International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 4 (2013), pp. 311-322

http://www. irphouse.com /ijict.htm

Intelligent Dynamic Time Quantum Allocation in MLFQ Scheduling

Deepali Maste[1], Leena Ragha[2] and Nilesh Marathe[3]

[1,2,3]*Computer Department, Ramrao Adik Institute of Technology, Nerul, Mumbai.*

Abstract

In today's world , computing systems serve many purposes, we can not imagine a world without them. So it gives us the ability to do tasks at great speeds and it opens a new world of possibilities that we are just starting to explore . One of the most important factors of computing systems is the ability to run several processes at once, or at least a good scheduler gives that impression. So to achieve this, the scheduler causes the system to switch quickly between processes. Each process runs for a certain time, and when a process times out the scheduler chooses which process to execute next on the CPU and for what amount of time. Therefore, the behaviour of a system mainly depends on the scheduler. The scheduler must balance fairness and efficiency to offer the appropriate behaviour. In this paper, we have proposed a new variant of MLFQ algorithm, in which time slice is assigned to each queue for MLFQ scheduling such that it changes with each round of execution dynamically and neural network is used to adjust this time slice again to optimize turnaround time. The overall performance of algorithm is to be observed to be improved by , MLFQ using dynamic time quantum and neural network over MLFQ using static time slice for each queue.

Keywords: Process Scheduling, Neural network, MLFQscheduling, artificial intelligence, turnaround time, dynamic time quantum.

1. Introduction

1.1 Scheduling

The process scheduler is a major component of operating systems and it is responsible for the assignment of CPU time to processes, while maintaining scheduling policy to obtain user interactivity, throughput, real time responsiveness, and more.

The most popular scheduling algorithms that adhere to user interactivity are priority based [1]. The basic idea is to keep the CPU busy as much as possible by executing a process until it must wait for an event, and then switch to another process. Processes alternate between consuming CPU cycles (CPU-burst) and performing I/O (I/O-burst) [2].

Scheduling is a complex decision making activity because of conflicting goals, limited resources and the difficulty in accurately modeling real world scenarios[13].

1.2 Scheduling Criteria

There are several different criteria to consider when trying to select the "best" scheduling algorithm for a particular situation and environment, including, CPU utilization, Throughput, Turnaround time, Waiting time, Load average, Response time. In general one wants to optimize the average value of a criteria (Maximize CPU utilization and throughput, and minimize all the others). However some times one wants to do something different, such as to minimize the maximum response time*[2]*. Sometimes it is most desirable to minimize the *variance* of a criteria than the actual value. i.e. users are more accepting of a consistent predictable system than an inconsistent one, even if it is a little bit slower.

1.3 Choice of Scheduling algorithm

When designing an operating system, a programmer must consider which scheduling algorithm will perform best for the use the system is going to see. There is no universal “best” scheduling algorithm, and many operating systems use extended or combinations of the scheduling algorithms above. For example Windows NT/XP/Vista uses a multilevel feedback queue,, a combination of fixed priority preemptive scheduling, round-robin, and first in first out.

1.4 MLFQ Scheduling

Multilevel feedback queue (MLFQ) algorithms are extensions to the more general MLQ algorithms. Silberchatz, Galvin and Gagne [2], algorithm partitions the ready queue into several separate queues. The processes are assigned to one queue. Based on proper of the process, such as memory size, process priority or process type. Each queue has its own scheduling algorithm. The main difference relies on the fact that, in MLFQ algorithms, jobs are allowed to move from one queue to another, according to their runtime behavior*[15]*.

By moving jobs from higher priority queues to lower priority ones, the algorithm guarantees that short jobs and I/O-intensive processes get to the CPU faster. After a

quantum of time jobs are decreased in importance and moved to a lower priority queue.

Artificial Intelligence and Scheduling

Artificial Intelligence (AI) aims at constructing artifacts (machines, programs) that have the capability to learn, adapt and exhibit human-like intelligence. Hence, learning algorithms are important for practical applications of AI. The field of machine learning is the study of methods for programming computers to learn.Research shows that Artificial Neural Networks (ANNs) can be used as a machine learning tool to study the scheduling process*[13]*.An ANN is a data-driven modeling tool that is able to capture and represent complex and non-linear input/output relationships.

Neural networks have several applications in scheduling. These are: searching networks (Hopfield net), probabilistic networks (Boltzmann machine), error-correcting networks (multilayer perceptron), competing networks and self-organizing networks. One of the advantages of using neural networks is that they can be trained for specific use cases by experts and advanced users and then re-used by less technically inclined users without requiring any knowledge about neural networks or schedulers.

Training a network is done completely offline. This means it is not required to have a modified kernel to train the networks. In fact, the networks can be trained (as well as tested) on any other OS if this is required. The feature data is extracted from a running modified kernel and written to a file which can be used for training and testing.

Neural Network in MLFQ

Attractive feature of Neural Network is the ability to learn. This provides a level of adaptability that does not exist in current scheduling disciplines. Detecting a bad state that leads to bad state , and procedure to avoid it is an example of the usefulness of this learning capability [14].

Neural networks are the ideal AI technique to use here because they can be deployed in different situations by training them for these. The input and output are continuous numbers, which the available process data usually is as well. It also allows for a kind of “brute force” search by experimentation. One can simply add features, even if it is still largely unclear how these features can be used to calculate process priorities. The network can “figure out” how to do this, if it is possible. Also, in case of noise in the input, neural networks degrade gracefully. When features are introduced by subsystems that are “equally important” for classification, the problem starts to resemble so-called P-type problems [16]. If this is the case, neural networks will perform better than algorithms that look at their input values one at a time in a hierarchical fashion, e.g. decision tree algorithms [3].

2. Literature Survey

2.1 "Comparison of scheduling techniques" ,Al-Husainy, M.A.F. [4]

Allocating CPU to a process requires careful attention to assure fairness and avoid process starvation for CPU. Different CPU scheduling algorithm have different properties and may favor one class of process over another numerous performance measures have been suggested for comparing CPU scheduling algorithms.

2.2 "Dynamic time quantum in RR" , Rami J. Matarneh [9]

Rami J. Matarneh founded that an optimal time quantum could be calculated by the median of burst times for the set of processes in the ready queue, unless if this median is less than 25ms. In such case, the quantum value must be modified to 25ms to avoid the overhead of context switch time .

2.3 "MLFQ analysis" ,Basney, Jim and Livny, Miron [5]

In Basney, smoothed competitive analysis is applied to multilevel feedback algorithm. Smoothed analysis is basically mixture of average case and worst case analysis to explain the success of algorithms. This paper analyses the performance of multilevel feedback scheduling in terms of the time complexity. Any performance enhancing approach can use this approach for performance analysis in terms of the time complexity.

2.4 "Recurrent NN in MLFQ" ,Becchetti, L., Leonardi, S. ,Marchetti S.A. [10]

In this paper Recurrent Neural Network has been used to optimize the number of queues and quantum of each queue of MLFQ scheduler to decrease response time of processes and increase the performance of scheduling. In this paper the proposed neural network takes inputs of the quantum of queues and average response time. After getting the required inputs, it takes the responsibility of finding relation between the specified quantum changes with an average response time. It can find the quantum of a specific queue with the help of optimized quantum of lower queues. Thus, this network fixed changes and specify new quantum, which overall optimize the scheduling time.

2.5 "Semantic cognitive scheduling", Shlomi , Avi and Igal Shilman [6]

Semantic cognitive scheduling method is presented. A scheduler that obeys the semantics cognitive scheduling paradigm takes into account the semantic progress of the processes in order to produce a schedule which will imply the maximal semantic progress in executing the tasks of the system. Here authors introduced a framework, define the general problem, provide a lower bound for the optimal algorithm in terms of time complexity and present dynamic and greedy competitive algorithms for the case of bounded state. Various ways investigated to use additional process dependency information in the form of process dependency graph. The process dependency graph is used to achieve effective scheduling.

2.6 "Fuzzy in MLFQ TQ allocation", Puneet Kumar , Nadeem and M Faridul Haque [11]

For using an operating system's resources more efficiently, multiprogramming plays an important part. And for multiprogramming to take place, scheduling plays a pivotal role. This policy of deciding which processes to run at a given time should attempt to maximize throughput, minimize latency, prevent process starvation etc. Various techniques are present to perform the said task. In this paper a new improved scheduling algorithm technique based on Fuzzy Logic has been proposed. The proposed algorithm has been implemented and compared with the existing FCFS and Round Robin. Here Fuzzy Logic has been used to decide a value for time quantum that is neither too large nor too small such that every process has reasonable response time and the throughput of the system is not decreased due to unnecessary context switches.

2.7 "MLFQ improvement" , Rakesh Kumar Yadav, Anurag Upadhayay [7]

CPU scheduling is a vital phenomenon of operating system. At present, numerous CPU scheduling algorithms are existing like FCFS (First come first serve), SJF (shortest job first), SRTF (Shortest remaining time first), Priority Scheduling, (RR) Round Robin scheduling , MLQ(multilevel queue). Efficiency and performance are not remaining satisfactory of these algorithms. MLFQ (Multilevel feedback queue) be one of most potential strategies, for CPU scheduling .It is further extension of multi-level queue scheduling algorithm while multilevel queue scheduling is results of combination of basic scheduling algorithms such as FCFS and RR scheduling algorithm. Therefore, research on these algorithms remains continuing till today. This paper, suggested a novel approach which will improve the performance of MLFQ (CPU) scheduling algorithm.

2.8 "Dynamic TQ in RR", Abbas Noon, Ali Kalakech, Seifedine Kadry [8]

Round Robin, considered as the most widely adopted CPU scheduling algorithm, undergoes severe problems directly related to quantum size.In this paper, authors propose a new algorithm, called AN, based on a new approach called dynamic-time-quantum; the idea of this approach is to make the operating systems adjusts the time quantum according to the burst time of the set of waiting processes in the ready queue.

2.9 "ESN in job scheduling", Julien Perez1, Balazs Keg, Cecile Renaud [12]

Two recurrent questions often appear when solving numerous real world policy search problems. First, the variables defining the so called Markov Decision Process are often continuous, that leads to the necessity for discretization of the considered state/action space or the use of a regression model, often non-linear, to approach the Q-function needed in the reinforcement learning paradigm. Second, the markovian hypothesis is made which is often strongly disputable and can lead to unacceptably suboptimal resulting policies. In this paper, the job scheduling problem in grid infrastructure is modeled as a continuous action-state space, multi-objective reinforcement learning problem, under realistic assumptions; So, formalizing the problem as a partially

observable Markov decision process , here used the algorithm of fitted Q-function learning using an Echo State Network. The experiment, conducted on simulation of real grid activity will demonstrate the significant gain of the method against native scheduling infrastructure and a classic feed forward back-propagated neural network for Q function learning in the most difficult cases.

3. The Proposed Work

MLFQ uses several queues with several quantum. Static time slice to the queue is limitation of MLFQ algorithm . In this paper, we have proposed a new variant of MLFQ algorithm. We propose algorithm in which time slice is assigned to each queue for MLFQ scheduling such that it changes with each round of execution dynamically and neural network is used to adjust this time slice again to optimize turnaround time. The overall performance of algorithm is to be observed to be improved by MLFQ using dynamic time quantum and neural network over MLFQ using static time slice for each queue.

Frequent context switches possible if time quantum is very small and the algorithm becomes FCFS if it very large. So we try to solve this problem by making choice of dynamic time slice appropriately, where the time slice are adjusted in every round according to dynamic priority.

3.1 DTQ allocation and Neural approach

Turn around time (tat) is the time gap between the instant of submission of process and the instant of its completion. Neural network is used to optimize turnaround time. Number of the queues and quantum of each queue affect the turnaround time directly. In this paper, we propose the algorithm for solving these problems and optimizing the turnaround time. In this algorithm Neural Network has been utilized to find the optimized quantum of each queue. Following method can be applied using neural network:

1. Run the programs with different special time slices which gives minimum turn-around-time (TaT).
2. Build the knowledge base of static and dynamic characteristics of the programs from the run traces obtained in step 1 and train them with the neural algorithm.
3. If a new program comes, classify it and run the program with this predicted Time slice.
4. If the new program instance is not in the knowledgebase, go to step 1.

Here dynamic quantum is calculated by calculating Bsp i.e. burst span , calculated by average of initial burst value of ,mid of the queue burst value and last burst value of queue. Also average of priorities and highest priority of queue is considered to calculate dynamic time quantum. Once the dynamic quantum is assigned to the queue , it is fed to the neural network which is used to adjust these quantum again to achieve optimized turnaround time.As shown in figure 1 Neural network updates the weights and then changes the quantum of the queues input and specifies a new quantum for

queues. We can find the effects of this change on average turnaround time, the new amounts of quantum to be given to the NN function .

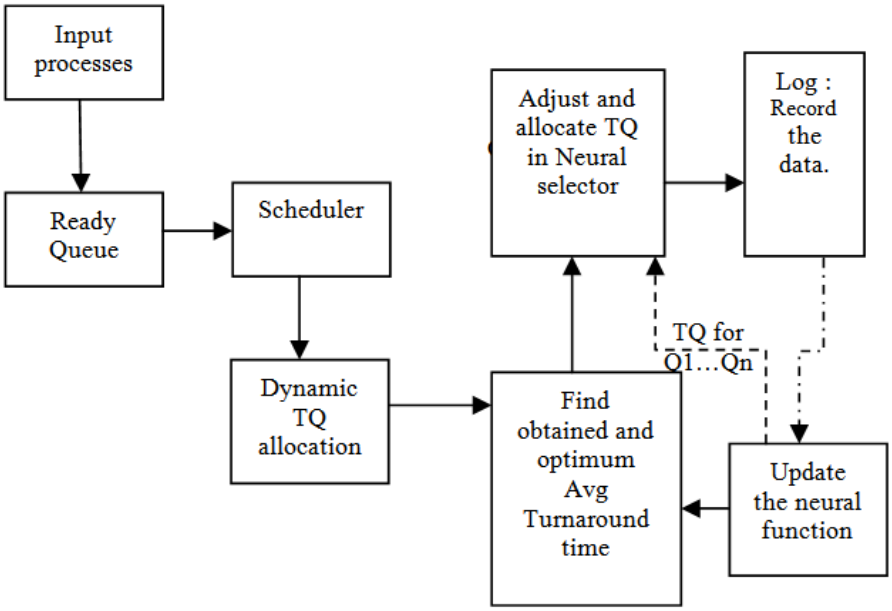

Figure 1: Architecture for Intelligent DTQ allocation.

The quantum of queues are fed back to the inputs in a recursive way, means only the new quantum of a specified queue is fed to the input and the other queues receive the former amounts as inputs. After replacing the new quantum of a specified queue in NN function, using pre-assumed default processes used to obtain the primary turnaround time, the new turnaround time caused by this change is found. Here, when a change is applied in the quantum of a specified queue, the number of queues can be changed. It is possible that reducing the quantum caused more processes are moved to the lower queues or a new queue is added to the number of required queues.

By characterizing or recognizing programs it may be possible to understand their previous execution history and predict their resource requirements. Here Figure 2 shows, how to minimize the TaT of programs by using NN technique. We discover certain static and dynamic characteristics of a program like input type, input size, text, process size and uninitialized data info size , taken as features which NN is used to optimize turnaround time. We call dynamic Time Quantum as the CPU burst time that minimizes turnaround time.

Input Layer Hidden Layer Output Layer

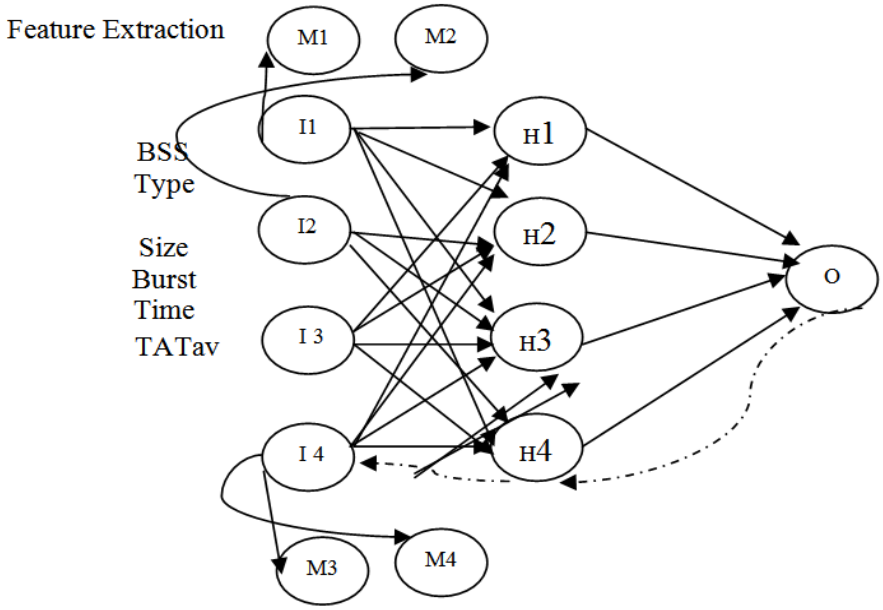

Figure 2: Design of the Network.

3.2 Parameter Selection

Text section is a place where the compiler put executable code of a program. BSS (Block Started by Symbol) is a section where all uninitialized variables are mapped. Turnaround time reduction rate slowly increases with the input size of the program . We consider here, processes of type computation bound and I/O bound. So here we need to study the execution behavior of several programs with its characteristics that can be used to predict the dynamic time quantum. We can take representative programs like matrix multiplication, sorting arrays, recursive, random number generator programs etc. And we use Pro-log file here which consists of fully labeled data about these processes for training.

3.3 3Proposed Algorithm

n = total no. of processes
Pri = No. of priorities
Prh= highest priority in queue
TQ= Time Quantum
AVpr=Average of priorities
Bsp= BurstSpan
Input : No of processes (P1, P2, …., Pn)
Burst time of processes (Bs1, Bs2,….Bsn)
Priority of processes (Pri1, Pri2……Prim)
Bi=initial burst value of queue
Bm=mid burst value of queue
Bl=last burst value of queue

Qx=Queue
Output : TATav = Average turnaround time,

1. Process arrives in ready queue with arrival time and predicted burst time .
2. Sort the processes into queues according to ascending order of burst time
3. Assign the priority to process in ascending order.
4. Update the Pro-Log file about input type , size, text, prog. Size and uninitialized data
5. For each queue x=1 to n repeat the following :
6. Calculate time quantum for each queue Qx as follows.
   TQ(Qx)=(Bsp * n) / (AVpr * Prh) ……. .Eq..1

Where Bsp=((Bi-initial+Bm-mid+Bl-last)/3) ……. .Eq..2

Calculate Average Turnaround time TATav

State : Using neural network find the optimum value of queue according to other queue quantum and the average turnaround time that is found in the previous stages. The state vector, x(t), of the network is governed by,

x(t+1)=f (Win Q(t) + W x(t) +Wfb TQ(t) ……Eq..3

where Win : describes the weights connecting the inputs to the network,

Q(t) is the input vector, W describes the recurrent weights, Wfb describes the feedback weights connecting the outputs back to the network, TAT(t) are the outputs.

The output TAT (t) is governed by TAT(t) = Wout .z(t); where z(t) = [x(t); Q(t)] is the extended state.

By including the input vector, the extended state allows the network to use a linear combination of the inputs in addition to the state to form the output.

1. Initialize network of N with random Win, W, and Wfb.
2. Sparsify W .
3. Train network:
   (a) Input training sequence and assign output TAT(t) to take on the training output.
   (b) Record extended state z over training period.
   (c) Find Wout to minimize TAT over the training period.

1. Consider the changes in other queue and update the quantum.
2. End for.

4. Conclusion

Generally, MLFQ algorithms achieve better results if compared to other scheduling policies, but the overhead introduced is usually higher. When designing an operating system, a programmer must consider which scheduling algorithm will perform best for the use the system is going to see. There is no universal “best” scheduling algorithm, and many operating systems use extended or combinations of the scheduling algorithms.

Neural networks are the ideal AI technique to use here because they can be deployed in different situations by training them for these. The input and output are continuous numbers, which the available process data usually is as well. It also allows for a kind of "brute force" search by experimentation. One can simply add features, even if it is still largely unclear how these features can be used to calculate process priorities. The network can "figure out" how to do this, if it is possible. Also, in case of noise in the input, neural networks degrade gracefully. But disadvantage of neural networks is that it is very hard to get any meaning out of their structure, because their knowledge is encoded sub-symbolically. When a network is trained for a new feature, it will not give much insight to how the feature relates logically to other features. But on the other hand, there has been some research with the intent to extract rules from neural networks . This could be useful to create more optimized rule-based schedulers after experimentation with the neural network scheduler. When features are introduced by subsystems that are "equally important" for classification .If this is the case, neural networks will perform better than algorithms that look at their input values one at a time in a hierarchical fashion.

References

[1] A. Silberschatz, Galvin, and G. Gagne, "Operating System Concepts", 2009 Wiley Inc.

[2] A. S. Tanenbaum, "Modern Operating Systems" 2009 Prentice Hall(book).

[3] J.R. Quinlan." Comparing connectionist and symbolic learning methods.",1994, Volume I: Constraints and Prospects, pages 445–456. MIT Press, 3.2, 3.5

[4] Al-Husainy, M.A.F.,"Best-job-first CPU scheduling algorithm" 2007, Inform. Technol. J.,Volume 6: Number 2, Pp: 288-293(pp)

[5] Basney, Jim and Livny, Miron,"Managing Network Resources in Condor",2000 9th IEEE Proceedings of the International Symposium on High Performance Distributed Computing, Washington, DC, USA.(pp)

[6] Shlomi Dolev , Avi Mendelson and Igal Shilman," Semantical Cognitive scheduling" , Nov. 2012 Technical Report # 13-02 Research in part by Microsoft and (Fronts).(pp)

[7] Rakesh Kumar Yadav, Anurag Upadhayay , "A fresh loom for Multilevel Feedback Queue Scheduling Algorithm", July 2012,International Journal of Advances in Engineering Sciences Vol.2, Issue 3.(pp)

[8] Abbas Noon1, Ali Kalakech2, Seifedine Kadry "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average" May 2011, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1. ISSN (Online): 1694-0814(pp)

[9] Rami J. Matarneh, "Self-Adjustment TQ in RR Algorithm Depending on Burst Time of the Now Running Processes",2009, American Journal of AS, Vol 6, No. 10.(pp)

[10] Becchetti, L., Leonardi, S. and Marchetti S.A., "Average-Case and Smoothed Competitive Analysis of the Multilevel Feedback Algorithm" 2006, Mathematics of Operation Research Vol. 31, No. pp. 85–108.(pp)

[11] Puneet Kumar , Nadeem and M Faridul Haque "Efficient CPU Scheduling Algorithm Using Fuzzy Logic", 2012 vol. 47, IACSIT Press, Singapore(pp)

[12] Julien Perez1, Balazs Keg "Non-Markovian Reinforcement Learning for Reactive Grid scheduling", Apr 2011 ,Cecile German Renaud -00586504, pp version 1 - 16 .(pp)

[13] Gary R.Weckman · Chandrasekhar V. Ganduri · David A. Koonce, "A neural network js scheduler" 2008 © Springer Media,19:191–201 DOI 10.1007/s10845-008-0073-9(pp)

[14] Dale Tonogai ,"AI in Operating system : An Expert scheduler", University of Californiya, Berkeley, 1988 ,Defence advanced research projects,1988(thesis)

[15] Paolo Di Francesco ,"Design and implementation of a MLFQ scheduler for the Bacula backup software" 2012 , Malarden University , Sweden.(thesis)

[16] Remzi H , Andrea C ,"Operating Systems", 2011, Four Easy Pieces (V0.4) (book).