Algorithmic Challenges in Multiscale Quantum Simulation of Strongly-Correlated Materials

supported by DOE ASCR and NNSA

Ichitaro Yamazaki
Department of Computer Science
University of California at Davis

CScADS Workshops 2007
Libraries and Algorithms for Petascale Applications

Next Generation Multi-Scale Quantum Simulation Software
for Strongly Correlated Materials
http://scicompforge.org/petamat

- SciDAC senior personnel
  - University of Cincinnati
    - Mark S. Jarrell, Physics (PI)
    - Karen Tomko, Electrical and Computer Engineering
  - Oak Ridge National Laboratory
    - Thomas A. Maier, Physics (co-PI)
    - Eduardo F. D'Azevedo, Computer Science and Mathematics
  - University of California at Davis
    - Zhaojun Bai, Computer Science (co-PI)
    - Sergey Y. Savrasov, Physics
    - Richard T. Scalettar, Physics

Modeling Materials at Petascal

- Study of Strongly-Correlated Materials;
  - Great technological promises.
    - most magnets and superconductors fall into this category.
  - Poorly understood.
    - properties of the materials are determined by complex interaction of electrons

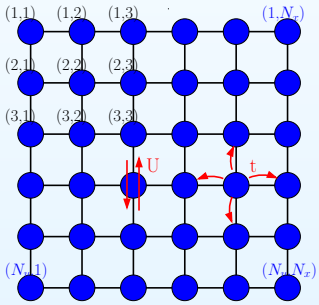
      long-ranged spin and charge correlations, competing ground states, complex phase diagrams, etc.
- Goals of the project;
  - Numeric: develop mathematical and numerical understanding of the model.
  - Algorithmic: develop massively parallel multi-scale method with a simple user interface.
  - Scientific: better understand their properties to improve and create new materials.
- Multiscale modeling
  - Short-length scales: $10 - 100$ electrons with QMC methods.
  - Intermediate-length scales: $100 - 1,000$ electrons using vertices from QMC.
  - Long-length scales: $1,000-$ electrons with mean field approximation.

Hubbard model

- Hubbard Hamilotian captures the exact correlation of neighboring electrons,

$$\mathcal{H} = \mathcal{H}_U + \mathcal{H}_t + \mathcal{H}_\mu,$$

  - potential energy U, kinetic energy t, and chemical energy μ of electrons.
- Study of the electron interactions on 2D lattice:



  - Number of lattice sites $N = N_x \times N_y$
    - N_x sites in x direction and N_y sites in y direction.
    - one electron per site on average, i.e., $\mu = 0$.
  - Discretize inverse temperature β into L intervals.

Computational kernel

Computational kernel of QMC simulation solves

$$Ax = b,$$

where $A = M^T M$ is SPD and

$$M = \begin{pmatrix} I & & & B_1 \\ -B_2 & I & & \\ & \ddots & \ddots & \\ & & -B_L & I \end{pmatrix}.$$

- Each $B_l \in \mathcal{R}^{N \times N}$ is defined as

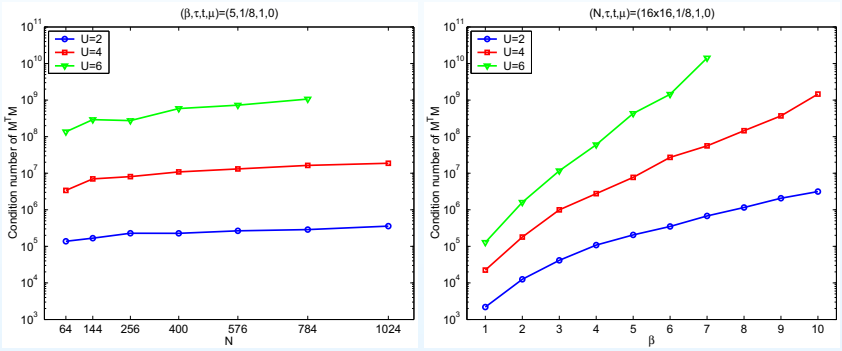
$$B_l = e^{t \triangle \tau K} e^{\nu V_l}$$

where

- $K \in \mathcal{R}^{N \times N}$ describes the lattice structure and stays same through simulation,
- $V_l = \text{diag}(h_{l,1}, h_{l,2}, \ldots, h_{l,N})$ is random config. and changes for each solution,
- $\triangle \tau = \beta / L$ is a discritization parameter, and ν is defined as $\cosh \nu = e^{\frac{U \triangle \tau}{2}}$.

Computational bottleneck

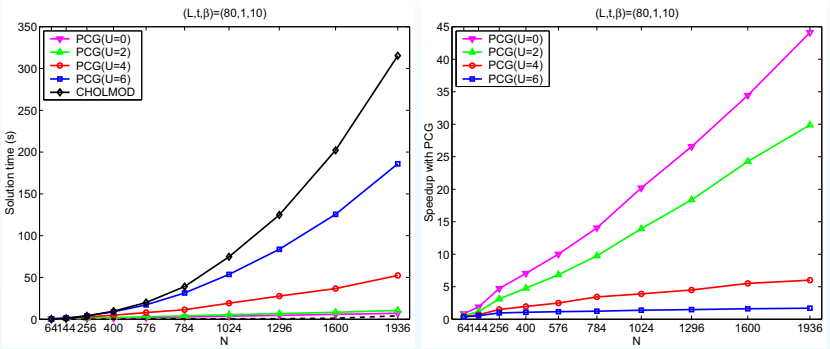
- Large and ill-conditioned system;
  - $N = \mathcal{O}(10^3)$, $L = \mathcal{O}(10^2)$, and $NL = \mathcal{O}(10^5)$.
  - Conditioning of the matrix changes with the parameters.



- The linear solver accounts for 80% - 90% of the total simulation time.
  - a large number of solutions are required for each simulation with different V_l.
  - A is large but sparse and structured:
    matrix-vector multiply in matrix-free form with $\mathcal{O}(NL)$-computation.
  - high-accuracy is not required for each solution.

Linear solver

- Comparison of PCG with Tim Davis' CHOLMOD in UMFPACK.



- Specialized direct solver with block-cyclic reduction;
  - dense $N \times N$ matrix operations;
    - matrix-matrix multiply and QR factorization.
  - $\mathcal{O}(N^3)$-computational complexity.
  - $\mathcal{O}(N^2 L)$-storage requirement.
- Toward "optimal" $\mathcal{O}(N)$-complexity.
  - My research focuses on preconditioning techniques.

New preconditioning technique

The preconditioner R is constructed based on the factorization of the form

$$A + \alpha \cdot D_A = RR^T + E,$$

where R is lower-triangular, and E is a symmetric error matrix. By combining

1. the static diagonal shifting of A with a scalar α, and
2. the dynamic updating of $E = RF^T + FR^T - S - S^T$;
   - F and S are strictly-lower triangular,
   - two-levels of drop tolerance σ_1 and σ_2 such that $\|F\| \leq \sigma_1$, $\|S\| \leq \sigma_2$, and

$$\sigma_1 \geq \sigma_2 = \alpha,$$

we can control the magnitude of entries in the residual matrix

$$I - R^{-T} A R^{-1} = \underbrace{F^T R^{-1} + R^{-1} F}_{\#1} - \underbrace{R^{-1}(S + \alpha D_A + S^T) R^{-T}}_{\#2},$$

where small enough σ_2 to make $\#1$ dominant results in an efficient yet high-quality R.

- Related works: [Manteuffel'80] [Kaporin'98]

New sparse data structure

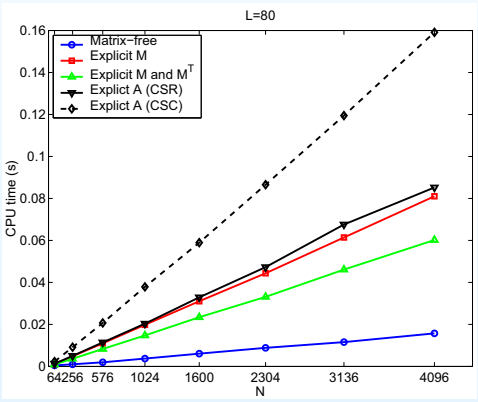
- Extended Compressed Sparse Column (ECSC) format;

$$\underbrace{valR,\ rowR,\ ptrR,}_{\text{CSC}}\quad\underbrace{linkR,\ headR,}_{\text{Row Access}}\quad\underbrace{nextF,}_{\text{Access } F}$$

  - avoids the storage of and operations with zero elements.
  - accommodates all the underlying access pattern.
  - never updates the data structure once non-zeros are stored.
  - increases the storage requirement by 70% from CSC format.
    - both R and F are sparse enough.
  - exploits memory hierarchy, but only for column-access.
- Note: for each solution, sparsity pattern is same for A but not for R.

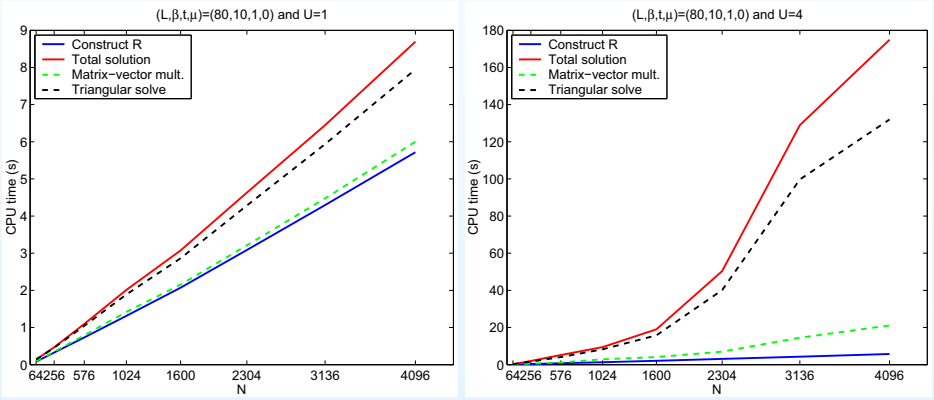
Sparse matrix-vector multiply

- Optimizing matrix-vector multiply with explicitly-stored matrix
  - same sparsity pattern of A for all solutions, but
  - the matrix-vector multiply is more efficient in matrix-free form for PCG iterations.



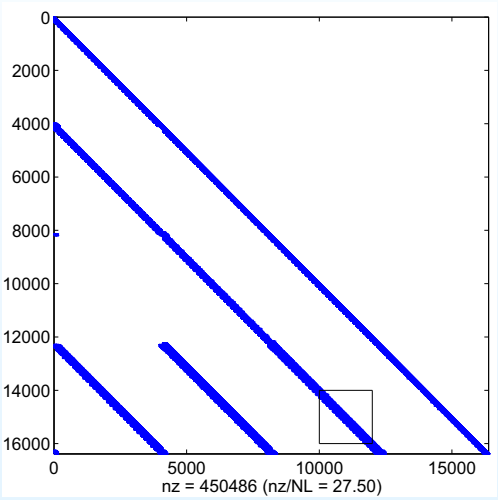
PCG bottleneck

- Matrix-vector product can be computed efficiently in a matrix-free form.
- Sparse triangular solve is the bottleneck.
  - R is stored in MSC format and applied with register-level tuning.



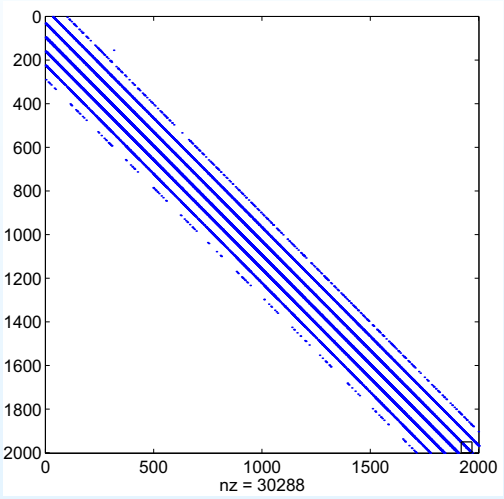
Sparse triangular solve

- Sparsity pattern of R, i.e., large and sparse blocks.
  - R with $(N, L) = (64 \times 64, 4)$.



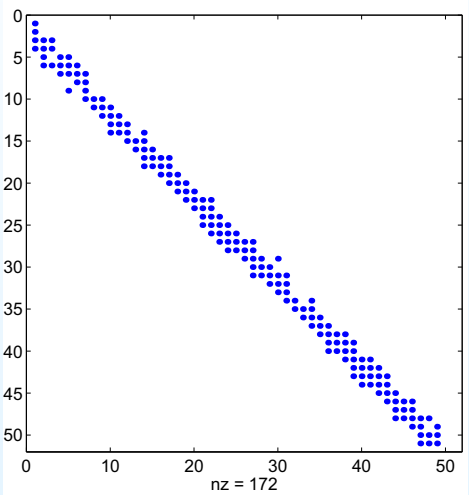
Sparse triangular solve

- Sparsity pattern of R, i.e., large and sparse blocks.
  - $R(14000:16000, 10000:12000)$ with $(N, L) = (64 \times 64, 4)$.



Sparse triangular solve

- Sparsity pattern of R, i.e., large and sparse blocks.
  - $R(15950:16000, 11920:11970)$ with $(N, L) = (64 \times 64, 4)$.



- Sparse optimization techniques like OSKI by BeBOP did not work.