Reconfigurable Gang Scheduling Algorithm

Luís Fabrício Wanderley Góes and Carlos Augusto Paiva da Silva Martins

Pontifical Catholic University of Minas Gerais, Post-Graduation Program in Electrical Engineering, CEP 30535-610
Av. Dom José Gaspar, 500 Belo Horizonte, Minas Gerais, Brasil
lfwg@uol.com.br, capsm@pucminas.br

Abstract. Using a single traditional gang scheduling algorithm cannot provide the best performance for all workloads and parallel architectures. A solution for this problem is an algorithm that is capable of dynamically changing its form (configuration) into a more appropriate one, according to environment variations and user requirements. In this paper, we propose, implement and analyze the performance of a Reconfigurable Gang Scheduling Algorithm (RGSA) using simulation. A RGSA uses combinations of independent features that are often implemented in GSAs such as: packing and re-packing schemes (alternative scheduling etc.), multiprogramming levels etc. Ideally, the algorithm may assume infinite configurations and it reconfigures itself according to entry parameters such as: performance metrics (mean utilization, mean response time of jobs etc.) and workload characteristics (mean execution time of jobs, mean parallelism degree of jobs etc.). Also ideally, a reconfiguration causes the algorithm to output the best configuration for a particular situation considering the system's state at a given moment. The main contributions of this paper are: the definition, proposal, implementation and performance analysis of RGSA.

1 Introduction

Nowadays, the service quality requirements of users and institutions increased. Thus, computer systems that provide many services (particularly, parallel machines) need to be highly utilized and provide a short response time for users jobs. Parallel job schedulers should match both requirements and workload (jobs) with resource availability (architecture, processors etc.) in order to maximize the system's performance. The main problem is that workload, requirements and resources change continuously. In order to solve this problem, many works have been developed to make job scheduling algorithms more flexible and adaptable [1], [12], [13], [14], [18], [20]. Up to now, a poorly explored solution is the use of reconfigurable computing concepts [3], [4], [13], [14], [16] in parallel job scheduling algorithms (like gang scheduling).

Reconfigurable computing emerged as a paradigm to fill in the gap between hardware and software, reaching better performance than software and more flexibility than hardware [3], [4], [16]. The reconfigurable devices including FPGAs (Field Programmable Gate Arrays) contain an array of computing elements or constructive blocks, whose functionalities are determined through the programming of configuration bits. Thus, an FPGA can implement different behaviors not established at design

D. Feitelson, L. Rudolph, and U. Schwiegelshohn (Eds.): JSSPP 2004, LNCS 3277, pp. 81-101, 2005.

time. Because of this, reconfigurable devices (hardware) are improving the solutions for problems from different areas [3], [4], [16].

Our basic idea in this paper is to use reconfigurable computing concepts in a parallel job scheduling algorithm (gang scheduling) to maximize system's performance. According to a deep bibliographic revision [3], [4], [13], [14], [16], we found works that apply reconfigurable computing in software, but we did not find a previous work that used it on algorithms. In [13], we used a first approach to build a reconfigurable algorithm of a static parallel job scheduling algorithm. We improved this first approach to reach our present stage.

Ideally, the algorithm may assume infinite configurations and it reconfigures itself according to entry parameters such as: performance metrics (utilization, mean response time of jobs etc.) and workload characteristics (mean execution time of jobs, mean parallelism degree of jobs etc.). Also ideally, a reconfiguration causes the algorithm to output the best configuration for a particular situation considering the system's state at a given moment.

Gang scheduling algorithms have been intensely studied in the last decade. They demonstrated many advantages over other parallel job scheduling algorithms, for instance, they: provide interactive response time for short jobs, through preemption; prevent long jobs from monopolizing processors; maximize the system's utilization etc [1], [2], [5], [6], [11], [12], [14], [18], [19], [20]. In our specific case it presents some interesting characteristics. It is composed of independent and well defined parts (packing and re-packing schemes, multiprogramming level, etc.) and each one has infinite possible solutions (implementations).

The **main objectives** of this paper are: to define, propose, develop and implement the RGSA; to analyze the performance of RGSA using simulation. The **main goal** is the implementation of RGSA in our simulation tool.

In this paper, we introduce the reconfigurable gang scheduling algorithm (RGSA) and relate it to other works in sections 2 and 3. In section 4, we present our experimental method: workload, metrics, configurations and parallel architecture used in our simulations. Section 5 presents the experimental results and the performance analysis comparing RGSA and other traditional gang scheduling algorithms. Finally, in section 6 we highlight our conclusions and future works.

2 Reconfigurable Gang Scheduling Algorithm (RGSA)

Extending the reconfigurable hardware definition, we define a reconfigurable algorithm as an algorithm that is composed of constructive blocks allowing its behavior to be modified through the form of its configuration.

A reconfigurable algorithm is composed of three layers: Configuration Control Layer (CCL), Reconfigurable Layer (RL) and Basic Layer (BL), as shown in Fig.1. The BL consists of a frame set and data structures. A data structure may be a list, a queue, an array or some structure that stores data. For example, in Fig.2 a wait queue (data structure) stores jobs (data).

A frame represents a part or phase of an algorithm. For example, in a gang scheduling algorithm, a frame may represent a packing scheme that fits a job inside the