Efficient Refinery Scheduling Heuristic in Heterogeneous Computing Systems

Sunita Bansal
Department of Computer Science & Information Systems
Birla Institute of Technology & Science
Pilani, Rajasthan, 333031, INDIA
sunita_bansal@bits-pilani.ac.in

Chittaranjan Hota
Department of Computer Science & Information Systems
Birla Institute of Technology & Science, Pilani
Hyderabad Campus, Hyderabd, AP, INDIA
hota@bits-hyderabad.ac.in

Abstract - **With the emergence of distributed systems, the problem of task scheduling has been arousing attention in recent past. Task scheduling is a NP-complete problem and it is more complicated under the distributed heterogeneous computing environment. To harness the potential of these systems, efficient scheduling algorithms are needed. This paper proposes a new distributed scheduling algorithm for independent tasks to be assigned optimally amongst available machines. The approach works in two phases. In first phase, it assigns a task according to the Min-min heuristic and in second phase, it improves the scheduling by using efficient refinery scheduling heuristic. The refinery heuristic balances the load across all the machines and reduces the make-span time of jobs. The results obtained using the proposed heuristic improves over the existing approaches.**

Index Terms - **Scheduling, Heuristics, Load balancing, Grid computing.**

I. Introduction

Grid computing is combination of computer resources from multiple administrative domains to reach a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of jobs [1]. Computational grid is a kind of grid environments, targeted at solving computationally intensive problems. To make effective and efficient use of the tremendous capabilities of the computational grids, efficient task scheduling algorithms are required. In high throughput computing, the grid aims to schedule large number of meta-tasks, with the goal of reducing the idle time of the machines, reducing the make-span and balancing the load well across the machines.

In this paper, an Efficient Refinery scheduling algorithm (ER) is proposed that assigns the tasks optimally across the machines. ER first assigns the tasks using Min-min heuristic [2]. Then, tasks are reallocated from highest completion time machine. In the reassignment, either the task is moved or swapped from the highest completion time machine to other machines. ER scores over Refinery because it does not require initial sorting, assigns task using benchmark heuristics and reallocates tasks.

The outline of this paper is organized as follows. Section II reviews related work on task scheduling in heterogeneous computing environment. Section III formulates the problem definition. Section IV describes our proposed scheduling for independent tasks scheduling in distributed heterogeneous computing systems. Evaluation of the performance of this strategy is presented in Section V. Experiment results are shown in section VI. Section VII discusses the conclusion and future work.

II. Related Work

The mapping heuristics can be grouped into two categories: on-line mode and batch-mode heuristics. In the on-line mode, a task is mapped onto a machine as soon as it arrives at the scheduler. In the batch mode, tasks are not mapped onto the machines as they arrive; instead they are collected into a set that is examined for mapping at prescheduled times called mapping events. A meta-task can include newly arrived tasks (i.e., the ones arriving after the last mapping event) and the ones that were mapped in earlier mapping events, but did not begin execution.

Braun et al. [2] have studied on-line and batch mode heuristics i.e., MCT (Minimum Completion Time), MET (Minimum Execution Time), Min-min and Max-min heuristics. MET scheduling algorithm assigns a task on a machine, which has minimum execution time. It does not care about the current load of the machine. MCT assigns a task on a machine which has minimum expected completion time. MCT does not assign a task on a machine which has minimum execution time. Min-min heuristic first finds minimum completion time of all unmapped tasks. Next, the task which has minimum completion time is selected and mapped to the machine. Finally, the newly mapped task is removed and the process repeats until all tasks are mapped. Max-min heuristic first finds out the minimum completion time of

doi:10.4304/jait.2.3.159-164

all unmapped tasks and assigns a task which has maximum completion time, on a machine. Max-min attempts to minimize the penalties incurred from performing tasks with longer execution time.

Casanova et al. [3] defined sufferage value as the difference between it is best minimum completion time and it is second-best minimum completion time. The tasks with high sufferage value takes precedence.

Parsa et al. [4] proposed Resources Aware Scheduling Algorithm (RASA) which is a combination of Min-min and Max-min strategies. They applied Max-min and Min-min algorithms, alternatively. Their heuristics used this strategy to execute small tasks before the larger ones. This is to avoid delays in executing larger tasks and to support concurrency in the execution of the large and small tasks.

Kamalam et al. [5] presented Min-mean mapping heuristic which first assigns a task based on Min-min algorithm. Then, the mean of all machine's completion time is taken. The machine whose completion time is greater than the mean value is selected for participating in scheduling. The tasks allocated to the selected machines are reallocated to the machines whose completion time is less than the mean value. It obtains less completion time, but takes more time to assign a task.

Bey et al. [6] introduced a make-span refinery approach to schedule unmapped tasks. This scheduling algorithm first sorts the tasks and then assigns the tasks according to Max-min algorithm, which is called initial task scheduling algorithm. Next, it tries to minimize the overall make-span time; using reassignment of the tasks between the machine that has maximum completion time and others in the system.

Kim et al. [8] studied eight dynamic mapping schemes that are Max-max, Max-min, Min-min, Queuing table, Suffrage, Slack suffrage, Switching and Percent best. The priorities, deadlines, and dynamic arrival of tasks were considered in their study. When loose deadlines were used, the Max-max heuristic and the Slack suffrage heuristic perform the best. When tight deadlines were used, the performance of all heuristics is degraded.

Shan et al. [9] introduced a QoS guided Min-min heuristic for independent Grid task scheduling problems. It classifies the tasks based on bandwidth requirement. First it assigns a high quality task and then low quality tasks are assigned using Min-min heuristic. Whenever the bandwidth requirements of all tasks are the same, it works like Min-min [2] algorithm.

Ehsan et al. [10] proposed High Standard Deviation First (HSTDF) heuristic, which considers the standard deviation of the expected execution time of a task as a selection criterion. The task which has high standard deviation must be assigned first. Time complexity of this algorithm is high.

Wu et al. [11] heuristic first ordered the tasks by their expected completion time and then the ordered sequence is segmented and finally it applies Min-min heuristic to these segments. This heuristic works better than Min-min where lengths of tasks are different. It executes first the long tasks and then shorter once.

III. Problem Definition

In this model static mapping of meta-tasks is considered. The accurate estimate of the expected execution time for each task on each machine is known as priory to execution and is contained within an ETC (Expected Time to Compute) matrix. In the ETC matrix row represents task (T_i) and column represents machine (M_j) estimated execution time. Using the ETC matrix model, the expected completion time of the task T_i on machine M_j is follows

$$CT(T_i, M_j) = RT(M_j) + ETC(T_i, M_j)$$

$RT(M_j)$ is the machine availability time which is the time at which machine M_j completes any previously assigned tasks.

The main aim of the heuristic scheduling algorithm is to minimize the make-span where,

$$\text{Make-span} = \text{Max}\,(CT(T_i, M_j))$$

The related definitions of proposed heuristic scheduling algorithm are as follows:

- ETC_{ij} - The amount of time taken by a machine M_j to execute T_i given that M_j is idle when T_i is assigned.
- CT_{ij} – The overall completion time of Ti on machine Mj
- ReadyTime (M_j) - The time at which M_j completes any previously assigned tasks.

IV. Efficient Refinery Scheduling Heuristic

ER assigns a job optimally in two phases. First phase, assigns tasks according to Min-min heuristic. Second phase, reassigns the tasks which are assigned in first phase. Reassignment is done by reallocating a task from highest completion time machine to other machines in the system. Tasks are reassigned either by swap strategy or by move strategy. Swap strategy retains the same number of tasks scheduled on each machine. While the move strategy changes the number of tasks on two machines.

First Phase: In our strategy, it is very important to select a better initial scheduling solution to achieve low make-span. Thus, we have chosen the Min-min heuristic. Min-min heuristic [2] is known as benchmark heuristic and gives the minimum make-span time. This heuristic first finds minimum completion time of all unmapped tasks. Next, the task which has minimum completion time is selected and mapped to the machine. Next, the newly mapped task is removed; the process repeats until all tasks are mapped.

Second Phase: In a distributed environment, mostly there will be only one machine which has the maximum completion time and other machines have less completion time. Therefore, to reduce the make-span of machine we move or swap the tasks between machines. Move procedure moves the task from one machine to another machine. Swap procedure swaps tasks between two machines. ER chooses the method which gives better result. ER is described in algorithms 1, 2, and 3.

ER heuristic (Algorithm-1) first assigns the tasks according to Min-min heuristic. Next, we find new reassignment using move or swap procedure. Based on

the outcome, i.e., whichever method gives less make-span time is selected, and then tasks are reassigned. This process works iteratively, until there is any change in the make-span.

Algorithm-2 describes a move procedure. It finds a task which can be removed from highest completion time machine M_{max} and be assigned to less completion time machine M_{less}. Then, it finds new make-span New_{max} and New_{less}. Finally, if new make-span (New_{max} and New_{less}) is less than the highest completion time ($Highest_{ct}$), then the $Highest_{ct}$ is updated.

Algorithm-3 shows the swap procedure. This method works similar to the move procedure. Instead of moving a task it swaps the tasks between two machines so that same numbers of tasks are retained on the machines.

First phase, assign task using Min-min heuristic and find $Highest_{ct}$
Do until make-span change
Find new' using m*ove_procedure()*
Find new" using swap_procedure()
If *(new' <=new")*
Make changes according to move procedure
Else
Make changes according to swap procedure
End Do

Algorithm-1 ER heuristic

Move_procedure()
For all task on M_{max}
Pick a task T_i
For all M_{Less}
Find New_{max} of M_{max} by removing task T_i
Find New_{less} by adding a task T_i
If New_{max} and New_{less} < $Highest_{ct}$
$Highest_{ct}$ = new make-span
End if

Algorithm-2 Move procedure

Swap_procedure()
For all task on M_{max}
Pick a task T_i
For all M_{Less}
Pick a task T_j from M_{less}
Swap task T_i and T_j of M_{max} and find New_{max}
Swap task T_j and T_i of M_{less} and find New_{less}
If New_{max} and New_{less} < $Highest_{ct}$
$Highest_{ct}$ = new make-span
End if

Algorithm -3 Swap procedure

ILLUSTRATIVE EXAMPLE

Consider a sample ETC matrix given in Table-1. It has 15 tasks and three machines. ER first assigns tasks according to Min-min that is shown in Table-2. Now, we find a new make-span from move or swap procedure. In Iteration-1 swap procedure swaps the task T4 and T8 from M1 to M2 to reduce the make-span of M2. This makes make-span 843(Table-3). Move procedure moves a T3 from M2 and assign to M0 and makes make-span is 840. Therefore, we choose the move procedure and results are shown in Table-4. Next, Iteration-2 swap procedure swaps the task T2 and T3 from M0 to M1 and makes a new make-span 764 (shown in Table-5). Move procedure moves a task T13 from M0 and assign to M1, results are shown in Table-6. Here, we choose move procedure. Next, Iteration-3 swap does not reduce the make-span. Move procedure further moves a T9 from M2 and assign to M1, results are shown in Table-7. Iteration-4 does not reduce the make-span further. Thus, ER make-span is 746 where as Refinery make-span is 837.

The complexity of the first phase is $O(m \cdot n^2)$ where m is the number of machines and n is the number of tasks. The complexity of second phase is $O(k \cdot m \cdot n^2)$ where k is the number of iteration when the make-span value does not changed, m is the number of machines and n is the number of tasks. So, the total complexity is max of both phase, which is $O(k \cdot m \cdot n^2)$.

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
M0	694	179	237	391	401	75	593	545	109	35	433	78	1	163	31
M1	604	279	237	469	451	667	75	69	25	52	11	386	271	111	271
M2	594	309	532	157	151	593	223	545	31	18	217	1	631	244	101

Table-1 ETC matrix

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	CT
M0		179				75							1	163	31	449
M1			237				75	69	25		11					417
M2	594			157	151					18		1				**921**

Table-2 Min-min heuristic

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	CT
M0		179				75							1	163	31	449
M1			237		451		75	69			11					**843**
M2	594			157					31	18		1				801

Table-3 Iteration-1 of ER (swap procedure)

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	CT
M0		179		391		75							1	163	31	**840**
M1			237				75	69	25		11					417
M2	594				151					18		1				764

Table-4 Iteration-1 of ER (move procedure)

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	CT
M0		179	237			75							1	163	31	686
M1				469			75	69	25		11					649
M2	594				151					18		1				**764**

Table-5 Iteration-2 of ER (swap procedure)

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	CT
M0		179		391		75							1		31	677
M1			237				75	69	25		11			111		528
M2	594				151					18		1				**764**

Table-6 Iteration-2 of ER (move procedure)

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	CT
M0		179		391		75							1		31	677
M1			237				75	69	25	52	11			111		580
M2	594				151							1				**746**

Table-7 Iteration-3 of ER (move procedure)

V. Simulation Procedure

To evaluate and compare the proposed heuristic with the other existing algorithms, a Java based simulator on Pentium IV has been developed.

The task arrivals are modeled by a Poisson random process [7]. It contains an ETC matrix which has the expected execution times of a task on all machines. The ETC matrix entries represented as ETC_{ij} values, which heuristic would use in its operation. The actual execution time of tasks may differ from the values that are in ETC matrix. In ETC matrix, a row represents the execution time of the corresponding task on different machines. The average variation along the rows is referred as the machine heterogeneity [8, 9]. Similarly, the average variations along the columns are referred as the task heterogeneity. One classification of heterogeneity is to divide it into high heterogeneity and low heterogeneity. Based on the above idea, four categories were proposed for the ETC matrix in (a) high task heterogeneity and high machine heterogeneity (HiHi), (b) high task heterogeneity and low machine heterogeneity (HiLo), (c) low task heterogeneity and high machine heterogeneity (LoHi), and (d) low task heterogeneity and low machine heterogeneity (LoLo). The ETC matrix can be further divided into two classes, consistent and inconsistent, which are orthogonal to the previous classifications. For a consistent ETC matrix, if machine M_x has a lower execution time than machine M_y for task T_k then the same is true for any task T_i. The ETC matrices which are not consistent are inconsistent ETC matrices. In addition to the consistent and inconsistent classes, a semi-consistent class could also be defined. A semi-consistent ETC matrix is characterized by a consistent sub-matrix. In the semi-consistent ETC matrices used here, 50% of the tasks define a consistent sub-matrix.

Let an ETC matrix have n_{max} rows and m_{max} columns. Random ETC matrices that belong to the different categories are generated in the following manner:

1. Let Γt be arbitrary constant quantifying task heterogeneity, being smaller for low task heterogeneity. Let N_t be a number picked from the uniform random distribution with range [1, Γt].

2. Let Γm be arbitrary constant quantifying machine heterogeneity, being smaller for low machine heterogeneity. Let Nm be a number picked from the uniform random distribution with range [1, Γm].

3. Sample *Nttmax* times to get a vector q[0..(tmax-1)].

4. Generate the ETC matrix, e[0..(tmax-1),0…(mmax-1)] by the following algorithm:

For *ti from* 0 to (*tmax*-1)
For *mj* from 0 to (*mmax*-1)
Pick a new value for *Nm*
e[i, j] = q[i] * *Nm*.
End for
End for

From the raw ETC matrix generated above, a semi consistent matrix could be generated by sorting the execution times for a random subset of the tasks on a random subset of machines. An inconsistent ETC matrix could be obtained simply by leaving the raw ETC matrix as such. Partially consistent and consistent ETC matrices are not considered in this study because they are least likely to arise in the current intended Management System for Heterogeneous Environments (MSHE).

In the experiments described here, the values of Γt for low and high task heterogeneities are 100 and 3000, respectively. The values of Γm for low and high machine heterogeneities are 10 and 1000, respectively. These heterogeneous ranges are based on one type of expected environment for MSHE.

A. *Performance Metrics:*

Make-span is the measure of the throughput of the Grid. It can be calculated using following equation:

$$\text{Make-span} = \max_{ti \varepsilon MT}(CT_i)$$

The less the make-span of a scheduling algorithm, the better it works.

VI. RESULTS

In order to evaluate the proposed approach, we have implemented the algorithms described in Section-IV, and compared our output with Min-min heuristic and Refinery scheduling heuristic for independent task scheduling in heterogeneous distributed computing.

The performance of the heuristic algorithm was evaluated by the average make-span of 100 results on the 100 ECTs generated by the same parameter. In all the experiments, the size of ETC is 16 machines and 512 tasks.

Figure-1 through Figure-4 compare the average make-span time of Min-min, Refinery and ER algorithm. We have shown graphs based on the task and machine heterogeneity. At X-axis machine and task heterogeneity with consistent, semi consistent and inconsistent tasks are shown and at Y-axis, average make-span time is shown. In all the cases our scheduler gives better performance over multiple different scenarios. Figure-5 shows the improvement of ER over refinery scheduling. ER performs well in all cases. There are 30% improvements in high heterogeneity and semi-consistent matrix (u-hi-hi-s) because there are more variances in the task and machine heterogeneity.

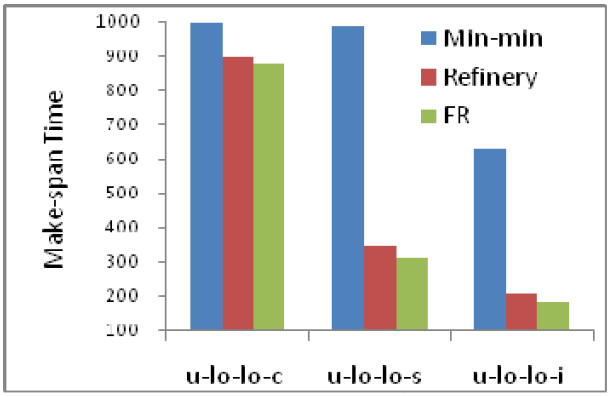

Figure-1 Make-span time of low task and low machine heterogeneity

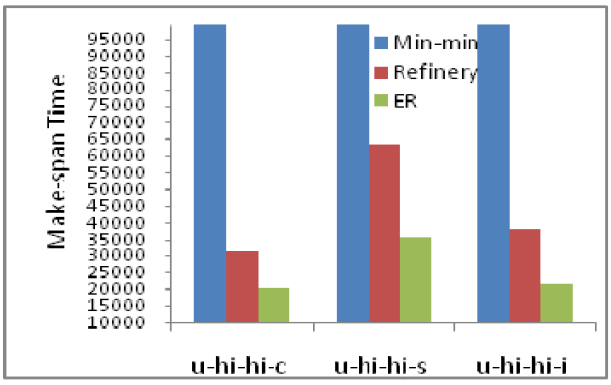

Figure-2 Make-span time of high task and high machine heterogeneity

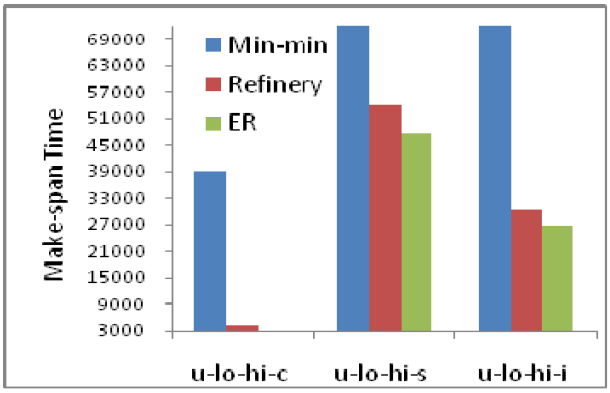

Figure-3 Make-span time of low task and high machine heterogeneity

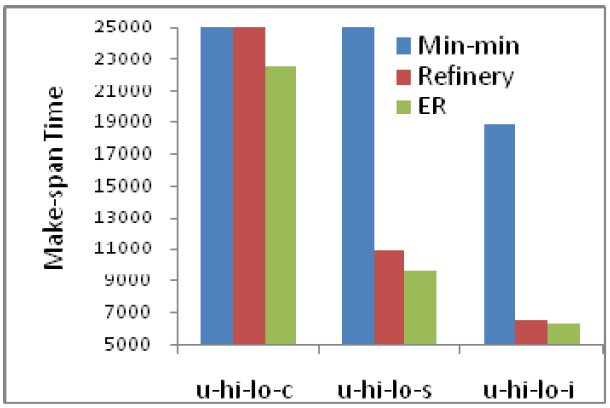

Figure-4 Make-span time of high task and low machine heterogeneity

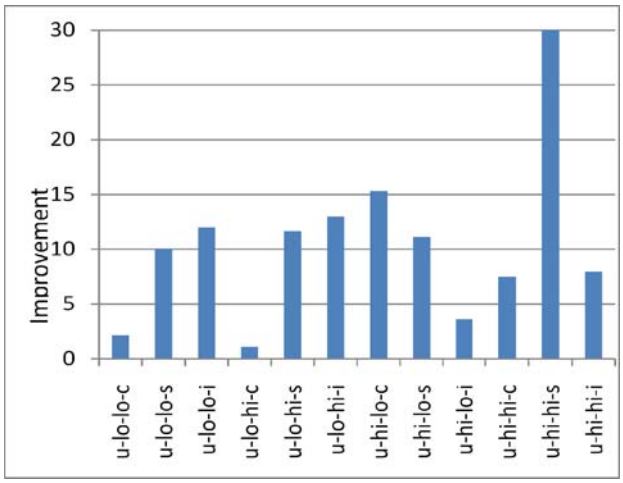

Figure-5 Improvement of ER over Refinery

VII. Conclusion And Future Work

The proposed heuristic scheduling algorithms and other existing ones are tested using the benchmark simulation model for distributed heterogeneous systems. The experimental results show that new heuristic scores over the existing heuristic algorithm in various systems and settings and also it delivers improved make-span on various heterogeneous environments such as job heterogeneity (high, low), machine heterogeneity (high, low), and consistency (consistent, semi-consistent or partially-consistent). In the future, we plan to test this method using a simulation model derived from the queue traces of existing heterogeneous systems.

References

[1] Foster and C. Kesselman, "The Grid: Blueprint for a Future Computing Infrastructure," Morgan Kaufmann Publishers, USA, 1999.

[2] Tracy D. Braun, Howard Jay Siegel, and Noah Beck, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks on to Heterogeneous Distributed Computing Systems," Journal of Parallel and Distributed Computing, Vol.61, No.6, June 2001, pp. 810-837.

[3] Casanova H., Legrand, A., Zagorodnov, D., and Berman, F. "Heuristics for Scheduling Parameter Sweep Applications in Grid Environment," Proceeding of 9th Heterogeneous Computing Workshop, Cancun, Mexico, May 2000, pp. 349-363.

[4] Saeed Parsa and Reza Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications Vol. 3, No. 4, December 2009, pp. 91-99.

[5] Kamalam G. K and MuralibhaskaranV, "A New Heuristic Approach: Min-mean Algorithm for Scheduling Meta-Tasks on Heterogeneous Computing Systems," International Journal of Computer Science and Network Security, Vol.10, No.1, January 2010, pp. 24-31.

[6] Kadda Beghdad Bey, Farid Benhammadia, Aicha Mokhtarib and Zahia Guessoumc, "Independent Task Scheduling in Heterogeneous Environment via Make-span Refinery Approach," International Conference on Machine and web Intelligence, Algiers, Algiers, October 2010, pp. 211-217.

[7] A. Papoulis, "Probability, Random Variables and Stohastic processes," MCGraw-Hill, New York, NY, 1984.

[8] Kim J K , Shivle S, and Siegel H J, "Dynamic Mapping in a Heterogeneous Environment with Tasks having Priorities and Multiple Deadlines," International Symposium on Parallel and Distributed Processing, Washington, May 2003, pp. 98-110.

[9] He Xiao Shan, Sun Xianhe, "QoS Guided Min-Min Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology, Vol.18, No.4, 2003, pp. 442-451

[10] Ehsan Ullah Munir, Jian-zhong Li, Sheng-fei Shi1, Zhao-nian Zou and Qaisar Rasool, "A New Heuristic for Task Scheduling in Heterogeneous Computing Environment," Journal of Zhejiang University Science, Vol.9, No.12, September 2008, pp. 1715-1723.

[11] M. Wu, W. Shu and H. Zhang, "Segmented Min-Min: A Static Mapping Algorithm for Meta-Tasks on Heterogeneous Computing Systems," Proceedings of the 9th Heterogeneous Computing Workshop, Cancun, Mexico, May 2000, pp. 375-385.

Sunita Bansal received her M Tech. (CS) degree from Banasthali Vidyapith, Banasthali, Rajasthan, India in 2005. She is working in Birla Institute of Technology & Science, Pilani since the year 2005. She is currently pursuing her PhD at BITS Pilani. She has published in conferences and journals over past few years. She is member of Computer Society of India (CSI); Indian Society for Technical Education (ISTE); Indian Science Congress Association (ISCA); International Association of Engineers, USA; and International Association of Computer Science and Information Technology, Singapore.

Chittaranjan Hota is currently Associate Professor and Head, Computer Science and Information Systems department at Birla Institute of Technology and Science, Pilani Hyderabad Campus, Hyderabad. He has completed his PhD from BITS Pilani. He is with BITS, Pilani since the year 2000. He had several visiting researcher and visiting professor assignments at International academic and research institutes abroad. He has published extensively in national and international conferences and journals. He is a life member of ISTE, IE, India, and ACM. His research interests are in the areas of Traffic Engineering, and Security in IP Networks, Peer-to-Peer Overlays, Mobile Wireless Networks, and Cloud Computing.