Simulated Annealing based Heuristic Approach for Dynamic Load Balancing Problem on Heterogeneous Distributed Computing System

Bibhudatta Sahoo, Sanjay Kumar Jena, and Sudipta Mahapatra

Abstract—Load balancing problem on Heterogeneous Distributed Computing System (HDCs) deals with allocation of tasks to computing nodes, so that computing nodes are evenly loaded. Due the complexity of dynamic load balancing problem majority of researchers uses heuristic algorithm to obtain near optimal solutions. We have used consistent ETC (Expected Time to Compute) matrix in to study the performance of simulated annealing (SA) algorithm to minimize the makespan. Simulated annealing based resource allocation algorithm uses sliding widow techniques to select the tasks to be allocated to computing nodes after number of iterations. A new codification suitable for simulated annealing algorithm has been introduced for dynamic load balancing on HDCS. We have also presented three algorithms for move sets representations for SA. Several simulations run have been made on proposed simulated annealing algorithm for dynamic load balancing on HDCS, and compare with conventional first fit (FF), and randomized algorithm. The effect of simulated annealing based dynamic load balancing scheme has been on comparisons with first-fit, and randomized heuristic algorithm with task scalability.

Keywords—Dynamic load balancing, simulated annealing, heterogeneous distributed system, makespan

I. Introduction

Distributed heterogeneous computing is being widely applied to a variety of large size computational problems. The large scale computing problems requires more computing time, which can be meat by utilizing the ideal computing time of the vast computing resources distributed over the globe. 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. Modern distributed computing technology includes clusters, the grid, service-oriented architecture, massively parallel processors, peer-to-peer networking, and cloud computing [37]. The central *or serial scheduler* 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 [20]. Due to heterogeneity of computing nodes, jobs encounter different execution times on different processors. Therefore, research should address scheduling in heterogeneous environment.

In task assignment approach, each process submitted by a user for processing is viewed as a collection of related tasks and these tasks are scheduled to suitable nodes so as to improve performance. In load sharing approach simply attempts to conserve the ability of the system to perform work by assuring that no node is idle while processes wait for being processed. In load balancing approach, processes submitted by the users are distributed among the nodes of the system so as to equalize the workload among the nodes at any point of time. Processes might have to be migrated from one machine to another even in the middle of execution to ensure equal workload. Load balancing strategies may be static or dynamic [3, 7, 38].

To improve the utilization of the processors, parallel computations require that processes be distributed to processors in such a way that the computational load is spread among the processors. Dynamic load distribution (also called load balancing, load sharing, or load migration) can be applied to restore balance [7, 20]. In general, load-balancing algorithms can be broadly categorized as centralized or decentralized, dynamic or static, periodic or non-periodic, and those with thresholds or without thresholds [3, 7, 11]. Central scheduler or serial scheduler or load balancing service should be able to effectively control the computing resource for dynamic allocation to the tasks [13]. We have used a centralized load-balancing algorithm framework as it imposes fewer overheads on the system than the decentralized algorithm [38]. The load-balancing problem aims 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 [21] 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 [41]. The complexity of dynamic load balancing increases with the size of HDCS and becomes difficult to solve effectively. The exponential solution space for the load balancing problem can searched using heuristic techniques(GA, Tabu search, SA) to obtained suboptimal solution in the acceptable time[16,17,38]. These Artificial intelligence techniques have been used by researchers and proven to be effective in solving many optimization problems. Simulated Annealing (SA), proposed by Kirkpatrick et al.[39,42], has been used as a popular heuristic to solve optimization problems. Genetic Algorithms are used as one the popular technique to search the solution space to obtain sub-optimal solution.

This paper considers the problem of finding an optimal solution for load balancing in heterogeneous distributed system using stochastic iterative dynamic load balancing. The rest of the paper is organized as follows. *Section 2* highlights the contribution of various researchers in the related area of load balancing on distributed computing system and solving dynamic load balancing problem with simulated annealing. *Section 3* discusses Heterogeneous distributed computing system (HDCS) structure and the linear programming formulation of load-balancing problem. *Section 4* describes the task model and stochastic iterative dynamic load balancing techniques for dynamic load distribution. *Section 5* outlines the design details of simulated annealing. Finally, conclusions and directions for future research are discussed in *Section 6*.

II. RELATED WORKS

Load balancing for distributed computing system is a problem that has been deeply studied for a long time. Different heuristic algorithms are used by researcher to find suboptimal solutions for homogeneous and heterogeneous distributed system. Dandamudi [10] addressed dynamic load sharing in distributed systems and established that load sharing improves performance by moving work from heavily loaded nodes to lightly loaded nodes. A general model for heterogeneous distributed/parallel computer system proposed by Li and Kameda [11] and used to formulate the multiclass job load balancing problem as a nonlinear optimization problem. An algorithmic approach to load balancing problem is presented in [19]. Different form of linear programming formulation of the load balancing problem has been discussed along with greedy, randomized and approximation algorithm to produce sub-optimal solutions to the problem. The solution to this intractable problem was discussed under different algorithm paradigm. Modeling of optimal load balancing strategy using queuing theory was proposed by Francois Spies (1996). This is one of the pioneer works reported in the literature that presents an analytical model of dynamic load balancing techniques as M/M/k queue and simulate with fundamental parameters like load, number of nodes, transfer speed and overload rate [7]. Most appropriate queuing model for homogeneous distributed system can be M/M/m/n, has been analyzed in [9]. Queuing-Theoretic models for parallel and distributed system can be found in [6, 8]. General Job scheduling problem of n tasks with m machines, is presented as an optimization problem in [8] to minimize the makespan. Jong-Chen Chen and *et al.* [12] investigated the contribution made by evolutionary learning on dynamic load balancing problems in distributed computing system. Bora Ucar and *et al.* have considered the assignment of communicating tasks to heterogeneous processors[28], that uses a task clustering method based upon execution time to allocate the task though the heuristic techniques. A classification of iterative dynamic load balancing technique is discussed in [28].

SA is a heuristic method that has been implemented to obtain good solutions of an objective function defined on a number of discrete optimization problems. Simulated Annealing (SA), proposed by Kirkpatrick et al.[39,42], has been used as a popular heuristic to solve several optimization problems to obtain sub-optimal solution. A heuristic algorithm based on simulated annealing is discussed [31], which guarantees good load balancing on grid environment. A comparative study of the three algorithms (Hill-climbing, simulated annealing and genetic algorithms) is then carried out in [30] considering performance criteria as the amount of search time.

Makespan minimization of scheduling problem on identical parallel machines using simulated annealing has been presented by Lee and et al. in [41]. Grid Computing is one of heterogeneous distributed computing system geographically dispersed among several entities. Fidanova used simulated annealing to obtain near optimal solutions for scheduling problem in large grid [1]. Researchers have examined, 11 different heuristics(Opportunistic Load Balancing, Minimum Execution Time, Minimum Completion Time, Min–min, Max–min, Duplex, Genetic Algorithm, Simulated Annealing, Genetic Simulated Annealing, Tabu, and A*) on Mixed-machine heterogeneous computing (HC) environments to minimize the total execution time of the metatask[16,17]. Rahmani and Rezvani presented a genetic algorithm for static scheduling, which is again improved by simulated annealing to obtain an improvised solution[43]. They have also established that running time depends on the number of task.

Several researchers used SA and GA for load balancing on distributed computing system; however majority of the papers have no specific representation for simulated annealing algorithms for load balancing. This paper presents detail frame work for the simulated annealing algorithm to solve dynamic load balancing problem using ETC matrix for *n* number of tasks on *m* computing nodes. We have also presented three algorithm for move sets representations (i) inversion, (ii) translation, and (iii) switching for SA.

III. HETEROGENEOUS DISTRIBUTED COMPUTING SYSTEM MODEL

A. *Heterogeneous distributed computing system*

Heterogeneous distributed computing system (HDCS) utilizes a distributed suite of different high-performance nodes, interconnected with high-speed links, to perform different computationally intensive applications that have diverse computational requirements [20, 22, 23, 24, 37]. 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 as shown in figure 3.1 [10, 19, 27]. 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 [11, 20, 27]. Heterogeneity in DCS can be expressed by considering three systems attributes *(i) Processor with computing node*, (ii) *memory*, and *(iii) networking* [27]. The metrics used to quantify the processor or node processing power by means of processing speed and represented with FLOPS (Floating point Operations per Second) and can be measured through LINPACK. Memory attributes are measured as the available memory capacity to support the process. The networking attributes are the link capacity associated with transmission medium, propagation delay and available communication resources [3].

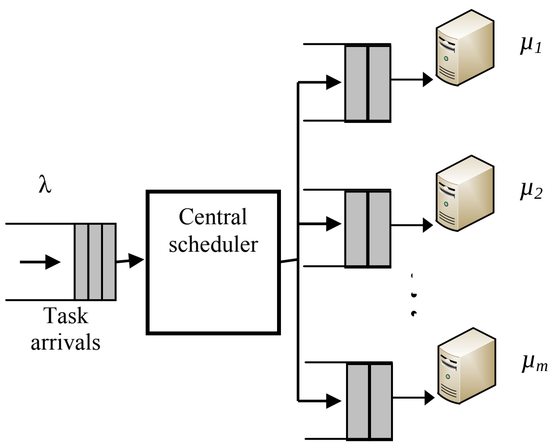

Figure: 3.1 Heterogeneous Distributed Computing System with central scheduler

In paper we have carried out simulation only considering processing power of the node, which can be represented as Markovian service time distribution [7, 9, 32]. In general, load-balancing algorithms can be broadly categorized as centralized or decentralized, dynamic or static, periodic or non-periodic, and those with thresholds or without thresholds [11, 20]. We have used a centralized load-balancing algorithm framework as it imposes fewer overheads on the system than the decentralized algorithm [1, 20]. Centralized load balancing algorithms requires the global information on computing nodes at a single location and the load balancing policy is initiated from the central location. Heterogeneity of architecture and configuration complicates the load balancing problem [20]. Heterogeneity can arise due to the difference in task arrival rate at homogeneous processors or processors having different task processing rates.

We have assumed that all computational tasks are capable of executed on any computing nodes of DCS. A single computing node that acts as a central scheduler or resource manager of the DCS collects the global load information of other computing nodes. Resource management sub systems of the HDCS are designated to schedule the execution of the tasks dynamically as that arrives 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. A basic assumption is that all computing nodes are always available for processing.

B. Load balancing problem in Heterogeneous distributed computing system

We have used the characterization model proposed by Shoukat Ali and et al as the basic framework to study the impact of system heterogeneity against different heuristic resource allocation algorithms [23]. We consider a heterogeneous distributed computing system (HDCS) consists of a set of M = $\{M_1, M_2, \ldots M_m\}$, m independent heterogeneous, uniquely addressable computing entity (computing nodes). Let there are T = $\{t_1, t_2, \ldots, t_n\}$ n number of tasks with each task t_i has an expected time to compute t_{ij} on node M_j. The entire task has expected time to compute on m nodes of HDCS. Hence the generalized load-balancing problem is to assign each task to one of the node M_j so that the loads placed on all nodes are as "balanced" as possible [19].

Let A(j) be the set of jobs assigned to node M_j; and T_j be the total time machine M_j have to work to finish all the task in A(j). Hence $T_j = \sum_{t_i \in A(j)} t_{ij}$; for all task in A(j). This is otherwise denoted as L_j and defined as load on node M_j. The basic objective of load balancing is to minimize make span, which is defined as maximum loads on any node (T = $\max_{j:1:m}$ (T_j). Let x_{ij} correspond to each pair (i, j) of node $M_j \in M$ and task $t_i \in T$.

- $x_{ij} = 0$; implies that task *i* not assign to node *j*.
- $x_{ij} = t_{ij}$; will indicate load of task *i* on node *j*.

For each task t_i we need $\sum_{j=1}^{m} x_{ij} = t_{ij}$; for all task $t_i \in T$

The load on node M_j can be represented as $L_j = \sum_{i=1}^{n} x_{ij}$, where $x_{ij} = 0$ whenever task $t_i \notin A(j)$. The load balancing problem aims to find an assignment that minimizes the maximum load. Let L be the load of a HDCS with m nodes. Hence the generalized load balancing problem on HDCS can be formulated as

Minimize L

$$\sum_{j=1}^{m} x_{ij} = t_{ij} \text{, for all } t_i \in T \quad (1)$$

$$\sum_{i=1}^{n} x_{ij} \leq L \text{, for all } M_j \in M \quad (2)$$

$x_{ij} \in \{0, t_{ij}\}$, for all $t_i \in T$ and $M_j \in M$

$x_{ij} = 0$, for all $t_i \notin A(j)$

Feasible assignments are one-to-one correspondence with x satisfying the above constraints [4]. Hence an optimal solution to this problem is the load L_i on a machine (corresponding assignment). The problem of finding an assignment of minimum makespan is NP-hard [19,21,29]. The problem is therefore untractable with number tasks or computing nodes (processors) exceeds a few units. The solutions to load balancing problem can be obtained using a dynamic programming algorithm with time complexity $O(n L^m)$, where L is the minimum makespan[19] The load balancing problem has been evenly treated, in both the fields of computer science and operation research. The algorithm approaches used for load balancing problem are roughly classified as (i) exact algorithms and (ii) heuristic algorithms [29, 45].

Queuing models are used as the key model for performance analysis and optimization of parallel and distributed system [11, 17]. The HDCS can be modeled as M/M/m/n (Markovian arrivals, Markovian distributed service times, m computing nodes as server, and space for $n \geq m$ tasks in the system) multi-server queuing system with m servers as computing nodes. However, the heterogeneous multi-server queuing systems are not adequately addressed in research with respect to certain quality of service [44].

The HDCS is modeled as M/M/m/n queuing system with node M_1 is the fastest computing node and M_m is the slowest computing node. Assume that service time follow exponential distribution with service rate so that $\mu_1 > \mu_2 > \ldots \mu_m$, where μ_I is the service rate of node M_i. The arrivals of the tasks at the central server or resource manager are modeled as Poisson with arrival rate λ. Each computing nodes can be modeled as shown in figure 2. The tasks that are to be executed at a node are under the control of local scheduler and the scheduling policy of the node is responsible for the execution of the assigned task. We have assumed FCFS policy is being used at computing nodes, which can be modeled as M/M/1 queuing system [44, 46].

IV. TASK MODEL AND ITERATIVE LOAD BALANCING TECHNIQUES

A. Task model on HDCS

In literature of distributed computing researchers have used two different task models as (i) Task graph(TG) or Task interaction graph(TIG)[7,8,9,43], (ii) expected time to compute(ETC) matrix[5,6,17,18,23]. The task graphs are both directed and undirected weighted graph that represents process or task to be executed, however majority of the models are not representing any mathematical model for quantifying task heterogeneity. In this paper we have use ETC matrix representation of task [23] that represents task heterogeneity and machine heterogeneity. The tasks are arriving from the different users or nodes to the central scheduler or or serial scheduler have the probability to be allocated to any of the m computing nodes. Hence the tasks are characterized by expected time to compute (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 nodes[23], in particular t_{ij} represent expected time to compute i^{th} task on machine M_j.

TABLE I
EXPECTED TIME TO COMPUTE (ETC) MATRIX

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

The ETC model presented in [23] are characterized by three parameters (i) machine heterogeneity, (ii)task heterogeneity and (iii)consistency. The task heterogeneity can be represented with two categories (i) *consistent* and (ii) *inconsistent*, here a consistent ETC matrix the computing nodes are arranged in the order of their processing capability or may be arranged as decreasing order of FLOPS. In particular a node M_i has a lower execution time than node M_j for task t_k , then $t_{ki} < t_{kj}$. Inconsistent ETC matrix is resulted in practice, when HDCS includes different type of machine architectures.(HPC clusters, Multi-core processor based workstations, parallel computers, work station with GPU units). In literature most of the task execution times are uniformly distributed[23, 24]. A consistent ETC matrix for ten tasks on five machines is shown on table II, which is taken from [23] .

To generate ETC matrix, we have used range base ETC generation technique discussed in [23] and added one component as arrival time of task. The arrival pattern of the task is based on Poisson distribution. For the analysis of the simulation results through the graph we have used expected completion time of task uniformly distributed {1, 500} time unit or seconds.

TABLE II
EXAMPLE OF CONSISTENT ETC MATRIX FOR 10 TASKS ON FIVE MACHINES

Node→ Task ↓	M1	M2	M3	M4	M5
t1	22	21	6	16	15
t2	7	46	5	28	45
t3	64	83	45	23	58
t4	53	56	26	42	53
t5	11	12	14	7	8
t6	33	31	46	25	23
t7	24	11	17	14	25
t8	20	17	23	4	3
t9	13	28	14	7	34
t10	2	5	7	7	6

B. Iterative centralized algorithms

We have used centralized load balancing algorithm, a central node collects the load information from the other computing nodes in HDCS. Central node communicates the assimilated information to all individual computing nodes, so that the nodes get updated about the system state. This updated information enables the nodes to decide whether to send the task to other nodes or accept new task for computation. The computing nodes depend on the information available with central node for all allocation decision. The two heuristic based resource allocation used to balance the load on computing nodes of HDCS are First Come First serve (FF), and Simulated Annealing (SA). A randomized resource allocation algorithm is selected along with the heuristic algorithms because the randomness can (probabilistically) guarantee average case behavior as well as it produces an efficient approximate solution to intractable problems. The FF algorithm follows the order of arrival time of the task with central scheduler. The *random task allocation algorithm* selects the node randomly from m nodes to allocate task t_j. SA based load balancing algorithm uses an iterative structure with stopping criteria as maximum number of iteration.

We have also assumed that tasks are independent and can be processed by any computing node in distributed environment. For stability it is also assumed that tasks must not be generated faster than the HDCS can process as shown in equation 3.

$$\sum_{i=1}^{n} \lambda_i \quad \leq \quad \sum_{j=1}^{m} \mu_j \qquad (3)$$

C. Coding Scheme for the Solution

Simulated annealing algorithms require a suitable representation and evaluation mechanism. In this case we have use a window structure of fixed length say k, with integer value assigned to individual element of the array of size k.. That on each step *k* no of task to be allocated to the computing node through simulated annealing with a minimized value of makespan. Task is assigned dynamically to the computing nodes on the fly. At the time of allocation there may be a large number of tasks are with central scheduler. A sliding window technique is used to select those tasks only that are in the window. The number of elements in the window is fixed is equal to the size of window. Figure 4.1, represents 10 tasks and their respective allocation to five computing node. Figure 4.2 shows the structure of allocation list, indicates the computing node. We have assumed that, current work load as dedicated tasks for each own node, so that the calculation of makespan is carried out from the time point when sliding window is selected.

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
M_5	M_3	M_5	M_3	M_2	M_2	M_4	M_4	M_1	M_1

Figure 4.1 allocation list of task to computing node

5	3	5	3	2	2	4	4	1	1

Figure 4.2 Allocation list

SA requires an appropriate representation to find the solution, we have used the window structure as shown in figure 4.2, the length of a array is the maximum number of task in the widow (window size) [38, 35]. The use of linear array helps to use the index as task number in the window so that a one dimensional list representation is selected. The individual element on window indicates the machine on which the corresponding task to be executed. Each window shows a possible allocation of computing nodes for which the makespan can be calculated from the ETC matrix. To prevent the nodes from overloading, before the task to be assigned to the node queue, a threshold is used. The percentage of acceptable queue for each node is calculated using formula:

$$\frac{number\ of\ acceptable\ node\ queues}{Total\ number\ of\ nodes\ in\ the\ system}$$

The higher the percentage leads to minimization of makespan[1]. Each computing nodes are modeled as M/M/1/k queue with maximum capacity to have k tasks in the system, so that it can also be a constraint on assignment.

D. Performance Metric

The performance analysis of allocation algorithms are based on three performance metric (i) makespan, (ii) average utilization, and (iii) acceptable queue size. The average utilization for a computing node can be calculated as the ration (makespan/Li). To prevent the nodes from overloading, before the task to be assigned to the node queue, a threshold is used. The percentage of acceptable queue for each node is calculated using formula:

$$\frac{number\ of\ acceptable\ node\ queues}{Total\ number\ of\ nodes\ in\ the\ system}$$

The higher the percentage leads to minimization of makespan [1]. Each computing nodes are modeled as M/M/1/k queue with maximum capacity to have k tasks in the system, so that it can also be a constraint on assignment.

Figure 4.3 shows the makespan=73 for the chromosome in figure 4.2 with corresponding average utilization (AU) of five computing nodes.

Node	A(i)			L_i	AU
1	t(9,1)=13	t(10,1)=2		15	0.2054
2	t(5,2)=12	t(6,2)=31		43	0.5890
3	t(2,3)=5	t(4,3)=26		31	0.4246
4	t(7,4)=14	t(8,4)=4		28	0.3835
5	t(1,5)=15	t(3,5)=58		73	1.0000

Figure 4.3 Makespan of the system

Node	Initial Load	A(i)		L_i	AU
1	9	t(9,1)=13	t(10,1)=2	24	0.3076
2	11	t(5,2)=12	t(6,2)=31	54	0.6923
3	7	t(2,3)=5	t(4,3)=26	38	0.4871
4	15	t(7,4)=14	t(8,4)=4	43	0.5512
5	5	t(1,5)=15	t(3,5)=58	78	1.0000

Figure 4.4 Makespan with initial load

Figure 4.4 shows the makespan=78 for the chromosome in figure 4.2 with corresponding average utilization(AU) of five computing nodes with considering current system load as initial load. The genetic algorithm uses fitness function to evaluate the quality of the task assignment for the chromosome is based on the [38] by Zomaya and The, defined by following equation:

$$fitness = \frac{1}{makespan} \times AU \times \frac{\#\, acceptable\ queues}{\#\, computing\ nodes}$$

Where AU is average utilization

V. Load Balancing Algorithm Using Simulated Annealing

SA is a heuristic method that has been implemented to obtain good solutions of an objective functions defined on a number of discrete optimization problem [16,31].The simulated annealing method mimics the physical process of heating a material and then slowly lowering the temperature(cooling) to decrease defects so as to minimize the system energy[17]. SA is implemented using iterative algorithm that only considers one possible solution for each task window at a time. The solution uses representation as the fixed window size for *k* number of task from the list of *n* tasks. The SA approaches randomly generates initial solution representing an allocation of tasks with a fixed window size. A new solution is generated based upon the neighborhood structure [26]. Temperature is used as a control parameter in SA and decreases gradually with each iteration. This decides the probability of accepting a worst solution at any step and commonly used a stopping criterion. The initial temperature is used as an integer value and decreased by a rate called annealing schedule [1, 26].

At each iteration Scheduling of tasks from a task set to different processors such that the loads of the assigned computing nodes is balanced, is a well-known instance of combinatorial optimization, which is tackled using the SA technique in the following steps. Task schedule (TS) is the linear representation of nodes on which the tasks are to be executed in order. We have use the similar structure as figure 3.1, to represent the task schedule TS = (ts1, ts2$_j$, ts3, …, ts*WIN_SIZE*). With n task to be scheduled on m computing nodes, simulated annealing based algorithm selects asset of *k* tasks from the task pool of *n* tasks, and generated an allocation for those tasks randomly on m machine. In next iteration the new allocated is based upon the move set representation. We are presenting three move sets representations (i) inversion, (ii) translation, and (iii) switching for SA. The details of these algorithms are presented with illustration as follows.

- **Inversion**

In the process of inversion, we select four randomly chosen consecutive nodes and replace it by the reverse order of the same node number. Following figure illustrates the process of 10 tasks on 5 nodes.

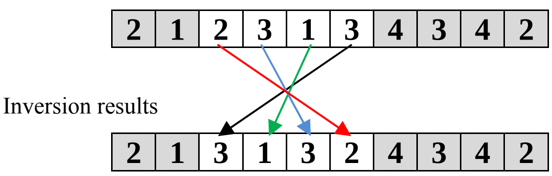

Figure 5.1 Allocation list on inversion

Algorithm* INVERSION *(TS, WIN_SIZE)

Input: TS = (ts1, ts2$_j$, ts3, …, ts10) Task Schedule

WIN_SIZE = Size of the Task Schedule *TS*

Output: TS*=(ts1, ts2$_j$, ts3, …, ts10) Task Schedule

1. Generate a random number S_1 to represent the starting point and another random number L_1 for the length of the substring.
2. Let *SS* = StringReverse (SubString (*TS*, S_1, L_1));
3. For *i* = 1 to *WIN_SIZE* repeat,
   a. if $i < S_1$ or ($i > S_1$ and $i >= S_1 + L_1$),
      S = concat (*S*, *TS* (*i*));
   b. if $i == S_1$, *S* = concat (*S*, *SS*);
   [End of for loop]
4. Return (*TS*);

- **Translation**

***Algorithm* TRANSLATION (*TS, WIN_SIZE*)**

Input: TS = (ts1, ts2$_j$, ts3, …, ts10) Task Schedule

WIN_SIZE = Size of the Task Schedule *TS*

Output: TS*=(ts1, ts2$_j$, ts3, …, ts10) Task Schedule

1. Generate a random number S_1 to represent the starting point and another random number L_1 for the length of the substring.
2. Generate a random number I_1 for the insertion point.
3. Let *SS* = SubString (*TS*, S_1, L_1);
4. For *i* = 1 to *WIN_SIZE* repeat,
   a. if $i <= I_1$ and ($i < S_1$ or ($i > S_1$ and $i >= S_1 + L_1$)), *S*= concat (*S*, *TS* (*i*));
   b. if $i == I_1$, *TS* = concat (*TS*, *SS*);
   c. if $i > I_1$ and ($i < S_1$ or ($i > S_1$ and $i >= S_1 + L_1$)), *S*= concat (*S*, *TS* (*i*));
   [End of for loop]
5. Return (*TS)*;

Translation is transformation functions that remove two or more consecutive nodes from the schedule and place it in between any two randomly selected consecutive nodes.

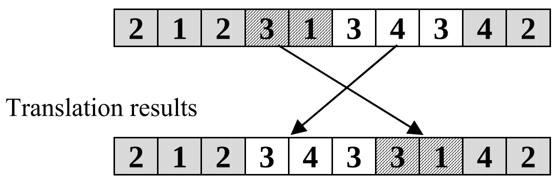

Figure 5.2 Allocation list on translation

- **Switching**

Move set can be constructed for the schedules using a switching function, which randomly select two nodes and switch them in a schedule. Generally speaking, the switching move set tends to rupture the original schedule and results in an allocation that has a makespan significantly different from that of the original allocation. Comparisons between inversion and switching move set can be found in [48]. Example of switching function is shown in figure 5.3.

2	1	2	3	1	3	4	3	4	2

2	1	4	3	1	3	4	3	2	2

Figure 5.3 Outcome of switching operation

***Algorithm* SWITCHING (*TS, WIN_SIZE*)**

Input: TS = (ts1, ts2$_j$, ts3, …, ts10) Task Schedule

WIN_SIZE = Size of the Task Schedule *TS*

Output: TS^*=(ts1, ts2$_j$, ts3, …, ts10) Task Schedule

1. Generate a random number *i* to represent the task 1 and another random number *j* to represent task 2.
2. swap (*TS*(*i*), *TS*(*j*));
3. Return (*TS*);

In our model, simulated annealing algorithm starts with generating initial schedule *TS* randomly for 10 tasks. Following that move set is created for an initial schedule, by any one of the three different methods *(i) Inversion* , *(ii) Translation* and *(iii) Switching* by selecting a random number between 1 to 3. A final allocation list for the tasks is obtained after 25 iteration. Tasks are allocated to the nodes and average utilization is calculated for those 10 tasks before selecting a next 10 tasks from the set of waiting tasks. The simulated annealing for dynamic load balancing outlined in for of algorithm ***SA_DLB.*** The algorithm ***SA_DLB*** called for maximum (n/ ***WIN_SIZE)*** times to allocate n tasks to the computing nodes.

***Algorithm* SA_DLB (*TS, WIN_SIZE*)**

Input: TS = (ts1, ts2$_j$, ts3, …, ts10) Task Schedule

WIN_SIZE = Size of the Task Schedule *TS*

Output: TS^*= (ts1, ts2$_j$, ts3, …, ts10) and AU($\boldsymbol{TS^*}$)

1. Calculate makespan for TS = ***ms***
2. For *i* = 1 to *25* repeat,
   a. Generate a random integer m from {1,2,3}
   b. if m = 1, call *INVERSION* **(*TS, WIN_SIZE*)** to create move set
   c. if m = 2, call *TRANSLATION* **(*TS, WIN_SIZE*)** to create move set
   d. if m = 3, call *SWITCHING* **(*TS, WIN_SIZE*)** to create move set
   e. calculate the makespan for the new move set $\boldsymbol{TS^*}$ as $\boldsymbol{ms^*}$
   f. if $\boldsymbol{ms^*} < \boldsymbol{ms}$ then $\boldsymbol{TS = TS^*}$
   [End of for loop]
3. Allocate the tasks to Nodes using ***TS*** and calculate average utilization(AU)
4. Return ($\boldsymbol{TS^*}$, ***AU***);

Common approaches used as the stopping criteria in simulated annealing algorithm (SA) are, (i) one may use a given number of iteration, or (ii) a time limit, or (iii) a given number of iteration without an improvement of the objective function value, (iv) value of the objective function limit as set by the user[25, 26]. We have used a fixed number of iteration proportional to number of task to be schedule on computing nodes. We have use Matlab to design our simulation programs. The experiment was conducted with n=1000 tasks on m=60 computing nodes. The simulation results are compared with two heuristic algorithms: *first fit* and *randomized* [17, 40].

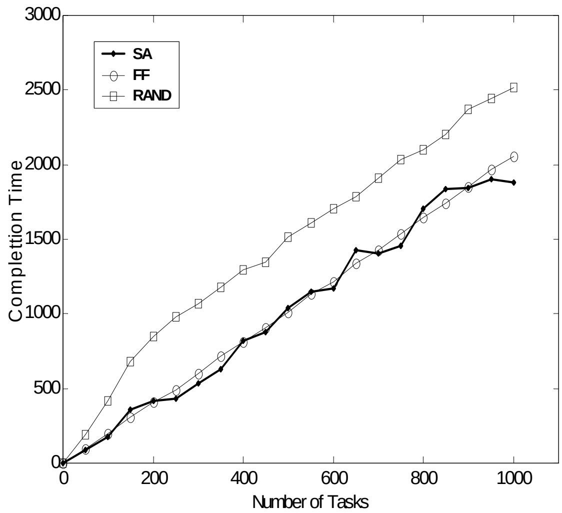

Figure 5.4 Completion Time of 1000 tasks on

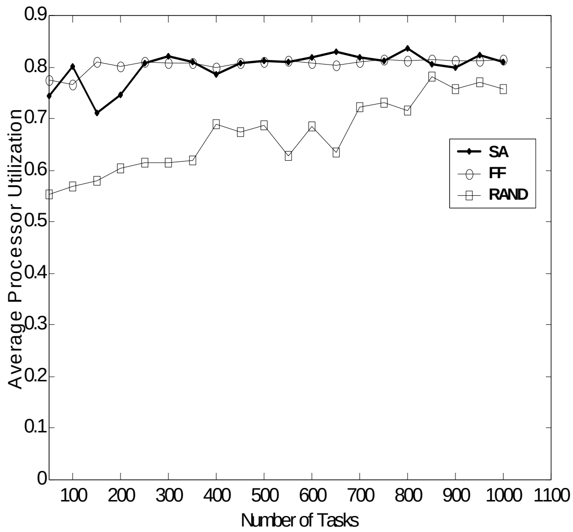

Figure 5.5 Average Processor Utilization

Randomized algorithms are known for efficient approxima te solutions to intractable problems with better complexity bounds. Moreover randomized algorithm is selected for performance comparison as it is simple to describe and implement than the deterministic algorithm. We executed several simulations on proposed simulated annealing algorithm for dynamic load balancing on HDCS, to compare with conventional first fit (FF), and randomized algorithm. The simulation results are presented in figure 5.4 and 5.5 with completion time and processor utilization respectively.

The Fast come first serve (FF) and randomized algorithms for resource allocation can make an instantaneous decision to allocation of the task to computing nodes, which results a shorter makespan. The SA-based load balancing algorithm shows very much similar performance to that of FF in both average *processor utilization* and completion time or *makespan*.

VI. CONCLUSION AND FUTURE WORK

Load balancing is being performed during runtime at various stages to keep the workload balance on different computing nodes of a HDCS. This paper presents in details, a SA based load balancing algorithm for HDCS with three algorithms to compute move set. We have proposed a coding scheme to represent the task assigned for execution to different computing node. We have simulated the behavior of different load balancing algorithm with our simulator developed using Matlab, where each task t_i is with the expected execution time t_{ij} on machine M_j. The results of the simulation with scalability of tasks are presented for conventional first fit (FF), randomized, and SA algorithm. This paper uses consistent ETC matrix to design load balancing algorithms, however further investigations may be carried out to design SA based load balancing algorithms for inconsistent and partially–consistent ETC matrix for tasks. Genetic algorithm have been proposed over the years for solving static and dynamic load balancing problems on distributed system. The coding method introduced in this paper can be used to design a genetic algorithm for dynamic load balancing in HDCS.

REFERENCES

[1] Fidanova, Stefka. "Simulated annealing for grid scheduling problem." In *Modern Computing, 2006. JVA'06. IEEE John Vincent Atanasoff 2006 International Symposium on*, pp. 41-45. IEEE, 2006.

[2] Wu, Min-You, Wei Shu, and Hong Zhang. "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems." In *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*, pp. 375-385. IEEE, 2000.

[3] Clawson, James. "Distributed computing architecture." U.S. Patent 6,112,304, issued August 29, 2000.

[4] Joanna Kolodziej , and Semee Ullah Khan,"Multi-level hierarchic genetic-based scheduling of independent job in dynamic heterogeneous grid environment", Information Science, vol. 214, pp.1-19, 2012.

[5] R. Thamilselvan, and P. Balasubramanie, "Analysis of various alternate crossover strategies for genetic algorithm to solve job shop scheduling problem", European Journal of Scientific Research, Vol. 64, No.4, 2011, pp.538-554.

[6] Onno Boxma, Ger Koole, and Zhen Liu,"Queueing-theoretic solution methods for models of parallel and distributed systems", Centrum voor Wiskunde en Informatica, Department of Operations Research, Statistics, and System Theory, 1994.

[7] Francois Spies, "Modeling of optimal load balancing strategy using queuing theory" Micro processing and Microprogramming, vol.41, 1996, pp.555-570.

[8] Caffrey, James, and Graham Hitchings. "Makespan distributions in flow shop scheduling." International Journal of Operations & Production Management 15.3 (1995): 50-58.

[9] R. Rindzevicius, D. Poškaitis, and B. Dekeris. "Performance measures analysis of M/M/m/K/N systems with finite customer population." Electr. Electr. Eng 3.67 (2006): 65-70.

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

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

[12] Jong-Chen Chen, Guo-Xun Liao, Jr-Sung Hsie, Cheng-Hua Liao: A study of the contribution made by evolutionary learning on dynamic load-balancing problems in distributed computing systems. Expert Syst. Appl. 34(1): 357-365 (2008)

[13] Sukumar Ghosh,. *Distributed systems: an algorithmic approach*. Vol. 13. Chapman & Hall/CRC, 2006.

[14] M. Nikravan, and M. H. Kashani. "A genetic algorithm for process scheduling in distributed operating systems considering load balancing." In *Proceedings 21st European Conference on Modelling and Simulation Ivan Zelinka, Zuzana Oplatkova, Alessandra Orsoni, ECMS*. 2007.

[15] A.J. Page, T.M. Keane, and T.J. Naughton. Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system. *Journal of parallel and distributed computing*, 70(7):758–766, 2010.

[16] Mitchell D. Theys, et al. "Mapping tasks onto distributed heterogeneous computing systems using a genetic algorithm approach." *Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Sciences* (2001): 135-178.

[17] Tracy D. Braun, Howard Jay Siegel, Noah Beck, Ladislau L. Bölöni, Muthucumaru Maheswaran, Albert I. Reuther, James P. Robertson et al. "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems." *Journal of Parallel and Distributed computing* 61, no. 6 (2001): 810-837.

[18] Tracy D. Braun, Howard Jay Siegel, and Anthony A. Maciejewski. "Static mapping heuristics for tasks with dependencies, priorities, deadlines, and multiple versions in heterogeneous environments." In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pp. 78-85. IEEE, 2002.

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

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

[21] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, 1979.

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

[23] S. Ali; H.J. Siegel, M. Maheswaran, D. Hensgen; , "Task execution time modeling for heterogeneous computing systems ," *Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings. 9th* , vol., no., pp.185-199, 2000.

[24] 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.

[25] Genetic Algorithm and Direct Search Toolbox 2, User Guide, The MathWorks, Inc., US, 2009.

[26] Kalyanmoy Deb, Optimization for Engineering Design: Algorithm and Examples, Prentice hall of India, 1998.

[27] Gamal Attiya, and Yskandar Hamam, "Task allocation for maximizing reliability of distributed system: A simulated annealing approach", Journal of parallel and Distributed Computing, vol.66, pp. 1259-1266, 2006

[28] Cheng-Zhong Xu, and Francis CM Lau. "Iterative dynamic load balancing in multicomputers." *Journal of the Operational Research Society* (1994): 786-796.

[29] Gamal Attiya, and Yskandar Hamam. "Two phase algorithm for load balancing in heterogeneous distributed systems." In *Parallel, Distributed and Network-Based Processing, 2004. Proceedings. 12th Euromicro Conference on*, pp. 434-439. IEEE, 2004.

[30] Talbi, E-G., and Traian Muntean. "Hill-climbing, simulated annealing and genetic algorithms: a comparative study and application to the mapping problem." In *System Sciences, 1993, Proceeding of the Twenty-Sixth Hawaii International Conference on*, vol. 2, pp. 565-573. IEEE, 1993.

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

[32] K. S. Trivedi, Probability and statistics with reliability, queuing and computer science applications, Prentice Hall of India, 2001.

[33] Thomas V. Christensen, "Heuristic Algorithms for NP-Complete Problems" Project report, Institute of Informatics and mathematical Modeling, Technical University of Denmark.2007.

[34] Daniel Grosu, and Anthony T. Chronopoulos, "Algorithmic Mechanisim Design for load balancing in Distributed system", IEEE transaction on system man and cybernetics-Part B: cybernetics, vol.34, No.1, PP. 77-84, 2004.

[35] Suman, Balram, and Prabhat Kumar. "A survey of simulated annealing as a tool for single and multiobjective optimization." *Journal of the operational research society* , Vol.57, No. 10, pp.1143-1160, 2005.

[36] R. H. J. M. Otten and L. P. P. P. van Ginneken. The Annealing Algorithm, Kluwer Academic Publishers, 1989.

[37] K K. Hwang, G.C. Fox, and JJ Dongarra., "Distributed and Cloud Computing: From Parallel Processing to the Internet of Things", Morgan Kaufmann, 2012.

[38] A.Y. Zomaya and Y.H. Teh, "Observations on using genetic algorithms for dynamic load-balancing", *IEEE Trans. on Parallel and Dist. Systems*, pp.899-911, September 2001.

[39] S. Kirkpatrick, C.D. Gelatt Jg., M. P. Vecchi, "Optimization by Simulated Annealing", Science 220, pp.671-680, 1983.

[40] Bibhudatta Sahoo, Dillip Kumar and Sanjay Kumar Jena, "Analysing the Impact of Heterogeneity with Greedy Resource Allocation Algorithms for Dynamic Load Balancing in Heterogeneous Distributed Computing System", International Journal of Computer Applications, Vol. 62, No.19, pp.25-34, January 2013. Published by Foundation of Computer Science, New York, USA

[41] Wen-Chiung Lee, Chin-China Wu, and Peter Chen, "A Simulated annealing approach to makespan minimization on identical parallel machines", International Journal of Advanced Manufacturing Technology, Vol.31, pp.328-334, 2006.

[42] S. Kirkpatrick, "Optimization by Simulated Annealing: Quantitative Studies", Journal of Statistical Physics, Vol.34, No.5/6, pp.975-986, 1984.

[43] Amir Masoud Rahmani and Mojtaba Rezvani, "A Novel Genetic Algorithm for Static Task Scheduling in Distributed Systems", International Journal of Compute Theory and Engineering, Vol.1, No1, April 2009.

[44] R. L. Haupt, and S.E. Haupt, "Practical genetic algorithm", John Willy and Sons, 2004.

[45] Helen D. Karatza Ralph C. Hilzer "Load Sharing in Heterogeneous Distributed Systems" Proceeding of the winter simulation conference, 2002, pp.489-496.

Bibhudatta Sahoo has 20years of Teaching Experience in undergraduate and graduate level in the field of computer Science & Engineering. He is presently Assistant Professor in the Department of Computer Science & Engineering, NIT Rourkela, INDIA. His technical interests include performance evaluation methods and modeling techniques distributed computing system, networking algorithms, scheduling theory, cluster computing and web engineering.

Sanjay Kumar jena is Professor in the department of Computer Science and Engineering NIT Rourkela. His areas of research interest include Database Engineering, Parallel Algorithm, Artificial Intelligence & Neural Computing, Computational Machines, Network security. He is the principal investigator of Information Security Education & Awareness Project for MIT, Government of India at NIT Rourkela.

Sudipta Mahapatra, obtained his M.tech. and Ph.D. degree in Computer Engineering from IIT, Kharagpur. He was in the Electronic Systems Design Group of Loughborough University, UK as BOYSCAST Fellow of DST Government of India, from March 1999 to March 2000. He is presently working as an Associate Professor in the Electronics & Electrical Communication Engineering department of IIT Kharagpur. His areas of research interest include Image and Video Compression, Optical and Wireless Networks, Parallel and Distributed Systems.