IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-6, June 2018 E-ISSN: 2347-2693

Fuzzy Min – Max Scheduling (FMiMaS) for Computational Grids

A. Kumar[1*], H. Pathak[2]

[1] Department of Computer Science, Gurukul Kangri Vishwavidyalaya, Haridwar (UK), India
[2]Department of Computer Science, Gurukul Kangri Vishwavidyalaya, Haridwar (UK), India

Available online at: www.ijcseonline.org

Accepted: 07/Jun/2018, Published: 30/Jun/2018

Abstract -With the fast expansion in wide area networks leading to availability of low cost fundamental computational resources, the popularity of computational grids has increased. Effective load balancing and scheduling are the key concerns for meeting QoS requirements of users for computational grids. Fuzzy Logic contributes to handle the uncertainties involved in processors' load and tasks' execution length during scheduling decisions to ensure a better load balancing in distributed systems. In an effort to enhance the previously proposed and implemented dynamic load balancing algorithms for hierarchical and distributed computational grids viz. DLBCGBH – H / D, Fuzzy based Min – Max Scheduling (FMiMaS) is proposed in this paper which when integrated with the Local scheduling proposed in DLBCGBH – H / D, devises its enhanced version viz. 'Hierarchical with Fuzzy' & 'Distributed with Fuzzy' approaches based on Hybrid Scheduling. It is implemented using GridSim 4.0 and the comparison of simulation results with DLBCGBH – H / D and Built-in Space Shared utility of GridSim 4.0 demonstrate tremendous improvements in terms of the performance metrics viz. Average Consumed Time, Average Processing Cost and Average Waiting Time.

Keywords**:** Computational Grid, Distributed, Hierarchal, Fuzzy Logic, Space Shared, Load Balancing, Scheduling, Binary Heaps, Fuzzy Min – Max Scheduling, Hybrid Scheduling, FMiMaS

1. INTRODUCTION

Several factors like resource sharing, scalability, etc. have taken computation to the era of Grid computing that permits desktop computers to take part in the activity of a global network when they are idle and thus enable large software systems to utilize extra hardware resources. When all the resources of inactive computer systems are gathered as a powerful computer system, a highly effective grid system arises [1]. Internet has become a boon to Grid computing by enabling to employ hardware resources that belong to numerous other systems. Rapid growth of computer usage has given rise to applications that uses the shared hardware and software resources (eg. memory, processor, files etc.) and ultimately increased the number of submitted jobs across the internet [2].

Grid computing is a system that permits us to link network resources and applications to make a large effective system that has the capability to do extremely complex jobs which a solitary personal computer could not accomplish. Grid systems allow the virtualization of a vast array of resources, in spite of their considerable heterogeneity [3]. In a real world scenario the volatile job arrival patterns and the unpredictable / asymmetrical computing capabilities cause a particular grid site to become overloaded while other grid sites may be under-loaded [4] which adversely affect the performance of grid system. Therefore, the heterogeneous and the dynamic environment of grids need load balancing in order to ensure the best usage of the performance of the grid nodes. Load balancing techniques play a vital role in determining the performance of the grid system by improving grid node utilization and reducing the usage of time. Hence, load balancing techniques should be "fair" in distributing the load across the grid nodes. The meaning of "fair" is that the difference between the "heaviest loaded" node and "lightest loaded" node should be minimized [5]. The factors that pose challenges for load balancing in Grid Systems include heterogeneity, autonomy, application diversity, dynamicity, adaptability, scalability, resource non-dedication, resource selection and computation – data separation [6].

The jobs that arrive from grid users are first placed in a job queue. Thereafter, as per the requirement of the load balancing schemes the job scheduler schedules the jobs from job queue to the dispatch queues of the appropriate computing node. The computing node executes the jobs and sends the computational results back to the associated grid user. Several parameters are used to measure the performance of the scheduling algorithm in grids such as resource utilization, response time, throughput, waiting time, reliability, communication overheads, processing cost etc. [7].

An effective scheduling approach considerably contributes to load balancing by deciding the load balanced schedule. As the information about the state of nodes during scheduling, exchanged at discrete intervals through message passing mechanism, reaches their destinations at variable latencies and hence becomes inevitably out of date to effectively estimate the global system state due to the rapid changes in the states of nodes / clusters. Moreover, many other factors lead to ambiguous information about the states of the nodes / clusters which in turn lead to uncertainty in scheduling decisions thereby causing load imbalance. In view of the aforesaid unavoidable uncertainties involved in grid environment, the Fuzzy Logic can be a suitable method as it deals with uncertain information [8 - 11].

Rest of the paper is organized as follows: Section 2 briefly presents the literature survey from the related domain. Section 3 explains the proposed work. Section 4 introduces with simulation environment, simulation scenarios and the simulation experiments carried out to analyze the performance against the relevant metrics. Section 5 presents the results of simulation experiments along with the performance analysis. Finally, the paper is concluded in Section 6 along with the future directions for research.

2. RELATED WORK

This section presents a brief overview of the various related researches that guided the efforts during the proposed work.

A hierarchical load balancing technique called PLBA is proposed in [12] which divides the load into different categories like lightly loaded, under-lightly loaded, overloaded, normally loaded based on variable threshold values. A threshold value, which can be found out using load deviation, is responsible for transfer the task and flow of workload information.

A dynamic load balancing algorithm with fuzzy logic is proposed in [13] which handle the issues of uncertainty and inconsistency faced by the previous algorithms. The algorithm reveals better response time than round robin and randomize algorithm respectively by 30.84% and 45.45%. Another fuzzy dynamic load balancing algorithm for homogenous distributed systems is proposed in [14] wherein inaccurate load information is dealt with the use of fuzzy logic and accordingly the load distribution decisions are made to maintain the overall stability of the system.

The design and implementation of a proposed fuzzy-logic-based scheme for dynamic load balancing in grid computing services is done in [15] by using a fuzzy logic inference system which uses some metrics to capture the variability of loads and specifies the state of each node of a cluster. Then, based on the overall nodes' states, the state of the corresponding cluster is defined in order to assign the newly arrived tasks such that load balancing among different clusters and nodes is accomplished. Being inspired from this approach, its enhanced version has also been proposed by the authors in [16] for dynamic load balancing in hierarchical computational grid using fuzzy logic.

Another fuzzy-logic-based self-adaptive job replication scheduling (FSARS) algorithm which considers the trust relationships between the participants is proposed in [17]. FSARS uses the security demand of the task and Trust level of the resources as the main parameters. The proposed method gives a vigorous performance against resource failures and improved scheduling achievement rate.

A fuzzy expert system for load balancing in a symmetric multiprocessor environment is proposed in [18] wherein an On-Demand based load balancing instead of the periodic load balancing is implemented to ensure fast and fair load balancing with minimal computational overhead. The time spent by a job in the system is considered as the main issue that needs to be minimized in the approach proposed in [19].

Min-Min heuristic minimizes makespan than the other heuristics but it fails to produce a load balanced schedule. A Load Balanced Min-Min (LBMM) algorithm that reduces the makespan and increases the resource utilization is proposed in [20] wherein the traditional Min-Min algorithm is executed in the first phase followed by the rescheduling of tasks to effectively use the unutilized resources in the second phase. Further, another scheduling algorithm based on Min–Min heuristic is proposed in [21] which first estimates the completion time of the tasks on each of resources and then selects the appropriate resource for scheduling. Further, RASA proposed in [22] alternatively applies Max-Min and Min-Min over scheduling process iterations so as to ensure effective utilization of resources leading to load balanced schedule.

3. PROPOSED SCHEDULING MODEL

Scheduling and Load Balancing in distributed system are closely related to each other. While the prior one is responsible for deciding the execution order of the tasks, the later one is responsible for ensuring that all the processing elements of distributed system are fairly loaded. Load balanced task scheduling is very important problem in complex grid environment. So task scheduling which is one of the NP-Complete problems becomes a focus of research in grid computing area. Efficiency of scheduling algorithms affects the user and service provider. Effectiveness of a scheduling algorithm is measured using response time, resource utilization, migrated task, cost, deadline and waiting time.

A Grid scheduling algorithm works at two levels - local scheduling and global scheduling. Local scheduling algorithms manage the nodes within site and improve the system performance, while global scheduling algorithms select the site and improve the task migration and cost. They are of much relevance in these days because user has to pay-

per-use. A hybrid scheduling policy is also recommended to be devised so as to club the positives of both the local & global scheduling policies. Further, existing research in using fuzzy logic for the purpose of scheduling and load balancing has only concentrated on utilizing fuzzy logic concepts in describing processors' load and tasks' execution length.

In view of the aforesaid facts, a Fuzzy Min – Max Scheduling (FMiMaS) model is proposed in Section 3.1 as a Global Scheduling policy which when integrated with the Local Scheduling proposed in DLBCGBH – H / D [23, 24], as proposed in Section 3.2, provides an efficient solution to load balancing problem in computational grids.

3.1 Fuzzy Min - Max Scheduling

To efficiently handle dynamic task arrivals, a fuzzy logic based min – max scheduling using binary heaps is proposed herein. The schematic of the proposed approach is shown in Figure-3.1. As shown in the figure, let there are n number of users who want to submit their jobs to the grid infrastructure for processing. Each of these n job lists contains a different number of jobs with different length of processes (i.e. Job Length) to be executed. The proposed model suggest to aggregate all these jobs at the Job Queue to apply a pre-processing technique named "Fuzzy Job Categorization" after which these aggregated jobs are categorized in to five different lists maintained in binary min and max heaps as explained in the description of the algorithm. Thereafter, a post-processing technique named "Job Lists Regeneration" is applied to create the requisite number of Dispatch Queues in view of the application specific requirements (or the number of users connected through the grid) by following a Min-Max strategy. The working of the two main algorithms involved in the proposed fuzzy min – max scheduling are described as follows:

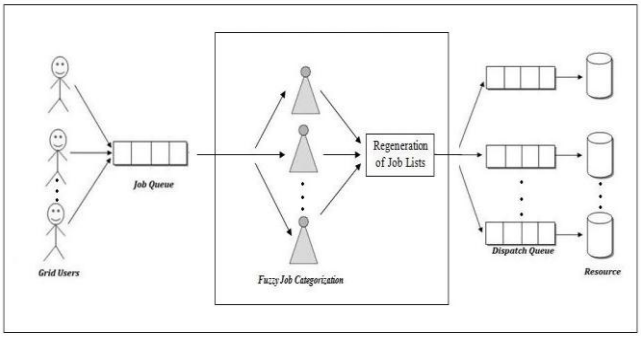

Figure-3.1: Schematic for Fuzzy Min - Max Scheduling

3.1.1 Fuzzy Job Categorization Algorithm

The working of this algorithm involves the application of Fuzzy Membership Function named "Job Category" for fuzzification of incoming jobs. The input variables to this Fuzzy membership function is "Resource Requirements of the job (i.e. Job Length)" and as output variable it produces the "Job Category" viz. "low", "low average", "average", "high average", and "high. Fuzzy membership function for the Output Variable "Job Category" is shown in Figure-3.2. Assuming that x represents the "Resource Requirements of Job (i.e. Job Length)", the formula for calculating the membership values for the fuzzy output variable "Job Category" is shown in Table-3.1. Thereafter, Binary Min-Heaps are created for "low", "low-average" and "average" categories of jobs viz. BH_{low},$BH_{low\text{-}average}$ & $BH_{average}$ while Binary Max-Heaps are created for the "high average" and "high" categories of jobs viz. $BH_{high\text{-}average}$ &BH_{high}. While creating & maintaining the Binary Heaps "Resource Requirements of the Job (i.e. Job Length)" is considered as the key value. The idea behind using the Binary Heaps is that they can efficiently accommodate the dynamically arriving tasks by efficiently reorganizing themselves so as to ensure that the heaviest and lightest jobs at any point of time are directly accessible through the root of the heap itself. The pseudo code for Fuzzy Job Categorization Algorithm is shown in Figure-3.3.

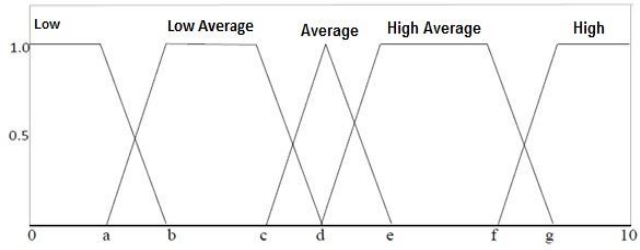

Figure-3.2: Fuzzy Membership Function for "Job Category"

Table-3.1: Fuzzy Membership Values for "Job Category"

$\mu_{low}(x)$	=	1	$x \le a$
		$(b-x)/(b-a)$	$a < x < b$
		0	$x \ge b$
$\mu_{low\ average}(x)$	=	0	$x \le a$
		$(x-a)/(b-a)$	$a < x < b$
		1	$b \le x < c$
		$(d-x)/(d-c)$	$c \le x < d$
		0	$x \ge d$
$\mu_{average}(x)$	=	0	$x \le c$
		$(x-c)/(d-c)$	$c < x < d$
		$(e-x)/(e-d)$	$d \le x < e$
		0	$x \ge e$
$\mu_{high\ average}(x)$	=	0	$x \le d$
		$(x-d)/(e-d)$	$d < x < e$
		1	$e \le x < f$
		$(g-x)/(g-f)$	$f \le x < g$
		0	$x \ge g$
$\mu_{high}(x)$	=	0	$x \le f$
		$(x-f)/(g-f)$	$f < x < g$
		1	$x \ge g$

3.1.2 Job Lists Regeneration Algorithm

As shown in the schematic for the Fuzzy Min - Max Scheduling in Figure-3.1, after categorization of jobs into five different categories, this algorithm restructures all the jobs into number of lists equivalent to the number of users connected through the grid. This number can also depend on the specific requirements of the grid application. Job Lists Regeneration Algorithm works on Min-Max strategy so as to ensure an effective scheduling wherein the short tasks are not suffered by the presence of long tasks while ensuring that long tasks are also processed simultaneously. This strategy ensures better load balancing by ensuring the effective utilization of resources .The pseudo code for the Job Lists Regeneration Algorithm is shown in Figure-3.4.

3.2 Enhancing DLBCGBH – H / D using Fuzzy Min – Max Scheduling (FMiMaS)

As the effective load balancing policy is always complemented by an efficient scheduling strategy. The Global Scheduling Policy FMiMaS, proposed in the previous section, is proposed to be integrated with the previously proposed approaches DLBCGBH – H / D to enhance the same through Hybrid Scheduling introduced by the integration of FMiMaS with DLBCGBH – H / D, thereby resulting into 'Hierarchical with Fuzzy' and 'Distributed with Fuzzy' approaches for dynamic load balancing in Computational Grids. The enhanced approaches are further implemented for studying through simulation.

Input: Number of users N, Users' job list $J_m = \{J_1, J_2, J_3, \dots, J_n\}$
Output: Five Lists of Jobs $J_l, J_{la}, J_a, J_{ha}, and\ J_h$
Process:

1. $max = findMax(J_m)$
2. $min = findMin(J_m)$
3. $for(i = 1; i \leq m; i++)$
   a. $if(j_i \geq 0\ \&\&\ J_i \leq max * 0.2)$
      i. $J_l = J_l.Add(J_i)$
   b. $else\ if\ (J_i > max * 0.2\ \&\&\ J_i \leq max * 0.4)$
      i. $J_{la} = J_{la}.Add(J_i)$
   c. $else\ if(J_i > max * 0.4\ \&\&\ J_i \leq max * 0.6)$
      i. $J_a = J_a.Add(J_i)$
   d. $else\ if(J_i > max * 0.6\ \&\&\ J_i \leq max * 0.8)$
      i. $J_{ha} = J_{ha}.Add(J_i)$
   e. $else\ if(J_i > max * 0.8)$
      i. $J_h = J_h.Add(J_i)$
   f. $end\ if$
4. End for

Figure-3.3: Fuzzy Job Categorization Algorithm

4. SIMULATION & PARAMETERS

The performance of the enhanced versions of DLBCGBH – H / D, called as 'Hierarchical with Fuzzy' and 'Distributed with Fuzzy' approaches, are analyzed through simulations using an application developed in Java using GridSim4.0, NetBeans IDE 8.1 and Apache Derby RDBMS. Different performance metrics used for performance comparisons are defined in this section:

A. Average Consumed Time

The average time consumed is the total amount of time required to process the entire request by the grid resources.

N : Total number of gridlets to be processed

CPU_i : Time for processing i^{th}gridlet

$$\text{Average Consumed Time} = \frac{1}{N}\sum_{i=1}^{N}(CPU_i + WT_i)$$

B. Average Processing Cost

The amount of mean cost required to execute all the submitted tasks is termed as the Average processing cost.

$cost_i$: Cost for processing i^{th}gridlet.

$$\text{AverageProcessingCost} = \frac{1}{N}\sum_{i=1}^{N} cost_i$$

C. Average Waiting Time

The time consumed before assigning a task to a server resource is termed as the waiting time. The Average Waiting Time is the mean time consumed for allocating the resource.

WT_i : Waiting Time for i^{th}gridlet

$$\text{AverageWaitingTime} = \frac{1}{N}\sum_{i=1}^{N} WT_i$$

Input: Categorized job list $L_{low}, L_{lowAvg}, L_{avg}, L_{highAvg}, L_{high}$
Output: Number of job list to process N
Process:

1. $workCount = L_{low} + L_{lowAvg} + L_{avg} + L_{highAvg} + L_{high}$
2. $LoopCount = \frac{workCount}{2}$
3. Initialize N list $L_n = \{L_1, L_2, \dots, L_n\}$
4. Temp=1;
5. Flag=0;
6. $for(i = 1; i \leq LoopCount; i++)$
   a. $jobList.Add(L_{low}^{i})$
   b. $jobList.Add(L_{high}^{length-temp})$
   c. $temp = temp + 1$
   d. $if(L_{low}.length == 0)$
      i. $L_{low} = L_{lowAvg}$
   e. $else\ if\ (L_{lowAvg}.length == 0)$
      i. $L_{low} = L_{Avg}$
      ii. $flag = 1$
   f. End if
   g. $if(L_{high}.length == 0)$
      i. $L_{high} = L_{highAvg}$
      ii. $temp = 1$
   h. $else\ if(L_{highAvg}.length == 0\ \&\&\ flag == 0)$
      i. $L_{high} = L_{Avg}$
   i. End if
7. $end\ for$
8. Boolean $f_2, \dots, f_n = false$
9. Boolean $f_1 = true$
10. $for(j = 1; j \leq jobList.length; j++)$
    a. $if(f_1 = true)$
       i. $L_1 = jobList_j$
       ii. $f_1 = false$
       iii. $f_2 = true$
       iv. Break;
    b. $if(f_2 = true)$
       i. $L_2 = jobList_j$
       ii. $f_2 = false$
       iii. $f_3 = true$
       iv. Break
    c. ….
       i. …..
       ii. ….
    d. $if(f_N = true)$
       i. $L_n = jobList_j$
       ii. $f_n = false$
       iii. $f_1 = true$
    e. End if
11. $end\ for$

Figure-3.4: Job Lists Regeneration Algorithm

5. PERFORMANCE ANALYSIS

A comparison of the performance of Fuzzy Hierarchical and Fuzzy Distributed approaches with the previously implemented algorithms DLBCGBH – H / D and Built-in Space Shared Utility of GridSim4.0 is done under this study. The datasets (i.e. real workload traces) for simulation experiments are downloaded from Parallel Workload Archive [25] and partitioned for experimental purposes.

In order to perform an in-depth study on the algorithms through the developed simulation software, their performance trend is analyzed for homogenous computing infrastructure by varying the size of datasets as well as the grid infrastructure viz. Number of Grid Resources, Number of Machines (M/c), and Number of Processing Elements (PE). Performances are recorded under all the three scenarios for the two enhanced approaches viz. Fuzzy Hierarchical and Fuzzy Distributed along with the previously proposed algorithms viz. DLBCGBH – H / D and Built-in Space Shared utility of GridSim 4.0 for three parameters in tabular form and shown in Appendix – A.

Results show the tremendous improvements are reflected by the integration of FMiMaS with DLBCGBH – H / D. However, these results are not included in graphs for the ease of representation. Graphs for different simulation scenarios are shown in the Graph Tables - 1, 2 & 3. For all the cases, Fuzzy Hierarchical approach is found to be comparatively expensive with respect to the Fuzzy Distributed approach.

Table 1: Graphs for Performance Analysis-Scenario-1

Simulation Scenario – 1			
Size of datasets	=	250-2500	Step 250 / 500
No. of Grid Resources	=	3	Fixed
Number of PE	=	1	Fixed
Number of Machines	=	1	Fixed

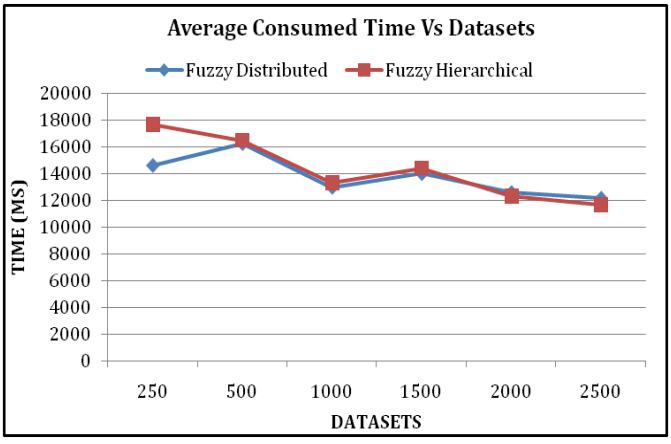

Fig-5.1.1: Datasets Vs Average Consumed Time

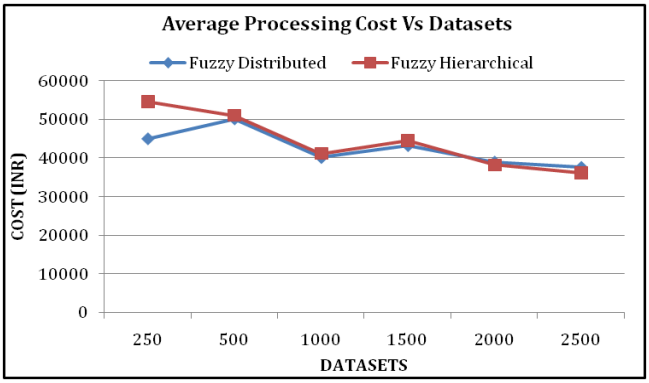

Fig-5.1.2: Datasets Vs Average Processing Cost

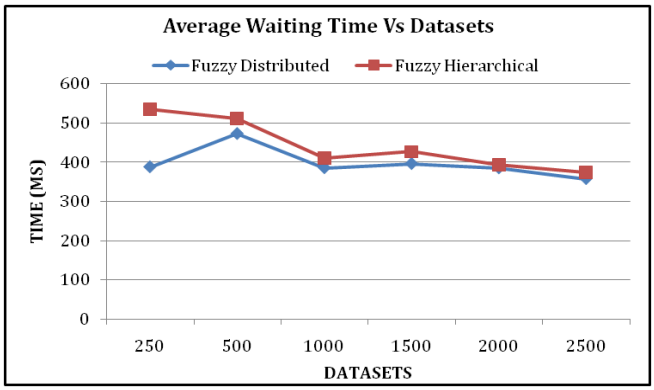

Fig-5.1.3: Datasets Vs Average Waiting Time

Table 2: Graphs for Performance Analysis-Scenario-2

Simulation Scenario - 2			
Size of datasets	=	250	Fixed
No. of Grid Resources	=	5 - 55	Step 10
Number of PE	=	10 - 60	Step 10
Number of Machines	=	15 - 65	Step 10

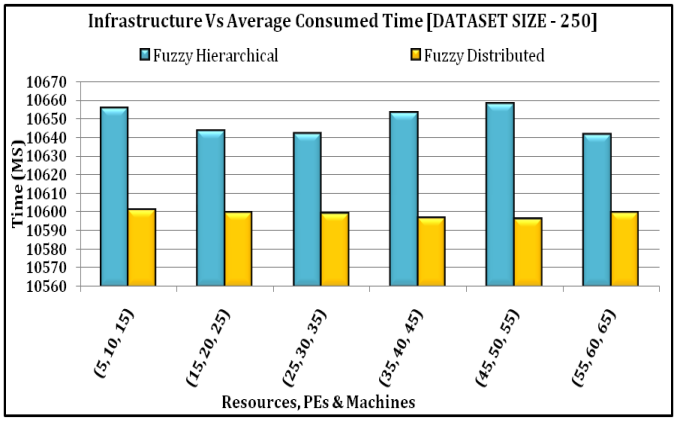

Fig-5.2.1: Infrastructure Vs Average Consumed Time

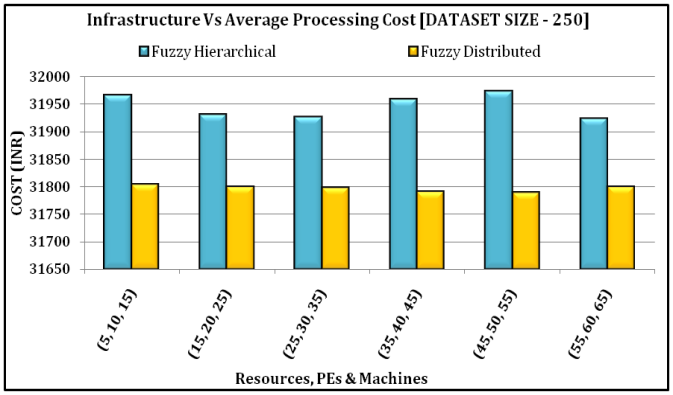

Fig-5.2.2: Infrastructure Vs Average Processing Cost

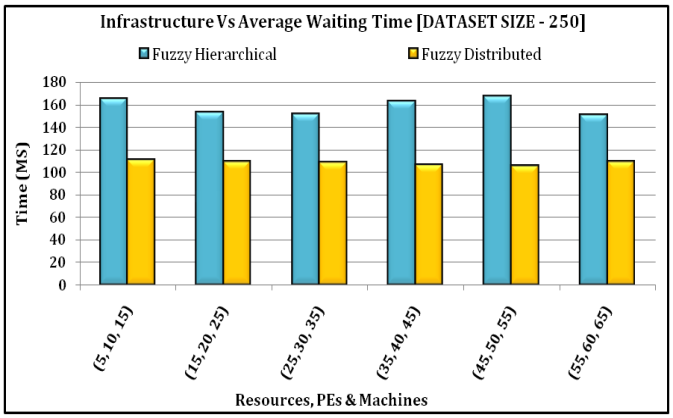

Fig-5.2.3: Infrastructure Vs Average Waiting Time

Table 3: Graphs for Performance Analysis-Scenario-3

Simulation Scenario - 3			
Size of datasets	=	2500	Fixed
No. of Grid Resources	=	5 - 55	Step 10
Number of PE	=	10 - 60	Step 10
Number of Machines	=	15 - 65	Step 10

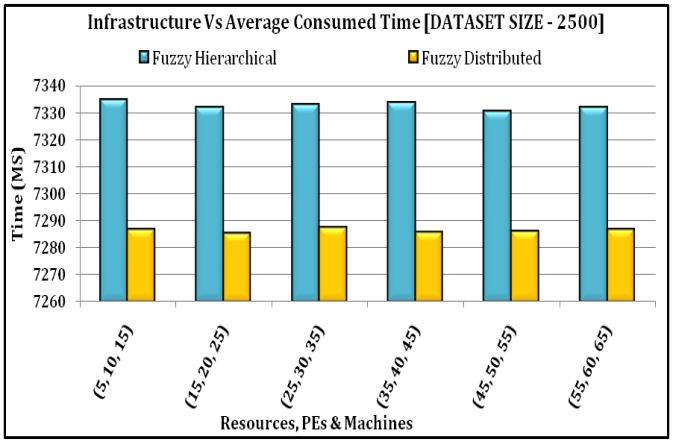

Fig-5.3.1: Infrastructure Vs Average Consumed Time

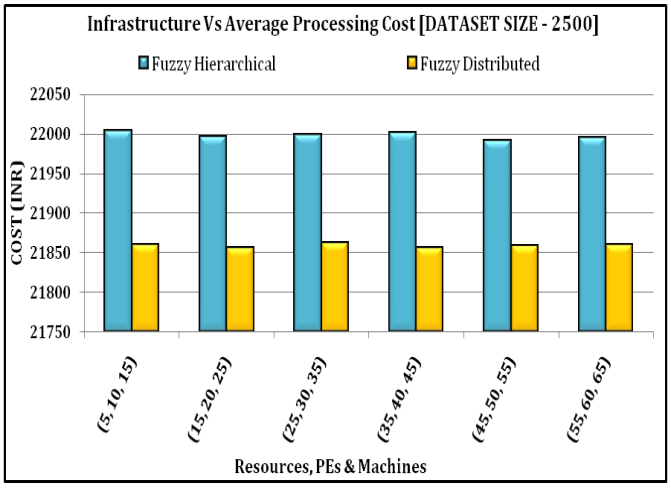

Fig-5.3.2: Infrastructure Vs Average Processing Cost

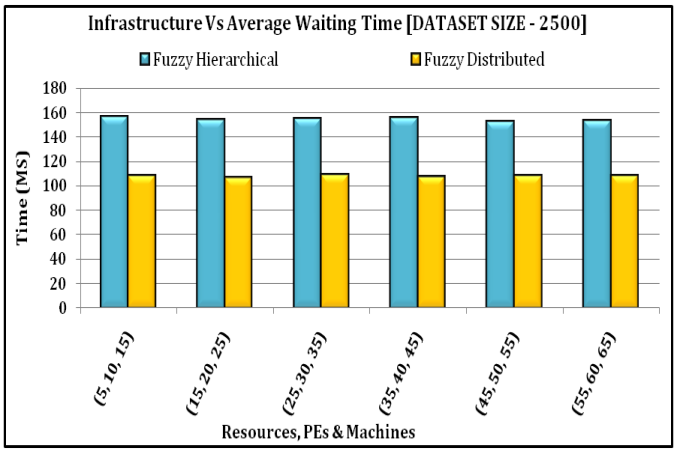

Fig-5.3.3: Infrastructure Vs Average Waiting Time

6. CONCLUSION & FUTURE SCOPE

The enhanced versions of DLBCGBH – H /D termed as Fuzzy Distributed & Fuzzy Hierarchical approaches, resulted by integrating DLBCGBH – H / D with Fuzzy Min – Max Scheduling (FMiMaS) to incorporate Hybrid Scheduling so as to effectively support the dynamic task arrivals, demonstrates tremendous improvements over Built-in Space Shared utility of GridSim 4.0 & DLBCGBH – H / D on all the performance metrics viz. Average Consumed Time, Average Waiting Time, and Average Processing Cost under simulation experiments carried out in Simulation Scenario 1, 2 & 3. Only the two fuzzy approaches are plotted in graphs as these are outperforming over the other three approaches. Fuzzy Distributed approach outperforms Fuzzy Hierarchical approach on all the performance metrics viz. Average Consumed Time, Average Processing Cost and Average Waiting Time. Moreover, with increased number of resources, the performance does not change much for the same size of datasets while improvements are reflected for the increased size of datasets as the values of all the parameters become almost stabilized.

Thus, Fuzzy Hierarchical and Fuzzy Distributed approaches are acceptable over DLBCGBH – H / D for computational grids in the homogenous computing infrastructure of grid resources comprising of similar number of nodes with similar / default PE ratings which proves the effectiveness of the Fuzzy Min – Max Scheduling (FMiMaS). As a matter of fact the improvements are due to the resultant hybrid scheduling caused by the integration of Global scheduling introduced by FMiMaS with the local scheduling implemented through DLBCGBH – H / D due to which the Average Waiting Time is tremendously decreased which further improves the Average Consumed Time and Average Processing Cost.

In future, the performance of proposed Fuzzy Hierarchical and Fuzzy Distributed approaches for load balancing in computational grids can also be analyzed in heterogeneous environment of computing resources. The two enhanced approaches viz. Hierarchical Fuzzy and Distributed Fuzzy can also be tested for fault tolerance. Improvements can also be tried by taking into account the characteristics of task viz. Resource Requirements, CPU Bound, Deadline etc. during its transfer to the best suited node in the grid environment. Further, the complexity of the proposed algorithms can mathematically be analyzed to strongly support the use of binary heaps.

Appendix – A

SIMULATION SCENARIO: 1					
Dataset Size	**Space Shared**	**Distributed**	**Hierarchical**	**Fuzzy Distributed**	**Fuzzy Hierarchical**
Parameter: Average Consumed Time					
250	25265.52	18025.09	23885.45	14600.32	17675.23
500	22932.21	19556.62	20874.07	16231.99	16490.52
1000	18543.24	15451.51	16845.24	12979.27	13307.74
1500	18756.04	17282.65	17578.98	13998.95	14414.76
2000	17783.72	16785.84	16225.99	12589.38	12331.75
2500	16886.85	14656.25	15751.69	12164.69	11656.25
Parameter: Average Processing Cost					
250	75796.56	54075.27	71656.35	43800.97	53025.70
500	68796.63	58669.86	62622.21	48695.98	49471.55
1000	55629.72	46354.53	50535.72	38937.81	39923.22
1500	56268.12	51847.95	52736.94	41996.84	43244.29
2000	53351.16	50357.52	48677.97	37768.14	36995.26
2500	50660.55	43968.75	47255.07	36494.06	34968.75
Parameter: Average Waiting Time					
250	14547.85	8812.45	12205.14	387.75	534.59
500	12215.52	10751.47	11651.51	473.06	510.34
1000	11542.81	8756.52	8539.95	385.29	409.92
1500	11120.74	9001.63	9066.96	396.07	426.15
2000	9126.32	8745.14	8021.21	384.79	393.04
2500	8542.99	8121.01	8514.63	357.32	372.94

SIMULATION SCENARIO: 2					
Infra-structure	**Space Shared**	**Distributed**	**Hierarchical**	**Fuzzy Distributed**	**Fuzzy Hierarchical**
Parameter: Average Consumed Time Dataset: 250					
(5, 10, 15)	15811.43	13145.45	14271.12	10601.44	10655.27
(15, 20, 25)	15251.32	13089.61	14259.12	10599.74	10643.50
(25, 30, 35)	15321.55	12920.01	14322.26	10599.33	10642.12
(35, 40, 45)	15329.45	13054.21	14652.87	10597.01	10653.23
(45, 50, 55)	15698.27	13515.65	14221.28	10596.36	10657.77
(55, 60, 65)	15325.22	12478.55	14175.58	10599.94	10641.49
Parameter: Average Processing Cost Dataset: 250					
(5, 10, 15)	47434.29	39436.35	42813.36	31804.32	31965.81
(15, 20, 25)	45753.96	39268.83	42777.36	31799.22	31930.50
(25, 30, 35)	45964.65	38760.03	42966.78	31797.99	31926.36
(35, 40, 45)	45988.35	39162.63	43958.61	31791.03	31959.69
(45, 50, 55)	47094.81	40546.95	42663.84	31789.08	31973.31
(55, 60, 65)	45975.66	37435.65	42526.74	31799.82	31924.47
Parameter: Average Waiting Time Dataset: 250					
(5, 10, 15)	4678.65	2365.12	3762.25	110.62	164.82
(15, 20, 25)	4588.23	2485.47	3721.01	108.92	152.68
(25, 30, 35)	4612.57	2455.98	3769.04	108.51	151.30
(35, 40, 45)	4783.01	2367.54	3798.25	106.18	162.41
(45, 50, 55)	4669.33	2566.01	3714.65	105.53	166.94
(55, 60, 65)	4789.25	2449.52	3725.33	109.11	150.66

SIMULATION SCENARIO: 3					
Infra-structure	**Space Shared**	**Distributed**	**Hierarchical**	**Fuzzy Distributed**	**Fuzzy Hierarchical**
Parameter: Average Consumed Time Dataset: 2500					
(5, 10, 15)	10289.30	8859.39	9626.89	7286.42	7334.56
(15, 20, 25)	10251.32	8711.53	9602.97	7285.01	7331.92
(25, 30, 35)	10221.62	8605.33	9615.33	7287.38	7333.02
(35, 40, 45)	10482.32	8701.68	9786.22	7285.33	7333.67
(45, 50, 55)	10458.11	8666.39	9728.27	7285.91	7330.30
(55, 60, 65)	10412.20	8706.14	9539.01	7286.56	7331.66
Parameter: Average Processing Cost Dataset: 2500					
(5, 10, 15)	30867.90	26578.17	28880.67	21859.26	22003.68
(15, 20, 25)	30753.96	26134.59	28808.91	21855.03	21995.76
(25, 30, 35)	30664.86	25815.99	28845.99	21862.14	21999.06
(35, 40, 45)	31446.96	26105.04	29358.66	21855.99	22001.01
(45, 50, 55)	31374.33	25999.17	29184.81	21857.73	21990.90
(55, 60, 65)	31236.60	26118.42	28617.03	21859.68	21994.98
Parameter: Average Waiting Time Dataset: 2500					
(5, 10, 15)	3365.21	1652.32	2465.02	108.26	156.36
(15, 20, 25)	3365.01	1684.29	2599.45	106.80	153.76
(25, 30, 35)	3371.36	1645.21	2541.21	109.16	154.83
(35, 40, 45)	3345.66	1587.99	2578.54	107.11	155.49
(45, 50, 55)	3355.25	1502.69	2696.21	107.69	152.10
(55, 60, 65)	3389.52	1545.45	2644.11	108.35	153.46

REFERENCES

[1] Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations", International Journal of High Performance Computing Applications, Volume 15, Number 3, 2001, pp. 200-222.

[2] Anuradha Sharma and SeemaVerma, "A Survey Report on Load Balancing Algorithm in Grid Computing Environment", International Journal of Advanced Engineering Research and Studies, Volume IV, Issue II, Jan.-March, 2015, pp. 128-132.

[3] Gaurav Sharma and JagjitKaur Bhatia, "A Review on Different Approaches for Load Balancing in Computational Grid", Journal of Global Research in Computer Science, Volume 4, No. 4, April 2013.

[4] Lu, Kai, RikySubrata, and Albert Y. Zomaya, "On the performance-driven load distribution for heterogeneous computational grids", Journal of Computer and System Sciences 73, No. 8, 2007, pp.1191-1206.

[5] Subrata, Riky, Albert Y. Zomaya, and Bjorn Landfeldt., "Artificial life techniques for load balancing in computational grids", Journal of Computer and System Sciences, Volume 73, Number 8, 2007, pp. 1176-1190.

[6] D.K. Patel et al., "Survey of load balancing techniques for grid", Journal of Network and Computer Applications, 65, 2016, pp.103–119.

[7] Yan, Kuo-Qin, Shun-Sheng Wang, Shu-Ching Wang, and Chiu-Ping Chang, "Towards a hybrid load balancing policy in grid computing system", Expert Systems with Applications, Volume 36, and Number 10, 2009, pp. 12054-12064.

[8] L.A.Zadeh, "From Computing with numbers to computing with words – from manipulation of measurements to manipulation of perceptions", Int. J. Appl. Math. Comp. Sc., Vol. 12, No. 3,2002,pp. 307-324.

[9] L.Cheung and Y.Kwok, "On load balancing approaches for distributed object computing systems", The Journal of Supercomputing, vol. 27, 2004, pp 149-175.

[10] K.Lu, R.Subrata, and A.Y.Zomaya, "An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites", Proceedings of 25th IEEE International Performance Computing and Communication Conference (IPCCC '06), 2006.

[11] Y.Li, Yuhang Yang and Rongbo Zhu, "A Hybrid Load Balancing Strategy of Sequential Tasks for Computational Grids", IEEE International Conference on Networking and Digital Society, 2009, pp. 112-117.

[12] Rathore N, Channa I, "Variable threshold based hierarchical load balancing technique in Grid", Engineering with Computers2014, pp. 1–19.

[13] Abbas Karimi and Faraneh Zarafshan, "A New Fuzzy Approach for Dynamic Load Balancing Algorithm", International Journal of Computer Science and Information Security(IJCSIS), Volume 6, Number 1, 2009.

[14] Ali M. Alakeel, "A Fuzzy Dynamic Load Balancing Algorithm for Homogenous Distributed Systems", World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering, Volume 6, Number 1, 2012.

[15] HelmyTarek et al. "Fuzzy logic-based scheme for load balancing in grid services", Journal of Software Engineering and Applications 5, 2012, 149.

[16] Anuj Kumar, Heman Pathak, "Dynamic Load Balancing in Heterogeneous Hierarchical Computational Grids using Fuzzy Logic (LBHHGF)", In Proceedings of International Conference on Advance Computing and Software Engineering (ICASE16), KNIT, Sultanpur, September 2016.

[17] Deepa N. K. and L. M. Nithya, "Fuzzy Logic Based Job Scheduling in Computational Grid with Minimum Communication and Replication Cost", International Journal of Innovative Research in Science, Engineering and Technology, Volume 3, Special Issue 1, February 2014.

[18] Mika RantonenaTapioFrantti and KaukoLeiviskä, "Fuzzy expert system for load balancing in symmetric multiprocessor systems", Expert Systems with Applications, Volume 37, Issue 12, December 2010, pp. 8711-8720.

[19] Kai Lu, RikySubrata and Albert Y. Zomaya, "On the performance-driven load distribution for heterogeneous computational grids", Journal of Computer and System Sciences, Elsevier Inc., Volume 73, 2007, pp. 1191–1206.

[20] T. Kokilavani and D.I. George Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", International Journal of Computer Applications (IJCA), Volume 20, Number 2, April 2011.
[21] Anousha S, Ahmadi M., "An improved Min–Min task scheduling algorithm in grid computing", In Proceedings of the international conference on grid and pervasive computing (GPC'13). Lecture Notes in Computer Science; 2013, vol.7861. p. 103–13.
[22] SaeedParsa, Reza Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm", "International Journal of Digital Content Technology and its Applications", volume3, No. 4, December 2009, pp. 91-99.
[23] A. Kumar, H. Pathak, "Dynamic Load Balancing for Computation Grids using Binary Heaps DLBCHBH – H / D)", International Journal of Computer Science and Engineering (2347 – 2693), Volume 6, Issue 5, 2018.
[24] Anuj Kumar, Heman Pathak, "A Comparative Study of Grid Load Balancing", International Journal of Computer Applications (0975 – 8887), Volume 179, No.18, February 2018.
[25] Parallel Workload Archive from http://www.cs.huji.ac.il/labs/parallel/workload/

Author's Profiles

Mr. Anuj Kumar pursued M.Sc. (Computer Science) from G.B.Pant University of Agriculture & Technology, Pantnagar and PGDM from Shri Ram Murti Smarak College of Engineering & Technology, Bareilly. He is currently working as Associate Professor, Department of Computer Applications, Shri Ram Murti Smarak College of Engineering & Technology, Bareilly. He is pursuing Ph.D. in Computer Science from Department of Computer Science, Faculty of Technology, Gurukul Kangri Vishwavidyalaya, Haridwar, India. His area of interest in teaching and research include Algorithms, Soft Computing and Distributed Systems.

Dr. Heman Pathak pursued M.Sc. (Computer Science) from BHU, Varanasi and Ph.D. in Computer Science from Gurukul Kangri Vishwavidyalaya, Haridwar. She is currently working as Associate Professor, Department of Computer Science, Faculty of Technology, Kanya Gurukul Campus, Dehradun (Second Campus of Gurukul KangriVishwavidyalaya, Haridwar, India). During her teaching experience of 19 years and research experience of 10 years, she has published more than 30 research papers in reputed International & National Journals and has attended 12 International and 08 National conferences. Her areas of interest in research include Distributed Systems.

