# Unveiling Eurora - Thermal and Power Characterization of the most Energy-Efficient Supercomputer in the World

Andrea Bartolini*§, Matteo Cacciari*, Carlo Cavazzoni†, Giampietro Tecchiolli‡ and Luca Benini*§
*DEI, University of Bologna, Italy, {*a.bartolini, matteo.cacciari, luca.benini*}@*unibo.it*
§ISS, ETH Zurich, Switzerland, {*barandre, lbenini*}@*iis.ee.ethz.ch*
†SCAI, CINECA, Italy, *c.cavazzoni@cineca.it*
‡Eurotech SpA, Italy, *giampietro.tecchiolli@eurotech.com*

***Abstract*—Eurora (EURopean many integrated cORe Architecture) is today the most energy efficient supercomputer in the world. Ranked 1st in the Green500 in July 2013, is a prototype built from Eurotech and Cineca toward next-generation Tier-0 systems in the PRACE 2IP EU project. Eurora's outstanding energy-efficiency is achieved by adopting a direct liquid cooling solution and a heterogeneous architecture with best-in-class general purpose HW components (Intel Xeon E5, Intel Xeon Phi and NVIDIA Kepler K20). In this paper we present a novel, low-overhead monitoring infrastructure capable to track in detail and in real-time the thermal and power characteristics of Eurora's components with fine-grained resolution. Our experiments give insights on Eurora's thermal/power trade-offs and highlight opportunities for run-time power/thermal management and optimization.**

## I. Introduction & Related Work

While electronics devices are facing critical technological walls (i.e. Power/Thermal/Utilization Walls) that are limiting the performance benefits of the transistor size scaling, the demand for more powerful supercomputers continue to increase. The TOP500 organization collects and ranks the worldwide peak performance, measured as Flops (floating point operation per second), of new supercomputer installations when running Linpack Benchmarks. Trends in the last twenty years show an exponential growth of peak performance that is predicted to enter the ExaFlops ($10^{18}$) scale in 2018 [7]. Today's most powerful Supercomputer, Tianhe-2, reaches 33.2 PetaFlops with 17.8 MW of power dissipation that increases to 24 MW considering also the cooling infrastructure [6]. These data show that Exascale supercomputers cannot be built by simply expanding the number of processing nodes of today systems as their power demand would increase unsustainably (hundreds of MW of power). According to [3], an acceptable value for an Exascale supercomputer is 20 MW. To reach this target, current supercomputer systems must achieve a significantly higher energy efficiency pushing towards a goal of 50 GFlops/W.

With the aim to lead supercomputers to improve energy efficiency, the Green500 organization ranks Top500 supercomputers by their energy efficiency [5]. In contrast to TOP500, the Green500 list looks into an energy efficiency metric, the GFlops per Watt (GOPS/W), for computers "big" enough to be consider supercomputer-class, i.e. passing the threshold of Top500. By looking from the Green500 perspective, the current fastest supercomputer (Tianhe-2) is only 32nd by delivering 1.9 GFlops/W. In addition to power per computation, which measures only the efficiency of the computational part, the extra power costs due to the cooling infrastructure, necessary to keep devices temperatures below dangerous values, must be considered as they contribute to reduce the energy efficiency of a supercomputer. To reduce this overhead there have been a shift from air cooling toward liquid cooling.

978-3-9815370-2-4/DATE14/ © 2014 EDAA

The Eurora Supercomputer, developed by Eurotech and Cineca [4] is today the most energy efficient supercomputer in the world. In July 2013, Eurora ranked first in the Green500 list, achieving 3.2 GFlops/W on the Linpack Benchmark with a peak power consumption of 30.7 KW, and improving by almost 30% the performance of the greenest supercomputer in the world. Eurora has been supported by PRACE 2IP project [16] and it will serve as testbed for next generation Tier-0 system. Its outstanding performance is achieved by adopting a direct liquid cooling solution and a heterogeneous architecture with best-in-class general purpose HW components (Intel® Xeon E5, Intel® Xeon Phi and NVIDIA Kepler K20). Eurora cooling solution is highly efficient and enables hot water cooling, that is suitable for hot water recycling and free-cooling solutions [9]. For its characteristics Eurora is a perfect vehicle for testing and characterizing next generation "greener" supercomputers.

Today's supercomputer cooling infrastructures are designed to sustain the peak power consumption. However, during everyday activity the machine load hardly reaches the 100% utilization and the workloads submitted by the users are characterized by different computational requirements [18]. This turns into cooling infrastructure over-design. In addition, operating systems are today capable of adapting the cores frequency according to the cores load to save energy. This augment the power consumption variability.

To reduce overheads induced by cooling over-provisioning, several works in the state-of-the-art propose to optimize the job dispatching and the frequency mapping taking advantages of non-uniformity in thermal and power evolutions [17], [11], [10], [9], [2]. Unfortunately, most of them rely on simulations and modeling assumptions and are not mature enough to execute safely in a supercomputer in production. For these reasons, a careful characterization is needed to highlight real thermal and power behaviors for upcoming next-generation supercomputers based on heterogeneous architectures and ultra-efficient liquid cooling.

Today's general purpose HW components feature built-in HW monitors [8], [14] directly accessible from the software stack and capable to track on-line the status and activity of the underlining HW. System administrators use a portion of these sensors to monitor and store the supercomputer usage for detecting failures and planning maintenance. IT monitoring tools are optimized for on-line data visualizations and traces storage, but usually disregard low-overhead fine-timescale sampling of large sets of HW sensors and off-line machine assisted post-processing [12], [1].

In this paper we first present a novel monitoring and data collection framework specifically designed for finely sampling all the Eurora HW sensors (load, frequency, power and temperature) of all the primary HW components (i.e. CPUs, GPUs, MICs) ensuring low-overhead and without perturbing the Eu-

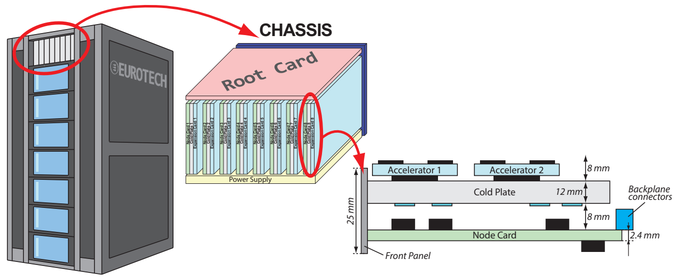


Fig. 1. Eurora architecture

rora workload and behavior. Second, we use the developed framework to characterize the power and thermal peculiarities of the Eurora platform under production workloads. The proposed framework is 100% compatible with the IT infrastructure of a supercomputer center in production.

In the remainder of the paper, Section II presents an overview of the Eurora Platform, Section III provides the details of our measurement infrastructure, and Section IV shows the results of our analysis. Finally, the conclusions are reported in Section V.

## II. Eurora Architecture

The biggest building block of the Eurora prototype is the rack. Each rack consists of up to 16 chassis, which host 16 node cards or 8 node cards plus 16 expansion cards (two per node), a root card, a backplane, liquid distribution pipes, and a touch screen panel for monitoring and control.

The Eurora system consists of a half-rack containing 8 stacked chassis, each of them designed to host 8 node cards and 16 expansion cards (see Fig. 1). The node card is the basic element of the system and consists of 2 Intel Xeon E5 series (SandyBridge) processors and 2 expansion cards configured to host an accelerator module. One half of the nodes use E5-2658 processors including 8 cores with 2.0 GHz clock speed (Max Turbo Frequency 2.8 GHz), 20 MB caches, and 95 W maximum TDP. The rest of the nodes use E5-2687W processors including 8 cores with 3.1 GHz clock speed (Max Turbo Frequency 3.8 GHz), 20 MB caches, and 150 W maximum TDP. 58 nodes have 16 GB of ECC DDR3 1.6 GHz, and a 160 GB SSD non volatile memory. The remaining 6 (with processors at 3 GHz clock rate) have 32 GB RAM. The accelerator modules can be Nvidia Tesla (Kepler) with up to 24 GB of GDR5 RAM and up to 2 TFlop peak DP and 250 W TDP, or, alternatively, Intel MIC KNC with up to 16 GB of GDR5 RAM and up to 1.4 TFlop peak DP and 245 W TDP.

Each node of Eurora currently executes a SMP CentOS Linux distribution version 6.3. The kernel is configured with NOHZ function disabled, hiperthreading HW support disabled and on-demand power governor [15]. These are common setting for high-performance supercomputers. Eurora interfaces with the outside world through two dedicated computing nodes, physically positioned outside the Eurora rack. The *login node* connect Eurora to the users. This node executes the batch job dispatcher and connects to the same shared file system visible directly accessible from all the computing nodes. The *master node* instead is connected to all the root cards and it is visible only to system administrators. In addition, both the *login node* and *master node* are connected to a MySQL server.

With respect to networking, Eurora adopted the Unified Network Architecture that consists of 3 different networks working together on the same machine: 2 fast independent networks (InfiniBand, 3D Torus) and a multi-level synchronization network. Depending on the executing application, the 3D torus is indicated for parallel user applications, the InfiniBand for I/O traffic and for connecting Eurora to the outside Storage Area Network (SAN). The latter synchronizes the CPUs reducing the OS jitter.

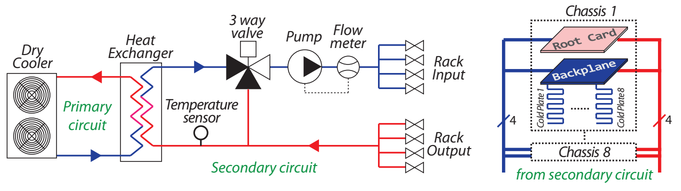


Fig. 2. Eurora simplified cooling circuit.

As its precursor Aurora, Eurora adopted a hot liquid cooling technology, i.e. the water inside the system can reach up to $50^\circ C$. This strongly reduces the cooling energy required for operating the system, since no power is used for actively cooling down the water, and the waste-heat can be recovered as energy source for other applications. A Siemens S7 PLC supervises the water circuit, shown in Fig. 2. It regulates the flow rate of the pump that pushes the water inside Eurora in order to avoid overpressure ($1\ bar \leq$ pressure $\leq 8\,bar$) and keep the temperature gap between the inlet and outlet water in the rack below $5^\circ C$. Moreover, it manages the opening of a three-way valve that mixes the inlet and outlet liquid in order to keep the temperature of the water close to a set value (greater than $18^\circ C$ to avoid condensation, and lower than $50^\circ C$ to prevent damage to the hardware). The current Eurora deployment, due to its prototyping status does not take advantage of free-cooling and the inlet water is refrigerated. Future research and testing activities on Eurora will characterize its behavior in free-cooling conditions.

The main water pipe inside the rack splits into four liquid distribution pipes that run parallel to the stacked chassis. In their tun, the pipes branch to reach each chassis according to a parallel configuration. The coolant is steered to the root cards and the cold plates that chill the computing hardware again with a parallel circuit. Finally, the water is collected and steered to the outlet pipes.

## III. Low-Level Monitoring Framework

In this section we describe the proposed framework for monitoring with low-overhead and fine time resolution the Eurora computational, energy and thermal performance. While the operating system and the application layers expose some of these values, key computational HW components have their own performance policies that may take decisions differently from the OS ones. As consequence of that the proposed framework relies on measurements directly obtained from HW counters that unveil behaviors that cannot be observed by high level monitoring tools.

The proposed framework is composed by a set of applications, daemons and scripts that collects at run-time the HW sensors values from the different HW components in each node. The generated traces are then off-line parsed and post-processed. Fig. 3 shows the building blocks of our solution. As previously introduced, Eurora is a heterogeneous architecture as the different nodes can contain two sets of eight cores CPUs (with nominal frequency of 2.1, 3.1 GHz)(CPU in figure) and two sets of HW accelerators: namely Intel Xeon Phi (MIC in figure) and Nvidia K20 GPU (GPU in figure).

In respect of nominal Eurora operating conditions our monitoring framework introduces five new *on-line* software components. Each of them is responsible to extract over time the sensors values from the main HW modules and save them as log *traces* in the shared filesystem repository. These software modules are time triggered, and for each node they concurrently monitor the power, the temperature and the activity of the two CPUs (cpu_stats), of the two GPUs (gpu_stats),

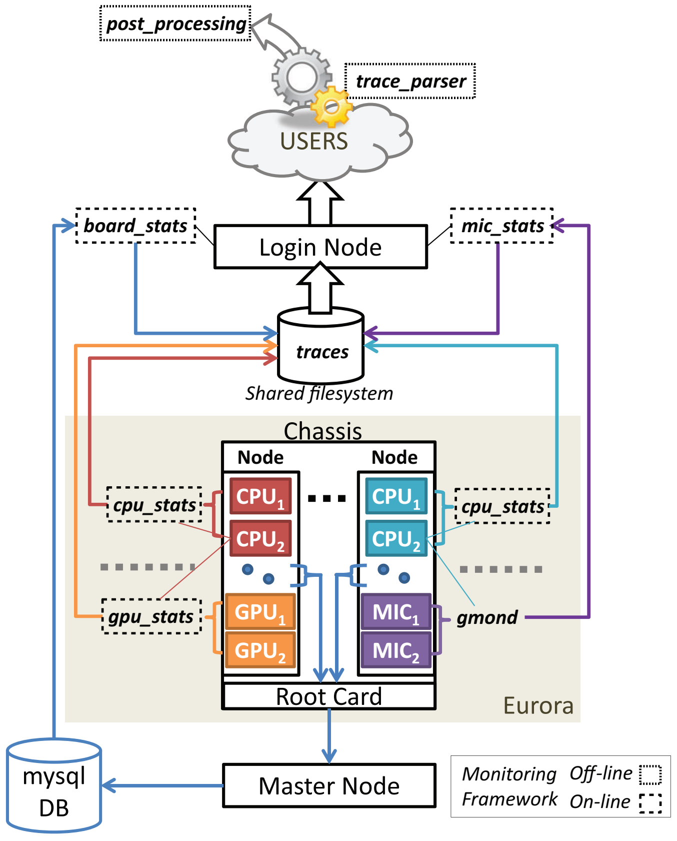


Fig. 3. Monitoring Framework

of the two Xeon Phi cards (mic_stats) and of the two board temperature sensors. In addition, we have designed two *off-line* software modules that parse and then post-process the collected data traces and enable statistical evaluation and modeling. As we will clarify in the following subsections, the software components have been designed to have negligible overhead on the computing nodes and to execute in userspace avoiding kernel modules and patches, not suitable for supercomputer in production phase. This characteristic enables our framework to work transparently during everyday supercomputer activities, unveiling real usage behaviors. In the following subsections we give more insights both on the on-line and off-line software components.

## A. On-line Node Monitoring

*1) cpu_stats:* This daemon, is written in C language, and executes with root privileges in each active node. The daemon is sequential and thus executes only in one core of the two CPUs present in each node. At each sample interval ($Ts_{CPU}$), *cpu_stats* reads the current date, opens in sequence all the *msr* device drivers (one for each active core - "/dev/cpu/coreId/msr") in the node, writes the raw data to a per-node trace file in the share filesystem (MSRnodeId), and finally sleeps until the next sample time. For each of them it reads the values of the available performance counters (UnHalted Core Cycles, Instructions Retired and UnHalted Reference Cycles) in the Performance Monitoring Unit (PMU), of the core temperature sensors, and of the time-step counter. Moreover, it reads the monitoring counter (power unit, core energy, dram energy, package energy and package temperature) present in the Intel Running Average Power Limit (RAPL) interface, respectively of the core 0 and 8 that belong to the two different physical CPUs. Table I shows for each counter the name and the quantity probed. Data are collected at the

TABLE I. PMU AND RAPL SENSOR LIST

| *PMU Sensor* | *Description* | *Size [bit]* |
|---|---|---|
| **UnHalted Core Cycles** | Counts the active cycles expired at the current applied frequency | 48 |
| **Instructions Retired** | Counts the number of instruction retired | 48 |
| **UnHalted Reference Cycles** | Counts the active cycles expired at the reference frequency | 48 |
| **Time Step Counter** | Counts all (including idle) the cycles expired at the reference frequency | 64 |
| **Digital Temperature Sensor** | Calibrated temperature sensors with accuracy of $\pm 1°C$ | 7 |
| *RAPL Sensor* | *Description* | *Size [bit]* |
| **power unit** | Energy counter resolution | 4 |
| **core energy** | Integrates the core energy consumption | 32 |
| **dram energy** | Integrates the dram energy consumption | 32 |
| **package energy** | Integrates the full-chip energy consumption | 32 |
| **package temperature** | Hottest core temperature | 7 |

beginning of each sample period, then all the *msr* are read in sequence for each core ($16cores \times (4msr\_read+1tsc\_read)+2CPUs \times 4msr\_read$) and only at the end of this operation all the values are written to the per node CPU trace file. This reduces the jitter between the different cores measurements. The values extracted allow to compute the following run-time per-core metrics: clocks per instruction, core load, core real frequency, core temperature. These metrics are generated off-line by the *trace-parser* basing on the raw trace files. Counter overflow detection and correction is offloaded to the off-line *trace-parser*. Every $Ts_{CPU}$, the *cpu_stats* daemon has an average execution time overhead of 13 ms on a single core out of 16 available for each node. We choose $Ts_{CPU} = 5s$ in the data reported in the experimental results Section.

*2) gpu_stats:* This daemon, written in python, executes with normal user privilege in each active node. The daemon uses the python bindings for the NVIDIA management library (pyNVLM). At each sample interval ($Ts_{GPU}$), the *gpu_stats* reads first the current date, then the GPU temperature, the GPU and Memory utilization, the GPU power consumption and the Graphics (Shader), Streaming Multiprocessor and Memory clock frequency in sequence for all the GPU cards present in the node. Once all the values are collected, it writes them to the per node GPU trace file (GPUnodeId) and sleeps until the next sampling interval. Every $Ts_{GPU}$, the *gpu_stats* daemon has an average execution time overhead of 23 ms, on a single core out of 16 available for each node. We choose a $Ts_{GPU} = 5s$ in the data reported in the experimental results Section.

*3) mic_stats:* Differently from other HW components the Intel Xeon Phi co-processors do not feature all the HW sensors directly available from an open driver or a management library. It has some management utilities that runs on the host operating system mapped to a small subset of sensors. It instead exports all the HW sensors through a proprietary Ganglia plugin[1]. As consequence of that, and differently from previous daemons, we designed the *mic_stats* to periodically (every $Ts_{MIC}$) read the Ganglia telnet xlm stream and extracts the entry related to the only Xeon Phi Cards. For each Xeon Phi Card found the *mic_stats* searches for the valid sensor name from a predefined *sensor_to_monitor* list and extracts the related numerical values. Then, it orders them consistently with the *sensor_to_monitor* list and print them to a unique MIC text trace file in the shared filesystem repository. Table II shows the monitored sensors. This daemon, written in bash and python, executes in the login node. Differently from *cpu_stats* and *gpu_stats* daemons that execute N (N=#active nodes) concurrent instances of the same daemon on each

[1]Ganglia is an open source scalable distributed monitoring system for high-performance computing systems.

TABLE II. INTEL® XEON PHI SENSOR LIST

| *Sensor Type* | *Value* | | |
|---|---|---|---|
| **Temperature** | Core Rail | Memory Rail | Uncore Rail |
| | Fan Inlet Die | Fan Outlet GDDR | Card |
| **Power** | 2x3 Plug | 2x4 Plug | PCIE |
| | Memory Rail Internal Supply | Core Rail Total Consumption | Uncore Rail |
| **Load** | Total Number of running processes | Total number of CPUs | One minute load average |
| | Bytes in per second | Bytes out per second | Cached memory |
| **DVFS** | CPU Speed | Memory Frequency | Core Voltage |

active node and stream the sensors output to the per node traces in the shared file system, the *mic_stats* is centralized and runs on the login node. Nevertheless *mic_stats* is only a collector point of values sent from the Ganglia monitor daemon *gmond* that is distributed across all computing nodes. It must be also considered that *mic_stats* overhead does not impact the Eurora overall performance as it executes in the login node that is not performance constrained. The *mic_stats* daemon has an average execution time overhead of 2.010 s every $Ts_{MIC}$. This overhead is compensate within the daemon code to really sampling the MIC sensors every $Ts_{MIC}$. We choose $Ts_{MIC} = 5s$ in the data reported in the experimental results Section.

*4) board_stats:* As node temperature sensors are only available from each chassis *root-card*, and root cards are visible only from the *master node* for security reasons, we have decided to use a MySQL database as collector point. In the *master node* a daemon executes which periodically (each $Ts_{BOARD}$) reads for all the nodes in a chassis the two board temperature sensors and writes these values to a dedicated table in the database. Then, the *board_stats* periodically executes on the *login node* and copies the new entries to a per node board sensor trace. The MySQL database is also used in parallel for thermal emergency monitoring from the administrator of CINECA. The *board_stats* daemon has an average execution time overhead of 2 s for reading all the board sensors for all the nodes every $Ts_{BOARD}$. We choose $Ts_{BOARD} = 5s$ in the data reported in the experimental results Section.

### B. Off-line Data Post-Processing

*1) trace_parser:* When triggered by the user, the off-line trace parser reads the traces collected in the shared filesystem repository and converts them to a set of intermediate trace files that are more suitable to be loaded and processed by higher level processing tools. The operation that are performed are:

- For each CPU trace (one for each active node), the *trace_parser* reads the file line-by-line and writes the content in two files: the first contains only the RAPL related metrics for the two CPUs, the second contains only the MSR related metrics for the total 16 cores. The RAPL raw measurements contain energy counters values. The parser computes the increment in energy from previous reads, detects and compensates possible overflow, computes the power by dividing the increment by the number of cycles counted from the previous measurement, and saves them to the intermediate file. For each line the parser returns the average power per sampled period ($Ts_{CPU}$) of the package, the core, and the DRAM of the two CPUs. The MSR raw measurements contain the absolute values for the PMU fix counters. The *trace_parser* reads these values line by line and computes the increments with respect to previous reads. In this phase, counter overflows are compensated. From the increments, the parser computes the following output metrics: clocks per instruction, core load, core real frequency, core temperature.
- The *trace_parser* reads the MIC trace by each line and splits the content in several intermediate files, one for each active MIC node.
- The *trace_parser* reads the Board trace by each line and splits the content in several intermediate files, one for each active node.
- GPU traces (one for each active GPU-node) are not parsed as they contain already the final metrics.

*2) post_processing:* The post-processing exploration task has been executed in Matlab [13] as it enables high level and powerful data management, plots and statistical toolboxes. The Matlab script first loads the traces from text files and then generates well structured ".mat" files that contains the computed metrics for each node and for each computational engine (CPU,GPU,MIC). Then, it directly uses the ".mat" files for computing the Eurora Characterization.

## IV. EXPERIMENTAL RESULTS

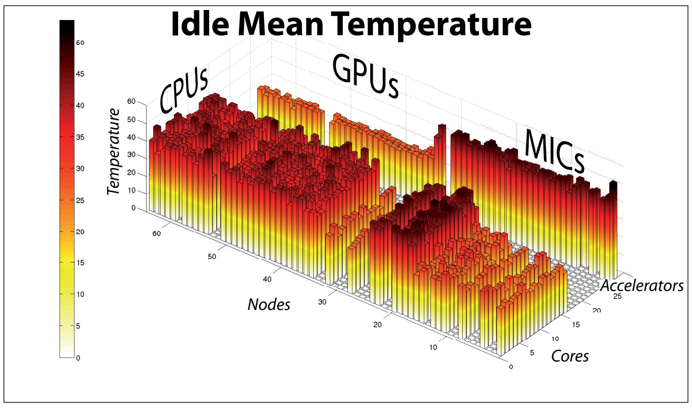
In this section we show a characterization of the overall Eurora Thermal/Power performance in a production workload. We obtained the characterization data by means of the framework described in Section IV. All the data presented refer to a sampling window of two days of normal Eurora production activity. For each node, the dataset is composed by a sets of structures, one for each of the following elements: CPU, MIC, GPU, Board. Each of these structure contains a set of timing traces. Each time traces is a vector representing a sampled physical quantity/metric collected by the framework. Within each structure, vectors are aligned and time consistent. Missing node entries are due to off-line nodes in the sampled windows. During measurements inlet cooling water is set to $24°C$ and outlet-inlet temperature difference is set to $5°C$.

We conducted three main explorations on the extracted data. The first one analyzes the spatial efficiency of the Eurora thermal dissipation/cooling system. The second one evaluates the worst-case thermal conditions, these are evaluated as maximum thermal gradients and peak temperatures. Finally, the third exploration evaluates how the Eurora computational to power efficiency changes with the workload.

### A. Eurora Cooling Efficiency

To evaluate the spatial difference in the Eurora cooling efficiency, for each node and for each structure (CPU, MIC, GPU), we have extracted from the dataset the trace portion that refers to zero load activity ($< 1\%$). On this data we computed the per node and per structure average. In Fig. 4 we plot the average temperature and average power. In both the plots on the x-axes is reported the nodeId while in the y-axes is reported the HW component. We have two sets of HW components the cores, and the HW accelerators. The cores are indexed from 1 to 16. Cores in the 1 to 8 range refers to one physical CPU whereas cores in range 9 to 16 refers to the second CPU. Along the nodes we have three different Intel Xeon E5 processors stepping: nodes 1-16 & 25-32 have a maximum frequency of 2.1GHz (CPUs-2100), nodes 17-24 have a maximum frequency of 2.2GHz (CPUs-2200), and nodes 33-64 have a maximum frequency of 3.1GHz (CPUs-3100). For the accelerators we have two entries since we have two accelerators per node. Nodes from 0 to 32 embed Intel Xeon Phi accelerators, whereas nodes from 33 to 64 embed Nvidia Kepler GPUs. The plot on the left shows on the z-axes the absolute temperature with linear scale, while the plot on the right shows on the same axes the total power consumption, using a logarithmic scale.

From the power plot (on the right) we can notice that within each class of computing engine (CPUs, MICs, and GPUs) the



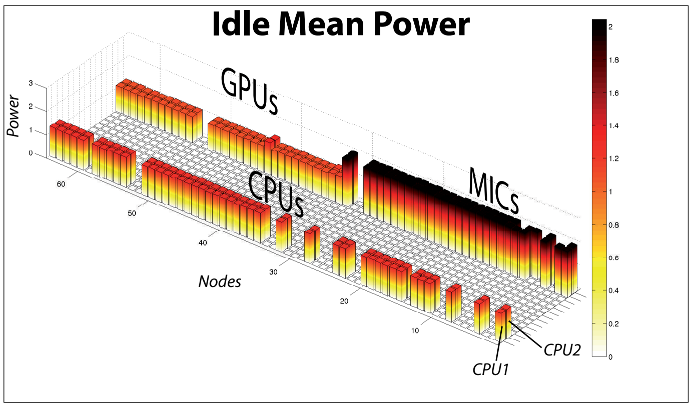


Fig. 4. CPUs, GPUs, MICs mean temperature (a) and power (b) during idle phases

idle power consumption is almost constant, whereas CPUs and GPUs have similar idle power consumption $\approx 10W$, MICs have significantly higher idle power consumption $\approx 100W$. We must notice that within each processing element class the idle power consumption is homogeneous. The picture changes if we look at the die temperature plot (on the left). While the MICs have higher idle temperatures than the GPUs and this is due to the different idle power consumptions, the cooling efficiency for the CPUs varies between the different nodes. Indeed, we can notice that for the same idle power consumption the nodes with CPUs-2100 have the lowest temperatures, while the highest ones are with CPUs-2200. By correlating this plot with the power plot we can notice that this is not caused by external heat interference as the coldest CPUs-2200 are the one positioned in the same node with the MIC accelerator, that is hotter than the GPUs. This leads to the following considerations valid for minimum low power consumption ranges. First, there is no thermal interference between the different computational engines. Second, within the same die, cores have similar thermal efficiency. Third, the nodes have different thermal efficiency accordingly to the processor stepping. This could be explained by a different package form factor or a different manufacturing yield of the cold plate. This last point justify the benefits of thermal-aware job-dispatching optimizations.

## B. *Eurora Worst-Case Temperature*

This subsection shows the Eurora worst working conditions from both temperature and power perspectives.

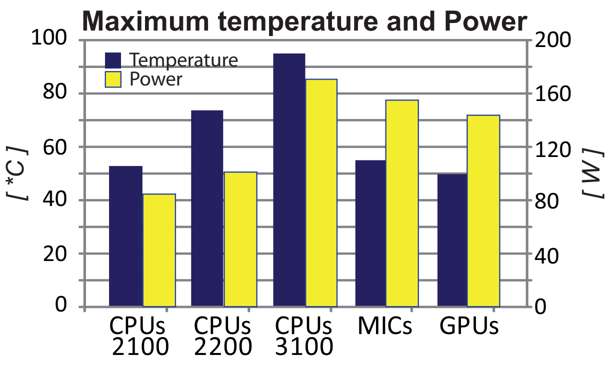


Fig. 5. CPUs, GPUs, MICs, Boards maximum absolute temperature

Fig. 5 shows on the x-axes the different classes of computing engine and in the y-axes reports both the maximum registered power consumption and temperature. From the figure we can notice that the peak powers and the peak temperatures vary significantly among the devices as well as the cooling efficiency (peak temperature over peak power). CPUs-3100 has almost the same peek power consumption of MICs, but almost the double of the die temperature. MICs and GPUs behave similarly and remain significantly colder than CPUs that can reach more than $90°C$ of peak temperature. The worst registered board temperature is almost $20°C$ higher than the inlet water temperature.

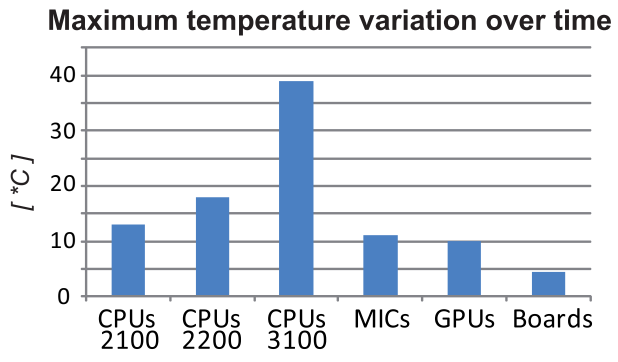


Fig. 6. CPUs, GPUs, MICs, Boards maximum temporal thermal gradient

In addition to peak temperatures we computed the maximum variation of temperature between two consecutive samples, that means 5 s for CPUs, MICs, GPUs and 60s for Boards. In Fig. 6 we show the maximum registered values for each class of computing engine. We can notice that these strongly depends on the computing engine class considered and are correlated with the peak power consumption. The maximum value registered is for the CPUs-3100 that may have almost $40°C$ of temporal thermal gradient. This justifies the benefits of fast dynamic thermal-aware control algorithms.

## C. *Eurora Power Efficiency*

The final test we have performed evaluates the Eurora power efficiency with respect to different load conditions. We focused this analysis only on the CPUs class of computing engine. Fig. 7a shows on the x-axes the average load on the different cores of a CPU, while on the y-axes shows the CPU power consumption. The plot shows that, as the load increases, the power consumption increases as well, but differently for each CPU stepping. Within each CPU stepping family, the power shows significant variations. This can either be due to workload properties as well as physical differences. At full load the CPU families consume significantly different. This is due to the different maximum frequency. We can also notice that the

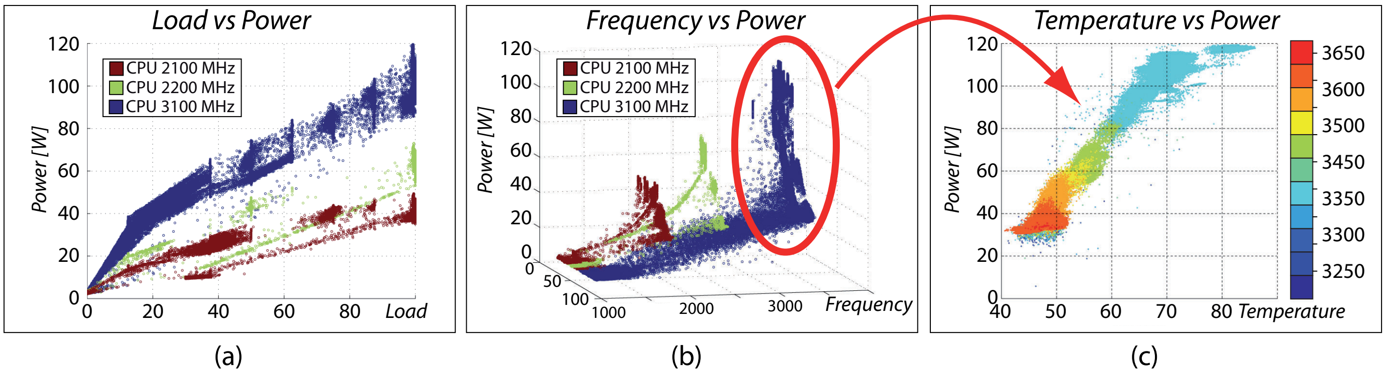


Fig. 7. CPUs, power efficiency (versus load)

power consumption saturates at higher loads. Fig. 7b shows, for the same results, on the x-axes the average real frequency within each core of the node, on the y-axes the nodeId and on the z-axes the total power consumption. We can notice that power consumption increases with the frequency and the peak power consumption depends on the CPUs stepping. Moreover, both the CPUs-2200 and CPUs-3100 are often in turbo mode as their average real frequency is higher than the maximum one. During these turbo mode periods their power consumption has an higher variability than when frequencies are lower. This is explained in Fig. 7c where we plot on the x-axes the average temperature of the cores in the same CPU and on the y-axes the CPU power consumption. This plot contains only the values for CPUs-3100 that are running at a frequency higher than 3.2GHz. We can notice a strong correlation between temperatures and power consumptions. But surprisingly we can notice that higher power values happen at lower frequencies, while higher frequencies are located at the lowest powers. This can be explained considering the turbo mode internal logic that selects higher frequencies when less cores are active in the CPU. From the same plot it must be noted that for a large portion of the trace (frequency @3350$MHz$) the turbo policy selects a lower frequency with respect to the maximum allowed, even if the CPU is far from the maximum thermal limits and TDP budget. This seems to suggest that the internal turbo logic is pessimistic and there are opportunities to improve its behave for systems with highly efficient cooling.

## V. Conclusion

In this paper we have presented a low-overhead framework for monitoring and collecting data to unveil the power-thermal trade-offs and peculiarity of Eurora, the "greenest" supercomputer in the world. Our monitoring framework enables fine samplings for all the Eurora HW sensors (load, frequency, power and temperature) and for all the primary HW components (i.e. CPUs, GPUs, MICs) with low-overhead and without perturbing the Eurora workload and behavior. The proposed framework is 100% compatible with the IT infrastructure of a supercomputer center in production. We then used the developed framework to characterize the power and thermal peculiarities of the Eurora platform under production workloads. The conducted analysis evaluate the Eurora cooling efficiency, the worst case operating conditions, and, finally, its power efficiency. Results of these analysis show nice opportunities for run-time energy/thermal optimizations at all the software levels (i.e. thermal aware job dispatching, dynamic frequency control and turbo mode).

## Acknowledgments

This work was partially supported by the FP7 ERC Advance project MULTITHERMAN (g.a. 291125) and by the PRACE 2IP EU project (g.a. RI-283493).

## References

[1] Zabbix: An enterprise-class open source distributed monitoring, May 2013.

[2] 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*, 24(1):170–183, 2013.

[3] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, and R. S. W. K. Yelick. Exascale computing study: Technology challenges in achieving exascale systems. Technical report, 09 2008.

[4] C. Cavazzoni. Eurora: a european architecture toward exascale. In *Proceedings of the Future HPC Systems: the Challenges of Power-Constrained Performance*, page 1. ACM, 2012.

[5] W. chun Feng and K. Cameron. The green500 list: Encouraging sustainable supercomputing. *Computer*, 40(12):50–55, 2007.

[6] J. Dongarra. Visit to the National University for Defense Technology Changsha, China. Technical report, University of Tennessee, 06 2013.

[7] J. J. Dongarra, H. W. Meuer, E. Strohmaier, et al. Top500 supercomputer sites. *Supercomputer*, 13:89–111, 1997.

[8] Intel Corporation. *Intel® 64 and IA-32 Architectures Software Developers Manual*. Number 325462-046. March 2013.

[9] J. Kim, M. Ruggiero, and D. Atienza. Free cooling-aware dynamic power management for green datacenters. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 140–146, 2012.

[10] J. Kim, M. Ruggiero, D. Atienza, and M. Lederberger. Correlation-aware virtual machine allocation for energy-efficient datacenters. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, pages 1345–1350, San Jose, CA, USA, 2013. EDA Consortium.

[11] D. Kudithipudi, Q. Qu, and A. Coskun. Thermal management in many core systems. In S. U. Khan, J. Koodziej, J. Li, and A. Y. Zomaya, editors, *Evolutionary Based Solutions for Green Computing*, volume 432 of *Studies in Computational Intelligence*, pages 161–185. Springer Berlin Heidelberg, 2013.

[12] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.

[13] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[14] Nvidia Corporation. *NVML API REFERENCE MANUAL*. Number 5.319.43. August 2013.

[15] V. Pallipadi and A. Starikovskiy. The ondemand governor. In *Proceedings of the Linux Symposium*, volume 2, pages 215–230. sn, 2006.

[16] PRACE. Partnership for Advance Computing in Europe.

[17] M. Sabry, A. Sridhar, J. Meng, A. Coskun, and D. Atienza. Greencool: An energy-efficient liquid cooling design technique for 3-d mpsocs via channel width modulation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(4):524–537, 2013.

[18] H. You and H. Zhang. Comprehensive workload analysis and modeling of a petascale supercomputer. In *Job Scheduling Strategies for Parallel Processing*, pages 253–271. Springer, 2013.