A Locality and Memory Congestion-aware Thread Mapping Method for Modern NUMA Systems
Mulya Agung, Muhammad Alfian Amrizal, Ryusuke Egawa, and Hiroyuki Takizawa

Background: Memory congestion in modern NUMA systems
- Non-uniform Memory Access (NUMA) Architecture is commonly used in HPC systems.
- Memory access in NUMA systems is not uniform.
- Remote-access communication is slower than local-access communication.
- Modern NUMA systems are susceptible to congested threads.
- Current CPUs can cause a massive load to the memory congestion.
- Maximizing locality can hurt performance due to the memory congestion.

Traffic congestion on memory controllers
A node (processor) consists of a set of CPUs that physically is associated with memory controllers and memory devices.

Thread Mapping
A technique to map the application's threads to CPU cores.

Locality and memory congestion-aware thread mapping
- Identifies the groups of threads from time-series data of communication using a weight k-means clustering method.
- Given a set of communication timestamps \((t_1, t_2, ..., t_n)\) and a set of clusters \((C_1, C_2, ..., C_k)\), the objective function: \(\sum_{i=1}^{k} \sum_{t \in C_i} Acomm(t) \cdot f(t)\), where \(Acomm\) is the number of communications.

Decongested Thread Locality (DeTLoc)
- A communication event (E): a pair of threads, timestamp, ...
- The communication events are detected from two consecutive access windows (at least one write) by different threads to the same cache line: A-current thread, B-Previous(A)
- The communication events will be treated as follows:
  - A-Write: The number of communications to the same cache line.
  - A-Read: The number of communications to the same cache line.

Step 1: Detecting communications between threads

Communication events
- The communication events detected from two consecutive access windows (at least one write) by different threads to the same cache line: A-current thread, B-previous(A).

Step 2: Identifying the group of threads that potentially cause congestion
- Identifies the groups of threads from time-series of communication data using a weighted k-means clustering method.
- Given a set of communication timestamps \((t_1, t_2, ..., t_n)\) and a set of clusters \((C_1, C_2, ..., C_k)\), the objective function: \(\sum_{i=1}^{k} \sum_{t \in C_i} Acomm(t) \cdot f(t)\), where \(Acomm\) is the number of communications.

Experiment results
- Performance monitoring results of the CG application.
- Ongoing work:
  - Evaluating the proposed method with larger problem sizes.
  - Evaluating the characteristics of applications that can benefit from the proposed method.

References

The authors would like to thank Kazuhiko Komatsu for valuable discussions on this work. This research is partially supported by JST CREST “An Evolutionary Approach to Construction of a Software Development Environment for Massively-Parallel Heterogeneous Systems”, and Grant-in-Aid for Scientific ResearchB(#16H06282).