A Scalable Diagonalization Framework for Tensor-Product Bitstring Selected Configuration Interaction

Enhua Xu

RIKEN Center for Computational Science
Computational Molecular Science Research Team
Team Principal: Takahito Nakajima

Outline

A. Introduction

B. Methodology

C. Results

D. Conclusions and Prospects

A. Introduction

Full Configuration Interaction (FCI)

In quantum chemistry, FCI provides the exact solution to the Schrödinger equation.

For N_α alpha and N_β beta electrons in M spatial orbitals, the total number of determinants is:

$$N_{\mathrm{FCI}} = \binom{M}{N_\alpha} \times \binom{M}{N_\beta}$$

The FCI wavefunction can be expressed as:

$$|\Psi_{\mathrm{FCI}}\rangle = \sum_{K=1}^{N_{\mathrm{FCI}}} c_K |D_K\rangle$$

where $|D_K\rangle$ denotes the K-th Slater determinant and c_K its coefficient.

Selected Configuration Interaction (SCI)

Due to the factorial growth of N_{FCI}, FCI rapidly becomes intractable as the system size increases.

In most chemical systems, only a small subset of determinants contributes significantly to the wavefunction.

This observation underlies selected configuration interaction (SCI) methods, which retain only those important determinants:

$$|\Psi_{\mathrm{FCI}}\rangle = \sum_{K=1}^{N_{\mathrm{FCI}}} c_K |D_K\rangle \longrightarrow |\Psi_{\mathrm{SCI}}\rangle = \sum_{K=1}^{n_{\mathrm{SCI}}} c_K |D_K\rangle$$

Developed SCI methods have been shown to provide highly compact wavefunctions, achieving near-FCI accuracy with only a small fraction of the FCI determinants for a wide range of molecular systems.

Algorithm Perspective

The core operation in CI diagonalization methods is a matrix-vector multiplication:

$$\mathbf{W} = H \cdot \mathbf{U}$$

H: Hamiltonian matrix (highly sparse and diagonally dominant)

U: eigenvector (the CI wavefunction)

$|\Psi_{\mathrm{FCI}}\rangle \rightarrow$ astronomically long vector

$|\Psi_{\mathrm{SCI}}\rangle \rightarrow$ much shorter vector

Including more determinants → improves the accuracy

Including more determinants → increases the length of the CI vector

However, most existing SCI implementations still employ replicated CI-vector storage. As a result, reported SCI calculations typically involve up to about 2 billion determinants.

Usable memory per node: ~24 GiB → $3.2*10^9$ determinants

On Fugaku, 158,976 nodes available → $5.1*10^{14}$ determinants

Fully distributed CI-vector storage!

Algorithmic Requirements

To enable extreme-scale SCI calculations, three key algorithmic requirements must be satisfied in the matrix-vector multiplication:

$$\mathbf{W} = H \cdot \mathbf{U}$$

H: on-the-fly Hamiltonian evaluation

U: fully distributed CI-vector storage

} Solve larger problems

Efficient Hamiltonian application → Solve problems faster

These requirements motivate us to reconsider: how determinants are organized in SCI calculations.

Tensor-Product Bitstring (TPB) Representation

Each determinant can be written as the tensor product of an α- and a β-bitstring:

$$|D_K\rangle = |S_w^\alpha\rangle \otimes |S_u^\beta\rangle$$

The FCI wavefunction can equivalently be written as:

$$|\Psi_{\mathrm{FCI}}\rangle = \sum_{w=1}^{L_\alpha} \sum_{u=1}^{L_\beta} c_{(w,u)} |S_w^\alpha\rangle \otimes |S_u^\beta\rangle$$

We refer to the representation as tensor-product bitstring (TPB) representation.

This α/β factorization has long been used in FCI algorithms to enable efficient Hamiltonian application on-the-fly with distributed CI-vector storage.

However, most SCI methods select determinants individually in a sparse and irregular manner.
→ the complete tensor-product form is no longer preserved.
→ how to organize the determinants?

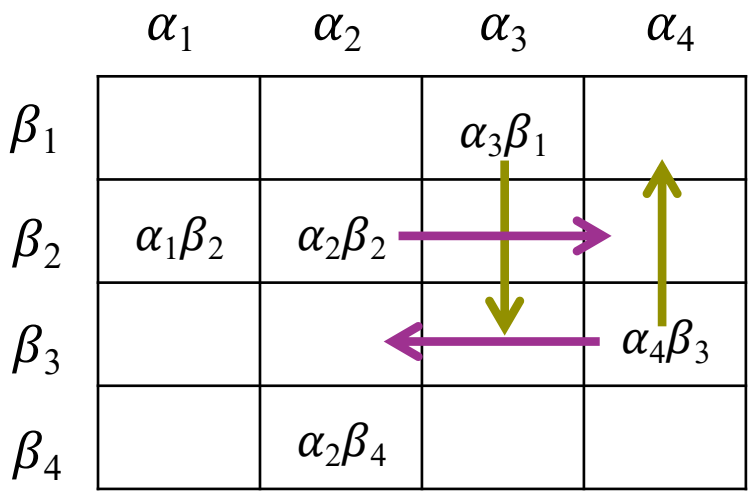
SCI Determinants in the TPB Representation

Consider the complete tensor-product closure: $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\} \otimes \{\beta_1, \beta_2, \beta_3, \beta_4\}$ which spans the full set of 16 determinants.

Suppose the selected determinants are: $\alpha_1\beta_2$, $\alpha_2\beta_2$, $\alpha_2\beta_4$, $\alpha_3\beta_1$, and $\alpha_4\beta_3$.

These determinants are stored in memory in a determinant-by-determinant manner.

However, if we embed them within the TPB representation, a structured bitstring-level indexing and connectivity pattern emerges naturally



reusable

horizontal: α-bitstring excitation (e.g. $\alpha_2 \leftrightarrow \alpha_4$)

vertical: β-bitstring excitation (e.g. $\beta_1 \leftrightarrow \beta_3$)

TPB Structure

We refer to this organizational pattern as the tensor-product bitstring (TPB) structure.

Within this organizing principle, fully distributed CI-vector storage and structured traversal enables efficient on-the-fly Hamiltonian application.

Building upon this structure, we develop the tensor-product bitstring SCI (TBSCI) approach.

Its algorithmic role will be discussed shortly.

B. Methodology

B1. Determinant organization and distributed CI-vector storage

B2. Distributed matrix-vector multiplication

B3. Efficient Hamiltonian computation

B4. MPI communication strategies

These four components are not independent modules, but form an integrated and tightly coupled framework.

B1, TPB Determinant Space

$$|\Psi_{\rm FCI}\rangle = \sum_{w=1}^{L_\alpha}\sum_{u=1}^{L_\beta} c_{(w,u)}|S_w^\alpha\rangle \otimes |S_u^\beta\rangle$$

In realistic systems, even the numbers of α- and β-bitstrings (L_α and L_β) can be astronomically large.

Suppose that a small set of important α- and β-bitstrings is identified, and we retain as: $\{S_w^\alpha \mid w = 1, 2, \ldots, l_\alpha\}$ and $\{S_u^\beta \mid u = 1, 2, \ldots, l_\beta\}$

The resulting tensor-product determinant set:

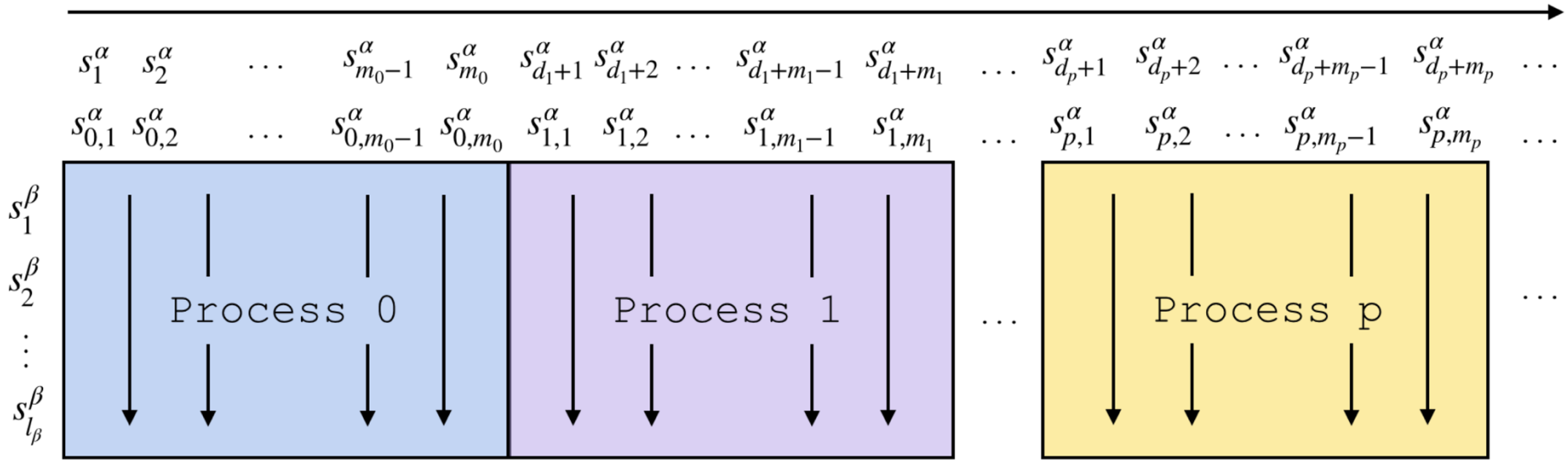
$$\mathcal{D}_{\rm TPB} = \left\{|S_w^\alpha\rangle \otimes |S_u^\beta\rangle \mid w = 1, \ldots, l_\alpha;\ u = 1, \ldots, l_\beta\right\}$$

which is referred to as the TPB determinant space.

Suppose all determinants in $\mathcal{D}_{\rm TPB}$ are retained in TBSCI calculations. Based on the "β first, then α" strategy, the global index K and the pair (w, u) are related by:

$$\begin{aligned} w &= \left\lfloor \frac{K-1}{l_\beta} \right\rfloor + 1, \quad u = \mathrm{MOD}(K-1, l_\beta) \\ K &= (w-1)l_\beta + u. \end{aligned}$$

B1, Distribute Storage of the CI Vector



Determinants are distributed across processes according to their α bitstring.

Denote m_p as the number of $\alpha-$bitstrings assigned to process p $(\sum_{p=0}^{N-1} m_p = l_\alpha)$, and d_p as the displacement index $(\sum_{i=0}^{p-1} m_i = d_p)$.

Define a segment as the subset of CI vector corresponding to all determinants sharing the same α bitstring. Here, each segment has a uniform length equal to l_β.

B1, TBSCI Determinant Space

TBSCI selects a determinant subset within $\mathcal{D}_{\text{TPB}}$:

$$\mathcal{D}_{\text{TBSCI}} = \left\{ |S_w^\alpha\rangle \otimes |S_u^\beta\rangle \mid w = 1, \ldots, l_\alpha;\ u \in \mathcal{I}_w^{(\beta)} \right\}$$

which is referred to as the TBSCI determinant space.

$\mathcal{I}_w^{(\beta)} \subseteq \{1, \ldots, l_\beta\}$ denotes the index set of β-bitstrings paired with the ω-th α-bitstring. It may range from the empty set to the full index set $\{1, \ldots, l_\beta\}$.

When $\mathcal{I}_w^{(\beta)} = \{1, \ldots, l_\beta\}$ for all ω, $\mathcal{D}_{\text{TBSCI}} = \mathcal{D}_{\text{TPB}}$.

Segment lengths are no longer uniform

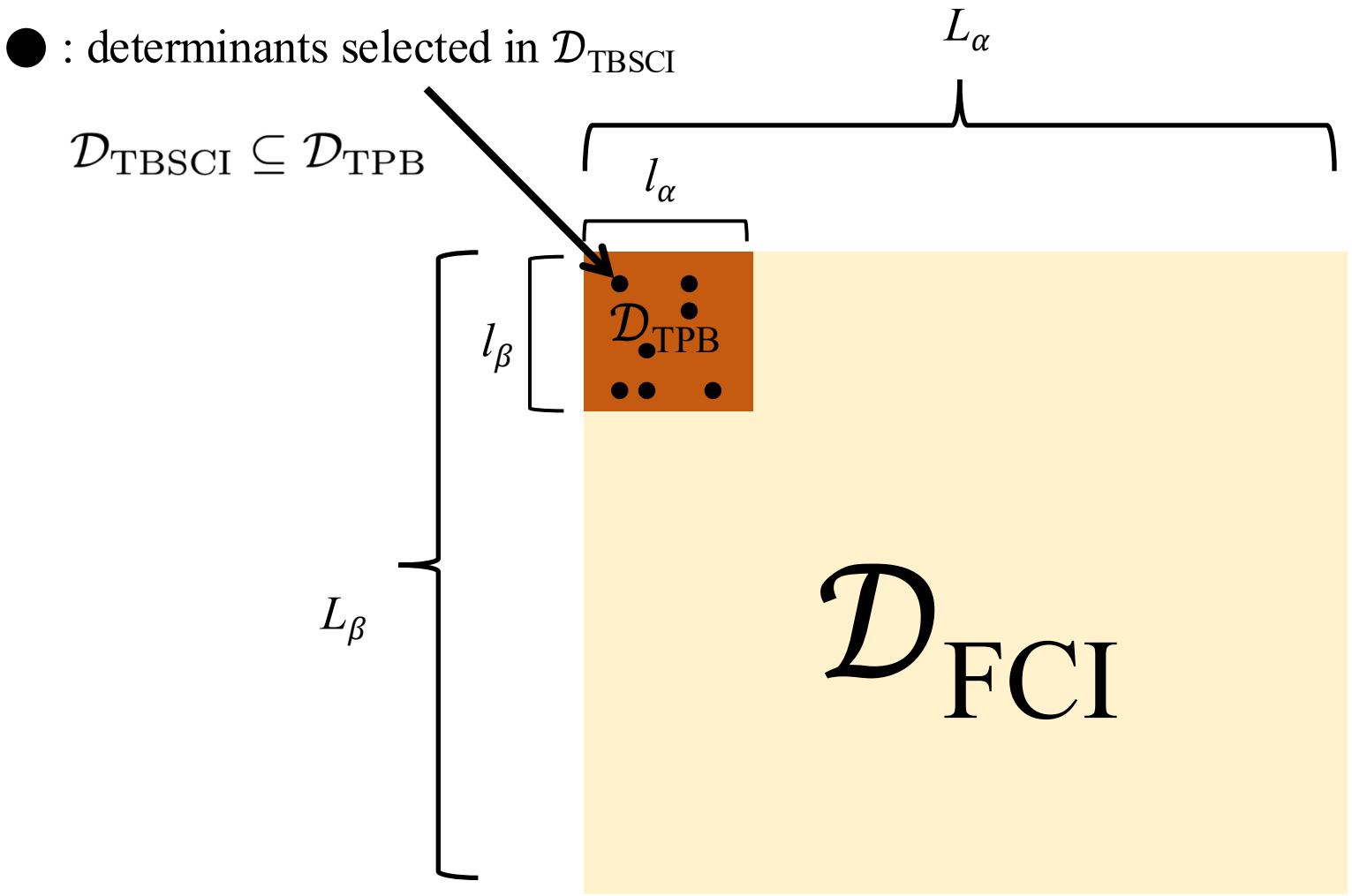
adjustments to the storage strategy

Replicated metadata for each segment:
– global offset
– owning process
– segment length

Replicated metadata for each process:
– global offset
– total length of its CI-vector portion

Locally stored data for each segment:

$\mathcal{I}_w^{(\beta)}$



Goal of this work

To assess the intrinsic compactness of the TPB representation, $\mathcal{D}_{\mathrm{TBSCI}}$ retains all symmetry-allowed determinants within $\mathcal{D}_{\mathrm{TPB}}$.

B2, Distributed Matrix-Vector Multiplication

In Davidson diagonalization, the core step is:

$$\mathbf{W} = H \cdot \mathbf{U}$$

After distributing the CI vector, each process computes its local contribution $\mathbf{W}_p$ as:

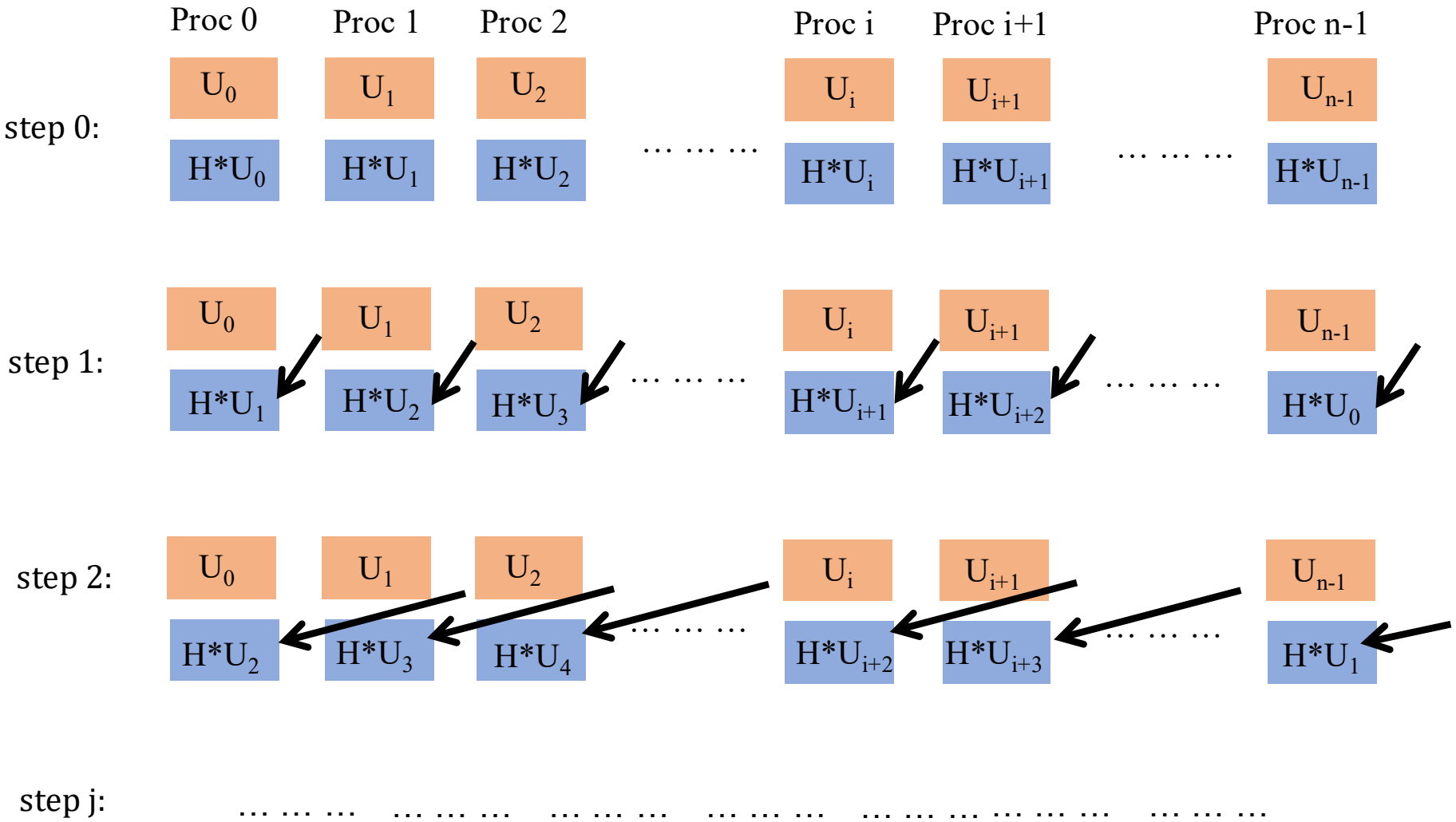
$$\mathbf{W}_p = \sum_q H_{p,q} \cdot \mathbf{U}_q$$

Each fetch-and-compute operation $\mathbf{W}_p \mathrel{+}= H_{p,q} \cdot \mathbf{U}_q$ is treated as one step.

Suppose there are *N* processes, a single Davidson iteration in principle involves *N* steps.

During the *N* steps of a single Davidson iteration, each node operates independently without global synchronization.

B2, Asynchronous Hamiltonian Application



B3, Efficient Hamiltonian Computation

$$\langle D_{(w,u)}|H|D_{(x,v)}\rangle \longrightarrow \langle S_w^\alpha|\langle S_u^\beta|H|S_x^\alpha\rangle|S_v^\beta\rangle$$

The Hamiltonian matrix is highly sparse!

$$\langle S_w^\alpha|\langle S_u^\beta|H|S_x^\alpha\rangle|S_v^\beta\rangle \neq 0 \longrightarrow \mathrm{DIFF}(S_w^\alpha, S_x^\alpha) + \mathrm{DIFF}(S_u^\beta, S_v^\beta) \leq 2$$

$$\mathrm{DIFF}(S_w^\alpha, S_x^\alpha) = 2,\ \mathrm{DIFF}(S_w^\beta, S_x^\beta) = 0 \qquad [2,0]\ \text{term}$$

$$\mathrm{DIFF}(S_w^\alpha, S_x^\alpha) = 1,\ \mathrm{DIFF}(S_w^\beta, S_x^\beta) = 1 \qquad [1,1]\ \text{term}$$

$$\longrightarrow \mathrm{DIFF}(S_w^\alpha, S_x^\alpha) = 1,\ \mathrm{DIFF}(S_w^\beta, S_x^\beta) = 0 \longrightarrow [1,0]\ \text{term}$$

$$\mathrm{DIFF}(S_w^\alpha, S_x^\alpha) = 0,\ \mathrm{DIFF}(S_w^\beta, S_x^\beta) = 2 \qquad [0,2]\ \text{term}$$

$$\mathrm{DIFF}(S_w^\alpha, S_x^\alpha) = 0,\ \mathrm{DIFF}(S_w^\beta, S_x^\beta) = 1 \qquad [0,1]\ \text{term}$$

$$\mathrm{DIFF}(S_w^\alpha, S_x^\alpha) = 0,\ \mathrm{DIFF}(S_w^\beta, S_x^\beta) = 0 \qquad [0,0]\ \text{term}$$

B3, TPB Structure: Implications for the Algorithm

Although TBSCI retains only a subset of determinants in $\mathcal{D}_{\mathrm{TPB}}$, the tensor-product indexing structure of the TPB representation is preserved.

As a consequence, for a fixed pair (S_w^α, S_x^α) , all β-side excitation candidates can be generated within the selected β-bitstring set $\{S^\beta\}$, and then filtered by the index sets $\mathcal{I}_w^{(\beta)}$ and $\mathcal{I}_x^{(\beta)}$.

↓

Excitation relations among β-bitstrings can be precomputed once and reused across all α-bitstring pairs.

↓

Construct the BETA_SINGLE_LINK and BETA_DOUBLE_LINK arrays, which store single and double excitation connectivity within $\{S^\beta\}$.

Algorithm 1: Local computation of matrix elements on process p at step q.

Data: Current process p, current step q; BETA_SINGLE_LINK and BETA_DOUBLE_LINK lists

```
1  for S_w^α ← S_{p,1}^α to S_{p,m_p}^α do
2      for S_x^α ← S_{q,1}^α to S_{q,m_q}^α do
3          case ← DIFF(S_w^α, S_x^α)
4          if case = 2 then
5              foreach S_v^β ∈ I_x^(β) do
6                  S_u^β ← S_v^β
7                  If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [2,0] term
8              end foreach
9          else if case = 1 then
10             foreach S_v^β ∈ I_x^(β) do
11                 foreach S_u^β ∈ BETA_SINGLE_LINK(:, S_v^β) do
12                     If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [1,1] term
13                 end foreach
14                 S_u^β ← S_v^β
15                 If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [1,0] term
16             end foreach
17         else if case = 0 then
18             foreach S_v^β ∈ I_x^(β) do
19                 foreach S_u^β ∈ BETA_SINGLE_LINK(:, S_v^β) do
20                     If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [0,1] term
21                 end foreach
22                 S_u^β ← S_v^β
23                 If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [0,0] term
24             end foreach
25         end if
26     end for
27     S_x^α ← S_w^α
28     foreach S_v^β ∈ the q-th batch of {S^β} do
29         foreach S_u^β ∈ BETA_DOUBLE_LINK(:, S_v^β) do
30             If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [0,2] term
31         end foreach
32     end foreach
33 end for
```

Firstly, loop over two sets of α bitstrings

Pre-computed and stored arrays

In one iteration, process p needs to fetch BETA_DOUBLE_LINK data from the target process q only once.

B3, Computational Scaling

In the framework of TBSCI, the **sparsity** of single and double excitations can be roughly estimated to be $\sqrt{N_{\mathrm{SCI}}/N_{\mathrm{FCI}}}$.

Scaling of [2,0] and [0,2]: $N_{\mathrm{SCI}} \cdot N_{\mathrm{occ}}^2 \cdot N_{\mathrm{vir}}^2 \cdot \sqrt{N_{\mathrm{SCI}}/N_{\mathrm{FCI}}}$

Scaling of [1,1] term: $N_{\mathrm{SCI}} \cdot N_{\mathrm{occ}}^2 \cdot N_{\mathrm{vir}}^2 \cdot N_{\mathrm{SCI}}/N_{\mathrm{FCI}}$

Method	Overall scaling	Dominant terms
SCI	$N_{\mathrm{SCI}} \cdot N_{\mathrm{occ}}^2 \cdot N_{\mathrm{vir}}^2 \cdot \sqrt{N_{\mathrm{SCI}}/N_{\mathrm{FCI}}}$	$[2,0],\ [0,2]$
FCI	$N_{\mathrm{FCI}} \cdot N_{\mathrm{occ}}^2 \cdot N_{\mathrm{vir}}^2$	$[2,0],\ [1,1],\ [0,2]$

B3, Computational Scaling of Conventional SCI

In conventional SCI implementations, the computational bottleneck is the mixed-spin [1, 1] term:

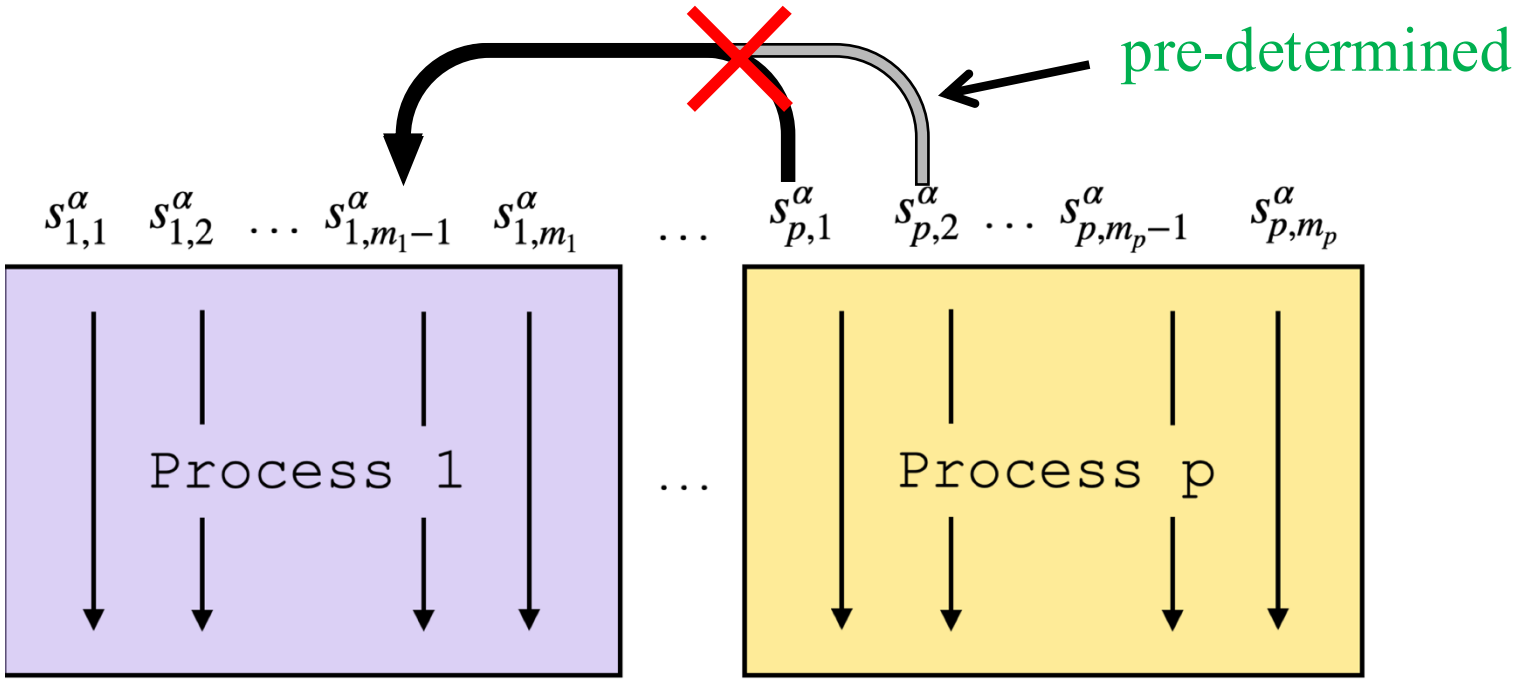
for $S_w^\alpha \leftarrow S_{p,1}^\alpha$ **to** S_{p,m_p}^α **do**
  for $S_x^\alpha \leftarrow S_{q,1}^\alpha$ **to** S_{q,m_q}^α **do**
    case $\leftarrow$ **DIFF**(S_w^α, S_x^α)
    if case $= 1$ **then**
      foreach $S_v^\beta \in \mathcal{I}_x^{(\beta)}$ **do**
        foreach $S_u^\beta \in \mathcal{I}_w^{(\beta)}$ **do**
          calculate $\langle S_w^\alpha | \langle S_u^\beta | \hat{H} | S_x^\alpha \rangle | S_v^\beta \rangle$ as [1,1] term
        end foreach
      end foreach

Explicit enumeration of determinant pairs

Computational scaling: $N_{\mathrm{SCI}}^{1.5} N_{\mathrm{occ}} N_{\mathrm{vir}} \sqrt{N_{\mathrm{SCI}}/N_{\mathrm{FCI}}}$

B4, MPI Communication Optimized Strategies

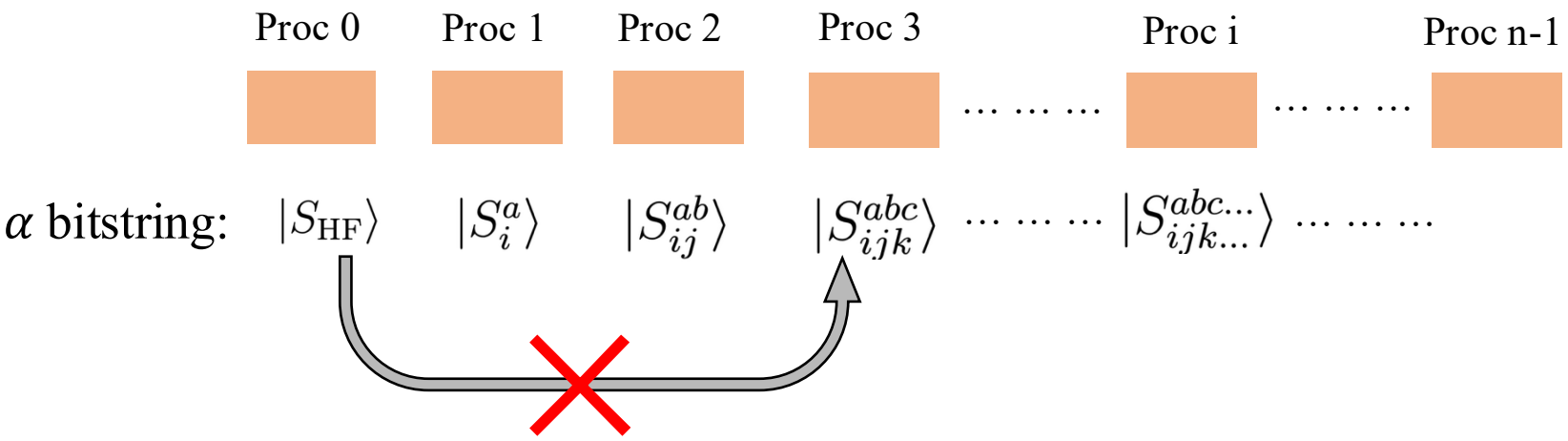
1. Avoid Unnecessary MPI Data Transfers



The overall MPI communication volume: N_{SCI}

B4, MPI Communication Optimized Strategies

2. Minimizing Long-Distance Communication in MPI



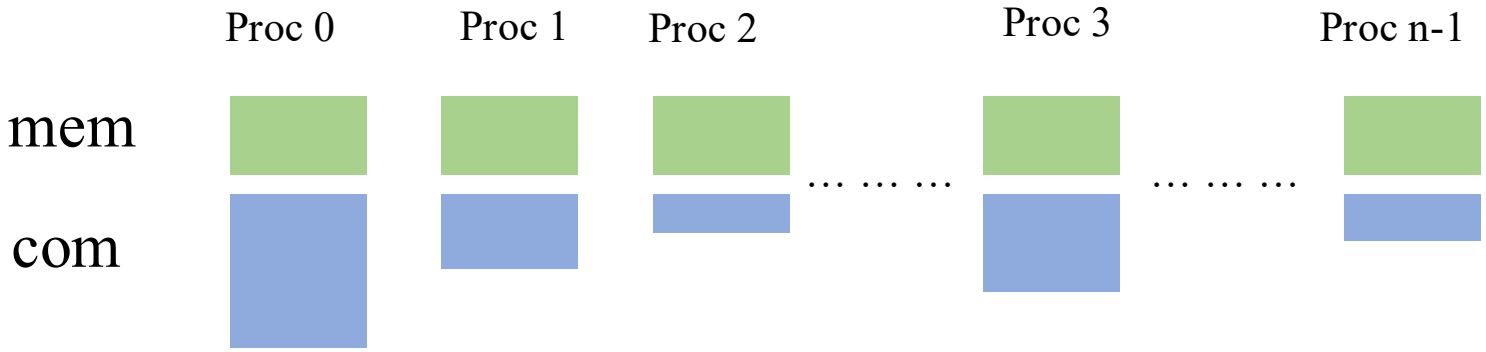
The communication between distant nodes is significantly reduced because the excitation differences between their assigned α bitstrings are more likely to be all greater than 2, making communication unnecessary.

B4, MPI Communication Optimized Strategies

3. Achieving Overall Balance in MPI

strict memory usage balance

severe computational load imbalance

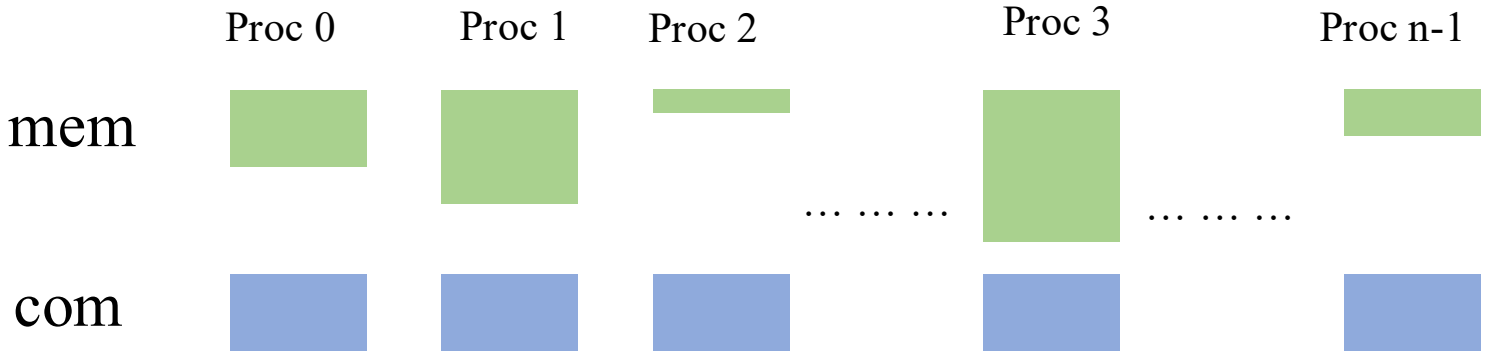


B4, MPI Communication Optimized Strategies

3. Achieving Overall Balance in MPI

strict computational load balance

highly uneven memory distribution



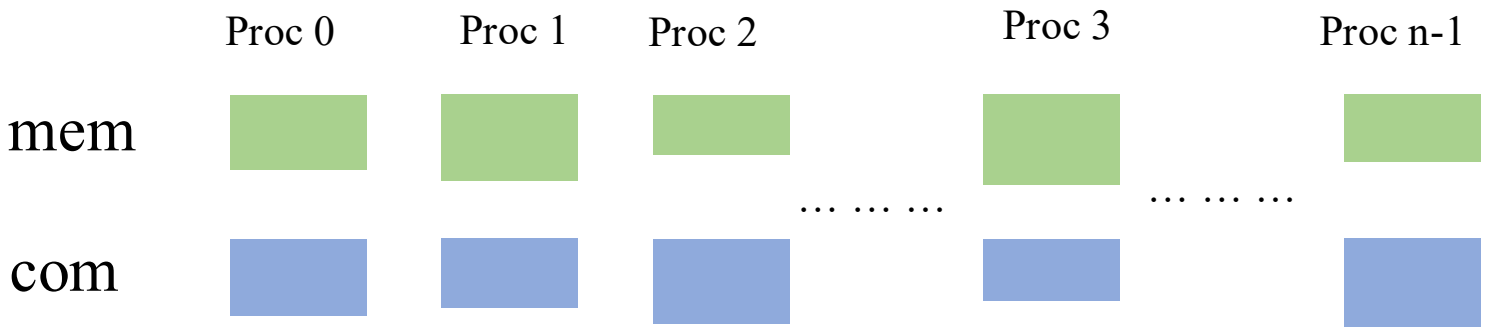
B4, MPI Communication Optimized Strategies

3. Achieving Overall Balance in MPI

Memory usage balance

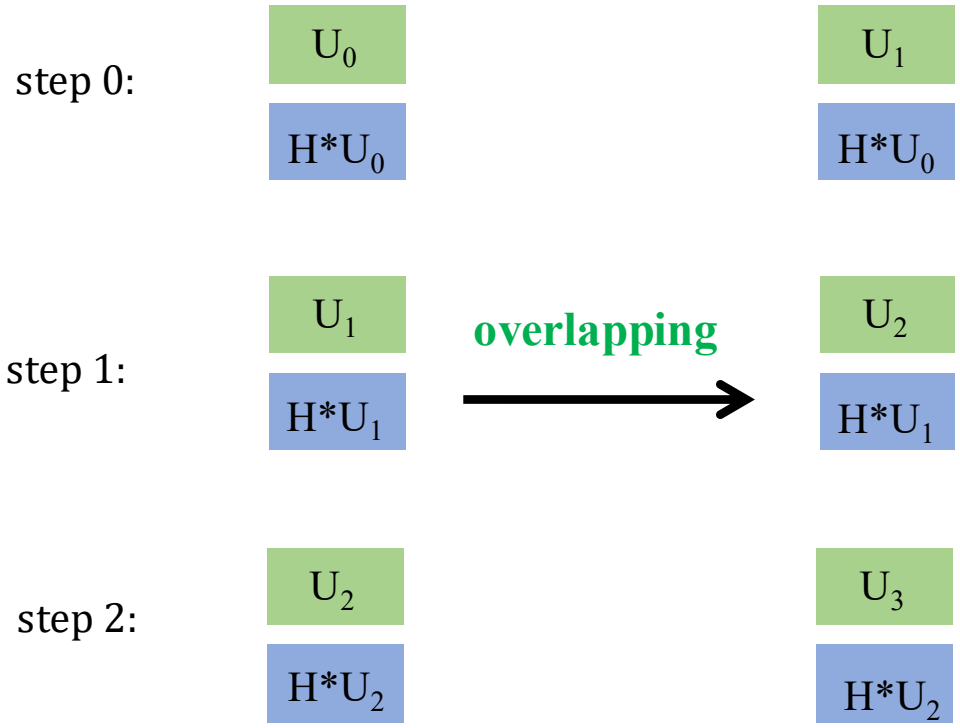
Computational load balance

} Overall balance



B4, MPI Communication Optimized Strategies

4. Overlapping Computation and Data Transfer to hide MPI Latency



B4, MPI Communication Optimized Strategies

4. Overlapping Computation and Data Transfer to hide MPI Latency

Let t_{cal} and t_{data} represent the computation time and the data retrieval time of one step, respectively.

If $t_{\text{data}} > t_{\text{cal}}$, we define the delay time to the wall time as: $t_{\text{delay}} = t_{\text{data}} - t_{\text{cal}}$

If $t_{\text{data}} \leq t_{\text{cal}}$, then $t_{\text{delay}} = 0$ even if the MPI communication congestion occurs!

We define T_{delay} as the sum of t_{delay} over N steps in one Davidson iteration of a process.

B4, MPI Communication Optimized Strategies

5. Calculation Reassignment

Parts of the calculation have higher data requirements.

Steps with low computational load produce short t_{cal}, making $t_{delay} > 0$ more likely.

Therefore, we assign [0,2] tasks to such steps as a reservoir to absorb potential t_{delay}.

High Data Task

```
for S_w^α ← S_{p,1}^α to S_{p,m_p}^α do
  for S_x^α ← S_{q,1}^α to S_{q,m_q}^α do
    case ← DIFF(S_w^α, S_x^α)
    if case = 2 then
      foreach S_v^β ∈ I_x^(β) do
        S_u^β ← S_v^β
        If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [2,0] term
      end foreach
    else if case = 1 then
      foreach S_v^β ∈ I_x^(β) do
        foreach S_u^β ∈ BETA_SINGLE_LINK(:, S_v^β) do
          If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [1,1] term
        end foreach
        S_u^β ← S_v^β
        If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [1,0] term
      end foreach
    else if case = 0 then
      foreach S_v^β ∈ I_x^(β) do
        foreach S_u^β ∈ BETA_SINGLE_LINK(:, S_v^β) do
          If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [0,1] term
        end foreach
        S_u^β ← S_v^β
        If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [0,0] term
      end foreach
    end if
  end for
```

Low Data Task

```
S_x^α ← S_w^α
foreach S_v^β ∈ the q-th batch of {S^β} do
  foreach S_u^β ∈ BETA_DOUBLE_LINK(:, S_v^β) do
    If S_u^β ∈ I_w^(β), calculate ⟨S_w^α|⟨S_u^β|Ĥ|S_x^α⟩|S_v^β⟩ as [0,2] term
  end foreach
end foreach
```

B4, MPI Communication Optimized Strategies

6. Odd/even processes fetch in opposite orders

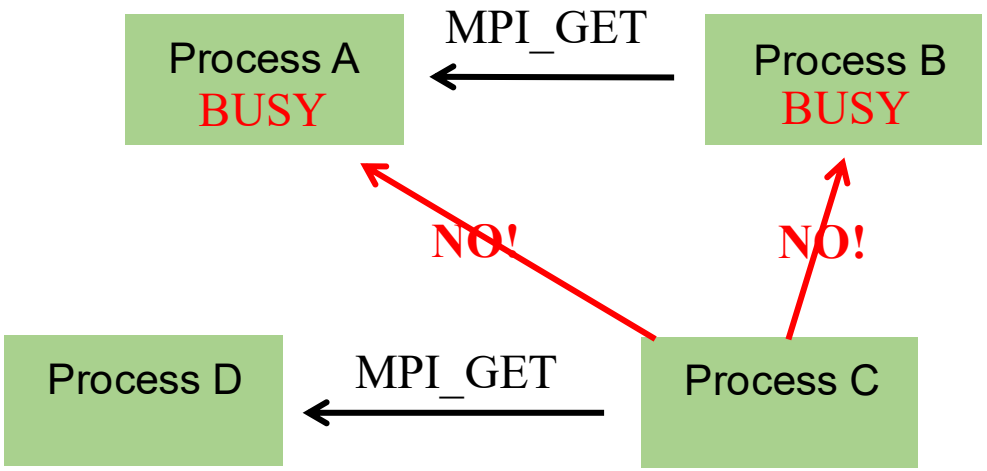
Odd-numbered processes fetch data :

Even-numbered processes fetch data：

to reduce the concentration of the MPI communication peaks

B4, MPI Communication Optimized Strategies

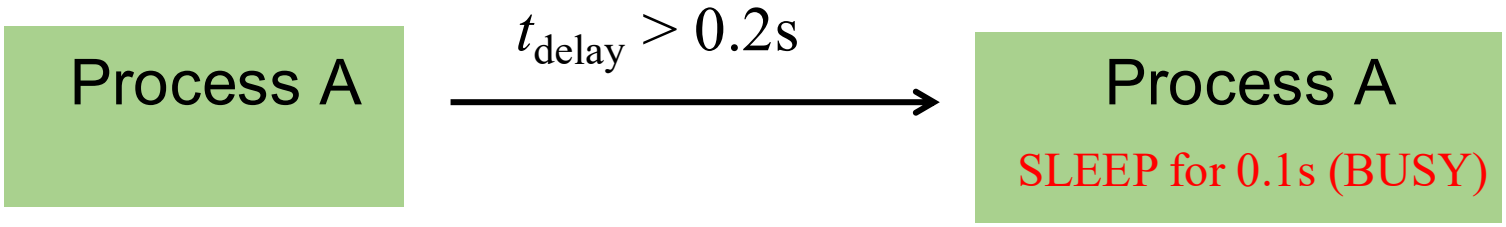
7. Check-if-busy dynamical scheduling



B4, MPI Communication Optimized Strategies

8. SLEEP Strategy under Severe MPI Congestion

MPI congestion behaves like a traffic jam: once congestion occurs, it induces blocking behavior which in turn worsens further congestion.



sleep time is included in t_{delay}.

C. Results

C1. Scalability and Parallel Efficiency

C2. Compactness of the TPB representation from SCI-derived bitstrings

C1, Scalability and Parallel Efficiency

To assess the parallel performance of the TBSCI eigensolver, we deliberately benchmark the code using FCI workloads.

For the same determinant count, FCI workloads involve:
- substantially heavier local Hamiltonian evaluations
- more intensive inter-process MPI communication than sparse SCI.

Large-scale FCI calculation therefore provides a particularly stringent stress test for the distributed CI-vector architecture and the associated MPI communication strategies.

C1, Toward FugakuNEXT

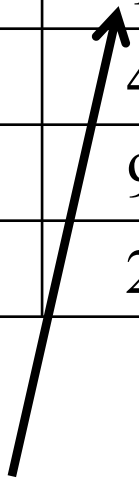
To enable GPU-oriented CI calculations, we also tested a mixed-precision (FP32) execution mode.

The Hamiltonian matrix-vector multiplication is performed in single precision, while other operations in Davidson diagonalization are still carried out in double precision.

This mixed-precision implementation is expected to be important for FugakuNEXT.

C1, FCI Benchmark Systems

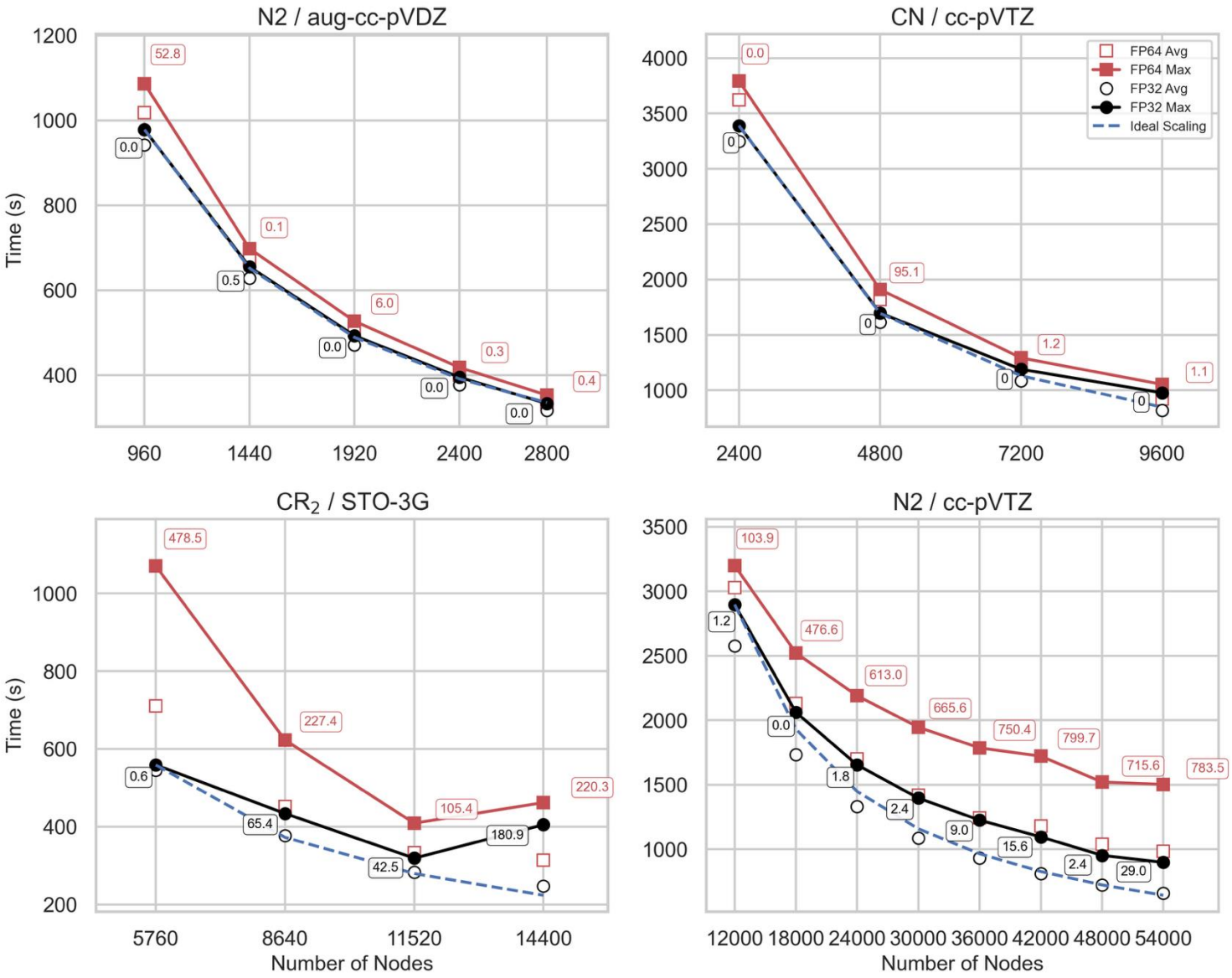
System	Active space	Molecular symmetry	FCI dimension
N_2 (aug-cc-pVDZ)	CAS(10e, 44o)	C_{2v}	1.47E11
CN (cc-pVTZ)	CAS(9e, 58o)	C_{2v}	4.86E11
Cr_2 (STO-3G)	CAS(24e, 24o)	D_{2h}	9.14E11
N_2 (cc-pVTZ)	CAS(10e, 58o)	C_{2v}	2.62E12



We run these FCI calculations using our TBSCI eigensolver. Even the smallest system already exceeds the scale of previously reported SCI calculations.

System	Nodes	FP64				FP32			
		Avg.	Max.	T_{delay}	NHs	Avg.	Max.	T_{delay}	NHs
N_2 (aug-cc-pVDZ)	960	1018	1086	52.8	290	942	978	0	261
	1440	670	698	0.1	279	628	655	0.5	262
	1920	505	527	6.0	281	471	493	0	263
	2400	400	418	0.3	279	377	395	0	263
	2880	337	353	0.4	282	317	333	0	264
CN (cc-pVTZ)	2400	3624	3793	0	2529	3247	3390	0	2260
	4800	1821	1910	95.1	2547	1615	1699	0	2265
	7200	1214	1292	1.2	2584	1084	1189	0	2378
	9600	920	1053	1.1	2808	819	977	0	2605
Cr_2 (STO-3G)	5760	711	1071	478.5	1714	545	559	0.6	894
	8640	452	623	227.4	1495	377	434	65.4	1042
	11520	333	409	105.4	1309	283	319	42.5	1021
	14400	314	462	220.3	1848	247	405	180.9	1620
N_2 (cc-pVTZ)	12000	3029	3199	103.9	10663	2577	2897	1.2	9657
	18000	2129	2522	476.6	12610	1734	2062	0	10310
	24000	1700	2190	613.0	14600	1333	1654	1.8	11027
	30000	1419	1946	665.6	16217	1087	1397	2.4	11642
	36000	1243	1786	750.4	17860	933	1226	9	12260
	42000	1180	1722	799.7	20090	811	1096	15.6	12787
	48000	1040	1521	715.6	20289	723	953	2.4	12693
	54000	986	1503	783.5	22545	661	899	29.0	13485

C1, Strong Scaling Tests



C2, SCI-Derived Bitstrings

DICE

↓

selected determinants

↓

SCI wavefunction

↓

bitstring weights

↓

important bitstring

Candidate determinants are generated using DICE.

Determinants with estimated amplitudes larger than a threshold ϑ are selected for diagonalization. In the present work, we set $\vartheta = 10^{-5}$.

After normalizing the SCI wavefunction, the weights of α- and β-bitstrings are evaluated and ranked.

Important α- and β-bitstrings are selected using a threshold δ relative to the corresponding HF α- and β-bitstrings.

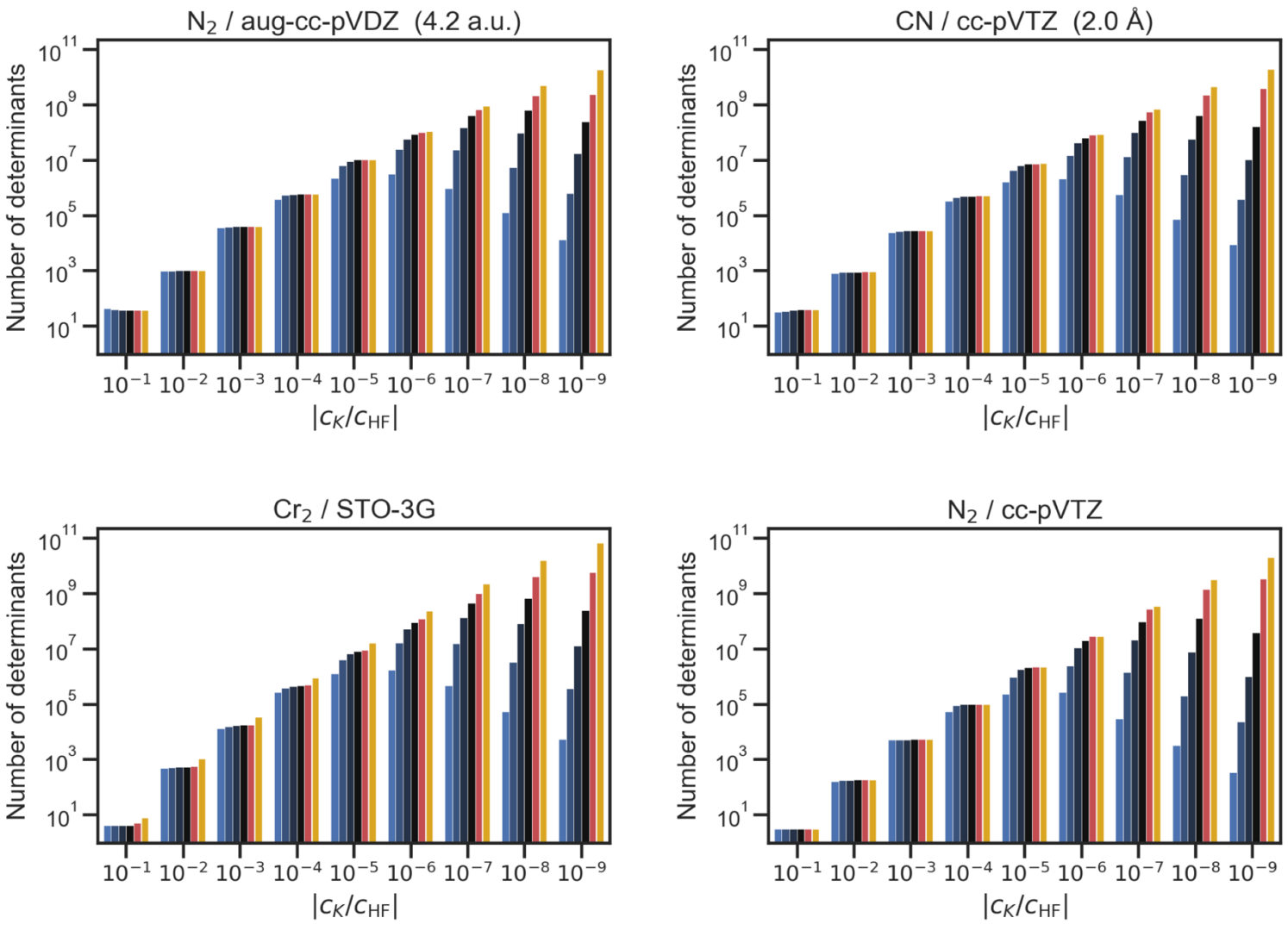
C2, N_2 (aug-cc-pVDZ)

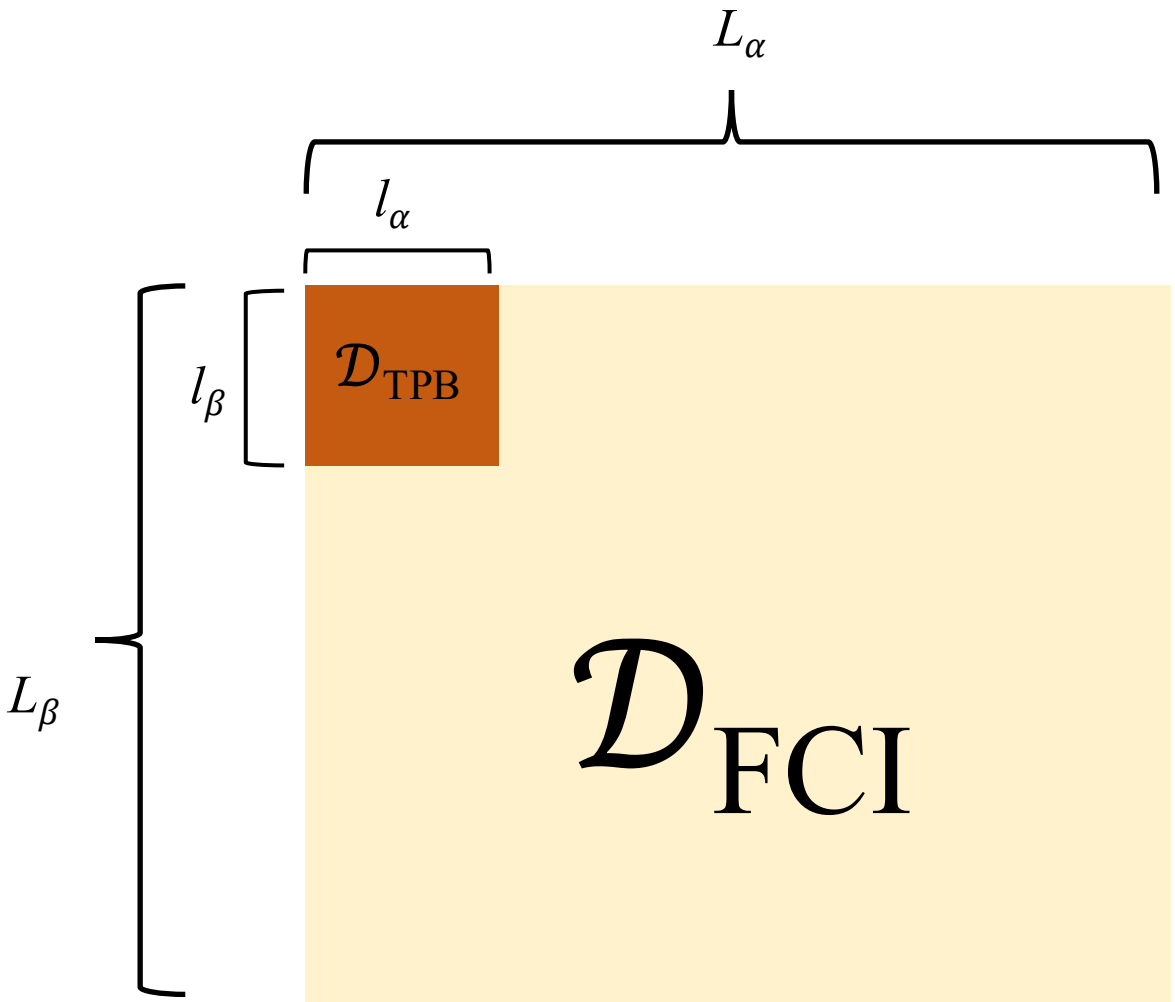
δ	$N_{\text{bitstrings}}$	N_{TBSCI}	$N_{\text{TBSCI}}/N_{\text{FCI}}$	$E_{\text{TBSCI}} - E_{\text{FCI}}$ (mH)	(nodes, seconds)
	$R = 2.068$ a.u.	$E_{\text{HF}} = -108.961045$		$E_{\text{FCI}} = -109.295263$	
1×10^{-6}	1,523	572,235	0.00039%	14.591	(12, 0.47)
1×10^{-7}	5,480	4,810,418	0.0033%	5.296	(12, 5.0)
1×10^{-8}	15,776	38,387,174	0.026%	1.616	(12, 49.3)
1×10^{-9}	38,620	228,641,788	0.16%	0.475	(12, 370.1)
0	124,480	2,872,138,682	1.95%	0.088	(36, 2111.1
	$R = 2.4$ a.u.	$E_{\text{HF}} = -108.875323$		$E_{\text{FCI}} = -109.258928$	
1×10^{-6}	1,870	742,908	0.00050%	15.249	(12, 0.70)
1×10^{-7}	6,375	6,480,851	0.0044%	5.386	(12, 8.1)
1×10^{-8}	17,853	48,556,961	0.033%	1.682	(12, 75.0)
1×10^{-9}	42,647	272,638,925	0.18%	0.512	(12, 545.7)
0	142,390	3,354,713,882	2.28%	0.081	(36, 3197.5)
	$R = 2.7$ a.u.	$E_{\text{HF}} = -108.748038$		$E_{\text{FCI}} = -109.182160$	
1×10^{-6}	2,240	1,008,784	0.00068%	15.725	(12, 1.2)
1×10^{-7}	7,441	8,705,921	0.0059%	5.534	(12, 14.1)
1×10^{-8}	20,242	61,107,582	0.041%	1.739	(12, 122.4)
1×10^{-9}	47,918	336,845,184	0.23%	0.525	(12, 891.2)
0	157,027	3,842,317,677	2.61%	0.080	(36, 4993.0)
	$R = 3.0$ a.u.	$E_{\text{HF}} = -108.618929$		$E_{\text{FCI}} = -109.108790$	
1×10^{-6}	2,748	1,444,334	0.00098%	16.239	(12, 2.3)
1×10^{-7}	9,105	12,703,807	0.0086%	5.520	(12, 25.9)
1×10^{-8}	24,013	83,584,179	0.057%	1.737	(12, 226.2)
1×10^{-9}	55,410	437,097,908	0.30%	0.531	(12, 1568.1)
0	173,593	4,454,632,169	3.02%	0.081	(36, 8005.7)
	$R = 3.6$ a.u.	$E_{\text{HF}} = -108.401280$		$E_{\text{FCI}} = -109.016333$	
1×10^{-6}	4,780	3,618,030	0.0025%	14.391	(12, 11.1)
1×10^{-7}	14,058	27,870,742	0.019%	4.996	(12, 117.2)
1×10^{-8}	36,022	175,911,036	0.12%	1.503	(12, 994.6)
1×10^{-9}	79,476	840,877,816	0.57%	0.437	(12, 3153.6)
0	206,186	5,761,751,654	3.91%	0.078	(48, 15262.6)
	$R = 4.2$ a.u.	$E_{\text{HF}} = -108.242916$		$E_{\text{FCI}} = -108.966039$	
1×10^{-6}	6,912	6,855,544	0.0047%	11.440	(12, 35.7)
1×10^{-7}	21,399	61,044,875	0.041%	3.502	(12, 426.5)
1×10^{-8}	50,284	333,094,790	0.23%	1.034	(12, 3095.3)
1×10^{-9}	106,006	1,450,439,300	0.98%	0.289	(24, 9237.9)
0	220,324	6,324,862,976	4.29%	0.071	(48, 26496.6)

C2, CN (cc-pVTZ), Cr2 (STO-3G) and N_2 (cc-pVTZ)

δ	$N_{\text{bitstrings}}$	N_{TBSCI}	$N_{\text{TBSCI}}/N_{\text{FCI}}$	$E_{\text{TBSCI}} - E_{\text{FCI}}$ (mH)	(nodes, seconds)
	CN	$R = 1.116$ Å	$E_{\text{HF}} = -92.219490$	$E_{\text{FCI}} = -92.569252$	
1×10^{-6}	1,984/1,652	1,415,808	0.00029%	17.985	(12, 2.6)
1×10^{-7}	6,630/5,040	11,207,206	0.0023%	6.510	(12, 34.0)
1×10^{-8}	19,914/13,362	79,673,340	0.016%	2.151	(12, 390.6)
1×10^{-9}	52,187/30,663	469,503,223	0.097%	0.677	(12, 3106.0)
0	305,847/125,366	12,654,398,243	2.60%	0.065	(144, 14019.6)
	CN	$R = 1.5$ Å	$E_{\text{HF}} = -92.059594$	$E_{\text{FCI}} = -92.476056$	
1×10^{-6}	2,477/1,919	1,768,335	0.00036%	17.844	(12, 4.3)
1×10^{-7}	8,279/5,621	15,071,035	0.0031%	6.181	(12, 59.6)
1×10^{-8}	23,647/14,219	98,778,650	0.020%	2.030	(12, 584.8)
1×10^{-9}	59,523/30,862	525,680,797	0.11%	0.645	(12, 4177.4)
0	311,092/114,033	10,549,477,483	2.17%	0.066	(144, 13893.2)
	CN	$R = 2.0$ Å	$E_{\text{HF}} = -91.844595$	$E_{\text{FCI}} = -92.324826$	
1×10^{-6}	5,292/2,965	4,743,784	0.00098%	13.904	(12, 37.8)
1×10^{-7}	15,933/8,039	36,786,548	0.0076%	4.788	(12, 346.8)
1×10^{-8}	42,851/18,919	218,222,058	0.045%	1.470	(12, 3739.1)
1×10^{-9}	97,277/37,180	956,531,522	0.20%	0.456	(24, 13296.6)
0	357,906/109,454	10,318,500,738	2.12%	0.060	(144, 43692.4)
	Cr_2	$R = 1.5$ Å	$E_{\text{HF}} = -2064.078900$	$E_{\text{FCI}} = -2064.808195$	
1×10^{-6}	5,293	3,820,595	0.00042%	24.535	(12, 6.0)
1×10^{-7}	17,326	40,118,388	0.0044%	9.346	(12, 98.1)
1×10^{-8}	47,239	291,894,399	0.032%	3.425	(12, 1287.5)
1×10^{-9}	107,597	1,495,787,547	0.16%	1.417	(24, 9897.9)
1×10^{-10}	199,571	5,116,950,531	0.56%	0.803	(36, 27027.4)
0	328,795	13,906,467,547	1.52%	0.582	(144, 27230.9)
	N_2	$R = 2.068$ a.u.	$E_{\text{HF}} = -108.984093$	$E_{\text{FCI}} = -109.375153$	
1×10^{-6}	1,530	586,634	0.000022%	20.668	(12, 0.60)
1×10^{-7}	5,472	5,147,844	0.00020%	8.036	(12, 6.0)
1×10^{-8}	16,861	42,751,597	0.0016%	2.633	(12, 71.8)
1×10^{-9}	43,914	288,849,332	0.011%	0.844	(12, 623.5)
0	260,795	12,554,326,295	0.48%	0.087	(144, 3772.6)

C2, Distribution of CI Coefficients Ratios $|C_K/C_{HF}|$





Determinants with the largest FCI coefficients are not randomly distributed, but are concentrated within a structured manifold induced by a small number of important α- and β-bitstrings.

C2, Cr2 (Ahlrichs SV) and N_2 (cc-pVQZ)

System	Active space	Molecular symmetry
Cr_2 (Ahlrichs SV)	CAS(48e, 42o)	D_{2h}
N_2 (cc-pVQZ)	CAS(10e, 108o)	D_{2h}

C2, Cr2 (Ahlrichs SV) and N_2 (cc-pVQZ)

δ	$N_{\text{bitstrings}}$	N_{TBSCI}	$E_{\text{TBSCI}} - E_{\text{benchmark}}$ (mH)	(nodes, seconds)
	Cr_2	$R = 1.5$ Å $E_{\text{HF}} = -2086.572971$	$E_{\text{DMRG}} = -2086.44478$	
1×10^{-6}	5,669	5,994,031	41.229	(12, 11.2)
1×10^{-7}	20,756	71,784,176	19.038	(12, 180.9)
1×10^{-8}	65,895	679,551,775	8.368	(12, 2409.7)
1×10^{-9}	181,976	5,061,656,122	3.752	(48, 6770.8)
1×10^{-10}	427,692	27,568,299,482	1.867	(386, 7228.9)
0	1,722,583	490,574,659,035	0.915	(7200, 10395.3)
	N_2	$R = 2.068$ a.u. $E_{\text{HF}} = -108.991735$	$E_{\text{FCIQMC}} = -109.40251$	
1×10^{-6}	1,916	887,486	30.961	(12, 1.28)
1×10^{-7}	7,268	10,169,032	13.662	(12, 20.6)
1×10^{-8}	24,819	100,565,129	4.798	(12, 306.5)
1×10^{-9}	73,249	815,730,549	1.423	(12, 3554.9)
0	826,347	146,434,093,579	-0.310	(2880, 10355.5)

We run these SCI calculations using our TBSCI eigensolver. For both systems, the largest SCI calculations exceeds the scale of previously reported SCI calculations.

D. Conclusions and Prospects

Conclusions

Conceptual foundation:

The TPB structure provides an organizing principle for indexing, traversing, and coupling determinants in Hamiltonian application, independent of how determinants are selected within the TPB representation.

HPC breakthrough:

A scalable TBSCI eigensolver enables the world's largest SCI calculations on Fugaku.

Scientific insight:

Near-FCI accuracy with only a tiny fraction of FCI determinants, revealing the intrinsic compactness of the TPB representation.

TBSCI OPENS A NEW STAGE FOR SCI

Prospects

Further refinement of determinant selection within the TPB representation.

Further improvement of the TBSCI eigensolver to reduce T_{delay} at larger node counts.

Extension of the distributed framework to incorporate second-order perturbative corrections (PT2).

Adaptation of the TBSCI framework to GPU-based supercomputers such as FugakuNEXT.

Physics > Chemical Physics

[Submitted on 13 Mar 2025 (v1), last revised 5 Mar 2026 (this version, v4)]

A Scalable Diagonalization Framework for Tensor-Product Bitstring Selected Configuration Interaction

Enhua Xu, William Dawson, Himadri Pathak, Takahito Nakajima

Selected configuration interaction (SCI) methods are effective for treating strongly correlated electronic systems, yet their scalability has long been limited by implementations that replicate the configuration interaction (CI) vector across processes, leading to severe memory bottlenecks. Here, we present a fully distributed diagonalization framework tailored for extremely large selected determinant spaces, directly addressing this major scalability bottleneck of modern SCI methods. The method is grounded in a tensor-product bitstring (TPB) representation, in which determinants are organized through a TPB structure constructed from selected alpha- and beta-bitstrings, and is referred to as tensor-product bitstring SCI (TBSCI). An efficient TBSCI eigensolver is developed based on a novel bitstring-based Hamiltonian evaluation algorithm together with a suite of MPI communication strategies designed to improve parallel efficiency. Large-scale full configuration interaction (FCI) benchmarks, employed as communication-intensive stress tests, demonstrate that the implemented TBSCI eigensolver continues to reduce the wall time for distributed diagonalization of 2.6 trillion determinants, reaching 54,000 nodes (more than 2.5 million cores) on supercomputer Fugaku. Beyond scalability, we investigate the structural compactness of the TPB representation and show that selecting alpha- and beta-bitstrings according to their collective weights in a reference SCI wavefunction yields TPB-based wavefunctions approaching the FCI limit while using only a small fraction of determinants. These results establish TBSCI as a scalable SCI methodology and provide evidence for the intrinsic compactness of the TPB representation.

Subjects: **Chemical Physics (physics.chem-ph)**
Cite as: arXiv:2503.10335 **[physics.chem-ph]**
(or arXiv:2503.10335v4 **[physics.chem-ph]** for this version)
https://doi.org/10.48550/arXiv.2503.10335

Submission history

From: Enhua Xu [view email]
[v1] Thu, 13 Mar 2025 13:13:59 UTC (196 KB)
[v2] Wed, 7 May 2025 02:18:07 UTC (201 KB)
[v3] Mon, 21 Jul 2025 03:07:40 UTC (999 KB)
[v4] Thu, 5 Mar 2026 07:27:41 UTC (686 KB)

TBSCI-202603 Public

Pin Watch 0 Fork 0

main 1 Branch 1 Tag Go to file t Add file Code

许恩华 and 许恩华 Update README: add BALANCE parameter description 5afd9bd · 2 weeks ago 2 Commits

bin	Initial frozen snapshot for TBSCI-202603 (JCP submission...	2 weeks ago
.gitignore	Initial frozen snapshot for TBSCI-202603 (JCP submission...	2 weeks ago
README.md	Update README: add BALANCE parameter description	2 weeks ago

README

TBSCI-202603

This repository provides executable binaries of the TBSCI eigensolver used in the associated manuscript. Bitstring preparation is performed using an external SCI tool as described in the manuscript.

Representative input and output data for selected benchmark systems are distributed via GitHub Releases.

Contents

- bin/ Prebuilt executables and corresponding makefiles:
  - `Fugaku_exe.out`, `Intel_exe.out`
  - `makefile_Fugaku`, `makefile_Intel`
- README.md This documentation.

Large input/output data files are NOT stored in the repository and must be downloaded separately from the Releases page.

About

Distributed tensor-produ
SCI eigensolver

Readme
Activity
1 star
0 watching
0 forks

Releases 1

TBSCI-202603 v1.0
2 weeks ago

Packages

No packages published
Publish your first package

Contributors

No contributors

William Dawson

Himadri Pathak

Takahito Nakajima

Thank you for your attention!