An ILP-based Real-Time Scheduler for Distributed and Heterogeneous Computing Environments

Eudald Sabaté[1], Maria A. Serrano[2], Eduardo Quiñones[3]

Barcelona Supercomputing Center (BSC), Spain

[1]eduald.sabate@bsc.es, [2]maria.serranogracia@bsc.es, [3]eduardo.quinones@bsc.es

***Keywords*—real-time, schedulability analysis, distributed computing environment, ILP**

ABSTRACT

The digitalization process is making cities to rapidly increase the amount of data to be processed upon which data analytics can extract valuable knowledge. However, this phenomenon is facing many important challenges. On one side, the advent of connected and autonomous vehicles challenges data analytics methods due to the need of accomplishing real-time requirements. On the other side, the dispersion nature of data sources makes current big data analytics methods, commonly designed to execute in centralized and computationally intensive (cloud-based) environments, not suitable for smart cities. The use of distributed computing environments composed of advanced parallel embedded processor architectures at the edge, e.g., NVIDIA Jetson, Kalray MPPA, can help alleviating the pressure on centralized cloud-based solutions, while providing the real-time guarantees needed to implement advanced mobility functionalities on cars and cities.

To do so, this work presents a novel scheduler (based on ILP formulation) to optimally distribute the computation across the compute continuum composed of multiple edge devices, while providing real-time guarantees. Our scheduler, implemented in the COMPSs distributed programming model developed at BSC, statically assigns tasks to those edge devices so that the overall response time of the workflow is minimized. It takes into account an execution time upper bound of the computation and communication existing in the workflow.

A. Introduction

Critical real-time systems are commonly modelled as a set of concurrent and periodic real-time tasks that implement the system functionalities [1]. Each real-time task is characterized with a *period*, a *deadline* and *direct acyclic graph (DAG)* composed of a set of nodes, implementing sub-functionalities, and a set of edges connecting the nodes, representing the precedence constraints among them. In this context, real-time scheduling techniques are key to: (1) efficiently execute the real-time tasks into computing resources, and (2) guarantee that the deadlines of the real-time tasks are met by means of schedulability analysis [2]. Interestingly, workflows in COMPSs [3], a task-based programming framework for distributed computing environments, are represented as a DAG, in which nodes correspond to COMPSs tasks and edges to the data dependencies existing among them. Unfortunately, current schedulability analysis are only applicable to systems executed on the same computing node and therefore, cannot be applied to COMPSs.

This work tackles the problem of efficiently distributing a COMPSs workflow across the compute continuum, while guaranteeing its timing constraints. To do so, we develop a schedulability analysis based on *Integer Lineal Programming* (ILP) formulation with a twofold objective: (1) to minimize the execution time of the COMPSs workflow by statically assigning COMPSs tasks to the computing resources that form the compute continuum; and (2) to provide an upper bound response time of the overall workflow by considering the timing characterization of each COMPSs task and the data transfer among them. This static allocation is then implemented by a dedicated COMPSs scheduler.

B. COMPSs

COMPSs offers a task-based and portable programming framework that facilitates the distribution and parallelization of sequential source code (written in Java, C/C++ or Python) in a distributed and heterogeneous computing environment, such as those existing in smart cities. In COMPSs, the programmer is responsible of identifying *COMPSs tasks* and the data dependencies existing among them, by annotating the sequential source code. The run-time scheduler is then in charge of distributing the COMPSs tasks among the available computing resources across the compute continuum, from on edge to cloud.

The COMPSs runtime incorporates several task schedulers implementing different allocation policies: (1) FIFO selects the first ready task and the first available computing resource; (2) FIFO + Data locality selects the first ready task and the computing resource that better exploits the data locality; (3) LIFO selects the last ready task and the first available computing resource; and (4) FIFO + Load Balancing selects the first ready task and the computing resource that better balances the overall load of the system. These strategies are intended to exploit the performance in HPC environments, but do not guarantee the real-time constraints of the system.

We tackle the problem of efficiently distributing a COMPSs workflow and guaranteeing the timing constraints, by developing a dedicated COMPSs scheduling strategy based on a static allocation of COMPSs tasks to computing resources derived with an ILP formulation.

C. ILP Scheduling Strategy

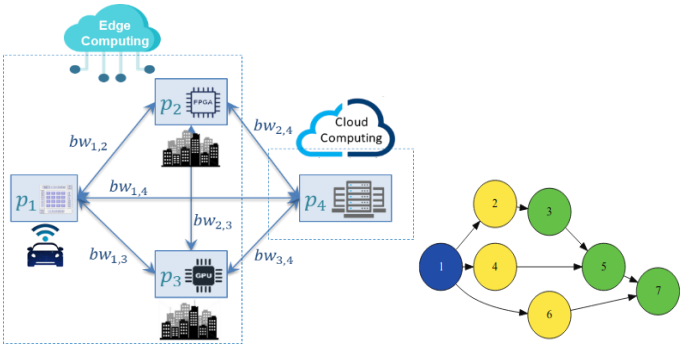
With the objective of incorporating real-time requirements while efficiently distributing the workflow, we propose a new scheduling strategy that allocates COMPSs tasks to computing resources, based on a predefined static allocation given by an ILP formulation. ILP is a well-known technique already used in real-time systems to compute the minimum makespan of real-time tasks represented as a DAG [4][5].

Concretely, our ILP scheduling strategy relies upon a system model that considers the following information: (1) a representation of the COMPSs workflow by means of a DAG, that includes the execution time upper bound of each COMPSs task and the characterization of the size of the data dependencies between tasks; and (2) a description of the compute continuum, including the computing resources and the communication network. Next, we briefly describe the system model considered.

C.1. Compute continuum model

The compute continuum model represents the available (and heterogeneous) computing resources interconnected by different network links. It is represented as a directed graph (digraph) where each node represents a computing resource characterized

by its type and computation capabilities, (e.g., GPU, CPU, Instruction Set Architecture (ISA) supported), and each edge represents the communication link between nodes, characterized by the transport bandwidth. Figure 1(a) shows an example of a compute continuum model representing an edge-cloud system, consisting of four computing resources: a computing resource available on a car, two computing nodes located within the street and a data center representing the city cloud services



(a) Compute continuum model *(b) Task model*

Figure 1. System model.

C.2. Task Model

A COMPSs workflow is represented by a DAG, where each node represents a COMPSs task, and each edge represents a data dependency existing between COMPSs tasks. Edges are characterized by the total size of the data involved in the dependency. If two dependent tasks execute in different computing resources, then this data must be transferred through the computer network. Edges also represent the execution order constraints, i.e., if there exists an edge between two nodes, then predecessor node must complete before successor node can begin its execution. Since each task may have different implementations, for each of the potential computing resources in which it can execute, each node is characterized by set of values corresponding to an upper bound of the execution time of the task in a given computing resource. Figure 1(b) shows an example of a task workflow, represented as a DAG, composed of seven nodes and eight edges.

D. Evaluation

This section provides a preliminary evaluation of the ILP scheduling strategy proposed.

D.1. Experimental Setup

We consider a Cholesky factorization [6] application (implemented in Python) and parallelized with COMPSs. The Cholesky factorization is commonly used for efficient linear equation solvers, Monte Carlo simulations, or to accelerate Kalman filters implemented, for example, in vehicle navigation systems to detect objects positions and compute trajectories. The COMPSs workflow, that processes a matrix of 2048 × 2048, generates a DAG composed of 30 COMPSs tasks. The execution time upper bound of each COMPSs tasks have been computing by selecting the maximum observed execution time of 50 executions.

Moreover, we consider a compute continuum model with two computing resources: (1) a four-core Intel(R) i7-7600U processor @ 2.80GHz and (2) a four core ARMv8 Processor rev 3 included in a NVIDIA Jetson TX2. The communication link between them is an IEEE 802.11g at 54 Mbps of bandwidth.

D.2. Results

Table 1 compares the execution time (in seconds) of the Cholesky factorization executed on our compute continuum model, considering our ILP-based scheduling strategy and the four different baseline COMPSs schedulers, i.e., FIFO, FIFO + Data locality, LIFO and FIFO + Load Balancing strategies.

Table 1. Execution time (in sec) of the Cholesky factorization.

ILP-based Scheduler	COMPSs baseline scheduler			
	FIFO	FIFO + Data Locality	LIFO	FIFO + Load Balancing
8.16	16.36	12.53	14.68	11.92

As observed, and for this particular example, our ILP scheduler strategy clearly outperforms the COMPSs baseline schedulers, reducing the execution of time of the Cholesky factorization by two when comparing to the COMPSs FIFO schedulers presented in Section B.

E. Conclusion and Future Enhancement

This work presents a static allocation strategy based on an ILP-formulation that minimizes the execution time of a COMPSs workflow, while providing an upper bound of its response time. The next steps are to evaluate our scheduling strategy with a more complex (and realistic) computing continuum model of a smart city, and considering a workflow that implements an advanced mobility functionality with data analytics methods.

F. Acknowledgements

The research leading to these results has received funding from the EU Horizon 2020 Programme under the CLASS Project (www.class-project.eu), grant agreement No 780622. An extended version of this work is planned to be submitted to ESWEEK 2019 (https://www.esweek.org/).

References

[1] S. Baruah, M. Bertogna, and G. Buttazzo. *Multiprocessor Scheduling for Real-Time Systems.* Springer, 2015.

[2] G. C. Buttazzo. *Hard real-time computing systems: predictable scheduling algorithms and applications*, volume 24. Springer Science & Business Media, 2011.

[3] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia. *ServiceSs: an interoperable programming framework for the Cloud*, Journal of Grid Computing, March 2014.

[4] M. A. Serrano, A. Melani, M. Bertogna, E. Quiñones. *Response-time analysis of DAG tasks under fixed priority scheduling with limited preemptions.* In DATE, 2016.

[5] A. Melani, M. A. Serrano, M. Bertogna, I. Cerutti, E. Quiñones, G. Buttazzo. *A static scheduling approach to enable safety-critical OpenMP applications.* In ASP-DAC 2017.

[6] N. Bašcelija. *Sequential and parallel algorithms for cholesky factorization of sparse matrices.* Mathematical Applications in Science and Mechanics, 2013.

Author biography

Eudald Sabaté was born in Barcelona, Spain, in 1995. He received the Bachelor degree in computer science from the Polytechnic University of Catalonia (UPC) in 2018, and is currently enrolled in the MSc in Innovation and Research in Informatics at UPC. Since April 2018, he has been working as a research student at BSC in the CAOS (Computer Architecture and Operating Systems) group.