A Distributed and Self-Calibrating Model-Predictive Controller for Energy and Thermal management of High-Performance Multicores

Andrea Bartolini, Matteo Cacciari, Andrea Tilli, Luca Benini
Email: a.bartolini,matteo.cacciari,andrea.tilli,luca.benini@unibo.it
University of Bologna, DEIS
via Risorgimento 2
40136 Bologna, Italy

Abstract

High-end multicore processors are characterized by high power density with significant spatial and temporal variability. This leads to power and temperature hot-spots, which may cause non-uniform ageing and accelerated chip failure. These critical issues can be tackled on-line by closed-loop thermal and reliability management policies. Model predictive controllers (MPC) outperform classic feedback controllers since they are capable of minimizing a cost function while enforcing safe working temperature. Unfortunately basic MPC controllers rely on a-priori knowledge of multicore thermal model and their complexity exponentially grows with the number of controlled cores.

In this paper we present a scalable, fully-distributed, energy-aware thermal management solution. The model-predictive controller complexity is drastically reduced by splitting it in a set of simpler interacting controllers, each allocated to a core in the system. Locally, each node selects the optimal frequency to meet temperature constraints while minimizing the performance penalty and system energy. Global optimality is achieved by letting controllers exchange a limited amount of information at run-time on a neighbourhood basis. We address model uncertainty by supporting learning of the thermal model with a novel distributed self-calibration approach that matches well the controller architecture.

I. Introduction

CPUs have become high-performance multi-cores, characterized by high power density and average power consumption. This leads to complex and expensive thermal dissipation solutions [1]. In addition, significant spatial and temporal variability of workloads leads to non uniform performance, power consumption and temperature distribution [2]. On-die hot-spots are subject to larger static power, due to the exponential dependency of leakage current on temperature. Moreover, hot spot areas age faster since degradation effects, such as NTBI, and HCI [3], are exponentially accelerated by high temperatures. This in turn may lead to early chip damage or failure. Hot-spot prevention uniquely based on worst-case thermal design is very expensive in terms of system cost and/or performance penalty. For this reason, significant effort has been devoted towards techniques to dynamically control the cores power dissipation in a temperature-aware fashion, i.e. aiming to enforce a safe working temperature across the die surface [4]. Today multiprocessors include hardware support for dynamic power and thermal management, based on introspective monitors [5], sensors and performance knobs. This infrastructure provides the sensors and the actuators for feedback control management policies.

978-3-9810801-7-9/DATE11/

A. Related Work

Power budgeting and power capping [6] techniques use built-in power meters as inputs to close-loop feedback controllers for constraining the power consumption to a given budget by reducing the cores clock frequencies. This approach has two main drawbacks: first, it relies on the availability of accurate and reliable power measurement infrastructures; second, in many cases it does not leads to a global energy reduction due to execution time overhead. Indeed, high power dissipation phases usually are correlated with CPU-bound computational ones. Thus power budgeting techniques happen to reduce the frequency mainly in this situation, leading to energy losses due to static power and to execution time linear dependency with frequency [7]. Different solutions have been presented to achieve a system energy reduction instead of power capping, but unfortunately the energy minimization alone cannot enforce a safe working temperature [7].

Closed-loop thermal control policies aim to address this problem. In [8], [9], [10] the authors show the benefit of feedback-control approaches vs. open loop policies based on temperature thresholding heuristics. Model predictive controllers (MPC) [11] [6] outperform classic feedback controller, which cannot take into account hard constraints in the state space. MPC controllers instead rely on an system model [12] to predict the future temperature while finding the optimal control action by solving a constrained optimization problem *for one or more control steps in the future*. Thus they generally lead to higher-quality control, providing that an effective thermal model is available.

Wang et al. [6] present a MPC that constraints both the power and the temperature of the cores while maximizing the performance. It uses power sensor as input to the optimization problem and to generate on-line a power to frequency linear model. This reduces the complexity of the controller (even though it can lead to sub-optimal controller corrective actions). Zanini et al. [11] assume a workload input requirement, so that the MPC minimizes the performance loss in tracking it while constraining the core temperatures. Internally it adopts a non-linear

frequency to power model, statically precomputed off-line avoiding usage of power sensors. Unfortunately the adopted model is simplistic and it does not consider the power dissipation dependency on workload properties, assuming it to be only related to core frequency.

Performance of MPC solutions strongly depends on the thermal model accuracy. Unfortunately, often an accurate model is not a-priori available. Different approaches have been proposed to identify it from HW introspective monitors. In [6] a first order thermal model is estimated using off-line least square method. Differently, Cochran et al. [13] extract a set of linear models to relate temperatures to workload characteristics and core frequencies. Eguia et al. [14] combine identification techniques with an overfitting remedying algorithm to reduce complexity and improve accuracy; nevertheless, the results need to be improved for fast power changes and thermal interactions between adjacent cores. all the above solutions exploit centralized control and deal with model identification of the whole die. Their complexity and computational burden increase fast with the number of cores. Therefore applying these solutions in upcoming many-cores [15] is very expensive. This complexity problem has been addressed in the control-theory literature. Several authors [16] [17] have shown how to reduce significantly the complexity of MPC by using distributed solutions. This approach is well-suited for large-scale systems and consists in dividing the control problem into sub-problems with lower complexity.

B. Contributions

In this paper we present a complete distributed solution that combines energy minimization, MPC based thermal capping and thermal model self-calibration. according to the incoming task workload characteristics, each local node first selects the minimum frequency (f_{EC}) that preserves the performance within a tolerable overhead. Second, if necessary each local MPC controller trims the frequency to ensure a safe working temperature. Local controllers jointly optimize global system operation by exchanging a limited amount of information at run-time on a neighbourhood basis. Third we address model uncertainty by self-calibration: each thermal controller node extracts automatically the local thermal model by applying a set of training stimuli and monitoring the thermal response of the neighbourhood area. The distributed controller strategy combined with the distributed thermal model calibration phase allow us to take advantage of the parallelism of the underlying multi-core platform by running different instances of the controller and self-calibration routine in parallel.

The paper is organized as follows. Section II introduces the key aspects of power and thermal issues in a multicore scenario. Section III describes the building blocks of the proposed solution. In Section IV the performance of the presented distributed energy-aware thermal controller are fully evaluated in a real uses case scenario by implementing it in a full-system virtual platform. Final conclusions are drawn in Section V.

II. Background Concepts

In a multicore scenario, the temperature distribution across the die area depends on the chip floorplan, its thermal environment, and the power consumption of the cores. The latter has been shown to be related to the operating point/performance level and workload characteristics, such as instruction type, density and data locality [18].

Fig. 1a shows the results of a set of tests performed to quantify the relationship existing between power, frequency and Clocks-Per-Instruction (CPI) of the running task, for each core in a general purpose multicore[1]. The dots represent the actual power consumption whereas the solid lines represent the fitting model curve extracted by these data, described by Eq. 1:

$$P = k_A freq * V_{DD}^2 + k_B + (k_C + k_D freq) * CPI^{k_E} \qquad (1)$$

From the figure we can notice that core dynamic power depends non-linearly on frequency, sublinearly on the CPI of the application and the two dependencies are coupled, CPI dependency is influenced by frequency value.

From Fig. 1b we can notice that a significant power reduction can be achieved, not only in cpu-bound program phases (low CPI) but also in memory-bound phases (high CPI). This is a key effects to use dynamic voltage and frequency scaling to obtain energy-efficiency instead of only power reduction. Indeed whereas scaling the frequency of a core executing a cpu-bounded task brings to a performance loss and to a energy inefficiency due to static power, scaling down the frequency of a core executing a memory-bound task does not lead to execution time overhead, thus the dissipated total energy is reduced.

To derive a complete thermal model of multicore dies, the "causal chain" $(Frequency, CPI) \rightarrow DissipatedPower \rightarrow Temperature$ can be split in two parts. The first part can be addressed separately in each core according to Eq. 1. Differently, the temperature variation in each point of the die will be affected by: the distribution of the power dissipated by all the cores, the current die temperature map and the chip physical properties. Nevertheless, the whole powers-to-temperatures model can be split in simpler interacting models by dividing the die area in regions, aligned with cores for obvious convenience.

According to the granularity usually required for thermal control, we can assume uniform power and temperature distributions in each core area. Then, recalling Fourier's law, the temperature of each core can be assumed dependent on its own dissipated power, ambient temperature and adjacent cores temperatures (boundary conditions). This assumption is actually straightforward for continuous time models only. When discrete-time models are considered, a larger coupling among cores has to be considered to account for the "chain of interactions" taking place during the blind intervals among samplings. Recalling again the Fourier's Law, the coupling among two cores will be inversely related to their distance and directly related to the sampling time period. Hence, the "equivalent neighbourhood" of a core depends on the floorplan combined with the adopted sampling.

To verify this assumption we took an example loosely correlated with the Intel® SCC experimental architecture [15]. The floorplan is fully tiled with 48 core/regions, each with an area of $11.82 mm^2$ and a maximum power consumption of 2.6W. We used this set-up with the HotSpot thermal analysis tool [19], stimulating it with a power step in the central core (21) while keeping all the other cores at zero power consumption. Fig. 1c shows with different colours the cores that increases their temperature as result of the central core power step after different time interval. We can notice that the radius of thermal influence of the central core increases with the time interval: within 50ms

[1] Intel® Xeon® X7350

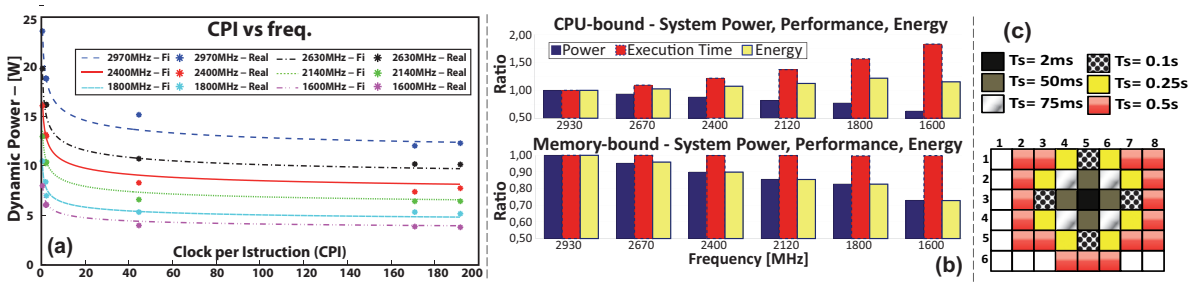


Fig. 1. Multicore exploration results

it impacts only the closest core along the four cardinal directions, whereas at 1s all cores are effected.

Thus, if a small sampling time is adopted with respect to thermal time-constants (50ms or less), the effect of time-discretization can be neglected, assuming the equivalent neighbourhood equal to the physical one. These considerations are the basis for developing our distributed thermal control.

Finally, results in [20] highlight that the thermal dynamics of each core is characterized by two time constants: a faster one, at a few ms, is related to the silicon surface, whereas the slower one, at a few seconds, is related to the heat spreader. This behaviour, needs to be carefully accounted in model identification and control design.

III. Architecture

Fig. 2 depicts the block diagram of the proposed solution. Each controller node (i) is made-up by three main parts:

- The Energy Controller (EC_i): at the k-th sampling instant, it takes as input the predicted CPI value of the running task for the following time interval ($CPI_i([k,k+1]|k)$) and produces as output the core frequency settings for the following (k to k+1) time interval ($f_{EC_i}(k)$) that minimizes the core power consumption while allowing a tolerable performance overhead.
- The MPC-based Thermal Controller: at the k-th interval, it receives as inputs the Energy Controller output frequency ($f_{EC_i}(k)$), its own core temperature ($T_i(k)$), the temperature of the physical neighbours ($T_{neig_i}(k)$)[2] and the ambient temperature ($T_{AMB}(k)$). Then, according to the safe reference temperature (T_{MAX}) at which the core temperatures ($T_i(k)$) must be constrained, the MPC algorithm adjusts the actual frequency command ($f_{TC_i}(k)$), minimizing the displacement from the Energy Controller requirement[3].
- The Thermal Model Self-Calibration Routine: it automatically derives, off-line, the local, but interacting, thermal prediction model adopted in MPC-based TC blocks (again according to Section II).

Section III-A describes the Energy Controller algorithm whereas Section III-B describes our thermal controller solution. Section III-C instead presents the thermal model self-calibration routine.

[2]The sampling time is assumed small enough.

[3]The computation and actuation times for EC and TC are assumed negligible with respect to sampling time interval. Hence, for mathematical modelling, control outputs are considered generated at the same instant of sampled inputs.

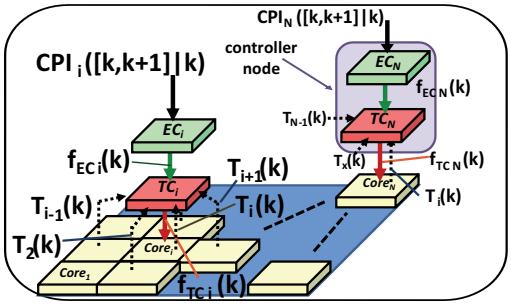


Fig. 2. General Architecture

A. Energy Controller

The goal of the Energy Controller (EC_i) is to provide the optimal frequency trajectory ($f_{EC_i}(k)$) to the Thermal Controller (TC_i). This frequency is selected to minimize the power consumption while preserving the system performance. Thus we can split the energy minimization problem from the temperature constraining one. The Energy Controller takes advantage of the parallel architecture by letting each core (i) compute autonomously the future frequency in line with the incoming workload requirements.

Considering an in-order architecture[4] and the average time needed to retire an instruction, composed by two terms: (T_{ALU}) portion of time spent in active cycles and (T_{MEM}) portion of time spent in waiting for memory cycles. Whereas the first term is proportional to the input frequency, the second one is constant to it and depends to the memory access latency.

Assuming f_M the maximum frequency allowed by the system, $f_{CK_i}(k) = \frac{f_M}{\alpha}$ a generic one and given the task CPI requirement for the interval(k), for the core i ($CPI_i(k)$), we can write the task execution time ($Time_i(k)$) as:

$$Time_{M_i}(k) = \#_{INST} \cdot [1 + (CPI_i(k) - 1)] \cdot \frac{1}{f_M} \quad (2)$$

$$Time_{CK_i}(k) = \#_{INST} \cdot [\alpha + (CPI_i(k) - 1)] \cdot \frac{1}{f_M} \quad (3)$$

By combining them the execution time overhead % can be represented as function of the new frequency $f_{CLK_i(k)}$ and $CPI_i(k)$ as reported in Eq. 4.

$$\%_i(k) = \frac{Time_{CK_i}(k)}{Time_{M_i}(k)} - 1 = \frac{\alpha + (CPI_i(k) - 1)}{1 + (CPI_i(k) - 1)} \quad (4)$$

[4]Multicore trend is toward in integrating high number of simpler processor[15].

Inverting the last equation (Eq. 5) we can find the future frequency ($f_{EC_i}(k)$) output of the Energy Controller (EC_i) that minimizes the power consumption for the given processor i running a task characterized by the $CPI_i([k,k+1]|k)$, while preserving the performance within a tolerable performance penalty (%).

$$f_{EC_i}(k) = \frac{f_M}{(1+\%)+(CPI_i([k,k+1]|k)-1)\cdot\%} \quad (5)$$

B. Thermal Controller

Our solution goal is tracking the desired cores frequencies, without crossing the cores temperature bounds imposed by each local MPC controllers of our distributed approach. As discussed in Section I, we use MPC controllers because it has been shown in [11] and [6], that well perform the previous aim. A MPC controller consists of two elements: the optimization problem solver and the model used for predictions.

First, each of our local controllers minimizes a quadratic cost function with constraint, formalized as:

$$min \sum_{k=0}^{N-1} \|Q\cdot(f_{TC}(t+k)-f_{EC}(t+k))\|_2 \quad (6)$$

$$\text{s.t.} \qquad 0 \le T(t+k|t) \le T_{MAX} \quad (7)$$

This formulation simply means that, to maximize the performance and respect the thermal constraint of one core, the differences (also called tracking error) between the desired core frequency f_{EC} and the frequency selected by the thermal controller f_{TC} must be minimized. At the same time the temperature T must be constrained below a thermal safety threshold, called T_{MAX}. Matrix Q is the non-negative weights matrix for the tracking error and N is the prediction horizon.

This optimization problem has been obtained by "locally projecting" the problem formulation proposed in [11], where a single MPC controller optimally constraints all the cores temperatures, maximizing the global performance. Later in this section we refer it as "centralized solution", where f_{EC}, f_{TC} and T are vectors rather than scalars.

Second, each local MPC regulator estimates future core temperature by using the following non-linear model for the single core:

$$\dot{x} = A\cdot x + B\cdot[P_{EC}, T_{AMB}, T_{NEIGH}]' \quad with \quad P_{EC} = g(f_{EC}, WL) \quad (8)$$

where x is the state vector (one is T), P_{EC} is the power consumption of the core, WL is the workload and T_{NEIGH} is the temperature contribution of the neighbours cores.

[21] shows that MPC model must be linear to have a convex problem and a simple and robust control solution. Thus we have confined the non-linearity outside the MPC, in the frequency to power relation (expressed by the $g(\cdot)$ function) and kept only the linear part of the prediction model inside the LQ MPC controller. As consequence the core power becomes the controlled variable, $g^{-1}(\cdot)$ transformation is needed to convert the output power (P_{TC}) in frequency command and the cost function can be expressed using core power error instead of core frequency error. This is shown in Fig. 3 that describes the distributed controller architecture.

The linear model accuracy is a key issue of MPC controllers. We preserve it by extracting the model parameters directly from the target multicore. This is achieved thanks to a self calibration routine explained in Section III-C. As discussed in Section II the core thermal transient shows

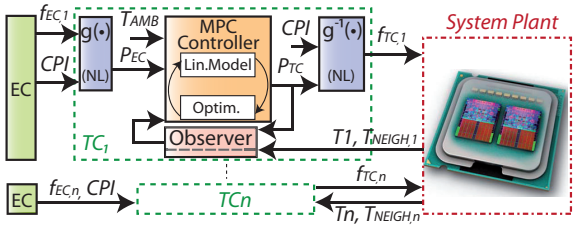
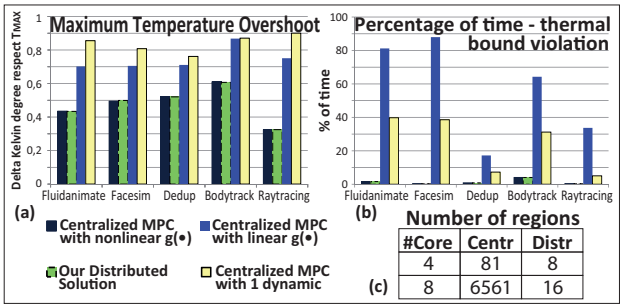


Fig. 3. Thermal Controller structure



#Core	Centr	Distr
4	81	8
8	6561	16

Fig. 4. Thermal MPC performance analysis

two time-scale responses, thus a second order linear model should be used for each core. This leads to a model order (number of states) larger than the number of temperature sensors (one per core). Therefore an observer block is required to estimate the state values, as in Fig. 3.

We tested our solution against the centralized one that, differently from [6], it uses a non linear power model and, differently from [11], it uses a more accurate power model ($g(\cdot)$) that account also the workload properties. We evaluate the benefits of these add-ons in trace-driven simulator[5] running Parsec2.1 [22] benchmark traces.[6] Fig. 4a and b, show the performance losses in the "centralized solution" by first, substituting the non-linear $g(\cdot)$ function with a linear[7] one (as in [6]) and secondly, by using a first order prediction model instead of a second order model (as in [6]). Both cases shown a significant performance worsening.

To reduce MPC computation complexity in [12] an explicit approach has been proposed to reduces the computational burden (one of the major drawbacks of the classical MPC strategy). It solves the optimization QP problem off-line for each system state $x(t)$, instead of solving it on-line for each time step. The obtained solution is a continuous piecewise affine function of the state ($u(x(t))$) similar to divide the state space in regions each one characterized by a different linear control law. Thus, depending on the current state value a different control law is applied on system. Unfortunately the explicit solution only moves the complexity from the computational cost to the memory usage [12](confirmed by results in [11]). Thus we can use the regions number of the explicit formulations as a complexity metric for the on-line controllers, since it directly relates to computational cost. Fig.4c shows that our distributed solution improves scalability and effectively reduces complexity. Whereas the number of regions of the centralized solution grows exponentially, the number of regions of the distributed one globally grows linearly.

[5]Internally it considers the four core floorplan presented in [18].

[6]Traces are profiled on a general purpose quad core.

[7]Best least-squares fit of the power data using as input the mean frequency and the mean workload

Indeed the complexity of the single controller node remains constant regardless the number of cores in the system. Also note that the distributed solution runs in parallel on all the cores encouraging robustness, safety and a lower computational burden for the cores, while the centralized one runs only on one core. Even though the distributed solution complexity is lower than the centralized one, Fig.4a and b shows that both the solutions have comparable performances.

C. Self-calibration routine

The knowledge of the thermal model of the multicore die is a MPC prerequisite. In many practical cases this model is not available or imprecise. We address model uncertainty by self-calibration: our thermal control framework directly learns the thermal model by monitoring the thermal response of the system after applying a set of training stimuli. This approach relies on system identification (SID) techniques. As shown in previous section and in [23], complex prediction models complicate the control action and cause overhead. Thus the implemented approach has two aims: capturing entirely the system thermal response and reducing the state space model of the plant that in reality it would have an infinite dimension.

Indeed our self-calibration routine is distributed: each core is associated with a self-identification agent that learns the local model. This approach perfectly fits our distributed control solution, since the regulator of each core directly exploits the identified local model for prediction. Secondly, it offers a low complexity solution to counteract the SID computational cost in large multicore systems. Indeed for MIMO model the SID complexity explodes with the number of inputs. Each agent implements an ARX (AutoRegressive eXogenous) model [24] [25]:

$$T(k) = \alpha_s \cdot T(k-1) + \cdots + \alpha_1 \cdot T(k-s) + \beta_{1,s} \cdot u_1(k-1) + \ldots$$
$$+ \beta_{1,1} \cdot u_1(k-s) + \beta_{2,s} \cdot u_2(k-1) + \cdots + e(k) \qquad (9)$$

where T is the temperature of the core (the model output), s is the model order, $u_i(\cdot)$ are the model inputs (the dissipated power of the core P_{EC}, the ambient temperature T_{AMB} and the temperatures of the neighbours units), $e(k)$ is a stochastic white process with null expected value representing the model error and α_i, $\beta_{i,j}$ are the identified parameters. As shown in Section II and in Eq. 9, each model is a simple MISO model: we have a single output and multiple inputs $u_i(\cdot)$. The core power consumption can be estimated from the core operating point and from the current workload characteristic (see Eq. 1 in Section II) or can be directly measured from power sensors present in recent MPSoC [15].

The self-calibration routine first forces a Pseudo-Random Binary Sequence (PRBS) power input to each core, while probing the cores temperature. Then it derives the parameters α and β by solving a least square problem that minimizes the error $e(k)$. To take into account both the slow and the fast dynamics highlighted in Section II, we use a second order ARX model[8]. Fig. 5a shows the model and plant temperature responses to a different PRBS from the self-calibration one.

[8]The identified models states have not a physical meaning. To match the core temperature with the first state of each model we apply a change of coordinate transformation to obtain a matrix $C = [I_n \mid 0_n]$

The performance of the identified model against the original one are evaluated by looking at the temperature response of each core running Parsec2.1 benchmarks. Fig. 5b shows the mean absolute errors of the identified model in a four cores floorplan[18] and in an eight cores floorplan obtained duplicating the previous one. The resulting errors are less than $0.3\,^{\circ}K$. The showed results are obtained running simulations on Matlab/Simulink environment. In a real system we expect to run the self-calibration routine during start up phase and each time the model behaviour differs from plant one.

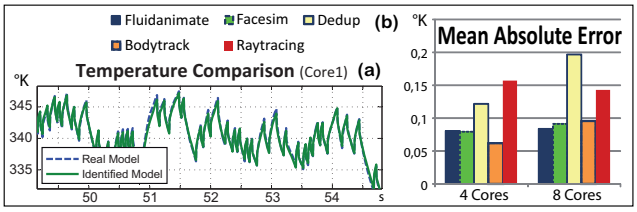


Fig. 5. Self-calibration routine results

IV. Experimental Results

In this section we illustrate the performance of our solution. We implemented it in a virtual platform environment [18]. This virtual platform emulates a general-purpose multicore running in a full system[9]. Indeed it simulates real benchmarks running on full O.S. It also supports per-core DVFS, and it estimates the power consumption and temperature evolution of the entire multicore. It exploits a MATLAB/Simulink interface that allows high level description of the controllers routines reacting to the emulated sensors and driving the performance knobs on the emulated system.

We implement our thermal controller solution in this framework as follows. For each core (i) in the emulated system we execute, with a time step of 1ms, both the Energy Controller(EC_i) and the Thermal Controller (TC_i) routines. The Energy Controller internally estimates the CPI every 1ms by using a last value prediction. This interval of time is comparable with modern O.S. Instead the Thermal Controller routine embeds, as presented in Section III-B, the explicit MPC implementation and estimates the full state vector with the state observer. Complexity analysis in Section IV-A demonstrates that the distributed solution has negligible run-time. Thus, the perturbation due to its computations to the program execution flow can be neglected.

A. Tests Results

In this section we show the results of the proposed solution running on the virtual platform. Each controller runs on each core of the target architecture [18] under different Parsec2.1 [22] benchmarks workloads. All the benchmarks have been executed with a number of tasks equal to the number of cores of the target architecture, and with the input set "simsmall", and constrain temperature ($T_{MAX} = 330\,^{\circ}K$)[10]. We run each Parsec benchmark under four possible configurations: Original, only Energy Controller(EC), Centralized Thermal Controller (Centr TC) and our Distributed Thermal Controller (Distr TC).

[9]The target system is composed by four x86, in order, cores with 32 KB private L1, 4MB shared L2 cache and 2GB of DRAM.

[10]Used thermal model is calibrated on a device with high performance thermal dissipation dynamics, indeed to stress our thermal controller we are forced to use a lower temperature constraint

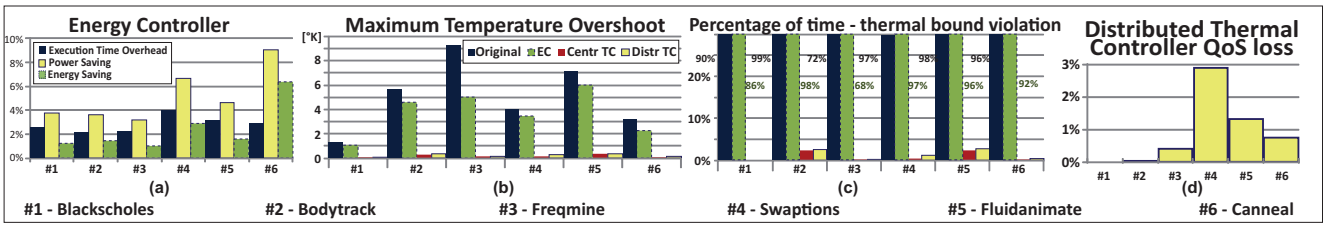


Fig. 6. Virtual platform test results

Fig. 6a shows the performance of the EC alone, while allowing a performance penalty of $\%_i(k) = 5\%$. We can notice that it is able to maintain the performance overhead under the selected threshold while achieving a significant power and energy saving.

The global solution performance are instead analysed by considering the maximum temperature overshoot with respect to the constraint, the percentage of time the temperatures violate the constraint (we consider the limit violated when temperature exceeds it of $0.1\,^{\circ}K$), and a metric that quantify the controller quality of service (QoS) degradation due to thermal constraint (QoS Loss). We decide to compute it as the mean squared error between the energy controller frequency target (f_{EC}) and the one applied to the system by the controller (f_{TC}). We relativized it against the centralized controller one. Fig. 6 shows the results collected. First, we can notice that the proposed distributed solution performs as well the centralized one. Fig. 6b shows the maximum overshoot in kelvin degree above the safe thermal threshold (T_{MAX}) whereas Fig. 6c shows that both solutions are capable of drastically reducing the portion of time in which each core runs out of the thermal bound. Looking at the QoS Loss performance figure (Fig. 6d), we notice that our proposed solution performs at the same level of the centralized one, with a degradation less than 3%. Finally in more symmetrical workloads[11], such as `swaptions`, `fluidanimate`, `canneal`, we noticed that the average frequency applied to the external cores (#1, #4) is kept lower (up to -14%) than the internal cores. This is a sign that the MPC controller is able to optimize the core frequency locally, taking advantage of the difference between the local thermal models. Indeed the external thermal models have less thermal dissipation headroom since thermal model considers the chip lateral boundary adiabatic [26].

V. Conclusions

We have presented a novel fully distributed, energy-aware, MPC-based thermal capping controller for modern multicore platforms. It works coupled with a model self-calibration routine, that allows the controller to automatically recognize the thermal properties of the underlying platform. It exploits a fully distributed architecture, that fits very well the multicore parallelism and distributed nature. We show that this approach performs similarly to the state-of-the-art centralized Thermal Model Predictive Controllers, but with a significantly lower computational cost. Indeed the global complexity cost of our solution scales linearly with the number of cores and it is fully parallel, whereas the centralized one scales exponentially and it parallelization is not trivial. We tested our solution in a real uses case scenario by running it in a complete virtual platform. The results show that our controller is capable to satisfy temperature constraints with high precision, while minimizing system energy.

[11]The parallel benchmark executes the same code on all the processors.

VI. Acknowledgements

This work was supported, in parts, by Intel Corp., Intel Labs Braunschweig and the EU FP7 Projects Pro3D (GA n. 248776) and Therminator (GA n. 248603).

References

[1] IDC. Worldwide server power and cooling expense 2006, 2010 forecast. http://www.sun.com/service/eco/IDCWorldwideServerPowerConsumption.pdf.
[2] Hanson H. et al. Thermal response to DVFS: analysis with an Intel® Pentium® m. In ISLPED '07, pages 219-224, 2007.
[3] Tiwari A. et al. Facelift: Hiding and slowing down aging in multicores. MICRO '08, pages 129-140, 2008.
[4] P Chaparro et al. Understanding the thermal implications of multi-core architectures. IEEE Transactions on Parallel and Distributed Systems,18(8):1055-1065, Aug. 2007.
[5] Intel Corporation. Intel® 64 and IA-32 Architectures Software Developer's Manual - Volume 3B, June 2009.
[6] Y. Wang, K. Ma and X. Wang, "Temperature-Constrained Power Control for Chip Multiprocessore with Online Model Estimation", *ISCA*, 2009.
[7] G. Dhiman, T. S. Rosing, "Dynamic voltage frequency scaling for multi-tasking systems using online learning", *ISLPED*, August 27-29, 2007, Portland, OR, USA.
[8] K. Skadron, T. Abdelzaher, M. R. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management". *Technical Report*. UMI Order Number: CS-2001-27., University of Virginia.
[9] M. Kadin, S. Reda, A. Uht, "Central vs. distributed dynamic thermal management for multi-core processors: which one is better?". *GLSVLSI* 2009. ACM, New York, NY, 137-140.
[10] Z. Wang, X. Zhu, C. McCarthy, P. Ranganathan, and V. Talwar, "Feedback Control Algorithms for Power Management of Servers". *FeBid*, Annapolis, MD, June 2008.
[11] F. Zanini, D. Atienza, L. Benini and G. De Micheli, "Multicore Thermal Management with model predictive control", *IEEE*, 2009.
[12] A. Bemporad, M. Morari, V. Dua and E.N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems", *Automatica*, Vol. 38, 2002, pp 3-20.
[13] R. Cochran, S. Reda, "Consistent runtime thermal prediction and control through workload phase detection". *DAC* 2010. ACM, New York, NY, 62-67.
[14] Thom J. A. Eguia, Sheldon X.-D. Tan, Ruijing Shen, Eduardo H. Pacheco and Murli Tirumala, "General Behavioral Thermal Modeling and Characterization for Multi-core Microprocessor Design". *DATE* 2010 Dresden
[15] J. Howard et al, "A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS", *ISSCC* 2010.
[16] E. Camponogara, D. Jia, H. Krogh and S. Talukdar, "Distributed Model Predictive Control", *IEEE Ctl. Sys. Mag.* 22 (2002) (1), pp. 44-52.
[17] R. Scattolini, "Architectures for distributed and hierarchical Model Predictive Control". *Journal of Process Control* (May 2009), 19 (5), pg. 723-731.
[18] A. Bartolini, M. Cacciari, A. Tilli, L. Benini and M. Gries "A Virtual Platform Environment for Exploring Power, Thermal and Reliability Management Control Strategies in High-performance Multicores", *GLSVLSI*, 2010.
[19] Huang Wei et al. "Accurate, pre-RTL temperature-aware design using a parameterized, geometric thermal model". *IEEE Trans. Comput.*, 57(9):1277-1288, 2008.
[20] W. Huang, K. Skadron, S. Gurumurthi, R. J. Ribando, and M. R. Stan. "Differentiating the roles of IR measurement and simulation for power and temperature-aware design", *ISPASS*, 2009.
[21] E. F. Camacho and C. Bordons, "Model Predictive Control" *Springer*, 1999.
[22] C. Bienia, S. Kumar, J. P. Singh and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications", *PACT*, 2008.
[23] F. Zanini, D. Atienza, G. De Micheli, S. P. Boyd, Online Convex Optimization-Based Algorithm for Thermal Management of MPSoCs, *GLSVLSI*, 2010.
[24] L. Ljung, "System Identification - Theory For the User", *2nd ed, PTR Prentice Hall* Upper Saddle River, N.J., 1999.
[25] R. Guidorzi, "Multivariable system identification", *Bononia University Press*, 2003
[26] G. Paci, M. Morari, V. Dua and E.N. Pistikopoulos, "Exploring temperature-aware design in low-power MPSoCs", *DATE*, Vol.1, 2006, pp 1-6.