COOLIP: Simple yet Effective Job Allocation for Distributed Thermally-Throttled Processors

Pratyush Kumar, Hoeseok Yang, Iuliana Bacivarov, Lothar Thiele
Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
{firstname.lastname@tik.ee.ethz.ch}

Abstract—**Thermal constraints limit the time for which a processor can run at high frequency. Such thermal-throttling complicates the computation of response times of jobs. For multiple processors, a key decision is where to allocate the next job. For distributed thermally-throttled procesosrs, we present COOLIP with a simple allocation policy: a job is allocated to the earliest available processor, and if there are several available simultaneously, to the coolest one. For Poisson distribution of inter-arrival times and Gaussian distribution of execution demand of jobs, COOLIP matches the 95-percentile response time of Earliest Finish-Time (EFT) policy which minimizes response time with full knowledge of execution demand of unfinished jobs and thermal models of processors. We argue that COOLIP performs well because it directs the processors into states such that a defined sufficient condition of optimality holds.**

I. Introduction

Most processors offer a range of operating frequencies which can be changed at runtime. For instance, the AMD Opteron 2218 series processor has five P-states with performance varying up to 2.6x [1]. The P-states with higher frequency can execute tasks faster, but consume more power. Choosing the appropriate P-state that balances performance and power objectives is a key challenge.

Besides power, another main concern is on-chip temperature. The current practice is to execute at high frequency for managed, short durations of time such that safe temperature thresholds are not exceeded. Consider the example of Turbo Boost in the Intel Sandy Bridge processors [2]. Turbo Boost can run the processor at up to 1.2-1.3x the Thermal Design Power (TDP). However, the high frequency can be sustained only for up to 30-60s. Subsequently, the processor must run at nominal TDP and build up a *thermal budget*. Thus, *availability of high operating frequencies is thermally-throttled*.

Multiple distributed processors may provide a solution to thermal-throttling. Under moderate utilization, some processors may have the thermal budget to execute at high frequency and absorb a burst of workload. After this, it may be the turn of other processors to execute at higher frequency. To take advantage of such *thermal-multiplexing*, we need a good policy to *allocate* incoming jobs on the processors. In this work, we study an allocation policy for homegeneous thermally-throttled processors to minimize the response time of jobs.

Allocating jobs considering thermal effects has been studied in two different domains. First, in the data-center context, job allocation is used to minimize response times and the high cooling costs [3], [4]. Second, for chip multi-processors, job allocation is used to meet deadlines and peak temperature requirements [5], [6]. In both these domains, job allocation is *co-optimized* along with the control of cooling mechanism in data-centers and custom frequency scaling in chip multi-processors. In contrast, we follow the approach suggested in [7] with a separation of concerns. The temperature is managed by some given policy, in our case by thermal-throttling of the processor, and the job allocation tries to minimize response time given such thermal-throttling. Apart from a decoupled design, this approach helps identify basic principles which can then be verified for the co-optimized case.

We present COOLIP, which stands for COOLest among Idle Processors. A job is allocated to the earliest available processor, and if there are several available simultaneously, to the coolest one. We study the 95-percentile response time of a trace of jobs with Poisson distribution of inter-arrival times and a Gaussian distribution of execution demands. Recent results highlight the importance of optimizing the *tail* of the response time distribution in data-centers [8]. We show that COOLIP, which only requires the current temperatures, performs similar to the Earliest Finish-Time (EFT) policy, which computes the optimal allocation decision to minimize response time with full knowledge of execution demand of unfinished jobs and thermal models of processors. This result is observed for varying ranges of the number of processors, the utilization and the workload parameters. We argue that this surprisingly good performance of COOLIP is because it directs the processors into states such that often a specific *sufficient condition* holds. This condition guarantees that the allocation made by COOLIP minimizes the response time of the next job.

The rest of the paper is organized as follows. In Section II we specify the thermal and workload models. In Section III we prove a sufficient condition under which a simple allocation policy optimally minimizes the response time of the next job. Based on this result, we define COOLIP in Section IV and compare its performance with other policies in Section V.

II. System Model

We consider N_p homogeneous thermally-throttled processors executing a trace of jobs in First-Come-First-Serve order.

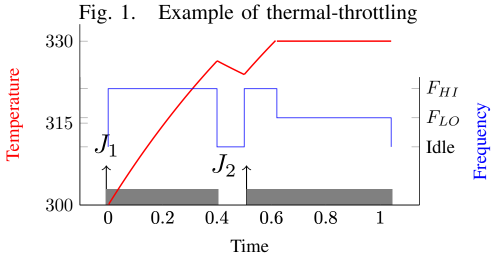
Thermal Model: In system-level studies, the thermal-model of a processor is specified as an equivalent RC network [9], [5], [10]. For such a model, for each frequency F at which

978-3-9815370-2-4/DATE14/©2014 EDAA

TABLE I
THERMAL MODEL OF AN EXAMPLE SYSTEM

T_{IDL}^{∞}	T_{LO}^{∞}	T_{HI}^{∞}	τ	F_{LO}	F_{HI}
300	330	380	1	1	1.5

Fig. 1. Example of thermal-throttling



the processor operates, the evolution of temperature follows

$$T(t) = T(0) + (T_F^{\infty} - T(0))(1 - \exp(-t/\tau)), \quad t \geq 0, \quad (1)$$

where T_F^{∞} is the steady-state temperature when continuously executing at the frequency F and τ is the rate constant. T_F^{∞} depends on the ambient temperature, power consumption at that frequency, and the thermal conductance between the processor and the ambient through its heat-sink [9]. The rate constant denotes the rate of heat transfer and depends on the thermal conductance and capacitance of the processor. When the processor is not executing any job, it is said to be idle with a steady-state temperature T_{IDL}^{∞}. The thermal parameters are the same for all processors. We neglect heat exchange between processors. This holds if the cooling mechanism isolates processors, or if the effect of heat exchange can be approximated by a higher ambient temperature.

Thermal-Throttling: Each processor has two frequencies F_{HI} and F_{LO} with steady-state temperatures T_{HI}^{∞} and T_{LO}^{∞}, respectively. Frequency at time t, denoted $F(t)$, is given as

$$\begin{aligned} F(t) &= F_{HI}, \quad \text{if } \; T(t) < T_{LO}^{\infty}, \\ &= F_{LO}, \quad \text{else.} \end{aligned} \quad (2)$$

The above throttling ensures that the temperature of a processor does not exceed T_{LO}^{∞}, independent of the workload.

Workload: We consider a trace of jobs with the standard assumptions: the inter-arrival times follow a Poisson distribution with rate λ and execution demands follow a Gaussian distribution with mean C_{μ} and standard deviation C_{σ}. Execution demand is the time required to finish the job at the lower frequency F_{LO}, and scales linearly with frequency.

Example: Consider thermal parameters shown in Table I. Throughout, all temperature values will be in Kelvin and time values in seconds. Let jobs J_1 and J_2 arrive at times 0 and 0.5 with execution demand of 0.6 each. The execution of these jobs on one processor is shown in Fig. 1. J_1 executes at the high frequency F_{HI}, but heats up the processor. This leads to thermal-throttling during the execution of J_2.

As seen in this example, with thermal-throttling the response time of a job depends on the previous execution profile of the processor. This complex dependence should be considered in allocating the jobs on multiple processors.

III. A SUFFICIENT CONDITION FOR MINIMIZING RESPONSE TIME

In this section, we will establish a condition under which a simple job allocation policy is the optimal one. This will then motivate the design of the COOLIP policy in the next section. We begin by defining two properties for each processor.

Definition 1 (Finish-time at time t)**.** *Assume no jobs are allocated on or after* t*. Then, finish-time of processor* p_i *at* t*, denoted as* $t_i^f(t)$*, is the time when* p_i *will finish executing all jobs allocated to it before* t*. Also,* $t_{\max}^f(t)$ *is the largest finish-time across all processors, i.e.,* $t_{\max}^f(t) = \max_{\{i:p_i \in P\}} t_i^f(t)$*.*

The finish-time specifies when a processor will be idle and hence ready to execute another job. If a processor p_i is already available at time t, then $t_i^f(t) = t$. Also, by earliest available processor we refer to the one with the smallest finish-time.

Definition 2 (All-Idle Temperature at time t)**.** *Assume no jobs are allocated on or after time* t*. The all-idle temperature of a processor* p_i *at time* t*, denoted as* $T_i^{ai}(t)$*, is the temperature of* p_i *at time* $t_{\max}^f(t)$*.*

The all-idle temperature of a processor is its temperature when all processors become idle. A processor which is idle earlier can cool down until all other processors are idle. Both the finish-times and the all-idle temperatures remain unchanged until a new job is allocated as given below.

Lemma 1. *If no jobs are allocated in the time interval* $[t_1, t_2]$*, then* $t_i^f(\cdot)$ *and* $T_i^{ai}(\cdot)$ *for every processor* $p_i \in P$ *remains invariant in the domain* $[t_1, t_2]$*.*

We are now ready to define the sufficient condition. The condition says that if the earliest available processor has the lowest all-idle temperature, then allocating the next job to that processor minimizes the job's response time.

Theorem 2. *At some time* t*, if for some processor* $p_u \in P$*,*

$$t_u^f(t) = \min_{\{i:p_i \in P\}} t_i^f(t), \quad \text{and} \quad (3)$$

$$T_u^{ai}(t) = \min_{\{i:p_i \in P\}} T_i^{ai}(t), \quad (4)$$

then allocating the next job to p_u *minimizes its response time.*

Proof: Let job J be allocated to $p_v \neq p_u$, and begin executing at time t. Then, from (3) p_u is idle at t, and from (4) p_u cannot have a higher temperature than p_v at time t. Since, response time of a job monotonically increases with the initial temperature of the processor [10], executing J on p_u cannot finish later than when executing on p_v. ■

The theorem establishes a sufficient condition. Indeed, it provides no guarantee when (3) and (4) are not satisfied by any processor. If the conditions are met, we still need to decide *when* to allocate the next job. We may either allocate the job as soon as possible, or we may wait and let the processors cool down. The following result, adapted from simliar results in [7], [10], establishes the optimality of a greedy solution.

Lemma 3. *Let at some time* t*, the processor* $p_u \in P$ *satisfy (3) and (4), then allocating the next job to* p_u *as soon as possible optimally minimizes the job's response time.*

In conclusion, if (3) and (4) are satisfied, then we wait for processor(s) to be idle and allocate the job to the coolest one. Importantly, this result does not depend on thermal or job parameters, or allocation decisions thus far.

IV. COOLIP POLICY

In this section, motivated by the sufficient condition of the previous section, we define the COOLIP policy. We also present a numerical case to understand how the policy works.

Definition 3 (COOLIP policy)**.** *Upon arrival, a job is inserted into a global First-In-First-Out (FIFO) queue. Whenever a processor is idle the job at the head of the global FIFO queue is allocated to it. If several processors are idle simultaneously, the job is allocated to the coolest processor.*

Note the economy in the inputs of COOLIP policy. It only requires (a) the temperatures of all processors, and (b) if the processors are idle or busy. Both of these are usually available. It does not need models for heat dissipation, the thermal-throttling conditions or execution demand of pending jobs. However, in COOLIP, a job is only allocated when some processor becomes idle. This requires a small communication latency between the FIFO queue and the processors.

COOLIP implicitly chooses the processor with minimum finish time by deferring the allocation until a processor is idle. If multiple processors are idle simultaneously, then the one with the smallest temperature has the smallest all-idle temperature (Lemma 1). Also COOLIP greedily allocates the job (Lemma 3). These properties lead to the following result.

Lemma 4. *If the sufficient condition of Theorem 2 is satisfied at some time t, then the COOLIP policy minimizes the response time of the next job allocated on or after t.*

COOLIP is optimal if (3) and (4) hold. How often does this occur? With a specific example, we will illustrate that COOLIP ensures that sufficient conditions often hold. To this end, we take three steps. First, we develop a scheme to represent the *state* of the processors at a given point of time. Then, for a given state we show how to compute the probability of satisfying the sufficient condition. Finally, we show how to compute steady-state probabilities of different possible states.

(a) A Scheme to Represent the State of the Processors: From Lemma 1, we know that finish times and all-idle temperatures are constant between allocation decisions. Thus, we can specify these parameters *once per job*. For every job J, we represent the state S by a set of tuples for each processor. For processor p_i its state is the tuple (t_i, T_i), where t_i and T_i are the time and temperature, respectively, when p_i becomes idle. t_i is specified *relative* to the arrival time of the job before J, denoted J'. If some processor p_i is idle before J' arrives and J' is not allocated to it, then we set $t_i = 0$ and T_i is the temperature of p_j at the arrival time of J'.

The valid tuples are bounded: t is a non-negative number upper-bounded by worst-case response time, while $T \in [T^{\infty}_{IDL}, T^{\infty}_{LO}]$. If we discretize the possible values of t_i and T_i, we obtain a finite, though approximate, set of states $\mathbf{S}$.

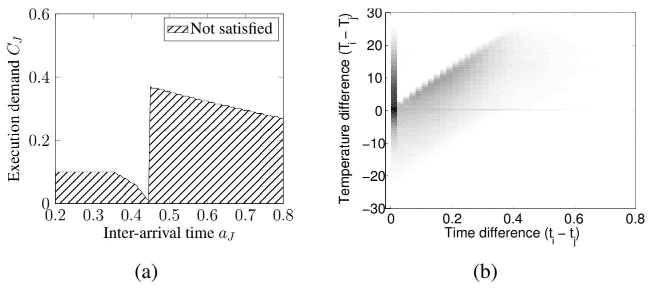


Fig. 2. (a) For state S^* satisfaction of sufficient condition. Not satisfied in shaded region. (b) Heat-map of the steady-state probability of different states. Darker points have higher probability, white region has negligible probability.

(b) Probability of Satisfying the Sufficient Condition from a given State: Consider a setup of two processors with thermal model as in Table I. For the workload, let $C_\mu = C_\sigma = 0.25$ and $\lambda = 0.5$. Let us study the allocation policy for a specific job J, when the state of the processors is

$$S^* = \{(t_1, T_1) = (0.35, 330), (t_2, T_2) = (0.45, 305)\}.$$

For each state $S \in \mathbf{S}$, we compute the probability, denoted $\rho(S)$, that (3) and (4) hold *after* the execution of J. Two random variables affect $\rho(S)$: the arrival time and the execution demand of J. Both these are governed by respective probability distributions. Let the inter-arrival time between J and the previous job be a_J, and the execution demand of J be C_J. Then, for different values of a_J and C_J, we can check if the sufficient condition is satisfied.

For the state S^*, we plot in Fig. 2(a) if (3) and (4) hold. Note the three distinct regions: $a_J < t_1$, $a_J \in [t_1, t_2]$ and $a_J > t_2$. Multiplying this data with the probability of the individual events, we compute $\rho(S^*) = 0.21$. The low value of $\rho(S^*)$ indicates that COOLIP is often not guaranteed to minimize response-time in the state S^*. Thus, COOLIP may not always make the optimal allocation decision.

(c) Steady-State Probability: We compute the steady-state probability distribution for each state, denoted $\eta(S)$. In other words, when executing the COOLIP policy, $\eta(S)$ is the long-term probability of visiting the state S. We construct the Markov transition probability matrix A, where the (i, j)th element is the probability of moving from state S_j to state S_i. The eigenvector of A gives the steady-state probability, η.

For the considered example, we plot in Fig. 2(b) the steady-state probability η, in a 2-D graph where the states are represented by the differences in the tuples for either processor. States with high η have either (a) $t_i = t_j = 0$, i.e., both processors are idle when the job arrives, or (b) $t_i > t_j$ and $T_i > T_j$, i.e., the processor with later finish time has a higher finish temperature. For states belonging to either case, we expect high probability of satisfying the sufficient condition. We confirm this numerically: For the most probable states accounting for about 75% of the total probability, the value of η for each state is 1, i.e., the sufficient condition is guaranteed to be true for each of these probable states.

From this example, we draw the following conclusions. COOLIP is not optimal and there are states such as the considered S^* where it can make poor decisions. However, when allocating a trace of jobs with COOLIP, the processors are often driven into states wherein COOLIP performs optimally.

V. Comparison with Other Allocation Policies

In this section, we will compare COOLIP with other policies. First we define the other policies we consider.

Earliest Finish-Time (EFT): The EFT policy computes the finish time if allocating the next job to each of the processors, and then picks the allocation that minimizes the finish time. This computation is done with full knowledge of the thermal models of the processors, the throttling conditions, and execution demand of all unfinished jobs. Thus, EFT optimally minimizes the response time of the next job.

Load Balancer (LB): The LB policy allocates the next job to the processor which has been allocated the smallest cumulative execution demand until then. LB requires the execution demand of all unfinished jobs.

Round Robin (RR): The RR policy allocates job to the least recently allocated processor.

Random Available Processor (RAP): The RAP policy allocates the next job to the first available processor. If there are several, it chooses randomly. Comparing COOLIP to RAP checks the importance of the temperature feedback.

In all the experiments, the thermal parameters are as in Table I. For different experiments, by performance of a policy (except EFT) we refer to the percentage difference between the 95-percentile response time of that policy and the 95-percentile response time of EFT policy. Each data point is the average of performance numbers for 100,000 different simulations.

(a) Varying utilization (Fig.3(a)): We consider two processors. We define utilization U as $U = \frac{C_\mu}{N_P \times \lambda}$. We set $C_\mu = C_\sigma = 0.25$, and vary λ according to the value of U.

(b) Varying execution demand (Fig.3(b)): We consider two processors. We vary C_μ, with $C_\sigma = C_\mu$. We fix $U = 0.8$ and compute λ, accordingly.

(c) Varying number of processors (Fig.3(c)): We fix $C_\mu = C_\sigma = 0.25$ and $U = 0.8$. We vary the number of processors N_P, and set set λ accordingly.

(d) Mixed-Type Workload (Fig.3(d)): We consider a trace of jobs from two streams with different parameters executing on two processors. We set $(C_\mu)_1 = 0.25$ and vary $(C_\mu)_2$. We set $U_1 = U_2 = 0.4$ and set λ_1 and λ_2 accordingly.

We make the following observations from the results.

- COOLIP matches the 95-percentile response-time of EFT. In all our experiments, the worst-case performance difference between COOLIP and EFT was a negligible 0.004%.
- LB is worse than COOLIP (by about 1%). Note that LB uses the execution demand of all pending jobs, which is often not known. In spite of this, LB performs worse than COOLIP, which instead uses the current temperatures.
- RR performs significantly worse than COOLIP (on average 2x), and shows strong dependence on different parameters. Thus, due to variability in job parameters thermal-multiplexing does not follow a cyclical order.
- RAP performs significantly worse than COOLIP (on average 1.5-2x), especially for high utilization. Note that RAP is similar to COOLIP, except that it does not choose the coolest processor when there are several available. This highlights the importance of the temperature feedback.

VI. Conclusions

The simple COOLIP policy effectively minimizes the 95-percentile response times of jobs on distributed thermally-throttled processors. While the choice of COOLIP is not always the optimal one, making a series of allocations with COOLIP prepares the right conditions under which it performs well. Further interest arises in heterogeneous processors and thermal coupling between processors. We believe that in such cases too, the current temperatures of the processors can be used to define simple yet effective allocation policies.

Acknowledgment: This work described in this paper has been partially supported by the TRANSCEND Strategic Action from Nano-Tera.ch.

References

[1] Dell, "Managing Data Center Power and Cooling," 2007. [Online]. Available: http://www.dell.com/downloads/global/power/ps1q07-20070204-AMD.pdf

[2] A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power management architecture of the 2nd generation Intel® Core microarchitecture," 2011. [Online]. Available: www.hotchips.org

[3] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, and J. S. Chase, "Balance of power: Dynamic thermal management for internet data centers," *Internet Computing, IEEE*, vol. 9, no. 1, pp. 42–49, 2005.

[4] R. Z. Ayoub and *et al*, "Temperature aware dynamic workload scheduling in multisocket cpu servers," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 30, no. 9, pp. 1359–1372, 2011.

[5] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *RTAS*. IEEE, 2009.

[6] Y. Xie and W.-L. Hung, "Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (MPSoC) design," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 45, no. 3, pp. 177–189, 2006.

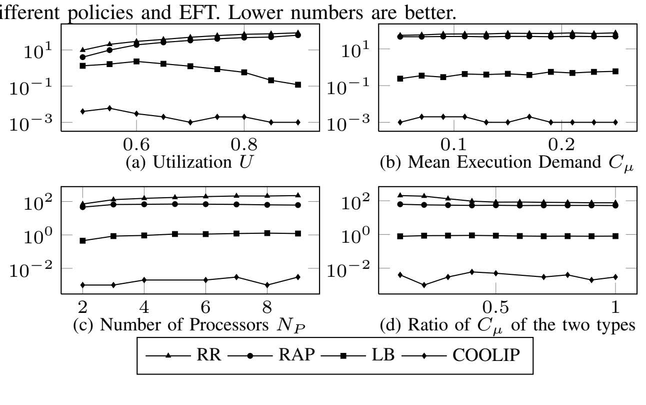
[7] S. Wang and R. Bettati, "Delay Analysis in Temperature-Constrained Hard Real-Time Systems with General Task Arrivals," in *RTSS*, 2006.

[8] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

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

[10] P. Kumar and L. Thiele, "Timing Analysis on a Processor with Temperature-Controlled Speed Scaling," in *RTAS*, 2012, pp. 77–86.

Fig. 3. Difference in percentage between 95-percentile response time of different policies and EFT. Lower numbers are better.


10^1
10^{-1}
10^{-3}
(a) Utilization U
(b) Mean Execution Demand C_μ
10^2
10^0
10^{-2}
(c) Number of Processors N_P
(d) Ratio of C_μ of the two types
RR
RAP
LB
COOLIP