Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2021.020207
ISSN 2790-0916 Vol. 2, Issue 2: 51-58

Coupled Distributed Systems Considering Multitasking Dynamic Scheduling Algorithms

Venpaty Velmugan*

Universiti Teknologi MARA, Malaysia

**corresponding author*

Keywords: Multitask Scheduling, Coupled Distributed System, Dynamic Scheduling, System Construction

Abstract: With the rapid development of Internet technology and the comprehensive arrival of the era of big data, the development of enterprise business scale and the rise of logic complexity make coupled distributed systems play an increasingly important role in enterprise-level applications. The multi-task dynamic scheduling scheme based on the proposed method has problems such as poor manageability, poor task scheduling ability and poor usability. Therefore, this paper studies and constructs the coupled distributed system considering the multi-task dynamic scheduling. This paper firstly introduces the relevant theories. In the system theory part, it mainly introduces the scheduling module, followed by the system design. In the system design part, the task scheduling is mainly designed. Finally, the system implementation is carried out. Scheduling accuracy is analyzed.

1. Introduction

With the rapid development of Internet technology and the full arrival of the era of big data, for large-scale enterprise-level applications, due to the complexity of daily business scenarios, the problem of multi-task scheduling is increasingly valued by enterprises and has become an enterprise-level application. An indispensable component in the application [1-2]. The problem of multitask scheduling in coupled distributed systems has always been a research hotspot. Considering the operating cost and practical economic benefits of multitask scheduling, some existing research works aim to reduce the system cost of task transmission or computation [3].

At present, many scholars have carried out in-depth research on multi-task dynamic scheduling and coupled distributed systems and achieved good results. For example, researchers such as Almeida A F use three different architectural patterns to decompose a monolithic application into microservices, and use detailed metrics to compare the two architectural styles, and argue that distributed systems provide a shift from the traditional way of building systems, the monolithic

system considered as the ancestor of microservices cannot meet the needs of today's large and complex applications [4]. Scholars such as Limam A proposed an enhanced faulty node detection method using interval weighting factors, which uses pseudo-random BCH codes for distributed network control systems to monitor node behavior, and the master node collects the single-bit BCH of each slave node. The replacement of the cyclic redundancy check CRC code by the code, this scheme can be used to detect and prevent serious damage caused by node failure [5]. Although there are many researches on multi-task scheduling and coupled distributed systems, there are few studies on coupled distributed systems considering multi-task dynamic scheduling. Multi-task dynamic scheduling is of great significance for the construction and research of coupled distributed systems [6-7].

The construction of the coupled distributed system in this paper is based on the consideration of multi-task dynamic scheduling. The structure of this paper can be divided into three parts, including related theory, system design and system implementation. In the related theory part, it mainly introduces the scheduling module and the multi-task cooperative distribution strategy. In the system design part, it mainly includes the design of loosely coupled system architecture, task scheduling system and adaptive scheduling strategy. In the system implementation part, the accuracy of task scheduling is tested and the scalability of adaptive scheduling is analyzed.

2. Related Theories

2.1. Scheduling Module

The main application of a distributed system is to provide a suitable computing platform for tasks [8]. In what form a task is represented in this platform, there must be a standardized agreement between the scheduling platform and the user [9]. In the coupled distributed simulation system, tasks are submitted in the form of working conditions. Task scheduling refers to the scheduling of tasks in working conditions. First, the management method of working conditions in this system must be defined, and then the management method of tasks in working conditions. In this way, the scheduling of the situation can be started [10]. Figure 1 shows the design of the scheduling module:

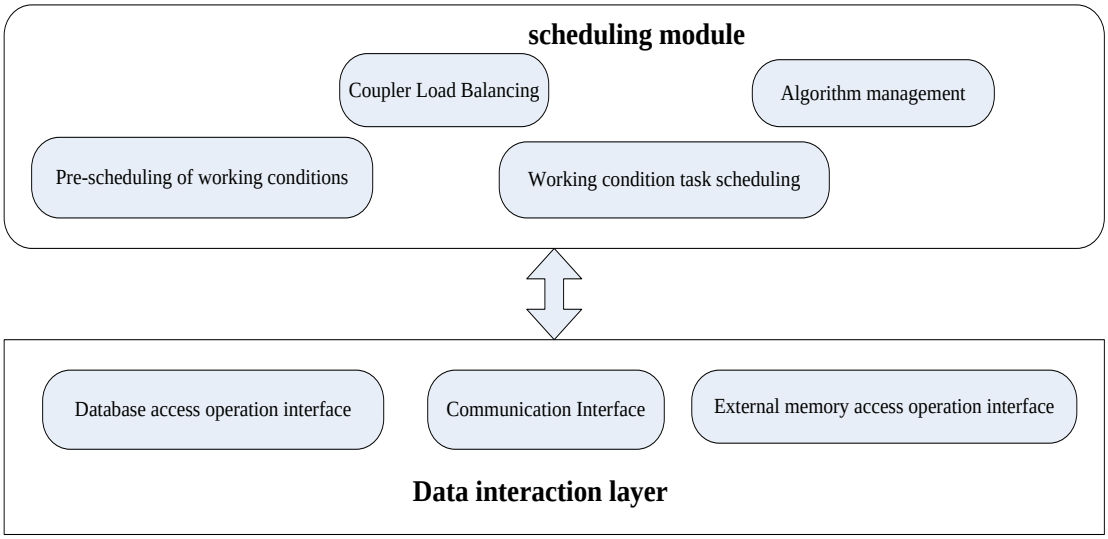

Figure 1. Design diagram of scheduling module

2.2. Multi-Task Collaborative Resource Allocation Strategy

It is impossible for the system to treat each resource scheduling data processing computing task equally, so it is necessary to weight each task to distinguish the importance of the task [11]. In the task-level resource scheduler in the coupled distributed system, different tasks are independent of each other, and their resources are isolated in the distributed cluster [12]. Because stream data processing tasks occupy resources in the system through preemption, that is, on a first-come, first-served basis [13-14]. So when there are multiple tasks in a distributed cluster, the first task to enter the system will get as many resources as it needs.

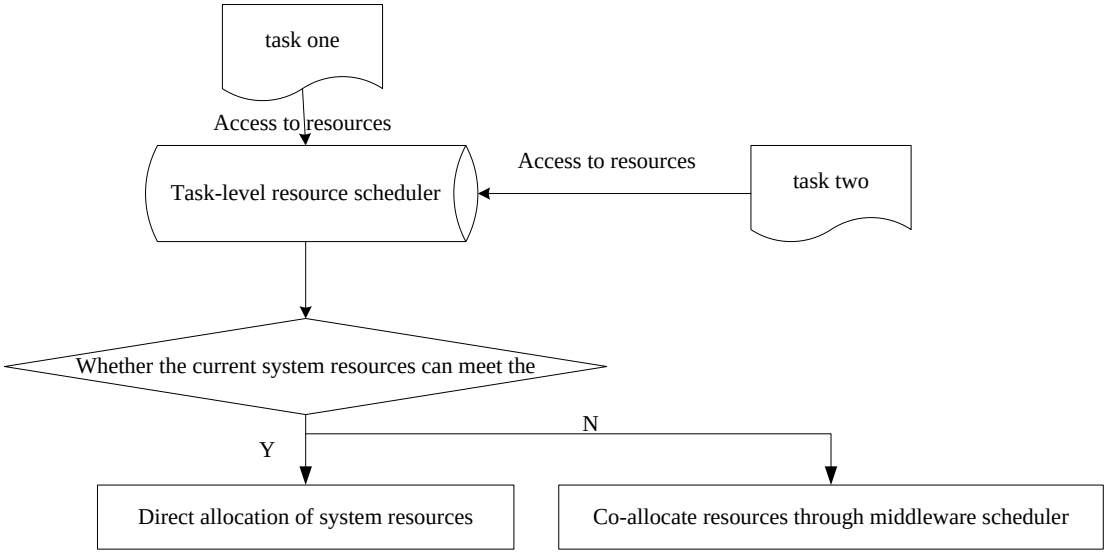

Figure 2. Multi-task collaborative resource allocation diagram

As shown in Figure 2, when different stream data processing tasks enter the cluster, the middleware scheduler needs to determine whether the current system resources can meet the task requirements. If it can be satisfied, the system will allocate resources directly through the middleware scheduler to meet the task requirements. If the current system cannot fully meet the resource requirements of each data processing task, the middleware scheduler will compare the weighted delay ratio of the streaming data processing tasks of each allocation scheme, and then calculate which scheme is more reasonable for the current system.

3. System Design

3.1. System Loosely Coupled Architecture Design

The coupled distributed system adopts the distributed micro-service architecture design as a whole, which can effectively improve the flexibility, reusability and scalability of the distributed timing task scheduling system [15]. The system is generally divided into four microservices, registry and database. The four microservices include task execution microservices, task scheduling microservices, task processing microservices and task alarming microservices. The functional modules of the distributed timing task scheduling system are shown in Figure 3. The system can be divided into four modules including a unified management module, a task scheduling module, a task processing module, and a task execution module. The unified management module includes

task basic management and task scheduling management, the task scheduling module includes task preemption, task activation and task triggering, the task processing module includes fragmentation processing and DAG processing, and the task execution module includes task registration and task execution.

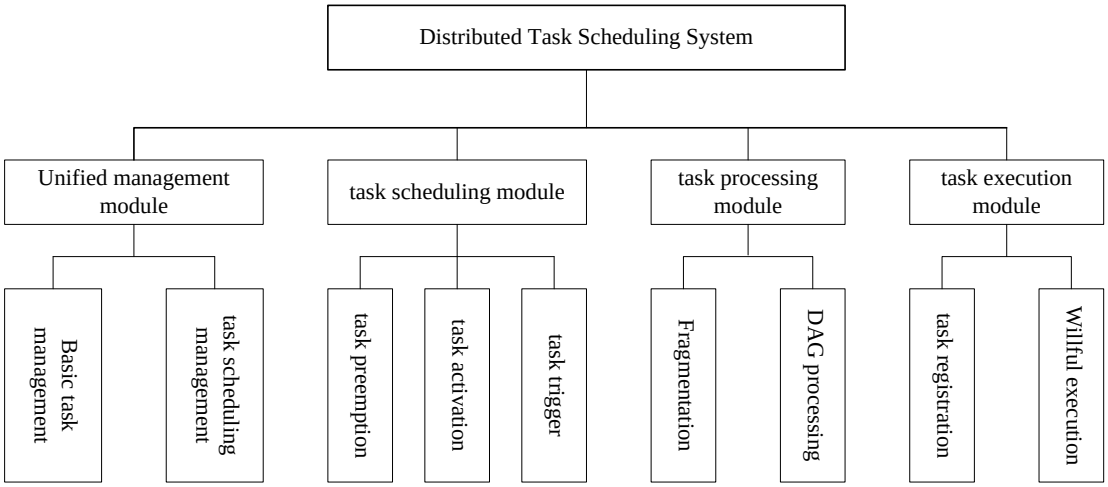

Figure 3. System function block diagram

3.2. Task Scheduling System

The number of iterations required for the model to converge varies for different types of tasks. Experience shows that the model will have better performance as the number of iterations increases. However, too many iterations will also lead to problems such as overfitting of the model and a low proportion of computing time. In addition, static scheduling cannot sense the dynamic changes of computing node performance, cannot achieve task migration, and has no good fault tolerance measures. Once the scheduling is abnormal, the training can only be restarted, which greatly wastes system resources and time. Therefore, this section proposes a task scheduling system for a coupled distributed system. The scheduling system can improve the portability, scalability and adaptability of training.

The task scheduling system can evaluate performance differences according to the resource statistics of each computing node. To represent node computing performance, the underlying task is executed multiple times over a period of time and the computing performance is sampled. Formula (1) and formula (2) are the calculation formulas for the actual performance of the node:

$$D_f = \alpha * \frac{1}{m} \sum_{g=n-m-1}^{m} D_{f,g} + \beta * D_{f,n}, \quad n \geq m+1 \tag{1}$$

$$D_f = D_{f,g}, \quad g < f+1 \tag{2}$$

D_f represents the final performance index of node f at the nth sampling point, $D_{f,g}$ is the final performance index of m sampling points before the current sampling point, and,$D_{f,n}$ represents the performance index of the basic task running at the current sampling point. If f is smaller than the window value of forward sampling,$D_{f,n}$ are directly used as the final performance index of this sampling. α and β are used as the weights of historical sampling and current sampling, and the sum of the two is 1. After the final performance calculation method of the node at the sampling point is given, the algorithm is called to complete the node and its resource allocation:

3.3. Adaptive Scheduling Strategy

Adaptive scheduling strategy is a scheduling strategy proposed on the basis of dynamic scheduling strategy, which can further reduce the synchronization time and endow the scheduling system with adaptability [16-17]. The adaptive scheduling strategy can realize the load balancing of data parallel or model parallel training. Load balancing, that is, considering the performance of each computing node to achieve adaptive allocation of tasks [18]. Redistributing tasks through load balancing is beneficial to improve the parallel efficiency of computing nodes. To address the dynamic performance differences arising during training, this section proposes an adaptive scheduling strategy. This strategy balances the dynamic performance differences between nodes by adjusting the amount of tasks.

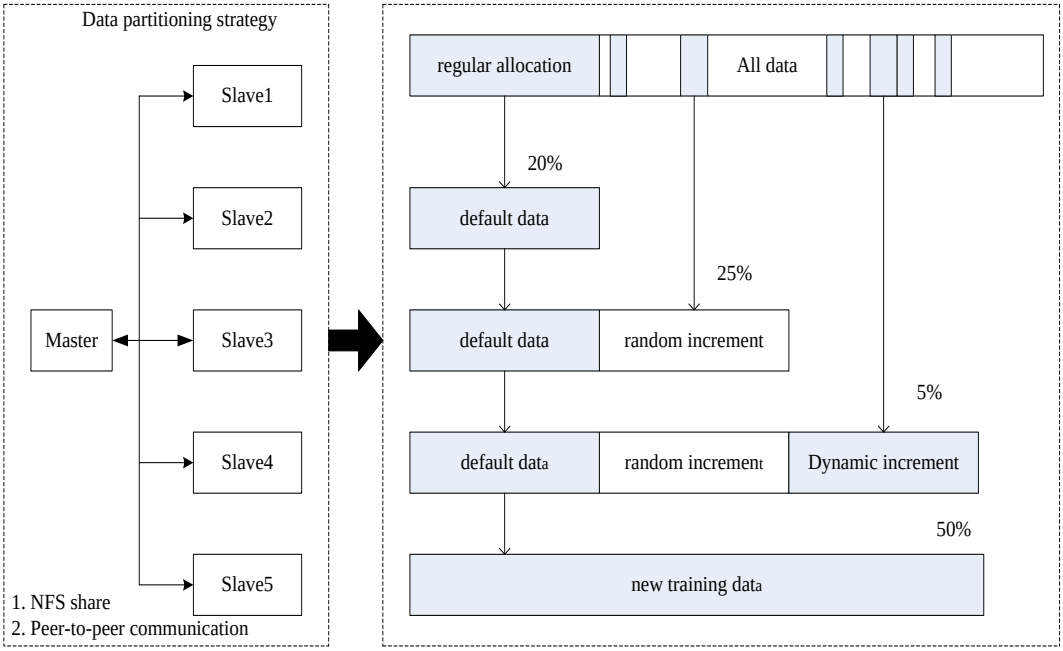

Figure 4. Using data partitioning to generate local training data graph

As shown in Figure 4, there are 5 worker nodes in the figure, so each worker node should hold about 20% of the training data of the complete dataset. Low number of iterations can lead to poor model accuracy or underfitting: high number of iterations can lead to model overfitting and poor performance on the test set. In order to improve the efficiency of task execution, the timing of outer iteration termination is determined by the task scheduling system and the specific number of iterations is not set.

4. System Implementation

4.1. Scheduling Accuracy Test

The test goal of the scheduling accuracy test is to test whether the scheduled tasks can be accurately scheduled at a predetermined time point under different concurrency of the distributed system. The test indicator is whether the average scheduling delay of the scheduled tasks can be controlled within 1s. The scheduling accuracy test of the system's single task scheduler is divided

into three groups, namely the single-thread group, the multi-thread group with 3 threads, and the multi-thread group with 5 threads. The following operations are performed for each test group: 10 rounds of tests are performed in scenarios with different concurrency of scheduled tasks, and the average delay time triggered by task scheduling is finally counted. The test results are shown in Figure 5.

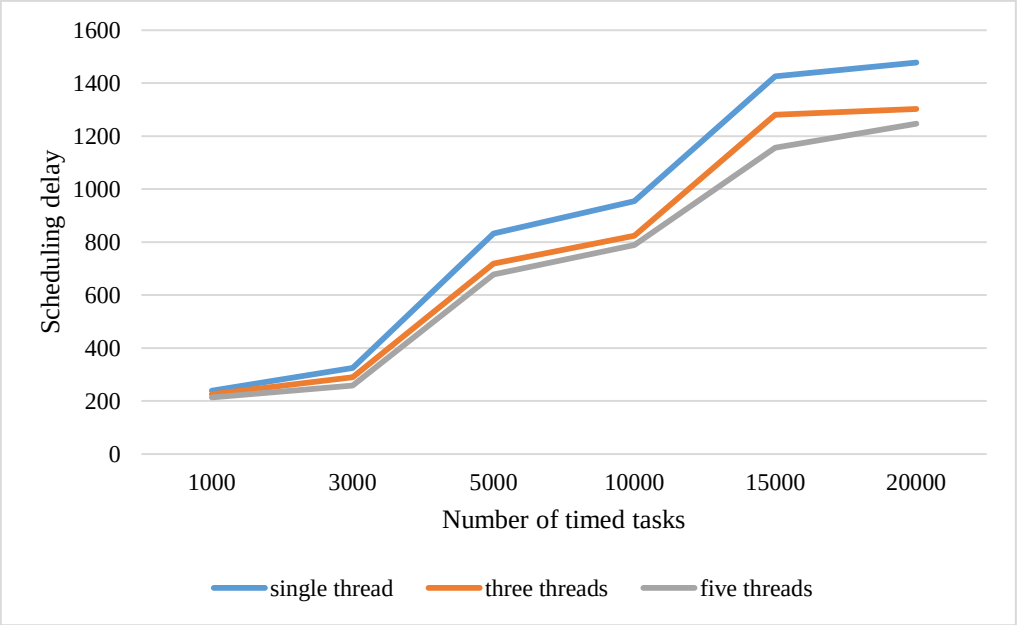

Figure 5. The relationship between the scheduling delay of a single task scheduler and the number of concurrent tasks

Based on the experimental results in Figure 5, the following conclusions can be drawn: The multi-task scheduling strategy of the coupled distributed system has good scheduling accuracy, and can control the average scheduling delay of timed tasks within ls under different task concurrency, basically within the acceptable range for users.

4.2. Scalability of Adaptive Scheduling

The adaptive scheduling strategy imposes penalty tasks on some nodes through data division to achieve node performance balance. In addition, the data partition strategy relies on the data augmentation function to expand the local task volume, and terminates the training in time through the early stop strategy to improve the convergence speed.

Table 1. Specific distribution of iteration time

Scheduling method	Training time(s)	Aggregation time(s)	Communication time(s)	Cumulative iteration
Adaptive scheduling	363	26	16	24
Non-adaptive scheduling	353	52	33	50

Table 1 shows the time distribution of two different scheduling strategies. When non-adaptive

scheduling is used, the average training time per round is about 7s, and the total training time is 353s. When using adaptive scheduling and applying data partitioning, the average training time per round is about 15s, and the total training time is 363s. The adaptive scheduling strategy uses the data partition strategy to balance the performance differences of computing nodes, and appropriately increases the amount of training data through the load factor to reduce the number of iterations and achieve the goal of convergence as soon as possible. From the above analysis, it can be seen that data partitioning has good performance and scalability, and can further realize self-adaptation on the basis of realizing dynamic scheduling strategy. This kind of self-adaptation is manifested in that the dynamic performance changes and iteration times of the adaptive computing nodes are adaptive to different tasks.

5. Conclusion

In this paper, the construction of the coupled distributed system takes into account the needs of multi-task dynamic scheduling, and meets the needs of the development of enterprise multi-task scheduling. In the system implementation part, through the analysis of the accuracy of system task scheduling, it is found that the multi-task scheduling accuracy of the coupled distributed system is good and can meet the needs of users. It has good performance and scalability, and can further realize self-adaptation on the basis of realizing dynamic scheduling strategy. Although the coupled distributed system proposed in this paper takes into account the dynamic scheduling of multitasking, there are still many deficiencies that need to be improved.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Wang X, Shahidehpour M, Jiang C, et al. Resilience Enhancement Strategies for Power Distribution Network Coupled With Urban Transportation System. Smart Grid, IEEE Transactions on, 2019, 10(4):4068-4079. https://doi.org/10.1109/TSG.2018.2848970

[2] Juliza J, Abdul R R, Hafiz F M, et al. Analysis on the Effect of Sensor Views in Image Reconstruction Produced by Optical Tomography System Using Charge-Coupled Device. IEEE transactions on image processing : a publication of the IEEE Signal Processing Society, 2018, 27(4):1689-1696. https://doi.org/10.1109/TIP.2017.2783620

[3] Mmc A, Vndc A, Sm B, et al. Asymptotic stability for a strongly coupled Klein-Gordon system in an inhomogeneous medium with locally distributed damping - ScienceDirect. Journal of Differential Equations, 2020, 268(2):447-489. https://doi.org/10.1016/j.jde.2019.08.011

[4] Almeida A F, Cavalcanti M M, Gonzalez R B, et al. Uniform decay rate estimates for the coupled semilinear wave system in inhomogeneous media with locally distributed nonlinear damping. Asymptotic Analysis, 2019, 117(41):1-45. https://doi.org/10.3233/ASY-191547

[5] Limam A, Boukhatem Y, Benabderrahmane B. General Decay Result for a Type III Thermoelastic Coupled System with Acoustic Boundary Conditions in the Presence of Distributed Delay. Mathematical Physics Analysis and Geometry, 2021(2):175-200. https://doi.org/10.15407/mag17.02.175

[6] Falsone A, Prandini M. A Distributed Dual Proximal Minimization Algorithm for Constraint-Coupled Optimization Problems. IEEE Control Systems Letters, 2020, 5(1):259-264.

[7] Webler C M, Zanchetta J P. Exponential stability for the coupled Klein–Gordon–Schrdinger equations with locally distributed damping in unbounded domains. Asymptotic Analysis, 2021, 123(3-4):289-315. https://doi.org/10.3233/ASY-201634

[8] AFD Almeida, MM Cavalcanti, JP Zanchetta. Exponential stability for the coupled Klein-Gordon-Schrdinger equations with locally distributed damping. Evolution Equations and Control Theory, 2019, 8(4):847-865. https://doi.org/10.3934/eect.2019041

[9] AF Almeida, MM Cavalcanti, JP Zanchetta. Exponential decay for the coupled Klein-Gordon-Schr ̈odinger equations with locally distributed damping. Communications on Pure & Applied Analysis, 2018, 17(5):2039-2061. https://doi.org/10.3934/cpaa.2018097

[10] Talaei B, Jagannathan S, Singler J. Output Feedback-Based Boundary Control of Uncertain Coupled Semilinear Parabolic PDE Using Neurodynamic Programming. IEEE Transactions on Neural Networks & Learning Systems, 2018, 29(4):1263-1274.

[11] Aguilar C S. Scheduling distributed clusters of parallel machines: primal-dual and LP-based approximation algorithms. Computing reviews, 2018, 59(11):605-605.

[12] Irom K, Kazi A, Akter T. Improvised Priority based Round Robin CPU Scheduling. International Journal of Computer Applications, 2018, 179(32):7-16. https://doi.org/10.5120/ijca2018916722

[13] Sohrawordi M, Ali U, Uddin M P, et al. A Modified Round Robin Cpu Scheduling Algorithm With Dynamic Time Quantum. International Journal of Advanced Research, 2019, 7(2):422-429. https://doi.org/10.21474/IJAR01/8506

[14] Popovic M, Kordic B, Popovic M, et al. Online algorithms for scheduling transactions on python software transactional memory. Serbian Journal of Electrical Engineering, 2019, 16(1):85-104. https://doi.org/10.2298/SJEE1901085P

[15] Qamhieh M, George L, Midonnet S. Stretching algorithm for global scheduling of real-time DAG tasks. Real Time Systems, 2019, 55(1):32-62. https://doi.org/10.1007/s11241-018-9311-1

[16] Shetty S C. Machine Learning Approach to Select Optimal Task Scheduling Algorithm in Cloud. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021, 12(6):2565-2580.

[17] Solayman H E. A Comparison of Scheduling parallel program tasks based on Java Applet. International Journal of Advanced Trends in Computer Science and Engineering, 2020, 2(9):1394 – 1403.

[18] Paul T, Hossain R, Samsuddoha M. Improved Round Robin Scheduling Algorithm with Progressive Time Quantum. International Journal of Computer Applications, 2019, 178(49):30-36. https://doi.org/10.5120/ijca2019919419