Energy-minimizing workload splitting and frequency selection for guaranteed performance over heterogeneous cores

Aditya Priya
adityapriya@iisc.ac.in
Indian Institute of Science Bangalore

Rajiv Choudhury*
rajiv.choudhury@iiitg.ac.in
Indian Institute of Information Technology Guwahati

Sujay Patni†
sujaypatni@gmail.com
Birla Institute of Technology & Science, Pilani

Himkant Sharma‡
himkant@kgpian.iitkgp.ac.in
Indian Institute of Technology Kharagpur

Moonmoon Mohanty
moonmoonm@iisc.ac.in
Indian Institute of Science Bangalore

Krishnasuri Narayanam
knaraya3@in.ibm.com
IBM Research, India

UmaMaheswari Devi
umamadev@in.ibm.com
IBM Research, India

Pratibha Moogi
pratibha.moogi@ibm.com
IBM Research, India

Preetam Patil
preetampatil@iisc.ac.in
Indian Institute of Science Bangalore

Parimal Parag
parimal@iisc.ac.in
Indian Institute of Science Bangalore

ABSTRACT

Heterogeneous computing involves CPU architectures that support more than one core type, and it aims to achieve energy efficiency while meeting the performance guarantees. This aim can be achieved by the operating system or the on-chip driver by exploiting the differential power-performance trade-off that heterogeneous cores offer. We characterize the power-performance trade-off for an Intel CPU with heterogeneous cores and provide a mathematical framework to study heterogeneous computing. In particular, we provide probabilistic workload split and operating frequency for all active cores that allow workload execution with minimal carbon emissions. We support the analytical findings with experimental evaluations for a few representative workloads. As compared to the default Linux frequency governors, our scheme can reduce the energy-delay product by up to 80%.

CCS CONCEPTS

• **Computer systems organization** → **Heterogeneous (hybrid) systems**; • **Hardware** → **Platform power issues**; • **Software and its engineering** → **Scheduling**; • **Computing methodologies** → **Modeling and simulation**.

This research was conducted when Rajiv Choudhury, Sujay Patni, and Himkant Sharma, were doing an internship at the Indian Institute of Science Bangalore.

E-Energy '24, June 04–07, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0480-2/24/06
https://doi.org/10.1145/3632775.3661968

KEYWORDS

Heterogeneous cores, energy optimization, workload scheduling, mean latency guarantees

ACM Reference Format:
Aditya Priya, Rajiv Choudhury, Sujay Patni, Himkant Sharma, Moonmoon Mohanty, Krishnasuri Narayanam, UmaMaheswari Devi, Pratibha Moogi, Preetam Patil, and Parimal Parag. 2024. Energy-minimizing workload splitting and frequency selection for guaranteed performance over heterogeneous cores. In *The 15th ACM International Conference on Future and Sustainable Energy Systems (E-Energy '24), June 04–07, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3632775.3661968

1 INTRODUCTION

The Information and Communications Technology (ICT) sector's current energy usage is close to 10% of global electricity consumption and is projected to increase to 20% by 2030 [21]. A significant fraction of ICT energy consumption is by data centers [8]. To combat the adverse impact on environmental sustainability caused by the above growth in energy demand, novel solutions are being proposed and developed by different players targeting different parts of the ICT space. While data center providers attempt to compensate for their carbon footprint through alternative targets such as Net-Zero, which focuses on the use of renewable energy, chip and server manufacturers are exploring energy-efficient hardware architectures. Despite efficiency improvements in CPU power, it is the most dominant components of the total server power consumption. It accounts for 58% of dynamic server power and 33% of total server power [11, 19, 29]. Further, cooling and provisioning costs are proportional to the total server power. Therefore, we focus on ways to reduce CPU power in this work. We note that memory power consumption is another important contributor to the server energy consumption. However, memory frequency and voltage are

constants during operation and are not load-dependent. Therefore, we do not consider energy consumption due to memory usage.

Computational workloads are time-varying and the service requirements are seldom constant [38]. Therefore, CPUs are rarely used at maximum utilization [11]. CPUs consume power when idling between busy phases, leading to energy wastage and lack of *energy proportionality*[1] that has become a key design goal for energy-efficient computing. Though it is possible to work around the lack of energy proportionality by putting CPUs to sleep when idle, low-power CPU sleep states are useful only to an extent. As observed in [39], CPUs need to be awakened quite often, and there is latency and power cost involved in the transitions to and from sleep states. Another approach to get better energy proportionality in modern CPUs is via using Dynamic Voltage and Frequency Scaling (DVFS) [44]. Using DVFS, one can dynamically slow down the clock frequency and supply voltage to reduce energy consumption. This can also reduce idle times, by reducing the intervals when the CPU is idle. However, reducing clock frequency can increase task completion times and there is latency involved with dynamic frequency switching. Further, frequent changes in voltage and frequency can be significantly detrimental to hardware reliability [51].

Fundamentally, there are two sources of non-proportionality: (a) CPUs consume static power primarily corresponding to leakage power [27], when CPUs are idle and not asleep, and (b) CPUs' dynamic power consumption while active is non-linear in the core frequency, especially at low utilization of 10-50% [11]. In order to improve power efficiency at all utilization levels of practical interest, servers offering high dynamic range and linear deviation (with workload) are desired. One proposed approach [56] for extending the dynamic range while maintaining high linearity is through computing architectures consisting of CPUs with heterogeneous cores having different power-performance trade-offs. A multi-core CPU with functionally non-identical cores is called a *heterogeneous or hybrid multi-core processor* (HMP). A HMP typically has more than one core micro-architecture in a single die, allowing distinct sets of cores to operate at different processing speeds, while sharing the same ISA, thus enabling performance and energy efficiency. Intel Core Processors, 12th generation onward [2], are examples of HMP CPUs, combining performance and efficiency cores (P-cores and E-cores) in the same package. P-cores support high clock speeds to maximize single-thread performance and responsiveness for compute-intensive workloads. E-cores are slower, consume less power, and are meant for background tasks. These Intel CPUs also support DVFS for both core types.

Realizing the energy-saving potential of HMPs requires a joint allocation and scheduling of workloads to all the cores. This entails (a) finding the workload allocation to the cores and (b) core frequency selection for each of the active cores. The overall objective is to meet the workload service level objectives (SLO's) while minimizing the energy consumption. Such an allocation scheme needs to be aware of the power-performance trade-off for the CPU cores being used, and the workload characteristics including arrival rate and service requirements. This can be achieved by (a) CPU characterization preferably through an analytical model for power as a function of frequency, supply voltage, and utilization, and (b) workload characterization through an analytical model for inter-arrival times and service requirements. Together these two models can predict the power-performance trade-off and help in making fast scheduling decisions. In this work, we focus on workloads that do not change considerably during one scheduling span and focus on determining (a) the number of sleeping and active cores, (b) the operating frequency selection at active cores, and (c) the workload split on active cores through thread assignments, such that service level objectives are met while minimizing power consumption.

Energy and performance-aware scheduling for hybrid cores can become significant in cloud environments due to the increasing adoption of *serverless computing*. Serverless systems are evolving to support heterogeneous workloads with varying processing characteristics and performance requirements. Efforts are already underway to instantiate such platforms over heterogeneous computing systems for improving performance and reducing resources, energy, and cost [20, 34]. Intel also recently announced plans for a modular SoC architecture that can support both P and E cores using compute chiplets for cloud workloads that need more than one CPU design [22]. The scheduling policies derived in this work can be extended to determine the right number of container replicas to enable different services on the right type of hardware based on the characteristics of the incoming workload. Our focus is on predictable workloads that remain unchanged for one scheduling span, e.g. modern batch processing workloads in cloud [36] that can consist of data and image analytics tasks as part of application workflow orchestration frameworks [23] and in HPC environments [55]. We note that our approach can be extended to dynamic, real-world scenarios by learning the changes to the workload at runtime, and this study can guide the design of efficient heuristics. We also note that our study can be utilized on public clouds, where workloads with similar characteristics can be aggregated and scheduled.

1.1 Related Work

The benefit of single-ISA heterogeneous multi-core architecture for energy optimization was first established in [33] by switching a multi-phased application with different phase execution characteristics among cores of a heterogeneous multi-core system composed of different generations of Alpha ISA processors. Heterogeneous scheduling has been studied in a variety of other across-chip heterogenic contexts. CPU-GPU collaboration for high-performance computing is explored in [18, 38, 41]. Multiple layers of parallelism exposed in modern hardware with symmetric multi-core processors are exploited in [14]. Maximizing the average instructions per cycle of a set of applications running simultaneously on an asymmetric multi-core processor system is studied in [45]. The tradeoff between average power reduction and SLA degradation is studied in [24] when a higher fraction of incoming tasks are scheduled on less performant but more power-efficient servers while letting idle servers sleep. In contrast, this work focuses on heterogeneity within a chip.

Theoretically, it has been shown that classical load-balancing policies designed for homogeneous servers perform poorly in the heterogeneous setting [10, 50, 54]. Existing work on scheduling over heterogeneous servers considers fixed service rates [15, 35, 37, 52]. Probabilistic splitting of workloads for performance optimization in

[1] *Energy proportionality* implies that the system's energy consumption should remain proportional to the utilization.

the heterogeneous setting was considered in [52]. These works did not consider service rate control and energy consumption. Energy-aware scheduling in Linux kernels using DVFS for multiple CPUs was proposed in [3]. The proposed scheduling was shown to be sub-optimal for a heterogeneous multi-core architecture [48, 49]. To the best of our knowledge, ours is the first work to combine scheduling on multicore CPUs with intra-chip heterogeneity and DVFS for optimizing power while meeting latency guarantees.

1.2 Key contributions

The key contributions of this work are as follows.

(1) We propose a power and performance model for CPUs with heterogeneous cores that is consistent with the CMOS CPU architecture of current generations in Section 2.
(2) We validate the system model through experimental characterization on an Intel X86-64 CPU (core i9-13900) with 8 P-cores and 16 E-cores in Section 4.1.
(3) We formulate the configuration problem of (a) the workload distribution among the cores, and (b) their operating frequencies in Problem 1 that minimizes the aggregate power consumption while meeting a mean sojourn time guarantee for the workload at each core.
(4) We propose HEMP—*Heterogeneity enabled Energy-Minimizer with Performance constraints*—an optimal analytical solution to the configuration problem in Theorem 3 which has strong theoretical guarantees, and is easily implementable in practical systems.
(5) We present numerical results in Section 4.2, and validate HEMP with experiments on a CPU with heterogeneous cores in Section 4.3.
(6) We compare the performance of HEMP with Linux frequency governors combined with the default CPU scheduler in Section 4.3.3 for image-processing and machine-learning inference workloads.

Notation: We denote the set of all positive integers by $\mathbb{N}$, the set of first n positive integers by $[n]$, the set of non-negative reals by $\mathbb{R}_+$, the set of probability measures on a finite set A by $\mathcal{M}(A) \triangleq \left\{p \in [0,1]^A : \sum_{a\in A} p_a = 1\right\}$.

2 SYSTEM MODEL

We consider a compute system with a single class of computational tasks, that can be offloaded to one of the N available cores. We model the arrival of computational tasks as a Poisson process with an aggregate arrival rate $N\lambda$. We consider a simple probabilistic load-balancing scheme where each arriving task is randomly assigned by the OS scheduler to one of the cores independently with an identical distribution (*i.i.d.*) where the common probability mass function (PMF) is denoted by $\gamma \in \mathcal{M}([N])$. It follows that γ_n is the thinning probability for arrivals to core n. If a core is busy, the task is queued in a per-core buffer[2] at the OS scheduler. We assume an arbitrarily large buffer serviced on a first come first served (FCFS) basis.

[2]In practice, the OS scheduler may have a common priority queue per CPU with a sophisticated priority scheme. However, modeling such queues is complex, and hence we assume a per-core queue for analytical tractability.

We consider the case of heterogeneous cores of two classes. The mutually exclusive sets of performance and efficiency cores are denoted by $\mathcal{N}_p \subseteq [N]$ and $\mathcal{N}_e = [N] \setminus \mathcal{N}_p$ respectively. Since each core is of type performance or efficiency, we denote the type of core n by

$$c_n \triangleq p\mathbb{1}_{\{n\in\mathcal{N}_p\}} + e\mathbb{1}_{\{n\in\mathcal{N}_e\}}. \tag{1}$$

We denote the number of performance and efficiency cores by $N_p \triangleq \left|\mathcal{N}_p\right|$ and $N_e \triangleq |\mathcal{N}_e| = N - N_p$, respectively.

2.1 Service time and requirement

We model the service requirements for computational tasks as an *i.i.d.* random sequence with a common exponential distribution having unit mean on a core with a unit task service rate. The task service rate μ_n at a core $n \in [N]$ for a given workload is proportional to the core frequency $f_n \in \mathcal{F}_{c_n}$, where $\mathcal{F}_{c_n}$ is the set of allowable frequencies for a core n. Hence the task service time is distributed exponentially, the service rate is proportional to the frequency, and the proportionality relation is given by

$$\mu_n \triangleq \alpha_{c_n} f_n. \tag{2}$$

The heterogeneity of the cores is reflected in the proportionality constant α_{c_n} that depends on the core type c_n. The performance cores have a larger proportionality constant for the task completion rate, i.e. $\alpha_p > \alpha_e$. That is, a performance core works faster than an efficiency core for the same operating frequency. Under this model, the service time distribution for computational tasks on core n with frequency f_n is exponential with a rate μ_n. We denote the limiting average of sojourn time by $\bar{W}_n$ averaged over all tasks joining a core n, and consider the guarantee that the mean sojourn time doesn't exceed a threshold w at any core n.

2.2 Power consumption

The average power consumption at a core in a time duration depends on whether the core was idle or working during this duration. If the core is idle for the entire time duration, then the average power consumed by the core n working at frequency f_n, is obtained by adapting the results of [47] and is denoted by

$$P_{\text{sta}}(c_n, f_n) \triangleq a_{c_n} V_n (e^{b_{c_n} V_n} - 1), \tag{3}$$

for positive core-dependent constants $a_p, b_p, a_e, b_e > 0$. If the core n operating at frequency f_n is working for the entire time duration, then the *additional* average power consumed is similarly adapted from the results of [47],

$$P_{\text{dyn}}(c_n, f_n) \triangleq \beta_{c_n} V_n^2 f_n, \tag{4}$$

for positive proportionality constants β_p and β_e which represent the effective capacitance of the CMOS system. We note that the two powers at core n are functions of the operating frequency f_n. However, these functions include undetermined variable voltage V_n. This is because voltage V_n is set by the CPU's internal governor to meet stability criteria depending on the operating frequency f_n. The exact voltage-frequency map is discussed in detail in Section 4.1. We note the following property.

Property 1. The operating voltage $V_n = V_{c_n}(f_n)$ at any core n is a convex non-decreasing function of operating frequency f_n.

CPU architectures also define sleep power states $\{C_1, \ldots, C_k\}$ as described in Section A. For simplicity of presentation, we focus on a single sleep state—the deepest sleep state C_k with the average power consumption denoted by $P_{\text{sleep}}(c_n)$ for a core n. Typically, $P_{\text{sleep}}(c_n)$ is significantly lower than the static power $P_{\text{sta}}(c_n, \min \mathcal{F}_{c_n})$. However, the core can't immediately start working when in a sleep state, and there is a non-negligible delay in waking a core from deep sleep. In the rest of the manuscript, we will assume that the cores that are sent to sleep state are decided at the beginning of the schedule, and the sleeping cores are not woken up for the duration of the schedule. In contrast, we note that an idling core can start working as soon as a task arrives.

We denote the limiting time average of power consumption at core n by $\bar{P}_n$. We have assumed that a sleeping core is never woken up, and hence $\bar{P}_n = P_{\text{sleep}}(c_n)$ implies that the thinning probability $\gamma_n = 0$. Further, if thinning probability $\gamma_n = 0$ for a core n, then the power consumption is minimized when it is asleep and hence $\bar{P}_n = P_{\text{sleep}}(c_n)$. Thus, we assume that the thinning probability $\gamma_n = 0$ for a core n if and only if $\bar{P}_n = P_{\text{sleep}}(c_n)$.

2.3 Problem formulation

We are interested in finding the thinning probability mass function $\gamma \in \mathcal{M}([N])$ and the sequence of operating frequencies $f \triangleq (f_n : n \in [N]) \in \mathcal{F}_p^{\mathcal{N}_p} \times \mathcal{F}_e^{\mathcal{N}_e}$ for all cores that minimizes the average power consumption aggregated over all cores while meeting the service requirement on limiting average of sojourn time at each core n.

Problem 1. Consider the set of feasible allocations

$$A \triangleq \left\{(\gamma, f) \in \mathcal{M}([N]) \times (\mathcal{F}_p^{\mathcal{N}_p} \times \mathcal{F}_e^{\mathcal{N}_e}) : \bar{W}_n \leqslant w\right\}.$$

Find the optimal allocation

$$(\gamma^*, f^*) \triangleq \arg\min \left\{ \sum_{n \in [N]} \bar{P}_n : (\gamma, f) \in A \right\}. \tag{5}$$

3 ANALYTICAL RESULTS

For a given allocation (γ, f), we note that the task arrival process at each core n is a thinned version of the aggregate homogeneous Poisson task arrival process of homogeneous rate $N\lambda$ and thinning probability γ_n. It follows that at each core n, the task arrival process is an independent Poisson process with a homogeneous rate $\lambda_n \triangleq N\lambda\gamma_n$. If the thinned Poisson arrival rate $\lambda_n = 0$ for any core n, then the power consumption is minimized when it is in sleep state, i.e. $\bar{P}_n = P_{\text{sleep}}(c_n)$. We first focus on the non-sleeping cores.

Definition 1. For any thinning PMF $\gamma \in \mathcal{M}([N])$, we define the set of non-sleeping or active cores with $\mathcal{N}_1 \triangleq \{n \in [N] : \gamma_n \neq 0\}$. Similarly, we define the set of active performance and efficiency cores as

$$\mathcal{N}_{p,1} \triangleq \left\{n \in \mathcal{N}_p : \gamma_n \neq 0\right\}, \quad \mathcal{N}_{e,1} \triangleq \left\{n \in \mathcal{N}_e : \gamma_n \neq 0\right\}. \tag{6}$$

The number of active cores is denoted by $N_1 \triangleq |\mathcal{N}_1|$, and the corresponding notation for active performance and efficiency cores are $N_{p,1} \triangleq \left|\mathcal{N}_{p,1}\right|$ and $N_{e,1} \triangleq \left|\mathcal{N}_{e,1}\right|$.

The service time for each task at an active core n is an independent exponential random variable with rate $\mu_n = \alpha_{c_n} f_n$. It follows that each active core n has an independent $M/M/1$ queue of tasks with the limiting average of task sojourn time [9] averaged over all incoming tasks

$$\bar{W}_n(c_n, f_n, \gamma_n) = \frac{1}{\mu_n - \lambda_n} = \frac{1}{\alpha_{c_n} f_n - N\lambda\gamma_n}. \tag{7}$$

The mean load on this core is $\rho_n \triangleq \frac{\lambda_n}{\mu_n}$, and the limiting average of idle time for this core is $1 - \rho_n$. Hence, the limiting average power consumption is

$$\bar{P}_n(c_n, f_n, \gamma_n) = \left[P_{\text{sta}}(c_n, f_n) + \frac{N\lambda\gamma_n}{\alpha_{c_n} f_n} P_{\text{dyn}}(c_n, f_n)\right] \mathbb{1}_{\{n \in \mathcal{N}_1\}} + P_{\text{sleep}}(c_n) \mathbb{1}_{\{n \notin \mathcal{N}_1\}}. \tag{8}$$

3.1 Optimal frequency selection

We first focus on solving the following sub-problem. Given a thinning probability mass function $\gamma \in \mathcal{M}([N])$, find the optimal feasible frequency allocation[3] $f \in \mathcal{F}_p^{\mathcal{N}_p} \times \mathcal{F}_e^{\mathcal{N}_e}$ that minimizes the average power consumption aggregated over all cores.

Problem 2. Consider a fixed thinning probability mass function $\gamma \in \mathcal{M}([N])$ and the set of feasible frequencies

$$A(\gamma) \triangleq \left\{f \in \mathcal{F}_p^{\mathcal{N}_p} \times \mathcal{F}_e^{\mathcal{N}_e} : (\gamma, f) \in A\right\}.$$

Find the optimal feasible frequency $f \in A(\gamma)$ that solves

$$f^*(\gamma) \triangleq \arg\min \left\{ \sum_{n \in [N]} \bar{P}_n : f \in A(\gamma) \right\}. \tag{9}$$

Lemma 1. *For a given thinning probability mass function $\gamma \in \mathcal{M}([N])$ and any active core $n \in \mathcal{N}_1$, the optimal feasible operating frequency that minimizes the average power consumption at each core n is given by*

$$f_n^*(c_n, \gamma_n) \triangleq \inf \left\{ f_n \in \mathcal{F}_{c_n} : f_n \geqslant \frac{1}{\alpha_{c_n}} \left(N\lambda\gamma_n + \frac{1}{w}\right) \right\}. \tag{10}$$

Proof. See Appendix C.2. □

Remark 1. From Lemma 1, we observe that the optimal frequency selection $f \in \mathcal{F}_p^{\mathcal{N}_p} \times \mathcal{F}_e^{\mathcal{N}_e}$ is completely determined for any given thinning probability mass function $\gamma \in \mathcal{M}([N])$.

Definition 2. We define the minimum average power consumption given a fixed thinning PMF γ as

$$\bar{P}_n(c_n, \gamma_n) \triangleq \bar{P}_n(c_n, f_n^*(c_n, \gamma_n), \gamma_n), \tag{11}$$

where optimal frequency allocation $f^*(\gamma)$ given γ is defined in (10) for all active cores $n \in \mathcal{N}_1$.

Assumption 1. The sets of feasible frequencies for both types of cores are typically different; both are sets of discrete frequencies. We will assume both sets to be continuous for the simplicity of analysis.

[3] Setting frequency of each core individually is not supported on all CPU architectures. However, frequency setting per core type for active cores suffices for optimality, as established in Theorem 2.

Remark 2. Under Assumption 1, the optimal frequency at core n with positive thinning probability γ_n that minimizes average power consumption while satisfying the mean sojourn time guarantee w at this core, is

$$f_n^*(c_n, \gamma_n) = f_{c_n}(\gamma_n) \triangleq \frac{1}{\alpha_{c_n}}\Big(N\lambda\gamma_n + \frac{1}{w}\Big). \tag{12}$$

We observe that $f_{c_n}(\gamma_n)$ is affine in thinning probability γ_n, and increasing in γ_n since $\frac{N\lambda}{\alpha_{c_n}} > 0$.

3.2 Optimal workload splitting

Thus, to find the optimal allocation (f^*, γ^*), we need to find the thinning feasible PMF γ that minimizes the average of aggregate power consumption at all cores.

Problem 3. Consider the optimal operating frequency $f_n^*(c_n, \gamma_n)$ defined in (12), and feasible thinning PMF set

$$\Gamma \triangleq \left\{\gamma \in \mathcal{M}([N]) : f_n^*(c_n, \gamma_n) \in \mathcal{F}_{c_n} \text{ for all } n \in \mathcal{N}_1\right\}.$$

Find the optimal thinning PMF that solves

$$\gamma^* \triangleq \arg\min\left\{\sum_{n\in[N]} \bar{P}_n(c_n, \gamma_n) : \gamma \in \Gamma\right\}, \tag{13}$$

where the minimum average power consumption $\bar{P}_n(c_n, \gamma_n)$ at each core n that meets the mean sojourn time guarantee w is defined in (11).

Remark 3. From Lemma 1, we observe that the optimal allocation (f^*, γ^*) that solves Problem 1 is given by $(f^*(c, \gamma^*), \gamma^*)$, where

$$f^*(c, \gamma) \triangleq (f_{c_n}(\gamma_n) : n \in \mathcal{N}_1) \tag{14}$$

is the optimal frequency allocation that solves Problem 2 for any thinning PMF γ and the optimal thinning PMF γ^* solves Problem 3.

We next discuss how to solve Problem 3, i.e. how to optimally split the workload in order to minimize average aggregate power consumption while meeting the mean sojourn time guarantee at each core.

3.2.1 When all cores are homogeneous and active.

Theorem 1. *Consider the case when $\mathcal{N}_{c_n,1} = [N]$ and Assumption 1 holds. Then, the optimal allocation for all cores $n \in [N]$ is*

$$\gamma_n^* = \frac{1}{N}, \qquad f_n^* = f_{c_n}\Big(\frac{1}{N}\Big) = \frac{1}{\alpha_{c_n}}\Big(\lambda + \frac{1}{w}\Big). \tag{15}$$

The minimum power consumption for N active homogeneous cores is

$$P_{c_n}(N\lambda, N) \triangleq NP_{\text{sta}}\left(c_n, f_{c_n}\Big(\frac{1}{N}\Big)\right) + N\lambda\frac{\beta_{c_n}}{\alpha_{c_n}}\left[V_{c_n}\left(f_{c_n}\Big(\frac{1}{N}\Big)\right)\right]^2. \tag{16}$$

Proof. See Appendix C.4. □

3.2.2 When all cores are homogeneous but not necessarily active.

Theorem 2. *Let $c \in \{p, e\}$ be the core type. Consider the case when $\mathcal{N}_c = [N]$, and Assumption 1 holds. There exists an optimal number of active cores $N_{c,1}^*$ such that an optimal allocation (γ^*, f^*) is*

$$\gamma_n^* = \frac{1}{N_{c,1}^*}\mathbb{1}_{\{n\in\mathcal{N}_{c,1}\}}, \quad f_n^* = \frac{1}{\alpha_c}\Big(\frac{N\lambda}{N_{c,1}^*} + \frac{1}{w}\Big), \quad n \in \mathcal{N}_{c,1}, \tag{17}$$

for any subset $\mathcal{N}_{c,1} \subseteq [N]$ of size $N_{c,1}^*$.

Proof. See Appendix C.6. □

Assumption 2. We will assume real number of active performance cores $N_{p,1} \in [0, N_p]$ and active efficiency cores $N_{e,1} \in [0, N_e]$ to simplify the problem.

Definition 3. Given aggregate arrival rate $N\lambda$ at N_p performance cores, the minimum average power consumption aggregated over all performance cores is defined as

$$P_{\text{tot}}(p, N\lambda, N_p) \triangleq \inf_{x_p\in[0,N_p]} P_{\text{tot}}(p, N\lambda, N_p, x_p).$$

Similarly, for the aggregate arrival rate $N\lambda$ at N_e efficiency cores, the minimum average power consumption aggregated over all efficiency cores is defined as ‘

$$P_{\text{tot}}(e, N\lambda, N_e) \triangleq \inf_{x_e\in[0,N_e]} P_{\text{tot}}(e, N\lambda, N_e, x_e).$$

Remark 4. Since $P_{\text{tot}}(p, N\lambda, N_p, x)$ and $P_{\text{tot}}(e, N\lambda, N_e, x)$ are jointly convex functions of $N\lambda$ and x (see Lemma 5 in Appendix C.5) and $[0, N_p]$, $[0, N_e]$ are convex sets, it follows from Lemma 2 part 2 that $P_{\text{tot}}(c, N\lambda, N_c)$ is convex in $N\lambda$ for $c \in \{p, e\}$.

3.2.3 Heterogeneous cores.

We now return to the optimal allocation in the case of heterogeneous cores. For any thinning PMF $\gamma \in \mathcal{M}([N])$, we can find the sets of active performance cores $\mathcal{N}_{p,1}$ and active efficiency cores $\mathcal{N}_{e,1}$ defined in Definition 1.

Theorem 3. *Under Assumption 1 and Assumption 2, there exists a unique split $(\delta_p^*, \delta_e^*) \in \mathcal{M}(\{p, e\})$, and optimal number of active cores $N_{p,1}^*, N_{e,1}^*$ that determine the optimal allocation (γ^*, f^*) that minimizes the aggregate power consumption while meeting the mean sojourn time guarantees at all N cores. The optimal thinning probability for all active cores is*

$$\gamma_n^* \triangleq \frac{\delta_p^*}{N_{p,1}^*}\mathbb{1}_{\{n\in\mathcal{N}_{p,1}\}} + \frac{\delta_e^*}{N_{e,1}^*}\mathbb{1}_{\{n\in\mathcal{N}_{e,1}\}}, \tag{18}$$

where $\mathcal{N}_{p,1} \subseteq \mathcal{N}_p$ and $\mathcal{N}_{e,1} \subseteq \mathcal{N}_e$ are active cores of size $N_{p,1}^*$ and $N*_{e,1}$ respectively. The operating frequency at each active core is

$$f_n^* \triangleq \frac{1}{\alpha_p}\Big(\frac{N\lambda\delta_p^*}{N_{p,1}^*} + \frac{1}{w}\Big)\mathbb{1}_{\{n\in\mathcal{N}_{p,1}\}} + \frac{1}{\alpha_e}\Big(\frac{N\lambda\delta_e^*}{N_{e,1}^*} + \frac{1}{w}\Big)\mathbb{1}_{\{n\in\mathcal{N}_{e,1}\}}. \tag{19}$$

Proof. See Appendix C.7. □

Definition 4. We can define the following minimizers for any core n and type $c_n \in \{p, e\}$

$$f_{c_n}^* \triangleq \inf_{f\in\mathcal{F}_{c_n}} \frac{P_{\text{sta}}(c_n, f) - P_{\text{sleep}}(c_n)}{\alpha_{c_n} f - \frac{1}{w}} + \frac{\beta_{c_n}}{\alpha_{c_n}} V_{c_n}(f)^2 + N_{c_n} P_{\text{sleep}}(c_n).$$

We can write the minimum values for core n and type $c_n \in \{p, e\}$

$$c^*_{c_n} \triangleq \frac{P_{\text{sta}}(c_n, f^*_{c_n}) - P_{\text{sleep}}(c_n)}{\alpha_{c_n} f^*_{c_n} - \frac{1}{w}} + \frac{\beta_{c_n}}{\alpha_{c_n}} V_{c_n}(f^*_{c_n})^2 + N_{c_n} P_{\text{sleep}}(c_n).$$

Remark 5. We observe from Definition 4 that the minimizing frequencies f^*_p, f^*_e do not depend on the normalized arrival rate λ. Further, these frequencies do not depend on the number of cores if the sleep powers are negligible.

Theorem 4. *Consider the case when Assumption 1 and Assumption 2 hold, and $f^*_p, f^*_e, c^*_p, c^*_e$ be as defined in Definition 4. We define*

$$\lambda_{p,0} \triangleq \frac{N_p}{N}\left(\alpha_p f^*_p - \frac{1}{w}\right), \quad \lambda_{e,0} \triangleq \frac{N_e}{N}\left(\alpha_e f^*_e - \frac{1}{w}\right). \tag{20}$$

Then for all normalized arrival rates $\lambda < \lambda_0 \triangleq \lambda_{p,0} \wedge \lambda_{e,0}$, the optimal workload split is $\delta^*_p = \mathbb{1}_{\left\{c^*_p < c^*_e\right\}}$.

Proof. See Appendix C.8. □

Remark 6. Theorem 4 provides an arrival rate region, in which only one type of core is selected for the entire workload. Further, we know the identical optimal frequency to be used on the active cores as given in Definition 4 which is independent of workload. We also know the number of active cores is x^*_p if $c^*_p < c^*_e$ and x^*_e otherwise. For sufficiently large mean sojourn time guarantees, we empirically observe that $c^*_e < c^*_p$ for the CPU we study.

4 EVALUATION

In this section, we describe the CPU characterization conducted for validating (a) linearity of service rate with core frequency, (b) convexity of operating voltage as a function of core frequency, and (c) dependence of static and dynamic power on the operating voltage. Further, we compute (a) the voltage-frequency relationship, and (b) the model parameters for power-frequency curves for both static and dynamic power. We next conduct numerical studies to obtain the optimal workload split δ^* between P and E cores, and correspondingly, the optimal number of active cores $N^*_{p,1}, N^*_{e,1}$.We conduct experiments on a heterogeneous CPU to show that under the probabilistic workload splitting, the aggregate power is minimized by HEMP. Finally, a comparison with common Linux frequency governors is presented.

4.1 CPU characterization

The test system's CPU was an Intel Core i9 13900K processor (Raptor Lake series) with two types of cores: eight P-cores and sixteen E-cores. Its P cores have a base frequency of 3GHz and a maximum turbo frequency of 5.6GHz. Its E cores have a base frequency of 2.2GHz and a maximum turbo frequency of 4.3GHz. In the absence of server-class processors with HMP yet, we consider the i9-13900K desktop CPU to be a suitable demonstrator of HMP capabilities.

4.1.1 System settings. The operating system on the system was Ubuntu 23.10 with Linux 6.5 kernel. We disabled simultaneous multithreading (SMT) (also called hyper-threading) on the CPU to ensure performance and power predictability. Evaluations of SMT on real-world applications have been mixed [42, 46]. While the maximum benefits can be up to 15% on a dual-socket system [28], these vary by applications [26], sometimes adversely [31], and require separate characterization to adjust model parameters, which we plan to do as future work. Further, we disabled the `intel_pstate` driver to bypass the processor's Energy-Performance Preference (EPP) logic and hardware-managed P-states, and enabled overclocking to turbo frequencies. This enabled setting specific core frequencies (beyond the base frequencies) from our scripts through ACPI driver and the userspace frequency governor. One disadvantage of using the ACPI driver is the limit of 15 on the number of frequency steps reported by the ACPI interface, hence the ACPI driver reports only a subset of the feasible frequencies to the userspace governor. To ensure steady-state thermal and power stability, we evaluated the system between 0.8–4.3 GHz frequencies. The available CPU frequencies presented by the ACPI driver in this range are as per the frequency set[4] listed in Table 1.

Core type	Frequency set[4] $\mathcal{F}$ in GHz
P or E	{0.8, 1, 1.3, 1.5, 1.8, 2, 2.3, 2.5, 2.8, 3, 3.3, 3.5, 3.8, 4, 4.3}

Table 1: Feasible frequency set for both types of cores.

The core 13900K processor does not allow sending cores individually to deep sleep states $\{C_6, C_8, C_{10}\}$ and those can be invoked at package level only [5]. These are not under the direct control of our scheduler, hence we disabled the package and core C-states through the BIOS for controlled experiments[5]. We left the core voltages untouched, to be determined as per CPU's internal VID table for each given frequency.

4.1.2 Workload benchmarks. Each computational task comprises of the execution of specified number of iterations of a given workload. We perform characterization, numerical, and experimental evaluation for two representative workloads:
(a) Image manipulation: We use the image manipulation benchmark (`538_imagick_r`) included in the SPECrate2017 floating-point benchmark suite under the SPEC-2017 benchmark package [17]. It uses Imagemagick [16], a commonly used batch image processing tool, in a single-threaded configuration. We execute the SPEC suite's `imagick` binary in a stand-alone mode without invoking it through `runcpu` to avoid the benchmark instrumentation overheads for short job runs.
(b) Machine Learning inference: As a machine learning inference workload, we use the TFLite Model Benchmark Tool [4] in a single-threaded CPU-only execution mode. The specific model used was 'NASNet mobile', a convolution neural network of size 21MB. A small model was chosen so that the inference iterations could be repeated for a desired number of times to compose tasks matching a given size distribution. The current section contains a detailed analysis and results for the image manipulation benchmark. Due to space limitations, we only provide the performance comparison with Linux governors in Section 4.3.4 for the machine learning inference benchmark.

[4] Although 2.2GHz is the base frequency for E-cores in i9-13900K processor, this frequency step did not appear on the subset reported by the ACPI driver for the given frequency range.
[5] However, the C_1 (active standby) state cannot be disabled. An idle cores are automatically sent by the CPU to C_1 state when it is not executing any instructions.

4.1.3 Measurement setup. CPU package power consumption is measured every 100ms through the RAPL mechanism [30] that reports accurate energy reading through the CPU's model-specific registers (MSR). RAPL is shown to accurately measure CPU package and core power [43]. CPU package power includes the core power, and the power consumption at the common components within the CPU, such as cache, internal bus, etc., shared between the cores [12]. The individual power consumption at these components is not directly measurable. Our estimated aggregate of static power at all cores includes the static core power and the power consumption at the shared components so that the total power estimate is closer to the real power consumed by the CPU. Core voltage values (common for all cores in i9-13900K CPU) are similarly measured through a corresponding MSR. We repeat the following characterization experiments for every frequency in $\mathcal{F}$ (Table1). To compute power characterization for each core type $c \in \{p, e\}$, the frequency for all cores of this type c is set at the given frequency. The frequency of the other core type is set to 800MHz. To measure static power, all cores are left idle. The measure the dynamic power, workload iterations are run in a loop pinned to each core of type c. Thus, the dynamic power characterization is performed at 100% core utilization for used cores, with unused cores idling in C_1 state[6]. Each data point was obtained by running the workload for 2 minutes.

4.1.4 Service rate and core voltage versus operating frequency. We plot the iteration completion rate (iterations/sec) as a function of core frequency in Fig. 1a. The plot validates (2), where the service rate at a core n is $\mu_n = \alpha_{c_n} f_n$ proportional to the core frequency f_n, and the proportionality constant α_{c_n} depends on the core type c_n.

Next, we plot the variation of core voltage as a function of core frequency in Fig. 1b. The voltage-frequency map for core n depends only on the core type $c_n \in \{p, e\}$. Performance and efficiency cores have different sets of feasible frequencies denoted by $\mathcal{F}_p$ and $\mathcal{F}_e$ respectively. The minimum operating voltages for performance and efficiency cores are $V_{p,0}$ and $V_{e,0}$, respectively. The corresponding minimum operating frequencies for performance and efficiency cores are defined as $f_{p,0} \triangleq \inf \mathcal{F}_{p,0}$ and $f_{e,0} \triangleq \inf \mathcal{F}_{e,0}$. For core n operating at frequency $f \in \mathcal{F}_{c_n}$, the operating voltage $V_n = V_{c_n}(f)$ is experimentally observed to be the following function of frequency,

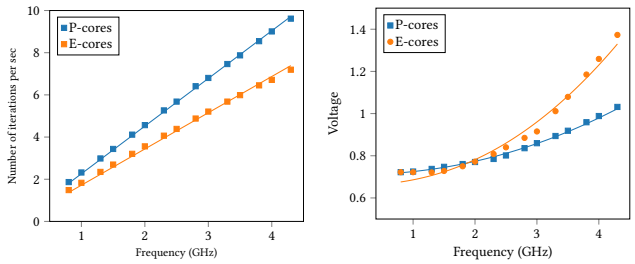
$$V_{c_n}(f) \triangleq V_{c_n,0} + r_{c_n} f^{s_{c_n}} \tag{21}$$

We note that the voltage of E-cores is higher than that of P-cores at identical frequencies. The proportionality constants α_p, α_e, linearity constants r_p, r_e, exponents s_p, s_e, minimum voltages $V_{p,0}, V_{e,0}$, and minimum feasible frequencies $f_{p,0}, f_{e,0}$ are listed in Table 2.

Type	α	β	a	b	r	s	V_0	f_0
P	0.0022	0.0014	0.1935	2.2703	$8.67e^{-9}$	2.08	0.722	800
E	0.0017	0.0008	0.1966	1.4789	$5.29e^{-9}$	2.23	0.722	800

Table 2: CPU model parameters from empirical characterization for Intel Core i9 13900K processor.

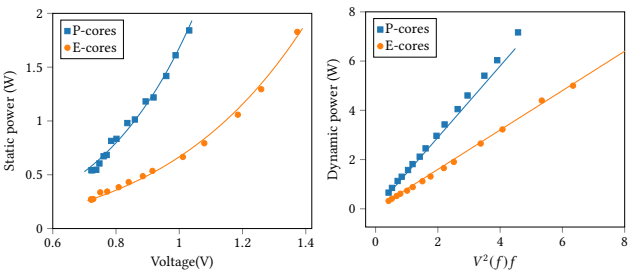
[6]except when the core is executing OS background tasks. The percentage of time spent executing OS background tasks in our test setup was observed to be below 0.02%.



(a) Number of iterations/sec. **(b) Operating voltage.**

Figure 1: Service rate and operating voltage.

4.1.5 Static and dynamic power. We note that the power consumption at a core consists of two components. Static power P_{sta} measured at zero utilization, and an *additional* dynamic power P_{dyn} when a core is working at full utilization. We plot the two empirical power measurements (dots) and the curve-fitted values (lines) in Fig. 2, for both core types. We plot the static power P_{sta} as a function of operating voltage V in Fig. 2a and observe that it closely follows the analytical expression in (3). Similarly, the plot of dynamic power P_{dyn} with respect to the product $V^2 f$ in Fig. 2b closely follows the analytical expression in (4).



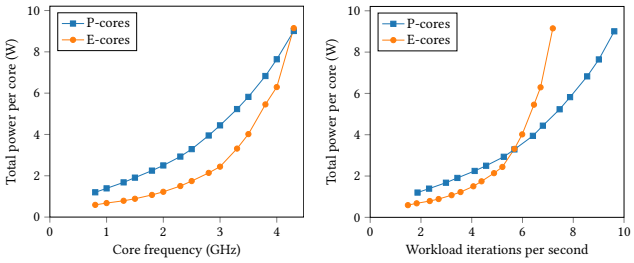
(a) Static power vs voltage. **(b) Dynamic power vs $V^2(f)f$.**

Figure 2: Static and dynamic components of CPU power.

4.1.6 CPU power versus core frequency and core service rate. We plot the per-core CPU power as a function of core frequency in Fig. 3a and as a function of workload service rate in Fig. 3b. As expected, total core power is a convex increasing function of core frequency. In fact, P-cores have higher power draw than E-cores for all frequencies. However, as the service rate for a P-core is higher than an E-core for the same core frequency, E-cores are more efficient at low service rates while P-cores become more efficient beyond the crossover point at 5.8 iterations/sec.

4.2 Numerical results

We validate the analytical results through numerical studies, using the parameters obtained through characterizations in Section 4.1. We assume the mean sojourn time guarantee at each core is $w = 4$ seconds. This is roughly one standard deviation from the mean service time for the slowest type of core at base frequency (E-cores at 2.2 GHz). In practice, workloads may specify a stricter latency



(a) Dependence on frequency. **(b) Dependence on service rate.**

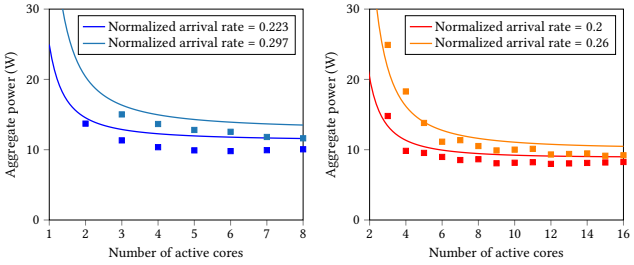
Figure 3: CPU package power per core.

bounds that can be incorporated in HEMP. We conducted numerical studies for the system parameters obtained by the power and the performance characterization of the heterogeneous core CPU in Section 4.1. In particular, we consider the number of efficiency and performance cores as $N_e = 16$ and $N_p = 8$ respectively. Therefore, the total number of cores $N = N_p + N_e = 24$. We assume that the static power follows (3) and the dynamic power follows (4) and the constant service rates at both core types follow (2), where constants $\alpha_p, \alpha_e, a_p, a_e, b_p, b_e, \beta_p, \beta_e$ are tabulated in Table 2.

The workloads under consideration comprise of discrete number of iterations of a benchmark (e.g., `imagick` image manipulation on a reference input image). Hence, the service requirement for tasks (number of iterations) was generated from an exponential distribution rounded to the nearest integer, with a mean of 8. We recall that the base frequencies for P-cores and E-cores are f_p = 3GHz and f_e = 2.2GHz respectively. For the given workload, the effective service rates for P and E cores are $\mu_p = 0.84$ and $\mu_e = 0.47$ tasks per second. For 16 E cores and 8 P cores, this implies the maximum base capacity of the system to be $8\mu_p + 16\mu_e = 14.286$ tasks per second. We normalize the arrival rate in terms of this base capacity and denote this normalized arrival rate by λ_{norm}.

In all of our theoretical studies, we have assumed the feasible set of frequencies to be continuous in Assumption 1, and the availability of a fractional number of cores in Assumption 2. These assumptions enabled a tractable analysis and proposal of theoretically optimal solutions. To align with the system reality, we remove both of these assumptions in our numerical studies. We numerically determined the optimal solution without these assumptions and found that the numerical solutions are close to the theoretically obtained solutions under the two assumptions. In the following discussion, we consider discrete set of frequencies $\mathcal{F}_p, \mathcal{F}_e$ (tabulated in Table 1), the number of active cores $N_{p,1} \in \{0, \ldots, N_p\}$, $N_{e,1} \in \{0, \ldots, N_e\}$, and workload split $\gamma \in \mathcal{M}([N_p])$ or $\gamma \in \mathcal{M}([N_e])$.

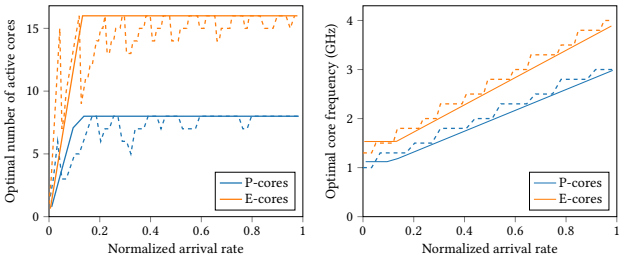
4.2.1 Homogeneous cores. We have plotted the total power aggregated over all cores of a single type as a function of the number of active cores for different normalized arrival rates λ_{norm} in Fig. 4, where each core has an identical feasible frequency at all active cores specified by (10) that meets the mean sojourn time guarantee $w = 4$ seconds at each core as prescribed by Theorem 1. We plot the power curve for expression obtained in Section 4.1 under Assumption 1 in solid lines and empirical values as squares, for P-cores in Fig. 4a and E-cores in Fig. 4b. Empirically obtained curves fit the predicted curves and validate our claims made in Lemma 5 and Theorem 2 regarding the convexity of aggregate power for each core type in the number of active cores for a fixed aggregate workload. In particular, we observe the existence of a unique optimal number of active cores to use for any given aggregate workload sent to that class of cores.



(a) P-cores. **(b) E-cores.**

Figure 4: Aggregate power.

We observed in Fig. 4 that for a fixed aggregate workload $N\lambda$ over a fixed core type, there exists an optimal number of active cores $N^*_{p,1}, N^*_{e,1}$. We plot the optimal allocation for homogeneous cores in Fig. 5 under Assumption 1 and Assumption 2 in solid lines and without these assumptions in dashed lines. In Fig. 5a we plot the variation of the optimal number of active cores with different normalized arrival rates λ_{norm}. We observe that the number of active cores increases almost linearly with the normalized arrival rate λ_{norm} for both core types until they saturate to their maximum value. From Theorem 2, we know that the optimal frequency at homogeneous cores is identical at all active cores such that they meet the mean sojourn time guarantee w. We plot the optimal frequency for both types of cores assuming the total workload is over a homogeneous set of cores in Fig. 5b. However, Theorem 4 suggests that the optimal frequencies are f^*_p, f^*_e and remain unchanged for low arrival rates $\lambda < \lambda_{p,0}$ for P-cores and $\lambda < \lambda_{e,0}$ for E-cores. We observe this phenomenon in Fig. 5b.



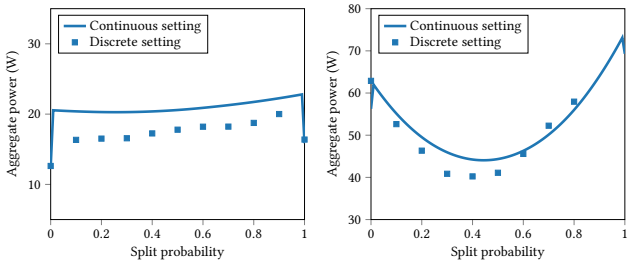
(a) Optimal core count. **(b) Optimal frequency.**

Figure 5: Optimal allocation of active cores and frequency.

4.2.2 Heterogeneous cores. For each workload split $\delta \in \mathcal{M}(\{p, e\})$, we have independent aggregate Poisson arrival of tasks with rates $N\lambda\delta_p$ and $N\lambda\delta_e$ at P-cores and E-cores respectively. For different values of split probability $\delta_p \in [0, 1]$, we have plotted the minimum

power consumption aggregated over all cores in Fig. 6 for two different values of normalized arrival rates $\lambda_{\text{norm}} \in \{0.210, 0.596\}$. We chose two nominal values of normalized arrival rate which were (a) in the feasibility region for the mean sojourn time guarantees and (b) sufficiently far apart to understand the behavior of optimal solutions under different loads.

As predicted by Theorem 5, the total aggregate average power is convex in the split probability δ_p for a fixed normalized arrival rate λ_{norm}. Further, as predicted by Theorem 4, we observe that at a low normalized arrival rate of $\lambda_{\text{norm}} = 0.210$, the total average power is affine in the split probability δ_p following (24).



(a) $\lambda_{\text{norm}} = 0.210$ **(b)** $\lambda_{\text{norm}} = 0.596$

Figure 6: Minimum power consumption vs. split probability.

We plot the optimal solution to Problem 1 with heterogeneous cores in Fig. 7. Dotted lines are numerical results in continuous values of number of cores and frequencies while squares correspond to results in discrete settings of the same. We plot the optimal workload split probability to P-core and E-cores in Fig. 7a as a function of increasing normalized arrival rate λ_{norm}. We numerically observe that $c_p^* > c_e^*$ in our case, for c_p^*, c_e^* defined in Definition 4. Thus, as predicted by Theorem 4, we observe that at low arrival rates $\lambda < \lambda_0 = 0.06$, i.e. normalized arrival rates $\lambda_{\text{norm}} < \lambda_{\text{norm},0} = 0.1$, all the workload is split among the E-cores. i.e. $\delta_p^* = 0$. Beyond this threshold, the optimal split is non-zero for P-cores and is concave increasing in normalized arrival rate λ_{norm}. That is, at low loads all the work is done at efficiency cores, and performance cores are woken up when the workload increases. As the workload keeps increasing, the optimal split saturates to the point where all cores are active and operating at their optimal frequencies. Fig. 7b shows the corresponding optimal number of active cores of each type given the optimal split $\delta^* = (\delta_p^*, 1 - \delta_p^*)$. We observe that for the normalized arrival rate $\lambda < \lambda_0$, no P-cores are active, and the number of active cores increases linearly with the normalized arrival rate until the saturation. Fig. 7c shows the optimal operating frequency at each active core for both core types. When the workload increases beyond $\lambda > 0.12$ i.e. $\lambda_{\text{norm}} > 0.2$, all the cores get active, and the core frequency has to be increased to accommodate the mean sojourn time guarantee w. We observe that these frequencies are in accordance with Theorem 1, and are identical for all cores of a particular type.

4.3 Experimental validation

We validate HEMP by implementing and experimentally evaluating its performance on Linux OS, and comparing its performance with that obtained in default Linux DVFS settings. We used the same test system, BIOS settings, and measurement setup as that used for empirical characterization in Section 4.1. HEMP performs the following tasks:

1. For a Poisson arrival of tasks with aggregate arrival rate $N\lambda$ having exponential service requirements and mean sojourn time guarantee w, it numerically obtains the optimal thinning probabilities γ^* and the optimal core frequency f^*. For all P and E cores with $\gamma^* > 0$, it sets their frequency at f_p^* and f_e^* respectively and it sets the frequency of idle cores ($\gamma^* = 0$) at the lowest frequency.
2. For every incoming task arrival, it randomly assigns the task to one of the active P or E cores with probability mass function γ^*.
3. If the assigned core is already busy running a task, hold the task assigned to a core in that individual core's virtual queue, to be executed when the queue becomes idle.

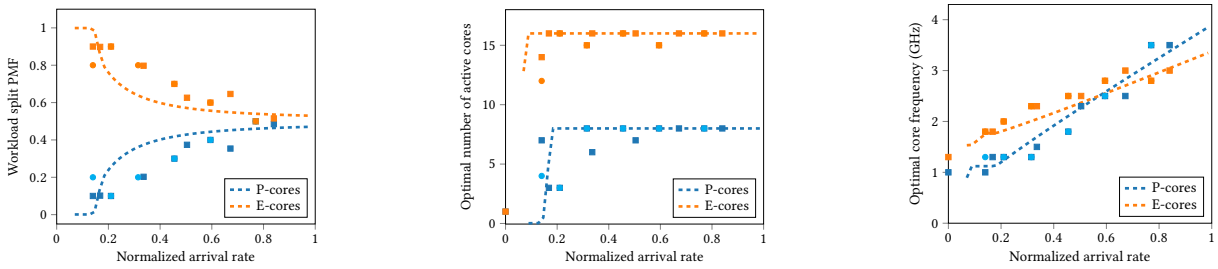
Workload generator: To experimentally evaluate HEMP, we also need to implement a task generator that generates realistic workloads matching the task distributions assumed in our model. The workload generator is designed to produce task arrivals matching the workload model described in Section 2 and Section 4.2. The tasks thus generated are submitted to whichever scheduler (e.g., HEMP or Linux's default Completely Fair Scheduler (CFS) [6] is being used in the experiment run.

4.3.1 Homogeneous cores. In Fig. 8, we experimentally validate the theoretical results which were validated numerically in Section 4.2.1. Again the curves experimentally confirm our claims made in Lemma 5 and Theorem 2 about the convexity of the total power in the number of active cores for a fixed aggregate workload and a single core type.

4.3.2 Heterogeneous cores. We now experimentally validate the numerical results for the heterogeneous setting as described in Section 4.2.2, with the same incoming workload and scheduling policy. In Fig. 9, we have plotted the total power aggregated over all cores as a function of split probability δ_p as in Fig. 6. We observe that the experimental curves are close to the theoretical curves, which validates the claims made in Section 4.2.2.

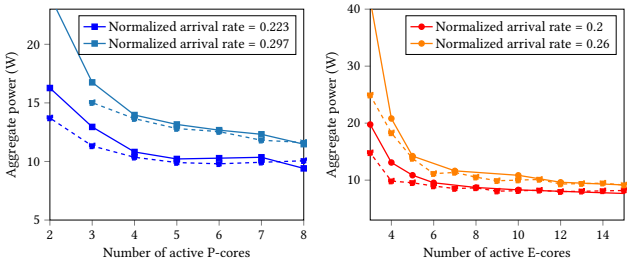
In Fig. 7 we have plotted the optimal solutions obtained experimentally and numerically as a function of normalized arrival rates. Dotted lines are numerical results from continuous settings, squares correspond to numerical results in discrete settings, and circles correspond to the experimental results. The experimental results validate the numerical studies plotted in Fig. 7 described in Section 4.2.2. We see that the scheduling policy obtained theoretically and numerically by HEMP is close to the experimentally obtained policy, which justifies our policy.

4.3.3 Comparison with Linux frequency governors. Next, we compare HEMP's performance with Linux frequency governors [7], namely powersave, performance, and schedutil, wherein HEMP and Linux governors received identical workload (SPEC-2017 image manipulation) task arrival patterns. For Linux governor experiments, no core pinning or frequency selection was applied, and the Linux OS's CFS [6] controlled the scheduling and allocation of tasks to cores. Both P and E cores were allowed to operate between 800 – 4300MHz under ACPI cpufreq driver control; rest of the system settings were identical to the CPU characterization setup.



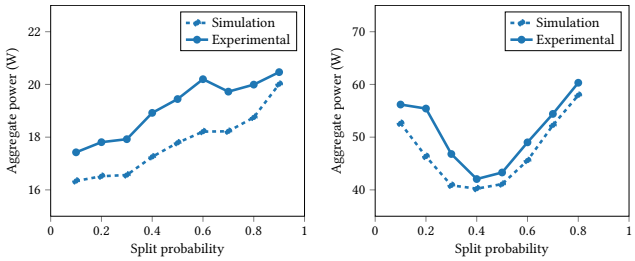
(a) Optimal workload split. (b) Optimal #active-cores. (c) Optimal frequency.

Figure 7: Optimal workload split between P and E cores.



(a) P-cores. (b) E-cores.

Figure 8: Aggregate power vs the number of active cores.



(a) $\lambda_{\text{norm}} = 0.210$. (b) $\lambda_{\text{norm}} = 0.596$.

Figure 9: Aggregate power versus split probability δ_p.

We note from Fig. 10a and Fig. 10b that performance and schedutil are aggressive in selecting higher frequencies at the cost of higher power while powersave prioritizes energy saving at the cost of higher sojourn time. HEMP, on the other hand, achieves CPU power close to powersave, while delivering stable sojourn time close to the set threshold w at all values of task arrival rate. Energy-delay product (EDP) [25] is often cited [32, 40] as a combined metric to quantify the performance-energy trade-off. Fig. 10c plots the EDP among the schemes compared. HEMP is more efficient than Linux governors at normalized arrival rate $\lambda_{\text{norm}} > 0.25$ and achieves up to 45% reduction at $\lambda_{\text{norm}} > 0.45$.

Disabling C-states may seem to constrain Linux frequency governors unfairly by limiting the available optimizations. Therefore, we repeated the Linux governor experiments after enabling C-states from BIOS[7]. The EDP plots in Fig. 10d for this scenario indicates that HEMP offers up to 35% reduction in EDP above $\lambda_{\text{norm}} = 0.45$. It should be noted that HEMP would similarly benefit from enabling C-states below $\lambda_{\text{norm}} = 0.2$ wherein P-cores are unused.

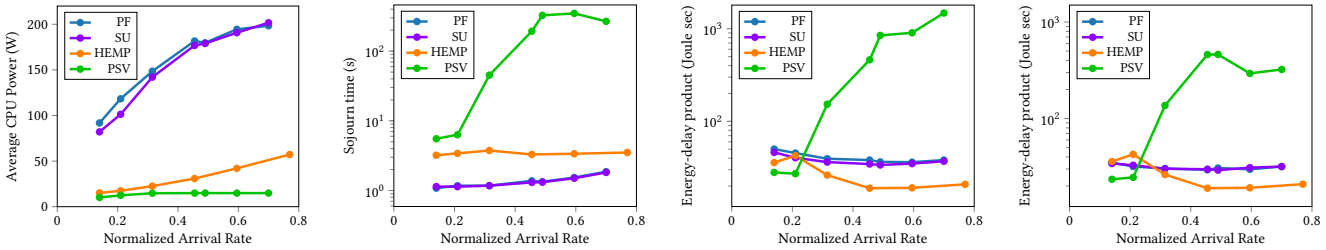
4.3.4 Machine learning workloads (TensorFlow Lite): Comparison with Linux frequency governors. Finally, we present the comparison of HEMP with Linux governors for a different workload—the TensorFlow Lite inference benchmark [4] running the NASNet Mobile model. CPU characterization with this workload yielded model parameters similar to Table 2 which were used for numerically obtaining the optimal thinning probabilities γ^* across all cores and the optimal core frequency f^* for active P and E cores. The service requirement for each task, which is supplied as an input to the workload generator as the number of runs of NASNet inference to execute, is computed similar to the image manipulation workload 4.2. The sojourn time guarantee w is taken to be 1.3 seconds, following the same logic as described in Section 4.2.

The plots for measured CPU power, sojourn time, and EDP are presented in Fig. 11. We note that HEMP is able to provide stable sojourn time guarantee 1.3s while consuming power closer to powersave governor. Similarly, HEMP offers 33-80% reduction in EDP for $\lambda_{\text{norm}} \in [0.2, 0.8]$. Even when C-states are enabled for Linux governor experiments, HEMP offers 30-73% improvement in EDP.

5 CONCLUSION AND FUTURE WORK

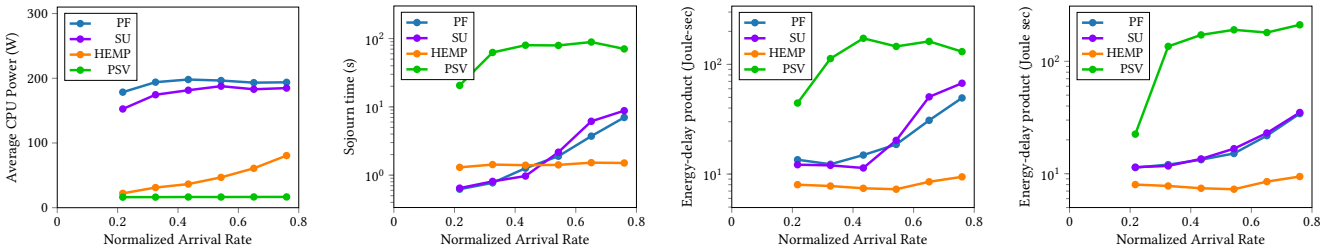
We have provided a power and performance model for a computer system with multi-core CPUs consisting of heterogeneous cores (HMP) that implement a given instruction set architecture (ISA). Using this, we analytically find the optimal allocation of the number of active cores, workload split across all the heterogeneous cores, and the operating frequency at each core. Due to the convexity of power functions, the optimal solution for cores of a fixed type is an equal division of workload among all active cores, with identical frequency for all. Thus, one only needs to find the optimal workload split between the core types and the number of active cores for each type. We conducted experiments on a HMP to find that the empirically found optimal solution to the static workload configuration is close to the analytically predicted optimal solution.

[7]Experiments for HEMP could not be repeated with C-states enabled due to paucity of time.



(a) Total power. (b) Sojourn Time. (c) Energy-delay product. (d) EDP with C-states enabled.

Figure 10: Comparison with Linux frequency governors: powersave (PSV), performance (PF), and schedutil (SU).



(a) Total power. (b) Sojourn time. (c) Energy-delay product. (d) EDP with C-states enabled.

Figure 11: Comparison with Linux frequency governors For TensorFlow Lite workload.

Our system model and corresponding analytical results may apply with minor enhancements to the ARM big.LITTLE architecture [1], which brings two different processors with similar ISA together to simulate a single CPU with heterogeneous cores by allowing only one of these processors to be active at any time. Our work will also be applicable to future HMPs as long as convexity of power functions holds true.

There are multiple future research directions. One direction is to study other possible and more flexible service guarantees with the potential to increase power savings. Examples include statistical service guarantees averaged over all tasks, guarantees on the variance of sojourn times, or on the tail of the sojourn time distributions. A second direction is the design of randomized solutions in which frequencies and core counts are modeled as random variables and the optimal allocation selects probability mass functions for these. This can eliminate discontinuities due to the discrete nature of these quantities.

Extensions to support dynamic workloads is another interesting direction. Workload-dependent splits like joining the core with the least workload or power-of-d variants of such splits, inclusion of multiple sleep states with different wake-up times, and on-demand changes to core frequencies can be considered here. In the absence of workload models, OS'es use system metrics such as task ready queue size, cache misses, core utilization, etc., as feedback signals to make heuristic scheduling decisions [48, 49]. A very interesting direction is to design data-driven adaptive approaches to learn workload configurations at runtime and adapt the scheduling policies gracefully to the changing environments.

With the increasing adoption of GPUs in large computing clusters, computing power would be an even more dominant percentage of energy consumption. GPU architectures are still evolving and power and performance measurements are either not available or not very accurate. Part of our study can be extended to the case of GPUs.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

This research was supported in part by IBM Research India under IBM-IISc Hybrid Cloud Lab (IIHCL) open research collaboration; in part by Qualcomm Inc. under Qualcomm University Relations 6G India; in part by Science and Engineering Research Board (SERB) Grant CRG/2023/008854; in part by UK-India Education and Research Initiative (UKIERI) Grant SPARC/2024/3927; and in part by Centre for Networked Intelligence (a Cisco Corporate Social Responsibility (CSR) Initiative) at Indian Institute of Science, Bengaluru. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] 2013. Power Management with big.LITTLE: A technical overview. https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/power-management-with-big-little-a-technical-overview.
[2] 2018. 12th Gen Intel Core Processors. https://www.intel.com/content/www/us/en/products/docs/processors/core/12th-gen-processors.html.
[3] 2018. Energy Aware Scheduling. https://docs.kernel.org/scheduler/sched-energy.html.
[4] 2022. "Performance Measurement | TensprFlow Lite". https://www.tensorflow.org/lite/performance/measurement
[5] 2023. 13th Generation Intel Core Processors Datasheet, Volume 1 of 2. https://www.intel.com/content/www/us/en/content-details/743844/13th-generation-intel-core-processors-datasheet-volume-1-of-2.html

[6] 2024. CFS Scheduler. https://docs.kernel.org/scheduler/sched-design-CFS.html.

[7] 2024. CPU Performance Scaling. https://docs.kernel.org/admin-guide/pm/cpufreq.html.

[8] International Energy Agency. [n. d.]. Data and Statistics. Web. https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks accessed: Sep 2023.

[9] Soren Asmussen. 1987. *Applied probability and queues*. Springer.

[10] Sayed A. Banawan and Nidal M. Zeidat. 1992. A Comparative Study of Load Sharing in Heterogeneous Multicomputer Systems. In *Annual Symposium on Simulation* (Orlando, Florida, USA) *(ANSS '92)*. IEEE Computer Society Press, Washington, DC, USA, 22–31.

[11] Luiz André Barroso and Urs Hölzle. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (2007), 33–37.

[12] Robert Basmadjian and Hermann de Meer. 2012. Evaluating and modeling power consumption of multi-core processors. In *International Conference on Future Systems (e-Energy)*. 1–10.

[13] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[14] Andre R Brodtkorb, Christopher Dyken, Trond R Hagen, Jon M Hjelmervik, and Olaf O Storaasli. 2010. State-of-the-art in heterogeneous computing. *Scientific Programming* 18, 1 (2010), 1–33.

[15] Yuan-Chieh Chow and Kohler. 1979. Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System. *IEEE Trans. Comput.* C-28, 5 (1979), 354–361.

[16] Standard Performance Evaluation Corporation. [n. d.]. ImageMagick: Convert, Edit, or Compose Digital Images. https://imagemagick.org/.

[17] Standard Performance Evaluation Corporation. 2017. The SPEC CPU 2017 benchmark package. https://www.spec.org/cpu2017/.

[18] Anthony Danalis, Gabriel Marin, Collin McCurdy, Jeremy S Meredith, Philip C Roth, Kyle Spafford, Vinod Tipparaju, and Jeffrey S Vetter. 2010. The scalable heterogeneous computing (SHOC) benchmark suite. In *Workshop on general-purpose computation on graphics processing units*. 63–74.

[19] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2016. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 732–794.

[20] Dong Du, Qingyuan Liu, Xueqiang Jiang, Yubin Xia, Binyu Zang, and Haibo Chen. 2022. Serverless Computing on Heterogeneous Computers. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '22)*. ACM, 797–813.

[21] EnerData. 2023. Between 10 and 20% of electricity consumption from the ICT sector in 2030? Web. https://www.enerdata.net/publications/executive-briefing/world-energy-consumption-from-digitalization.pdf accessed: Sep 2023.

[22] Chris Gianos. 2023. Architecting for Flexibility and Value with Next Gen Intel® Xeon® Processors. https://hc2023.hotchips.org/assets/program/conference/day1/Platforms/HC2023.Intel.Gianos.v7.pdf

[23] Joe Goldberg. 2018. Modern Batch Processing: A Thing of the Past or Essential Discipline? Web page. https://www.bmc.com/blogs/modern-batch-processing-thing-past-essential-discipline/ accessed: Feb 2024.

[24] Alexander Golovin, Robert Basmadjian, Sergey Astafiev, and Alexander Rumyantsev. 2023. Little's Law in a Single-Server System with Inactive State for Demand-Response in Data Centers with Green SLAs. In *Companion Proceedings of ACM e-Energy* (Orlando, FL, USA) *(e-Energy '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 91–97.

[25] R. Gonzalez and M. Horowitz. 1996. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits* 31, 9 (1996), 1277–1284.

[26] Wessam M Hassanein, Layali K Rashid, and Moustafa A Hammad. 2008. Analyzing the Effects of Hyperthreading on the Performance of Data Management Systems. *International Journal of Parallel Programming* 36, 2 (2008), 206–225.

[27] Mark Horowitz, Elad Alon, Dinesh Patil, Samuel Naffziger, Rajesh Kumar, and Kerry Bernstein. 2005. Scaling, power, and the future of CMOS. In *IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest.* 7–15.

[28] Bill Jones. 2023. Will Hyper-Threading Improve Processing Performance? Web page. https://www.dasher.com/will-hyper-threading-improve-processing-performance/ accessed: Feb 2024.

[29] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. 2010. Virtual machine power metering and provisioning. In *ACM Symposium on Cloud Computing (SoCC '10)*. 39–50.

[30] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 3, 2 (Mar 2018), 1–26.

[31] Donald Kinghorn. [n. d.]. Hyper-Threading may be Killing your Parallel Performance. Web page. https://www.pugetsystems.com/labs/hpc/hyper-threading-may-be-killing-your-parallel-performance-578/ accessed: Feb 2024.

[32] R. Kumar, K.I. Farkas, N.P. Jouppi, P. Ranganathan, and D.M. Tullsen. 2003. Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction. In *IEEE/ACM International Symposium on Microarchitecture (MICRO-36)*. 81–92.

[33] Rakesh Kumar, Dean M Tullsen, Parthasarathy Ranganathan, Norman P Jouppi, and Keith I Farkas. 2004. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. *ACM SIGARCH Computer Architecture News* 32, 2 (2004), 64.

[34] Vincent Lannurien, Laurent D'Orazio, Olivier Barais, Esther Bernard, Olivier Weppe, Laurent Beaulieu, Amine Kacete, Stéphane Paquelet, and Jalil Boukhobza. 2023. HeROfake: Heterogeneous Resources Orchestration in a Serverless Cloud – An Application to Deepfake Detection. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 154–165.

[35] Ronald L. Larsen and Ashok K. Agrawala. 1983. Control of a Heterogeneous Two-Server Exponential Queueing System. *IEEE Transactions on Software Engineering* SE-9 (1983), 522–526. https://api.semanticscholar.org/CorpusID:13941700

[36] T. Li, S. Ying, Y. Zhao, and J. Shang. 2024. Batch Jobs Load Balancing Scheduling in Cloud Computing Using Distributional Reinforcement Learning. *IEEE Transactions on Parallel & Distributed Systems* 35, 01 (jan 2024), 169–185.

[37] Woei Lin and P. Kumar. 1984. Optimal control of a queueing system with two heterogeneous servers. *IEEE Trans. Automat. Control* 29, 8 (1984), 696–703.

[38] Ravi Reddy Manumachu and Alexey Lastovetsky. 2022. On Energy Nonproportionality of CPUs and GPUs. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 34–44.

[39] David Meisner and Thomas F. Wenisch. 2012. DreamWeaver: Architectural Support for Deep Sleep. In *International Conference on Architectural Support for Programming Languages and Operating Systems* (London, England, UK) *(ASPLOS XVII)*. Association for Computing Machinery, New York, NY, USA, 313–324.

[40] Andreas Merkel, Jan Stoess, and Frank Bellosa. 2010. Resource-conscious scheduling for energy efficiency on multicore processors. In *European Conference on Computer Systems* (Paris, France) *(EuroSys '10)*. Association for Computing Machinery, New York, NY, USA, 153–166.

[41] Sparsh Mittal and Jeffrey S Vetter. 2015. A survey of CPU-GPU heterogeneous computing techniques. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 1–35.

[42] Lucía Pons, Josué Feliu, José Puche, Chaoyi Huang, Salvador Petit, Julio Pons, María E. Gómez, and Julio Sahuquillo. 2022. Effect of Hyper-Threading in Latency-Critical Multithreaded Cloud Applications and Utilization Analysis of the Major System Resources. *Future Generation Computer Systems* 131 (2022), 194–208.

[43] Thomas Rauber, Gudula Rünger, Michael Schwind, Haibin Xu, and Simon Melzner. 2014. Energy measurement, modeling, and prediction for processors with frequency scaling. *The Journal of Supercomputing* 70 (12 2014), 1451–1476.

[44] Nikzad Babaii Rizvandi, Javid Taheri, and Albert Y. Zomaya. 2011. Some observations on optimal frequency selection in DVFS-based energy consumption minimization. *J. Parallel and Distrib. Comput.* 71, 8 (2011), 1154–1164.

[45] Juan Carlos Saez, Alexandra Fedorova, David Koufaty, and Manuel Prieto. 2012. Leveraging core specialization via OS scheduling to improve performance on asymmetric multicore systems. *ACM Transactions on Computer Systems (TOCS)* 30, 2 (2012), 1–38.

[46] Subhash Saini, Haoqiang Jin, Robert Hood, David Barker, Piyush Mehrotra, and Rupak Biswas. 2011. The impact of hyper-threading on processor resource utilization in production applications. In *International Conference on High Performance Computing*. 1–10.

[47] Abul Sarwar. 1997. CMOS power consumption and C_{pd} calculation. Web page. https://www.ti.com/lit/an/scaa035b/scaa035b.pdf?ts=1714742752758

[48] Claudio Scordino, Luca Abeni, and Juri Lelli. 2018. Energy-aware real-time scheduling in the linux kernel. In *ACM Symposium on Applied Computing*. 601–608.

[49] Claudio Scordino, Luca Abeni, and Juri Lelli. 2019. Real-time and energy efficiency in Linux: theory and practice. *ACM SIGAPP Applied Computing Review* 18, 4 (2019), 18–30.

[50] S. Shenker and A. Weinrib. 1989. The optimal control of heterogeneous queueing systems: a paradigm for load-sharing and routing. *IEEE Trans. Comput.* 38, 12 (1989), 1724–1735.

[51] Jayanth Srinivasan, Sarita V. Adve, Pradip Bose, and Jude A. Rivers. 2004. The impact of technology scaling on lifetime reliability. In *International Conference on Dependable Systems and Networks*. 177–186.

[52] Xueyan Tang and Samuel T Chanson. 2000. Optimizing static job scheduling in a network of heterogeneous computers. In *International Conference on Parallel Processing*. IEEE, 373–382.

[53] Inc. UEFI Forum. 2022. Advanced Configuration and Power Interface ""(ACPI)"" Specification Release 6.5. https://uefi.org/sites/default/files/resources/ACPI_Spec_6_5_Aug29.pdf

[54] Abel Weinrib and Scott Shenker. 1988. Greed is not enough: adaptive load sharing in large heterogeneous systems. *IEEE INFOCOM* (1988), 986–994.

[55] Mark Whitney. 2023. Best Practices for Running HPC Batch Jobs. Web page. https://rescale.com/blog/batch-job/ accessed: Feb 2024.

[56] Daniel Wong and Murali Annavaram. 2012. KnightShift: Scaling the Energy Proportionality Wall through Server-Level Heterogeneity. In *IEEE/ACM International Symposium on Microarchitecture*. 119–130.

A CPU POWER-SAVING STATES

The ACPI specification [53] defines common vendor-agnostic interfaces enabling robust operating system (OS) directed power management of devices and systems. The ACPI specification defines a few common system power states as shown in Fig. 12. They are classified as global (G), sleep (S), and CPU (C) states, numbered from 0 (no power-saving) in increasing order of power-saving. We focus on the '$G_0 - S_0$: working' state of the CPU, within which the CPU can be in one of $\{C_0, C_1, \ldots, C_k\}$ states. Contained within the C_0 state are several performance states $\{P_0, \ldots, P_k\}$ that define the voltage and frequency supplied to the CPU. A vendor-specific CPU architecture (e.g., [5]) determines the number of such states supported, physical parameters, and the control primitives for these states. Recent CPUs allow control of the C and P states on a per-core basis. Some of the C and P states are under the control of the OS although the CPU hardware/firmware may override them due to thermal or interrupt servicing considerations.

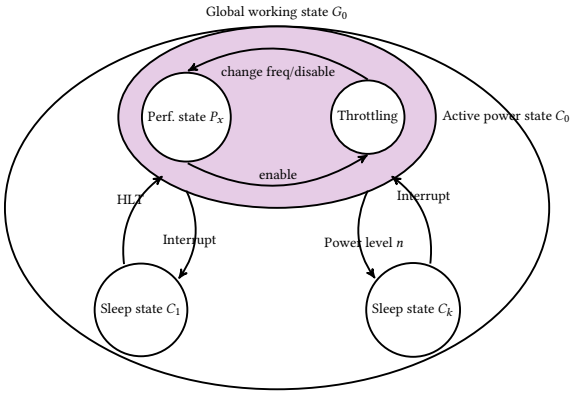


Figure 12: Power states of a working CPU

B CONVEX FUNCTIONS

Lemma 2. *[13, Chapter 3.2] Assume that $f, g : \mathbb{R}_+ \to \mathbb{R}_+$ are convex increasing functions, then we have the following*

a) Affine functions in one variable are convex.
b) fg is also convex and increasing.
c) $f(x)+g(1-x)$ is also a convex function over the domain $[0, 1]$.
d) $f \circ g$ is also convex and increasing.
e) $af + bg$ is also convex and increasing for any $a, b \in \mathbb{R}_+$.
f) If $h(x, t) = tf(\frac{x}{t})$, then $h : (\mathbb{R}_+)^2 \to \mathbb{R}_+$, called the perspective of f is also a convex function on $(\mathbb{R}_+)^2$. Consequently, for any fixed $(x, t) \to h(x, t)$ is convex on $\mathbb{R}_+$.
g) If $f : \mathbb{R}^2 \to \mathbb{R}$ is jointly convex in both arguments, and $C \subseteq \mathbb{R}$ is a convex non-empty set, then $g(x) \triangleq \inf_{y \in C} f(x, y)$ is a convex function.

Proof. Part 2, 2, and 2 can be verified directly. Convexity of resulting functions in parts 2, 2, 2, and 2 follow directly from [13, Chapter 3.2]. The fact that the resulting functions are increasing is immediate for parts 2 and 2, and for part 2 it follows from the additional assumption of non-negativity. □

C PROOFS

C.1 Convexity of static and dynamic power

Lemma 3. *For any active core $n \in \mathcal{N}_1$, the following are convex increasing functions of frequency f_n.*

a) The static power $P_{\text{sta}}(c_n, f_n)$,
b) the dynamic power per frequency $P_{\text{dyn}}(c_n, f_n)/f_n$, and
c) the average power consumption $\bar{P}_n(c_n, f_n, \gamma_n)$.

In addition, the average power consumption $\bar{P}_n(c_n, f_n, \gamma_n)$ is affine and increasing in thinning probability γ_n.

Proof. Let n be an active core. From Eq. (3) and Eq. (4) for static and dynamic power respectively, it follows that P_{sta} and P_{dyn}/f_n are convex increasing functions of voltage V_n. From Property 1, the operating voltage $V_n(c_n, f_n)$ at core n is a convex non-decreasing function of frequency f_n.

a) Since $P_{\text{sta}}(c_n, f_n)$ is the composition of $P_{\text{sta}}(V_n)$ and the voltage function $V_n(c_n, f_n)$, the result follows from Lemma 22.
b) Since $P_{\text{dyn}}(c_n, f_n)/f_n$ is the composition of $\frac{P_{\text{dyn}}(V_n)}{f_n}$ and the voltage function $V_n(c_n, f_n)$, the result follows from Lemma 22.
c) From (8) for any active core n, it follows that the average power consumption $\bar{P}_n(c_n, f_n, \gamma_n)$ at node n is a positive linear combination of $P_{\text{sta}}(c_n, f_n)$ and $P_{\text{dyn}}(c_n, f_n)/f_n$. Thus, it is also convex and increasing from Lemma 22.

From (8) for any active core n, it follows that average power consumption $\bar{P}_n(c_n, f_n, \gamma_n)$ at node n is affine in γ_n, and increasing from the positivity of the other terms. □

C.2 Proof of Lemma 1

From Eq. (7), it follows that for a fixed thinning probability γ_n the mean sojourn time $\bar{W}_n$ at any core n is decreasing function of frequency f_n. Therefore, we have $\bar{W}_n \leqslant w$ for any frequency $f_n \geqslant f_n^*(c_n, \gamma_n)$. For any core n with non-zero thinning probability $\gamma_n > 0$, the average power consumption is an increasing function of frequency f_n from Lemma 3, and hence the result follows.

C.3 Convexity of static and dynamic power under optimal frequency allocation

Lemma 4. *Under Assumption 1, an active core n operating at a frequency $f_{c_n}(\gamma_n)$ defined in (12) satisfies the mean sojourn time constraint w while minimizing the average power consumption for a given thinning probability γ_n. The following powers at core n are convex increasing in the positive thinning probability γ_n.*

a) The static power $P_{\text{sta}}(c_n, f_{c_n}(\gamma_n))$,
b) the scaled dynamic power $\gamma_n P_{\text{dyn}}(c_n, f_{c_n}(\gamma_n))/f_{c_n}(\gamma_n)$, and
c) the power consumption $\bar{P}_n(c_n, \gamma_n)$ at core n defined in (11).

Proof. Let n be an active core.

Under Assumption 1, it follows from Remark 2, that the optimal operating frequency $f_n^*(c_n, \gamma_n) = f_{c_n}(\gamma_n)$ for an active core n is an affine increasing function of positive thinning probability γ_n.

a) Since $P_{\text{sta}}(c_n, f_n^*(c_n, \gamma_n))$ is the composition of two convex increasing functions $P_{\text{sta}}(c_n, f_n)$ and $f_n^*(c_n, \gamma_n)$, it is convex increasing in γ_n from Lemma 22.
b) Since $P_{\text{dyn}}(c_n, f_n^*(c_n, \gamma_n))/f_n^*(c_n, \gamma_n)$ is the composition of two convex increasing functions $P_{\text{dyn}}(c_n, f_n)/f_n$ and $f_n^*(c_n, \gamma_n)$,

it is convex increasing in γ_n from Lemma 22. Since γ_n is linear non-negative and increasing, it follows from Lemma 22 that the scaled dynamic power $\frac{\gamma_n}{f_n^*(c_n,\gamma_n)} P_{\text{dyn}}(c_n, f_n^*(c_n,\gamma_n))$ is also convex increasing in positive thinning probability γ_n.

c) The average power consumption at core n with a positive thinning probability $\gamma_n > 0$ and operating frequency $f_n^*(c_n,\gamma_n)$

$$\bar{P}_n(c_n,\gamma_n) = P_{\text{sta}}(c_n, f_n^*(c_n,\gamma_n)) + \frac{N\lambda\gamma_n}{\alpha_{c_n}} \frac{P_{\text{dyn}}(c_n, f_n^*(c_n,\gamma_n))}{f_n^*(c_n,\gamma_n)},$$

is a positive linear combination of two convex increasing functions of γ_n, and hence is convex increasing in γ_n from Lemma 22.

□

C.4 Proof of Theorem 1

We show the result holds when all cores are of performance type. In this case, $\mathcal{N}_{p,1} = [N]$ and $c_n = p$ for all active cores n. Further, the minimum power consumption at core n while meeting the mean sojourn time guarantee of w is ensured by setting the operating frequency at $f_n^*(p,\gamma_n) = f_p(\gamma_n)$. The minimum power consumption at performance core n for a given positive thinning PMF γ is written as

$$\bar{P}_n(p,\gamma_n) = P_{\text{sta}}(p, f_p(\gamma_n)) + \frac{N\lambda\gamma_n}{\alpha_p} \frac{P_{\text{dyn}}(p, f_p(\gamma_n))}{f_p(\gamma_n)}.$$

From the convexity of $\bar{P}_n(c_n,\gamma_n)$ in γ_n from Lemma 4 for a fixed c_n, we obtain that

$$\frac{1}{N}\sum_{n=1}^{N} \bar{P}_n(p,\gamma_n) \geqslant \sum_{n=1}^{N} \bar{P}_n(p, \frac{1}{N}).$$

This implies that $\gamma_n^* = \frac{1}{N}$ for all active performance cores $n \in [N]$. We can repeat the proof when all cores are of efficiency type.

C.5 Joint convexity of aggregate power under optimal frequency allocation

Lemma 5. *Let $c \in \{p, e\}$ be the core type. Consider a fixed number N of c type cores with a given set of non-sleeping cores $\mathcal{N}_{c,1}$ and continuous operating frequency $\mathcal{F}_c$. The minimum aggregate power that satisfies the mean sojourn time guarantee w at each active core $n \in \mathcal{N}_{c,1}$ is $P_c(N\lambda, N_{c,1})$, where $P_c(N\lambda, x) : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ is jointly convex in $N\lambda$ and x.*

Proof. We show the result holds when all cores are of performance type. From Theorem 1, the optimal thinning probability for non-sleeping cores $n \in \mathcal{N}_{p,1}$ is $\gamma_n^* = \frac{1}{N_{p,1}}$. Thus, we can write the optimal operating frequency for all non-sleeping cores $n \in \mathcal{N}_{p,1}$ as

$$f_n^*(p,\gamma_n^*) \triangleq \frac{1}{\alpha_p}\Big(\frac{N\lambda}{N_{p,1}} + \frac{1}{w}\Big).$$

The minimum aggregate power consumption $P_p(N\lambda, N_{p,1})$ at all $N_{p,1}$ non-sleeping cores can be written as

$$N_{p,1} P_{\text{sta}}\left(p, \frac{1}{\alpha_p}\Big(\frac{N\lambda}{N_{p,1}} + \frac{1}{w}\Big)\right) + \frac{N\lambda}{\alpha_p}\beta_p V_p\left(\frac{1}{\alpha_p}\Big(\frac{N\lambda}{N_{p,1}} + \frac{1}{w}\Big)\right)^2. \quad (22)$$

We observe that the average power consumption at each active performance core $n \in \mathcal{N}_{p,1}$ is given by

$$\bar{P}_n(c_n,\gamma_n^*) = P_{\text{sta}}(p, f_n^*(p,\gamma_n^*)) + \frac{N\lambda\gamma_n^*}{\alpha_p} \frac{P_{\text{dyn}}(p, f_n^*(p,\gamma_n^*))}{f_n^*(p,\gamma_n^*)}.$$

Thus, $P_p(N\lambda, N_{p,1}) = N_{p,1}\bar{P}_n\Big(c_n, \frac{N\lambda}{N_{p,1}}\Big)$. From Lemma 44, it follows that $\bar{P}_n(p,\gamma_n)$ is a convex function of γ_n. We observe that $P_p(N\lambda, N_{p,1})$ is the perspective of $\bar{P}_n(p,\gamma_n)$, hence jointly convex in $N\lambda$ and $N_{p,1}$ from Lemma 2, part (2). We can repeat the proof when all cores are of efficiency type. □

C.6 Proof of Theorem 2

We show the result holds when all cores are of performance type. We observe that $\mathcal{N}_p = [N]$ and the set of sleeping cores are $[N] \setminus \mathcal{N}_{p,1}$. We can apply Theorem 1 to active cores $\mathcal{N}_{p,1}$ to obtain the optimal allocation in (17) for any set of active cores $\mathcal{N}_{p,1}$. We can find the aggregate average power consumption for all N cores

$$P_{\text{tot}}(p, N\lambda, N, N_{p,1}) \triangleq P_p(N\lambda, N_{p,1}) + (N - N_{p,1})P_{\text{sleep}}(p). \quad (23)$$

We observe that $P_{\text{tot}}(p, N\lambda, N, x)$ is the sum of affine function $(N - x)P_{\text{sleep}}(p)$ and convex function $P_p(N\lambda, x)$ for $x \in \mathbb{R}_+$. Thus, the aggregate average power consumption $P_{\text{tot}}(p, N\lambda, N, \cdot) : [0, N] \to \mathbb{R}_+$ is a convex function, and the minima is achieved at some unique $x_p^* \in [0, N]$. Since the number of cores $N_{p,1}$ is not a real number, (a) we can find the ceiling and the floor of the optimal real number x_p^*, and (b) compare the two power consumptions to find an optimal number of cores $N_{p,1}^*$. This optimal number is unique if the power consumption at the ceil and the floor of x_p^* are not identical. We can repeat the proof when all cores are of efficiency type.

C.7 Proof of Theorem 3

We assume that $\delta \triangleq (\delta_p, \delta_e) \in \mathcal{M}(\{p, e\})$ is the thinning PMF across performance and efficiency cores. This implies that the aggregate arrival rate to N_p performance and N_e efficiency cores are $N\lambda\delta_p$ and $N\lambda\delta_e$ respectively. For these fixed aggregate arrival rates, the minimum aggregate average power consumption at performance and efficiency cores are given by $P_{\text{tot}}(p, N\lambda\delta_p, N_p)$ and $P_{\text{tot}}(e, N\lambda\delta_e, N_e)$ respectively, as defined in Definition 3.

These power consumptions are achieved at performance and efficiency cores by letting some of the cores sleep, thinning the aggregate arrivals equally among all active cores of each type, and operating all active cores n at an identical frequency for each core type. Thus, for a fixed splitting δ, the minimum aggregate average power consumption at all N cores is achieved when each core n operates at frequency $f_n^*(c_n, \frac{\delta_{c_n}}{N_{c_n,1}})$ that minimizes the average power consumption at each core type while satisfying the mean sojourn time guarantee w at each active core n. For each core type, the number of active cores is optimized to minimize this power consumption.

It suffices to show that there is an optimal split δ among the two core types. From Remark 4, both $P_{\text{tot}}(p, N\lambda, N_p)$ and $P_{\text{tot}}(e, N\lambda, N_e)$ are convex in $N\lambda$ under Assumption 2, and since $\delta_p + \delta_e = 1$, it follows from Lemma 22 that the total aggregate power in convex in δ_p. Thus, there exists a unique $\delta_p^* \in [0, 1]$ that minimizes the aggregate power consumption.

C.8 Proof of Theorem 4

For a given workload split $\delta \in \mathcal{M}(\{p,e\})$ and denoting the number of active performance and efficiency cores by x_p, x_e, we can write the optimal identical operating frequencies at the two types of active cores that meet the mean sojourn time guarantee w at each active core, are

$$f_p \triangleq \frac{1}{\alpha_p}\Big(\frac{N\lambda\delta_p}{x_p} + \frac{1}{w}\Big), \quad f_e \triangleq \frac{1}{\alpha_e}\Big(\frac{N\lambda\delta_e}{x_e} + \frac{1}{w}\Big).$$

From this equation, we can write the number of active cores x_p, x_e in terms of the operating frequencies f_p, f_e. Under the theorem hypothesis of zero sleep powers, workload split δ, and operating frequencies f_p, f_e, the minimum average power consumption for core n of type $c_n \in \{p,e\}$ can be written as

$$P_{\text{tot}}(c_n, N\lambda\delta_{c_n}, N_{c_n}, f_{c_n}) \triangleq N\lambda\delta_{c_n}\Big(\frac{P_{\text{sta}}(c_n, f_{c_n}) - P_{\text{sleep}}(c_n)}{\alpha_{c_n} f_{c_n} - \frac{1}{w}} + \frac{\beta_{c_n}}{\alpha_{c_n}} V_{c_n}(f_{c_n})^2\Big) + N_{c_n} P_{\text{sleep}}(c_n).$$

We observe that the optimal operating frequency f_p^*, f_e^* at both the cores is independent of the split δ and is defined in Definition 4. When $\lambda < \lambda_0$, then we observe that $x_p^* < N_p$ and $x_e^* < N_e$. We observe that

$$\begin{aligned} &\min_{\delta_p, f_p, f_e} P_{\text{tot}}(p, N\lambda\delta_p, N_p, f_p) + P_{\text{tot}}(e, N\lambda\delta_e, N_e, f_e) \\ &\geqslant \min_{\delta_p} N\lambda(\delta_p c_p^* + \delta_e c_e^*). \end{aligned} \tag{24}$$

The result follows from the fact that the minimum aggregate power consumption across all cores is a convex combination.