Timing Analysis on a Processor with Temperature-Controlled Speed Scaling

Pratyush Kumar and Lothar Thiele
Computer Engineering and Networks Laboratory,
ETH Zurich, Switzerland
E-mail: {kumarpr, thiele}@tik.ee.ethz.ch

***Abstract*—Several recent works consider the problem of temperature-constrained scheduling of jobs. In such attempts, speed of the processor and the execution of jobs is software-controlled such that temperature and performance constraints are met. An alternative approach is to use measurements from temperature sensors to actuate the speed of the processor as a feedback control loop. Though such a solution explicitly and independently meets the thermal constraints, the analysis of the real-time properties of tasks served by such a processor is not straightforward. In this paper, we study this problem for a variable stream of jobs characterized by an input arrival rate. We show that an intuitive notion of monotonicity extends to such a processor. Using this property, we present an analytical technique to determine the worst-case delay suffered by jobs. The presented technique efficiently and tightly determines the delay as a function of the initial temperature. The simplicity of this analysis motivates further analysis and mainstream use of such systems.**

I. Introduction

As we continue to require more functional complexity out of fabricated silicon, the power densities of our systems, correspondingly, continue to rise. Hardware cooling solutions such as heat sinks and air cooling techniques, though constantly improving, are unable to keep pace. Consequently, the peak temperatures of our systems are higher than ever.

Different communities have grappled with this impending "temperature wall" differently. The VLSI design community has continued its focus on low-power design with a search for material and technology innovations. Chip packagers are studying novel cooling solutions such as liquid cooling. Software and computer engineers have focussed on Dynamic Thermal Management (DTM) techniques such as Dynamic Voltage Scaling (DVS), system throttling, task migration, to lower system temperature. The real-time community has been concerned about tasks meeting deadlines on such hot chips with DTM techniques enabled. The underlying question is: "How do we design and analyze systems with both temperature and performance constraints?"

Several research works have attempted to answer, in part, the above problem. The common approach is to concretely define a policy of the DTM technique. For instance, for a DVS-enabled processor, each software task may be assigned a fixed frequency of operation. The policy is so chosen that temperature and/or performance metrics are optimized within temperature and/or performance constraints. Examples of such research works include [1], [2], [3] and [4].

The fundamental limitation with any such approach is that the DTM policy is verified or evaluated for a *precise* model of the system. Given the nature of the problem, this model spans widely comprising the physical properties of heat diffusion, architectural properties of power generation and the software properties of jobs' arrival and execution times. It is not uncommon to expect these properties of a system to be uncertain at design-time: VLSI engineers have often highlighted process variations, whereas real-time system theorists routinely expect variable execution times and job arrival patterns. Such variability can prove to be detrimental: If due to a software bug the execution time of a task is larger than expected, the temperature constraint can be violated and this may lead to a system failure.

What are the potential ways to cope with such imprecise models? The brute-force approach would be to consider and design for all possible variations of the model. This makes the problem computationally expensive and perhaps epistemically uninteresting. The standard engineering approach would be to consider sensitivity analysis to identify confidence levels across the parameter space and then derive design guidelines. While we think that such an approach would be instructive, we did not find any existing work in this direction. Another approach would be the isolation of the sources of variability, whereby, by design, certain system parameters are not affected by uncertainties in other parameters of the system. One such example from the real-time operating systems world is the use of servers to execute different tasks while isolating timing faults. As a final approach, one can theoretically argue, if possible, that it is sufficient to study only a sub-space of the possible model variations. The remaining model variations should be shown to perform "better" with respect to the considered system properties (for example lower temperature or higher throughput).

In this work, we show that a combination of the last two of the above mentioned approaches can be applied to the problem of thermal-aware scheduling. In particular, we show that with the use of reactive speed scaling as proposed in [5] we can isolate thermal failures from the uncertainties in the software model. Then, to consider uncertainties in the software model, we theoretically show that analyzing only

one worst-case trace of jobs, is sufficient to compute a tight bound on the provided performance.

Reactive speed scaling was proposed in [5] as a means to control the speed of a DVS-enabled processor based on the current temperature of the processor. In other words, the temperature sensor and the speed actuator form a closed feedback control loop: the higher the temperature the lower the set speed. The control law is such that the processor's temperature would not exceed a threshold, if the assumed power and thermal models are correct. We interpret this as isolating the thermal constraint from variability in the software model. Independent of the imprecisions such as larger than expected execution times, a processor with such a control-loop would meet the temperature constraint.

Though the above approach isolates thermal constraint from software uncertainties, it greatly complicates standard real-time analysis of tasks. By letting the speed of the processor be set based on its temperature, the response times of jobs are more difficult to evaluate. This is especially so because temperature is a state variable that depends on the earlier behavior of the processor. Furthermore, if the stream of jobs shows variability in arrival patterns and execution times (for instance jitter of a periodic task) the analysis is further complicated. In this paper, we consider the problem of computing the worst-case delay suffered by jobs of a stream that conform to a given arrival rate [6]. Towards this end, we define a desirable monotonicity principle, which is proved to hold for general thermal and power models. Using the monotonicity principle, we show how analyzing only one worst-case trace of the system suffices to characterize the maximum possible delay that can be suffered by any job.

Our results show that the worst-case trace depends on the initial temperature of the system. As a key contribution, we present a unifying analysis technique that holds for all initial temperatures of the processor. Importantly, the analysis that we present is both tight and efficient to compute. This simple analysis technique motivates the analysis and use of such systems for mainstream real-time applications.

Organization of the paper The rest of the paper is organized as follows. We define the model and analysis objectives in Section II. In Section III, we define and prove the monotonicity property of the considered system. The main results of the paper are presented in Section IV, where we consider each case of initial temperature separately. We demonstrate the theoretical results by numerical experiments in Section V. We present concluding remarks in Section VI, and compare our work to existing results in Section VII.

II. System Model and Problem Definition

In this section, we detail the models we consider, namely models that describe (a) the speed scaling rules of the processor, (b) the thermal properties of the system, and (c) the timing properties of the jobs executing on the processor. Our focus has been to define the models as abstractly as possible, so as to generalize the results to a large class of systems.

A. Speed Scaling Rule

We consider a processor, denoted as $\mathbf{P}$, that is characterized at all times by a temperature T and a speed s. The temperature of the processor is sensed with a temperature sensor, while the speed of the processor is actuated according to a speed scaling rule given as

$$\begin{aligned} s(t) &= f(T(t)), && \text{if there are pending jobs,} \\ &= 0, && \text{else,} \end{aligned} \quad (1)$$

where f is a non-negative piecewise-constant non-increasing function. In words, when there is no pending work to execute, $\mathbf{P}$ is idle (zero speed), and when there is work to execute the speed set is higher if the temperature is lower, as given by the function f.

The global minimum of the function f is denoted as $s^{\min}$ and the steady state temperature of $\mathbf{P}$ when executing at this speed is denoted as $T^{\max}$. It is required that the domain of the function f does not exceed beyond $T^{\max}$ and $f(T^{\max}) = s^{\min}$. Then it follows that, independent of the arrival times of jobs and their execution times, the speed scaling rule ensures that the temperature of $\mathbf{P}$ never exceeds $T^{\max}$.

In the above model we have made two assumptions. Firstly, like in [5], [7], we assume zero delay between sensing and actuation. Considering non-zero control-loop delay further complicates the analysis and is a direction for future work. As the second assumption, like several other researchers in [1], [2], [3], [4] and [8] we consider the processor to be thermally homogenous. Considering multiple thermal "nodes" is an important direction in extending the results to multi-core systems. Note that this assumption does not restrict the cooling infrastructure (see Appendix).

B. Power and thermal model

Modelling the thermal properties of a system is an active line of research deriving from on the one hand the physical phenomenon of heat dissipation and on the other hand the architectural property of power consumption of the processor. This wide scope renders such modelling practically challenging. Nevertheless, for obtaining useful theoretical insights and design guidelines, abstractions of these models are employed.

The power consumed by a processor, based on the CMOS technology, is a convex function of the operating frequency or speed. Denoting $P(t)$ as the power consumed by the processor at time t, we have

$$P(t) = \phi(s(t)), \quad (2)$$

where ϕ is a convex increasing function of the speed $s(t)$, such that $\phi(0) = 0$. Any static power consumption (such as leakage power), both constant and temperature-dependent, can be absorbed into the thermal parameters as discussed in detail in the Appendix.

The role of the thermal model is to transform the power consumed by the processor $P(t)$ to the temperature of the processor $T(t)$. We assume that this thermal model is linear and time-invariant (abbreviated as LTI). This assumption has been made in earlier works [1], [2], [3], [4] and [8], and is valid for the conductive heat diffusion process. Consequently, there exists a characteristic impulse response $h(t)$ such that

$$T(t) = (P * h)(t), \tag{3}$$

where $*$ is the convolution operator defined as

$$(f * g)(t) = \int_0^t f(u)g(t-u)du \tag{4}$$

Furthermore, we assume that $h(t)$ is monotonically non-increasing, i.e.,

$$h(t_1) \geq h(t_2), \quad \forall\ t_1 \leq t_2. \tag{5}$$

The physical interpretation of this assumption is that the effect of consuming power on the temperature of the system, decreases over time (as the heat dissipates to the cooler surroundings). While this seems intuitively true, we formally show in the Appendix, that it is holds for the conductive thermal model. Furthermore, we show that the assumptions also hold if the leakage power of the processor depends on the temperature, thereby being relevant for the highly leaky modern systems.

We assume that the ambient temperature is constant. The linear nature of the system enables us to represent the temperature of the processor relative to this constant ambient temperature. To simplify expressions, throughout the paper, we assume that the temperature of the processor when in steady-state with the ambient while in the idle mode is 0.

C. Task model

A task is a stream of jobs that need to be executed on the processor. In this work, we assume that jobs of the same task are executed in First-Come-First-Serve (FCFS) manner. From the perspective of real-time analysis, the task is characterized by the execution demand of the jobs. We define execution demand of a job as the time required to complete the job on $\mathbf{P}$ running at unit speed. We make the standard assumption that execution demand linearly scales with the speed of $\mathbf{P}$.

As discussed earlier, variability in the arrival and execution times of jobs are expected and need to be considered. For instance, in a periodic job with jitter, the jitter may be exhibited at unknown times. To formally capture such variability, the notion of input arrival rate has been used in network calculus [6], and subsequently real-time calculus [9]. For one trace of job arrivals, let $R(t)$ denote the execution demand of the jobs arriving in the interval $[0, t]$, i.e., all jobs arriving in $[0, t]$ require a total of $R(t)$ amount of time to execute on $\mathbf{P}$ running at unit speed. Let $\mathbf{R}$ denote the set of possible traces $R(t)$ that can be exhibited by the system. Then, the input arrival rate α complies to the set $\mathbf{R}$ iff

$$\alpha(\Delta) = \sup_{t \geq 0}(R(t+\Delta) - R(t)), \quad \forall\ R \in \mathbf{R}. \tag{6}$$

In words, $\alpha(\Delta)$ bounds the execution demand of all jobs arriving in any time interval of length Δ.

D. Problem description

Given is a processor $\mathbf{P}$ with speed scaling rule and power and thermal models as described. This processor is executing a stream of jobs characterized by an input arrival rate as described. We want to compute the worst-case delay suffered by any job of the stream. Such a worst-case delay can then be used to check if real-time properties such as deadlines are met. We restrict our analysis to jobs arriving in a specified time-window of $[0, \tau]$, where τ is some arbitrary given number. The delay suffered by the jobs of the stream is likely to depend on the initial temperature of the system, denoted as $T(0)$. We consider three cases:

1) Zero Initial Temperature $T(0) = 0$. (Recall that the temperature is specified relative to the idle steady-state processor temperature.)
2) Maximum Initial Temperature: $T(0) = T^{\max}$.
3) Intermediate Initial Temperature: $T(0) \in (0, T^{\max})$.

III. The Monotonicity Principle

In this section, we establish an important result that will enable us to analyze the timing guarantees provided by $\mathbf{P}$. First, we define a *unit cycle*, denoted as u, as the smallest measure of the execution time of jobs. Conveniently, u can be defined to be 1-cycle long (one clock tick). We assume that the speed of $\mathbf{P}$ is constant during the execution of a unit cycle.

Lemma 1. *Consider the execution of a unit cycle on* $\mathbf{P}$. *Increasing the starting temperature of* $\mathbf{P}$ *cannot lead to an earlier finish time of the unit cycle.*

Proof: The speed-scaling rule of (1) is such that f is monotonically non-increasing with the temperature of the system. Thus, increasing the temperature cannot increase the speed of execution of the unit cycle and thus cannot decrease the finish time. ■

Lemma 2. *Consider the execution of a unit cycle on* $\mathbf{P}$. *Delaying the execution of this unit cycle cannot lead to an earlier finish time.*

Proof: We prove this by contradiction. Consider two schedules S_1 and S_2. Both schedules start at time 0 with the same arbitrary initial temperature of $\mathbf{P}$. S_1 executes the unit cycle from time 0, whereas S_2 is idle until time $p > 0$ and then executes the unit cycle. Let the finish time of the unit cycle be earlier in S_2 than in S_1. Then, there exists a time $q > p$ such that both S_1 and S_2 have completed exactly the same amount of work until q. Let T_1 and T_2 denote the temperature trace of $\mathbf{P}$ for the schedules S_1 and S_2, respectively. Let P_1 and P_2 denote the power traces of $\mathbf{P}$ for schedules S_1 and S_2, respectively. The two schedules perform the same amount of work in $[0, q]$ but schedule S_1 runs at a lower speed. Then, from the convexity of function ϕ defined in (2), we know that

$$\int_0^{d_2} P_1(t)dt \leq \int_0^{d_2} P_2(t)dt. \tag{7}$$

Furthermore, P_1 is constant, whereas P_2 is non-decreasing. Combining this with the monotonically decreasing impulse response h we have

$$(P_1 * h)(q) \leq (P_2 * h)(q). \tag{8}$$

Since the initial temperature of the $\mathbf{P}$ for the two schedules is the same, we have $T_1(q) \leq T_2(q)$. However, at time q, the speed of $\mathbf{P}$ in S_2 is greater than in schedule S_1. Given the monotonic nature of f of (1) we have $T_2(q) < T_1(q)$. This is a contradiction and hence S_2 does not finish executing the unit task before S_1 does. ■

Lemma 3. *Consider the execution of a single unit cycle on* $\mathbf{P}$. *Delaying the execution of this unit cycle cannot lead to a lower temperature of* $\mathbf{P}$ *measured at the finish time of the unit cycle in the delayed execution.*

Proof: Let S_1 and S_2 be two schedules defined in the proof of Lemma 2. Let d_1, d_2 be the finish times of the unit task in schedules S_1 and S_2, respectively. Then, we need to show that $T_1(d_2) \leq T_2(d_2)$. In S_1, $\mathbf{P}$ runs at a constant speed from time 0 to d_1 and then is idle from d_1 to d_2. In S_2, $\mathbf{P}$ is idle from 0 to q and then runs at a higher constant speed from time q to d_2. The two schedules complete the same amount of work in the interval $[0, d_2]$ and schedule S_1 runs at a lower speed. Then from the convexity of function ϕ of defined in (2), we know that

$$\int_0^{q} P_1(t)dt \leq \int_0^{q} P_2(t)dt. \tag{9}$$

Further, P_1 is monotonically non-increasing, whereas P_2 is monotonically non-decreasing. As before, using the monotonic natures of the impulse response h and the speed-scaling rule f, we have $T_1(d_2) \leq T_2(d_2)$. ■

Note that the convexity of ϕ, the monotonicity of functions f and h, are the only properties that suffice to prove the above results. In particular, one does not need the exact solution of the thermal differential equation to prove these results. We are now ready to define and derive the monotonicity principle.

Theorem 4 (Monotonicity Principle)**.** *Consider a* given *stream of jobs served by* $\mathbf{P}$. *Delaying the arrival time of any job of the stream cannot lead to an earlier finish time of any* subsequent *job of the stream.*

Proof: We prove this by induction on the total number of unit cycles of all jobs of the considered stream.

Induction Hypothesis: Delaying the execution of any job or increasing the starting temperature of $\mathbf{P}$ will not lead to an earlier finish time of any job.

Basis: For one unit cycle we showed in Lemma 1 and 2, that the induction hypothesis is true.

Inductive Step: If the hypothesis holds for all streams with execution demand c unit cycles, then it holds for all streams with execution demand $(c + 1)$ unit cycles.

Let R be a stream of jobs with a total execution demand of $(c + 1)$ unit cycles. Let S_1 and S_2 be two schedules of executing R on $\mathbf{P}$ such that in S_1 all jobs are executed without any delay, whereas in S_2 some jobs are executed with delay. Consider the first unit cycle of R. If the execution of this unit cycle is not delayed in S_2, then both schedules S_1 and S_2 behave exactly similarly until the execution of the first unit cycle and thus the problem is reduced to a stream of tasks of c unit cycles which according to the inductive step satisfies the hypothesis. Let the execution of the first unit cycle be delayed in S_2. Let d_1 and d_2 denote the finish time of this unit cycle in the two schedules S_1 and S_2. From Lemma 3, we know that $d_2 \geq d_1$. Two cases arise, depending on the arrival time of the second unit cycle of R, denoted as η.

Case (a): $\eta \geq d_2$. Applying Lemma 4 to S_1 and S_2 for the fixed-time of observation of η, we have, $T_1(\eta) \leq T_2(\eta)$. Consider the trace R starting from η onwards. The execution demand of this trace is c cycles. Then, according to the induction hypothesis increasing the starting temperature cannot lead to the earlier finish time of any job. As $T_1(\eta) \leq T_2(\eta)$, no job of S_2 can finish before the same job of S_1.

Case (b): $\eta < d_2$. Modify S_1 to form a new schedule S_1' such that the second unit cycle of R is delayed to start executing from d_2. From the induction hypothesis, from the second unit cycle onwards, no job of schedule S_1 will finish any later than the same job in S_1'. From the induction hypothesis, starting from time d_2, no job of S_2 can finish before than

the same job from S_1'. Hence, no job of S_2 finishes before the same job of S_1. ■

To put the above result in perspective, consider the case of a constant-speed processor serving a stream of jobs in FCFS manner. It can be seen that delaying the execution of any job will not lead to a earlier finish time of any subsequent job. This can be interpreted as: Delaying a job does not decrease the interference the job has on subsequent jobs. The derived results says that, even for processors with temperature-controlled speed scaling with the model assumptions, this desirable monotonic relationship between arrival time and interference holds.

A second point relates to the optimal use of such a processor. The monotonicity principle implies that optimal execution of a given stream of jobs on $\mathbf{P}$, with respect to response time, is realized by the greedy processing of pending jobs. In other words, no additional control by delaying the execution of jobs (called shaping, as used in [10]) is required to reduce the delay.

IV. Delay Analysis

In this section, we present the main results of this paper, that of the delay analysis of $\mathbf{P}$. We consider three cases of initial temperatures separately. For each of these cases, we identify the trace of the arrival of jobs, conforming to the input arrival rate α, where one of the jobs suffers the worst-case delay. We call such a trace the worst-case trace. We can simulate the worst-case trace and compute the value of the worst-case delay.

A. Zero Initial Temperature

We now consider the problem for the case of a processor with no prior workload, i.e., $T(0) = 0$. We show in the subsequent result that independent of the thermal, power and speed-scaling parameters of $\mathbf{P}$, there exists a uniquely characterized worst-case trace and a uniquely characterized job of that trace which experiences the worst-case delay. Recall that input traces are defined the domain $[0, \tau]$, for some given τ.

Theorem 5. *The tight worst-case delay when a stream of jobs with input arrival rate α is executed on $\mathbf{P}$ with $T(0) = 0$, is suffered by the last job of the trace R^*_{zero} defined as*

$$R^*_{\text{zero}}(t) = \alpha(\tau) - \alpha(\tau - t), \quad t \in [0, \tau]. \tag{10}$$

Proof: Let the last job of the trace R^*_{zero} be denoted J^*. We prove the above by contradiction. Let there exist some other trace of arrival of jobs, R, such that the job arriving at time $t'(\leq \tau)$, denoted as J', has a response time larger than J^*. Let t'' be defined as follows

$$R(t') = \alpha(t' - t''). \tag{11}$$

Since R conforms to the input rate α, we have $t'' \in [0, t']$.

Applying the monotonicity principle on R, we can delay the arrival of jobs arriving in $[0, t']$, without decreasing the response time of J'. This can be done until

$$\begin{aligned} R(t) &= \alpha(t' - t'') - \alpha(t' - t), && t \geq t'', \\ &= 0, && t < t''. \end{aligned} \tag{12}$$

It can be seen that delaying any job any further results in a trace that does not conform to the input rate α. The modified trace R defined as above in the interval $[t'', t']$ is a suffix of the trace R^* defined in (10). More formally,

$$R(t) = R^*_{\text{zero}}(t + (\tau - t')), \quad t'' \leq t \leq t'. \tag{13}$$

Let us compare the common parts of the traces, i.e., $R(t'' : t')$ and $R^*_{\text{zero}}((\tau - (t' - t'')) : \tau)$. While serving trace R, the buffer of pending jobs is empty just before t'', whereas while serving R^* there could be pending jobs at time $\tau - (t' - t'')$. Then, from Lemma 2, for J' to suffer a larger delay than J^*, the temperature of $\mathbf{P}$ while executing R at time t'' should be strictly higher than the temperature of $\mathbf{P}$ while executing R^*_{zero} at time $\tau - (t' - t'')$. However, this is a contradiction, as the temperature of $\mathbf{P}$ while executing R remains the minimum possible value of 0 in $[0, t'']$.

R^*_{zero} conforms to the input arrival rate α and thus the result is tight. ■

As a consequence of the monotonicity principle, we were able to identify the worst-case trace of job arrivals, R^*_{zero}, with respect to the delay suffered by jobs served by $\mathbf{P}$ starting from zero initial temperature. Importantly, under the model assumptions, R^*_{zero} does not depend on the values of parameters of the speed-scaling rule, the power consumption or the thermal impulse response. Note that in R^*_{zero} burst of jobs, if any, appears at the end of the time horizon. As mentioned before, to compute the value of the worst-case delay we simply simulate the execution of R^*_{zero} on $\mathbf{P}$ and observe the delay of the last job.

B. Maximum Initial Temperature

We now consider the case $T(0) = T^{\max}$. We arrive at the solution differently in comparison to the previous section. Firstly, we present the following definition

$$\texttt{Del}(f, g, \tau) = \sup_{t \in [0,\tau]} \{\inf\{u \in [t, \tau] \ : \ f(t) \leq g(t + u)\}\}.$$

The above function defines the maximum horizontal distance between functions f and g, in the domain $[0, \tau]$.

Theorem 6. *The tight worst-case delay when a stream of jobs with input arrival rate α is executed on $\mathbf{P}$ with $T(0) = T^{\max}$ is given by*

$$d = \texttt{Del}(\alpha(\Delta),\ s^{\min}\Delta,\ \tau) \tag{14}$$

Proof: We will first show that d defined as above is an upper-bound on the worst-case delay under the given conditions. We will then show the tightness by producing a valid trace of jobs, $R^*_{\max}$, such that at least one job suffers a delay exactly equal to d.

Recall that $s^{\min}$ is the lowest speed of $\mathbf{P}$ when there are pending jobs to execute, and $f(T^{\max}) = s^{\min}$. Then, it follows that in any busy interval of length Δ, the minimum service provided by $\mathbf{P}$ is $(s^{min} \cdot \Delta)$. From network calculus [6] this implies that the service curve of $\mathbf{P}$ is lower-bounded by $\beta(\Delta) = s^{min} \cdot \Delta$. From the definition of the service curve, the delay suffered by any job of the stream is indeed upper-bounded by d defined as in (14).

We now prove the tightness. Let Δ' and Δ'' be the smallest numbers satisfying

$$\alpha(\Delta') = s^{min}(\Delta' + d), \tag{15}$$
$$\alpha(\Delta'') = s^{min}\Delta''. \tag{16}$$

Then, it follows from the sub-additive property of α [6] that $\Delta' \leq \Delta''$. Consider the trace of jobs $R^*_{\max}$ defined as

$$R^*_{\max}(t) = \alpha(t), \quad t \in [0, \min(\tau, \Delta'')] \tag{17}$$

For this trace, the definitions of Δ' and Δ'' implies that if the jobs of $R^*_{\max}$ are executed on a constant-speed processor executing at speed $s^{\min}$, we have a busy interval from $[0, \Delta'']$ and the job arriving at Δ' (which is before the end of the first busy interval) suffers the largest delay which equals d.

Now consider the execution of $R^*_{\max}$ on $\mathbf{P}$. Since, the initial temperature is $T^{\max}$, $\mathbf{P}$ will behave exactly like a constant-speed processor running at $s^{\min}$ until the end of the first busy interval. From the result that $\Delta' \leq \Delta''$, we know that all jobs of $R^*_{\max}$ belong to the first busy interval, and thus $\mathbf{P}$ behaves exactly like a constant-speed processor running at s^{min}. Thus, some job of $R^*_{\max}$ suffers from the delay d. ∎

Note that the for the two cases considered so far, the worst-case traces are markedly different. In the case of zero initial temperature, in the worst-case trace, R^*_{zero}, we had the burst of jobs, if any, towards the end of the time-horizon. On the other hand, in $R^*_{\max}$, we have the bursts of jobs, if any, towards the start. For an example arrival rate this difference can be noticed in Figure 1. Another difference is with respect to computing the worst-case delay. For zero initial temperature case, we had to rely on simulation to compute the delay suffered by the last job. However, for the case of maximum initial temperature, computation of worst-case delay is decidedly simpler: compare the input arrival rate of the jobs with the slowest speed of the processor.

C. Intermediate Initial Temperature

We presented the results of delay analysis for the case of zero and maximum initial temperatures. In both cases, we applied different principles that enabled identification of the worst-case trace. So the natural question is what happens when the initial temperature is between the two extremes. We answer this question and develop a unifying analysis technique in this section.

Towards the above end, we define a model of an artificial processor $\mathbf{P}$'. We will show, in the subsequent result, that analysis of worst-case traces on $\mathbf{P}$' helps provide a unifying picture. $\mathbf{P}$' has a speed scaling rule, power model and thermal model exactly like that of $\mathbf{P}$. But $\mathbf{P}$' is differentiated from $\mathbf{P}$ by an additional rule which states that the temperature of $\mathbf{P}$' is never allowed to fall below a given *temperature threshold*. Note that we do not change the thermal model. The temperature of the system is computed according to the standard thermal and power models. If the temperature is below the threshold, it is enforced to be equal to the threshold. In other words, the temperature of $\mathbf{P}$' is *clipped* from below to the threshold. A $\mathbf{P}$' with zero temperature threshold is equivalent to $\mathbf{P}$. A $\mathbf{P}$' with maximum temperature threshold is equivalent to a constant-speed processor running at speed $s^{\min}$.

Theorem 7. *The tight worst-case delay when a stream of jobs with input arrival rate α is executed on $\mathbf{P}$ with $T(0) = T^{\text{inter}}$, for some given $T^{\text{inter}} \in (0, T^{\max})$, is equal to the delay suffered by the last job of the trace R^*_{zero} defined in (10) when served by $\mathbf{P}$' with temperature threshold equal to T^{inter}.*

Proof: Let the last job of R^*_{zero} be denoted as J^*.

We first prove that the upper-bound holds by contradiction. Let there be some other stream R executing on $\mathbf{P}$ such that the job J' arriving at t', suffers a delay larger than that suffered by J^* when served by $\mathbf{P}$'. As before, we can use the monotonicity principle to delay the arrival of jobs without decreasing the delay suffered by J'. With such delays we can modify R until

$$\begin{aligned} R(t) &= R(t') - \alpha(t' - t), && t \geq t'', \\ &= 0, && t < t'', \end{aligned} \tag{18}$$

where t'' is as defined in (11).

Now we can compare the two equal parts of the traces R and R^*_{zero}, namely $R(t'' : t')$ and $R^*_{\text{zero}}(\tau - (t' - t'') : \tau)$. There are no pending jobs at time t'', when R is served by $\mathbf{P}$.

Whereas there may be pending jobs when R^*_{zero} is served by $\mathbf{P}$' at time $\tau-(t'-t'')$. By repeated use of Theorem 2 and the definition of $\mathbf{P}$', for the hypothesis to hold, the temperature of $\mathbf{P}$ at time t'' when serving R, must be strictly larger than that of $\mathbf{P}$' at time $\tau-(t'-t'')$ where serving R^*_{zero}. But $\mathbf{P}$ is idle from in $[0, t'']$ and thus its temperature cannot be larger than T^{inter} which is the lower-bound on the temperature of $\mathbf{P}$'. Thus, we have a contradiction.

Since the delay is computed on an artificial model $\mathbf{P}$', we still have to show that the bound is tight when jobs execute on $\mathbf{P}$. Let us consider the trace R^*_{zero} being executed by $\mathbf{P}$'. Let ρ be the latest time when the temperature of $\mathbf{P}$' is *forced* to be equal to T^{inter}. If the temperature is never forced to be equal to T^{inter}, then $\mathbf{P}$' behaves equivalent to $\mathbf{P}$, and thus the bound is tight, as R^*_{zero} conforms to α. In this case we set $\rho = 0$. Otherwise, $\rho > 0$. Since, the temperature of $\mathbf{P}$' at ρ^- cannot be lower than T^{inter}, it implies that there are no pending jobs to execute at time ρ. Thus, starting from an empty buffer at ρ, $\mathbf{P}$' executes all jobs that arrive subsequently, without forcing the temperature to T^{inter}. This implies that if we have a trace R^*_{inter} defined as

$$R^*_{\text{inter}}(t) = R^*_{\text{zero}}(t+\rho) - R^*_{\text{zero}}(\rho), \quad t \in [0, \tau-\rho]. \tag{19}$$

$$= \alpha(\tau-\rho) - \alpha(\tau-\rho-t), \quad t \in [0, \tau-\rho]. \tag{20}$$

then the behavior of $\mathbf{P}$ and $\mathbf{P}$' when executing R^*_{inter} is identical. Thus, the delay suffered by the last job of R^*_{inter} on $\mathbf{P}$ is the same as the delay suffered by the last job of R^* on $\mathbf{P}$'. This proves the tightness. ■

D. Discussion

Can we apply the methodology for the intermediate temperature case to the other two cases? It is clear to see that extending it for the case when $T(0) = 0$ is straightforward. In this case $\mathbf{P}$' has a threshold temperature of 0 and is indistinguishable from $\mathbf{P}$. Thus, ρ as used in the proof of Theorem 7 would be 0 and by (19) R^*_{inter} and R^*_{zero} are equal.

Can a similar statement be made for the case $T(0) = T^{\max}$? For this case, $\mathbf{P}$' has a threshold temperature of $T^{\max}$, which implies that $\mathbf{P}$' always executes jobs at the constant speed of $s^{\min}$. Then it can be shown formally that the value ρ would be exactly equal to $\max(0, \tau - \Delta'')$, where Δ'' is as defined in (16). We then obtain R^*_{inter} by substituting this ρ in (20). In either case, it can be shown that the delay suffered by the last job of such an R^*_{inter} when served at a constant speed $s^{\min}$ is exactly equal to d as defined in (14).

As has been illustrated, for different initial temperatures, the worst-case traces could be different, depending on the value of ρ. However, by defining the artificial model $\mathbf{P}$' we need to only consider a single worst-case trace as defined by R^*_{zero}. The delay of the last job of this trace when served by $\mathbf{P}$' is a tight upper-bound for the delay of any job when executing on $\mathbf{P}$. We have thus effectively reached our goal of handling variations in the software model by only analyzing a smaller sub-space, in this case one worst-case trace.

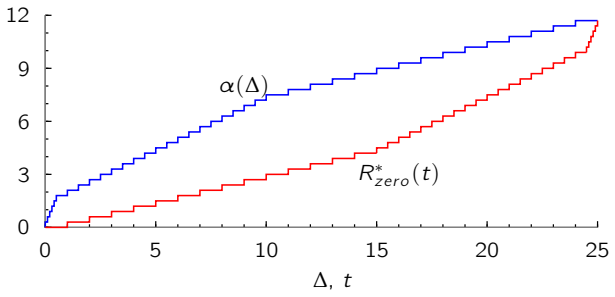

Figure 1. Input arrival curve

As a final comment, we would like to highlight the design principle that we can derive from this analysis. As a direct consequence of the monotonicity principle, for all initial temperatures, the worst-case delay of a stream of jobs is monotonic with respect to the input arrival rate. For instance, if there are two software designs where the input arrival rates are α_1 and α_2, respectively, with $\alpha_1 > \alpha_2$, then independent of the parameters of $\mathbf{P}$ or the initial temperature, the worst-case delay of α_1 cannot be smaller than that of α_2. This clear separation of principles satisfyingly reinforces the common idea that bursty output must be avoided, if possible.

V. Experimental Results

We first discuss the parameters of the system as used in the following experimental results. The speed scaling rule is given by

$$\begin{aligned} f(T) &= 2, && 0 \le T < 30, \\ &= 1.414, && 30 \le T < 50, \\ &= 1, && T \ge 50. \end{aligned}$$

The power model is given by $P(s) = 15s^2$. The thermal impulse response is given by $h(t) = \exp(-0.3t)$. The input arrival rate is as shown in Figure 1. The slowest speed $s^{\min}$ is 1, and the steady-state temperature at this speed is $T^{\max} = 50$, which coincides with the domain of f. Thus, all model assumptions are satisfied.

A. Zero Initil Temperature

We first consider the case of zero initial temperature. In this case the worst-case trace R^*_{zero}, given by (10), is shown in Figure 1. The trace of the temperature and speed of $\mathbf{P}$ for executing this trace is shown in Figure 2. The worst-case delay is obtained to be 1.3, and is indeed suffered by the last job of the trace.

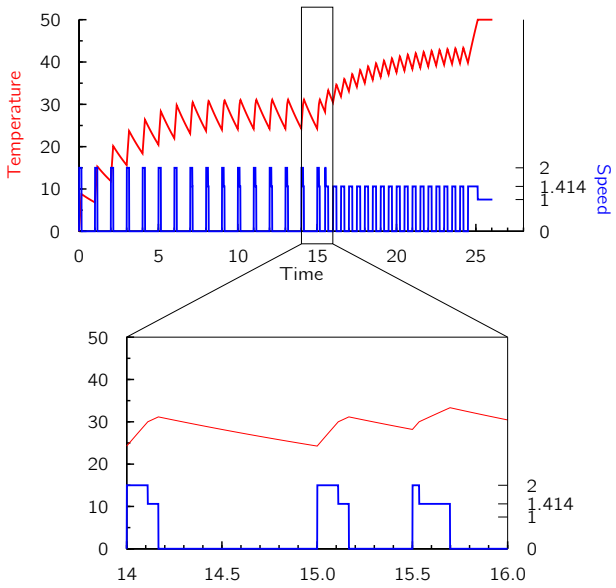

Figure 2. Temperature and speed traces for $T(0) = 0$

B. The length of the time horizon

We have presented the analysis where we observe the arrivals of jobs within an interval $[0, \tau]$, for a given τ. It is easy to see that the worst-case delay cannot decrease on increasing τ. The question then is how to choose a small value of τ, while expecting a reasonably accurate worst-case delay computation.

The choice of τ is related to how input arrival curves are specified. In principle, arrival curves have to be specified for all $\Delta \geq 0$. However, α is specified to be asymptotically periodic beyond a certain Δ. Let us call this the tail of the α curve. In the considered α of Figure 1, this tail starts at 10. Recollect that in the worst-case trace R^*_{zero}, the tail arrives at the beginning. Then, τ should be chosen to be large enough that the temperature of $\mathbf{P}$ while executing the tail also changes periodically, i.e., the temperature of $\mathbf{P}$ at the start of subsequent periods must be approximately the same.

For our example, from Figure 2, it is evident that while executing the tail from $[0, 15]$, the system reaches a periodic behavior. So, the chosen value of $\tau = 25$ is large enough. This is also clear from Figure 3. In all subsequent simulations, we set $\tau = 25$.

C. Simulation on $\mathbf{P}$'

We now consider a higher initial temperature, $T(0) = 35$. From Theorem 7, we simulate the trace R^*_{zero} on $\mathbf{P}$' with temperature threshold 35. The trace is shown in Figure 5. Note how the temperature is forced to the threshold of 35. The worst-case delay is indeed suffered by the last job and equals 1.055. The latest time instant when the temperature is forced to the temperature threshold, denoted as ρ, is 15.

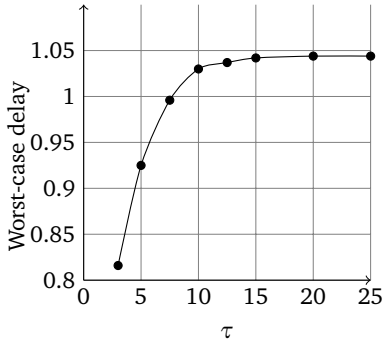

Figure 3. Worst-case delay variation with the time horizon τ.

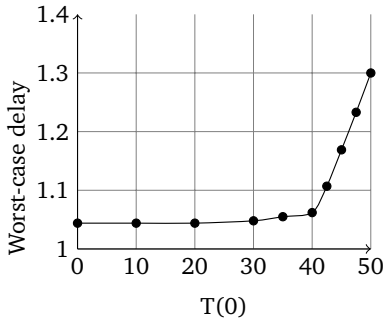

Figure 4. Worst-case delay variation with the time horizon the initial temperature $T(0)$.

This can be used to compute the worst-case trace of jobs for execution on $\mathbf{P}$ using (20).

D. The role of the initial temperature

The worst-case delay depends on the initial temperature. This relationship is plotted in Figure 4. As expected, the relationship is monotonic. For the special case of maximum initial temperature, $T(0) = 50$, we can validate the result obtained by computing the worst-case delay using (14). This is graphically illustrated in Figure 6. The obtained delays in either method are both 1.3.

It is easy to see that ρ (and thus the length of the worst-case trace on $\mathbf{P}$) should be monotonic with respect to the initial temperature. Indeed, for all $T(0) < 24.3$ we have

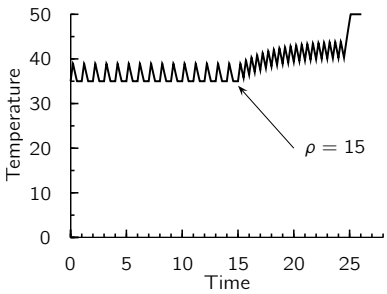

Figure 5. Simulation of $\mathbf{P}$' with temperature threshold of 35

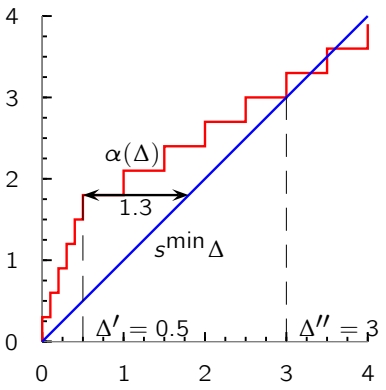

Figure 6. Computing the delay for the maximum initial temperature using (14)

$\rho = 0$, for all $T(0) \in [24.3, 40.6]$ we have $\rho = 15$ and otherwise $\rho = 24.5$. The interpretation is that in the worst-case trace, the burstier part of the trace arrives earlier as the initial temperature increases.

VI. Concluding Remarks

Thermal behavior of a processor is governed by several system parameters. Dealing with variability in these parameters is essential in guaranteeing temperature constraints. Controlling the speed of a processor based on feedback from temperature sensors provides advantage of providing a peak temperature guarantee independent of timing properties of the task. However, analysis of real-time properties guaranteed by such a processor seems to be more involved. In this work, we first argued that an intuitive notion of monotonicity can be extended to such systems, under the considered model assumptions. With the use of such a property we analyzed the worst-case delay of jobs when served by the processor. Importantly, for all initial temperatures, the presented analysis provides a tight delay bound obtained by simulating a single trace of the system. The ease of the presented analysis motivates further study and mainstream use of such processors.

There are two directions of future work. One direction is to design analysis techniques for multi-core systems with partitioned scheduling, where the workload for each processor is fixed. In such a system, the temperature of one core can affect that of another and thus the presented theory has to be broadened. Another direction is to consider multiple streams of jobs under a given scheduling policy being served by a single processor. It will be interesting to evaluate if monotonicity principle can be extended to other scheduling policies other than FCFS.

VII. Related Work

Several recent works have considered the problem of thermal-aware real-time scheduling. However, the focus of the work has been on speed scaling that is based on temperature feedback. The first work that considered the use of feedback control for thermal management seems to be [11]. The authors showed that feedback control can be used to provide adaptive and localized thermal management. [12] discusses in greater detail the design of the feedback control law, while authors in [13] demonstrate the use of model-predictive control. However, none of these works simultaneously consider real-time properties. As far as we know, the only literature on real-time analysis of such processors are the papers by Wang and Bettati [5] and [7]. We thus focus on a more detailed comparison to the these works.

The presented Lemmas 1 and 2 are similar in scope to results of Lemma 1 of [7]. However, our results are applicable to a larger class of systems because of the following generalizations: (a) the speed scaling rule considered in [7] has only two different speeds, whereas we allow for any piecewise-constant non-increasing function f to define the speed scaling rule, (b) the power consumption model is any general convex function in our paper whereas in [7] a specific model has been used, (c) the thermal model in our system is only specified by a monotonically decreasing impulse response, whereas results of [7] are presented for a specific model. It is to be noted that the specifics of the model in [7] (which we abstract) are used to prove the results there.

In [7], the authors also present the delay analysis of a stream of jobs characterized by an input arrival rate α. Apart from having the above model restrictions the presented results have further limitations. The authors consider the initial temperature as an unknown and assume that the maximum temperature is reached at some point of time. The presented results correspond to our second case when we assume that the initial temperature is $T^{\max}$. For this case, the authors present the delay analysis in Theorem 1 of [7], where the worst-case delay is to be computed by solving potentially very large number of non-linear inequalities. It is not even clear how many such inequalities would have to be solved. As noted by our result in Theorem 6, computing a tight upper-bound on the worst-case delay is straight-forward: compute the horizontal distance between $\alpha(\Delta)$ with $s^{min}\Delta$. We have also highlighted how the solutions for the other cases of initial temperatures are interesting and a unifying analysis technique can be formulated.

Acknowledgements

This work was supported by EU FP7 projects EURETILE and PRO3D, under grant numbers 247846 and 249776, respectively.

References

[1] N. Bansal, T. Kimbrel, and K. Pruhs, "Dynamic Speed Scaling to Manage Energy and Temperature," in *FOCS*, 2004.

[2] J.-J. Chen, S. Wang, and L. Thiele, "Proactive Speed Scheduling for Real-Time Tasks under Thermal Constraints," in *RTAS*, 2009.

[3] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *ICCAD*, 2007.

[4] D. Rai, H. Yang, I. Bacivarov, J.-J. Chen, and L. Thiele, "Worst-Case Temperature Analysis for Real-Time Systems," in *DATE*, 2011.

[5] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," *Real-Time Systems*, vol. 39, no. 1-3, 2008.

[6] J.-Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Lecture Notes in Computer Science, Springer, 2001.

[7] S. Wang and R. Bettati, "Delay analysis in temperature-constrained hard real-time systems with general task arrivals," in *RTSS*, 2006.

[8] P. Kumar and L. Thiele, "Thermally optimal stop-go scheduling of task graphs with real-time constraints," in *ASP-DAC*, 2011.

[9] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *ISCAS*, 2000.

[10] P. Kumar and L. Thiele, "Cool Shapers: Shaping real-time tasks for improved thermal guarantees," in *DAC*, 2011.

[11] K. Skadron, T. F. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management," in *HPCA*, 2002.

[12] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. D. Koutsoukos, and H. Wang, "Feedback thermal control for real-time systems," in *RTAS*, 2010.

[13] F. Zanini, C. N. Jones, D. Atienza, and G. D. Micheli, "Multicore thermal management using approximate explicit model predictive control," in *ISCAS*, 2010.

[14] W. Huang and *et al*, "Hotspot: A compact thermal modeling methodology for early-stage vlsi design," *IEEE Trans. VLSI Syst.*, vol. 14, no. 5, 2006.

APPENDIX

Modelling scope of the thermal model

Recall that we consider a thermal model which is entirely described by the impulse response of the temperature for a unit power consumed, denoted as $h(t)$. We assume that this function is monotonically decreasing. We illustrate here how some commonly used models can be formulated using this model.

Conductive thermal model: A commonly used thermal model is the conductive heat dissipation model, wherein the temperature of the processor evolves according to the following differential equation

$$GT(t) + C\frac{dT(t)}{dt} = P(t), \tag{21}$$

where G and C are thermal parameters. Note that we have already discussed how the value of ambient temperature can be considered to be 0, without loss of generality. Then it can be shown that the above system has a monotonically decreasing impulse response given as

$$h(t) = \frac{1}{C}\exp\left(\frac{-t}{C/G}\right). \tag{22}$$

Temperature-dependent power consumption: In newer technologies leakage power is a significant proportion of the total power consumption. Unfortunately, leakage power increases with temperature. Let this relationship be modelled as $P(t) = \phi(s(t)) + \eta \cdot T(t) + \nu$, where ϕ is the convex function which denotes the dynamic power and the leakage power is represented as a linear function of the temperature. Then, the differential equation can be written as

$$(G - \eta)T(t) + C\frac{dT(t)}{dt} = \phi(s(t))\nu \tag{23}$$

This is similar in form to (21). Thus, if the temperature-dependence of power consumption or thermal parameters can be approximated to be linear, the impulse response exists and is monotonically decreasing.

Non-lumped cooling system: As the final modelling scope, we consider a system where the cooling network is more elaborate. As a reference, we cite the model of the cooling system in [14]. Here the active heat generating processor is interfaced with the ambient through several layers including substrates, pads, heat spreaders and sinks. In such a system, the thermal model would be a more elaborate RC network with multiple nodes, where each node has a capacitance to the ground and nodes are interconnected with positive resistances. It can be shown that, for any such network, the impulse response of the temperature of the heat-generating node is monotonic. Thus, as long as the temperature sensor is placed at the single heat generating node, the model assumptions hold for non-lumped cooling systems.

What cannot be modelled: If the heat transfer is governed by non-linear physical processes such as radiation, or if the heat generation is itself distributed, then the thermal system cannot be modelled under the assumed framework.