Time-Predictable Parallel Programming Models

Maria A. Serrano
Barcelona Supercomputing Center (BSC) and
Technical University of Catalonia (UPC), Barcelona, Spain
Email: maria.serranogracia@bsc.es

Eduardo Quiñones
Barcelona Supercomputing Center (BSC), Barcelona, Spain
Email: eduardo.quinones@bsc.es

***Abstract*—Embedded Computing (EC) systems are increasingly concerned with providing higher performance in real-time while HPC applications require huge amounts of information to be processed within a bounded amount of time. Addressing this convergence and mixed set of requirements needs suitable programming methodologies to exploit the massively parallel computation capabilities of the available platforms in a predictable way. OpenMP has evolved to deal with the programmability of heterogeneous many-cores, with mature support for fine-grained task parallelism. Unfortunately, while these features are very relevant for EC heterogeneous systems, often modeled as periodic task graphs, both the OpenMP programming interface and the execution model are completely agnostic to any timing requirement that the target applications may have. The goal of our work is to enable the use of the OpenMP parallel programming model in real-time embedded systems, such that many-cores architectures can be adopted in critical real-time embedded systems. To do so, it is required to guarantee the timing behavior of OpenMP applications.**

I. Introduction

High performance computing (HPC) has been for a long time the realm of a specific community within academia and specialized industries. Similarly, embedded computing (EC) has also focused mainly on specific systems with specialized and fixed functionalities for which timing requirements were considered more important than performance requirements. However, with the ever-increasing availability of more powerful processing platforms, alongside affordable and scalable software solutions, both HPC and EC are extending to other sectors and application domains.

As a result, a new type of applications is crossing the boundaries between the HPC and the EC domains. For such applications, the correctness of the result is dependent on both performance and timing requirements, and the failure to meet either of them is critical to the functioning of the system. In this context, it is essential to guarantee the timing predictability of the performed computations.

The use of parallel programming models is fundamental to exploit the performance of current and future many-core architectures, while providing good programmability (and so productivity) of high performance systems. Among the different models, OpenMP [1] has become one of the most used parallel programming models due to its simplicity and scalability in shared memory and heterogeneous systems. The latest specifications of OpenMP incorporate a tasking model that enables very sophisticated types of fine-grained and irregular parallelism, in which the programmer may define explicit tasks and their related data dependencies, as well as an advanced accelerator execution model to deal with data movement and efficient computation in heterogeneous architectures.

Unfortunately, OpenMP tasking and accelerator models were created for a very different purpose than describing real-time applications modeled as task graphs. However, its syntax and execution model retain certain similarities to that formalism that could make it a good candidate to fill the existing gap between: (i) a convenient programming model for heterogeneous many-cores and (ii) state-of-the-art techniques for scheduling with timing guarantees.

Our work focuses on adopting the current OpenMP v4.5 specification to provide timing guarantees so that it can be used in critical real-time embedded systems.

II. OpenMP timing characterization

The first specifications of OpenMP (up to version 2.5) were focused on a thread-centric model to exploit massively data-parallel and loop-intensive types of applications. The latest specifications of OpenMP (versions 3.0, 4.0 and 4.5) have evolved to a task-centric model which enables very sophisticated types of fine-grained and irregular parallelism, including support for heterogeneous computing. This new model, known as *tasking model*, provides a very convenient abstraction of parallelism, being the run-time in charge of scheduling tasks to threads. Despite this model lacks any notion of real-time scheduling semantics, such as deadline, period or WCET, its structure and syntax have certain similarities with the Directed Acyclic Graph (DAG) real-time scheduling model [2] used to analyse the timing behavior of parallel execution in real-time. As an example, Figure 1a shows an OpenMP program composed of five tasks and Figure 1b shows the corresponding OpenMP-DAG, see more details in [3].

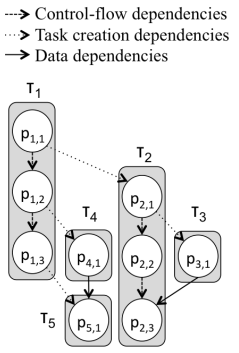
This tight correspondence between the structure and syntax of an OpenMP program and the DAG model makes OpenMP a firm candidate to be adopted in future real-time systems.

III. Response Time Analysis of OpenMP DAGs

In real-time systems the correctness of each computation depends not only on its logical result but also on the time instant at which such result is produced. Failure to respond within a specified time interval is considered as a faulty response. Thus, a key property of a real-time system is time predictability. The sporadic DAG scheduling model is currently under investigation in the real-time community to address time predictability of parallel computation. In the sporadic DAG model, each real-time task (called DAG-task) is represented with a directed acyclic graph (DAG).

```
1 #pragma omp parallel {
2 #pragma omp single {   // τ1
3   p1,1
4   #pragma omp task {   // τ2
5     p2,1
6     #pragma omp task   // τ3
7     { p3,1 }
8     p2,2
9     #pragma omp taskwait
10    p2,3
11  } p1,2
12  #pragma omp task depend (out:a)
13  { p4,1 }             // τ4
14  p1,3
15  #pragma omp task depend (in:a)
16  { p5,1 }             // τ5
17 }}
```

(a) OpenMP source code.



(b) OpenMP DAG.

Fig. 1: Example of an OpenMP program composed of tied tasks (a) and its corresponding OpenMP-DAG (b).

In the following subsections we briefly describe our research on computing the response time analysis of real-time OpenMP applications represented as DAGs. The response time analysis computes the worse-case response time a system may take, in this case, when being scheduled in a parallel processor with a given number of cores. Thus, when compared with the application deadline we can provide OpenMP applications with real-time guarantees.

A. Response Time Analysis of an OpenMP task

Despite being OpenMP a convenient candidate to be adopted in future real-time systems, it incorporates features that limit its practical usability in real-time systems. The most notable example is the distinction between tied and untied tasks. Tied tasks force all parts of a task to be executed on the same thread that started the execution, whereas a suspended untied task is allowed to resume execution on a different thread. The execution model of tied tasks has serious implications on the response time analysis of OpenMP applications, making difficult to adopt it in real-time environments.

In [6] we analyze, from a timing perspective, the two tasking existing models in OpenMP: tied and untied. The considerations drawn in this work suggest that using tied tasks inside time-critical applications is not recommendable because of the inherent pessimism that underlies the timing analysis of such tasks and the conceptual difficulties behind the construction of an accurate schedulability test. We also show that a simple schedulability analysis of OpenMP programs is possible whenever untied tasks are involved. This suggests that the use of untied tasks would be preferable for parallel applications in a real-time context, since it would permit to exploit a parallel execution model in a predictable way.

B. Response Time Analysis of a set of OpenMP tasks

When consideirng several OpenMP applications with different priorities, preemption is a key concept since it allows the operating system allocate the core to tasks requiring urgent service. The limited preemption (LP) approach has been proposed in the literature to reduce the run-time overhead due to preemptions and still preserve the schedulability of the system. According to this approach, a task implicitly executes in non-preemptive mode and preemption is allowed only at predefined locations inside the code, called preemption points. In this way, a task is divided into a number of non-preemptive chunks (also called sub-tasks); if a higher priority task arrives between two preemption points of the running task, preemption is postponed until the next preemption point.

Interestingly, the LP approach with fixed preemption points resembles the OpenMP task execution model [3]. Therefore, in [4] and [5] we evaluate the LP strategy for DAG-based task-sets; we show the necessary conditions under which DAG tasks may experience lower and higher priority task interference for different LP sub-approaches.

IV. Future Work

Our future work focuses on the schedulability analysis of the OpenMP accelerator model for heterogeneous architectures. We will investigate novel time predictable scheduling solutions for the OpenMP accelerator model in current many-core heterogeneous architectures. In these architectures, the main problems lie in: (1) the proper scheduling of data transfers and computation on acceleration devices, and (2) the characterization of the multiple interferences and inter-dependencies that may arise among the simultaneous access of various accelerator devices in a many-core system.

V. Acknowledgment

This work was funded by the EU project P-SOCRATES (FP7-ICT-2013-10) and the Spanish Ministry of Science and Innovation under contract TIN2015-65316-P.

References

[1] OpenMP Application Program Interface, Version 4.5. November 2015.
[2] S. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, L. Stougie, and A. Wiese. A generalized parallel task model for recurrent real-time processes. In *RTSS*, 2012.
[3] R. Vargas, et. al. OpenMP and Timing Predictability: A Possible Union? In *18th Design, Automation and Test in Europe Conference (DATE)*, 2015.
[4] M. A. Serrano, A. Melani, M. Bertogna, and E. Quinones. Response-time analysis of DAG tasks under fixed priority scheduling with limited preemptions. In *DATE*, March 2016.
[5] M. A. Serrano, A. Melani, S. Kehr, M. Bertogna, and E. Quinones. An analysis of lazy and eager limited preemption approaches under dag-based global fixed priority scheduling. In *ISORC*, May 2017 (to apperar).
[6] M. A. Serrano, A. Melani, R. Vargas, A. Marongiu, M. Bertogna, and E. Quinones. Timing characterization of OpenMP4 tasking model. In *CASES*, 2015.

Maria A. Serrano is a PhD. student in the Department of Computer Science at the Barcelona Supercomputing Center (BSC), Spain, since 2014. She studied a B.S / M.S in Computer Science Engineering (2006-2013) and a M.S. in Systems Engineering and Computer Science (20013-2014) at the University of Zaragoza, Spain. Her research interests focus on Real-Time systems, specifically on many-cores embedded processors and on providing parallel programming models with timing guarantees.