Modeling and Evaluation of Application-Aware Dynamic Thermal Control in HPC Nodes

Daniele Cesarini, Andrea Bartolini, Luca Benini

Modeling and Evaluation of Application-Aware Dynamic Thermal Control in HPC Nodes

Daniele Cesarini[†], Andrea Bartolini[†], and Luca Benini[†‡]

[†]DEI, University of Bologna, 40136 Bologna, Italy
[‡] IIS, Swiss Federal Institute of Technology, 8092 Zurich, Switzerland
{daniele.cesarini,a.bartolini}@unibo.it
lbenini@iis.ee.ethz.ch

Abstract. As side effects of the end of Dennard's scaling, power and thermal technological walls stand in front of the evolution of supercomputers towards the exaflops era. Energy and temperature walls are big challenges to face for assuring a constant grow of performance in future. New generation architectures for HPC systems implement HW and SW components to address energy and thermal issues for increasing power and efficient computing in scientific workload. In thermal-bound HPC machines, workload-aware runtimes can leverage hardware knobs to guarantee the best operating point in term of performance and power saving without violating thermal constraints.
In this paper, we present an integer-linear programming formulation for job mapping and frequency selection for thermal-bound HPC nodes. We use a fast solver and workload traces extracted from a real supercomputer to test our methodology. Our runtime is integrated into the MPI library, and it is capable of assigning high-performance cores to performance-critical processes. Critical processes are identified at execution time through a mathematical formulation, which relies on the characterization of the application workload and on the global synchronization barriers. We demonstrate that by combining long and short horizon predictions with information on the critical processes retrieved from the programming model, we can drastically improve the performance of the target application w.r.t. state-of-the-art DTM solutions.

Keywords: HPC, thermal model, power model, workload model, energy saving, thermal constraint, DTM, MPI, runtime, ILP, quantum espresso

1 Introduction

Driven by Moore's law, the trend in increasing performance of CPUs has seen as collateral effects the rapid increase of power consumption and power density that in turn have limited the achievable performance and caused an acceleration of chip aging. Cooling and heat generation are rapidly becoming the key limiters for high performance processors, especially for HPC and data centres which typically host clusters of thousands of high-performance processors.

In High-Performance Computing (HPC) nodes the maximum safe temperature at which the processing elements can run depends on the cooling technologies. For instance, Intel Xeon E5-26XX v3 server class processors have specifications on the maximum silicon temperature ranging from 69^oC to 101^oC according to the package thermal resistance (cost) and the nominal thermal design power (TDP)[1]. To enforce these safe working temperatures, HPC nodes use active-cooling solutions which translate in additional power consumption.

Dynamic thermal management (DTM) has been studied to limit the cooling effort by controlling and reducing the heat generation. This is achieved monitoring the HW thermal sensors and the application workload reacting on CPU dynamic voltage and frequency scaling (DVFS) states. New generation multicore CPUs, which are used in HPC systems, can apply a different voltage and frequency to each core independently [17]. This opens new scenarios for fine-grain DVFS control in DTM solution. Operating Systems use feedback loops between sensors and DVFS states of each core to scale down frequency states to avoid thermal hazards. Indeed, several solutions explore proactive techniques for DTM strategies to improve performance in thermal-bound systems [2, 12, 20, 21]. DTM strategies take advantages of the heterogeneity in the thermal dissipation of cores, which is related to chip and board design, and manufacture, to maximize the performance. However, these approachs often results in a performance unbalance between the cores. Coldest cores run faster than hottest cores.

Applications in HPC take advantage of the parallel architecture to speed up the execution of large scale simulations and workloads. The message passing interface (MPI) programming model is the de facto standard in HPC programs for splitting the workload in tasks that execute in parallel in the HPC machine. During the execution of an MPI-based application, the tasks alternate phases of computation on local data with phases of data exchange and synchronization. A critical design parameter in MPI applications is the balancing of the workload between the tasks, and the minimization of the waiting time for each tasks in the synchronization points [19, 25]. Critical tasks, in a specific code segment, are the ones which carry on the most workload and arrive late at a synchronization point. In practice, they limit the application speed in the specific code segment. Application developers and users in HPC systems parameterize the application configuration to balance the workload between the tasks. This intended to limit the slowdown induced by critical tasks. As previously seen, DTM techniques can create local unbalance between cores to maximize processor's throughput. This can be significantly detrimental for application performance as it may slow down critical tasks in parallel applications. However, this can be translated into an advantage for DTM strategies. Indeed, critical tasks could be assigned to the coldest cores at the application start-up phase and could reward critical tasks slowing down the less critical ones. In this chapter we focus on this problem, creating an application-aware dynamic thermal management runtime for HPC processors.

[1] Intel Xeon®Processor E5 v3 Family Thermal Guide

We present a DTM solution for HPC systems to increase performance of thermal-bound HPC systems exploiting thermal capacitances. We first propose a novel thermal model description derived from state-space representation of a real HPC node. We study the sensitivity of the application walltime to frequency changes in the communication phases. Our exploration reveals that the penalty in the application walltime caused by the frequency reduction decreases proportionally with time spent in the MPI library. After that, we focus our work on the workload distribution of of a real supercomputer's application. We identify the presence of critical tasks, which will be prioritized w.r.t. the other MPI tasks. Secondly, we present two novel ILP formulations for thermal-aware task mapping and frequency selection for large parallel heterogeneous many-core. We propose a task criticality model which relies on a mathematical formulation; this model considers application workload and synchronization constraints to reduce the slack times. We use the thermal characteristics of the compute node to formulate both the ILP problems. In this context we explore the impact of the time horizons at which future temperatures are predicted in the efficacy of the proposed DTM solution. We then show that our optimization models can significantly improve the performance in supercomputer environments without inducing significant overhead in time-to-solution.

This chapter is an extension of the conference paper [8]. We extend the previous work by: 1) A detailed analysis of the power consumed by the main components (core and uncore) in supercomputer's node under different DVFS operational states. 2) A detailed analysis of the workload distribution in our target HPC application among the MPI tasks , and of the task criticality in periods of tens seconds. 3) Proposing new module that we implement the proposed thermal controller called "Task Criticality Generator". This module is responsible to profile, calculate the MPI activity using a new proposed mathematical formulation, and identify the task criticality of each task in each time period. 4) Evaluating the performance trade-offs given by the "Task Criticality Generator".

The chapter is organized as follows. Section 2, presents state-of-the-art works on thermal management. Section 3 characterizes thermal proprieties of a scientific computing node and reports a study on workload unbalance in a target scientific application. Section 4 shows our DTM solution for thermal-aware mapping and control based on ILP formulation and task criticality generator. Section 5 reports experimental results. While Section 6 describes the conclusions of this work.

2 Related Work

Several works were focused on thermal-aware workload allocation based on DVFS strategies. Those techniques include: (i) on-line optimization policies [11, 4, 10, 32], which are based on predictive models and embedded sensors to read the current temperatures on the system; (ii) scheduling approaches for off-line allocation [26, 24] which rely on simplified thermal models, usually embedded in the target platform [4] or simulating chip temperature [31].

Today's thermal management works range from mobile to large scale parallel machine, like supercomputer and HPC systems. Xie et. al. [30] show that mobile systems are thermally constraint. Interestingly, thermal constraints, come from user experience and not from silicon limits. Conficoni et al. [9] show that the power cost of HPC cooling depends on several factors, for instance IT power consumption, the cooling control policies, and the ambient temperature. On the other hand, the power consumption is intertwined with workload execution and computation phases [23, 7], which can produce high thermal heterogeneity between nodes and CPUs. For this reason, over-provisioning cooling design can causes severe inefficiencies.

Wang et. al. [29] show that fan power can account for up to 23% of typical server power and scales super-linearly with node utilization. Authors in [6] extract a predictive thermal model directly from the multicore device correlating power, performance and thermal sensors implemented in HW. They show that the thermal evolution of a multicore device can be modeled with a linear state-space representation. the leakage-power dependency from temperature can be modeled as a perturbation of the state matrix of the thermal model. Due to different materials present in the heat dissipation path, the thermal transient is multi-modal with time constants that vary from ms to tens of seconds. Beneventi et al. [5] shows in an Intel based computing nodes with 36 physical cores, that the increased number of processors integrated in the same die generates significant thermal gradients and this thermal heterogeneity can be exploited by thermal/aware MPI task allocation to reduce the fan speed and power without impacting the application performance.

To find a close form solution of the fast mapping problem under thermal constraint, Hanumaiah [18] assumes the absence of direct thermal exchange from the hot to the cold cores of the same die. Mutapcic et al. [24] formulate a convex optimization problem to control the speed of the processor, which is subject to environment thermal constraints. They solve it with a specialized algorithm. However, their optimization algorithm does not cover the case of an higher number of cores than the number of tasks (some cores remain in idle state).

Predictive controls are often based on thermal and optimization models which can guarantee a safe-working condition applying performance constraints to the systems. Rudi et al. [28] have developed an Integer Linear Programing (ILP) model for task allocation and frequency selection to avoid thermal hazards in many-core architectures. This thermal control is able to leverage on the idleness of the cores when tasks are less than the number of available cores allocating tasks on the coldest cores and leaving hottest ones in idle states. The limit of [28] is the task allocation, which is not handled by the systems.

There are even significant works on energy-aware MPI library. Rountree et al. [27] use DVFS mechanism to reduce the frequency when there are no critical tasks running on the CPU. Adagio is not only one that use predictions to improve energy efficiency with DVFS techniques [14, 15, 22]. Instead, Eastep et al. [13] improve performance in power-constraint system balancing node's power budget

to speed up critical tasks. However, these solutions do not consider thermal constraint systems where CPU performance are limited to respect the safe-working temperature.

3 Workload and Thermal Modelling of HPC

Dynamic thermal management policies aim to reduce the cooling effort and power by adapting the processing element's performance to ensure a safe working temperature. In this section, we first introduce the nomenclature and the thermal properties of HPC nodes with direct measurements. Then, we extract from real scientific parallel workload a model linking the performance knob to the real performance of the final application. Finally, we analyze how the application workload is distributed among all the cores.

We took as a target machine an HPC system based on an IBM NeXtScale cluster. Each node of the cluster is equipped with two Intel Haswell E5-2630 v3 CPUs, with 8 cores with 2.4 GHz clock speed and 85W Thermal Design Power (TDP, [17]). This supercomputer is ranked in the Top500 supercomputer list [1].

3.1 Thermal Model

We focus our attention on a single node of the cluster as the rack is constructed by replication of the same node. To understand the thermal properties of a computing node, we have executed three main stress tests on which we have: (i) Kept the system in idle and measured the total power and the temperature for each core after ten minutes; (ii) We then have executed a stressmark[2] in sequence on each core of each socket in the node, leaving idle the remaining ones. We maintained the workload constant for ten minutes and measured the power consumption and the temperature, we used this test to extract the maximum steady state temperature gradient. Finally, (iii) we have simultaneously executed the stressmark for ten minutes in all the cores of the node and measured the temperature and the power consumption. In all the previous tests the temperature and power values are measured using an infrastructure similar to the one presented in [3], the Turbo mode was disabled to avoid power consumption to workload dependency. The results of our analysis are reported in table 1.

As we will see in the experimental results section, we used the extracted characteristics to create a thermal model using a distributed RC approach [4], with one tuned RC per core to have similar thermal characteristics as the measured ones.

3.2 Power Model

To model the impact of DVFS states on the target system, we have re-executed the stressmark in each core while scaling down the frequency for each core in all

[2] cpuburn stressmark by Robert Redelmeier: it is a single-threaded application which takes advantage of the superscalar architecture to load the CPU

Table 1. Thermal Model

AVG temperature - Idle cores	15.93^oC
AVG temperature - Active cores	33.39^oC
Gradient - Idle cores	4.47^oC
Gradient - Active cores	4.79^oC
Gradient - Active core vs idle cores	8.05^oC
Stady-state time	$120sec$

the available speed steps. We maintained each configuration for ten minutes and we measured the power consumed by each CPU. We collected these measurements in a lookup-tables (LUTs), one for each CPU. We then used the LUTs to compute the power dissipated by each CPU on each available frequency. We measured a total power of 17.86 W when all cores in a computing node are idle. The total power raises to 92.44 W when all the cores are active. We then extracted the power consumed by each core at each DVFS level with an average standard deviation in between cores of 0.1 W. The average uncore region of the CPUs contribute for 11.84 W and 17.85 W respectively when idle or active. The Figure 1 shows the average power consumption for each core of the system at all available frequency levels.

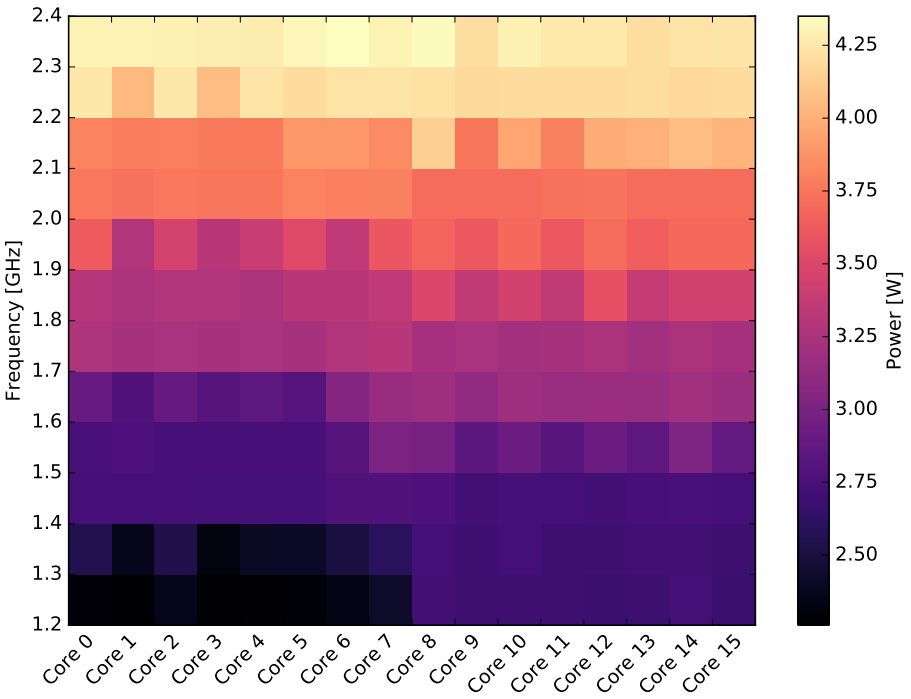

Fig. 1. Average power consumption of cores at all available frequency levels.

3.3 Workload Model

A HPC application can be seen as the composition of several tasks executed in a distributed environment, interconnected with a low-latency high-bandwidth network. HPC communications happen by sending explicit messages through a standard MPI programming model which takes advantage of the high-performance interconnect sub-system. Usually, tasks are composed by computational intensive phases on independent data segments interrupted by synchronization points and communications. This characteristic impacts the sensitivity of the application to each core's performance as computational imbalance can lead to longer synchronization phases.

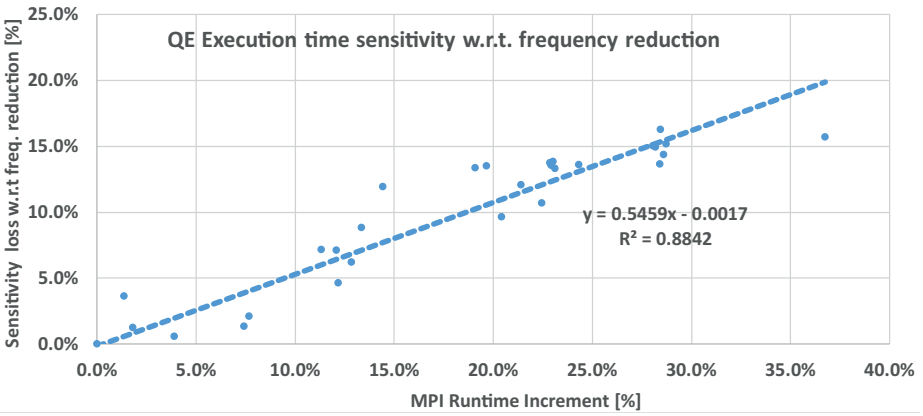

Fig. 2. Sensitivity loss w.r.t the reduction of frequency compared with the increment of the time spent into MPI library

As support to this statement, in this work we use as benchmark Quantum ESPRESSO (QE) [16], which is a real application widely used from the scientific community in high-end supercomputers. Moreover, QE main computational kernels include dense parallel linear algebra and 3D parallel FFT, which are both relevant in many HPC applications. In our test we use a Car-Parrinello (CP) simulation, which prepares an initial configuration of a thermally disordered crystal of a chemical element by randomly displacing the atoms from their ideal crystalline positions. This simulation consists of a number of tests that have to be executed in the correct order.

In the following experiment, we have explored how the different ratio of active code and MPI library for each QE task changes the impact of frequency scaling on the overall application execution time. We computed QE-CP on two computing nodes with 32 MPI tasks to increase the number of results respect

to a run on a single node. We run QE-CP 32 times. At each run we configured sequentially one core of the 32 at minimum frequency while the other are maintained at the maximum. We compared it with the run in which all the cores are at the nominal frequency. We then correlated the overall QE-CP slow down and the MPI percentage of the slowed down task. Figure 2 shows that the impact of frequency reduction increases with percentage of MPI library present in each task. This result is in line with what was shown in [27]. We can use it for extracting on-line the sensitivity to frequency for each MPI task. In this work, we take advantage of this information to address energy saving at execution time.

3.4 Workload Distribution

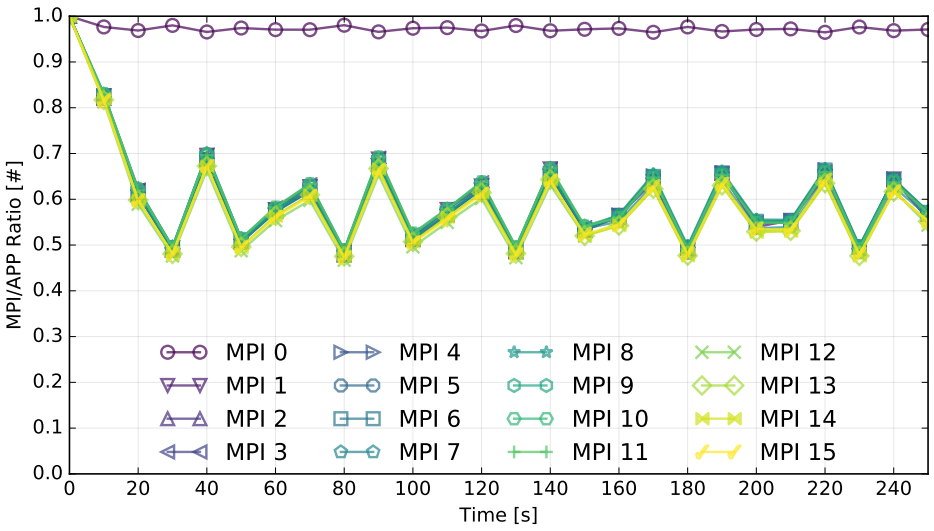

Fig. 3. Ratio of the time spent in application phases and MPI phases for each core and every 10 seconds.

While figure 2 shows the workload unbalance for the entire application run, it does not show how this unbalance is distributed in time -at a finer granularity-. In this section we explore how the workload is spread among all the MPI tasks and in time. We computed QE-CP on a single compute node with 16 MPI tasks. For each MPI task, we extract the time spent in the application and we compare it with the time spent in the MPI library. Every 10 seconds, we calculate the ratio between application time and MPI time, we plot this result in figure 3. We can see that the MPI task 0 spends more time in the application with respects to the others. In our benchmark, the core that slows down the application execution mostly is the core that runs the MPI task 0. If we slow down this core, we will have the highest penalty in the total execution time.

In the next section, we will see how this information can be extracted and considered in the thermal management problem.

4 HPC Optimal Thermal control

In this section, we present a Dynamic and Thermal Management (DTM) ILP formulation, namely the Optimal Thermal Controller (OTC), which matches all the requirements of HPC systems and proactive thermal control: (i) limiting the future temperature of all the cores below a critical threshold by selecting the proper frequency for each core; (ii) maximizing the application performance (frequency of all the cores); (iii) identifying cores that host critical tasks to promote their performance; (iv) slowing down the cores' frequency during communication.

As shown in Figure 4, the OTC operates at node level and it is composed of two main components: the thermal-aware task mapper and controller and the energy-aware MPI wrapper.

The thermal-aware task mapper and controller (TMC) is triggered: (a) after the job scheduler has deployed the parallel application on the reserved portion of the HPC machine for its execution; (b) periodically, with period T_s, and (c) at the start/end of every MPI call. At scheduling point (a) the TMC specifies the task to core mapping which will be maintained until the application completion. Clearly, if a critical task is mapped to a thermally inefficient core this will more likely cause a severe degradation of the final application performance. To capture the task criticality, we use a task criticality generation module, which intercepts every MPI call and extracts the time spent in both application and MPI library. At every scheduling point, this runtime uses a mathematical formulation based on the timestamps of the MPI calls to identify the criticality level (later named task criticality) for each task, as will be described in 4.1. At scheduling point (b), the TMC selects the optimal frequencies to be applied to the different cores for the following interval (to maintain the future cores' temperature below a safe threshold). Our OTC solution solves the scheduling points (a) and (b) with an ILP formulation and custom solver strategies as described in 4.2 and 4.3.

The energy-aware MPI wrapper (EAW) is event-driven and acts as a bridge between the MPI synchronization primitives and the core's frequency selection. This programming model interface is reactive and reduces the core's frequency when the MPI library is busy waiting. When the execution flow returns to the application code, the frequency is restored to the one selected by the Thermal Controller.

4.1 Task Criticality Generator

The per-task criticality level is calculated based on the time spent by the task in the application and waiting in the global synchronization points for each time interval. It is not sufficient to consider only the total time spent in the application during the last interval to compute a criticality level. We need to consider each

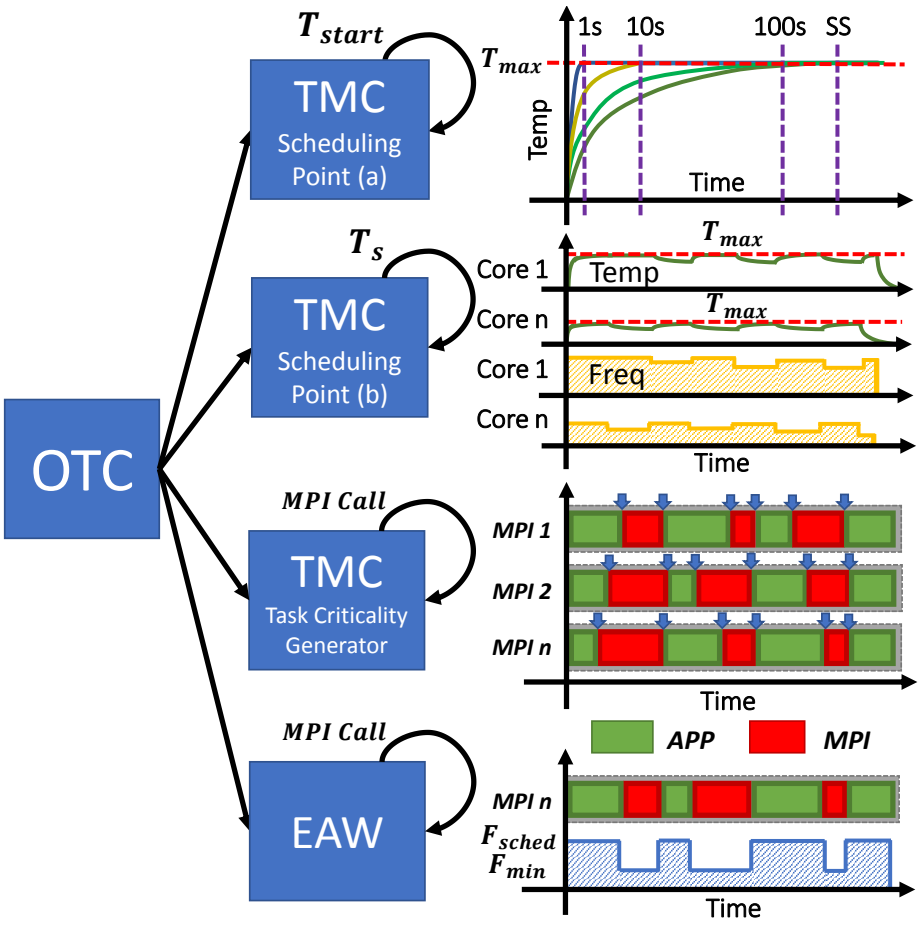

Fig. 4. Optimal thermal controller at node level

global synchronization point independently and for each of them compute the waiting time of each task.

We use a mathematical model to extract the per-task criticality level between two global synchronization points and we calculate the criticality of each task for all the global synchronization points in an interval. We define the criticality level for each task in this interval time as the average of the criticality levels weighted by the time which lasts between each pair of global synchronization points.

Figure 5 shows a general HPC application section enclosed by two global synchronization points where all the MPI tasks are involved. Every time that a MPI task encounters a global synchronization point, it must wait all other tasks to continue its execution. For each task, we identify three major time points which we base our model on. These are T_l, T_s, and T_e which represent the exit

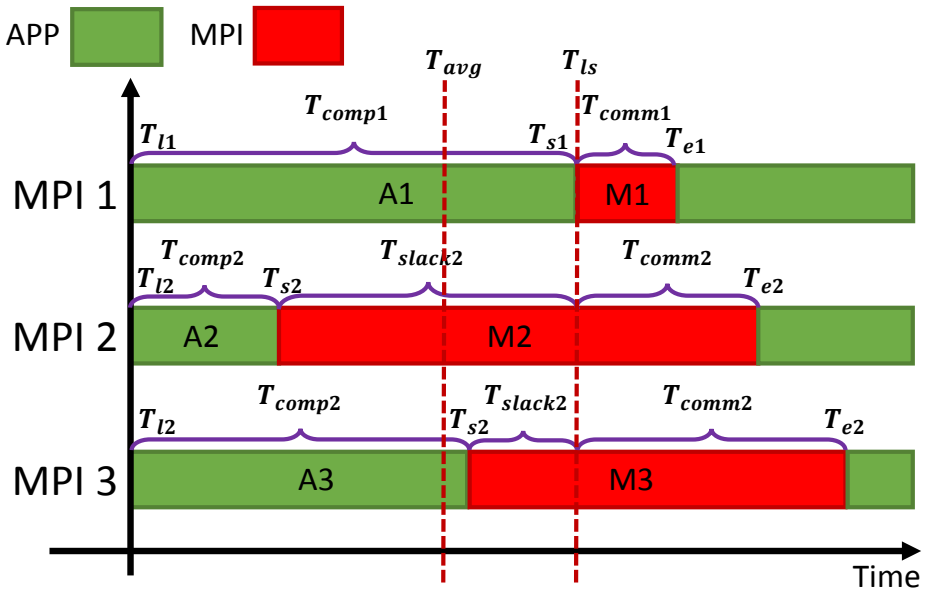

Fig. 5. General HPC application section with our naming convention for the mathematical model to calculate the criticality for each MPI task.

time of the last MPI call, the start time of the current MPI call, and the exit time of the current MPI call respectively. We use $[i]$ as the index to identify the MPI task id.

$$T_{ls} = MAX(T_{s[i]}) \tag{1}$$
$$T_{comp[i]} = T_{s[i]} - T_{l[i]} \tag{2}$$
$$T_{slack[i]} = T_{ls} - T_{s[i]} \tag{3}$$
$$T_{comm[i]} = T_{e[i]} - T_{ls} \tag{4}$$
$$T_{avg} = AVG(T_{s[i]}) \tag{5}$$
$$\delta_i = \frac{T_{comp[i]}}{T_{avg} - T_{l[i]}} = \frac{T_{s[i]} - T_{l[1]}}{T_{avg} - T_{l[i]}} \tag{6}$$

The last task that enters the global synchronization point unlocks all the waiting tasks which can now continue their execution. T_{ls} in eq. (1), identifies the time at which the last task enters in the synchronization point. For each application section and for each task $[i]$ we define as computation time $T_{comp[i]}$ in eq. (2) the time spent in the application code and MPI time the time spent in the MPI library. The latter is composed by two factors: (i) $T_{slack[i]}$ in eq. (3), which represents the time that a task spends in the MPI library waiting the last task reaching the synchronization point, (ii) $T_{comm[i]}$ in eq. (4), which identifies the time spent to exchange data. T_{avg} in eq. (5) is the average of all the

$T_{comp[i]}$. We compute the task criticality level δ_i in eq. (6) as the ratio between the $T_{comp[i]}$ and the T_{avg}. This metrics is proportional to the unbalance between the tasks in each application section.

4.2 The First Step Problem - FSP

This optimization problem is solved during the initialization of the application. Its purpose is to allocate the application tasks on the available cores and selecting for each of them the maximum frequency which meets the thermal constraint T_{max} in the prediction interval (PI_{FSP}). As we will see in the experimental results, the prediction interval (i.e. the time horizon) plays an important role. Indeed, if it is too short, the TMC cannot predict the impact of a task allocation on long term core's temperature as its effect is hidden by the thermal capacitance, making the problem trivial. On the contrary if the time horizon is too long the TMC cannot take advantage of the thermal capacitance for sustaining short time power burst.

In addition, not all tasks have the same criticality. This is captured by the optimization model which maximizes the frequency of the highest critical task penalizing the frequencies of other ones in case a thermal limit is reached. The optimization model considers K tasks to be assigned to N cores where the number of tasks is lower or equal to the cores i.e., $K \leq N$. Each core can be configured with a frequency in a set of M level of frequencies. The Objective Function ($O.F.$) maximizes the sum of frequencies of all active cores γ_{jf} weighted by the criticality δ_i of the task assigned on that core. To model the problem, we use two sets of binary decision variables:

$$x_{jf}^{i} = \begin{cases} 1 & \text{if core } j(j = 1, \ldots, N) \text{ works at frequency} \\ & f(f = 1, \ldots, M) \text{ executing task } i(i = 1, \ldots, K) \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

$$y_j = \begin{cases} 1 & \text{if core } j(j = 1, \ldots, N) \text{ is idle,} \\ 0 & \text{otherwise, i.e., if it is working.} \end{cases} \tag{8}$$

We can formulate the following ILP model with three constraints to model the assignments and the thermal bounds:

$$O.F. = max \sum_{i=1}^{K} \sum_{f=1}^{M} \sum_{j=1}^{N} \delta_i \gamma_{jf} x_{jf}^{i} \tag{9}$$

$$\sum_{j=1}^{N} \sum_{f=1}^{M} x_{jf}^{i} = 1 \tag{10}$$
$$(i = 1, \ldots, K)$$

$$\sum_{i=1}^{K}\sum_{f=1}^{M} x_{jf}^{i} + y_j = 1 \tag{11}$$
$$(j = 1, \ldots, N)$$

$$\sum_{j=1}^{N} GS_{jl}\left(\boldsymbol{p}_j y_j + \sum_{i=1}^{K}\sum_{f=1}^{M} p_{jf} x_{jf}^{i}\right) + T_l^0 + T^a \leq T_{MAX} \tag{12}$$
$$(l = 1, \ldots, N)$$

The constraint (10) specifies that a task must be assigned only on a single core, which works at a given frequency. In addition, it specifies that all the N tasks must be assigned. Constraint (11) is needed to determine the y decision variables which represent the idle cores. These variables are used in constraint (12) in case there are less tasks than cores i.e., $K \leq M$n. Finally, constraints (12) guarantee that the temperature of each core does not exceed T_{max} over the prediction interval (PI_{FSP}). In the last constraint (12), GS is a gain matrix with dimension $N \times N$. This matrix is used to calculate the increment of temperature of all the cores when a core is subjected to a constant power input for PI_{FSP} seconds . T_0^l represents the dependency of the future temperature(@ PI_{FSP}) from the current core's temperature. These values can be derived from a state-space thermal model as described by [28]. T_a is the ambient temperature. When tasks are less than cores the decision variable y_i is used in conjunction with the vector of idle powers $\bar{p}$, to add the idle power components.

4.3 The i-th Step Problem - ISP

After the tasks have been assigned to the cores in the FSP the TMC has to periodically solve, at a finer time scale, the assignment problem of frequencies to cores only. The ISP has the same objective function as FSP 4.2 as well as the same thermal model formulation. However the prediction interval for the ISP (PI_{ISP}) can be generally different from the FSP.

Differently from the previous case, the model considers only active cores (T) because the thermal constraints cannot be broken by an idle core. This reduces the overall complexity. As tasks have been already allocated in FPS in this model, tasks and core do not need separate variables, thus a criticality is referred to a core.

$$x_{rf} = \begin{cases} 1 & \text{if core } r(r = 1, \ldots, T) \text{ works at frequency} \\ & f(f = 1, \ldots, M), \\ 0 & \text{otherwise.} \end{cases}$$

The ISP model has fewer constraints than FSP due the lower number of variables.

$$O.F. = max \sum_{a \in A} \sum_{f=1}^{M} \delta_a \gamma_{af} x_{af} \tag{13}$$

$$\sum_{f=1}^{M} x_{af} = 1 \tag{14}$$

$$(\forall a \in A)$$

$$\sum_{a \in A} \sum_{f=1}^{M} GS_{la} p_{af} x_{af} + \sum_{i \in I} GS_{li} \boldsymbol{p}_i + T_l^0 + T^a \leq T_{MAX} \tag{15}$$

$$(\forall l \in A)$$

The constraint (14) bounds each core to a selected frequency. The constraint (15) guarantees the thermal limits imposed on the model. Where the set $A = a_i$ contains the index of the active cores and the set $I = i_i$ contains the index of idle cores directly defined from the solution of FSP. Where $A \cap I$ is empty. In general, the ISP problem is computationally simpler than the FSP problem due to the much lower number of decision variables and constraints.

In the next section we will evaluate the performance of the proposed TMC in a realistic scenario and under different trade-offs in between the predicted horizons of the FSP and ISP problems.

5 Experimental Results

In this section, we first describe the emulation framework we have created, starting from the results of the characterization of computing nodes and real scientific workload conducted in Section 3. We use this emulation framework to study the implication of the prediction interval/horizon and the task criticality generator in the thermal-aware task mapping and control of supercomputer nodes.

5.1 Emulation Framework

Our emulation framework is composed by the following components:

(i) The workload traces. The traces have been extracted using a commercial tracing and profiling tool called Intel Trace Analyzer and Collector. The traces contain all the MPI activities (MPI call, data transfer, source/destination MPI task) with time instants. These have been extracted for the QE-CP running on a computing node.

(ii) The thermal simulator. We have created a first order discrete state-space model matched with the computing node as described in Section 3.1. The model has a sample time of 10ms (Ts_{TM}), and as state variables has the temperature

of each core of the node. Each core's power is computed with the power model presented in Section 3.2. Workload traces which have resolution than the 10ms have been averaged on this period to produce the percentage of time in which each task was in the MPI library for each (Ts_{TM}) interval. We use this value to model the energy-aware MPI wrapper impacts on core's power consumption.

(iii) The thermal-aware task mapping and control problem. The TMC optimization problem proposed in Section 3 has been solved using IBM Ilog CPLEX 12.6.1. The emulator calls CPLEX each time there is a new TMC problem to be solved. This happens once at the application start (FSP) and periodically each ISP interval Ts_{ISP} which matches the prediction interval in the ISP problem (PI_{ISP}).

At each CPLEX call, the emulator builds a new instance of the problem with the new thermal parameters and the criticality of the tasks and it waits for CPLEX results. During the waiting time the emulator is frozen, in this way the overhead time does not impact on the chronological MPI events. CPLEX has been executed on the same machine of the emulation framework, which is our HPC node, therefore the time overheads reflect real measurement.

5.2 Evaluation of Prediction Horizons

In this section, we will explore how change the frequency level for high and low critical tasks using different prediction horizon for FSP and ISP problem. We conducted the following experiments with different prediction intervals for both FSP and ISP problems. We considered PI_{FSP}=1s,10s,100s,steady state (SS) and PI_{ISP}=1s,10s,100s,steady state (SS) because the thermal propagation in our system is in the order of tens of seconds as we reported in table 1. In the following, we name these tests with the notation $PI_{FSP} - PI_{ISP}$. It must be noted that 1s-1s represent state-of-the-art DTM solutions with no thermal-aware task-to-core mapping, while SS-SS represents state-of-the-art static DTM solutions.

For all the experiments, we set the temperature limit to 65% of maximum temperature which can be reached by the hottest core at the maximum frequency.

Figure 6 shows on the y-axis the temperature evolution of the coldest core (#0) for five cases. Namely no thermal control active, no thermal control active (NoTMC,NoEAW) but energy-aware MPI wrapper active (NoTMC,EAW), TMC active with (1s-1s), (SS-1s), (SS-SS). For the same configurations, the Figure 7 shows on the y-axis the temperature evolution of the hottest core. Clearly, according to the capability of the FSP problem, to predict the long term thermal evolution the higher critical (HC) task will be mapped on the coldest core. Indeed from Figure 6, we can notice that if no TMC calls are executed, the coldest core executes a low critical task. When the FSP is empowered with a steady-state thermal predictor instead the TMC allocates the higher critical task on the coldest core and manages to run it always at the maximum frequency. Vertical spikes of the frequency are caused by the energy-aware MPI wrapper, which sets the minimum frequency of the core during the MPI phases. As a consequence, the maximum temperature reached by NoTMC-EAW is lower than

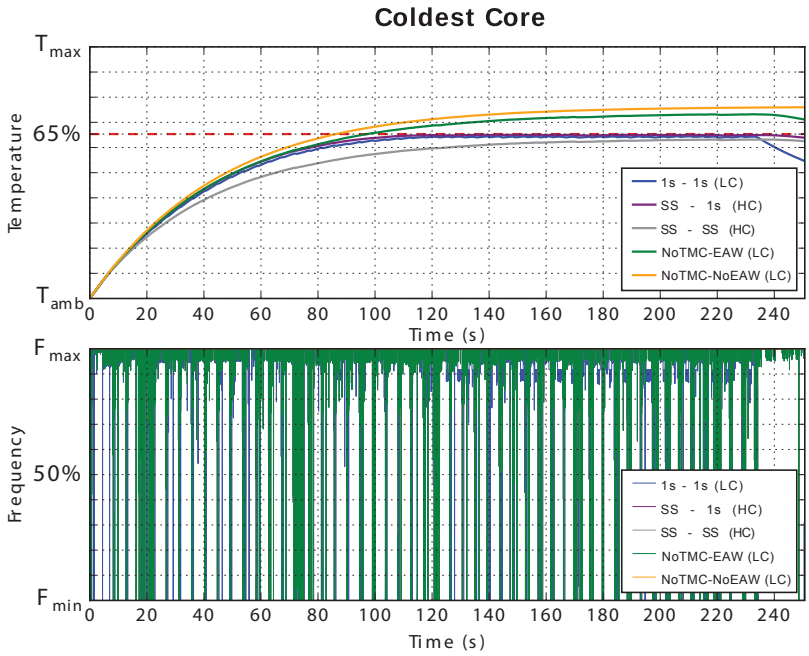

Fig. 6. Temperature and frequency evolution for the coldest core of the system - core #0

NoTMC-NoEAW; showing its effectiveness in reducing the power consumption. Differently, short time horizons (1s-1s) in the FSP do not allow the solver to "see" the constraint and thus lead to a sub-optimal task mapping allocation. As a consequence, the high critical task need to be frequency limited to meet the thermal constraint as the thermal capacitance effect vanishes.

5.3 Evaluation of the Task Criticality Generator

As previously introduced, the task criticality is a key parameter for the final application performance. Figure 8 shows the penalty in term of the execution time of application, when we consider equal criticality for each task respected of the one obtained by the TMC task criticality generator presented in section 4.1.

Figure 8 reports on the x-axis the cores where the highest critical task is allocated. We can see when the highest critical task is located on the core #6 or on the core #8 we have the highest and lowest penalty in the execution time, respectively 21.18% and 0.33%. When the root MPI task is located on the core #8, we have a lucky case, this means that it runs on the "coldest" core of the system where the TMC runtime can easily increase the core's frequency without violating the thermal constraint. On the other hand, when the root MPI task

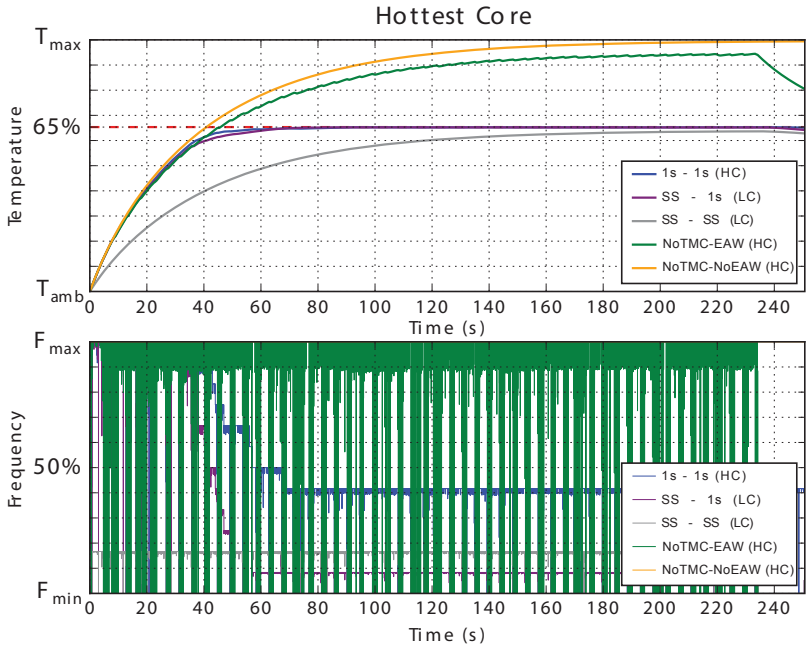

Fig. 7. Temperature and frequency evolution for the hottest core of the system - core #14

is located on the "hottest" core we have a high penalty due the difficult of the runtime to increase the frequency on that core. To conclude, we can evidently see that in all cases the TMC criticality generator outperforms the cases with task with the same criticality.

5.4 Performance Gain

Figure 9 depicts the average frequency of the cores that host the highest critical tasks and the average frequency for all the cores in each configuration. Interestingly, in all the cases the highest critical task never reaches the maximum average frequency. This is the effect of the energy-aware MPI wrapper which reduces the core frequency during MPI calls.

The error bars show the variance for each configuration among different executions of the same QE-CP problem while moving the highest critical task from the MPI root task to another one. This means that if we shift the default position of highest critical task from 0 to 15 in the MPI rank all the configuration with predict interval in the FSP (PI_{FSP}) of 1 and 10 seconds we have a huge variation. This can be explained by the fact that in both experiments the FSP has a prediction horizon which is too short to see the effect of long term thermal

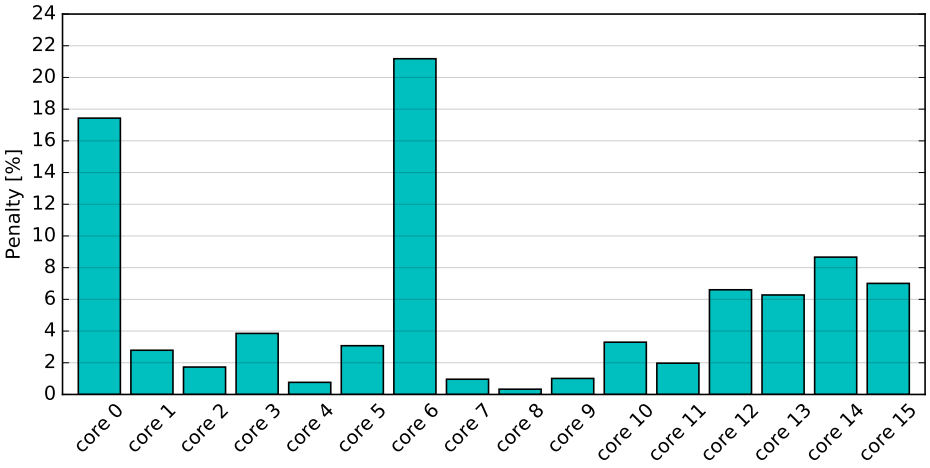

Fig. 8. Execution time penalty in benchmarks with equal per-task criticality level w.r.t. the benchmark with the TMC criticality generator. Every run identify on which core was pinned the highest critical task.

evolution and thus it cannot predict which core will hit the thermal constraint. For this case the allocation FSP problem is trivial and tasks are allocated on the first available core following a simple numerical binding where the task 0 will be allocate to the core 0 and so on. This binding is also the default on the Intel MPI library. In this particular case, if the highest critical task is lucky, it will be pinned on a "cold" core. Vice versa, if the highest critical task is unlucky, it will be mapped on a "hot" core. At the steady-state the frequency of the core will be limited by the ISP to respect the thermal constraint. On the other cases, the PI_{FSP} is always enough to sense the thermal constraint. The optimization model will avoid the binding of the highest critical task on a "hot" core. In this case the highest critical task will be pinned on a "cold" core allowing the highest critical task to work at maximum frequency.

We take as a baseline the SS-SS configuration, which model state-of-the-art solutions based on static allocation of tasks and frequency. The 1s-1s and 10s-10s induces performance penalties on the highest critical task, while they lead to an increase of performance of the 4.97% and 4.50% respectively in average in all the cores. For the remaining configurations, we measure no penalty for the highest critical tasks and a gain of to 7.46%, 7.06% and 3.65% respectively for the configuration SS-1s, SS-10s and SS-100s. These results show that short horizon predictive models pay off in the ISP as it allows to take advantage of the thermal capacitance. In the next section, we will add to this conclusion the solver overhead.

Overhead time Figure 10 shows cumulative overhead for different configurations and quantify the induced performance loss as it sums up to the execution

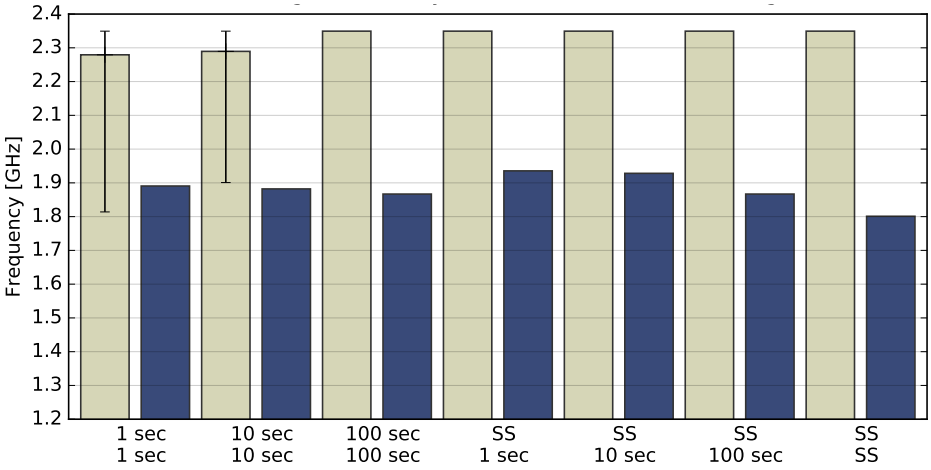

Fig. 9. Comparison between average core frequency and the frequency of the highest critical core using different configuration for the optimization problem.

time. The FSP bars represent the overhead time of the FSP problem solved only once at the application start, while the ISP bars are the sum of the overhead times of all iterations of the ISP solver.

For all the instances and the configurations, the solver is capable of finding the optimal solution. CPLEX allow to bound the solution time by the so called deterministic ticks, we use this approach to limit the solution time in case of harder problem. Authors of [28] show for a 60 core instance that the optimally gap always reduces below the 0.002% with a maximum number of 180 ticks.

We can see that for the 1s-1s and 10s-10s configuration the FSP solver time is negligible. After 1 second or 10 seconds the thermal transient has not reached the thermal constraint, for this reason the solution is trivial and consequently the solution immediately converge. Instead, all the other configurations have an average overhead time of 0.59% of total execution time.

The total overhead time for the ISP significantly changes when we vary the PT_{ISP} and the Ts_{ISP}. Obviously, the ISP with a prediction interval of 1 second will be called hundred times more than a ISP with a prediction interval of 100 seconds. The results respect this trend, in particular for 1 seconds of prediction interval leads to an average penalty of 10.20% of total execution time, which makes this configuration worse than a static allocation (SS-SS) as cause of the solution overhead (7.46% of performance gain - 10.20% of overhead). Interesting the 10 seconds case (SS-10s) reduces the total penalty to the 0.64% which in conjunction to the 7.06% of performance gain w.r.t. the static-allocation lead to an overall performance gain of the 6%. At 100 seconds the total overhead penalty decreases to the 0.09%. However, for this case the performance gain in only of the 3.46% making it less performing than the SS-10s case.

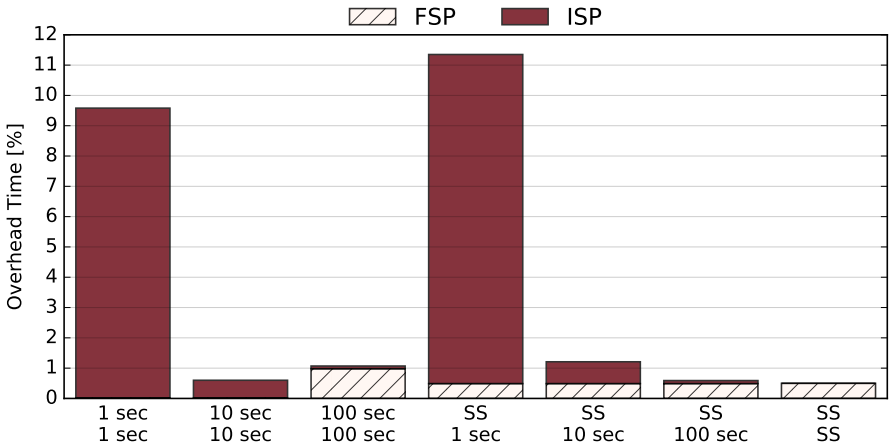

Fig. 10. Cumulative overhead induces by the optimization problem using different configuration for the optimization problem.

6 Conclusion

In this chapter, we propose a thermal-aware mapping and control of thermally-bound HPC nodes. Our system implements a novel ILP formulation for thermal-aware optimization and an exploration analysis on the workload application to address performance promoting critical MPI tasks. Our work is focused on real HPC hardware and workload. We extracted thermal characteristics as well as workload traces to study the workload distribution to identify critical MPI tasks. Our control system relies on these information to optimize the task allocation and the frequency selections in thermal-constraint HPC nodes.

In the experimental section, we compared our system with state-of-the-art DTM solutions which dynamically control only the frequency selection of the cores or can choose a statically task allocation with a specific frequency. Our experimental results show that using a long-time horizon for the task allocation and a short time horizon for selecting DVFS levels at execution time, our solution can lead up to 6% performance gain including overheads. Moreover, our task criticality model embedded in our DTM system can avoid the pinning of critical tasks on hot cores where OTC cannot promote this task with high frequency. This can cause high performance degradation up to 21.18% of the entire application execution.

Acknowledgments

Work supported by the EU FETHPC project ANTAREX (g.a. 671623), EU project ExaNoDe (g.a. 671578), and EU ERC Project MULTITHERMAN (g.a. 291125).

References

1. Top500.org. top 500 supercomputer sites, 2017.
2. R. Ayoub, S. Sharifi, and T. S. Rosing. Gentlecool: Cooling aware proactive workload scheduling in multi-machine systems. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 295–298. European Design and Automation Association, 2010.
3. A. Bartolini, M. Cacciari, C. Cavazzoni, G. Tecchiolli, and L. Benini. Unveiling eurora - thermal and power characterization of the most energy-efficient supercomputer in the world. In *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '14, pages 277:1–277:6, 3001 Leuven, Belgium, Belgium, 2014. European Design and Automation Association.
4. A. Bartolini, M. Cacciari, A. Tilli, and L. Benini. A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, March 2011.
5. F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini. Cooling-aware node-level task allocation for next-generation green hpc systems. *management*, 1:6, 2016.
6. F. Beneventi, A. Bartolini, A. Tilli, and L. Benini. An effective gray-box identification procedure for multicore thermal modeling. *IEEE Transactions on Computers*, 63(5):1097–1110, May 2014.
7. D. Cesarini, A. Bartolini, and L. Benini. Benefits in relaxing the power capping constraint. In *Proceedings of the 1st Workshop on AutotuniNg and aDaptivity AppRoaches for Energy Efficient HPC Systems*, ANDARE '17, pages 3:1–3:6, New York, NY, USA, 2017. ACM.
8. D. Cesarini, A. Bartolini, and L. Benini. Prediction horizon vs. efficiency of optimal dynamic thermal control policies in hpc nodes. In *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, Oct 2017.
9. C. Conficoni, A. Bartolini, A. Tilli, G. Tecchiolli, and L. Benini. Energy-aware cooling for hot-water cooled supercomputers. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 1353–1358, San Jose, CA, USA, 2015. EDA Consortium.
10. 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*, 28(10):1503–1516, 2009.
11. A. K. Coskun, T. S. Rosing, and K. Whisnant. Temperature aware task scheduling in mpsocs. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1659–1664. EDA Consortium, 2007.
12. A. K. Coşkun, K. Whisnant, K. C. Gross, et al. Static and dynamic temperature-aware scheduling for multiprocessor SoCs. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(9):1127–1140, 2008.
13. J. Eastep, S. Sylvester, C. Cantalupo, F. Ardanaz, B. Geltz, A. Al-Rawi, F. Keceli, and K. Livingston. Global extensible open power manager: A vehicle for hpc community collaboration toward co-designed energy management solutions.
14. V. W. Freeh, N. Kappiah, D. K. Lowenthal, and T. K. Bletsch. Just-in-time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs. *Journal of Parallel and Distributed Computing*, 68(9):1175–1185, 2008.
15. R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron. Cpu miser: A performance-directed, run-time system for power-aware clusters. In *2007 International Conference on Parallel Processing (ICPP 2007)*, pages 18–18. IEEE, 2007.

16. P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, et al. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of physics: Condensed matter*, 21(39):395502, 2009.
17. P. Hammarlund, R. Kumar, R. B. Osborne, R. Rajwar, R. Singhal, R. D'Sa, R. Chappell, S. Kaushik, S. Chennupaty, S. Jourdan, et al. Haswell: The fourth-generation Intel core processor. *IEEE Micro*, (2):6–20, 2014.
18. V. Hanumaiah, S. Vrudhula, and K. S. Chatha. Performance optimal speed control of multi-core processors under thermal constraints. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 1548–1551, April 2009.
19. K. A. Huck and J. Labarta. Detailed load balance analysis of large scale parallel applications. In *Parallel Processing (ICPP), 2010 39th International Conference on*, pages 535–544. IEEE, 2010.
20. H. Khdr, S. Pagani, M. Shafique, and J. Henkel. Thermal constrained resource management for mixed ilp-tlp workloads in dark silicon chips. In *Proceedings of the 52nd Annual Design Automation Conference*, page 179. ACM, 2015.
21. H. Khdr, S. Pagani, E. Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel. Power density-aware resource management for heterogeneous tiled multicores. *IEEE Transactions on Computers*, 66(3):488–501, 2017.
22. M. Y. Lim, V. W. Freeh, and D. K. Lowenthal. Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs. In *SC 2006 conference, proceedings of the ACM/IEEE*, pages 14–14. IEEE, 2006.
23. M. Maiterth, T. Wilde, D. Lowenthal, B. Rountree, M. Schulz, J. Eastep, and D. Kranzlmiiller. Power aware high performance computing: Challenges and opportunities for application and system developers x2014; survey tutorial. In *2017 International Conference on High Performance Computing Simulation (HPCS)*, pages 3–10, July 2017.
24. S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D. Micheli. Temperature-aware processor frequency assignment for mpsocs using convex optimization. In *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2007 5th IEEE/ACM/IFIP International Conference on*, pages 111–116, Sept 2007.
25. O. Pearce, T. Gamblin, B. R. de Supinski, M. Schulz, and N. M. Amato. Quantifying the effectiveness of load balance algorithms. In *Proceedings of the 26th ACM International Conference on Supercomputing*, ICS '12, pages 185–194, New York, NY, USA, 2012. ACM.
26. D. Puschini, F. Clermidy, P. Benoit, G. Sassatelli, and L. Torres. Temperature-aware distributed run-time optimization on mp-soc using game theory. In *Symposium on VLSI, 2008. ISVLSI'08. IEEE Computer Society Annual*, pages 375–380. IEEE, 2008.
27. B. Rountree, D. K. Lownenthal, B. R. De Supinski, M. Schulz, V. W. Freeh, and T. Bletsch. Adagio: making dvs practical for complex hpc applications. In *Proceedings of the 23rd international conference on Supercomputing*, pages 460–469. ACM, 2009.
28. A. Rudi, A. Bartolini, A. Lodi, and L. Benini. Optimum: Thermal-aware task allocation for heterogeneous many-core devices. In *High Performance Computing Simulation (HPCS), 2014 International Conference on*, pages 82–87, July 2014.
29. Z. Wang, C. Bash, N. Tolia, M. Marwah, X. Zhu, and P. Ranganathan. Optimal fan speed control for thermal management of servers. In *ASME 2009 InterPACK Conference collocated with the ASME 2009 Summer Heat Transfer Conference and the ASME 2009 3rd International Conference on Energy Sustainability*, pages 709–719. American Society of Mechanical Engineers, 2009.

30. Q. Xie, M. J. Dousti, and M. Pedram. Therminator: A thermal simulator for smartphones producing accurate chip and skin temperature maps. In *Low Power Electronics and Design (ISLPED), 2014 IEEE/ACM International Symposium on*, pages 117–122, Aug 2014.
31. Y. Xie and W.-L. Hung. Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (mpsoc) design. *The Journal of VLSI Signal Processing*, 45(3):177–189, 2006.
32. F. Zanini, D. Atienza, L. Benini, and G. D. Micheli. Thermal-aware system-level modeling and management for multi-processor systems-on-chip. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 2481–2484, May 2011.