World Academy of Science, Engineering and Technology
International Journal of Energy and Power Engineering
Vol:18, No:5, 2024

Energy-Aware Scheduling in Real-Time Systems: An Analysis of Fair Share Scheduling and Priority-Driven Preemptive Scheduling

Su Xiaohan, Jin Chicheng, Liu Yijing, Burra Venkata Durga Kumar

***Abstract*—**Energy-aware scheduling in real-time systems aims to minimize energy consumption, but issues related to resource reservation and timing constraints remain challenges. This study focuses on analyzing two scheduling algorithms, Fair-Share Scheduling (FFS) and Priority-Driven Preemptive Scheduling (PDPS), for solving these issues and energy-aware scheduling in real-time systems. Based on research on both algorithms and the processes of solving two problems, it can be found that FFS ensures fair allocation of resources but needs to improve with an imbalanced system load. And PDPS prioritizes tasks based on criticality to meet timing constraints through preemption but relies heavily on task prioritization and may not be energy efficient. Therefore, improvements to both algorithms with energy-aware features will be proposed. Future work should focus on developing hybrid scheduling techniques that minimize energy consumption through intelligent task prioritization, resource allocation, and meeting time constraints.

***Keywords*—**Energy-aware scheduling, fair-share scheduling, priority-driven preemptive scheduling, real-time systems, optimization, resource reservation, timing constraints.

I. Introduction

ONE critical aspect of operating system optimization is scheduling, which involves determining the order and allocation of system resources and time constraints to different tasks or processes. Scheduling algorithms significantly impact system performance, responsiveness, and resource utilization [15]. In the context of real-time systems, scheduling algorithms must consider tasks not only deadlines and resource allocation but also energy efficiency to address the growing concerns of resource constraints and environmental impact [8]. By analyzing the energy awareness of these scheduling algorithms, this research aims to solve key issues in scheduling in real-time systems, determining their effectiveness in minimizing energy consumption additionally while meeting real-time task deadlines and completing the resource reservation.

FFS is designed to provide equitable resource allocation among multiple tasks or processes, ensuring fairness and preventing resource starvation [11]. On the other hand, PDPS aims at prioritizing and scheduling tasks based on their significance, making sure that high-priority tasks are assigned precedence over lower-priority tasks [3].

Our research project focuses on the analysis of two scheduling algorithms, namely FFS and PDPS, in the environment of real-time systems [8]. The goal is to investigate issues solved in energy awareness by these algorithms. And we will also conduct research on the solving process. Through research, the article will provide insights into the strengths and limitations of each algorithm and discuss the possible improvements for each algorithm.

This study's findings will contribute to optimizing operating systems in real-time systems by providing a comprehensive understanding of the benefits and challenges associated with FFS and PDPS in terms of energy awareness [3]. Ultimately, the research aims to enhance the design and implementation of scheduling algorithms in operating systems to achieve improved energy efficiency and performance in real-time computing environments.

II. Background

A. Introduction of Energy-Aware Scheduling

Energy-aware scheduling is a scheduling method that optimizes job sequencing and resource allocation in order to reduce a system's energy consumption. Scheduling that takes into account energy sensing takes into account how energy-intensive computer systems are.

Through judicious job scheduling and resource management, energy sensing scheduling aims to optimize energy in the following areas, such as mobile computing, renewable energy integration, IoT. It makes an effort to prevent resource wastage and cut back on excessive energy use.

B. Real-Time Systems

There are actually two main types of real-time systems:

- Hard real-time systems: In these kinds of systems, missing a deadline can lead to catastrophic consequences, such as failure of safety-critical systems or loss of life. There are many real-world applications of hard real-time systems, including aircraft control systems, medical devices, and nuclear reactor control systems. In hard real-time systems, meeting timing constraints is significantly crucial.
- Soft real-time systems: In these kinds of systems, missing a deadline might cause degraded performance or temporary disruptions, but it does not lead to catastrophic consequences for its real-world application, including video streaming, voice communication, and online gaming.

Su Xiaohan, Jin Chicheng, Liu Yijing, and Dr. Burra Venkata Durga Kumar are with the School of Computing & Data Science, Xiamen University Malaysia, DULN009(B) Jalan Sunsuria, Bandar Sunsuria, 43900 Sepang, Selangor Darul Ehsan, Malaysia (e-mail: EEE1909245@xmu.edu.my, CST2109157@xmu.edu.my, CST2009133@xmu.edu.my, venkata.burra@xmu.edu.my).

Although meeting the deadline of each task is still important in soft real-time systems, occasional deadline misses can be tolerated [9].

C. The Application of Energy-Aware Scheduling in Real-time System

Real-time systems can use less energy, utilize energy more effectively, and schedule tasks such that they are completed in real time. It supports the dependability and sustainability of real-time systems by meeting time requirements while consuming the least amount of energy. As a result, it has been used in many real-time system fields.

An IoT heterogeneous multiprocessor system on a chip (MPSoC) job scheduling challenge was investigated by a team from China in order to maximize security quality while meeting energy, real-time, and task precedence requirements [22]. Besides, a new energy-efficient routing technique called geographic routing time transfer (GRTT) is suggested in the field of workforce monitoring in order to employ sensor node topology information for target tracking and coverage applications [17]. Last but not least, a team conducted a study with a focus on cloud computing, which processes data from fog computing that has been gathered by devices. By introducing Dynamic Voltage and Frequency Scaling (DVFS) technology, they develop a kind of energy-conscious strategy that uses less energy [6].

III. Issue

A. Issues of Energy-Aware Scheduling in Real-Time Systems

Energy-aware scheduling in real-time systems brings additional challenges. There are some key issues specific to energy-aware scheduling in real-time systems, for example, time constraints, resource reservation, DVFS, predictability, overhead considerations, performance degradation and so on. In real-time systems, energy-aware scheduling aims to optimize the execution of tasks, which is about task prioritization. And meeting time constraints and resource reservation are the two main ways to prioritize the tasks based on our research. Therefore, in this paper, we mainly focus on the issues of time constraints and resource reservations of energy-aware scheduling in real-time systems.

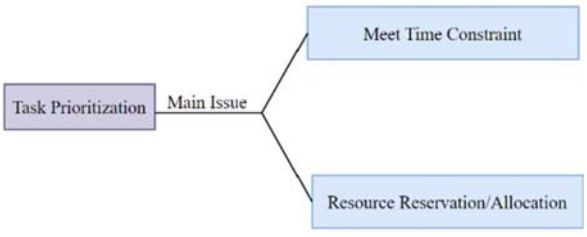

Fig. 1 Main Issues in Task Prioritization

B. Issue of Resource Reservations of Energy-Aware Scheduling in Real-Time Systems

A real-time operating system is capable of handling jobs with stringent deadlines. High response time requirements for jobs thus become a problem, which can result in deteriorated system performance or significant breakdowns. When it comes to energy-aware scheduling, its application is intended to reduce the system's energy usage. As a result, the order in which tasks are scheduled and the reservation of resources become problematic.

C. Issue of Time Constraints of Energy-Aware Scheduling in Real-Time Systems

In real-time systems, timing constraints (the time of tasks should meet) are one of the most critical requirements that should be satisfied for it to make it function correctly; the components of the systems can interact well, and they can respond promptly to other external events. The issue of meeting timing constraints in real-time systems revolves around scheduling tasks and ensuring that tasks are executed within corresponding deadlines while maintaining high performance as well as efficiency. In this case, the process of task prioritization can be conducted smoothly. In order to solve this issue, there are various scheduling algorithms and techniques being applied to the efficient management of task execution. And the brief process can be shown in Fig. 2 [2].

IV. Algorithms

A. Introduction of Fair-Share Scheduling Algorithm

FFS is a scheduling algorithm in the fields of operating systems where the usage of CPU is distributed among different system users or groups equally, which is opposed to the equal distribution of resources in the processes [1]. It can dynamically and equally distribute the time quanta to its objects [4].

One common way of applying FFS is implementing the round-robin scheduling strategy at each level of abstraction recursively (users, processes, groups, etc.). The time quantum that round-robin requires is arbitrary, and the same results will be produced as an equal division of time. Here is an example. There are four users (A, B, C, D), and each of them is executing a process at the same time; all the users can get the same share of the CPU cycles with the help of the scheduler that splits the CPU periods equally, which is $(100\%/4)$ 25%.

TABLE I
EXAMPLE OF DISTRIBUTION CPU TIME OF PROCESS BY USING FFS

Process 1	Process 2	Process 3	Process 4
25%	25%	25%	25%

TABLE II
EXAMPLE OF DISTRIBUTION CPU TIME OF SUB-PROCESSES BY USING FFS

	Sub1 of Process 2
Sub1 of Process 1	$50\% * 25\% = 12.5\%$
$50\% * 50\% = 25\%$	Sub2 of Process 2
	$50\% * 25\% = 12.5\%$
	Sub3 of Process 2
Sub2 of Process 1	$50\% * 25\% = 12.5\%$
$50\% * 50\% = 25\%$	Sub4 of Process 2
	$50\% * 25\% = 12.5\%$
Process 1	Process 2

The scheduler nevertheless logically separates any additional sub-processes into another layer of partitioning if there are any. For instance, if two processes are running simultaneously and each of them has a different number of subprocesses to

complete, the algorithm will divide the total time available for those processes.

Besides, this algorithm gives users and queues fair access to resources by dividing the processing power of the LSF cluster so that monopolizing the resources of the cluster by user or queue is impossible, and no queue will be starved.

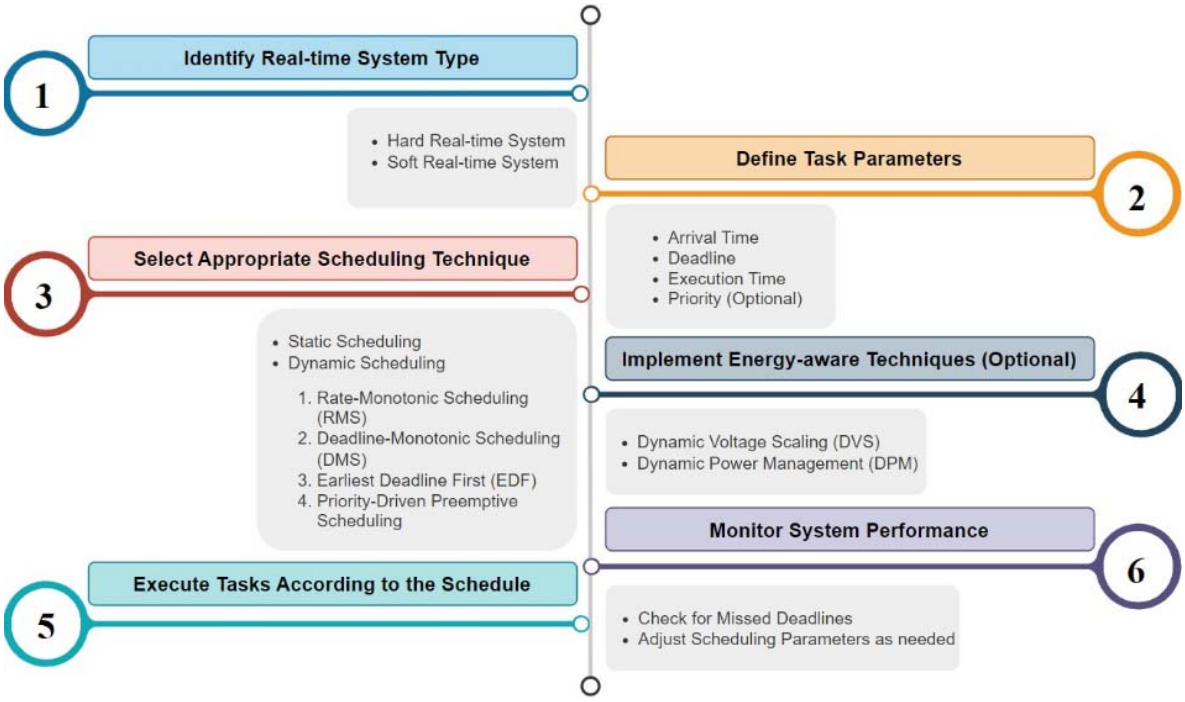

Fig. 2 Process of Solving Issue of Time Constraints

The following details the algorithm's operation: Each user or group is first allocated a certain number of shares. These shares represent a small portion of the cluster's available resources. The most important users or groups will receive the greatest number of shares. The dynamic priority of a user may be determined by the share assignment.

The dynamic priority of the person who submitted the job is more significant than the sequence in which jobs are queued up. When utilizing FFS, LSF will place the first job, which belongs to the user with the highest dynamic priority, in the queen [7].

Fig. 3 illustrates the entire scheduling procedure. After launching, this algorithm will allocate resources depending on the stated task priority while also keeping an eye on resource utilization. We continue the procedure and allot resources for the task until it is finished if the scheduling is not finished. If the schedule is finished, we go on to the conclusion.

B. Resolving Resource Reservations in Energy-Aware Scheduling

Resource reservations are balanced through FFS. While FFS can guarantee it, energy-aware scheduling tries to reduce system energy consumption. The system may allocate the resources needed for energy-sensitive tasks based on the fair share of each user or job with the help of FFS. Inadequate resources for other jobs are avoided as a result of preventing particular users or tasks from abusing resources excessively. In addition, the system can set limits on each user's or task's energy usage to keep it within the allotted share. This allows the system to optimize energy use and prevents energy usage from going above or below expectations.

C. Introduction of Priority-Driven Preemptive Scheduling Algorithm

PDPS Algorithm is an algorithm that can be used to help with the management of task prioritization in real-time systems, where tasks are assigned priorities based on their time constraints or other standards. In PDPS, tasks with higher priorities are executed first and right before tasks with lower priorities. Besides, if there are tasks with higher priorities, the execution process can even be preempted. Moreover, PDPS is adaptable to different system requirements and constraints; for example, it can be able to provide a framework for employing other energy-aware scheduling strategies.

As we know, the key issue in addressing timing constraints is to find the most suitable algorithm and technique that can make sure all tasks are executed within the corresponding deadline with high efficiency and good performance, which is also a significant issue in energy-aware real-time systems.

According to Fig. 3, PDPS is included in the "Dynamic Scheduling" category. Also, PDPS is an algorithm integrated with some energy-saving techniques like Dynamic Voltage Scaling (DVS) and Dynamic Power Management (DPM). With the help of these two techniques, PDPS Algorithm would be a great solution for dealing with timing constraints in real-time systems.

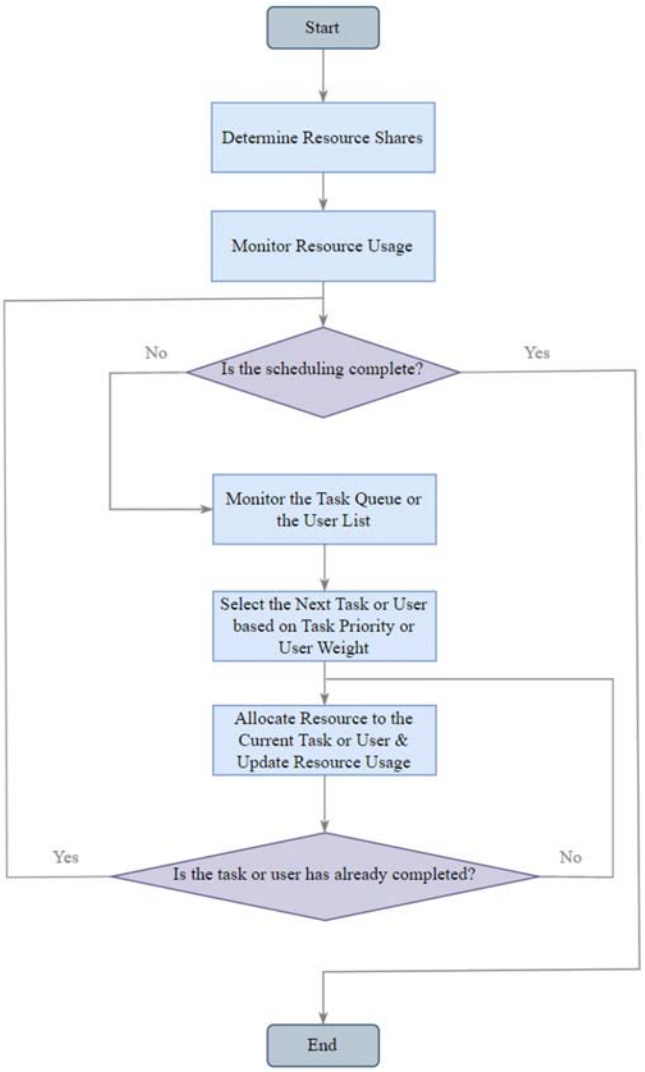

Fig. 3 Process of FFS

D.Resolving Time Constraints in Energy-Aware Scheduling for Real-Time Systems

After incorporating PDPS into energy-aware scheduling, the issue of timing constraints can be effectively addressed. This scheduling method can also be adapted to address energy-aware scheduling by considering the energy consumption associated with each task. The following is a detailed explanation of how PDPS solves timing constraints in energy-aware scheduling for real-time systems.

1. Task Model

Before PDPS is conducted, each task in the system would be defined by a tuple $(C\_i, D\_i, T\_i, E\_i, P\_i)$, where C_i is the computation time, D_i is the relative deadline, T_i is the period, E_i is the energy consumption, and P_i is the priority of the task respectively. Besides, tasks are ordered by their priority levels, with higher priority tasks having lower priority values (e.g., $P1 < P2$) [10].

2. Task Prioritization

In this process, PDPS assigns priorities to tasks based on their importance and criticality. The important task will be assigned with higher priority and require faster response time or stricter time constraints so that tasks are all scheduled and executed in a timely manner.

a. Energy-Aware Priority Assignment

The priority assignment is executed by considering both timing constraints and energy consumption. Generally, the fixed-priority assignment will be used by PDPS so that tasks are assigned fixed priorities that remain unchanging during execution. Also, it simplifies scheduling decisions and analysis, which allows for predictable behaviors and facilitates schedulability analysis, enabling the determination of whether all tasks can be successfully scheduled within their timing constraints or not.

Based on research, one approach proposed by Guo et al. [5] is to use the Rate Monotonic Algorithm (RMA) or the Deadline Monotonic Algorithm (DMA) to assign priorities according to their periods or deadlines, respectively. Then, the tasks with similar priorities can be ordered based on their energy consumption, with lower energy-consuming tasks having higher priority.

b. Response Time Analysis

Particularly, PDPS enables response time analysis, which estimates the worst-case execution time and response time of each task. It helps determine if timing constraints are met, making sure that all tasks complete their execution within their deadlines. Also, PDPS can provide guarantees on task response times, allowing for proper system design and validation after considering the worst-case execution times [21].

3. Preemptive Scheduling

Preemption is crucial for meeting timing constraints since it makes sure that high-priority tasks will not be delayed or blocked by lower-priority tasks. Thus, the scheduler follows the preemptive approach, meaning that a task with higher priority can preempt a task with lower priority currently being executed, or lower-priority tasks would be preempted or temporarily suspended to allow the tasks with higher priority to execute.

When a task arrives, the scheduler will check whether the task has a higher priority than the currently running task. If so, the running task is preempted, and the task with higher priority begins execution. Otherwise, the new task is placed in the queue according to its priority, as others do. Putting it in another way, when a task with higher priority becomes ready to be executed, it preempts the currently running task with lower priority, enabling the system to respond promptly to the more critical events.

4. Task Suspension and Resumption

PDPS allows for task suspension and resumption, which means a lower-priority task can be temporarily suspended to give way to a higher-priority task, as mentioned above. Once the higher-priority task completes or is preempted by an even task with higher priority, the lower one is resumed. Task suspension and resumption will facilitate efficient scheduling and ensure that high-priority tasks obtain the necessary

resources to meet their timing constraints. In this case, task suspension and resumption are actually a small part of preemptive scheduling.

5. Priority Inheritance

Priority inheritance protocols can be applied with PDPS to mitigate priority inversion problems. Priority inversion happens when a high-priority task is blocked or delayed by a lower one holding a shared resource. Priority inheritance ensures that the priority of the task accessing the shared resource is temporarily elevated to the highest priority among the tasks waiting for that shared resource. This approach can prevent priority inversion and help maintain the timely execution of critical tasks efficiently.

6. Energy-Awareness

Energy awareness can be achieved by employing DVFS techniques. The scheduler can then adjust the operating frequency and voltage of the processor to minimize energy consumption while still meeting the time constraints of each task. This can be done by considering the energy consumption and remaining slack time (difference between the deadline and the current time) of each task.

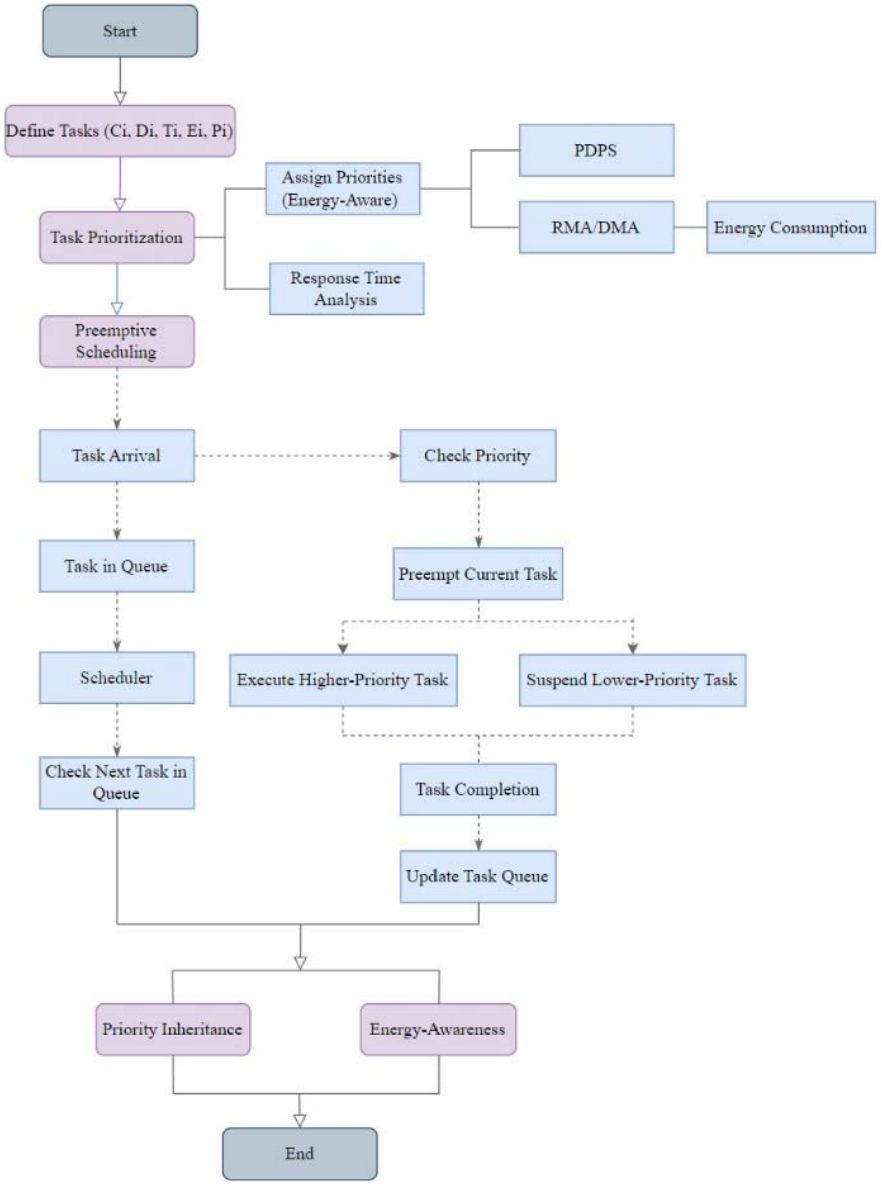

Fig. 4 Process of How PDPS Algorithm Solves the Issue of Time Constraints of Energy-Aware Scheduling in Real-Time Systems

By applying the technique like PDPS, energy-aware scheduling in real-time systems can effectively solve the problem of timing constraints.

Overall, the task model setting, prioritization of tasks, energy-aware priority assignment, response time analysis, support for preemption, task suspension and resumption, priority inheritance, and energy awareness collectively ensure that critical tasks receive the necessary resources and execute

within their deadlines, ensuring timing correctness in the system.

V. Discussion

A. Advantages of Fair-Share Scheduling Algorithm

FFS enables various users or tasks with various priorities. According to their fair share of resources, each user or job is assigned a priority level. It is beneficial for prioritizing work and ensuring that time-sensitive jobs are given the required resources and completed on schedule. FFS is furthermore flexible. In order to adapt to the current workload and user/task needs, it can dynamically alter resource allocation based on system conditions. In real-time systems, this algorithm aids in maximizing resource usage and responsiveness.

B. Limitation of Fair-Share Scheduling Algorithm

The imbalanced system load is one of the FFS's drawbacks. Even though FFS strives to allocate resources fairly, when the system load is imbalanced, some jobs may face delays or resource shortages.

Another restriction is the difficulty of task prioritizing. Priorities must be established for various tasks in FFS to ensure equitable resource distribution. Making the proper settings and choosing the proper priorities, however, are difficult undertakings. Inappropriate order, the performance of other processes can be impacted when certain tasks use up too many system resources due to settings.

When the system is overloaded, or there are not enough resources, FFS might not be able to guarantee real-time performance. Higher-level scheduling algorithms and strategies are needed for projects with stringent real-time requirements in order to guarantee their real-time performance.

C. Advantages of Priority-Driven Preemptive Scheduling Algorithm

In PDPS, tasks with higher priorities are executed before lower ones, and running tasks can be preempted by incoming higher-priority tasks. In this way, the problem of time constraints of energy-aware scheduling in real-time systems can be easily solved.

Moreover, there are also some optimizations of PDPS for the process of meeting time constraints in task prioritization; for example, preemption and response time analysis, DVFS (which optimizes energy consumption without violating timing constraints), schedulability analysis, response time reduction, predictability and determinism (which provides a predictable and deterministic scheduling framework, then allows for accurate analysis and prediction of task execution and resource usage to enhance system reliability) and so on [16].

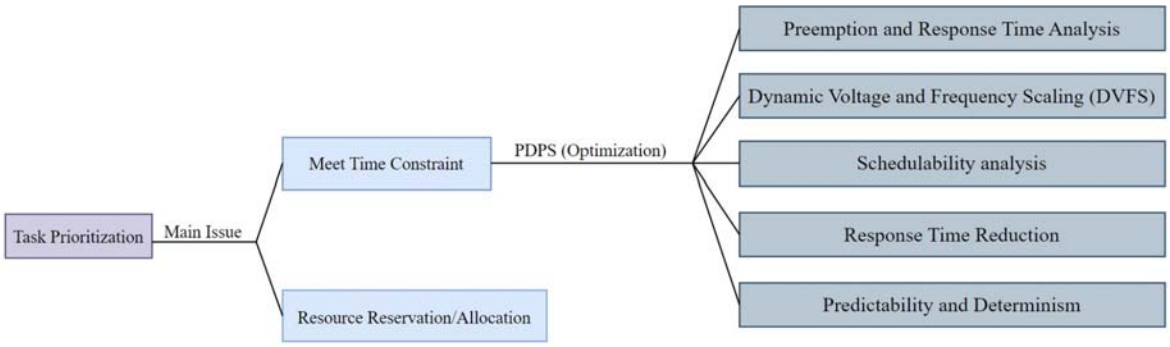

Fig. 5 Optimization Ways of PDPS

D. Limitation of Priority-Driven Preemptive Scheduling Algorithm

Although PDPS helps to deal with several challenges, it is important to note that energy-aware scheduling in real-time systems is a complex problem, and PDPS may not be sufficient enough to address all issues happened. Therefore, additional considerations are needed, such as energy models, system-level optimizations, and balancing energy efficiency with timing requirements, which may still be required for effective energy-aware scheduling in real-time systems.

VI. Recommendation

A. Improvement for Fair-share Scheduling Algorithm

1. Task Packing Algorithm to Maximize CPU Utilization

Task Packing algorithm can be used, which is based on the Knapsack problem. It concentrates on the idle cycles of unbalanced applications where one or more CPUs are set aside from execution. To make it, a minimum number of CPUs are chosen and work on useful jobs of the parallel application tasks with undegraded performance, using oversubscription. Besides, "in place" is one of the characteristics of the Task Packing algorithm, which can make full use of idle cycles generated at synchronization points of unbalanced applications [18].

2. Fair-share Management for Resource Reservation

To optimize the resource reservation, we improve it by the following method: The user will submit a request to the system. The system is informed by the provider of the resources that are accessible. The system then determines each provider's fair share. The provider in question will receive the entire resource if the number of requests exceeds the fair share. Following this, the system will update the details. When the system receives the provider with the largest share (apart from those on the entire list), it will provide them with the requested resource. Then, it will stop if the provider calls the final request; else, carries on

with the initial flow. The entire system is focused on equitable resource distribution and fair share resource management [19]. And Fig. 6 is the flow chart to explain the whole procedure.

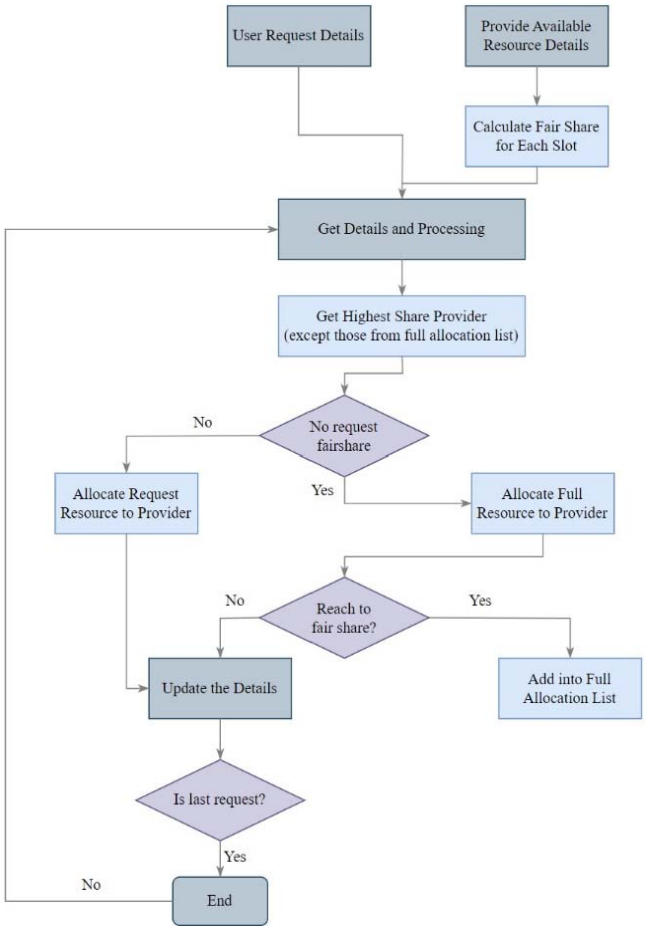

Fig. 6 Process of Fair-Share Resource Management Method [19]

B. Improvement for Priority-Driven Preemptive Scheduling Algorithm

As mentioned above, two techniques, DVS and DPM, can be applied with PDPS to improve its performance in solving time constraints problems. In detail, these techniques help in reducing energy consumption by adjusting the processor's voltage and frequency, as well as by managing power states [20]. Therefore, the PDPS algorithm can be extended to include energy-aware features by considering the following factors:

- Task priority: Tasks are prioritized based on their deadlines. Higher-priority tasks have tighter deadlines and must be executed before lower-priority tasks.
- Task execution time: The time required to complete each task should be considered. Longer tasks consume more energy, so the scheduler aims to minimize the overall task execution time.
- Processor states: The scheduler takes into account the different power states that the processor can be in, such as active, idle, and sleep modes. Each state correspondingly has a different power consumption level.

And originally, the scheduling process of PDPS in dealing with the time constraints of energy-aware scheduling should be as the example shown in Table III.

TABLE III
EXAMPLE OF SCHEDULING PROCESS OF PDPS

Task	Arrival Time	Deadline	Execution Time	Priority
A	0	10	4	1
B	2	14	5	2
C	4	20	7	3

According to Table III, there are three tasks, namely A, B, and C, with different arrival times, deadlines and execution times. Then their priorities will be assigned by PDPS based on deadlines so that A has the highest priority and then B and C.

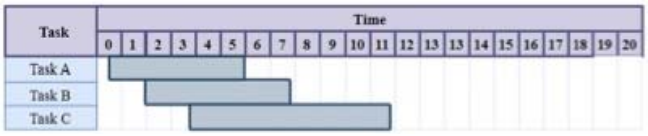

Fig. 7 Example of Execution Process

In detail, as we can see from Fig. 7, Task A starts executing immediately at time 0, followed by Task B at time 4, and finally, Task C conducts at time 9. The scheduling process ensures that all tasks meet their respective time constraints while also minimizing energy consumption by reducing the overall task execution time. More than that, after improvement by applied techniques of DVS and DPM, PDPS will be more efficient in minimizing energy consumption [14]. For example, after expanding Table I with power states and energy consumption, the effectiveness of these two techniques can be explained.

TABLE IV
EXAMPLE OF SCHEDULING PROCESS OF PDPS WITH POWER STATES AND ENERGY CONSUMPTION

Task	Arrival Time	Deadline	Execution	Priority	Power State	Energy Consumption
A	0	10	4	1	Active	12 mJ
B	2	14	5	2	Active	15 mJ
C	4	20	7	3	Active	21 mJ

1. Dynamic Voltage Scaling

This technique can adjust the processor's voltage and frequency depending on the workload of the task to reduce energy consumption. There are three frequency levels: high, medium, and low [12].

TABLE V
EXAMPLE OF FREQUENCY LEVELS OF DIFFERENT TASKS

Task	High Frequency	Medium Frequency	Low Frequency
A	12 mJ	9 mJ	7 mJ
B	15 mJ	11 mJ	9 mJ
C	21 mJ	16 mJ	13 mJ

Based on Table V, DVS will be applied by the scheduler to select the most energy-efficient frequency level for each task without violating their time constraints. For example, Task A can be executed at low frequency (energy consumption then from 12 mJ to 7 mJ) and still meet its deadline.

2. Dynamic Power Management

DPM can switch the processor between different power states to save energy. When there is no task running, the processor can enter an idle or sleep state, consuming less energy than in the active state. The energy consumption for different power states is shown in Table VI [13].

TABLE VI
EXAMPLE OF ENERGY CONSUMPTION FOR DIFFERENT POWER STATES

Power State	Energy Consumption
Active	5 mJ/s
Idle	2 mJ/s
Sleep	0.5 mJ/s

The scheduler can apply the DPM technique to minimize energy consumption during idle periods. For example, after Task A finishes execution at time four and before Task B starts at time 4, the processor can enter the idle state for a short period, reducing energy consumption.

By applying both DVS and DPM techniques, PDPS can effectively minimize energy consumption while ensuring that all tasks meet their timing constraints, which means that PDPS can be improved. This approach allows for a balance between performance and energy efficiency in real-time systems.

In conclusion, the fusion of Blockchain and AI technologies holds significant potential for transforming supply chain finance and enhancing the scalability of distributed systems. This amalgamation presents an innovative solution to the prevailing scalability issues by improving transaction speed, security, and data handling capacities. Blockchain's decentralized nature eliminates intermediaries, boosts transparency, and enhances security. However, it also introduces the Blockchain Trilemma, encompassing problems around security, scalability, and decentralization, es.

VII. CONCLUSION

Through the article, we have analyzed two scheduling algorithms, FFS and PDPS, in the environment of energy-aware scheduling in real-time systems. Both algorithms have advantages in solving certain issues in energy-aware scheduling, such as resource reservation and timing constraints. However, they also have limitations that can be addressed through improvements.

FFS ensures fair allocation of resources among tasks but may struggle when the system load is imbalanced. Implementing a task-packing algorithm and improving fair-share management can optimize resource utilization and reservation.

PDPS prioritizes tasks based on their criticality and meets timing constraints through preemption. However, it relies heavily on task prioritization and may not be inherently energy efficient. Applying techniques like DVS and DPM can improve its energy efficiency while still maintaining timeliness.

In summary, combining the strengths of both algorithms, along with implementing improvements with energy-aware features, can optimize scheduling in real-time systems to achieve both timeliness and energy efficiency. Future work should focus on developing hybrid scheduling techniques that incorporate the advantages of FFS for fairness and priority-driven scheduling for time constraints while minimizing energy consumption through intelligent task prioritization, resource allocation, and processor management.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Dr. Burra Venkata Durga Kumar for his invaluable guidance and mentorship throughout the process of writing this paper. His expertise in the field of energy-aware scheduling and real-time systems has been instrumental in expanding our knowledge and enhancing our understanding of this complex subject matter. His continuous support and encouragement have been pivotal in completing this research. Furthermore, we extend our heartfelt appreciation to all the authors whose works have been cited in this paper. Their significant contributions to the field of distributed systems have formed the foundation upon which our research is built. Their relentless efforts and dedication to advancing knowledge in this domain have inspired us and enriched our understanding of the subject.

REFERENCES

[1] Adibhatla, B. et al. (2021) "Top Interview Questions for a Data Engineer Job Profile," Analytics India Magazine (Preprint). Available at: https://analyticsindiamag.com/top-interview-questions-for-a-data-engineer-job-profile/.

[2] Burns, A. and Davis, R.H. (2017) "A Survey of Research into Mixed Criticality Systems," ACM Computing Surveys, 50(6), pp. 1–37. Available at: https://doi.org/10.1145/3131347.

[3] Chen, Y., Li, Y., & Chen, J. (2018). Research on Real-Time Scheduling Algorithm Based on Energy Efficiency. Journal of Physics: Conference Series, 1053(1), 012131. Available at: https://doi.org/10.1088/1742-6596/1053/1/012131

[4] GeeksforGeeks (2021) "Fair share CPU scheduling," GeeksforGeeks (Preprint). Available at: https://www.geeksforgeeks.org/fair-share-cpu-scheduling.

[5] Guo, Y., Wang, K., & Yang, J. (2018). Energy-aware real-time task scheduling for single-processor systems: A survey. Journal of Systems Architecture, 84, 1-14. Available at: https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=scheduling-understand-fair-share.

[6] Hosseinioun, P. et al. (2020) "A new energy-aware tasks scheduling approach in fog computing using a hybrid meta-heuristic algorithm," Journal of Parallel and Distributed Computing, 143, pp. 88–96. Available at: https://doi.org/10.1016/j.jpdc.2020.04.008.

[7] IBM Documentation (no date). Available at: https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=scheduling-understand-fair-share.

[8] Kumar, S., & Singh, K. (2019). Energy-aware scheduling algorithms for real-time systems: A review. Journal of Ambient Intelligence and Humanized Computing, 10(5), 1787–1800. Available at: https://doi.org/10.1007/s12652-018-0798-3

[9] Liu, C. L., & Layland, J. W. (2019). Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 20(1), 46–61. Available at: https://doi.org/10.1145/321738.321743

[10] Maghsoudlou, H., Afshar-Nadjafi, B. and Niaki, S.T.A. (2021) "A framework for a preemptive multi-skilled project scheduling problem with time-of-use energy tariffs," Energy Systems, 12(2), pp. 431–458. Available at: https://doi.org/10.1007/s12667-019-00374-8.

[11] Pagani, S., Cardellini, V., Grassi, V., & Lo Presti, F. (2021). Fair share scheduling for cloud computing: A survey. Journal of Network and Computer Applications, 183, 102976. Available at: https://doi.org/10.1016/j.jnca.2021.102976

[12] Piao, X. and Park, M. (2015) "On-Line Dynamic Voltage Scaling for EDZL Scheduling on Symmetric Multiprocessor Real-Time Systems," International Journal of Multimedia and Ubiquitous Engineering (Preprint). Available at: https://doi.org/10.14257/ijmue.2015.10.7.18.

[13] Praveenchandar, J. and Tamilarasi, A. (2022) "Retraction Note to Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing," Journal of Ambient Intelligence and Humanized Computing, 14(S1), p. 115. Available at: https://doi.org/10.1007/s12652-022-03970-2.

[14] Qin, Y. et al. (2022) "Dynamic voltage scaling based energy-minimized partial task offloading in fog networks," Wireless Networks, 28(8), pp. 3337–3347. Available at: https://doi.org/10.1007/s11276-022-03052-3.

[15] Rahman, M. M., Islam, M. R., & Islam, M. M. (2020). A Survey of Real-Time Scheduling Algorithms for Multiprocessor Systems. In Proceedings of the International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (pp. 124–131). Available at: https://doi.org/10.1145/3388231.3388245

[16] Rubaiee, S. and Yildirim, M. (2019) "An energy-aware multiobjective ant colony algorithm to minimize total completion time and energy cost on a single-machine preemptive scheduling," Elsevier, 127, pp. 240–252. Available at: https://doi.org/10.1016/j.cie.2018.12.020.

[17] Sangaiah, A.K. et al. (2021) "Energy-Aware Geographic Routing for Real-Time Workforce Monitoring in Industrial Informatics," IEEE Internet of Things Journal, 8(12), pp. 9753–9762. Available at: https://doi.org/10.1109/jiot.2021.3056419.

[18] Utrera, G., Farreras, M. and Fornes, J. (2019) "Task Packing: Efficient task scheduling in unbalanced parallel programs to maximize CPU utilization," Journal of Parallel and Distributed Computing, 134, pp. 37–49. Available at: https://doi.org/10.1016/j.jpdc.2019.08.003.

[19] Vaghela, F.N. and Serasiya, S. (2022) "Fair Share Management for Resource Allocation in Multi Cloud Environment," International Journal of Progressive Research in Engineering Management and Science, 02(05), pp. 32–35.

[20] Zhang, Y. et al. (2020) "Interval optimization based coordination scheduling of gas–electricity coupled system considering wind power uncertainty, dynamic process of natural gas flow and demand response management," Energy Reports, 6, pp. 216–227. Available at: https://doi.org/10.1016/j.egyr.2019.12.013.

[21] Zhang, Y. (2023) "Energy efficient non-preemptive scheduling of imprecise mixed-criticality real-time tasks," Sustainable Computing: Informatics and Systems, 37, p. 100840. Available at: https://doi.org/10.1016/j.suscom.2022.100840.

[22] Zhou, J. et al. (2020) "Security-Critical Energy-Aware Task Scheduling for Heterogeneous Real-Time MPSoCs in IoT," IEEE Transactions on Services Computing, 13(4), pp. 745–758. Available at: https://doi.org/10.1109/tsc.2019.2963301.