International Journal of Engineering and Techniques - Volume 1 Issue 5, Sep-Oct 2015

RESEARCH ARTICLE OPEN ACCESS

Round Robin Scheduling Algorithm to Control Processes that Grow Non-Deterministically

Hind HazzaAlsharif[1] , Razan Hamza Bawareth[2]
Computer Science Department
[1,2](Faculty of Computing and Information Technology King Abdul Aziz University Jeddah, Saudi Arabia.)

Abstract:

Scheduling is one of the most important fundamental concepts in operating systems. Round Robin (RR) scheduling is one of the most widely used algorithms in multitasking and real time environments [1]. In multitasks environments, it is necessary to select the process among a set of processes. The CPU is allocated to the selected process and controlled by the scheduling algorithm. Each scheduling algorithm has its own needs, advantages and disadvantages. In this paper we have studied RR and focused on Dynamic Round Robin algorithm (DRR) and its impact on the processes, which grow in its burst time in a non-deterministic way with different sets of phenomenon and compared the results.

Keywords **— RR Scheduling, processes, burst time.**

I. INTRODUCTION

RR scheduling algorithm is the most widely used scheduling algorithm due to its fairness and starvation free concepts. Fairness has been a desirable criterion of a schedule ever since concur-rent execution of independently applications be-came possible in time shared systems [2]. RR al-gorithm works in such that the CPU allocated to each process for a time unit. A time quantum is a fixed time for each process in equal portions and in circular order [3]. If the time quantum is too long, the response time is high. However, if the time quantum is too small, this will cause unnec-essarily frequent context switch leading to more overheads resulting in less throughput. The per-formance of the system in DRR depends on the choice of anhalf time quantum, which is dynamic in nature. Apparently, this leads in reducing con-text switching, average waiting time AWT and average turnaround time ATT.

In processes with growing burst time, where a CPU is allocated to a process and if the process is not com-pleted yet, the process has to go back to the queue. Thus the time of the process is going to grow again in a non-deterministic way. Thus, we have to deal with such kind of processes in order to complete its execution with satisfying system performance.

We may have this sort of processes in network with growing buffers, which becomes especially desirable through the increasing use of parallel systems in multiuser environments with the interconnection network shared by several users at the same time. So fair allocation of bandwidth at links within a network is a necessary requirement for ensuring that the performance is not affected when another possibly misbehaving flow tries to send packets at a rate faster than its fair share. In multiuser environments, the protection guaranteed by fair scheduling of packets improves the isolation between users, a quality strongly desired by customers of parallel systems [4]. Many other applications also exist.

II. RELATED WORK

Allison and Celso [5] have discussed a new weighted variant of the minimum carry-over effects value problem in RR. The problem was formulated by integer programming and an algorithm based on the hybridization of the Iterated Local Search (ILS) meta-heuristic with a multi-start strategy. They have obtained numerical results to validate the effectiveness of the hybrid heuristic. Their results confirm successful cases of the hybridization of the ILS meta-heuristic with multi-start strategies.

Fattahand Leung [6] have proposed a new scheduling algorithm for packet cellular networksknown as Wireless Deficit Round Robin (WDRR). This scheduler has low implementation complexity and offers low delay bound, tight fairness index, and good isolation property. In error-prone channels, the algorithm provides short-term fairness among sessions that perceive a clean channel, long-term fairness among all sessions, ability to meet specified throughput objectives for all sessions, and graceful service degradation among sessions that received excess service. Analysis and simulation were used to verify the WDRR properties.

Kanhere and others [7] has presented a simple, efficient and easily implementable scheduling algorithm called Elastic Round Robin (ERR). It is designed to satisfy the unique needs of switching, which is popular in interconnection networks of parallel systems. Their work proved that ERR is efficient with packet work complexity. They analytically derived the relative fairness bound of ERR. They've also derived the bound on the start-up latency experienced by a new flow that arrives at an ERR scheduler. Finally, they have presented simulation results comparing the fairness and performance characteristics of ERR with other scheduling disciplines of comparable efficiency. They found that neither Deficit Round Robin (DRR) nor Surplus Round Robin (SRR) is ideally suitable for use in wormhole networks. While ERR is suitable for use in Internet routers and has better fairness and performance characteristics than previously known scheduling algorithms of comparable efficiency, including DRR and SRR.

Batcher and others [8] have introduced a flexible Dy-namic Round-Robin Scheduling (DRRS) as a flexible framework for improving the performance of multi-tasking embedded systems during run time. The monitoring of the system performance is used to monitor the effect of the task rescheduling. They've achieved incrementally improvements in performance under changing system conditions. Techniques such as DRRS can be very useful in other techniques for improvements that are used in embedded systems.

The proposed method in [9] by Bashir and others, has used a fuzzy logic to compute a suitable time quantum for a given CPU scheduling scenario so, the throughput of the system is not going to decrease due to unnecessarily context switches. Using fuzzy logic methods is computationally efficient; it works well with optimization and adaptive techniques on the way to extract proper knowledge about a data set. The proposed system, which is called Fuzzy Inference System (FIS) used two inputs which are the number of users or processes in the system and the average burst time of the processes in the ready queue, in order to produce the most suitable time quantum for a given CPU scheduling scenario.

In [1], Nayak and othershave proposed a new variant of RR scheduling algorithm known as Improved Round Robin (IRR) Scheduling algorithm. IRR algorithm used the median method to determine the half time quantum with sorting the burst times in specific orders or use it randomly. IRR has showed its significant improvement in RR as it produced better turnaround and waiting times and most important reduced the number of context switching in observable way.

Salil and others in [10], and Wadee and others in [11] have presented a scheduling discipline called Elastic Round Robin (ERR), which is simple, fair, and efficient. They have shown that the work complexity of ERR is O(1) and, therefore, can be easily implemented in networks with large numbers of flows. In comparison to other scheduling disciplines of similar efficiency, such as Deficit Round Robin (DRR) and Surplus Round Robin (SRR), ERR has better fairness properties, as well as a lower start-up latency bound. Among scheduling disciplines of comparable efficiency, DRR and SRR come closest to ERR in fairness. However, neither DRR nor SRR is suitable for use in networks, where the length of time a packet occupies the link is not known before a decision to transmit the packet is made.

III. EXPERIMENT SITTING

Round Robin techniques used under experiment:

The following approaches have been proposed for RR to be used in this study: Fixed Quantum Round Robin (FQRR), Half Quantum Round Robin (HQRR), and Minimum Quantum Round Robin (MQRR). Each process has been tested on different burst times.

Fixed Quantum Round Robin (FQRR):

In RR scheduling algorithm, which is based on a fixed time quantum. The CPU scheduler selects a process from the ready queue and allocates one time quantam to each process. The scheduler selects the first process from the queue, sets a timer to interrupt after one quantum, and dispatches the process. If the process is still running at the end of the quantum, the CPU is preempted and the process is added at the end of the ready queue. If the process is finished before the end of the quantum, the process voluntarily releases the CPU. In either case, the CPU scheduler assigns the CPU to the next process in the ready queue [3].

Half Quantum Round Robin (HQRR):

In this algorithm, we assign a variable time quantum to the process, which is known as dynamic time quantum. This time depends on the burst time of all processes. It is calculated using ceiling of burst time divided by two. For example, if we have 4 processes P1, P2, P3, and P4 and each process have a dynamic burst time. So, the half time quantum will be the $\lceil$burst time/2$\rceil$. Table 1 illustrates the example.

Processes	Burst time	halftime quantam
P1	6	3
P2	8	4
P3	2	1
P4	4	2

Table 1:Illastratinghalf time quantam for HQRR

Minimum Quantum Round Robin (MQRR):

Minimum quantum RR has improved over both fixed quantum round robin and half quantum round robin. It is derived by assigning the minimum burst time in each time unit to all the processes that are in the queue to be a time quantum. In this algorithm time quantum equal minimum burst time for each time unit.

IV. COMPARATIVE STUDY AND RESULTS

In this paper, we have closely observed the behaviour of each of the above techniques. We conducted the test on deterministic (predefined) and non-deterministic (growing) burst time processes. Example (1) is used. The average turnaround time

Techniqe	FQRR	HQRR	MQRR
Deterministic burst time processes	15	17.75	14
Non-deterministic burst time processes	20.75	Infinite loop	25.75

ATT has been calculated for each technique. Table 2 shows the results.

Table 2: Average turnaround time for RR techniques under experiment for deterministic burst time processes and non-deterministic burst time processes.

In the non-determenstic burst time processes scenario, techniques have been applied on processes that grow in non-deterministic way. According to Table 2, we have observed that, FQRR is the best technique, but this result is not guaranteed. Because the growing burst time value is dynamic. Also, we can say that sometimes the MQRR is the best approach for the applied algorithm. But again we cannot guarantee this result as well for the same reason. However, the worst approach apparently is the HQRR with 50% increase in the average turnaraound time, because at each time, since the process will grow as it takes only half of it is needed burst time only, so it wouldn't finish and will enter to an infinite loop which results is a very large system overhead. Figure 1 illastrates the comparative results.

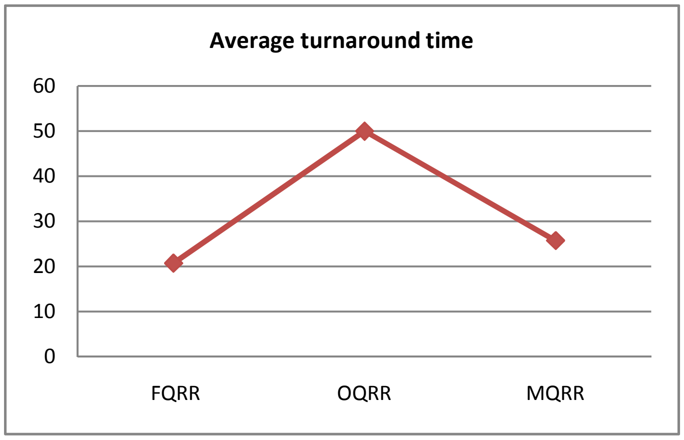

Figure 1: Non-deterministic burst time processes

In the deterministic burst time processes scenario, techniques have been applied on processes with predefined burst time. According to Table 2, we have observed that, MQRR is the best technique. Each time unit is assigned by the minimum time quantum depending on different burst times processes. In this approach, each time unit has at least one process to be finished. On the other hand the HQRR showed the worst performance with infite proceeding loop. Figure 2 illustrates the comparative results.

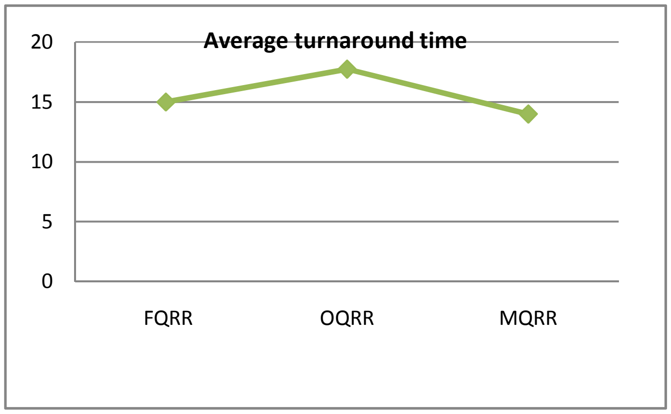

Figure 2: Deterministic burst time processes

V. CONCLUSION

RR is the most common scheduling algorithm. In this paper, we've studied DRR that improved on average turnaround times, and its effect on deterministic and non-deterministic burst time processes. Minimum quantum RR showed the best performance in non-deterministic burst time processes. At each time unit, a minimum burst time process will be assigned as a time quantum to all the processes. Half time quantum RR is the worst technique in deterministic burst time processes. It won't finish and will enter to an infinite loop, which would result in a very large system overhead.

REFERENCES

[1] D. Nayak, S. K. Malla, and D. Debashree, "Improved Round Robin Scheduling using Dynamic Time Quantum,"International Journal of Computer Applications (0957-8887), vol. 38, no. 5, pp. 34-38, Jan 2012.

[2] Sarkar. Arnab, Chakrabarti. Partha P., and Kumar. Rajeev."Frame-Based Proportional Round-Robin". IEEE TRANSACTIONS ON COMPUTERS, VOL. 55, NO. 9, SEPTEMBER 2006.

[3] A. Silberschatez, G. Gagne, and P. Galvin, *Operating System Concepts*, 7th ed., J. Willy, Ed. USA: Willy, 2005.

[4]H.Sethu,C.B.Stunkel,andR.F.Stucke,aI, "IBM RS/6000 SP interconnection network topologies for large systems",o Proc. Int'l Conf. Parallel Processing, Aug. 1998.

[5] Guedes. Allison and Ribeiro. Celso, "A heuristic for minimizing weighted carry-over effects in round robin tournaments", Journal of Scheduling, Vol. 14, Issue 6, pp 655-667, Springer US 2011.

[6] Fattah. Hossam and Leung.Cyril, "An Improved Round Robin Packet Scheduler for Wireless Networks", International Journal of Wireless Information Networks, Vol. 11, No. 1, Kluwer Academic Publishers-Plenum Publishers, January 2004.

[7] Kanhere, S.S. , Sethu, H. , and Parekh, A.B., "Fair and Efficient Packet Scheduling Using Elastic Round Robin", IEEE Transactions on Parallel and Distributed Systems, Vol.13, Issue 3, p(324 – 336), Mar 2002.

[8] Batcher, K.W. and Walker, R.A., "Dynamic Round-Robin Task Scheduling to Reduce Cache Misses for Embedded Systems", Design, Automation and Test in Europe, 2008, 10-14 March 2008.

[9] Doja, M.N. ,Alam, B. , and Biswas, R. , “Finding Time Quantum of Round Robin CPU Scheduling Algorithm Using Fuzzy Logic”, 2008 International Conference on Computer and Electrical Engineering. p(795-798), 2008 IEEE.

[10] Kanhere.Salil S., Sethu. Harish, and Parekh.Alpa B., “Fair and Efficient Packet Scheduling Using Elastic Round Robin”. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 13, NO. 3, MARCH 2002.

[11] W. Alhalabi, M. Kubat, and M. Tapia, "A Tool to Personalize the Ranking of the Documents Returned by an Internet Search Engine", Journal of Convergence Information Technology, vol. 2,no. 3, pp.6-10, 2007.