Robust identification of thermal models for in-production High-Performance-Computing clusters with machine learning-based data selection

Robust identification of thermal models for in-production High-Performance-Computing clusters with machine learning-based data selection

Federico Pittino*, Roberto Diversi*, Luca Benini*†, Andrea Bartolini*
*Department of Electrical, Electronic and Information Engineering (DEI), University of Bologna, Italy
{federico.pittino, roberto.diversi, luca.benini, a.bartolini}@unibo.it
†Integrated Systems Laboratory, ETH Zurich, Switzerland {lbenini}@iis.ee.ethz.ch

***Abstract*—Power and thermal management are critical components of High-Performance-Computing (HPC) systems, due to their high power density and large total power consumption. The assessment of thermal dissipation by means of compact models directly from the thermal response of the final device enables more robust and precise thermal control strategies as well as automated diagnosis. However, when dealing with large scale systems "in production", the accuracy of learned thermal models depends on the dynamics of the power excitation, which depends also on the executed workload, and measurement nonidealities, such as quantization. In this paper we show that, using an advanced system identification algorithm, we are able to generate very accurate thermal models (average error lower than our sensors quantization step of 1°C) for a large scale HPC system on real workloads for very long time periods. However, we also show that: 1) not all real workloads allow for the identification of a good model; 2) starting from the theory of system identification it is very difficult to evaluate if a trace of data leads to a good estimated model. We then propose and validate a set of techniques based on machine learning and deep learning algorithms for the choice of data traces to be used for model identification. We also show that deep learning techniques are absolutely necessary to correctly choose such traces up to 96% of the times.**

I. Introduction

High performance computing (HPC) systems are designed to be at the cutting edge of computing capabilities. To achieve this goal, HPC installations are characterized by high computational power density and as a consequence, by high power density as well as large total power consumption. Indeed HPC systems have 2-4x higher rack power density w.r.t. server and industrial datacentre installations, with a per rack power envelope ranging between 20-100 kW [1]. High power density and envelope are obviously critical for HPC system management and operation.

Today one of the most powerful supercomputers in Top500 is Sunway TaihuLight which consumes 15.3 MW for delivering 93 Petaflops. One of the previous first ones, Tianhe-2, consumes 17.8 MW for "only" 33.2 Petaflops. However, the power consumption increases to 24 MW when considering also the cooling infrastructure[2]. Such an amount of cooling power serves to prevent thermal issues. In fact, the performance of the processing elements is actively controlled by the internal firmware logic, which modulates chip voltage and frequency for maximizing the clock speed while satisfying power and thermal constraints. However these mechanisms are usually reactive, threshold-based and take significant safety margins: authors in [3] show that, for hot-water liquid cooled nodes, the processors are incapable of employing thermal throttling by using DVFS states to prevent the critical thermal threshold to be reached.

To solve these issues, several works in the literature [4], [5], [6], [7], [8], [9], [10], [11], [12] propose to take advantage of proactive thermal and power management strategies. These strategies all rely on the availability of compact predictive power and thermal models, capable of predicting future power consumption and temperature of the system and, even more importantly, to build a clear understanding on the sensitivity of these on workload parameters and hardware knobs that can be controlled at run time. Such models allow to estimate and model the power consumption of the entire CPUs and their cores based on workload characteristics extracted through performance counters and micro-architectural usage. Thanks to that, an optimizer can leverage these models to find the maximum clock frequency to apply based on the current usage of the micro-architecture while satisfying a global power budget or thermal constraints.

Compact models can be used in combination with optimization and artificial intelligence techniques to select in a robust fashion the optimal operating points from the target power and temperature and the current conditions [4], [5], [6], [7], [8], as well as to estimate hotspots and peak temperatures [12]. Moreover, such compact models can be used also for detecting anomalous changes in the behaviour of the system, for example due to a failure. However, the strategies for learning these models rely on design-time parameters [10] that cannot cope with manufacturing variability, which makes each chip different from the others. Moreover, differences in deployment conditions and aging which can induce very significant differences in compact model parameters even for nominally identical nodes. In addition, such models have been applied only to single node systems operating in a test environment and therefore cannot easily be utilized at the scale of a full system in production without causing significant calibration costs of all its nodes (e.g. bringing the HPC machine off-line periodically for power and thermal characterization).

In case an identified compact thermal model needs to

be periodically updated during the lifetime of the system, additional challenges arise concerned with identifying models under production workloads, which cannot be calibrated to the needs of the model identification process. In fact, it is well known that the input signals may affect significantly the results of an identification experiment [13], [14]. The choice of a suitable input signal (when possible) has been extensively treated in the system identification literature [13], [14]. In many identification methods the input data are not available but the type of input excitation is assumed to be known [15], [16], [17]. When dealing with real workloads, as in our case, it is critical to evaluate if a given input (workload) will lead to a sufficiently accurate estimated model. However, even if [13], [14] suggest that persistently exciting of sufficiently high order as well as a low condition number is a requirement for high estimation accuracy, very few previous works have verified if these conditions are sufficient for discriminating real workload traces that lead to good or bad identified models. To the best of our knowledge, the only attempt to define a procedure of data selection for system identification has been reported in [18]. The authors investigate the usage of two metrics, one connected to the model's matrix condition number and the other to the cross-correlation between input and output, and they find that only a combination of the two metrics can discriminate between good and bad windows for identification. The method is however only applied to a single sequence in a specific use-case, and therefore the general performance of the model has not been assessed.

This paper aims at bridging the gap between the theoretical identification of thermal models and their application to real workloads on in-production large HPC system. The paper is organized as follows. Sec. II presents a review with the state-of-the-art and highlights the open problems. Sec. III describes the theoretical foundations of our system identification models. Sec. IV describes our deployment scenario, consisting of a large HPC system under real workloads, and presents the results of thermal identification on our data. Given the large amounts of available data (14 days of continuous operation), we divide it in time windows of 12 hours each, which also enables us to highlight the problem of selecting the right window of data for good model identification. Finally, Sec. V explores the problem of window selection and it shows that only by means of machine learning and deep learning algorithms an accurate selection can be performed.

II. Related work

Several strategies have been proposed in the last decade for extracting compact thermal models directly from a processor chip's thermal response to a given power/workload stress input [19], [20], [21], [22], [8], [23], [24], [25], [26], [11]. The simplest ones do not account for the multimodal nature of the thermal transient caused by the different building materials and their relative time-constants (i.e. die, heat-spreader and heatsink) [8]. In [19], [20], a first order dynamic thermal model is identified by solving a linear Least Squares (LS) optimization problem. Sharifi et al. [27] show that, when a model is available, it can be used effectively to filter out measurement noise using a Kalman filter.

Coskun et al. [21] use an Auto-Regressive Moving Average (ARMA) technique for predicting the future thermal evolution of each core. The derived model predicts future temperature by using only its previous values. Since it does not account directly for workload-to-power dependency, a Sequential Probability Ratio Test (SPRT) technique is used to rapidly detect changes in the statistical residual distribution (average, variance) and, then, to re-train the model, when it is no longer accurate. Juan et al. [22] uses a combination of a K-means clustering and an Auto-Regressive (AR) model to learn a compact model for fast thermal simulation. This approach is effective only when starting from a highly accurate thermal model of the HW. Moreover, the missing exogenous terms in both of the above approaches leads to neglecting the direct link between dissipated power and temperature. Then, the learning of a large number of different models is required to capture the characteristics of different functional units and program phases.

Huang et al. [10] propose HotSpot, a simulation tool to compute the thermal transients, thermal map and steady-state temperature in electronics devices. Whereas the tool allows to accurate model several thermal-related effects given the chip floorplan and some tuning parameters, HotSpot it is not suitable to on-line predict the temperature of a real manufactured chip and a full system. Pagani et al. [12] propose an approach to compute the peak transient temperature in a chip floorplan by analytically solving the response of the state-space dynamic model representing the heat dissipation. The authors show that their approach can compute the peak transient temperature quickly than HotSpot. To apply this approach to a real manufactured chip it is required to extract a thermal model tuned on the specific component.

Reda et al. [11] propose a method for estimating thermal models and power consumption only from the measurements of thermal sensors and total power consumption, without making a-priori assumption on the core's power consumption as well as model structure/parameters. The results are promising but, as in other works, to correctly identify the thermal model it is required to excite the system with a controlled input (power steps long enough to reach steady state conditions). Bartolini et al. [8] present a distributed model learning approach based on a set of Auto-Regressive eXogenous (ARX) models. Each core executes its own model learning routine generating a local thermal model. The model is used internally, in each core, by a local model-predictive controller. However this approach has been applied only to simulated systems and it is based on the assumption that per-core power traces and thermal sensor outputs are accurate and without noise.

Indeed, standard ARX models are suitable to represent the so-called "process noise"[1], but are based on the assumption that input and output data are accurate and not affected by

[1]this is a stochastic process usually injected as additional (unknown) input in order to represent unavoidable model approximations

measurement noise [14], [13]. Beneventi et al. [23] present an Output Error system identification strategy that is robust to quantization noise on the input temperature measurements. This is achieved by adding to the basic optimization problem a set of linear constraints that filter out the model parameters that are not physically valid. The approach is validated on a quad-core server platform. Unfortunately, the proposed methodology cannot handle "process noise".

Previous works have shown that the relation between the core temperature and the dissipated power can be described by a purely dynamic ARX model [25], [24]. ARX models are widely used in system identification since they constitute the simplest way of representing a dynamic process in the presence of uncertainties [14]. Two important features of these models are the possibility of obtaining asymptotically unbiased estimates of their parameters by means of least squares and the absence of stability problems of the associated optimal one step-ahead predictors [14]. Nevertheless, it has been shown in [24] that the classic MISO ARX model is not able to describe properly the thermal dynamics of the system because the estimated models are characterized by relevant negative poles and/or complex conjugate poles. This is in contrast with the physics of thermal systems, where only real positive poles can exist. As explained in [24], this problem is due to the presence of a significant level of measurement noise.

To take into account the presence of this noise, MISO ARX models with noisy input and output have been considered [25], [24]. These models belong to the family of errors-in-variables models and cannot be identified by means of standard least squares and prediction error methods [28]. In [25] Diversi et al. introduced a bias compensated least squares approach for identifying noisy ARX models, which has been extended in [24] with a distributed implementation. These works have been conducted on the Intel Single Chip Cloud computer test device which featured "cheap" ring oscillators as thermal sensors. In [26] a Frisch scheme-based approach is applied to a server class processor operating in free-cooling with variable ambient temperature and the built-in state-of-the-art thermal sensors are affected by quantization noise. The obtained results prove the robustness of the approach.

To extract the models all the above mentioned works [25], [24], [26] rely on the capability of testing the system with Pseudo Random Binary Sequences (PRBS) workloads, where each core, synchronously with the thermal response measurements (with a regular sub-second sampling time) can be forced to execute at in ether a low workload/power (idle) state or high workload/power (power virus) state to emulate a Gaussian distribution of the power stimulus. The binary workload is chosen as it allows to pre-characterize precisely the power consumption of each of the two workload states and thus to create input vectors not affected by measurement noise in conjunction with an exciting workload.

In a realistic scenario, it is interesting to understand if the previously mentioned approaches can be applied to generic user workloads without a priori guarantees on the persistent excitation of the workload as well as whiteness of its spectral components. Indeed the system identification theory requires to excite the system under test in all its modes to be capable of correctly identify its model. However to track physical parameter changes as well as to detect abnormal changes in this parameters it is important to understand: (i) if real workloads, with no guarantees on their excitation, can be used to learn accurate models, and (ii) how to filter out workload windows which would not lead to accurate model identification.

Tackling these open problems is the main contribution of our work. In particular, point (i) above is discussed in Sec. IV, while point (ii) is the topic of Sec. V.

III. Methods

The objective of our work is the development and verification of a thermal model methodology and its integration in a monitoring framework for large scale HPC cluster (Tier0/Tier1) composed by several racks of computing nodes, each of which embedding one or more processing elements (i.e. multi-core CPUs). Such a model is targeting the applications of thermal control and anomaly detection. As far as the requirements are concerned, it has to:

- be predictive in time;
- have good accuracy and be stable in all operating conditions;
- provide temperature estimation at a core level, for most powerful control;
- be able to be estimated on real workloads, in order not to require down-time periods of the cluster for runs of ad-hoc workloads for model estimation.

In order to meet all these requirements, we have decided to extend the distributed and scalable monitoring framework that we presented in [29], [30]. As far as the thermal model is concerned, we have used a variation to the algorithm presented in [26], which makes use of an identification algorithm (see Sec. III-B).

As we discussed, in our conditions several challenges arise:

- the identification algorithms run under the assumption that the system to monitor is stressed with a very exciting (ideally white) input pattern; this condition is very likely to be not verified in real workloads;
- models identification should rely on the fact that there can be time windows where the above conditions for good model identification are verified, but it is not clear how to identify such windows.

In the following we describe in detail the derivation of the models and their integration into the monitoring framework [29], [30].

A. Power model

The thermal model [26] uses as input the power of all cores in a package. Since in the architecture we considered there is no available measurement of the power per core, but only at a package level, the first step is to derive an estimate for the cores' powers. For this purpose, we have modified the model

TABLE I
CHOSEN METRICS

Metric name	Description
freq · C0	(MHz) Actual core frequency multiplied by ratio of core time in state C0
freq_pkg · C0_pkg	(MHz) Actual package frequency multiplied by ratio of package time in state C0
1 - C0_pkg	Ratio of package time in all states except C0

in [30] to obtain a power model at the core level and not at the package level.

The power model is based on the measurement of appropriate metrics derived from the performance counters provided by the CPU architecture (Intel in our case). The procedure used to select the appropriate metrics is explored in detail in [30], here we report only the most relevant points for the paper to be self-contained.

Starting from the performance counters, we have derived a set of metrics by applying transformations and non-linear combinations of different counters. Such metrics comprise measurements of all cores in a package, and of the uncore part. These measurements are meant to cover all phenomena relevant to power consumption, from physical measurements (like temperature and frequency), to the current power-saving state, to the load on the system (eg., instructions per second, cache utilization and others). To choose a subset of the metrics, among all the available ones, we have calculated Pearson's correlation of the metrics with the measured package power, and by retaining only the smallest subset with the highest correlation we derived the ones reported in Tab. I. Note that the set comprises both core metrics and CPU metrics, because our power model needs to estimate also the power dissipated by the uncore. We have verified that these metrics are the smallest subset which gives good accuracy (a decrease from 97% to 91% for the number of points below the 9.7W threshold with respect to the results in [30], see the results in Sec. IV-B), and that the addition of other metrics does not increase the model's performance considerably. All used metrics have also been rescaled in order to lie in the interval $[0, 1]$, in order to avoid numerical instabilities in the model.

The power model is then derived using a linear regression algorithm directly on the chosen metrics. It can be cast in the form:

$$P_{pkg} = \sum_{i \in M_{unc}} \alpha_i m_i + \sum_{k \in C} \sum_{i \in M_{core}} \beta_{i,k} m_{i,k} + \lambda \left(\sum_{i \in M_{unc}} \alpha_i^2 + \sum_{i \in M_{core}} \beta_{i,k}^2 \right) \quad (1)$$

where M_{unc} and M_{core} are the sets of uncore and core metrics, respectively, C the set of cores, m_i and $m_{i,k}$ the values of the metrics, α_i and $\beta_{i,k}$ the regression coefficients and λ the regularization parameter [31]. Such a formulation allows us to partition the package predicted power in two sets of contributions (ignoring the regularization term):

$$P_{pkg} = P_{unc} + \sum_{k \in C} P_k \quad (2)$$

with obvious definitions of the symbols comparing with Eq. 1. With this formulation we can easily extract the partial powers, i.e., P_{unc} is an estimate of the power dissipated by the uncore and P_k is the power dissipated by core k. The regression coefficients are derived through a least-squares procedure, and to this purpose we have used the "ml" package from Apache Spark 2.2.0.

Note that also for the derivation of the power model we use real workloads, and this poses the same challenges we discussed above and in [30]. However, since the accuracy of the power model is not critical and the model itself is relatively simple, we have verified that we are always able to derive a model with relatively good stability and accuracy if we use a long enough interval for the training data, in our case generally 3 days as in [30]. Moreover, as we showed in [30], the training of the power model and its evaluation can be done very efficiently in our scalable framework and they impose only a minimal overhead.

B. Thermal model of a core

The thermal dynamics of a single core of a node are represented by means of a MISO model linking the core's temperature (model output) to the powers of all the node's cores (model inputs). Note that we have decided to model the effects of the cores on each other exclusively through their power consumption and not using directly their temperatures at the previous time instant. This choice is justified both by the facts that the measurements of the temperatures are noisy and that power and temperature are inherently connected measurements, being a change in temperature a consequence of the dissipated power, and therefore including both quantities is not expected to provide great additional information. This is also a result of the time discretization, which makes power of the neighbour cores a direct input to the core's temperature. As discussed in [26], a standard ARX (AutoRegressive eXogenous) model is not suitable to describe the thermal dynamics of a core because of the presence of a significant additive noise corrupting the temperature readings. Therefore, we will adopt the following MISO ARX model with additive output noise

$$\bar{T}(t) + \sum_{i=1}^{n} a_i \bar{T}(t-i) = \sum_{k=0}^{N_c} \sum_{i=1}^{n} b_{ki} P_k(t-i) + w(t) \quad (3)$$

$$T(t) = \bar{T}(t) + v(t), \quad (4)$$

where

- $\bar{T}(t)$ is the actual (unknown) core temperature;
- n is the model order, i.e. the memory of the difference equation;
- N_c is the number of cores of the CPU;
- $P_1, \ldots, P_{N_c}$ are the dissipated powers of all the cores of the node and P_0 denotes the uncore power P_{unc};

– $w(t)$ is the equation error (process noise), assumed to be a zero mean white process with variance σ_w^2;
– $T(t)$ is the measured core temperature;
– $v(t)$ is the additive measurement error, assumed to be a zero mean white process with variance σ_v^2, uncorrelated with $w(t)$.

By defining the vectors

$$\varphi_T(t) = [\, -\bar{T}(t-1) \; -\bar{T}(t-2) \; \dots \; -\bar{T}(t-n)\,]^T \tag{5}$$

$$\varphi_P(t) = [\, P_0(t-1) \; \dots \; P_0(t-n) \, P_1(t-1) \; \dots \; P_1(t-n) \; \dots \; P_{N_c}(t-1) \; \dots \; P_{N_c}(t-n)\,]^T \tag{6}$$

$$\theta_A = [\, a_1 \, a_2 \; \dots \; a_n \,]^T \tag{7}$$

$$\theta_B = [\, b_{01} \; \cdots \; b_{0n} \, b_{11} \; \cdots \; b_{1n} \; \cdots \; b_{N_c 1} \; \cdots \; b_{N_c n} \,]^T \tag{8}$$

model (3)-(4) can be rewritten in the compact form

$$\bar{T}(t) = \varphi_T^T(t)\, \theta_A + \varphi_P^T(t)\, \theta_B + w(t) \tag{9}$$

$$T(t) = \bar{T}(t) + v(t), \tag{10}$$

The identification problem to be solved consists in estimating the model coefficients $a_i, i = 1, \dots, n$, $b_{ki}, k = 0, \dots, N_c, i = 1, \dots, n$ on the basis of N samples of the measured temperature $T(t)$ and of the powers $P_j(t), j = 0, \dots, N_c$. The equation error variance σ_w^2 and the additive noise variance σ_v^2 are also estimated.

The adopted identification algorithm is an evolution of that presented in [26] and is essentially based on the following equations

$$(\Sigma - \tilde{\Sigma})\, \bar{\theta} = 0 \tag{11}$$

$$\Sigma_q\, \bar{\theta} = 0 \tag{12}$$

where

$$\bar{\theta} = \left[\, 1 \, a_1 \; \cdots \; a_n \; b_{01} \; \cdots \; b_{0n} \; \cdots \; b_{N_c 1} \; \cdots \; b_{N_c n} \,\right]^T \tag{13}$$

$$\tilde{\Sigma} = \text{diag}\, [\sigma_v^2 + \sigma_w^2 \; \underbrace{\sigma_v^2 \; \cdots \; \sigma_v^2}_{n} \; \underbrace{0 \quad \cdots \; 0}_{(N_c+1)n}] \tag{14}$$

$$\Sigma = E\,[\varphi(t)\, \varphi^T(t)] \tag{15}$$

$$\Sigma_q = E\,[\varphi_q(t)\, \varphi^T(t)] \tag{16}$$

and

$$\varphi(t) = [\, -T(t) \; \dots \; -T(t-n) \, P_0(t-1) \; \dots \; P_0(t-n) \; P_1(t-1) \; \dots \; P_1(t-n) \; \dots \; P_{N_c}(t-1) \; \dots \; P_{N_c}(t-n)\,]^T \tag{17}$$

$$\varphi_q(t) = [\, P_0(t-1) \; \dots \; P_0(t-q) \; P_1(t-1) \; \dots \; P_1(t-q) \; \dots \; P_{N_c}(t-1) \; \dots \; P_{N_c}(t-q)\,]^T. \tag{18}$$

$E[\cdot]$ denotes the expectation operator. The integer q in (18) is a user-chosen parameter. It can be noted that (11)-(12) is a system of equations where the unknowns are the model coefficients and the noise variances whereas the matrices Σ and Σ_q can be directly estimated from the available data:

$$\hat{\Sigma} = \frac{1}{N-n} \sum_{t=n+1}^{t=N} \varphi(t)\, \varphi^T(t) \tag{19}$$

$$\hat{\Sigma}_q = \frac{1}{N-q} \sum_{t=n+q+1}^{t=N} \varphi_q(t)\, \varphi^T(t). \tag{20}$$

To apply the above mentioned identification approach, the sample matrices $\hat{\Sigma}$ and $\hat{\Sigma}_q$ need to be non singular. This implies that the dissipated powers (input signals) $P_0, P_1, \dots, P_{N_c}$ have to be persistently exciting of sufficiently high order. Nevertheless, as pointed out in [24], the non-singularity of the matrices is often not sufficient to get satisfactory results. For this reason, the application of the algorithm proposed in [26] can be difficult in the framework considered in this paper as we use real workloads and not ad-hoc excitations. This means that it is highly likely that the input is not highly exciting. In [24] some metrics (like the matrix condition number) have been proposed to evaluate the quality of the identified model, and in this work we test such assumptions. In the framework under study it is therefore necessary to include a block which performs the window selection procedure. More precisely, given a set of input-output data, the aim of this block is to evaluate the goodness of the set w.r.t. the implementation of the identification algorithm.

In order to evaluate the performance of the model, as in [26], we rely on a Kalman filter for the prediction of the temperature, using the identified thermal model. The filter is based on the following state space representation of the model (3)–(4):

$$x(t+1) = A\, x(t) + B\, u(t) + G\, w(t+1) \tag{21}$$

$$T(t) = C\, x(t) + v(t) = \bar{T}(t) + v(t) \tag{22}$$

where

$$A = \begin{bmatrix} -a_1 & 1 & 0 & \cdots & 0 \\ -a_2 & 0 & \ddots & \ddots & \\ \vdots & \vdots & & \ddots & \\ \vdots & \vdots & & & 1 \\ -a_n & 0 & \cdots & \cdots & 0 \end{bmatrix} \quad B = \begin{bmatrix} b_{01} & \cdots & b_{N_c 1} \\ b_{02} & \cdots & b_{N_c 2} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ b_{0n} & \cdots & b_{N_c n} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \qquad G = C^T,$$

and

$$u(t) = [\, P_0(t) \, P_1(t) \; \dots \; P_{N_c}(t) \,]^T. \tag{23}$$

The filter allows to compute, at each time step t, the state prediction $\hat{x}(t+1|t)$ and then the prediction $\hat{\bar{T}}(t+1|t) = C\, \hat{x}(t+1|t)$ of the actual core temperature $\bar{T}(t+1)$. This prediction can subsequently be compared with the measured temperature $T(t+1)$ and the prediction error (innovation)

$$\varepsilon(t+1) = T(t+1) - \hat{\bar{T}}(t+1|t) \tag{24}$$

can be exploited as a model performance index.

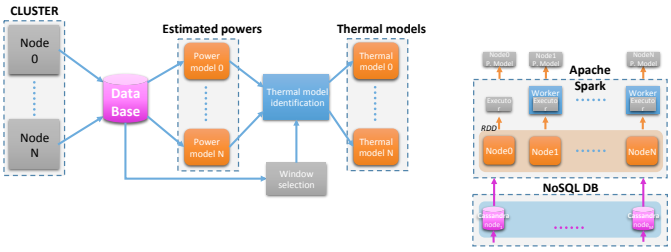


Fig. 1. Model-learning framework

An overview of the framework architecture, where the power and thermal prediction modules and the window selection block are highlighted, is presented in Fig. 1.

Remark 1: Model (3)-(4) refers to a single core of the CPU. In order to model the thermal dynamics of the whole N_c-core CPU, we rely on a multi-input multi-output (MIMO) representation consisting in a set of N_c MISO ARX model with additive noise of type (3)-(4) or (9)-(10). By denoting the actual and measured temperatures of the i–th core ($i = 1, 2, \ldots, N_c$) as $\bar{T}_i(t)$ and $T_i(t)$ and with $w_i(t)$, $v_i(t)$ the corresponding process and measurement noise, the MIMO model representing the thermal dynamics of the CPU is described by the following equations

$$\begin{cases} \bar{T}_1(t) & = {\varphi_T^1}^T(t)\,\theta_A^1 + \varphi_P^T(t)\,\theta_B^1 + w_1(t) \\ T_1(t) & = \bar{T}_1(t) + v_1(t) \\ \bar{T}_2(t) & = {\varphi_T^2}^T(t)\,\theta_A^2 + \varphi_P^T(t)\,\theta_B^2 + w_2(t) \\ T_2(t) & = \bar{T}_2(t) + v_2(t) \\ & \vdots \\ \bar{T}_{N_c}(t) & = {\varphi_T^{N_c}}^T(t)\,\theta_A^{N_c} + \varphi_P^T(t)\,\theta_B^{N_c} + w_{N_c}(t) \\ T_{N_c}(t) & = \bar{T}_{N_c}(t) + v_{N_c}(t) \end{cases} \tag{25}$$

where $\varphi_T^i(t)$ is a vector of type (5) whose entries are samples of $\bar{T}_i(t)$, whereas the vectors θ_A^i, θ_B^i contains the coefficients of the i–th core MISO model. To estimate the CPU thermal model, the identification procedure previously described is performed for each core.

IV. Results

A. Test bed

For our experimentation, we implemented the framework from Fig. 1 on 40 of the 516 nodes of a cluster in a working production system (Galileo at CINECA [32]) as a case study. Each node is equipped with two 8-cores Intel Haswell CPUs (E5-2630 v3 @ 2.40GHz) and 128GB of DRAM. The nodes have been monitored over a period of 17 days of normal operation, and all metrics are acquired at a constant sampling time of 2s. On the monitored nodes and during the entire time period there were 117 active users and a total of 3612 jobs were submitted, with an average of 31 users and 90 jobs per node (note that each user can submit multiple jobs, each using several nodes). The power prediction algorithms are instead run on a separate service node (Intel Haswell E5-2670 v3 @ 2.30GHz, 24 cores, 128GB DRAM, 4 NVIDIA GeForce GTX 1080 Ti), where the Apache Spark environment and all processing utilities are installed.

We want again to stress out the fact that, unlike most of the previous literature on power models (for example, [33], [34], [35], [36], [37]) and our previous work on thermal models ([26], [24]), in this work we trained and applied our power and thermal prediction models in a production environment while the machine was fully operational and running user jobs, where each node has a different workload which can drastically change over time, and not on custom-defined workloads on single nodes.

B. Power model

For the training of the power model, we have selected the first 3 days of operation of all nodes, and these data have not been used further in the definition of the thermal model. A different model is trained for each node and package, to reflect the variability between the nodes. The regularization parameter λ is set to 0.001 as in [30], which has proven to be a robust value to maximize accuracy.

To obtain an assessment of the model's performance, we have calculated the absolute error between the estimated and measured total package powers for all remaining 14 days of operation. For each node and package, we then calculate the percentage of time instants where such error is below a certain threshold. Fig. 2 then shows a summary between all nodes and packages of the percentage of points where the power error is below two thresholds (3.23W and 9.68W, equivalent to the 1°C and 3°C errors we used in [30]). Each point on the x-axis is a different number of nodes considered both for training and testing, and for each of these and for both power thresholds they are shown, between all considered nodes and packages, the median number of points below the threshold (circles) together with the 25% and 75% percentiles (error bars). By comparing Fig. 2 with the results in [30], we note that the new model performs slightly worse than the old one, as expected because we use significantly less features and we are estimating also the individual cores contributions. However the additional error is not very significant, and anyways for more than 90% of the points the error in package power is below 10W, which corresponds to roughly 1W error per core (assuming uniform distribution of the power) and about 10% of the maximum power. As we will see in the following, these errors are indeed small and do not prevent us from obtaining very accurate thermal models.

C. Thermal model

For the identification and evaluation of the thermal models we have used the 14 days of measurements remaining from the power models training. We have then chosen to further subdivide this data in shorter time windows and, for each node and core, use each window to identify a different model. One of the reasons behind this choice is to be comparable with

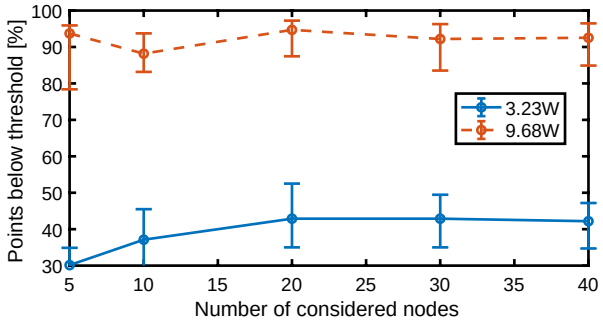


Fig. 2. Overview of the error in estimating the total package power. The two lines show, between all nodes and sockets, the median (circle) and 25th and 75th percentiles (error bar) of the percentage of points where the power error due to the estimation is lower than a threshold, as we did in [30].

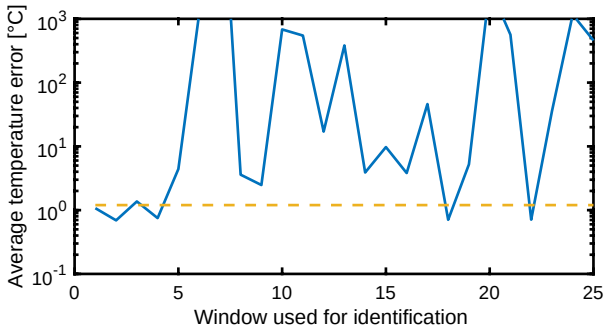
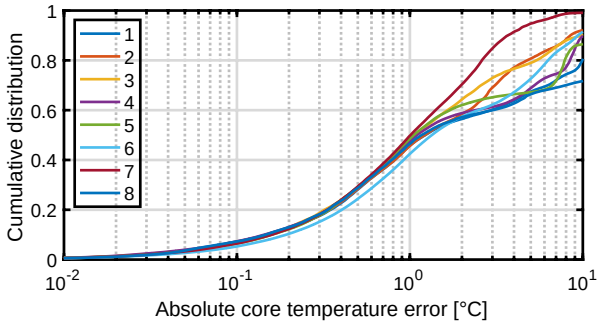
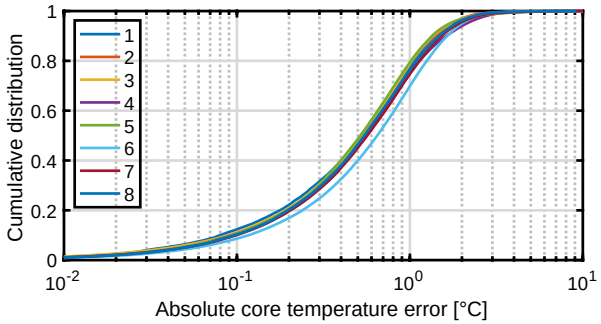


Fig. 3. Average of the temperature error across all cores for one package in the cluster, changing the window used for the thermal model identification. The dashed horizontal line corresponds to 1.2°C.



(a) window 16



(b) window 22

Fig. 4. Distribution of the temperature error for all cores of one package in the cluster, changing the window used for the thermal model identification.

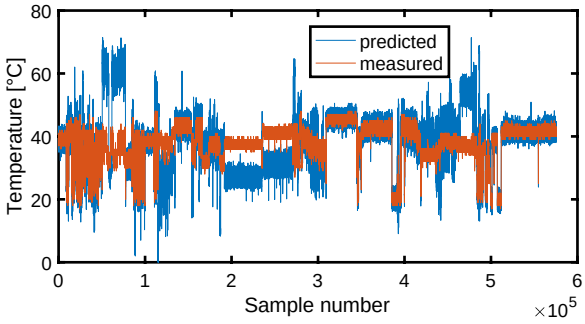
previous works [26] which have used relatively short windows (∼1.5 hours) for the identification. Another reason is that, since the workload can vary considerably during the entire time frame, by comparing for each core the models identified in various windows enables us to perform a thorough cross-validation of our results. On the other hand, the time windows cannot be too short, since the workloads have relatively large time constants and a short time window would not be able to easily capture all the possible states of the system. For all these reasons, we have chosen to divide the time frame in 25 windows, each ∼12 hours long and consisting of ∼22k samples (since we have a sampling time of 2 seconds).

For the model evaluation, instead, we have performed an extensive per core cross-validation by calculating the temperature prediction, using the Kalman filter, of each of the 25 core models (each one identified in a different window) on the entire 14 days of measurements. This procedure allows us to test if a model is able to retain a good performance in all operating conditions of the system, much beyond the duration of its identification dataset.

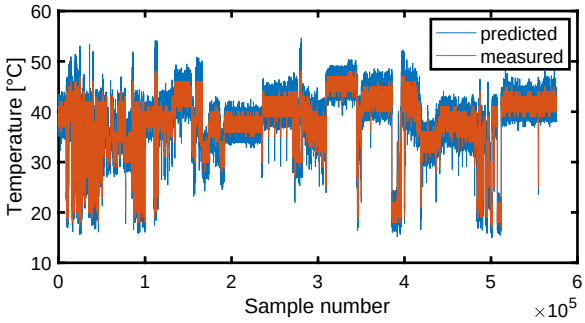
Going more into the details of the models, for each core of a node, a MISO ARX model with additive output noise of order $n = 2$ has been identified. The thermal dynamic of each core is thus characterized by two poles. The suitability of this choice has been proved in [26]. The estimated model on each window is then been used to compute the temperature predictions on all the remaining data by means of a Kalman filter, as discussed in Sec. III-B. We want again to stress the fact that, to assess the model's performance, we have performed a very strict cross-validation, since we test the model on an amount of data which is 25 times larger than the one used for identification. Fig. 3 reports the average of the error between the predicted and measured temperatures across all cores in one package for the whole available time frame of 14 days, changing the window used for the thermal model identification. It is evident that there is a great variability between the models identified in different windows. In particular, some of the models have excellent performance, with an average error often below 1.2°C (the dashed horizontal line), which is only slightly higher than the quantization step of 1°C.

In order to analyze the data in more detail, Fig. 4 shows the distribution of the error separately for all cores in the package choosing two windows for the identification, one with small error and one with high error. Moreover, Fig. 5 chooses one of the cores and shows also the time trace of both the predicted and the measured error in these situations. It is once again evident that, if the model is identified correctly, the performance is excellent, while, if the window chosen for the identification has bad properties, applying a system identification algorithm to real workload data can lead to poor estimations.

This finding is confirmed also by comparing the results presented in this work to what we obtained in [26], where



(a) window 16



(b) window 22

Fig. 5. Time traces of the measured and predicted temperatures for one core of one package in the cluster, changing the window used for the thermal model identification.

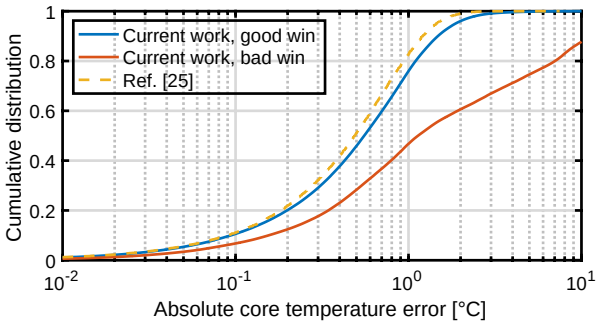


Fig. 6. Comparison between the distributions of the temperature error for all cores of one package in the cluster, using for the identification the same windows from Fig. 4 (solid lines), and of the temperature error reported in [26] (dashed line).

we derived a similar thermal model for one package of the same architecture considered here. In [26], however, we were able to execute an ad-hoc synthetic workload on the machine, specifically designed to be almost white, in order to rapidly excite all possible states of the system. Fig. 6 shows then a comparison between the results from Fig. 4 and from [26]. We note in the best case an almost negligible decrease in accuracy with respect to [26], which is an excellent result since in this work we do not employ ad-hoc benchmarks and our testing time of the model is much longer (14 days compared to 1.5 hours). On the other hand, once again a poor choice of the identification window leads to a very inaccurate model. It is therefore of utmost importance to devise a robust procedure for the a-priori evaluation of an identified model, or even for the choice of an appropriate window of data to use. This necessity becomes even more prominent when the models are used for control and anomaly detection.

V. Window Selection

As we have discussed, an appropriate choice for the data window to use for identification is of paramount importance. Such a choice can be made in two ways:

1) a-priori, before the identification, basing the choice upon the properties of solely inputs (powers) and outputs (temperatures);
2) using the results of the identification, looking for example at the poles, the condition number or the whiteness of the residuals.

Ideally, method 1) above should be preferable, as long as the algorithm for window selection is computationally very efficient. On the other hand, it is not guaranteed that a good window selection can be performed using only the results of the identification.

In order to evaluate the model's performance, we have derived a ground truth a-posteriori by using a Kalman filter on new data (see Sec. III-B). Such a method cannot be used for the window selection, both because it is highly computationally intensive and also since it might lead to delays or problems in a system that is meant for control or anomaly detection.

We have decided to perform the window selection on a per-core basis, i.e., every core in the package has its own ensemble of chosen windows, that can be (and likely in most cases will be) shared across most cores in one package. Note that the thermal model for one core uses as input also the estimated powers of all other cores and of the uncore in the package, so we expect that a selection on a per-core basis will still take into account the relative difference in activity between the cores.

In the following we test multiple algorithms on both methods for window selection. All the algorithms lie in the field of supervised machine learning, and therefore need to define labeled training and test sets. The goal of the algorithms is to perform a classification, calculating a likelihood of the windows to be good (windows whose identified model has good performance) or bad ones. Contrary to the identification of the thermal models, where we have derived a different model for each node and core to reflect their inherent variability, for the window selection phase we derive a single model using the data from all nodes and cores. Our choice is justified by the fact that the properties that define whether a time window is suitable for identifying a good thermal model should be general and not dependent on the particular characteristics of a single core.

The data used for training of all machine learning models has been handled in the same way. All 25 windows per package have been divided by core, resulting in 16k total windows. These have been randomly and independently reordered, and the 80% of those has been used for the training phase, the

remaining for the test phase. A window has been labelled as good if the average error of the Kalman filter on the entire data using the parameters identified in that window is below a threshold (1.2°C in our case), its standard deviation is also below a threshold (1.5°C) and the poles are stable, real and not too low (we set a threshold of 0.8 based on Fig. 8 and on the results in [26]). For better robustness of the algorithm, we have excluded from the training set the windows with medium temperature error (between 1.2°C and 1.5°C) and standard deviation (between 1.5°C and 2°C). All these thresholds have been calibrated by inspection of the data, and the threshold on the error standard deviation is necessary to exclude models where the error distribution deviates too much from a normal distribution (see for example Fig. 4).

The performance has been evaluated by classifying the windows from the test set, identifying the predicted good windows (i.e., the ones where the computed likelihood is above a given threshold) and showing the average temperature error that we would incur if we chose that window for the identification. Note that the misclassified good windows (i.e., the windows that are predicted of being good while in fact they are not) are the worst-case scenario, since in this case we would use for temperature control a bad model, while the misclassified bad windows are not as critical.

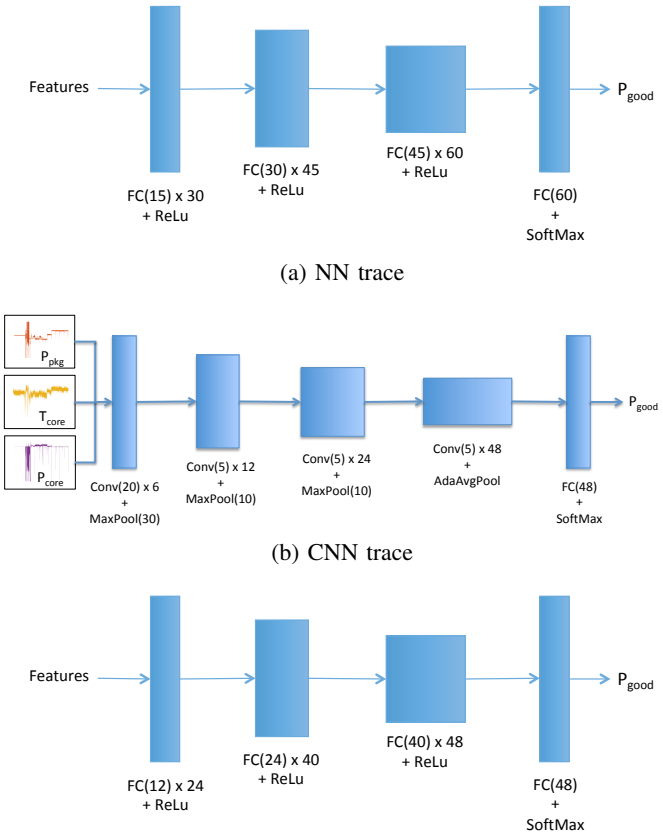
We should also note that the approach we follow, even though verified on a particular system with one thermal model in mind, is general, and therefore can be easily adapted to any situation where a model, with a real-valued time series as output, has to be identified on real data, where the inputs are an arbitrary number of real-valued time series. This aspect is further discussed in Remark 2 at the end of this section.

A. *Selection based on time traces*

For the selection based on time traces, we have then chosen to use as input the time traces of the cores' and the uncore partial powers (as estimated by the model from Sec. IV-B) and the core measured temperature. Since we have observed in the previous section that the identified model performs badly if the poles are very different between the cores, we have also decided to include the time trace of the measured package power, in order to obtain a measurement of similarity between all cores in a package.

We have evaluated three choices of algorithms, which will be presented in detail in the following part of this section. In summary, two of them rely on custom-defined features calculated on the time traces (namely, a classical SVM with RBF kernel [38] and a fully connected neural network [39]) and the last one is a 1D Convolutional Neural Network (CNN) [39] applied directly to the time traces. All neural network models have been implemented in PyTorch [40], levering the multi-GPU capability of our computing system.

As far as testing is concerned, the output of the neural networks is a number in the interval $[0, 1]$ which represents the likelihood of the window being good. If we set the decision threshold at 0.5, all windows will be classified either good or bad. We can also decide to use a different threshold, for example 0.8, and in this case only windows with likelihood $l > 0.8$ are considered good. As for the remaining windows, we have decided to leave all the ones with likelihood in the range $[0.2, 0.8]$ as unclassified. This allows us to define three classes: (i) good windows $l > 0.8$, to be used for identification, (ii) bad windows $l < 0.2$, to be discarded, (iii) unclassified windows $0.2 < l < 0.8$, for which we are not sure whether they can be used or not, and they can be a pool to choose from if no window gets classified as good.



(a) NN trace

(b) CNN trace

(c) NN identification

Fig. 7. Architecture of the Neural Networks used for window selection.

TABLE II
CHOSEN FEATURES FOR THE SVM AND NN ALGORITHMS USING THE TIME TRACES, WHERE THE SIGNAL x CAN STAND EITHER FOR PACKAGE POWER OR CORE'S TEMPERATURE OR PARTIAL POWER.

Feature definition	Description
$\mathrm{E}\left[\lvert\mathrm{corr}\,(x)\rvert\right]$	Average of the absolute value of the signal's auto-correlation with up to 100 samples delay
$\mathrm{E}\left[\bar{x}\right]$	Average of the low-passed signal
$\mathrm{std}\left[\bar{x}\right]$	Standard deviation of the low-passed signal
$\max\left[\bar{x}\right] - \min\left[\bar{x}\right]$	Maximum span of the low-passed signal
$\mathrm{E}\left[\lVert\mathrm{fft}\,(x)\rVert\right]$	Average of the norm of the signal's FFT

Classification based on custom features: As already outlined, we have used both a SVM algorithm with radial basis functions and a fully connected neural network. The architecture of the network is shown in Fig. 7a and it has been derived following the standard suggestions for neural networks [39]. It consists of four fully connected layers each

followed by a rectified nonlinearity (ReLu) and finally a softmax classification layer. The optimizer used for training is Adam, with learning rate 0.001 and weight decay 10^{-4} and using as loss function the binary cross-entropy. The training phase was stopped after 3000 iterations, where the training loss had converged to a value of 0.35.

Both algorithms use the same input, which consists in a pre-defined set of features on the power and temperature time traces. The features have been chosen in order to empirically reproduce the characteristics that we have observed in good windows. Some of the features are defined on a low-pass version of the traces (denoted as $\bar{x}$ for a signal x), where the window signal has been further divided into 20 subwindows and averaged. Tab. II reports a summary of the defined features.

Classification based on CNN: The architecture of the chosen CNN is shown in Fig. 7b, and it has again been derived following the standard suggestions for convolutional networks [39]. It is composed of four 1D convolutional layers, each one doubling the number of channels of its input, followed by max-pooling, and in the last layer an adaptive average pooling to bring down the dimension of the time trace to one followed by a softmax classification layer. In the training phase, a dropout layer was also inserted before the average pooling, with a drop probability of 0.5.
The optimizer used for training is again Adam, with learning rate 0.001 and weight decay 10^{-5} and using as loss function the binary cross-entropy. The training phase was stopped once the training loss had been below a threshold (0.18) for at least 5 iterations.

B. Selection based on identification results

In order to understand if the choice of a machine learning algorithm for the selection based on identification results is appropriate, we have started considering only simple metrics, i.e., the modulus of the maximum pole and the condition number of the matrix $\hat{R}$ for each core and window, where (see Sec. III-B):

$$\hat{R} = \begin{bmatrix} \hat{\Sigma}^T & \hat{\Sigma}_q^T \end{bmatrix}^T .$$

In fact, on the one hand, unstable models (where the poles magnitude is greater than 1) do not comply with the physics of the system and, on the other hand, we know from theoretical considerations that very high condition numbers are undesirable [14], [13]. As Fig. 8 shows, these metrics can give very good indications to classify between good and bad windows, however there is still a large number of outliers. Our analysis is compatible with the results presented in [18], where it is shown that the condition number itself is not enough to choose a window and the addition of other metrics is mandatory. However, in our case, due to the much greater amount of data with respect to [18], we have observed that no simple combination of features is able to deliver a good window selection. We have chosen therefore to resort to a machine learning algorithm, considering all the available metrics that we could derive from the identification algorithm.

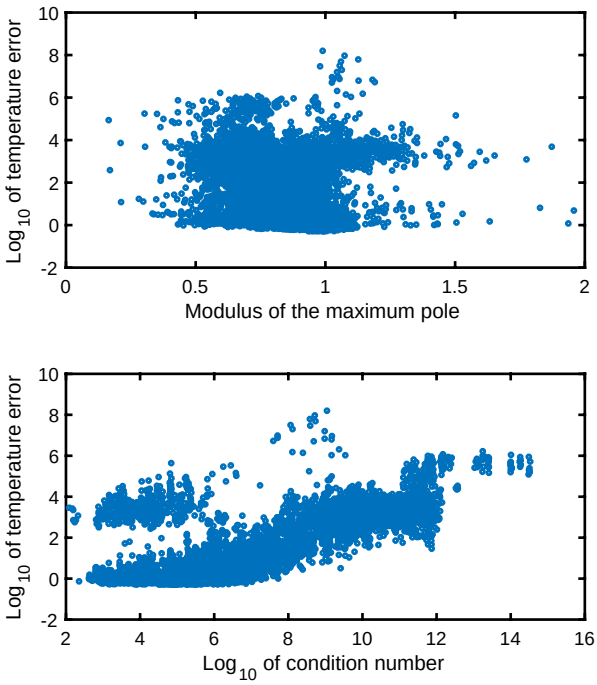
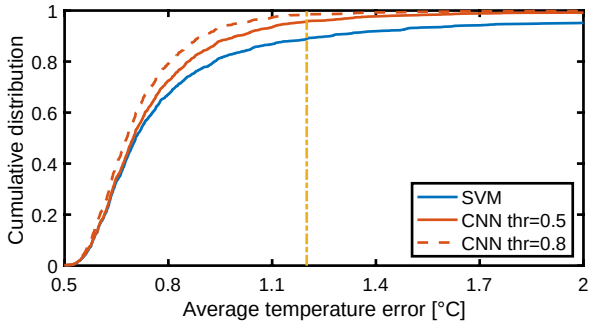


Fig. 8. Correlation between the condition number or the maximum pole modulus and the temperature error for all cores and windows. For the magnitude of the maximum poles, we have calculated an average of 0.95 ± 0.03 for the good windows and of 0.85 ± 0.17 for the bad ones.

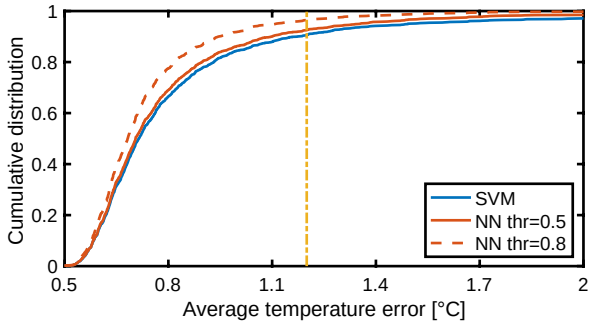
TABLE III
CHOSEN FEATURES FOR THE SVM AND NN ALGORITHMS USING THE RESULTS FROM THE MODEL IDENTIFICATION.

Feature definition	Description
$\mathrm{Re}\, p_i$	Real parts of the poles of the A matrix
$\mathrm{Im}\, p_i$	Imaginary parts of the poles of the A matrix
σ_w	Standard deviation of the equation error
σ_v	Standard deviation of the measurement noise
$\mathrm{corr}\,(res)$	First four autocorrelations of the identification residuals
$\log_{10}\left(\mathrm{cond}\left[\hat{R}\right]\right)$	Condition number of the matrix $\hat{R}$
$\log_{10}\left(\min \mathrm{svd}\left[\hat{R}\right]\right)$	Minimum singular value of the matrix $\hat{R}$

As already outlined, we have used again both a SVM algorithm with radial basis functions and a fully connected neural network. The architecture of the network is shown in Fig. 7c and it has been derived very similarly to the one in Fig. 7a. It consists of four fully connected layers each followed by a rectified nonlinearity (ReLu) and finally a softmax classification layer. The optimizer used for training is once again Adam, with learning rate 0.001 and weight decay 10^{-4} and using as loss function the binary cross-entropy. The training phase was stopped once the training loss had been below a threshold (0.15) for at least 5 iterations.
Both algorithms use the same input, which consists in a set of features derived from the results of the identification algorithm. These features are reported in Tab. III.



(a) Time traces



(b) Identification features

Fig. 9. Cumulative distributions (ECDFs) of the average temperature error for the windows chosen by the classification algorithm using either directly the time traces or the features coming from the identification, and varying the decision threshold for the neural networks. The chosen windows differ between the algorithms, so the ECDFs do not have the same number of points. The vertical dashed line is the 1.2°C threshold.

TABLE IV
CLASSIFICATION ACCURACY FOR BOTH ALGORITHMS, VARYING THE DECISION THRESHOLD FOR THE CNN. NOTE THAT ALL PERCENTAGES ARE NORMALISED BY THE NUMBER OF CLASSIFIED WINDOWS, EXCEPT FOR THE LAST LINE (UNCLASSIFIED WINDOWS), WHICH IS NORMALISED BY THE TOTAL NUMBER OF WINDOWS IN THE TEST SET.

Metric definition	Algorithm	Threshold	Value
Percentage of misclassified good windows	SVM trace	-	7.6%
	NN trace	0.5	10%
		0.8	3%
	CNN trace	0.5	4.6%
		0.8	2%
	SVM identification	-	7.8%
	NN identification	0.5	5%
		0.8	1.8%
Percentage of misclassified bad windows	SVM trace	-	6.8%
	NN trace	0.5	6.8%
		0.8	2.9%
	CNN trace	0.5	3.5%
		0.8	1.3%
	SVM identification	-	3.2%
	NN identification	0.5	4.2%
		0.8	1.3%
Percentage of correctly classified windows	SVM trace	-	86%
	NN trace	0.5	83%
		0.8	94%
	CNN trace	0.5	91%
		0.8	96%
	SVM identification	-	89%
	NN identification	0.5	90%
		0.8	96%
Percentage of unclassified windows (relative to the size of the test set)	SVM trace	-	0%
	NN trace	0.5	0%
		0.8	42%
	CNN trace	0.5	0%
		0.8	16%
	SVM identification	-	0%
	NN identification	0.5	0%
		0.8	18%

C. Classification results

Initially we concentrate on the study of the misclassified good windows, i.e., the cases where the algorithm classifies the window as being good while in reality it is not. Fig. 9a reports, for the SVM and CNN algorithms on the time traces and varying the decision threshold for the CNN, the cumulative distributions of the average temperature error on the entire data when the model has been identified in a window that is chosen by the algorithm as a good window. We immediately note that the CNN performs much better than the SVM. Moreover, if we raise the decision threshold to 0.8, the number of outliers is almost zero.

A similar situation can be observed in Fig. 9b, which reports the same results for the SVM and NN algorithms on the identification results. Also in this case the SVM provides the lowest accuracy, while the NN with decision threshold 0.8 gives excellent performance. Moreover, comparing Fig. 9a and Fig. 9b, we note that the CNN on time traces and the NN on identification results perform very similarly.

Finally, Tab. IV reports the percentages of correctly and incorrectly classified windows for all cases. It also shows the decrease in yield when the decision threshold is increased from 0.5 to 0.8. From the results reported in the table, we again conclude that the best performance is achieved with the CNN on time traces and the NN on identification results. Increasing the decision threshold to 0.8 allows much better performance, with the drawback of a relatively small decrease in yield.

Note that, even though the NN on identification results perform very similarly to the CNN on time traces, the former needs also to run the identification procedure before passing the results through the neural network, while the latter works directly on the raw signals.

Remark 2: As already mentioned, although in this paper we focus our attention on the MISO ARX model with additive output noise (ARX + noise) model (3)-(4), the proposed window selection approach can be applied to a large class of input-output models. For instance, the method can be easily adapted to more classical models like ARX and Ouput Error (OE), that have already been used for identifying the thermal dynamics of many-core systems on chip [8], [23]. More precisely:

– The selection algorithms based on time traces described in Subsection V-A are fully model-independent since the involved features are the signal properties reported in Table II or the signal themselves (as for the CNN). These algorithms can thus applied to ARX and OE models without any variation.

– The selection algorithms based on identification results described in Subsection V-B are not model independent, but most of the features reported in Table III are quite general.

TABLE V
ESTIMATED DISCRETE-TIME POLES p_1, p_2 FOR ALL CORES OF ONE PACKAGE.

	ARX + noise		ARX		OE	
core	p_1	p_2	p_1	p_2	p_1	p_2
1	0.961	0.020	0.877	−0.443	0.954	−0.463
2	0.958	0.245	0.876	−0.456	0.972	−0.823
3	0.952	0.192	0.865	−0.452	0.968	−0.038
4	0.955	0.163	0.861	−0.456	0.871	0.143
5	0.954	0.031	0.870	−0.437	0.968	−0.037
6	0.950	0.055	0.867	−0.442	−0.996	0.897
7	0.959	0.140	0.886	−0.449	0.973	−0.178
8	0.953	0.308	0.880	−0.448	0.966	−0.516

For example, all the considered types of models (ARX+noise, ARX, OE) lead to estimated poles, variance of the equation error and sequence of residuals so that the proposed algorithms can be easily applied to these models with small variations. Finally, it is worth to remember that ARX and OE models have not been considered in the paper as they do not lead to suitable results for thermal modelling when the measured temperatures are affected by significant additive noise, as discussed in [25], [24], [26]. This is confirmed by the results described in Table V, that reports the identified poles associated with ARX + noise, ARX and OE models for all the eight cores of one package. All models have been identified by using the input-output sequences of a good data window. From Table V, it is clear that only the poles associated with ARX + noise models are compliant with the physics of thermal systems, where only real positive poles can exist.

VI. CONCLUSIONS

In this work we have demonstrated the identification and application of a thermal model, suitable for power and thermal control and anomaly detection, to the nodes of an HPC cluster in production. We have shown that the performance of the model is excellent, with an average error on the temperature predicted using a Kalman filter very close to the quantization step. This result is even more relevant since the thermal models have been identified using real workloads on the nodes, and not ad-hoc excitations which would have required to put the production machine off-line.

In order to achieve the best model performance, it is crucial to accurately choose the data to use for the model identification. To this purpose, we have analyzed and compared a variety of approaches based on machine learning and deep learning techniques that can reliably choose the appropriate windows of data given real workloads. Our finding is that this choice of window is indeed a non-trivial problem, which only sophisticated deep learning algorithms can accurately address.

Our work paves the way to the application of thermal models on an HPC cluster in a scalable and efficient way, without requiring neither large computational overheads nor down-times of the machines for model calibration.

VII. ACKNOWLEDGEMENTS

This work was supported by the EU FETHPC project ANTAREX (g.a. 671623). The authors would like also to thank Andrea Borghesi from the University of Bologna for providing the numbers on the usage of the cluster and Francesco Beneventi from the University of Bologna for his support on the framework and on the data collection.

REFERENCES

[1] L. Gilly, "Data centre design standards and best practices for public research high performance computing centres," Ph.D. dissertation, uzh, 2016.

[2] J. Dongarra, "Visit to the national university for defense technology changsha, china," *Oak Ridge National Laboratory, Tech. Rep., June*, 2013.

[3] A. Moskovsky, E. Druzhinin, A. Shmelev, V. Mironov *et al.*, "Server level liquid cooling: Do higher system temperatures improve energy efficiency?" *Supercomput. Front. Innov.: Int. J.*, vol. 3, no. 1, pp. 67–74, Jan. 2016.

[4] A. Verma, P. Ahuja, and A. Neogi, "Power-aware dynamic placement of hpc applications," in *Proceedings of the 22Nd Annual International Conference on Supercomputing*, ser. ICS '08. New York, NY, USA: ACM, 2008, pp. 175–184.

[5] I. Rodero, H. Viswanathan, E. K. Lee, M. Gamell *et al.*, "Energy-efficient thermal-aware autonomic management of virtualized hpc cloud infrastructure," *Journal of Grid Computing*, vol. 10, no. 3, pp. 447–473, Sep 2012.

[6] F. Zanini, D. Atienza, C. N. Jones, L. Benini *et al.*, "Online thermal control methods for multiprocessor systems," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 1, p. 6, 2013.

[7] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini, "Cooling-aware node-level task allocation for next-generation green hpc systems," in *High Performance Computing & Simulation (HPCS), 2016 International Conference on*. IEEE, 2016, pp. 690–696.

[8] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini, "Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 170–183, 2013.

[9] A. Mutapcic, S. Boyd, S. Murali, D. Atienza *et al.*, "Processor speed control with thermal constraints," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, pp. 1994–2008, Sept 2009.

[10] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan *et al.*, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, May 2006.

[11] S. Reda, K. Dev, and A. Belouchrani, "Blind identification of thermal models and power sources from thermal measurements," *IEEE Sensors Journal*, vol. 18, no. 2, pp. 680–691, 2018.

[12] S. Pagani, J. Chen, M. Shafique, and J. Henkel, "MatEx: Efficient transient and peak temperature computation for compact thermal models," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2015, pp. 1515–1520.

[13] T. Soderstrom and P. Stoica, *System Identification*. Cambridge, UK: Prentice-Hall, 1989.

[14] L. Ljung, *System Identification – Theory for the User*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[15] N. Xie and H. Leung, "Blind identification of autoregressive system using chaos," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 9, pp. 1953–1964, Sept 2005.

[16] C.-A. Lin and Y.-S. Chen, "Blind identification of mimo channels using optimal periodic precoding," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 4, pp. 901–911, Apr 2007.

[17] S. A. Fattah, W.-P. Zhu, and M. O. Ahmad, "A novel technique for the identification of arma systems under very low levels of snr," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 7, pp. 1988–2001, Aug 2008.

[18] A. H. Ribeiro and L. A. Aguirre, "Selecting transients automatically for the identification of models for an oil well," *IFAC-PapersOnLine*, vol. 48, no. 6, pp. 154–158, 2015.

[19] Y. Yang, Z. Gu, C. Zhu, R. P. Dick *et al.*, "Isac: Integrated space-and-time-adaptive chip-package thermal analysis," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 26, no. 1, pp. 86–99, 2007.

[20] R. Cochran and S. Reda, "Consistent runtime thermal prediction and control through workload phase detection," in *Design Automation Conference*, June 2010, pp. 62–67.

[21] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Utilizing predictors for efficient thermal management in multiprocessor socs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1503–1516, Oct 2009.

[22] D. C. Juan, H. Zhou, D. Marculescu, and X. Li, "A learning-based autoregressive model for fast transient thermal analysis of chip-multiprocessors," in *17th Asia and South Pacific Design Automation Conference*, Jan 2012, pp. 597–602.

[23] F. Beneventi, A. Bartolini, A. Tilli, and L. Benini, "An effective gray-box identification procedure for multicore thermal modeling," *IEEE Transactions on Computers*, vol. 63, no. 5, pp. 1097–1110, 2014.

[24] R. Diversi, A. Tilli, A. Bartolini, F. Beneventi *et al.*, "Bias-compensated least squares identification of distributed thermal models for many-core systems-on-chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 9, pp. 2663–2676, Sept 2014.

[25] R. Diversi, A. Bartolini, A. Tilli, F. Beneventi *et al.*, "SCC thermal model identification via advanced bias-compensated least-squares," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 230–235.

[26] R. Diversi, A. Bartolini, F. Beneventi, and L. Benini, "Thermal model identification of supercomputing nodes in production environment," in *Proc. of the 42nd Annual Conference of IEEE Industrial Electronics Society (IECON 2016)*, 2016, pp. 4838–4844.

[27] S. Sharifi and T. S. Rosing, "Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 10, pp. 1586–1599, Oct 2010.

[28] R. Diversi, R. Guidorzi, and U. Soverini, "Identification of ARX and ARARX models in the presence of input and output noises," *European Journal of Control*, vol. 16, no. 3, pp. 242–255, 2010.

[29] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini, "Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 1038–1043.

[30] F. Pittino, F. Beneventi, A. Bartolini, and L. Benini, "A scalable framework for online power modelling of high-performance computing nodes in production," in *Proceedings of The 2018 International Conference on High Performance Computing and Simulation (HPCS)*, 2018.

[31] H. Trevor, T. Robert, and F. JH, "The elements of statistical learning: data mining, inference, and prediction," 2009.

[32] "Galileo, the italian tier-1 cluster for industrial and public research," CINECA, http://www.hpc.cineca.it/hardware/galileo, Tech. Rep.

[33] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," *SIGARCH Comput. Archit. News*, vol. 34, no. 5, pp. 185–194, Oct. 2006.

[34] G. T. Chetsa, L. Lefèvre, J.-M. Pierson, P. Stolf *et al.*, "Exploiting performance counters to predict and improve energy performance of hpc systems," *Future Generation Computer Systems*, vol. 36, no. Supplement C, pp. 287 – 298, 2014.

[35] M. Witkowski, A. Oleksiak, T. Piontek, and J. Weglarz, "Practical power consumption estimation for real life hpc applications," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 208–217, 2013.

[36] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das *et al.*, "Accurate and stable run-time power modeling for mobile and embedded cpus," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, Jan 2017.

[37] M. Chadha, T. Ilsche, M. Bielert, and W. E. Nagel, "A statistical approach to power estimation for x86 processors," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2017, pp. 1012–1019.

[38] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[40] A. Paszke, S. Gross, S. Chintala, G. Chanan *et al.*, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.