# Large-Scale Implementation of the Density Matrix Renormalization Group Algorithm:

## Performance Analysis on the Intel® Broadwell and Knights Landing Architectures

James VANCE*, Marcello DALMONTE, Ivan GIROTTO
*The Abdus Salam International Centre for Theoretical Physics*
*Strada Costiera 11, 34151 Trieste, Italy*
*jvance@ictp.it

5 March 2018

# Introduction

# Two-dimensional many-body quantum systems

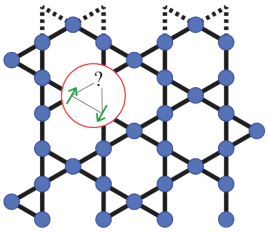



**Figure 1:** Spin-1/2 Kagome Heisenberg antiferromagnet, Yan et al Science 2011 [1]

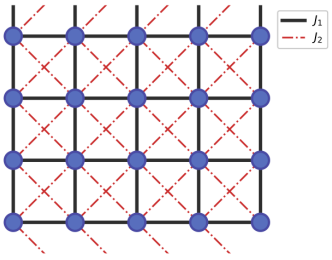



**Figure 2:** $J_1$-$J_2$ XY model on a square lattice

- Crucial in understanding high-$T_c$ superconductivity, frustrated magnetism, and topological phases of matter [2–5]
- Intractable with *exact diagonalization* (large number of states) [6] and *quantum Monte Carlo* (negative sign problem) [7]

# Density Matrix Renormalization Group (DMRG)

- Variational method originally used to study the ground state properties of many-body quantum systems in 1D [8]
- Systematic truncation of Hilbert space, keeping only *m* basis states with minimal loss of information
- Well-established method used in condensed matter [9] and quantum chemistry [10]

# Towards 2D Systems

- DMRG has been proven effective for 2D but requires more states to be kept after truncation [11–13]

- Results in large sparse matrices and more intensive operations requiring a scalable parallel solution
  $\rightarrow$ **HPC approach**

# Objectives

- Implement DMRG calculations for distributed-memory architectures
- Enable developed code to run efficiently on a Tier-0 world-class HPC infrastructure
- Perform large-scale DMRG calculations on 1D and 2D frustrated magnets
- Shed light on some paradigmatic models of frustrated magnetism that are relevant to experiments (e.g. cold gases in optical lattices and superconducting circuits)

# Implementation

# Infinite-size DMRG algorithm

**Block notation**: $\mathcal{B}(n, D)$

block of $n$ sites with $D$ states [14]

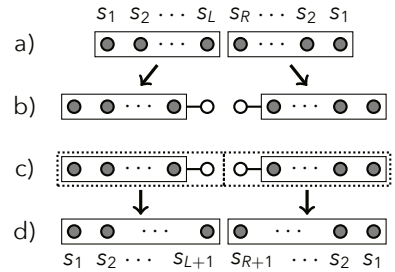



**Figure 3:** Infinite-size algorithm [9]

**Local dimension**: $d$

e.g. for spin-1/2 systems $d = 2$

a) **Initial blocks**

b) **Add one site to each:**

$\mathcal{B}(L, m) \rightarrow \mathcal{B}(L + 1, dm)$

$\mathcal{B}(R, m) \rightarrow \mathcal{B}(R + 1, dm)$

c) **Build and diagonalize the Superblock:***

$\mathcal{B}(L + R + 2, d^2m^2)$

d) **Truncate Basis**†

$\mathcal{B}(L + 1, dm) \rightarrow \mathcal{B}(L + 1, m)$

$\mathcal{B}(R + 1, dm) \rightarrow \mathcal{B}(R + 1, m)$

*Computationally intensive operations: **Construction** and **diagonalization** of the Hamiltonian

†**Full eigendecomposition** of small density matrices and **rotation** of block operators

# Libraries for Parallel Linear Algebra

**Portable, Extensible Toolkit for Scientific Computation** (PETSc)

- data structures and routines for scalable scientific applications
- overlap in computation and communication during sparse matrix-vector multiplications (spMVM)

**Scalable Library for Eigenvalue Problem Computations** (SLEPc)

- eigenvalue decomposition of large sparse matrices [15]
- **Eigenvalue Problem Solver** (EPS) - iterative diagonalization algorithms that use efficient spMVM from PETSc

# The Kronecker Product

Construction of the enlarged blocks and superblock involve linear combinations of Kronecker products of matrices

$$\mathbf{C} = \sum_n a_n \mathbf{A}_n \otimes \mathbf{B}_n \tag{1}$$

We provided our own implementation **MatKronProdSum**

- **Submatrix-collection** for non-local rows
- **Preallocation** of matrix memory
- **Parallel local generation** of resultant matrix elements
- **Index slicing** for selection of states

# Performance Analysis

# System Architecture

| Partition | **Marconi A1 (BDW)** | **Marconi A2 (KNL)** |
|---|---|---|
| Processors | 2 × 18-core Intel Xeon E5-2697 v4 (Broadwell) at 2.3 GHz | 68-core Intel Xeon Phi 7250 CPU (Knights Landing) at 1.40 GHz |
| Cores | 36 cores/node | 68 cores/node |
| Extensions | AVX 2.0 | AVX-512 |
| RAM | 128 GB/node | 16 GB/node MCDRAM<br>96 GB/node DDR4 |
| Max Memory Bandwidth | 76.8 GB/s‡ | MCDRAM: 475-490 GB/s<br>DDR4: 90 GB/s § |
| Network | Intel Omnipath, 100 Gb/s | |

‡Intel specifications [16]

§Streams benchmark [17] in FLAT mode; Marconi A2 operates in CACHE mode

# Implementation Details

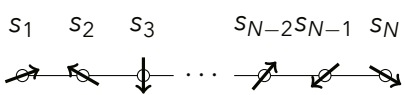
## PETSc 3.7 and SLEPc 3.7

- BDW: Pre-compiled binaries on Marconi
- KNL: `--xOPTFLAGS="-g -xMIC-AVX512 -O3 -mP2OPT_hpo_vec_remainder=F" --with-memalign=64`

## EPS Solver

- Type: Krylov-Schur (Thick-Restart Lanczos)
- Relative Tolerance: $10^{-12}$

## Test Case: 1D Heisenberg XXZ Hamiltonian




$$\mathbf{H} = \sum_{i=1}^{N-1} \left( J_z \mathbf{S}_i^z \mathbf{S}_{i+1}^z + \frac{J}{2} \left[ \mathbf{S}_i^+ \mathbf{S}_{i+1}^- + \mathbf{S}_i^- \mathbf{S}_{i+1}^+ \right] \right)$$

# Quantum Number Conservation

- Implemented conservation of total magnetization $S^z_{\text{tot}} \equiv S^z_L + S^z_{\sigma^L} + S^z_{\sigma^R} + S^z_R = \text{const}$
- Superblock Hamiltonian $H_{L \circ\circ R}$ **matrix size is reduced**
- Reduced density matrices $\rho_{L\circ}$ become **block-diagonal**

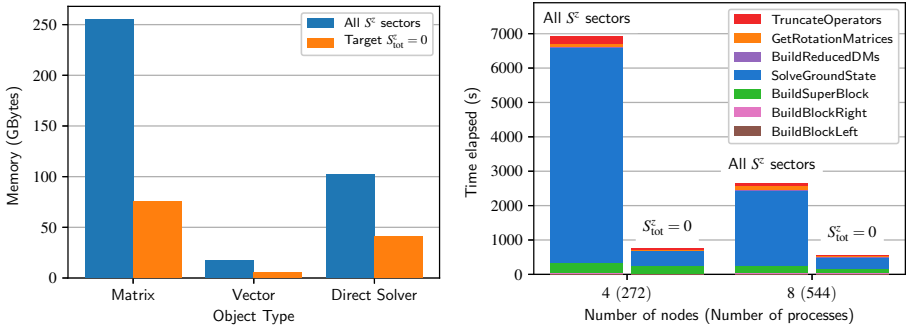



**Figure 4:** Memory consumption (left) and time to solution (right) comparing with and without targeting magnetization sector $S^z$, as measured on 4-8 nodes of Marconi KNL

# Performance Results $m = 768$

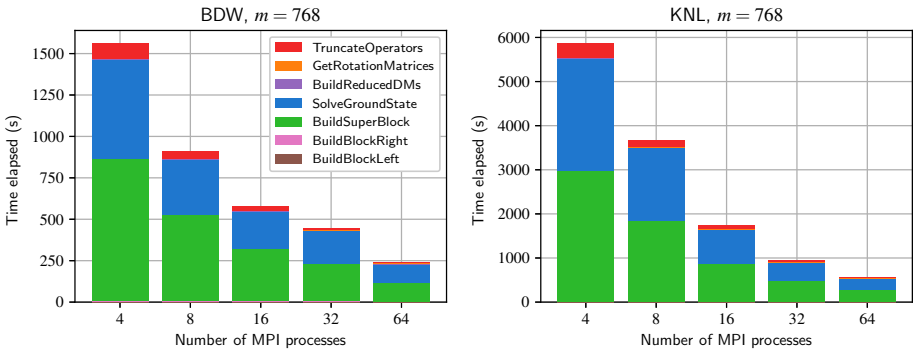



**Figure 5:** Elapsed times for $m = 768$ on 1-2 nodes with varying number of MPI processes

# Performance Results $m = 768$

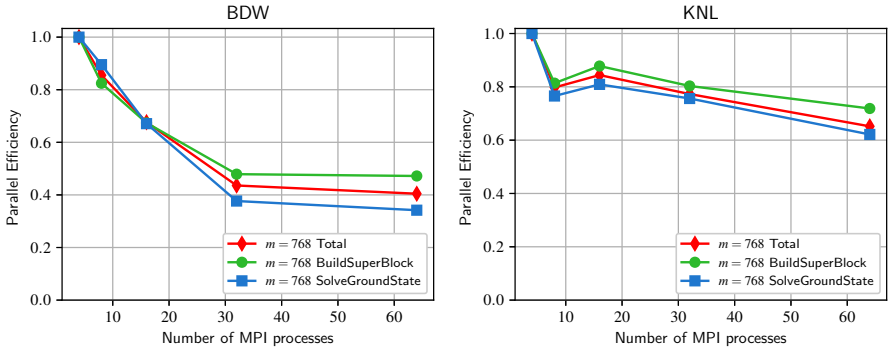



**Figure 6:** Parallel efficiency for $m = 768$ on 1-2 nodes with varying number of MPI processes

# Performance Results (Towards larger $m$)

## Full-Node vs Half-Node

| | Full-node case | Half-node case |
|---|---|---|
| **BDW** | 36 procs/node | 18 procs/node |
| **KNL** | 68 procs/node | 34 procs/node |

## Motivation for Running at Half-Node

- Greater memory bandwidth for each process
- Reduced memory capacity pressure
- More unknowns assigned to each process means less communication vs. computation during spMVM

# Performance Results $m = 1024$ on BDW

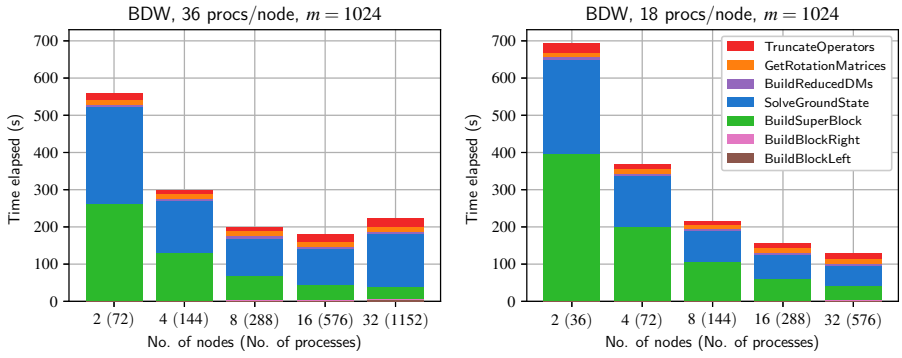



**Figure 7:** Elapsed times for $m = 1024$ kept states on BDW for full-node (left) and half-node (right) cases

# Performance Results $m = 1024$ on KNL

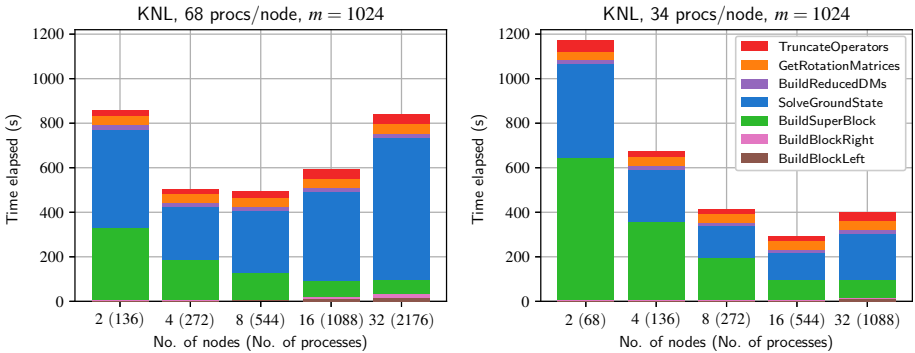



**Figure 8:** Elapsed times for $m = 1024$ kept states on KNL for full-node (left) and half-node (right) cases

# Performance Results $m = 1024$

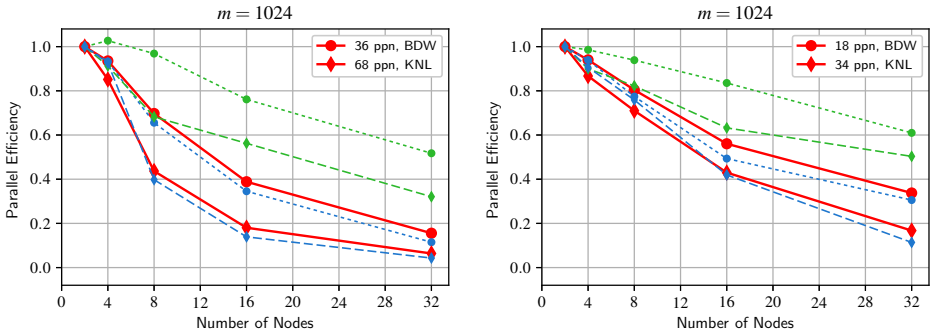



**Figure 9:** Parallel efficiencies for the *full-node case* (left) and *half-node case (right)* with $m = 1024$ kept states, shown for ■ all steps, ■ BuildSuperBlock and ■ SolveGroundState

# Performance Results $m = 1536$ on BDW

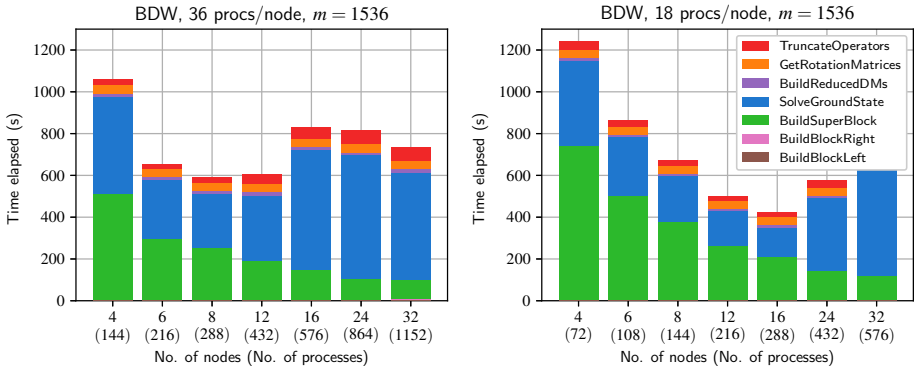



**Figure 10:** Elapsed times for $m = 1536$ kept states on BDW for full-node (left) and half-node (right) cases

# Performance Results $m = 1536$ on KNL

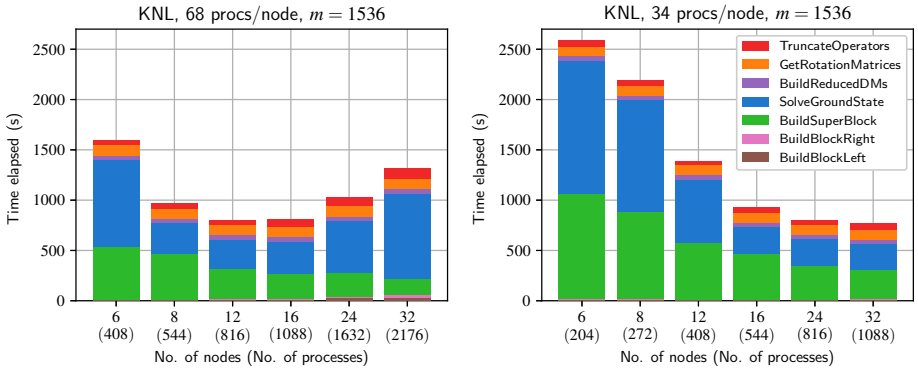



**Figure 11:** Elapsed times for $m = 1536$ kept states on KNL for full-node (left) and half-node (right) cases

# Performance Results $m = 1536$

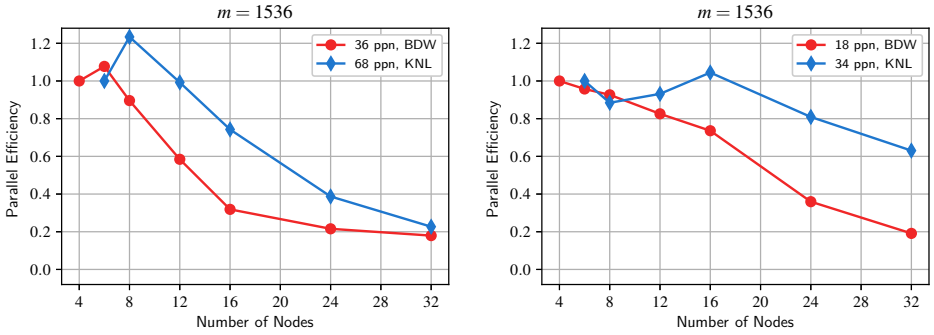



**Figure 12:** Parallel efficiency for the *full-node case* (left) and *half-node case (right)* with $m = 1536$ kept states

# Performance Results $m = 2048$

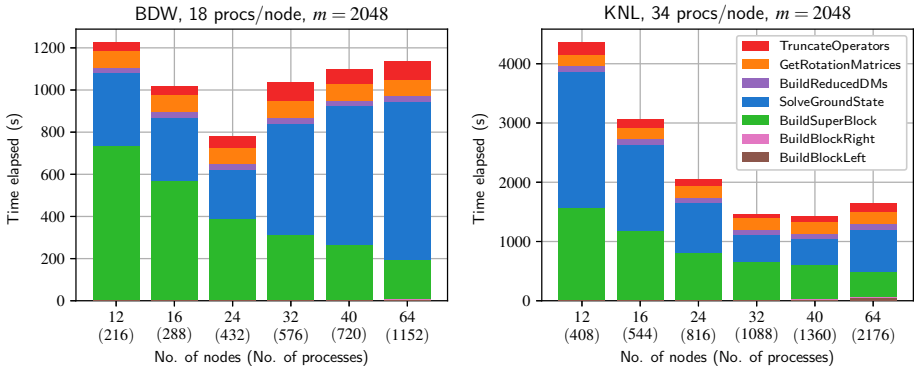



**Figure 13:** Elapsed times for the half-node case with $m = 2048$ kept states

# Performance Results $m = 2048$

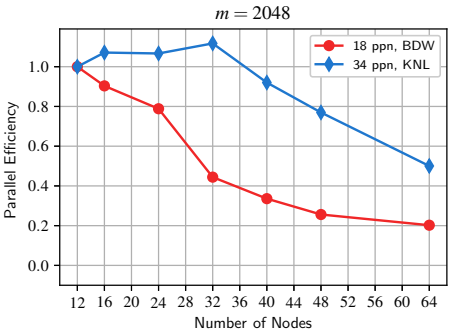



**Figure 14:** Parallel efficiency for the half-node case with $m = 2048$ kept states. KNL exhibits superscaling effects due to MCDRAM and DDR4 having a large difference in memory bandwidth

# Performance Analysis

**General Observations**

- Good scaling behavior of **BuildSuperBlock** for all cases due to optimally parallel **MatKronProdSum** routine
- Scalability of **SolveGroundState** (SLEPc) is better in the **half-node case**
- BDW is **faster** but KNL has **better parallel efficiency** and exhibits superscaling due to high-bandwidth memory

# Scaling of Computational Resources: Memory/Iteration

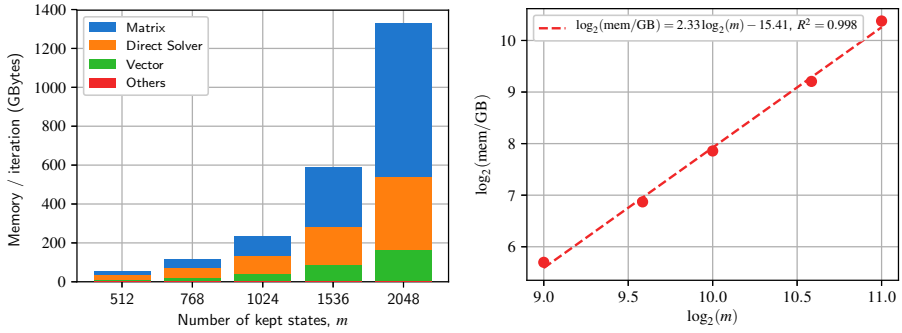



**Figure 15:** Scaling behavior of memory consumed with the number of kept states $m$ on 16 nodes of Marconi KNL ($\approx$ 100 GB/node)

**Estimation**:

Given 3600 KNL nodes with 310TB of memory, we can target up to $m = 22,000$ states

# Scaling of Computational Resources: Elapsed Time/Iteration

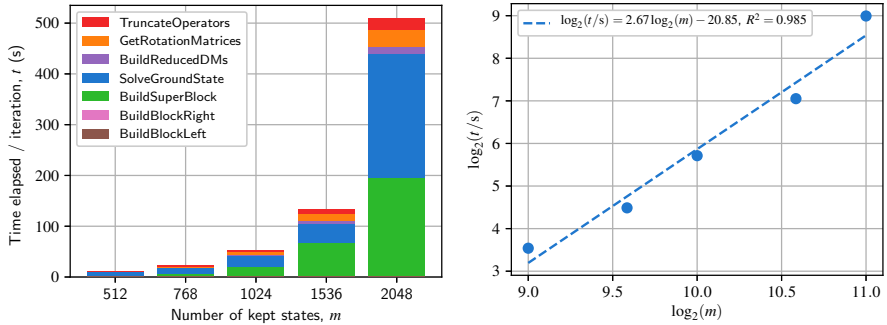



**Figure 16:** Scaling behavior of elapsed time with the number of kept states $m$ on 16 nodes of Marconi KNL running at half-node

**Estimation**:

If we target $m = 22000$ states on 3600 KNL nodes, we would consume 85,000 cpu-h/iteration

# Discussion

**Limitations to scalability**

1. Eigensolver is limited by computation vs communication during spMVMs
2. Different *m* require different minimum numbers of nodes
3. spMVM is also limited by the memory bandwidth per core

**To achieve best performance**

1. Limit the total number of processes to reduce communication vs computation
2. Use enough nodes to fit the problem in memory
3. Identify a good minimum number of processes per node so that each process still gets high memory bandwidth

# Conclusions

# Conclusions

- Presented our **massively parallel DMRG implementation**
- **Assessed its performance** on up to thousands of cores on the KNL and BDW sections of Marconi
- **Identified limitations in scalability**: communication bottlenecks, memory size, and memory bandwidth
- **Measured scaling of resources** allowing us to predict time and memory needed for any *m*
- **Demonstrated the framework** for full DMRG calculations of 2D systems for large number of kept states, allowing us to provide compelling evidence of topological order in experimentally relevant model systems

# Ongoing and Future Work

# Ongoing and Future Work

- **ISCRA C project** to study topological phases in 2D
- **2D finite-size DMRG** - same workflow but more Hamiltonian terms and greater memory consumption
- **Efficient matrix-free approach** using `MATSHELL`
- **Exploit KNL in flat mode** using `memkind` to control data between MCDRAM and DDR4

# Thank you
## for your attention!

Special thanks:


ICTP-TRIL



# References I


[1] S Yan, D A Huse, and S R White. “Spin-Liquid Ground State of the S = 1/2 Kagome Heisenberg Antiferromagnet”. In: *Science* 332.6034 (2011), pp. 1173–1176.

[2] Patrick a. Lee, Naoto Nagaosa, and Xiao-Gang Wen. “Doping a Mott insulator: Physics of high-temperature superconductivity”. In: *Rev. Mod. Phys.* 78.1 (2006), pp. 17–85. ISSN: 0034-6861. DOI: 10.1103/RevModPhys.78.17. URL: http://link.aps.org/doi/10.1103/RevModPhys.78.17.

[3] C. Lacroix, P. Mendels, and F. Mila, eds. *Introduction to Frustrated Magnetism*. Springer Series in Solid-State Sciences Vol. 164, 2010.

[4] H C Jiang, Z Y Weng, and D N Sheng. “Density Matrix Renormalization Group Numerical Study of the Kagome Antiferromagnet”. In: *Physical Review Letters* 101.11 (Sept. 2008), pp. 117203–4.

# References II


[5] E Gibney and D Castelvecchi. *Physics of 2D exotic matter wins Nobel*. Vol. 538. Nature, 2016.

[6] Mischa Thesberg and Erik S Sørensen. "An Exact Diagonalization Study of the Anisotropic Triangular Lattice Heisenberg Model Using Twisted Boundary Conditions". In: *arXiv.org* 1 (June 2014), p. 115117. arXiv: `1406.4083v2 [cond-mat.str-el]`.

[7] M Troyer and U J Wiese. "Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations". In: *Physical Review Letters* 95.12 (2005).

[8] Steven R White. "Density matrix formulation for quantum renormalization groups". In: *Physical Review Letters* 69.19 (1992), pp. 2863–2866.

[9] U Schollwöck. "The density-matrix renormalization group". In: *Reviews of Modern Physics* 77.1 (Jan. 2005), pp. 259–315.

# References III


[10] Garnet Kin-Lic Chan and Sandeep Sharma. "The Density Matrix Renormalization Group in Quantum Chemistry". In: *Annual Review of Physical Chemistry* 62.1 (May 2011), pp. 465–481.

[11] Susumu Yamada, Masahiko Okumura, and Masahiko Machida. "Direct Extension of Density-Matrix Renormalization Group to Two-Dimensional Quantum Lattice Systems: Studies of Parallel Algorithm, Accuracy, and Performance". In: *Journal of the Physical Society of Japan* 78.9 (Sept. 2009), pp. 094004–5.

[12] E M Stoudenmire and Steven R White. "Studying Two-Dimensional Systems with the Density Matrix Renormalization Group". In: *Annual Review of Condensed Matter Physics* 3.1 (Mar. 2012), pp. 111–128.

[13] F B Ramos and J C Xavier. "N-leg spin-S Heisenberg ladders: A density-matrix renormalization group study". In: *Physical Review B* 89.9 (Mar. 2014), pp. 094424–7.

# References IV


[14] Gabriele De Chiara et al. “Density Matrix Renormalization Group for Dummies”. In: *Journal of Computational and Theoretical Nanoscience* 5.7 (July 2008), pp. 1277–1288.

[15] J. E. Roman et al. *SLEPc Users Manual*. Tech. rep. DSIC-II/24/02 - Revision 3.7. D. Sistemes Informàtics i Computació, Universitat Politècnica de València, 2016.

[16] Intel. *Intel Xeon Processor E5-2697 v4*. accessed 24 November 2017. 2017. URL: https://ark.intel.com/products/91755/Intel-Xeon-Processor-E5-2697-v4-45M-Cache-2_30-GHz.

[17] Karthik Raman (Intel). *Optimizing Memory Bandwidth in Knights Landing on Stream Triad*. accessed 24 November 2017. 2017. URL: https://software.intel.com/en-us/articles/optimizing-memory-bandwidth-in-knights-landing-on-stream-triad.

# Single DMRG Iteration

**BuildBlockLeft** and **BuildBlockRight**

Grow blocks by adding one site $\dim(L) = m \rightarrow \dim(L\circ) = dm$

$$\mathbf{O}_{L\circ} = \sum_n \mathbf{O}_{L,n} \otimes \mathbf{s}_n \tag{2}$$

**BuildSuperBlock***

Form the superblock Hamiltonian $\dim(L \circ \circ R) = d^2m^2$

$$\mathbf{H}_{L\circ\circ R} = \sum_n a_n \mathbf{O}_{L\circ,n} \otimes \mathbf{O}_{\circ R,n} \tag{3}$$

**SolveGroundState***

Diagonalize using Lanczos iteration

$$\mathbf{H}_{L\circ\circ R}|\psi_0\rangle = E_0|\psi_0\rangle \tag{4}$$

***Computationally-intensive operations**

**BuildReducedDMs**

Obtain the reduced density matrices

$$\rho_{L\circ} = \mathrm{Tr}_{\circ R}\,|\psi_0\rangle\langle\psi_0| \tag{5}$$

**GetRotationMatrices**

Get $m$ largest eigenstates of $\rho_{L\circ}$ and form $dm \times m$ matrix $U_{L\circ}$

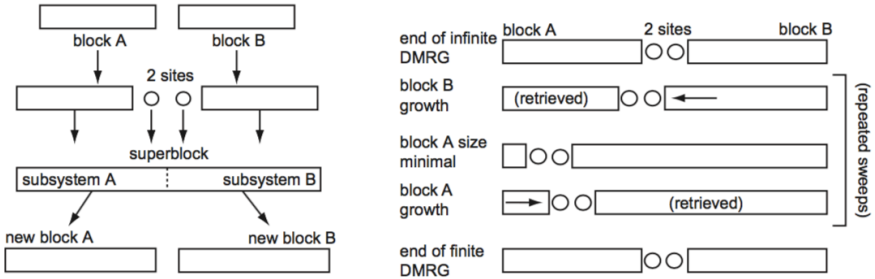
$$\rho_{L\circ} \approx \sum_{\alpha=1}^{m} \omega_\alpha\,|\alpha\rangle_{L\circ\,L\circ}\langle\alpha| \equiv U_{L\circ} \tag{6}$$

**TruncateOperators**

Rotate each operator to the new basis $\dim(L\circ) = dm \rightarrow \dim(L\circ) = m$

$$\tilde{\mathbf{O}}_{L\circ} = (U_{L\circ})^\dagger\,\mathbf{O}_{L\circ}U_{L\circ} \tag{7}$$

# Infinite and Finite DMRG Algorithms




- Infinite DMRG (left) is computationally cheap but convergence is poorer especially since the shape of our 2D lattice is not preserved
- Finite-size DMRG (right) helps correct this by applying sweeps and optimizing the wave function

# Scaling of Computational Resources

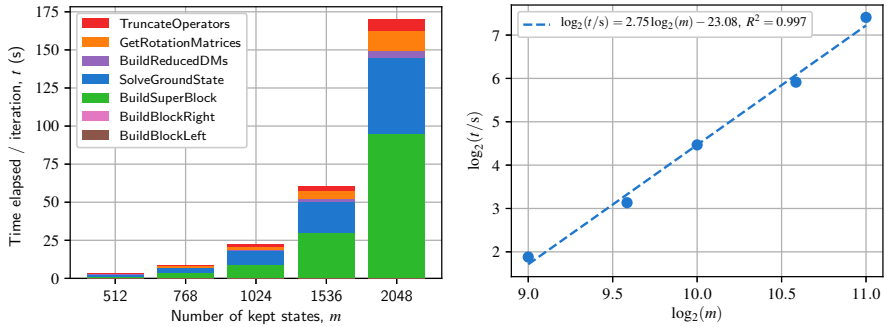



**Figure 17:** Scaling behavior of elapsed time with the number of kept states $m$ on 16 nodes of Marconi BDW running at half-node

# Scaling of Computational Resources

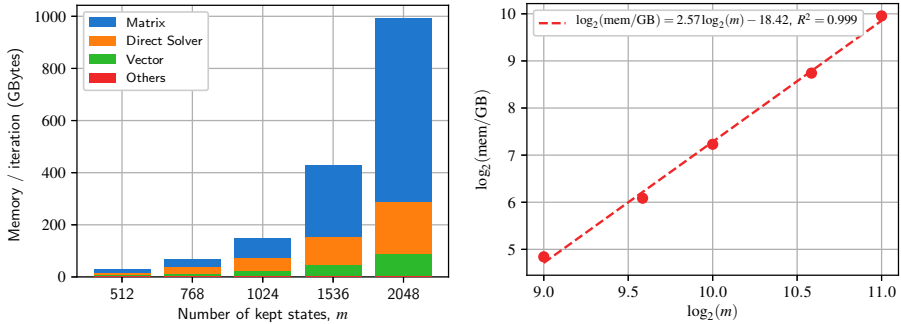



**Figure 18:** Scaling behavior of memory consumed with the number of kept states $m$ on 16 nodes of Marconi BDW

# 1D Traversal of Square Lattice

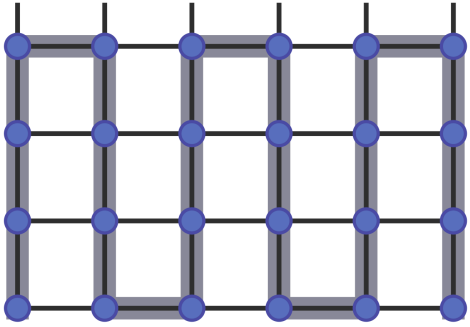


**Figure 19:** One-dimensional traversal of the square lattice creating long-range interactions