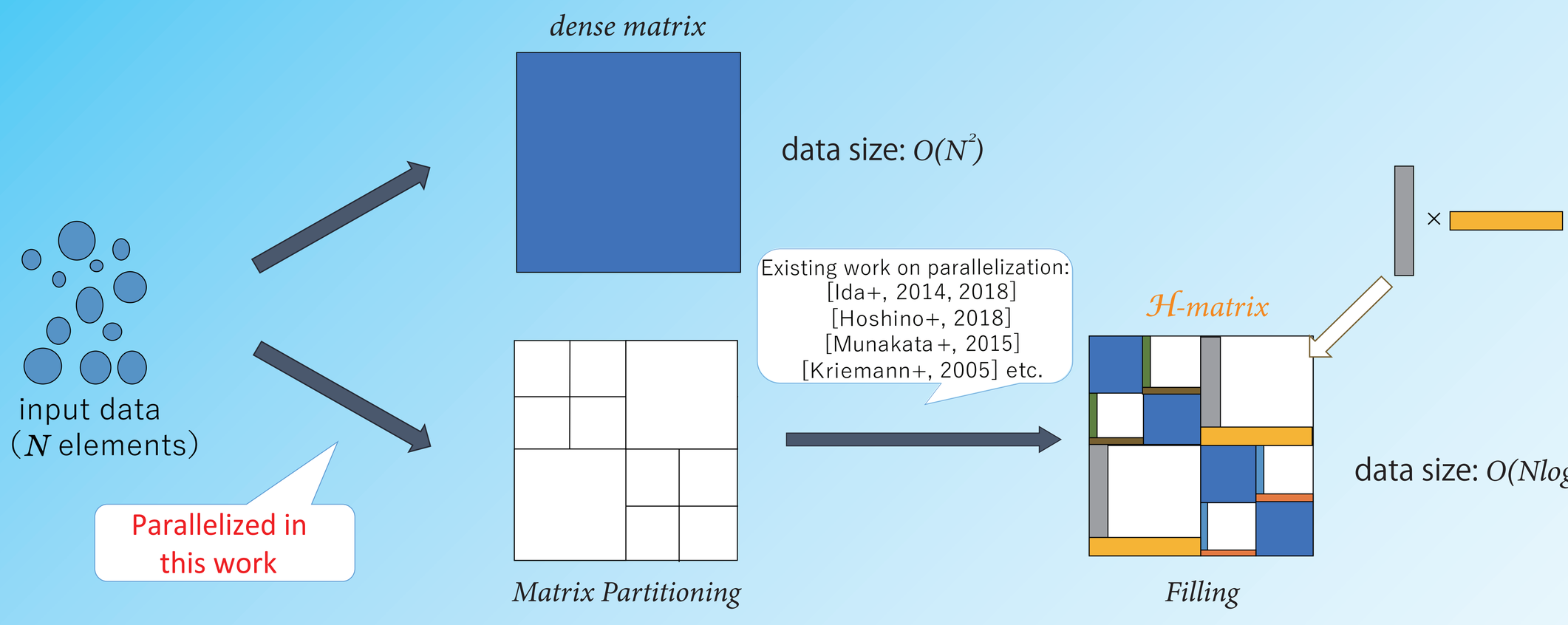
Implementation of Partitioning of Hierarchical Matrices using Task Parallel Languages

Zhengyang Bai[†1] Tasuku Hiraishi[†1] Hiroshi Nakashima[†1] Akihiro Ida[†2] Masahiro Yasugi[†3]

†1: Kyoto University †2: the University of Tokyo †3: Kyushu Institute of Technology

Hierarchical Matrix ($\mathcal{H}$ - matrix)

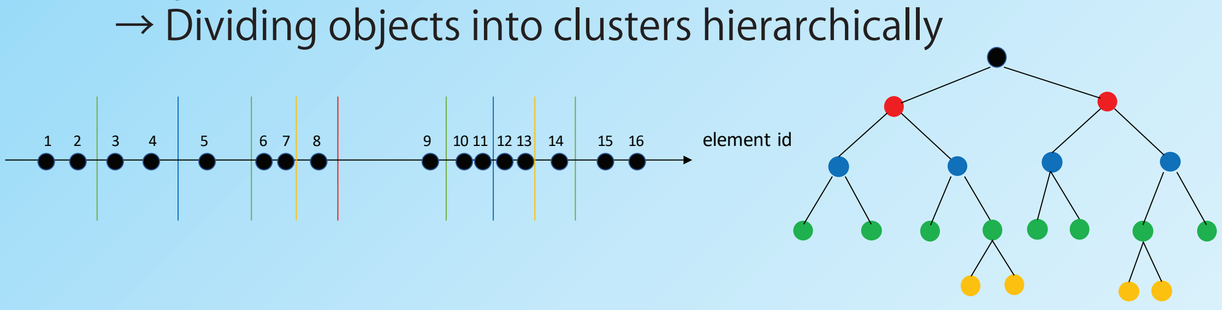
- An approximated form that represents $N \times N$ correlations of N objects



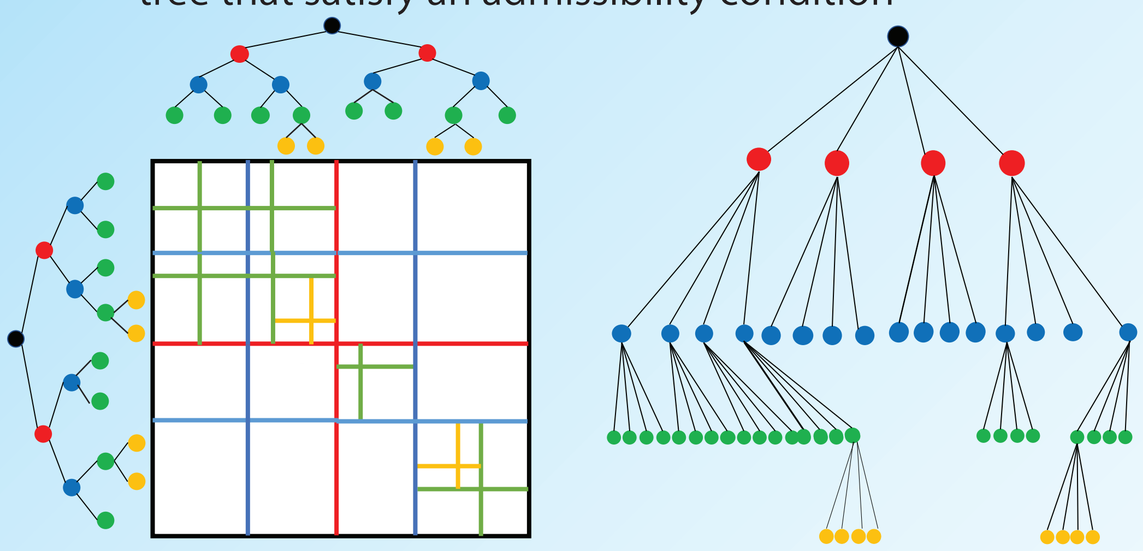
Parallelization of Matrix Partitioning using Task Parallel Languages

Matrix Partitioning

- Step 1: Cluster tree (CT) construction
  → Dividing objects into clusters hierarchically



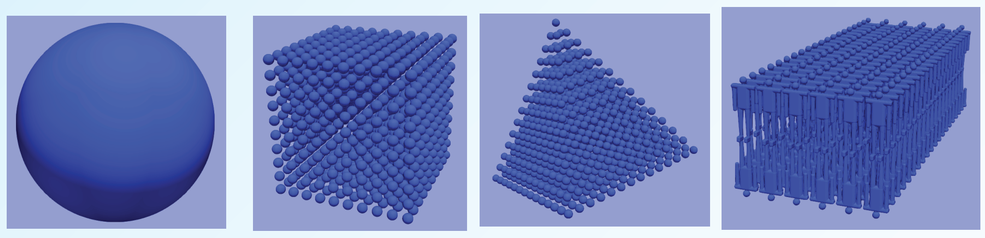
- Step 2: Block cluster tree (BCT) construction
  → Observe all cluster pairs at the same level of the cluster tree that satisfy an admissibility condition



- CT and BCT are unpredictably unbalanced
  → We used task parallel languages, Cilk Plus and Tascell, to get good load balance
  - Parallelized recursive calls using spawn/sync (Cilk Plus) or do_two (Tascell)
  - In CT construction, we also parallelized the inside of each recursive step

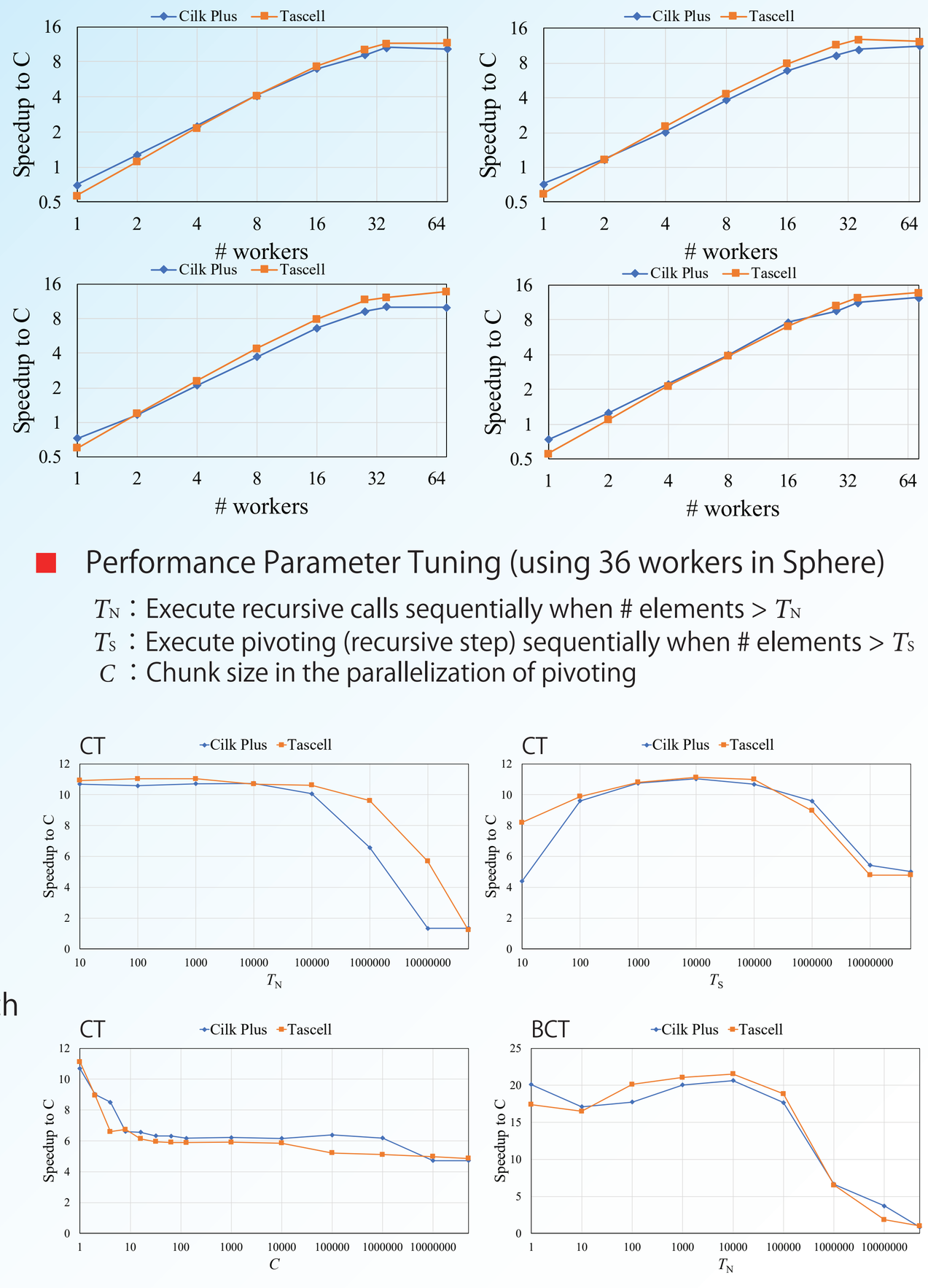
Evaluation

- Data sets



Sphere	SphereCube	SpherePyramid	Humans
# elements: 50,000,000	101,250,000	102,768,750	98,320,000

- Speedup (Xeon Broadwell 18 cores × 2 socket, HT enabled)



- Performance Parameter Tuning (using 36 workers in Sphere)

T_N : Execute recursive calls sequentially when # elements > T_N
T_S : Execute pivoting (recursive step) sequentially when # elements > T_S
C : Chunk size in the parallelization of pivoting

This work was supported in part by JSPS KAKENHI Grant Numbers JP17H01749, JP17K00099 and JP19H04087