DOI: http://dx.doi.org/10.26483/ijarcs.v16i4.7322

ISSN No. 0976-5697

Volume 16, No. 4, July-August 2025

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

TEMPERATURE-AWARE TASK SCHEDULING IN MOBILE SYSTEMS: A SURVEY OF ALGORITHMS AND HARDWARE SOLUTIONS

Sandeep Gupta
SATI, Vidisha, India

***Abstract*—**Temperature-based task scheduling has become a relevant remedy to thermal safety without compromising performance and energy efficiency. Increasing complexity and performance needs of mobile and embedded processors have placed energy consumption and thermal stability as system design issues. The increased compactness and power of the devices cause a problem of too much heat being produced which might lead to reduced performance, low reliability and hardware short life. Here the strategy to alleviate such issues has come in the form of temperature-aware task scheduling, which has proven to be an effective way of optimizing task scheduling to address both performance and thermal limits. Through this survey, a rigorous survey of algorithmic as well as hardware-level approaches in thermal-aware scheduling is presented. It discusses both the techniques of static and dynamic schedules, such as heuristic scheduling and optimization programming and predictive scheduling. Lt also examines the hardware-enabled approaches like Dynamic Voltage and Frequency Scaling (DVFS), thermal sensors, and thermal throttling services. The application in real-life settings is also presented and remarks provided on the measures of performance applied to measure performance of scheduling strategies in the review. With the insight into the existing research and transforming trends, this paper seeks to convey an in-depth view of the space that is temperature-aware scheduling in mobile systems and direct the future advancement of thermally efficient designing systems.

***Keywords*—**Temperature-aware scheduling, task scheduling algorithms, DVFS, multi-processor system-on-chip (MPSoC), embedded processors, low-power design.

I. INTRODUCTION

The growth in processing capability of current (micro)processors is exponential and a striking rise in energy consumption on all computing platforms including small handheld computers and large-scale data centers has mirrored this expansion of processing power. This surge has led to significant heat emissions, resulting in high operating temperatures that adversely affect processor performance and reliability [1]. Elevated temperatures not only degrade the lifespan of chips but also risk permanent hardware damage. Consequently, manufacturers have set strict thermal thresholds and incorporated cooling systems that operate almost continuously [2]. However, the energy consumed and heat emitted by these cooling mechanisms contribute further to the overall system energy footprint.

The scheduling of tasks is very important. Contrasted with task allocation, which handles the timing and sequencing of tasks once allocation is complete, task scheduling is formally described as the problem of assigning a set of tasks to a set of processors (or robots) in a way that optimizes multiple objectives, such as total completion time, workload balancing, or travel distances [3]. Efficient scheduling must now also account for thermal constraints to ensure reliable and energy-efficient operation.

As processor power consumption continues to rise rapidly with each generation, energy and temperature management have become a critical issue [4]. This is especially challenging since advances in cooling technology have not kept pace with the increase in heat generation. To mitigate this, a range of research efforts have focused on incorporating energy and thermal awareness into scheduling algorithms at the system and operating system levels. Modern processors support mechanisms such as Dynamic Voltage Scaling (DVS), which enable the real-time control of processor speed and voltage, allowing for dynamic thermal and energy management.

These days, thermal management isn't limited to central processing units. In the world of electric vehicles and mobile devices, batteries and related components are a big cause for concern [5]. Battery performance and safety are highly dependent on temperature conditions; hence effective thermal management systems are necessary [6]. Additionally, the 3G/4G network interface, a crucial part of mobile devices, uses a lot of power. The main culprit here is tail energy, which maintains power to the radio interface long after data transmission has ended [7]. Together, these challenges highlight the importance of developing temperature-aware task scheduling strategies that not only optimize performance and energy consumption but also maintain thermal safety across heterogeneous mobile computing environments [8].

A. *Structure of the paper*

The structure of the paper is as follows: **Section II** discusses the Classification of Temperature-Aware Task Scheduling Approaches; **Section III** explores software-based scheduling Algorithms; **Section IV** presents Hardware-Assisted Thermal Management Techniques; **Section V** offers a comprehensive literature review of recent RL-based energy optimization approaches; and **Section VI** concludes with key findings and future research directions.

II. CLASSIFICATION OF TEMPERATURE-AWARE TASK SCHEDULING APPROACHES

A number of techniques are employed in temperature-aware task scheduling for mobile systems in order to control thermal behavior while preserving system performance. Although many earlier models focused on energy and thermal management at the micro-architecture level, it shifts the focus to the operating system level for thermal management. When it comes to reducing peak temperatures, most previous research has focused on multi-core systems, which allow for the dynamic migration of tasks between cores [9]. But recent work has demonstrated that thermal efficiency can be enhanced even in systems with

only a single core by exploiting differences in heat generation between tasks. Managing processor temperature more efficiently can be accomplished by smart time scheduling of tasks with considerations of their thermal property. Dynamic thermal management (DTM) hardware continually reads the temperature given on the chip, and acts in response to the temperature going too high. To mitigate overheating, the system automatically reduces the CPU frequency commonly by half, and potentially down to one-fourth or one-eighth, if necessary, thereby reducing heat output. The cooling system is always running, bringing the processor's temperature closer to that of the surrounding environment.

A. Thermal Management in Mobile Systems

Thermal management in mobile systems goes beyond conventional energy-saving mechanisms by ensuring that chip temperatures remain within safe thermal thresholds [10]. When a device approaches or exceeds these thresholds, throttling techniques are typically employed to reduce temperature by lowering performance or temporarily disabling certain components. Most of the approaches discussed in this section focus specifically on managing thermal issues in mobile MPSoC (Multi-Processor System-on-Chip) platforms. Although numerous thermal management strategies have been proposed over the years, the majority of research has focused on general-purpose processors and Network-on-Chip (NoC) architectures, rather than the distinct characteristics and constraints of mobile Many-Core System-on-Chip (MPSoC) architectures [11]. However, modern computing devices from smartphones to complex cyber-physical systems increasingly rely on MPSoC architectures to meet growing demands for performance and energy efficiency. As a result, effective and targeted thermal management solutions tailored to mobile MPSoC platforms are becoming critically important.

B. Static vs. Dynamic Scheduling

Task scheduling in mobile systems is broadly classified into Static and Dynamic approaches. Static Scheduling (SS) makes scheduling decisions during the compilation process and is apt to program segments where the control flow and execution time are predictable. Dynamic Scheduling (DS) on the other hand modifies its decisions at run time thus it is suitable for workloads with variable or unpredictable behavior [12]. DSS is a kind of hybrid architecture, where SS is used to optimize the regularly program areas and DS is used on the rest of the program parts. This moderate approach improves performance and performance efficiency at the same time. There is a succinct comparison of the two methods in Table I:

TABLE I. STATIC VS. DYNAMIC SCHEDULING

Aspect	Static Scheduling (SS)	Dynamic Scheduling (DS)	Hybrid (Static + Dynamic Scheduling - DSS)
Definition	Scheduling decisions made at compile time	Scheduling decisions made at runtime	Combines static and dynamic scheduling to balance performance and area
Use Case	Suitable for code sections with simple control flow and fixed latency	Ideal for irregular, unpredictable control flows	Static parts identified (e.g., via pragmas), and remaining code scheduled dynamically
Flexibility	Less flexible; does not adapt to runtime conditions	Highly flexible; adapts to dynamic system behavior	Offers a balance between flexibility and efficiency
Performance Optimization	Optimized for known, deterministic workloads	Optimized for variable workloads	Targets minimal area usage and maximal performance
User Involvement	Compiler or user annotates fixed-schedule regions	System autonomously handles scheduling	Current implementations require user annotations; future work aims for automation
Integration Strategy	Applied to specific parts of a program	Applied to the remaining parts after static regions are handled	Areas that remain static are handled as opaque entities while the remainder of the program is dynamically scheduled.

DSS a compromise between SS and DS, which seeks low area and high performance, illustrated in Figure 1:

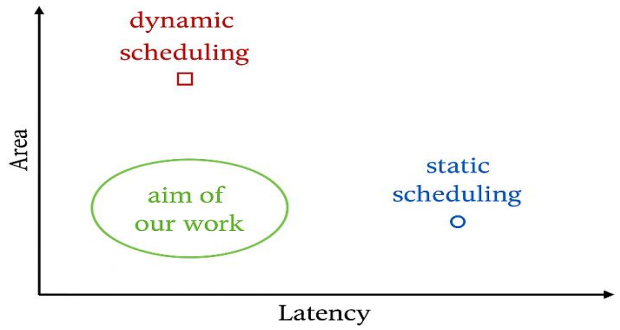

Fig. 1. A Sketch Comparing the Design Quality of Different Scheduling Approaches.

C. Reactive vs. Proactive Scheduling

Reactive vs. Proactive Scheduling Reactive-proactive scheduling has the twin features of combining two complementary steps to arrange the execution of tasks in an uncertain setting. During the proactive period, a baseline plan is developed based on the statistical information in order to foresee disturbances. This plan strong which implies that it is not overly affected by unplanned events when implementing the plan. Reactive scheduling comes into action when deviations or disruptions are experienced during the run time, and this is done by modifying the baseline schedules to adapt to the real-time differences [13]. The plan which is thus obtained is known as the realized schedule. Robustness in this case is the capability of base schedule that is able to take up or defend against interruptions with a minimum performance loss. Robustness has been discussed in many fields in various ways, resulting in varied definitions and criteria of evaluation. This means an

increased requirement for common standards of robustness and measurement procedures.

D. Thermal-Aware Real-Time Scheduling

Real-time task scheduling with temperature management is called thermal-aware real-time scheduling. This way, jobs can be limited by time constraints and also be sure to stay within safe temperature ranges. Particular real-time schedulers, like Earliest Deadline First (EDF) or Rate Monotonic (RM) schedulers, benefit from thermal awareness in embedded and mobile systems when timing guarantees are crucial and thermal headroom is minimal. These are dynamic modifications of task hierarchies with temperature, recovery of idle time to cool down, and using forward-looking schedule slippage to prevent overheating. The scheduler either relies on real-time temperature data or predictive models to make decisions and hence high-priority tasks run on time without thermal throttling. Such a balance between thermal control and time accuracy is critical to ensuring reliability as well as performance of thermally limited platforms, e.g. wearables, smartphones, and automotive embedded systems.

III. Software-Based Scheduling Algorithms

The application of software scheduling algorithms is critical to optimal task execution within computing contexts and more specifically, within operating systems, real-time systems and cloud computing systems. These algorithms are calculated to decide the order and time, at which the process or tasks would be assigned to the resources like CPU or virtual machines. Priority Scheduling, Round Robin, First-Come, First-Served (FCFS), and Shortest Job Next (SJN) are some of the most well-known scheduling strategies. Other, more practical methods like Multilevel Queue and Multilevel Feedback Queue are also suggested [14]. To meet strict timing constraints in real-time systems, techniques such as Earliest Deadline First (EDF) and Rate Monotonic Scheduling (RMS) are used. Trust-aware and energy-efficient scheduling schemes are emerging in distributed environments as well as in cloud computing environments with the intention of delivering reliability, security, and maximization of resource use availability. The choice of scheduling algorithm significantly impacts system performance metrics like turnaround time, throughput, and response time, making it a critical aspect of system design and resource management.

A. Thermal-Aware DVFS (Dynamic Voltage and Frequency Scaling)

As shown in Figure 2, the suggested temperature-aware DVFS architecture is based on. Because logic gates' delays change with temperature, a ring oscillator can double as a thermometer. The decrease in oscillation frequency (f_{osc}) [15] is due to an increase in the gate delay as a consequence of rising ambient temperature. When the need to detect temperature arises, a counter is set to operate for a predetermined length and used to track this oscillator's output. The possibility of chip overheating is indicated by a decreased (f_{osc}) If the counter value drops below a set threshold. As a result, the system reduces the supply voltage, which in turn minimises power usage. Lessening the supply voltage helps lower the chip temperature because, as shown in Equations (1) and (2) f, Dynamic power and leakage both drop with lower voltages.

$$P_{dynamic} = \propto CV^2{}_{D_D} f \quad (1)$$

$$P_{leagage} = I_{leagage}.V_{D_D} \quad (2)$$

As the temperature drops, gate delays shorten, and (f_{osc}) increases. To improve performance, the supply voltage can be increased when the counter value goes beyond a certain threshold, which indicates that there is enough cooling. For Dynamic Voltage Scaling (DVS), the procedure is shown in Figure 2 (a), where the adjustment is made to only V_{D_D}; for DVFS, both V_{D_D} The clock frequency f is scaled, as seen in Figure 2(b). Notably since lower V_{D_D} Leads to longer gate delays, the clock frequency must also be reduced in DVFS schemes to maintain timing reliability.

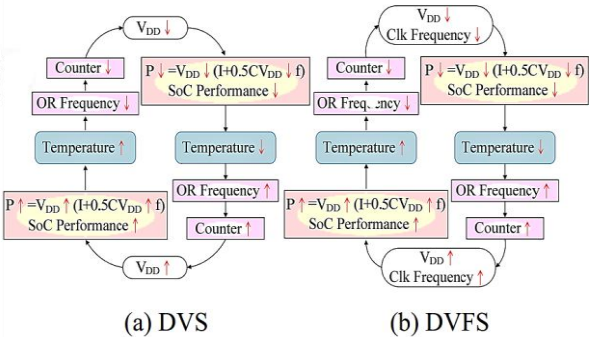

Fig. 2. Dynamic Voltage Scaling (DVS)

B. Heuristic-Based Scheduling Algorithms

In order to complete a task, a resource must be chosen from a pool of potential options that meet all of the criteria [16]. Although all resources in the list meet the minimum requirements, a scheduling algorithm is essential to determine the most suitable resource for efficient task execution. Heuristic-based scheduling algorithms are widely used to make such decisions, aiming to optimize performance, minimize completion time, or balance load. Key heuristic algorithms include:

- **Min-Min Algorithm:** The Min-Min method starts with the set of all unassigned tasks, which is called the meta-task (MT) set. The first step is to determine the shortest amount of time that each MT task is likely to take. The second step is to find the resource that can do the work with the overall minimal predicted time. The earliest jobs are given priority in this technique.
- **Max-Min Algorithm:** The second phase of the Max-Min method is different from Min-Min, while the two algorithms share a similar structure. It uses the maximum expected completion time from the MT set to assign the job to the corresponding resource, rather than the task with the lowest completion time. In an effort to decrease total make span, this strategy gives higher priority to longer tasks, with the goal of tackling tasks of longer durations first.
- **Switcher Algorithm:** The minimum completion times of unassigned jobs are used to determine the Switcher algorithm's dynamic choice between the Min-Min and Max-Min methods. It finds the position in the list when the time difference between two consecutive jobs is greater than the standard deviation. Based on this comparison, it switches to the more appropriate algorithm for each scheduling instance.
- **Suffrage Algorithm:** Each task's minimum and second minimum predicted completion times are calculated using this method. A person's suffrage value is the difference between these two numbers. The task with the highest suffrage value indicating that its optimal resource assignment is more critical is then scheduled on the machine that offers the minimum completion time.

C. Thermal-Aware Allocation and Scheduling

The thermal-aware ASP considers temperature when finding a solution to the thermal issue. The total system temperature is determined by the power consumption, size, and location of each processor element (PE) on the embedded system platform. Hotspot, a thermal modelling program, is used to get the temperature profile [17]. Hotspot provides a simple compact model that takes into consideration the internal heat dissipation of each PE as well as the exterior heat flow between PEs. For each function block, Hotspot can generate accurate temperature estimates by taking the system topology and power usage into account. In order to initiate the thermal-aware application programming interface (API), transmit to the Hotspot the aggregate power consumption of all PEs as well as the power consumed by the ongoing scheduled task. By averaging the temperatures returned from the Hotspot, they may compute the dynamic criticality, which was previously defined. Minimizing the average temperature is the goal of replacing the Pow term with the newly added Avg. Temperature. This objective also necessitates lowering the maximum temperature.

IV. Hardware-Assisted Thermal Management Techniques

Mobile and embedded systems require high efficiency and thermal stability, which can be effectively achieved through hardware-assisted thermal management. This strategy incorporates specialized hardware components, e.g. thermal sensors, dynamic thermal management (DTM) engines and thermal throttling, into processor and system-on-chip (SoC) designs. This allows dynamically varying performance-related settings to be made, including supply voltage, clock frequency, and workload distribution, among processing cores [18]. Such real-time thermal information can be used right away to adjust to thermodynamic changes and therefore system reliability and effectiveness. Hardware-based thermal control entails DVFS modules which are typically embedded into processors. The dynamic power optimization used in these modules is based on thermal feedback that prevents overheating of the modules compared to when power consumption is kept constant to allow optimal performance at the higher levels [19]. DVFS helps to reduce the chances of temperature failures by diffusing power draw under the condition of high temperature, though the processing capabilities are not reduced much.

A. System Hardware Model

The physical embodiment and structure of a computer system are specified in the system hardware model, which implements the software and performs the computation. All these models commonly have the processing unit (CPU/GPU), memory unit [RAM, cache], storage unit (HDD and SSD), input/output devices, and communication ports. In a multi-core or distributed implementation, the hardware model involves some headcount of processors or nodes that are interconnected and each has its memory since processing of information occurs [20]. The design of effective scheduling algorithms relies on the model, as it affects execution time, resource competition, and any communication overhead. Performance optimization, balanced load, and the ability to enable concurrency in tasks across different computing architecture types, such as embedded systems, clouds, and HPCs, require a precise understanding of hardware architecture.

B. Thermal Management Strategies

The ability to manage thermal performance, battery pack life and security of electric vehicle (EV) battery packs relies on thermal management strategies. Such strategies usually involve active cooling, passive cooling, and cooling insulation, hoping to keep the individual operating temperatures optimal, and avoid thermal runaway (TR). The most effective methods in cooling include active cooling where liquid is used to cool down the heat produced during charging and releasing the heat. These systems actively circulate coolant around battery cells which gives accurate thermal regulation. Passive cooling techniques on the other hand, such as the utilization of phase change materials (PCMs) assist in the process of the temperature variation by not bringing in extra energy instead [21]. High-load periods refuse to be wasted by PCMs, which are able to absorb the surplus to keep the heat till its release in a few moments and become originators of energy-efficient thermal stability. Safety is also increased by thermal insulation that reduces the outside thermal influences, provides stable internal temperatures. This minimizes the chances of getting overheated and guarantees regular battery performance. The synergy between these strategies allows EV battery systems to achieve comprehensive thermal performance, higher operation safety, and battery life.

C. Hotspot Thermal Model

Hotspot computes temperatures of processors by simulating the thermal behavior of an electronic system in a manner similar to how electrical networks are analyzed, with heat flow modeled much as the current flow of a network of thermal resistors and capacitors. Floorplan The input to the simulator is a floorplan, which describes the physical arrangement, placement, and connectivity of the different processor elements [22]. The heatsink, fan and thermal interface material are also provided in the thermal model in order to provide realistic simulation. A variation of a previous floor layout is adapted to the new setup here, with 4 cores in the layout branched out but each one is smaller in size. Every core includes all elements to handle an out-of-order pipeline. Four cores are also interconnected with each other by single L2 cache and on-chip interconnection because this is a typical architecture of multicore processors. Hotspot can do both steady-state and transient temperature profiles. Although a considerable number of past research have focused on steady-state analysis, which is applicable in introducing approximations of long-term thermal conduct even in the presence of short-time simulation duration, it can prove quite challenging to utilize steady-state analysis in certain instances, specifically at periods that are much less than the periods of thermal response.

V. Literature Review

This literature Summary brings new developments in the field of temperature-sensitive task scheduling, including heuristic, optimization, and machine learning methods. A particular focus is given to energy efficiency, thermal stability, and performance on a wide range of platforms, even in cloud, fog, embedded, and trusted execution environments.

Kasturi et al. (2025) an Energy and Temperature conscious scheduling policy based on the Earthworm Optimization Algorithm (EOA) to effectively optimize energy consumption, thermal status and performance of systems. The proposed EOA takes the foraging behavior that the earthworm uses and optimizes Task Scheduling (TS) by finding energy-efficient nodes and assigning workload to minimize temperature hotspots. Introducing energy consumption rates as well as temperature barrier into the fitness function allows the algorithm to dynamically assign the tasks in order to minimize the energy usage and avoid the thermal overload on both cloud and fog layers. The results of the experiments prove that the EOA-based scheduling lowers overall energy consumption and sustains constant temperature levels by a significant margin as compared

to the traditional algorithms, such as HDDPGTS, EEOA, and MAO [23].

Li et al. (2024) examine performance penalty in parallel implementation of the cross-device resource sharing tasks. Then, a novel multi-task perceiving and scheduling framework (MTPS) is proposed to guarantee the quality of service of the parallel tasks. The basic idea of MTPS is to first build a master-slave system model to reorganize mobile devices under the same network. Then, MTPS perceives the running cross-device resource sharing tasks and schedules the parallel execution of multiple tasks to avoid mutual interference. Experimental results on real devices show that MTPS can reduce the average completion time of file sharing by 63.5%, and maintain at least 24 frames per second for screen casting at optimal levels in the presence of other tasks. The prevalence of cross-device resource sharing enables users to utilize various device resources of the connected mobile devices seamlessly [24].

Peng et al. (2023) a two-stage temperature-aware scheduling scheme for approximate tasks based on homogeneous multicore platforms, solving approximate task scheduling problem under time, energy, reliability, and temperature constraints. In the first stage, it can design a heuristic scheduling algorithm to perform task-to-processor assignment and preliminarily task frequency selection to minimize energy and make span. In the second stage, it can design a temperature optimization algorithm that combines DVFS and slack distribution to ensure that processor temperature is lower than the temperature threshold. Compared with the existing methods, the experimental simulations show that method achieves 97.6% runtime reduction and 14.9% peak temperature reduction [25].

Wang et al. (2023) a thorough evaluation of the efficiency of the freely available TEE encryption algorithm. Next suggest an ETS-TEE, a task scheduling technique that is very energy efficient. Policy considers the intricacy of TA jobs through the application of deep learning. Offloading to an edge server and local device modelling are two ways these tasks are dynamically scheduled. They use a Raspberry Pi 3B to evaluate the method as both a local mobile device and an edge server. The results show that the method reduces energy consumption by an average of 38.0% and increases speed by 1.6 times compared to the default scheduling strategy on the local device. Creating a trusted execution environment that is both quick and secure, greatly reduces the performance hit that mobile devices might impose, ensuring the secure execution of programs [26].

Nong et al. (2022) an architectural framework that takes energy economy into account when managing resources, taking thermal factors into account. A layered architecture underpins the framework, which includes a suite of intuitive client tools and middleware that accounts for temperature conditions while allocating tasks within and across data centers. The creation of a data center-specific job scheduling component that takes thermal considerations into account is the primary emphasis of this article. In light of upcoming efforts to reduce data center energy costs, this component is essential for maintaining a stable temperature distribution within a single data center [27].

Pourmohseni et al. (2022) a strategy for migrating tasks in order to optimize thermal performance in systems with multiple cores and different types of hardware. The proposed strategy is based on a thermally safe analytical power-budgeting method that uses Dynamic Voltage and Frequency Scaling (DVFS) for electricity and heat management. The migration policy aggressively implements DVFS in an effort to optimize the system's performance while simultaneously guaranteeing thermal safety. To achieve this goal, it iteratively fine-tunes the distribution of active cores in the system (via suitable migration decisions) to maximize their thermally safe power budget, which enables them to operate on higher frequencies without being overheated [28].

Lee (2021) effectively controls the temperature of embedded systems by combining two procedures: dynamic thermal management in real-time and a utilization bound that takes thermal awareness into account. In order to adhere to the chip temperature restriction which is influenced by variables like system configurations, workloads, environmental conditions, and chip cooling capacity the former specifies a maximum allowable processor utilization. The latter takes into account the thermal-aware utilization bound and optimizes the execution rates of specific tasks. By studying a vehicle controller, they were able to improve system utilization by 18.2% over earlier approaches and demonstrate the thermal-aware utilization bound [29].

Table II presents a comparative summary of recent temperature-aware task scheduling strategies, highlighting diverse algorithmic approaches, key findings, encountered challenges, and prospective future research directions in mobile systems

TABLE II. LITERATURE SUMMARY ON TEMPERATURE-AWARE TASK SCHEDULING IN MOBILE SYSTEMS

Author	Study On	Approach	Key Findings	Challenges	Future Directions
Kasturi et al. (2025)	Energy and temperature-aware scheduling in cloud and fog systems	Earthworm Optimization Algorithm (EOA) using energy-efficient node selection and thermal constraints	Reduced energy consumption and thermal hotspots compared to HDDPGTS, EEOA, and MAO	Balancing real-time performance and environmental constraints	Extend EOA to heterogeneous edge environments and real-time IoT scenarios
Li et al. (2024)	Temperature-aware scheduling for approximate tasks on homogeneous multicore platforms	Two-stage scheduling scheme. Heuristic task-to-processor assignment with preliminary DVFS for energy and makespan optimization.	Achieved 97.6% runtime reduction and 14.9% peak temperature reduction compared to baseline methods	Balancing trade-offs among energy efficiency, reliability, and temperature within task deadlines; complexity in coordinating	Extend to heterogeneous systems, integrate machine learning for dynamic prediction, explore real-time adaptive scheduling for dynamic workloads

		Temperature optimization using DVFS and slack distribution		scheduling and thermal management	
Peng et al. (2023)	Scheduling of approximate tasks under time, energy, and temperature constraints	Two-stage heuristic: task-to-core assignment with frequency scaling, followed by DVFS + slack distribution for thermal control	Achieved 97.6% runtime reduction and 14.9% peak temperature reduction	Handling approximation accuracy with thermal limits	Incorporating adaptive approximation models and extending to multi-application systems
Wang et al. (2023)	Conserving Power Through Efficient Task Scheduling in TEE	Deep learning-based ETS-TEE policy on mobile–edge systems	Ensured secure execution while achieving a 38.0% decrease in energy consumption and a 1.6-fold increase in speed.	TEE overhead and device limitations	Explore hybrid secure task offloading with dynamic trust assessment
Nong et al. (2022)	Middleware for thermally-aware work scheduling in cloud data centers	Layered architecture with thermal-aware scheduling middleware and distributed cluster monitoring	Improved temperature regulation and decreased power consumption	Scalability across data centers, middleware integration	Develop intelligent thermal prediction modules and inter-data center thermal coordination
Pourmohseni et al. (2022)	Migration of thermally-aware tasks in diverse many-core architectures	DVFS-based analytical power-budgeting with proactive task migration	Maximized safe thermal operation and system performance	Overhead in continuous migration and prediction	Integrate AI-based decision models for predictive core activation and migration planning
Lee (2021)	Thermal-aware scheduling in embedded systems with real-time constraints	Dynamic thermal management in real-time and thermally aware utilization-bound	Enhanced system utilization by 18.2% while meeting thermal restrictions for the associated chips	Dependence on cooling environment and dynamic workloads	Apply to safety-critical systems like automotive ECUs with formal guarantees

VI. Conclusion And Future Work

This study of temperature-aware task scheduling strategies in mobile and embedded systems emphasizes the critical interplay between energy efficiency and thermal management. The paper outlined key scheduling models, including dynamic and static approaches, proactive versus reactive strategies, and hardware-assisted methods such as DVFS and thermal sensing. Recent advances demonstrate how algorithmic and architectural innovations help mitigate thermal hotspots, prolong device lifespan, and maintain optimal system performance. While significant progress has been made, particularly in multi-core and heterogeneous processor platforms, challenges persist in scalability, real-time responsiveness, and platform-specific optimizations. Integrating task scheduling with advanced thermal modeling tools, such as Hotspot, further enhances the effectiveness of thermal-aware design strategies. However, most existing approaches are tailored to specific platforms and lack generalizability across diverse hardware configurations.

Future research should focus on developing adaptive and predictive scheduling algorithms that utilize artificial intelligence to anticipate thermal events in real-time. A promising direction lies in integrating machine learning models with hardware-level thermal feedback to enable self-optimizing systems. Furthermore, expanding the scope of temperature-aware scheduling to include emerging platforms such as edge AI devices, wearable computing, and automotive embedded systems can ensure broader applicability. Addressing the limitations of current cooling mechanisms and incorporating robust energy models for next-generation batteries will be essential. Ultimately, achieving a synergistic balance between energy, performance, and thermal behavior remains a key challenge for designing sustainable and intelligent mobile systems.

VII. References

[1] V. Panchal, “Energy-Efficient Core Design for Mobile Processors: Balancing Power and Performance,” Int. Res. J. Eng. Technol., vol. 11, no. 12, pp. 1–11, 2024.

[2] E. Bampis, D. Letsios, G. Lucarelli, E. Markakis, and I. Milis, “On multiprocessor temperature-aware scheduling problems,” J. Sched., vol. 16, no. 5, pp. 529–538, 2013, doi: 10.1007/s10951-013-0319-z.

[3] C. Rema, P. Costa, M. Silva, and E. J. S. Pires, “Task Scheduling with Mobile Robots—A Systematic Literature Review,” Robotics, vol. 14, no. 6, p. 75, 2025, doi: 10.3390/robotics14060075.

[4] J. Zidar, T. Matić, I. Aleksi, and Ž. Hocenski, “Dynamic Voltage and Frequency Scaling as a Method for Reducing Energy Consumption in Ultra-Low-

Power Embedded Systems," Electronics, vol. 13, no. 5, p. 826, Feb. 2024, doi: 10.3390/electronics13050826.

[5] S. Gupta and A. Mathur, "Enhanced Flooding Scheme for AODV Routing Protocol in Mobile Ad Hoc Networks," in 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies, IEEE, Jan. 2014, pp. 316–321. doi: 10.1109/ICESC.2014.60.

[6] S. Shelare, K. Aglawe, M. Dhande, S. Wagmare, M. Giripunje, and P. Sirsat, "Battery Thermal Management System: A Review on Recent Progress, Challenges and Limitations," MATEC Web Conf., vol. 405, p. 02004, 2024, doi: 10.1051/matecconf/202440502004.

[7] Z. Tang, S. Guo, P. Li, T. Miyazaki, H. Jin, and X. Liao, "Energy-Efficient Transmission Scheduling in Mobile Phones Using Machine Learning and Participatory Sensing," IEEE Trans. Veh. Technol., vol. 64, no. 7, pp. 3167–3176, 2015, doi: 10.1109/TVT.2014.2350510.

[8] R. Patel, "Remote Troubleshooting Techniques for Hardware and Control Software Systems: Challenges and Solutions," Int. J. Res. Anal. Rev., vol. 11, no. 2, pp. 933–939, 2024, doi: 10.56975/ijrar.v11i2.311510.

[9] M. Chrobak, C. Dürr, M. Hurand, and J. Robert, "Algorithms for temperature-aware task scheduling in microprocessor systems," Sustain. Comput. Informatics Syst., vol. 1, no. 3, pp. 241–247, 2011, doi: 10.1016/j.suscom.2011.05.009.

[10] A. Hassan, "Thermal and Energy Efficiency Management of Mobile MPSoC: A Review," Arid. J. Basic Appl. Res., vol. 4, no. 1, 2024, doi: 10.55639/607.414039.

[11] V. Panchal, "Mobile SoC Power Optimization : Redefining Performance with Machine Learning Techniques," IJIRSET, vol. 13, no. 12, pp. 1–17, 2024, doi: 10.15680/IJIRSET.2024.1312117.

[12] J. Cheng, L. Josipovic, G. A. Constantinides, P. Ienne, and J. Wickerson, "DASS: Combining Dynamic Static Scheduling in High-Level Synthesis," IEEE Trans. Comput. Des. Integr. Circuits Syst., vol. 41, no. 3, pp. 628–641, 2022, doi: 10.1109/TCAD.2021.3065902.

[13] M. İ. Ulucak and H. Gökçen, "Dynamic Scheduling in Identical Parallel-Machine Environments: A Multi-Purpose Intelligent Utility Approach," Appl. Sci., vol. 15, no. 5, 2025, doi: 10.3390/app15052483.

[14] M. Ben Ahmed and A. A. Boudhir, Innovations in Smart Cities and Applications, vol. 37, no. January. in Lecture Notes in Networks and Systems, vol. 37. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-74500-8.

[15] Y. W. Yang and K. S. M. Li, "Temperature-aware dynamic frequency and voltage scaling for reliability and yield enhancement," Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC, pp. 49–54, 2009, doi: 10.1109/ASPDAC.2009.4796440.

[16] K. Gupta and M. Singh, "Heuristic-based task scheduling in grid," Int. J. Eng. Technol., vol. 4, no. 4, pp. 254–260, 2012.

[17] M. N. Shehzad et al., "Thermal-aware resource allocation in earliest deadline first using fluid scheduling," Int. J. Distrib. Sens. Networks, 2019, doi: 10.1177/1550147719834417.

[18] İ. Çetin, E. Sezici, M. Karabulut, E. Avci, and F. Polat, "A comprehensive review of battery thermal management systems for electric vehicles," Proc. Inst. Mech. Eng. Part E J. Process Mech. Eng., vol. 237, no. 3, pp. 989–1004, Jun. 2023, doi: 10.1177/09544089221123975.

[19] R. Patel, "Advancements in Data Center Engineering : Optimizing Thermal Management , HVAC Systems , and Structural Reliability," Int. J. Res. Anal. Rev., vol. 8, no. 2, pp. 991–996, 2021.

[20] O. L. Abraham, M. Asri Bin Ngadi, J. Bin Mohamad Sharif, and M. Kufaisal Mohd Sidik, "Task Scheduling in Cloud Environment–Techniques, Applications, and Tools: A Systematic Literature Review," IEEE Access, vol. 12, no. October, pp. 138252–138279, 2024, doi: 10.1109/ACCESS.2024.3466529.

[21] M. Murugan et al., "A Comprehensive Review of Thermal Management Methods and Ideal System Design for Improved Electric Vehicle Battery Pack Performance and Safety," Energy Sci. Eng., vol. 13, no. 3, pp. 1011–1036, 2025, doi: 10.1002/ese3.2081.

[22] T. A. N. T. Perera, T. M. D. Nayanajith, G. Y. Jayasinghe, and H. D. S. Premasiri, "Identification of thermal hotspots through heat index determination and urban heat island mitigation using ENVImet numerical micro climate model," Model. Earth Syst. Environ., 2022, doi: 10.1007/s40808-021-01091-x.

[23] S. B. Kasturi, S. S. H. Raju, N. Srikanth, K. Anusha, M. Revathi, and S. K. Medishetti, "EOA: Energy and Temperature Aware Scheduling in Cloud-Fog Computing Environment," in 2025 5th International Conference on Pervasive Computing and Social Networking (ICPCSN), 2025, pp. 112–119. doi: 10.1109/ICPCSN65854.2025.11035542.

[24] W. Li, H. Li, L. Yang, L. Qiao, and L. Shi, "MTPS: A Multi-Task Perceiving and Scheduling Framework Across Multiple Mobile Devices," IEEE Trans. Mob. Comput., vol. 23, no. 12, pp. 15048–15061, Dec. 2024, doi: 10.1109/TMC.2024.3450577.

[25] C. Peng, L. Mo, J. Liu, and D. Niu, "Thermal-Aware Approximate Real-Time Task Scheduling for Multicore Embedded Systems," in 2023 China Automation Congress (CAC), 2023, pp. 6439–6444. doi: 10.1109/CAC59555.2023.10450488.

[26] H. Wang, L. Cai, X. Hao, J. Ren, and Y. Ma, "ETS-TEE: An Energy-Efficient Task Scheduling Strategy in a Mobile Trusted Computing Environment," Tsinghua Sci. Technol., vol. 28, no. 1, pp. 105–116, 2023, doi: 10.26599/TST.2021.9010088.

[27] J. Nong, J. Chen, Y. Wang, W. Qin, and X. He, "Thermal-aware Energy Efficient Task Scheduling Framework," in 2022 IEEE 21st International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS), 2022, pp. 304–309. doi: 10.1109/IUCC-CIT-DSCI-SmartCNS57392.2022.00055.

[28] B. Pourmohseni, S. Wildermann, F. Smirnov, P. E. Meyer, and J. Teich, "Task Migration Policy for Thermal-Aware Dynamic Performance Optimization in Many-Core Systems," IEEE Access, vol. 10, pp.

33787–33802, 2022, doi: 10.1109/ACCESS.2022.3162617.

[29] Y. Lee, "Thermal-Aware Design and Management of Embedded Real-Time Systems," in 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021, pp. 1252–1255. doi: 10.23919/DATE51398.2021.9474042.