Parallel Computing 37 (2011) 410–427

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Mapping workflow applications with types on heterogeneous specialized platforms

Anne Benoit [b], Alexandru Dobrila [a,*], Jean-Marc Nicod [a], Laurent Philippe [a]

[a] Laboratoire d'Informatique de Franche-Comté (LIFC), Université de Franche-Comté, France
[b] Laboratoire de l'Informatique du Parallélisme (LIP), ENS Lyon, Université de Lyon, CNRS, INRIA, UCBL, France

ARTICLE INFO

Article history:
Available online 25 December 2010

Keywords:
Coarse-grain workflow applications
Types
Failures
Throughput
Complexity results
Linear programming
Heuristics

ABSTRACT

In this paper, we study the problem of optimizing the throughput of coarse-grain workflow applications, for which each task of the workflow is of a given type, and subject to failures. The goal is to map such an application onto a heterogeneous specialized platform, which consists of a set of processors that can be *specialized* to process one type of tasks. The objective function is to maximize the throughput of the workflow, i.e., the rate at which the data sets can enter the system. If there is exactly one task per processor in the mapping, then we prove that the optimal solution can be computed in polynomial time. However, the problem becomes NP-hard if several tasks can be assigned to the same processor. Several polynomial time heuristics are presented for the most realistic *specialized* setting, in which tasks of the same type can be mapped onto the same processor, but a processor cannot process two tasks of different types. Also, we give an integer linear program formulation of this problem, which allows us to find the optimal solution (in exponential time) for small problem instances. Experimental results show that the best heuristics obtain a good throughput, much better than the throughput obtained with a random mapping. Moreover, we obtain a throughput close to the optimal solution in the particular cases on which the optimal throughput can be computed (small problem instances or particular mappings).

1. Introduction

Mapping applications onto parallel platforms is a difficult challenge. Several scheduling and load-balancing techniques have been developed for homogeneous architectures (see [1] for a survey), but the advent of heterogeneous platforms, such as grid platforms or even clouds, have rendered the mapping problem even more difficult.

In this paper we deal with scheduling and mapping strategies for coarse-grain workflow applications, for which each task of the workflow is of a given type, and subject to failures. In such applications, a series of data sets enters the workflow and progresses from task to task until the final result is computed. The type of a task determines its computation requirements, and the failure rate of a task determines an amount of extra computations to be done when failures occur. After an initialization delay, a new data set is completed every period and it exits the workflow. The *period* is therefore defined as the longest cycle-time to operate a task, and it is the inverse of the *throughput* that can be achieved. We target *coarse-grain* applications and platforms in which the cost of communications is negligible in comparison to the cost of computations.

The scheduling problem is to map such an application onto a target platform, with the objective to minimize the period, or equivalently, to maximize the throughput. Each task of the workflow must be mapped onto a processor, and the target plat-

* Corresponding author.
E-mail addresses: Anne.Benoit@ens-lyon.fr (A. Benoit), adobrila@lifc.univ-fcomte.fr (A. Dobrila), jmnicod@lifc.univ-fcomte.fr (J.-M. Nicod), lphilippe@lifc.univ-fcomte.fr (L. Philippe).

0167-8191/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.parco.2010.12.001

form is heterogeneous. Moreover, each processor can be specialized to process one type of tasks. We consider several mapping rules: we require the mapping to be *one-to-one* (a processor is assigned at most one task), or *specialized* (a processor is assigned a set of tasks of same type), or fully *general*. On the theoretical side, our main contributions are (i) a polynomial time algorithm to find the one-to-one mapping which minimizes the period, (ii) the proof that the problem becomes NP-hard for specialized and general mappings, and (iii) an integer linear programming formulation of the specialized mapping problem.

In practice, the specialized mapping problem has many applications, not only in the field of distributed computing and workflows, but also for production systems. In the distributed computing context, an example of workflow application would be processing images in batches on a SaaS (Software as a Service) platform. For such an application, the failure rate may be impacted by the complexity of the service [2]. Another field of application is the important problem of sequencing pipelined queries on a set of web services, as presented in [3]. Each service does not provide the same amount of data on the output as on the input, because of the *selectivity* of the services; this problem is quite similar to our problem of mapping unreliable tasks, where service selectivities correspond to task failure rates. We further consider typed services.

On the production side, a relevant use case is a micro-factory [4] composed of several cells that provide functions as assembly or machining. Due to the piece, actuator and cell sizes, it is impossible for human operators to directly interfere with the physical system. So it needs a highly automated command. The complexity of this command makes it mandatory to develop a distributed system to support this control. But, at this scale, the physical constraints are not totally controlled so there is a need to take faults into account in the automated command. Typically, the tasks to be performed in the micro-factory are typed.

The paper is organized as follows. Section 2 gives an overview of related work. Then, Section 3 details the application, the platform and execution models, and the optimization problems. The complexity results are given in Section 4. Section 5 provides an integer linear program to solve the specialized mapping problem, which turns out to be NP-hard, and we propose several polynomial time heuristic solutions to this problem. The heuristics are extensively studied through a set of experiments in Section 6, and finally we conclude and give future work directions in Section 7.

2. Related work

Workflow scheduling [5,6] is a vast domain largely studied in the literature. There are two main contexts in which workflow applications can be found. First, in grid computing, workflows are described as collections of tasks that are processed in a particular order to optimize an objective function. This is a popular programming model for streaming applications like DSP, medical or video/audio encoding/decoding applications [7,8]. In the same way, query optimization over web services is a challenge [3,9]. Moreover, some data-driven (or macro-dataflow) workflow applications are based on the coarse-grain pipelined parallelism processing paradigm [10,11]. Second, in factories [4], workflow supports supply chain, logistics or business process.

In both contexts, we aim at scheduling workflow applications onto a target platform, in order to optimize some criteria. Most of the time, performance-related criteria are targeted, such as the completion time [12], the latency [13] or the throughput [14]. Two or more criteria, even if antagonists, can be combined, as for instance the latency/throughput combination [15]. Some other criteria are nowadays being tackled, such as the energy consumption or the reliability, which must be optimized while satisfying some requirements on the performance criteria, see for instance [16–18]. In this paper, we focus only on the classical problem of the throughput maximization.

Several structures of task graphs have been studied. In [19], the workflow is a classical pipeline, i.e., the graph is a linear chain, while in [20], it is a direct acyclic graph (DAG) with a single-entry and a single-exit. Precedence constraints are applied between the tasks as well as communication costs which represent the amount of data to be transferred from a task to the next one. In this paper, we focus mainly on linear chain graphs, but the results can be easily extended to any DAG without forks (in-tree). Moreover, when considering low communication overhead in parallel computing, communication-free techniques can be used [21]. In our context of coarse-grain workflow applications, communications can be overlapped by computations.

With the advent of large distributed systems, processors are more and more prone to failures, and one must find a way to deal with it. The main issue for fault tolerant systems [22,23] is to overcome the failure of a node, a machine or a processor. To deal with those faulty machines, the most common method used is to replicate [24,2] the data, i.e., do redundant computations on several machines in order to increase the probability to get a correct result. Once all the replicated jobs are done, a vote algorithm [25] is often used to decide which result is the right one.

In real time systems, another model called window-constrained model [26] can be used. In this model one considers that, for y messages, only x ($x \leqslant y$) of them will reach their destination. The y value is called the window. The looses are not considered as a failure but as a guarantee: for a given network, a window-constrained scheduling [27,28] can guarantee that no more than x messages will be lost for every y sent messages. In this paper, the window-constrained based failure model is adapted to a distributed system. We deal with the loss of intermediate results. This context is close to the web service query optimization problem using a selectivity rate less than one [29]. Finally, our optimization problem is quite similar to the problem of scheduling unreliable jobs on parallel machines [30].

3. Framework

We outline in this section the characteristics of the applicative framework and target platform. Then we describe the execution model. Finally we define the optimization problems that are tackled in this paper.