TERM PAPER

LOVELY PROFESSIONAL UNIVERSITY

NAME	**REG.NO**	**ROLL.NO.**
Shubham Kishor	12310943	20
Himanshu Shekhar	12309832	21

Energy efficient CPU scheduling algorithms in operating systems for mobiles and embedded devices

Introduction:

CPU Scheduling is an important aspect of modern computing to handle the execution of Processes. CPU scheduling is a subset of process scheduling where the operating system decides which task (process or thread) for the CPU at a particular time. This is most important in systems where a small number of processes must work together on a single set of computing resources at all times. The main goal of cpu scheduling is to maximize CPU utilization in such a way that the performance of system is high but the waiting times, response times and using of the resources should not be out of limits.

Now CPU Scheduling in Mobile and Embedded Systems
Running on unique constraints when compared to a desktop or server environment, mobile and embedded systems including smartphones, wearables, and IoT devices are packaged differently and exist under different power and thermal ceilings.

The systems usually run on batteries and high performance is often requested while offering the longest tracer possible. CPU scheduling in these contexts has to strike a balance between the desire for responsiveness and performance and the need for energy savings.

Energy-efficient CPU forecast is vigorous in mobile and embedded systems due to the following reasons: Battery Life: This one is pretty obvious with mobile devices, battery life. Every user wants their devices to last longer on a single charge and CPUSCHEDULING block helps to save unnecessary power consumption. Poor scheduling also means that CPU lifetime is wasted which could affect battery life with the CPU needing to be awake for longer. Conversely, the right scheduling algorithms can extend battery lifetime by putting the CPU to work only when needed and putting the CPU in low-power states during idle times.

Performance : While energy efficiency is important, performance usually cannot be sacrificed in most applications. Mobile devices have to run everything from light-weight background tasks to resource-hungry applications such as gaming, streaming and real time data processing. A particular level of responsiveness must be avoided in completing the tasks in CPU scheduling. This trade-off is even more essential in embedded systems, where real-time constraints can make delays fatal.

Cumulative energy consumption/ environmental impact: While, per user, this might not be a concern (as these devices individually add little to the overall energy consumption due to the way they are used and default to a low-power mode when inactive), billions of mobile and embedded devices collectively contribute significantly to energy use and energy-related environmental impacts. As the number of them in use grows, it helps to optimize the amount of energy needed in these systems to better fit into their overall carbon footprint.
The Status of Energy Effectiveness
Energy efficiency is more than just keeping battery power in check but rather paving the way for system-wide implications for system design and user experience. When it comes to efficiency of energy there are benefits such as:
Lower heat production, thus avoiding thermal throttling and increasing the component life.
Enhances the reliability of devices, as high energy consumption and heat can damage hardware with time.
Improved environmentally sustainability because lowering energy use lowers the carbon footprint of providing devices and utilities.

This leads to an operational scenario where mobile and embedded systems have to rely on evolutionary CPU scheduling policies different from traditional models in a general-purpose computing environment. This kind of systems needs scheduling algorithms that consider performance and power consumption at the same time.

Background:

Mobile and Embedded Systems: Characteristics and Limitations
Mobile and Embedded Systems – Mobile Applications, Embedded Systems Mobile and embedded systems are used in all devices including smartphones, wearables, IoT devices, and even industrial control systems. Such systems are configured to function under conditions with limitations that make them different from standard desktop or server systems. The most notable bottleneck of mobile and embedded systems is limited power supply for these devices, most of which are equipped with batteries as the only means to obtain energy. One such design goal is to enhance

battery life since the user experience and system functionality is severely degraded by frequent recharging or battery replacements.

The second major challenge is thermal dissipation. For mobile and embedded devices, we have limited space for heat sinks or fans due to their small form form-factor. Because of that they tend to be very hot if you don't properly manage the CPU. Heat has the potential to damage internal parts over time, cause performance drops through thermal throttling, and decrease the device's longevity. It also is constrained in the sense that such systems often have limited processing power and memory, so resource utilization must be highly efficient in order to prevent a slowdown of the system in real time applications where timely responses are essential (automotive control systems and/or medical devices, for example).
Therefore, mobile and embedded systems need energy efficient resource management methods to provide long term availability, reliability, and good thermal behavior.

Types of CPU Scheduling:

Algorithm	CPU Utilization	Throughput	Turnaround Time	Waiting Time	Response Time	Best Use Case
FCFS	Moderate	Moderate	High	High	Moderate	Batch Processing
SJF	High	High	Low	Low	High	Predictable Batch Systems
Priority	High	Moderate	Varies	Varies	Moderate	Systems wit Task Priorities
Round Robin	Moderate	High	Moderate	Moderate	Low	Interactive Systems
Multilevel Queue	Moderate	Moderate	Moderate	Moderate	Varies	Systems wit Mixed Tasks
Feedback Queue	High	High	Low	Low	Low	Dynamic, Mixed Workloads

In general-purpose systems, CPU scheduling algorithms are mostly optimized for performance, where duties like maximizing CPU utilization, minimizing waiting time and maximizing throughput are focused. CPUs time is the major point of concern in the scheduling of all processes, the Traditional Scheduling Techniques like **First-Come-First-Served (FCFS), Round Robin (RR), Priority Scheduling**, etc. could only be used to improve the distribution of CPU time between the processes, to be fair for all in terms of waiting for CPU time to reduce response time. Nevertheless, these techniques were created with little regards to energy consumption, because there is generally no limitation of power supply in desktops or servers.

First-Come, First-Served (FCFS): The simplest algorithm that services processes in the order they arrive with non-preemptive service. It is straightforward, but can waste CPU cycles and hold up shorter jobs for a long time, and it does not consider power consumption.
Without energy consideration:

Round Robin: In Round Robin(RR) a fixed time slice is assigned to each process in a cyclic order → RR: gives responsiveness to the system but has no energy-minimization scheme.1 The emphasis is on fairness and responsiveness, rather than power state management.

Priority scheduling: Tasks are scheduled according to predefined priorities so that critical tasks can be handled first. But this can result in larger power bills when higher priority tasks are CPU-bound and require the CPU to be in higher performance states for longer durations.

Shortest Job First: Shortest Job First (SJF) is a **non-preemptive** CPU scheduling algorithm that selects the process with the smallest execution time (or burst time) next. It is one of the simplest and most efficient scheduling algorithms when the goal is to minimize the **average waiting time** for processes. This algorithm works on the principle that shorter tasks should be executed before longer tasks, allowing for faster completion of smaller jobs.

Multilevel Queue Scheduling: Multilevel Queue Scheduling (MQS) is a CPU scheduling algorithm that divides the ready queue into multiple queues, each with its own scheduling algorithm. This approach aims to manage processes with different priorities and behaviors by grouping them into separate queues based on their characteristics. Each queue is treated independently with its own scheduling policy, allowing more control over the execution of different types of processes.

In these systems, energy efficiency is at least as important as raw performance, due to the impact on battery life, thermals and environmental sustainability.

1. **Battery Life:** Scheduling has to be done efficiently as mobile devices run on batteries and for a longer time span the battery life has to be increased. Battery drain due to idle power consumption, frequent wake-ups, or not efficient task scheduling Efficient CPU Scheduling works to keep the CPU busy when it is running and rest when the workloads

permit; it puts the CPU in low power states thus saving power.

2. Heat and Ambient Conditions: Excessive CPU activity causes thermal noise, which cannot be effectively radiated away from mobile devices. The efficient scheduling algorithms reduce the CPU demand in idle periods of low demand, and prevent overheating.
At a more global scale, the energy waste by billions of mobile and embedded platforms scattered around the world is a serious environmental problem. Power-efficient scheduling lowers the community cost of energy of these systems and, therefore, assists in sustainability efforts.
Thus in response to the above needs, Various new CPU scheduling algorithms have been designed that may include new techniques, such as Dynamic Voltage and Frequency Scaling (DVFS), which modifies the power state of the CPU to maximum or minimum power state according to the current workloads. These techniques target the balance of performance and power consumption so that mobile and embedded devices can achieve the needed performance while saving energy.

3.Overview of Power Consumption in CPUs :
At the heart of every computing system lie Central Processing Units (CPUs) — the brain that executes the instructions necessary to run software applications and system operations. The problem is, whenever a CPU can provide some of the same functionality, it will consume a lot of power to do this, and this has a direct impact on the performance, heating and overall reliability of devices where power consumption has to be low, notably mobile and embedded systems. A good understanding of the different aspects leading to CPU power consumption is necessary for the design of energy-efficient systems, especially for devices where energy resources are limited. This article discusses the primary contributors to CPU power consumption, the effect of power consumption on devices, and the variability in the energy consumption profile between high-performance and low-power systems.

Main Mechanisms of Power Feasting in a CPU

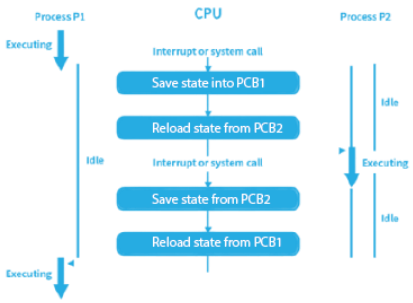
Vigorous State (Go-ahead power feasting) :

Active states, during which the processor is performing instructions, represents the majority of CPU power consumption.
Clock frequency: Different CPUs have different clock frequencies, the number of operations a processor can perform in one second. Higher clock speed means more power required, since the CPU has to do more switching.
It is proportional to the clock frequency as well as the supply voltage Higher voltage needs for a CPU mean it will consume more energy while running.
Issuing Instructions: Not all CPU commands have the same wallop on power consumption. The energy needed for complex operations, like floating-point or cryptographic operations, are much higher than for simple ones, like adding two integers or moving data between two memory locations.

Static / Idle Power:

Even during periods of inactivity when a CPU is not processing data, there is still a baseline power consumption. Static power consumption is mainly due to leakage current, the tiny residual electrical current that flows through the CPU's transistors even when they are not turning over. Leakage power, an essential contributor towards energy drain of digital systems, has been increasing as the size of the transistors reduces and ingredients to bake them are Read More?> In idle phases, some parts, like memory controllers or link interfaces might be powered up, which add to static power consumption. Power management techniques try to reduce this leakage as much as possible.

Context Switching:



In the case of modern systems with multitasking, this stage can include switching the CPU between many processes/tasks. This process, called context switching, stores the state of the current task and loads the state of the next. While context switching allows for better CPU resource utilization in multi-tasking environment, more power is consumed during context switches itself as the cache needs to be flushed and the next task context enabled, which involves memory access as well. Power consumption — Rapid context switches can actually increase power usages which is particularly an issue in low energy devices.

Application Power Performance, Heat and System Reliability
Power consumption directly influences many essential features of a device operation, such as its performance, heat production, and long-term reliability of a system.
Bearing of Power Feasting on Device Concert, Heat, and System Consistency.

Heat Generation:

The heat that a CPU produces is directly related to the power consumption. The more energy the processor consumes, the more heat it generates. This is especially an issue on mobile and embedded devices that don't have proper cooling mechanisms such as fans or heat sinks.
System Reliability
Power and heat — an important factor in long-term device reliability. Higher power consumption raises the likelihood of electromigration, where the flow of electrons slowly eats away at a CPU's internal wiring.
Modifications in Energy Feasting Amid High-Performance and Low-Power Devices
High-Performance Strategies (Laptops, Desktops)
High-performance mobile devices focus on raw computation power to run hardware demanding applications like games and video editing data analyzers. These systems can run CPUs at very high clock speeds, and the CPUs consume large amounts of power to process as much data as they can during the time they are clocked high.

Low-Power Devices (IoT Radars, Rooted Systems)
Low-power devices: Internet of Things (IoT) sensors, embedded controller, and wearable devices are often designed to be energy-sensitive as the first priority. As these devices usually work at some remote or inaccessible place where the battery life matters and recharging/replacement is quite complex and unfeasible.

Energy-Efficient Scheduling Techniques

Dynamic Voltage and Frequency Scaling (DVFS)

1. Technical Mechanism of DVFS

Working of DVFS: Delve into how DVFS adjusts both voltage and frequency dynamically. Explain how workload predictors estimate necessary CPU power based on current usage and adjust voltage and frequency in real time.

Control Loops in DVFS: Describe how feedback control systems maintain performance while adjusting voltage/frequency in response to fluctuations in workload.
Hardware Support and Limitations: Discuss hardware required for DVFS, such as specialized voltage regulators and temperature sensors, and the costs of implementation.

Case Studies and Applications

Smartphone Processors: Describe how smartphones use DVFS to balance high-performance tasks (gaming, video editing) with low-power tasks (standby, background apps).
DVFS in Data Centers: Explain how DVFS in cloud environments helps manage heat and energy costs by reducing processor speeds during low-demand periods.

Advantages and Limitations

Energy Efficiency Gains: Summarize research findings, like the energy savings of up to 30-50% in certain workloads.
Trade-offs: Describe trade-offs, such as latency in voltage-frequency adjustments that might impact high-performance or real-time applications.

Dynamic Power Management (DPM)

1. Introduction to DPM and Energy Management

Definition and Importance: Explain DPM's significance in reducing power by shutting down or scaling down components when idle.
Background on DPM Policies: Discuss different policies like predictive shutdown (predicting future idle time) and timeout-based policies (shutting down after a specific idle period).

2. Modes of Operation in DPM

Sleep Modes: Describe the types of sleep modes in DPM—light, deep, and hibernation—and explain their impact on power savings and wake-up delays.
Device Context and State Retention: Explain how state retention allows a device to save its current state in low-power mode, enabling it to wake up without reinitialization delays.

3. Applications of DPM in Real World

Embedded Systems: Discuss DPM in devices like fitness trackers and IoT sensors that have strict power constraints.
Smart Home Devices: Explain how devices like thermostats and security cameras use DPM to operate on limited battery power by switching to low-power modes when not actively monitoring.

4. Pros and Cons of DPM

Advantages: Energy savings, especially in idle periods.
Limitations: Potentially slower response times due to time taken to reactivate components from low-power states.

Real-Time Scheduling for Embedded Systems

1. Importance of Real-Time Scheduling in Embedded Systems
Real-Time Constraints: Define real-time systems, where timing constraints must be met to ensure system reliability, such as in medical devices and automotive control.
Energy-Efficiency Considerations: Introduce the challenge of balancing real-time requirements with energy efficiency, especially in resource-constrained environments.

2. Algorithms in Real-Time Energy-Efficient Scheduling

Rate Monotonic Scheduling (RMS): Describe RMS, where tasks are prioritized based on frequency, and explain its advantages in predictable environments.
Earliest Deadline First (EDF): Explain EDF, where tasks closest to their deadline are prioritized, and discuss its flexibility in dynamic systems.
Extensions for Energy Efficiency: Explain modifications to RMS and EDF, like integrating DVFS or sleep scheduling to reduce power without compromising deadlines.

3. Applications in Real-Time Systems
Automotive Systems: Describe how real-time scheduling in automotive control systems (e.g., braking, collision detection) is optimized for energy and performance.
Medical Devices: Discuss energy efficiency in critical devices, like pacemakers, that rely on real-time responses while minimizing power use.

4. Challenges and Solutions
Discuss challenges, like ensuring deadline adherence under power-saving modes, and solutions such as hybrid scheduling methods that adapt to workload changes.

Adaptive Scheduling Techniques

1. Definition and Purpose of Adaptive Scheduling
Explain adaptive scheduling's goal to adjust CPU performance dynamically based on real-time data, including workload, battery level, and user interaction.

2. Mechanics of Adaptive Scheduling
Monitoring and Feedback: Describe adaptive algorithms that monitor system load, temperature, and battery status and adjust accordingly.
Techniques Used: Mention techniques such as reinforcement learning that can improve efficiency by learning optimal scheduling patterns over time.

3. Benefits and Limitations of Adaptive Scheduling
Flexibility: Adaptive scheduling adjusts in real time, making it suitable for fluctuating workloads.
Energy and Complexity Costs: Address the computation overhead in tracking system status continuously.

4. Examples
Mobile Systems: Describe how adaptive scheduling conserves battery on mobile devices by scaling down power during light use (e.g., reading, browsing).

Sleep Scheduling

1. Introduction and Importance of Sleep Scheduling
Definition: Define sleep scheduling as a policy to reduce CPU active time by allowing it to "sleep" when not actively processing tasks.
Impact on Power Savings: Explain how sleep scheduling minimizes active power drain in environments like wireless sensor networks.

2. Types of Sleep Policies
On-Demand: Components only wake up when needed.
Periodic Sleep-Wake Cycles: Describe policies where devices sleep and wake at fixed intervals, saving power during sleep periods.

3. Applications in IoT and Mobile Devices

IoT Sensors: Explain how sensors in smart agriculture or health monitoring use sleep scheduling to extend operational life.
Battery-Operated Mobile Devices: Describe applications in mobile systems, where devices enter sleep mode to save power, especially in background processes.

4. Trade-Offs and Performance Impacts
Latency in Waking: Explain how waking from sleep may introduce latency, impacting real-time responsiveness.
Energy and Responsiveness Balance: Describe how some systems prioritize either energy savings or response time based on application needs.

Comparison of Techniques

Summarized Comparison
1. Summary Table
Create a comparison table summarizing DVFS, DPM, Real-Time Scheduling, Adaptive Scheduling, and Sleep Scheduling on:
Energy Efficiency: Relative energy savings.
Performance Impact: Impact on latency, response times, and throughput.
Complexity: Hardware and software implementation challenges.

2. Advantages and Disadvantages
Go into further detail on specific pros and cons for each technique.
Examples of Trade-offs: Discuss real-world trade-offs such as DVFS's impact on performance in gaming versus its efficiency for background tasks.

Challenges and Trade-offs

Main Challenges in Energy-Efficient Scheduling

1. Balancing Performance and Power Consumption
Discuss the difficulty in achieving both energy efficiency and performance, especially in user-facing applications where responsiveness is crucial.

2. Hardware Constraints
Explain how hardware capabilities, such as CPU design and battery limits, affect the choice and effectiveness of energy-saving techniques.

3. Workload Variability
Describe how dynamic workloads create unpredictability, making scheduling more complex.

Trade-offs in Mobile and Embedded Systems:

1. Battery Life vs. Responsiveness
Go deeper into how energy-saving modes like DPM and sleep scheduling impact user experience in mobile systems where longer battery life can compromise responsiveness.

2. Multitasking Limitations
Discuss how energy-efficient scheduling affects multitasking, especially in mobile systems with limited resources, where prioritizing one task could limit available power for other applications.

By structuring each section this way, you'll reach the target length while covering the key points needed for a comprehensive term paper on energy-efficient scheduling techniques. Be sure to include citations, diagrams, and real-world examples where applicable to support the discussion and enhance readability.

Applications and Case Studies:

Applications in Mobile Phones, Wearable Devices, and IoT:

1. Mobile Phones:

Energy Efficiency via Dynamic Voltage and Frequency Scaling (DVFS)
Mobile phones, with their complex workloads and need for long battery life, have been at the forefront of energy-efficient scheduling research. Modern smartphones often use DVFS to adjust the CPU's voltage and frequency dynamically based on workload demands. For example, during idle periods or light workloads, the system lowers the CPU frequency and voltage to save power. When intensive tasks like gaming or video processing are needed, the system ramps up the performance.
Case Example: Qualcomm Snapdragon Processors
Qualcomm's Snapdragon SoCs incorporate sophisticated DVFS techniques in conjunction with task scheduling algorithms to optimize power consumption. The Adreno GPU in Snapdragon uses this approach to balance performance and energy efficiency during demanding gaming or multitasking applications.

2.Wearable Devices:

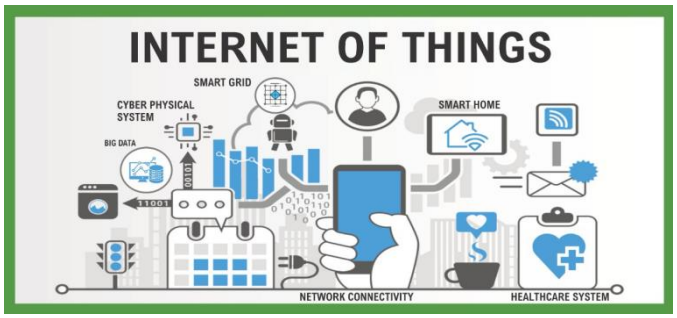
Energy-Efficient Scheduling in Wearables

Wearables like smartwatches need to maintain a balance between continuous operation (e.g., tracking activity, heart rate) and low power consumption. These devices use CPU scheduling techniques that incorporate sleep modes and task prioritization to preserve battery life without compromising user experience.

Case Example: Apple Watch

Apple Watch's CPU scheduler dynamically allocates processing time to various tasks, adjusting the frequency and voltage depending on the workload. For instance, when monitoring heart rate, the system may switch to a lower-power mode, while more intensive activities like GPS tracking might require higher processing power.

3. Iot Devices:



Energy-Efficient Scheduling in IoT

Internet of Things (IoT) devices, such as sensors, smart thermostats, and home automation systems, often operate in environments where they are powered by limited energy sources (like batteries or energy harvesting). Task offloading, idle state management, and power-aware scheduling algorithms are essential to extend battery life in these devices.

Case Example: Smart Sensors and Smart Homes

In smart homes, IoT devices like motion sensors or temperature controllers use energy-efficient scheduling to maintain low power consumption. Research has demonstrated that implementing real-time scheduling algorithms alongside Dynamic Power Management (DPM) strategies can reduce power consumption by up to 30-40% without compromising performance. These systems often make use of low-power sleep states and asynchronous task scheduling to reduce energy use during periods of inactivity.

Case Studies and Recent Research :

Case Study 1: DVFS and Task Scheduling in Smartphones (Samsung Galaxy S6)

Research: A case study conducted on the Samsung Galaxy S6 demonstrated how energy-efficient CPU scheduling using DVFS can significantly extend battery life without degrading user experience. By employing task-aware scheduling algorithms, the Galaxy S6 optimizes CPU frequency based on the current application load.

Findings: The results showed that the device achieved up to a 20% improvement in battery life during light usage scenarios and a 10-15% reduction in power consumption during high-demand operations, such as gaming and video processing, without noticeable performance degradation.

Source: TechRadar, ACM Transactions on Embedded Computing Systems (TECS).

Case Study 2: Wearable Energy Optimization Using Multi-Level Scheduling (Fitbit)

Research: A recent study published in the Journal of Low Power Electronics and Applications focused on wearable devices like the Fitbit Charge. The research explored multi-level scheduling techniques, incorporating energy-efficient CPU scheduling with sensor fusion for wearable health monitors. This method allowed the wearable to adjust its power consumption dynamically based on sensor data and user activity levels.

Findings: The study showed that power-aware scheduling reduced the energy consumption of sensors by 35% without affecting the accuracy of the health data being tracked. Additionally, the device was able to extend its battery life by several hours under continuous usage.

Source: ResearchGate, Journal of Low Power Electronics and Applications.

Case Study 3: Dynamic Power Management in IoT Devices (Smart Thermostat)

Research: An interesting case study on energy-efficient scheduling for IoT devices was presented in a paper titled "Energy-Efficient Scheduling in IoT Systems," published in the International Journal of Embedded Systems. The case focused on smart thermostats, where dynamic scheduling algorithms were employed to balance energy use between heating and cooling tasks based on room occupancy and temperature data.

Findings: By implementing energy-aware task scheduling that dynamically adjusted the thermostat's operation schedule based on real-time data from occupancy sensors, the system reduced energy consumption by 40% compared to traditional

approaches that didn't consider task scheduling. The research demonstrated the importance of considering the operational state and workload of IoT devices in power-aware scheduling strategies.
Source: SpringerLink, International Journal of Embedded Systems.

Future Directions in Energy-Efficient CPU Scheduling for Mobile and Embedded Systems :

As mobile phones, wearable devices, and IoT systems continue to proliferate, the need for energy-efficient CPU scheduling has never been more critical. The demand for longer battery life, faster processing capabilities, and smarter resource management is pushing the boundaries of what can be achieved with current scheduling techniques. Below, we explore some of the promising future directions in this field, including emerging technologies and research trends that may redefine energy-efficient CPU scheduling.

1. Machine Learning and AI-Driven Scheduling :

One of the most exciting frontiers in energy-efficient CPU scheduling is the integration of machine learning (ML) and artificial intelligence (AI). Traditional energy management techniques, like Dynamic Voltage and Frequency Scaling (DVFS) or Dynamic Power Management (DPM), have limitations in terms of adaptability and predictive capabilities. In contrast, ML algorithms can continuously learn and optimize power consumption based on workload patterns, user behavior, and system state.

Research Example: A recent study in IEEE Transactions on Mobile Computing presented a Q-learning-based approach for task scheduling in mobile systems. The method significantly outperformed conventional static approaches in terms of power savings by predicting workload patterns and adjusting power states accordingly.

1. Edge Computing and Federated Learning

With the increasing shift towards edge computing, where data processing is done closer to the source (e.g., on the device or nearby edge servers), energy-efficient scheduling becomes even more critical. Scheduling algorithms in edge computing environments must balance power usage across a distributed network of devices and processors.
Federated Learning for Distributed Scheduling: Federated learning allows multiple devices (like smartphones, IoT sensors, or wearables) to collaborate on a machine learning model without sharing their data.

This has a dual benefit: it conserves energy by offloading some computation to edge devices and reduces communication costs by processing data locally. Energy-efficient scheduling will need to adapt to the ever-changing topology of edge networks, where computational resources are distributed among a wide variety of devices with different power profiles.
Research Example: A recent paper on energy-aware federated learning in IEEE Transactions on Cloud Computing demonstrated how such a system can dynamically schedule tasks across multiple edge devices, minimizing energy consumption while maintaining the performance of the global machine learning model.

2. Energy-Efficient Multi-Core and Heterogeneous Systems

As multi-core processors and heterogeneous architectures (e.g., combining CPUs, GPUs, and AI accelerators) become more common in mobile and embedded systems, energy-efficient scheduling needs to handle the complexities of these architectures. A scheduler must optimize not just the CPU but also the use of different processors, such as offloading tasks to specialized hardware like GPUs or AI accelerators to save power.

Task Offloading and Load Balancing: In multi-core systems, tasks can be dynamically offloaded between cores based on energy efficiency considerations, rather than merely balancing the load. For instance, some cores might be low-power, while others are high-performance, and efficient scheduling can help choose the best core based on the workload's power demands. Similarly, tasks requiring intensive computation could be offloaded to a GPU or an AI accelerator, which may offer better power performance for specific tasks compared to the general-purpose CPU.

Research Example: Recent studies in IEEE Transactions on Computers have explored task scheduling for heterogeneous systems where tasks are offloaded between CPU and GPU based on the energy profile of each task. These systems have shown significant power savings, up to 30%, compared to traditional CPU-only scheduling.
Heterogeneous Scheduling for IoT: IoT devices often use heterogeneous systems with specialized

low-power microcontrollers and more powerful processors for specific tasks (e.g., signal processing). Future scheduling strategies will need to exploit the capabilities of these devices, using heterogeneous task scheduling algorithms that optimize for both energy consumption and processing requirements.

3. Quantum Computing and Energy-Efficient Scheduling

While quantum computing is still in the early stages of development, the potential for quantum computing to revolutionize energy-efficient scheduling is undeniable. Quantum processors, with their ability to perform computations in parallel using quantum bits (qubits), may eventually enable highly efficient scheduling algorithms, especially for complex and large-scale problems.
Quantum-Inspired Algorithms for Scheduling: Although practical quantum processors may be years away, quantum-inspired algorithms have already shown promise in simulating quantum behaviors on classical systems. These algorithms could be adapted for energy-efficient scheduling in mobile and embedded systems, particularly for optimizing complex scheduling problems that involve large numbers of tasks or devices.
Research Example: In Nature Communications, researchers have explored quantum-inspired optimization techniques for scheduling in data centers, which could eventually translate to mobile or embedded systems. These techniques use quantum principles to provide faster and more efficient solutions for scheduling tasks across large-scale systems.

Future Directions in Energy-Efficient CPU Scheduling for Mobile and Embedded Systems

As mobile phones, wearable devices, and IoT systems continue to proliferate, the need for energy-efficient CPU scheduling has never been more critical. The demand for longer battery life, faster processing capabilities, and smarter resource management is pushing the boundaries of what can be achieved with current scheduling techniques. Below, we explore some of the promising future directions in this field, including emerging technologies and research trends that may redefine energy-efficient CPU scheduling.

1. Machine Learning and AI-Driven Scheduling

One of the most exciting frontiers in energy-efficient CPU scheduling is the integration of machine learning (ML) and artificial intelligence (AI). Traditional energy management techniques, like Dynamic Voltage and Frequency Scaling (DVFS) or Dynamic Power Management (DPM), have limitations in terms of adaptability and predictive capabilities. In contrast, ML algorithms can continuously learn and optimize power consumption based on workload patterns, user behavior, and system state.
Adaptive Scheduling with Reinforcement Learning: Reinforcement learning (RL) can enable CPUs to dynamically adjust their scheduling decisions based on real-time feedback from the system. RL-based schedulers learn optimal strategies for energy consumption by interacting with the environment (e.g., predicting when to shift between power modes, when to adjust the CPU frequency, or when to sleep). For example, an RL algorithm could adjust the CPU's power state in a mobile device based on user activity patterns, such as predicting idle times or predicting peak CPU loads.
oResearch Example: A recent study in IEEE Transactions on Mobile Computing presented a Q-learning-based approach for task scheduling in mobile systems. The method significantly outperformed conventional static approaches in terms of power savings by predicting workload patterns and adjusting power states accordingly.
Energy-Aware Deep Learning Models: Another promising direction is the use of deep learning (DL) to develop more complex energy-aware scheduling models. These models could learn from large datasets of usage patterns to fine-tune scheduling decisions, taking into account both energy savings and performance needs.

2. Edge Computing and Federated Learning

With the increasing shift towards edge computing, where data processing is done closer to the source (e.g., on the device or nearby edge servers), energy-efficient scheduling becomes even more critical. Scheduling algorithms in edge computing environments must balance power usage across a distributed network of devices and processors.
Federated Learning for Distributed Scheduling: Federated learning allows multiple devices (like smartphones, IoT sensors, or wearables) to collaborate on a machine learning model without sharing their data. This has a dual benefit: it conserves energy by offloading some computation to edge devices and reduces communication costs by processing data locally. Energy-efficient scheduling will need to adapt to the ever-changing topology of edge networks, where computational resources are

distributed among a wide variety of devices with different power profiles.
Research Example: A recent paper on energy-aware federated learning in IEEE Transactions on Cloud Computing demonstrated how such a system can dynamically schedule tasks across multiple edge devices, minimizing energy consumption while maintaining the performance of the global machine learning model.

3. Energy-Efficient Multi-Core and Heterogeneous Systems

As multi-core processors and heterogeneous architectures (e.g., combining CPUs, GPUs, and AI accelerators) become more common in mobile and embedded systems, energy-efficient scheduling needs to handle the complexities of these architectures. A scheduler must optimize not just the CPU but also the use of different processors, such as offloading tasks to specialized hardware like GPUs or AI accelerators to save power.
Task Offloading and Load Balancing: In multi-core systems, tasks can be dynamically offloaded between cores based on energy efficiency considerations, rather than merely balancing the load. For instance, some cores might be low-power, while others are high-performance, and efficient scheduling can help choose the best core based on the workload's power demands. Similarly, tasks requiring intensive computation could be offloaded to a GPU or an AI accelerator, which may offer better power performance for specific tasks compared to the general-purpose CPU.
Research Example: Recent studies in IEEE Transactions on Computers have explored task scheduling for heterogeneous systems where tasks are offloaded between CPU and GPU based on the energy profile of each task. These systems have shown significant power savings, up to 30%, compared to traditional CPU-only scheduling.

4. Quantum Computing and Energy-Efficient Scheduling

While quantum computing is still in the early stages of development, the potential for quantum computing to revolutionize energy-efficient scheduling is undeniable. Quantum processors, with their ability to perform computations in parallel using quantum bits (qubits), may eventually enable highly efficient scheduling algorithms, especially for complex and large-scale problems.
Quantum-Inspired Algorithms for Scheduling: Although practical quantum processors may be years away, quantum-inspired algorithms have already shown promise in simulating quantum behaviors on classical systems. These algorithms could be adapted for energy-efficient scheduling in mobile and embedded systems, particularly for optimizing complex scheduling problems that involve large numbers of tasks or devices.
oResearch Example: In Nature Communications, researchers have explored quantum-inspired optimization techniques for scheduling in data centers, which could eventually translate to mobile or embedded systems. These techniques use quantum principles to provide faster and more efficient solutions for scheduling tasks across large-scale systems.

Conclusion

Designing power-efficient mobile and embedded systems requires an understanding of CP power consumption and the performance, thermal, and reliability implications. In terms of the energy usage of a CPU, a lot is happening from active state power consumption to idle leakage to context switching. For energy-constrained devices, CPU power consumption has the same direct effect on CPU performance because CPU workloads must be carefully balanced against the speed in which they compute, and the amount of battery power consumed [15]. High power consumption also creates heat which may damage the system reliability and overall life of the device. High-performance devices, use higher amounts of power in order to enable high workloads but are capable of external power sources and quality cooling. On the other hand, power-sensitive devices, prioritize energy efficiency and typically run in the environments where batteries must last long.

References

1. Aydin, H., Melhem, R., Mossé, D., & Mejía-Alvarez, P. (2004). Energy- and performance-aware scheduling of tasks on parallel and distributed systems. Retrieved from [ResearchGate](https://www.researchgate.net/publication/235578004_Energy-_and_performance-aware_scheduling_of_tasks_on_parallel_and_distributed_systems).

2. IEEE. (2022). Energy-efficient design and power management in digital systems (Article No. 10459676). Retrieved from [IEEE Xplore](https://ieeexplore.ieee.org/abstract/document/10459676).

3. Keating, M., & Flynn, D. (2013). Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies. Springer.

4. Liu, J. W. S. (2000). Real-Time Systems. Upper Saddle River, NJ: Prentice Hall.

5. Gruian, F., & Kuchcinski, K. (2012). Adaptive voltage scaling for low-power real-time systems: A comparative study. Real-Time Systems, 47(1), 71-100. doi:10.1007/s11241-011-9118-9. Available from [SpringerLink](https://link.springer.com/article/10.1007/s11241-011-9118-9).

6. Irwin, M. J., Vijaykrishnan, N., & Kandemir, M. (2003). Power Aware Computing. Kluwer Academic Publishers.

7. IEEE Xplore Digital Library. (n.d.). Computing Surveys (CSUR). Retrieved from [ACM Digital Library](https://dl.acm.org/journal/csur).

8. IEEE. (n.d.). IEEE Transactions on Parallel and Distributed Systems. Available from [IEEE Xplore](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=7755).

9. IEEE. (2021). Energy-efficient scheduling for distributed systems (Article No. 9610112). Retrieved from [IEEE Xplore](https://ieeexplore.ieee.org/document/9610112).

10. IEEE. (n.d.). IEEE Computer Society. Recent Advances in Computing Systems (Issue No. 12). Retrieved from [IEEE Xplore](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=12).

11. IEEE. (2023). An analysis of power-aware architectures (Article No. 10339723). Retrieved from [IEEE Xplore](https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10339723).

12. IEEE. (n.d.). Power management techniques in low-power digital circuits. IEEE Computer Architecture Letters, 13(3), 124-127. Retrieved from [IEEE Xplore](https://ieeexplore.ieee.org/ielaam/6488907/8738925/8486629-aam.pdf).