Chapter 5
A Resource-Aware Dynamic Load-Balancing Parallelization Algorithm in a Farmer-Worker Environment

M. Leeman
Cisco, Belgium

ABSTRACT

This paper describes an algorithm for dynamically assigning tasks to processing entities in a world where each task has a set of resource or service requirements and each processing entity a set of resources or service capabilities. A task needs to be assigned to a node that offers all required services and the set of tasks is finished within a minimal execution time frame. Dependability and adaptability are inherent to the algorithm so that it accounts for the varying execution time of each task or the failure of a processing node. The algorithm is based on a dependable technique for farmer-worker parallel programs and is enhanced for modeling the time constraints in combination with the required configuration set in a multidimensional resources model. This paper describes how the algorithm is used for dynamically load balancing and parallelizing the nightly tests of a digital television content-processing embedded device.

INTRODUCTION

In environments where processing power is limited or where continuously increasing processing power is needed, parallelization is the main method for speeding up tasks. These tasks can be divided into equally sized subtasks. Through the application of multiple instruction stream, multiple data stream (MIMD) techniques, the subtasks are assigned in parallel to available processing entities (Hennessy & Patterson, 2003). When processing a subtask fails or when processing is delayed due to decreased performance, the subtask can be reassigned. Processing nodes can be added when they become available or can be removed upon failure.

DOI: 10.4018/978-1-4666-2056-8.ch005

In some applications, however, neither subtasks nor processing entities are mirrors of each other. Some subtasks can be executed only on one or more processing nodes, as only these nodes offer the services or configuration the subtask requires. In this respect, subtasks are recognized as having certain requirements, and each processing entity as offering a set of capabilities. Subtasks need to be assigned to nodes meeting the subtask's requirements. If no node offers the required capabilities, the subtask cannot be executed.

Furthermore, the complete set of subtasks often needs to be completed as soon as possible. Some subtasks are more time-consuming than other ones, and even the processing time of one subtask can differ between two executions. Hence the varying execution length requires a dynamic assignment process. Assigning these subtasks is multidimensional: one dimension for each requirement, and another for the time constraints.

The next section describes the basic algorithm for parallelizing tasks with a farmer-worker dependable model. Afterward, a new algorithm is described that assigns tasks dynamically, such that the requirements of each subtask are met and the complete set of subtasks is finished with minimal delay, even if some processing nodes fail. A test case is covered in which this algorithm assigns nightly tests to devices that process digital television streams in the telecom domain. Finally, some future work is described.

THE FARMER-WORKER ALGORITHM AND ITS DEPENDABLE EXTENSION

The dependable multiresource dynamic algorithm is based on an elaboration of the traditional farmer-worker algorithm. In the most commonly known basic farmer-worker model, a "farmer" processing entity grabs the input data or tasks, divides the tasks into subtasks, and feeds the subtasks in parallel to processing entities called "workers." The farmer collects the results and glues them together to one resulting processed output.

The farmer needs to go through quite some sequential processing between two runs. The input data needs to be grabbed, divided, and dispatched to the workers, and afterward the worker's processing output needs to be collected, merged, and finalized to one result. In video processing, where video is received from a camera and images extracted and divided for further processing, real-time behavior is important. Furthermore the failure of one worker endangers the processing of the complete task.

Therefore a dependable extension of the farmer-worker model has been proposed (De Florio, Deconinck, & Lauwereins, 1997) and, later, a corresponding framework library, RAFTnet (Leeman, Leeman, De Florio, & Deconinck, 2003). This extension describes a "dispatcher" and a "collector." The workers subscribe themselves for processing with the dispatcher. The farmer grabs the input data and sends the list of subtasks to the dispatcher, which assigns them to idle workers. While these workers are processing subtasks and the dispatcher is assigning them, the farmer can grab the input data for the next run.

Once a worker has processed a subtask, it notifies the dispatcher of its idle state so that a new subtask can be assigned to it. The output is sent to the collector. This entity collects the output from each worker and notifies the dispatcher that this subtask has been processed. From the moment a subtask is finished, the dispatcher also notifies the farmer, which sends as a response the corresponding subtask from the next run to the dispatcher.

Other algorithms have been proposed by similarly introducing an additional farmer or collector process in a flat (Chan & Abramson, 2001) or a hierarchical model (Aida, Futakata, & Tomotaka, 2006; Berthold, Dieterle, Loogen, & Priebe, 2008). Further improvement can be achieved by exploiting algorithm characteristics like dynamic grain size, denoting the task can be