Exploiting big.LITTLE Batteries for Software Defined Management on Mobile Devices

Zichen Xu, *Member, IEEE*, Jie Zhou, Wenli Zheng, *Member, IEEE*, Yuhao Wang, *Senior Member, IEEE*, and Minyi Guo, *Fellow, IEEE*

Abstract—Battery service time is a critical constraint on the availability and functionality of mobile devices. Equipping larger batteries may mitigate such deficiency yet it raises the challenges on thermal limit and physical size. Changing the battery chemistry is another solution, which however usually benefits the energy efficiency of only part of the applications, depending on their software behaviors. To address the challenges of battery energy efficiency and heat dissipation in limited physical space, we propose CAPMAN , a management framework that jointly optimizes the *c*ooling and *a*ctive *p*ower *man*agement in a smartphone, a typical mobile device, equipped with a hybrid battery pack. We establish the framework with three components. First, we abstract the correlation among the batteries, devices and software into a finite state machine model, whose state transitions can be triggered by actions like system calls and user activities. Second, we propose a battery scheduling algorithm that determines the more suitable battery for cooling/active power use, with respect to the dynamic software behaviors and their impact on the hardware states, based on a Markov decision process (MDP). Third, we design a facility for joint cooling and active power management by coordinating TECs and batteries. With the three major designs, CAPMAN realizes software defined management that schedules heterogeneous batteries and TEC cooling in a timely manner. In addition, CAPMAN provides an online algorithm with a proved O(1.05)-competitiveness performance. With a pair of big.LITTLE batteries, we prototype CAPMAN on multiple popular smartphones and a PYNQ development board. The evaluation with real-world workloads shows that compared to the current mainstream, CAPMAN can achieve 114 percent longer battery service time under skewed loads; compared to the state-of-the-practice baselines, CAPMAN shows 55 percent performance gain and 53 percent less energy use on average. Those results approve that big.LITTLE batteries with sophisticated software defined management is an effective way to prolong the battery service times on mobile devices.

Index Terms—Software-defined battery, battery scheduling, MDP-based approximation, power/performance evaluation

1 Introduction

Running out of battery is a common experience for the users of many mobile devices. While a variety of Apps with more popular and rich functions are developed every year, they are also becoming more resource- and power-consuming. For example, it is reported that Instagram accounts for 30 percent of active battery consumption [46]. High power consumption not only drains the battery quickly, but also easily overheats the device [18]. Therefore, cooling and power management is essential for the design of mobile systems to improve user experience.

To address the cooling and power challenge, researchers have done enormous work on both hardware and software. The hardware solutions attempt to equip mobile devices with larger battery packs and auxiliary facilities, such as battery banks [12], phase-change material devices [51], and surface cooling fans [53]. However, those hardware facilities degrade user experience due to the huge increases in terms of volume in size and carry-on weight. Additionally, active cooling methods usually solve the thermal problem with higher power consumption. On the other hand, the software solutions, such as fetch toggling [59], dynamic frequency and voltage scaling (DVFS) [34], load migration [25], and chip-level thermal shutdown [16], usually throttle/halt the workload processing of some active devices to save power consumption. Hence those solutions may not work for interactive or streaming Apps (e.g., games or video players on smartphones), as they cannot tolerate significant performance degradation and still yield to fast battery drain and high temperature. Up till now, it is still challenging to reduce the power consumption and temperature of mobile devices without user experience degradation.

Recently, a hybrid energy system proposed by Microsoft and Tesla, explores the idea of software-defined battery (SDB), which exploits heterogeneous battery packs to support various software patterns with high energy efficiency. Our research develops this idea and asks: if the battery permits, *how could our system saves the power consumption of a mobile device by adapting heterogeneous batteries to diverse software behaviors?* Saving power consumption would also reduce the heat dissipation and lower the demand of active cooling.

- *Zichen Xu is with the College of Computer Science and Technology, Nanchang University, Nanchang, Jiangxi 330031, China. E-mail: xuz@ncu.edu.cn.*
- *Jie Zhou is with the Department of Computer Science and Engieering, Tongji University, Shanghai 200092, China. E-mail: JieZhou0805@outlook.com.*
- *Wenli Zheng and Minyi Guo are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: zhengwenli@sjtu.edu.cn, guo-my@cs.sjtu.edu.cn.*
- *Yuhao Wang is with the Department of Electrical and Computer Engineering, Nanchang University, Nanchang, Jiangxi 330031, China. E-mail: wangyuhao@ncu.edu.cn.*

Manuscript received 29 June 2020; revised 18 Oct. 2020; accepted 21 Oct. 2020. Date of publication 2 Nov. 2020; date of current version 5 May 2022. (Corresponding author: Wenli Zheng.) Digital Object Identifier no. 10.1109/TMC.2020.3035236

We begin exploring the answer to this question with adapting the SDB definition [10] on the chemistry of different batteries. We first study different software behaviors to find their power characteristics, which indicate their preferences about battery chemistries and cooling techniques. As such, we obtain three important observations as follows: (1) Software performance and user experience can be significantly influenced by the battery chemistry. For example, one battery chemistry can achieve 37.6 percent longer service time than another when running PCMark [5]. Different battery properties, like energy density and discharge rate, compromise with each other, which exhibits the potential to jointly optimize the power efficiency by selecting the most appropriate battery for each specific software behavior. However, due to the fast variation of software behaviors, frequently switching batteries may cause additional energy loss and heat dissipation. (2) Battery properties, software behaviors, and correlated hardware states (like device power states), can be modeled into combinatorial states in power management [41]. The actions, such as system calls and user activities, transit these states from one to another. This motivates us to use a graph to formulate the problem. (3) Resource-intensive Apps can cause thermal problems like hot spots (i.e., surface temperature that exceeds 45–[55]) and hence call for active cooling, which increases the active power consumption.

Based on the three observations, in this paper, we propose *CAPMAN*, a **c**ooling and **a**ctive **p**ower **man**agement framework, which models the demand of power and cooling, and jointly optimizes the use of heterogeneous batteries and active cooling. CAPMAN works on a mobile device supported by the one-pair battery design that we propose, i.e., big.LITTLE batteries, with which the power management problem becomes categorizing diverse software behaviors into using either the big or LITTLE battery, for optimized battery energy efficiency.

The concept of big.LITTLE follows the heterogeneous design, as the big.LITTLE core from ARM [2]. We apply this idea to the battery design in order to exploit the different advantages of each battery chemistry. While the ARM's technology meets the power demand of CPUs, big.LITTLE batteries meet the battery energy consumption given the power demand of all the device components, minimizing the energy loss caused by the variation of power demand. Hence the two technologies can cooperate orthogonally.

We establish CAPMAN with three major designs. First, based on the hierarchical structure of a mobile system that consists of the battery layer, device layer and software layer, we abstract the correlation among the different layers into a finite state machine model, whose state transitions can be triggered by actions like system calls and user activities.

Second, we propose a battery scheduling algorithm that determines the more suitable battery for cooling/active power use, with respect to the dynamic software behaviors and their impact on the hardware states, based on Markov decision process (MDP). Since a modern smartphone can have hundreds of Apps, tens of devices, and two batteries, the number of states explodes in time-series analysis. We derive a structural similarity approximation method to accelerate the convergence of MDP computation and the decision making, proven to achieve a worst case $O(1/(1-\rho))$-competitiveness performance, where ρ is a controlled discount factor. When we allow 5 percent similarity approximation relax, the algorithm in CAPMAN is O(1.05)-competitiveness.

Third, we design a management facility, both hardware and software, to coordinate the thermoelectric coolers (TECs) [11] and batteries. A TEC is an efficient cooling device yields to runtime active power consumption, while this active power burst can be addressed by our big.LITTLE batteries and predicted by our Markov model.

With the three major designs, CAPMAN realizes software defined management that schedules big.LITTLE batteries and TEC cooling on mobile devices in a timely manner.

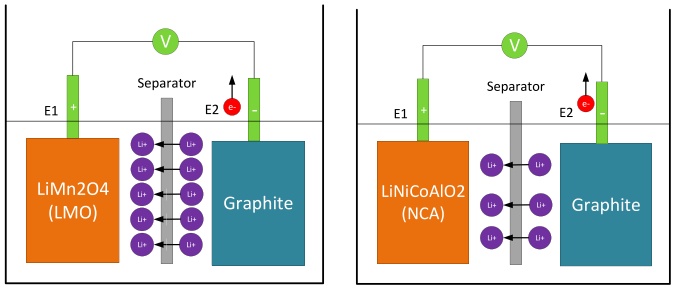
We have implemented a prototype of CAPMAN on multiple mainstream smartphones and a PYNQ development board, on which we conduct experiments with real-world traces and workloads. Results show that compared to the current mainstream, CAPMAN can achieve 114 percent longer battery service time while maintaining the ambient temperature, even under skewed loads; compared to state-of-the-practice baselines, CAPMAN shows 55.08 percent performance gain and 53.27 percent less energy use on average. Our prototype design introduces minimal extra weight (less than 5 gram) and negligible volume overhead, and the extra devices can fit into a typical smartphone with proper implementation. The major contributions are as follows:

- We study the characteristics of multiple newly proposed battery chemistries and active cooling techniques, and motivate the research on cooling and active power management of mobile devices for improved software performance and user experience.
- We integrate the idea of big and LITTLE batteries, which can significantly prolong the battery service time but cause hot spots, with active cooling techniques, which can address the hot spots gracefully.
- We propose CAPMAN and prove its online scheduling scheme can help to achieve a worst-case $O(1/(1-\rho))$-competitiveness performance.
- We evaluate CAPMAN with real-world workloads, and show it can double the battery service times with the same battery capacity.

The rest of this paper is organized as follows. Section 2 highlights the background and motivation about a heterogeneous battery design. Section 3 describes CAPMAN and proves that it retains the performance guarantee. Section 4 describes how to implement a prototype of CAPMAN. Section 5 empirically studies the performance of CAPMAN. Section 6 provides a brief on the related work. At last, Section 7 concludes the paper.

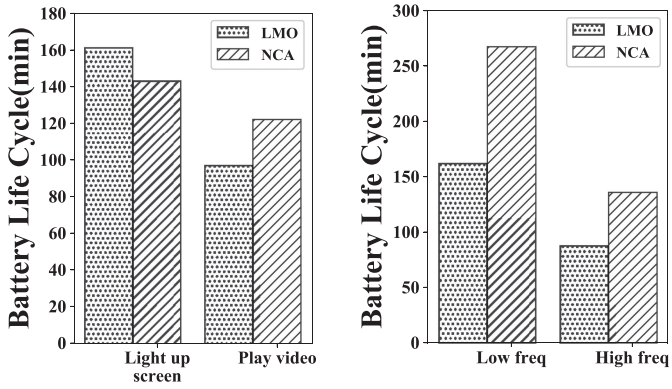
2 Background and Motivation

Both the hardware and software of mobile devices have rapidly developed in the last decade. Typically, a modern smartphone usually runs a variety of applications with a powerful CPU. Consequently, the power consumption and heat dissipation also rise sharply. In the past, a fully charged phone can sustain two or three days, and now it has to be charged two or three times a day [44], since smartphone manufacturing limits the size of available batteries. In order to at least mitigate the power constraint, previous



(a) $LiMn_2O_4$ (LMO) (b) $LiNiCoAlO_2$ (NCA)

Fig. 1. LMO (a) and NCA (b) batteries behave significantly different in releasing electrons, or power supply.



(a) Behaviors (b) Frequencies

Fig. 2. Different user behaviors (a) and frequencies of switching phone on/off (b) may favor different battery chemistries in a Nexus 6.

efforts focus on smart utilization of batteries, using various control, optimization, and scheduling techniques [17]. As aforementioned in Section 1, recent studies suggests different batteries for different software patterns, invoking battery pack design composed of heterogeneous battery chemistries. The battery properties is significantly affected by the electrode materials. Fig. 1 shows that under the same conditions, The $LiMn_2O_4$ (LMO) battery has more electrons to exchange than the $LiNiCoAlO_2$ (NCA) battery during the same period of time. In other words, the discharge rate of the LMO battery can be much higher than that of the NCA battery.

In this paper, we refer to a battery with a high energy density as a *big* battery, which usually comes with a low discharge rate (like NCA), and a battery with a small energy density as a *LITTLE* battery, which usually comes with a high discharge rate (like LMO).

To meet specific software requirements with the more proper battery, in this section, we deeply study the correlation between battery chemistries, system power consumption, and software behaviors. As such, the study enables our system modeling to accurately profile and control the power and performance in the design of CAPMAN for big. LITTLE battery powered systems.

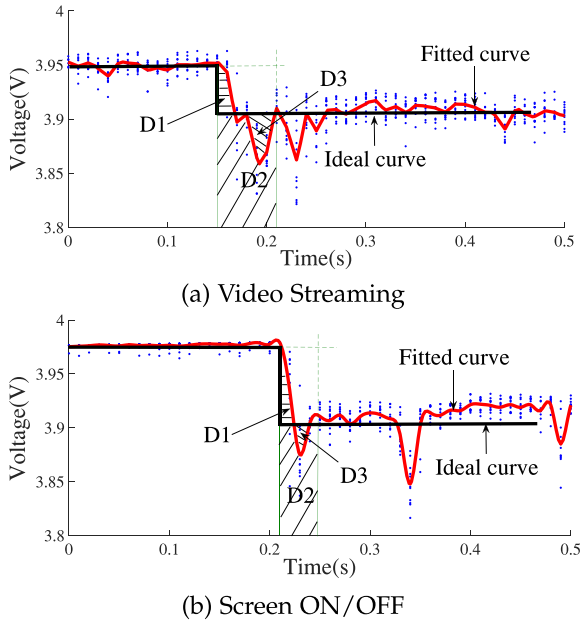
2.1 Battery Chemistries and User Behaviors

The first step of our work is to understand the power saving potential from using heterogeneous batteries. We test a Nexus 6 phone with Android 5.0.1, collect its battery discharge cycle, and repeat the same tests with different batteries.[1] Fig. 2 shows that using two battery chemistries (LMO and NCA) with the same capacity (2500mAh) leads to significantly different discharge cycles, in testing scenarios like lighting up the screen, playing video, or keeping turning on and off the phone at different frequencies.

Fig. 2a shows that when keeping the phone on and idle, using the LMO battery can sustain 14.3 percent longer than using the NCA battery. If the phone plays some videos, the result reverses. The NCA battery can outperform the LMO battery with 24 percent longer service time. One explanation is these simple behaviors expose different power demand patterns that favor different battery discharge features. Fig. 2b shows that when repeatedly turning the phone on and then off, the NCA battery is always better at handling this short burst of power demand, prolonging the discharge cycle time. However, when the frequency of such a behavior increases, from every minute (low) to every second (high), the advantage of using the NCA battery decreases, from 46 percent longer service time to only 35 percent. As such, the nonlinear changes of battery service time performing various behaviors supported by different batteries motivate to model the power demand surge and its frequency, with a consideration of battery properties.

2.2 Active Power Savings

A common practice in electric vehicles (EVs) is to schedule the right battery to use when the load changes, while a similar method could be also beneficial on mobile devices. Xu *et al.*[60] discover the V-edge phenomenon, i.e., when a new power demand arrives, the battery output voltage first quickly drops, and then rises up at a lower level than the initial voltage, leading to the so-called V-edge, as shown in Fig. 3. We further extend this direction on measuring



(a) Video Streaming

(b) Screen ON/OFF

Fig. 3. Power saving potentials from serving a heavy [(i.e., video streaming (a)) and light (i.e., Screen ON/OFF (b)] workload.

1. More detailed experimental setup can be found in Section 5.

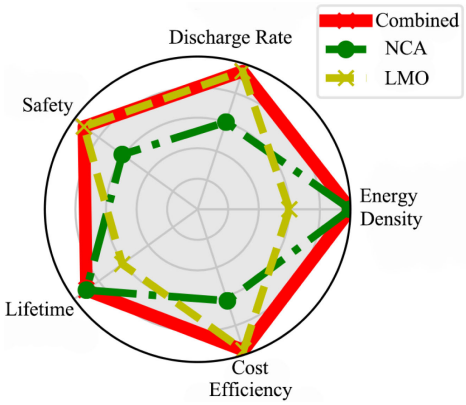


Fig. 4. Metrics comparison between popular smartphone batteries.

battery output voltage with multiple loads and batteries, using an Agilent 34410A multi-meter [1].

Figs. 3a and 3b illustrate the voltage drops when playing the video and turning on/off the screen, respectively. The blue dots are the collected voltage samples, the red curves are the fitted curves, and the black lines present the ideal cases.

After the output voltage settles down, the ideally estimated power leak consumption is $D2 + D3$ while the actual consumption is $D1 + D2$. The area $|D3 - D1|$ is the potential power saving we seek. Thus we have the general principles: if this V-edge phenomenon happens frequently, we are looking for one battery chemistry that minimizes $D1$, i.e., the LITTLE battery; if the settling time is long, we need a different battery that maximizes $D3$, i.e., the big battery.

Hence software, including applications and user behaviors, using different batteries leads to different discharge time. If we can choose the most suitable battery according to the software, then the longest battery discharge time can be achieved, thereby improving the user experience. Therefore, the battery problems must be addressed at the system level.

2.3 Battery Features at Different Dimensions

To comprehensively characterize the big and LITTLE batteries, we perform literature study on currently available Li-ion battery features for smartphone power supply [3]. Here we provide a radar map of popular smartphone batteries, such as NCA and LMO, on important features, with normalized data shown in Fig. 4. We obtain two important observations from this map. First, no single battery chemistry provides the optimal coverage in all the five dimensions, i.e., discharge rate, energy density, cost, lifetime, and safety. However, combining various batteries can help to achieve this goal. Second, a fully mixed battery pack is complex to schedule yet hard to reason the optimal scheduling solution, as proved in [8]. In this study, we mainly focus on the power savings in one discharge cycle, and thus without loss of generality, we pick two batteries that perform almost orthogonal in important features, such as discharge rate and energy density, to suit our battery scheduling to software behaviors. In this paper, we call it big.LITTLE battery.

2.4 Active Power Management With Cooling

Since powering and cooling commonly affect each other, adapting a big.LITTLE battery design can also complicate

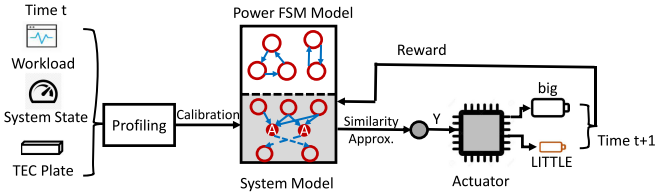


Fig. 5. The CAPMAN framework. Shaded part is CAPMAN's contributions.

the smartphone cooling. One issue is that frequent battery switch and fast voltage drop raise temperature on some spots of a mobile device. The traditional cooling method in a smartphone is a cooling plate that dissipates heat evenly and slowly, which is insufficient to prevent hot spots from leading to unexpected additional energy loss and heat dissipation. To address this problem, we introduce an active cooling technique, i.e., Thermoelectric Cooling (TEC), a promising heat sink, into the design of our management framework.

Recently, TEC has been used for electronic cooling [32], with the advantages on size, quietness, and high reliability. It is widely used in aerospace and other industrial products for different cooling purposes [45]. A typical TEC works with two p-type and n-type semiconductor beadings electrically connected. When a direct current passes through a TEC, the conductor drains heat from the cold side of the TEC to the hot side [15]. TEC can reduce the temperature fast and to the spot, with an expense on an active power surge. This power surge shall also be considered and supported in the big.LITTLE battery scheduling design. Thus, it motivates us to design CAPMAN, to jointly optimize cooling and active power management for big.LITTLE battery powered systems.

3 CAPMAN Design

CAPMAN schedules big.LITTLE batteries as a cooling and active power management framework, in order to prolong the service time that suits software demand. It supports system modeling on power consumption, profiles the runtime cooling and active power cost, and controls big and LITTLE batteries. CAPMAN targets software with these features:

- Software is accessed frequently enough to invoke transition between device power states;
- User interaction can turn on and off a phone frequently. Yet this leaking power surge is stable when the same operation happens;
- On the scale of one discharge cycle, i.e., duration between two device charges, the arrivals of software demands are frequent with a skewed distribution.

Fig. 5 highlights our overall design of CAPMAN. CAPMAN collects runtime workload, TEC, and system statistics for the whole system power profiles. The device power modeling is an extensively studied area [14]. CAPMAN adapts the finite-state machine model (C. Hu et al [41]), and treats all power profiles as metrics, depending on related power states. Using the power profile, the scheduling decision process is formulated as a Markov decision process (MDP). Based on the calculated maximum likelihood from

TABLE 1
Battery Model

Battery	Cost Efficiency	Lifetime	Discharge Rate	Energy Density	Result
$LiCoO_2$(LCO)	⋆⋆	⋆⋆⋆	⋆⋆	⋆⋆⋆⋆⋆	big
$LiNiCoAlO_2$(NCA)	⋆⋆⋆	⋆	⋆⋆⋆	⋆⋆⋆⋆⋆	big
$LiMn_2O_4$(LMO)	⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆	LITTLE
$LiNiMnCoO_2$(NMC)	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	LITTLE
$LiFePO_4$(LFP)	⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆	LITTLE
$LiTi_5O_{12}$(LTO)	⋆	⋆⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆	LITTLE

the MDP, an algorithm can enable CAPMAN to extract the right decision. However, we prove that the algorithm is complex in time that may not provide the right battery decision on time. As such, we further develop an online approximation algorithm based on MDP similarity. We prove the theoretical bound of our online algorithm, which outputs the battery selection onto CAPMAN implementation. CAPMAN can switch between batteries in milliseconds. The subsequent sections discuss system modeling, algorithms, and actuator design in details.

3.1 System Modeling

CAPMAN operates the battery and cooling modules of a smartphone system for cooling and active power management, and the decisions are made based on relevant models.

Battery Model. Due to the heterogeneity in energy storage capacity, instantaneous power discharge capacity, etc., some batteries are suitable for power demands with unique characteristics. We investigate six types of widely used lithium batteries and summarize their major properties in Table 1. We can find the two properties energy density and discharge rate, determine the energy storage capacity and instantaneous power discharge capacity, and they are anti-correlation. A battery with higher energy density can store more energy given the same volume, but discharge less electricity instantaneously. Such a battery is more suitable for the scenarios like playing a video, whose discharge time is long but gently changes. In contrast, a battery with a large discharge rate is preferred when the discharge power changes dramatically, e.g., when an application is launched, or a user lights up an inactive phone.

Based on the two properties, we classify those batteries into two categories: the batteries with high energy density as big batteries, and those with large discharge rates as LITTLE batteries, which is shown in Table 1. Without loss of generality, we select $LiMn_2O_4$(LMO) and $LiNiCoAlO_2$(NCA) as the LITTLE battery and big battery in our setup, respectively.

Cooling Model. The surface temperature of a phone, as part of the user experience, deserves to be considered. Hence, CAPMAN should include a cooling model estimating the surface temperature based on power consumption. We show a typical temperature distribution across the main components in a smartphone (the top row in Fig. 6). For the best cooling effect, TECs are placed upon the CPU, to cool the hottest component in the smartphone.

We use a cooling model [19] to describe how a TEC works, which is related to the thermoelectric coefficient S_T, resistance R and thermal conductivity K. The heat Q_c transferred through a TEC can be calculated by Equation (1),

$$Q_c = S_T T_c I - \frac{1}{2} I^2 R - K(T_h - T_c), \tag{1}$$

where I is the operating current in Ampere, and T_h and T_c are the temperatures on the hot and cold sides of the TEC, respectively, in Kelvin. Equation (1) illustrates that the heat dissipation rate of a TEC is not simply proportional to its operating current. This is also demonstrated by the bottom of Fig. 6, which shows the relationship between the cooling efficiency of a TEC and the operating current. Here, the cooling efficiency is calculated by the temperature difference between the two sides of a TEC. As the operating current increases, the cooling efficiency gradually increases at first, reaches the maximum when the operating current is around 1.0A (i.e., its rated operating current), and then gradually decreases. Therefore, for the best cooling efficiency, we propose to maintain the TEC at its rated operating current. In the next part, we build power models of TEC, CPU, screen and WiFi, and analyze the power consumption of each one.

Power Model. Previous work has conducted extensive research on the power modeling of smartphone components, especially those that consume major energy. Thus to build the active power model for CAPMAN, we adopt the commonly used power models for CPU, screen and WIFI, and integrate them with the power model of TEC. The four power models are listed in Table 2. The CPU power consumption is linearly related to its utilization given a specific frequency level [52]. The screen power consumption depends on the brightness [9]. In Table 2, α_b, α_w, and C_{Screen} are power coefficients, which respectively represent power

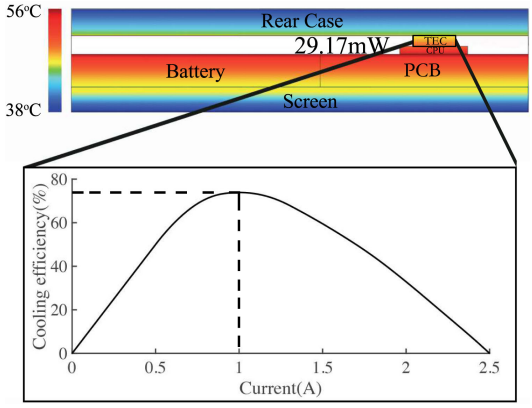


Fig. 6. Top: Temperature distribution in mobile phones, with red for high temperature and blue for normal temperature. The thickness of TEC is generally 2mm. For the convenience of labeling, we have enlarged it. Bottom: Relationship between TEC heat dissipation and its operating current.

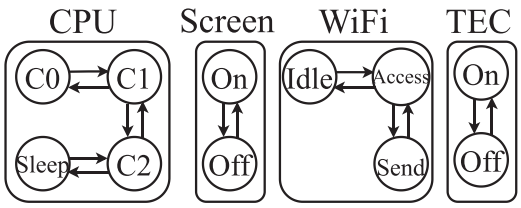


Fig. 7. Hardware status and state transitions in CAPMAN.

consumption as a function of the brightness level of black and white pixels. Here we take the average of the power consumed by the white and black pixels as an approximate power consumption for each pixel. The parameter B_{level} indicates the brightness level of the mobile phone interface, ranging from 0 to 255. The power consumption of WIFI is piece-wise linearly related to the packet rate [62]. The TEC power can be derived based on its operating current I [19], which can be obtained by Equation (1).

3.2 Finite-State Power Machine in CAPMAN

Traditional power modeling in computing devices does not directly translate into a correct battery management decision. With the given states and transitions in-between (Fig. 7), we may not be able to mark which device state can be translated into which battery (i.e., big or LITTLE) to use. The connections between software behavior and hardware demand, as well as hardware power states and battery type are the key design factors in CAPMAN.

Software to Hardware. Every click or touch in the human-software interaction leads to some unique responses in the software, and eventually in the hardware state changes. As aforementioned, all user behavior can be classified into two categories, in a coarse-grained manner. The first type is a constant activation of an App or a series of Apps, such as watching videos. This leads to a stably predictable sequence of hardware use. With the help of our system modeling in Table 2, we can easily find which device stays in what state, and calculate the power demand accordingly. Another typical behavior is the intermittent activation of the phone, as mentioned in [43]. In this case, user occasionally checks his/her phone, for emails or short messages, and closes the phone within seconds. At the same time, hardware devices are turned on and off in a relatively short period of time, causing bursty power demand to local energy storage. This behavior, accounts for 65 percent of battery use, as recorded in [48]. This is usually missed in power-aware design literature yet essential for an efficient battery management design. Our power model can capture this change from tensors between nodes in the Fig. 7.

Hardware to Battery. It is very difficult to label which battery to use associate with different power state from different hardware. Unlike most previous work in power modeling for battery-powered devices, our work focuses on state transitions and their frequency instead of which state the device stays on. It is because the battery behavior responses to sudden power demand changes, instead of a constant power demand. Thus, one unique design in CAPMAN is *CAPMAN models variation in the power demand distribution more than the actual power value*. There are still two challenges in making a right battery decision in practice. (1) Most hardware state reads from operating system are at the second level while a plausible battery switch shall be done in at least millisecond scale. The incorrect granularity mapping from a second level to millisecond level could prevent an efficient battery management design at the system level. (2) Power state automata only omit the current power demand. We need to build a prediction mechanism using this automata to guide the correct use of batteries. To address these two challenges from using power automata, we use a Markov process to model the decision process, a.k. a the battery scheduling as the correctness in set mapping between different granularity and predictability.

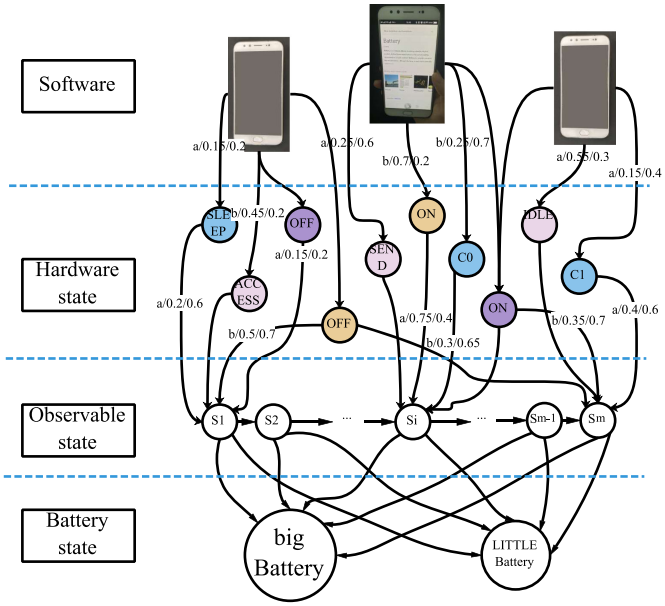


Fig. 8. The markov decision process in CAPMAN. The phone is waked up to receive a Wikipedia update.

3.3 Markov-Based Battery Scheduling

Fig. 8 presents our Markov representation of one-round battery scheduling. The hardware state layer reacts to the upper software demand changes, e.g., the screen on event that wakes the entire phone and begins to receive Internet data, in the form of a MDP model $\mathcal{M} = \{S, A, T, R\}$. S and A are the finite sets of states and actions, which in CAPMAN are the device power state vector in Fig. 7 and the system call vector [41], respectively. $T : S \times A \times S \rightarrow [0, 1]$ and $R : S \times A \times S \rightarrow [0, 1]$ are the *state transition function* and the *reward function*. In our system model, for example, when the phone wakes, the CPU state turns from sleep to C0 and the screen is switched from off to on, CAPMAN switches the battery supply from big to LITTLE, in order to meet this short power surge. Our model presents it as $T(\{\text{SLEEP}, \text{OFF}, \ldots, \text{big}\}, a, \{C0, \text{ON}, \ldots, \text{LITTLE}\})$, which gives the probability that the phone is awake, and $R(\{\text{SLEEP}, \text{OFF}, \ldots, \text{big}\}, a, \{C0, \text{ON}, \ldots, \text{LITTLE}\})$ is the reward for taking such action a. Specifically, the reward is a function of a normalized variable in [0,1] and CAPMAN can compute the distribution with the mean represented as μ_i. Without loss of generality, in our current setup, these reward distributions are *i.i.d*.

One problem with the classic MDP representation is that it does not distinguish between actions that lead to a battery

switch decision and other internal transitions between the state nodes. In our case, we need to reduce the unnecessary state transitions between devices, so as that CAPMAN could improve its performance. To tackle this problem, we consider the following MDP graph representation.

The MDP graph for our $\mathcal{M} = \{S, A, T, R\}$ can be defined as a graph model $G_{\mathcal{M}} = \{\mathrm{V}, \Lambda, E, \psi, p, r\}$, which is a directed bipartite graph with the *state* nodes (V) and *action* nodes (Λ). We only generate this graph, in which the action node $v \in \Lambda$ connects two state nodes $u \in \mathrm{V}$ have different battery states. E represents a set of decision edges from state nodes to action nodes, and ψ is the set of transition edge from action nodes to state nodes, as the solid and dash lines in Fig. 5, respectively. We complete the graph transformation according to [27]. At the beginning of generating the correlated MDP graph, the decision edge is unweighted while transition edges are weighted by transition probability p and a reward r. It is clear that the designed MDP $\mathcal{M}$ corresponds with the $G_{\mathcal{M}}$ in a one-to-one relationship. Therefore, solving the $G_{\mathcal{M}}$ provides a unique solution to our original MDP problem.

3.4 Runtime Calibration

For battery scheduling, CAPMAN is able to find future battery states because of the MDP graph $G_{\mathcal{M}}$. CAPMAN computes the n^{th}-order optimality by searching and updating the MDP graph, and predicts the correct policy π to control the batteries. Classic solutions [30] show that the order of the polynomials could be large enough that the theoretically efficient algorithms are not efficient in practice, not to mentioned that our battery scheduling shall be done at circuit level with time granularity of micro to milliseconds. To simplify the search and solving the entire MDP graph, we propose to use a structural similarity method [54] that computes the similarity between MDP graph at different order such that the decision can be extract from history patterns without actual recompute the entire graph. Further, this computation works as an index for the decision process, that can be executed when the device is not busy at the background.

As mentioned above $G_{\mathcal{M}}$ is a bipartite, so state nodes and action nodes are always neighbor nodes to each other. The state similarity δ_{S} and the action similarity δ_{A} are defined first in Equation (2) before we present our structural similarity definition and runtime calibration algorithm:

$$\begin{aligned} \delta_{\mathrm{S}}(u, v) &\stackrel{\text{def}}{=} 1 - \sigma_{\mathrm{S}}(u, v), \forall u, v \in \mathrm{V}, \\ \delta_{\mathrm{A}}(a, b) &\stackrel{\text{def}}{=} 1 - \sigma_{\mathrm{A}}(a, b), \forall a, b \in \Lambda. \end{aligned} \tag{2}$$

State/Action Similarity Recursion. To define similarity, like SimRank [29], we have δ_S and δ_A from state nodes and action nodes. It is easy to understand that, if and only if their out-neighbors are similar, two nodes are similar. We repeat this process to find all similarities. On this basis, we define

$$\delta_{\mathrm{S}}(u, v) = \begin{cases} 0, & \text{if } u = v, \\ 1, & \text{if } u \text{ or } v \text{ (not both), is absorbing,} \\ d_{u,v}, & \text{if both } u \text{ and } v \text{ are absorbing.} \end{cases} \tag{3}$$

Here, a state is absorbing when the out-degree of the node is zero, which is the target state for battery scheduling in practice. Therefore, the configuration of $d_{u,v} \in [0, 1]$ is a description of the relationship between the target states depending on the application. $d_{u,v} \equiv 1$ and $d_{u,v} \equiv 0$ are two special cases, indicating that the two target states should be identified as completely different or the same, respectively. To compute state similarity σ_S and σ_A, we adopt distance computation using Hausdorff distance and earth mover's distance (EMD), which are suitable for graph calculation, to compute state similarity σ_S and σ_A as in Equation (4):

$$\begin{aligned} \sigma_{\mathrm{S}}(u, v) &= C_{\mathrm{S}} \cdot (1 - \delta_{\mathrm{Haus}}(N_u, N_v; \delta_{\mathrm{A}})), \\ \sigma_{\mathrm{A}}(a, b) &= 1 - (1 - C_{\mathrm{A}})\delta_{\mathrm{rwd}}(a, b) \\ &\quad - C_{\mathrm{A}}\delta_{\mathrm{EMD}}(p_a, p_b; \delta_{\mathrm{S}}), \end{aligned} \tag{4}$$

where $0 < C_S, C_A < 1$ is the parameter to measure the importance of the transition similarity and the reward similarity.

Algorithm 1. Structural Similarities Recursion

Input: MDP graph $G_M = (V, \Lambda, E, \Psi, p, r)$
Parameter: Discount factors $C_{\mathrm{A}} \in (0, 1)$
Output: Solution $(\sigma^*_{\mathrm{S}}, \sigma^*_{\mathrm{A}})$ to the recursion
// Initialization
1: $\boldsymbol{S} \leftarrow \boldsymbol{I}_{|V|\times|V|}, \boldsymbol{A} \leftarrow \boldsymbol{I}_{|\boldsymbol{\Lambda}|\times|\boldsymbol{\Lambda}|}$
// Iterative computation.
2: **While** NOT S and A converge
3: **for** all $a \in N_u$ and $b \in N_v (u, v \in V, u \neq v)$ **do**
4: $d \leftarrow \mathrm{EMD}(p_a, p_b; G_M, \mathbf{1} - \mathrm{S})$
5: Compute $\boldsymbol{A}_{a,b}$ with C_{A}, d and S
6: **for** all $u, v \in V$ with $N_u \neq \emptyset$ and $N_v \neq \emptyset$ **do**
7: Compute $\boldsymbol{S}_{u,v}$ with C_{S}, d and A
8: **return** $(\sigma^*_{\mathrm{S}}, \sigma^*_{\mathrm{A}}) \leftarrow (\boldsymbol{S}, \boldsymbol{A})$

3.5 Recursive Computation

The iterative algorithm for computing σ^*_{S} and σ^*_{A} is shown in algorithm 1 by repeating the recursion.

The algorithm computes similarity between states and actions to find the finalized pair, with which we can explore the battery decision. Note that in Line 4, Algorithm 1 calls for a the successive shortest path (SSP) algorithm [56] to compute the distance between two distributions, which is the EMD parameter.

Space and Time Complexity Analysis. Given the graph $G_{\mathcal{M}} = (V, \Lambda, E, \Psi, p, r)$ of the MDP $\mathcal{M} = (S, A, T, R)$, to store S and A, algorithm 1 requires $\Theta\left(|V|^2 + |\Lambda|^2\right) = O\left(|\mathrm{S}|^2|\mathrm{A}|^2\right)$ space SSP takes $O\left(K^2_{\max}\right)$ working memory, where $K_{max} \leq |V|$ is the maximum out-degree of action nodes in $G_{\mathcal{M}}$. In our case, our finite MDP has 50 state nodes and over 200 system calls recorded (i.e., cardinality in S and A, respectively). In our experiments, the memory footprint records no more than 400kB for this similarity exploration. Given a precision ϵ which is predefined (e.g., $\epsilon = 0.01$), SSP is guaranteed to end in $O\left(\frac{1}{\epsilon^2} \cdot \left(K^2_{\max} + K_{\max} \log K_{\max}\right)\right) = O(10^6)$, which is a constant in milliseconds. To further speed up the computation, we use Dijkstra's algorithm with a Fibonacci heap. Each iteration of Algorithm 1 (Lines 3-7) makes $\Theta(|\Lambda|^2)$ calls to SSP. It takes $\Theta(|V|^2 L^2_{\max})$ time to compute the Hausdorff distances, where $L_{\max} \leq |A|$ is the maximum

out-degree of state nodes in GM. Therefore, the overall time cost is $O\left(N \cdot |S|^2|A|^2 K_{\max}^2/\epsilon^2\right)$, which is linear to N, the number of iterations before convergence.

Uniqueness and Stability. By showing that Algorithm 1 always terminates, we prove that σ_{S}^* and σ_{A}^* are well-defined. We assume that after the k-th execution of Lines 3-7 of Algorithm 1 (k = 1, 2, ...), $S^{(k)}$ and $A^{(k)}$ are versions of the matrices **S** and **A**. Besides, let $S^{(0)}$ and $A^{(0)}$ be the contents of S and A right before the algorithm enters the main loop. It is easy to see that when the order k increases, the sizes of S^k and A^k is always less than S^{k+1} and A^{k+1}. Meanwhile, when introducing discount factor ($0 < C_S, C_A < 1$), we have $\mathrm{S}^i \in [0,1]$ and $\mathrm{A}^i \in [0,1]$ for all $i > 0$. Thus, the k increases to ∞, we have

$$\begin{aligned} \lim_{k\to\infty} S^{(k)} &= \sigma_{\mathrm{S}}^* \in [0,1], \\ \lim_{k\to\infty} A^{(k)} &= \sigma_{\mathrm{A}}^* \in [0,1]. \end{aligned} \tag{5}$$

Thus, Algorithm 1 always terminates correctly with the unique solution $\left(\sigma_{\mathrm{S}}^*, \sigma_{\mathrm{A}}^*\right)$.

Upper Bound. Given the graph $G_M = (V, \Lambda, E, \Psi, p, r)$ and an arbitrary initial state $u_0 \in V$, followed by a probabilistic policy $\pi : V \times \Lambda \to [0,1]$, there will be a state transitions track:

$$u_0 \xrightarrow[[r_1]]{a_0 = \pi(u_0)} u_1 \xrightarrow[[r_2]]{a_1 = \pi(u_1)} u_2 \xrightarrow[[r_3]]{a_2 = \pi(u_2)} \cdots$$

Given a discount factor $\rho \in (0,1)$, the state value of $u \in V$ under policy π, written $\mathcal{V}_u^\pi$, is the expected total accumulative return starting from the state node u, i.e.,

$$\mathcal{V}_u^\pi \overset{\text{def}}{=} \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \rho^k r_{k+1} | u_0 = u\right]. \tag{6}$$

Analogously, the action value of $a \in \Lambda$ under policy π is

$$\mathcal{P}_a^\pi \overset{\text{def}}{=} \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \rho^k r_{k+1} | a_0 = a\right]. \tag{7}$$

Then, the optimal value functions $\mathcal{V}^*$ and $\mathcal{Q}^*$ under the optimal policy π^* are considered.

The Bellman equations [50] state that

$$\mathcal{V}_u^* = \max_{a \in N_u} \mathcal{P}_a^*, \tag{8}$$

$$\mathcal{P}_a^* = \sum_{u \in N_a} p(a,u)\left(r(a,u) + \rho \mathcal{V}_u^*\right). \tag{9}$$

We show that σ_S^* and σ_A^*, the proposed distance measures, can be used to bound the difference between the optimal values. Therefore, we have:

$$\begin{aligned} \left|\mathcal{V}_u^* - \mathcal{V}_v^*\right| &\le \frac{1}{1-\rho} \cdot \delta_S^*(u,v), \\ \left|\mathcal{P}_a^* - \mathcal{P}_b^*\right| &\le \frac{1}{1-\rho} \cdot \delta_A^*(a,b). \end{aligned} \tag{10}$$

Let $C_S = 1$ and $C_A = \rho$, meaning two state nodes are diverged and transition similarity weight bounded for the competitiveness, since the reward function $r \in [0,1]$, $\sum_{k=0}^{\infty} \rho^k = \frac{1}{1-\rho}$. That is, we relax the similarity discount factor from empirical data, which sets $\rho = 0.05$, the upper bound of Algorithm 1 is within O(1.05)-competitiveness, compared to the optimal policy. This competitiveness guarantees the upper bound performance of battery scheduling in CAPMAN.

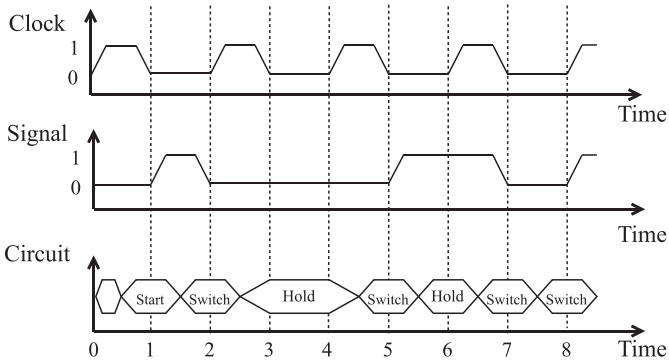


Fig. 9. Timing diagram for battery switching.

3.6 Actuator

Actuator converts the output of computed MDP state into battery selection decisions. In big.LITTLE batteries, the battery decision is a binary choice between switching from big to LITTLE and vice versa. For this implementation, we use a simple digital logic to control the switch using high/low voltage in TTL gates. Fig. 9 highlights a sample of our designed signal. The control process starts at time 1, where the voltage signal raises to the high level. Each voltage flip (e.g, $0 \to 1$ or $1 \to 0$) indicates a switch event. Otherwise, the system holds to the same battery. In Fig. 9, the battery switch flips at time 2, 5, 7, 8. Each flip can cause extra heat, which invokes TEC to cool the system down. As shown later in Section 5, CAPMAN actually favors LITTLE battery due to frequently wake TEC to cool the phone actively.

4 Implementation

In this section, we detailed our CAPMAN prototype in implementation, as illustrated in Fig. 10. We built the prototype to support two kinds of battery powered devices, a phone and a FPGA development board (i.e., PYNQ-Z2 [6]). The phone is used to illustrate CAPMAN in real world scenario. The FPGA board helps to understand what exactly happens in power consumption from each single device, affected by CAPMAN on hybrid batteries.

4.1 Profile and Monitor

The approximation in CAPMAN abstracts patterns, operations, and user interactions from the software level into a device power state machine, as described in our system model in Section 3.1. As illustrated in Fig. 7, we consider a limited number of power states for each device in mobile phones. The connection between these power states are system calls and binder message as *actions*, e.g., when the package number p is larger than 100kB (i.e., t in the third row of Table 2) in Android 5.0.1, the WIFI is switched to a high power state. These power profiles are obtained offline, measured from a multimeter [4].

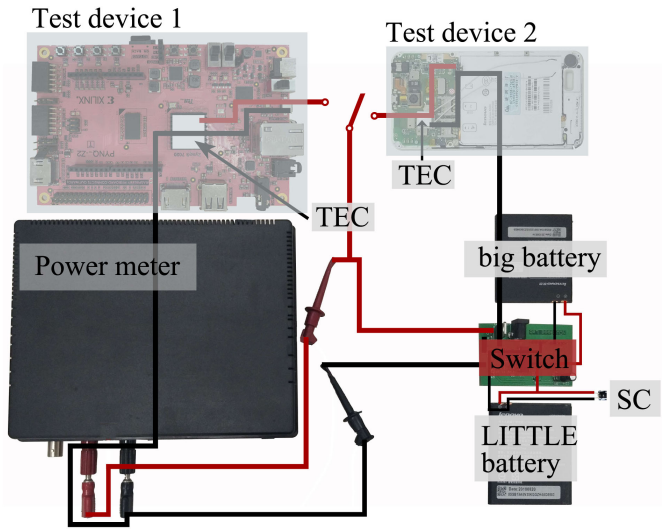


Fig. 10. The full prototype of big.LITTLE battery support for mobile devices, such as PYNQ board(test device 1) and phone(test device 2). We implement a switch facility to convert CAPMAN algorithm output into battery switch signal. For the LITTLE battery, the voltage is unstable in spike. We installed a super-capacitor to boost and filter the LITTLE output, such that CAPMAN can have a reliable power supply.

In all, the whole system is connected and measured in Fig. 10.

For active cooling, CAPMAN adds a TEC device, covering the CPU in both the mobile phone and FPGA board. The TEC allows dynamic cooling efficient when powered with different Ampere, as shown in Fig. 6. We have profiled our TEC chip offline, finding its maximum cooling efficiency point and keep it in that way. Thus, the TEC power model works as an on/off scheme in CAPMAN. In the physical setup, TEC is powered on directly from the switch facility when the temperature is higher than 45°C threshold. On a rooted phone, we can get the current CPU frequency through the ADB command. By calculating the CPU frequency and the crystal frequency of the hardware circuit, we can figure out how many states the prediction system should predict at least.

4.2 The Switch Facility

To implement the big.LITTLE battery support, we design a physical implementation with power monitors and the switch facility, as shown in Fig. 10.

The Switch taps its internal clock on communicating with the smartphone. The Switch installs an oscillator with a 20 kHz range, which allows us to produce big.LITTLE battery switch at a millisecond scale. The circuit schematic of the Switch in illustrated in Fig. 11. The Switch operates at different voltage level. When a different battery demand arrives, the comparator raises to 3.5V, indicating a upper signal that turns on the left Mos Tube in Fig. 11. If received signal flips again, the specified voltage drops to 0.3V, making the switch turn onto the right Mos Tube, which eventually picks the right battery. As such, CAPMAN can manage the battery supply from big.LITTLE batteries.

CAPMAN is also implemented on the PYNQ development board, as shown in Fig. 10. From the perspective of hardware architecture, the PYNQ board equips Xilinx ZYNQ SoC, which is an FPGA SOC platform that combines programmable logic (PL) and programmable system (PS). There are many isolated devices, including WiFi chips, screen connector, etc., allowing us to measure the individual power such that we can evaluate the performance from CAPMAN specifically.

TABLE 2
Power Models

Component	Model	Citation
CPU	$P^{CPU} = \gamma_{freq}^{CPU} \times \mu + C_{CPU}$ $\mu : utilization, 0 \leqslant \mu \leqslant 100$ $freq : frequency\,index, freq = 0, 1, 2 \cdots, n$	[52]
Screen	$P^{Screen} = \left(\frac{\alpha_b + \alpha_w}{2} \times B_{\text{level}}\right) + C_{Screen}$ $\alpha_b,\ \alpha_w,\ C_{Screen} : power\,coefficients$ $B_{\text{level}} : brightness\,level,\ 0 \leqslant B_{\text{level}} \leqslant 255$	[9]
WIFI	$P^{WiFi} = \begin{cases} \gamma_l^{WIFI} \times p + C_l & if\ p \leqslant t \\ \gamma_h^{WIFI} \times p + C_h & if\ p > t \end{cases}$ $p : packet\,rate,\ t : threshold$	[62]
TEC	$P^{TEC} = \alpha I \Delta T + I^2 R$ where I can be calculated from Equation (1).	[19]

Since the space available for the battery in a mobile phone is limited, the hybrid battery pack contains two batteries each with a smaller size than usual (e.g., 1/2 of the usual size). In the case of the same capacity, CAPMAN reduces the total mass of the batteries due to higher energy density. In addition, the battery packaging technology [28] can mold the battery into different shapes. We are able to reduce the size of the batteries by designing a suitable shape.

5 Evaluation

In this section, we prototype CAPMAN onto our physical testbed and workloads to evaluate its performance.

5.1 Hardware/Software Setup

As aforementioned, we have built a physical test platform to evaluate the runtime performance of CAPMAN. The testbed extends our prototype in Fig. 10 to power meters (i.e., Agilent 34410A multi-meter). We install the big.LITTLE battery pack including one $LiNiMnCoO_2$ (LMO) and $LiNiCoAlO_2$ (NCA) each. The voltage comparator is an LM339AD chip that outputs battery switch signal from Pin 10. For active cooling, we use an ATE-31-2.2A TEC, weighting less than 2 gram. We perform tests onto PYNQ-Z2 and three phones, with CPU frequency ranging from 1040 kHz

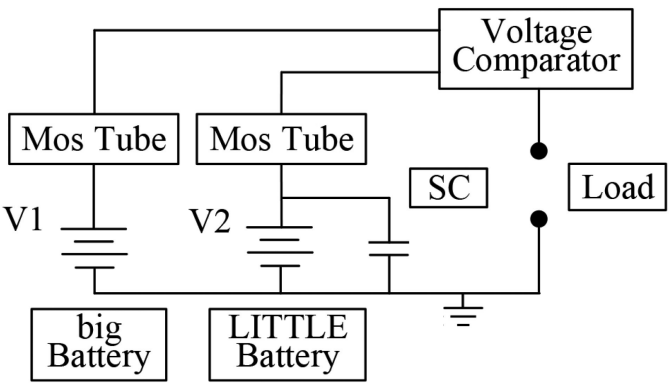


Fig. 11. Circuit diagram of the battery switcher.

TABLE 3
Average Power Costs of all Hardware States in TESTED Devices

Hardware	CPU [7]				Screen [37]	
Status	C0	C1	C2	Sleep	Off	On
Power(mW)	612	462	310	55	22	790
Hardware	WiFi [23]				TEC [52]	
Status	Idle	Access	Send		Off	On
Power(mW)	60	1284	1548		0	29.17

to 2000kHz, with installed Android ROM version 5.0-7.1. All of our experiments are going under the ambient temperature, e.g., 25°C. The detailed power profile of each device used in our test is shown in Table 3.

Workloads and Traces. We verified the performance of CAPMAN using real world workloads and traces. Each benchmark perform a diverse software behavior that can verify the performance of CAPMAN in the wild.

The benchmarks are as follows:

- *Geekbench* is a resource intensive benchmark. This workload always fulfills the system utilization, making the power profile easier to predict.
- *PCMark* is a CPU intensive benchmark, modified with occasional user interactions. This is used to test CAPMAN behavior when software pattern changes.
- *Video* is a stable workload that keeps playing short videos.
- *η-Static* is a mixed workload batch controlled by η, where η is the ratio for mixing *PCMark* and *Video* workloads.

Baselines. We mainly evaluate CAPMAN with the following baselines:

- *Oracle* is a baseline based on offline analysis, serving ground truth.
- *Practice* is the baseline that phone equips a single battery with the same capacity.
- *Dual* deploys big.LITTLE batteries but always uses LITTLE battery first.
- *Heuristic* is a dual battery baseline with a CPU utilization-based prediction model in Table 2. It only considers the power consumption of the CPU and is a simple but widely used baseline.

We first measured the service time of the hybrid single battery pack with the same volume. The experiment measures the time duration from fully discharge to the system is automatically shutdown. In order to analyze the scheduling of heterogeneous batteries at a finer granularity, we tested the battery usage of the five baselines in each battery scheduling cycle and combined it with workflow analysis. In terms of thermal control, we measured the temperature of the mobile phone mainboard and the corresponding TEC working condition. Together, we combine the colelcted heat data with the workflow characteristic to do a detailed analysis.

5.2 One Discharge Cycle Performance

We first plot an one-discharge-cycle performance of CAPMAN with other baselines. Fig. 12 illustrates this comparison using six different workloads nameed *Geekbench*, *PCMark*, *Video* and three setups of $\eta - *$ static workloads. The green line in figures is the fitted curve and the green dots are data collected from multiple simulation

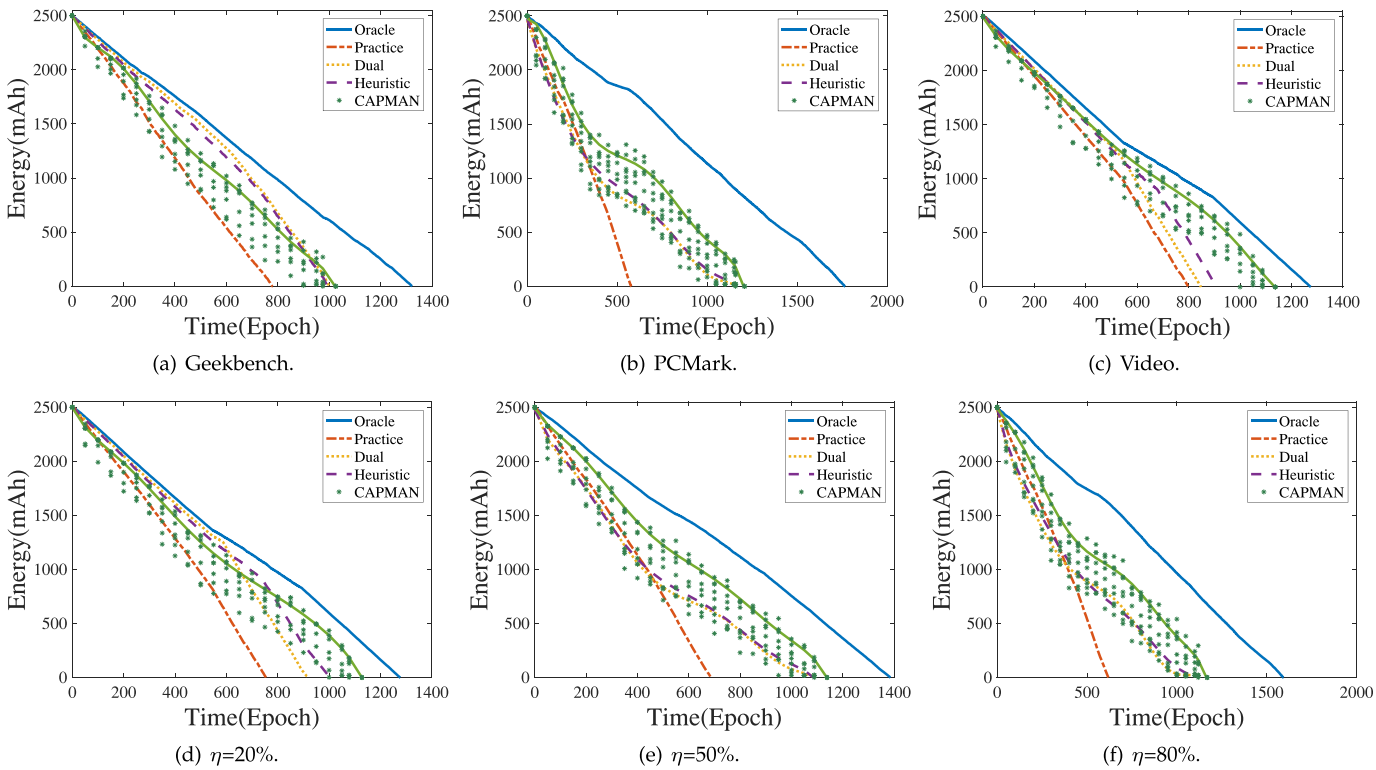
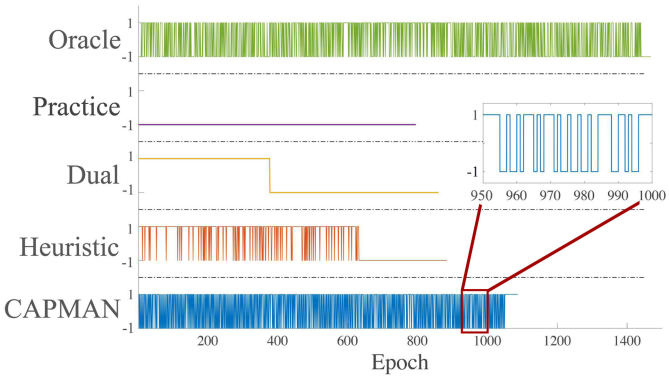
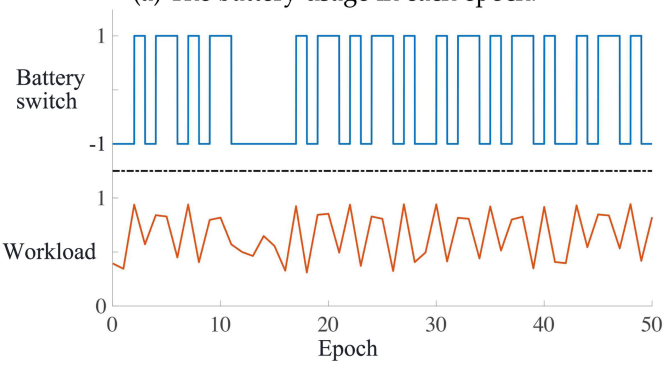


Fig. 12. Performance comparison of CAPMAN and different baselines. (The green dots are data collected from all trials using CAPMAN and the green line is the fitting curve.)



(a) The battery usage in each epoch.



(b) The impact of workflow.

Fig. 13. Battery switching (a) and the impact of workflow on it (b).

experiments. It is worth mentioning that, CAPMAN also includes the power consumption of running TEC here, except for the typical high power consumption hardwares.

In Geekbench, there are many CPU and memory intensive jobs, leading to a fully occupied system. In this workload, CAPMAN is not very different from *Dual* and *Heuristic*, (Fig. 12a). It is because CAPMAN spends extra power on maintaining the MDP representation of the system and constantly updates the similarity, which is unnecessary in stationary workloads like Geekbench. However, compared with the *Practice*, CAPMAN can still prolong 50 percent more service time. For the PCMark workload, the system is not fully utilized. Then, CAPMAN gradually learns the state behavior, and reacts to the right battery decision. Thus, CAPMAN improves the performance by 21.3, 25.7 percent, compared to *Dual* and *Heuristic* at last (Fig. 12b), though the energy drains fast in the beginning. When the workload is Video, which demand becomes dynamic, CAPMAN can significantly outperform other baselines, namely 53.27, 55.08, and 67.1 percent longer service time than *Heuristics*, *Dual*, and *Practice*, respectively (Fig. 12c). Note that, it is also very close to the theoretical best–*Oracle*, within 9.6 percent less service time.

When executing a mixed workload (Figs. 12e, 12f), CAPMAN extends 76, 105, and 114 percent more service time than *Practice*. Compared to a single battery with the same capacity, CAPMAN smartly scheduling heterogeneous dual batteries and therefore doubles the service time on average. CAPMAN shows a closer curve than all the baselines to the offline optimal *Oracle* at most times.

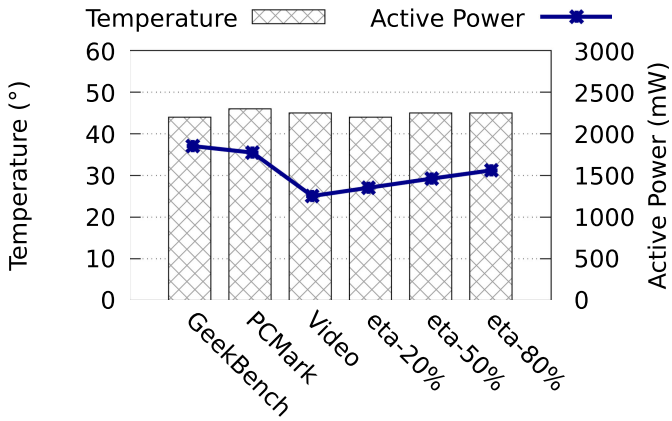


Fig. 14. Cooling and active power consumption of different workloads.

5.3 Battery Scheduling

In order to analyze the heterogeneous battery scheduling in more detail, we show the battery usage of five baselines in each battery scheduling cycle in Fig. 13a, where "-1" means using a big battery and "1" means using a LITTLE battery. In a system with two batteries, if one battery is exhausted, the system can only choose to use another battery, as shown at the end of the CAPMAN curve. Here we show a zoom-in from the battery scheduling decision in CAPMAN. Because CAPMAN can accurately predict and use the correct battery to respond to energy demand, it has a better performance than other baselines, which is closest to *Oracle*. *Heuristic* only makes predictions based on CPU utilization. In this baseline, LITTLE battery is usually inefficiently selected, causing the LITTLE battery to unnecessarily drain out thus affecting performance. *Dual* always calls the LITTLE battery first to meet the energy demand. Due to the characteristics of the workflow, the LITTLE battery can better meet the energy demand such that saves energy. Therefore, *Dual* has better performance than *Practice*. If the workflow behaves smoothly which favors the discharge profile of the big battery, then Practice could have a longer discharge cycle than *Dual*.

Changes in workflow directly affect battery scheduling. Fig. 13b reveals the relationship between workflow and battery scheduling.

After we have processed and normalized the original data of the workflow to [0,1], the value of the data directly reflects the dramatic changes in the workflow. The drastic situation of workflow changes is basically the same as the battery selection. When the workflow changes drastically, CAPMAN will choose a LITTLE battery to deal with, otherwise it will use a big battery. CAPMAN has a 84.2 percent accuracy rate to select the correct battery according to the changes in the workflow, which allows CAPMAN to save energy to a large extent, extend the battery life of the mobile phone and improve the user experience.

5.4 Cooling and Active Power Management

Fig. 14 presents how CAPMAN handles cooling when the active power varies. In all workloads, CAPMAN can maintain the temperature at the predefined temperature, which is around 45 degree. When the whole system works at its designed highest utilization and the active power reaches to 2300mW, CAPMAN boots up the TEC to reduce the mainboard temperature. The active power is much smaller when

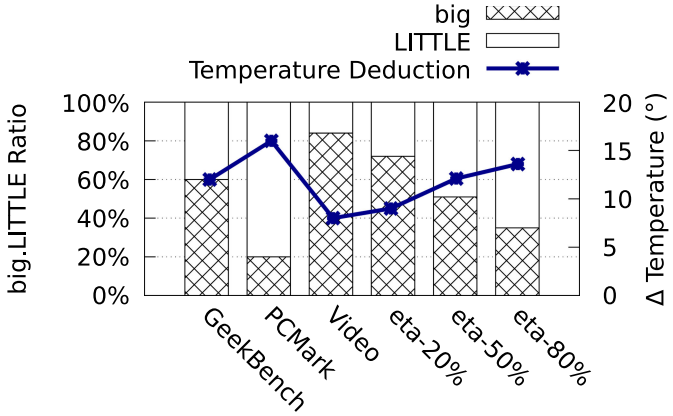


Fig. 15. The relationship between big.LITTLE ratio and temperature reduction in different workloads.

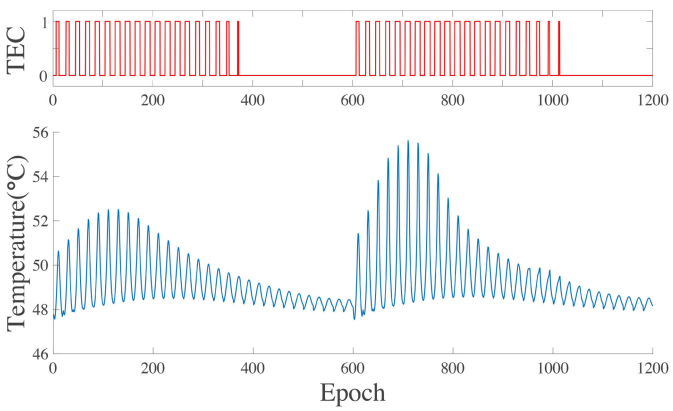


Fig. 16. Top: The working status of TEC, where "1" means TEC is working, "0" means TEC is off. Bottom: The temperature of the phone's mainboard.

the workload is less intensive, such as the Video workload, due to both mobile phone and TEC cooling consume less energy.

To further illustrate the performance, here we measure temperature reduction and compare with the case without TEC. And we show the ratio of activation time between the big and LITTLE battery in Fig. 15. CAPMAN selects the appropriate battery based on the current state of the system. It is clear that when more dynamic power surges arrive in the system, which could be either a CPU intensive job or the whole system boosts up, LITTLE battery takes in charge. In these case, TEC is highly likely to be on for active cooling. Therefore, in PCMark and $\eta - 80\%$, it reduces the most temperature beyond the default cooling plate.

In order to analyze the heat dissipation of the mobile phone more fine-grained, we plot a comparison chart of the working condition of the TEC and the temperature of the mobile phone. The independent temperature of each module is difficult to measure, so we use a thermal simulator [58] to get mobile phone temperature. In the implementation, we placed the TEC in the hottest part of the entire mobile phone,which is CPU. So here we mainly focus on the temperature of the mobile phone main board, as shown at the bottom of Fig. 16. When the temperature of the mobile phone exceeds the threshold, the TEC starts to work and effectively reduces the temperature of the mobile phone. Once the temperature is below the threshold, TEC stops working. Due to the influence of the workflow, when the

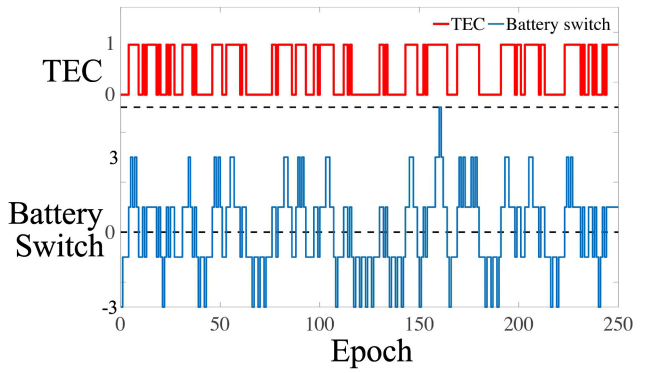


Fig. 17. The impact of TEC on battery scheduling.

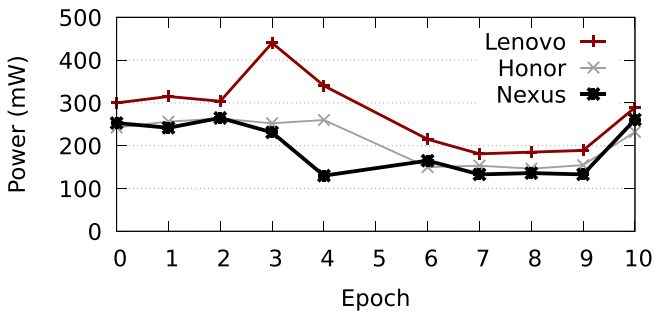


Fig. 18. A snapshot of CAPMAN on different phones.

temperature of the mobile phone raises, the TEC starts to work again.

The opening of TEC requires the support of large currents, which greatly affects battery scheduling. In Fig. 17, the thick red line is the working condition of TEC, where "1" indicates that the TEC starts to work, and "0" indicates that the TEC is off; the blue line indicates the battery scheduling situation, where "-1" indicates that CAPMAN uses a big battery, and "1" indicates CAPMAN uses a LITTLE battery. The opening of the TEC requires the support of a large current, and at this time the system will prefer a LITTLE battery. When the TEC is turned on, the usage rate of the small battery increases significantly.

5.5 Stability and Scalability

We test CAPMAN onto three different phones, namely Nexus, Honor, and Lenovo. One snapshot of runtime performance is shown in Fig. 18. Under the same workload traces, CAPMAN shows similar active power management between different phones, raising from 100mW to 450mW. In our implementation, we build CAPMAN within the OS ROM, such that all Android phones can have system support for big.LITTLE batteries.

In CAPMAN design, ρ is an important factor that measures the performance competitiveness between CAPMAN and optimal solution. However, setting the value of ρ is very subtle, for setting it too high could also lead to more recursion calculation in Algorithm 1. We show the impact of ρ on the computation overhead in Fig. 19. This impact varies from different phones, as the computation resource varies. However, all curves show the similar exponential behavior when ρ increases. When ρ reaches 1, the battery

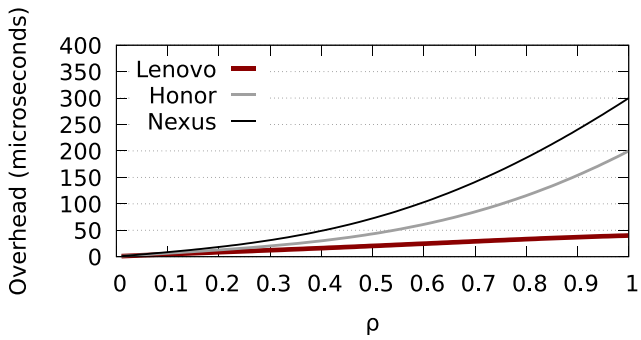


Fig. 19. The impact of the discount factor ρ.

control unstable because the overhead reaches to 300 microseconds in Nexus. Thus, for each device, CAPMAN needs to recalibrate to find a suitable configuration.

In order to further verify the scalability of CAPMAN, we tested its performance on PYNQ and plotted the performance of one-discharge-cycle. Fig. 20 shows a comparison of using two workflows to light the LED and flash the LED. The green dots are data collected from multiple simulation experiments for CAPMAN and the green line is the fitted curve. Compared with Practice, CAPMAN can extend the service time by 49.2 and 42.5 percent respectively. In other words, whether it is a stable workflow or a workflow that changes frequently, CAPMAN is better than Practice.

Limitations and Discussions. Different active states of components in the SoC are the true victim of power demand dynamics. Therefore, if we allow a finer-grained heterogeneous battery management, we may (1) achieve better battery service quality; (2) allow more battery chemistries in supporting computing systems. Second, in the current setup, we only use new batteries and ignore the aging problem.

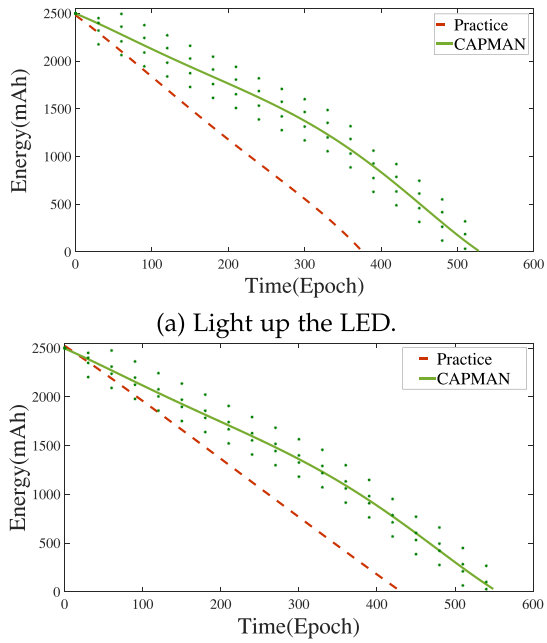
In practice, battery performances and attributes vary along the serving time. The influence of usage time on the battery status is another interesting research topic. Besides, affected by the workflow, a certain battery may be used up early in the experiment, resulting in one battery supply in the rest of the service time. A scale-out design of battery scheduling is worthy of further discussion.

6 Related Work

CAPMAN is a framework designed for cooling and active power management in battery-powered system domain. Related work are from battery scheduling and power and cooling management as follows.

Battery Scheduling. Despite the considerable effort that has been spent, today's smartphones still face many performance challenges [24]. Battery life is a critical performance and user experience metric on mobile devices [38]. Falaki *et al.* [22] develop the prototype that manages multiple batteries for electric vehicles. Ziyou Song *et al.* [47] propose a HESS including multiple batteries and supercapacitors(SC) for electric buses. Jin Lu *et al.* [33] further provide a more detailed controller design for battery charging. Bruno Pollet and his colleagues [42] highlight the current status of hybrid battery, and fuel cell electric vehicles in the electrochemical market.

There are many work that leverage hybrid batteries in different areas such as EV [47], datacenters [20], [61], smart grid [13], etc. Our paper is one of the first attempts to



(a) Light up the LED.

(b) Flash the LED.

Fig. 20. Performance comparison of CAPMAN and different baseline on PYNQ. (The green dots are data collected from multiple simulation experiments for CAPMAN and the green line is the fitted curve.)

provide system support for battery scheduling, based on the SDB idea from [10].

Power and Cooling Management. Power management has been studied for a long time in battery-powered device domain. Carroll *et al.* [14] propose a basic power modeling set. Y. Hu *et al.* [27] propose a power analog circuit level analysis. Their works pile as the early pioneers in the modeling field, however, not in energy conservation management. Clara Martinez *et al.* [35] present a comprehensive analysis of energy management strategies (EMSs) towards blended mode and optimal control.

L. Sun *et al.* [49] conducted an extensive experimental evaluation of the WiFi power consumption model of smartphones. Our work focuses on battery management to prolong smartphone service times.

Heat dissipation limits the device performance and battery lifetime [36]. To reduce the temperature, plenty of research has been devoted [21], [39], [40] for thermal modeling. In this field, the work from Dai [19] is closest to ours. They propose to use a small thermoelectric generator (TEG) to generate surge power, in order to support active cooling using thermoelectric coolers. However, adopting a TEG device into a smartphone is another challenge for manufacturing. Any of those solutions (including ours) jointly managing the power and cooling can work together with passive cooling methods like vapor chamber solutions [26], [31], [57]. Our work focuses on a more practical aspect for cooling and active power management: *exploiting the existing battery chemistries to harvest more service times.*

7 Conclusion

Modern smartphone design is constrained by the power and thermal wall. Battery engineering suggests a hybrid battery pack can perform better than a single battery. To provide the software-defined management on the big.LITTLE battery, we

propose CAPMAN, a cooling-aware battery management facility that effectively extends the service time and enables efficient TEC cooling on phones. CAPMAN models power profiles and provides runtime calibration for battery scheduling, as well as provides an online algorithm with a proved worst-case O (1.05)-competitiveness performance. We implement CAPMAN on popular smartphones and find it can significantly extend (114 percent) the service time while maintaining the ambient temperature. Compared to state-of-the-practice baselines, CAPMAN shows an average 55.08 percent performance gain with 53.27 percent less power use.

Acknowledgments

This work was supported by NSFC funding No. 61702250 and 61702329, MST National R&D Key project No. 2018Y FB14043033, Nuclear High Foundation Special Fund 2018ZX01035-101, and Jiangxi Thousands of Talents project No. jxsq106018. The authors would like to thank Prof. Hao Wang on reviewing the theory proof and Prof. Xiaorui Wang on discussing and inspiring the original research question. They would also like to thank all reviewers for their valuable comments.

References

[1] Agilent 34410a multimeter. Accessed: Dec. 4, 2019. [Online]. Available: https://www.keysight.com/
[2] Arm big.little core. Accessed Oct. 10, 2020. [Online]. Available: https://zh.wikipedia.org/wiki/Big.LITTLE
[3] Battery university. Accessed: Dec. 4, 2019. [Online]. Available: https://batteryuniversity.com/
[4] The keysight power meter. Accessed: Dec. 4, 2019. [Online]. Available: https://www.keysight.com/en/pd-692834-pn-34410A/digital-multimeter-6-di%git-high-performance?&cc=CN&lc=chi
[5] Pcmark. Accessed May 10, 2020. [Online]. Available: https://benchmarks.ul.com/zh-hans/pcmark10
[6] Pynq. Accessed May 10, 2020. [Online]. Available: https://www.pynq.io/
[7] A. Abdelmotalib and Z. Wu, "Power consumption in smartphones (hardware behaviourism)," *Int. J. Comput. Sci. Issues (IJCSI)*, vol. 9 no. 3, p. 161, 2012.
[8] A. Agrawal, K. Shah, A. Kumar, and R. Chandra, "Battery scheduling problem," in *Int. Conf. Theory and Appl. Models Comput.*, pages 1–12. Springer, 2019.
[9] F. A. Ali, P. Simoens, T. Verbelen, P. Demeester, and B. Dhoedt, "Mobile device power models for energy efficient dynamic offloading at runtime," *J. Syst. Softw.*, 113:173–187, 2016.
[10] A. Badam *et al.*, "Software defined batteries," in *Proc. 25th Symp. Operating Syst. Princip.*, 2015, pp. 215–229.
[11] A. B.-C. and P. Wang, "On-chip hot spot remediation with miniaturized thermoelectric coolers," *Microgr. Sci. Technol.*, vol. 21 no. 1, pp. 351–359, 2009.
[12] M. Bartlett and B. Sherrill, "Battery pack including an emergency back-up battery for use in mobile electronic devices, June 1 2010," US Patent 7,728,549.
[13] T. Bocklisch, "Hybrid energy storage systems for renewable energy applications," *Energy Procedia*, vol. 73, pp. 103–111, 2015.
[14] A. Carroll *et al.*, "An analysis of power consumption in a smartphone," in *Proc. USENIX Annu. Techn. Conf.*, 2010, vol. 14, pp. 21–21.
[15] Y.-W. Chang, C.-H. Cheng, W.-F. Wu, and S.-L. Chen, "An experimental investigation of thermoelectric air-cooling module," *Int. J. Mech. Aerosp. Ind. Mechatronic Manuf. Eng.*, vol. 1, 2007Art. no. 9.
[16] G. Chemla, "Integrated circuit temperature monitoring and protection system, 1998, " US Patent 5,805,403.
[17] C.-F. Chiasserini and R. R. Rao, "Energy efficient battery management," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 7, pp. 1235–1245, Jul. 2001.
[18] V. Chiriac, S. Molloy, J. Anderson, K. E. Goodson, "A figure of merit for smart phone thermal management," *Fig. Merit Smart Phone Thermal Manage.*, p. 16, 2017.
[19] Y. Dai, T. Li, B. Liu, M. Song, and H. Chen, "Exploiting dynamic thermal energy harvesting for reusing in smartphone with mobile applications," *ACM SIGPLAN Notices*, vol. 53, pp. 243–256, 2018.
[20] W. Di, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. Fathy, "Energy storage in datacenters: What, where, and how much?," *ACM Sigmetrics Perform. Eval. Rev.*, vol. 40, no. 1, pp. 187–198, 2012.
[21] B. Egilmez, G. Memik, S. Ogrenci-Memik, and O. Ergin, "User-specific skin temperature-aware dvfs for smartphones," in *Proc. IEEE Des., Autom. Test Eur. Conf. Exhib.*, 2015, pp. 1217–1220.
[22] M. H. Falaki, "Automating Personalized Battery Management on Smartphones," PhD thesis, UCLA, 2012.
[23] R. Friedman, A. Kogan, and Y. Krivolapov, "On power and throughput tradeoffs of wifi and bluetooth in smartphones," *IEEE Trans. Mobile Comput.*, vol. 12, no. 7, pp. 1363–1376, Jul. 2013.
[24] Y. Gao *et al.*, "Whom to blame? Automatic diagnosis of performance bottlenecks on smartphones," *IEEE Trans. Mobile Comput.*, vol. 16, no. 6, pp. 1773–1785, Jun. 2017.
[25] K. Hasebe, T. Niwa, A. Sugiki, and K. Kato, "Power-saving in large-scale storage systems with data migration," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci.*, 2010, 266–273.
[26] S.-S. Hsieh, R.-Y. Lee, J.-C. Shyu, and S.-W. Chen, "Thermal performance of flat vapor chamber heat spreader," *Energy Convers. Manage.*, vol. 49, no. 6, pp. 1774–1784, 2008.
[27] Y. Huh, "Future direction of power management in mobile devices," in *Proc. IEEE Asian Solid-State Circuits Conf.*, 2011, pp. 1–4.
[28] A. N. Jansen, K. Amine, A. E. Newman, D. R. Vissers, and G. L. Henriksen, "Low-cost, flexible battery packaging materials," *JOM*, vol. 54, no. 3, pp. 29–32, 2002.
[29] G. Jeh and J. Widom, "SimRank: A measure of structural-context similarity," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2002, pp. 538–543.
[30] N. K. Jong and P. Stone, "State abstraction discovery from irrelevant state variables," in *Proc. 19th Int. Joint Conf. Artif. Intell.*, 2005, vol. 8, pp. 752–757.
[31] Y. Koito, H. Imura, M. Mochizuki, Y. Saito, and S. Torii, "Numerical analysis and experimental verification on thermal fluid phenomena in a vapor chamber," *Appl. Therm. Eng.*, vol. 26, no. 14–15, pp. 1669–1676, 2006.
[32] A. D. Kraus and A. Bar-Cohen, *Thermal Analysis and Control of Electronic Equipment*, Washington, DC, USA: Hemisphere Publishing Corp., 1983, p. 633.
[33] J. Lu, T. S. Kelly, and L. Cheung, "Battery management system and method, 2013," US Patent 8,583,955.
[34] D. Ma and R. Bondade, "Enabling power-efficient DVFS operations on silicon," *IEEE Circuits Syst. Mag.*, vol. 10, no. 1, pp. 14–30, First Quarter 2010.
[35] C. M. Martinez, X. Hu, D. Cao, E. Velenis, B. Gao, and M. Wellers, "Energy management in plug-in hybrid electric vehicles: Recent progress and a connected vehicles perspective," *IEEE Trans. Veh. Technol.*, vol. 66, no. 6, pp. 4534–4549, Jun. 2017.
[36] P. Mercati, T. S. Rosing, V. Hanumaiah, J. Kulkarni, and S. Bloch, "User-centric joint power and thermal management for smartphones," in *Proc. 6th Int. Conf. Mobile Comput. Appl. Serv.*, 2014, pp. 98–105.
[37] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 317–328.
[38] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 317–328.
[39] J. Park and H. Cha, "Aggressive voltage and temperature control for power saving in mobile application processors," *IEEE Trans. Mobile Comput.*, vol. 17, no. 6, pp. 1233–1246, Jun. 2018.
[40] F. Paterna, J. Zanotelli, and T. S. Rosing, "Ambient variation-tolerant and inter components aware thermal management for mobile system on chips," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, 2014, pp. 1–6.
[41] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang, "Fine-grained power modeling for smartphones using system call tracing," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 153–168.
[42] B. G. Pollet, I. Staffell, and J. L. Shang, "Current status of hybrid, battery and fuel cell electric vehicles: From electrochemistry to market prospects," *Electrochimica Acta*, vol. 84, pp. 235–249, 2012.

[43] W. Qi *et al.*, "Construction and mitigation of user-behavior-based covert channels on smartphones," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 44–57, Jan. 2018.
[44] A. Rice and S. Hay, "Measuring mobile phone energy consumption for 802.11 wireless networking," *Pervasive Mobile Comput.*, vol. 6, no. 6, pp. 593–606, 2010.
[45] S. B. Riffat and X. Ma, "Thermoelectrics: A review of present and potential applications," *Appl. Thermal Eng.*, vol. 23, no. 8, pp. 913–935, 2003.
[46] S. Rosen, A. Nikravesh, Y. Guo, Z. M. Mao, F. Qian, and S. Sen, "Revisiting network energy efficiency of mobile apps: Performance in the wild," in *Proc. Internet Meas. Conf.*, 2015, pp. 339–345.
[47] Z. Song, H. Hofmann, J. Q. Li, X. Han, and M. Ouyang, "Optimization for a hybrid energy storage system in electric vehicles using dynamic programing approach," *Appl. Energy*, vol. 139, pp. 151–162, 2015.
[48] Norazah Mohd. Suki and Norbayah Mohd. Suki, "Mobile phone usage for m-learning: Comparing heavy and light mobile phone users," *Campus-Wide Inf. Syst.*, vol. 24, pp. 355–365, 2007.
[49] L. Sun, H. Deng, R. K. Sheshadri, W. Zheng, and D. Koutsonikolas, "Experimental evaluation of WiFi active power/energy consumption models for smartphones," *IEEE Trans. Mobile Comput.*, vol. 16, no. 1, pp. 115–129, Jan. 2017.
[50] R. S. Sutton and A. G. Barto," *Reinforcement Learning: An Introduction*," Cambridge, MA, USA: MIT press, 2018.
[51] F. L. Tan and S. C. Fok, "Thermal management of mobile phone using phase change material," in *Proc. 9th Electron. Packag. Technol. Conf.*, 2007, pp. 836–842.
[52] F. L. Tan and S. C. Fok, "Methodology on sizing and selecting thermoelectric cooler from different tec manufacturers in cooling system design," *Energy Convers. Manage.*, vol. 49, no. 6. pp. 1715–1723, 2008.
[53] E. Wang, "Mobile electronic devices with integrated personal cooling fan, August 11 2011," US Patent App. 12/919,473.
[54] H. Wang, S. Dong, and L. Shao, "Measuring structural similarities in finite MDPs," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 3684–3690.
[55] V. Wienert *et al.*, "Local thermal stress tolerance of human skin," *Anasthesie Intensivtherapie Notfallmedizin*, vol. 18, no. 2, pp. 88–90, 1983.
[56] S. J. William, "Optimal flow through networks," *Operations Res.*, vol. 10, pp. 476–499, 1962.
[57] S.-C. Wong, K.-C. Hsieh, J.-D. Wu, and W.-L. Han, "A novel vapor chamber and its performance," *Int. J. Heat Mass Transfer*, vol. 53, no. 11–12, pp. 2377–2384, 2010.
[58] Q. Xie, M. J. Dousti, and M. Pedram, "Therminator: A thermal simulator for smartphones producing accurate chip and skin temperature maps," in *Proc. Int. Symp. Low Power Electron. Des.*, 2014, pp. 117–122.
[59] Q. Xie, J. Kim, Y. Wang, D. Shin, N. Chang, and M. Pedram, "Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2013, pp. 242–247.
[60] F. Xu, Y. Liu, Q. Li, and Y. Zhang, "V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 43–55.
[61] Y. Hu, H. Sun, J. Gu, L. L. Tao, and Li and, "HEB: Deploying and managing hybrid energy buffers for improving datacenter efficiency and economy," in *Proc. ACM/IEEE 42nd Annu. Int. Symp. Comput. Archit.*, 2015, pp. 463–475
[62] L. Zhang *et al.*, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardware/Softw. Codesign Syst. Synthesis*, 2010, pp. 105–114.

Zichen Xu (Member, IEEE) received the PhD degree from the Ohio State University. He is currently a full professor at Nanchang University, China. His research interests include data-conscious complex system, including profile analysis, system optimization, and storage system design/implementation. He is also a member of the ACM.

Jie Zhou received the undergraduate degree from Nanchang University. She is currently working toward the graduate degree from Tongji University. Her research interests include effective energy utilization of IoT devices.

Wenli Zheng (Member, IEEE) received the PhD degree from the Ohio State University, in 2016. He is currently a tenure-track assistant professor at the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests mainly include data center power management, cloud computing, and distributed systems.

Yuhao Wang (Senior Member, IEEE) received the PhD degree in space physics from Wuhan University, Wuhan, China, in 2006. Since 2006, he has been an associate professor at Mobile Laboratory, School of Information Engineering, Nanchang University, Nanchang, China. During the summer of 2008, he was as a visiting professor with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada, where he conducted research on radio channel modeling and communication evaluation in Smart Homes, vehicular ad hoc networks (VANETs), and sensor networks for multimedia applications supported by the Mathematics of Information Technology and Complex Systems Project. His current research interests include radio measurements, channel modeling, radio link and network simulation, wireless cellular networks, VANETs, and software-defined radio.

Minyi Guo (Fellow, IEEE) received the PhD degree in computer science from the University of Tsukuba, Tsukuba, Japan. He is currently a Zhiyuan chair professor at Shanghai Jiao Tong University, Shanghai, China. His research interests include parallel and distributed computing, Big Data, and parallelizing compilers. In 2007, he received the Recruitment Program of Global Experts and Distinguished Young Scholars Award from the National Natural Science Foundation of China. He is also on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems* and the *IEEE Transactions on Computers*.