Proceedings of the 21st International Conference on Automation & Computing, University of Strathclyde, Glasgow, UK, 11-12 September 2015

Organization Based Intelligent Process Scheduling Algorithm (OIPSA)

[1]Munam Ali Shah, [2]Muhammad Bilal Shahid, [3]Sijing Zhang, [4]Safi Mustafa, [5]Mushahid Hussain

[1,2,4,5]Department of Computer Science, [3]Department of Computer Science and Technology
[1,2,4,5]COMSATS Institute of Information Technology, Islambad, Pakistan
[3]University of Bedfordshire, Luton, UK
[1]mshah@comsats.edu.com, [1]hallian20@gmail.com, [3]sijing.zhnag@beds.ac.uk, [4]safi07@live.com, [5]mushahidh@yahoo.com

***Abstract*— In a multi-tasking environment, the purpose of a scheduling algorithm is to give CPU time to each process in such a way that maximum throughput could be achieved. One way is to manually set the priority of the processes but conventional operating systems put equivalent scheduling policies on each set of process and do not observe organizational preferences. It is believed that every organization performs same set of tasks most of the time. Tasks performed by an organization must be given priority according to their level of activeness rather than some hard rules defined at a design level. In this paper, we propose a novel algorithm that schedule processes according to the organization's need. Our proposed Organization Based Intelligent Process Scheduling Algorithm (OIPSA) intelligently learns the processes that are frequently used within an organization's operating system and give priority to the users' most wanted processes. The results show that OIPSA decreases response time, waiting time and turnaround time for the organization preferred processes and enhance the overall efficiency of the system when compared with conventional scheduling algorithms.**

Keywords---OS process, scheduling, Organization preference, scheduling algorithm

I. INTRODUCTION

We are in an era in which everyone is moving towards making intelligent and efficient systems. Intelligence is also required at organizations' operating systems that can adapt organizational preferences and behave accordingly. In recent years, there have been quite a few improvements and efforts made towards the development of adaptive systems that make use of reflection model [1]. Reflection modelling is extensively recognized as a very effective and manageable mechanism for runtime adaption and redesign of a system [1].

In a multi-tasking or multi-programming environment, processes when fetched in the main memory are always competing for the CPU time. When one process is in execution cycle other processes are waiting for their turn or waiting for some other event to occur such as I/O operation. This is the main reason, scheduling is important for an operating system. Scheduling decides which process will get the CPU at a given time and which process will be put in the wait condition. Some of the goals that Scheduling algorithm should accomplish to determine performance and efficiency of Scheduling algorithms are throughput, turnaround time, response time, fairness and waiting time [5][6]. Scheduling is also considered as an organized procedure through which process, thread and flow of data is controlled and is given access to different system resources [2]. Scheduling is mainly performed for load balancing and to offer quality of services [3]. All the scheduling algorithms perform some common tasks such as *multi-tasking* (to keep CPU as busy as possible) and *multiplexing* (to transmit multiple data using single physical channel) [2], [4].

Process scheduling has been always a point-of concern for operating system designers. At present, there are lots of scheduling algorithms designed, developed and implemented. Some examples are: First Come First Serve (FCFS) [12], Shortest Job First (SJB), Round Robin [RR], Multi Queue Scheduling and many more [5]. However, there will be always a need for a better and more efficient scheduling algorithm which can use CPU resources efficiently and fairly [6]. In some scenarios, an operating system is unable to select an algorithm at design level which can target organizational preferences and help in improving the throughput of the system [7][10][11] Hard rules defined at design level cannot produce desirable results for majority of the organizations and users get same priority level issues. A similar scenario is personalized operating systems in handheld devices but there exist no personalized operating system at organizational level. In these circumstances, an organization's preference based operating system can perform better.

In this paper, a novel scheduling algorithm called Organization Based Intelligent Process Scheduling Algorithm (OIPSA) is proposed. OIPSA considers the organizations preferences and schedules the processes accordingly. The rest of the paper is organized as follows: Section II reviews the existing literature. Section III provides detailed description of the proposed OIPSA. Section IV provides performance evaluation and comparison of OIPSA with the help of

experiments and results. The paper is concluded in Section V.

II. Related Work

Generally, multi-processors operating system focuses on using the maximum time of CPU and to use CPU resources efficiently for all types of processes. Oldest, easiest, fairest and most commonly used scheduling algorithms is Round Robin (RR) [8]. The aim of its design is specifically for clock/time sharing systems. In RR scheduling algorithm, multiple processes can be present in the ready queue at same time. Scheduler's job is to allocate a short amount of time known as quantum to each process. In this way, all the processes can have CPU time fairly and can progress simultaneously [11]. Recently, a lot of work is being done in the field of CPU computational power and bandwidth of network, few studies show support of multimedia job using CPU. Nieh and Monica [9] proposed a scheduler for multimedia applications, called SMART, which is categorized the CPU time. This scheme supports conventional time sharing applications and real time system as well.

There is also a significant amount of research done in the field of process scheduling using artificial intelligence techniques such as fuzzy logic. One of the techniques for scheduling using fuzzy logic was to decide time quantum for those jobs which are neither too big nor too small. In this way, every job gets a reasonable CPU time and also, the throughput of the system does not reduce due to needless context switching [5]. Sungsoo Lim and Sung-Bae Cho [7], proposed that scheduling must be done according to the type of processes rather than using uniform scheduling policy for each set of process. They also suggested that user preference must be taken into account during scheduling. This will result in the more user friendly behavior of the operating system.

Another technique which uses AI focuses on the priority of relationship between activities instead of Absolute Time Quantum. It was proved that using this technique scheduling will be efficient even if set of activities are not well defined and also when the time of task to complete is large [10]. The static scheduling algorithm in [11] claims that operating system should tune its priorities accordingly [11].

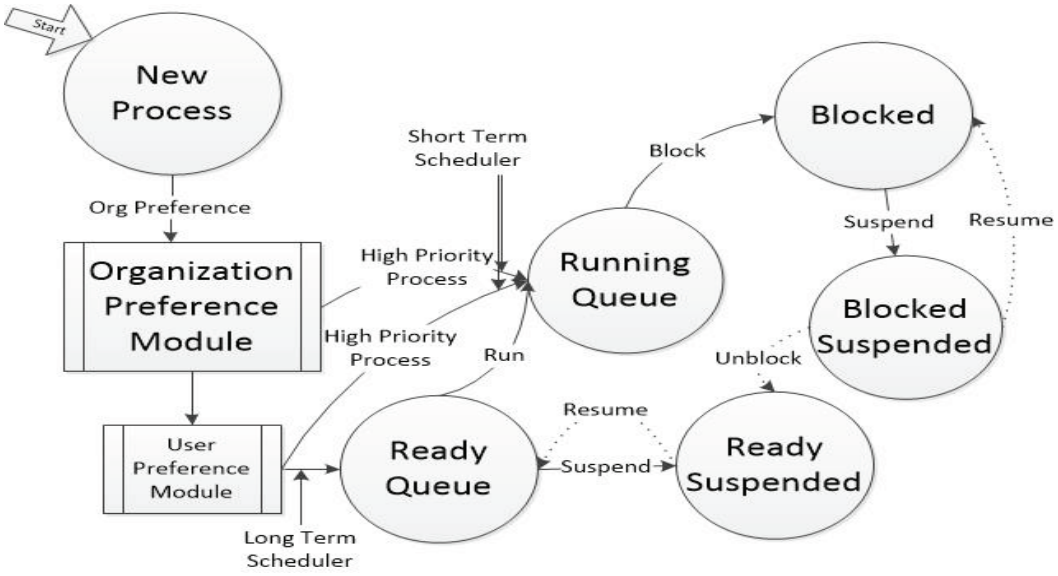

Figure 1. Process State Diagram with Organization Preference Module

III. Organization Based Intelligent Process Scheduling Algorithm (OIPSA)

This section highlights salient features of our proposed OIPSA algorithm. We start our discussion by first presenting the hypothesis for the proposed algorithm and then present the methodology and design for OIPSA.

A. Hypothesis

We believe that each organization perform certain tasks repeatedly and regularly. The users within that organization perform tasks according to their need in general and organizational needs in particular. Using this assumption, we assign priorities to the certain organizational processes. This results in a better working environment and more optimal results than the one which are given by conventional scheduling algorithms. We believe that it is difficult to handle individual processes. To validate our assumption, we perform simulations on different scheduling algorithms and our proposed OPISA. We assume that organization preferred processes are already assigned top priority and are stored in the system. Our

assumption is based on the Windows operating system which stores application preferred data in the following registry:

HKEY_LOCAL_USER/MACHINE\Software\Micros-oft\Windows.

B. Methodology

Simulation experiments and results are used to prove that proposed algorithm performs better in most of the cases. The input values are provided in Table 1. For our simulation, we use open source simulator. A file is taken as an input which is used as a parameter and is run for the proposed OIPSA and for other conventional scheduling algorithms. After this, we compare results and evaluate the performance of each of the algorithm with OIPSA.

C. Design

When a new process is started, it becomes am input for the A new process when started, will become input of Organization Preference Module as shown in Figure. 1. Organization Preference Module considers some predefined rules for new processes. On the basis of these rules, the organizational preference module put the process in one of three queues, i.e., *high priority process queue*; *medium priority process queue* and *low priority process queue*. Long-term Scheduler decides which process should be in ready queue based on multiple priority queue. As the new process is placed in the ready queue, short-term scheduler checks if the process submitted by the long-term scheduler has more priority than the process which is currently running in the running queue. If the recently submitted process doesn't have priority over the running process, then it waits. Otherwise, an interrupt is send to the CPU. The process that was in the running state is sent to the blocked queue and the newly submitted process is given to the CPU. In Figure 1, it can be observed that the Organization Preference Module can send process to running queue if it is up to certain level, and sends to User Preference Module, which can send process directly to running queue if a priority of new process is up to certain level of user importance. This ensures that whenever Organization Preferred Process is created, it gets CPU in no time reducing response time, turnaround time and waiting time.

As discussed above, conventional operating systems usually schedule each and every process uniformly without taking notice of Organization's Preferences, or User's Preferences, because they are hard coded at design level. The designers of the operating system cannot decide target audience of operating system as the user of operating systems are very versatile in nature. Every organization uses operating system for different uses. Some use it for handling complex tasks, while some use it for information processing. Some users use the operating system for software development purposes and few use it for routine tasks. Scheduling decided at design level can be better in certain cases but at the same time it can be devastating for some other organizations. While considering user preferences, the single system will work according to user needs. This will overcome the drawback of prioritizing the preferences in a static fashion. Our proposed OIPSA adjusts the user needs and organizational preferences in a dynamic way.

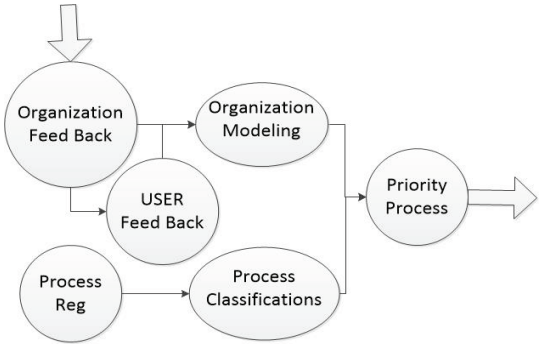

Figure 2. Core Modules of the OIPSA

OPISA is comprised of three major modules as shown in Figure. 2. The *Process Classification* module, which classifies a process into high, medium and low priority processes. Second module is, *Organization Modelling* module, which models the organization's most used processes or organization preferences. This module also checks the user feedback and take organization feedback into consideration. Third module is *Priority Process* module which classifies the priority of each process at run time. The input for the priority module is based on the output from Priority Classification and Organization Modelling modules.

D. Process Classification

Figure. 3 depicts the processes within Priority Classification module. For the Priority Classification, we are keeping a track of processes and number of times they have been a resident in the job queue. The more identical the processes are in job queue, the more they will get priority. Also, according to some specified rules they will be placed in either High Priority Pool, Medium Priority Pool or Low Priority Pool.

E. Process Priority Decesion

In routine, the operating system schedules processes at system level without taking into account

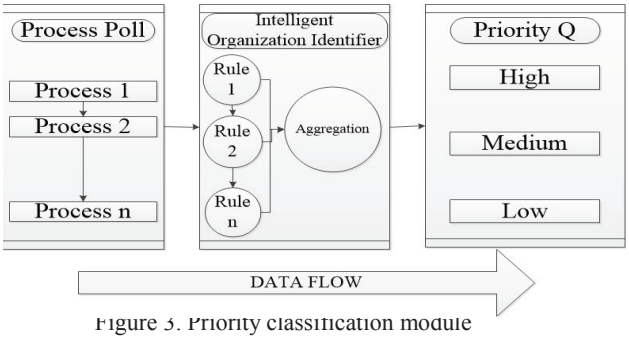

Figure 3. Priority classification module

the organization's liking and disliking. Therefore, if an organization wishes to alter the scheduling policy, the organization must change priority by themselves or they must reset the operating system. Both these tasks are time taking and hard for novice users. Using

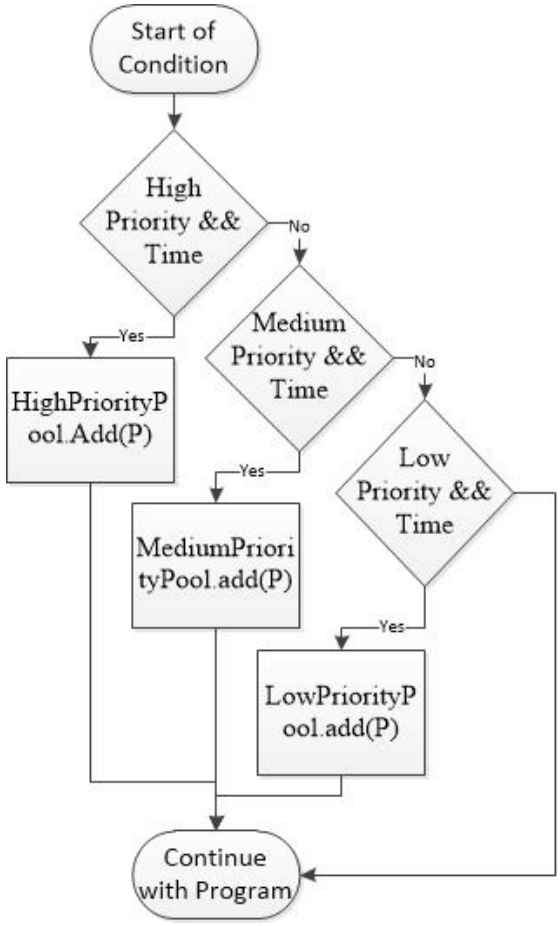

Figure 4. Flowchart of Priority Scheduling in OIPSA

OIPSA, an operating system can keep the users of an organization satisfied without the need of changing priorities or resetting operating system by themselves. The OIPSA only needs to model preferences of organization. If organization is satisfied, the users comes under its umbrella automatically.

F. *Implementation*

We combine our technique of assigning priorities with one of the implemented scheduling algorithms, i.e., Priority Scheduling Algorithm [11]. The flow chart of the proposed OIPSA is provided in Figure. 4.

Table 1: Input for OIPSA

Process	Burst Time	Priority
1	10	5
2	20	2
3	30	3

IV. EXPERIMENTAL RESULTS

In this section, we perform experiments on our proposed OIPSA. We take a set of three processes with different priority values and different burst times. Table. 1 shows the processes and the associated values. For the first experiment, we choose a data source that is suitable for FCFS algorithms. Same is repeated with opposite values and priorities. It could be observed that jobs with the short burst time are scheduled first. In this case, FCFS performs better when jobs with the shorter burst time are first and jobs with the longer burst time are at the end of the queue. This reduces turnaround time, waiting time and response time.

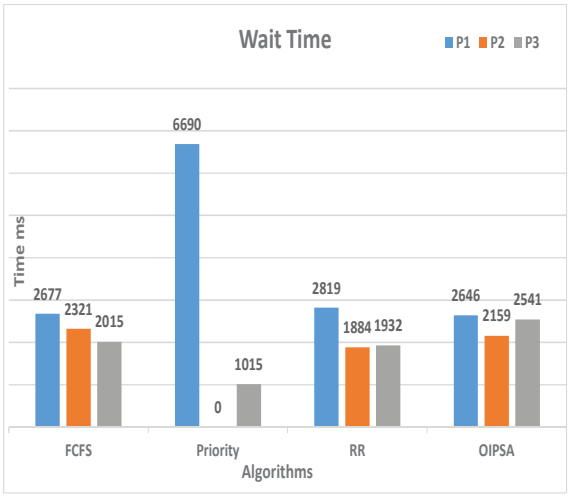

Figure 5. Waiting time of Individual Processes (P=Process)

A. *Waiting time of Individual Processes*

From above chart (Figure. 5), we can see that FCFS has performed pretty well for all the processes, as they have very low waiting time. Round Robin (RR) performed satisfactory for all the processes. If we observe the performance of our proposed OIPSA algorithm, we see the results obtained closely match with that of FCFS algorithm. This is because OPISA doesn't give same quantum time to all the processes. Higher priority process gets more time than lower priority process and that is why its waiting time is very close to the FCFS for this set of input.

B. *Response Time of Individual Processes*

In Figure 6, we can observe that response time of Round Robin (RR) for all the processes is comparatively lowest because of its fair quantum time to all the processes. For Priority algorithm, the processes with more priority get CPU very frequently then the processes with low priority. The response time in FCFS is better as shorter processes are first and longer burst time processes are latter.

If we observe the performance of the proposed OIPSA, it could be noticed that the OIPSA is not even close to RR algorithm. The reason is that OPISA does not give fair quantum time for all the processes. The higher priority will get more quantum time that is why

response time for each process is higher than RR Algorithm.

C. *Turnaround Time of Individual Processes*

From Figure 7, we can see that the turnaround time for the FCFS is satisfactory as short CPU burst job are at start of queue. High CPU burst jobs are at the end of queue. P2 for priority algorithm has the least turnaround time as of lowest priority process. RR algorithm does not perform better when it comes to turnaround time as it believes in fairness.

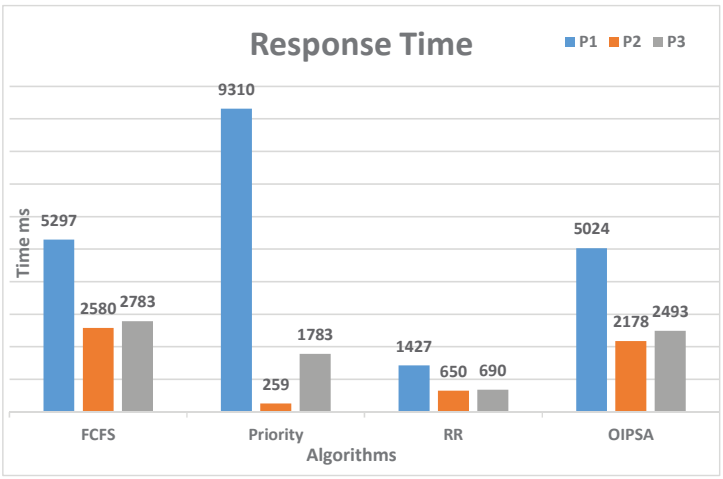

Figure 6.Responce time of Individual Processes (P=Process)

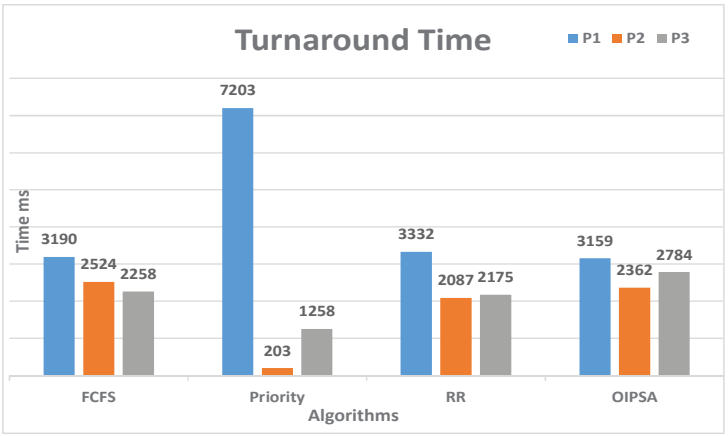

Figure 7. Turnaround Time of Individual Processes (P=Process)

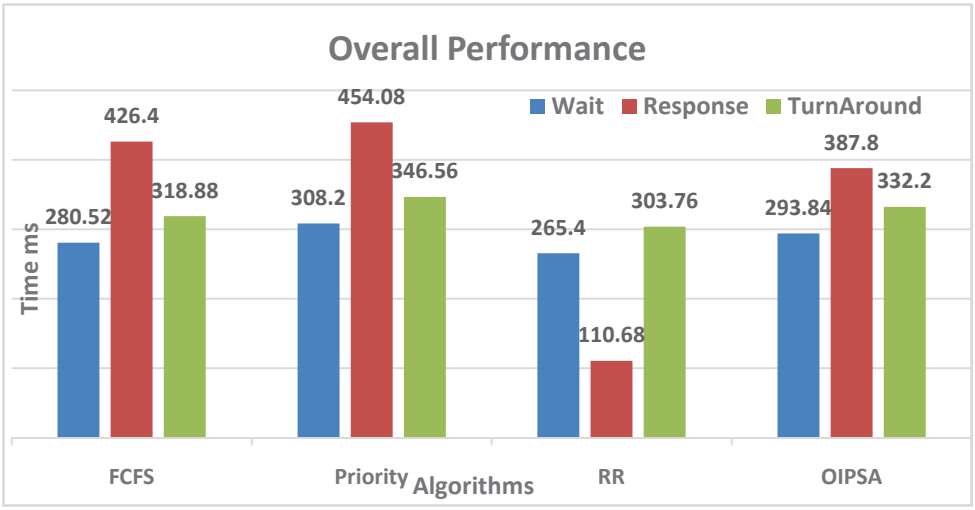

Figure 8. Overall Performance of all Algorithms

Notice the performance of the proposed OPISA in Figure 7, we see that it performed well for P1 when compared with the Priority Algorithm as its priority increased as the time passes. P2 turnaround time is higher than P2 of Priority, as of low priority. In this case too, OIPSA is still very close to the FCFS.

D. Overall Performance of all Algorithms for the same data

The average values of all the experiments for the discussed algorithms have been computed and plotted in Figure 8. It could be observed that on average, FCFS gives optimal values and the proposed OIPSA is pretty close to FCFS.

V. CONCLUSION

In this paper we proposed a novel algorithm which uses organization's preferences as a basis for scheduling. Our proposed algorithm classifies each process into *high*, *medium*, and *low* priority pool and then depicts the organization preferences. Comparing with existing algorithms and user preference algorithms, OIPSA gives better results. It schedules according to organization's needs. User personal preferences are ignored as user comes under the umbrella of an organization. An important point to note here is that the performance of OIPSA at start will be just satisfactory but as OIPSA learns organization's preferences, it will perform much better and will enhance overall efficiency of the system according to organization's needs. In future, we will be adding more preferences. User preference will be encapsulated in organization preferences. We aim to enhance the existing algorithms to meet the organization needs. A hybrid approach comprising of organization's need and user preferences will also form part of our future work.

ACKNOWLEDGMENT

The principal author would like to acknowledge the grant of funding by the Higher Education Commission (HEC), Pakistan to present this paper.

REFERENCES

[1] G. Coulson, G. Blair, and P. Grace, "On the performance of reflective systems software," IEEE Int. Conf. Performance, Comput. Commun. 2004, pp. 763–770.

[2] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating systme concepts",1998.

[3] B.A. Shirazi"Introduction to Scheduling and Load Balancing," 1990.

[4] A. A. Aburas and V. Miho, "Fuzzy Logic based algorithm for uniprocessor scheduling," in 2008 International Conference on Computer and Communication Engineering, 2008, pp. 499–504.

[5] P. K. Varshney, N. Akhtar, and M. F. H. Siddiqui, "Efficient CPU Scheduling Algorithm Using Fuzzy Logic," vol. 47, no. Iccts, pp. 13–18, 2012.

[6] A. M. Wang, "Fuzzy-Based Scheduling Algorithm," pp. 1–12, 2010.

[7] S. Lim and S. Cho, "Intelligent OS Process Scheduling Using Fuzzy Inference with User Models," pp. 725–734, 2007.

[8] J. Nieh, "2001 USENIX Annual Technical Conference," 2001.

[9] "The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Applications." [Online]. Available: http://suif.stanford.edu/papers/nieh97b/paper.html. [Accessed: 06-Apr-2014].

[10] P. Laborie, "Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results," Artif. Intell., vol. 143, no. 2, pp. 151–188, Feb. 2003.

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

[12] U. Schwiegelshohn and R. Yahyapour, "Analysis of first-come-first-serve parallel job scheduling," *SODA*, 1998.