Proposal of a Transparent Dynamic Core Allocation Method for MPI/OpenMP Hybrid Parallelization

Yusuke Ishihara (ishihara.yusuke.s4@dc.tohoku.ac.jp) [1] , Masatoshi Kawai[1] , Sichen Tao[1] , Hiroyuki Takizawa[1]
[1]Tohoku University

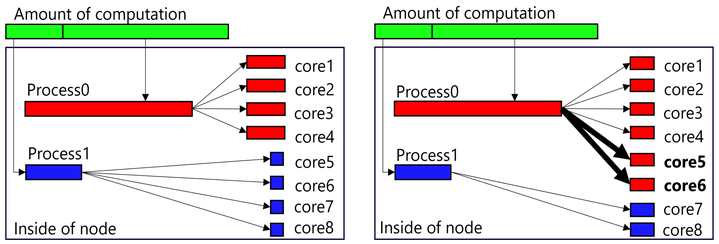
Background & Goal

■Load Imbalance in HPC

- Load balancing at both process and thread levels is critical for maximizing performance in MPI/OpenMP hybrid environments.
- Applications like H-matrices suffer from unpredictable load imbalances, resulting in CPU idle time and wasted power.

■DCB (Dynamic Core Balancing)

- DCB is a method that dynamically changes the number of allocated cores at runtime according to the computational load of each process.
- It achieves **both performance improvement and power saving** by allocating more cores to heavily loaded processes and putting surplus cores to sleep.



■Challenges

- Conventional DCB requires users to modify source code and manually insert API calls.
- This results in high implementation costs and low portability.

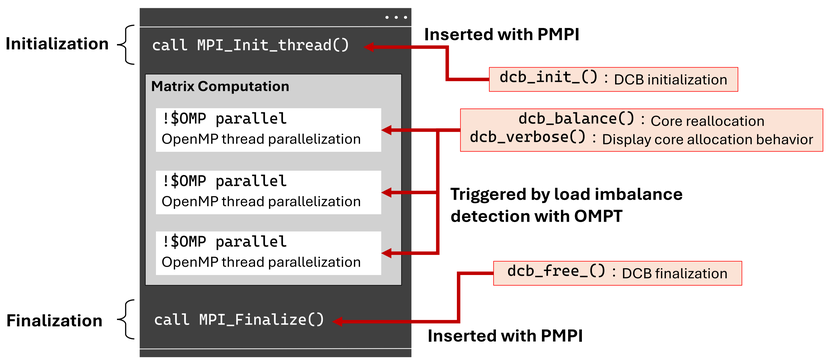
■Objectives

- Develop a **Transparent DCB** method requiring **zero source code modifications** to user programs.
- Maintain or improve performance through automatic load-aware core binding.

Proposed Method

■Approach

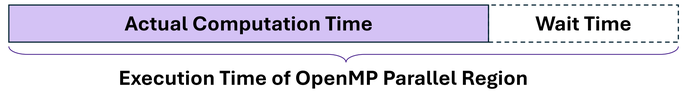
- Utilizes standard APIs, specifically OpenMP Tools (OMPT) and Profiling MPI (PMPI), to apply DCB transparently and automatically.



■Dynamic Parameter Calculation

- Eliminates user-dependent static parameters by dynamically calculating load from process execution times.

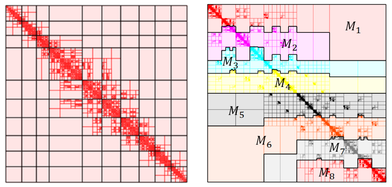
① Measure the execution time of OpenMP parallel regions using OMPT.
② Calculate **"actual computation time"** (load indicator) by subtracting PMPI-measured wait times from the OMPT-measured total time.



Evaluation

■Evaluation Environment

- **Hardware:** Miyabi-C Supercomputer (Intel Xeon CPU Max 9480, 112 cores/node, 128 GiB HBM2e)
- **Software:** Intel oneAPI DPC++/C++ Compiler 2023.2.0 / Intel MPI Library 2021.10.0 / OpenMP 5.1



Example of Lattice $\mathcal{H}$-matrix parallelization

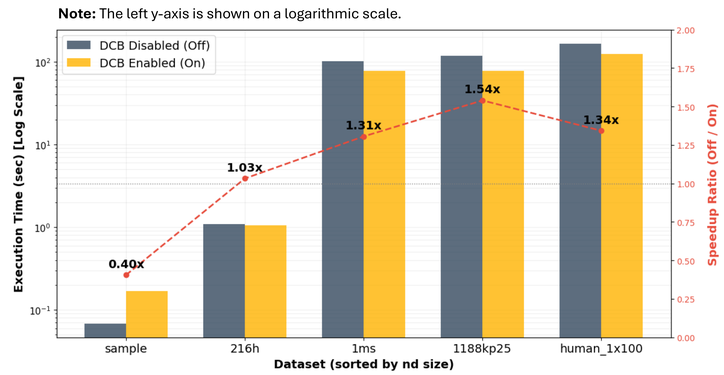
- **Target Application:** Lattice $\mathcal{H}$-matrix, selected for its highly irregular workload and inherent load imbalance.
- **Datasets:** Utilized five datasets with varying problem sizes (nd**: 600 to 1.96M**) to evaluate load balancing effectiveness under different scales.

Dataset	Unknowns (nd)
input_sample.pbf	600
input_216h.pbf	21,600
input_1ms.pbf	1,004,400
input_1188kp25.pbf	1,188,000
input_human_1x100.pbf	1,966,400

■Single-Node Performance

- Achieves up to a **1.54x speedup** on large datasets compared to execution without DCB.
- Reduces total energy consumption by up to **32.0%**; although higher CPU utilization increases active power draw, the significantly shorter execution time yields overall savings.
- For the smallest dataset, the runtime overhead of OMPT/PMPI interception and core reallocation may outweigh the benefit of DCB due to its short execution time.

Note: The left y-axis is shown on a logarithmic scale.

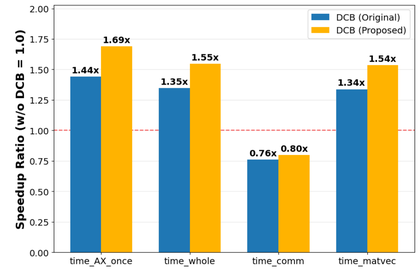


■Multi-Node Performance

- Reduces total execution time by approximately **6.1%** in a 16-node (1,792 cores) environment.
- Decreases total system energy consumption by **10.6%**, demonstrating an effective performance-power balance in large-scale distributed memory environments.

■Comparison with Conventional DCB

- Achieves a **1.55x speedup**, outperforming the conventional manual method (**1.35x**) that relies on static parameters.
- Enables highly accurate load balancing by utilizing the **"actual computation time"** of each process as a load indicator.



- The increased MPI time may result from changes in process synchronization after core reallocation; further profiling is required to identify the dominant factor.

Conclusions & Future Work

■Conclusion

- **Establishes a transparent DCB method** via OMPT and PMPI integration, requiring no source code modifications.
- **Eliminates adoption barriers** for existing applications, outperforming conventional DCB with static parameters while improving system-wide energy efficiency.

■Future Work

- Optimize execution thresholds dynamically to minimize runtime overhead for extremely small-scale problems.
- Integrate hardware performance counters to utilize cache miss rates and memory bandwidth as high-precision load indicators.
- Explore diverse MPI+OpenMP process mapping patterns and further evaluate scalability in larger multi-node environments.

References

[1]Kawai, M., Ida, A., Hanawa, T. and Nakajima, K.: Dynamic Core Binding for Load Balancing of Applications Parallelized with MPI/OpenMP, Computational Science – ICCS 2023, Cham, Springer Nature Switzerland, pp. 378–394 (2023).
[2]Kawai, M., Ida, A., Hanawa, T. and Hoshino, T.: Optimize Efficiency of Utilizing Systems by Dynamic Core Binding, International Conference on High Performance Computing in Asia-Pacific Region Workshops (HPCAsia WS 2024), pp. 1–10 (2024).
[3]Ida, A.: Lattice H-matrices on distributed-memory systems, IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 389–398 (2018).
[4]Eichenberger, A. E., Mellor-Crummey, J., Schulz, M., Bates, M., Chaimov, K., Copty, S., DelSignore, J., Elken, R., Fried, S., Krentel, M. et al.: OMPT: An OpenMP tools application programming interface for performance analysis, International Workshop on OpenMP, Springer, pp. 171–185 (2013).
[5]University of Tokyo Information Technology Center: Supercomputer System Miyabi, https://www.cc.u-tokyo.ac.jp/supercomputer/miyabi/service/. (Accessed: July 31, 2026).

■Acknowledgments

The authors utilized ChatGPT (OpenAI) and Gemini (Google) for proofreading and language refinement; however, the authors bear full responsibility for the final content