



(RESEARCH ARTICLE)



Analysis of energy saving scheduling algorithms for multiprocessor systems

M Sreenath * and P A Vijaya

Department of ECE, BNMIT, Bengaluru, Karnataka, India.

International Journal of Science and Research Archive, 2022, 07(02), 614–623

Publication history: Received on 20 November 2022; revised on 28 December 2022; accepted on 30 December 2022

Article DOI: <https://doi.org/10.30574/ijrsra.2022.7.2.0368>

Abstract

Modern battery-powered devices' energy usage has become one of the most crucial design considerations in the near past. This presents an extra issue in a multi processor environment. The key development criteria and goal is to lower system operating costs through low energy consumption, an extended battery life, and less heat dissipation. When the system's resource consumption is subpar, it can occasionally lead to decreased performance levels since more energy is needed to complete tasks. The amount of CPUs and cores affects how much energy the system uses while performing activities. If the system executes a more no of tasks, may face different difficulties to maintain a good performance. A process scheduler uses particular scheduling methods to assign different programmes to the CPU. Priority Scheduling, Shortest Job Next, and FCFS are some of the most popular process scheduling techniques. The method that controls how much CPU time is given to processes and threads is known as a strategy for planning.

Keywords: Scheduling Algorithms; Multiprocessor; Energy Saving; Performance

1. Introduction

DPM & DVFS are both frequently employed in multiprocessor based real time systems to lower CPU power usage. For instance, to save electricity, our laptop might switch to a sleeping mode. The computer stays in a low-power sleep state, but only wakes up to process tasks when necessary. Another example application of DPM is to realize virtual machine management for cloud computing. In that scenario, virtual machines are activated on demand. The power manager schedules those rare used virtual machines into an idle state. An idle state will turn to a power-off state if it waits for a long time with no job arrived. DPM aims to switch the system into a power-saving mode as long as possible, while guaranteeing the system still provides prompt response for tasks. On the other hand, DVFS realizes power-saving purpose by decreasing frequency and voltage of a processing unit while maintaining certain quality of service (QoS). Since decreasing frequency and voltage increases task executing time, a common goal to achieve QoS is to reduce frequency as much as possible on the condition that task will be finished on time. DVFS and DPM hybrid methods can be found in [1–3]. This thesis focuses on a single-core CPU case. DVFS is a sub-topic of energy-aware scheduling [4]. First, single-core algorithms can be divided into DVFS and DPM. DVFS can be broken down into 3 sub-categories: static slack, dynamic slack, and static& dynamic slack.

2. Literature review

The planning and mapping tasks are the most difficult parallel computation problems, and to solve them, exceptional excellent performance computing is needed, which entails breaking the issue down into smaller tasks and concentrating on those simultaneously. To ensure efficient use of the system's, the application subtasks are allocated to supporting processors and scheduled for execution to optimize execution time and ensure load balancing amongst the processors in line with their progress. It's possible that the underlying infrastructure is uniform. Infrastructure that is homogeneous can make use of the same tools, resources, and capabilities. While machines with diverse infrastructure

* Corresponding author: M Sreenath

have varying efficiency, speed, and interconnection. The CBHD algorithm defined by clustered based heterogeneous earliest finish time (HEFT) with duplication, two of the most important task scheduling methods for heterogeneous machines. Data redundancy is required for the CBHD technique to function better. In accordance with a comparative evaluation of these algorithms, the CBHD performs load balancing, which is regarded as one of the essential performance elements in a dynamic environment. It also executes more quickly than the two algorithms. There are now two versions of the CBHD algorithm, most likely one without replication and one with replication. The makespan could be extended using the CBHD technique without duplication, decreasing performance. In a dynamic setting, the CBHD technique with replication has achieved the key performance characteristics, including load balancing, maximum utilization, and minimal makespan. Generally speaking, minimal makespan, load balancing, and CPU usage is almost satisfied in the created CBHD [5].

Initial particles in the PSO method are generated at random. Sadly, randomness can make it less likely for the algorithm to arrive at the best option. To improve the effectiveness of the traditional PSO technique, the Best-Fit (BF) method has been incorporated into the PSO algorithm, i.e. created according to the BF instead of creating initial population randomly. The normal PSO algorithm's other phases are left alone. In contrast contrary, the TS technique is a neighborhood search method that makes use of the flexible memory approach and intelligent search. Its purpose is to direct other algorithms away from the local optimality trap. As a result, it has been used to resolve additional optimization issues, including task scheduling. In order to prevent being stuck at the optimal solutions and to expedite the search process, TS algorithm and PSO algorithm have been combined. The PSO method serves as the foundation for the proposed BFPSOTS method, which combines the Tabu Learning Algorithm, PSO, and Best-Fit algorithms. Consequently, the fundamentals of the common PSO method will be covered first. To attain higher performance of the whole system, it is planned that the PSO algorithm will be improved in future work employing other greedy algorithms (such as Worst-Fit) and taking dependent jobs into account rather than independent tasks [6]. The IoT and cloud computing platforms combine to expand the potential for creating new applications and delivering them in the real world. But creating effective IoT applications is difficult in the present IoT landscape. Making sure the IoT ecosystem is secure is crucial in addressing these issues.

The security problems in IoT applications were attempted to be solved using traditional cloud models, however they were not successful in providing the best answer. Computing concepts like cloud technology and corner society are amalgamated to address the security challenges. The integration of the suggested solution by the reliability assessment methodology and usage template to solve the traffic adjusting issue in cloud applications. In order to optimize the model's potential, the structure of corner environment is developed with effective corner protocols that use fewer resources. The suggested solution offers the versatility of load, the cloud usage pattern and loaded the corner protocol's usage grammar pattern, which results in the creation of IoT apps. The findings of the study proved that any IoT application must use the Trust examination approach. The trust model distinguishes the percentage of the system's afflicted internal components better than a system without trust model. In order to offer an effective, safe place for the IoT and computing coupled application, the corner community and usage pattern is essential. The combination of corner society, corner regulation, usage template, and cloud computing delivers the best result for the Network of Environment. A revolutionary method is envisaged for the future to create a productive IoT-cloud ecosystem [7].

Objectives of the work

The basic objective is to distribute or allocate the computation time to all the tasks in a system by using scheduling algorithms to decrease the optimum execution times of all activities. The maximum duration it takes for every task to be performed, tardy tasks, etc., are decreased by using scheduling algorithms to distribute or allocate processor time among all the jobs in a system. Enhancing the efficiency of a system as a result. The major goals are to show that setting up primary and backup processes in separate processors at a moderate pace works, and to assess how well these methods function with various strategies for various applications.

- Functional comparative analysis is done using scheduling algorithm parameters on two processors. Turnaround, Process, burst, and waiting time parameters are used to assess the effectiveness of the SJF, RR, FCFS, Priority, and SJF-LJF algorithms.
- These measures are compared and analysed the scheduling algorithms behaviour by average wait and turnaround time, which weren't implemented in any of the previous studies.
- It was found that scheduling algorithms given results not as per expected. In order to achieve the expected outcomes based on typical average waiting, turnaround time, and energy use, a new scheduling algorithm was created as a result of this investigation.

3. Material and methods

The study has employed five approaches in order to compare the suggested methodology with the current methodology, i.e. Priority, SJF-LJF scheduling techniques, FCFS, RR, and SJF. In the present, the turnaround, burst, and waiting times of the proposed methodology are compared.

3.1. First Come & First Serve Scheduling Algorithm

This non-preemptive planning technique executes a designated activity a single at a period in a sequence. If a new procedure satisfies the requirements, it will go to the rear of the line. As a result, it is the final operation in the queue. This method advances the method preceding it in the queue if a novel process comes after it right now. If an ongoing activity remains stalled, every next activity in the pipeline is performed, and when the stalled project is completed, it is moved towards the conclusion of the pipeline. Batch systems typically employ it. It is quite simple to implement this FCFS method. However, because it is non-preemptive, there are times when CPU utilization is unnecessary. The Convoy effect, or the inability to use resources concurrently, originated from non-preemptive since the average wait time is not optimum. Consider a situation in which there are multiple IO-bound processes and a single CPU limited process. The processes that are IO-bound should queue until a process that is CPU limited receives CPU. Better would have been for the IO-bound activity to use the CPU for a while before switching to IO devices. Given n activities and their burst timings, the goal is to apply the FCFS scheduling method to calculate the mean turnaround and average wait. The most basic scheduling method is FIFO, often known as FCFS. Simply said, processes are FIFO queued and enter the ready queue in the order they arrive. In this case, Initial, its first procedure might be performed, and only then would the 2nd operation was fully completed. In this case, zero is the result if all processes start at the same time.

3.2. Longest Job and Shortest Job First strategies

3.2.1. Least or Shortest Job First

The strategy reorders the tasks so that the operation with the shortest burst duration is selected for the subsequent execution. It is employed to shorten the typical time that other processes spend waiting to be executed. This might be preventative or not. Shortest Remaining Time First is its preventive variant (SRTF). A job is added to the waiting list when it arrives based on its burst time. SJF decreases the average waiting time since it serves processes with shorter burst times before serving those with longer burst times. The key advantage of applying this strategy is that it prolongs average wait times and turn-around times, improving the efficiency of the system. It may penalize processes with longer burst times while minimizing average wait time. Processes having longer execution times have a tendency to be left on the ready queue while smaller processes get the job done if there are quicker processes available. In severe circumstances, it is possible that only processes with short execution times will be handled, while processes with long execution times will have to wait indefinitely. This algorithm's shortcoming is its dearth of longer-running processes.

3.2.2. SJF (Shortest Job First) Scheduling Algorithm

- The most effective strategy for cutting waiting times is this.
- These are employed in batch systems.
- There are two varieties:
 - Non-Preventative
 - Pre-emptive
- The processor must be informed of the activities as burst and duration timings in order for it to be correctly executed, which is almost never attainable.
- If every task or activity can be accessed simultaneously, this scheduling method is excellent. All arrival timings are either equal or all arrive at the same time.

3.2.3. Problem with Non-Preemptive SJF

Problem with non-preemptive SJF: If operations arrive at various times, some jobs may arrive later while not all activities are on the waiting list at zero time. In such a scenario, processes with short burst periods periodically need to wait for the implementation and operation of the current process to complete. Non-preemptive SJF does not pause an ongoing task or process to begin executing a new one when a new simple process is received. Because of this, there is a specific problem in a particular as starvation, when an extraordinarily lengthy period of time must pass before a shorter procedure may begin. If there is an increase in shorter professions, however the problem can be remedied by employing the idea of ageing.

3.2.4. Preemptive Shortest Task First Scheduling

When a process with a shorter execution times arrives, the present activity is preempted or suspended, as well as the smaller job is performed first. Jobs are placed into the waiting list as they arrive. However, longest job first (LJF) is the absolute reverse of least job first. The jobs are rearranged with the intention of choosing one that has the greatest burst time for execution next. When a process starts processing using this type of non-preemptive scheduling strategy, it cannot be stopped in the middle of it; instead, any further processes can only be performed just after assigned procedure has completed processing and been terminated. Based on its execution times, a job is added into the ready queue when it arrives. It's also possible for this to be preemptive or not. Longest Remaining Time First is its preventive kind (LRTF). A preemptive, non-preemptive scheduling approach is used. The right approach of cutting waiting time. In production settings where the necessary computation time is known in advance, implementation is simple. Incredibly difficult to accomplish in active schemes where the necessary CPU time is unknown. The computer ought to be aware of how long a task will take beforehand.

Advantages

- Quick decisions are made in short processes.
- The system also employs very minimal overhead because decisions are only made when a process is finished or one more is included.
- The algorithm need just compare the operation that is already running with the new process whenever a process is added; it can ignore any other processes that are currently waiting to run.

Disadvantages

- It may lead to process hunger, similar to shortest job first.
- If short processes are consistently introduced, long processes may be postponed forever.

3.2.5. Cons

- For just a particular set of processes, this algorithm yields extremely high average waiting times and average turnaround times. This could have a convoy effect.
- It is possible for a brief process to never be executed while the system continues to run longer processes.
- It slows down processing, which lowers the system's usage and efficiency.

3.3. Algorithm for scheduling Round Robin

A cyclic process is what defines the Round Robin (RR) scheduling method, and this is how the jobs are carried out. The CPU uses the scheduling mechanism known as round robin to carry out the job. When according to the Round Robin tasks for the periodic arrangement as follows, but whenever there is no chance for the tasks for executing due to because of cyclic scheduling. The tasks along with the period line and deadline will be assigned to the period. There no processor utilization ratio when compare to all other methods. The CPU uses the scheduling mechanism known as round robin to carry out the job. The round robin method was created especially for work and time sharing systems. Although the preemption feature adds the ability to move between processes, it's indeed similar to the first come, first served scheduling approach. A defined quantum, sometimes referred to as a time slice, is a brief period of time. The ready queue functions like a circular line. The circular queue, also known as the ready queue, is where this algorithm keeps all of its processes. The tails of the waiting list receives each new process. With the help of this technique, the CPU ensures that temporal slots (any rational number) are allocated to every process in equal quantities and in a circular manner, taking care of all processes equally and without regard to priority. Cyclical Executive is another name for it.

Each process inside the job (process) waiting line is guaranteed a time slot on the CPU by RR. Time slicing is the practice of allocating discrete blocks of time (referred to as quanta) to applications vying for CPU time. A period of time is just the amount of CPU time that each task will require for during an RR algorithm iteration. Although preemption occurs after a time slice, all jobs are completed using the FCFS method. The task either will be completed within the specified time frame or it will be redirected to the end of the queue and processed later. Since every job is essentially given the same priority, this is a drawback. Additionally, RR encourages quick virtual procedures. The requirement that the time slot value be accurate, is another issue with RR. Context switching takes a long time compared to actual CPU work if the time slot value is set too low. Additionally, the time slice value must be low enough to keep RR from degrading into a non-preemptive FCFS algorithm. Preemptive or non-preemptive prioritization scheduling options are available. Priority scheduling, irrespective of type, faces issues with famine. The three ways that a process can lose complete control of the CPU are task completion, task readiness if it is more crucial, and the emergence of a wait state. Lower priority procedures may be neglected as higher priority processes become ready.

3.3.1. Round Robin Strategy (RR)

In computing, application network schedulers use this technique. Time slices, or time quanta as it is more commonly known, all processes are processed equally, without respect to urgency, in equal portions that are distributed to everyone in a circular order. Scheduling using RR is straightforward, easy to carry out, and liberated from appetite. Other types of strategies could use round-robin procedures [8]. Scheduling issues, like the scheduling of data packets in computer networks. It is an idea for an operating system. The method draws its name from the well-known round-robin principle, in which each person alternately receives an equal portion of a resource. A round-robin scheduler typically uses period, giving every job a specific time or quantum and to reasonably schedule activities, halting the job if it is not accomplished by that point. The task is continued once another time slot is designated for that procedure. If the application crashes or changes to waiting within the time quantum allotted to it, the scheduler selects the initial task in the waitlist to be executed. A technique that created gigantic jobs would be preferred over other processes when time was limited or when the quanta were large in comparison to the amount of the work [9]. After reaching the duration quota, The RR approach is a preemptive algorithm since the controller removes the CPU activity.

3.4. Scheduling CPUs in priority

Learn about the behavior of the priority scheduling algorithm, its operation, and its benefits and drawbacks. In the least Job First method, a process's precedence is typically the reverse of the CPU access time, i.e., lower the priority of that process and higher the CPU burst time. The scheduling is made based on important each process that the most urgent process is addressed first, trailed by the processes with lower priority in that order. Scheduling priorities enables the special attention to be established either internally or externally, so it is not necessarily the CPU burst time that is set as the opposite. FCFS is used to carry out tasks with the same priority. When a process' internal priority is stated, it could be chosen based on factors such as needed memory, time constraints, the quantity of files, the ratio of CPU to I/O burst, etc. On the other hand, external priorities are established based on variables beyond the OS, such as the importance of the operation, the expense of employing the hardware and software resources, the production factor, etc.

Types of Priority Scheduling Strategies

Priority scheduling can take one of two forms:

- Preemptive: If a process to which the waiting list was added does have a greater priority than a process that is already in motion. When a system is preempted, the adaptive system's evaluation is halted. Instead, the newly arriving process is given the CPU to run on.
- Non Preemptive: In this case, the inbound approach is placed at the head of the waiting list, suggesting it will be treated after the present procedure has finished executing, if a new process comes with a higher priority than the one that is now running.

Given n activities and their burst timings, the goal is to apply the FCFS scheduling method to calculate the mean waiting & turnaround time. Simply said, processes are FIFO queued enter the line in the proper sequence. The first action will be carried out first in this case, and the second procedure won't start until the initial procedure is finished entirely. In this case, zero is the result if all processes start at the same time.

- Completion Time: The moment a process is finished running.
- Turnaround Time: the amount of time between finishing and arriving.
- The interval between the turnaround and peak times is known as wait time.
- Turnaround time minus burst time equals waiting time.

The SURE approach is a real-time, non-work conserving method designed to save system power usage while ensuring the application's chronological correctness. If there are open jobs to complete, the SURE algorithm purposely adds waiting times to the schedule, which is not work conserving. To create a recurring program for internet operation or live streaming, the SURE methodology can be run offline for greater flexibility and possibly better energy savings. Due to the fact that SURE is executed online, there must be a cost-benefit trade-off, which will vary according on the application. SURE is not a magic bullet for energy savings, as is true of the majority of scheduling algorithms.

4. Methodology

The method is comparable to the methods used to assess power dissipation and to assess I/O devices, all of which were offline scheduling approaches [10, 11]. Additionally, as the amount of energy a saving algorithm saves is what matters most when evaluating it, examined different offline algorithms. An online SURE implementation will result in the same energy savings as an offline simulation, provided that the jobs finish within their worst case completion time. The

expense of making adjustments to the pre-calculated slack table because of insertion delay, which has cost $O(n)$, is the major cost difference between both the online and offline SURE techniques. Data sheets of device's manufacturer were used to determine a device's power needs and state switching periods, otherwise, experimental measurements were made. Semiconductor offers an information sheet on the CPU that includes a list of the normal CPU energy needs for each of its several power modes. Though, the merchant omits to mention how long it often takes to change power modes. In order to determine the typical timeframes needed to switch between power states and the typical amount of energy used in each power state, a number of experiments were carried out. The calculated average times are utilized to figure out the typical power state transition time used in the simulations, though, because this data is not given by the supplier. The efficiency gains of the 3 methods are compared using their normalized efficiency gains. The amount of electricity conserved by a DPM technique in comparison to not using a DPM technique is measured by the normalized energy efficiency. The normalized energy savings are defined by energy saving Approach [12]. Each experiment involved, 500 task sets with random utilization ($U < 1$) were generated at random. Each task set contained a set of jobs, ranging from 1 to 20, at random.

The timeline serves to mark the exact point at which a component must be changed to the excited phase in order to guarantee that it happens before the beginning of the task it is being utilized for. At this stage in the operation, an event has been dispatched, placing the device in the excited phase. When a job is finished and the engine determines that a device has to be transferred to the idle condition, the power state change functions similarly. Because of the assumption of a synchronous periodic task set, only the first hyper period needs to be simulated. On average, SURE conserves more energy, when TDU (total device use) increases, the normalized energy savings decrease. The cause for this is that as gadgets are utilized more often, the duration of time they can be left idle decreases. In some circumstances, the EEA-EDF strategy may be more energy-efficient than the SURE method. This is because, while SURE tries to limit the amount of switches that are operational at any particular time, it is unable to guarantee that the total amount of switches is decreased. In heuristic algorithms, it is typical for a locally optimal decision to have a globally inferior outcome. The percentage of switch reductions calculated by Equation is another indicator will be divided with Number of Device Switches with EEA-EDF = Percentage of Switch Reductions [13].

5. Results and discussion

Enter the choice to execute specific functionalities

- Execute all schedule algorithms
- Execute all scheduling tasks related functionalities
- Exit
 - Enter time quantum: 2
 - Enter the number of process: 2
 - Enter the burst time of process in msec 1: 8
 - Enter the burst time of process in msec 2: 5

Table 1 FCFS Algorithm

Process	Burst Time (msec)	Waiting time (msec)	Turnaround time (msec)
p1	8.000000	0.000000	8.000000
p2	5.000000	8.000000	13.000000
Avg_WT: 4.000000			
Avg_TAT: 10.500000			

Table 2 SJF Algorithm

Process	Burst Time (msec)	Waiting time (msec)	Turnaround time (msec)
p1	5.000000	0.000000	5.000000
p2	8.000000	5.000000	13.000000
Avg_WT: 2.500000			
Avg_TAT: 9.000000			

Table 3 RR Algorithm

Process	Burst Time (msec)	Waiting time (msec)	Turnaround time (msec)
p1	8.000000	0.000000	8.000000
p2	5.000000	-2.000000	3.000000
Avg_WT: -1.000000			
Avg_TAT: 5.500000			

Table 4 Priority Algorithm

Process	Priority	Burst Time (msec)	Waiting time (msec)	Turnaround time (msec)
p1	2	8.000000	0.000000	8.000000
p2	1	5.000000	8.000000	5.000000
Avg_WT: 4.000000				
Avg_TAT: 6.500000				

Table 5 Modified SLF-LJF Algorithm

Process	Burst Time (msec)	Waiting time (msec)	Turnaround time (msec)
p0	5.000000	0.000000	5.000000
p1	8.000000	5.000000	13.000000
Avg_WT: 2.500000			
Avg_TAT: 9.000000			

An algorithm for non-preemptive scheduling is called longest job first (LJF). This approach is based on the processes burst times. Depending on their burst times, or in descending order of burst times, the prepared queue is updated with the processes. The traits were noted SLF-LJF like Average waiting and turnaround time based on burst time and processors used for execution, and Table 5 shows the related result. This algorithm is based, as its name implies, on the principle that the operation with the longest burst time is handled first. FCFS is used to break ties when two methods share the same burst time; the activity that came first is handled first. The corresponding results verified in table 1. Preemptive scheduling is used in this approach to treat each stage equally. There is a quantum, or running time, interval for each process to maintain a queue and each technique is subject to quantum. A task is relegated to the rear of the line once it has finished its quantum, at which point a new mechanism is initiated. Servers and desktop computers both use the round robin scheduling method. The time quantum's size affects how well the RR Algorithm performs. Observed the characteristics of RR like Average waiting and turnaround time based on burst time and processors used for execution, and table 3 shows the related result. Table 2, which shows the SJF relevant result, demonstrates the queuing and turnaround time restrictions. In Table 1, the proper result of the priority algorithm is displayed.

It has also been found that the suggested algorithm tends to have longer waiting, turn around, and burst times. It could be improved in the future to use any other legal application. Priority driven scheduling is the most used preemptive scheduling technique and added a ground-breaking algorithm to the suggested method. Many real-time applications benefit from it, including those where a job's priority whether something is CPU-bound in quasi platforms determines how it works. Planning tools with static tables must be redistributed in order to meet deadlines and guarantee that safety-critical activities get the attention they require. The capability of static schedules is evaluated using static table-driven techniques. A plan that is the result of the planning process is used to identify when a job may begin to be carried out utilizing dynamic planning-based approaches.

It has historically been the goal of Dynamic Power Management (DPM) strategies in integrated practical cases to reduce the power consumption dynamically that happens whenever a CMOS gate in a Processor changes. On received little attention, the power used by I/O devices and other subsystems as well as processor leakage power. However, non-real time devices have seen a significant amount of research attention with I/O-based DPM approaches. These methods, which switch I/O devices to reduced power modes based on a number of regulations, cannot be applied due to the erratic design of the rules, actual events. Reduced power consumption while retaining temporal accuracy is thus the goal of conserving energy in embedded practical applications. Propose three scheduling methods with increasing complexity to solve this issue. Given their simplicity and the low the price of changing energy states, these strategies offer significant energy reductions. But generally speaking, SURE is more advantageous the more energy is required to change power levels. The power efficiency provided by the SURE approach decrease as the price of shifting power modes rises, notably when compared to EA-EDF and EEA-EDF. The problem of finding a processes contribute that consumes the lowest amount of power from the I/O devices. Instead of choosing the best alternative, the main objective was to automate processes that can be employed remotely and reduce the energy consumption of numerous shared machines.

Table 6 WT and TAT of different scheduling algorithms

Name of the algorithm	Processor Name	Burst Time (msec)	Waiting Time (msec)	Turnaround Time (msec)
FCFS	P1	8.000000	0.000000	8.000000
	P2	5.000000	8.000000	13.000000
SJF	P1	5.000000	0.000000	5.000000
	P2	8.000000	5.000000	13.000000
RR	P1	8.000000	0.000000	8.000000
	P2	5.000000	-2.000000	3.000000
PA	P1(2)	8.000000	0.000000	8.000000
	P2(1)	5.000000	8.000000	5.000000
SLF-LJF	P1	5.000000	0.000000	5.000000
	P2	8.000000	5.000000	13.000000

The specific steps involved in putting the performance-enhanced scheduling method shown in Fig. 6 into practice. Observed the parameters of average waiting and turnaround time dependent on burst time and processors used for execution. The relevant result can be verified in Table 7. By creating the whole tree of anticipatory strategies for a certain job set and choosing the technique that used the lowest amount of energy, the SURE & MES for a task list were evaluated. This was accomplished by using a Depth-First-Search approach, which involved creating a tree of H levels and scanning the tree at each time-tick. Every level takes preemptive plans into account, thus for the sake of argument, think of CPU idle as a work with indefinite deadline and implementation length. No subsequent jobs at that position in the hierarchy are considered if a job is late without additional branching, the following job in the same level if a job has not yet been released. Simply said, this indicates that at any given time, only consider scheduling ready jobs.

Table 7 Avg_WT and Avg_TAT of different scheduling algorithms

Algorithm	Avg_WT	Avg_TAT
FCFS	4	10.5
SJF	2.5	9
RR	-1	5.5
Priority	4	6.5
SJF-LJF	2.5	9

Every level computes the energy. The least energy value for the specified task set once all schedules have been completed. This approach has an $O((n+1)^{H-1})$ complexity, where the task set's there are n overall jobs, and H is the duration of the excitable phase [14].

Table 8 Comparison with the Minimum Energy Scheduler

H	MAKESPAN/PERFORMANCE TIME		POWER/ENERGY CONSUMPTION (J)	
	Strategy of MES	Strategy of SURE	Strategy of MES	Strategy of SURE
≤ 10	< 1 Sec	< 1 Sec	74	78
≤ 20	> 30 Min	< 1 Sec	121	123
≤ 30	> 30 Min	< 1 Sec	116	120
≤ 40	> 30 Min	< 1 Sec	176	180
≤ 50	> 30 Min	< 1 Sec	138	138
≤ 60	> 1 Day	< 1 Sec	164	164

Conducted studies for H with task temporal parameters and DRS that varied from 10 to 60 time units. Table 8 displays the worst-case values for the Minimum Energy method's execution time and comparing the provided task's minimum energy value to the SURE algorithm. Since it was taking many days to do some task sets, halted when $H > 60$. As per new approach higher orders of the hyper period, table 5.6 demonstrates that the SURE schedule's energy savings are greater than 90% of the ideal option. Furthermore, SURE strategy can be computed in several orders of magnitude less time than the MES schedule.

6. Conclusion

This chapter's major goal is to examine and contrast the SJF-LJF, FCFS, SJF, RR, and Priority algorithms' performance indicators. Algorithms for scheduling are tested on two processors to ensure proper operation. Running time, energy-time ratio, total energy, and total time are investigated for scheduling strategies including SJF-LJF, FCFS, SJF, RR, and Priority. MES and SURE scheduling algorithms taken 74 and 78 joules energy consumption respectively. The proposed approach could be tweaked in the future to execute jobs faster, less energy consumption and hence improve performance by allocating time slots to available CPUs. The complete number of processors has been efficiently employed as a result of this method. This paper can be concluded by stating functional comparative analysis have distinctive information in comparison with existed scheduling algorithms. The proposed PeSche (Performance improved Scheduling) method can be implemented together with the process, burst, waiting, and turnaround time on an embedded real-time system.

Compliance with ethical standards

Acknowledgments

I am thankful to my Research Supervisor Dr. P A Vijaya, Professor and Head in the Department of Electronics and Communication Engineering, BNMIT, Bengaluru. for her valuable guidance and suggestions in analyzing and testing throughout the period of doing this research work. I wish to express my sincere thanks to Management, BNMIT, Bengaluru. for providing ample facilities to do the research work.

Disclosure of conflict of interest



No conflict of interest.

References

- [1] Gerards, M. E., & Kuper, J. (2013). Optimal DPM and DVFS for frame-based real-time systems. *ACM Transactions on Architecture and Code Optimization (TACO)*, 9(4), 1-23. DOI: 10.1145/2400682.2400700

- [2] M. K. Bhatti, C. Belleudy, and M. Auguin, "Hybrid power management in real time embedded systems: An interplay of DVFS and DPM techniques," *RealTime Syst.*, vol. 47, no. 2, pp. 143–162, 2011. 113. DOI: 10.1007/s11241-011-9116-y
- [3] Chen, G., Huang, K., & Knoll, A. (2014). Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination. *ACM Transactions on Embedded Computing Systems (TECS)*, 13(3s), 1-21. DOI: 10.1145/2567935
- [4] Bambagini, M., Marinoni, M., Aydin, H., & Buttazzo, G. (2016). Energy-aware scheduling for real-time systems: A survey. *ACM Transactions on Embedded Computing Systems (TECS)*, 15(1), 1-34. DOI: 10.1145/2808231
- [5] Abdelkader, D. M., & Omara, F. (2012). Dynamic task scheduling algorithm with load balancing for heterogeneous computing system. *Egyptian Informatics Journal*, 13(2), 135-145. DOI: 10.1016/j.eij.2012.04.001
- [6] Alkhashai, H. M., & Omara, F. A. (2016). An enhanced task scheduling algorithm on cloud computing environment. *International Journal of Grid and Distributed Computing*, 9(7), 91-100. DOI: 10.14257/ijgdc.2016.9.7.10
- [7] Mohanty, S. N., Radhika, A., Dahiya, V., Pattanaik, C. R., & Krishamoorthy, S. (2020). An efficient amalgamation of computational models to ensure a secure IoT environment. *International Journal of Control and Automation*. 13(2), 235-243.
- [8] El Amrani, C., & Gibet Tani, H., "Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data," *Journal of Data Mining & Digital Humanities*, vol. 7, no. 2, pp. 1943-3581, 2018. DOI: 10.46298/jdmdh.3104
- [9] Athokpam Bikramjit Singh, Sathyendra Bhat J., Ragesh Raju, Rio D'Souza, "A Comparative Study of Various Scheduling Algorithms in Cloud Computing," *American Journal of Intelligent Systems*, vol. 7, no. 3, pp. 68-72, 2017. DOI: 10.5923/j.ajis.20170703.06
- [10] Lee, Y. H., Reddy, K. P., & Krishna, C. M. (2003, July). Scheduling techniques for reducing leakage power in hard real-time systems. In *15th Euromicro Conference on Real-Time Systems, 2003. Proceedings.* (pp. 105-112). IEEE. DOI: 10.1109/EMRTS.2003.1212733
- [11] Swaminathan, V., & Chakrabarty, K. (2003). Energy-conscious, deterministic I/O device scheduling in hard real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(7), 847-858. DOI: 10.1109/TCAD.2003.814245
- [12] Krishnapura, R., Goddard, S., & Qadi, A. A. (2004). A dynamic real-time scheduling algorithm for reduced energy consumption. *CSE Technical reports*, 72.
- [13] Tia, T. S. (1995). Utilizing slack time for aperiodic and sporadic requests scheduling in real-time systems. *University of Illinois at Urbana-Champaign*.
- [14] Krishnapura, R., Goddard, S., & Qadi, A. A. (2004). A dynamic real-time scheduling algorithm for reduced energy consumption. *CSE Technical reports*, 72. Technical Report # TR-UNL-CSE-2004-0009.

Author's short biography

	<p>M Sreenath: received B.Tech degree from JNTU Hyderabad and M.Tech degree from JNTU Anantapuramu. Pursuing Ph.D in the Dept. of ECE., at BNMIT, Bengaluru (under VTU, Belagavi). Areas of interests are Embedded Real Time systems, Scheduling Algorithms, Microprocessors & Microcontrollers and Control Systems</p>
	<p>Dr. P. A. Vijaya: received Ph.D and M.Tech degrees from IISc, Bangalore, PGDCA from Mysore University and MBA(Information Technology) from SMUDE. Currently working as Professor and Head in the Department of E.C.E., BNMIT, Bengaluru, India. Areas of interests are Operating Systems, Embedded Systems, Real time systems, RTOS and Distributed Systems. She has held several prestigious positions under VTU, Belgaum, India. She has around 125 papers publication in reputed journals and IEEE international conferences and 2 book chapters to her credit. She has guided 7 research scholars for their Ph.D. Degrees and currently guiding 3 Ph.D. students. She is a member of IEEE, ISTE, IEI and other prestigious technical bodies.</p>