Memory-aware Task Mapping for Heterogeneous Multi-Core Systems

Yifan Jin
Tohoku University
Japan
jinyifan@hpc.is.tohoku.ac.jp

Mulya Agung
The University of Edinburgh
UK
mulya.agung@ed.ac.uk

Jiaheng Liu
Tohoku University
Japan
jiaheng@hpc.is.tohoku.ac.jp

Hiroyuki Takizawa
Tohoku University
Japan
takizawa@tohoku.ac.jp

1. INTRODUCTION

As a new trend in processor design, a multi-core processor has evolved to employ a heterogeneous multi-core architecture integrating some kinds of cores with different performance and energy characteristics on a single chip. In addition, the Non-Uniform Memory Access (NUMA) architecture has become a de facto standard in modern HPC systems. Almost all large-scale computing systems can be regarded as NUMA systems.

Because task mapping could significantly affect the usage of systems' heterogeneity and memory resources, proper task mapping will be a key to high performance and energy efficiency.

2. MOTIVATION

This work focuses on NUMA systems built with heterogeneous multi-core processors. Most of the conventional studies on NUMA awareness mainly assume to use homogeneous processors and/or processor cores, while most of the conventional studies on heterogeneous multi-core architectures mainly assume the homogeneous interconnection among cores [1, 2]. However, due to the emergence of the new system configuration, a task mapping method will no longer be suitable for the new system configuration unless it simultaneously considers both two factors, NUMA memory awareness and core heterogeneity.

The objective of this work is to deal with the task mapping problem on the new system configuration by properly combining memory-aware task mapping and heterogeneity-aware task mapping.

3. APPROACH

The basic workflow of the task mapping strategy proposed in this work is shown in Figure 1. Considering the diversity of system configuration and application characteristics, this work proposes a task mapping strategy that switches between two priority options to determine the best mapping for the target application on the target system.

The two proposed priority options are the memory-aware priority option and the heterogeneity-aware priority option. Choosing one of the priority options will prioritize the impact of that factor at the task mapping. However, it does not mean that the influence of the other factor on the mapping result is completely ignored because some applications might be sensitive to both memory and core performance. Our proposed priority option switching mechanism selects appropriate priority options for individual systems and applications considering their performance characteristics.

4. EVALUATIONS

Figure 2 shows some evaluation results on a simulator. We simulate a NUMA system consisting of two types of cores with different clock speeds. Two applications with different characteristics (Rodinia-LavaMD and Splash2-fft) are executed on this system

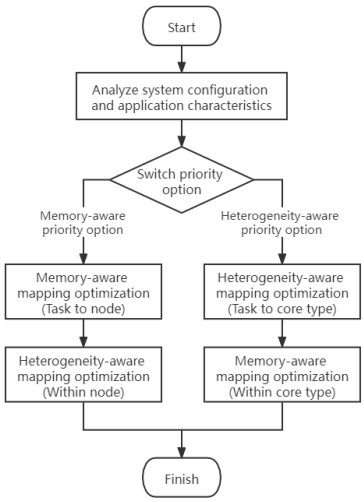

Figure 1. Workflow of the proposed mapping strategy

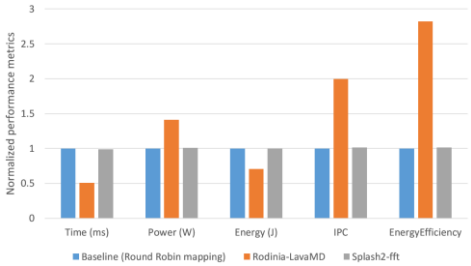

Figure 2. Performance and energy impacts of using the heterogeneity-aware priority on different applications

using the proposed heterogeneity-aware priority option. Compared with the round-robin mapping policy as baseline, Rodinia-LavaMD can get significant improvements in performance and energy efficiency while the Splash2-fft cannot. The results show that heterogeneity-aware priority option is suitable only for the former application. As also shown in our previous work [1], the memory-aware priority option is suitable only for some of the applications, but not for all. We have shown that only one priority option cannot provide a suitable mapping for all kinds of applications. This shows the necessity of the two mapping priority options and the switching mechanism proposed in this work. Our work will also investigate the key performance characteristics of applications that can be used to determine the mapping priority.

REFERENCES

[1] Agung, M., Amrizal, M. A., Egawa, R., & Takizawa, H. (2020). Deloc: A locality and memory-congestion-aware task mapping method for modern numa systems. *IEEE Access*, *8*, 6937-6953.

[2] Ding, J. H., Chang, Y. T., Guo, Z. D., Li, K. C., & Chung, Y. C. (2014). An efficient and comprehensive scheduler on Asymmetric Multicore Architecture systems. *Journal of Systems Architecture*, 60(3), 305-314.