Work-In-Progress: Response Time Bounds for Typed DAG Parallel Tasks on Heterogeneous Multi-cores

Meiling Han, Nan Guan, Jinghao Sun, Qingqiang He, Qingxu Deng and Weichen Liu

1Northeastern University, China
2The Hong Kong Polytechnic University, Hong Kong
3Nanyang Technological University, Singapore

Abstract—Heterogeneous multi-cores utilize the strength of different architectures for executing particular types of workload, and usually offer higher performance and energy efficiency. In this paper, we study the worst-case response time (WCRT) analysis of typed DAG tasks on heterogeneous multi-cores, where the workload of each vertex in the DAG is only allowed to execute on a particular type of cores. The only known WCRT bound for this problem is grossly pessimistic and suffers the non-self-sustainability problem. In this paper, we propose two new WCRT bounds. The first new bound has the same time complexity as the existing bound, but is more precise and solves its non-self-sustainability problem. The second new bound explores more detailed task graph structure information to greatly improve the precision, but is computationally more expensive. We prove that the problem of computing the second bound is strongly NP-hard if the number of types in the system is a variable, and develop an efficient algorithm which has polynomial time complexity if the number of types is a constant. Experiments with randomly generated workload show that our proposed new methods are significantly more precise than the existing bound while having good scalability.

I. INTRODUCTION

Multi-cores are more and more widely used in real-time systems, to meet rapidly increasing requirements in performance and energy efficiency. To fully utilize the computation capacity of multi-cores, software should be properly parallelized. A representation that can model a wide range of parallel software is the DAG (directed acyclic graph) task model, where each vertex represents a piece of sequential workload and each edge represents the precedence relation between two vertices. Real-time scheduling and analysis of DAG parallel task models have raised many new challenges over traditional real-time scheduling theory with sequential tasks, and have become an increasingly hot research topic in recent years.

Many modern multi-cores adopt heterogeneous architectures. Examples include Zynq-7000 [7] and OMAP1/OMAP2 [3] that integrate CPU and DSP on the same chip, and the Tegra processors [6] that integrate GPU and CPU on the same chip. Heterogeneous multi-cores utilize specialized processing capabilities to handle particular computational tasks, which usually offer higher performance and energy efficiency. For example, [8] showed that a heterogeneous-ISA chip multi-processor can outperform the best same-ISA homogeneous architecture by as much as 21% with 23% energy savings and a reduction of 32% in energy delay product.

In this paper, we consider real-time scheduling of typed DAG tasks on heterogeneous multi-cores, where each vertex is explicitly bound to execute on a particular type of cores. Binding code segments of the program to a certain type of cores is common practice in software development on heterogeneous multi-cores and is supported by mainstream parallel programming frameworks and operating systems. For example, in OpenMP [4] one can use the proc_bind clause to specify the mapping of threads to certain processing cores. In OpenCL [5], one can use the clCreateCommandQueue function to create a command queue to certain devices. In CUDA [1], one can use the cudaSetDevice function to set the following executions to the target device.

II. PRELIMINARY

A. Task Model

We consider a typed DAG task $G = (V, E, \gamma, c)$ to be executed on a heterogeneous multi-core platform with different types of cores. $S$ is the set of core types (or types for short), and for each $s \in S$ there are $M_s$ cores of this type ($M_s \geq 1$). $V$ and $E$ are the set of vertices and edges in $G$. Each vertex $v \in V$ represents a piece of code segment to be sequentially executed. Each edge $(u, v) \in E$ represents the precedence relation between vertices $u$ and $v$. The type function $\gamma : V \times S$ defines the type of each vertex, i.e., $\gamma(v) = s$, where $s \in S$, represents vertex $v$ must be executed on cores of type $s$. The weight function $c : V \times \mathbb{R}_0^+$ defines the worst-case execution time (WCET) of each vertex, i.e., $v$ executes for at most $c(v)$ time units (on cores of type $\gamma(v)$).

If there is an edge $(u, v) \in E$, $u$ is a predecessor of $v$, and $v$ is a successor of $u$. If there is a path in $G$ from $u$ to $v$, $u$ is an ancestor of $v$ and $v$ is a descendant of $u$. We use $\text{pre}(u)$, $\text{suc}(u)$, $\text{ans}(u)$ and $\text{des}(u)$ to denote the set of predecessors, successors, ancestors and descendants of $u$, respectively. Without loss of generality, we assume $G$ has a unique source vertex $v_{src}$ (which has no predecessor) and a
unique sink vertex $v_{\text{sink}}$ (which has no successor). We use $\pi \in G$ to denote $\pi$ is a path in $G$. A path $\pi = \{\tau_1, \cdots, \tau_k\}$ is a complete path iff its first vertex $\tau_1$ is the source vertex of $G$ and last vertex $\tau_k$ is the sink vertex. We use $\text{vol}(G)$ to denote the total WCET of $G$ and $\text{vol}_s(G)$ the total WCET of vertices of type $s$:

\[
\text{vol}(G) = \sum_{u \in V} c(u), \quad \text{vol}_s(G) = \sum_{u \in V \land \gamma(u) = s} c(u).
\]

The length of a path $\pi$ is denoted by $\text{len}(\pi)$ and $\text{len}(G)$ represents the length of the longest path in $G$:

\[
\text{len}(\pi) = \sum_{u \in \pi} c(u), \quad \text{len}(G) = \max_{\pi \in G}\{|\text{len}(\pi)|\}.
\]

B. Existing WCRT Bound

To our best knowledge, the only known WCRT upper bound for the considered model was developed in an early work [2]:

**Theorem II.1 (OLD-B).** The WCRT of $G$ is bounded by:

\[
R(G) \leq \left(1 - \frac{1}{\max_{s \in S}\{M_s\}}\right) \times \text{len}(G) + \sum_{s \in S} \frac{\text{vol}_s(G)}{M_s}.
\]  

**OLD-B** is not only pessimistic but also suffers the problem of being non-self-sustainable with respect to processing capacity. More specifically, the value of the WCRT bound in (1) may increase when the number of cores of (some type) increases.

III. Our New Methods

In this section, we first proposed a method which is not only more precise than **OLD-B** (with the same time complexity), but also self-sustainable. But the new method also pessimistic, then we proposed the second method.

A. The First New Method

We start with introducing some useful concepts.

**Definition III.1.** The scaled graph $\hat{G} = (V, E, \hat{c}, \hat{\gamma})$ of $G = (V, E, c, \gamma)$ has the same topology $(V$ and $E)$ and type function $\gamma$ as $G$, but a different weight function $\hat{c}$:

\[
\forall v \in V : \hat{c}(v) = c(v) \times (1 - 1/M_{\gamma(v)}).
\]

**Theorem III.1 (NEW-B-1).** The WCRT of $G$ is bounded by:

\[
R(G) \leq \text{len}(\hat{G}) + \sum_{s \in S} \frac{\text{vol}_s(\hat{G})}{M_s},
\]

where $\hat{G}$ is the scaled graph of $G$.

According to the Equation (2), we can get the following corollary:

**Corollary III.1.** **NEW-B-1** is self-sustainable with respect to each $M_s$.

B. The Second New Method

Our first new WCRT bound **NEW-B-1** is more precise than **OLD-B**, but still very pessimistic. In this section, we present a new method which is more precise than **NEW-B-1**.

For each vertex $v \in V$, $\text{par}(v)$ denotes the set of vertices that have the same type as $v$ but are neither ancestors nor descendants of $v$. Let $\pi = \{\tau_1, \cdots, \tau_k\}$ be a critical path, $\text{ivs}(\pi, s)$ is defined as

\[
\text{ivs}(\pi, s) = \bigcup_{\tau_i \in \pi \land \gamma(\tau_i) = s} \text{par}(\tau_i).
\]

Then the WCRT of the task $G$ is shown in the following theorem:

**Theorem III.2 (NEW-B-2).** The WCRT of $G$ is bounded by

\[
R(G) \leq \max_{\pi \in G}\{\tilde{R}(\pi)\}
\]

where

\[
\tilde{R}(\pi) = \text{len}(\pi) + \sum_{s \in S} \sum_{v \in \text{ivs}(\pi, s)} c(v)/M_s.
\]

Comparing with **NEW-B-1**, **NEW-B-2** is more precise, and according to the Equation (5), so we have

**Corollary III.2.** **NEW-B-2** strictly dominates **NEW-B-1** and **NEW-B-2** is self-sustainable with respect to each $M_s$.

**NEW-B-2** requires to compute the maximum of $\tilde{R}(\pi)$ among all paths in the graph $G$. It is computationally intractable to explicitly enumerate all the paths, the number of which is exponential. Can we develop efficient algorithms of (pseudo-)polynomial complexity to compute $\max_{\pi \in G}\{\tilde{R}(\pi)\}$? Unfortunately, this is impossible unless $P = NP$.

**Theorem III.3.** The problem of computing $\max_{\pi \in G}\{\tilde{R}(\pi)\}$ is strongly NP-hard.

In realistic heterogeneous multi-core platforms, the number of core types is usually not very large. If the number of types is a constant, we proposed an algorithm to compute $\max_{\pi \in G}\{\tilde{R}(\pi)\}$ with complexity $O(|V|S^{1/2})$. That is to say, the problem is actually in $P$ if the number of types is a constant. Instead of explicitly enumerating all the possible paths, our algorithm will use abstractions to represent paths in the graph searching procedure.

**REFERENCES**

[1] NVIDIA CUDA[online]. https://developer.nvidia.com/cuda-zone.
[2] Jeffrey M. Jaffe. Bounds on the scheduling of typed task systems. SIAM Journal on Computing, 9(3):541–551, 1980.
[3] OMA[online]. https://en.wikipedia.org/wiki/OMAP.
[4] OpenMP[online]. Openmp application program interface version 4.5. https://www.openmp.org/.
[5] OpenCL[online]. https://www.khronos.org/opencl/.
[6] Tegra Processors[online]. https://www.nvidia.com/object/tegra.html.
[7] Zynq-7000 SoC[online]. https://www.xilinx.com/products/silicon-devices/soc/zyng-7000.html.
[8] Ashish Venkat and Dean M. Tullsen. Harnessing ISA diversity: Design of a heterogeneous-isa chip multiprocessor. In ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014, pages 121–132. IEEE Computer Society, 2014.