2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) | 978-1-7281-7586-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/PIMRC50174.2021.9569613

Optimal CPU Frequency Scaling Policies for Sustainable Edge Computing

Yu Luo*, Lina Pu†, and Chun-Hung Liu*
* Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS, 39762
†Department of Computer Science, University of Alabama, Tuscaloosa, AL 35487
e-mail: yu.luo@ece.msstate.edu; lina.pu@ua.edu; chliu@ece.msstate.edu

***Abstract*— Sustainable edge computing (SEC) is a promising technology that can reduce energy consumption and computing latency for the mobile Internet of things (IoT). By collecting solar or wind energy from the environment, an SEC cloudlet outside the electric grid can provide powerful computing capabilities for resource-constrained mobile IoT devices. Considering significant density variation of sustainable energy over time, the SEC cloudlet needs to dynamically adjust the clock frequency of the central processing unit (CPU) to balance energy consumption and computing power. In this paper, we consider the limited energy storage of the cloudlet and develop an offline optimal CPU frequency scaling policy to maximize the overall computing power of the cloudlet within a certain period of time. The tightest string policy that gives a graphical viewpoint of the optimal CPU frequency scaling is found.**

I. Introduction

In recent years, mobile edge computing (MEC) is emerging as a new computing paradigm for the mobile Internet of things (IoT) [1]. By deploying small-scale servers, called cloudlets, at the edge of the Internet, IoT devices can offload computing tasks to the cloudlet for preprocessing, thereby significantly reducing response time and saving network bandwidth between the IoT network and cloud servers.

Current research on MEC assumes that the cloudlet is always connected to the electric grid. In this case, how to minimize the energy consumption of the cloudlet while meeting the deadline of each task is the main objective. To achieve this goal, strategies that offload tasks from mobile devices to the cloudlet and manage the computing power of the cloudlet have been comprehensively studied [2]–[4].

Sustainable cloudlets that can harvest solar or wind energy will greatly improve the scalability and sustainability of existing mobile IoT networks and allow the deployment of cloudlets in the area where power grids are not available. As will be introduced in the paper, it is realistic to use solar or wind energy to power a high-performance cloudlet to provide resource-limited IoT devices with considerable computing power to perform complex tasks in the wild.

In SEC, managing the computing power of cloudlets is a new challenge, as the power density of wind and solar energy is not constant, but changes over time [5]. The energy harvested from the environment may not always allow cloudlets to run at full speed. We should carefully study how to manage power consumption of the cloudlet in a dynamic energy environment to maximize computing performance.

In order to effectively use the harvested energy, the clock frequency of the central processing unit (CPU) needs to be adjusted carefully since it determines the computing power of the cloudlet. Generally, the energy consumption rate of CPU is approximately proportional to the square of the clock frequency [6]. As a result, from the energy perspective, increasing the clock frequency to improve computing power is not energy efficient. The optimal CPU frequency scaling becomes a challenging task as both the energy efficiency and the limitations of task deadlines need to be considered.

In this paper, we consider the scenario where the energy that can be collected in the future is known. We aim at optimizing the overall computing power so the total number of computing tasks that can be completed by cloudlet in a specific time period is maximized. To achieve this objective, we build a model that can convert the CPU frequency management into an optimization problem with a concave objective function and several convex constraints. The constraints can prevent the cloudlet from violating the energy storage limitation and the energy causality constraint (sustainable energy cannot be used before it arrives). By solving the Karush-Kuhn-Tucker (KKT) conditions, we obtain the tightest string policy that provides a graphical viewpoint of the optimal CPU frequency scaling.

The rest of the paper is organized as follows: Section II introduces the related work. The system model of the CUP frequency management for the SEC cloudlet is given in Section III. In Section IV, we formulate the optimization problem to maximize the computing power of the cloudlet. The solutions are provided in Section V and evaluated in Section VI. We conclude our work in Section VII.

II. Related Work

CPU frequency scaling and task offloading play a crucial role in MEC to minimize the energy consumption of cloudlets without missing task deadlines of resource-limited devices. It needs to comprehensively consider the deadline of each task, the quality of the wireless channel, the topology of the tasks (sequential, parallel, or general dependency), and the power consumption of communications and computations and properly allocate energy and time between computation and wireless communication [2]–[4].

In [2], joint task allocation and CPU frequency scaling are modeled as a stochastic optimization problem, which takes into account the dynamics of task arrival and variation of

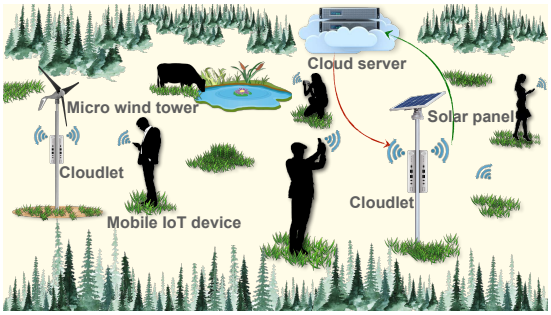

Figure 1. Network architecture of SEC with mobile IoT devices.

wireless channel states. This work aims at minimizing the energy consumption subject to the queue delay constraint of the cloudlet. In [3], it assumes that mobile devices can harvest energy from environments. How to manage the transmission power of the mobile device and the CPU frequency of the cloudlet is studied. The goal of that work is to minimize the execution delay and task dropping cost of the device. In [4], the authors consider an application that combines MEC with the wireless power transfer (WPT) technology, where a power station can transmit energy packets to charge mobile devices. In the work, both the binary offload scheme and the partial offload scheme are studied to maximize the computation efficiency (the ratio of the total computed bits to the consumed energy) of the cloudlet.

Unlike existing research, we assume that the cloudlet rather than IoT devices can collect renewable energy to drive a high-performance CPU. The feasibility of powering cloudlet with renewable energy and how to optimize the CPU frequency of the cloudlet in a dynamic energy environment have not yet been studied, which are the focus of this paper.

III. System Model

In this section, we briefly introduce the SEC architecture with mobile IoT. Then, an optimization problem is formulated to optimize the performance of computation-intensive IoT networks.

A. Network Architecture

We consider a mobile IoT network, where many mobile devices are running computation-intensive applications in the wild, such as AR and target recognition. Due to size, energy, and heat dissipation limitations, performing all tasks locally is inefficient. In order to reduce energy consumption and computing latency, mobile devices offload their tasks to nearby sustainable cloudlets, as shown in Fig. 1.

Different from mobile IoT devices, the cloudlet can have a relatively large size, so that it can scavenge solar or wind energy from the surrounding environment to power high-performance CPUs. Taking wind energy as an example, even with a micro wind turbine (weight: 12 kg, rotor-swept area: 1.2 m^2), it can harvest energy at 177 W and 524 W power when the wind speeds are 11 m/s (24.6 mph) and 20 m/s (44.7 mph), respectively [7]. These energy harvesting rates are sufficient to drive high-performance server processors, such as Intel Xeon Gold 6328HL [8] or AMD EPYC 7501 [9], the thermal design point (TDP) of which are 165 W and 175 W, respectively.

Taking solar energy as another example, the peak intensity of solar energy in non-shaded areas can reach 600 W/m^2 [10]. The power conversion efficiency of commercial solar panels is between 15% and 20% [11]. Therefore, a two square meters (1.4 m$\times$1.4 m) solar panel can generate 180 W to 240 W of power, which can easily drive high-performance CPUs.

After receiving computing tasks from mobile devices, the cloudlet will preprocess the raw data (e.g., feature extraction or data compression), and then upload useful information to the cloud server via satellite internet constellation (e.g., Starlink by SpaceX [12]) for further data processing, or directly download computing results to IoT devices. With the assistance of the cloudlet, internet traffic can be greatly alleviated and the workload of mobile devices can be significantly reduced.

B. Mathematical Model

Denote the clock frequency of the cloudlet CPU at time t by $f_c(t)$. According to processor design [13], the CPU's power consumption can be divided into three parts: the short-circuit power, the transistor leakage power, and the dynamic power, where the last part dominates the others when the CPU is running. Therefore, we use the dynamic power to approximate the total power consumption of the cloudlet.

According to the circuit theory [6], the dynamic power at time t, which is denoted by $P_d(t)$, can be calculated as

$$P_d(t) = \alpha f_c(t) V_c^2(t), \tag{1}$$

where α is a constant related to the processor architecture and $V_c(t)$ is the CPU power supply voltage at time t. Furthermore, f_c is proportional to V_c [13]. By adopting dynamic voltage scaling (DVS) technology, modern processors can dynamically scales down the voltage based on the frequency requirement [14]. Consequently, the CPU clock frequency can be written as a function of the dynamic power:

$$f_c(t) = \beta P_d^{\frac{1}{3}}(t), \quad P_d \geq 0, \tag{2}$$

where β is the frequency scaling coefficient, which is a constant greater than zero.

In nature, the power density of sustainable energy changes over time. Let $p_h(t)$ be the incident power at time t. According to the energy causality, the cloudlet cannot use the energy that has not yet arrived. As a result, the total energy consumed by cloudlet cannot exceed the cumulative harvested energy, i.e.,

$$\int_0^t P_d(u)\,\mathrm{d}u \leq \int_0^t p_h(u)\,\mathrm{d}u, \qquad \forall t \geq 0. \tag{3}$$

Assume the maximum capacity of the energy storage used by the cloudlet is E_{max}. To avoid battery overflow, the difference between the cumulative harvested energy and the total consumed energy cannot over E_{max}, i.e.,

$$\int_0^t p_h(u)\,\mathrm{d}u - \int_0^t P_d(u)\,\mathrm{d}u \leq E_{max}, \quad \forall t \geq 0. \tag{4}$$

The computing power of the cloudlet linearly increases with the CPU clock frequency to execute a fixed amount of machine

codes. Without loss of generality, assuming that it takes CPU an average of one clock cycle to execute a machine code, then according to the relationship between f_c and P_d given in (2), we can formulate the following optimization problem to optimize the overall computing power, so the total number of machine codes that can be executed by the cloudlet within time period, $[0,\, T_p]$, is maximized:

$$\begin{aligned} &\textbf{\textit{P1:}}\ \max_{P_d(t)\geq 0} \int_0^{T_p} \beta P_d^{\frac{1}{3}}(t)\,\mathrm{d}t, && T_p>0,\\ \textbf{s.t.}\ &\textbf{\textit{C1:}}\ \int_0^t P_d(u)\,\mathrm{d}u \leq \int_0^t p_h(u)\,\mathrm{d}u, && \forall t\in[0,T_p],\\ &\textbf{\textit{C2:}}\ \int_0^t p_h(u)\,\mathrm{d}u - \int_0^t P_d(u)\,\mathrm{d}u \leq E_{max}, && \forall t\in[0,T_p]. \end{aligned} \tag{5}$$

The objective function of **P1** enables the cloudlet to achieve the best computing performance in a certain period of time, which is suitable for computation-intensive IoT applications.

IV. Processing Power Optimization

Based on the optimization problem **P1** given in Section III-A, this section studies how to manage the CPU clock frequency in order to maximize the computing power of the cloudlet within a certain period of time.

To solve **P1**, we first discretize p_h in (5). Let Δt represent a short time period, which is an aliquot part of T_p. The time interval $[n\Delta t, (n+1)\Delta t]$ is referred to as the time slot n, which is represented by t_n. When Δt is small enough, it is reasonable to assume that the incident power of sustainable energy remains constant within a time slot, then we have that

$$p_h(t) = p_h[n], \quad t\in[n\Delta t,(n+1)\Delta t)\,,\ n=0,1,2\ldots. \tag{6}$$

Let $E_h[n] \triangleq \Delta t\, p_h[n]$ be the energy received in the n^{th} time slot, then the accumulative harvested energy by time t is

$$\int_0^t p_h(u)\,\mathrm{d}u = \sum_{i=0}^{n} E_h[i], \quad t\geq 0,\ n=0,1,2\ldots. \tag{7}$$

Substituting (7) into the constraints of (5), then we can obtain the following Lemma and Corollary:

Lemma 1. *Under the optimal policy, the CPU clock frequency remains unchanged within a time slot.*

Corollary 1. *For a given total amount of energy consumed in a certain period of time, the computing power can be maximized if the CPU clock frequency remains the same.*

Proof. As shown in (2), the CPU clock frequency is a concave function of the dynamic power. Therefore, the proof of Lemma 1 and Corollary 1 can refer to the proof of inequality (2.8) in the BT-problem of [15]. □

According to Lemma 1, the optimal clock frequency of the CPU in time slot n can be expressed in a discrete form:

$$f_c^*(t) = f_c^*[n],\ t\in[n\Delta t,(n+1)\Delta t)\,,\ n=0,1,2\ldots. \tag{8}$$

In addition, from Lemma 1 and (2), it can be realized that $P_d(t)$ in (5) becomes a piece-wise linear function of t. Let $P_d[n]$ represent $P_d(t)$ at time $n\Delta t$, then we have that

$$f_c[n] = \beta P_d^{\frac{1}{3}}[n], \qquad n=0,\ldots,N_p, \tag{9}$$

where $N_p = T_p/\Delta t - 1$.

Through the above discretization process, the continuous optimization problem **P1** can be converted into a piece-wise optimization problem:

$$\begin{aligned} &\textbf{\textit{P2:}}\ \max_{P_d[i]\geq 0} \sum_{i=0}^{N_p} \beta P_d^{\frac{1}{3}}[i]\Delta t, && N_p=0,1,2\ldots,\\ \textbf{s.t.}\ &\textbf{\textit{C1:}}\ \sum_{i=0}^{n} P_d[i]\,\Delta t \leq \sum_{i=0}^{n} E_h[i], && n=0,\ldots,N_p,\\ &\textbf{\textit{C2:}}\ \sum_{i=0}^{n} E_h[i] - \sum_{i=0}^{n} P_d[i]\,\Delta t \leq E_{max}, && n=1,\ldots,N_p, \end{aligned} \tag{10}$$

In the optimization problem **P2**, the objective function is concave because it is a linear combination of concave functions. In addition, **C1** and **C2** in (10) are composed of linear constraints, which are convex. Consequently, there exist KKT multiplier sets $\boldsymbol{\mu} = \{\mu_0,\ldots,\mu_{N_p}\}$ and $\boldsymbol{\lambda} = \{\lambda_1,\ldots,\lambda_{N_p+1}\}$ to make the following conditions hold [16]:

Stationarity:

$$\begin{aligned} \nabla_{P_d^*[i]}\mathcal{L} = \nabla_{P_d^*[i]}&\left(\sum_{i=0}^{N_p}\beta P_d^{\frac{1}{3}}[i]\Delta t\right)\\ &-\sum_{n=0}^{N_p}\mu_n\nabla_{P_d^*[i]}\left(\sum_{i=0}^{n}P_d[i]\,\Delta t-\sum_{i=0}^{n}E_h[i]\right)\\ &-\sum_{n=1}^{N_p}\lambda_n\nabla_{P_d^*[i]}\left(\sum_{i=0}^{n}E_h[i]-\sum_{i=0}^{n}P_d[i]\,\Delta t-E_{max}\right)=0, \end{aligned} \tag{11}$$

where $\mathcal{L}$ is the Lagrangian function and $\nabla_x(\cdot)$ denotes the partial derivative with respect to x.

Complementary slackness:

$$\begin{cases} \mu_n\left(\sum_{i=0}^{n}P_d[i]\,\Delta t-\sum_{i=0}^{n}E_h[i]\right)=0, & n=0,\ldots,N_p, \quad (12)\\ \lambda_n\left(\sum_{i=0}^{n}E_h[i]-\sum_{i=0}^{n}P_d[i]\Delta t-E_{max}\right)=0, & n=1,\ldots,N_p. \quad (13) \end{cases}$$

Dual feasibility:

$$\begin{cases} \mu_n\geq 0, & n=0,\ldots,N_p,\\ \lambda_n\geq 0, & n=1,\ldots,N_p. \end{cases} \tag{14}$$

In the following section, we will introduce how to use the KKT conditions to find the optimal solution for **P2** from the graphical perspective.

V. Graphical Perspective of Processing Power Optimization

This section studies how to adjust the CPU clock frequency from the graphical point of view to maximize the computing power of the cloudlet. We first construct an energy feasibility tunnel based on the constraints of the energy causality and the energy storage capacity. Afterward, based on Lemma 1 and Corollary 1, we give a solution to **P2**, which is called the tightest string policy.

A. Energy Feasibility Tunnel

In Fig. 2, X-axis is the time and Y-axis is the accumulative energy. The energy feasibility tunnel forms the constraints to the dynamic energy based on **C1** and **C2**. As shown in the figure, at any time, the dynamic energy cannot fall outside the tunnel: If it excesses the upper bound of the tunnel, the energy causality constraint will be violated; if it is below the lower bound of the tunnel, the constraint of the energy storage capacity will not hold.

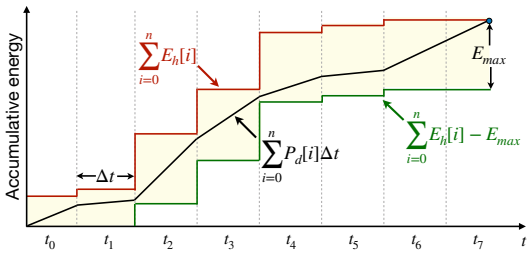

Figure 2. Energy feasibility tunnel

The dynamic energy curve, i.e., $\sum_{i=0}^{n} P_d[i]\,\Delta t$, is admissible if it is in the energy feasibility tunnel. In problem **P2**, we aim at finding an admissible curve to maximize $\mathscr{F}(P_d)$, where

$$\mathscr{F}(P_d) = \sum_{i=0}^{N_p} \beta P_d^{\frac{1}{3}}[i], \qquad N_p = 0, 1, 2 \ldots. \tag{15}$$

The accumulative energy consumed by the CPU increases monotonically with time, therefore according to Lemma 1, we can obtain the following Corollary:

Corollary 2. *Under the optimal policy, the dynamic energy curve touches neither the upper bound nor the lower bound of the energy feasibility tunnel in a time slot.*

Proof. The accumulative energy increases monotonically with time. Therefore, if the dynamic energy curve touches the lower bound of the energy feasibility tunnel at time T_l, where $T_l \in (m\Delta t, (m+1)\Delta t)$, the curve will be below the lower bound of the energy feasibility tunnel between $(m\Delta t, T_l)$, which violates the constraint of energy storage capacity (i.e., curve (e) in Fig. 3).

According to Lemma 1, if the optimal dynamic energy curve reaches the upper bound of the energy feasibility tunnel at time T_u, then the curve will exceed the upper bound of the energy feasibility tunnel between $(T_u, (m+1)\Delta t)$, which violates the energy causality constraint (i.e., curve (d) in Fig. 3). □

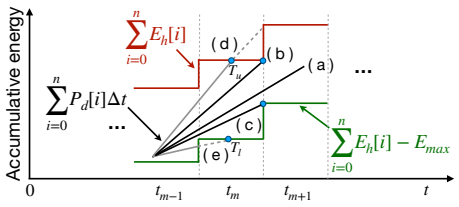

Figure 3. Three possible states of the dynamic energy curve and the end of time slot m.

According to Corollary 2 and the constraints of energy causality and energy storage capacity, at the end of t_m, the optimal dynamic energy curve has only three potential states: (a) passing through the energy feasible tunnel, (b) reaching the upper bound of the tunnel, and (c) touching the lower bound of the tunnel shown in Fig. 3. States (d) and (e) will not occur.

B. Solution of Processing Power Optimization

By solving the KKT stationarity condition in (11), the following result can be obtained:

$$P_d^*[i] = \left[\frac{3}{\beta}\left(\sum_{n=i}^{N_p}\mu_n - \sum_{n=i}^{N_p}\lambda_n\right)\right]^{-\frac{3}{2}}, \quad N_p = 0, 1, 2 \ldots. \tag{16}$$

Combining (16) with the complementary slackness and the dual feasibility of the KKT conditions, we can obtain the following lemmas:

Lemma 2. *Under the optimal policy, the CPU clock frequency remains unchanged when the energy storage is neither full nor empty (i.e.,* $0 < \sum_{i=0}^{m}(E_h[m] - P_d[i]\Delta t) < E_{max}$, $\forall\, m \in [0, N_p - 1]$: $f_c^*[m] = f_c^*[m+1]$*).*

Proof. The dynamic energy curve in Lemma 2 corresponds to the state (a) in Fig. 3. In this state, we have that

$$\begin{cases} \sum_{i=0}^{m} P_d[i]\,\Delta t - \sum_{i=0}^{m} E_h[i] \neq 0, & (17) \\ \sum_{i=0}^{m} E_h[i] - \sum_{i=0}^{m} P_d[i]\Delta t - E_{max} \neq 0. & (18) \end{cases}$$

Substituting (17) and (18) into the complementary slackness of the KKT conditions, (12) and (13), it can be obtained that $\mu_m = 0$ and $\lambda_m = 0$. Then, according to (16), it can be obtained that

$$\begin{aligned} P_d^*[m] &= \left[\frac{3}{\beta}\left(\mu_m + \sum_{n=m+1}^{N_p}\mu_n - \lambda_m - \sum_{n=m+1}^{N_p}\lambda_n\right)\right]^{-\frac{3}{2}} \\ &= \left[\frac{3}{\beta}\left(\sum_{n=m+1}^{N_p}\mu_n - \sum_{n=m+1}^{N_p}\lambda_n\right)\right]^{-\frac{3}{2}} = P_d^*[m+1]. \end{aligned} \tag{19}$$

From (19) and the relation between the CPU clock frequency and the dynamic power given in (9), we have that $f_c^*[m+1] = f_c^*[m]$. □

Lemma 3. *Under the optimal policy, the CPU clock frequency increases monotonically when the energy storage becomes empty (i.e., $\sum_{i=0}^{m} P_d[i]\Delta t = \sum_{i=0}^{m} E_h[m]$, $\forall\, m \in [0, N_p - 1]$: $f_c^*[m+1] \geq f_c^*[m]$).*

Proof. The dynamic energy curve in Lemma 3 corresponds to the state (b) in Fig. 3. In this state, we have that

$$\begin{cases} \sum_{i=0}^{m} P_d[i]\,\Delta t - \sum_{i=0}^{m} E_h[i] = 0, & (20) \\ \sum_{i=0}^{m} E_h[i] - \sum_{i=0}^{m} P_d[i]\Delta t - E_{max} \neq 0. & (21) \end{cases}$$

Substituting (21) into the complementary slackness of the KKT conditions given in (13), it can be obtained that $\lambda_m = 0$. Substituting (20) into (12), and then according the dual feasibility of the KKT conditions given in (14), we have that $\mu_m \geq 0$. Fianlly, according to (16), it can be obtained that

$$\begin{aligned} P_d^*[m] &= \left[\frac{3}{\beta}\left(\mu_m + \sum_{n=m+1}^{N_p}\mu_n - \lambda_m - \sum_{n=m+1}^{N_p}\lambda_n\right)\right]^{-\frac{3}{2}} \\ &\leq \left[\frac{3}{\beta}\left(\sum_{n=m+1}^{N_p}\mu_n - \sum_{n=m+1}^{N_p}\lambda_n\right)\right]^{-\frac{3}{2}} = P_d^*[m+1]. \end{aligned} \quad (22)$$

From (22) and the relation between the CPU clock frequency and the dynamic power described in (9), we have that $f_c^*[m+1] \geq f_c^*[m]$. □

Lemma 4. *Under the optimal policy, the CPU clock frequency decreases monotonically when the energy storage becomes full (i.e., $\sum_{i=0}^{m} P_d[i]\Delta t = \sum_{i=0}^{m} E_h[m] + E_{max}$, $\forall\, m \in [0, N_p - 1]$: $f_c^*[m+1] \leq f_c^*[m]$).*

Proof. The dynamic energy curve in Lemma 4 corresponds to the state (c) in Fig. 3. In this state, we have that

$$\begin{cases} \sum_{i=0}^{m} P_d[i]\,\Delta t - \sum_{i=0}^{m} E_h[i] \neq 0, & (23) \\ \sum_{i=0}^{m} E_h[i] - \sum_{i=0}^{m} P_d[i]\Delta t - E_{max} = 0. & (24) \end{cases}$$

Substituting (23) into the complementary slackness of the KKT conditions given in (12), we can obtain $\mu_m = 0$. Substituting (24) into (13), and then according to the dual feasibility of the KKT conditions given in (14), we have $\lambda_m \geq 0$. Finally, according to (16), it can be obtained that

$$\begin{aligned} P_d^*[m] &= \left[\frac{3}{\beta}\left(\mu_m + \sum_{n=m+1}^{N_p}\mu_n - \lambda_m - \sum_{n=m+1}^{N_p}\lambda_n\right)\right]^{-\frac{3}{2}} \\ &\geq \left[\frac{3}{\beta}\left(\sum_{n=m+1}^{N_p}\mu_n - \sum_{n=m+1}^{N_p}\lambda_n\right)\right]^{-\frac{3}{2}} = P_d^*[m+1]. \end{aligned} \quad (25)$$

From (25) and the relation between the CPU clock frequency and the dynamic power given in (9), we have that $f_c^*[m+1] \leq f_c^*[m]$. □

Lemma 5. *Under the optimal policy, the cloudlet consumes all the harvested energy by the end of the last slot (i.e., $\sum_{i=0}^{N_p} P_d^*[i]\,\Delta t = \sum_{i=0}^{N_p} E_h[i]$).*

Proof. If energy is not exhausted in the last time slot with the optimal $P_d^*[i]$, $i = 1, \ldots, N_p$, we can always find $P_d'[N_p] > P_d^*[N_p]$ that consumes all collected energy. Because $\mathscr{F}(\cdot)$ in (15) is a monotonically increasing function of P_d, thus $\mathscr{F}(P_d'[N_p]) > \mathscr{F}(P_d^*[N_p])$, which means that $P_d^*[N_p]$ is not optimal. Therefore, the optimal policy must consume all harvested energy in the last time slot. □

C. Tightest String Policy

Here, we use an example to introduce how to utilize the Lemmas and Corollaries proved in Section IV and Section V-B to optimize the computing power for the cloudlet.

As shown in Fig. 4, we first mark several turning points in the feasible energy tunnel as p_i, $i = 0, \ldots, 8$, where p_0 is the starting point. The optimal dynamic energy curve can be obtained through the following steps.

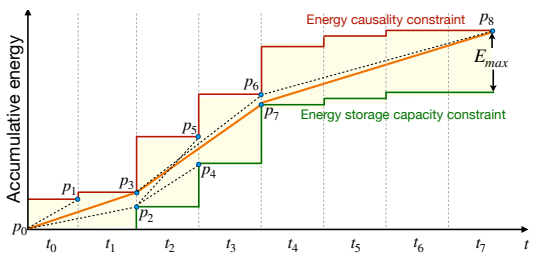

Figure 4. The tightest string policy for computing power optimization.

Step 1: Connect p_0 with all turning points in the energy feasibility tunnel. Remove the strings that have any portion fall outside the tunnel. The rest of the strings, which are $\overline{p_0p_1}$, $\overline{p_0p_2}$, and $\overline{p_0p_3}$, are considered as admissible starting strings.

Step 2: According to Corollary 2, among all admissible starting strings, we keep the one with the longest duration and remove others. If two starting strings have the longest duration, such as $\overline{p_0p_2}$ and $\overline{p_0p_3}$ in Fig. 4, and then go to the next step to examine each retained string.

Step 3: We check $\overline{p_0p_2}$ first. Let p_2 be the new starting point, and then repeat Step 1 and Step 2 to obtain all admissible strings, $\overline{p_2p_4}$ and $\overline{p_2p_5}$. Because $\overline{p_0p_2}$ hits the lower bound of the energy feasibility tunnel, which means that the energy storage is full. In this case, according to Lemma 4, the CPU will reduce the frequency and the dynamic power in the next slot. As a result, the slope of $\overline{p_2p_i}$ must be smaller than that of $\overline{p_0p_2}$. However, it can be observed from Fig. 4 that the slopes of $\overline{p_2p_4}$ and $\overline{p_2p_5}$ are both greater than the slop of $\overline{p_0p_2}$. Therefore, $\overline{p_0p_2}$ needs to be removed from the admissible strings, and only $\overline{p_0p_3}$ is retained.

Step 4: Let p_3 be the new starting point, then we repeat Step 1 and Step 2 to obtain all admissible strings, which are $\overline{p_3p_6}$ and $\overline{p_3p_7}$. We first check $\overline{p_3p_6}$. According to Lemma 5, the

cloudlet must spend all received energy at the end of the time slot. Therefore, the end point of the dynamic energy curve is p_8. As shown in the figure, $\overline{p_3p_6}$ reaches the upper bound of the energy feasibility tunnel. In this case, the CPU will increase the frequency and the dynamic power in the next slot based on Lemma 3. Therefore, the slope of $\overline{p_6p_i}$ should be greater than that the slope of $\overline{p_3p_6}$, which is unsatisfactory. Therefore, $\overline{p_3p_6}$ is removed from the admissible strings. Finally, $\overline{p_3p_7}$ and $\overline{p_7p_8}$ are retained as the optimal solution.

Step 5: After obtaining the optimal strings through Step 1 to Step 5, the optimal dynamic power of the CPU in each time slot is available, which is the slope of the strings. Finally, the optimal CPU frequency can be calculated by (9).

The optimal strings obtained through the above steps are the tightest that follows the Lemmas. The corresponding CPU frequency scaling policy is called the tightest string policy. To be specific, assume that a thread ball is placed in the energy feasibility tunnel. We tie one end of the thread ball to the starting point p_0, and then withdraw the thread at the endpoint p_8. The process will not stop until the thread is fully tightened. Finally, the thread left in the tunnel has the shortest length and the tightest shape. Similar observation can also be found in the transmission scheduling of wireless communications with deadline constraints [15].

VI. Performance Evaluation

This section evaluates the performance of different CPU frequency scaling strategies. We first briefly introduce the simulation configuration. Then, we compare the performance of three different offline strategies.

A. Simulation Setup

In the simulation, we will evaluate the performance of the optimal CPU frequency scaling policy in different settings, and compare it with the following two frequency management strategies:

- **Average scaling strategy:** In this strategy, the cloudlet estimates the average of the energy harvesting rate, and adjusts the CPU frequency so that the dynamic power is equal to the average energy harvesting rate.
- **Greedy scaling strategy:** In this strategy, the cloudlet will spend all the collected energy at the end of each time slot to maximize short-term computing power.

The simulations are carried out on MATLAB. In the simulation, the length of each time slot is $\Delta t = 10$ min. The default value of the average energy harvesting rate is $\bar{p}_h = 96$ W, thus the average amount of energy that the cloudlet can receive in each time slot is $\bar{E}_h = 16$ Wh or 5.76×10^4 J. The frequency scaling coefficient is set to $\beta = 0.565 \times 10^9$. With this frequency scaling coefficient, when the dynamic power is 10 W and 150 W, the CPU clock frequencies are 1.2 GHz and 3.0 GHz, respectively.

B. Performance Comparison

In Fig. 5, we evaluate the impact of the energy storage capacity on the average clock rate ($\bar{f}_c$) of different CPU scaling strategies. The amount of energy harvested in each time slot follows a truncated Gaussian distribution that lies between 0 Wh and 50 Wh with a mean of 16 Wh and a standard deviation of 10 Wh. The truncated Gaussian distribution guarantees that the energy harvesting rate is a positive value less than 300 W, which is the peak energy harvesting rate that a 2.5 m^2 solar panel with 20% energy conversion efficiency can be achieved in an unshadowed area. The results presented in the figure are the average of 10 simulations.

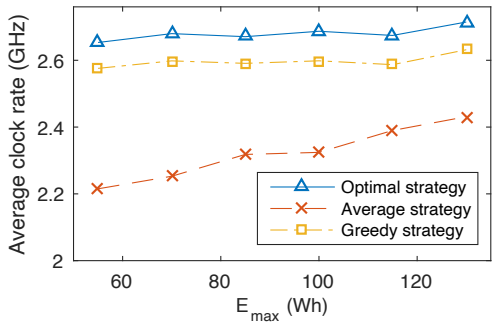

Figure 5. Impact of energy storage capacity on the performance of different CPU frequency management strategies.

As illustrated in Fig. 5, the performance of the average CPU frequency scaling strategy is much lower than that of the other two strategies, especially when the energy storage capacity is low. In addition, the optimal strategy and the greedy strategy are less affected by the variation of E_{max}. As a result, in the optimal strategy and the greedy strategy, $\bar{f}_c$ is almost a constant relative to E_{max}. For example, when the maximum capacity of energy storage increases from 55 Wh to 130 Wh, $\bar{f}_c$ with the average frequency management strategy increases from 2.22 GHz to 2.43 GHz. In this case, the computing power of the cloudlet is improved by 8.6%. On the contrast, if the optimal strategy is adopted, $\bar{f}_c$ slightly increases from 2.65 GHz to 2.71 GHz. As a result, there is only a 2.5% improvement in computing power.

Compared with the optimal strategy, the performance of the greedy strategy is relatively low due to the inefficient energy utilization. Specifically, as shown in (2), the clock frequency of the CPU is a concave function of the dynamic power. Therefore, it is not efficient to improve the computing power of the cloudlet by increasing the dynamic power. In fact, if the maximum capacity of the energy storage is unlimited and all tasks have no deadline, the most efficient way to use the harvested energy is to perform the task at the lowest frequency, thereby maximizing the total clock cycles for a given energy consumption. However, the greedy strategy consumes all collected energy at the highest possible power. The energy collected in past time slots cannot be saved for future use, resulting in low energy utilization.

Compared with the greedy strategy, the average CPU frequency scaling policy has higher energy efficiency due to the lower average dynamic power. However, the energy harvesting efficiency of the average strategy is relatively low because it uses the collected energy in a conservative manner. As a consequence, when the dynamic power of the average

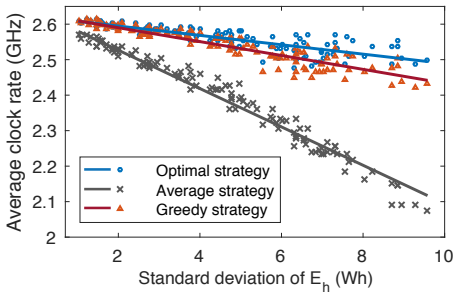

Figure 6. Impact of energy variation on the performance of different CPU frequency management strategies.

strategy is lower than the energy harvesting rate, the energy storage may overflow. Increasing the maximum capacity of the energy storage can improve the performance of the average CPU frequency scaling strategy, making it gradually approach the optimal solution by reducing the probability of battery overflow.

In Fig. 6, we evaluate the impact of energy dynamics on the performance of the three CPU frequency management strategies. In the figure, the maximum capacity of the energy storage and the average energy harvesting rate are set to $E_{max} = 85\,\text{Wh}$ and $\bar{p}_h = 16\,\text{Wh}$, respectively. Denote the standard deviation of E_h as σ_h, which can indicate the variation of energy intensity. In order to clearly show the relationship between the performance of the three strategies and σ_h, we perform the linear fit to all discrete points, and show the results with the solid line in the figure.

As shown in Fig. 6, when the fluctuation of energy intensity becomes large, the performance of all three CPU management strategies decreases linearly. According to Lemma 2, it can be realized that an ideal energy feasibility tunnel should be the one that allows the CPU frequency to remain constant throughout the whole tunnel. In this case, the tightest string will be a single line segment connecting the start and end of the tunnel. To achieve this, the height of the steps in the energy feasibility tunnel needs to be consistent, which requires the fluctuation of the energy intensity to be as small as possible. Otherwise, the slope of the tightest string will keep changing to meet the constraints of the energy causality and energy storage capacity, thereby reducing the efficiency of energy utilization.

From Fig. 6, it can be observed that the average frequency management strategy is more sensitive to energy variation than the greedy strategy and the optimal policy. For instance, when the standard deviation of E_h increase from 1 Wh to 9 Wh, $\bar{f}_c$ with the average frequency management strategy is reduced from 2.58 GHz to 2.15 GHz, that is, the computing power decreases by 16.7%. In the same situation, the average clock rate of the greedy strategy is slightly reduced from 2.61 GHz to 2.45 GHz, that is, the computing power is only reduced by 6.1%. This is because, with the increase of σ_h, the harvested energy is very likely to be underutilized. As a result, there is a high probability that the energy storage overflows in some time slots, which greatly reduces the energy harvesting efficiency.

VII. Conclusion

In this paper, we studied the CPU frequency scaling problem of the cloudlet with energy harvesting capability. The optimization problem subject to the constraints of energy causality and energy storage capacity is developed to maximize the computing power of the CPU within a certain period of time. By solving the KKT conditions, we conclude several lemmas which is later used to obtain the optimal CPU frequency management, namely, the tightest string policy. The tightest string policy provides a graphical viewpoint of optimal CPU frequency scaling.

Acknowledgement

This work is supported in part by the US National Science Foundation under Grant No. 2051356.

References

[1] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: architecture, advances and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[2] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "TOFFEE: task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, 2019.

[3] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[4] F. Zhou and R. Q. Hu, "Computation efficiency maximization in wireless-powered mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3170–3184, 2020.

[5] J. Cochran, M. Miller, O. Zinaman, M. Milligan, D. Arent, B. Palmintier, M. O'Malley, S. Mueller, E. Lannoye, A. Tuohy *et al.*, "Flexibility in 21st century power systems," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2014.

[6] K. De Vogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "The energy/frequency convexity rule: modeling and experimental validation on mobile devices," in *proceedings of International Conference on Parallel Processing and Applied Mathematics*. Springer, 2013, pp. 793–803.

[7] H. Mamur, "Design, application, and power performance analyses of a micro wind turbine," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 23, no. 6, pp. 1619–1637, 2015.

[8] Intel Company, "Intel Xeon Gold 6328HL Processor," 2021, [Accessed: Feb, 2021]. [Online]. Available: https://www.intel.com/content/www/us/en/products/processors/xeon/scalable/gold-processors/gold-6328hl.html

[9] AMD Company, "AMD EPYC 7501," 2021, [Accessed: Feb, 2021]. [Online]. Available: https://www.amd.com/en/products/cpu/amd-epyc-7501

[10] G. Papadakis, P. Tsamis, and S. Kyritsis, "An experimental investigation of the effect of shading with plants for solar control of buildings," *Energy and buildings*, vol. 33, no. 8, pp. 831–836, 2001.

[11] Ossila Company, "Perovskites and perovskite solar cells: an introduction," 2018, [Accessed: Feb, 2021]. [Online]. Available: https://www.ossila.com/pages/perovskites-and-perovskite-solar-cells-an-introduction

[12] SpaceX Company, "Starlink," 2021, [Accessed: Feb, 2021]. [Online]. Available: https://www.starlink.com

[13] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 13, no. 2, pp. 203–221, 1996.

[14] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *proceedings of the 41st Annual Design Automation Conference*, 2004, pp. 868–873.

[15] M. A. Zafer, "Dynamic rate-control and scheduling algorithms for quality-of-service in wireless networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[16] Y. Luo, L. Pu, Y. Zhao, W. Wang, and Q. Yang, "A nonlinear recursive model based optimal transmission scheduling in RF energy harvesting wireless communications," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3449–3462, 2020.