International Journal of Basic Sciences and Applied Computing (IJBSAC)
ISSN: 2394-367X, Volume-2 Issue-1, September 2016

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, the computing nodes and the communication network, is known before the application execution, static mapping can be exploited. In the general form of static mapping, a parallel application is modelled using a module interaction graph (MIG). In the MIG model, the vertices represent application modules and the edges represent inter-module communications. There are no precedence relations between tasks. A module incurs an execution cost that may vary from one processor to another, and two interacting modules that are not assigned to the same processor incur a communication cost. Certain resource constraints, such as memory and processing load constraints, may be present at each processor. The goal of the module assignment is to minimize the sum of the total execution and communication costs by appropriately allocating the modules to the processors without violating any of the constraints.

Due to its key importance on performance, the module assignment problem has been extensively studied and numerous methods have been reported in the literature. These allocation schemes can be classified into two categories. First, there are the exact methods that try to find the optimal allocation for the given objective. The existing approaches are developed using different strategies such as graph theoretic techniques [2], integer programming [3], and state space search [4, 5, 6]. However, as the problem is NP-hard for more than three processors [4], these methods are limited by the amount of time and memory needed to obtain an optimal solution since they grow as exponential function of the problem order.

On the other hand, heuristic algorithms provide fast and effective means for obtaining suboptimal solutions. These techniques require less computation time than exact methods. They are useful in applications where an optimal solution is not obtainable within a critical time limit. They are also applicable to large-size problems. Therefore, development of effective heuristic procedures is gaining importance among researchers. Different algorithms are used for developing heuristic methods such as genetic algorithm (GA) [7, 8], simulated annealing (SA) [9], hybrid particle swarm optimization (HPSO) [10], harmony search (HS) [11] and honey bee mating optimization (HBMO) [20].

Because of the intractable nature of the module assignment problem, new efficient techniques are always desirable to obtain the best-possible solution within a reasonable amount of computation time. The Dynamic greedy (DG) heuristic is an effective stochastic local search algorithm recently developed for combinatorial optimization problems which has exhibited state-of-the-art performances for several problems from computer science and engineering, such as set covering problems [12, 13], flow shop scheduling problems [14, 15], Sequencing single-

Revised Version Manuscript Received on September 08, 2016.

Surinder Kumar, Assistant Professor, Department of Mathematics, Dayanand Anglo Vaidik College, Chandigarh (Punjab). India.

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 $EEC = \{x_{ik}\}_{N*K}$ be the estimated execution cost matrix where x_{ik} 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_{ki} as distance-related communication cost associated with one unit of data transferred from processor *'k'* to processor *'l'*, such that if modules *'i'* and *'j'* are executed on processors 'k' and 'l' respectively, then a communication cost of $c_{ik}d_{kl}$ is incurred. The distance metric is symmetric, i.e., $d_{kl}= d_{lk}$. 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 *'ψ'* 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 *Ω* 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

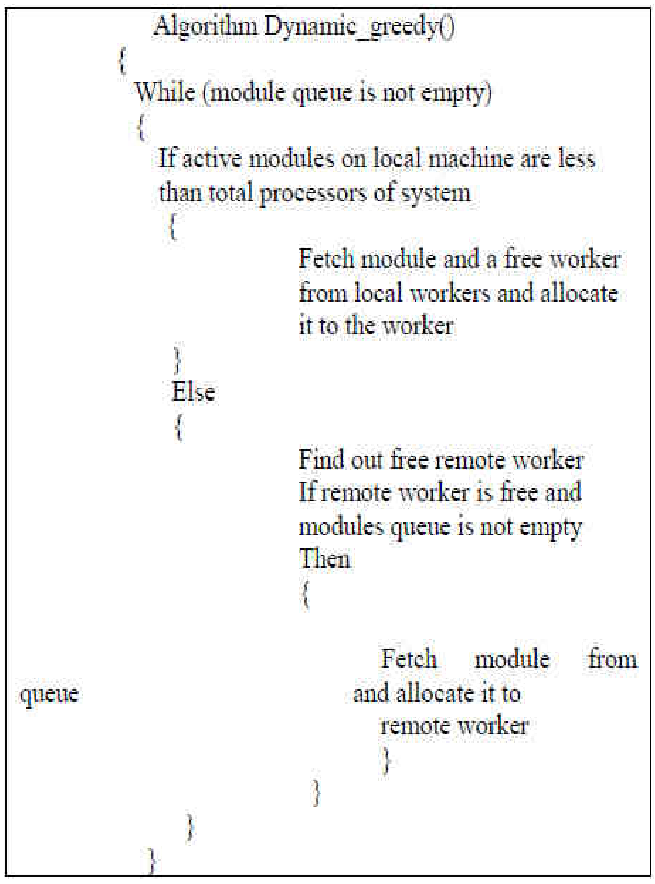
$$Minimize Cost(\psi) = \sum_{i=1}^{N} X_{i\psi(i)} + \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} C_{ij} d_{\psi(i)\psi(j)} \qquad for\,all\ \psi \in \Omega \qquad (1)$$

$$Such\,that \qquad \sum \psi(i) r_i \le R_k \qquad \forall\ k = 1,2,3,\ldots,K \qquad (2)$$

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.



```
Algorithm Dynamic_greedy()
{
  While (module queue is not empty)
  {
    If active modules on local machine are less
    than total processors of system
     {
              Fetch module and a free worker
              from local workers and allocate
              it to the worker
     }
     Else
     {
              Find out free remote worker
              If remote worker is free and
              modules queue is not empty
              Then
              {
                  Fetch module from queue
                  and allocate it to
                  remote worker
                  }
              }
     }
}
```

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_1}
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. j:= 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

1. S Casavant, T., Kuhl, J.G.,1988. “A taxonomyof scheduling in general-purpose distributed computing systems”. IEEE Transaction on Software Engineering 14 (2), 141–154.
2. Stone, H.S., 1977. “Multiprocessor scheduling with the aid of network flow algorithms”. IEEE Transactions on Software Engineering SE 3 (1), 85–93.
3. Ernst, A., Jiang, H., Krishnamoorthy, M., 2006. “Exact solutions to module allocation problems”. Management Science 52, 1634–1646.
4. Chern, M.S., Chen, G.H., Liu, P., 1989. “An LC branch-and-bound algorithm for module assignment problem”. Information Processing Letters 32,61–71.
5. Sinclair, J.B., 1987. “Efficient computation of optimal assignments for distributed modules”. Journal of Parallel and Distributed Computing 4, 342–361.
6. Tom, A.P., Murthy, C.S.R., 1999. “Optimal module allocation in distributed systems by graph matching and state space search”. Journal of Systems and Software 46 (1), 59–75.
7. Chockalingam, T., Arun kumar, S., 1995. “Genetic algorithm based heuristics for the mapping problem”. Computer and Operations Research 22, 55–64.
8. Hadj-Alouane, A.B.,Bean, J.C., Murty, K.G., 1999. “A hybrid genetic/optimization algorithm for a module allocation problem”. Journal of Scheduling 2,189–201.
9. Hamam, Y., Hindi, K.S., 2000. “Assignment of program modules to processors: a simulated annealing approach”. European Journal of Operational Research 122, 509–513.
10. Yin, P.Y., Yu,S.S., Wang, P.P., Wang, Y.T., 2006. “A hybrid particle swarm optimization algorithm for optimal module assignment in distributed systems”. Computer Standard and Interface 28, 441–450.
11. Zou, D.X., Gao, L.Q., Li, S., Wu, J.H., Wang, X., 2010. “A novel global harmony search algorithm for module assignment problem”. Journal of Systems and Software 83 (10), 1678–1688.
12. Jacobs, L.W., Briscoe, M.J., 1995. A local-search heuristic for large set-covering problems”. Naval Research Logistics Quarterly 42, 1129–1140.
13. Marcher, E., Steinbeck, A., 2000.“An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling”. Lecture Notes in Computer Science 1803, 367–381.
14. Pan, Q.K., Wang, L., Zhao, B.H., 2008. “An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with make span criterion”. International Journal of Advanced Manufacturing Technology 38, 778–786.
15. Ruiz, R., Stützle, T., 2007. “A simple and effective iterated greedy algorithm for the permutation flow shop scheduling problem”. European Journal of Operational Research 177 (3), 2033–2049.
16. Ying, K.C., Lin, S.W., Huang, C.Y., 2009. “Sequencing single-machine tardiness problems with sequence dependent setup times using an iterated greedy heuristic”. Expert Systems with Applications 36, 7087–7092.
17. Dubois-Lactose, J., López-Ibã nez, M., Stutz, T. “A hybrid TP + PLS algorithm for bi-objective flow-shop scheduling problems”. Computers and Operations Research, in press.
18. T.D. Braun, D. Hansen, R.F. Freund, H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, B. Yao, “A comparison of eleven static heuristics for mapping a class of independent modules onto heterogeneous distributed computing systems”, Journal Parallel and Distributed Compute. 61 (6) (2001)

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

81–837.
19. G. Ritchie, J. Levine, "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments", in: Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group, 2004.
20. Qin-Ma Kang, Hong He, Hui-Min Song, Rong Deng. "Module allocation for maximizing reliability of distributed computing systems using honeybee mating optimization". Original Research Article Journal of Systems and Software, Volume 83, Issue 11, November 2010, Pages 2165-2174.