A Research Survey and Analysis for CPU Scheduling Algorithms using Probability-Based Study

Shweta Jain¹, Saurabh Jain²

¹Assistant Professor, Department of Computer Applications, Shri R.G.P. Gujarati Professional Institute, Indore, INDIA
²Associate Professor, Department of Computer Applications, Shri Vaishnav Institute of Technology & Science, Indore, INDIA

ABSTRACT

Operating System (OS) is also known as resource manager because its prime responsibility is to manage the resources of the computer system. An operating system controls all the operation that constantly and continuously manages the resources available around the system in optimum way. Scheduling is a fundamental and most important OS function which is essential to an operating system’s design that managing multiple queues of processes in order to minimize delay and to optimize performance of the system. CPU Scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. Hence, the performance of system depends on which scheduling algorithm used. In this paper, we will discuss the various approaches that can be used for this purpose and elaborate the research analysis in this field. A probability-based Markov chain analysis is done in order to determine the performance of these algorithms.

Keywords---- Markov Chain, Scheduler, Scheduling, Stochastics etc.

I. INTRODUCTION

The scheduling is a methodology of managing multiple queues of processes in order to minimize delay and to optimize performance of the system in the environment where queues of processes exist with servers. Scheduling refers to set of rules, policies and mechanism that govern the order in which resource is allocated to the various processes and the work is to be done. A scheduler is an OS module that implements the scheduling policies and its primary objective is to optimize the system performance according to the criteria set by the system designers. The Scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them. Different CPU scheduling algorithms have different properties, and the choice of a particular
algorithm may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Many criteria have been suggested for comparing these algorithms that characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following:

- Utilization/Efficiency: keep the CPU busy 100% of the time with useful work or maximum utilization of CPU.
- Throughput: maximize the number of jobs processed per hour.
- Turnaround time: from the time of submission to the time of completion, minimize the time batch users must wait for output.
- Waiting time: Sum of times spent in ready queue, minimize it.
- Response Time: time from submission till the first response is produced, minimize response time for interactive users.
- Fairness: make sure each process gets a fair share of the CPU.

We present scheduling approaches which lead to improvements in overall system performance, and allow the system to achieve graceful degradation as system load increases.

III. SCHEDULING OF PROCESSES

One of the fundamental functions of an operating system is scheduling in which sharing of computer resources between multiple processes. Processor time is also a resource that is costlier than other resources and that’s why proper attention to paid to use the processor time efficiently. So what kind of scheduling algorithm can be there for better utilization of processor time and in which situation which algorithm is better to use. The intention should be allowed as many as possible running processes at all time in order to make best use of CPU. CPU scheduling has strong effect on resource utilization as well as overall performance of the system in a way that meets system objectives. These scheduling activities are performed by schedulers who are having three separate functions: a long term (an admission or job or high level scheduler), a mid-term or medium-term and a short-term (dispatcher or CPU scheduler) that relates time scale with performing which functions. The long-term or admission scheduler decides which jobs or processes are to be admitted to the ready queue; that is, a new process is created. This is a decision to add a new process to a set of processes that are currently active. The mid-term scheduler decides "swapping of processes out" or "swapping in" (also referred as "paging out" or "paging in"). The short-term scheduler (also known as the CPU scheduler) decides which of processes in the ready queue, Thus the short-term scheduler makes scheduling decisions much more frequent than the long-term or mid-term schedulers. This scheduler can be preemptive, implying that it is capable of forcibly removing processes from a CPU when it decides to allocate that CPU to another process, or non-preemptive (also known as "voluntary" or "co-operative"), in that case the scheduler is unable to force processes off the CPU. Fundamentally, scheduling is a matter of managing queues to minimize queuing delay and optimize performance of queuing environment.

IV. SCHEDULING ALGORITHMS

Basically, there are two types of algorithms: Preemptive and Non-preemptive.

- In preemptive, scheduling decision can be taken on run time where high priority process takes more attention that is advantageous in deadlock. In non-preemptive, no preference is given to higher priority process that treats to the all process fairer but short jobs have to wait for longer jobs to be completed. The other commonly used criteria can be categorized along two directions that can make a distinction between user-oriented and system-oriented criteria.

1. PRIORITY-BASED

Each process is assigned a certain level of priority that corresponds to the relative importance of the event that it services. The processor is normally allocated to the highest-priority process among those that are ready to execute. Higher-priority processes usually preempt execution of the lower-priority processes. This form of scheduling, called priority-based preemptive scheduling, is used by a majority of real-time systems. As a job is waiting, raise its priority so eventually it will have the maximum priority which is a solution to the problem of starvation. If there are many processes with the maximum priority, it uses FCFS among those with max priority (risks starvation if a job doesn’t terminate) or can use RR.

2. FIRST-COME FIRST-SERVED

First-Come, First-Served (FCFS) is the simplest CPU scheduling algorithm in the concept of first-in first-out (FIFO) or a strict queuing scheme in which the process that requests the CPU first is allocated the CPU first. This is non preemptive policy in which implementation of the policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. As each process becomes ready, it joins the ready queue. In the queuing model, turnaround time is waiting time plus service time.

3. ROUND ROBIN

RR scheduling is often regarded as a "fair" scheduling discipline. It is also one of the best-known scheduling disciplines for achieving good and relatively
evenly distributed terminal response time. It reduces the penalty that jobs suffer with FCFS is to use preemption based on a clock. A clock interrupt is generated at periodic intervals. The performance of round robin scheduling is very sensitive to the choice of the time slice. For this reason, duration of the time slice is often made user-tunable by means of the system generation process. The time interval between each interrupt may vary. Generally RR is effective in interactive environments such as time-sharing systems or transaction processing system. One drawback of RR is its relative treatment of processor-bound and I/O-bound processes.

4. SHORTEST PROCESS NEXT
Another approach, shortest process next (SPN) policy, which is non preemptive to reduce the bias in favor of long processes. Main idea behind is that if CPU is free to allocate to the process having small burst time or shortest expected processing time is selected next in which the short process will jump to the head of the queue past longer jobs. Therefore this scheduling policy is also referred by Shortest Job First (SJF) in which the jobs are sorted on the basis of total execution time needed and then it runs the shortest job first. It has the shortest average waiting time. Hence decreases the total waiting for all jobs and hence decreases the average waiting time as well. If two processes have same burst time then FCFS technique is applied. The shorter the job, the better service it will provide. It is optimal algorithm among the all algorithm because of less average waiting time. This scheduling policy can starve processes that require a long burst. So it proves beneficial in long-term-scheduling.

5. SHORTEST REMAINING TIME NEXT
SRTN is a provably optimal scheduling discipline in terms of minimizing the average waiting time of a given workload. SRTN scheduling is done in a consistent and predictable manner, with a bias towards short jobs. With the addition of preemption of SPN/SJN, an SRTN scheduler can accommodate short jobs that arrive after commencement of a long job. Preferred treatment of short jobs in SRTN tends to result in increased waiting times of long jobs in comparison with FCFS scheduling, but this is usually acceptable. The SRTN discipline schedules optimally assuming that the exact future execution times of jobs or processes are known at the time of scheduling. Even more detailed knowledge of the duration of each individual processor burst is required. Dependence on future knowledge tends to limit the effectiveness of SRTN implementations in practice, because future process behavior is unknown in general and difficult to estimate reliably, except for some very specialized deterministic cases. Predictions of process execution requirements are usually based on observed past behavior, perhaps coupled with some other knowledge of the nature of the process and its long-term statistical properties, if available.

6. MULTI-LEVEL QUEUE

The scheduling policies discussed so far are more or less suited to particular applications, with potentially poor performance when applied inappropriately. One should use in a mixed system, with some time-critical events, a multitude of interactive users, and some very long non-interactive jobs. One approach is to combine several scheduling disciplines. A mix of scheduling disciplines may best service a mixed environment, each charged with what it does best. One way to implement complex scheduling is to classify the workload according to its characteristics, and to maintain separate process queues serviced by different schedulers. This approach is often called multiple-level queues (MLQ) scheduling. This discipline maintains responsiveness to external events and interrupts at the expense of frequent preemption’s. An alternative approach is to assign a certain percentage of the processor time to each queue, commensurate with its priority. MLQ scheduling may also impose the combined overhead of its constituent scheduling disciplines.

7. MULTI-LEVEL FEEDBACK QUEUE
In a MLQ scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between the queues. Multiple queues in a system may be used to increase the effectiveness and adaptiveness of scheduling in the form of multiple-level queues with feedback. Multi-level feedback queue (MLFQ) scheduling, however, allows a new process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a process uses too much CPU time, it will be moved to a lower priority queue. This scheme leaves I/O-bound and interactive processes in the higher priority queues. Similarly, a process that waits too long in a lower priority queue may be moved to a higher priority queue. Rather than having fixed classes of processes allocated to specific queues, the idea is to make traversal of a process through the system dependent on its run-time behavior.

V. PROBABILISTIC BASED MODELS

Probabilistic-based models will be applied to different kind of scheduling schemes with simulation study. In a probabilistic model, we make two assumptions with the required model:
1. Movement to the next state is entirely dependent on the current state.
2. We can make a suitable estimate of the transition probabilities between states.

Types of Model
Stochastic Models are used to compare alternative decisions. Independence of random variables is a very restrictive assumption in stochastic modeling.

**Stochastic process**

The set of possible values of an individual random variable $X_n$ (or $X(t)$) of a stochastic process $\{X_n, n \geq 1\}$, $\{X(t), t \in T\}$ is known as state space. The state space is discrete if it contains a finite or a denumerable infinity of points; otherwise, it is continuous. For example, if $X_n$ is the total number of sixes appearing in the first $n$ throws of a die, the set of possible values of $X_n$ is discrete. We can write $X_n = Y_1 + \ldots + Y_n$, where $Y_i$ is discrete random variable denoting the outcome of the $i^{th}$ throw and $Y_i = 1$ or 0 according as the $i^{th}$ throw shows six or not. Secondly, consider $X_n = Z_1 + \ldots + Z_n$, where $Z_i$ is the continuous random variable assuming values in $[0, \infty)$. Here the set of possible values of $X_n$ is the interval $[0, \infty)$, and so the state space of $X_n$ is continuous.

**Definition:**

If a stochastic process in discrete time may be discrete state space. A stochastic process with state variable $X_t$ is said to possess the Markov Property if

$$P(X_{t+1} = i_{t+1}|X_0 = i_0, X_1 = i_1, \ldots, X_t = i_t) = P(X_{t+1} = i_{t+1}|X_t = i_t)$$

1. The probabilities that a stochastic process moves to a new state depends only on the current state;

2. The probabilities are independent of all past events.

So this transition probabilities is referred by Markov Chain Model.

**Markov Chain**

Consider a simple coin tossing experiment for a number of times. The possible outcome of each trial is two: head with probability, say $p$, and tail with probability $q$, $p + q = 1$. Let us denote head by 1 and tail by 0 and random variable denoting the result of $n^{th}$ toss by $X_n$. Then for $n = 1, 2, 3, \ldots$, $Pr\{X_n = 1\} = p$, $Pr\{X_n = 0\} = q$.

There is a sequence of random variables $X_1, X_2, \ldots$ The trails are independent and the result of $n^{th}$ trial does not depend in any way on the previous trial numbered 1, 2, \ldots $(n-1)$.

Consider now the random variable given by the partial sum $S_n = X_1 + \ldots + X_n$. The sum $S_n$ gives the accumulated number of heads in the first $n$ trials and its possible values are 0, 1, \ldots, $n$. We have $S_{n+1} = S_n + X_{n+1}$. Given that $S_n = j$ ($j = 0, 1, 2, \ldots, n$), the random variable $S_{n+1}$ can assume only two possible values: $S_{n+1} = j$ with probability $p$; these probabilities are not at all affected by the values of the variables $S_1, \ldots, S_n$. Thus $Pr\{S_{n+1} = j+1|S_n=j\} = p$ and $Pr\{S_{n+1} = j|S_n=j\} = q$.

**Definition:** The stochastic process $\{X_n, n=0, 1, 2\ldots\}$ is called Markov chain, if, for $j, k, j_1, \ldots, j_{n-1} \in \mathbb{N}$ (or any subset of $\mathbb{N}$), $Pr\{X_n = k / X_{n-1} = j, X_{n-2} = j_1, \ldots, X_0 = j_{n-1}\} = Pr\{X_n = k / X_{n-1} = j\} = p_{jk}$ (say) Whenever first member is defined.

**Transition Probability Matrix:**

The transition probabilities $p_{jk}$ satisfy $p_{jk} \geq 0$, $\sum_k p_{jk} = 1$ for all $j$.

These probabilities may be written in the matrix form

$$P = X^{(n-1)}$$

$$P = \begin{bmatrix}
    p_{11} & p_{12} & p_{13} & \cdots \\
    p_{21} & p_{22} & p_{23} & \cdots \\
    p_{31} & p_{32} & p_{33} & \cdots \\
    \vdots & \vdots & \vdots & \ddots
\end{bmatrix}$$

This is called the transition probability matrix of the Markov chain. The $P$ is a stochastic matrix, i.e. a square matrix with non-negative elements and unit row sums.

**VI. LITERATURE REVIEW**

The operating system plays a major role in managing processes arriving in the form of multiple queues. The arrival of a process is random along with their different categories and types. All these require scheduling
algorithms to work over real time environment with special reference to task, control and efficiency. Jain et al. (2015) presented a Linear Data Model Based Study of Improved Round Robin CPU Scheduling algorithm with features of Shortest Job First scheduling with varying time quantum. Chavan and Tikekar (2013) derived a new CPU scheduling algorithm called an Optimum Multilevel Dynamic Round Robin Scheduling Algorithm, which calculates intelligent time slice and changes after every round of execution. Suranauwarat (2007) used simulator in operating system to learn CPU scheduling algorithms in an easier and a more effective way. Sindhu et al. (2010) proposed an algorithm which can handle all types of process with optimum scheduling criteria. Hieh and Lam (2003) discussed smart schedulers for multimedia users. Saleem and Javed (2000) developed a comprehensive tool which runs a simulation in real time, and generates useful data to be used for evaluation which is useful for the design and development of modern operating systems, for measuring the performance of different scheduling algorithms and for the understanding and training of students.

Shukla et al. (2009) incorporated only three processors along with three queues and the procedure of thread scheduling is examined in light of Markov chain model. A simulation study is incorporated to support the findings. Shukla and Jain (2007 a) have a discussion on the use of Markov chain model for multilevel queue scheduler in an operating system. Shukla and Jain (2007 b) proposed a data model based Markov chain model to study the transition phenomenon and a scheduling scheme is designed and compared through deadlock-waiting index measure with simulation study. Shukla et al. (2010 a) proposed a Markov chain model for a general setup of multi level queue scheduling and evaluate Performance. Shukla et al. (2010 b) took Data model approach and Markov Chain based analysis of multilevel queue scheduling and the scheduler is assumed to perform random movement on queue over the quantum of time. Shukla et al. (2010 c) presented a new CPU scheduling scheme in the form of SL Scheduling which is found useful and effective to estimate the total processing time of all the processes present in ready queue waiting for their processing using a numerical study. Shukla et al. (2010 d) designed a new scheduling scheme Group Lottery Scheduling (GLS) in multi-processor environment to predict ready queue processing time using two variants involved Type-I and Type-II allocation of jobs whose variabilities are compared. Shukla et al. (2010 e) proposed a Data Model Approach in State Probability Analysis of Multi-Level Queue Scheduling and evaluated performance effectiveness.

Shukla et al. (2011) used probability models to get estimates of system parameters in multiprocessors environment to overcome the queue length using lottery scheduling. Shukla and Ojha (2010) proposed a data model based Markov chain model to study the transition phenomenon and a general class of scheduling scheme is designed through a proposed deadlock index measure to evaluate by simulation study. Shukla et al. (2009) used a model based study to compare several scheduling schemes of the class and defined an index measure in the mixture of FIFO and round robin algorithms. Sisodia, and Garg (2011) are used a Markov chain model to compare several scheduling schemes which is a mixture of FIFO and round robin in terms of model based study approach using system simulation procedure. Shukla and Jain (2012) presented an efficient method to predict about total time needed to process the entire ready queue if only few are processed in a specified time. Confidence internals are calculated based on PPS-LS and compared with SRS-LS. The PPS-LS found better over SRS-LS.

Naldi (2002) presented a Markov chain model for understanding the internet traffic sharing among various operators in a competitive market. Medhi (1991) elaborated study of a variety of stochastic processes and their uses in the queues management, various field applications. He also developed a Markov chain model for the study of uncertain rainfall phenomenon. Some other useful contributions are for different CPU scheduling by Silberschatz and Galvin (2010), Stalling (2004), Tanenbaum and Woodhull (2000) and Dhamdhere (2009).

VII. SCOPE OF PROPOSED STUDY

In OS, there are different algorithms but all algorithms have their own deficiency and limitations. In this research, some probabilistic models will be proposed to improve the performance of different CPU scheduling. This research proposed new approaches for scheduling algorithm which helps to improve the efficiency of CPU using a probability model based study in the environment of different scheduling, assuming the random movement of scheduler over various processes and queues. The proposed work will be useful for the operating system designers by incorporating some new schemes in currently available scheduling algorithms. It is also possible that some algorithms can be merged to get the optimized performance of operating system. Where the performance of different CPU scheduling schemes will be analyzed under probability based models. Simulation study will support this proposed discussion.

VIII. CONCLUSION

In this paper the researcher introduces the concept of CPU Scheduling algorithms and states that how Probability-based model may be applied on it for setting up approaches and trend. After reviewing the available literature that explored some important issues and challenges associated with CPU Scheduling algorithms. This review work certainly will be helpful for the
upcoming researchers who want to carry on their research in the application of Operating Systems and measures its performance.

REFERENCES