Future Generation Computer Systems 33 (2014) 1–10

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Maximizing stochastic robustness of static resource allocations in a periodic sensor driven cluster

Jay Smith [a,b,*], Anthony A. Maciejewski [b], Howard Jay Siegel [b,c]

[a] Lagrange Systems, Boulder, CO 80302, USA

[b] Colorado State University, Department of Electrical and Computer Engineering, Fort Collins, CO 80523–1373, USA

[c] Colorado State University, Department of Computer Science, Fort Collins, CO 80523–1373, USA

HIGHLIGHTS

- Novel methodology for creating robust resource allocations given a tight deadline.
- Resource allocation heuristics that maximize robustness.
- Local search operator for a Genetic Algorithm including search space analysis.
- Local search based path relinking crossover operator for a Genetic Algorithm.

ARTICLE INFO

Article history:
Received 15 December 2012
Received in revised form
27 September 2013
Accepted 4 October 2013
Available online 24 October 2013

Keywords:
Robustness
Heterogeneous computing
Resource management
Static resource allocation
Distributed computing

ABSTRACT

This research investigates the problem of robust static resource allocation for distributed computing systems operating under imposed Quality of Service (QoS) constraints. Often, such systems are expected to function in an environment where uncertainty in system parameters is common. In such an environment, the amount of processing required to complete a task may fluctuate substantially. Determining a resource allocation that accounts for this uncertainty—in a way that can provide a probability that a given level of QoS is achieved—is an important area of research. We have designed novel techniques for maximizing the probability that a given level of QoS is achieved. These techniques feature a unique application of both path relinking and local search within a Genetic Algorithm. In addition, we define a new methodology for finding resource allocations that are guaranteed to have a non-zero probability of addressing the timing constraints of the system. We demonstrate the use of this methodology within two unique steady-state genetic algorithms designed to maximize the robustness of resource allocations. The performance results for our techniques are presented for a simulated environment that models a heterogeneous cluster-based radar data processing center.

1. Introduction

In parallel and distributed computing, multiple compute nodes are collectively utilized to simultaneously process a set of applications to improve the performance over that of a single processor [1,2]. Often, such computing systems are constructed from a heterogeneous mix of machines that may differ in their capabilities, e.g., available memory, number of floating point units, clock speed, and operating system. This paper investigates robust resource allocation for a large class of heterogeneous computing (HC) systems that operate on *periodically updated* sensor data sets. Sensors (e.g., radar systems, sonar) in this environment produce new data sets at a fixed period Λ (see Fig. 1). Often, the period at which the sensor produces new data sets is fixed by the sensor and cannot be extended to account for long-running applications. Because these sensors typically monitor the physical world, the characteristics of the data sets they provide may vary in a manner that impacts the execution times of the applications that must process them. Suppose that each input data set must be processed by a collection of N independent applications that can be executed concurrently on the available set of M heterogeneous compute nodes. The goal of resource allocation heuristics in this environment is to allocate the N tasks to the M compute nodes such that all of the applications finish each data set in less than Λ time units, i.e., the makespan of a resource allocation must be less than or equal to Λ.

In this environment, the allocation of compute nodes to applications can be considered static, i.e., all of the applications that are to be executed are known in advance and are immediately available for execution upon the arrival of a new data set. Furthermore,

* Corresponding author at: Colorado State University, Department of Electrical and Computer Engineering, Fort Collins, CO 80523–1373, USA. Tel.: +1 7208395378.
E-mail addresses: jay@lagrangesystems.com (J. Smith), aam@colostate.edu (A.A. Maciejewski), hj@colostate.edu (H.J. Siegel).

0167-739X/$ – see front matter
http://dx.doi.org/10.1016/j.future.2013.10.001

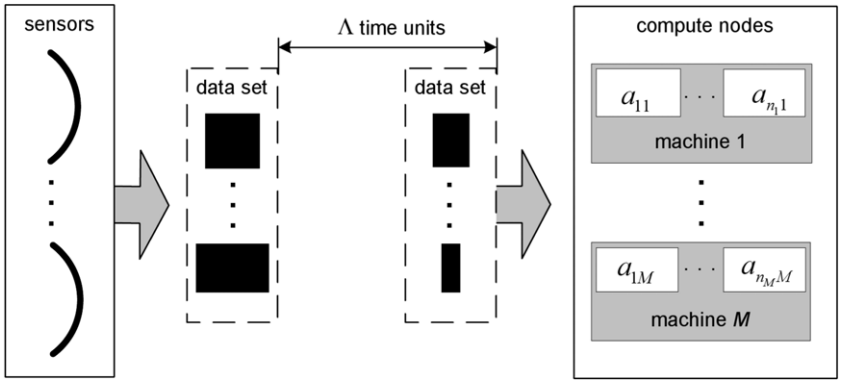

Fig. 1. Major functional units and data flow for a class of systems that must periodically process data sets from a collection of sensors, where a_{ij} corresponds to the ith application assigned to machine j.

the resource allocation remains fixed over a sequence of data sets. We assume that historical application execution time data for each application on each compute node in the HC system is available.

Because this is an HC system, the execution times for each of the N independent applications differ across the M compute nodes. Resource allocation in such computing environments has been shown in general to be an NP-hard problem (e.g., [3,4]). Thus, designing techniques for resource management in such environments is an active area of research (e.g., [5,6,1,7–10]).

We consider an environment where the required time period Λ is fixed by the sensor platform. Because a new data set arrives every Λ time units, the completion time of all applications must be less than or equal to Λ. Thus, ensuring that the system is ready to begin processing the next data set upon arrival without delay. However, unpredictable differences in the characteristics of the input data sets may result in a significant change in the execution times of the applications that must process the data. These differences may prevent some applications from completing because it would cause the makespan of the resource allocation to exceed Λ. *Robust* design for such systems involves determining a resource allocation that can account for uncertainty in application execution times in a way that maximizes the probability that all applications will complete within Λ time units. This probability is used to quantify robustness in this environment.

We first demonstrate several techniques for maximizing the robustness of resource allocations using both a greedy heuristic and more complex approaches, such as Genetic Algorithms. We show that the complex heuristics perform well given a "loose" Λ completion time constraint. However, given a tight Λ constraint all of the initial approaches routinely fail to produce resource allocations with a non-zero probability of completing by Λ. Our results show that by first minimizing the expected makespan of our resource allocations, we can easily find resource allocations with very small but provably non-zero robustness values. Although these resource allocations are poor solutions to our original problem of finding robust resource allocations, they can be used as seeds for our Genetic Algorithm techniques for maximizing robustness. The results of our simulation study demonstrate the utility of this approach for maximizing the robustness of resource allocations given both loose and tight Λ completion time constraints.

The major contributions of this paper include (1) a novel methodology for generating robust resource allocations given tight completion time constraints, (2) the design of resource allocation heuristics that can leverage this methodology to maximize the robustness of resource allocations subject to a constraint on the maximum allowable completion time Λ, (3) design of a local search technique for this problem domain and the associated search space analysis, and (4) the design of a path relinking crossover operator that utilizes local search within a Genetic Algorithm.

The remainder of this work is organized in the following manner. Section 2 describes the model of stochastic compute node completion times used in this work. A brief introduction to the stochastic robustness framework is presented in Section 3 along with an introduction to using the stochastic robustness metric within a heuristic. Four search based algorithms designed for this environment and one simple greedy heuristic are described in Section 4. The parameters of the simulation study used to evaluate a direct maximization of robustness are discussed in Section 5. Section 6 defines our methodology for generating resource allocations that are guaranteed to have non-zero robustness even for tight Λ values. This section presents the details of a steady-state genetic algorithm that successfully applies this approach for defining an initial population that can be used in each of our complex heuristics. The results of this combined approach are provided in Section 6.4. A sampling of the relevant related work is presented in Section 7. Section 8 concludes the paper.

2. System model environment

In this environment, we are concerned with allocating a set of N independent applications to a collection of M dedicated heterogeneous compute nodes. Each of the N applications must process data that is produced by the system's sensors. In a static resource allocation, all of the applications to be executed are known in advance of performing a resource allocation. In this environment, although the data sets to be processed vary every Λ time units, the applications executed are the same and their assignment to compute nodes remains fixed.

Application execution times are known to be data dependent. The data sets produced by the system sensors vary with changes in the real world, causing application execution times to vary in unpredictable ways. For this reason, we model the execution time of each application i ($1 \leq i \leq N$) on each compute node j ($1 \leq j \leq M$) as a random variable [11], denoted by η_{ij}. We assume that, based on historical data or experiments, probability mass functions (pmfs) describing the possible execution times for each application on each compute node exist and are available for each η_{ij}. (See [12] for some techniques for estimating these pmfs.)

Using the execution time pmfs, we can produce a completion time distribution for each compute node for a given resource allocation. The finishing time of each compute node is calculated as the sum of the execution time random variables for each application assigned to that compute node [13]. Let n_j be the number of applications assigned to compute node j. The pmf for the finishing time of compute node j, referred to as a local performance characteristic ψ_j, can be expressed as follows:

$$\psi_j = \sum_{i=1}^{n_j} \eta_{ij}. \tag{1}$$