Energy Saving and Thermal Management Opportunities in a Workload-Aware MPI Runtime for a Scientific HPC Computing Node

September 2017

Energy Saving and Thermal Management Opportunities in a Workload-Aware MPI Runtime for a Scientific HPC Computing Node

Daniele CESARINI [a] Andrea BARTOLINI [a] and Luca BENINI [a,b]
[a] *DEI, University of Bologna, 40136 Bologna, Italy*
[b] *IIS, Swiss Federal Institute of Technology, 8092 Zurich, Switzerland*

Abstract. With the advent of a new generation of supercomputers characterized by tightly-coupled integration of a large-number of powerful processing cores in the same die, energy and temperature walls are looming threats to the growth in computational power.

Scientific computing is characterized by a single application running in parallel on multiple nodes and cores until termination. The message-passing programming model is a widely adopted paradigm for explicitly handling data-sharing between processes of the same application. As an effect of the MPI communication patterns among different processes, the application is characterized by phases which can be exploited by OS power manager. In addition, the large number of cores integrated in the same silicon die introduces large thermal capacitance as well as on-die thermal heterogeneity. Jointly exploiting local workload unbalance and computational node heterogeneity can open interesting opportunities for advanced thermal and energy management. In this paper, we present an exploratory work to assess these opportunities and their limiting factors. We analyze application workload and we identify opportunities to reduce energy consumption and their impact on performance. We test our methodology on a widely-used quantum-chemistry application demonstrating potential benefits of combining the application flow with power and thermal management strategies.

Keywords. HPC, thermal model, power model, energy, MPI, runtime, scientific workload

1. Introduction

Nowadays, it is well established that the pace dictated by the Moore's law on technological scaling comes at the cost of increasing power consumption and leads to thermally-bound computing systems. Supercomputers as well as data centers are on the cutting edge of this crisis, because of aggressive performance, integration density and sustainable power budget [16,19].

The most powerful supercomputer in Top500 is *Sunway TaihuLight* which consumes 15.3 MW to deliver 93 PetaFLOPs. The second one, *Tianhe-2* (ex 1st) consumes 17.8 MW for "only" 33.2 PetaFLOPs. However, the power consumption increases to 24 MW when considering also the cooling infrastructure [9]. Such an amount of cooling power serves to prevent thermal issues. Increasing the inlet coolant temperature reduces the cooling cost, but impacts thermal budget. Druzhinin et al. [17] reports a performance drop of 10% caused by thermal throttling when raising inlet cooling water from 19^oC up to 65^oC in a direct liquid cooling supercomputer.

Beneventi et al. [4] show in an Intel-base computing node with 36 physical cores, that the increased number of processors integrated on same die generates significant thermal gradients and heterogeneity. On the same silicon die, they measure up to 24^oC of temperature difference between active cores and idle cores, and more than 7^oC of thermal heterogeneity under homogeneous workload.

In high performance computing nodes, the maximum safe temperature at which processing elements can run depends on cooling technologies. Intel Xeon E5-26XX v3 HPC class processors have different specifications on the maximum silicon temperature which ranges from 69^oC to 101^oC according to the package thermal resistance (cost) and the nominal thermal design power (TDP)[1]. Dynamic thermal management (DTM) has been introduced to reduce the worst-case cooling effort by controlling and limiting, when necessary, the heat generation. Operating systems use reactive controllers to maintain processors under a critical temperature. Several state-of-the art approaches explore proactive techniques to improve DTM performance [1,8].

CPU power is the largest contributor to power cost on modern servers (at idle state contributes up to 40% of total power and 66% at peak power [2]). Today's CPUs use hardware mechanisms to adapt performance to workload to become more energy efficient. Dynamic voltage and frequency scaling (DVFS) techniques can change clock rate during application execution to reduce the overall energy consumption. However, in a supercomputing environment, user applications run in isolation on a portion of the machine and are accounted based on execution time. The peculiar usage model of supercomputing hardware and applications, which are not interactive, nor latency critical, but require sustained peak performance, is not yet well understood in terms of unique opportunities for domain specific power and thermal management.

This work focuses on the evaluation of potential benefit in combining state-of-the-art energy-thermal management strategies with the MPI programming model which is pervasive in real-life HPC applications. Our contributions are:

1. A formal model for the energy and thermal management problem in HPC computing nodes. This includes reactive power management, to exploit computation vs communication phases, and proactive thermal management, to take advantage of: (i) heterogeneity in the core thermal dissipation, (ii) heterogeneity in application threads composition and (iii) thermal capacitance.
2. We characterized the node of a supercomputing system today ranked 151 in the TOP500 list [10] in terms of thermal heterogeneity and inertia (i.e. capacitance) and impact of HW power management mechanisms.
3. We characterized a real scientific quantum-chemistry parallel application, measuring the thread unbalance and communication-to-computation ratio.
4. We evaluated i) the potential impact of reactive power management strategies which can lead up to 12.39% energy reduction in this context; ii) the potential of proactive thermal management solutions which can lead up to 6% performance improvements when running the system in a thermally constrained environment.

The paper is organized as follows. Section 2 presents related works, Section 3 characterizes HW and SW characteristics of scientific computing relevant to power and thermal management solutions. Section 4 defines properties of an optimal policy for this context, while Section 5 quantifies achievable benefits and opportunities.

2. Related work

Our exploration work ranges from thermal and power modeling for HPC system to energy-aware MPI runtimes and thermal-aware allocation schedulers. In this section, we review the state-of-the-art in this field.

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

Several works have investigated thermal-aware workload allocation, making use of mechanisms such DVFS to prevent the activation of more drastic cooling measures. Those approaches include: (1) on-line optimization policies [6,7,14] based on predictive models and taking advantage of run-time temperatures read from hardware sensors; (2) off-line allocation and scheduling approaches [18], usually embedding a simplified thermal model of the target platform [5,3].

There are even significant works on energy-aware MPI runtimes. Adagio [20] uses DVFS mechanism to reduce CPU frequency during communication phases trying to predict the duration of each task adjusting the CPU frequency. A task is identified as a period of computation between two MPI calls. Adagio correctly identifies tasks when there are iterative computational phases interspersed with communication phases. However, several HPC applications do not respect the same communication pattern, for instance [15], this leads to a misprediction that causes performance penalty.

This work uses proactive techniques to reduce energy consumption in MPI phases. Moreover, it does not consider thermal-bounded environments where computational tasks must slow down to avoid overheating. Differently, our runtime adjusts the CPU frequency when the execution flow calls MPI functions leaving performance decisions for computational tasks to the thermal management scheduler which assures a safe-working environment.

3. HPC Architecture and Workload Characterization

3.1. HW and SW for HPC Machine

HPC machines are clusters composed of tens to thousands of high-end nodes interconnected with a low-latency, high-bandwidth network. Usually each node is composed of multiple sockets, each socket hosts multiple cores that share a main memory and the network subsystem. Nodes are accommodated in racks located in server rooms of a data center. Server rooms require a cooling system to remove the heat produced by the racks.

HPC resources are requested from users through a batch queue system which partitions the HPC machine in sub-clusters for limited lifetime. HPC applications are quite different from general server workload. At execution time, a single scientific computing application runs exclusively on the allocated sub-clusters. HPC software consists of multiple instances of the same application that can run on different nodes. Ideally, application instances are composed by computational intensive phases on independent data segments alternated by synchronization points and communication phases. MPI APIs are used to support explicit data communication between application instances.

3.2. Power and Thermal Management

The power management states of computing elements are divided in sub-groups. The P-States include dynamic voltage and frequency (DVFS) operating points which target the reduction of active power. C-States instead target idle power reduction strategies. Both P-States and C-States are numbered from 0 to n. Higher number means higher power saving. However, in case of C-State this means also longer transitions in and out of the state itself. In recent Intel architectures [12] P-States can be selected independently for each core. C-States are instead defined independently for cores and the ”uncore” region.

The P-States are handled by the Linux O.S. by means of SW frequency governors. C-States are triggered from the firmware of the CPU but the OS can provide hints on the appropriate C-State to the hardware through O.S. idle governors. Table 1 shows the different architecture impacts and dependencies for the different cores and package C-States.

Table 1. C-States

Package C-States	Core C-States: C0	C1	C3	C6	
C0					Active State
C1E					Lower P-State
C2					Only L3 Snoop
C3					Flush L3 - Off
C6					Low Voltage
	Active State	Clock Gated	Flush L1,L2 Off	Power Gated	

Green cells represent a valid configuration for Core and Package C-States, red are invalid ones.

Table 2. 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
Time to reach steady state	$120sec$

Table 3. Quntum ESPRESSO - Energy

	BW MPI	IB MPI	IB MPI - C1 lim
Exe Time	$261.52sec$	$283.48sec$	$283.48sec$
Avg Power	$68.19W$	$62.91W$	$63.22W$
Energy	$17,901J$	$17,833J$	$18,048J$
Idleness	0.00%	15.80%	15.80%
C1	0.00%	9.74%	15.80%
C3	0.00%	0.66%	0.00%
C6	0.00%	5.40%	0.00%

3.3. Thermal and Power Characterization of a Supercomputer Node

We took as target high performance computing infrastructure, which is a Tier-1 HPC system based on an IBM NeXtScale cluster. Each node of the system is equipped with 2 Intel Haswell E5-2630 v3 CPUs, with 8 cores with 2.4 GHz nominal clock speed and 85W Thermal Design Power (TDP, [13]). Taking into account also miscellaneous components, the overall system TDP is about 360 KW. As regards the software infrastructure, SMP CentOS Linux distribution version 7.0 with kernel 3.10 runs on each node. This Tier-1 supercomputer is currently classified in the Top500 supercomputer list [10].

3.3.1. Thermal Model

We start our work towards the exploitation of the thermal characteristics of the HPC system. We focus our attention on a single node of the cluster.

To understand the thermal properties of a computational node we have executed three main stress tests on which we have (i) kept the system in idle and measured the power and each per-core temperature after ten minutes, (ii) we have executed a stressmark[2] in sequence on each core of each socket in the node, leaving the remaining ones idle. We maintained a costant workload for ten minutes and measured power consumption and temperature. This test has been used to extract the maximum gradient between cores. Finally, (iii) we have executed the stressmark for ten minutes in all the cores of the node simultaneously and we measured the temperature and the power consumption. In all the previous tests, temperature and power values were measured by periodically accessing the machine specific registers, the Turbo mode was disabled. Results of our analysis are reported on Table 2.

[2]cpuburn stressmark by Robert Redelmeier: it takes advantage of the superscalar architecture to maximize the CPU power consumption

Table 4. Power model - Active

	C0 - Turbo	C0 - Max Freq	C0 - Min Freq
Puncore	17.13*W*	17.13*W*	12.76*W*
Pcore	6.38*W*	5.47*W*	3.16*W*

Table 5. Power model - Idle

	C1	C3	C6
Puncore	12.76*W*	11.90*W*	11.84*W*
Pcore	1.32*W*	0.38*W*	0.00*W*

3.3.2. Power Model

In addition to the previous tests, we have re-executed the stressmark in different configurations (number of cores executing the stressmark, Turbo mode enabled/disabled, and using different frequency levels) while limiting C-States[3]. We maintained each configuration for ten minute and we measured the power consumed by each CPU.

Table 4 and 5 report the power split between core and uncore regions. Table 4 shows them for different P-States (C0=active) (Turbo, Max, Min Frequency) while Table 5 shows them for different C-States. The power of the uncore in *C0 - Turbo* and *C0 - Max Freq* is equal because uncore component is not affected by the Turbo frequencies. Scaling down the frequency (Min freq) however reduces the uncore power. Core power instead scales proportionally with the frequency. Idle power instead can be seen in Table 5 from which we can see that *C1* further reduces the power of 58% w.r.t. *C0 - Min Freq* and that *C3* significantly cuts the idle power by the 71% for the core, but only marginally for the uncore. *C6* instead zeroes the core power but reduces the uncore power only by 1%. Clearly, these results show that to reduce energy, parallel programming runtime (i.e. MPI) should prefer *C1* and *C3* states instead of P-States (DVFS) during idle intervals (communication phases). We will use this thermal and power model characterization in power management space exploration in Section 5.

3.4. Scientific Workload

3.4.1. Quantum ESPRESSO

In this work, we use Quantum ESPRESSO (QE) [11] as a real-life workload scientific application to evaluate power and thermal management opportunities in a real HPC infrastructure. QE is a software suite for molecular dynamics. Its main computational kernels include dense parallel linear algebra and 3D parallel fast Fourier transform, which are both relevant for many HPC applications. In detail, we use a *Car-Parrinello* (CP) simulation, which prepares an initial configuration of a thermally disordered crystal of chemical elements by randomly displacing the atoms from their ideal crystalline positions. This simulation consists of a number of computation kernels that have to be executed in the correct order.

3.4.2. MPI Runtime and Workload Characterization

We use Intel *MPI Library 5.1* as the runtime for communication and Intel *ICC/IFORT 16.0* in our toolchain. We choose Intel software stack because it is currently used in our target system and in most of HPC machines [10] based on Intel processors. This runtime tries to achieve maximum performance by adopting to default a busy-waiting (BW MPI) policy that forces CPU polling on a network fabric controller during communication phases and synchronization points. Alternately programmers can specify an interrupt-based (IB MPI) communication. With IB communication, HPC applications release the processor during communication phases leaving the control to the O.S. which triggers idle state (C-State). To isolate the impact of deep C-States from the different communication mechanisms, we also run the interrupt-base communication while limiting the deepest C-State to C1 (IB MPI - C1 limited).

Table 3 shows the results of QE using different configurations of MPI runtime. For all the benchmarks we compare: execution time, average power and total energy for both

[3] C-States in Intel architectures can be limited through dedicated machine specific registers

sockets, we use idleness which represents the percentage of idle time w.r.t. total time and as well as the C1, C2, C3 percentage that is measured through HW counters. *BW MPI* benchmark, we can see as the effect of the busy-waiting communication has 0% of idleness and C1,2,3 states. Differently *IB MPI* shows a significant percentage of idleness which is exploited by entering in deep C-States. Due to the HW policy, which triggers transition to C-States using time-outs, the specific communication phase can finish before the deepest C-State is reached (C6). Thanks to *IB MPI* configuration, the application consumes 8% less power than *BW MPI*. This however does not cause an energy reduction due to an increase in execution time when *IB MPI* is used. This could be caused by recovering from deep C-States. In *IB MPI - C1 limited* only C1 states are used, which are characterized by a negligible transition time. When comparing its execution time with the one of *IB MPI*, we notice that there is no variation. This demonstrates that the slowdown in *IB MPI* w.r.t. *BW MPI* is not related by recovering from deep C-State but to the overhead of interrupt handler.

4. Energy Thermal Management Policies

As we have seen in the previous section, HPC workload contains communication-induced idle times, which are opportunities to reduce energy consumption. However due to the IB MPI, this does not turn in a significant energy reduction due to increased execution time related to the specific MPI communication implementation and due to the HW C-State promoting policy. In the assumption that these implementation issues can be solved, we are still missing an evaluation of the potential energy-saving achievable by exploiting these communication slacks.

In addition, we also see that supercomputer workload faces significant unbalance between tasks of the same application (i), but these processes cannot be migrated among cores, as they have to run with the same binding till the end of the application (ii). Moreover, computing nodes show a significant thermal heterogeneity (iii) and long thermal transients (iv). By combining these characteristics with state of the art thermal and power management solution we can envision that a thermal and energy management strategy should include:

i) *Reactive Policy*: Minimizes the overall energy by reacting to application phases. It triggers the transition to a lower power state (P- or C-State) during communication slack.

ii) *Proactive policy*: It minimizes energy by maximizing the performance when the computing resources are pushed against their thermal limit. It selects at the application start time the MPI process to core binding to guarantee that computational demanding processes run at the maximum performance. At execution time, it acts to the core frequency to ensure that each core operates in a safe temperature range.

In the next section, we will quantify the achievable savings for reactive and proactive power and thermal management policies.

5. Assessing Energy Saving Opportunities

Real workload traces have been recorded from runs of QE (BW) on computational nodes using Intel MPI tracing tools. These traces contain all MPI activities (MPI call, data transfer, source/destination IDs) with a time stamp. The traces are provided as inputs to our policy energy estimation script. The reactive policy consists of selecting a given idle state C[1:6] or DVFS state (Px Min Freq) for each MPI communication phase and C0 state when active. This emulated an ideal reactive policy with zero transition overhead useful to quantify the best-case saving. Results are reported in Section 5.1. Proactive policy consists of a thermal-aware task mapping and control problem. We implement and extension of the ILP formulation proposed in [21] to assign at each process a priority

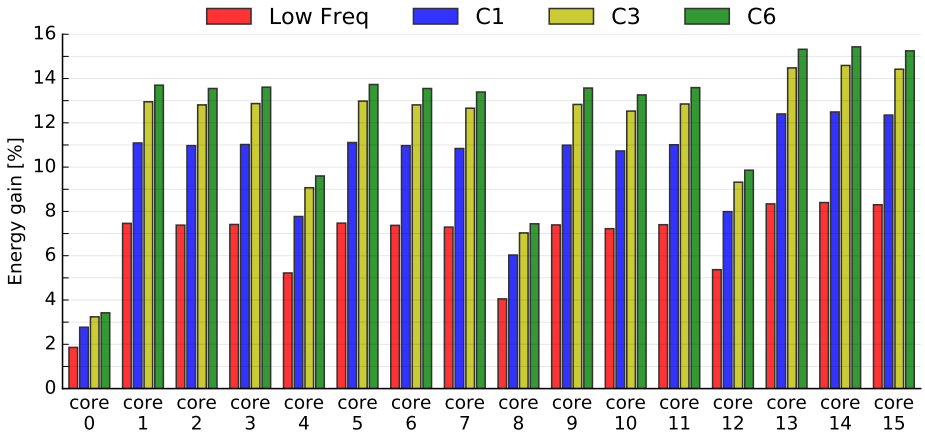

Figure 1. Energy saving of MPI with *reactive policy* w.r.t. the MPI BW

weight and account for it in the cost function which now is $max\sum(Prior_c, i * freq_c, i)$ where c is the core and i is the time interval. The ILP problem uses constraints to select a per-core frequency that forces future core temperatures to be below a critical temperature. Future temperature is predicted through a thermal model. As task migration has very significant performance penalty in MPI workloads, we split the problem in two parts: (P1) intial MPI process to core mapping and (P2) on-line frequency selection. We evaluate the effect of the prediction horizon on both P1 and P2. Indeed, long horizon in P1 means better accounting in the MPI process allocation of the steady-state thermal heterogeneity. Instead for P2, long horizon means worse exploitation of the thermal capacitance but lower overheads. The optimization problems are solved using IBM Ilog CPLEX 12.6.1 running on the same machine, therefore the time overheads can be comparable with application time. Finally, our evaluation framework is composed by a power and thermal model. We used a time constant discrete state-space model matched with the computational node as described in table 2. The model has a sample time of 10ms, and as state variables has the temperature of each core of the node. Core power is computed with the power model described in Table 4 and 5 extended with additional P-States. Workload traces that have higher temporal accuracy than the 10ms, have been averaged among this period to produce the percentage of time in which each task was in the MPI runtime for each 10ms interval.

5.1. Reactive

In this section, we quantify the maximum achievable energy saving for a reactive policy under ideal conditions: no interrupt, library and transitions overheads. We use the simulator and the MPI traces as explained above. We compare the results with the baseline policy of Intel MPI BW considering the reactive policy to select *Low Freq*, *C1*, *C3* or *C6* during communication slack.

Figure 1 shows with different bars the low-power state used by the reactive policy during communication phases. In y-axis is reported the achievable energy-saving w.r.t. to baseline busy waiting for each core (x-axis). From the plot we can first recognize that different cores have different communication-to-computation ratio, which leads to a significant workload unbalance in QE. This is reflected by a different achievable saving for each core. When comparing the achievable energy-reduction, Figure 1 shows that C-States are more effective in saving energy during communication phases. Indeed, while reducing the frequency to *Low Freq* saves in average 7%, using C1 reduces the energy

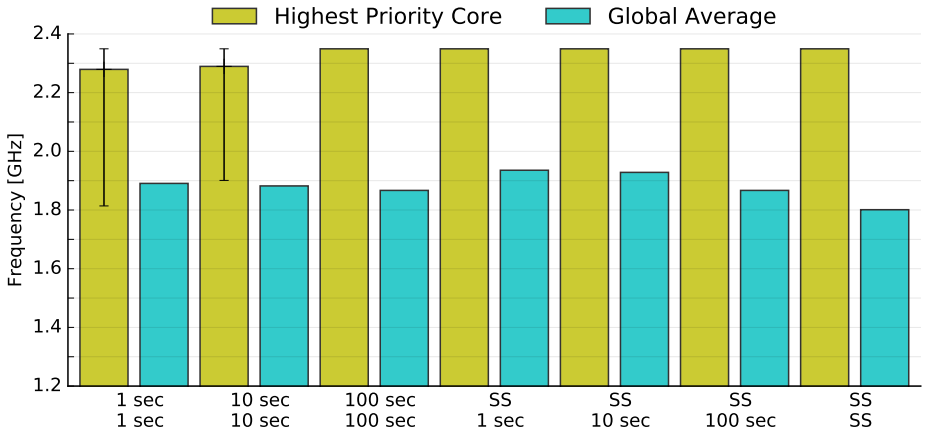

Figure 2. Frequency of the highest priority core and global average using different configurations

consumption by 10%. Using *C3* and *C6* leads to an additional saving reaching 11.7% and 12.4% respectively. Considering the different latencies intrinsic to these low-power states, from our analysis we suggest that MPI runtime should maximize the usage of C1 during communication slack with an energy saving opportunity of 10% for QE[4].

5.2. Proactive

In this section, we evaluate the opportunities for proactive thermal management. The test is conducted by assigning the highest priority to the MPI process with rank 0 which is the most critical. We assume the reactive policy to work by select *Low Freq* during communication phases.

Figure 2 depicts on the y-axis the average frequency of the cores that host highest priority tasks and the average frequency for all the remaining cores. On the x-axis we report the different predictive horizon. This is composed by two numbers: in the top we report the P1 horizon at the bottom the P2 horizon. SS means the prediction at the steady state. The error bars show the variance for each configuration among different executions of the same QE problem while moving the highest priority job[5] from different cores. High bars mean high performance because the scheduler can set an higher frequency respecting thermal constraints. From the figure, we can notice that *1s-1s* and *10s-10s* induces performance penalties on the high priority task, while they lead to an increase of performance of the 5% and 6% respectively in average in all remaining cores. For the remaining configurations, we measure no penalty for high priority tasks and a gain of the 8%, 7% and 4% respectively for the configuration *SS-1s*, *SS-10s* and *SS-100s*. These results shows that short horizon predictive models pay off in P2 as it allows to take advantage of the thermal capacitance.

Figure 3 shows the associated overhead for the scheduler. Low bars mean low overhead quantified in percentage which represent the a sum of seconds for all iterations during application execution. We take as a baseline *SS-SS* configuration, which model state-of-the-art solutions based on static allocation of jobs and frequencies.

From Figure 3 we can see that for the *1s-1s* and *10s-10s* configuration the P1 solve time is negligible. This is because these horizons are not enough to predict the thermal ef-

[4]In this exploration we consider only Core C-States because Package C-States are always close to 0% of utilization due the continuous presence in C0/C1 state on at least one core

[5]In our benchmark the highest priority job is the root MPI process identified with job ID 0

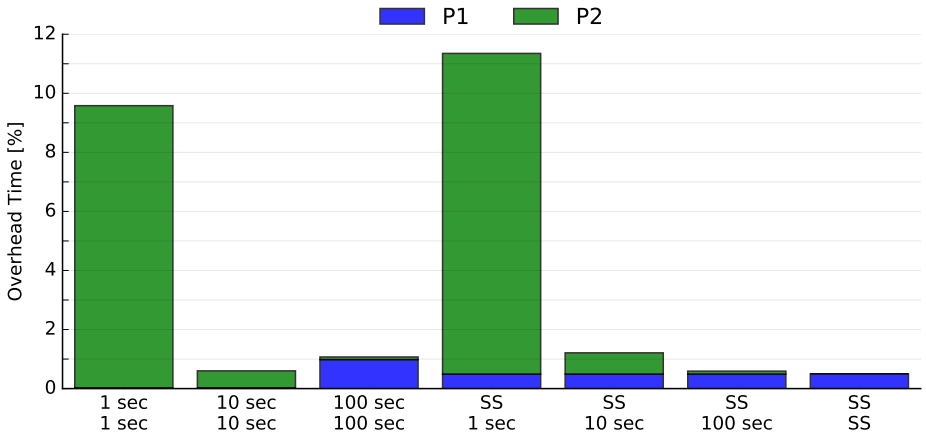

Figure 3. Cumulative overheads of the proactive scheduler for both P1 and P2 problem

fects when transients are expired. For this reason, the solution is trivial and consequently the optimization algorithm immediately converge. Instead, all other configurations cause an average overhead time of 0.59% of total execution time.

When looking at P1, short horizon will lead to more calls to the solver and thus a large overhead is expected. The results respect these trends, in particular for 1 seconds of prediction interval it leads to an average penalty of 10% of total execution time, which makes this configuration worse than the static allocation (SS-SS) 8% of performance gain with 10% of overhead. Interesting the 10 seconds case (SS-10s) reduces the total penalty below 1% which, in conjunction to 7% of performance gain, leads to an overall performance gain of 6%. At 100 seconds the total overhead penalties decrease to 0.1%. However, for this case the performance gain is only of 4% leading to a worse performance than the *SS-10s* case. As a matter of fact, proactive thermal management solutions can take advantage of the thermal capacitance and heterogeneity in compute nodes by smart static core mapping (P1) and dynamic frequency selections. We evaluated that long horizons are required by static process-to-core mapping and medium predictive horizons are the best for dynamic thermal adaptation as they can exploit the thermal capacitance as well as producing reduced overhead. When these optimal configurations are chosen, up to 6% of performance gain can be obtained w.r.t. static thermal optimization.

6. Conclusion

In this paper, we presented a novel exploration of energy and thermal management policies. Differently from state-of-the-art solutions, we focused our analysis on a real supercomputing system from which we modeled the real thermal and power characteristics as well as we extracted real scientific workload traces. We developed an energy-aware MPI wrapper to explore the maximum energy that could be saved during communication phases. We implemented thermal-aware allocation schedulers, which are induced by MPI that guarantees a safe working temperature while it maximizes performance on critical task. For thermal-aware allocation schedulers we explore different configurations to find-out the best performance point in thermally-bounded systems.

Our results show that energy-aware MPI wrapper can save up to 12% of energy during communication phases. For the online DVFS selection our implementation can lead up to 6% performance gain including overheads while ensuring that high priority task run always at the maximum frequency.

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] 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.

[2] L. A. Barroso, J. Clidaras, and U. Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] J. Dongarra. Visit to the national university for defense technology changsha. *China, University of Tennessee*, 199, 2013.

[10] J. J. Dongarra, H. W. Meuer, E. Strohmaier, et al. Top500 supercomputer sites. *Supercomputer*, 11:133–133, 1995.

[11] 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.

[12] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer. An energy efficiency feature survey of the intel haswell processor. In *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*, pages 896–904. IEEE, 2015.

[13] 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*, 34(2):6–20, 2014.

[14] V. Hanumaiah, S. Vrudhula, and K. S. Chatha. Performance optimal online dvfs and task migration techniques for thermally constrained multi-core processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(11):1677–1690, 2011.

[15] D. J. Kerbyson, A. Vishnu, and K. J. Barker. Energy templates: Exploiting application information to save energy. In *2011 IEEE International Conference on Cluster Computing*, pages 225–233. IEEE, 2011.

[16] L. B. Kish. End of moore's law: thermal (noise) death of integration in micro and nano electronics. *Physics Letters A*, 305(3):144–149, 2002.

[17] A. Moskovsky, E. Druzhinin, A. Gromov, A. Shmelev, V. Mironov, and A. Semin. Server level liquid cooling: Do higher system temperatures improve energy efficiency? *Supercomputing frontiers and innovations*, 3(1):67–74, 2016.

[18] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. De Micheli. Temperature control of high-performance multi-core platforms using convex optimization. In *2008 Design, Automation and Test in Europe*, pages 110–115. IEEE, 2008.

[19] M. K. Patterson. The effect of data center temperature on energy efficiency. In *Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. ITHERM 2008. 11th Intersociety Conference on*, pages 1167–1174. IEEE, 2008.

[20] 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.

[21] 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.