Module Allocation for Maximizing Reliability of Distributed Computing Systems using Dynamic Greedy Heuristic

Surinder Kumar

Abstract: This paper deals with the problem of module allocation (i.e., to which processor should each task of an application be assigned) in heterogeneous distributed computing systems with the goal of maximizing the system reliability. The module assignment problem for more than three processors is known to be NP-hard, and therefore satisfactory suboptimal solutions obtainable in an acceptable amount of time are generally sought. We propose a new intelligent technique based on dynamic module allocation which uses greedy search algorithm for this problem. Performance of the algorithm depends on number of modules, number of processors, and the ratio of average communication time to average computation time and module interaction density of application. The effectiveness and efficiency of our algorithm is compared with recently proposed module allocation algorithms for maximizing system reliability available in literature.

Keywords: Module assignment, Distributed computing, Reliability, Dynamic Greedy heuristic, Module interaction graph.

I. INTRODUCTION

Distributed computing (DC) systems have been widely deployed for executing computationally intensive applications with diverse computing requirements. A DC system generally consists of a suite of geographically distributed dissimilar processors interconnected via communication networks. In such a system, a parallel application can be decomposed into a number of cooperating modules that are distributed to the various processors for execution. In reality, however, the performance of a parallel application running on a DC system heavily depends on the mapping of modules partitioned from the application onto the available processors in the system, referred to as the module assignment problem which, if not properly handled, can nullify the benefits of DC systems. Module assignment can be performed statically or dynamically [1]. Static module assignments take place during compile time before running the application and remain unchanged until the end of the execution. In contrast, dynamic module assignments are performed at runtime. Since static mapping does not incur overheads on the execution time of the mapped application, more complex mapping algorithms than the dynamic ones can be adopted. When all information needed for the assignment, such as the structure of the parallel application, the execution costs of modules, the amount of data to be transferred among modules,
machine tardiness problems [16], multi objective optimization problem [17], just to name a few. Thus, we intend to further extend the application of DG to the module assignment problem in the distributed computing systems. To the best of our knowledge, this study is the first to pioneer the use of DG heuristic for the problem considered. The remainder of this paper is organized as follows. After formulating the problem in Section 2, the proposed DG heuristic is elaborated in Section 3. Finally, some concluding remarks are made in Section 4.

II. PROBLEM FORMULATION

The general problem of optimally mapping independent modules to machines in a DC suite has been shown to be (weakly) NP-complete. To address this problem, a number of heuristics have been proposed and can be categorized into fast and slow algorithms according to the time it takes to obtain the sub-optimal solution. Slow heuristics, such as by ant optimization and by genetic algorithm, take a significantly longer time than fast heuristics, however, they aim to find better solutions.

In [18] eleven heuristics are compared and it is concluded that the greedy heuristic min–min performs well in comparison to the other techniques. Paper [19] reports that the technique of ant optimization outperforms min-min and genetic algorithm at the expense of a much longer mapping process. However, only fast DG heuristics can be adopted in the following situations, where the mapping process is performed during the execution of the mapped modules. There exists a large body of the literature covering many module and heterogeneous computing models. In this paper, we consider the module assignment problem with the following characteristics.

A distributed application is characterized by a task interaction graph (MIG) \( G(V, E) \), where \( V \) is a set of \( N \) nodes indicating the \( N \) modules of the application, and \( E \) is a set of edges specifying the communication requirements among these tasks. A weight \( c_{ij} \) associated with the edge between modules \( i' \) and \( j' \) represents the amount of data to be transferred between the two modules. The processors in the system are heterogeneous. Hence, a module will incur different execution costs if it is executed on different processors. Let \( K \) be the number of processors in the DC systems and \( EEC = \{x_{ij} \mid x_{ij} \in \mathbb{R} \} \) be the estimated execution cost matrix where \( x_{ij} \) denotes the execution cost of module \( i' \) on processor \( k' \). On the other hand, all of the communication channels are assumed to be non-uniform. That is, an identical amount of data, if transmitted through different communication channels, will incur different communication costs. Define \( d_{ij} \) as distance-related communication cost associated with one unit of data transferred from processor \( k' \) to processor \( i' \), such that if modules \( i' \) and \( j' \) are executed on processors \( k' \) and \( i' \) respectively, then a communication cost of \( c_{ij}d_{ij} \) is incurred. The distance metric is symmetric, i.e., \( d_{ij} = d_{ji} \). Furthermore, we assume that no communication cost is incurred if two interacting modules are assigned to the same processors.

The allocation constraints depend on the characteristics of both the application involved (resource requirements by the modules) and on the available resource capacities of the processors in the system. To describe the allocation constraints, let \( r_i \) denote the resource requirement of module \( i' \) and let \( R_p \) denote the available resource capacity of processor \( p' \).

A particular module assignment can be represented by an integer vector \( \psi \) of size \( N \) which is a mapping from the set of modules to the set of processors. It contains the indices of the processors to which each module is allocated, i.e. \( [i] = k \), if module \( i' \) is allocated to processor \( k' \). Let \( \Omega \) be the set of all mappings from the set of modules to the set of processors. Our objective is to minimize the total execution and communication costs incurred by the module assignment subject to the resource constraint. Hence, the considered module assignment problem can be formulated as

\[
\text{MinimizeCost}(\psi) = \sum_{i=1}^{N} X_{i\psi(i)} + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C_{ij}d_{\psi(i)\psi(j)} \quad \text{for all } \psi \in \Omega \tag{1}
\]

\[
\text{Such that } \sum_{i=1}^{N} \psi(i)r_i \leq R_k \quad \forall \ k = 1, 2, 3, \ldots, K \tag{2}
\]

In the above formulation, objective function (1) consists of two parts. The first is the sum of the execution costs and the second the sum of the communication costs incurred between interacting modules residing on different processors. Constraint (2) ensures that the total resource requirements of the modules assigned to each processor must not exceed its resource availability.

III. DYNAMIC GREEDY HEURISTIC

The Dynamic Greedy (DG) algorithm is nothing but a simple greedy algorithm applied on dynamic distributed computing systems. In this approach, we are using a simple greedy search algorithm or greedy heuristic to obtain the next appropriate processor so that it maximizes the system reliability.
System reliability can be maximized by decreasing communication time in the whole process of module allocation as to minimize the ratio of communication time to computation time. As the communication time of local machine is nearly zero, so we will select the local worker for module allocation first or we will give higher priority to the local processor than remote processor. We will also see that the processor that we are using for allocation is free or not. If the processor is free or ideal then a module can be allocated to that processor. (In initial step to algorithm we should have all modules in module queue).

**A. Greedy Algorithm**

1. n := length[s]
2. A := {a_i}
3. J := 1
4. For k:= 2 to n do
5.   If s_k> = f_j // compatible activity
6.   then A := A union {a_k}
7. k:= k
8. Return A

**IV. OBJECTIVES**

- To learn the Greedy algorithmic paradigm
- To apply Greedy methods to solve several optimization problem
- To analyse the correctness of greedy algorithms

**V. CONCLUSION**

To the best of our knowledge, this is the first report on the application of dynamic greedy heuristic to the module assignment problem in distributed computing systems. We used the simple greedy heuristic algorithm to minimize the total time required to execute the application. Furthermore, the DG has the advantages that it has fewer parameters that need to be tuned than the competing algorithms, and it is a rather simple, easily implementable algorithm compared to HPSO algorithm and HBMO algorithm. We are currently extending the application of the proposed DG algorithm to another version of the module assignment problem where each processor and each communication link has a failure ratio and the goal is to maximize the system reliability for accomplishing the module execution.

**ACKNOWLEDGMENT**

We take this opportunity to express our deepest gratitude and appreciation to all those who have helped us directly or indirectly towards the successful completion of this paper.

**REFERENCES**