ISC High Performance 2022:
Accelerating Simulated Quantum Annealing with GPU and Tensor Cores

Presenter: Yi-Hua Chung

Advisor: Shih-Hao Hung

Yi-Hua Chung, Cheng-Jhih Shih, Shih-Hao Hung

/

Department of Computer Science & Information Engineering, National Taiwan University

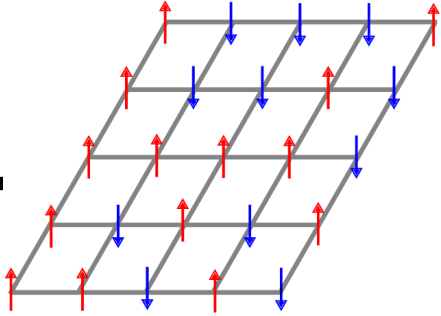
Yi-Hua Chung, r09944072@csie.ntu.edu.tw

Outline

- Introduction
  - Ising Model
  - Simulated Quantum Annealing (SQA)
  - Acceleration Strategy Adopted by Tohoku University (SOTA)
  - Tensor Cores (TCs)
- Methodology
  - Hierarchical Update (HU)
  - Utilizing the Tensor Cores
- Performance Evaluation
  - Five Experiments
- Conclusion

Introduction

4 Ising Model



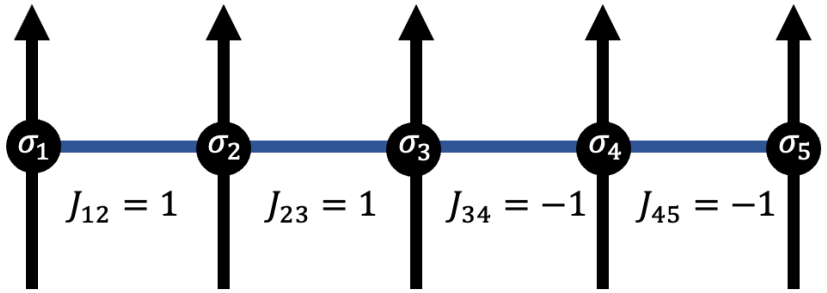
- Definition of Ising model
  - The Ising model is a mathematical model that doesn't correspond to an actual physical system. It's a huge lattice with **N** spins (σ_i), where each spin can be in one of two states **(+1, -1)**
  - The interaction terms between the connected spins are called **couplings** (J_{ij}). Besides, we can formulate the Hamiltonian of Ising model as:

$$H(\sigma) = -J \sum_{<ij>} \sigma_i \sigma_j - h \sum_j \sigma_j.$$

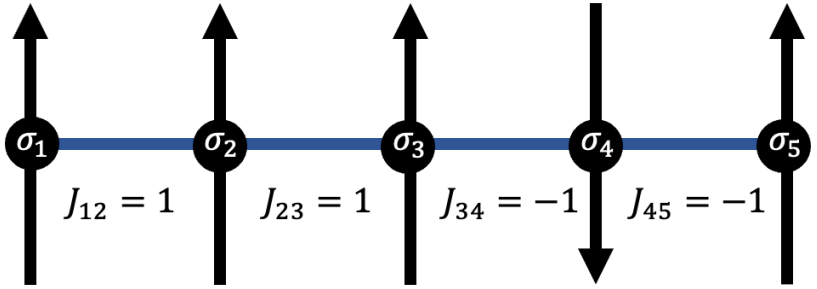
- When solving the Ising problem, our goal is to find a set of spin configuration that minimize the Hamiltonian of the system
  - For example, solving a 1D Ising model
    - (Next page)

5

$$H(\sigma) = -J \sum_{<ij>} \sigma_i \sigma_j - h \sum_j \sigma_j.$$



(…)



- Assume $h = 0$:
- Initial condition:

  $H(\sigma) = -[1 * (1 * 1) + 1 * (1 * 1) + (-1) * (1 * 1) + (-1) * (1 * 1)] = 0$

- Stage 1. flipping σ_1 from +1 to -1

  $H(\sigma) = -[1 * (-1 * 1) + 1 * (1 * 1) + (-1) * (1 * 1) + (-1) * (1 * 1)] = 2$ (X)

- Stage 2. flipping σ_2 from +1 to -1

  $H(\sigma) = -[1 * (1 * -1) + 1 * (-1 * 1) + (-1) * (1 * 1) + (-1) * (1 * 1)] = 4$ (X)

- Stage 3. flipping σ_3 from +1 to -1

  $H(\sigma) = -[1 * (1 * 1) + 1 * (1 * -1) + (-1) * (-1 * 1) + (-1) * (1 * 1)] = 0$ (X)

- Stage 4. flipping σ_4 from +1 to -1

  $H(\sigma) = -[1 * (1 * 1) + 1 * (1 * 1) + (-1) * (1 * -1) + (-1) * (-1 * 1)] = -4$ (O)

- Stage 5. flipping σ_5 from +1 to -1

  $H(\sigma) = -[1 * (1 * 1) + 1 * (1 * 1) + (-1) * (1 * -1) + (-1) * (-1 * -1)] = -2$ (X)

(…)

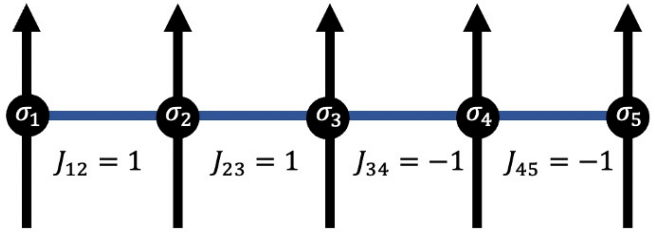
- Try to find out the best arrangement for the Ising problem according to the given coupling terms

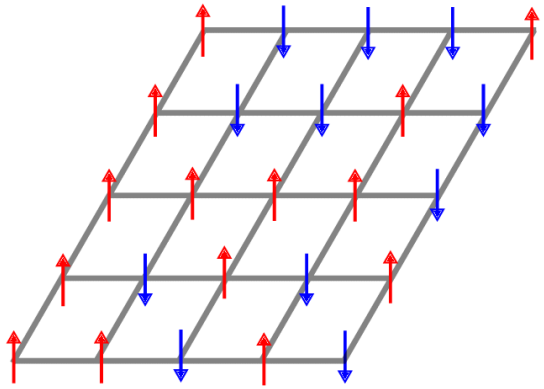
6 Cont'd

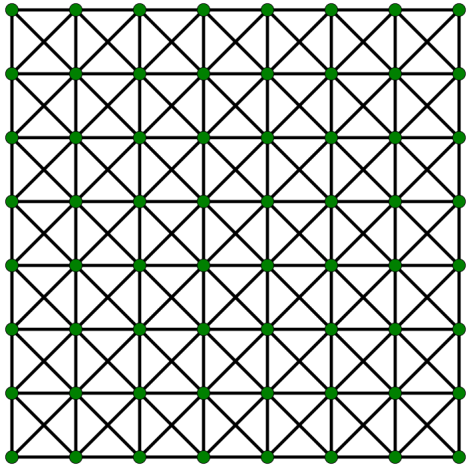
- When solving the Ising problem, our goal is to find a set of spin arrangements that minimize the total energy of the system
  - For example, solving a 1D Ising model *(done)*
  - **It is a combinatorial problem (Ising model: NP-Complete)**
  - By reducing NPC to Ising model, we can solve some NP-hard problems by solving the Ising model
- Generally, the Ising problem is currently solved in a heuristic
- One of the heuristic algorithms "**Simulated Quantum Annealing**" is an algorithm for solving combinatorial optimization problems

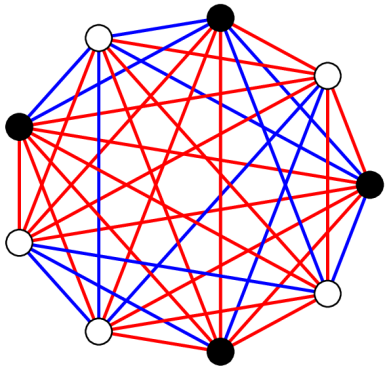
7 Cont'd

$$H(\sigma) = -J \sum_{<ij>} \sigma_i \sigma_j - h \sum_j \sigma_j.$$





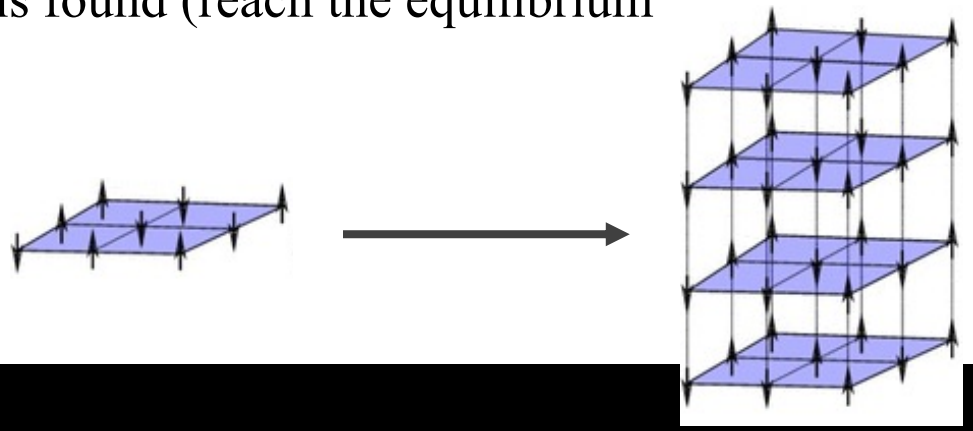




- There are many different Ising models. Different Ising models have different degrees of connectivity
- The higher spin connectivity for an Ising model, the more difficult for it to be accelerated
- In our work, we accelerate solving the **fully-connected Ising model** by adopting **simulated quantum annealing (SQA)** algorithm

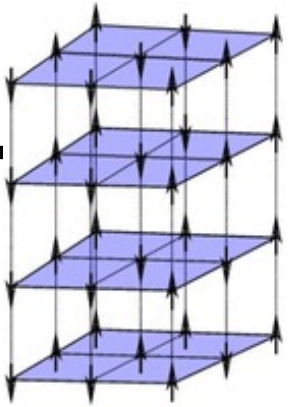
8 Simulated Quantum Annealing (SQA)

- Simulate quantum annealing (SQA) is a quantum-inspired algorithm
  - To simulate the quantum phenomenon, SQA duplicates original Ising model into M copies. This multi-trotter Ising model is called a transverse field Ising model (TFIM)
  - Different from the original Ising model, each spin has the connections with the spin in the same position located at the previous and the next trotter of TFIM
- A path integral quantum Monte Carlo method
  - Given an initial temperature $T(0)$ and magnetic field $\Gamma(0)$, and random generate a set of spin configuration initially
  - As time increases (MC-step increases), the value of $T(t)$ will gradually decrease, so the spins flip until a set of spin configuration with the lowest energy is found (reach the equilibrium with the environment)



9 Spin Energy of TFIM

$$H(\sigma) = -J \sum_{<ij>} \sigma_i \sigma_j - h \sum_j \sigma_j.$$



- Hamiltonian for the Fully-connected TFIM [ref-1, ref-2] :
  - $H_{TFIM} = -\sum_{m=1}^{M} \left(\sum_{(i,j)} \frac{J_{ij}}{M} \sigma_{i,m}\sigma_{j,m} + \sum_{(i)} \frac{h_i}{M} \sigma_{i,m} - J^{\dagger} \sum_{(i)} \sigma_{i,m}\left(\sigma_{i,m-1} + \sigma_{i,m+1}\right) \right)$,

    for $J^{\dagger} = -\frac{T}{2} * \ln\left(\frac{\coth(\Gamma(t))}{MT}\right) \in const$ (under a fixed MC-step)
- By equation above, we know the spin energy for $\sigma_{i,m}$, the *i-th* spin on *m-th* trotter is:
  - $spin\_energy_{i,m} = -\left(\sum_{j=1}^{N} \frac{J_{ij}}{M} \sigma_{i,m}\sigma_{j,m} + \frac{h_i}{M} \sigma_{i,m} - J^{\dagger} \sigma_{i,m}\left(\sigma_{i,m+1} + \sigma_{i,m-1}\right) \right)$
- Let $J^{\dagger\prime} = J^{\dagger}/\mathrm{M}$:
  - $spin\_energy_{i,m} = -\sigma_{i,m}\left(\sum_{j=1}^{N} J_{ij}\sigma_{j,m}\right) - \sigma_{i,m}h_i + J^{\dagger\prime}\sigma_{i,m}\left(\sigma_{i,m+1} + \sigma_{i,m-1}\right)$
- Most of the problem can be solved by setting $h_i = 0$
  - $spin\_energy_{i,m} = -\sigma_{i,m}\left(\sum_{j=1}^{N} J_{ij}\sigma_{j,m}\right) + J^{\dagger\prime}\sigma_{i,m}\left(\sigma_{i,m+1} + \sigma_{i,m-1}\right)$

ref-1: "Relationship between d-Dimensional Quantal Spin Systems and (d+1)-Dimensional Ising Systems: Equivalence, Critical Exponents and Systematic Approximants of the Partition Function and Spin Correlations "

ref-2: (IEEE international conference on rebooting computing (ICRC) "An ising computer based on simulated quantum annealing by path integral monte carlo method"

Effect of a Spin-Flip

- Why do we need to know the spin energy of a spin?
  - Goal: Find a set of spin configurations with the lowest energy under the fixed environment (reach the equilibrium with the outside system)
    - During annealing, we try to flip the spins of the TFIM to get the different spin configuration
  - Spin energy for σ_i, the *i-th* spin on *m-th* trotter
    - $spin\_energy_{i,m} = -\sigma_{i,m}\left(\left(\sum_{j=1}^{N} J_{ij}\sigma_{j,m}\right) - J^{\dagger'}\left(\sigma_{i,m+1} + \sigma_{i,m-1}\right)\right)$
  - Hence, we need to know: (1) the spin energy of the spin before a spin-flip (2) calculate the delta energy caused by a spin-flip

Implementation of SQA Algorithm (1)

Algorithm 1: SQA Algorithm

```
1  Coupling Matrix ← Couplings Data
2  Randomly initialize σ ∈ {−1, 1}
3  local_field ← 0
4  // Construct local-field energy
5  for m = 1 to M do
6  |  for i = 1 to N do
7  |  |  for j = 1 to N do
8  |  |  |  local_field_{i,m} += J_{i,j} * σ_{j,m}
9  |  |  end
10 |  end
11 end
```

Initialization

```
12 for t = 1 to MC-STEP do
13 |  for i = 1 to N do
14 |  |  for m = 1 to M do
15 |  |  |  ΔH = σ_{i,m}(local_field_{i,m} − J†(σ_{i,m+1}+σ_{i,m−1}))
16 |  |  |  // Judge_to_Flip
17 |  |  |  if e^{−ΔH/T} > random() then
18 |  |  |  |  σ_{i,m} = −σ_{i,m}
19 |  |  |  |  // Update local-field energy
20 |  |  |  |  for j = 1 to N do
21 |  |  |  |  |  local_field_{j,m} += 2 * J_{i,j} * σ_{i,m}
22 |  |  |  |  end
23 |  |  |  end
24 |  |  end
25 |  end
26 end
```

- Initialization
  1. Random generate N spin states for M trotters
  2. Prepare the local-field for all the spins
- From the previous page

$$spin\_energy_{i,m} = -\sigma_{i,m}\left(\sum_{j=1}^{N} J_{ij}\sigma_{j,m}\right) + J^{\dagger\prime}\sigma_{i,m}(\sigma_{i,m+1} + \sigma_{i,m-1})$$

local-field

Implementation of SQA Algorithm (1)

Algorithm 1: SQA Algorithm

```
1  Coupling Matrix ← Couplings Data
2  Randomly initialize σ ∈ {−1, 1}
3  local_field ← 0
4  // Construct local-field energy
5  for m = 1 to M do
6      for i = 1 to N do
7          for j = 1 to N do
```

$$spin\_energy_{i,m} = -\sigma_{i,m}\left(\sum_{j=1}^{N} J_{ij}\sigma_{j,m}\right) + J^{+'}\sigma_{i,m}(\sigma_{i,m+1} + \sigma_{i,m-1})$$

```
12 for t = 1 to MC-STEP do
13     for i = 1 to N do
14         for m = 1 to M do
15             ΔH = σ_{i,m}(local_field_{i,m} − J†(σ_{i,m+1}+σ_{i,m−1}))
16             // Judge_to_Flip
17             if e^{−ΔH/T} > random() then
18                 σ_{i,m} = −σ_{i,m}
19                 // Update local-field energy
20                 for j = 1 to N do
21                     local_field_{j,m} += 2 * J_{i,j} * σ_{i,m}
22                 end
23             end
24         end
25     end
26 end
```

- We can only flip a spin in a time
  - Since all the spins on the same trotter are connected, we can only flip a spin in a time
  - Besides, if a spin flip, all the local-field of the spins on the same trotter need to be update

---

- Update local-field
  - $local\_field_{k,m} = \sum_{j=1\neq i}^{N} J_{k,j}\sigma_{j,m} + J_{k,i}\sigma_{i,m}$
- As *k-th* spin flip on *m-th* trotter for *i-th* spin-flip:
  - $local\_field_{k,m}{}' = \sum_{j=1\neq i}^{N} J_{k,j}\sigma_{j,m} + J_{k,i}\sigma_{i,m}{}'$
  - $local\_field_{k,m}{}' - local\_field_{k,m}$

  $= (\sum_{j=1\neq i}^{N} J_{k,j}\sigma_{j,m} + J_{k,i}\sigma_{i,m}{}') - (\sum_{j=1\neq i}^{N} J_{k,j}\sigma_{j,m} + J_{k,i}\sigma_{i,m})$
  - $= J_{k,i}\sigma_{i,m}{}' - J_{k,i}\sigma_{i,m}$
  - $= J_{k,i}\sigma_{i,m}{}' - J_{k,i}(-\sigma_{i,m}{}') = 2 * J_{k,i}\sigma_{i,m}{}'$

For $e^{\frac{-\Delta H}{T}}$: (IEEE international conference on rebooting computing (ICRC) “An ising computer based on simulated quantum annealing by path integral monte carlo method”

13 SOTA of SQA and the Adopted Methodology by GPA-Based SOTA

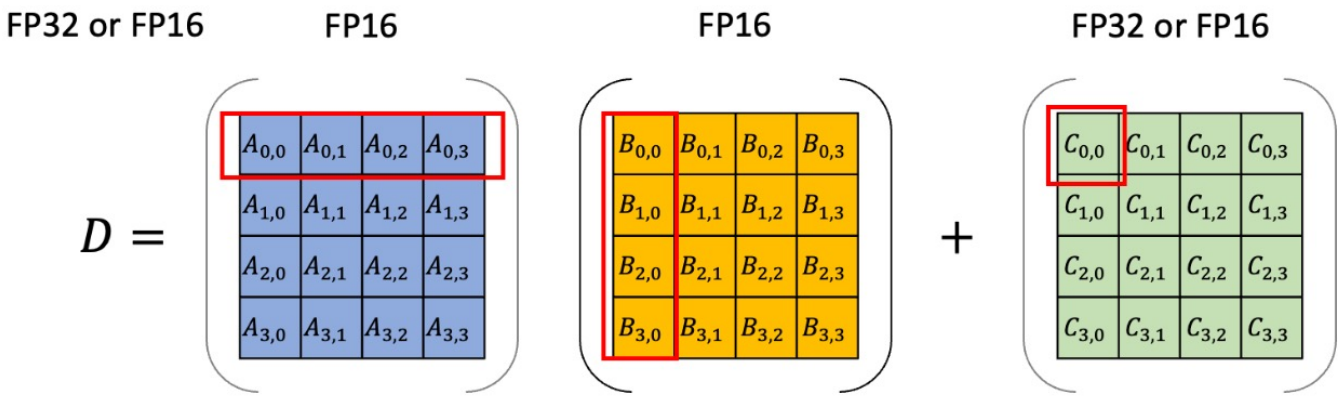
- Time complexity of SQA
  - Per MC-step (N spins, M trotters)
  - $O(M * N) * O(N) \rightarrow O(M * N^2)$
- Acceleration are needed, state of the art (SOTA, both conducte
  - **GPU-based:** A GPU-based quantum annealing simulator for ful spatial and temporal parallelism (2020, IEEE Access)
  - **FPGA-based:** Highly-parallel FPGA accelerator for simulated c Transactions)
- Parallel the calculation of updating local-field by using accele
- Ideas proposed by Tohoku University
  - Parallel processing of spins belonging to three different trotters

Algorithm 1: SQA Algorithm

```
1  Coupling Matrix ← Couplings Data
2  Randomly initialize σ ∈ {−1, 1}
3  local_field ← 0
4  // Construct local-field energy
5  for m = 1 to M do
6      for i = 1 to N do
7          for j = 1 to N do
8              local_field_{i,m} += J_{i,j} * σ_{j,m}
9          end
10     end
11 end

12 for t = 1 to MC-STEP do
13     for i = 1 to N do
14         for m = 1 to M do
15             ΔH = σ_{i,m}(local_field_{i,m} − J†(σ_{i,m+1}+σ_{i,m−1}))

16             // Judge_to_Flip
17             if e^{−ΔH/T} > random() then
18                 σ_{i,m} = −σ_{i,m}

19                 // Update local-field energy
20                 for j = 1 to N do
21                     local_field_{j,m} += 2 * J_{i,j} * σ_{i,m}
22                 end
23             end
24         end
25     end
26 end
```

14 Tensor Cores (TCs)

- NVIDIA released Volta microarchitecture featuring specialized computing units called "**Tensor Cores (TCs)**"
- TCs are used in: (1) Deep Neural Networks (DNN), (2) **Matrix multiplication**
- How to use?
  - **cuBLAS**, CUTLASS, WMMA
- Data type supported by cuBLAS :
  - FP32, FP16, BP16, …



$$D_{0,0} = A_{0,0} * B_{0,0} + A_{0,1} * B_{1,0} + A_{0,2} * B_{2,0} + A_{0,3} * B_{3,0} + C_{0,0}$$

Ref: cuBLAS (CUDA Basic Linear Algebra Subprograms)
toolkit: https://docs.nvidia.com/cuda/cublas/index.html

Methodology

Methodology

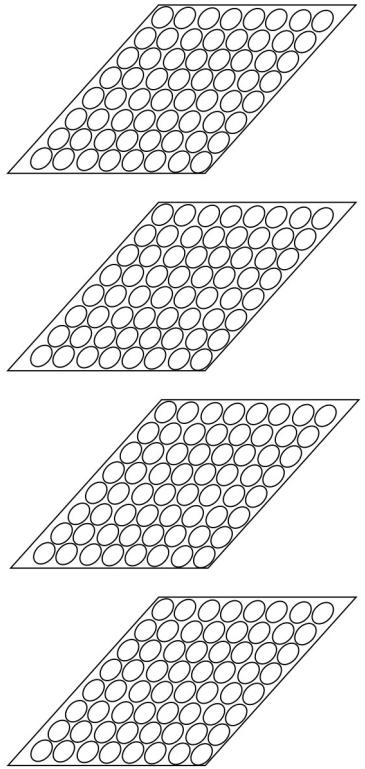
- **Hierarchical Update** - Re-forming the SQA algorithm to implement the algorithm more parallelly
- **Utilizing the Tensor Cores** - Not only **GPU** is used, **Tensor Cores (TCs)** provided by NVIDIA are also used to accelerate the algorithm in our work

Methodology - Hierarchical Update

- ***Hierarchical Update*** *- Re-forming the SQA algorithm to implement the algorithm more parallelly*
- **Utilizing the Tensor Cores** - Not only **GPU** is used, **Tensor Cores (TCs)** provided by NVIDIA are also used to accelerate the algorithm in our work

18 Hierarchical Update (HU)

- Different from the original SQA algorithm where each layer of trotter can only flip one spin at a time, HU will **only update the local-field of the some spins** but all the spins on the same trotter



*N = 64 (8*8), M = 4*

s1	s9	s17	s25	s33	s41	s49	s57
s2	s10	s18	s26	s34	s42	s50	s58
s3	s11	s19	s27	s35	s43	s51	s59
s4	s12	s20	s28	s36	s44	s52	s60
s5	s13	s21	s29	s37	s45	s53	s61
s6	s14	s22	s30	s38	s46	s54	s62
s7	s15	s23	s31	s39	s47	s55	s63
s8	s16	s24	s32	s40	s48	s56	s64

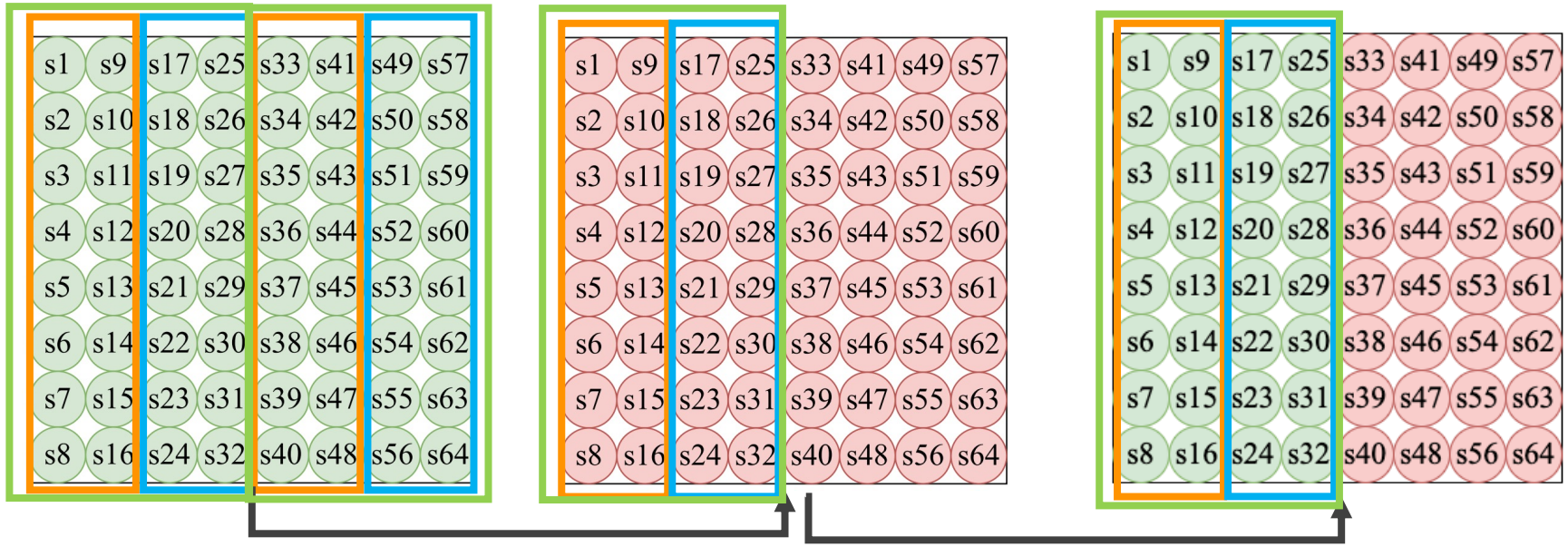
Spins' energies are correct

s1	s9	s17	s25	s33	s41	s49	s57
s2	s10	s18	s26	s34	s42	s50	s58
s3	s11	s19	s27	s35	s43	s51	s59
s4	s12	s20	s28	s36	s44	s52	s60
s5	s13	s21	s29	s37	s45	s53	s61
s6	s14	s22	s30	s38	s46	s54	s62
s7	s15	s23	s31	s39	s47	s55	s63
s8	s16	s24	s32	s40	s48	s56	s64

Spins' energies are not correct

19 Cont'd

- block_size (*blk_sz*) is set
  - Then we split the N spins into $\frac{N}{(blk\_sz)}$ blocks (Green), and each block is separated into two small blocks (orange and blue) assume *blk_sz =32*

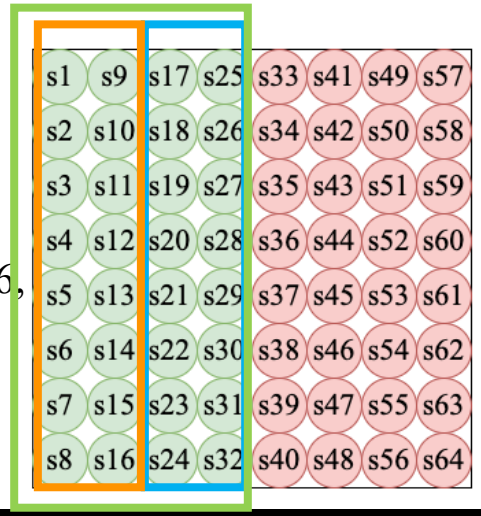


- Action: judge_to_flip s1,
  If s1 flip, then all the local-field of the spins on the same trotter are wrong
- We update all the local-field of the spins in the same block (by adding $2 * J_{k,i}\sigma_{i,m}{}'$)

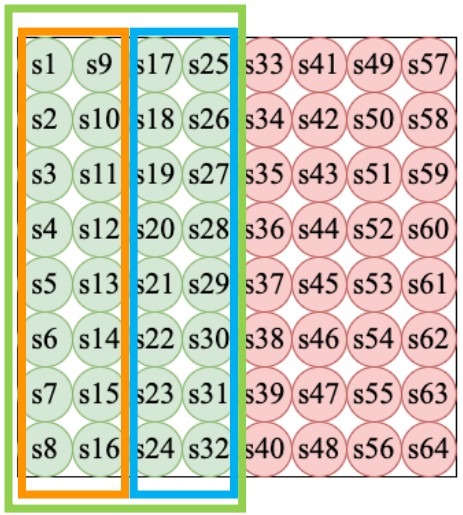
- Action: judge_to_flip s1,
  If s1 flip, then the local-field of the spins on the same trotter are wrong
- We update the local-field of the spins in the same block (by adding $2 * J_{k,i}\sigma_{i,m}{}'$)
- Action: judge_to_flip s2,
- If s2 flip, update the local-field for the spins in the block

(…)

- Action: judge_to_flip s16,
- We only need to update the local-fields for the spins in the block

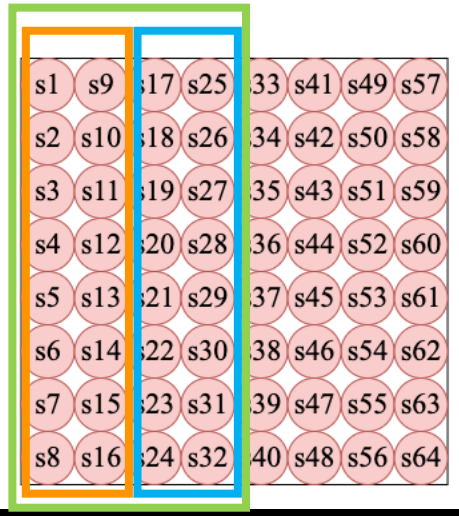


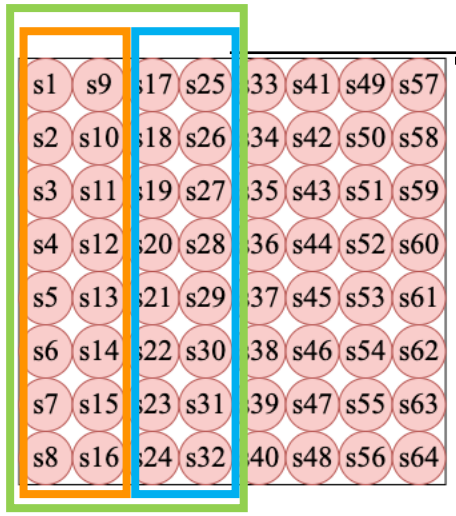
- Action: judge_to_flip s17



(…)

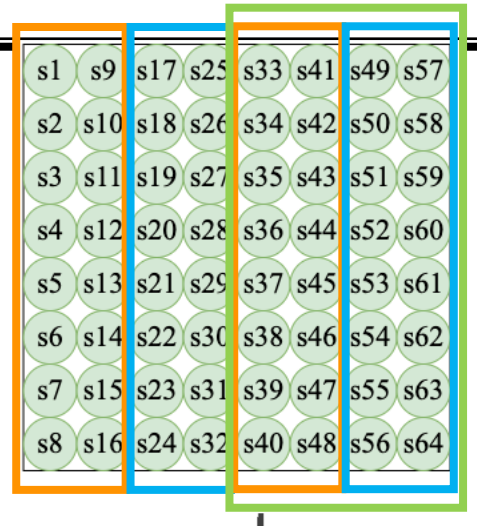
- Until all spins in the orange are flipped, then flip the spins in the blue



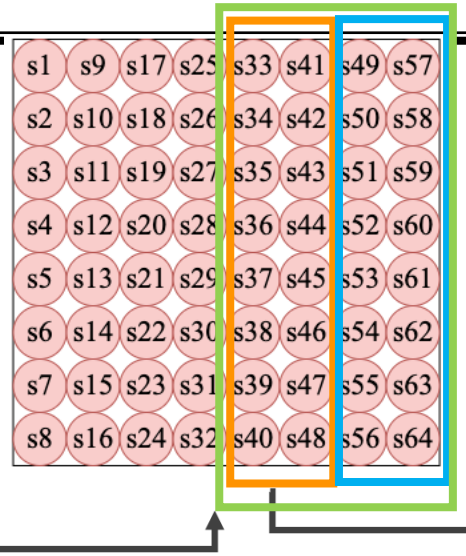


- After all the spins are judged to flip in the block, then all the local-field of spins are incorrect

- Then we cost O(N) to update all the local-fields for N spins (the same as the original method)



- And heading to flip the spin in the next block

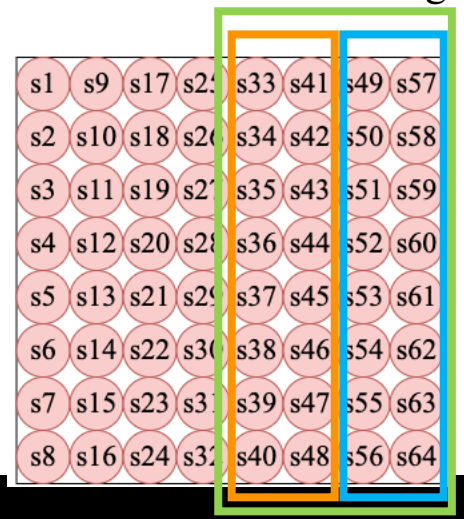


- Action: judge_to_flip s33, If s33 flip, the local-field of the spins on the same trotter are wrong

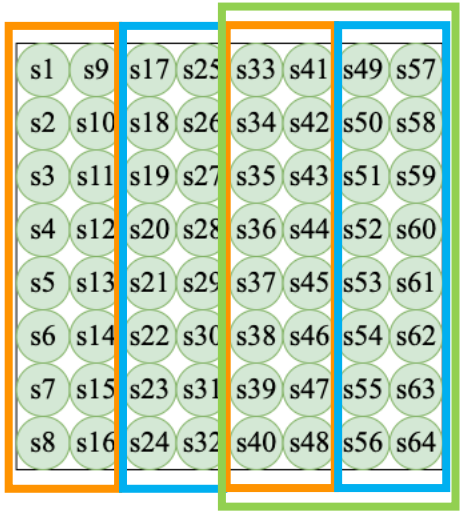
- We update the local-field of the spins in the same block (by adding $2 * J_{k,i}\sigma_{i,m}{}'$)

(…) (…)

- judge_to_flip all the spins in the orange small block, then heading to flip all the spins in blue

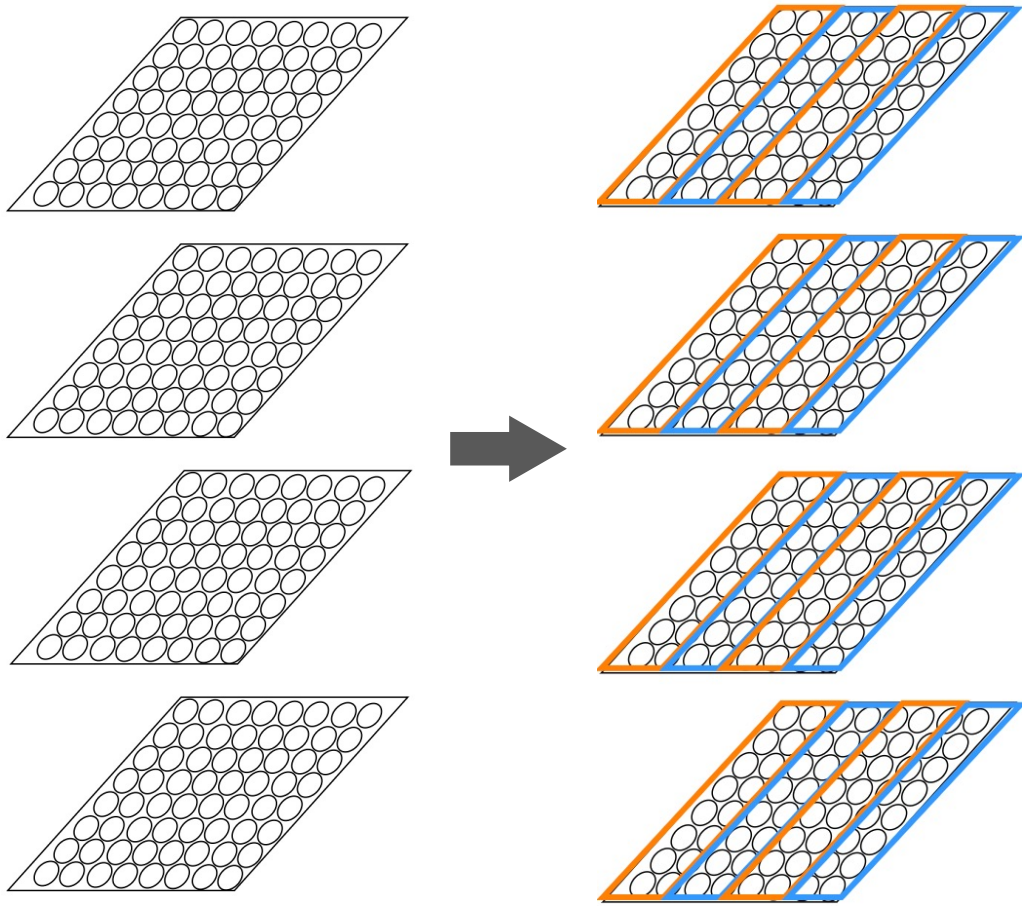


- Then we cost O(N) to update all the local-fields for N spins (the same as the original method)

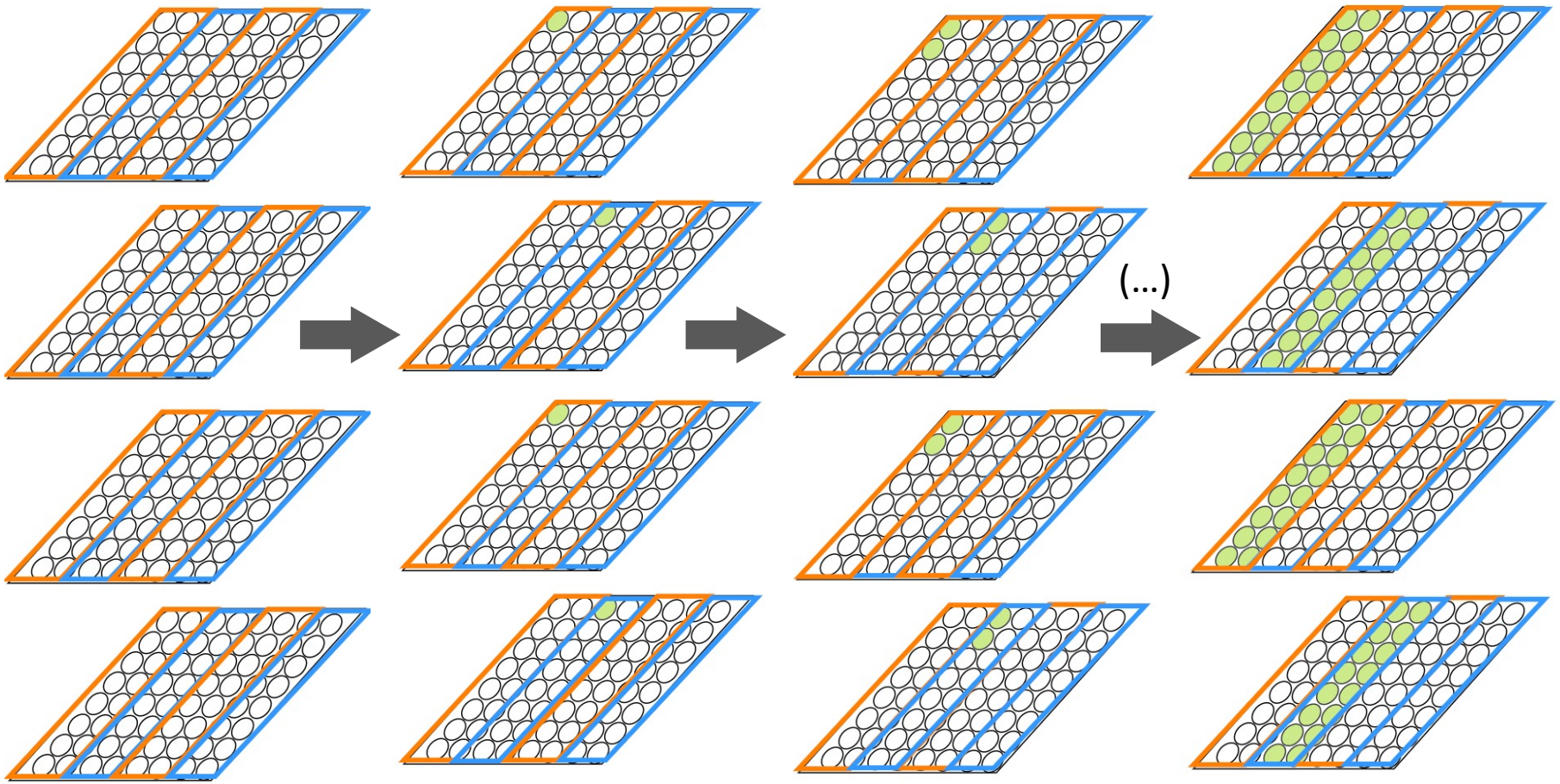


22 Flip Different Block on the Different Trotter

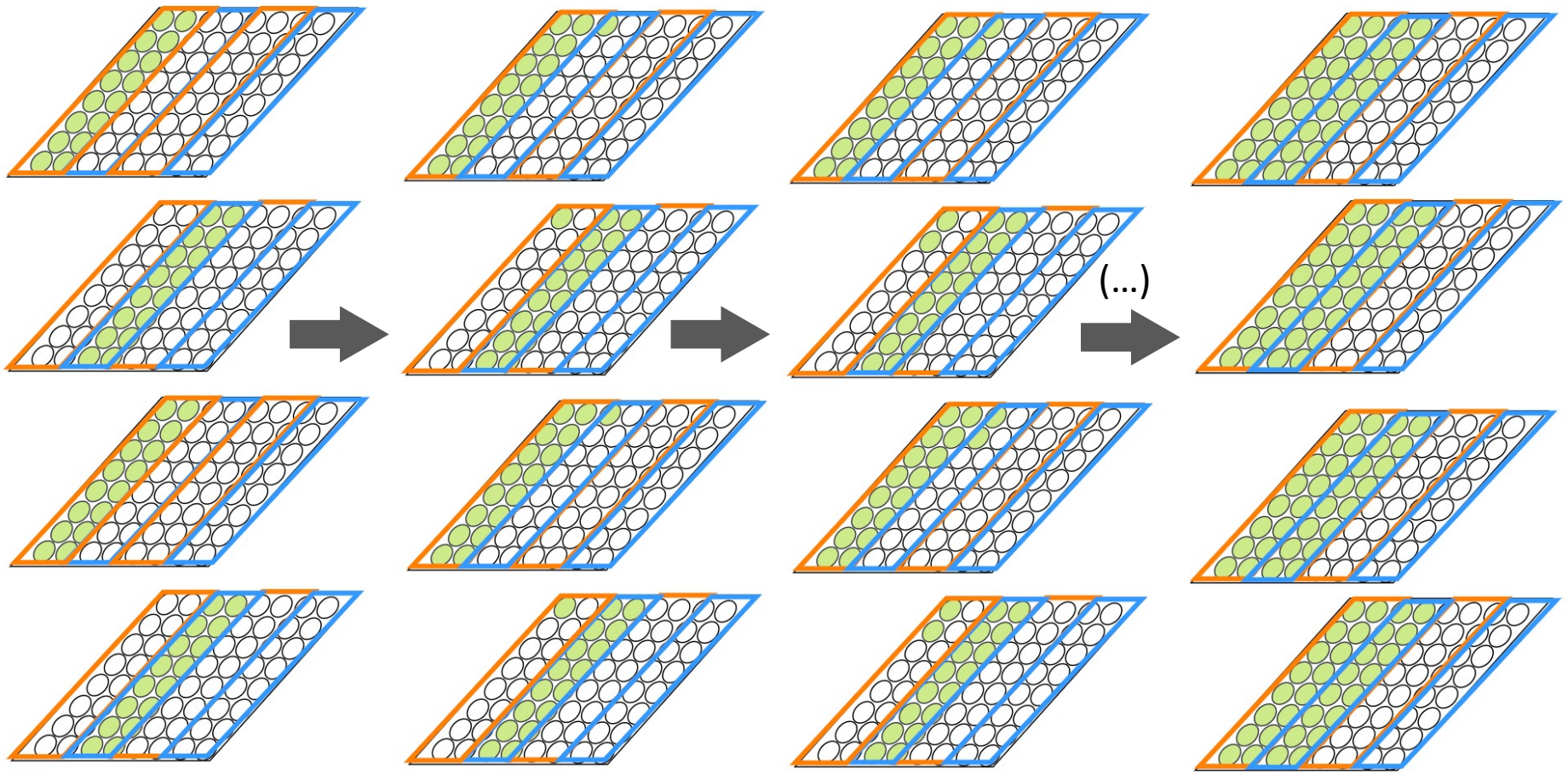
- By flipping different small block of each trotter to maintain the parallelism on M-dimension



- 1-th round
  - Odd trotter → flip the first orange block
  - Even trotter → flip the first blue block
  - Odd trotter → flip the first blue block
  - Even trotter → flip the first orange block
  - __syncthreads();
- Update al the *M*N* local-field
- 2-th round
  - Odd trotter → flip the first orange block
  - Even trotter → flip the first blue block
  - Odd trotter → flip the first blue block
  - Even trotter → flip the first orange block
  - __syncthreads();
- Update al the *M*N* local-field



- 1-th round
  - **Odd trotter → flip the first orange block**
  - **Even trotter → flip the first blue block**



- 1-th round
  - Odd trotter → flip the first orange block
  - Even trotter → flip the first blue block
  - **Odd trotter → flip the first orange block**
  - **Even trotter → flip the first blue block**
  - __syncthreads();

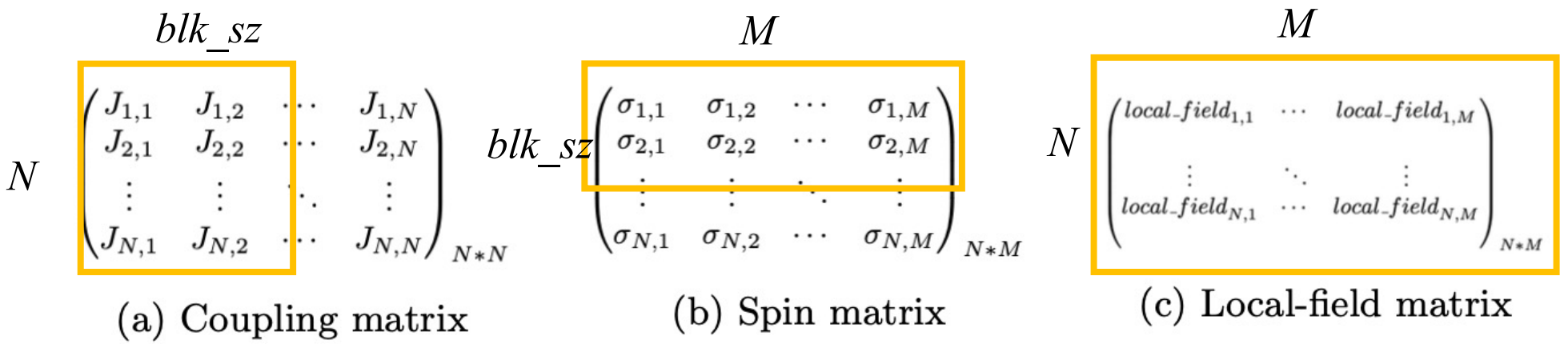
Update al the *M*N* local-field

Methodology - Utilizing the Tensor Cores

- **Hierarchical update** - Re-forming the SQA algorithm to implement the algorithm more parallelly
- ***Utilizing the Tensor Cores*** *- Not only* ***GPU*** *is used,* ***Tensor Cores (TCs)*** *provided by NVIDIA are also used to accelerate the algorithm in our work*
  - While update the M*N local-field, we can reach a higher acceleration by using TCs, a special designed microarchitecture to conduct matrix multiplication

26 Updating Local-Field by Using TCs

- After judge_to_flip *blk_sz* spins on *M* trotters, we need to update *M*N* local field
- During update, we only need to conduct the matrix multiplication between coupling matrix and spin matrix for $N * blk\_sz * M$ to update all the local-field of *M*N* spins



(a) Coupling matrix (b) Spin matrix (c) Local-field matrix

27 Analysis for the Proposed Methodology

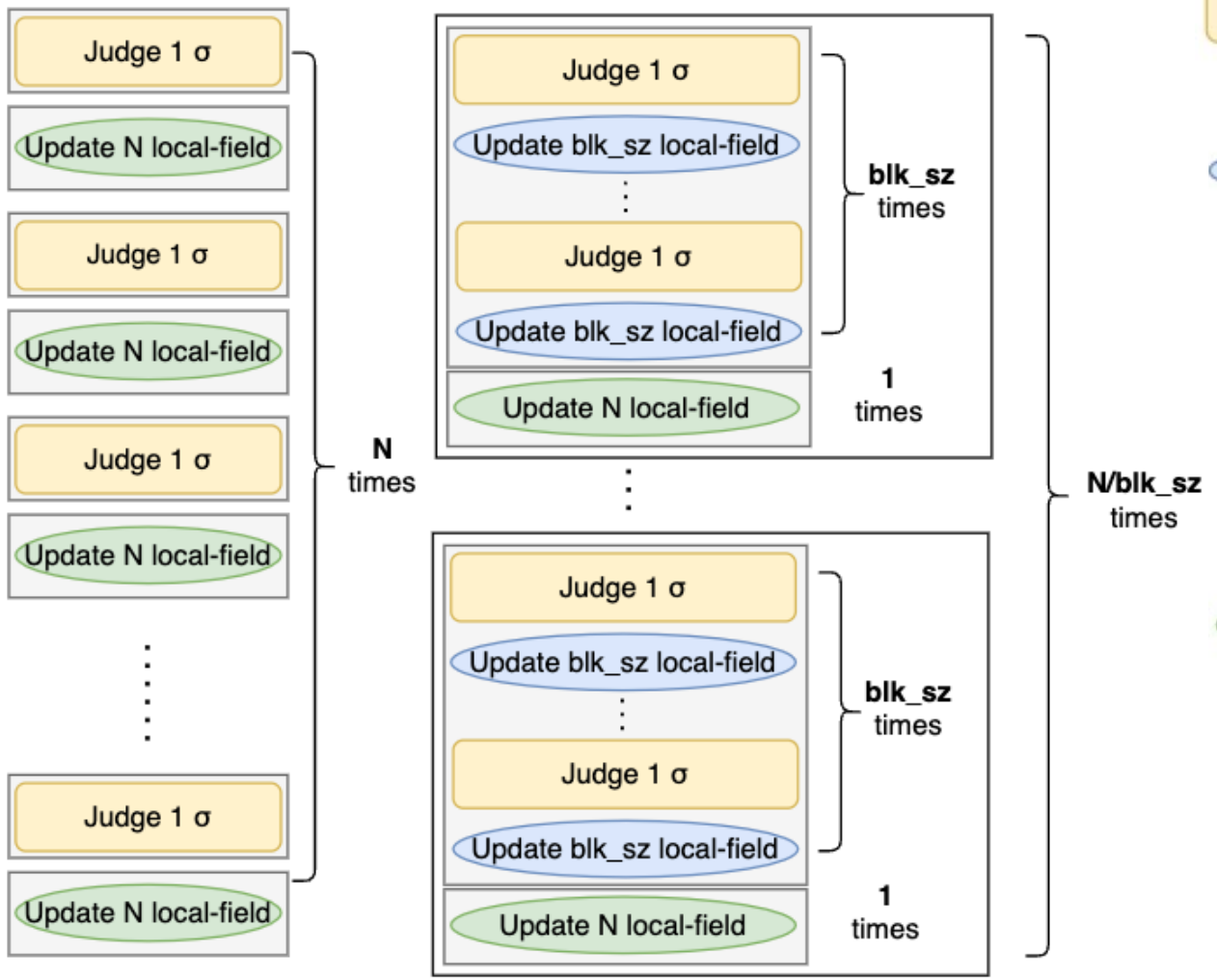
: kernel launch

Judge 1 σ

N times, sequential part

Update N local-field

N times, parallel part
For one spin: $2 * J_{k,i} * \sigma_{i,m}{}'$
For N spins, N times:
Multiplication: $(2N) * (N)$
Addition: $(1N) * (N)$



Judge 1 σ

N times, sequential part

Update blk_sz local-field

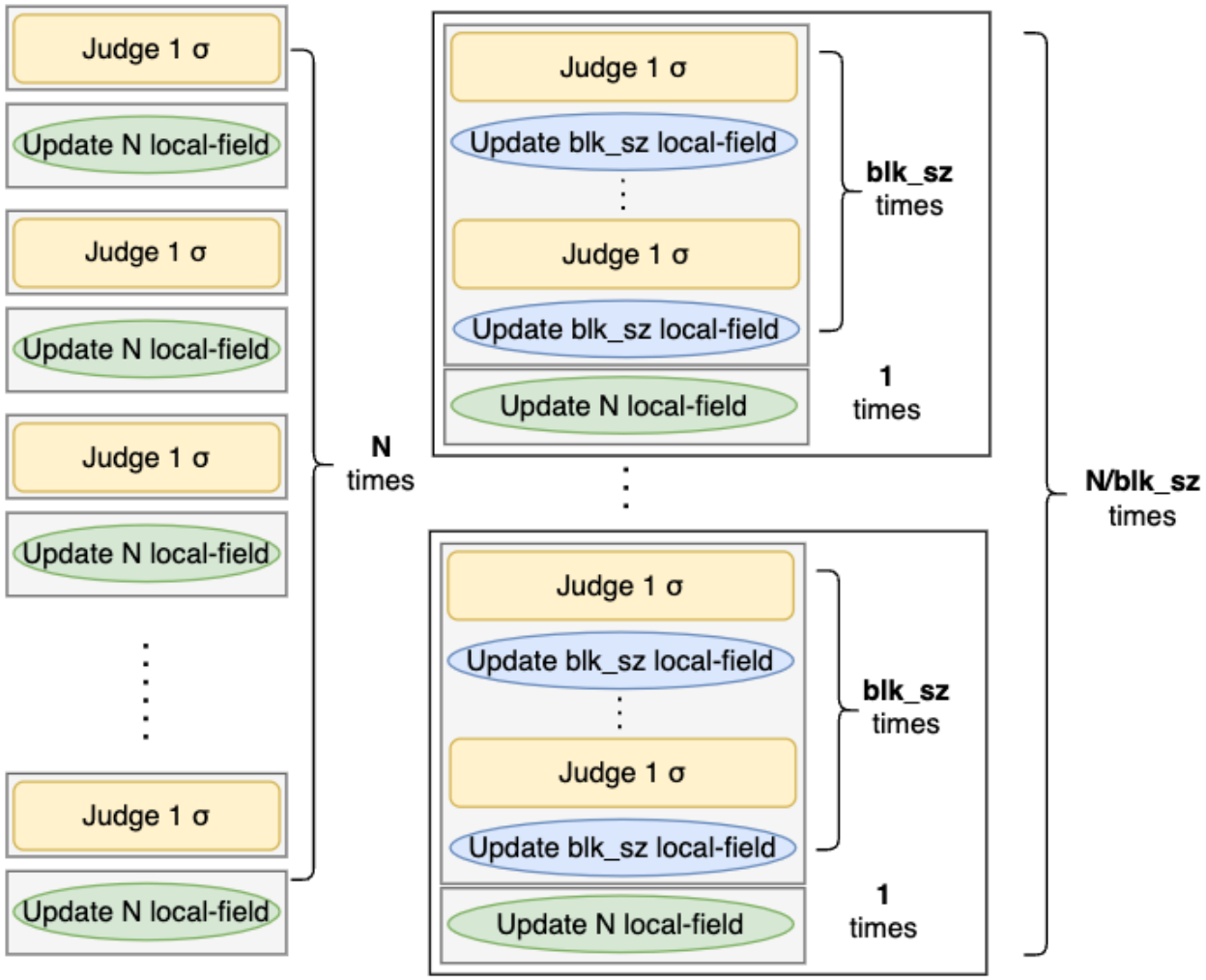
$blk\_sz * (N/blk\_sz)$ times, parallel part
For one spin: $2 * J_{k,i} * \sigma_{i,m}{}'$
For blk_sz spins, N times:
Multiplication: $(2 * blk\_sz) * (N)$
Addition: $(1 * blk\_sz) * (N)$

Update N local-field

(N/blk_sz) times, parallel part
For one spin: $2 * J_{k,i} * \sigma_{i,m}{}'$
For N spins, N/blk_sz times:
Multiplication: $(2N) * (N/blk\_sz)$
Addition: $(1N) * (N/blk\_sz)$

Without Hierarchical Update **With Hierarchical Update**

Cont'd

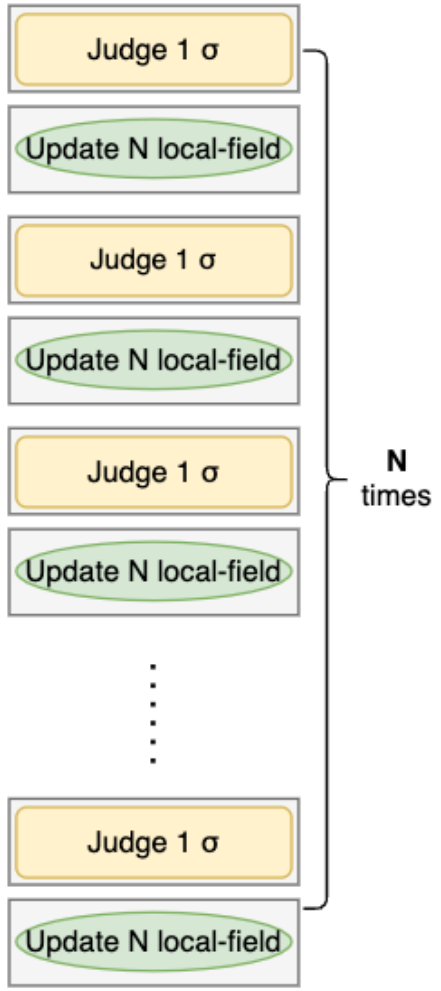


- With adopting the HU, the times to launch the GPU kernel is less than the method without using HU
  → **Less launch overhead**
- Since *blk_sz* is small, computing data can be placed in the memory of local processor, but the global memory of the GPU
  → **Faster memory access**
- More acceleration can be done by using TCs to calculate the green part
  → **Tensor Cores acceleration**
- We still maintain the parallelism in the M-dimension

SQA Algorithm -1

Algorithm 1: SQA Algorithm

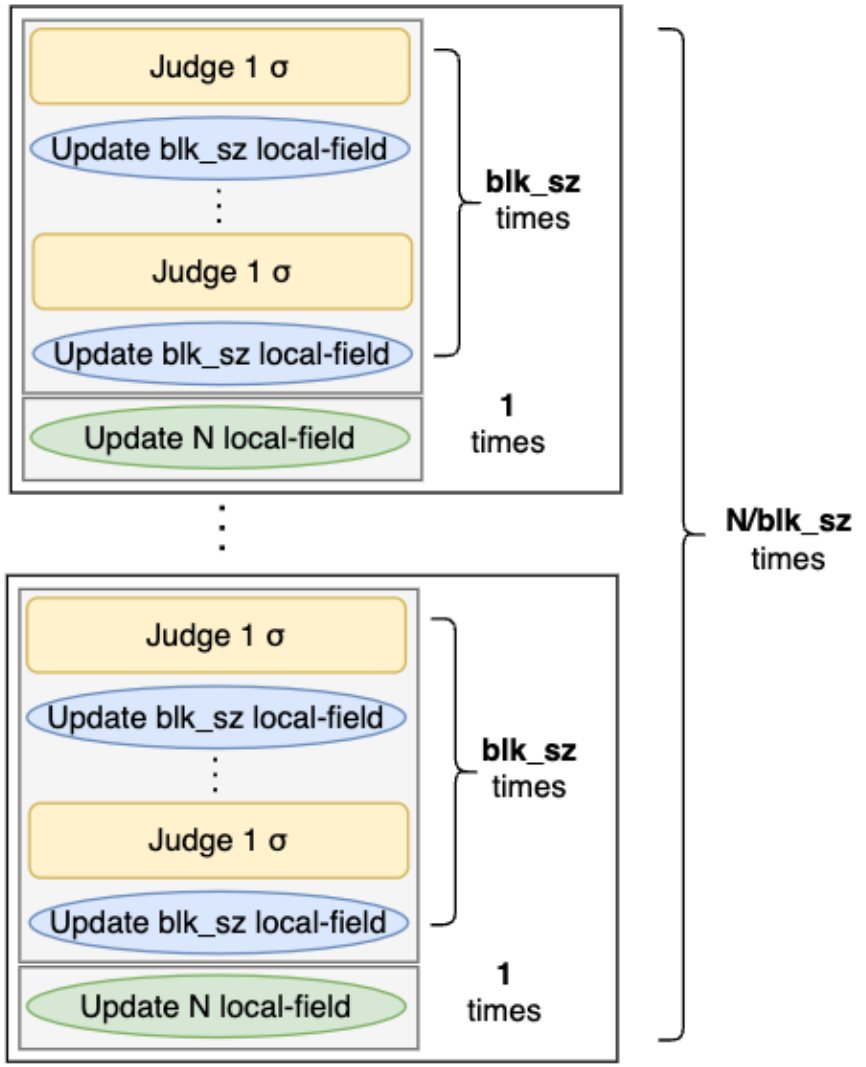
```
Coupling Matrix ← Couplings Data
Randomly initialize σ ∈ {−1, 1}
local_field ← 0
// Construct local-field energy
for m = 1 to M do
    for i = 1 to N do
        for j = 1 to N do
            local_field_{i,m} += J_{i,j} * σ_{j,m}
        end
    end
end

for t = 1 to MC-STEP do
    for i = 1 to N do
        for m = 1 to M do
            ΔH = σ_{i,m}(local_field_{i,m} − J†(σ_{i,m+1}+σ_{i,m−1}))

            // Judge_to_Flip
            if e^{−ΔH/T} > random() then
                σ_{i,m} = −σ_{i,m}

                // Update local-field energy
                for j = 1 to N do
                    local_field_{j,m} += 2 * J_{i,j} * σ_{i,m}
                end
            end
        end
    end
end
```



SQA Algorithm -2

Algorithm 2: Proposed Algorithm

```
1  Coupling Matrix ← Coupling Data
2  Randomly initialize σ ∈ {−1, 1}
3  local_field ← 0
4  // Construct_local_field_Energy, using Tensor Cores
5  for m = 1 to M do
6      for i = 1 to N do
7          for j = 1 to N do
8              local_field_{i,m} += J_{i,j} * σ_{j,m}
9          end
10     end
11 end
12 for t = 1 to MC-STEP do
13     for i = 1; i ≤ N / blk_sz; i++ do
14         // Judge_to_Flip (ii = i * blk_sz)
15         for m = 1; m ≤ M; m++ do
16             for j = 1; j ≤ blk_sz; j++ do
17                 ΔH = σ_{ii+j,m} (local_field_{ii+j} − J†(σ_{ii+j,m+1}+σ_{ii+j,m−1}))
18                 if e^{−ΔH/T} > random() then
19                     σ_{ii+j,m} = −σ_{ii+j,m}
20                     for k = 1; k ≤ blk_sz; k++ do
21                         local_field_{ii+k} += 2 * J_{ii+j,i+k} * σ_{ii+j,m}
22                     end
23                 end
24                 __syncthreads()
25             end
26         end
27         // Update local_field Energy, using Tensor Cores
28         for m = 1; m ≤ M; m++ do
29             for j = 1; j ≤ N; j++ do
30                 for k = 1; k ≤ blk_sz; k++ do
31                     local_field_{j,m} += 2 * J_{j,k} * σ_{k,m}
32                 end
33             end
34         end
35     end
36 end
```



Performance Evaluation

Performance Evaluation

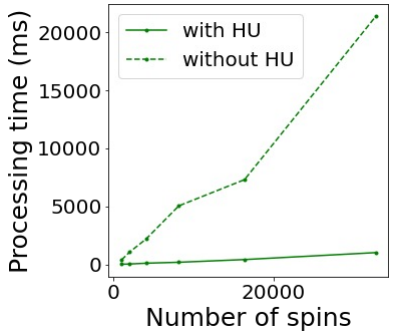
- Exp -1. Benefits of Hierarchical Update
- Exp -2. Per-Step Annealing Time
  - Compared with Tohoku University (both GPU and FPGA Version)
- Exp -3. The Choice of *blk_sz*
- Exp -4. The Impact of Tensor Cores
- Exp -5. The Quality of Solution
  - MAX-CUT problem, Gset served by Stanford University

Experimental Environment

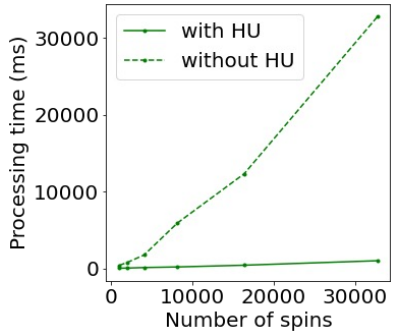
- Environment for exp -1, exp -3, exp -4, and exp -5
  - An Intel(R) Core(TM) i9-9900KF CPU running at 3.60GHz with Ubuntu 18.04.5 LTS OS and a GeForce RTX 3080 GPU with CUDA 11.3 library
- Environment for exp -2
  - For a fair comparison, we run the experiment 2 on a setup that consists of an Intel(R) Xeon(R) Gold 6154 CPU chip running at 3.00GHz with CentOS Linux release 7.8.2003(Core) and a **Tesla V100-SXM2** GPU with the CUDA 11.3 library
  - Tohoku University uses **Quadro GV100 GPU**, which adopts the same Volta architecture as the Tesla V100 and performs single- precision FP operations at 14.8 TFLOPS/s, while the V100-SXM2 delivers 15.7 TFLOPS/s.
    - Our setup has a 6% advantage over Tohoku's

Benefits of Hierarchical Update

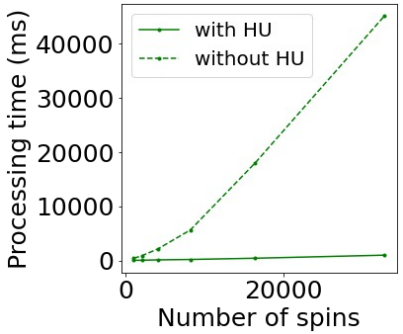
- 100 annealing steps with and without the hierarchical update strategy
- It is clear that the proposed HU strategy effectively accelerates the annealing process, especially when the number of spins (N) is large



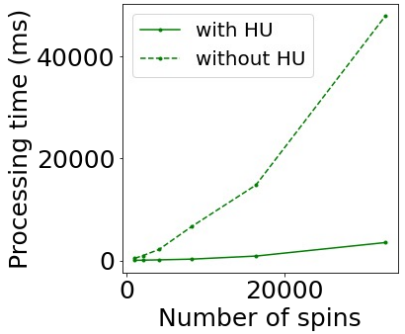
M = 4



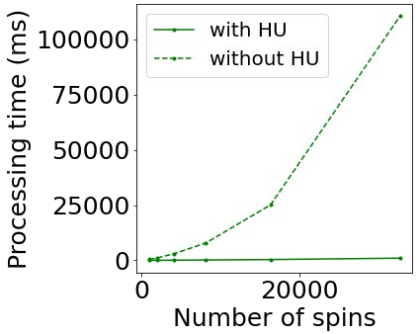
M = 8



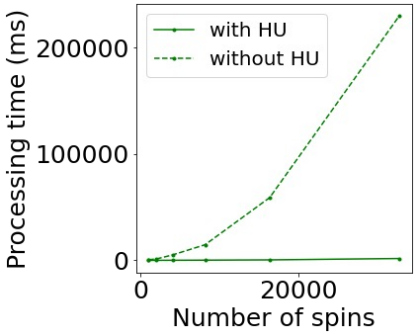
M = 16



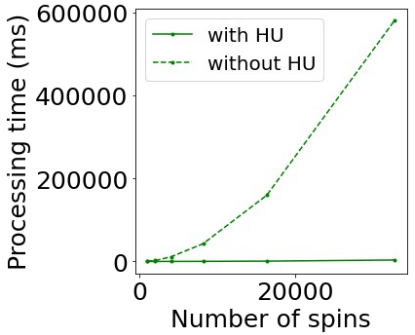
M = 32



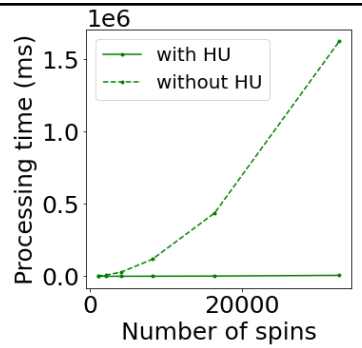
M = 64



M = 128



M = 256



M = 512

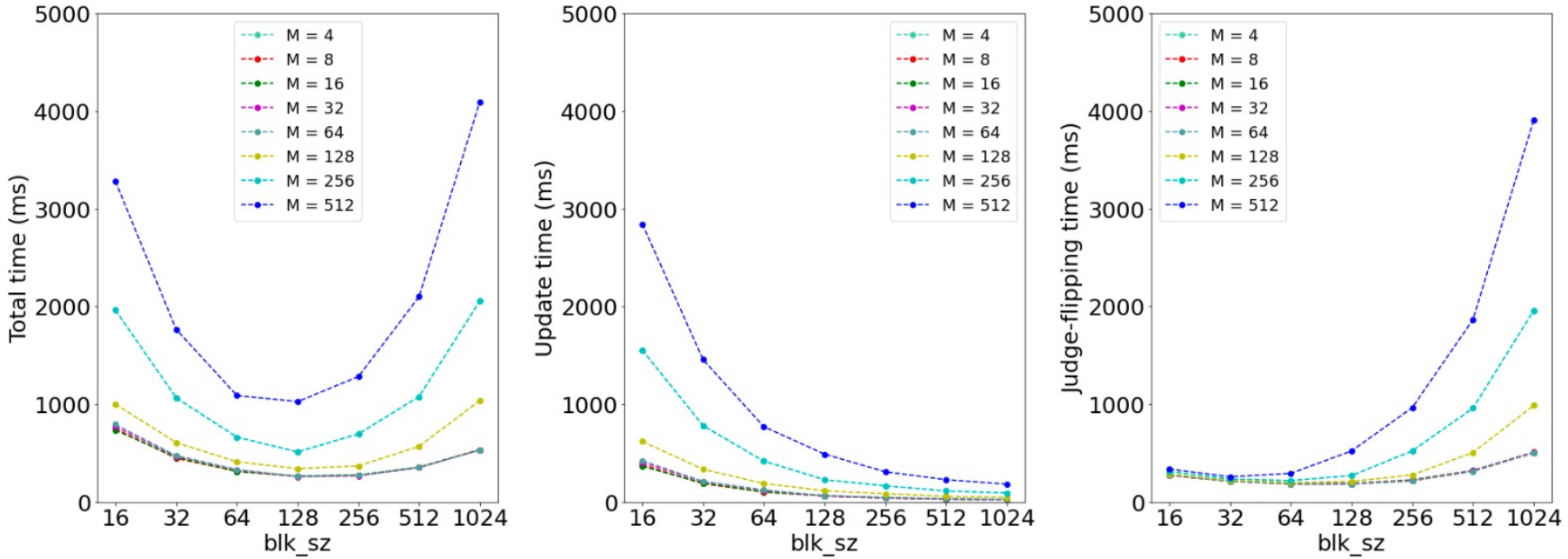
Per-Step Annealing Time

Method	N	M=4	M=16	M=64	M=256	M=512
Tohoku_GPU	4096	12.0 (1X)	15.0 (1X)	32.0 (1X)	215.0 (1X)	826.0 (1X)
Tohoku_FPGA		4.61 (2.6X)	5.11 (2.9X)	N/A	N/A	N/A
HU w/o TC		1.7 (6.9X)	1.8 (8.3X)	2.1 (15.1X)	4.4 (48.6X)	**6.9 (119.5X)**
HU w/ TC		1.7 (6.9X)	1.7 (8.8X)	1.8 (17.7X)	3.3 (65.2X)	**5.3 (156.0X)**
Tohoku_GPU	8192	35.0 (1X)	44.0 (1X)	75.0 (1X)	269.0 (1X)	898.0 (1X)
Tohoku_FPGA		17.45 (2.0X)	18.7 (2.4X)	N/A	N/A	N/A
HU w/o TC		3.4 (10.3X)	3.9 (11.2X)	4.7 (15.9X)	11.3 (23.9X)	18.6 (48.4X)
HU w/ TC		3.6 (9.8X)	3.5 (12.4X)	3.7 (20.1X)	8.1 (33.3X)	13.5 (66.7X)
Tohoku_GPU	16384	151.0 (1X)	169.0 (1X)	239.0 (1X)	1009.0 (1X)	2130.0 (1X)
Tohoku_FPGA		65.77 (2.3X)	70.33 (2.4X)	N/A	N/A	N/A
HU w/o TC		7.5 (20.3X)	9.3 (18.1X)	13.2 (18.1X)	32.1 (31.4X)	56.8 (37.5X)
HU w/ TC		7.4 (20.3X)	7.7 (22.0X)	8.9 (26.9X)	20.9 (48.4X)	36.6 (58.1X)
Tohoku_GPU	32768	1003.0 (1X)	1069.0 (1X)	1301.0 (1X)	4754.0 (1X)	8732.0 (1X)
Tohoku_FPGA		264.93 (3.8X)	N/A	N/A	N/A	N/A
HU w/o TC		19.7 (50.8X)	27.5 (38.9X)	37.1 (35.1X)	104.0 (45.7X)	**184.8 (47.2X)**
HU w/ TC		19.3 (52.0X)	20.6 (51.8X)	24.9 (52.2X)	59.8 (79.5X)	**100.9 (86.6X)**
Tohoku_GPU	65536	N/A	N/A	N/A	N/A	N/A
Tohoku_FPGA		N/A	N/A	N/A	N/A	N/A
HU w/o TC		54.1	82.8	129.6	376.6	683.1
HU w/ TC		54.2	62.0	81.9	203.2	352.8

- Unit: (ms)
- Our solution (HU w/o TC) is 7X to 48X faster than Tohoku GPU
- With the Tensor Cores, our solution (HU w/ TC) provides 7X to 86X speedup over the baseline
- As M grows, the acceleration from TCs becomes larger

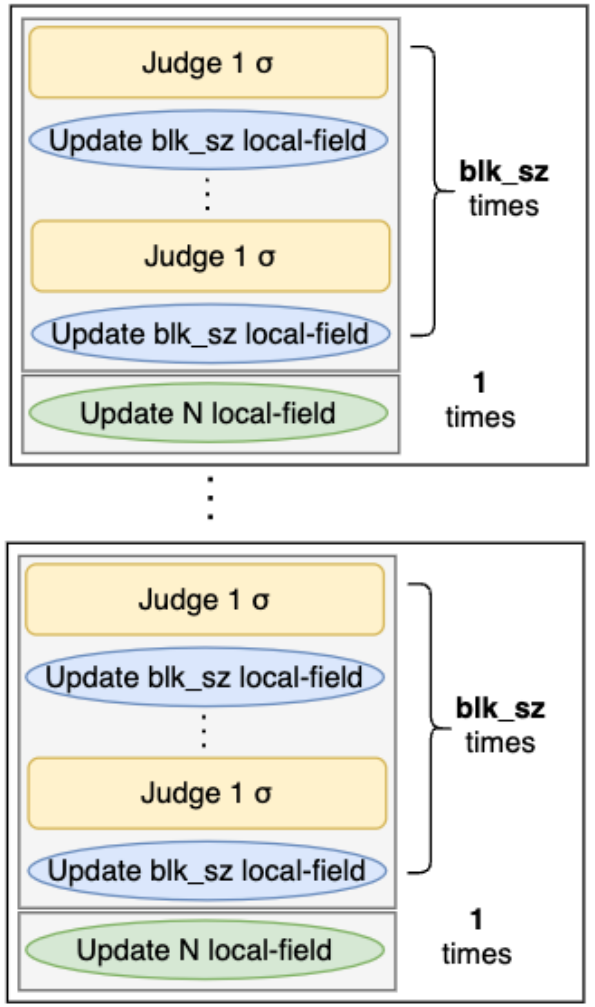
The Choice of *blk_sz*

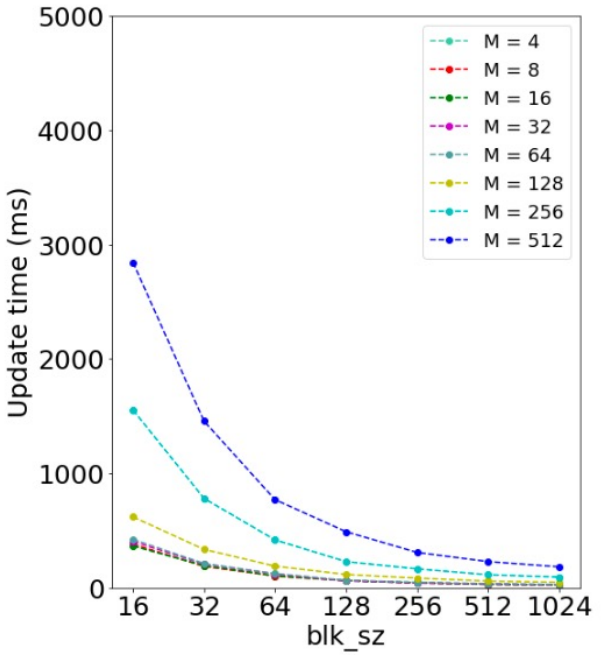
- The choice of *blk_sz* impacts the effectiveness of the performance with the hierarchical update
- To evaluate the impact of *blk_sz* , we profile the annealing time under various *blk_sz*
- 100 annealing steps, N =8192, M = [4, 512]



(a) Total-annealing time (b) Update_Local-field time (c) Judge_to_Flip time

Cont'd

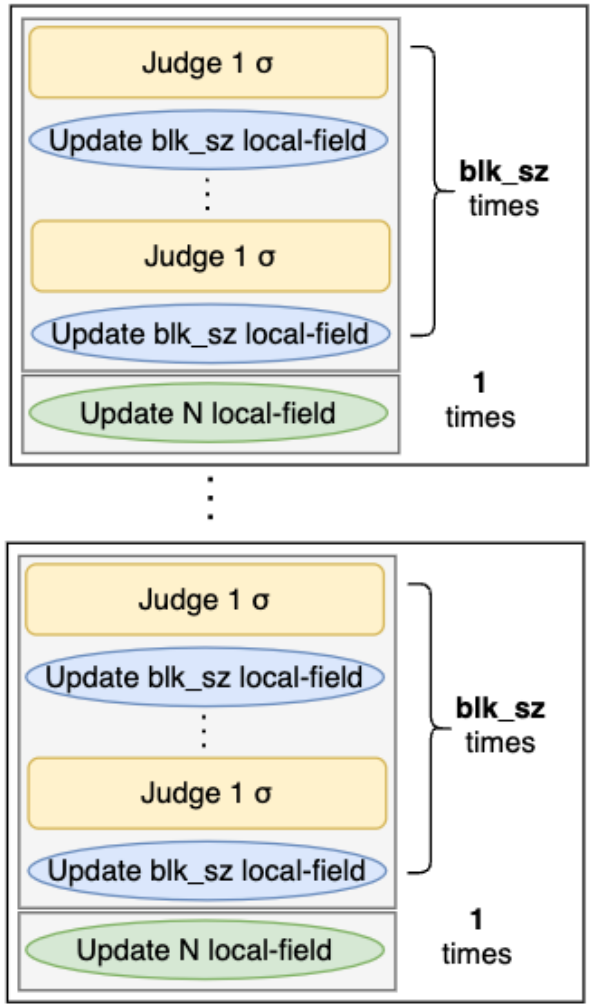


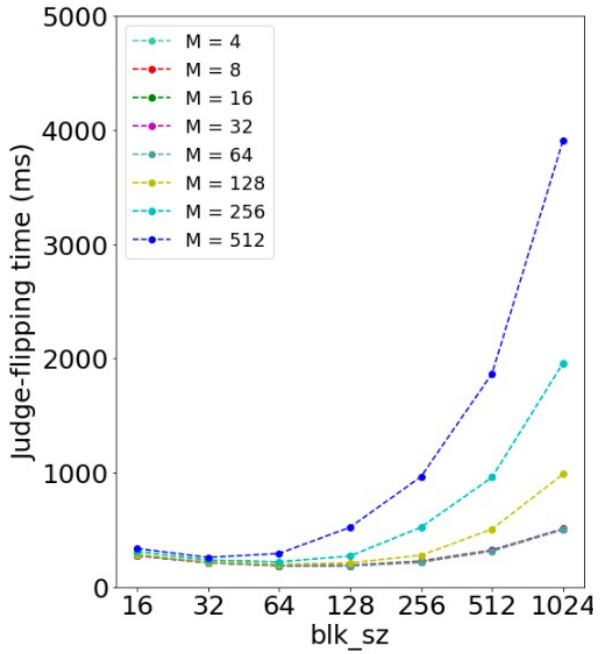


(b) Update_Local-field time

- As shown in Fig.b
  - The time required to update the local-field energies decreases as *blk_sz* increases
  - The larger *blk_sz* enables more efficient utilization of the Tensor Cores to perform the matrix multiplication

Cont'd

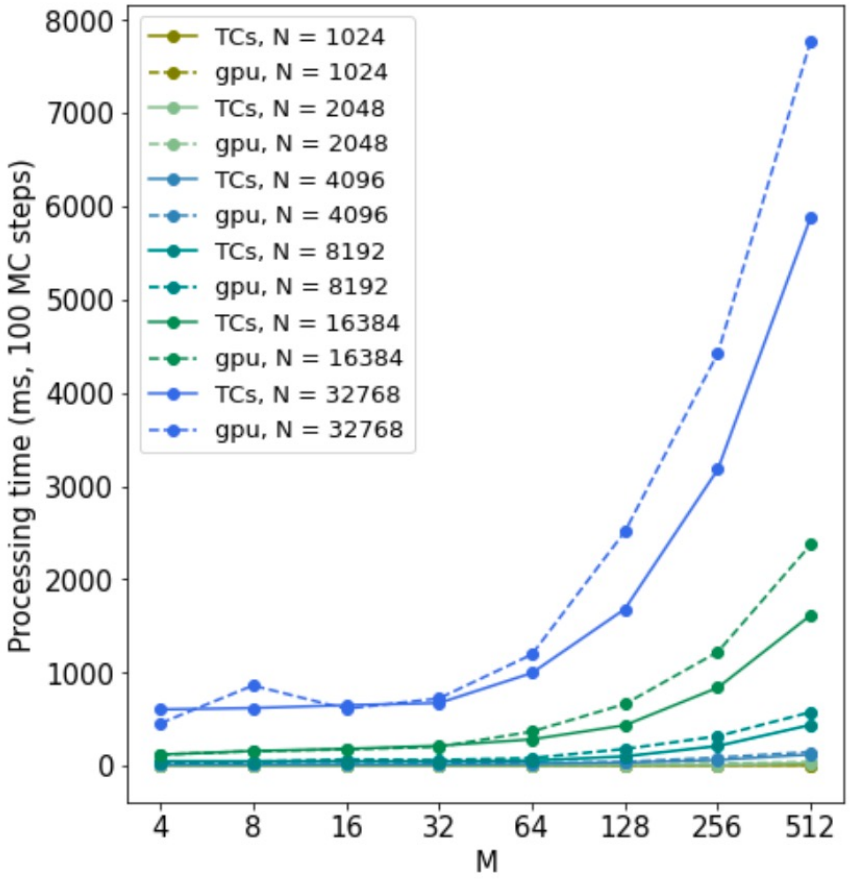




(c) Judge_to_Flip time

- Fig.c shows that the time required by Judge_to_Flip decreases initially, but increases when *blk_sz* becomes large
  - From *blk_sz* =16 – 128, the time increase, because the larger *blk_sz* reduces the the kernel launch and the GPU cores become fully utilize to perform the computing
  - From *blk_sz* = 128 – 1024, the time increase, when the *blk_sz* becomes larger, it takes more time to examine more spin-flip and requires more time to update the local-field within the same block
- The choice of *blk_sz* depends mainly on the system architecture. We set *blk_sz* to 128 to produce most of the experimental results in this paper

The Impact of Tensor Cores

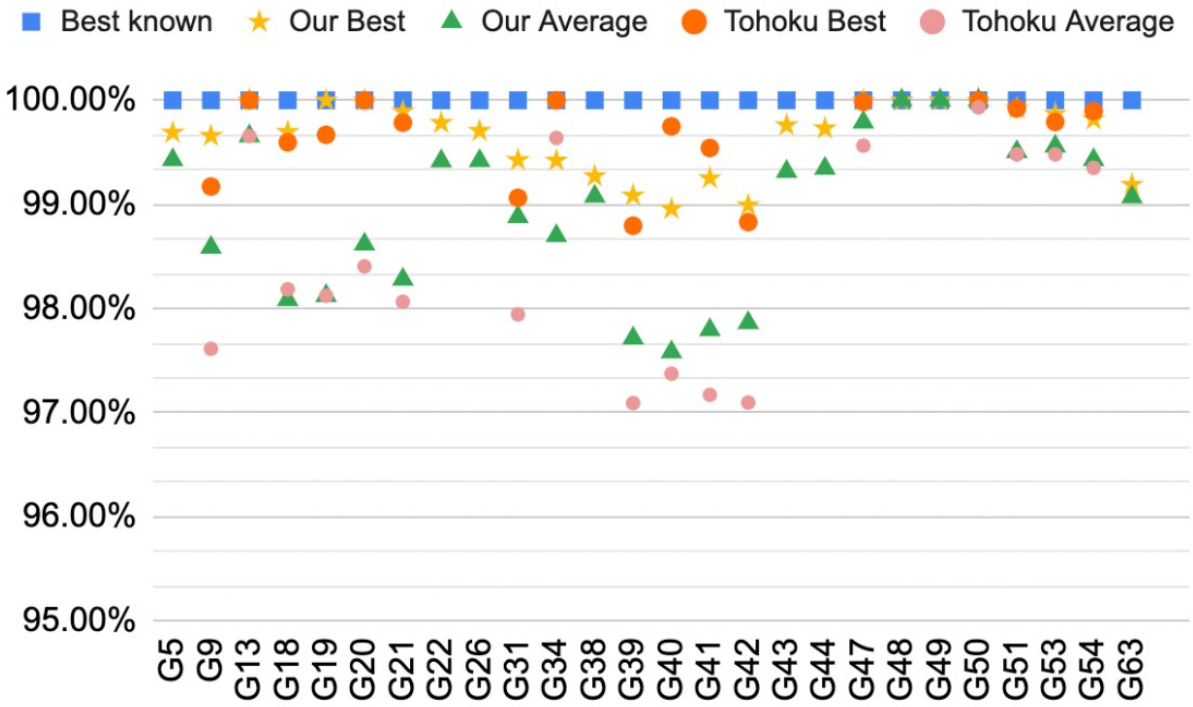


- Comparing the time cost by updating the local-field with or without using TCs
- Data types of coupling matrix and spin matrix: half precision, local-field matrix: single precision
- With N = [1024, 32768], M = [1, 512], $blk\_sz = 128$
- In those cases of $M \geq 32$, the programs accelerated by TCs are faster than those which use the cuda cores
  - Besides, as M grows, the acceleration by TCs becomes larger

The Quality of Solution

- Problem: Gset, MAXCUT, Stanford University
- Annealing steps: 1000 steps (the same as Tohoku)
- By adjusting T(0) and the decreasing value of the temperature per MC-step to find better solution quality

Gset	N	#Edges	Best Known	Gset	N	# Edges	Best Known
G5	800	19176	11631	G41	2000	11785	2405
G9	800	19176	2054	G42	2000	11779	2481
G13	800	1600	582	G43	1000	9990	6660
G18	800	4694	992	G44	1000	9990	6650
G19	800	4661	906	G47	1000	9990	6657
G20	800	4672	941	G48	3000	6000	6000
G21	800	4667	931	G49	3000	6000	6000
G26	2000	19990	13328	G50	3000	6000	5880
G31	2000	19990	3309	G51	1000	5909	3848
G34	2000	4000	1384	G53	1000	5914	3850
G38	2000	11779	7688	G54	1000	5916	3852
G39	2000	11778	2408	G63	7000	41459	27045



Conclusion

Conclusion

- Accelerating SQA algorithm to solve the Ising problems
- What we have done in this work
  - **Hierarchical Update** - Re-forming the SQA algorithm to implement the algorithm more parallelly
  - **Utilizing the Tensor Cores** - Not only **GPU** is used, **Tensor Cores (TCs)** provided by NVIDIA are also used to accelerate the algorithm in our work
- Original state of the art (SOTA) at that time
  - Both SOTAs accelerated by different accelerators are done by Tohoku University
  - **GPU-based:** A GPU-based quantum annealing simulator for fully-connected ising models utilizing spatial and temporal parallelism (2020, IEEE Access)
  - **FPGA-based:** Highly-parallel FPGA accelerator for simulated quantum annealing (2019, IEEE Transactions)
- Achievement
  - Faster than SOTA GPU-based solution: 1.7x - 86.6x
  - Faster than SOTA FPGA-based solution: 2.65x - 13.68x
  - Maintaining the solution quality no less than 97.5% (Gset (MAX-CUT) provided by Stanford)

Thank you for your time!

/

Any Questions/Ideas/Thoughts/Feedback?

Yi-Hua Chung
Contact me: r09944072@csie.ntu.edu.tw