

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/330244137>

## A Locality and Memory Congestion-aware Thread Mapping Method for Modern NUMA Systems

Poster · November 2018

CITATIONS  
0

4 authors, including:



Mulya Agung

Tohoku University


13 PUBLICATIONS 10 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Checkpoint Restart Technologies for Hierarchical Storages (Japanese: 階層型ストレージのチェックポイントリスタート技術) [View project](#)

READS  
57



Muhammad Alfian Amrizal

Tohoku University

19 PUBLICATIONS 20 CITATIONS

SEE PROFILE



# A Locality and Memory Congestion-aware Thread Mapping Method for Modern NUMA Systems

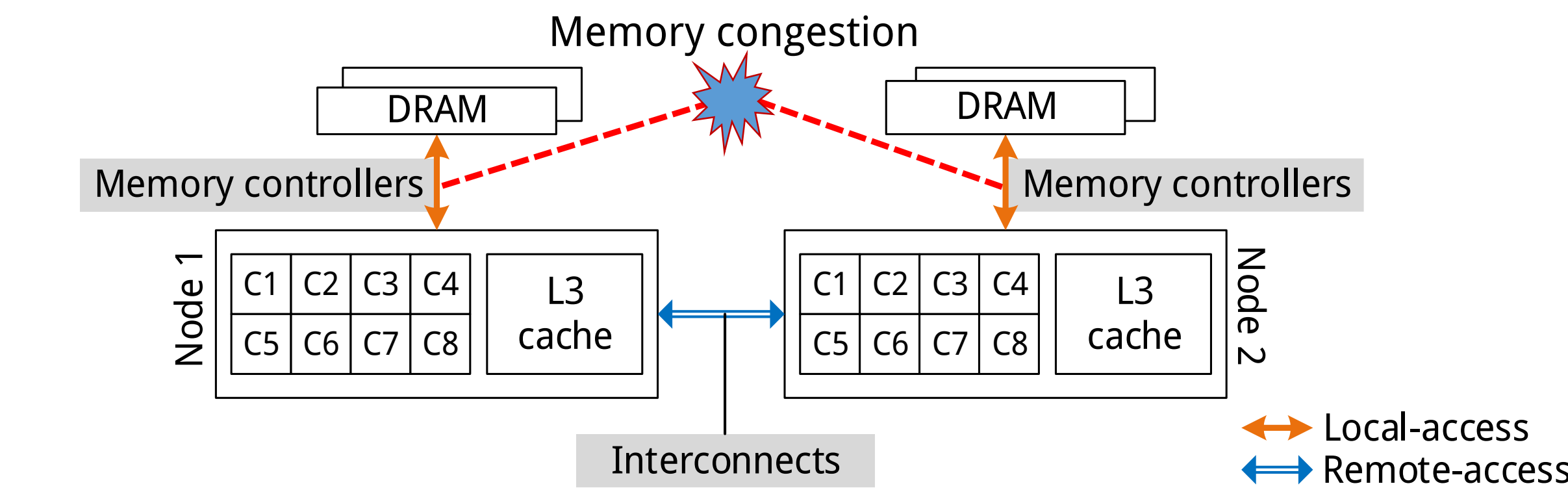
Mulya Agung, Muhammad Alfian Amrizal, Ryusuke Egawa, and Hiroyuki Takizawa

## Background: Memory congestion in modern NUMA systems

- **Non-unified 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 congestion**
  - ⇒ Current CPUs can cause a massive load to the memory controllers
  - ⇒ Maximizing locality can hurt performance due to the memory congestion<sup>[1]</sup>

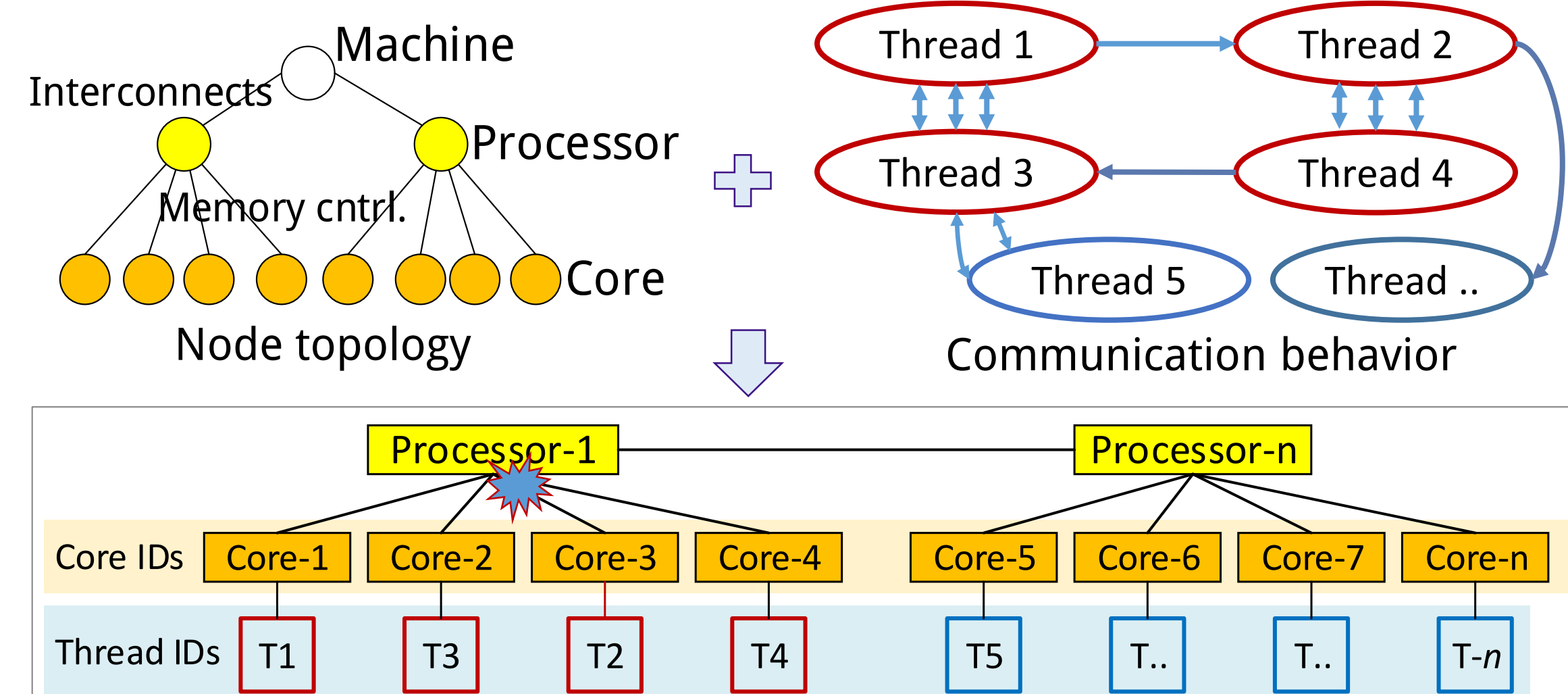
## Traffic congestion on memory controllers

A **node** (processor) consists of a set of CPU cores that is physically associated with memory controllers and memory devices.



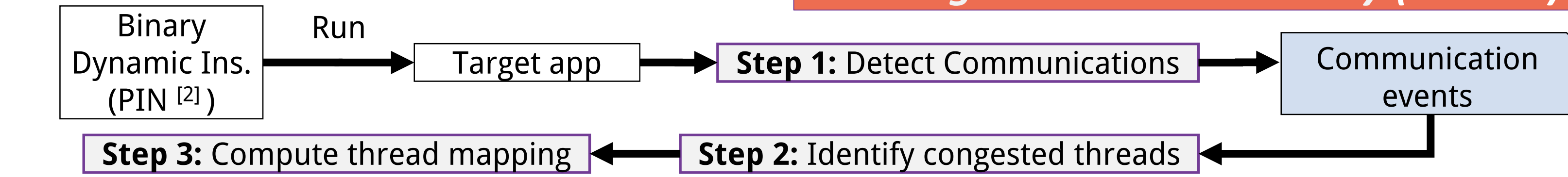
## Thread Mapping

A technique to map the application's threads to CPU cores

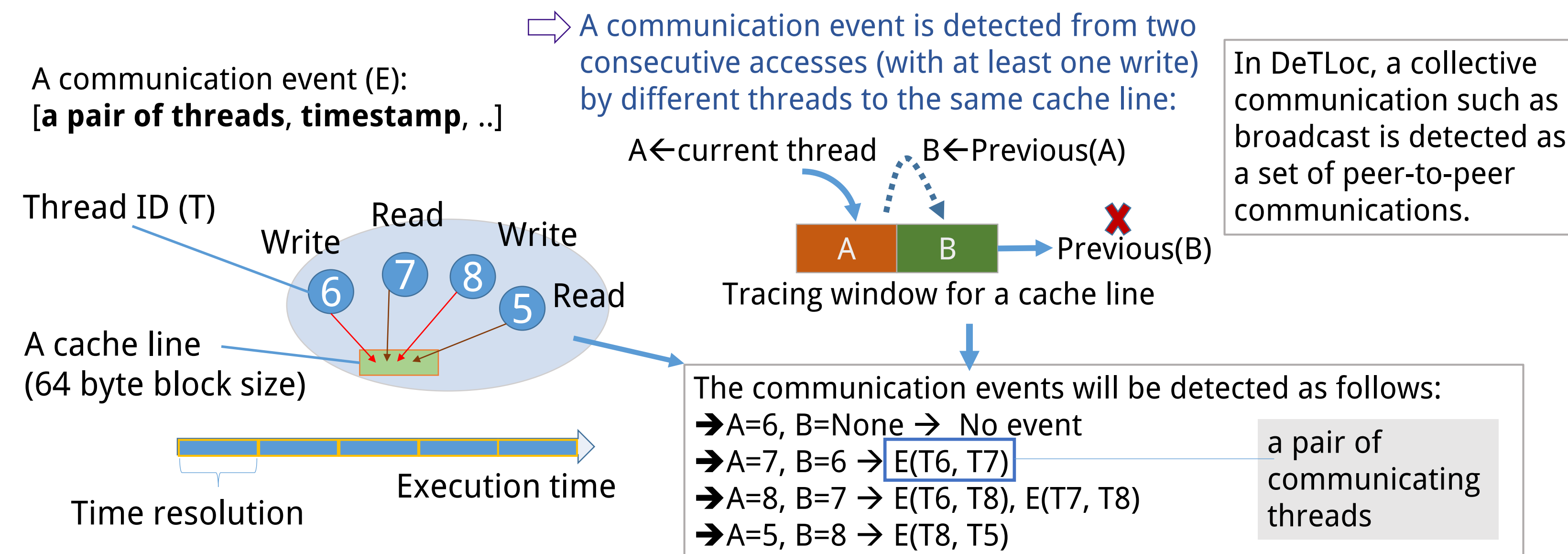


## Locality and memory congestion-aware thread mapping

### Decongested Thread Locality (DeTLoc)



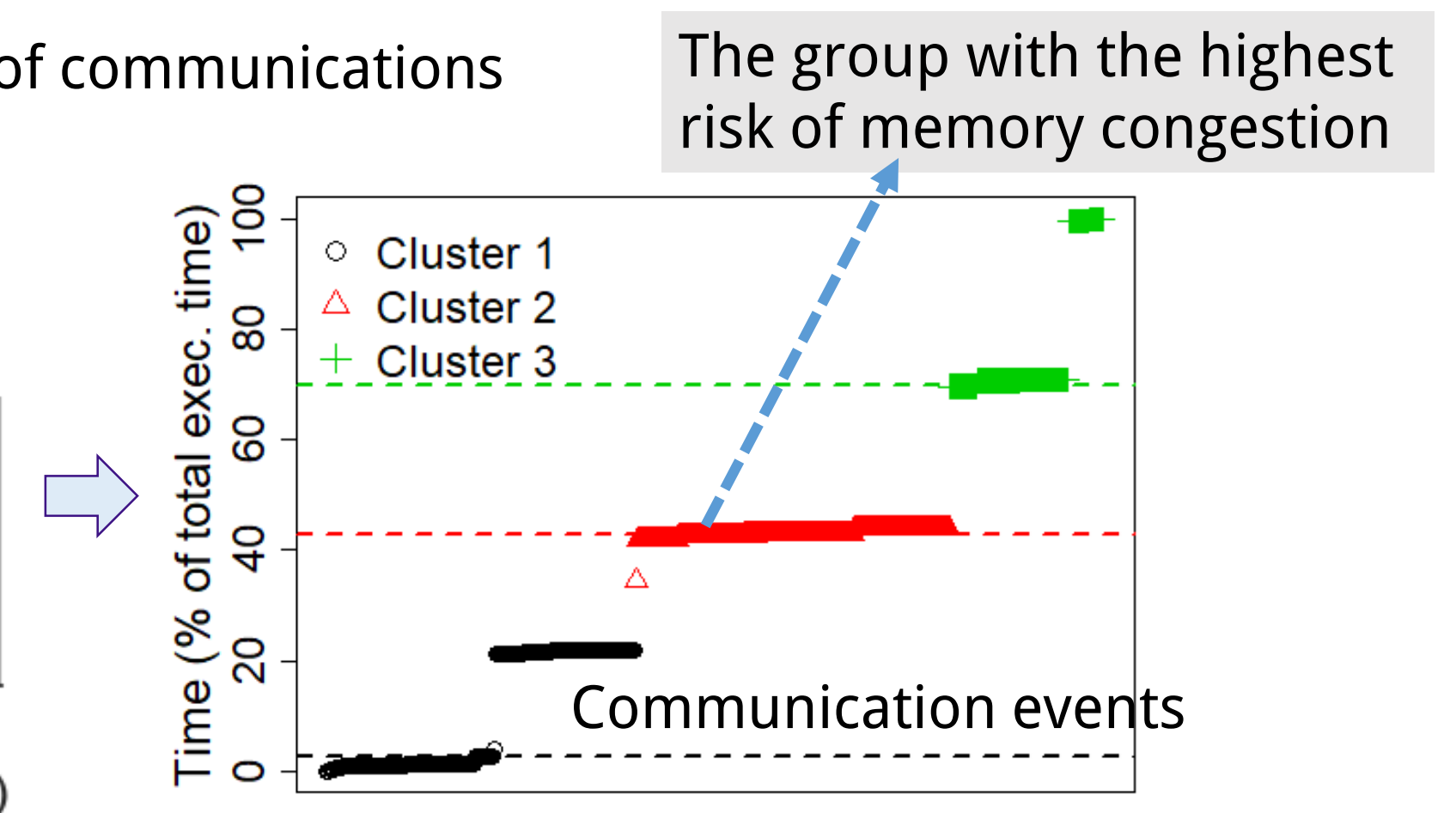
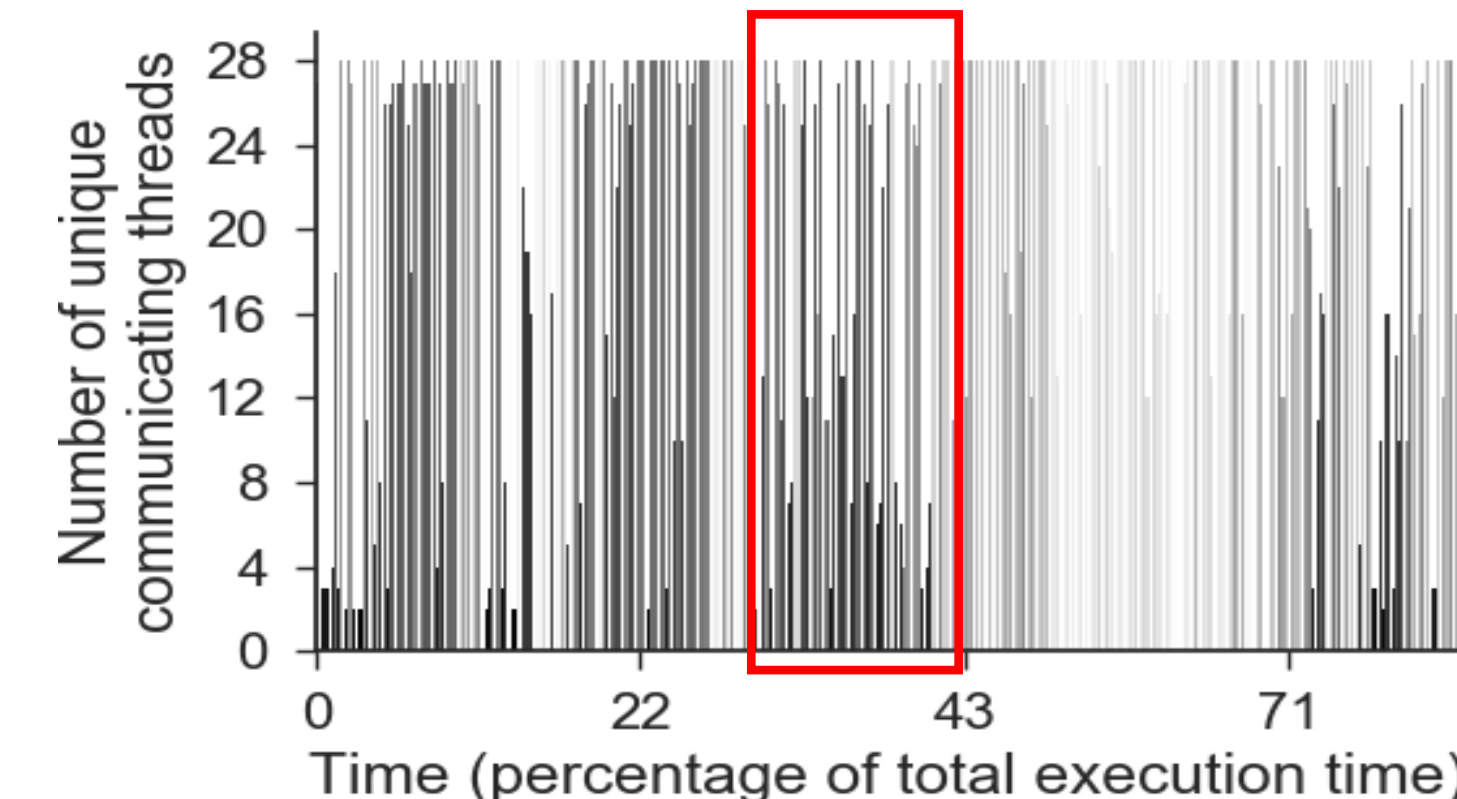
### Step 1: Detecting communications between threads



### Step 2: Identifying the group of threads that potentially cause the congestion

- Identify the groups of threads from time-series data of communication events using a weighted k-means clustering method.
- Given a set of communication timestamps  $\{t_1, t_2, \dots, t_n\}$  and a set of clusters  $\{C_1, C_2, \dots, C_k\}$ , the objective function of the k-means:

$$\sum_{i=1}^k \sum_{x \in C_i} Acomm_t \|t - \bar{C}_i\|^2, \text{ } Acomm: \text{the number of communications}$$

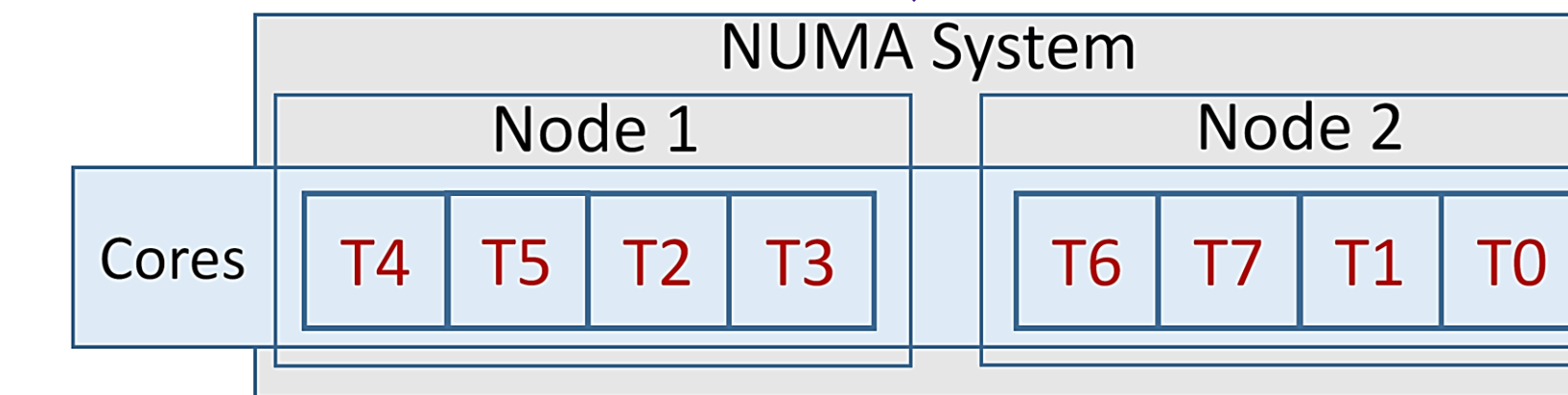


## Step 3: Computing the thread mapping

From the highest to the lowest  $L_c$

Group-2	Group-3
Pair-3 (T4, T5)	Pair-2 (T2, T3)
Pair-4 (T6, T7)	Pair-5 (T0, T1)
Pair-2 (T2, T3)	Pair-4 (T6, T7)
Pair-1 (T1, T2)	Pair-1 (T1, T2)

From the highest to the lowest  $W_p$



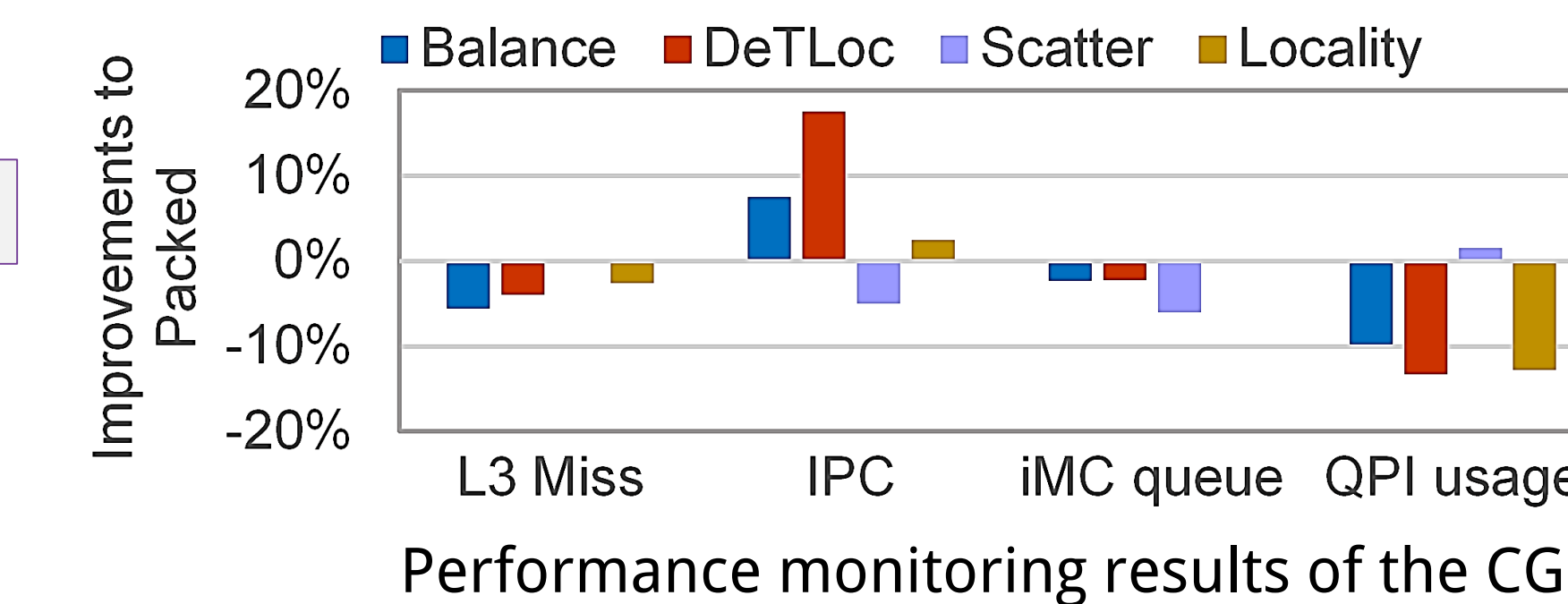
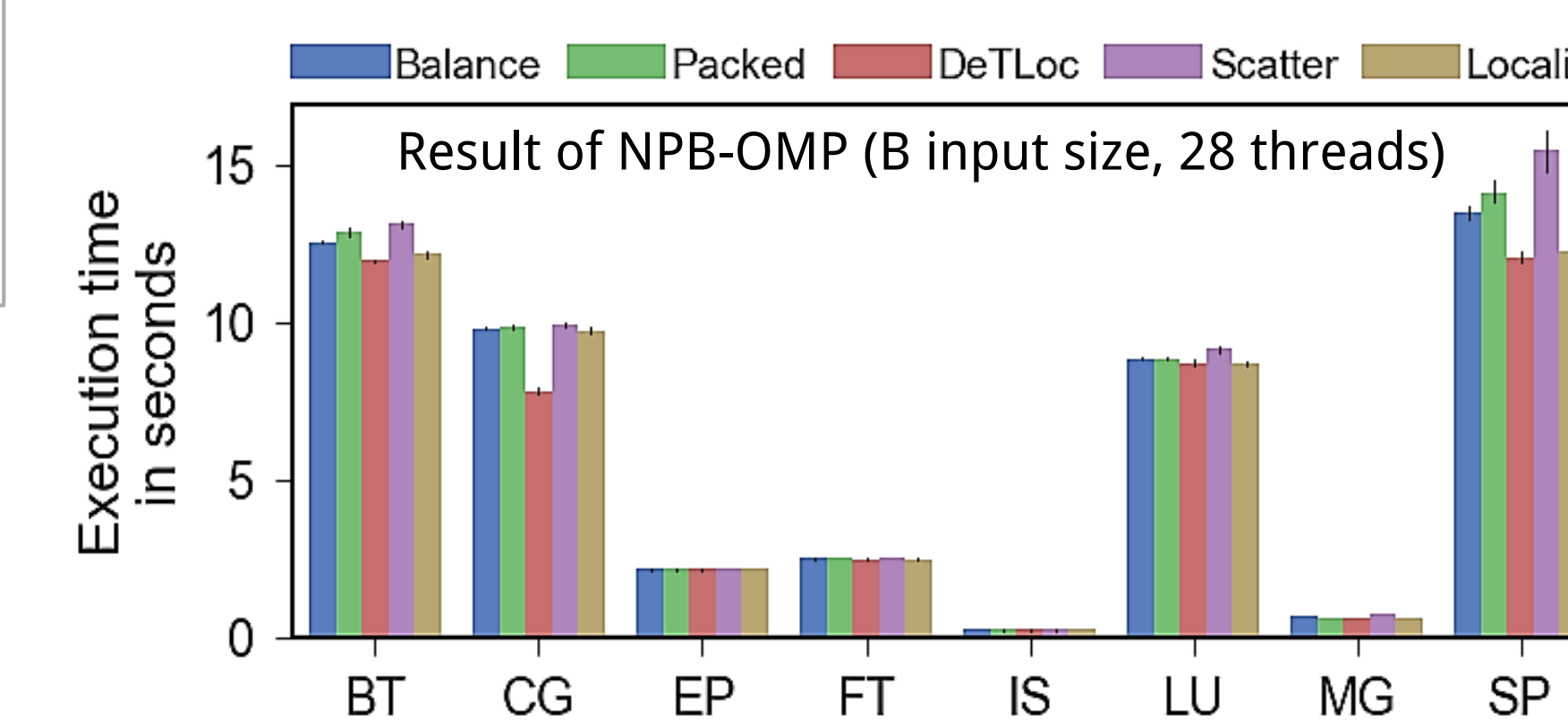
$$W_p = \frac{Acomm_p}{\sum_{i=1}^P Acomm_i}, L_c = \sum_{i=1}^{P_c} W_{p_i}$$

$P_c$ : number of thread pairs in group  
 $P$ : total number of pairs

**Idea:** map a thread pair to the same node (**improve locality**), while also mapping the thread pairs in the same group to the different nodes to spread the loads of the group (**reduce memory congestion**).

Note that a thread can belong to different pairs in one group. In such a case, avoiding the congestion will increase the number of remote-accesses. If the congestion level of the target application is low, the performance will become worse. We will discuss this issue in our future work.

## Experiment results



- Experiments on a 2-node Intel-based NUMA system with two Intel Xeon E5-2680v4 processors, and each node consists 14 cores (2.4 GHz) and 64 GB RAM.
- DeTLoc has been compared with **Packed**, **Scatter**, **Balance**-based, and **Locality**-based mapping methods.

## Conclusions

On average, DeTLoc can achieve shorter execution times than that of the other methods by simultaneously reducing the memory congestion and the number of remote-access. For a heavily congested application such as CG, DeTLoc can achieve 20% performance improvement compared with locality-based and balance-based thread mapping.

## Ongoing works

- Evaluating the proposed method with larger nodes and larger problem sizes
- Evaluating the characteristics of applications that can benefit from the proposed method

## References

- [1] Mulya Agung, Muhammad A. Amrizal, Kazuhiko Komatsu, Ryusuke Egawa, and Hiroyuki Takizawa. 2017. A Memory Congestion-Aware MPI Process Placement for Modern NUMA Systems. In 2017 IEEE 24th International Conference on High Performance Computing (HiPC). 152–161.
- [2] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. 2005. Pin: building customized program analysis tools with dynamic instrumentation. SIGPLAN Not. 40, 6 (June 2005), 190–200.

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 Research(B) #16H0282

## Acknowledgements