EasyChair Preprint

№ 3215

CAPMAN: Cooling and Active Power Management in big.LITTLE Battery Supported Devices

Jie Zhou, Zichen Xu, Wenli Zheng and Yuhao Wang

April 20, 2020

CAPMAN: Cooling and Active Power Management in big.LITTLE Battery Supported Devices

Jie Zhou
Nanchang University
zhoujie@email.ncu.edu.cn

Zichen Xu*
Nanchang University
xuz@ncu.edu.cn

Wenli Zheng*
Shanghai Jiao Tong University
zheng-wl@cs.sjtu.edu.cn

Yuhao Wang
Nanchang University
wangyuhao@ncu.edu.cn

Abstract*—Modern smartphone is far from being ubiquitous due to limited energy capacity. Recent research suggests that heterogeneous batteries may expose power saving opportunities that fit dynamic software patterns. Yet it is still a challenge on thermal and power management for a hybrid battery pack in a smartphone. To address this challenge, we propose a system framework, called CAPMAN , which supports joint optimization of *c*ooling and *a*ctive *p*ower *man*agement in smartphones. The framework consists of three major techniques: 1) A Markov decision process (MDP) technique that models battery types, and cooling/active power use from state and action nodes; 2) A structural similarity approximation that speeds up the convergence of MDP computation, providing battery scheduling decisions; 3) A TEC and battery management facility to realize the cooling and active power management. In addition, CAPMAN provides an online algorithm with a proved worst-case O($\frac{1}{1-\rho}$)-competitiveness performance, where ρ is the factor of discount. We have prototyped CAPMAN with popular smartphones and heterogeneous batteries, and evaluated them with real-world workloads. Results show that CAPMAN can achieve 114% longer service time under skewed loads, compared to the original phone. Compared to the state-of-the-practice baselines, CAPMAN shows 55% performance gain and 53% less energy use on average. As such, CAPMAN approves that big.LITTLE batteries with a careful system design is an effective way to prolong smartphone service times.

I. Introduction

Current battery-powered devices suffer from limited battery capacity. It is worse when popular Apps with rich functions are usually resource-intensive, draining a considerable amount of energy and leaking high surface temperature. This fast power-drain behavior overheats the device [15]. Thus, to achieve better user experience, cooling and power management is essential to a battery-powered device design.

System researchers have spent relentless effort on cooling and power management techniques over the last decade, such as fetch toggling [41], dynamic frequency and voltage scaling (DVFS) [27], load migration [21], and chip-level thermal shutdown [13]. These traditional techniques work with a tradeoff on the performance and power cost, and hence may not work for interactive apps, as they cannot tolerate this significant performance degradation and still yield to a high power consumption and heat dissipation. Smartphone industry tries to adapt larger battery packs and secondary facilities, such as battery banks [10], phase-change material devices [35], and surface cooling fans [37], in order to at least mitigate the power and thermal problem. However, these methods degrade user experience due to their huge volume in size and additional carry-on weight requirement. Additionally, active cooling methods usually solve the thermal problem with higher power consumption. Recently, a work [8] from Microsoft and Tesla propose the idea of software-defined battery (SDB), which introduces power saving opportunities from using heterogeneous battery packs to support various software patterns. Our research develops this idea and asks: if the battery permits, *how could our system supports dynamic cooling and power management from heterogeneous batteries in the mobile system domain?*

To address this problem, we first adapt the SDB definition [8] on the chemistry of different batteries. We then empirically study what cooling and power characteristics of different batteries are favored by software. As such, we obtain important observations as follows:

First, although battery performance may be evaluated in various dimensions, properties like energy density, discharge rate, and cost efficiency are essential to software performance and experience. For example, running specific apps, such as PCMark [3], one battery chemistry may sustain 37.6% more discharge time than another. Different properties of batteries 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, frequently switching batteries may cause additional energy loss and heat dissipation. Second, battery features, software patterns, and correlated device power states, can be modeled into combinatorial states in power management [32]. Actions, such as system calls and user activities, transit these states from one to another. Third, thermal problems caused by resource-intensive apps are usually hot spots (i.e., surface temperature that exceeds 45°C [39]) that call for extra engineering efforts to enable active cooling, which also increases the active power demand.

To exploit the aforementioned observations, in this paper, we propose a **c**ooling and **a**ctive **p**ower **man**agement framework, or *CAPMAN*, which models and optimizes cooling and active power management. CAPMAN works on a system supported by the one-pair battery design that we propose, i.e., big.LITTLE batteries, with which the power management problem becomes categorizing different power demands into either the big or LITTLE battery use. CAPMAN highlights

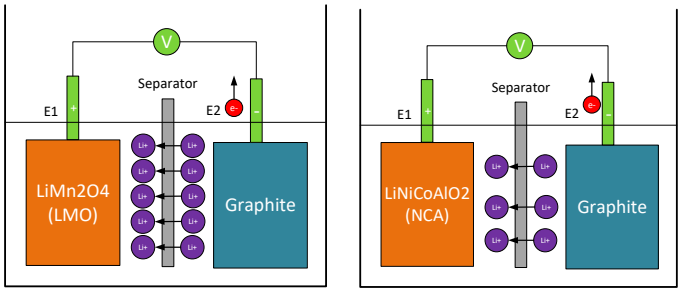
three techniques. (1) A modeling process that abstracts the correlation among layers of batteries, devices, and software, using a finite state machine model, whose state transitions can be triggered by resource demand, system calls, etc. states connected by tensors. Each tensor represents an action of state transition from resource demand, system calls, etc. (2) A Markov decision algorithm that schedule the right battery to use. Since a modern smartphone can have hundreds of apps, tens of devices, and two batteries, the number of states explodes in time-series analysis. To enable CAPMAN to make the right decision within a given time, we derive an approximate online algorithm, proven to achieve a worst-case $O(\frac{1}{1-\rho})$-competitiveness performance. (3) A Thermoelectric Cooler (TEC) [9] and battery management facility, both hardware and software, to realize the scheduling decision. TEC is an efficient cooling device yields to runtime active power consumption, while this active power burst can be addressed by our big.LITTLE battery and predicted by our Markov model. As such, we provide a solution for cooling and active power management in big.LITTLE battery powered systems.

For evaluation, we have prototyped CAPMAN on multiple mainstream smartphones and analyze it with real-world traces and workloads. Results show that CAPMAN can achieve 114% longer service time of smartphones while maintaining the ambient temperature even under skewed loads. Compared to state-of-the-practice baselines, CAPMAN shows 55.08% performance gain and 53.27% less energy use on average. In our prototype design, the total weight of all extra devices is less than 5 gram. The volume overhead of CAPMAN is negligible and the devices can fit into a typical smartphone with proper implementation.

The major contributions are listed below:

- We tackle the thermal and power management of battery-powered devices for improved software performance and user experience. For this, we conduct characterization study of several newly proposed energy storage and cooling facilities, and motivate the opportunity for cooling and active power management.
- We argue the idea of big and LITTLE batteries can significantly prolong the battery life time with a penalty of creating hot spots, while our active cooling facility can address this limitation gracefully.
- We prove the online scheduling scheme of CAPMAN can help to achieve a worst-case $O(\frac{1}{1-\rho})$-competitiveness performance.
- We evaluate CAPMAN with realworld benchmarks, CAPMAN can double the service times than original phone with the same battery capacity.

The remainder of this paper is organized as follows. Section II highlights our motivation on a heterogeneous battery design. Section III describes CAPMAN and proves that it retains performance guarantee. Section IV describes CAPMAN implementation. Section V empirically studies CAPMAN performance. Section VI provides a brief on the related work. At last, Section VII concludes the paper.



(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.

II. System Identification and Motivation

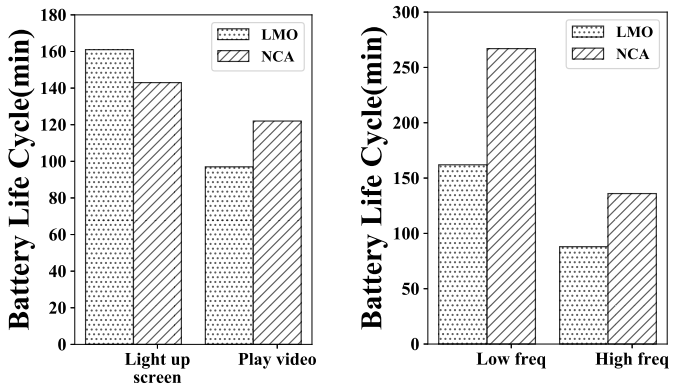
As aforementioned in Section I, recent efforts focus on smart utilization of batteries, using various control, optimization, and scheduling techniques [14]. The smartphone manufacturing limits the size of available batteries, invoking battery pack design composed of heterogeneous battery chemistry, to meet specific software requirements. This calls for a deep understanding on the correlation between battery chemistries, system power consumption, and software patterns. Figure 1 shows that $LiMn_2O_4$ (LMO) has more electrons exchange than $LiNiCoAlO_2$ (NCA) in the same time. In other words, the discharge rate of LMO is much higher than that of NCA.

In this section, we refer to a battery with a high energy density as a *big* battery (low discharge rate), and a battery with a small energy density as a $LITTLE$ battery (high discharge rate). As such, the study enables our system modeling to accurately profile and control the power and performance in CAPMAN design for big.LITTLE battery powered system.

Battery Chemistry and User Behaviors. We first try to understand the power saving potential from using different batteries. We perform different apps on a Nexus 6 phone with Android 5.0.1, collect its discharge cycle (e.g., battery on time), and repeat the same experiment with different batteries [1]. Though running simple applications, such as keeping the phone screen on, and streaming video, or keep turn on and off the phone at different frequency scales, using two battery chemistries with the same capacity (i.e., 2500mAh), as $LiMn_2O_4$ (LMO) and $LiNiCoAlO_2$ (NCA), leads to significantly different discharge cycles, as shown in Figure 2.

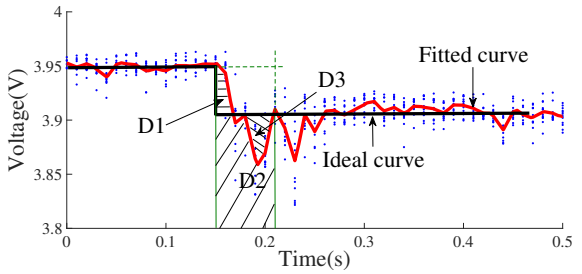
Figure 2(a) shows that using the LMO battery can sustain 14.3% longer than using the NCA battery when keeping the phone on and idle. If the phone plays some videos, the result reverses. The NCA battery can outperform 24% longer than that of LMO. One explanation is these simple applications expose different power demand patterns that favor different battery discharge features. Given the same application, e.g., turning phone on then off, repeatedly, as shown in Figure 2(b), the NCA battery is always better at handling this short burst of power demand, prolonging the discharge cycle time. However, when such behavior frequency increases, from low

[1]More detailed experimental setup can be found in Section V

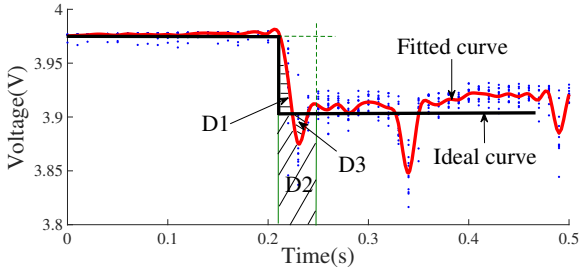


(a) Applications (b) Frequencies

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



(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.

(e.g., each minute) to high (i.e., each second), the relative benefit from using the NCA battery decreases, from 46% longer service time to 35%. As such, this nonlinear behavior of running various apps supported by different batteries calls for a modeling on the power demand surge and its frequency, with a consideration of battery features.

Active Power Savings. To meet load changes, a smartphone may schedule right battery to use. This is already a common practice in electric vehicles (EVs). Fengyuan Xu et al. [42] discovers the V-edge phenomenon. That is when a new power demand arrives, the battery output voltage first quickly drops, and then rises up at a relative lower level than the initial voltage, i.e., the V-edge, shown in Figure 3. We further extend this direction on measuring output voltage with multiple loads and batteries, using an Agilent 34410A multi-meter [1].

Figure 3(a) and (b) illustrate the voltage drop when streaming video and lighting up the screen, respectively. The blue

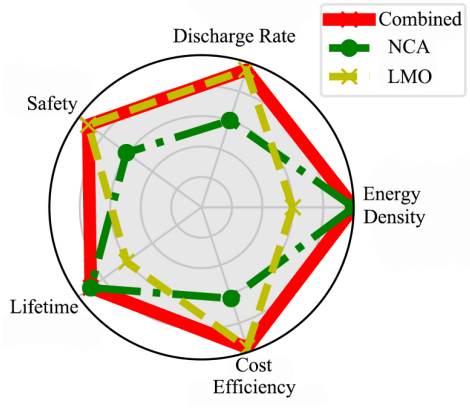


Fig. 4. Metrics comparison between popular smartphone batteries.

dots are collected voltage samples, the red line is the fitted curve, and the bold line is the ideal case. After one peak of power demand, the estimated ideal 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, if this V-edge curve happens frequently, we are looking for one battery chemistry that minimizes $D1$, called *the LITTLE battery* or if the duration of the curve is long, we need one different battery that maximizes $D3$, called *the big battery*.

Battery Features at Different Dimensions. In order to find the batteries with above features, we perform literature study on current available Li-ion battery features used in smartphone power supply [2]. With the same battery capacity, different battery chemistries outperform others in different dimensions. To this end, we provide a radar map of popular smartphone batteries, such as NCA and LMO, on important features, with normalized data shown in Figure 4. There are two important observations from this map. First, no single battery 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 [6]. In this report, we mainly focus on the power savings in one discharge cycle, and thus, without losing any generosity, 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 demand. In our paper, we call this big.LITTLE battery.

Active Power Management with Cooling. Adapting a big.LITTLE battery design further complicates the smartphone design. One issue is that frequently switching between batteries and fast voltage drop raises temperature on some spots. Traditional cooling method in smartphone is a cooling plate that dissipates heat evenly and slowly. Yet hot spots lead to unexpected additional energy loss and heat dissipation. To address this problem, we introduce active cooling technique, i.e., Thermoelectric Cooling (TEC), a very promising heat sink, into the phone design. Recently, TEC has been used for electronic cooling [25], with the advantages on size, quietness, and high reliability. TEC can reduce the temperature fast and

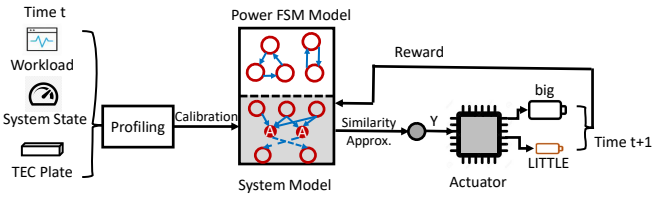


Fig. 5. The CAPMAN framework. Shaded part is CAPMAN's contributions.

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 support cooling and active power management for big.LITTLE battery powered smartphones.

III. 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.

Figure 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 [12]. CAPMAN adapts the finite-state machine model (C. Hu et al [32]), 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 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.

A. 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. Different batteries are suitable for power demands with different characteristics. We investigate six types of widely used lithium batteries and summarize their major properties in Table I. The two properties energy density and

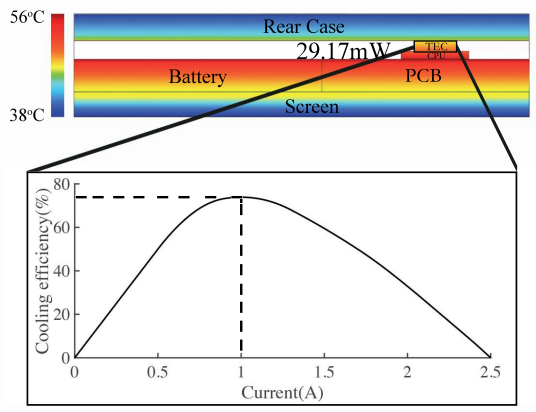


Fig. 6. Top: Temperature distribution in mobile phones, with red for high temperature and blue for normal temperature.The thickness of TEC is generally 2mm. For the convenience of labeling, we have enlarged it. Bottom: Relationship between TEC heat dissipation and its operating current.

discharge rate determine the energy storage capacity and instantaneous power discharge capacity. We can find that 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 with long discharge time but gentle changes, such as playing a video. 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 and then turns it off soon. 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 I. 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 top half of Figure 6 shows a typical temperature distribution across the main components in a smartphone. For the best cooling effect, TECs are placed upon the CPU, to cool the hottest component.

We use a cooling model [16] 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),

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

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. We can see that the heat dissipation rate of TEC is not simply proportional to its operating current. This is also demonstrated by Figure 6, which shows the relationship between the temperature difference between the two sides of TEC and the operating current. As the operating current increases from 0, the temperature difference gradually increases at first, reaches the maximum when the operating current is around 1.0 A (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.

TABLE I
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**

TABLE II
POWER MODELS.

Component	Model	Citation
CPU	$P^{CPU} = \gamma_{freq}^{CPU} \times \mu + C_{CPU}$ μ : $utilization, 0 \leqslant \mu \leqslant 100$ $freq$: $frequency\,index, freq = 0, 1, 2 \cdots, n$	[36]
Screen	$P^{Screen} = \left(\frac{\alpha_b + \alpha_w}{2} \times B_{\text{level}}\right) + C_{Screen}$ α_b, α_w, C_{Screen} : $power\,coefficients$ B_{level} : $brightness\,level$, $0 \leqslant B_{\text{level}} \leqslant 255$	[7]
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 : $threshold$	[44]
TEC	$P^{TEC} = \alpha I \Delta T + I^2 R$ where I can be calculated from Equation (1).	[16]

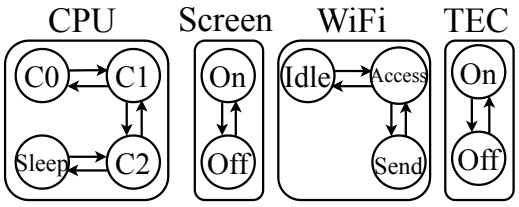


Fig. 7. Hardware status and state transitions in CAPMAN.

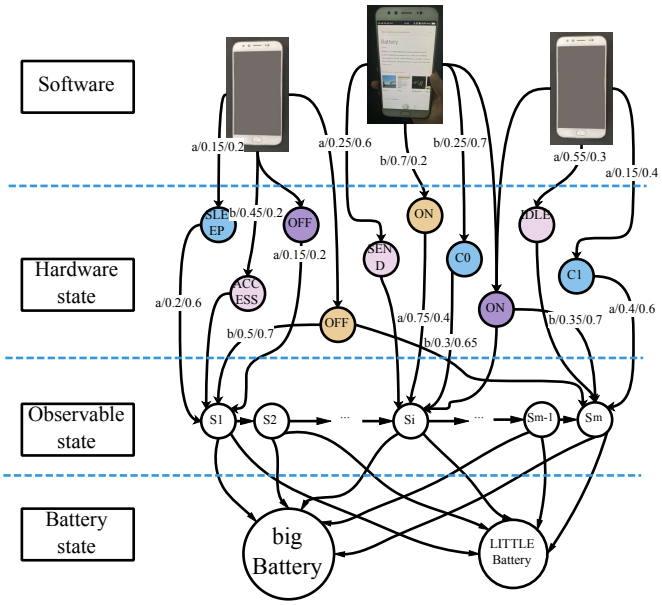


Fig. 8. The Markov Decision Process in CAPMAN. The phone is waked up to receive a Wikipedia update.

Power Model. Power modeling of smartphone components, especially those major energy-consuming ones, has been extensively studied in previous research. Thus 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 II. The CPU power consumption is linearly related to its utilization given a specific frequency level [36]. The screen power consumption depends on the brightness [7]. The power consumption of WIFI is piece-wise linearly related to the packet rate [44]. The TEC power can be derived based on its operating current I [16], which can be obtained by Equation (1).

B. System Modeling and Profiling

Figure 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 Figure 7 and the system call vector [32], 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 $T(\{\text{SLEEP}, \text{OFF}, ..., \text{big}\}, a, \{C0, \text{ON}, ..., \text{LITTLE}\})$, which gives the probability that the phone is awake, and $R(\{\text{SLEEP}, \text{OFF}, ..., \text{big}\}, a, \{C0, \text{ON}, ..., \text{LITTLE}\})$ 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. 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 $G_{\mathcal{M}} = \{\text{V}, \Lambda, E, \psi, p, r\}$, 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 is the 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 Figure 5, respectively. 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.

C. Runtime Calibration

For battery scheduling, the MDP graph $G_{\mathcal{M}}$ allows CAPMAN to find future battery states. 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 [24] 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 simply the search and solving the entire MDP graph, we propose to use a structural similarity method [38] 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, thus the out-neighbors of a state node are always action nodes, whereas those of an action node are always state nodes. Before we present our structural similarity definition and runtime calibration algorithm, let us first define the state similarity δ_{S} and the action similarity δ_{A} in Equation (2):

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

State/Action Similarity Recursion. To define similarity, like SimRank [23], we have δ_S and δ_A from state nodes and action nodes. It is easy to understand that two nodes are similar if and only if their neighbors are similar. We repeat this process to find all similarities. As the base case, we define

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

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 movers distance (EMD) as in Equation (4):

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: $d \leftarrow \mathrm{EMD}\left(p_a, p_b; G_M, \mathbf{1} - \mathrm{S}\right)$
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** $(\sigma^*_{\mathrm{S}}, \sigma^*_{\mathrm{A}}) \leftarrow (\boldsymbol{S}, \boldsymbol{A})$

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

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

D. Recursive Computation

Algorithm 1 shows the iterative algorithm for computing σ^*_{S} and σ^*_{A} by repeating the recursion. The algorithm computes similarity between state 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 [40] 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 $\mathcal{M} = (S, A, T, R)$, Algorithm 1 requires $\Theta\left(|V|^2 + |\Lambda|^2\right) = O\left(|\mathrm{S}|^2|\mathrm{A}|^2\right)$ space to store S and A. 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 predefined precision ϵ (e.g., ϵ = 0.01), SSP is guaranteed to terminate in $O\left(\frac{1}{\epsilon^2} \cdot \left(K^2_{\max} + K_{\max} \log K_{\max}\right)\right) = O\left(10^6\right)$, which is almost constant. To further speed up the computation, we use Dijkstras algorithm with a Fibonacci heap. Each iteration of Algorithm 1 (Lines 3-7) makes $\Theta\left(|\Lambda|^2\right)$ calls to SSP. Computing the Hausdorff distances takes $\Theta\left(|V|^2 L^2_{\max}\right)$ time, where $L_{\max} \leq |A|$ is the maximum out-degree of state nodes in GM. The overall time cost is therefore $O\left(N \cdot |S|^2|A|^2 K^2_{\max}/\epsilon^2\right)$, which is linear to *N*, the number of iterations before convergence.

Uniqueness and Stability. We now prove that σ^*_{S} and σ^*_{A} are well-defined by showing that Algorithm 1 always terminates. Let $S^{(k)}$ and $A^{(k)}$ be versions of the matrices **S** and **A** after the k-th execution of Lines 3-7 of Algorithm 1 (k = 1, 2, $\cdots$). In addition, 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 S^k is always less than S^{k+1} and S^{k+1}. Meanwhile, when introducing discount factor ($0 < C_S, C_A < 1$), we have $S^i \in [0,1]$ and $A^i \in [0,1]$ for all $i > 0$. Thus, the k increases to ∞, we have

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

Thus, Algorithm 1 always terminates correctly with the unique solution (σ_S^*, σ_A^*).

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

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

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.,

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

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

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

Now, consider the optimal value functions $\mathcal{V}^*$ and $\mathcal{Q}^*$ under the optimal policy π^*. The Bellman equations [34] 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 the proposed distance measures, σ_S^* and σ_A^*, can be used to bound the difference between the optimal values. Therefore, we have:

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

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 $r \in [0,1]$, $\sum_{k=0}^{\infty} \rho^k = \frac{1}{1-\rho}$. Detailed proof is skipped due to space limit. That is, if we relax the similarity discount factor and let $\rho = 0.05$, 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.

E. 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. Figure 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 Figure 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 V, CAPMAN actually favors LITTLE battery due to frequently wake TEC to cool the phone actively.

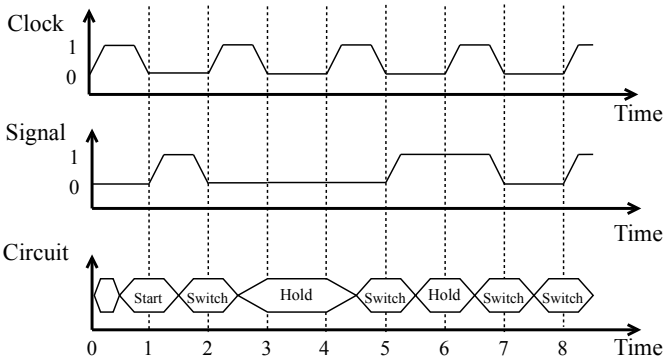


Fig. 9. Timing diagram for battery switching.

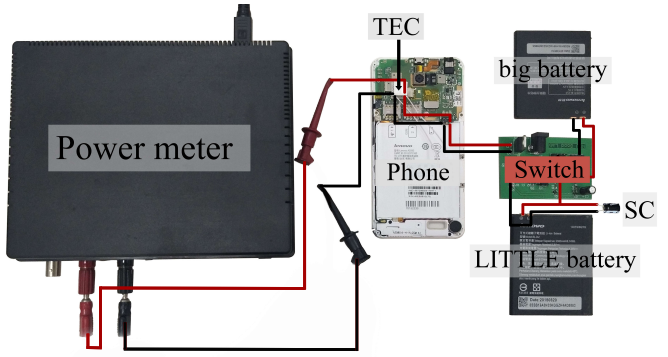


Fig. 10. The full prototype of big.LITTLE battery support for mobile devices. We implement a switch facility to convert CAPMAN algorithm output into battery switch signal. For the LITTLE battery, the voltage is unstable in spike. We installed a supercapacitor to boost and filter the LITTLE output, such that CAPMAN can have a reliable power supply.

IV. IMPLEMENTATION

In this section, we detailed our CAPMAN prototype in implementation, as illustrated in Figure 10.

Profile and Monitor. The approximation in CAPMAN abstracts patterns, operations, and user interactions from the software level into a set of device power states. As illustrated in Figure 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 II) in Android 5.0.1, the WIFI is switched to a high power state. These power profiles are

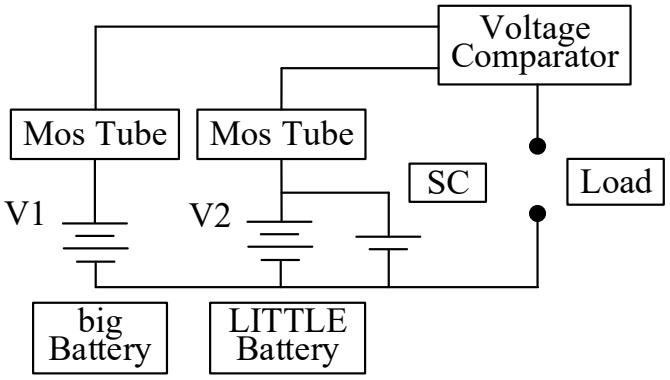


Fig. 11. Circuit diagram of the battery switcher.

obtained offline, measured from a multimeter. In all, the whole system are connected and measured in Figure 10.

For active cooling, CAPMAN adds a TEC device, covering the CPU spot in implementation. Without making the system further complicated, we profile our TEC chip offline, and always power it at its maximum cooling efficiency. Thus, the TEC works with an on/off model in CAPMAN. In the physical setup, TEC is powered on directly from the switch facility when the temperature is higher than 45°C threshold.

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 Figure 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 Figure 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 Figure 11. If received signal flips again, the specified voltage drops to 0.3V, causing the comparator switch to the right Mos Tube, that turning on the LITTLE battery. As such, CAPMAN can manage the battery supply from big.LITTLE batteries.

V. Results

In this section, we evaluate the performance of CAPMAN by prototyping it onto our physical testbed and workloads.

Hardware/Software Setup. We build a physical test platform to evaluate the runtime performance of CAPMAN. The testbed extends our prototype in Figure 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 three phones, with CPU frequency ranging from 1040kHz to 2000kHz, with installed Android ROM version 5.0-7.1. The detailed power profile of each device used in our test is shown in Table III.

Workloads and Traces. We verified the performance of CAPMAN using real world workloads and traces. Each benchmark 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.

TABLE III
Average power costs of all hardware states in tested devices.

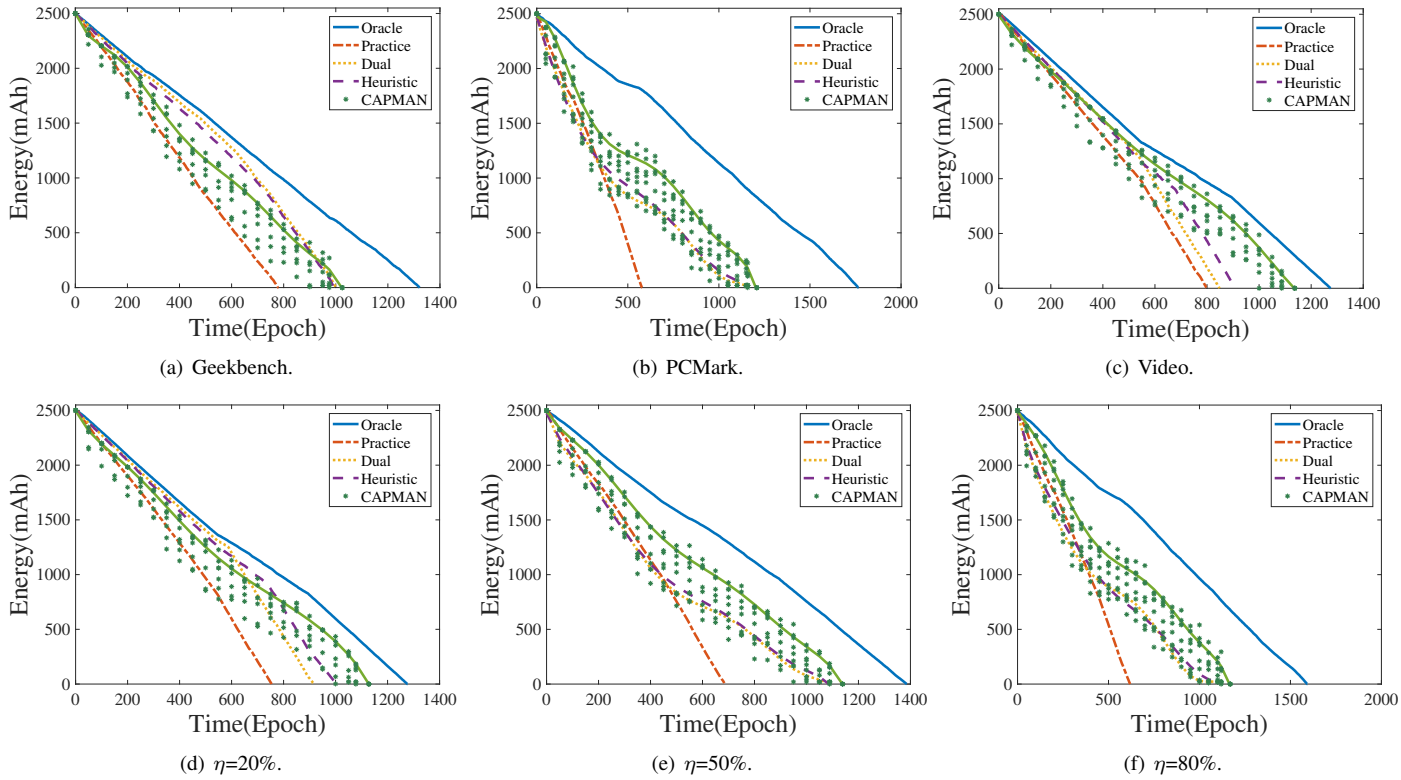
Hardware	CPU [4]				Screen [29]	
Status	C0	C1	C2	Sleep	Off	On
Power(mW)	612	462	310	55	22	790
Hardware	WiFi [20]				TEC [36]	
Status	Idle	Access	Send		Off	On
Power(mW)	60	1284	1548		0	29.17

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 utilization-based prediction model in Table II.

One Discharge Cycle Performance. We first plot an one-discharge-cycle performance of CAPMAN, compared to other baselines. Figure 12 illustrates this comparison using six different workloads, namely, *Geekbench, PCMark, Video* and three setups of $\eta - *$ static workloads. The green dots are data collected from multiple simulation experiments and the green line is the fitted curve. It is worth mentioning that, in addition to the typical high power consumption hardwares, CAPMAN also includes the power consumption of running TEC here.

In Geekbench, the system is fully occupied with CPU and memory intensive jobs, leading to a fully occupied system. In this workload, CAPMAN works in a way similar to *Dual* and *Heuristic*, (Figure 12(a)). It is because CAPMAN spends extra on maintaining the MDP representation of the system and constantly updates the similarity, which is unnecessary in stationary workloads like Geekbench. However, CAPMAN can still prolong 50% more service time than the *Practice*. For the PCMark workload, as the system is not fully utilized. CAPMAN gradually learns the state behavior, and reacts to the right battery decision. Thus, though the energy drains fast in the beginning, CAPMAN improves the performance by 21.3%, 25.7%, compared to *Dual* and *Heuristic* at last (Figure 12(b)). When the software demand becomes dynamic, such as Video, CAPMAN can significantly outperform other baselines, namely 53.27%, 55.08%, and 67.1% longer service



(a) Geekbench. (b) PCMark. (c) Video.

(d) η=20%. (e) η=50%. (f) η=80%.

Fig. 12. Performance comparison of CAPMAN and different baselines. (The green dots are data collected from multiple simulation experiments for CAPMAN and the green line is the fitted curve.)

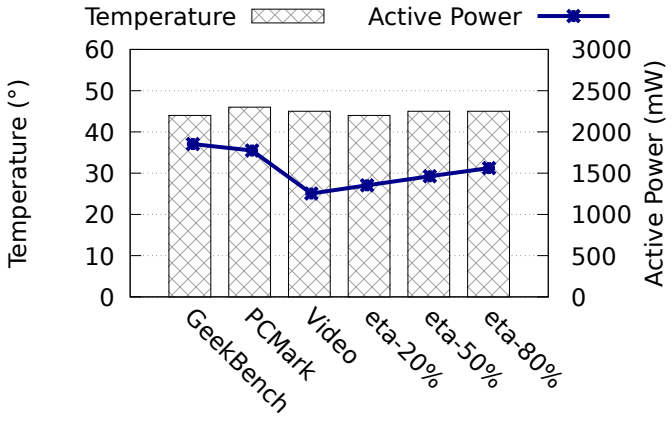


Fig. 13. Cooling and active power consumption of different workloads.

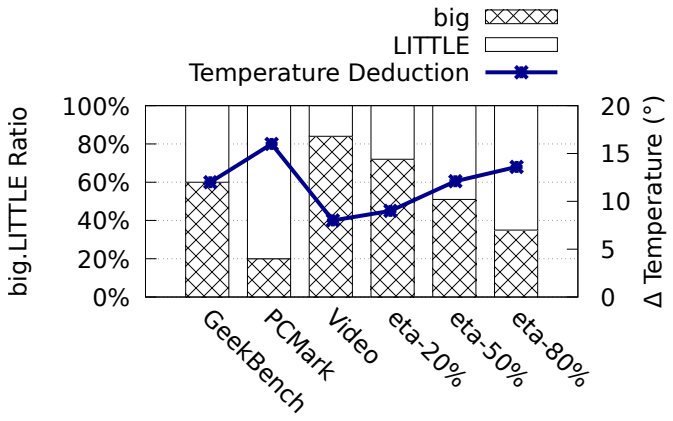


Fig. 14. The relationship between big.LITTLE ratio and temperature reduction in different workloads.

time than *Heuristics, Dual,* and *Practice*, respectively (Figure 12(c)). Note that, it is also very close to the theoretical best–*Oracle*, within 9.6% less service time.

When executing a mixed workload (Figure 12(e)-(f)), CAPMAN extends 76%, 105%, and 114% more service time than *Practice*. On average, CAPMAN can double the service time by smartly scheduling heterogeneous dual batteries, compared to a single battery with the same capacity. CAPMAN shows a closer curve than all the baselines to the offline optimal *Oracle* at most times.

Cooling and Active Power Management. Figure 13 presents how CAPMAN handles cooling when the active power varies. In all workloads, CAPMAN can maintain the temperature around 45 degree, as predefined. When the active power reaches to 2300mW and the whole system works at its designed highest utilization, CAPMAN boots up the TEC to reduce the surface temperature. When the workload is less intensive, such as the Video workload, the active power is much smaller, due to less consumption from both mobile phone and TEC cooling.

To further illustrate the performance, here we show the performance between temperature reduction, which compares to no TEC, and the ratio of activation time between big and LITTLE batteries in Figure 14. CAPMAN selects the

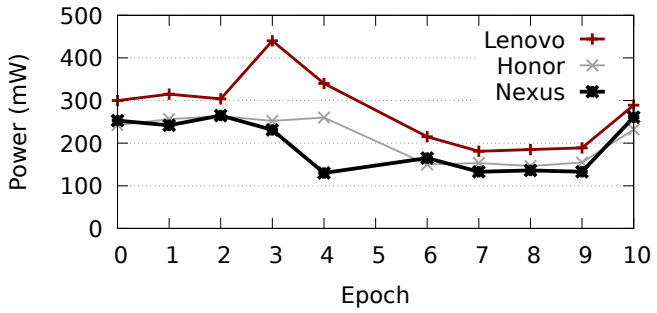


Fig. 15. A snapshot of CAPMAN on different phones.

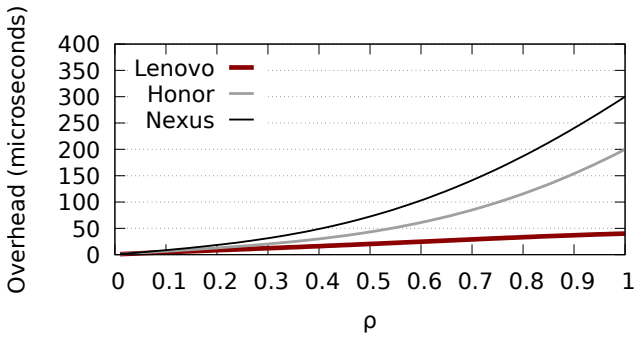


Fig. 16. The impact of the discount factor ρ.

appropriate battery based on the current state of the system. It is clear that when LITTLE battery takes in charge, it means more dynamic power surges arrive in the system, which could be either a CPU intensive job, or the whole system boosts up. In either 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.

Stability and Scalability. We test CAPMAN onto three different phones, namely Nexus, Honor, and Lenovo. One snapshot of runtime performance is shown in Figure 15. CAPMAN shows similar active power management between phones, under the same workload traces, 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 a ρ too high could also lead to more recursion calculation in Algorithm 1. We show the impact of ρ on the computation overhead in Figure 16. This impact varies from different phones, as the computation resource varies. However, all curves show an exponential behavior when ρ increases. When ρ reaches 1, the overhead reaches to 300 microseconds in Nexus, which makes the battery control unstable. Thus, for each device, CAPMAN needs to recalibrate to find a suitable configuration.

VI. 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 cooling and power management as follows.

Battery Scheduling. Battery life is a critical performance and user experience metric on mobile devices [30]. Falaki et al. [19] develop the prototype that manages multiple batteries for electric vehicles. Jin Lu et al. [26] further provide a more detailed controller design for battery charging. There are many work that leverage hybrid batteries in different areas such as EV [33], datacenters [17], [43], smart grid [11], etc. Our paper is one of the first attempts to provide system support for battery scheduling, based on the SDB idea from [8].

Power and Cooling Management. Power management has been studied for a long time in battery-powered device domain. Carroll et al. [12] propose a basic power modeling set. Y. Hu et al. [22] 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. [28] present a comprehensive analysis of energy management strategies (EMSs). Yuvraj Agarwal et al. [5] present the Cell2Notify power management framework to reduce the high energy consumption of WiFi. Our work focuses on battery management to prolong smartphone service times. To reduce the temperature, plenty of research has been devoted [18], [31] for thermal modeling. In this field, the work from Dai [16] 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. Our work focuses on a more practical aspect for cooling and active power management: *exploiting the existing battery chemistries to harvest more service times.*

VII. Conclusion

Modern smartphone design is constrained by its power and thermal wall. Battery engineering suggests a pack of big.LITTLE batteries can perform better than one single battery with the same capacity. To provide the system support for big.LITTLE batteries, 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 an online algorithm with a proved worst-case O($\frac{1}{1-\rho}$)-competitiveness performance. We implement CAPMAN on popular smartphones and find it can significantly extend (114%) the service time while maintaining the ambient temperature. Compared to state-of-the-practice baselines, CAPMAN shows an average 55.08% performance gain with 53.27% less power use.

VIII. Acknowledgement

This work is supported by NSFC funding No. 61702250 and 61702329, MST National RD Key project No. 2018YFB14043033, and Jiangxi Thousands of Talents project No. jxsq106018. We would like to thank Prof. Hao Wang on reviewing the theory proof and all anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] Agilent 34410a multimeter. https://www.keysight.com/. Accessed December 4, 2019.

[2] Battery university. https://batteryuniversity.com/. Accessed December 4, 2019.

[3] Pcmark. https://benchmarks.ul.com/zh-hans/pcmark10. Accessed December 4, 2019.

[4] Ahmed Abdelmotalib and Zhibo Wu. Power consumption in smartphones (hardware behaviourism). *International Journal of Computer Science Issues (IJCSI)*, 9(3):161, 2012.

[5] Yuvraj Agarwal, Ranveer Chandra, Alec Wolman, Paramvir Bahl, Kevin Chin, and Rajesh Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 179–191. ACM, 2007.

[6] Aakash Agrawal, Krunal Shah, Amit Kumar, and Ranveer Chandra. Battery scheduling problem. In *International Conference on Theory and Applications of Models of Computation*, pages 1–12. Springer, 2019.

[7] Farhan Azmat Ali, Pieter Simoens, Tim Verbelen, Piet Demeester, and Bart Dhoedt. Mobile device power models for energy efficient dynamic offloading at runtime. *Journal of Systems and Software*, 113:173–187, 2016.

[8] Anirudh Badam, Ranveer Chandra, Jon Dutra, Anthony Ferrese, Steve Hodges, Pan Hu, Julia Meinershagen, Thomas Moscibroda, Bodhi Priyantha, and Evangelia Skiani. Software defined batteries. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 215–229. ACM, 2015.

[9] Avram Bar-Cohen and Peng Wang. On-chip hot spot remediation with miniaturized thermoelectric coolers. *Microgravity Science and Technology*, 21(1):351–359, 2009.

[10] Matthew Bartlett and Brad Sherrill. Battery pack including an emergency back-up battery for use in mobile electronic devices, June 1 2010. US Patent 7,728,549.

[11] Bocklisch and Thilo. Hybrid energy storage systems for renewable energy applications. *Energy Procedia*, 73:103–111.

[12] Aaron Carroll, Gernot Heiser, et al. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, volume 14, pages 21–21. Boston, MA, 2010.

[13] Guy Chemla. Integrated circuit temperature monitoring and protection system, September 8 1998. US Patent 5,805,403.

[14] C-F Chiasserini and Ramesh R Rao. Energy efficient battery management. *IEEE journal on selected areas in communications*, 19(7):1235–1245, 2001.

[15] Victor Chiriac, Steve Molloy, Jon Anderson, Ken Goodson, and Victor Chiriac. A figure of merit for smart phone thermal management. *A Figure of Merit for Smart Phone Thermal Management*, page 16, 2017.

[16] Yuting Dai, Tao Li, Benyong Liu, Mingcong Song, and Huixiang Chen. Exploiting dynamic thermal energy harvesting for reusing in smartphone with mobile applications. In *ACM SIGPLAN Notices*, volume 53, pages 243–256. ACM, 2018.

[17] Wang Di, Chuangang Ren, Anand Sivasubramaniam, Bhuvan Urgaonkar, and Hosam Fathy. Energy storage in datacenters: What, where, and how much? *Acm Sigmetrics Performance Evaluation Review*, 40(1), 2.

[18] Begum Egilmez, Gokhan Memik, Seda Ogrenci-Memik, and Oguz Ergin. User-specific skin temperature-aware dvfs for smartphones. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1217–1220. IEEE, 2015.

[19] Mohamamd Hossein Falaki. *Automating personalized battery management on smartphones*. PhD thesis, UCLA, 2012.

[20] Roy Friedman, Alex Kogan, and Yevgeny Krivolapov. On power and throughput tradeoffs of wifi and bluetooth in smartphones. *IEEE Transactions on Mobile Computing*, 12(7):1363–1376, 2012.

[21] Koji Hasebe, Tatsuya Niwa, Akiyoshi Sugiki, and Kazuhiko Kato. Power-saving in large-scale storage systems with data migration. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 266–273. IEEE, 2010.

[22] Y. Huh. Future direction of power management in mobile devices. In *IEEE Asian Solid-State Circuits Conference 2011*, pages 1–4, Nov 2011.

[23] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[24] Nicholas K Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, pages 752–757. Citeseer, 2005.

[25] Allan D Kraus and Avram Bar-Cohen. Thermal analysis and control of electronic equipment. *Washington, DC, Hemisphere Publishing Corp., 1983, 633 p.*, 1983.

[26] Jin Lu, Todd Scott Kelly, and Lee Cheung. Battery management system and method, November 12 2013. US Patent 8,583,955.

[27] Dongsheng Ma and Rajdeep Bondade. Enabling power-efficient dvfs operations on silicon. *IEEE Circuits and Systems Magazine*, 10(1):14–30, 2010.

[28] Clara Marina Martinez, Xiaosong Hu, Dongpu Cao, Efstathios Velenis, Bo Gao, and Matthias Wellers. Energy management in plug-in hybrid electric vehicles: Recent progress and a connected vehicles perspective. *IEEE Transactions on Vehicular Technology*, 66(6):4534–4549, 2016.

[29] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 317–328. ACM, 2012.

[30] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 317–328, New York, NY, USA, 2012. ACM.

[31] Francesco Paterna, Joe Zanotelli, and Tajana Simunic Rosing. Ambient variation-tolerant and inter components aware thermal management for mobile system on chips. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2014.

[32] Abhinav Pathak, Y Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, pages 153–168. ACM, 2011.

[33] Ziyou Song, H Hofmann, JQ Li, Xuebing Han, and Minggao Ouyang. Optimization for a hybrid energy storage system in electric vehicles using dynamic programing approach. *Applied Energy*, 139:151–162.

[34] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[35] FL Tan and SC Fok. Thermal management of mobile phone using phase change material. In *2007 9th Electronics Packaging Technology Conference*, pages 836–842. IEEE, 2007.

[36] FL Tan and SC Fok. Methodology on sizing and selecting thermoelectric cooler from different tec manufacturers in cooling system design. *Energy conversion and management*, 49(6):1715–1723, 2008.

[37] Eric Wang. Mobile electronic devices with integrated personal cooling fan, August 11 2011. US Patent App. 12/919,473.

[38] Hao Wang, Shaokang Dong, and Ling Shao. Measuring structural similarities in finite mdps. In *International Joint Conference on Artificial Intelligence*, pages 3684–3690, 2019.

[39] V Wienert, H Sick, et al. Local thermal stress tolerance of human skin. *Anasthesie, Intensivtherapie, Notfallmedizin*, 18(2):88–90, 1983.

[40] S JEWELL William. Optimal flow through networks. *Operations Research*, 10:476–499, 1962.

[41] Qing Xie, Jaemin Kim, Yanzhi Wang, Donghwa Shin, Naehyuck Chang, and Massoud Pedram. Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 242–247. IEEE, 2013.

[42] Fengyuan Xu, Yunxin Liu, Qun Li, and Yongguang Zhang. V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 43–55, Lombard, IL, 2013. USENIX.

[43] Yang, Hu, Hongbin, Sun, Juncheng, Gu, Longjun, Liu, Tao, and Li and. Heb: Deploying and managing hybrid energy buffers for improving datacenter efficiency and economy.

[44] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114. ACM, 2010.