

ACCELERATING TOTAL-J CALCULATIONS

Hasan Metin Aktulga (LBL)

in collaboration with

Chao Yang, Esmond Ng (LBL)

Pieter Maris, James Vary (ISU)

Umit Catalyurek (OSU)

Total-J Overview

Alternative to M-scheme

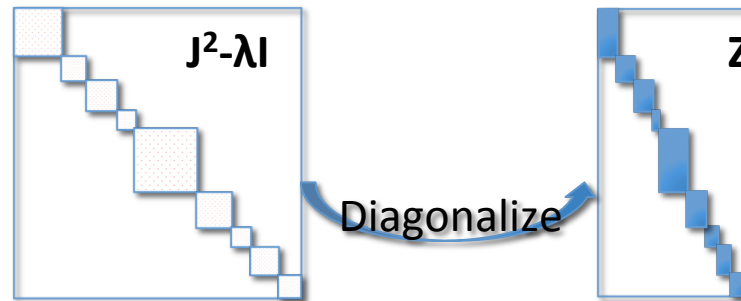
Large number of low energy states for given J

Applications:

- investigating nuclear level densities
- evaluating scattering amplitudes

Major Stages of Total-J Calculations

- 1 Construction of the invariant subspace Z



- 2 Projection of the Hamiltonian (H) into Z



- 3 Sparse matrix diagonalization on the projected H

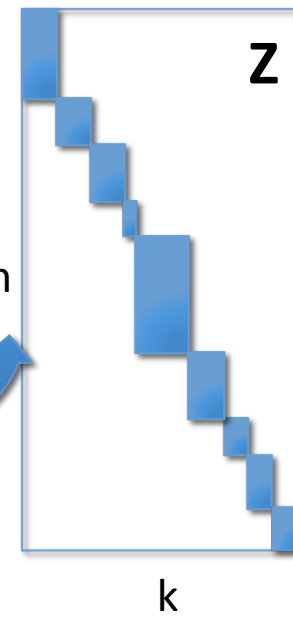
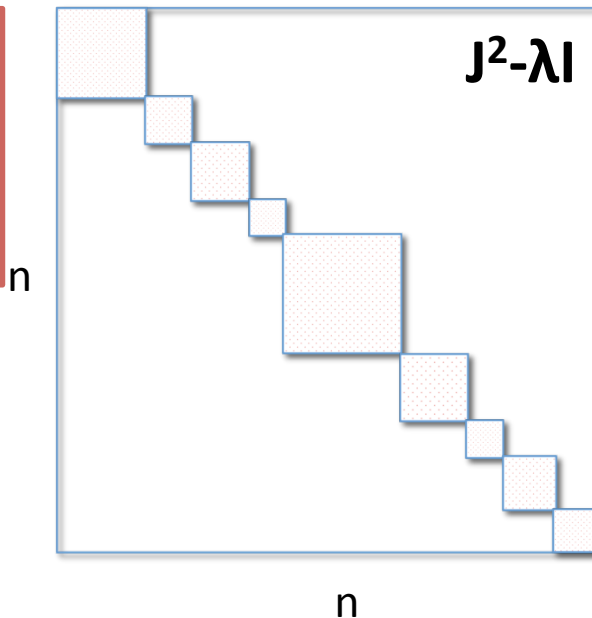
Construction of Invariant Subspace

Compute the **null space** of $J^2 - \lambda I$, $\lambda = J(J+1)$

Block diagonal

Sparse blocks

Task = block



Block diagonal

Dense blocks

Block = basis

Two major considerations

- Diagonalization of each sparse $J^2 - \lambda I$ block
- Efficient parallel calculation

Methods for $J^2-\lambda I$ Block Diagonalization

1 Randomized Rank-Revealing QR (RQR)

$N = \text{random } w/\text{Gaussian}$

$$[Q', R'] = \text{QR}(N)$$

vs.

$$\text{QR}(J^2_i)$$

$$[Q, R] = \text{QR}(J^2_i Q'^T)$$



observed up to
4x speed-up

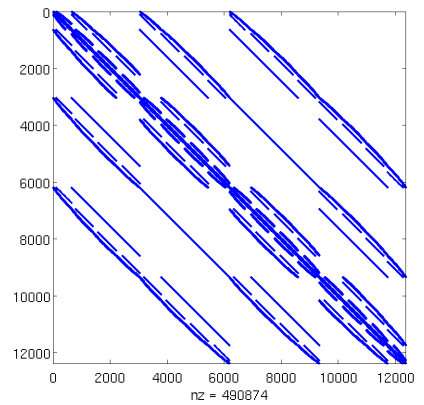
2 Polynomial Accelerated Subspace Iteration (PASI)

Subspace iterations on $p(J^2_i)$

$p(w)$ is a band-pass filter for $w=\lambda$

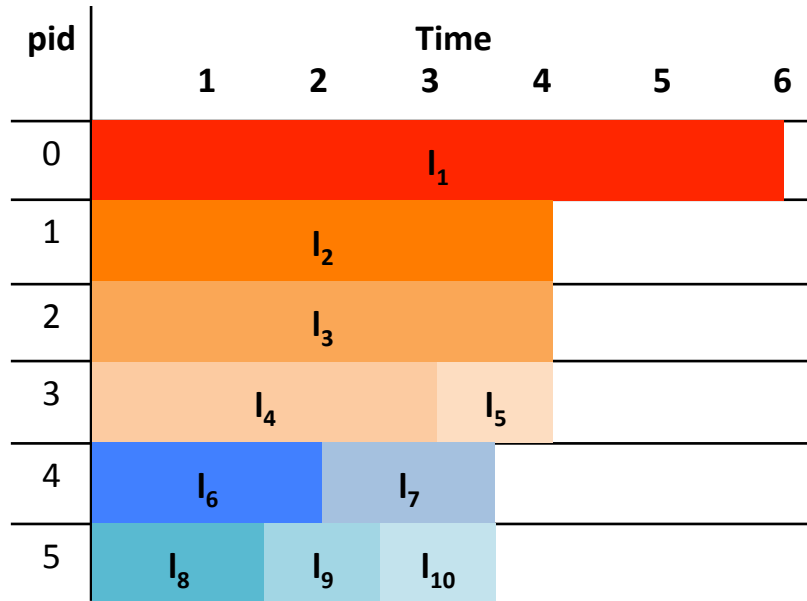
3 Shift-Invert Lanczos (SIL)

Lanczos iterations on $(J^2_i - \sigma I)^{-1}$, σ target shift near λ

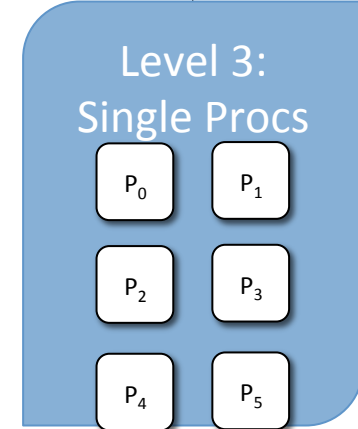
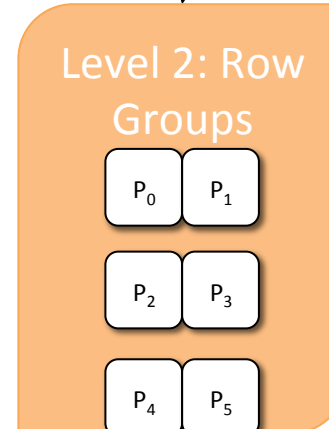
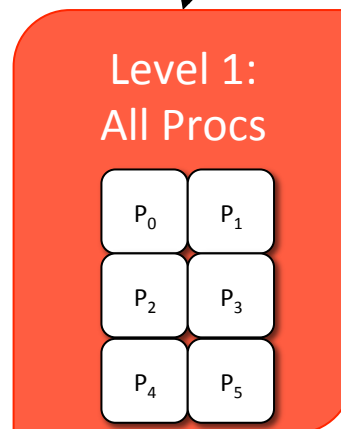
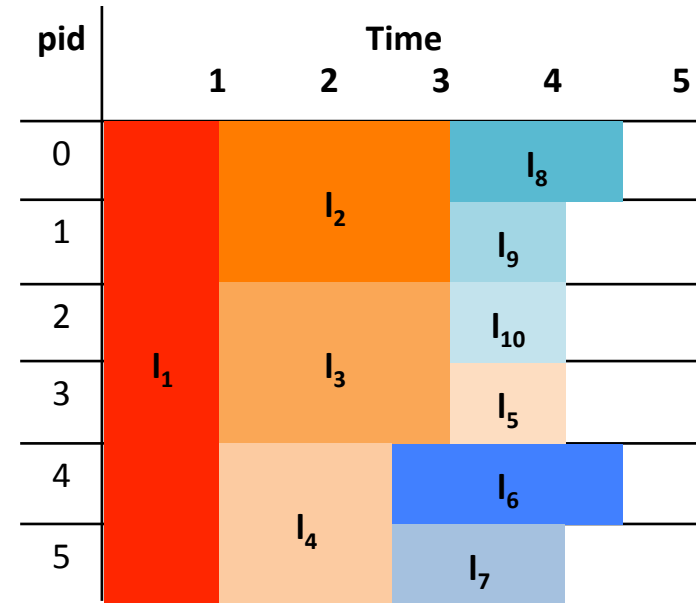


Nonzero pattern of J^2_i

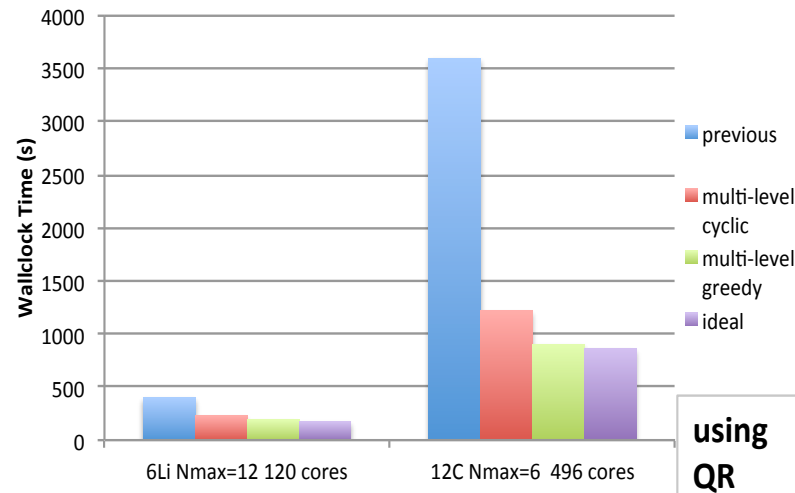
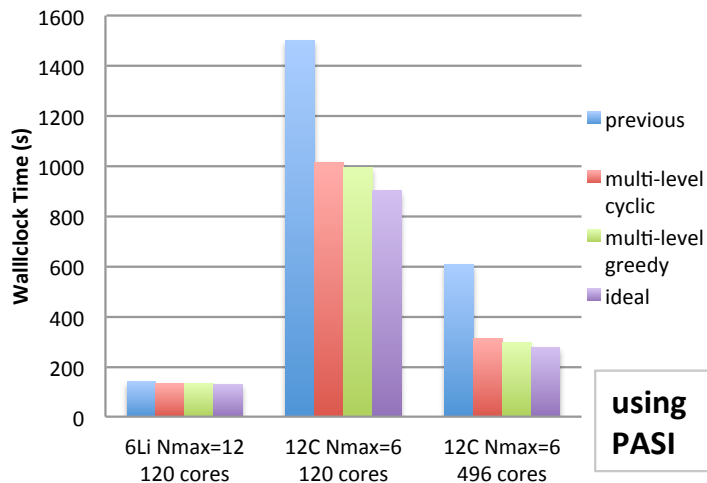
Load Balancing Null Space Computations



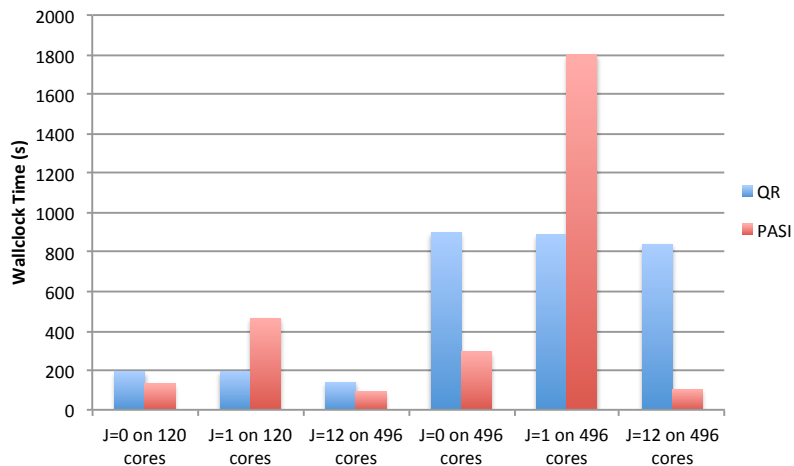
Multi-level Greedy Load Balancing



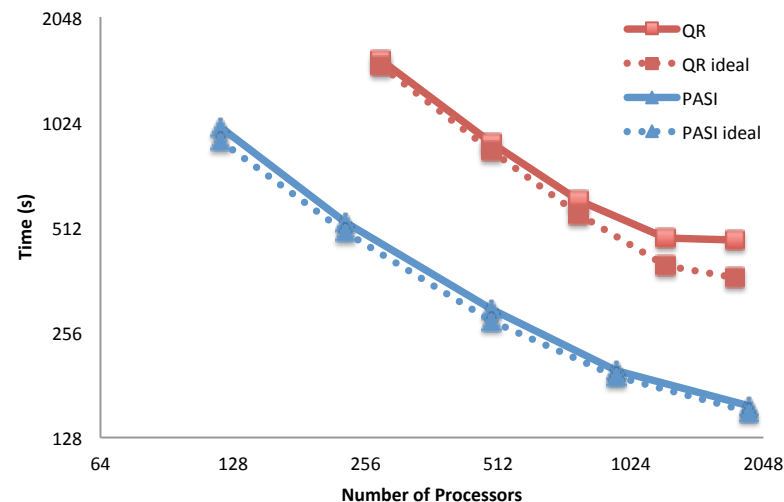
Previous implementation vs. Multi-level Greedy Load Balancing



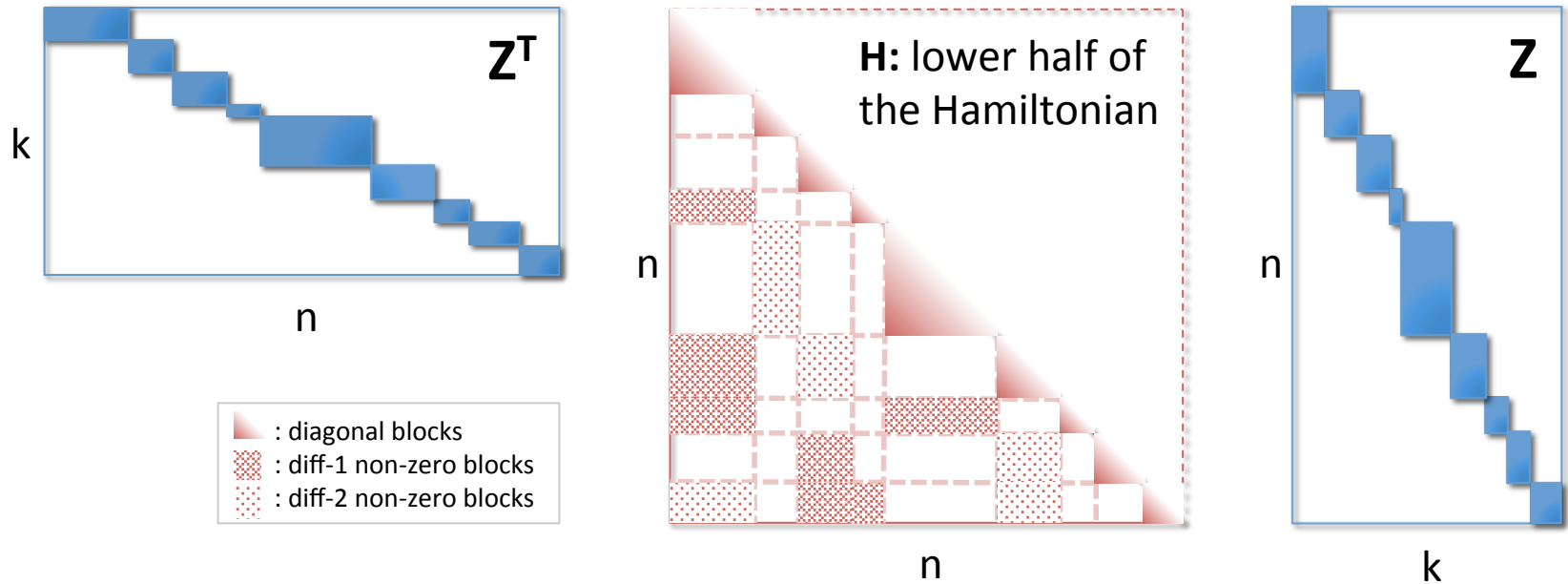
RQR vs. PASI



^{12}C , $N_{\max}=6$ Scaling

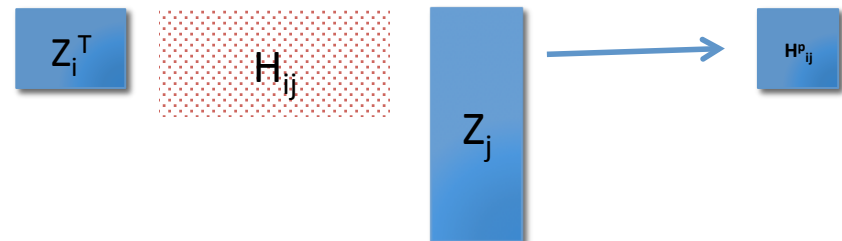


Subspace Projection of the Hamiltonian

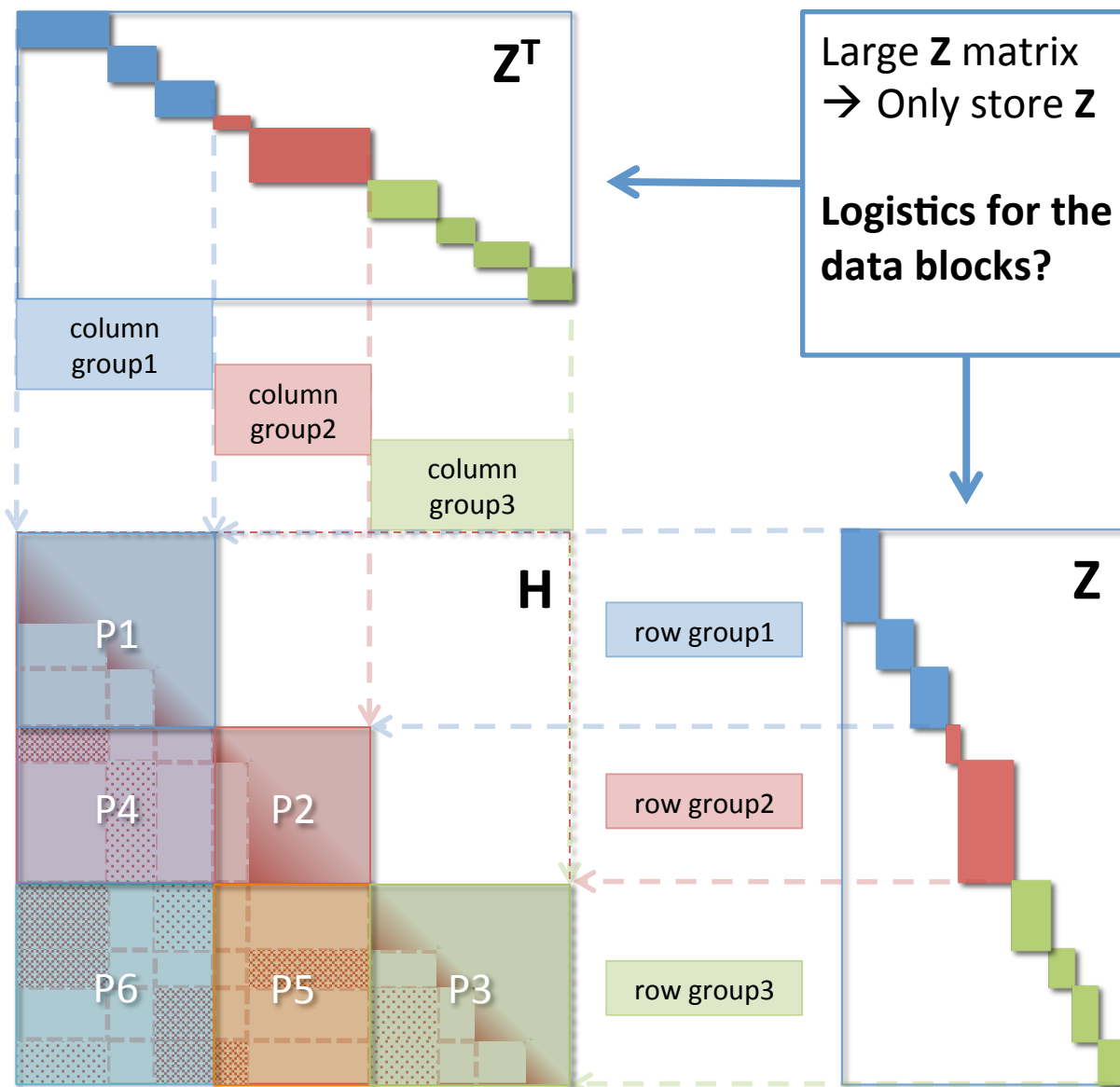


Each non-zero H_{ij} block defines a task:

1. construct H_{ij}
2. bring the data blocks, Z_i and Z_j
3. project block by block: $Z_i^T (H_{ij} Z_j)$



Problem Decomposition



Large Z matrix
→ Only store Z

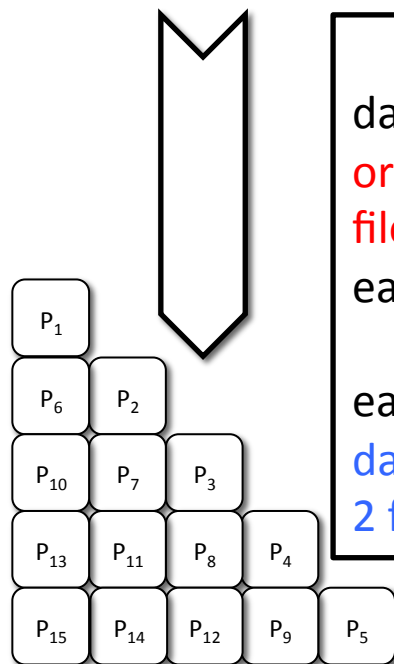
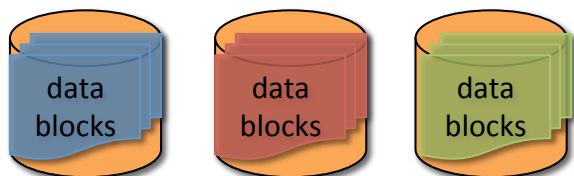
Logistics for the
data blocks?

Distribute
many-body
groups over
diagonal
processors

H constructed
on the fly!

Out-of-core vs. In-Core Approaches

I/O System

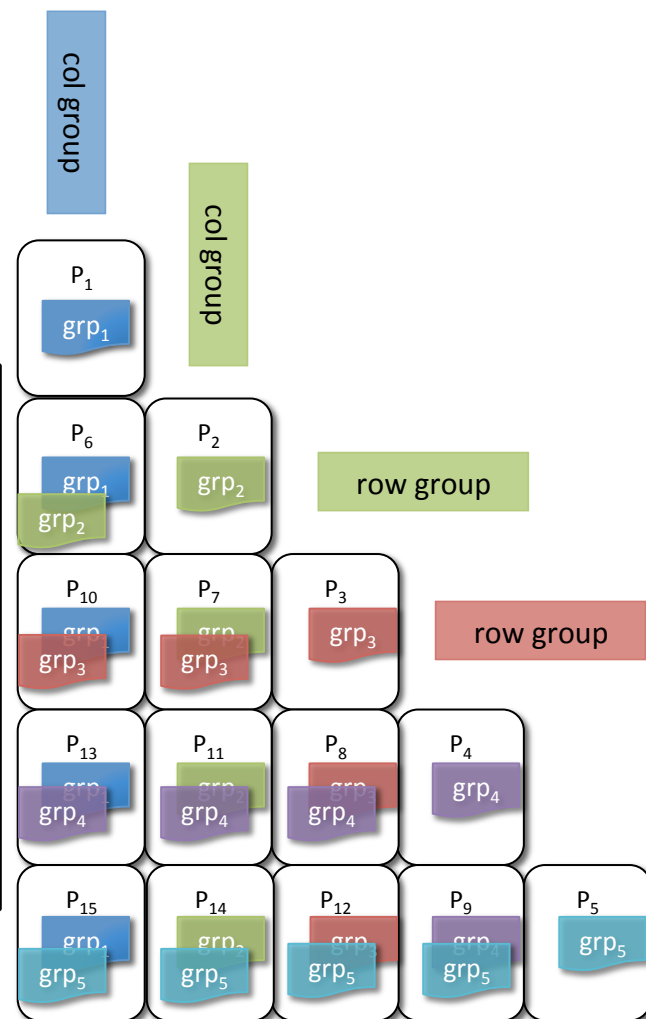


oocore
 data blocks
 organized into
 files → one for
 each diag proc

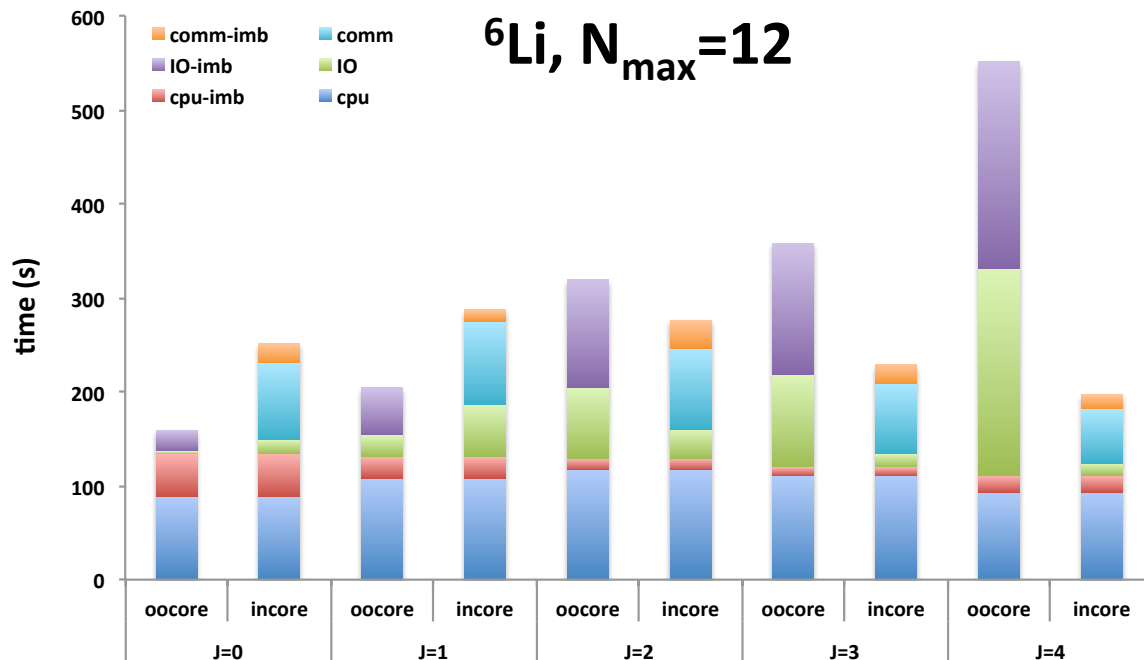
each proc reads
 data blocks from
 2 files (1 if diag)

incore
 data blocks
 distributed over
 row&col groups

One-sided MPI
 communication
 with bundling

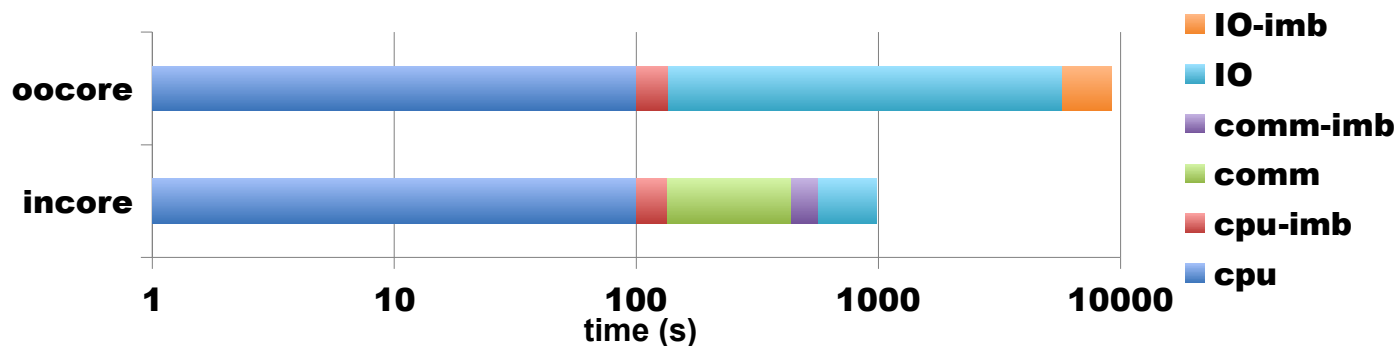


Out-of-core vs. In-Core Performance



${}^6\text{Li}, N_{\text{max}}=14$
J=3

Much larger problem, bigger gain!



Future Work

Better **load models** for **invariant** subspace calculations

Heuristics for **load balancing** in **subspace projection**
that also minimize **communication overheads**

Reduce **communication overheads** in **sparse matrix diagonalization** stage: both for M-scheme and J-scheme