Proceedings Paper:

Gao, yiyang, Chen, Haoran, Fang, Xinwei orcid.org/0000-0003-3630-2249 et al. (1 more author) (2024) A Structure-Aware DAG Scheduling and Allocation on Heterogeneous Multicore Systems. In: 2024 IEEE 14th International Symposium on Industrial Embedded Systems (SIES). 2024 IEEE 14th International Symposium on Industrial Embedded Systems, 23-25 Oct 2024 IEEE , CHN , pp. 26-33.

https://doi.org/10.1109/SIES62473.2024.10767927

A Structure-aware DAG Scheduling and Allocation on Heterogeneous Multicore Systems

Yiyang Gao[1], Huixuan Yi[1], Haoran Chen[1], Xinwei Fang[2], Shuai Zhao[1]
[1] School of Computer Science and Engineering, Sun Yat-sen University, China
[2] Department of Computer Science, University of York, UK
{gaoyy26, yihx3, chenhr56}@mail2.sysu.edu.cn, xinwei.fang@york.ac.uk, zhaosh56@mail.sysu.edu.cn

2024 IEEE 14th International Symposium on Industrial Embedded Systems (SIES) | 979-8-3503-8863-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/SIES62473.2024.10767927

***Abstract*—With the ever-increasing complexity of real-time applications, heterogeneous architectures are often applied, with tasks modelled as a Direct Acyclic Graph (DAG) to reflect their execution dependency. However, existing methods assign node priority based on a single node characteristic (*e.g.*, execution time), which cannot leverage the DAG structure to improve performance. In addition, the allocation methods solely consider the ready nodes, which neglects their impact on the upcoming nodes, prolonging the DAG makespan. This paper introduces a novel DAG scheduling algorithm for heterogeneous real-time systems that overcomes the limitations of existing methods. First, a novel node-level priority assignment is proposed that fully exploits the DAG structure to enhance the timing performance. Then, an allocation is constructed that speeds up the execution of high-priority nodes, with upcoming nodes taken into account. The experimental results demonstrate that the proposed method outperforms the existing one up to 14.35% in the DAG makespan.**

***Index Terms*—Direct Acyclic Graphs, Heterogeneous Multicore Systems, List Scheduling, Node Characteristic**

I. Introduction

With the rapid development of high-performance architectures, heterogeneous systems comprising CPUs with various processing speeds are widely deployed to execute computationally intensive tasks with complex execution dependency relationships [1]. A Directed Acyclic Graph (DAG) is a model to encapsulate these dependencies inherent in such tasks [2].

Within a DAG task as shown in Fig. 1, each node signifies a discrete computation unit, capable of independent execution, while the directed edges between nodes denote the dependency relationship. For example, an edge from v_0 to v_1 indicates that the execution of v_1 can commence only after the completion of n_0. Nodes without such dependency can be executed in parallel, such as v_1 to v_5 in the figure. The complexity of DAG scheduling is well-documented, with the problem being categorized as NP-hard [3].

Traditional scheduling approaches for DAG tasks predominantly adopt list heuristic strategies, which consist of two pivotal phases: the prioritization of nodes and the subsequent selection of an appropriate core for each ready node [4], [5]. Despite their widespread application, existing approaches often utilize heuristic methods that assign priority based on simplistic node characteristics, including the length of execution path [4], communication cost [6] or in/out-degrees [5], failing to comprehensively utilize and leverage the multifaceted characteristics and structural differences of these DAG nodes. Furthermore, core allocation strategies in current approaches

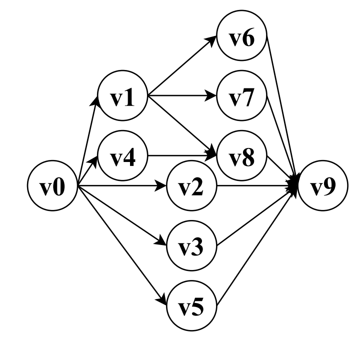

Fig. 1: An example DAG task with ten nodes.

tend to focus narrowly on immediate optimization [7], based on currently available nodes, neglecting the broader implications on upcoming nodes that may hold greater significance.

Contributions. To address these limitations, this paper introduces a novel Structure-aware DAG Scheduling and Allocation (SSA-DAG) for heterogeneous real-time systems, in which the processing speeds of cores can be different. The constructed method identifies the nodes with a high impact on the DAG makespan based on an in-depth analysis of the node structural characteristics. Then a priority assignment and an allocation method of nodes are constructed for SSA-DAG, which prioritize and allocate such nodes on cores with a high processing speed, effectively speeding up their executions. With the SSA-DAG constructed, we provide an end-to-end solution for DAG tasks on heterogeneous systems that fully exploit the structural information to enhance the system timing performance. The contributions of the paper are summarized as follows:

- A *Node Structural Importance* (NSI) metric reflects the importance of a node on DAG makespan using the DAG structure. This provides the foundation for exploiting the DAG structure to improve timing performance.
- A node priority assignment based on NSI, which prioritizes nodes with structural characteristics considered.
- A dynamic allocation strategy with a lookahead mechanism, which considers both the ready and upcoming nodes and reduces DAG makespan by speeding up the execution of high-priority nodes.
- Comprehensive experiments that evaluate the performance of the proposed method over existing ones [4], achieving an improvement of 12.41% (up to 14.35%) in reducing DAG makespan.

Organization. The rest of the paper is organized as follows: Sec. II describes the state-of-the-art in heterogeneous DAG

Corresponding author: Shuai Zhao, zhaosh56@mail.sysu.edu.cn.

scheduling. Sec. III presents the system and task model. Sec. IV introduces the proposed node priority assignment, and Sec. V presents the constructed dynamic allocation strategy. Sec. VI reports and analyzes the experimental results, followed by the conclusion in Sec. VII.

$$rank_d(v_i) = \max_{v_j \in pre(v_i)} \{rank_d(v_j) + \frac{W_j}{\overline{s}}\} \tag{2}$$

where W_i refers to the workload of v_i and $\overline{s}$ denotes the average speed of all cores. Specifically, CPOP [4] takes the sum of the $rank_u$ and $rank_d$ as the priority of each node,