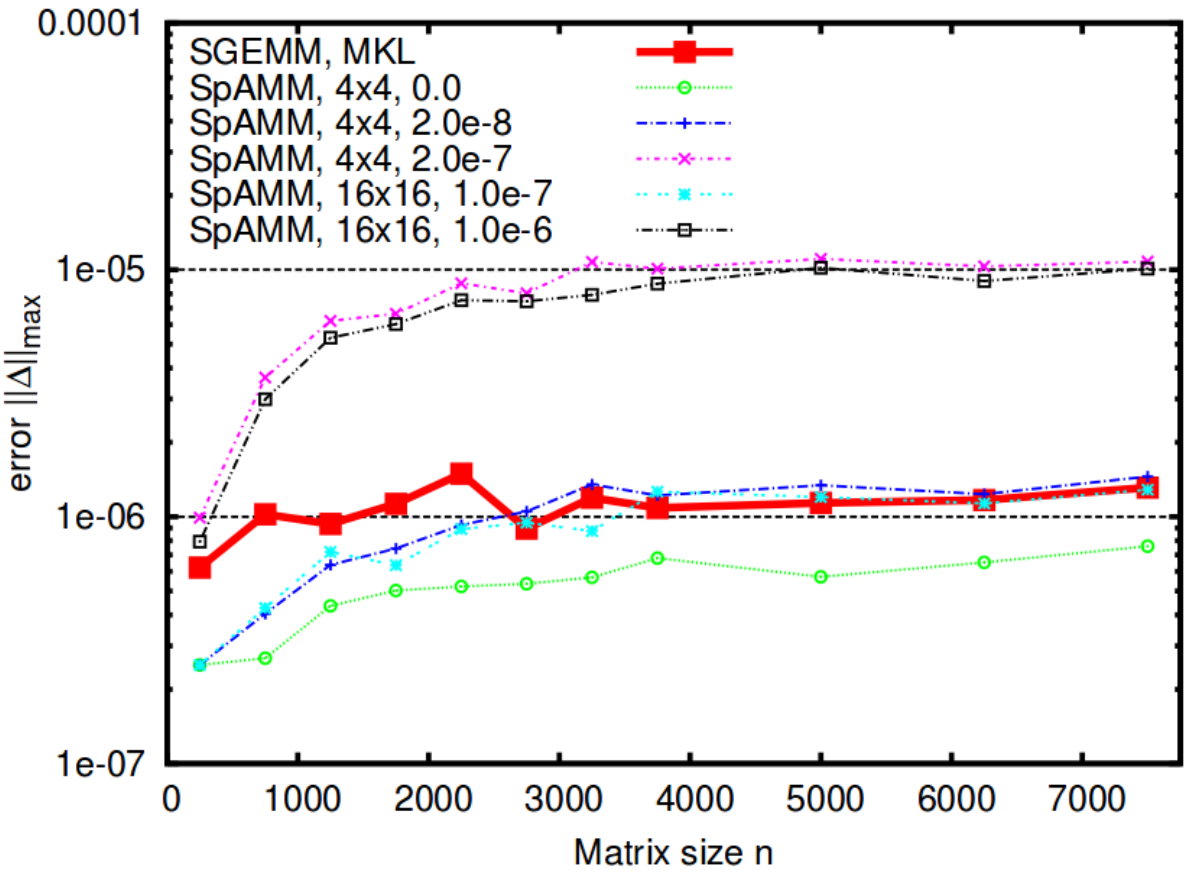


SpAMM For Dense Decay Matrices

Bock & Challacombe, SIAM J. Sci. Comput., 35(1), C72

Product matrix is asymptotically sparse, but only much, much later

Fill in of small blocks at negligible cost yields $O(N)$ scaling even for dense matrices



In single, SpAMM beats MKL SGEMM in error

Recursion w/locality more accurate than row-column

What About an Optimized SpAMM?

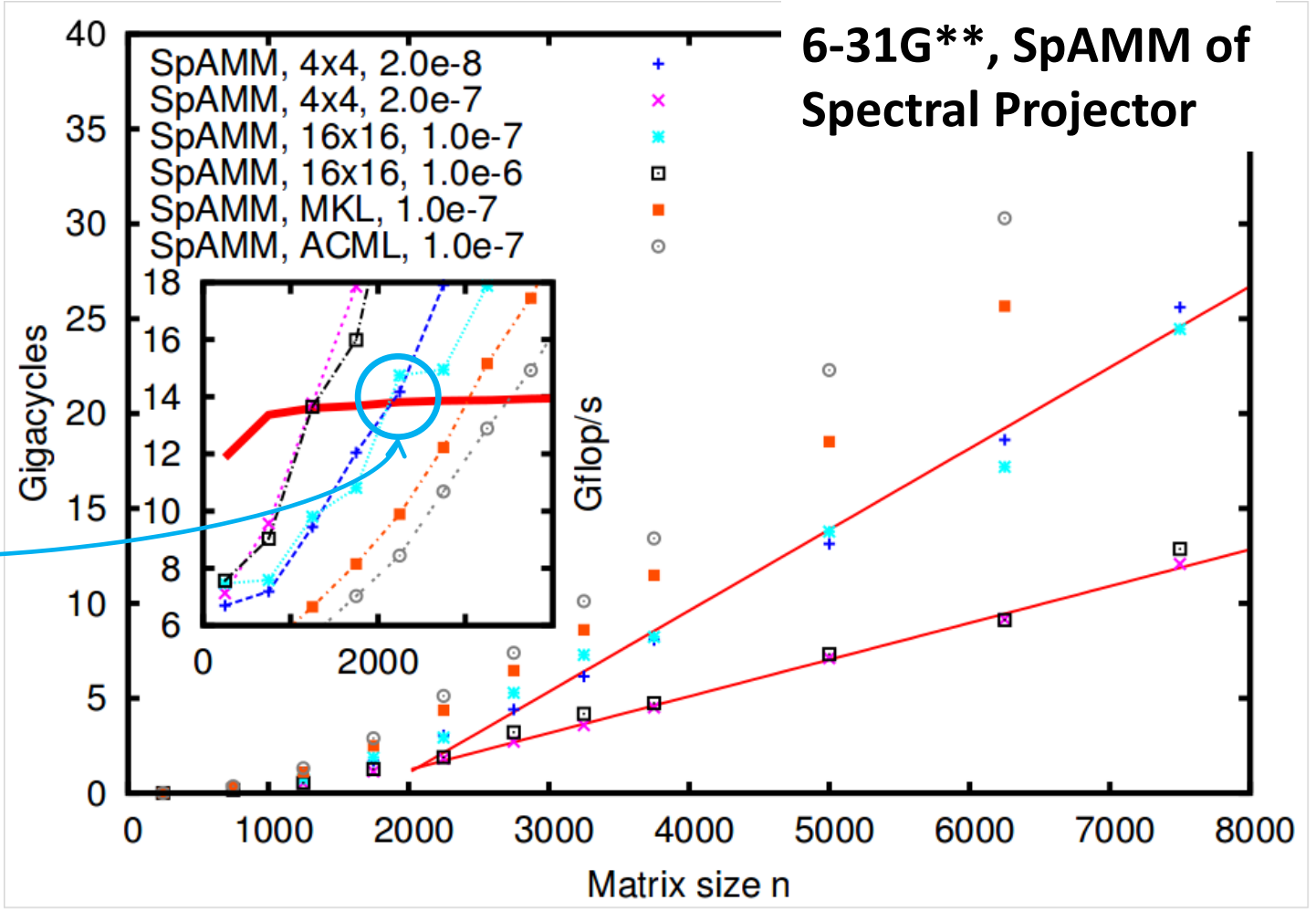
Bock & Challacombe, SIAM J. Sci. Comput., 35(1), C72

Assembly coded SpAMM in single. Don't try this at home…
Informs use of pragmas, OpenMP 4.0 directives & *etc.*

50% of peak with 4x4 blocking

Crossover with MKL/SGEMM at N=2000, same error

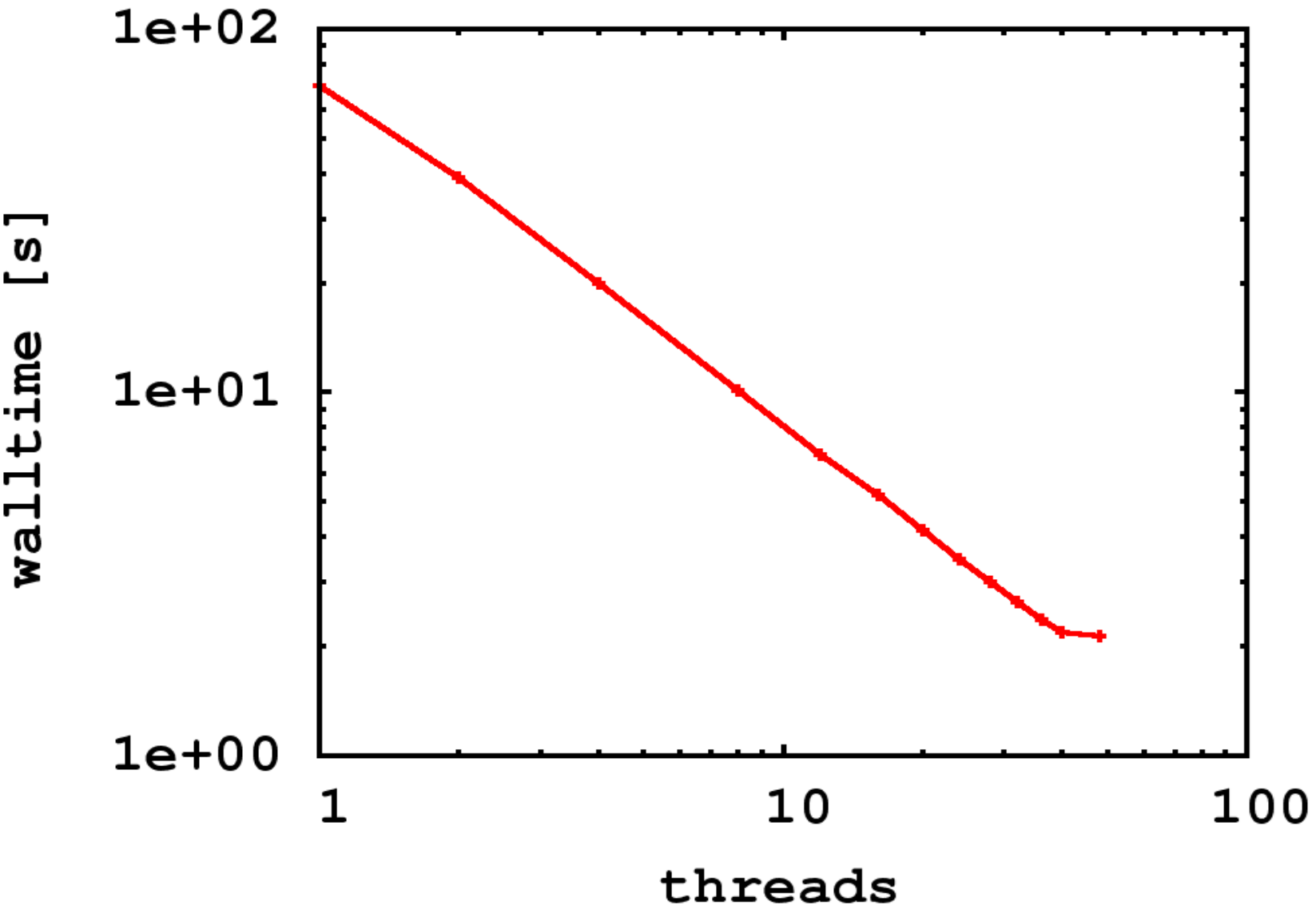
Can we do as well with directives (openmp 4)?



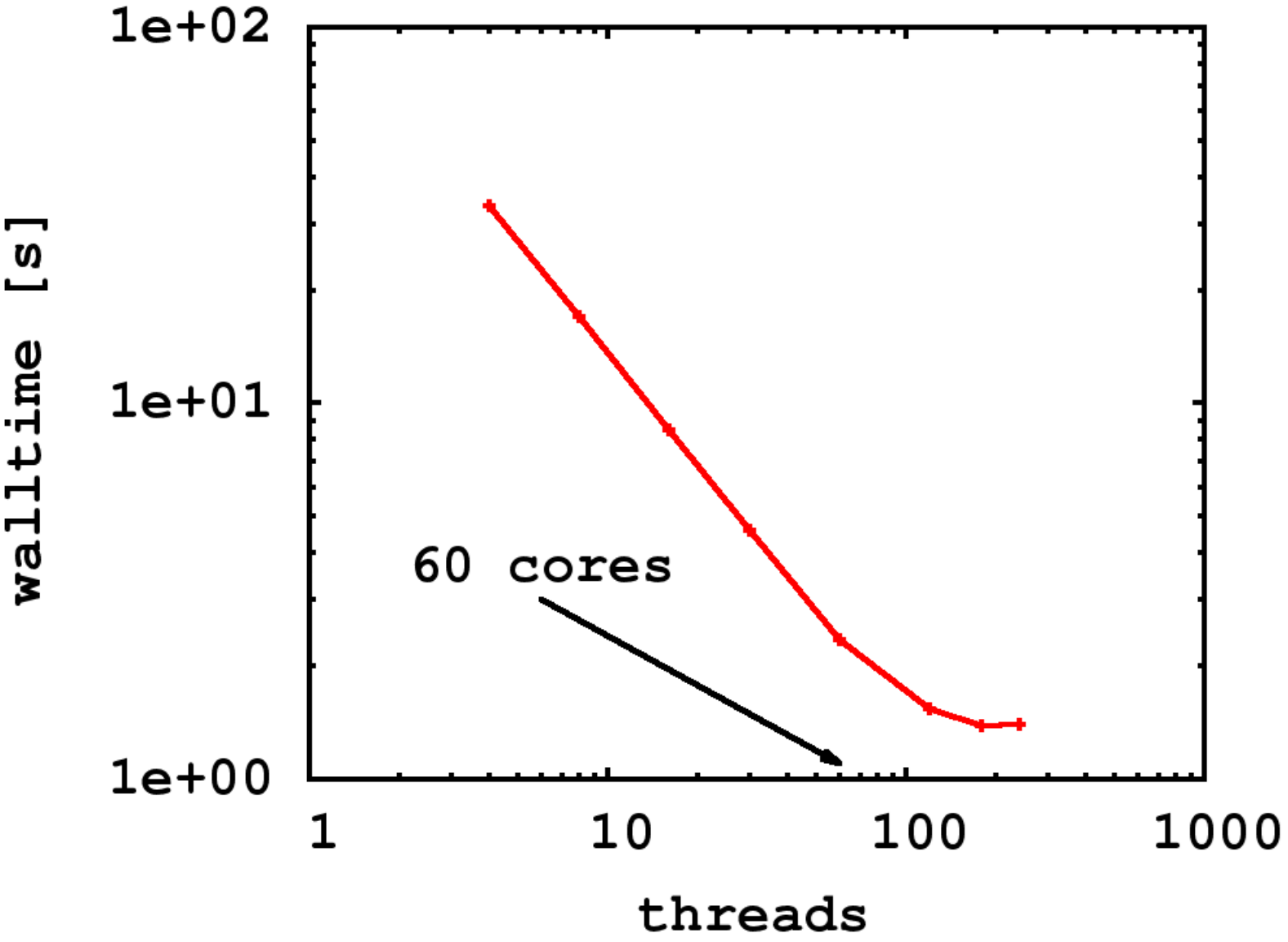
SpAMM - OpenMP 3.0

```
1: function MULTIPLY(τ, tier, A, B, C)
2:     if tier < depth then
3:         for all {i, j, k | C_ij ← A_ik B_kj} do
4:             if ‖A_ik‖ ‖B_kj‖ > τ then                    ▷ Eq. 1
5:                 Create untied OpenMP task
6:                 MULTIPLY(τ, tier+1, A_ik, B_kj, C_ij)
7:             end if
8:         end for
9:         OpenMP taskwait
10:    else
11:        Acquire OpenMP lock on C
12:        C ← C + A × B          ▷ Dense product, e.g. BLAS
13:        Release OpenMP lock on C
14:    end if
15: end function
```

SpAMM – Parallel Efficiency on Magny Cours



SpAMM - Parallel Efficiency on Xeon Phi



Weak Scaling and the Parallel SpMM

Bowler *et. al*, arXiv:1402.6828 ← very entertaining comment …

So far, no code has demonstrated a parallel SpMM beyond ~ 4 atoms/core ($p < N/4$). While enabling bigger systems, does not enable high throughput (timesteps!)

Best results to date involve work homogenization, precluding $p \gg N$

Argue that no SpMM employing matrix decomposition can be *O* (*N*) & access strong scaling limit

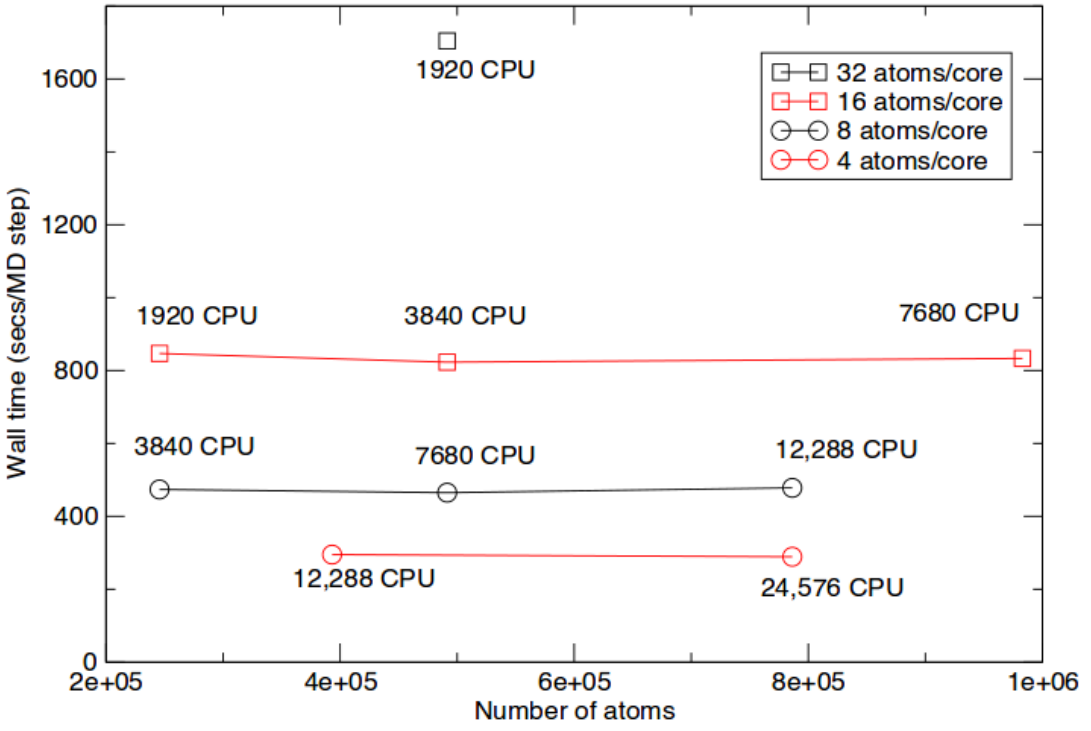


FIG. 1: Weak scaling for CONQUEST on the K computer, showing scaling to around 200,000 cores (8 cores per CPU).

SpAMM - Charm++

- Quadtree linked list → 2D chare array per tier
- Recursive multiply → 3D chare array per tier
- GreedyComm LB after each multiply

```
1: function MULTIPLY(τ, A, B, C)
2:     for t ≥ 0 ∧ t < d do
3:         convolution[t].prune()
4:     end for
5:     convolution[d].multiply()
6:     convolution[d].store()
7: end function
```

SpAMM in the Strong Scaling Limit

Bock & Challacombe, SIAM J. Sci. Comput, Accepted w/ revs (2015)

- ✓ *n*-body is communication optimal (recent Yellik & Demmel) & task parallelism supported by openmp3 & charmm++
- ✓ Quantum locality → data locality via SFC heuristics
- ✓ **Decomposition in 3-D task space, not 2-D data space**
- ✓ Persistence load balancing and mixed openmp3/charm++ for iteration functional approximation. In this case, spectral projection of 6-31G** density matrices (sign/step)
- ✓ Harder to scale with fast decay (bulk water has fast decay)

Persistence Load Balancing in SpAMM Spectral Projection (Sign/Step Function): Strong Scaling with Strong Decay ($H2O_n$)