Adaptive RoundRobin

Dang Vu Tuan Kiet[1,2], Truong Binh Minh[1,2], Nguyen Do Trung Kien[1,2], and Tran Nguyen Bao[1,2]

[1] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Vietnam
[2] Vietnam National University Ho Chi Minh City, Vietnam

Abstract. The traditional Round Robin scheduling is fair but can be slow due to frequent context switching with short time slices. It can also neglect short processes if the time slices are long. This research introduces a new Adaptive Round Robin algorithm that uses a deep learning technique called self-attention to automatically adjust the time slice based on what the user wants to prioritize and the current mix of processes. Early results show it can achieve 88.8% accuracy while still being fast with 3,000 FPS. Code is available at here.

1 Introduction

CPU scheduling plays a critical role in ensuring your computer runs smoothly. It acts like a traffic conductor for the CPU, allocating its time efficiently among various programs running on your system. This prevents the CPU from being idle while processes wait, keeps things fair by ensuring all programs get a turn, and minimizes wait times which speeds things up overall. CPU scheduling even allows prioritizing important tasks and creates the illusion of multitasking on a single CPU. In short, it's the behind-the-scenes hero that keeps your computer running efficiently and responsively.

The round-robin algorithm is a CPU scheduling approach that prioritizes fairness for multiple programs running on a system. It works similarly to a round-robin game where everyone gets a turn for a designated amount of time. In the CPU world, processes take turns using the CPU for a short, fixed-time slice called a quantum. If a process doesn't finish its work within its allotted quantum, it's politely put back in line to wait for another turn. This cyclical method ensures each process gets a chance to run regularly, preventing any single program from monopolizing the CPU and keeping the system responsive for users, even if some tasks require more time to complete.

This approach offers several advantages. First, it guarantees fairness by giving all processes an equal opportunity to use the CPU. No single process can starve for resources waiting for a CPU hog to finish. Additionally, round-robin scheduling is relatively simple to implement and understand, making it a popular choice for various systems. It also provides good responsiveness, as processes get serviced frequently, leading to a faster-perceived performance for users.

However, the round-robin algorithm also has its drawbacks. A crucial factor in its effectiveness is the selection of the quantum time. Setting the quantum too short can lead to excessive context-switching overhead. Context switching involves saving the state of the current process and loading the state of the next process, which takes time and slows down overall system performance. On the other hand, a quantum time that's too long can defeat the purpose of fairness. If a process has a much longer burst time (total execution time) than the quantum, it ends up dominating CPU usage for extended periods, essentially behaving like a non-preemptive system and potentially causing starvation for shorter processes. Finding the optimal quantum time depends on the specific workload and can be a challenge for the operating system.

This work proposes a novel framework that tackles this challenge by utilizing the power of deep learning models to create an adaptive system for calculating quantum time. This system considers two crucial aspects: the user's desired performance metric (think responsiveness or fairness) and the real-time makeup of the process queue (the types and execution times of processes waiting for the CPU). By analyzing this data, the deep learning model can dynamically adjust the quantum time, ensuring efficient resource allocation and prioritizing tasks based on user preference. This framework has the potential to significantly improve the effectiveness of round-robin scheduling by striking a balance between fairness and responsiveness in a way that adapts to real-world situations.

2 Related work

Round Robin: Recent years, there are several work on how to find the optimal quantum time for RoundRobin Scheduling algorithm. These work can be devided into two type of algorithm. The first type calculate a optimal quantum time before start to schedule, [5] use linear programming to solve for a optimal quantum time base on the process list, other work [1], [6] use some statistic value of burst time of process list to choose quantum time like average, median, max min dispersion,... The second type dynamic adjust the quantum time during scheduling. Some works can be found in [7], [3], [4].

Deep Learning and Sequential Model: The field of deep learning has exploded in recent years, driven by a trifecta of factors: massive datasets, ever-increasing computing power, and the ability of deep learning models to handle complex tasks without needing explicit programming. This has led to breakthroughs in image recognition, natural language processing, and self-driving cars. Within deep learning, sequential models have seen remarkable progress. Building on the foundation of Recurrent Neural Networks (RNNs), Transformers [8] have revolutionized the field. Their ability to capture long-range dependencies has pushed the state-of-the-art in tasks like machine translation and text generation. Even in vision tasks [2] , the concept of Transformers is proving fruitful by treating image patches like tokens, achieving cutting-edge results when applied.

3 Adaptive RoundRobin

3.1 Problem Definition

Consider a set, denoted by $\mathcal{M}$ that encompasses all possible methods for calculating the quantum time for a list of upcoming processes. We're also interested in a metric vector, $\mathcal{W} \in [0,1]^5$, which defines the relative importance of five key metrics: average response time, average waiting time, average turnaround time, and variance of response time.

Imagine a specific list of processes where the burst time and arrival time of each process are known. If $\mathcal{M}$ is sufficiently large and well-designed, then there will always exist at least one method within this set that minimizes the dot product between the metric vector $\mathcal{W}$ and the corresponding vector containing the calculated metric values for that specific process list.

However, design a good enough set $\mathcal{M}$ is nearly impossible so we need to approximate it with following set and accept the best method m in it which is implied that:

$$m = \underset{m \in \mathcal{M}}{\operatorname{argmin}} \mathcal{W} \cdot m_W$$

To enhance the Round Robin algorithm's performance , this work will concentrate on building a model that can choose the best method $m \in \mathcal{M}$ for a list of process and metric vector $\mathcal{W}$ with predefined method set $\mathcal{M}$.

3.2 Architecture

Since the number of processes in each sample can vary, addressing this challenge requires a sequential model. Inspired by [8] and [2] approaches, we propose a sequential classification model. This model takes a process list and a metric vector as input and aims to identify the optimal method $m \in \mathcal{M}$.

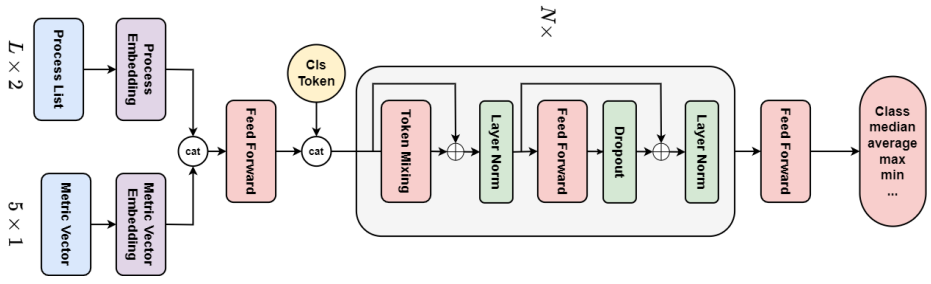

Fig. 1: Overall architecture of Adaptive RoundRobin

At first, We break down the process list and metric vector into a sequence of elements. Each process and each metric value is treated as a token, similar to a word in a sentence. These tokens are first passed through separate embedding

layers, which are essentially simple linear transformations that map them into a higher-dimensional space. Subsequently, the embedded process and metric tokens are concatenated (joined together) and fed into a two-layer Feed-Forward Neural Network (FFN). This FFN learns non-linear relationships within the combined data, allowing the model to capture complex patterns for optimal method selection.

After that, we follow [2] and feed embedding sequential to several encoder block with multi head attention as token mixing for feature combination between each token (each process) of the input. Here the learnable token for classification is also use and we padding it to the end of the sequential when training and inference. (the reason for choosing this will be revisit in ablation study). The detail architecture can be see in Fig. 1.

4 Experiments

4.1 Data Generation

To generate data for training the proposed model, we first randomly generate several sample which contain a list of process and correspond metric vector. Each process in process list contain two value of a process that is burst time and arrival time (assume that we know these two values). Then we turn by turn calculate all five metrics value for each method in method set $\mathcal{M}$ and label the best method with the one return the minimum dot product between metric vector and its metric value.
The training set is contain 19000 samples with number of process in each sample is variable and randomly between 3 and 20. The test set is also randomly from the same distribution and contain 1000 samples.
We choose method set $\mathcal{M}$ contain 10 method in Table 1 and we found that the data is unbalanced which turn out that some method in method sets is more better than other method. As shown in Fig. 2, harmonic mean and min max dispersion, each method occupy around 42% of sample, which mean if we only use one of these two method the accuracy of the model is around 42%, and our propose method at least need to beat this accuracy.

4.2 Result

Detail Implemnetation. For efficient, we design our model as small as possible with only 15755 parameters. The dimension of hidden feature, d_model is equal to 32 and we just use 1 encoder block. For easy to optimize we use cross entropy loss function and Adam optimizers with $lr = 0.0001$ and weight decay 0.000001.
Result and Discussion. Our model achieve 88.8% accuracy on test set which is quite high and can beat the best method (in previous part) which is only achieve almost 42% accuracy. Our model can also run in real time with 3000 FPS on GeForce RTX 3060.

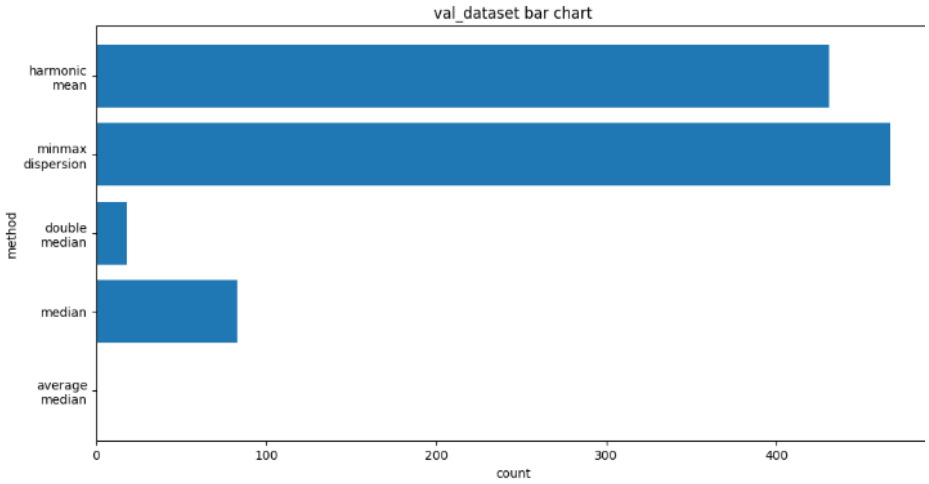

Fig. 2: Validation Data Distribution

Table 1: All method in method set $\mathcal{M}$. B_i is burst time of process i and B is burst time list.

Method	Equation
Median	$Median(B)$
Average	$Mean(B)$
Average Max Mean	$(Mean(B) + Max(B)) \times 0.5$
Median Max Geometric Mean	$\sqrt{Median(B) \times Max(B)}$
Median Max Mean	$(Median(B) + Max(B)) \times 0.5$
Average Median Mean	$(Median(B) + Mean(B)) \times 0.5$
Average Median Geometric Mean	$\sqrt{Median(B) \times Mean(B)}$
Min Max Dispersion	$Max(B) - Min(B)$
Harmonic Mean	$len(B) / \sum_i \frac{1}{B_i}$
Double Median	$2 \times Median(B)$

4.3 Ablation Study

We found that there are two important setting that can affect the performance of the model: position encoding and position of classification token.
Our experiments, as shown in Table 2, reveal a significant performance difference depending on the CLS token placement. Placing the CLS token in the middle of the sequence yields considerably lower results compared to placing it at the end. However, this performance gap can be mitigated by incorporating positional encoding. This technique, inspired by [8] , injects information about the token's relative position within the sequence. In our case, using sinusoidal positional encoding led to a noteworthy improvement of 5.1%. Interestingly, combining the final CLS token placement with positional encoding resulted in a less effective approach. This is because positional encoding introduces artificial spacings between elements in the sequence that don't truly exist. For instance, shuffling the order of the process list would disrupt the positional encoding for each process, rendering it inaccurate. This is a key advantage of self-attention mechanisms over traditional sequential models like RNNs or LSTMs. Self-attention can effectively handle non-positional sequential inputs, making it a more suitable choice for this specific task.

Table 2: Effect on Model Selection

				Accuracy
Self-Attention	Position Encoding	MiddleCls Token	LastCls Token	
✓	x	✓	x	79.6
✓	x	x	✓	88.8
✓	✓	✓	x	84.7
✓	✓	x	✓	87.2

5 Conclusion

This work introduces a novel Adaptive Round Robin scheduling algorithm that dynamically selects quantum time based on the process list and user-defined metric vectors. This approach allows for customization based on specific user needs. Our initial evaluation demonstrates promising performance, suggesting Adaptive Round Robin as a valuable tool for enhancing process scheduling. Future research will focus on validating the model's generalizability across diverse data generation methods. We are also committed to continuous improvement, striving to make the model even more effective, robust, and efficient.

6 Acknowledgement

We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

A huge thanks to TickLab Student Organization for giving us the computation resource that made this project possible.

References

1. Abdelkader, A., Ghazy, N., Zaki, M.S., ElDahshan, K.A.: Mmmrr: a modified median mean round robin algorithm for task scheduling https://inass.org/wp-content/uploads/2022/09/2022123153-2.pdf
2. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv (Cornell University) (Jan 2020). https://doi.org/10.48550/arxiv.2010.11929, https://arxiv.org/abs/2010.11929
3. HIRANWAL, S., ROY, D.K.: Adaptive round robin scheduling using shortest burst approach based on smart time slice https://csjournals.com/IJCSC/PDF2-2/Article%208.pdf
4. H.S.Behera, Mohanty, R., Nayak, D.: A new proposed dynamic quantum with re-adjusted round robin scheduling algorithm and its performance analysis https://www.ijcaonline.org/volume5/number5/pxc3871291.pdf
5. Saeidi, S., Alemi Baktash, H.: Determining the optimum time quantum value in round robin process scheduling method https://www.mecs-press.org/ijitcs/ijitcs-v4-n10/IJITCS-V4-N10-8.pdf
6. Sakshi, N., Sharma, C., Sharma, S., Kautish, S., Alsallami, S.a.M., Khalil, E., Mohamed, A.W.: A new median-average round robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time. Alexandria Engineering Journal /Alexandria Engineering Journal **61**(12), 10527–10538 (Dec 2022). https://doi.org/10.1016/j.aej.2022.04.006, https://doi.org/10.1016/j.aej.2022.04.006
7. Sharma, A., Kakhani, M.G.: Analysis of adaptive round robin algorithm and proposed round robin remaining time algorithm https://inass.org/wp-content/uploads/2022/09/2022123153-2.pdf
8. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. arXiv (Cornell University) (Jan 2017). https://doi.org/10.48550/arxiv.1706.03762, https://arxiv.org/abs/1706.03762