Improving Temperature Prediction Accuracy Using Kalman and Particle Filtering Methods

Baver Ozceylan*[1], Boudewijn R. Haverkort[2], Maurits de Graaf[3], Marco E. T. Gerards[1]

[1] University of Twente, Enschede, the Netherlands
[2] Tilburg University, Tilburg, the Netherlands
[3] Thales Nederland B.V., Huizen, the Netherlands

* Corresponding Author: b.ozceylan@utwente.nl

Abstract

Predicting the device temperature is crucial for high performance mobile devices since a high temperature reduces the device reliability and lifetime, and increases the power dissipation per processing activity. For these reasons, thermal models are used to predict the temperature and schedule the workloads according to these predictions. This means that more accurate predictions can improve the reliability, lifetime and energy-efficiency of devices. We introduce two different generic methods to extend a thermal model to improve the prediction accuracy. The first method is to extend a thermal model with a Kalman filter. This approach enables a device to adapt to environmental changes more easily and to reduce the effect of noise by combining sensor data and dynamic behavior of the system. However, it assumes every random variable to be normally distributed. The second method is to extend a thermal model with a particle filter. In addition to the ability of adapting better to environmental changes, this approach enables a device to approximate any arbitrary distribution to reduce the effect of noise. Both methods are applicable to any dynamic thermal model to improve its prediction accuracy. Our experimental results show that the new methods indeed improve the prediction accuracy.

1 Introduction

The computational power of mobile devices has grown dramatically over the last decades and thermal management has become a prime concern in the design of such devices. This has two main reasons. The first reason is that increased computational power aggravates the overheating of these devices, which, in turn, decreases the reliability due to the increase of failure rates under higher temperatures. Moreover, the variety of applications running on these devices may create large temporal heat dissipation bursts. This non-uniformity of the temperature profile further decreases the reliability and the performance [1]. The second reason is that leakage power, which depends exponentially on processor temperature, becomes an essential part of the total power consumption. As reported in [2], with the developments in electronic chip manufacturing technologies, transistor process geometries are reduced progressively. This leads to less power dissipation per processing activity, which implies a decrease in dynamic power. On the other hand, this reduction of transistor process geometries increases the leakage current (i.e., the leakage power), which is independent from the processing activities. Due to the decrease in dynamic power and increase in leakage power, the same use case consumes less power now than before. However, the overall power consumption has been increasing rapidly due to more advance functionality in mobile devices. Therefore, as stated in [2], leakage power is more and more dominating the total power dissipation; and because it increases exponentially with temperature, thermal management positively affects energy efficiency.

Reactive thermal management techniques activate cooling devices, such as fan or throttle processor speeds, according to the current temperature measurement; as such, reactive thermal management techniques typically take action only if a certain temperature threshold is reached. Such a method has already been implemented in the Linux Kernel as default thermal governor, which periodically samples the temperatures of the CPU cores using on-chip thermal sensors. On the other hand, *proactive* thermal management techniques are more advanced. They employ temperature predictions in order to manage the available resources more effectively, which might include actions such as adjusting the workloads and/or the operating frequency. Although proactive methods have higher overhead than reactive methods, proactive methods generally perform better for modern mobile devices due to the high computational power requirements of the applications in general [3].

The system temperature can be predicted by statistical methods, as described in [4], which use autoregressive moving averages (ARMA) to estimate future temperatures. On the other hand, more recent studies [5-8] combine statistical methods and the dynamic model of the system, thus combining past data and dynamic temperature behavior. In [5], the authors use a thermal model for such predictions. They describe an experimental setup with a furnace to find the model parameters; during the experiments, they measure the temperature and power consumption of the system (using on-board sensors). Although this method gives accurate results, it is not robust, which means that it does not adapt to environmental changes, since the parameters are determined for a specific environment. Additionally, although every

processor has at least one temperature sensor due to safety reasons, not every processor has a power sensor. [6] introduces a similar concept, however, only using temperature sensors. In [7], we introduced a thermal model and a method to find the system parameters. Our method only uses built-in temperature sensors and is applicable easily to each and every device since it does not require any additional hardware, such as a furnace or power sensor. In order to deal with different environments, [8] uses a look-up table that contains error correction coefficients for different system states. This look-up table aims to adapt the system to environmental changes.

In this paper, we propose more accurate and robust future temperature prediction methods. Although our previously introduced method [7] can be applied periodically to adapt to environmental changes, we further improve it here, by extending the model first using the Kalman filtering method and then using the particle filtering method. In addition, we experimentally validate the new methods, that is, we compare (i) our original model, (ii) the model improved with a Kalman filter, and (iii) the model improved with a particle filter. We use an ODROID-XU4 platform for the experiments. This platform is composed of the Samsung Exynos 5422 System-on-Chip. It contains a big cluster, which consist of four ARM Cortex-A15 CPUs, and a small cluster, which consist of four ARM Cortex-A7 CPUs. It also contains four built-in temperature sensors. In this paper, we use the average of the four sensor measurements as the observed temperature of the system. Additionally, this system allows us to control the operating frequency of each cluster separately. Hence, we group the CPUs corresponding to each cluster. This means that our experimental system has two processing units that affects the system temperature according to the rate of processing activities.

Our aim is to accurately predict the system temperature using a thermal model. In Section 2, we briefly explain the thermal model from [7] and the prediction method based on only this model. Then, we introduce two different methods to improve the prediction accuracy. In Section 3, we explain the Kalman filtering method and extend our model with a Kalman filter. In Section 4, we explain the particle filtering method and extend our model with a particle filter. In Section 5, we evaluate these two extended methods and discuss the benefits of them. Section 6 concludes the paper.

This paper makes the following main contributions:

- We improve the accuracy of temperature predictions using a Kalman filter and validate this with experimental results.
- We improve the accuracy of temperature predictions using a particle filter and validate this with experimental results.
- We explain and compare the advantages of these two new methods.
- Any thermal model can be improved with these methods to predict the temperature.

2 Thermal Model

The thermal models in the literature [5-8] for computing the future temperature based on the current temperature and the rate of processing activities are mostly derived from the well-known heat transfer equation:

$$C\frac{dT(t)}{dt} = -GT(t) + P(t), \qquad (1)$$

where C and G are thermal conductance and capacitance. respectively. $P(t)$ and $T(t)$ represent respectively the power dissipation and the temperature at time t. In this paper, we use our previous work [7] to model the power dissipation as $P(t) = G(T_a(t) + u(t))$, where $u(t)$ represents the rate of processing activities and $T_a(t)$ is the ambient temperature, which depends on the environment. Although the ambient temperature can change over time, we consider it constant in this section since the change is slow. Therefore, the first order differential equation (1) can be solved as:

$$T(t) = T_0 e^{\frac{-t}{\tau}} + T_a\left(1 - e^{\frac{-t}{\tau}}\right) + \frac{e^{\frac{-t}{\tau}}}{\tau}\int_0^t u(\sigma)e^{\frac{-\sigma}{\tau}}d\sigma, \qquad (2)$$

where T_0 refers to the current temperature and τ is the thermal time constant, which equals C/G. According to our model in [7], we write the rate of processing activities as:

$$u(t) = \boldsymbol{c}^T(t)\boldsymbol{\Gamma}\boldsymbol{f}(t), \qquad (3)$$

where $\boldsymbol{\Gamma}$ is a coefficient matrix, $\boldsymbol{f}(t)$ is the frequency vector and $\boldsymbol{c}(t)$ is the utilization vector. Since our experimental system has two processing units, we define the frequency vector $\boldsymbol{f}(t) = [1, f_1(t), f_2(t)]^T$, where $f_1(t)$ and $f_2(t)$ refer to the operating frequency of the big and small clusters over time, respectively, and the utilization vector $\boldsymbol{c}(t) = [1, c_1(t), c_2(t)]^T$, where $c_1(t)$ and $c_2(t)$ refer to the utilization ratio of the big and small clusters over time, respectively. We use our method in [7] to determine the model parameters, which are $\boldsymbol{\Gamma}$, τ and T_a.

We now explain the prediction procedure using only the thermal model (2). In our experiments, the frequency and utilization of each processing unit, i.e., $f_1(t)$, $f_2(t)$, $c_1(t)$ and $c_2(t)$, are randomly changed every 10 seconds. The observed temperature data are collected with the sampling period Δ using the on-chip thermal sensors. We estimate the temperature at these discrete time points. In our experiments, Δ is around 0.55 second, which means that $\Delta \ll 10s$. For this reason, we may assume that $u(t)$ is constant between two consecutive discrete time points, i.e., $u(t) = u_k$ in the time interval $[k\Delta, k\Delta + \Delta]$ and we compute u_k using (3). We denote the observed temperature of the system by z_k, which refers the measurement at the k^{th} time point. Since the

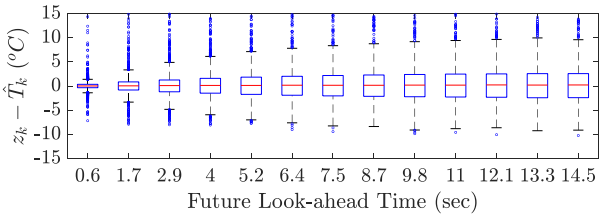

Figure 1: The error for the estimations with the standard model [7]

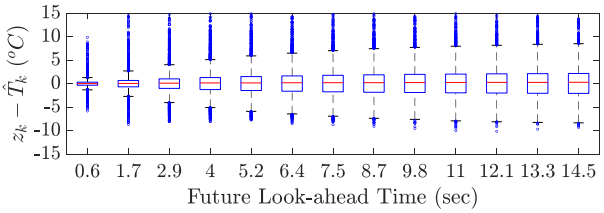

Figure 2: The error for the estimations with the model improved with a Kalman filter

measurements are noisy, we can write $z_k = T_k + \tilde{n}_s$, where T_k is the real temperature at $t = k\Delta$ and $\tilde{n}_s$ refers to the sensor noise with $\tilde{n}_s \sim N(0, \sigma_s^2)$. The estimated temperature at the k^{th} time point is denoted by $\hat{T}_k$, with $k\epsilon\{0,1,2,\dots\}$. At $t = 0$, the current temperature is estimated using the sensor data, i.e., $\hat{T}_0 = z_0$, and the future temperature is estimated using (2) as follows:

$$\hat{T}_k = \hat{T}_{k-1}\beta + T_a(1-\beta) + \frac{\beta}{\tau}\int_{(k-1)\Delta}^{k\Delta} u(\sigma)e^{\frac{-\sigma}{\tau}}d\sigma,$$

where $\beta = e^{-\Delta/\tau}$. After solving the integral, we obtain:

$$\hat{T}_k = \hat{T}_{k-1}\beta + T_a(1-\beta) + u_{k-1}(1-\beta). \qquad (4)$$

This means that we first measure the temperature to estimate the current temperature, i.e., $\hat{T}_0 = z_0$, and then predict the temperature $\hat{T}_k$, $\forall k\epsilon\{1,2,\dots\}$ by applying (4) iteratively.

For validation purposes, we conducted a 24 hours-long experiment on an ODROID-XU4 and store the frequency, processor utilization and temperature data. After collecting the frequency, utilization and temperature data, we randomly sample 10000 15-seconds-long data sets. For each data set, We use the first data point to estimate the current temperature as $\hat{T}_0 = z_0$; subsequently we estimate the future temperature with (4). Figure 1 shows the prediction error, i.e., $z_k - \hat{T}_k$, for different future time points. Each bin contains the results from 10000 time points and shows the error distribution.

The measurement noise $\tilde{n}_s$ has a significant effect on the prediction accuracy since we directly use the measurements data as $\hat{T}_0 = z_0$, which means $\hat{T}_0 = T_0 + \tilde{n}_s$, and then we use $\hat{T}_0$ to predict the future by applying (4) iteratively, which further increases the effect of the noise $\tilde{n}_s$ leading to growing confidence intervals in the estimations. Additionally, even though the knowledge of the rate of processing activities $u(t)$ can be obtained and the sampling period Δ can be fixed before executing a task, they may undergo small changes during the execution. These nondeterministic changes due to the processing activities in the system, often called the "process noise", further decreases the prediction accuracy. Furthermore, the current method is not adaptable to changes in the environment. To address these issues, we propose two different methods, which combine this model (i) with a Kalman filter (in Section 3) and (ii) with a particle filter (in Section 4).

3 Kalman Filter Extension

In this section, we aim to decrease the effect of the measurement noise and the process noise on the prediction accuracy by extending the model with a Kalman filter. The noise problems can be addressed with methods that are developed for target tracking problems [9], which estimate the position of a target by combining sensor data (such as GPS data) and the well-known relations between position, velocity and acceleration. The most common solution approach is using a Kalman filter, which requires a stochastic dynamic system and normally distributed random variables.

We develop a discrete time Kalman filter with period Δ to estimate the system temperature $T(t)$ and the ambient temperature $T_a(t)$ hence the system can adapt to changes in the environment. To ease the notation, T_k and $T_{a,k}$ refer respectively to the system temperature and the ambient temperature at the k^{th} time point. To represent the environmental changes, we define $T_{a,k} = T_{a,k-1} + \tilde{n}_a$, where $\tilde{n}_a \sim N(0, \sigma_a^2)$. The rate of processing activities has no effect on the ambient temperature, which means the process noise is uncorrelated with $\tilde{n}_a$. Hence, we can define the dynamic system (2) as follows:

$$T_k - T_{a,k} = (T_{k-1} - T_{a,k-1})\beta + u_{k-1}(1-\beta) + \tilde{n}_u, \qquad (5)$$

where $\tilde{n}_u \sim N(0, \sigma_u^2)$ represents the process noise. Because of the noise components $\tilde{n}_u$ and $\tilde{n}_a$, T_k and $T_{a,k}$ are random variables for all $k\epsilon\{1,2,\dots\}$. For the Kalman filter, we can define the state vector as $\boldsymbol{x}_k = [T_k - T_{a,k}, T_{a,k}]^T$ so that its two elements are uncorrelated. Therefore, the covariance matrix $\mathbf{Q}$ can be expressed as:

$$\mathbf{Q} = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_a^2 \end{bmatrix}.$$

Moreover, we define the control vector as $\boldsymbol{u}_k = [u_k, 0]^T$. The Kalman filtering method has two steps, which are the prediction step and the update step. In the

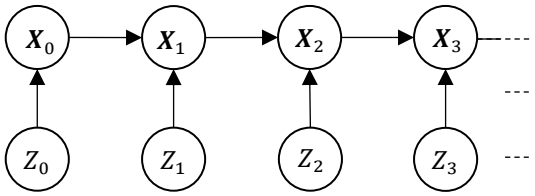

Figure 3: Markov model for the particle filter

prediction step, if $\hat{\boldsymbol{x}}_{k-1}$ is the best estimate, we compute the state vector in the next step by applying (5) as $\hat{\boldsymbol{x}}'_k = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1}$, where the prediction matrix $\boldsymbol{F}$ is:

$$\boldsymbol{F} = \begin{bmatrix} \beta & 0 \\ 0 & 1 \end{bmatrix},$$

and the control matrix as:

$$\boldsymbol{B} = \begin{bmatrix} 1-\beta & 0 \\ 0 & 0 \end{bmatrix}.$$

For each iteration, a new covariance matrix $\boldsymbol{P}_k$ is computed. This matrix expresses the uncertainty related to the previous estimation and the uncertainty from the environment. If $\boldsymbol{P}_{k-1}$ is the covariance matrix in the previous step, we compute $\boldsymbol{P}'_k = \boldsymbol{F}\boldsymbol{P}_{k-1}\boldsymbol{F}^T + \boldsymbol{Q}$. In the subsequent update state, a Kalman filter combines the estimated state in the prediction step, $\hat{\boldsymbol{x}}'_k$, with the current measurement z_k as $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}'_k + \boldsymbol{K}(z_k - \boldsymbol{H}\hat{\boldsymbol{x}}'_k)$, where $\boldsymbol{K}$ is the so-called Kalman gain, which is defined as $\boldsymbol{K} = \boldsymbol{P}'_k\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}'_k\boldsymbol{H}^T + \sigma_s^2)^{-1}$ and $\boldsymbol{H} = [1,1]$ since the measurement z_k corresponds to the system temperature T_k. In the update step, also, the covariance matrix $\boldsymbol{P}_k$ is updated with respect to the Kalman gain. The new covariance matrix to be used in the next iteration is $\boldsymbol{P}_k = \boldsymbol{P}'_k - \boldsymbol{K}\boldsymbol{H}\boldsymbol{P}'_k$. [10]

We now explain the prediction procedure using the thermal model (2) and the Kalman filter. As in the previous section, we still apply (4) iteratively to estimate the future temperature. However, we now keep track of the temperature with the Kalman filter so that we have better knowledge about the current temperature T_0 and the ambient temperature T_a. This means that, if the best estimate for the current state is $\hat{\boldsymbol{x}}_k$, we estimate the current temperature as $\hat{T}_0 = \boldsymbol{H}\hat{\boldsymbol{x}}_k$ and the ambient temperature as $T_a = \boldsymbol{C}\hat{\boldsymbol{x}}_k$, where $\boldsymbol{C} = [0,1]$. Then, we estimate the future temperature by applying the model function (4) iteratively.

For validation purposes, we conducted a 24 hours-long experiment on an ODROID-XU4 and store the frequency, processor utilization and temperature data. After collecting the frequency, utilization and temperature data, we randomly sample 10000 45-seconds-long data sets. For each data set, we use the first 30 seconds to train the Kalman filter, which means adapt to the environment; subsequently we estimate the future temperature with (4). Figure 2 shows the prediction error, i.e., $z_k - \hat{T}_k$, for different future time points, similar to Figure 1 in Section 2. We observe that the confidential intervals are smaller. This is also shown in Figure 5.

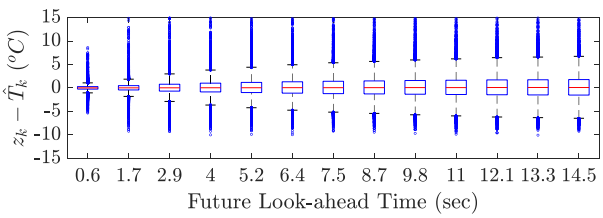

Figure 4: The error for the estimations with the model improved with a particle filter

4 Particle Filter Extension

In this section, we aim to decrease the effect of the measurement noise and process noise on the prediction accuracy by extending our model with a particle filter [11]. The particle filtering method is a sequential Monte Carlo algorithm to compute the probability distribution for a Markov process. To do so, the particle filtering method uses a set of so-called particles, i.e., a set of samples, to represent an arbitrary distribution. In this method, in order to better model the process noise, we consider the sampling period as a random variable, which means that the time between the $(k-1)^{\text{th}}$ and k^{th} time points is Δ_k. We subsequently define the state vector $\boldsymbol{x}_k = [T_k - T_{a,k}, T_{a,k}, \Delta_k]^T$. We denote its elements by $\boldsymbol{x}_k(1) = T_k - T_{a,k}$, $\boldsymbol{x}_k(2) = T_{a,k}$ and $\boldsymbol{x}_k(3) = \Delta_k$, and we define the stochastic process as $\{\boldsymbol{X}_k\}_{k\geq 0}$, which is a Markov process; Figure 3 shows the corresponding Markov model. Instead of finding the best estimates as in Section 3, we find the probability distributions in this section. If the system is in state $\boldsymbol{X}_k$, which means at the k^{th} time point, our aim is to find the probability distribution $p(\boldsymbol{X}_k = \boldsymbol{x}_k)$ given the probability distribution of the previous state. This distribution is represented by a set of particles $S^{\boldsymbol{X}}_{k-1} = \{\boldsymbol{x}^{(j)}_{k-1}\}_{j\in\{1,\dots,N\}}$ and their importance weights $S^{w}_{k-1} = \{w^{(j)}_{k-1}\}_{j\in\{1,\dots,N\}}$, where N is the number of used particles. This means that the weight $w^{(j)}_{k-1}$ represents the probability $\boldsymbol{X}_{k-1} = \boldsymbol{x}^{(j)}_{k-1}$, i.e., $p\left(\boldsymbol{X}_{k-1} = \boldsymbol{x}^{(j)}_{k-1}\right) = w^{(j)}_{k-1}$. The particle filter first creates a new set of particles for the previous state $\boldsymbol{X}_{k-1}$ as $S'^{\boldsymbol{X}}_{k-1} = \{\boldsymbol{x}^{(i)}_{k-1}\}_{i\in\{1,\dots,M\}}$, where $M \geq N$, by randomly sampling M particles from the set $S^{\boldsymbol{X}}_{k-1}$ with respect to the weights given by the set S^{w}_{k-1}. This set may contain the same element more than once. In doing so, we can represent the probability distribution of the previous state. Then, it iterates the system for each particle $\boldsymbol{x}^{(i)}_{k-1} \epsilon S'^{\boldsymbol{X}}_{k-1}$ to the current state using the model function (4) as:

$$\boldsymbol{x}^{(i)}_k(1) = \boldsymbol{x}^{(i)}_{k-1}(1)\beta^{(i)}_k + \boldsymbol{x}^{(i)}_k(2) + u_{k-1}\left(1-\beta^{(i)}_k\right), \quad (6a)$$

$$\boldsymbol{x}^{(i)}_k(2) = \boldsymbol{x}^{(i)}_{k-1}(2) + \tilde{n}_{\mathrm{a}}, \quad (6b)$$

$$\boldsymbol{x}^{(i)}_k(3) = \boldsymbol{x}^{(i)}_{k-1}(3) + \tilde{n}_{\Delta}, \quad (6c)$$

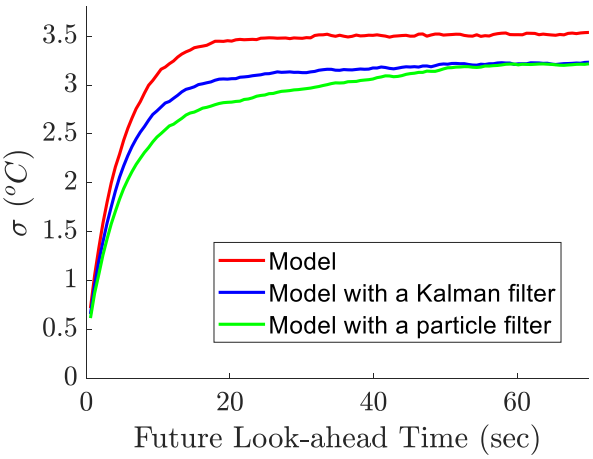

Figure 5: The standard deviation of the prediction errors

where $\beta_k^{(i)} = e^{\frac{-x_k^{(i)}(3)}{\tau}}$, $\tilde{n}_a \sim N(0, \sigma_a^2)$ and $\tilde{n}_\Delta \sim N(0, \sigma_\Delta^2)$. Hence, we have an updated set of particles for state $\boldsymbol{X}_k$ as $S_k'^{\boldsymbol{X}} = \{\boldsymbol{x}_k^{(i)}\}_{i\in\{1,\dots,M\}}$. At this point, we take a temperature measurement to find the importance weight of each particle. This means that the particle filter computes a weight for each particle $\boldsymbol{x}_k^{(i)} \epsilon S_k'^{\boldsymbol{X}}$ using the observation data z_k as $w_k^{(i)} = f(z_k|\boldsymbol{X}_k = \boldsymbol{x}_k^{(i)})$. The function $f(z_k|\boldsymbol{X}_k)$ is the likelihood function. We assume that the measurement noise is normally distributed hence:

$$f(z_k|\boldsymbol{X}_k = \boldsymbol{x}_k) = \frac{1}{\sigma_s\sqrt{2\pi}} e^{-\frac{(z_k - x_k(1) - x_k(2))^2}{2\sigma_s^2}}.$$

Therefore, we have $S_k'^{w} = \{w_k^{(i)}\}_{i\in\{1,\dots,M\}}$. In the final step, the particle filter reduces the size of the sets $S_k'^{\boldsymbol{X}}$ and $S_k'^{w}$ from M to N by eliminating the least probable particles so that we obtain $S_k^{\boldsymbol{X}}$ and S_k^{w}, which represent the posterior probability distribution $p(\boldsymbol{X}_k|z_0, \dots, z_k, u_0, \dots, u_k)$. The noise components $\tilde{n}_a$, $\tilde{n}_\Delta$ and the elimination of the least probable particles make the particle filtering method robust so that it can approximate any distribution and can adapt to environmental changes. Moreover, realizing the distributions by particles allows us to use random variables in non-linear computations, like $\beta_k = e^{-x_k(3)/\tau} = e^{-(x_{k-1}(3)+\tilde{n}_\Delta)/\tau}$ in (6).

We now explain the prediction procedure using the thermal model (2) and the particle filter. To start with, we keep track of the temperature using the particle filter. Considering that the system is at the k^{th} time point, we have better knowledge about the current temperature T_k, the ambient temperature $T_{a,k}$ and the probability distribution of the time between two consecutive temperature measurements due to the particle filter. This means that, if the best approximation for the posterior probability distribution $p(\boldsymbol{X}_k|z_0, z_1, \dots, z_k)$ is represented by the set of particles $S_k^{\boldsymbol{X}}$ and the set of importance weights S_k^{w}, we can compute a new set of particles for a future time point $k' > k$, i.e., $S_{k'}^{\boldsymbol{X}} = \{\boldsymbol{x}_{k'}^{(j)}\}_{j\in\{1,\dots,N\}}$, using (6) iteratively. Then, we have the probability distribution for the future time point k', i.e. $p(\boldsymbol{X}_{k'}|z_0, \dots, z_k, u_0, \dots, u_{k'})$ and estimate the temperature at this time point as:

$$\hat{T}_{k'} = \frac{\sum_{j=1}^{N} w_{k'}^{(j)} (\boldsymbol{x}_{k'}^{(j)}(1) + \boldsymbol{x}_{k'}^{(j)}(2))}{\sum_{j=1}^{N} w_{k'}^{(j)}}.$$

To validate this approach, we employ a similar procedure as in Section 3. Figure 4 shows the prediction error, i.e., $z_k - \hat{T}_k$, for different future time points, similar to Figure 1 in Section 2 and Figure 2 in Section 3. As can be observed, the confidence intervals are tighter than for the previous methods. This is also shown in Figure 5.

5 Discussion

In this paper, we compare the three methods: (i) the original model, (ii) the model improved with a Kalman filter and (iii) the model improved with a particle filter. Figure 5 shows the standard deviation of the prediction errors as a function of the future look-ahead time. As expected, estimating the far future is less accurate then the near future, hence, the standard deviation increases. However, significant improvements are obtained with the application of the Kalman and particle filtering methods.

The key advantages of using a Kalman filter are that it adapts to environmental changes and can compensate for the error resulting from the process noise. In our case, the ambient temperature can change over time. Moreover, the time between two consecutive temperature measurements does not exactly equal to the sampling period due to the process noise, which means it has jitter. Although the change in the environmental temperature is slow and the process noise is low, the effect is significant in the long run. For this reason, the Kalman filter can make better estimations. It combines the past data with the dynamic model equation to compute the posterior probability of the current temperature. This results in an accurate representation of the current temperature with a reduced confidence interval. Figure 5 shows that the Kalman filtering extension decreases the standard deviation of the prediction error. The disadvantage of a Kalman filter is that it assumes each variable to be normally distributed, which might not be always the case in practice. Although this approximation still gives accurate estimations in our case since the distributions have single peak points, the asymmetric characteristics of the distributions leaves room for improvement. These asymmetric characteristics can be observed in Figures 1 and 2. The reason of this asymmetry is the process noise. Although the temperature measurements are taken periodically, a time point where the system takes a measurement may shift in a positive or a negative direction due to the process noise. This time shift also affects subsequent time points. The Kalman filter is able to adapt to

these small time shifts. However, the probability of the positive and negative time shifts are considered equal by the Kalman filter.

Particle filtering methods are able to approximate any arbitrary distribution using a sequential Monte Carlo algorithm. For this reason, we use a Particle filter to overcome the disadvantage of a Kalman filter. In our case, the particle filter finds an approximated probability distribution for the process noise and the ambient temperature. Moreover, it adapts to the changes in the distributions. Another advantage of the Particle filtering method is that it allows non-linear operations since the distributions are represented by sets of particles instead of functions, like in the Kalman filtering method. Therefore, we can compute the future temperature more accurately since the relation between the process noise $\tilde{n}_\Delta$ and β is non-linear, i.e., $\beta \propto e^{-\tilde{n}_\Delta/\tau}$. The benefit of the particle filtering method is shown in Figure 5. In the figure, the standard deviation of the particle filter converges to the standard deviation of the Kalman filter in the far future. The reason is that the distribution for the process noise converges to a Gaussian distribution in the far future.

Moreover, although we have focused on the specific dynamic model that we introduced in [7], the new methods of Section 3 and Section 4 can be applied to other dynamic models as well since these methods require a dynamic system that derived from the well-known heat transfer equation (1), which is mostly the case for the models in the literature.

6 Conclusion

We propose two generic methods that can be combine with any dynamic thermal model to improve the prediction accuracy of processor temperatures, based on Kalman filtering and particle filtering. We validated these two improved methods experimentally using an ODROID-XU4 platform, and compared the performance with that of our previously developed method, which only uses a simple thermal model. We show that Kalman filtering will already improve the accuracy since it can adapt to environmental changes. However, particle filtering even further improves the prediction accuracy since it does not require normally distributed random variables and it can adapt to any arbitrary distribution.

Literature

[1] A. Iranfar, M. Kamal, A. Afzali-Kusha, M. Pedram and D. Atienza, "TheSPoT: Thermal Stress-Aware Power and Temperature Management for Multiprocessor Systems-on-Chip," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1532-1545, Aug. 2018, doi: 10.1109/TCAD.2017.2768417.

[2] K. Sekar, "Power and Thermal Challenges in Mobile Devices," in *Proceedings ACM of the 19th Annual International Conference on Mobile Computing & Networking (MobiCom '13)*, 2013, pp. 363–368, doi: 10.1145/2500423.2505320.

[3] A. K. Singh, S. Dey, K. R. Basireddy, K. McDonald-Maier, G. V. Merrett and B. M. Al-Hashimi, "Dynamic Energy and Thermal Management of Multi-Core Mobile Platforms: A Survey," in *IEEE Design & Test*, 2020, doi: 10.1109/MDAT.2020.2982629.

[4] A. K. Coskun, T. S. Rosing and K. C. Gross, "Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1503-1516, Oct. 2009, doi: 10.1109/TCAD.2009.2026357.

[5] G. Bhat, G. Singla, A. K. Unver and U. Y. Ogras, "Algorithmic Optimization of Thermal and Power Management for Heterogeneous Mobile Platforms," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 544-557, March 2018, doi: 10.1109/TVLSI.2017.2770163.

[6] A. Prakash, H. Amrouch, M. Shafique, T. Mitra and J. Henkel, "Improving mobile gaming performance through cooperative CPU-GPU thermal management," *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1-6, doi: 10.1145/2897937.2898031.

[7] B. Ozceylan, B. R. Haverkort, M. de Graaf and M. E. T. Gerards, "A Generic Processor Temperature Estimation Method," *2019 25th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*, 2019, pp. 1-6, doi: 10.1109/THERMINIC.2019.8923636.

[8] E. W. Wächter, C. de Bellefroid, K. R. Basireddy, A. K. Singh, B. M. Al-Hashimi and G. Merrett, "Predictive Thermal Management for Energy-Efficient Execution of Concurrent Applications on Heterogeneous Multicores," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 6, pp. 1404-1415, June 2019, doi: 10.1109/TVLSI.2019.2896776.

[9] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333-1364, Oct. 2003, doi: 10.1109/TAES.2003.1261132.

[10] G. Welch and G. Bishop, "An introduction to the Kalman filter," Chapel Hill, NC, USA, Tech. Rep., 1995.

[11] A. Doucet and A. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*, D. Crisan and B. Rozovsky, Eds., Oxford, U.K.: Oxford University Press, 2011, ch. 24, pp. 656-704.