International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 6 – No. 1, September 2013 – www.ijais.org

Preemptive Task Partitioning Strategy (PTPS) with Fast Preemption in Heterogeneous Distributed Environment

[1]Rafiqul Zaman Khan
(Associate Professor)
Department of Computer Science,
Aligarh Muslim University, Aligarh.

[2]Javed Ali
(Research Scholar)
Department of Computer Science,
Aligarh Muslim University, Aligarh.

Abstract
Efficient preemptions in the scheduling of real time systems cause optimal overhead in parallel computing systems. Periodic and sporadic tasks are exists in the real time systems. The periodic tasks may be divided into the synchronous and asynchronous categories. The management of the resource sharing in the parallel computing can be powerfully achieved by preemptive scheduling. Fast preemptions are necessary to achieve the high degree parallelism. In this paper, Earliest Starting Time parameter expended up to a large degree of heterogeneity. We compare the proposed algorithm with the existed well known algorithms: preemptive MCP and FPS algorithms. The result shows better performance of the PTPS in terms of average NSL and running time complexities.

Keywords
Preemptive Scheduling, Normalized Schedule Length, Directed Acyclic Graph, Parallel Computing etc.

1. Introduction
Non-preemptive scheduling strategies [1, 7, 20, 6, 19, 16, 9] are discussed in the literature. MCP (Modified Critical Path), HEFT (Highest Level First) with estimated time algorithm [10], the earliest time first algorithm EFT [6] and dynamic level scheduling algorithm(DLS) are some well known non-preemptive scheduling algorithms. PTS (Preemptive Task Scheduling) algorithms shows low scheduling cost and better load balance than the existing list scheduling algorithms. Time complexities of PTS is better than time complexities of MCP, HLEFT, ETF, DLS algorithms. In these scheduling algorithms turnaround time and CPU utilization are the metrics to evaluate the performance of the system. FCFS scheduling shows worst average job slow down than SJF [17]. Starvation problem may be occurs in FCFS because some long jobs having more priority than short jobs may take very long execution time. This problem shows long delays and very low throughput. Job fairness parameter shows better results in non-preemptive scheduling. It is clear that non-FCFS scheduling are less fair than FCFS due to the starvation problems. Fairness of job may be calculated by the delay in time due to delayed of a later arriving job. If actual start time of a job is greater than its fair start time then the job is treated as unfair. FCFS scheduling algorithms do not shows always better results in terms of fairness than another scheduling algorithm is that provide better response time [21].

The complexities of the scheduling algorithms play a important role for the execution of the processes. Low time complexities shows good performance in all performance metrics [10, 15, 16]. Scheduling cost is proportional to the number of processors used for the scheduling of large number of tasks. This is a major problem for the futuristic due to the increasing number of processors. A scheduling algorithm with fast preemptions can speed up the compilation process. The process must be started as soon as possible for minimization of execution time. If waiting time is shorter then turnaround time is also affected by the earliest short time of the process. Hence throughput is increases if earliest starting time is decreases. Preemptive scheduling may be categories in following categories:

a) Priority based preemptive scheduling: In these scheduling tasks are allocated to the processors according to their priorities.

b) Resource sharing: In this approach, all tasks are allocated to the processors simultaneously. Every tasks get equal time quantum for the execution of threads.

c) Implementation of the preemptive approaches by the thread shares low context switching overhead.

2 Resource sharing
In preemptive scheduling utilization of the resources increased due to the following considerations. These considerations avoid deadlock due to regular preemptions of the resources.

1. Designed algorithm must satisfy the requirement of parallel processing.
2. The algorithm should have low NSL (Normalized Schedule Length) and communication overhead.
3. The algorithm must achieve high performance efficiency and low response time in multi-core systems.
4. At every instant of time, a resource should be hold at most one task.
5. In resource scheduling algorithms deadlock condition must be avoided. To ensure the absence of deadlock condition at least one condition from the below must be satisfied.
    a) Allocation of the resources should be in non sharable mode. Every participating resource in the multiprocessing environment should not be shared by multiple tasks.

b) If any resource allocated to a particular task then another task should wait until it's released by allocated tasks.

c) The resources should not be allocated in the circular manner (Firs allocated resource requested by last resource and second allocated resource must be requested first resource and so on.)

d) No-preemption condition must be satisfied.

6. The proposed algorithm satisfies fixed priority preemptions and it's capable to schedule the large number of tasks simultaneously.

In DAG based preemptive scheduling, a partial portion of the task can be reallocated to the different processors [23,22 ,24,25]. While in the case of no-preemptive scheduling, the allocated processors can't be re-allocated until it finishes the assigned tasks. Flexibility and resource utilization of the preemptive scheduling is more than non-preemptive scheduling in theoretical manner. Practically, re-assigning the partial part of the task causes extra overhead. Preemptive scheduling shows polynomial time solutions while non-preemptive proved NP-complete [26]. Scheduling with duplication upon different processors is also NP-complete. Communication delay amongst the preemptive tasks is more du to preemptions of processors. Preemptive and non-preemptive scheduling approaches investigated by [27] in homogeneous computing architectures. They used independent tasks without having precedence constraints. In DAG, condition of precedence constraints is inserted by Wang and [28] for preemptive and non-preemptive scheduling. Earliest time factor inserted by them to construct scheduling in list scheduling strategies.

3.1 Proposed Algorithm for Preemptive Scheduling:

1. Assign each $t_i \rightarrow P_j$
2. ***for*** all processors

   generate priority queue(t_i)$\rightarrow P_s$

   sort ascending EST(t_i)$\rightarrow P_s$

   end for
3. *calculate AvgFT (t_i) for* P_{all}
4. ***while*** (ready-list not empty) ***do begin***
5. Select task t_i (highest priority) from priority queue;
6. Select the processor P_j for task t_i according to EST;
7. Assigned $t_i \rightarrow P_j$
8. Delete task t_i from the ready-queue;
9. Update ready -queue;
10. compute FT all unscheduled tasks;
11. Schedule smallest FT tasks;
12. Move task $t_i \leftrightarrow P_j$
13. ***end while***

3.1.1 Description of the Scheduling Algorithm:

There are two phase in the proposed scheduling algorithm. In first phase priorities are assigned to the tasks according to their communication time execution time. While in second phase Earliest Start Time(EST) is calculated and tasks are scheduled upon the processors according to their processing capabilities. Each task of the example is assigned on the fastest processor. Nodes are sorted according to their ascending order Earliest Start Time. The first node having highest priority is numbered as 0. Next priority nodes are numbered in the ascending order of natural numbers. After assigning the priorities there is no dependency exist amongst the nodes. If dependency is found the new group of the tasks are generated to remove the dependency of the nodes. In this manner dependency from the tasks of DAG are removed and these tasks are assigned independently.

After calculating Earliest Starting Time the tasks are assigned to the selected processors. When the task is finished then it's removed from the ready queue. Ready queue is updated after the deletion of the tasks. Finish time of the remaining tasks is calculated as follows:

$$FT = t_{exec\ time} + \min\left(EST_{ij} + FT_{i,p}\right) \quad (1)$$

Smallest finish time tasks are assigned prior to the next smallest finish time. Due to the preemption nature of the tasks they can move to each other computing processors. So idle time of the processors decreases due to the optimal and fast preemptions of the tasks upon the group of heterogeneous processors.

Symbol	Definition
t_i	i^{th} Task
P_f	Fastest Processor
P_s	Selected Processor
P_{all}	All Processors
EST_{ij}	Earliest Staring Time of i^{th} task upon j^{th} processor
$FT_{i,p}$	Finish Time of i^{th} task upon j^{th} processor

Table 1: Symbol and Their Description

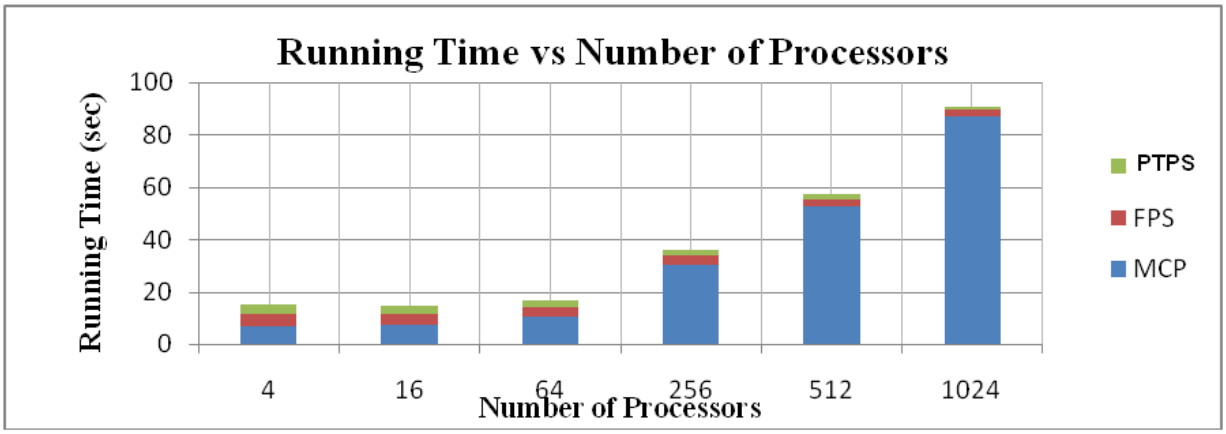
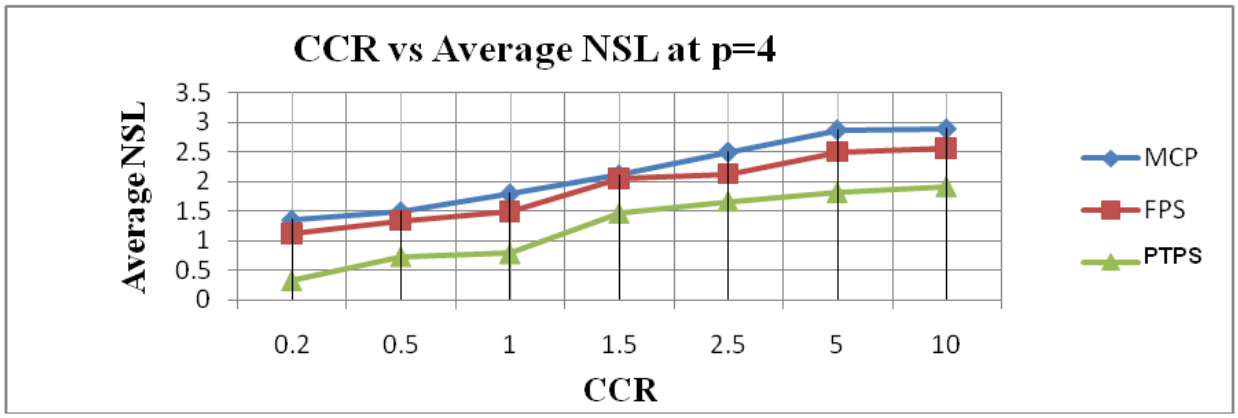
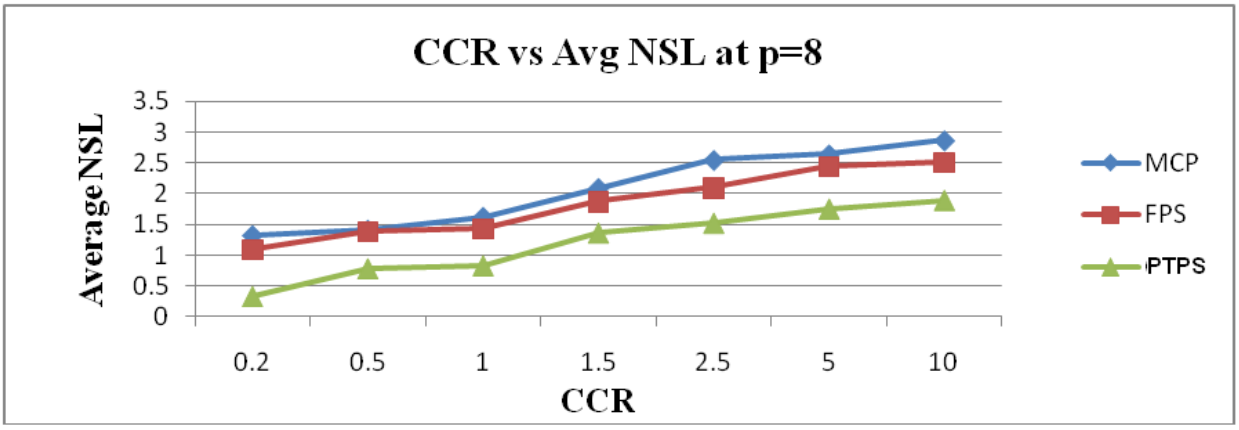


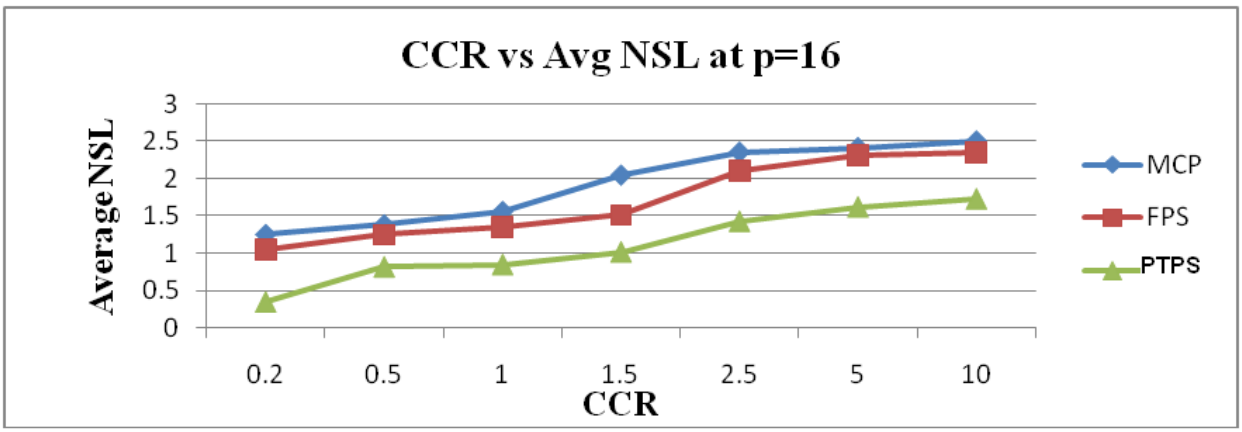
Figure (1): Running Time vs Number of Processors Analyasis



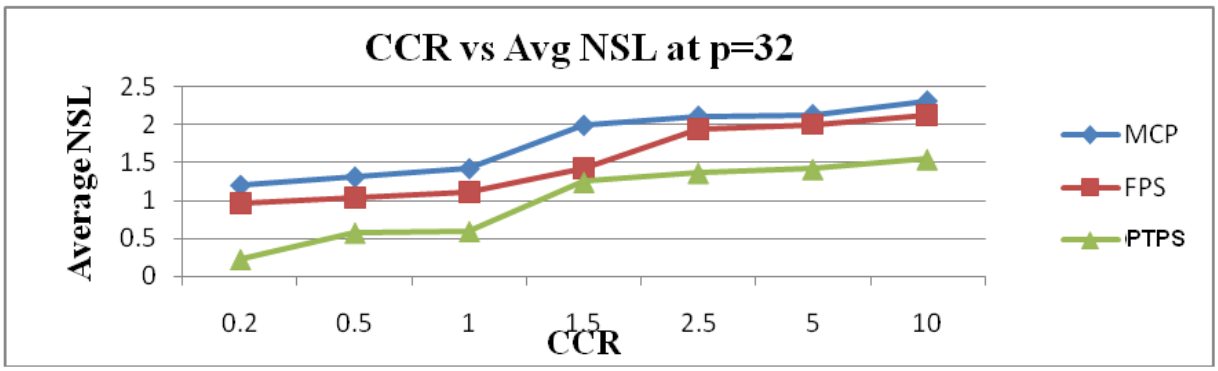
Figure(2): CCR vs Average Normalized Schedule Length Analysis at 4 processors



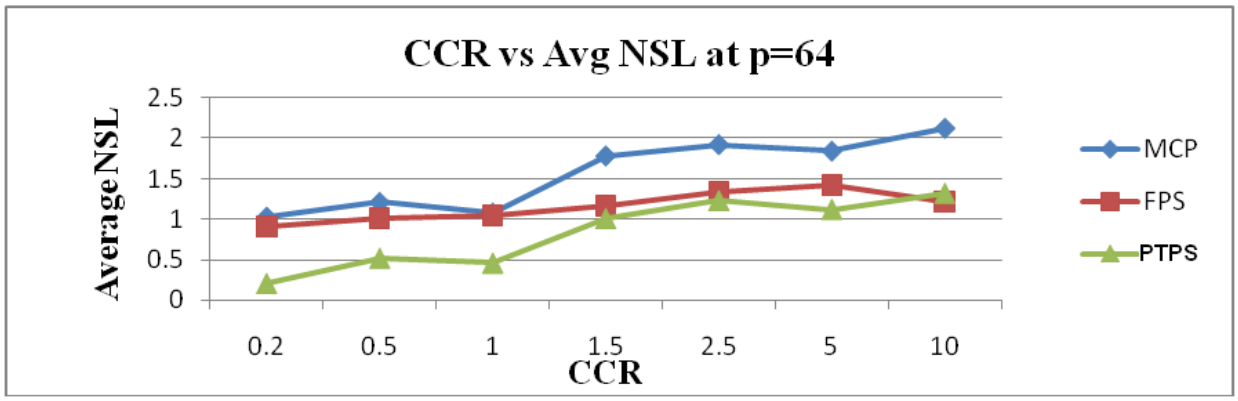
Figure(3): CCR vs Average Normalized Schedule Length Analysis at 8 processors



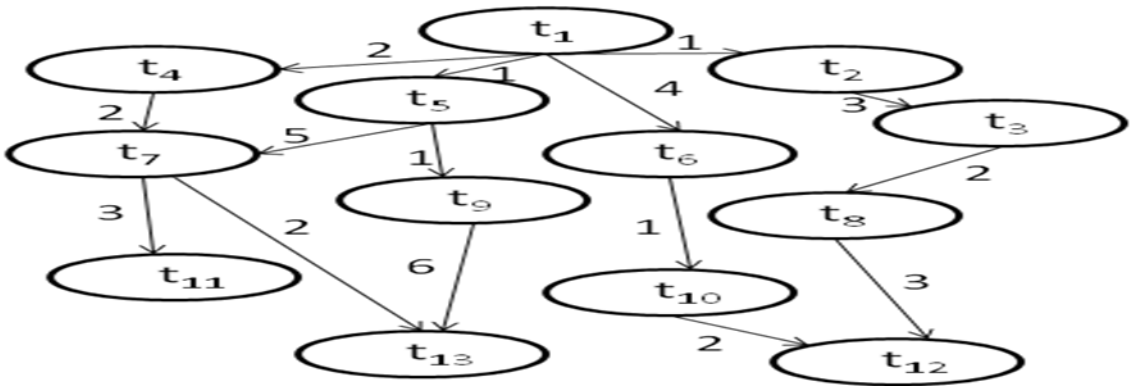
Figure(4): CCR vs Average Normalized Schedule Length Analysis at 16 processors



Figure(5): CCR vs Average Normalized Schedule Length Analysis at 32 processors



Figure(6): CCR vs Average Normalized Schedule Length Analysis at 64 processors



Figure(7): DAG Example for Simulation

4 Experimental Phase

We conducted simulation based experimental evaluation of the proposed algorithm against existed well known FPS (Fast Preemptive Scheduling) and preemptive MCP (Minimum Critical Path) algorithms. In Figure (1), running time (sec) variation with the different number of processors is demonstrated. It can be observed that MCP shows very large running time for the range of processors {4, 16, 64, 256, 512, 1024} in the comparison of FPS and PTPS scheduling algorithms.While the running time of PTPS is 11.91 % less than running time of FPS.

Figure (2) shows the behavior of average NSL (Normalized Schedule Length) against CCR range (0.2, 0.5, 1, 1.5, 2.5, 5, 10) values. When CCR values increases then average NSL value is also increases. At highest value CCR=10 for p=4, NSL value of PTPS decreases by FPS and MCP 11.41 % and 33.56 % respectively. This estimates that if communication cost increases then overhead is also increases. We examine the performance of the proposed algorithm for the range (p=4, 8, 16, 32, 64) of processors. Figure (3-6) shows that optimal results are obtained by PTPS algorithm at the different number of processors. It may be observed that if number of processors increases then average NSL value is continuously going to decreases. So PTPS shows better performance upon the compared algorithms FPS and preemptive MCP scheduling algorithms.

5 Conclusion

This paper contribute the best possible scheduling approach PTPS for heterogeneous computing environments. PTPS is based on the Earliest Start Time factor which provides the optimum processing capability of the existing machines. This approach gives the optimal schedule at very low complexity of the existing scheduling algorithms. A large number of experiments conducted with large number of task graphs. PTPS shows better results over preemptive MCP and FPS in terms of Normalized Schedule Length. Which is a very important parameter for the communication and computational costs of the task partitioning strategies.

6 Future Work

The proposed algorithm may be extended for the group of heterogeneous computing environment. The scalability of the proposed algorithm may be extended at very large scale.

References:

[1] L. George, N. Rivierre, and M. Spuri, (1996). Preemptive and Non-Preemptive Real-Time Uni- Processor Scheduling, Institut National de Rechercheen Informatique et en Automatique, Tech. Rep., 45-50.

[2] S. Baruah and S. Chakraborty,(2006). Schedulability analysis of non-preemptive recurring real-time tasks. *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 12-15.

[3] L. George, N. Rivierre, and M. Spuri. ,(1996).Preemptive and non-preemptive real-time uniprocessor scheduling. Research Report RR-2966, INRIA, 89-90.

[4] K. Jeffay, D. Stanat, and C. Martel., (1991).On non-preemptive scheduling of period and sporadic tasks. *Real-Time Systems Symposium, 1991. Proceedings., Twelfth*,129–139.

[5] Ramaprasad and F. Mueller., (2008).Bounding worst-case response time for tasks with non-preemptive regions. *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE*, 58–67.

[6] R. Dobrin and G. Fohler., (2004).Reducing the number of preemptions in fixed priority scheduling. *Real-Time Systems, 2004. ECRTS 2004. Proceedings. 16th Euromicro Conference on*, pages 144–152.

[7] C.-G. Lee, J. Hahn, Y.-M. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, and C. S. Kim. (1998).Analysis of cache-related preemption delay in fixed-priority preemptive scheduling. *IEEE Trans. Comput.*, 47(6):700–713.

[8] A. Burns. ,(1994).Preemptive priority based scheduling: An appropriate engineering approach. *In S. Son, editor, Advances in Real-Time Systems*, 225–248.

[9] N. Megow, M. Uetz, and T. Vredeveld, (2006).Models and algorithms for stochastic online scheduling", Mathematics of Operations Research, vol. 31, 513-525.

[10] R.H. Moring, A.S. Schulz, and M. Uetz, (1999).Approximation in stochastic scheduling: the power of LP-based priority policies, 1. ACM , vol. 46, 924-942.

[11] Sahni, S. K, (1976).Algorithms for scheduling independent tasks. ACM, vol. 23, 116-127.

[12] R. I. Davis and A. Burns, (2009). Priority Assignment for Global Fixed Priority Pre-Emptive Scheduling in Multiprocessor Real-Time Systems," in *2009 30th IEEE Real-Time Systems Symposium*, Technical report YCS-2010-451, Department of Computer Science, University of York. IEEE, 398–409.

[13] E. Bini and G. Buttazzo, (2005). Measuring the performance of schedulability tests. *Real-Time Systems*, vol. 30, no. 1, 129–154.

[14] C. Li, C. Ding, and K. Shen, (2007) .Quantifying The Cost of Context Switch.In *Proceedings of the 2007 workshop on Experimental computer science*, no. June. ACM, 67-69.

[15] A. Bastoni, (2010).Cache-Related Preemption and Migration Delays : Empirical approximation and Impact on Schedulability.In *Sixth International Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, 33–34.

[16] C. L. Liu and J. W. Layland, (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, vol. 20, no. 1,46–61.

[17] N. Guan, W. Yi, Q. Deng, Z. Gu, and G. Yu, (2011).Schedulability analysis for non-preemptive fixed-priority multiprocessor scheduling. *Journal of Systems Architecture*, vol. 57, no. 5, 536 – 546.

[18] D. I. Katcher, H. Arakawa, J. K. Strosnider., (1993).Engineering and analysis of fixed priority schedulers. IEEE Transactions on Software Engineering, 19(9),920-934.

[19] M. Bertogna and S. Baruah, (2010). Limited preemption of scheduling of sporadic task systems. IEEE Transactions on Industrial Informatics, 45-49.

[20] S. Altmeyer, R. Davis, and C.Maiza, (2011). Cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. In *Proc. 32nd IEEE Real-Time Syst. Symp. (RTSS.11)*, Vienna,Austria, 23-28.

[21] M. Bertogna, O. Xhani, M. Marinoni, F. Esposito, and G. Buttazzo, (2011).Optimal selection of preemption points to minimize preemption overhead. In *Proc. 23rd Euromicro Conf. Real-Time Syst. (ECRTS'11)*, Porto, Portugal, Jul. 6–8, , 217–227.

[22] Cheng, R., Gen, M., And Tsujimura, Y., (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms—I: representation. *Comput. Ind. Eng. 30*, 4, 983–997.

[23] Gonzalez, M. J., Jr.,(1977). Deterministic processor scheduling. *ACM Comput. Surv. 9*, 3 (Sept.), 173–204.

[24] Horvath, E. C., Lam, S., Sethi, R., (1977). A level algorithm for preemptive scheduling. *J. ACM 24*, 1 (Jan.), 36–47.

[25] Rayward-Smith, V. J., (1987).The complexity of preemptive scheduling given interprocessor communication delays. *Inf. Process. Lett. 25*, 120–128.

[26] Coffman, E. G. and Graham, R. L. (1972). Optimal scheduling for two-processor systems. *Acta Inf. 1*, 200–213.

[27] Blazewicz, J., Weglarz, J., and Drabowski, M. (1984). Scheduling independent 2-processor tasks to minimize schedule length. *Inf. Process. Lett. 18*, 267–273.

[28] Chung, Y.-C. and Ranka, S. (1992). Applications and performance analysis of a compile-time optimization approach for list scheduling algorithm on distributed memory multiprocessors. In *Proceedings of the 1992 Conference on Supercomputing* (Supercomputing '92, Minneapolis, MN, Nov. 16–20), R. Werner, Ed. IEEE Computer Society Press, Los Alamitos, CA, 515–525.

BIOGRAPHY AUTHORS:

Dr. Rafiqul Zaman Khan:

Dr. Rafiqul Zaman Khan, is presently working as a Associate Professor in the Department of Computer Science at Aligarh Muslim University, Aligarh, India. He received his B.Sc Degree from M.J.P Rohilkhand University, Bareilly, M.Sc and M.C.A from A.M.U. and Ph.D (Computer Science) from Jamia Hamdard University. He has 18 years of Teaching Experience of various reputed International and National Universities viz King Fahad University of Petroleum & Minerals (**KFUPM**), K.S.A, Ittihad University, U.A.E, Pune University, Jamia Hamdard University and AMU, Aligarh.

He worked as a **Head** of the Department of Computer Science at Poona College, University of Pune. He also worked as a **Chairman** of the Department of Computer Science, AMU, Aligarh. His Research Interest includes Parallel & Distributed Computing, Gesture Recognition, Expert Systems and Artificial Intelligence. Presently **04** students are doing PhD under his supervision. He has published about **35** research papers in International Journals/ConferencesNames of some Journals of repute in which recently his articles have been published are International Journal of Computer Applications (ISSN: 0975-8887), **U.S.A**, Journal of Computer and Information Science (*ISSN: 1913-8989*), **Canada**, International Journal of Human Computer Interaction (*ISSN: 2180-1347*), **Malaysia,** and Malaysian Journal of Computer Science(*ISSN: 0127-9084*), **Malaysia**. He is the Member of Advisory Board of International Journal of Emerging Technology and Advanced Engineering (IJETAE), Editorial Board of International Journal of Advances in **Engineering** & Technology (IJAET), International Journal of Computer Science Engineering and Technology (IJCSET), International Journal in Foundations of Computer Science & technology (IJFCST) and Journal of Information Technology, and Organizations (JITO).

Javed Ali:

Javed Ali is a research scholar in the Department of Computer Science, Aligarh Muslim University, Aligarh. His research interest include parallel computing in distributed systems. He did BSc.(Hons.) in mathematics and MCA from Aligrah Muslim University ,Aligarh. He published seven international research papers in reputed journals. He received state level scientist award by the government of India.