Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 6 (15): 3164-3166, 2006
ISSN 1812-5654

Dynamically Changing Priorities Algorithm for CPU Scheduling

[1]M. Raghunadh Acharya, [2]Ashok Sahai, [3]Ravi Gedela and [3]Srinivas Mukundam
[1]Department of Statistics, ICFAI Business School, Hyderabad, India
[2]Departments of Mathematics and Computer Science, The University West Indies,
St. Augustine, Trinidad and Tobago
[3]Satyam Computer Services, Hyderabad, India

Abstract: Many algorithms are developed and have been used for Scheduling the Central Processing Unit (CPU) in a typical Multiprocessing Environment. In this research a study is carried on how recently processed requests can be taken up for deciding the priorities (or) order for scheduling the CPU Processing. The results are found to be very favorable, as per our simulation study.

Key words: Stochastic processes, Markov chain, transition probability matrix, steady-state probabilities

INTRODUCTION

A brief introduction to CPU scheduling: The assignment of physical processors to processes allows processors to accomplish work. The problem of determining when processors should be assigned and to which processes is called Processor scheduling or CPU scheduling.

When more than one process is to be run, the operating system must decide which one first. The part of the operating system concerned with this decision is called the scheduler and algorithm it uses is called the scheduling algorithm.

Few CPU scheduling algorithms are as follows:

First-In-First-Out (FIFO): Perhaps, First-Come-First-Served algorithm is the simplest scheduling algorithm. Processes are dispatched according to their arrival time on the ready queue. Being a non preemptive discipline, once a process has a CPU, it runs to completion, the major drawback of this scheme is that the average time is often quite long.

Shortest-Job-First (SJF): Is a non-preemptive discipline in which waiting job (or process) with the smallest estimated run-time-to-completion is run next. In other words, when CPU is available, it is assigned to the process that has smallest next CPU burst. The obvious problem with SJF scheme is that it requires precise knowledge of how long a job or process will run and this information is not usually available.

Priority scheduling: The basic idea is straightforward each process is assigned a priority and priority is allowed to run. Equal-Priority processes are scheduled in FCFS order. The Shortest-Job-First (SJF) algorithm is a special case of general priority scheduling algorithm.

Priority can be defined either internally or externally. Internally defined priorities use some measurable quantities or qualities to compute priority of a process. Examples of internal priorities are:

- Time limits.
- Memory requirements.
- File requirements, for example, number of open files.
- CPU Vs I/O requirements.

Externally defined priorities are set by criteria that are external to operating system such as:

- The importance of process.
- Type or amount of funds being paid for computer use.
- The department sponsoring the work.
- Policies.

A major problem with priority scheduling is indefinite blocking or starvation (Millan, 2001; Galvin, 2002). A solution to the problem of indefinite blockage of the low-priority process is ageing. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long period of time.

Round robin: One of the oldest, simplest, fairest and most widely used algorithm is round robin (RR). In the round robin scheduling, processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or a quantum.

Corresponding Author: Ashok Sahai, Departments of Mathematics and Computer Science, the University West Indies, St. Augustine, Trinidad and Tobago, India

If a process does not complete before its CPU-time expires, the CPU is preempted and given to the next process waiting in a queue. The preempted process is then placed at the back of the ready list.

Round Robin Scheduling is preemptive (at the end of time-slice) therefore it is effective in time-sharing environments in which the system needs to guarantee reasonable response times for interactive users.

The only interesting issue with round robin scheme is the length of the quantum. Setting the quantum too short causes too many context switches and lower the CPU efficiency. On the other hand, setting the quantum too long may cause poor response time and approximates FCFS.

In any event, the average waiting time under round robin scheduling is often quite long.

In general, most of the multi user operating systems (Millan, 2001; Tanenbaum, 2000) use the combination of SJF, Priority and Round Robin algorithms for effective utilization of CPU.

The priorities can be set either externally or internally. For effective utilization of CPU, Priorities to various processes can be set by a study of type and time of processes during a day. For example there can be more number of LOGON processes during the day time in an organization and more backup jobs during evening. So logon processes should be given top priority in the morning hours, where as backup jobs should be given higher priority than other jobs during closing hours. A typical case of a soft ware company dealing in Credit/ Debit Card transactions, in a month first week there are more transactions and in the middle of month there back end operations, in the last week of the month credits. The priorities can be set depending on the frequency (Hit ratio) of the processes during a time slice (a time slice should be minimum of 3 to 5 h).

A brief introduction to stochastic processes and Markov chains: A family of random variables $\{X(t), t \geq 0\}$ is indexed by time parameter t, then process X (t) is called a stochastic processes (Bhat, 1984; Kendall, 1953).

The set of possible values of Indexing parameter is called parameter space, which can be either continuous or discrete. The possible values X (t) assume are called as states and the set of possible values is called as state-space, which can be continuous or discrete.

A discrete parameter discrete state space stochastic process $\{x_{n+1} = t_{n+1}/x_n = t_n, x_{n-1} = t_{n-1}, x_{n-2} = t_{n-2}, \ldots, x_0 = t_0\} = \{x_{n+1} = t_{n+1}/x_n = t_n\}$ is called as Markov Chain.

This research deals with identifying priorities of application processing jobs that are arriving at CPU processing in a multi processing environment. The priorities identified are mostly for immediate application i.e., from next moment or for application in immediate near future i.e., from next cycle onwards.

In this research a CPU Scheduling algorithm is discussed which will consider the traffic intensity (hit ratio) of the jobs coming to the CPU in the recent past and most recent ones and decide on priorities, rather than a predefined format of priorities these priorities will change from time to time based on the requests coming to CPU.

Formulating dynamically evolving priorities CPU scheduling as a Markov chain model: Let $T_1, T_2, T_3, T_4, \ldots, T_n$ be the tasks that are coming for processing at CPU, let $P_1, P_2, P_3, P_4, \ldots, P_n$ be the ratios or probabilities of these tasks based on the past data.

Let X_i be the state of the processor when it is processing the task $T_{i,Pij}$ be the probability that the processor is executing the task T_j at n + 1st step and in nth step it executed task T_i

$P_{ij} = \{X_{n+1} = j/Xn = i\}$ which can be referred to as transition probability and the Matrix which gives all the possible values is referred to as transition probability matrix.

$$p=((p_{ij}))=\begin{pmatrix} p_0 & p_1 & p_2 & \cdots & 1-\sum_{1=0}^{n-2} p_1 \\ p_0 & p_1 & p_2 & \cdots & 1-\sum_{1=0}^{n-2} p_1 \\ 0 & p_0 & p_1 & \cdots & 1-\sum_{1=0}^{n-3} p_1 \\ 0 & 0 & p_0 & \cdots & 1-\sum_{1=0}^{n-4} p_1 \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ \vdots & \cdots & \cdots & \cdots & 1-P_0 \end{pmatrix}$$

since the times taken by the CPU are in nano seconds and the requests coming up to the processor are very high hence the priorities keep changing from time in the study taken by the authors it is observed that the priorities decided are stable for a couple of hours which can be consider to be the behavior of the system in the long run or steady-state. The priorities are different for different schedules hence the priorities are taken accordingly.

The priorities for a specific schedule are computed, as follows:

The steady state probabilities or the probabilities or priorities in the long run are $\pi_0, \pi_1, \pi_2, ..., \pi_{n-1}$.

Assuming that steady state is achievable, the steady-state probability vector $\pi = (\pi_0, \pi_1, \pi_2, \ldots\ldots \pi_{n-1})$; $\pi_l > 0$, for l = 0,1,……., n-1, can be found as a solution to the system of equations $\pi P = \pi$, in conjunction with

$$\sum_{l=0}^{n-1} \pi_l = 1$$

The priorities obtained are used for that slot proved very efficient effective.

A software code is written for a multiprocessing credit card transactions environment and the comparative results showed more than 15% efficiency.

REFERENCES

Bhat, U.N., 1984. Elements of Applied Stochastic Processes, John Wiley and Sons, New York.

Galvin, S., 2002. Operating System Concepts, 5th Edn., John Willy and Sons.

Kendall, D.G., 1953. Stochastic processes occurring in the theory of Queues and their analysis by the method of Imbedded Markov Chain. Anal. Math. Statistics, 24: 338- 354.

Millan, M., 2001 Operating Systems. 2nd Edn., Tata Mc. Grahil. l.

Tanenbum, A.S., 2000. Modern Operating Systems, 2nd Edn., Pearson Asia.