Advances In CPU Scheduling Algorithms in Operating Systems

Lei Li

School of Software Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi, 710000, China

seohying@stu.xjtu.edu.cn

Abstract. CPU scheduling algorithms significantly influence Operating System (OS) performance. Many studies have proposed optimized scheduling algorithms. This paper outlines the main characteristics of common scheduling algorithms, including First Come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR), Priority Scheduling, and Highest Response Ratio Next (HRRN). It reviews comparative performance research on these algorithms and mentions their optimizations to summarize the progress in this field. This paper also analyses two specific operating systems - Time-Sharing Systems and Embedded Systems, and introduces scheduling algorithms applicable to these two systems and their typical applications. The review finds that SJF & SPTF perform better than FCFS and RR in terms of waiting time, while HRRN also significantly outperforms RR. The optimization of the scheduling algorithm mainly focuses on RR, by clustering techniques and dynamically resizing the quantum time. For time-sharing systems, RR is the most common strategy, while the Linux kernel adopts the Completely Fair Scheduler (CFS). For embedded systems, the real-time scheduling algorithms, Earliest Deadline First(EDF) and Least Slack Time(LST), are applicable to soft real-time systems.

Keywords: CPU scheduling algorithms; Performance; Round Robin; Time-Sharing System; Embedded System.

1. Introduction

Operating System (OS) is program that acts as an intermediary between a user of a computer and the computer hardware. The OS manages the various resources of the computer, including the CPU, the core of the computer, and controls the running of user programmes and the operation of I/O devices. The goal of OS is to make computer systems easy and efficient to use for users. There are many different types of operating systems. Time-Sharing systems are characterized by those multiple users sharing use of the same computer by time slice. Real-time systems are systems with well-defined fixed time constraints. Embedded systems are hardware and software systems that are embedded in a larger device to fulfill a specific function.

A process represents the dynamic execution of a program. When a process is created, it waits to be scheduled into the ready queue, which is called long-term scheduling. After entering the ready queue, the process has to wait for the CPU to be allocated, which is called short-term scheduling (or CPU scheduling). The scheduler is a program of the operating system that performs scheduling. CPU scheduling determines which process or thread runs when, which is critical for obtaining maximum CPU utilization with multiprogramming.

The scheduler's scheme for selecting processes is called scheduling algorithm. A scheduling algorithm is non-preemptive if once it assigns CPU to a process, it only de-allocates CPU when the process finishes. Conversely, a scheduling algorithm is preemptive if it may de-allocates CPU from a process when the process is running. Typical scheduling algorithms include First Come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR), etc.

CPU scheduling algorithms play a critical role in determining the overall performance, efficiency, and fairness of resource allocation within a computing system. In recent years, many studies have been devoted to the optimisation of scheduling algorithms. Hark outlines the proposed scheduling strategies which are based on the typical scheduling algorithms [1]. Other researchers propose the optimised versions of RR, for RR plays an important role in the time-sharing system [2,3,4,5]. In terms of specific systems. Gao proposed optimised CFS for time-sharing systems [6]. Kamoyedji et al proposed dynamic job scheduling under User-PC computing systems [7]. While Wang et al

proposed optimisation of real-time scheduling algorithms in embedded systems [8,9,10]. In addition, some researchers have analysed and experimented on the performance of several scheduling algorithms [11,12,13]. As the complexity of modern computer architectures and the diversity of workloads continue to evolve, the need for advanced and adaptable scheduling techniques has become increasingly important.

This paper introduces common CPU scheduling algorithms in operating systems and discuss their main characteristics and performance, as well as summarizes and analyzes the recent research. For some specific systems, such as time-sharing system, some scheduling algorithms from recent research are presented. Finally, the deficiencies in the current research on CPU scheduling algorithms and future research trends are discussed to provide reference for the next research in this field.

2. Overview and analysis of CPU scheduling algorithms

2.1. Outline of CPU scheduling algorithms

Common CPU scheduling algorithms include FCFS, SJF, RR, and priority scheduling for a single ready queue.

FCFS is the simplest scheduling algorithm, which selects the first arriving process for execution. FCFS is a typical non-preemptive strategy, where the Processes are executed in the same order as the processes arrive [1]. However, FCFS has relatively long average job waiting time [11].

In Round Robin, each process gets a small unit of CPU time (time quantum). After this time has elapsed, the process is preempted and added to the end of the ready queue. RR is a common preemptive strategy. The performance of RR is related to the time quantum size [2]. When the time quantum size is large (when most processes can finish in one time quantum), RR is close to FCFS; when the time quantum size is small, RR has a shorter response time, but it also adds extra time overhead for context switching [2].

SJF selects the process with shortest CPU burst time for execution. The preemptive SJF is also known as SRTF. Under ideal conditions, SJF is considered the optimal scheduling algorithm, for it gives the shortest average waiting time for a given set of processes. It moves the short process before the long process and the decrease in the waiting time of the short process is greater than the increase in the waiting time of the long process [14]. However, the ideal SJF is difficult to implement because the size of the job is hard to learn before running the job and can only be estimated.

Priority Scheduling associates a priority number with each process, and selects the process with the highest priority for execution (e.g., SJF is priority scheduling with the length of the next CPU burst as the priority number). Static priority is determined at the time of process creation and remains constant throughout its lifetime, but can lead to starvation---low priority processes may never execute [1]. Dynamic priority means that the priority of a process can be changed as the process advances based on factors such as the process's wait time, resource usage, etc., in order to obtain better scheduling performance.

Highest Response Ratio Next (HRRN) is implemented based on the dynamic priority algorithm, which prioritizes the response ratio (RT) and select the process with the highest RT for execution, where the response ratio is defined as:

$$\text{Response Ratio}=\frac{\text{waiting time of a process so far + estimated run time}}{\text{estimated run time}}=1+\frac{\text{waiting time of a process so far}}{\text{estimated run time}}. \quad (1)$$

The priority of long jobs in HRRN increases as the waiting time increases to avoid starvation [15].

In addition, in systems with multiple ready queues, strategies such as multilevel queue scheduling and multilevel feedback queue are used for scheduling.

Multilevel queue scheduling assigns jobs to different queues, each with its own scheduling policy and priority. Scheduling between queues is usually based on a fixed priority, or according to the time quantum allocated.

Multilevel feedback queue (MLFQ) manages multiple queues with different priority levels, and the priority of jobs can be dynamically adjusted according to their behaviors, with jobs completed in a shorter period of time maintaining a high priority, and jobs running for a longer period of time gradually decreasing in priority [1].

2.2. Performance analysis of CPU scheduling algorithms

For the performance of scheduling algorithms, the main criteria include CPU utilization, which is the ratio of CPU busy time in total time; throughput, defined as the number of processes completed per time unit; turnaround time, which is the time interval from submission to completion; waiting time, referring to the time a process waits in the ready queue for scheduling; and response time, which is the time interval from the submission of the request to the first response occurrence.

In some recent researches on the performance of scheduling algorithms, the above metrics are commonly used for comparative studies, while the results of simulations and comparisons show the relative performance advantages and disadvantages of the common scheduling algorithms.

Pemasinghe et al.'s research introduced the simulation of FCFS, RR, SJF, SRTF, Priority Scheduling. Among the result of the simulation, SJF&SPTF have the shortest average turnaround time and waiting time, while RR has the longest average turnaround time and waiting time [11].

Tufegdžić et al. tested FCFS, SJF, SPTF, RR using object-oriented programming, with two sets with the same and different process arrival times, respectively. The results for both sets indicated that SJF&SPTF performs the best, while RR had the longest average turnaround time and waiting time as well [12].

Bibu et al. made a comparative study of FCFS and SJF by generating process sequences and presented the results by graphs, which also shows that SJF significantly outperforms FCFS in terms of average turnaround time and waiting time [13].

A comparison test between HRRN and RR by Benny et al. with three given processes cases shows that the average waiting time with HRRN scheduling is quite shorter than the RR. The researchers concluded from their experiments that the reason for the phenomenon is that RR scheduling interrupts jobs and is not specific to each process, while HRRN assigns priority to jobs based on their attributes [15].

2.3. Recent advancement of scheduling algorithms

Much research has recently focused on proposing and improving scheduling techniques. Among the various scheduling algorithms, Round Robin are one of the most extensively studied and applied techniques, for its importance in operating systems such as time-sharing systems.

Mostafa et al. proposed a optimised RR scheduling which cluster similar processes in the queue by K-means clustering, and allocate the same time quantum to the processes under the clustering [2].

In terms of time quantum size adjustment, Sharma et al. suggested the dynamic calculation of time quantum size from the median and average of CPU burst times of the processes, successfully reducing average waiting time and turnaround time [3]. Shafi et al. proposed ADRR which dynamically adjusts time quantum size according to the CPU burst time, and evaluated it to find that ADRR is better than RR, IRR, OMDRR, PRR in terms of performance 4. Moreover, Fiad et al. provides an improved model for calculating time quantum sizes, and proved it improves the performance [5].

3. Scheduling algorithms for specific systems

3.1. Time-Sharing System

Time-sharing systems are based on the mechanism of time slices, which divide CPU time into units and distribute them to terminals. The terminal starts running when it gets the CPU and interrupts running when it runs out of time slices. Time-sharing systems allow multiple users to share computers, foreground programs (input focus of the terminal), and background programs (not interacting with users) to run simultaneously, and are the prototype for many modern operating systems.

Round Robin is the most common scheduling strategy in time-sharing systems, because it also uses a time-slice mechanism, allocating equal time intervals to each process. The strategies are mentioned in section 2.3 of this paper, are improved Round Robin available for time-sharing systems.

In addition to RR, Completely Fair Scheduler (CFS) is also used in time-sharing systems. In CFS, processes in the queue are set with a vruntime that will grow as the process runs, while the vruntime grows slower for processes with higher priority. The queue sorts the processes in the red-black trees by the vruntime, hence processes can be allocated fairly based on running conditions and priorities.

A typical application of CFS is as the default scheduler for Linux, since it has been introduced to the Linux kernel in version 2.6.23, 2007 [16]. The lastest Linux scheduler is modularly designed to handle particular scheduling tasks through different classes of schedulers, CFS is one of the scheduling classes. But CFS has the drawback of contention on hardware resources. Gao Z discussed the shortcomings of CFS and the problem of the multi-resource fair scheduling, and suggested the Dominant Resource Fair Queueing Scheduler (DRFQS) as an improvement [6].

3.2. Embedded System

Scheduling algorithms for embedded systems are often constrained in terms of resources, such as limited processor computing capacity, memory, and power. These constraints require the design of scheduling algorithms to balance computational complexity, memory utilization, energy consumption. Some systems also have more sensitive time requirements, and scheduling for these systems is known as real-time scheduling. Typical real-time scheduling algorithms in embedded systems include Earliest Deadline First (EDF), RR, Least Slack Time (LST), and Genetic Based Adaptive Scheduling [17]. FCFS, SJF, MLFQ, CFS are common policies of non-real time scheduling in embedded system.

Earliest Deadline First (EDF) schedules jobs by their deadlines and can dynamically adjust priorities of jobs. The primary job of the EDF is to ensure the schedulability of the system, i.e., to ensure that tasks satisfy time constraints. Wang et al. proposed the IEDF (Improved Earliest Deadline First) algorithm based on the queuing theory model and EDF, and proved that IEDF has improved in terms of performance [8]. On the other hand, energy consumption is also an issue for EDF. El Ghor H et al. suggested the energy saving-dynamic voltage and frequency scaling (ES-DVFS) base on the dynamic voltage and frequency scaling (DVFS) technique, which is a optimality of EDF. This algorithm aims to decrease the CPU energy consumption [9].

EDF is adopted in many microprocessors. Interlocked-Hardware-Microkernel Plasma (IHM-Plasma), is a microprocessor which uses hardware to implement the EDF. Krause et al. showed that the IHM-Plasma processes about 174% more tasks per second than the original Plasma which running the EDF algorithm by software, which is a large benefit for hard real-time systems (with strict time limits) [18].

Least Slack Time (LST) prioritizes the job the least slack time, where slack time is defined as the deadline minus the remaining processing time. This means that LST selects the most urgent jobs to process first. EDF and LST are more appropriate for scheduling of process in soft real-time operating system. Teraiya et al. tested EDF and LST by a periodic task set, and found that the EDF and LST perform well in underloaded situations and conversely not perform well in overloaded situations [10].

4. Conclusion

In this paper, various standard scheduling algorithms are summarized and assessed, to find out their main charateristics. The main criteria for evaluating performance are introduced, and some studies on the performance of scheduling algorithms are reviewed. These studies mainly illustrate that SJF & SPTF have better performance than FCFS and RR in terms of average turnaround time and waiting time, while HRRN has a shorter average waiting time compared to RR. Improvements to common scheduling algorithms mainly focus on RR, by clustering similar processes and dynamically adjusting time quantum sizes.

This paper discusses the features of two popular Oses - Time-Sharing Systems and Embedded Systems - and states out the scheduling algorithms applicable to them. RR is the most common strategy for time-sharing systems, while CFS is applied to the Linux kernel with the drawback of contention on hardware resources. For embedded systems, special real-time scheduling algorithms, such as EDF and LST, are applicable to many soft real-time systems.

This paper found that current research on performance comparison of scheduling algorithms is usually based on a given sequence, and focuses on waiting time and turnaround time. Future work needs to be based more on simulations with random process sequences and on other metrics such as CPU utilization. Other scheduling algorithms, including SJF, HRRN, etc., need to be paid more attention to for optimization of scheduling algorithms. This paper also has a shortcoming in discussing the performance and application of scheduling algorithms in time-sharing and embedded systems.

References

[1] Harki N, Ahmed A, Haji L. CPU scheduling techniques: A review on novel approaches strategy and performance assessment[J]. Journal of Applied Science and Technology Trends, 2020, 1(1): 48-55.

[2] Mostafa S M, Amano H. Dynamic round robin CPU scheduling algorithm based on K-means clustering technique[J]. applied sciences, 2020, 10(15): 5134.

[3] Sharma C, Sharma S, Kautish S, et al. A new median-average round Robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time[J]. Alexandria Engineering Journal, 2022, 61(12): 10527-10538.

[4] Shafi U, Shah M A, Wahid A, et al. A novel amended dynamic round robin scheduling algorithm for timeshared systems[J]. Int. Arab J. Inf. Technol., 2020, 17(1): 90-98.

[5] Fiad A, Maaza Z M, Bendoukha H. Improved version of round robin scheduling algorithm based on analytic model[J]. International Journal of Networked and Distributed Computing, 2020, 8(4): 195-202.

[6] Gao Z. Multi-resource Fair Scheduler in Linux[D]. University of Waterloo, 2022.

[7] Kamoyedji A, Funabiki N, Htet H, et al. A proposal of dynamic job scheduling algorithm considering CPU core utilization for User-PC computing System[C]//2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW). IEEE, 2021: 268-271.

[8] Wang Y, Zhou K, Wang Z, et al. Research on real-time embedded software scheduling model based on edf[J]. IEEE Access, 2020, 8: 20058-20066.

[9] El Ghor H, Aggoune E H M. Energy efficient scheduler of aperiodic jobs for real-time embedded systems[J]. International Journal of Automation and Computing, 2020, 17: 733-743.

[10] Teraiya J, Shah A. Analysis of dynamic and static scheduling algorithms in soft real-time system with its implementation[C]//Soft Computing: Theories and Applications: Proceedings of SoCTA 2018. Springer Singapore, 2020: 757-768.

[11] Pemasinghe S, Rajapaksha S. Comparison of CPU scheduling algorithms: FCFS, SJF, SRTF, Round Robin, priority based, and multilevel queuing[C]//2022 IEEE 10th Region 10 Humanitarian Technology Conference (R10-HTC). IEEE, 2022: 318-323.

[12] Tufegdžić M, Jevremović V, Petrović Z. Estimation of CPU Scheduling Algorithms Efficiency Using Object Oriented Programming[J]. quantum, 2022, 1: 4.

[13] Bibu G D, Nwankwo G C. Comparative analysis between first-come-first-serve (FCFS) and shortest-job-first (SJF) scheduling algorithms[J]. 2019.

[14] Putra T D. Analysis of Preemptive Shortest Job First (SJF) Algorithm in CPU Scheduling[J]. International Journal of Advanced Research in Computer and Communication Engineering, 2020, 9(4): 41-45.

[15] Benny R, Wirawan I. Comparison analysis of round robin algorithm with highest response ratio next algorithm for job scheduling problems[J]. International Journal of Open Information Technologies, 2022, 10(2): 21-26.

[16] Chen J, Banerjee S S, Kalbarczyk Z T, et al. Machine learning for load balancing in the linux kernel[C]//Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems. 2020: 67-74.

[17] Kabilesh S K, Stephensagayaraj A, Anandkumar A, et al. Resemblance of Real Time Scheduling Algorithms for Real Time Embedded Systems[J]. Journal of Optoelectronics and Communication, 2020, 2(3).

[18] Krause I, Dantas L P, Gimenez S P. Impact in the Parallel Processing of IHM-Plasma Using the Earliest-Deadline-First Algorithm for the Task-Scheduler Realized by Hardware[J]. Journal of Integrated Circuits and Systems, 2023, 18(1): 1-8.