International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 16 (2014), pp. 1611-1624

http://www. irphouse.com

Analysis of Improved Neural MLFQ Scheduler

Deepali Maste[1], Dr. Leena Ragha[2], Prof Nilesh Marathe[3]

[1, 2, 3]Computer Department, [1, 2, 3]Ramrao Adik Institute of technology, Nerul, Mumbai
[1]deepali. maste@gmail. com [2]leena. ragha@gmail. com
[3]nilesh. marathe@rait. ac. in

Abstract

The scheduling task becomes more difficult when a feasible schedule does not exist and the goal is to minimize some measure of delinquency, often termed as tardiness. Tardiness of an individual job under a given schedule is defined as the amount of time by which the job finishes after its prescribed deadline, and is considered to be zero if the job finishes on or before the deadline. Assuming there are runnable processes, a process should always be running. If there are more processes than processors in a system, some processes will not always be running. These processes are waiting to run. Deciding which process runs next and for what amount of time when given a set of runnable processes, is the fundamental decision that scheduler must take. There is no universal "best" scheduling algorithm, and many operating systems use extended or combinations of the scheduling algorithms. In this work we use Neural Network technique to learn the execution behavior of known processes to minimize average turnaround time in MLFQ scheduling. By experimentation we can conclude that predictive quantum scheduling could reduce average turnaround time in the range of 4% to 30% in average. If processes with large variations in parameters scheduled together then it can be more helpful in prediction of time quantum by Neural Network. At present Operating System has not used process's execution behavior history to schedule processes so we find our work interesting in that context.

Keywords—average turnaround time, Recurrent Neural Network, Process Scheduling, MLFQ scheduling, dynamic time quantum, process attributes, standard deviation

I. Introduction

Scheduling

User interactivity and process completion time are major concern in operating

systems such as Microsoft Windows and Linux [1]. The most popular scheduling algorithms that adhere to user interactivity are priority based [2]. In multiprogramming systems, when there is more than one ready processes, the operating system must decide which one to activate. The decision is made by the part of the operating system called the scheduler, using a scheduling algorithm. The scheduler is concerned with deciding policy, not providing a mechanism[3]. Turnaround time is the time required for a particular process to complete, from submission time to completion (Wall clock time)[2]. The scheduling algorithm is the last component of a scheduling system. It has the task to generate a valid schedule for the actual stream of submission data in an on-line fashion. A good scheduling algorithm is expected to produce very good if not optimal schedules with respect to the objective function while not taking 'too much' time and 'too many' resources to determine the schedule. Unfortunately, most scheduling problems are computationally very hard.

MLFQ Scheduling

Unix and Windows NT use a strict multilevel feedback queue scheduling algorithm. Modern operating systems support up to 160 queues in which a process is placed, depending on its priority[4]. The MLFQ scheduler is priority-preemptive, where a running process can be preempted from the CPU by another process that has a higher priority. The multi-level feedback queue is an excellent example of a system that learns from the past to predict the future. Such approaches work when jobs have phases of behavior and are thus predictable, of course one must be careful with such techniques, as they can easily be wrong and drive a system to make worse decisions than they would have with no knowledge at all.

Neural Network in Scheduling

NN is composed of several layers of processing elements or nodes. These nodes are linked by connections, with each connection having an associated weight. The weight of a connection is a measure of its strength and its sign is indicative of the excitation or inhibition potential. A NN captures task-relevant knowledge as part of its training regimen. This knowledge is encoded in the network in: the architecture or topology of the network, the transfer functions used for nonlinear mapping and a set of network parameters (weights and biases) [6]. Learning from past history is a fundamentally ill posed. A model may fit past data well but not perform well when presented with new inputs. Recurrent NNs (RNNs), leverage the modeling abilities of NNs for time series forecasting. Feedforward NNs have done well in classification tasks such as handwriting recognition, however in dynamical environments, techniques needed that account for history. In RNN, signals passing through recurrent connections constitute an effective memory for the network, which can then use information in memory to better predict future time series values [5].

LITERATURE SURVEY

Semantic cognitive scheduling method is presented by Shlomi, Avi and Igal Shilman

[1]. 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. The process dependency graph is used to achieve effective scheduling[1].

Rami J. Matarneh [6] 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.

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 so Al-Husainy [7] suggested numerous performance measures for comparing CPU scheduling algorithms. Becchetti, L., Leonardi, S., Marchetti S. A. [8] suggested Recurrent Neural Network 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. They proposed that 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[8].

Julien Perez1, Balazs Keg, Cecile Renaud [9] models the job scheduling problem in grid infrastructure 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.

Round Robin, considered as the most widely adopted CPU scheduling algorithm, undergoes severe problems directly related to quantum size. Abbas Noon, Ali Kalakech, Seifedine Kadry [8] proposed a new algorithm, AN based 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.

Puneet Kumar, Nadeem and M Faridul Haque [10] proposed a new improved scheduling algorithm technique based on Fuzzy Logic. Their 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. Basney, Jim and Miron [11] tried to apply smoothed competitive analysis to multilevel feedback algorithm. Smoothed analysis is basically mixture of average case and worst case analysis to explain the success of algorithms. Atul Negi, Senior Member, IEEE, Kishore Kumar P[12] shows that machine learning algorithms are very efficient in the characterization of process. The C4. 5 decision tree algorithm achieved good prediction (91% to 94%), which indicates that when suitable attributes are used, a certain amount of predictability does exist for known programs. In this paper

experiments show that 1:4% to 5:8% reduction in turnaround time is possible and this reduction rate slowly increases with the input size of the program. From the experiments, author finds the best features : input size, program size, bss, text, rodata and input type of a program that can characterize its execution behavior. This technique can improve the scheduling performance in a single system[12].

After studying various aspects of scheduling techniques and methods used in improvement of those algorithms, we can not neglect the fact that MLFQ is most widely used algorithm in recent OS and research to improve it, is still going on. We will move forward to focus on our proposal, MLFQ and use of NN to improve performance of MLFQ scheduling algorithm in the next section.

THE PROPOSED WORK [13]

Neural Approach

Here ready processes arrives for execution and demands CPU time. Processes sorted according to predicted burst time in ascending order and divided among multiple queues. Burst span time is calculated for each queue and dynamic time quantum is calculated for each queue. RNN is used to optimize turnaround time and then it compares old and new turnaround time, lesser one is selected.

We need to control a running process' quantum automatically and dynamically. A process' desired quantum can be determined by analyzing its behavior, this is most easily measured by the OS. This behavior classifies a process as “interactive”, “multimedia”, “cpu-bound” or any other classes like “size of initialized data”, ”size of readonly data”etc. one wishes to distinguish from. The user defines the classes and the priorities to be assigned to processes that fall into these classes. If the Neural Network classifies a process based only on its most recent activity profile, processes will most likely be assigned different quantum on each classification cycle because of these fluctuations. To lessen this fluctuation, the NN has to be enhanced with some form of memory. So we use Recurrent Neural Network here.. Networks' memory is used to store the feature values on the inputs at previous time steps. Processes are unrelated so at one point in time we have the features of one process clamped onto the input, but the memory will contain a value related to the inputs of the previous, unrelated processes. This will hamper learning rather than facilitate it. The input at network cycle t is completely unrelated to the input at cycle $t - 1$. If there are k processes, the input at cycle t is related to that at cycle $t-k$, but only if no processes were created or destroyed in that time span. This becomes too complicated too quickly. Requirement is a memory of input history per process. The global history which represents the relationship between processes is important in scheduling. The RNN scheduler can focus on the main issues of calculating a quantum for each process based on their isolated behavior.

Dynamic Time Quantum allocation

As shown in fig 1, RNN 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

RNN function. 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 this function, using pre-assumed default processes used to obtain the primary turnaround time, the new turnaround time caused by this change is found. This approach is aimed to present an intelligent algorithm to optimize the average turnaround time. It can be been optimized by learning the NN. Learning time of the network is directly related to the amounts of input data. RNN used to recognize the trend information of time series data. Reason to use it is, it produces a trace of its behavior and keeps a history of its previous states. Quantum of queues and the average turnaround time are the inputs of the RNN. Average turnaround time, is fed to the RNN as an input fed back from output, so the network finds the relation of the quantum change of a specified queue with the average turnaround time and the quantum of other queues. We try to optimize the average turnaround time by assuming change in quantum of specified queue. By characterizing or recognizing programs it may be possible to understand their previous execution history and predict their resource requirements as resource we consider here is Time Quantum[13].

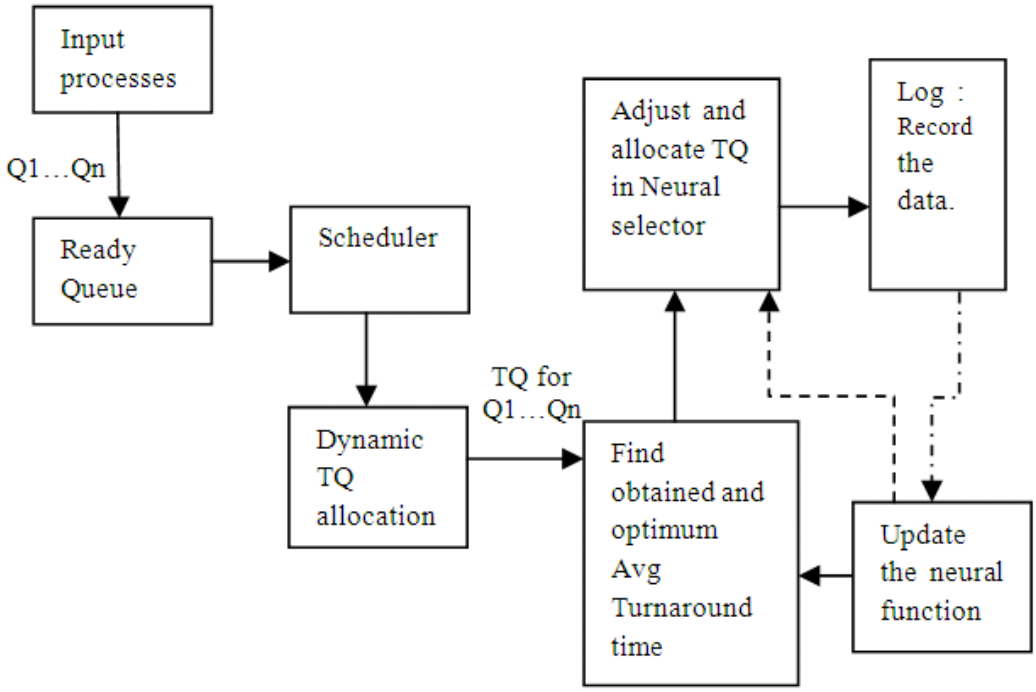

Figure 1 Architecture for DTQ allocation[13]

Parameter Selection

Text section is a place where the compiler put executable code of a program. 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 searching in array, matrix operations, 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 [13].

By characterizing or recognizing programs it may be possible to understand their previous execution history and predict their resource requirements. So input neurons are characteristics of a program like bt,. rodata,. text,. data from which RNN extracts information into memory. Hidden layer neurons and is used to optimize turnaround time as shown in fig 2 [Design-of-RNN], input neurons as process characteristics, hidden neurons will use extracted features, Levenberg-Marquardt backpropagation function can be used to train the network. After running the network and updating activations, the new activations of the memory nodes are extracted from the network. This is repeated for the next process. For a particular process, the memory will be a function of the features at the previous time steps of only that process. Weight updating is based on the network error value of every process. Memory strength can be modified by changing the weight values of the recurrent connections. In RNNs signals passing through recurrent connections constitute an effective memory for the network, which can then use information in memory to better predict future value[13].

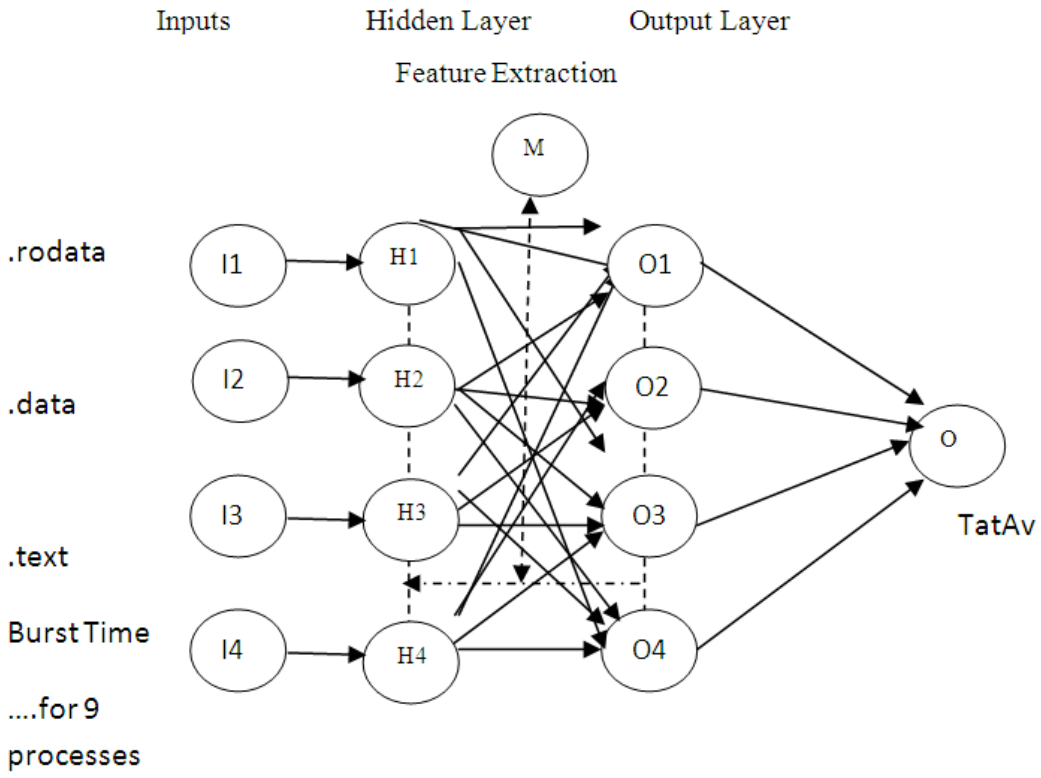

Fig 2 Design of RNN for proposed approach [13]

Proposed Algorithm [13]

n = total no. of processes
Pri = No. of priorities
Prh= highest priority in queue
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).
TQ= Time Quantum
Bi=initial burst value of queue, Bm=mid burst value of queue
Bl=last burst value of 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 i/p type, i/p size, text, program size and uninitialized data info size.
5. For each queue x=1 to n repeat the following :
5. 1 Calculate time quantum for each queue Qx as follows. TQ (Qx)= (Bsp * n) / (AVpr * Prh)
Where Bsp= ((Bi-initial+Bm-mid+Bl-last)/3) Calculate Average Turnaround time TaTAv
5. 2 Using RNN find the optimum value of queue according to other queue quantum and the average turnaround time that is found in the previous stages.

IMPLEMENTATION AND EXPERIMENTAL SETUP

The experimental procedure is divided into four phases. In the first phase, we create the data set from the program's run traces and make the data base with the static and dynamic characteristics of the programs. In the second phase, we design and simulate MLFQ scheduler to study behavior of MLFQ scheduling in terms of average turnaround time by using dynamic time quantum, in the third phase we use Recurrent NN to find more appropriate time quantum to get optimum average turnaround time by providing it more information about the process to decide upon time quantum and in the fourth phase we execute scheduler by neural approach as time quantum will be given to the scheduler suggested by RNN.

Create dataset

Processes considered here are programs in c. As discussed earlier we consider here process attributes (burst time),. data (the size of the initialized data that contribute to the size of the program's memory image size),. rodata (read-only data size that typically contribute to a non-writable segment in the process image,. text (the executable instructions size (bytes)of a program). We used readelf and time commands to get the attributes. We build the data set of 300 execution instances of programs like array processing. We collected the data like the above for 15 programs

with different input sizes, different data initialization and different number of constant declarations in the program. Data of 300 instances of the above programs was collected and made into database. bt is considered in milliseconds and. text,. data and. rodata converted from hex to decimal in bytes, Following table 1 shows the Input and Output of RNN. Input sample data collected by readelf and time command, output is average turnaround time (TaTAv) predicted by RNN.

Table 1 Input and Output of RNN MLFQ Scheduler

	I/P					O/P
Sr. No.	**Process**	**Bt**	**. data**	**. rodata**	**. text**	**TaTAv**
1	p1	1718	54	110	626	4668
2	p2	1818	54	80	578	
3	p3	858	46	135	770	
4	p4	5598	58	122	658	
5	p5	48	54	251	658	
6	p6	2038	58	177	1506	
7	p7	1628	62	357	818	
8	p8	2068	190	113	514	
9	p9	3068	58	46	978	
10	p10	835	55	289	1112	2483
11	p11	895	51	222	889	
12	p12	385	99	209	895	
13	p13	55	67	390	927	
14	p14	835	71	198	895	
15	p15	355	59	174	927	
16	p16	1715	71	172	1233	
17	p17	3335	43	243	1167	
18	p18	1815	51	208	984	

In Recurrent Neural Network implementation phase we provide the additional information about the processes as data, rodata and text along with the information of burst time, to predict the average turnaround time for a set of 9 processes. Setting up Recurrent NN in matlab, Input layer as 36 inputs (9 processes with 4 parameters each) Hidden Layer as 1 (27 neurons) and Output Layer as 1 (average turnaround time). Activation is fed forward from input to output through “hidden layers”

Training the network with 300 instances of process inputs, best performance is achieved after 8 epochs.

Weight bias data, 1. 4617-1. 3568-1. 2349-1. 1013 1. 0067-0. 9049-0. 7816-0. 6581 0. 5702-0. 4631 0. 3460…..

Randomly divide training data in two sets, training and validation. For each number of hidden neurons we want to try, we train our network with the training set and evaluate the error in the validation one. The lowest validation error gives us a good estimation of the best training parameter.

Levenberg-Marquardt backpropagation technique is used for training the network. After every such epoch, compute the error. Stop the training when the error falls below a predetermined threshold, or when the change in error falls below another predetermined threshold, or when the number of epochs exceeds a predetermined maximal number of epochs. Many (order of thousands in nontrivial tasks) such epochs may be required until a sufficiently small error is achieved.

Result and Analysis

Test data of 160 instances of 20 processes generated and the model is tested to analyze the result. We analyzed the cases where we could be able to achieve our target as to minimize the average turnaround time. The processes for which we get the best results are analyzed with all its attributes and its behavior, RNN respond in different way for different pattern or behavior of processes. Pattern of processes and its attribute values provided to the scheduler affects to the average turnaround time. Following is the comparative study of best case, worst case and average case as per result given by MLFQ scheduler using RNN approach.

In best case scheduler was able to achieve minimum TaTAv (Average Turnaround time) as suggested by RNN as shown in fig 3. And in worst case scheduler was not able to achieve minimum TaTAv (Average Turnaround time) as suggested by NN. Results were much better without the prediction of NN, fig 4 shows the worst cases where actual average turnaround time is more than the predicted average turnaround time.

Table 2 Best Case of RNN prediction in MLFQ scheduling

TaTAv before prediction	TaTAv after prediction
7440	7337
2522	2414
4147	3925
5447	5164
7483	7170
6718	6364
7170	6702
7391	6905
7440	6905
8013	7378
8013	7337
6109	5372
4668	3925
3225	2414
3319	2340
8013	6905
7483	6364
6702	516

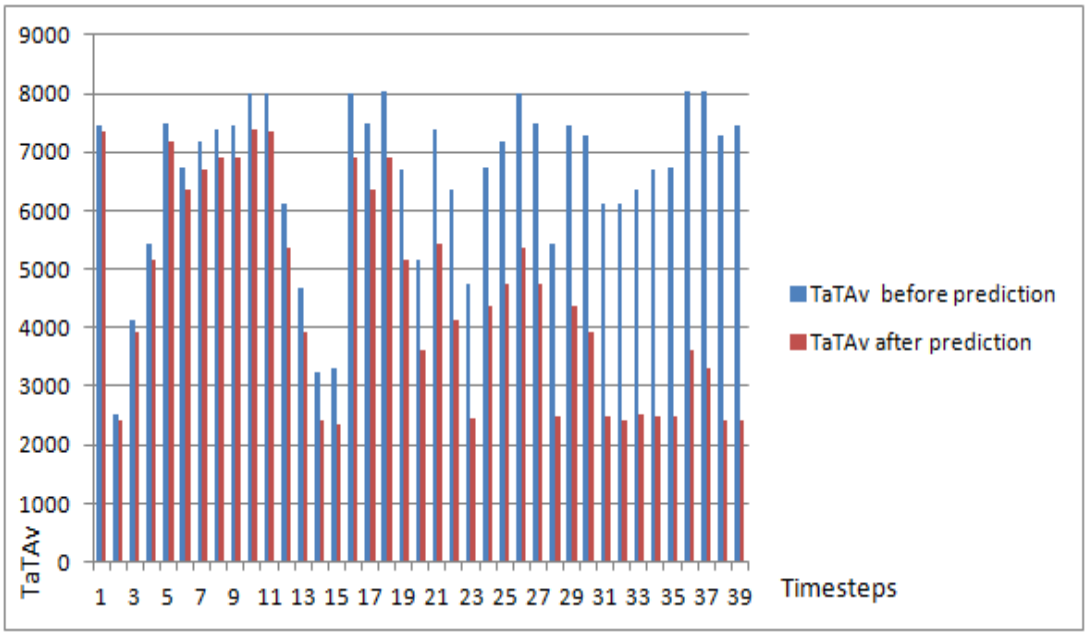

Fig 3 Best Case of RNN prediction in MLFQ scheduling

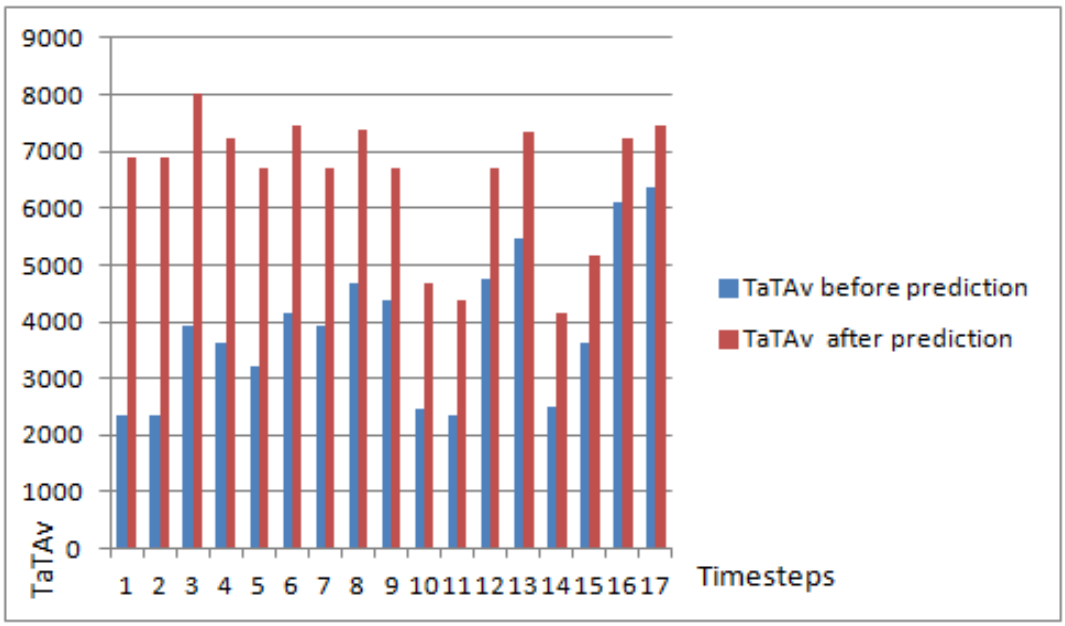

Fig 4 Worst Case RNN prediction in MLFQ scheduling

Behavior of process input attributes and its effect on RNN prediction

One aspect of most sets of data is that the values are not all alike; indeed, the extent to which they are unalike, or vary among themselves is of basic importance. Measuring the behavior of input parameters to find out how these process attributes affect in worst case and best cases. Let us observe that the dispersion of a set of data. If we

observe the data we could say that more the variations in data more accurate are the results of the scheduler. Standard deviation is a measure of the spread or dispersion of a set of data. It is calculated by taking the square root of the variance. In other words, the more widely the values are spread out, the larger the standard deviation. Sample variance is a measure of the spread of or dispersion within a set of sample data. The sample variance is the sum of the squared deviations from their average divided by one less than the number of observations in the data set. Measuring variability is of special importance in inference. The range of a set of data is the difference between the largest value and the smallest. Although frequency distributions can take on almost any shape or form, most of the distributions we meet in practice can be described fairly well.

If we refer to the table 3 and 4 for deviation in burst time and the range of values in best and worst cases then we could easily figure out that the varied process burst time gives more accurate prediction in best case. But when it is less spread among its values as inputs then it is little difficult for RNN to predict the accurate output. In the same way if we refer to table 5 and 6 and table 7 and 8 results RNN works well in predicting if different size data bundled together and if there are more differences in initialized data segment (. data), and size of readonly data (. rodata) of processes so our RNN prediction goes well with it. As opposed to this if we observe the data of worst cases of each input then combination of more similar data as similar kind of processes scheduled together then RNN gives the inaccurate result as RNN has failed to relate it and predict it about the output. In all cases if there is more deviation and variance and if data at a time covers maximum range then it gives the best results.

Table 3 Pattern of input 'bt (burst time)' in Best case for seven time steps

Standard Deviation	Sample Variance	Range
1556. 399	2422378	5550
3387. 001	11471775	9280
4030. 324	16243511	9570
1294. 161	1674853	3020
1294. 161	1674853	3020
2988. 746	8932603	9240
2988. 746	8932603	9240

Table 4 Pattern of input 'bt (burst time)' in Worst case for six time steps

Standard Deviation	Sample Variance	Range
778. 0335	605336. 1	2020
1096. 563	1202450	2970
858. 1299	736387	2996
1105. 918	1223054	3280
858. 3079	736692. 5	2963
1079. 872	1166123	2923

Table 5 Pattern of input '. data ' in Best case for seven time steps

Standard Deviation	Sample Variance	Range
45. 053055	2029. 778	144
16. 384274	268. 4444	52
7. 7746025	60. 44444	24
19. 843835	393. 7778	64
19. 843835	393. 7778	64
44. 993827	2024. 444	144
44. 993827	2024. 444	144

Table 6 Pattern of input '. data' in Worst case for six time steps

Standard Deviation	Sample Variance	Range
16. 61325	276	56
15. 6205	244	48
8. 717798	76	28
12. 66886	160. 5	41
11. 14426	124. 194	35
13. 78405	190	42

Table 7 Pattern of '. rodata' in Best case for seven time steps

Standard Deviation	Sample Variance	Range
95. 711952	9160. 778	311
100. 72294	10145. 11	288
106. 53925	11350. 61	311
44. 989196	2024. 028	118
34. 989196	2020. 028	115
81. 348701	6617. 611	253
80. 348501	6517. 611	243

Table 8 Pattern of '. rodata' in Worst case for six time steps

Standard Deviation	Sample Variance	Range
35. 769323	1279. 444	117
54. 80293	3003. 361	152
14. 503831	210. 3611	44
75. 707405	5731. 611	211
56. 682841	3212. 944	194
55. 493243	3079. 5	175

CONCLUSION

The purpose of this study is to analyze a Recurrent NN model for MLFQ scheduler and evaluate the performance of this scheduler in terms of average turnaround time. This study was undertaken to apply Recurrent NNs to the recognition of scheduled process patterns. Each process runs for a certain time, then scheduler chooses which process to execute and for what amount of time. The scheduler must balance fairness and efficiency to offer the appropriate behavior. It was found that attributes of process we used over here such as burst time of process, size of executable instructions in process, size of the readonly data in the process and size of initialized data in the process affect on average turnaround time. By learning the behavior of processes in this way we can minimize turnaround time. According to our experimental data we find that scheduling of different kind of processes together affects the average turnaround time. RNN we used here helps us to learn the pattern of processes grouped together and predict about the effective average turnaround time. It also suggests TQ (Time Quantum) to be given for the set of processes as we are providing enough information about processes. The RNN models are not only important for the forecasting of time series but also generally for the control of the dynamical system. Furthermore, a new method for examining scheduler was established by searching for temporal context of processes. It became clear that the method was effective in minimizing average turnaround time in MLFQ scheduler.

FUTURE SCOPE

Our future work will include extending our technique to evaluate performance of scheduler for other kind of processes. So for that we need to study various parameters of the that process affecting turnaround time or response time whichever is required to minimize according to application. This could be useful to create more optimized rule-based schedulers after experimentation with the Recurrent Neural Network scheduler..

REFERENCES

[1] Fletcher G. P and Hinde, "Interpretation of Neural Networks as Boolean Transfer Functions", Elsevier B. V. Vol. 7, No. 3, pp 207-214. (1994), pp 207-214

[2] Shlomi Dolev and Avi Mendelson and Igal Shilman, "Semantical Cognitive scheduling", Fronts (2012).

[3] Abbas Noon and Ali Kalakech and Seifedine Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average", IJCSI 3, 1 ISSN : 1694-0814 (2011).

[4] F. J. Corbato, M. M. Daggett, R. C. Daley, "An Experimental Time-Sharing System", SJCC Paper of IFIPS. (1962).

[5] A. Silberschatz and P. B. Galvin and G. Gagne, "Operating System Concepts", Wiley Inc. (2009).

[6] Tal B., "Neural Network Based System of Leading Indicators", CIBC World markets (2003).

[7] Ehsan Saboori and Shahriar Mohammadi and Shafigh Parsazad, "A new scheduling algorithm for server farms load balancing" (2010).

[8] J. R. Quinlan, "Comparing connectionist and symbolic learning methods. In Computational Learning Theory and Natural Learning Systems", MIT Press (1994), 445 456.

[9] Stuart J. Russel and Peter Norvig, "Artificial Intelligence: A Modern Approach", Prentice-HallInc (1995).

[10] Jeffrey L. Elman, "Finding structure in time. Cognitive Science" (1990), 179–11.

[11] Russell S. and Norvig, "Artificial Intelligence: A Modern Approach", London: Prentice Hall (2003).

[12] Atul Negi Senior Member IEEE and Kishore Kumar P., "Applying Machine Learning Techniques to improve Linux Process Scheduling", University of Hyderabad (2003).

[13] Deepali Maste, Dr. Leena Ragha, Prof Nilesh Marathe, " Intelligent Dynamic Time Quantum Allocation in MLFQ Scheduling ", International Journal of Information and Computation Technology, Volume 3, number 4, ISSN 0974-2239, 2013