Observing the Performance of Greedy algorithms for dynamic load balancing in Heterogeneous Distributed Computing System

Bibhudatta Sahoo#, Dilip Kumar*, Sanjay Kumar Jena#

#Department of Computer Science & Engineering, NIT Rourkela, India

**Department of Computer Science & Engineering, NIT Jamshedpur, India*

[1] bdsahu@nitrkl.ac.in

[2] dilip.cse@nitjsr.ac.in

[3]skjena@nitrkl.ac.in

Abstract— **Distributed systems have been an active research area in computer science for the last decade, task allocation and load balancing have been a major issue associated with such systems. The load-balancing problem, attempts to compute the assignment with smallest possible makespan (i.e. the completion time at the maximum loaded computing node). Load balancing problem is a NP hard problem. This paper discusses the performance of some simple heuristic algorithms to solve load balancing problem with makespan as performance metric to minimize the makespan**.

Keywords— **Heterogeneous distributed system, dynamic load balancing, makespan, greedy heuristic.**

I. Introduction

Distributed heterogeneous computing is being widely applied to a variety of large size computational problems. These computational environments are consists of multiple heterogeneous computing modules, these modules interact with each other to solve the problem. In a Heterogeneous distributed computing system (HDCS), processing loads arrive from many users at random time instants. A proper scheduling policy attempts to assign these loads to available computing nodes so as to complete the processing of all loads in the shortest possible time.

The *resource manager* schedules the processes in a distributed system to make use of the system resources in such a manner that resource usage, response time, network congestion, and scheduling overhead are optimized. There are number of techniques and methodologies for scheduling processes of a distributed system. These are *task assignment*, *load-balancing*, *load-sharing* approaches [7],[9],[10]. Due to heterogeneity of computing nodes, jobs encounter different execution times on different processors. Therefore, research should address scheduling in heterogeneous environment. The distributed nature of underlying resources presents problems not present in closely coupled systems, such as communication overheads, or heterogeneity of resources. A poor allocation of tasks to processors could nullify the benefits of using a distributed system by inefficiently utilizing the system resources [2].

. The load-balancing problem, aim to compute the assignment with smallest possible makespan (i.e. the completion time at the maximum loaded computing node) The load distribution problem is known to be NP-hard [4],[5] in most cases and therefore intractable with number of tasks and/or the computing node exceeds few units. Here, the load balancing is a job scheduling policy which takes a job as a whole and assign it to a computing node [2].This paper considers the problem of finding an optimal solution for load balancing in heterogeneous distributed system. The rest of the paper is organized as follows. The next section discusses Heterogeneous distributed computing system (HDCS) structure and the load-balancing problem. *Section 3* describes the different dynamic load distribution algorithms using greedy paradigm. We have simulated the behaviour of different greedy load balancing algorithm with our simulator developed using Mat lab. The results of the simulation present the performance of resource allocation algorithms with scalability of computing nodes and varying tasks arrival in *Section 4.* Finally, conclusions and directions for future research are discussed in *Section 5*

II. System and problem model

A. Heterogeneous Distributed Computing System

Heterogeneous distributed computing system (HDCS) utilizes a distributed suite of different high-performance machines, interconnected with high-speed links, to perform different computationally intensive applications that have diverse computational requirements. Distributed computing provides the capability for the utilization of remote computing resources and allows for increased levels of flexibility, reliability, and modularity. In heterogeneous distributed computing system the computational power of the computing entities are possibly different for each processor [1],[3],[4]. A large heterogeneous distributed computing system (HDCS) consists of potentially millions of heterogeneous computing nodes connected by the global

Internet. The applicability and strength of HDCS are derived from their ability to meet computing needs to appropriate resources [2],[3],[9]. Resource management sub systems of the HDCS are designated to schedule the execution of the tasks that arrive for the service. HDCS environments are well suited to meet the computational demands of large, diverse groups of tasks. The problem of optimally mapping also defined as matching and scheduling.

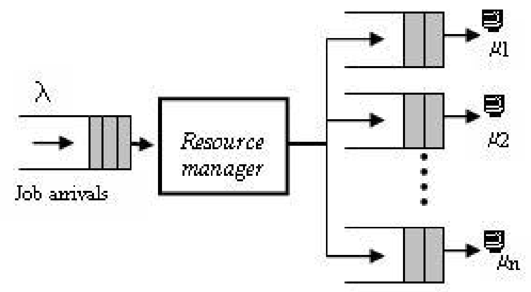

Fig. 1 A model of distributed computing system

We consider a heterogeneous distributed computing system (HDCS) consists of a set of m { $M_1, M_2, \ldots M_m$} independent heterogeneous, uniquely addressable computing entity (computing nodes). Let there are n number of jobs with each job j has a processing time t_j are to be processed in the HDCS with m nodes. Hence the generalized load-balancing problem is to assign each job to one of the node M_i so that the loads placed on all machine are as "balanced" as possible [5].

B. *Model of load balancing problem on Distributed System*

This section presents a mathematical model for load balancing problem based on min-max criterion. Objective of this formulation is to minimize the load at the maximum loaded processor. Let A(i) be the set of jobs assigned to machine Mi; hence the machine Mi needs total computing time $T_i = \sum_{j \in A(i)} t_{ij}$ which is otherwise known as *(Li)*load on machine Mi. The basic objective of load balancing is to minimize make span, which is defined as maximum loads on any machine *(T* = maxi *Ti*). This problem can be expressed as linear programming problem, with the objective to (load of the corresponding assignment).

Minimize L

$$\sum_i x_{ij} = t_{ij} \text{ , for all j} \in \text{A(i)}$$

$$\sum_j x_{ij} < L, \text{ for all i} \in M$$

$$x_{ij} \in \{0, \mathrm{t_{ij}}\}$$

$$x_{ij} = \{0, \mathrm{t_{ij}}\} \quad \text{, for all j} \in \text{A(i), i} \in M_j$$

$$x_{ij} = 0 \text{ , for all j} \in \text{A(i), i} \notin M_j$$

Where $M_j \subseteq M$;set of machines to which the job j can be assigned.

The problem of finding an assignment of minimum makespan is NP-hard [5]. The solutions to this can be obtained using a dynamic programming algorithm *O(n L^m)*, where L is the minimum makespan.

Due to the complexity of load balancing problem, most of researchers proposed heuristic algorithms, while optimal algorithm are developed for only restricted cases or for small problems[4]. Greedy algorithmic technique always makes the choice that looks best at the moment to solve optimization problems with the hope that this choice will lead to a globally optimal solution. In this paper we have analyse the performance of resource allocation schemes on the HDCS where all computing nodes are heterogonous and characterised by the different service rate μ. For any to computing nodes M_i, and M_j , $\mu_i \neq \mu_j$.

C. *Task model*

The tasks are arriving from the different nodes to the resource manager has the probability to be allocated to any of the m computing nodes. Hence the tasks are characterized by expected time of completion (ETC) on all m computing nodes, can be represented as follows, In ETC matrix, the elements along a row indicate the execution time of a given task on different machine[12] [14]

$$\begin{array}{cccccccc} & M_1 & M_2 & \cdots & M_J & \cdots & M_m \\ T_1 & t_{11} & t_{12} & \cdots & t_{1j} & \cdots & t_{1m} \\ T_2 & t_{21} & t_{22} & \ldots & t_{2j} & \ldots & t_{2m} \\ T_i & t_{i1} & t_{i2} & & t_{ij} & & t_{im} \\ T_n & t_{n1} & t_{n2} & & t_{nj} & & t_{nm} \end{array}$$

These tasks are arrived to the system with Poisson distribution and their expected times to complete the execution are uniformly distributed. Simulation is carried out with the ETC matrix and the different greedy allocation policy.

III. GREEDY RESOURCE ALLOCATION ALGORITHMS FOR LOAD BALANCING

A. *Simple greedy algorithm for load Balancing*

In a HDCS, each task can have a different execution time on each machine. We have used heuristic algorithms to map tasks into the computing nodes of the HDCS. The simple greedy base algorithm is as follows.

```
Algorithm 1: Greedy resource allocation algorithm.
Input: MaxTask, MaxNode.
Output: makespan
 1:  Generate ETC Matrix.
 2:  Generate Arrival Times for each task with
     arrival rate (λ).
 3:  for i=1 to MaxTask
 4:      for j=1 to MaxNode
 5:      Allocate task i to Node with minimum load.
 6:      end for
 7:  end for
 8:  Compute make span.
```

A simple greedy algorithm [5] makes one pass the jobs in any order and assigns the job **j** *to the computing node* **M_i** *that having minimum load so far. The allocations* *are based upon the ETC of n tasks on m computing nodes.*

B. Load balancing algorithm with greedy heuristic

We have implemented three different greedy heuristic task allocation algorithm based on (i) first come First serve (FCFS) based on arrival of the task to the resource manager, (ii) max-min, and Min-max. We have also used a randomized scheduler that represents the average major of the makespan. [14] The scheduler starts the tasks in the order of their submission. If resources are not sufficient to start the task then the scheduler waits until tasks can be started. The other tasks in the submission queue are stalled. The first come first serve (FCFS) algorithms uses job arrival time as the heuristic to assign the jobs to the computing nodes having minimum ETC value. The details of FCFS algorithm that uses ETC matrix and arrival time to decide allocation are shown below.

```
Algorithm 2: First Come First Served (FCFS) Resource
allocation algorithm
Input: MaxTask, MaxNode, ETC Matrix, Arrival Times
Output: Make span, Allocation
// B[ ] holds the completion times of Nodes
  1:  B [1: MaxNode]=0
  2:  // Allocation of Tasks to Nodes
  3:  for i=1 to MaxTask do
  4:      if (i ≤ MaxNode ) then
  5:          Allocate Task $T_i$ to Node $P_i$
  6:          Waiting Time($T_i$)=0
  7:          TurnAroundTime($T_i$)= ETC(i,i);
  8:      B(i) =TurnAroundTime($T_i$)+ArrivalTime($T_i$)
  9:      else
 10:          $P_j$ =Node with minimum value in B
 11:          Allocate Task $T_i$ to Node $P_j$
 12:          Waiting Time(Ti)=B($T_i$)-Arrival Time($T_i$)
 13:  TurnAroundTime($T_i$)=
                         Wating Time($T_i$)+ETC(i, $P_j$)
 14:  B( $P_j$) = TurnAroundTime($T_i$)+ArrivalTime($T_i$)
 15:  end if
 16:  end for
 17:  // Make span computation
 18:  Make span = Maximum(B)
```

The Max-min heuristic sends the task with maximum completion time for execution. This strategy is useful in a situation where completion time for tasks varies significantly. Using this heuristic, the tasks with long completion time are scheduled first on the best available machines and executed in parallel with other tasks. This leads to better load-balancing and better total execution time. The Min-max heuristic also assigns the task with minimum ETC on a machine to that machine so as to minimize makespan. A frame work for max-min algorithm is described as follows.

```
Algorithm3: Max-Min Resource Allocation Algorithm
Input: MaxTask, MaxNode, ETC Matrix, Arrival Times
Output: Make Span, Allocation
// B[] holds the completion times of Nodes
  1:  B[1:MaxNode] =0
  2:  for i=1 to MaxTask
  3:  $P_i$= Node that has minimum completion time for $T_i$
  4:      $t_i$=Minimum completion time
  5:  end for
  6:  for i=1 to MaxTask
  7:      if (B($P_i$) = = 0)
  8:          B($P_i$) = Arrival Time($T_i$)+ $t_i$
  9:      elseif (Arrival Time($T_i$) > B($P_i$))
 10:          B($P_i$) = Arrival Time($T_i$)+ $t_i$
 11:      else
 12:          B($P_i$) = B($P_i$) + $t_i$
 13:      end if
 14:  end for
 15:  // Make span computation
 16:  Make span=maximum(B)
```

IV. Simulation results

Simulation is carried out with the simulator designed using Mat lab. The number of task arrives are based on Poisson distribution with their expected time of completion of different computing nodes are generated using uniform distributions. The allocations of computing nodes are decided by the central resource manager. All the greedy algorithms are fast and polynomial bounded irrespective of their ability to generate sub-optimal solutions for optimization problem.

I

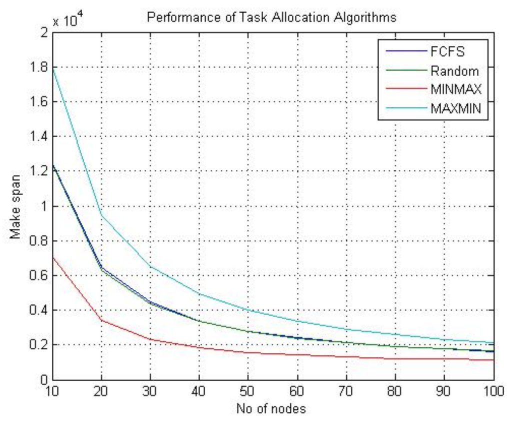

Fig. 2 Performance of heuristic algorithms against the number of computing node

We have simulated against the makespan for a task pool of 500 tasks that arrives using Poisson distribution. It is assumed that the service rate of the task is higher than the rate of arrival of the task, so that whatever the task arrives at a point of time to the resource manager entered to the task queue. The observation from Fig 2 leads to the selection of no of nodes fur further simulation. We have use node size to be 60 so as to study the performance of greedy algorithms on HDCS

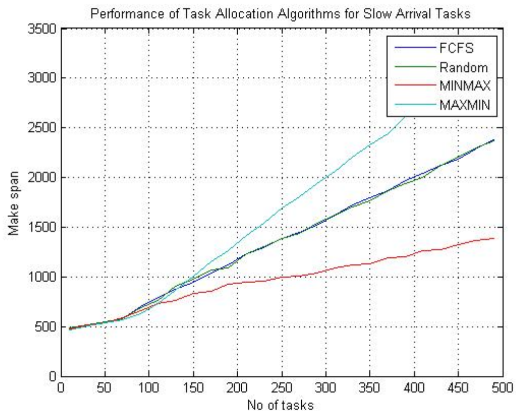

Fig. 3 Makespan for 60 nodes for slow arrival of tasks

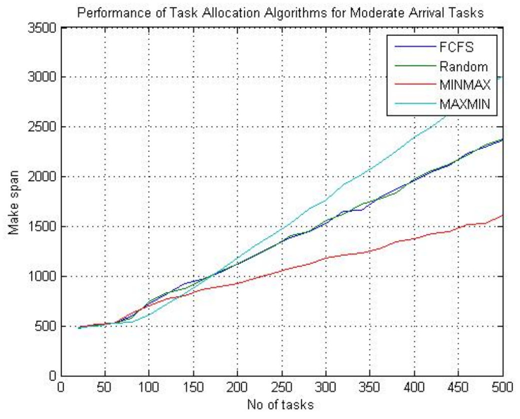

Fig. 4 Makespan for 60 nodes for moderate arrival of tasks

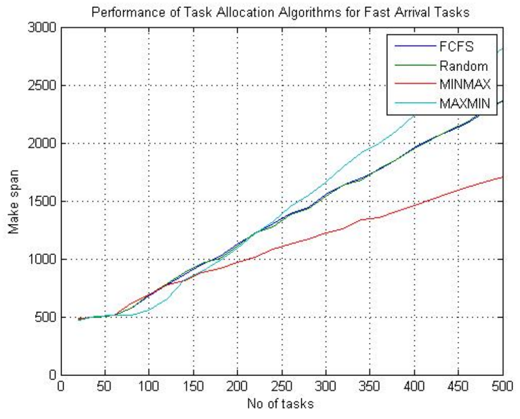

Fig. 5 Makespan for 60 computing nodes for fast arrival of tasks

V. Observation and Conclusions

This paper studies performance of different greedy algorithms to solve dynamic load balancing in heterogeneous computing system. Simulation results indicate that the performance of min-max algorithm found to be best method irrespective no of computing nodes with the system. We analyzed the performance of allocation schemes with different arrival rate of task on system with heterogeneous computing capability. The simulation results indicate the optimal performance by min-max algorithm.

References

[1] Marta Beltran, Antonio Guzman, and Jose Luis Bosque, Dealing with heterogeneity in load balancing algorithm, Proc. 5th IEEE International Symposium on Parallel and Distributed Computing, Timisoara, Romania, 2006, 123-133.

[2] Jie Li, & Hisao Kameda, Load balancing problems for multiclass jobs in distributed/parallel computer systems, IEEE Transactions on Computers, 47(3), 1998, 322-332.

[3] H. C. Lin, and C. S. Raghavendra, A dynamic load-balancing policy with a central job dispatcher (LBC), IEEE Transactions on Software Engineering, 18(2), 1992, 148-158.

[4] Gamal Attiya & Yskandar Hamam, Two phase algorithm for load balancing in heterogeneous distributed systems, Proc. 12th IEEE Euromicro conference on Parallel, Distributed and Network-based processing, Coruna, Spain 2004, 434-439.

[5] Jon Kleinberg & Eva Tardos, Algorithm Design (Pearson Education Inc. 2006).

[6] Helen D. Karatza, & Ralph C. Hilzer, Load sharing in heterogeneous distributed systems, Proceedings of the Winter Simulation Conference, 1, San Diego California, 2002 Page(s): 2002, 489 – 496.

[7] Jie Wu, Distributed system design,(CRC press, 1999)

[8] Y.Zhang, H.Kameda & S.L.Hung, Comparison of dynamic and static load-balancing strategies in heterogeneous distributed systems, IEE proceedings in Computer and Digital Techniques,144(2), 1997, 100-106.

[9] Bora Ucar, Cevdet Aykanat, Kamer Kaya, & Murat Ikinci, Task assignment in heterogeneous computing system, Journal of parallel and Distributed Computing, 66, 2006, 32-46.

[10] Marta Beltran, Antonio Guzman, & Jose Luis Bosque, Dealing with heterogeneity in load balancing algorithm, Proc. 5th IEEE International Symposium on Parallel and Distributed Computing, Timisoara, Romania, 2006, 123-132.

[11] Sivarama P. Dandamudi, Sensitivity evaluation of dynamic load sharing in distributed systems, IEEE Concurrency,6(3), 1998, 62-72.

[12] H. C. Lin, and C. S. Raghavendra, A dynamic load-balancing policy with a central job dispatcher (LBC), IEEE Transactions on Software Engineering, 18(2), 1992, 148-158.

[13] B. A. Shirazi, A. R. Hurson, & K. M. Kavi, Scheduling and load balancing in parallel and Distributed systems, CS press, 1995.

[14] H. J. Siegel, and S. Ali, Techniques for mapping tasks tomachines in heterogeneous computing, Systems, Journal of Systems Architecture, 46(8), 2000, 627—639.

[15] G. Schmidt, Scheduling with limited machine availability, European Journal of Operational Research, Elsevier, 121(1), 2000, 1-15.