Ab Initio Nuclear Structure Calculations on High Performance Computing systems

Pieter Maris

Iowa State University
pmaris@iastate.edu

DOE-SciDAC3 NUCLEI
NESAP project at NERSC
INCITE award at ALCF and OLCF

Progress in Ab Initio Techniques in Nuclear Physics
TRIUMF, Vancouver BC, March 1, 2017

Outline

High-Performance Computing

No-Core Configuration Interaction approach

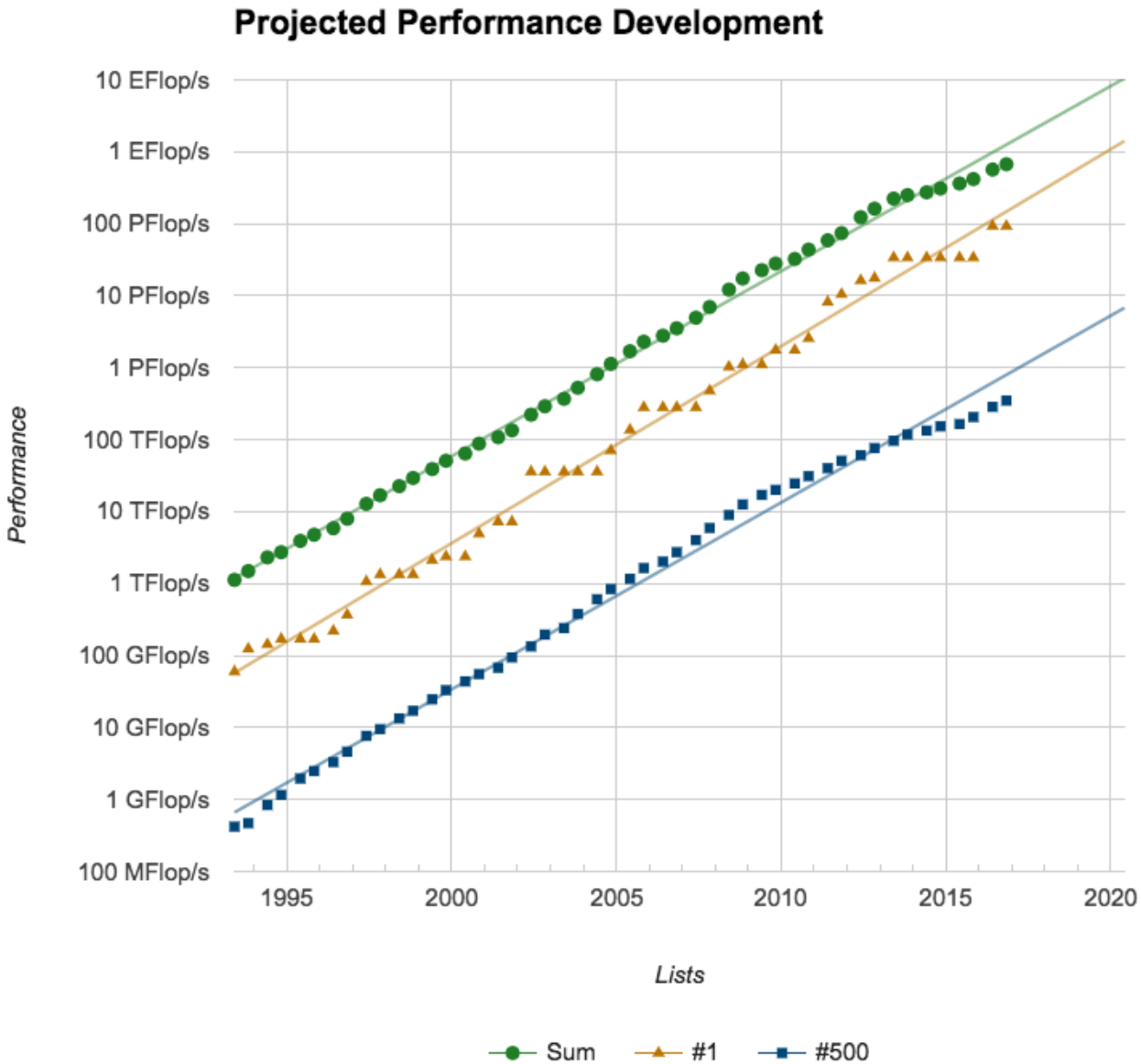
Many-Fermion Dynamics for nuclear structure

Algorithmic improvements: Lanczos vs. LOBPCG

Porting and Tuning MFDn for Intel Xeon Phi 'KNL'

Results for *p*-shell nuclei with chiral EFT up to N^2LO

High-Performance Computing: Moore's law



Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719
7	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	206,720	9,779.0	15,988.0	1,312
9	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
10	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	4,233

Top 10 (Nov. 2016)

- top 2 from China
- 5 from the US
  - 3 Titan @ ORNL
    Cray XK7 (GPU's)
  - 4 Sequoia @ LLNL
    IBM BG/Q
  - 5 Cori @ NERSC
    Cray XC40 (KNL)
  - 9 Mira @ ANL
    IBM BG/Q
  - 10 Trinity @ LANL/SNL
    Cray XC40

High-Performance Computing: Challenges

- Parallel computing
  - Initially: Shared memory or distributed memory parallel systems
  - Currently: Systems have shared and distributed memory
    - use OpenMP within a node and MPI between nodes
- Accelerators
  - GPU's (NVIDIA), Xeon Phi (Intel), ...
  - Initially: as co-processor
  - Now/Soon: self-hosted
- Vectorization
  - Xeon Phi (KNL) has 512-bit vector units (8 double precision floats)
- Increasing performance gap between processor and memory
  - Available memory and memory bandwidth per PU decreases
  - Data locality and data placement is crucial

Highly nontrivial to achieve good performance

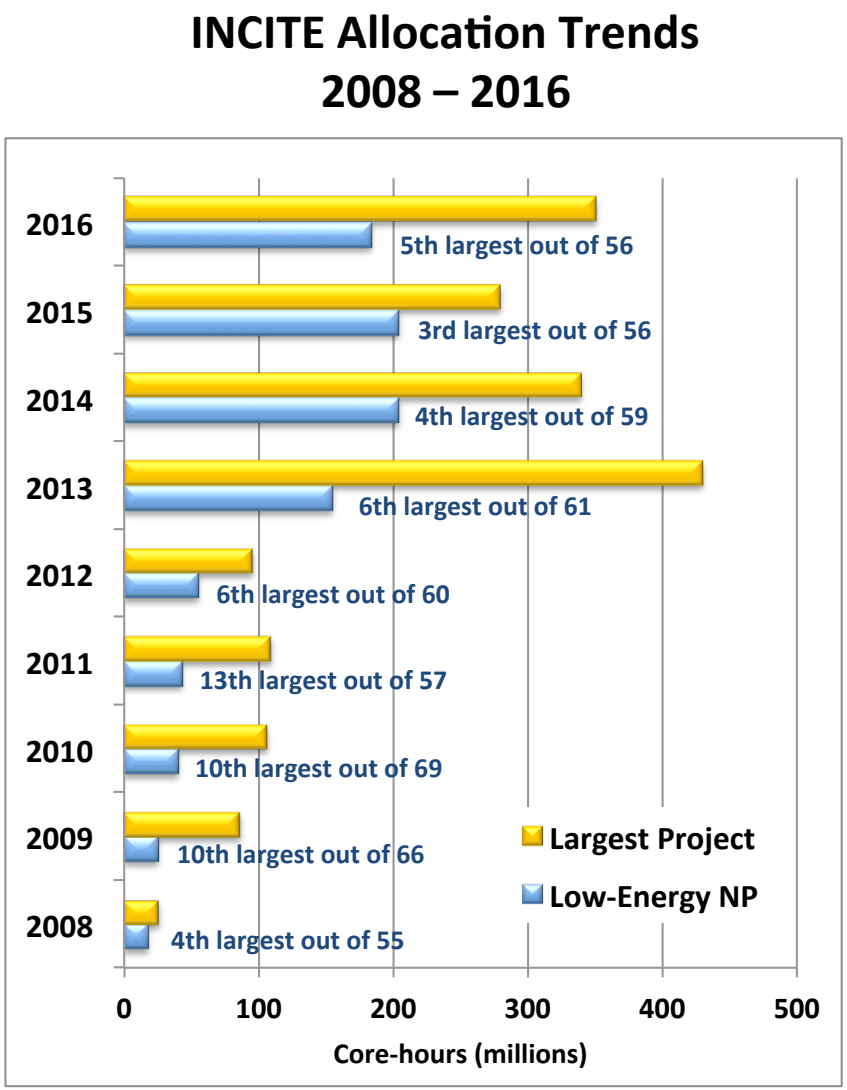
- Need to collaborate with applied mathematicians and computer scientists

Nuclear Structure Calculations

- Computational methods
  - Configuration Interaction (NCSM, and various variants thereof)
  - Coupled Cluster
  - In-Medium SRG
  - Many-Body Perturbation Theory
  - Nuclear Lattice Simulations
  - Quantum Monte Carlo (GFMC, AFDMC)
  - Self-Consistent Green's Functions
  - ...

  all have advantages and disadvantages
  all need (large) computers to obtain results
  with quantifiable uncertainties
- High-Performance Computing systems can be useful provided we can efficiently utilize the available computing power
  - nontrivial to do so ...

HPC usage on DOE leadership class facilities



INCITE allocations

- largest allocation 2008 – 2015: Lattice QCD
- largest allocation 2016: optimize coal burner designs
- allocation 2017:
  - 90 M core hours on Titan (CC, NCSM, IUMD)
  - 80 M core hours on Mira (GFMC, NCSM)

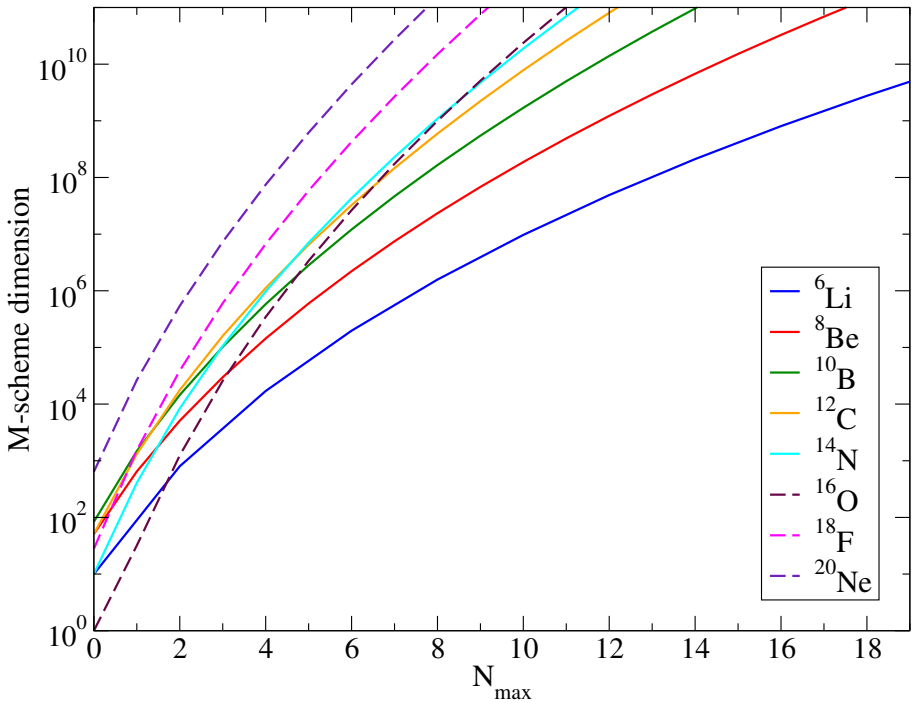
Moving to exascale

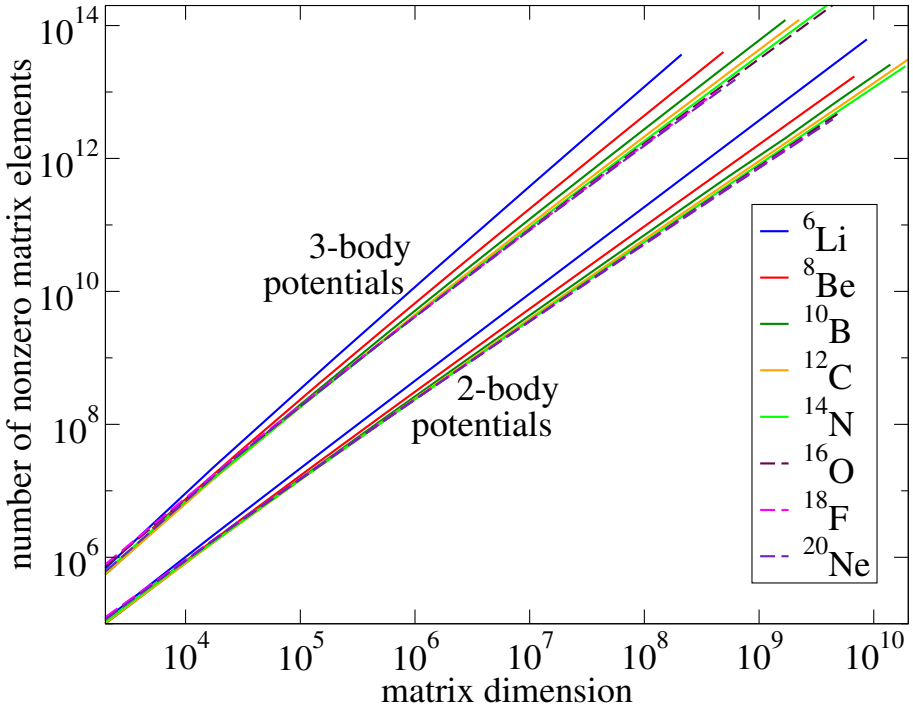
- Cori @ NERSC
  - 9,304 Intel Xeon Phi 'KNL' nodes
  - NESAP early science project: MFDn
  - user access 2017
- Summit @ ORNL
  - $\sim$3,400 compute nodes
  - multiple IBM POWER 9 CPUs and NVIDIA Volta GPUs per node
  - over 512 GB memory per node
  - CAAR early science project: NUCCOR (Gaute Hagen)
  - peak power consumption 10 MW
  - user access 2018
- Aurora @ ANL
  - over 50,000 compute nodes
  - next-generation Intel Xeon Phi (Knights Hill)
  - over 7 PB DRAM and persistent memory
  - peak power consumption 13 MW
  - user access 2019

No-Core Configuration Interaction approach

- Expand wavefunction in basis states $|\Psi\rangle = \sum a_i |\Phi_i\rangle$
- Express Hamiltonian in basis $\langle\Phi_j|\hat{\mathbf{H}}|\Phi_i\rangle = H_{ij}$
- Diagonalize Hamiltonian matrix H_{ij}
- No-Core: all A nucleons are treated the same
- Complete basis $\longrightarrow$ exact result
  - caveat: complete basis is infinite dimensional
- In practice
  - truncate basis
  - study behavior of observables as function of truncation
- Computational challenge
  - construct large sparse symmetric matrix H_{ij}
  - obtain lowest eigenvalues & -vectors corresponding to low-lying spectrum and eigenstates

NCCI approach – Main Challenge





- Increase of basis space dimension with increasing A and $N_{\max}$
  - need calculations up to at least $N_{\max} = 8$ for meaningful extrapolation and numerical error estimates
- More relevant measure for computational needs
  - number of nonzero matrix elements
  - current limit 10^{13} to 10^{14} (Cori, Mira, Titan)

Many-Fermion Dynamics for nuclear structure

Configuration Interaction code for nuclear structure calculations

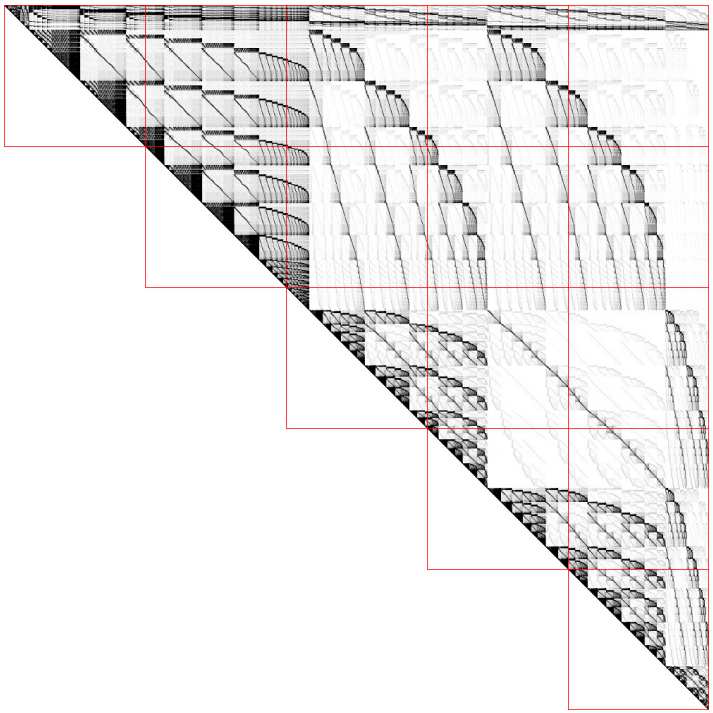
- Platform-independent, hybrid OpenMP/MPI, Fortran 90 (+ some C)
- Construct of many-body matrix H_{ij}
  - determine which matrix elements can be nonzero
  - evaluate and store nonzero matrix elements in compressed sparse block format (CSB)
- Obtain lowest eigenpairs using Lanczos algorithm or LOBPCG
  - eigenvalues: energy levels
  - eigenvectors: wavefunctions
  - most compute-intensive kernels
    - Lanczos: Sparse Matrix Vector Multiplication (SpMV)
    - LOBPCG: Sparse Matrix Matrix Multiplication (SpMM)
- Calculate observables from wavefunctions

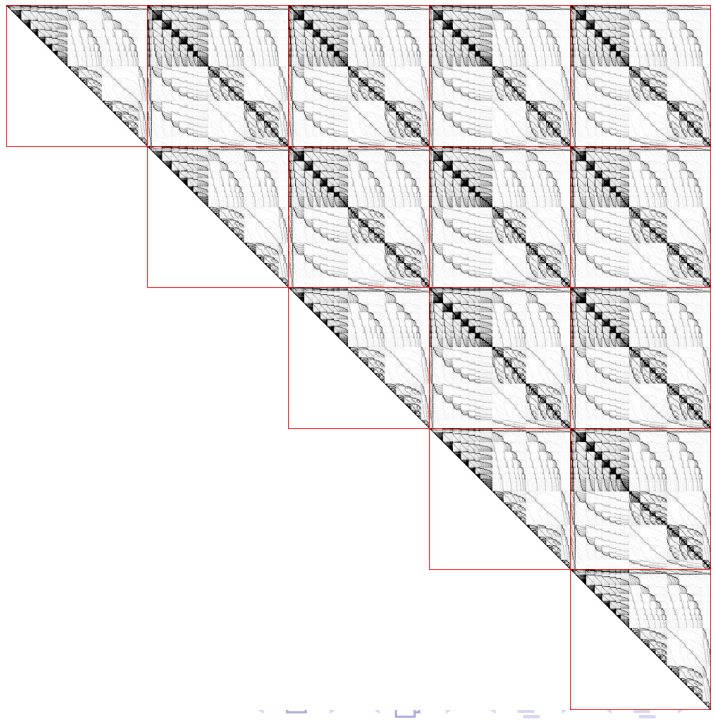
Biggest computational challenge

- Effective use of aggregate memory
  - calculations limited by aggregate memory 10^{13} to 10^{14} nonzero matrix elements (80 to 800 TB)

Distributed symmetric matrix

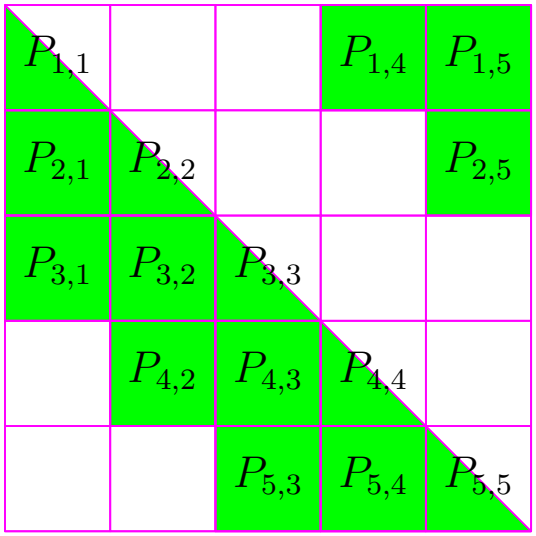
- Store only half the matrix (upper or lower triangle)
- Have to do SpMV and SpMV^T with same data structure
- Load-balancing
  - 2-dimensional distribution of matrix over MPI ranks
  - local load determined by number of nonzero matrix elements
  - can be achieved by even distribution of many-body (n, l, j) orbitals

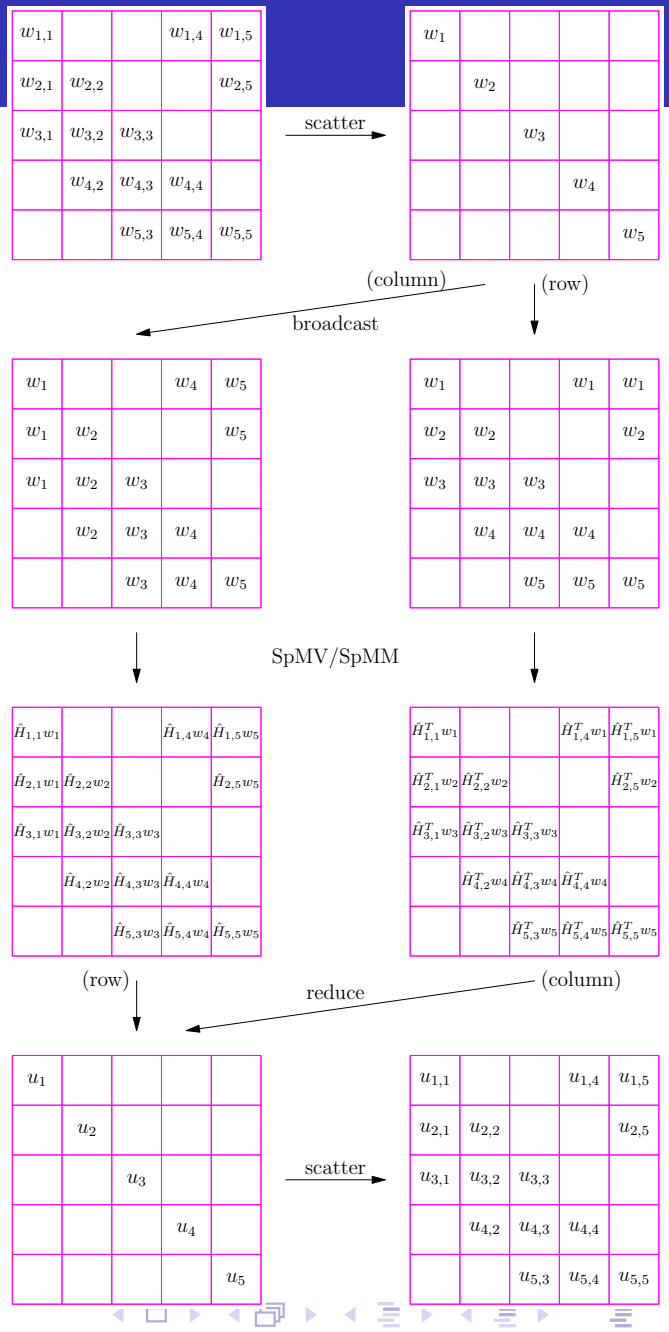




Efficient distributed SpMV

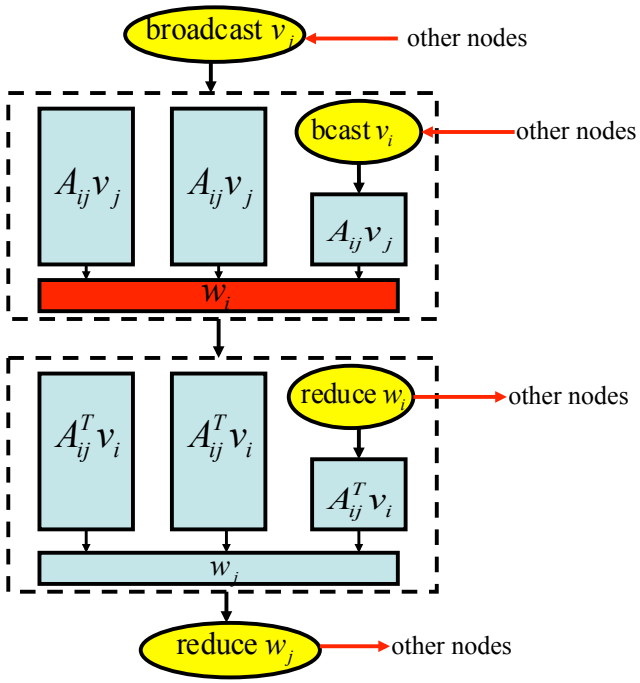
- Communication needs to be load-balanced as well
- Vectors distributed over all processors for orthogonalization

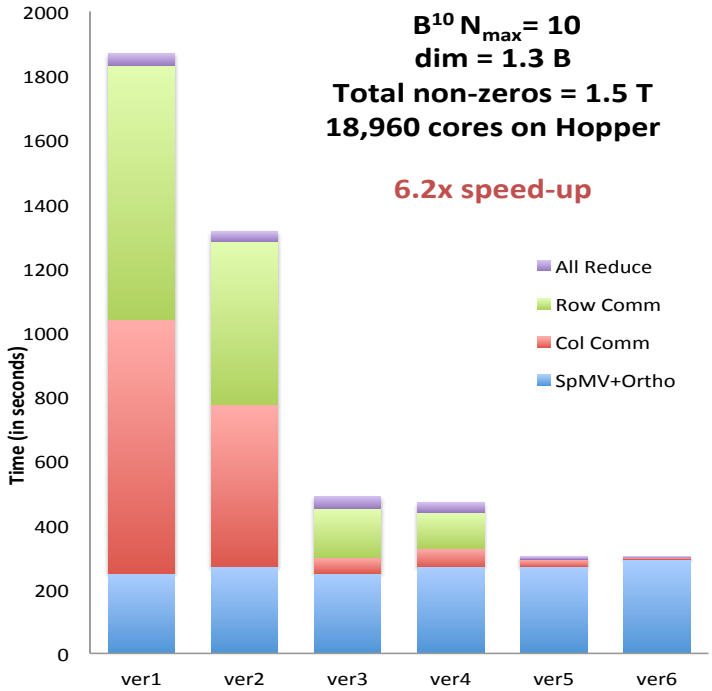




Efficient distributed SpMV – MPI communication

Aktulga, Yang, Ng, PM, Vary, Concurr. Comput. 26 (2014), doi:10.1002/cpe.3129

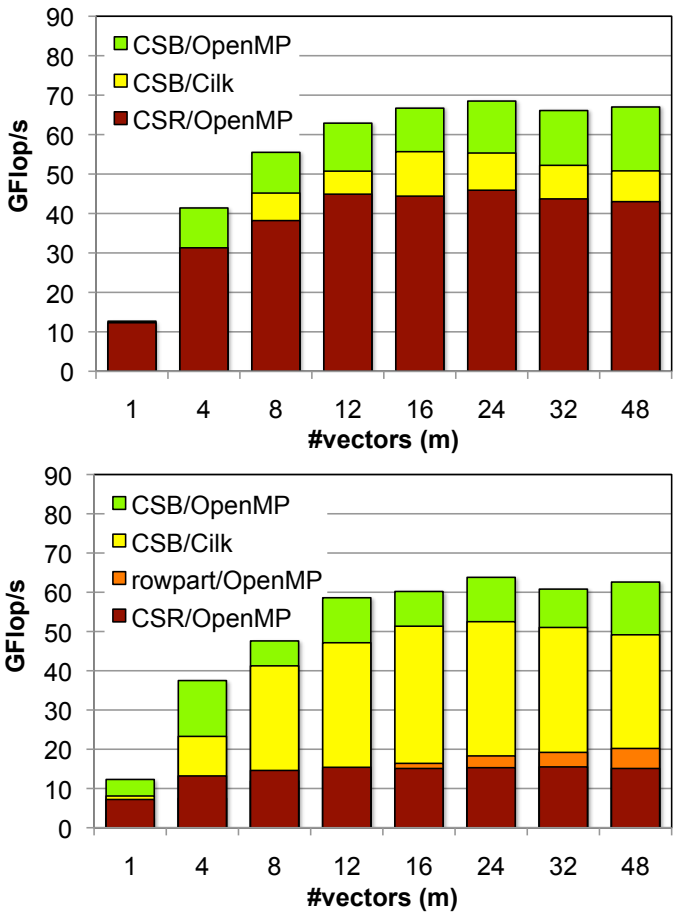




- Overlap communication with computation
- Optimize mapping onto network topology for non-overlapping communication

see also Oryspayev, PhD thesis 2016, ISU

Symmetric SpMV/SpMM implementation



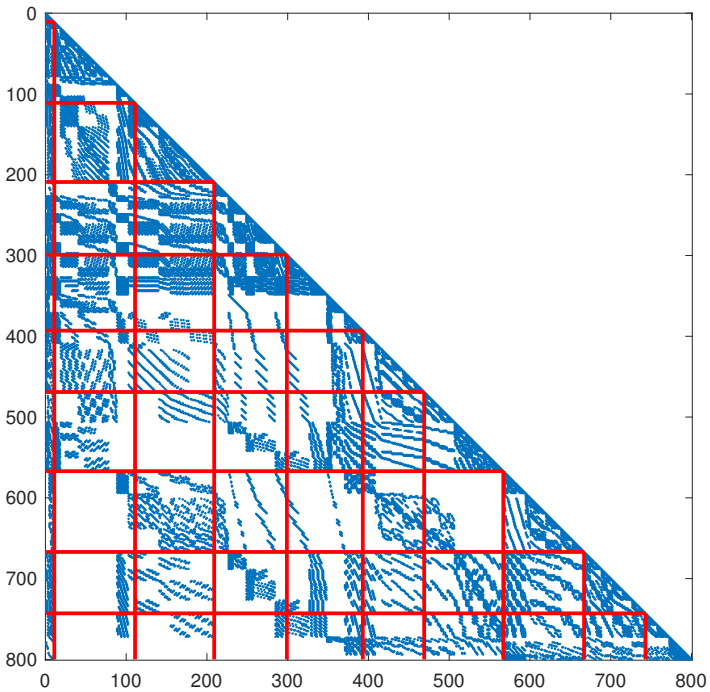
Intel Ivy Bridge (Edison @ NERSC)

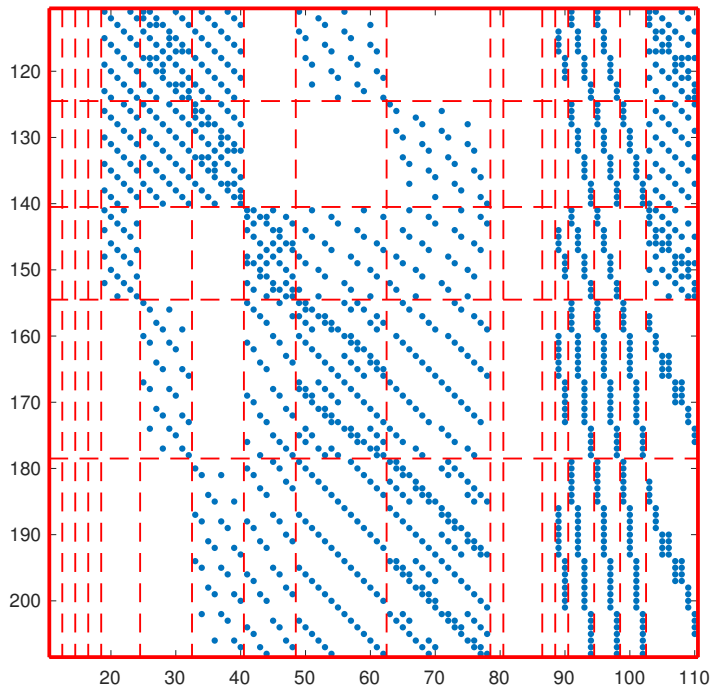
Aktulga, Afibuzzaman, Williams, Buluç, Shao, Yang, Ng, Maris, Vary, DOI 10.1109/TPDS.2016.2630699

- Compressed sparse row (CSR)
  - ok for SpMV
  - need private output vectors for SpMV^{T} to avoid race condition
  - prohibitively expensive on many-core architectures
- Compressed sparse block (CSB)
  - improves data locality and cache performance
  - allows for efficient OpenMP parallelization within nodes, for both SpMV (top) and SpMV^{T} (bottom)
- Block algorithm (LOBPCG)
  - SpMV on 'set of vectors' allows for vectorization

Matrix sparsity structure

Consider 'diagonal' MPI rank





- Nonzero tiles of varying size (dashed lines), defined by bra and ket many-body (n, l, j) orbitals
- Tiles are combined to form (approximately) square blocks (CSB), with boundaries coinciding with tile boundaries(solid lines)

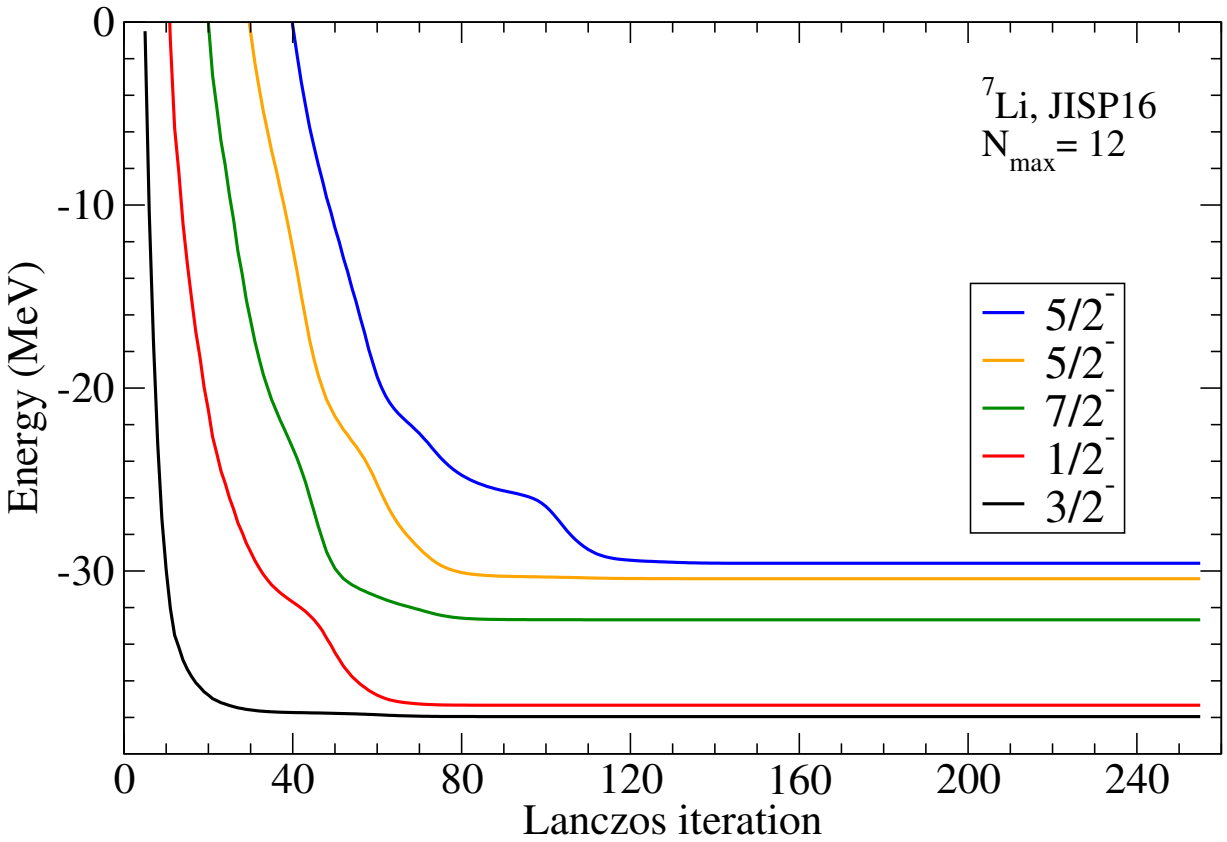
Lanczos algorithm

as implemented in MFDn

Let H be a symmetric matrix. Then H can be reduced to a symetric tri-diagonal matrix T via orthogonal unitary transformations, $H = Q_n T_n Q_n^T$

- For $i = 1$, set $\beta_1 = 0$ and initial vector q_1 with $||q_1|| = 1$
- While (not converged) do
  1. compute $p = H\,q_i$ — i.e. perform Sparse Matrix-Vector Multiplication
  2. compute $\alpha_i = q_i^T \cdot (H\,q_i)$ — i.e. perform dot-product
  3. compute $k = p - \alpha_i q_i - \beta_i q_{i-1}$
  4. (orthogonalize k w.r.t. q_i for numerical stability) — more dot-products
  5. compute $\beta_{i+1} = ||k||$ — and one more dot-products
  6. set $q_{i+1} = k/||k||$
  7. increment $i = i + 1$
  8. check (convergence) — diagonalize small tridiagonal matrix
     - obtain eigen-values λ and -vectors v of T_n — LAPACK
     - compute $\beta_i |\frac{v_i}{\lambda}|$ for each desired eigenvalue
- Compute approximate eigenvectors of H from T_n and Q_n

Lanczos algorithm



- dimension 252 million, with 400 billion nonzero matrix elements
- runs on 124 nodes Edison at NERSC using 496 MPI ranks with 6 OpenMP threads/MPI
- total runtime less than 10 minutes

- Lowest 5 eigenvalues of T_n after n Lanczos iterations
- Note: in MFDn we use single-precision for H and $Q = \{q_i\}$ but double-precision for dot-products and T_n for numerical stability

LOBPCG

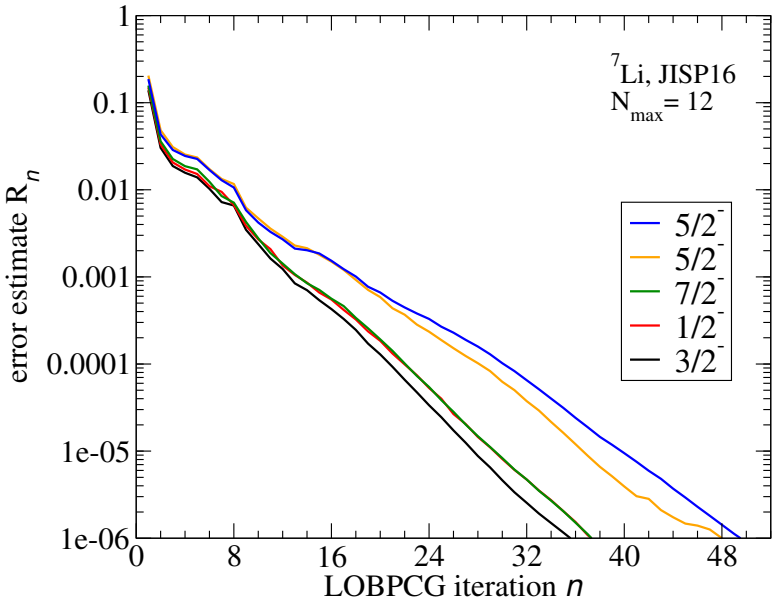
in collaboration with applied mathematicians from Berkeley

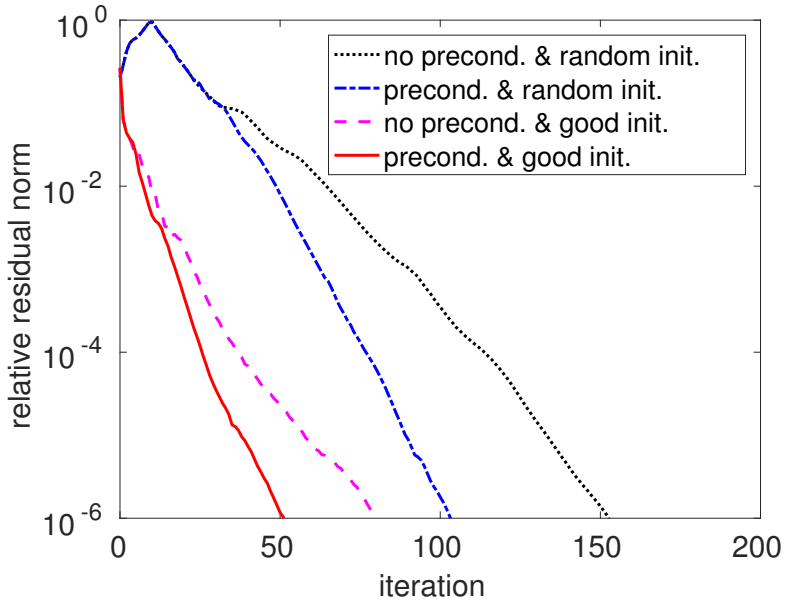
Locally Optimized Block Preconditioned Conjugate Gradient

- Set initial guess for $X^{(1)}$ consisting of k orthonormal vectors
  - ideally, consisting of approximate eigenvectors
  - e.g. smaller basis space, different H.O. parameter $\hbar\omega$, ...
- While (not converged) do
  1. apply preconditioner T
     - preconditioning is an art ...
     - kinetic energy is likely to be efficient, but too expensive
     - diagonal matrix element is cheap, but not efficient
     - compromise:
       diagonal tiles of H, based on many-body (n, l, j) orbitals
  2. orthonormalize using Cholesky QR
  3. compute $H X^{(i)}$ Sparse Matrix-Matrix Multiplication
  4. do LOBPCG magic ...
  5. check convergence
- $X^{(n)}$ consists of k orthonormal eigenvectors

LOBPCG

Shao, Aktulga, Yang, Ng, PM, Vary, arXiv:1609.01689 [cs.NA]



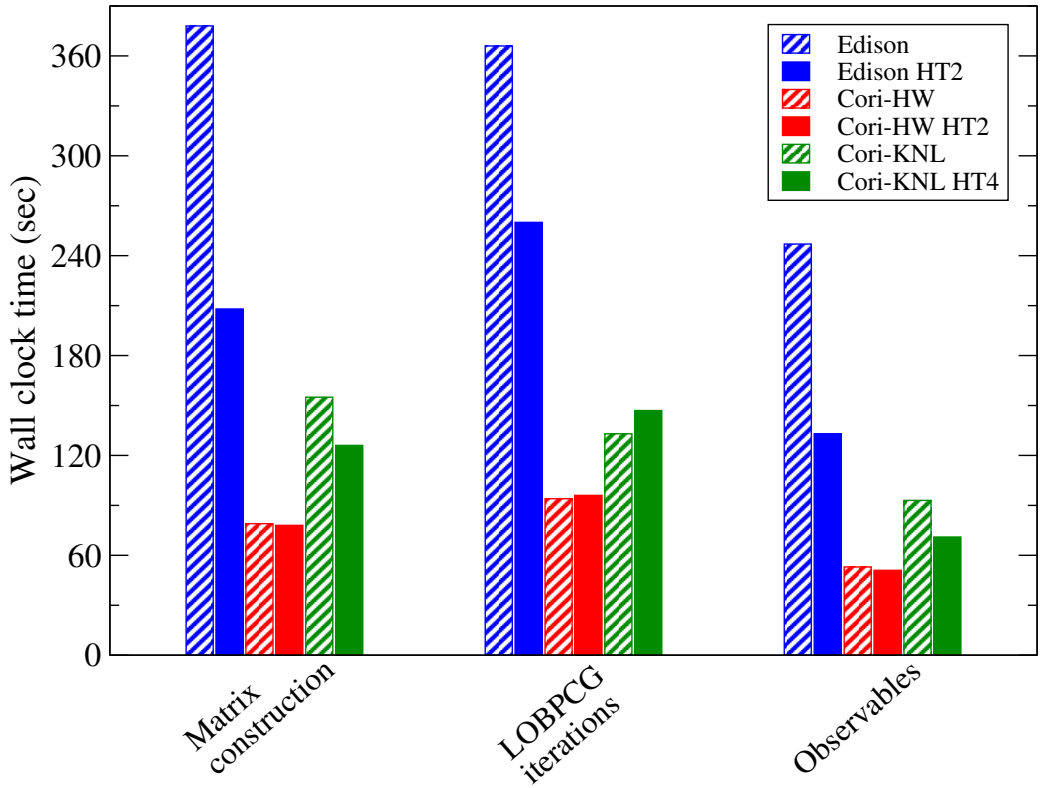


- Blocks of 8 vectors, targeting lowest 5 eigenstates
- $N_{\max} = 8$: 114 iterations in 6.5 seconds (using random initial vectors)
- $N_{\max} = 10$: 67 iterations in 19.8 seconds
- $N_{\max} = 12$: 50 iterations in 109.4 seconds
- Despite doing approximately 1.6 times more work in SpMV/SpMM, LOBPCG factor of 2 faster than Lanczos

HPC systems at NERSC

- Edison (in production since 2013)
  5,586 Intel 'Ivy Bridge' nodes
  - two 12 cores @ 2.4 GHz, 2 hyper-threads/core
  - one 256-bit-wide vector units per core
  - 64 GB DDR3 memory per core
- Cori-I (in production since 2016)
  2,004 Intel Xeon 'Haswell' nodes
  - 32 cores @ 2.3 GHz, 2 hyper-threads/core
  - two 256-bit-wide vector units per core
  - 128 GB DDR4 memory per core
- Cori-II (limited user access)
  9,304 Intel Xeon Phi 'Knights Landing (KNL)' nodes
  - 68 cores @ 1.4 GHz, 4 hyper-threads/core
  - two 512-bit-wide vector units per core
  - 96 GB DDR4 memory, plus 16 GB MCDRAM (high-bandwith)
  - aggregate memory: 1 PB

Edison vs. Cori-Haswell vs. Cori-KNL



^{10}B at $N_{max} = 8$

- dimension 160 million
- number of nonzero's 124 billion
- 30 compute nodes (could run on 15 nodes on Cori)

- Porting – no problem
- Without tuning, Cori-KNL significantly slower than Cori-HW
- Hyperthreading improves performance Edison, but not necessarily on Cori

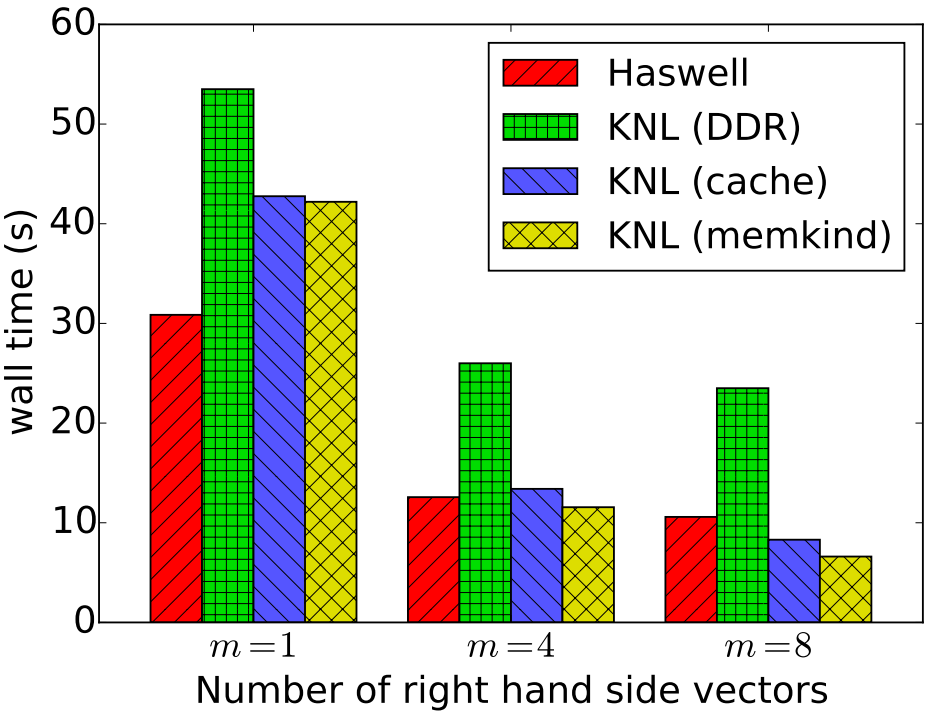
Tuning single-node performance on KNL B0 whitebox

- Single-node performace using MFDn proxy
  - local workload of one node out of 5,000 nodes production run
    - construction of local matrix with dimension $117,805,679 \times 116,805,483$ and 7.5×10^9 nonzero matrix elements
    - local SpMV/SpMM and transpose SpMV/SpMM
    - no orthonormalization, no communication
- Tuning for KNL
  - optimize memory placement
  - explore MPI and OpenMP scaling within node
  - improve cache re-use and vectorization
    - use compiler report to see which loops vectorize automatically
    - use OpenMP4 directives for manual vectorization
    - split complicated innerloops into smaller and simpler subloops tuned to vector length and/or cache size
    - improve data locality
- Compare to Intel Haswell

SpMV and SpMM performance – DDR vs. MCDRAM ?

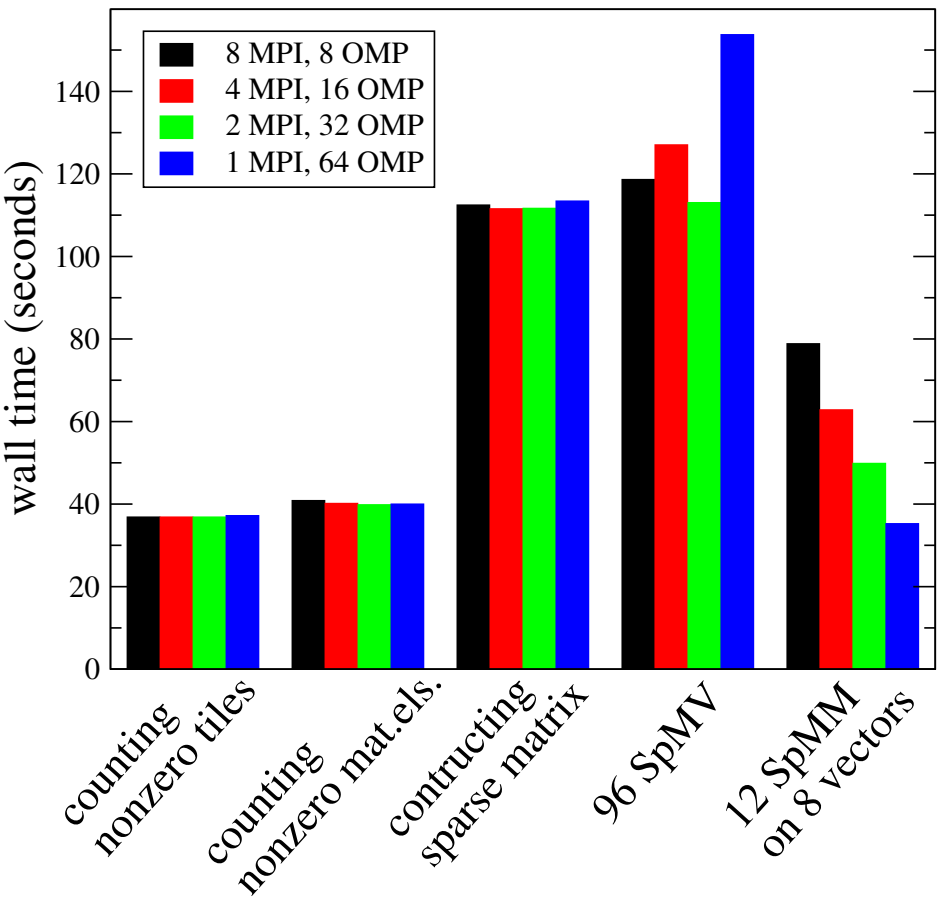
Cook, Maris, Shao, Wichmann, Wagner, O'neill, Phung and Bansal, LNCS 9945, 366 (2016), DOI 10.1007/978-3-319-46079-6_26

- 16 GB MCDRAM (high-bandwidth memory) can be used as extended cache, or explicitly managed
- Data placement using memkind library and FASTMEM directives



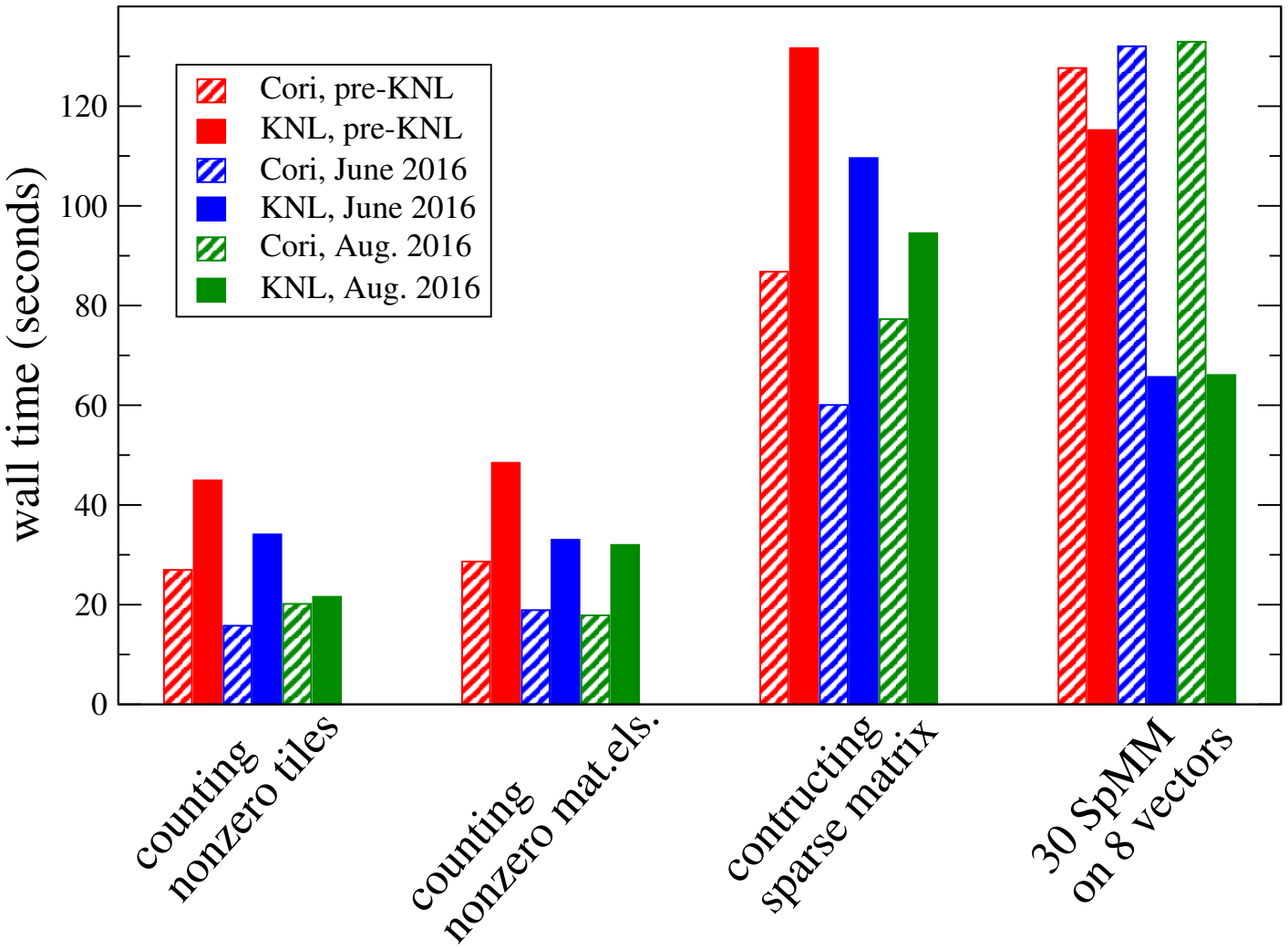
- SpMV on single vector
  - no vectorization, KNL slower than Haswell
  - both cache mode and quad+flat with vectors in MCDRAM improve performance
- SpMM on 4+ vectors
  - quad+flat with vectors in MCDRAM most efficient
  - KNL more efficient than Haswell

OpenMP vs. MPI on single KNL node ?



- No performance difference for setup and matrix construction
- SpMV more efficient with more MPI ranks?
  - local vectors smaller
  - more cache re-use?
- SpMM more efficient with fewer MPI ranks
  - smaller combined memory footprint
  - 8 vectors on 1 MPI rank barely fit in MCDRAM

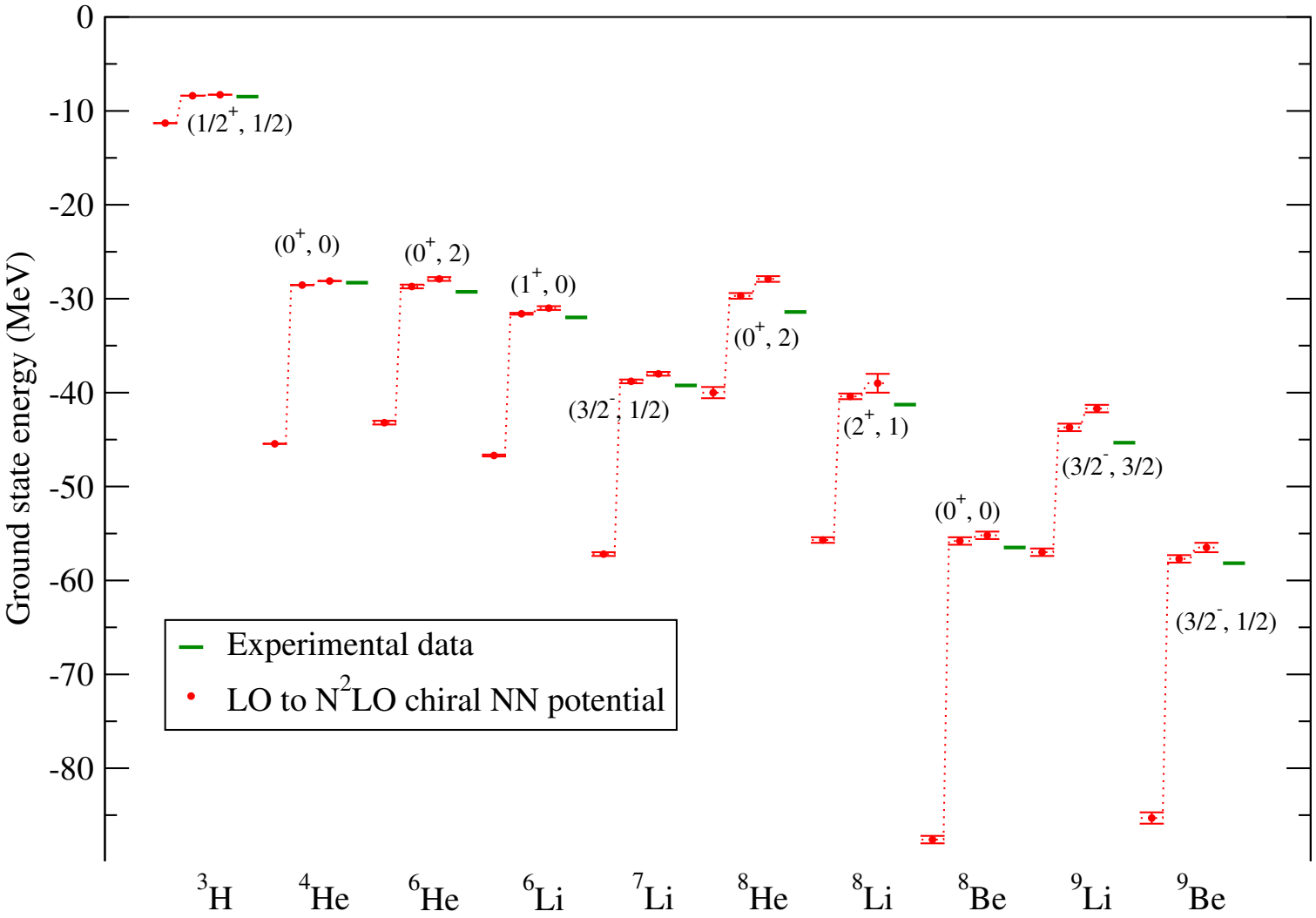
Summary of performance KNL vs. to Haswell



Summary of current status on Intel KNL

- Single-node performance matrix construction
  - similar performance as 'pre-optimized' code on Intel Haswell
  - hyper-threading helps, but more work to be done?
- Single-node LOBPCG diagonalization
  - explicit data management in MCDRAM
  - factor of 1.5 to 2.0 improvement over Intel Haswell
- Large-scale runs
  - load balancing of computational load good
  - bottleneck: MPI communication during LOBPCG diagonalization
    - communication volume 8 to 16 times larger than with Lanczos
    - one MPI rank per node: collective comm. by one core at a time
      MPI standard allows more threads to perform MPI communication
      however, MPI standard only guarantees correctness, not efficiency
      in practice collective MPI calls get serialized . . .
    - better communication performance with 4 to 16 MPI ranks per node,
      but overall memory footprint and communication volume increase
- Work in progress/under consideration:
  writing our own multithreaded collective MPI communication?

Ground state energies up to N^{2}LO for $A = 3$ to $A = 9$

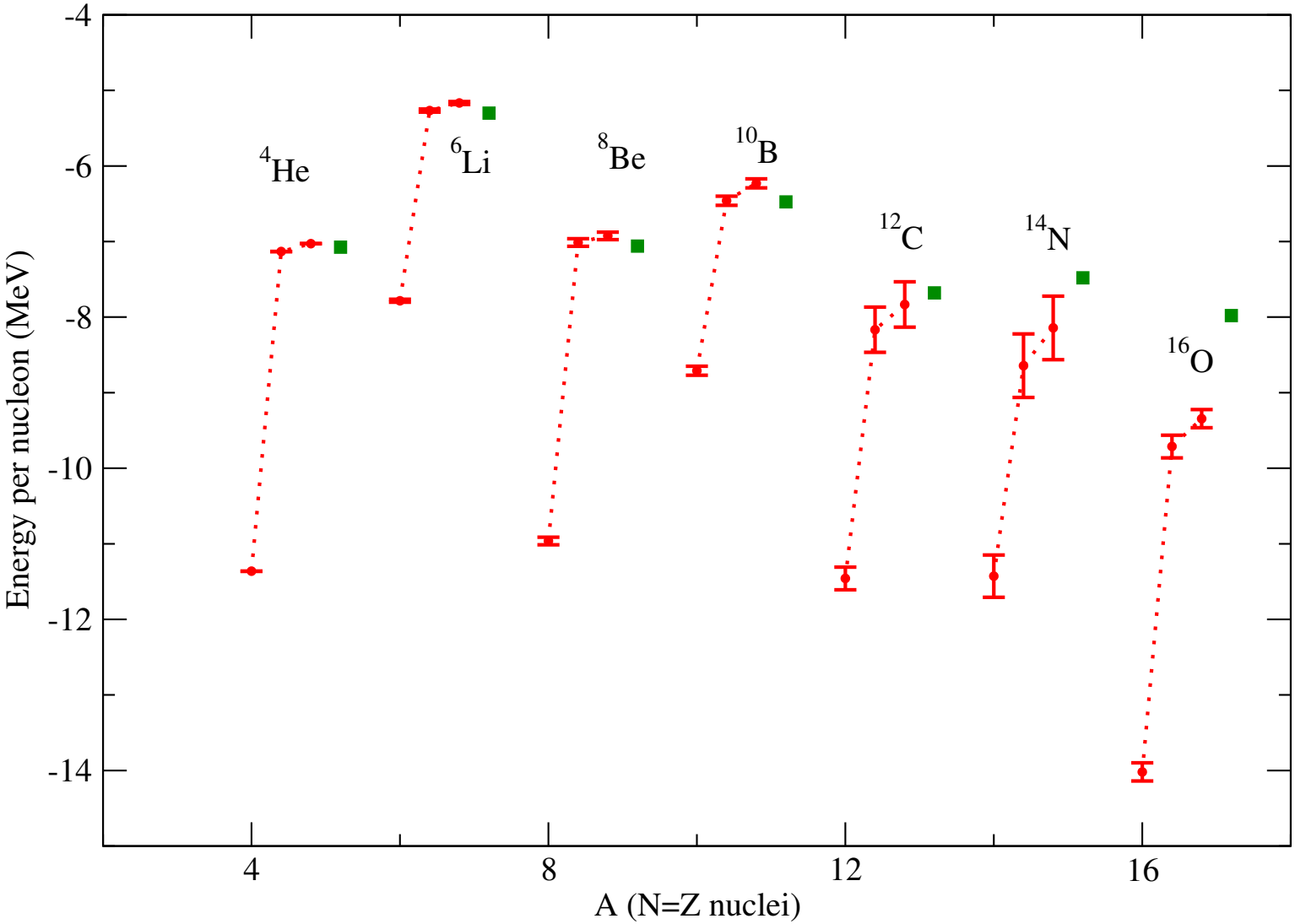


- NN only
- $R = 1.0$ fm
- many-body uncertainties shown

LENPIC collaboration

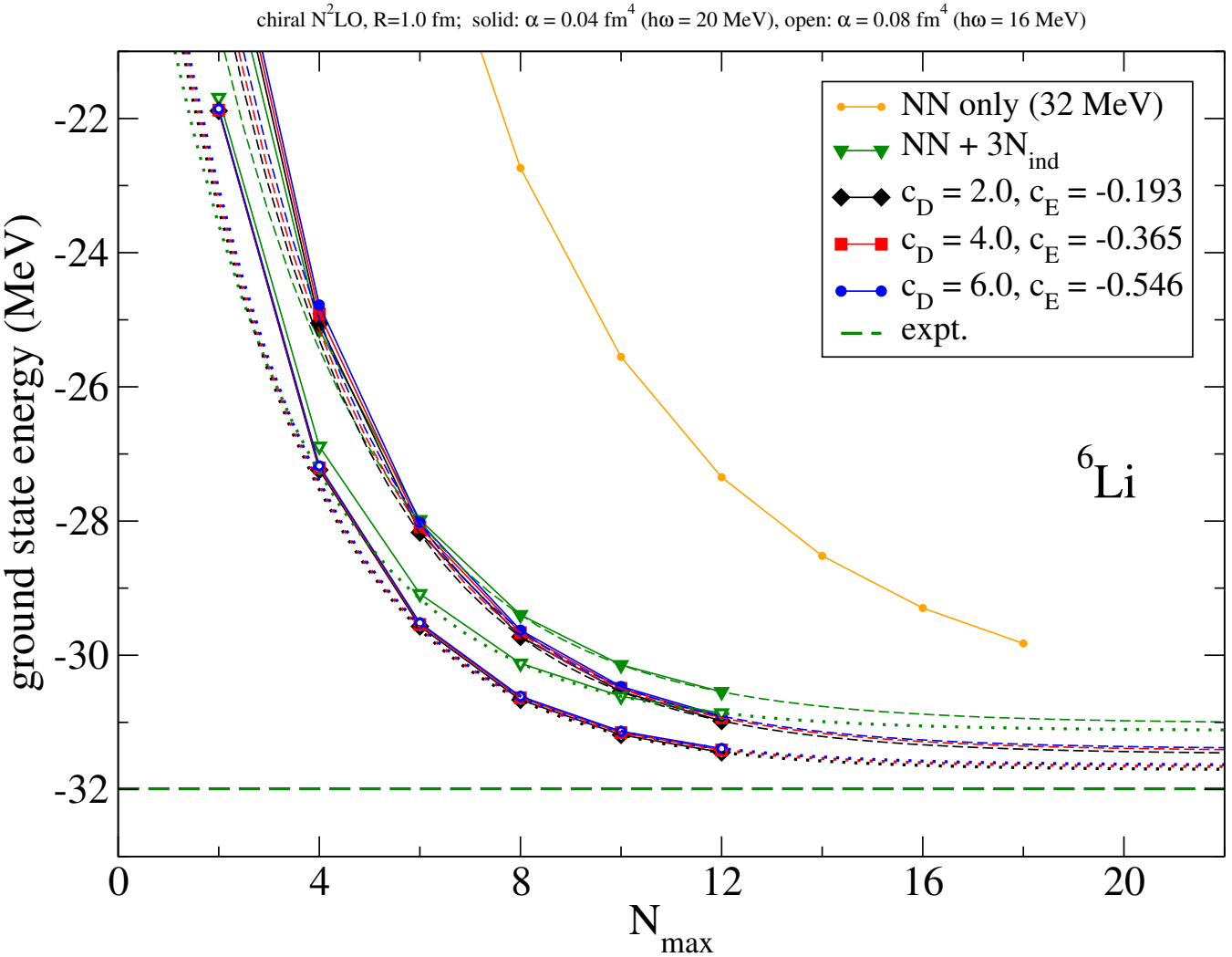
in preparation

$N = Z$ Ground state energies up to N^{2}LO



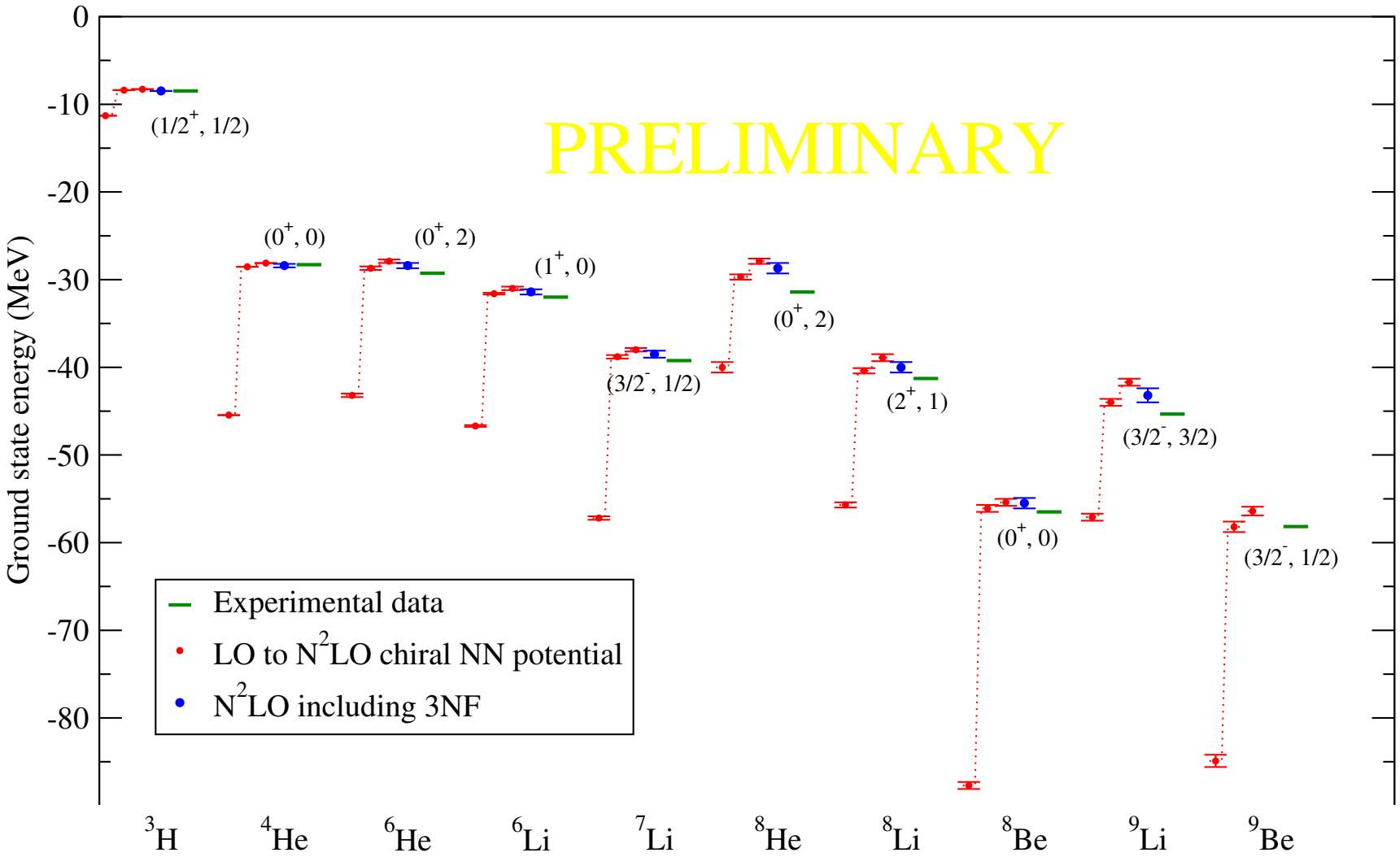
Including N^{2}LO 3N interaction: ^{6}Li

LENPIC collaboration, work in progress

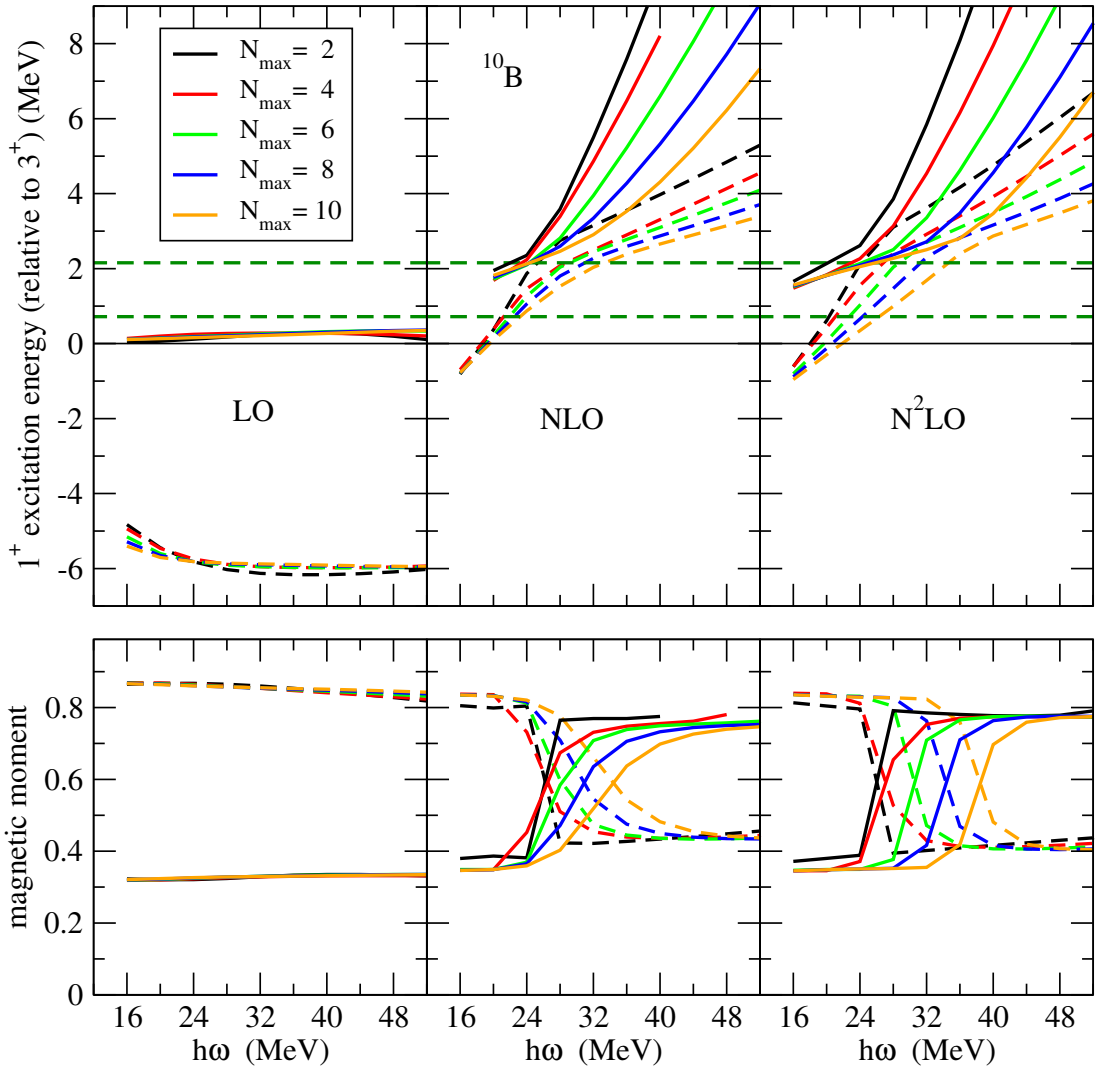


- extrapolation uncertainty ~ 0.1 MeV
- dependence on SRG α ~ 0.2 MeV
- dependence on (c_D, c_E) ~ 0.1 MeV
- gs energy NN only -31.0 ± 0.2 MeV with 3NF -31.4 ± 0.3 MeV

Ground state energies up to N^{2}LO including 3NF



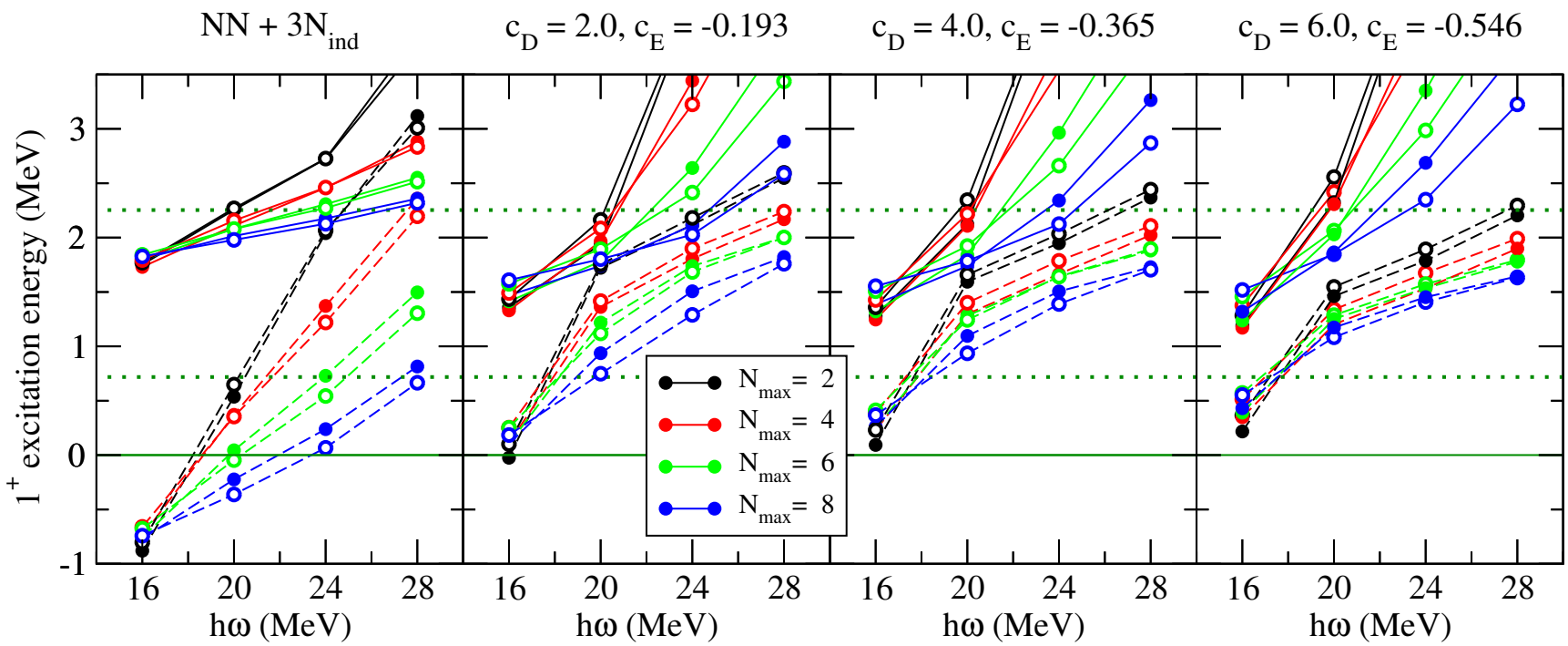
Spectrum ^{10}B: 1^+ states (NN-only)



- Two low-lying 1^+ levels
  - LO: well seperated
  - NLO (and higher): mix and cross, depending on basis parameters ($N_{\max}, \hbar\omega$)
- Can be distinguished by e.g. magnetic moments
  - state with $\mu \sim 0.4$ and $E_x \sim 2$ to 3 MeV
  - state with $\mu \sim 0.8$ and E_x strongly dependent on basis

Jurgenson et al. PRC87 (2013)

Spectrum ^{10}B at N^{2}LO: influence of 3NFs



- At N^{2}LO without 3NF's: lowest 1^+ below 3^+
- With 3NF's correct 3^+ ground state
- Preferred LEC's: $(c_D, c_E) = (6.0, -0.546)$
- Numerical uncertainties hard to estimate due to mixing ...

Conclusions and Outlook

- HPC expected to reach exascale capabilities by 2020
- Highly nontrivial to efficiently utilize current & future HPC systems
  - Need to collaborate with applied mathematicians and computer scientists
- Systematic calculations for *p*-shell nuclei
  - Order-by-order in χEFT
  - Same interactions also used for $A = 3$ and $A = 4$
    - Faddeev and Faddeev–Yakubovsky calculations
    - benchmark for NCCI calculations
  - Same interactions also used for heavier nuclei
    - IM-SRG and CC
    - benchmark with NCCI calculations for ^{16}O
- Uncertainty Quantification
  - Many-body method – dependence on basis space
  - Renormalization – SRG parameter dependence
  - Nuclear interaction – order in χEFT expansion