Power-Aware Job Scheduling on Heterogeneous Multicore Architectures

Matteo Chiesi, Luca Vanzolini, Claudio Mucci, Eleonora Franchi Scarselli, *Member, IEEE*, and Roberto Guerrieri

Abstract—This paper presents a power-aware scheduling algorithm based on efficient distribution of the computing workload to the resources on heterogeneous CPU-GPU architectures. The scheduler manages the resources of several computing nodes with a view to reducing the peak power. The algorithm can be used in concert with adjustable power state software services in order to further reduce the computing cost during high demand periods. Although our study relies on GPU workloads, the approach can be extended to other heterogeneous computer architectures. The algorithm has been implemented in a real CPU-GPU heterogeneous system. Experiments prove that the approach presented reduces peak power by 10 percent compared to a system without any power-aware policy and by up to 24 percent with respect to the worst case scenario with an execution time increase in the range of 2 percent. This leads to a reduction in the system and service costs.

Index Terms—Power management, power measurement, multi-GPU, scheduling, power capping, prediction

1 Introduction

The computational power required by scientific, engineering and financial applications is unattainable on today's most advanced multi-core CPUs [1]. For this reason, GPUs have been proposed as accelerators for intensive workloads in large scale supercomputers [2], leading to the development of powerful heterogeneous high performance computing systems (HPC). For example the new supercomputer Titan, developed by ORNL [3], uses a heterogeneous architecture coupling conventional 16-core CPUs and GPU accelerators.

In such a scenario power has become the most critical issue. These systems can reach a peak power consumption of tens of MW and in 2012 the supply cost over their useful life exceeded the initial capital investment [4]. In addition to supply cost abatement and positive environmental implications, the limitation of the worst case power scenario leads to a cost saving due to the lower complexity and capacity of the cooling systems needed. Traditionally, supercomputers were designed to sustain the worst case operating condition. However this scenario is very rare and oversized power supply and cooling systems involve additional costs. Thus in order to hold down the costs, supercomputers are nowadays designed with a better-than-worst-case policy [5]. In this situation the power consumption is constantly monitored and if the operating condition overlaps the predetermined power threshold, the power budget required by each node is adjusted to run safely under the maximum physical limitation.

Power capping defined as a strategy to limit peak power under a predetermined threshold is strongly influenced by the jobs activated on the computing system nodes. Hence techniques are needed which allow one to dynamically control the peak power while keeping system performance as high as possible [6]. In particular simultaneous execution of jobs (concurrency) leads to a performance enhancement effect, but also to an increase in power consumption. On the contrary, when concurrency is decreased, both performance and power consumption decrease.

In this framework, this paper presents a job-level scheduling algorithm that aims to limit the worst case power condition below a predetermined budget during the concurrent execution of jobs in a heterogeneous computing system coupling CPU cores and GPU accelerators. The need for power-saving policies allowing control of power consumption, depending on the jobs being activated on the nodes, has already been recognized [7]. The open challenge is to find an effective way to reduce peak power while keeping concurrent execution of jobs as high as possible. The paper is organized as follows. The rest of Section 1 discusses related work and the contributions of this paper. Section 2 describes the power measuring system. Section 3 discusses the scheduling algorithm herein proposed. Performance evaluation is presented in Section 4. Section 5 contains the discussion and some conclusions are drawn in Section 6.

1.1 Related Work

Several studies have been carried out addressing the need to limit power consumption. Some of them exploit dynamic voltage and frequency scaling (DVFS) in order to achieve a power reduction in HPC systems [8]. In [9] the approach presented was to reduce the clock frequency on nodes which had been assigned small computation load. The algorithm

- *M. Chiesi, E. Franchi Scarselli, and R. Guerrieri are with the Advanced Research Center on Electronic Systems for Information and Communication Technologies E. De Castro (ARCES), University of Bologna, viale Carlo Pepoli 3/2, 40123 Bologna, Italy. E-mail: {mchiesi, efranchi, rguerrieri}@arces.unibo.it.*
- *L. Vanzolini and C. Mucci are with STMicroelectronics, Agrate Brianza, (MB), Italy. E-mail: {luca.vanzolini, claudio.mucci}@st.com.*

Manuscript received 23 Sept. 2013; revised 13 Feb. 2014; accepted 28 Mar. 2014. Date of publication 1 Apr. 2014; date of current version 6 Feb. 2015. Recommended for acceptance by M. Guo.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2014.2315203

developed in [10] presents a power-aware DVFS run-time system that performs power reduction with small performance loss. Another work [11] provides a power-aware scheduling algorithm for applications with deadline constraint. In this approach DVFS is used to minimize power consumption meeting the deadline specified by users. However none of these approaches are designed to keep the power consumption under a preset threshold. Other methods exploit DVFS in order to keep the maximum power lower than a predetermined power constraint. In [12] power is shifted between resources, observing how they are being used, while keeping the total power consumption lower than a given budget. Since frequency assignment is performed at a very fine grain, applying this approach to large scale systems could involve high overheads. In [5] the technique presented uses feedback control to keep the system within predetermined power constraints managing the CPU performance. The scheduling algorithm developed in [13] uses integer linear programming to assign a CPU frequency before executing a selected job in order to remain below the predetermined budget. Even though DVFS is common for CPU-based infrastructures [14], it is relatively new for heterogeneous systems based on GPUs. It must also be noted that scheduling algorithms based on dynamic voltage and frequency scaling deliver a suboptimal response for short-time workloads because they rely on reaction instead of prediction and for short workloads this reaction can occur after the transition. In this case the amount of performance loss is related to the number of transitions in the workload and the lag between request and capacity [15]. Approaches that aim to reduce power consumption according to the jobs being activated on the node have been explored. In order to do this, the power consumption of several library functions may be characterized for different CPU performance. For example, in [16] a power performance comparison between LAPACK [17] and PLASMA [18] libraries was made using the setup developed in [7]. The work presented in [19] uses the same measure setup [7] in order to develop a job-centric model. The purpose of these works is to understand how a program can be modified to improve performance with respect to system run-time and power consumption. In [20] a method of profile-based power-performance optimization is presented. In this work a program is split into several regions and for each one the frequency which minimizes the power-performance ratio is selected.

All of these works are based on multicore CPU and not on heterogeneous CPU-GPU architectures. In addition, most of the algorithms described in the literature do not consider the concurrent execution of several jobs on different cores as a target to be optimized in order to keep peak power under a predetermined budget. However, in heterogeneous computing systems where GPUs are the most power-consuming devices, the simultaneous execution of GPU kernels may lead to overlapping high power profiles, causing generation of power absorption peaks which could be avoided with a smart distribution of the workload to the resources.

1.2 Contributions of This Work

This paper presents a predictive power-aware scheduling algorithm which provides a real-time allocation of computationally-intensive jobs to the nodes of a heterogeneous computing system, with a view to keeping the peak power under a predetermined budget, mitigating the worst case power condition.

The basic idea behind the algorithm is to adopt a two-step approach. First, the power consumption of a GPU kernel library is characterized. Jobs activated on the system nodes utilize these kernels to accelerate intensive computational cores. From the user viewpoint this characterization does not affect the programming model at all. However, each time a new kernel is added to the library, its power consumption must be characterized. Second, this characterization is then used to develop a model capable of adjusting the start time of a job depending on its GPU kernel calls, and selecting the node on which to activate it, taking into account the jobs that are already running on the system. This approach limits peak power requirements and enables the system not to exceed the predetermined budget. This is achieved without performance reductions caused by frequency and voltage scaling as proposed in [5], since it is obtained by considering the different profiles associated with each kernel in order to avoid concurrent execution of the most power-consuming jobs on the same node. The specific contributions of this paper may be summarized as follows:

- A low-cost measurement system has been developed to extract the power profile of jobs running on heterogeneous computer architectures. This system has been designed to make up for the lack of standard hardware sensors in the computing nodes used as basic blocks of high performance systems [7].
- A power-aware scheduling algorithm to manage the resources of several computing nodes has been developed. The scheduler manages the start times and the nodes on which to run the jobs. The goal is to minimize peak power absorption (such as may happen during simultaneous execution of several jobs) while keeping concurrency as high as possible.
- A quantitative analysis has been carried out in order to demonstrate that the algorithm significantly reduces peak power requirements during parallel job execution, mitigating the worst case power condition.

2 Power Measuring System

In order to develop a job-level power model it is necessary to characterize the power consumption of the jobs performed on the system nodes. Common power profiling techniques allow one to measure the consumption of a single computer component such as a processor or a co-processor [21], [22], [23], [24]. Other setups have been proposed to perform fine grain profiling analysis of an entire system [7], [25]. However, they are based on expensive measuring instruments.

In [23] three options are discussed to measure the power consumption of a job running on heterogeneous CPU-GPU platform: (1) measuring the power consumption of the GPU card; (2) measuring the input of the power supply unit (PSU); (3) measuring the output of the power supply unit. Measuring the power consumption of the GPU card is the

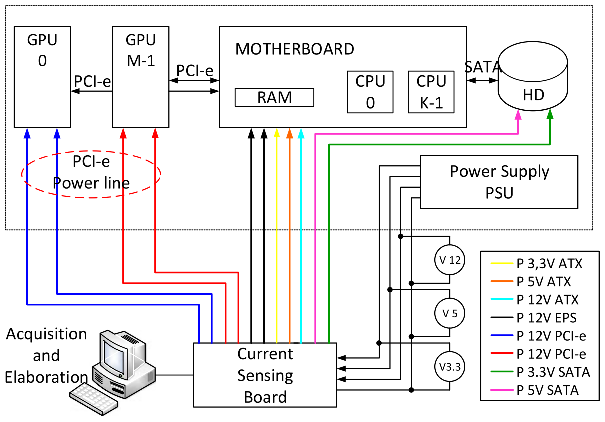


Fig. 1. Measuring setup for the generic computing node.

most direct way, although it is not precise because the GPU is a co-processor and needs a CPU as a host. Thus the power measured is only a part of the power required to run the job. Measuring the PSU input includes the power loss by the power supply unit which may be 20 percent or more of the total power dissipation [23]. However, measuring the total PSU output will not allow one to understand how each computer component contributes to the total power consumption, whereas in order to develop a job-level model one needs to measure the power consumption of each system component. Fig. 1 shows the acquisition system used in this work. It has been designed to measure the power consumption of each target system component:

- the motherboard (ATX 12 V, 5 V and 3.3 V);
- the additional power supply for CPU (EPS 12 V);
- the GPUs (PCI-e 12 V);
- the hard disk (SATA 5 V and 3.3 V).

The current-sensing board is composed of 16 Hall effect current sensors capable of measuring currents in the range 0-30 A (Allegro ACS713) and 0-50 A (Allegro ACS758). The sensor outputs provide voltage values proportional to the currents measured. Low-pass filters are connected between sensor outputs and microcontroller inputs (STM32F) in order to improve the signal-to-noise ratio. Analog-to-digital converters internal to the microcontroller sample the current data. The board is connected to a PC used to acquire data via USB. A Java interface is set up to manage data acquisition. When the connection between the board and the PC is set, the microcontroller starts to send one data packet per second. Using all 16 channels, the maximum sample rate is 1,600 samples/s for each channel. The current samples are then multiplied by the voltage values measured with a Fluke device and data are post-processed by Matlab in order to obtain the power profile of each job.

3 Power-Aware Scheduler

High performance computing relies on computing nodes equipped with multicore devices at each node and a distributed resource management system (DRMS). Users submit jobs which have to be assigned to the cores. The DRMS sorts and assigns jobs to the available resources following a scheduling policy. Hence by changing the DRMS policy it is possible to limit peak power consumption.

TABLE 1
Nomenclature Table of the Algorithm

P_C	Power Capping : predetermined power constraint
$P_{job\,(m)}$	Maximum power consumption of *mth*-job
$P_{node\,(n)}$	Power consumption of *nth*-node
$P_{idle\,(n)}$	Power consumed by the *nth*-node in idle state
N	number of nodes
M	number of cores for each node

The proposed power-aware scheduler is developed and implemented starting from two common scheduling policies (First-in-first-out and Backfilling first fit).

- *First in first out (FIFO)*. First-in-first-out is a simple scheduling strategy. New jobs that must be executed are placed at the end of the queue. When a resource becomes available the first job in the queue is activated.
- *Backfill algorithm (BFF)*. Backfill is a policy which allows the scheduler to run jobs out of arrival order. When there are not enough resources to run the first job in the queue, other jobs in the queue are checked in order to find a job that could be executed without exceeding the additional constraint imposed by the algorithm. Usually backfill allows the scheduler to start lower-priority jobs so long as they do not delay the first job in the queue. Execution is therefore limited to the resources available and the time available before the expected start time of the first job. Various backfill strategies can be used. In this work a backfill first fit strategy has been implemented: the list of feasible jobs is filtered, selecting the first one which fits the constraints.

The proposed power-aware scheduler is capable of predicting the behavior of each node, every time a new application is ready to be activated. Each job discussed in this paper is composed of a low computational part running on a CPU host, and a data-intensive computational kernel running on a GPU.

3.1 The Scheduling Algorithm

In order to simplify the design of the scheduling algorithm, three hypotheses were formulated:

- during the entire execution of a GPU kernel, the power profile is assumed constant and equal to the maximum value;
- power consumption of GPU kernels is not sensitive to changes in input data set values;
- different input data size for the same GPU kernel lead to the same peak power with different durations.

The validation of these hypothesis will be discussed in Section 4.1. Parameters used in the scheduling algorithm are defined in Table 1.

The purpose of the scheduler is to assign jobs to the available resources, limiting the maximum power consumption of each node to below the predetermined constraint (P_C) while keeping the parallelism as high as possible. P_C is set via software and can be adjusted by the user. Power consumption of all kernels comprising the

```
Input: $N; P_C; P_{node}; P_{job}$
Output: $selected\ node\ (n)$
 1: $n \leftarrow none$
 2: $P_{min} \leftarrow P_C$
 3: for $i = 0$ to $N - 1$ do
 4:     $P_{tmp} = P_C - P_{node\,(i)} - P_{job}$
 5:     if $(0 \leq P_{tmp} \leq P_{min})$ then
 6:         $P_{min} \leftarrow P_{tmp}$
 7:         $n \leftarrow i$
 8:     end if
 9: end for
10: return
```

Fig. 2. Node selection.

library is characterized a priori using the measuring system described in Section 2. Thus, the scheduler already knows the maximum power consumption of each job (P_{job}) that could be activated on the nodes. As previously mentioned, these contributions are assumed constants and equal to their maximum values. The detailed description of how the maximum power consumption of each job has been obtained is reported in Section 4.1.

Each GPU in the system is a stand-alone device running an independent job. Hence, the total power consumption of the *nth*-node during the concurrent execution of M jobs is computed by adding up the power consumed by the node in idle state (P_{idle}) and the power consumption of jobs (P_{job}) which are running on that node, as shown in (1),

$$P_{node\,(n)} = P_{idle\,(n)} + \sum_{m=0}^{M-1} P_{job\,(m)} \quad \forall n \in N. \tag{1}$$

Once the *mth*-job has been executed, the power consumption of the *nth*-node is updated as shown in (2),

$$P_{node\,(n)} = P_{node\,(n)} - P_{job\,(m)}. \tag{2}$$

When a new job needs to be activated on the system the DRMS checks if there are nodes with resources available and if these nodes will meet the power constraint when executing the job, as reported in (3),

$$P_C \leq P_{node\,(n)} + P_{job}. \tag{3}$$

Hence, if the condition shown in (3) is verified for some nodes, the scheduler assigns the application according to a minimum power-slot policy as shown in the algorithm reported in Fig. 2.

Fig. 3 explains graphically what is discussed above. The system shown is composed of four nodes ($N = 4$), each one equipped with four GPUs ($M = 4$). At time T_{start} the scheduler has to select the node on which the new job is to be activated. The first node (*NODE 1*) is already running four jobs so it has no resources available. The second node (*NODE 2*) has one GPU available. However, if the job were to be run on the node, the P_C threshold would be exceeded. The job needs to be executed on *NODE 3* or *NODE 4* if the power budget is not to be exceeded. The DRMS performs the scheduling according to a minimum power-slot policy as illustrated in the proposed algorithm. The strategy is to activate the job in the node with the smallest "power-slot" able to keep the power consumption under the predetermined threshold. This is done in order to keep the largest

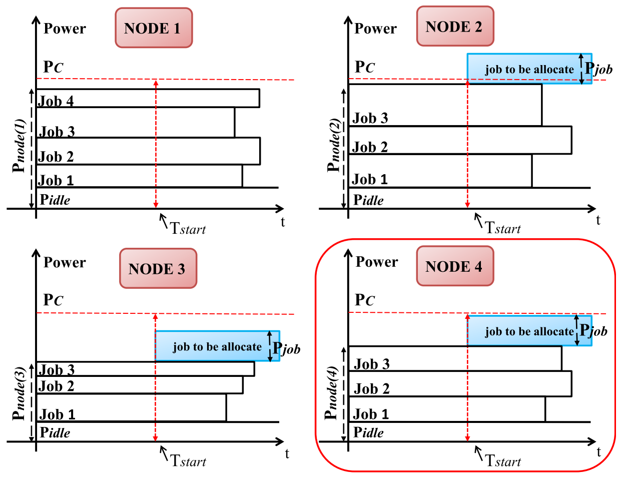


Fig. 3. Example of scheduling: four nodes, each one composed of four cores.

"power-slot" free for a more power-consuming job. In this example the job is activated on *NODE 4* while *NODE 3* is left free for a more power-consuming job. If all nodes are busy or the constraints are not met, the job will wait to be scheduled in the queue in accordance with the scheduling policy selected. Thus the scheduler manages both power and GPU as finite resources. The algorithm can easily be extended to rack level instead of node level. In this case, each time a new job is activated, the power needs to be controlled at rack enclosure level [6] instead of node level. Equation (3) can be rewritten as:

$$P_{rack} \leq \sum_{n=0}^{N-1} P_{node_{(n)}} + P_{job}. \tag{4}$$

Once the characterization of jobs has been completed, the scheduler makes its decision at run time (on-line), selecting the most suitable candidates out of the current set of tasks ready-to-run. The algorithm is non-preemptive in that the currently executing task will not be preempted until completion.

4 Performance and Evaluation

4.1 Job Characterization

Six jobs in the field of linear algebra (see Table 2) were developed, starting from the code samples available in [26], [27]. The power consumption of these jobs was characterized by changing the dimensions and the values of the input data. The configuration of the computing node used for this work is detailed as follows (see also Fig. 1 with $K = 2$ and $M = 4$):

TABLE 2
Power Consumption of GPU Jobs

Job	P_{job}
Matrix Multiplication	160 W
Matrix Multiplication (cublas)	220 W
Eigenvalues	170 W
Triangular Matrix Inversion	190 W
Matrix Transpose	180 W
Scalar Product	110 W

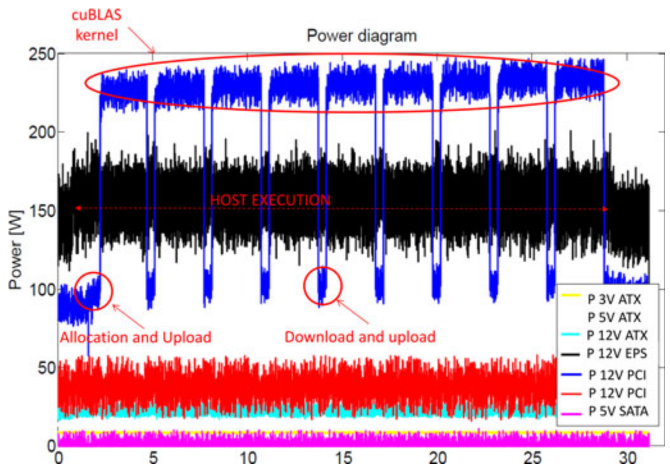


Fig. 4. Power profile measured during a matrix multiplication on GPU.

- Motherboard SuperMicro X8DTG-QF;
- Two Intel Xeon E5520 CPUs @ 2.27 GHz;
- 24 GB RAM;
- Two NVIDIA GeForce GTX 590 (4 GPUs).

The contributions measured in order to obtain an estimation of the power consumption during the computation of a job are:

- the consumption of the motherboard (ATX);
- the additional power supply for CPU (EPS);
- the power of the GPUs (PCI-e);
- the consumption of the hard disks (SATA).

The idle power consumption of the computing node (P_{idle}) is 240 W. Each job was characterized in the same operating conditions and during its execution no competing tasks were performed.

The heterogeneous programing model supported by GPU implies a system composed of a host (CPU) and a GPU each with their own separate memory. Kernels operate out of GPU memory, so the run-time provides functions to allocate, deallocate and copy GPU memory, as well as transfer data between host memory and GPU memory [26]. This architecture is reflected in the power profile of each job. A small increase in the job power consumption will be detected during data upload and download. However the most time- and power-consuming phase is the kernel execution. Fig. 4 shows an example of job power profile obtained with the monitoring system developed.

The figure shows a characterization of a matrix multiplication performed by multiplying two input matrices comprising 30,720 × 30,720 elements. Since the GPU memory has limited space, the computation was carried out by decomposing the matrix into nine different sub-matrices of dimensions 10,240 × 10,240. Two main contributions can be observed: the black line which shows the consumption of the motherboard and the blue line which represents the power supply of a GPU. As shown in Fig. 4, a GPU job starts with allocation of the GPU memory and copying of data from host to GPU (contribution indicated in Fig. 4 with *allocation and upload*). Once the data have been allocated on the GPU memory, the kernel execution starts. This phase always coincides with the highest power in the job (contribution indicated in Fig. 4 with *cuBLAS kernel*). After computation of a kernel, results are copied from device to host memory and new input data are uploaded on the device memory (contribution indicated in Fig. 4 with *download and upload*). Once the job is finished, the device memory is released. The other contributions reported in Fig. 4 (e.g., SATA) are negligible. All the jobs studied in this work, performed with different sizes of input matrix and different input data set values, follow the trend discussed above and shown in Fig. 4. The characterization proves the hypothesis discussed in Section 3. The power profile of a kernel is not sensitive to changes in the input data set values. In addition, experiments demonstrate that different data sizes for the same kernel lead to power profiles with very similar power peaks which can be approximated to the same peak value. The different duration of these kernels depends on the computational complexity of the kernel selected. A summary of the GPU job power consumption is reported in Table 2. The table shows that the jobs differ considerably in power consumption value. These values have been obtained by subtracting the idle power consumption of the GPU from its maximum power consumption during the execution of the job.

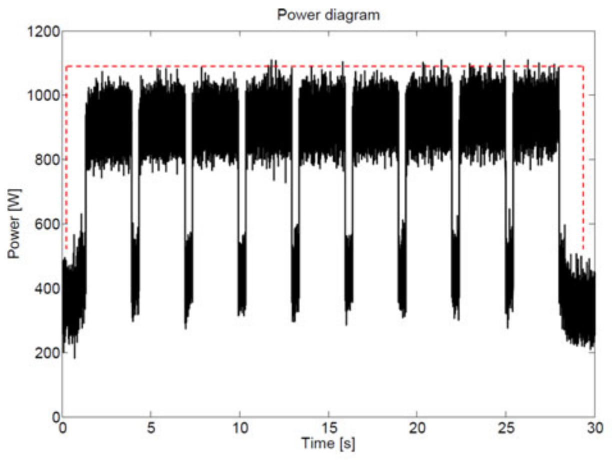
As previously described in the algorithm, characterization of jobs which can be activated on the target system allows the scheduler to predict what the power consumption will be, knowing which jobs are currently running on the node. An example of what has been discussed above is shown in Fig. 5. The consumption profile is obtained by considering four concurrent executions of the previously characterized matrix multiplication. Fig. 5a shows the prediction obtained by adding the profile previously characterized, while Fig. 5b shows the consumption measured during run-time. Substituting the values in (1), the estimated peak power consumption is computed,

$$P_{node\,(n)} = \left(240 + \sum_{m=0}^{3} 220\right) W = 1120\ W.$$

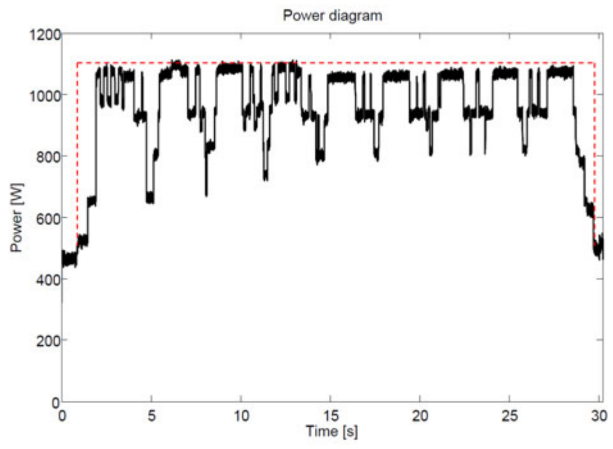
The dashed lines at the top of the two profiles show the approximation introduced in the algorithm (i.e., power constant and equal to the maximum power value). The two profiles show the same trend. The fluctuations are more evident in the theoretical case due to the fact that the plot represents the sum of four identical contributions, so that noise components are visibly amplified.

4.2 Experimental Setup

The scheduler was evaluated on four computing nodes ($N = 4$) equal to that described in Section 4.1. Hence the total number of GPUs used is 16. The algorithm was tested for generating, executing and measuring 10 workloads of 1,000 job requests selected from the previously characterized jobs. In order to create the workload, a Markov chain model was used [28], [29]. Each job requires 1 GPU and it is assumed that there is no data dependence between any jobs. Each measurement was ended after all jobs had finished. The workloads were generated so as to have more concurrent jobs needing to be activated than resources available. This was done in order to verify the performance of the algorithm during high-demand periods.



(a) Theoretical



(b) Experimental

Fig. 5. Comparison between theoretical and experimental evaluation of total power absorption during four concurrent matrix multiplications.

Power profiles obtained using the proposed power-aware scheduling algorithm were compared with the results obtained by executing and measuring the same jobs without the power-aware characteristic, so as to evaluate the trade-off between performance and peak power reduction.

As shown in the previous section, the worst case scenario of a node, (considering the previously characterized jobs) corresponds to four concurrent executions of matrix multiplication (*cublas*), bringing the total peak power up to 1,120 W ($P_{peak\{WC\}}$) as shown in Fig. 5. Although this situation is very rare, the power supply has to be designed so as to sustain this condition.

4.3 Analysis of Results

Allocation of workloads to resources was evaluated for the two different policies (FIFO, BFF) while changing the constraint on the maximum power value attainable by the system (P_C). Since the scheduling is done according (3), changing P_C the maximum power consumed by the node and the execution time of the entire workload change as well. Several indices are introduced to evaluate the performance of the proposed technique. A detailed description of these follows:

4.3.1 Peak Reduction (PR)

The peak reduction is computed comparing the peak power values obtained executing the workload with and without the power aware characteristic as shown in (5):

$$PR = \frac{P_{peak\{ST\}} - P_{peak\{PA\}}}{P_{peak\{ST\}}} \cdot 100, \tag{5}$$

with

$$P_{peak} = \max(\{P_{node\,(n)} : n = 0, \dots, N-1\}), \tag{6}$$

where $P_{peak\{ST\}}$ is the peak power value measured during the execution of the algorithm without the power-aware characteristic while $P_{peak\{PA\}}$ is the peak power value measured during the execution of the power-aware version.

4.3.2 Peak Reduction with Respect to the Worst Case Power Scenario (PW)

Peak reduction with respect to the worst case power scenario is defined using (5) by substituting the power value measured during the execution of the algorithm without the power aware characteristic ($P_{peak\{ST\}}$) with the worst case power value ($P_{peak\{WC\}}$, in this case 1,120 W).

4.3.3 Increase in Time (T)

The increase in computation time is obtained comparing the execution time of the workload with and without the power aware characteristic as shown in (7):

$$T = \frac{T_{max\{PA\}} - T_{max\{ST\}}}{T_{max\{ST\}}} \cdot 100, \tag{7}$$

with

$$T_{max} = \max(\{T_{W\,(n)} : n = 0, \dots, N-1\}), \tag{8}$$

where T_W is the workload execution time.

4.3.4 Increase in Energy (EC)

The increase in the energy consumption is evaluated following (9):

$$EC = \frac{E_{\{PA\}} - E_{\{ST\}}}{E_{\{ST\}}} \cdot 100, \tag{9}$$

where the total energy consumption E is computed as shown in 10

$$E = \sum_{n=0}^{N-1} \left(\int_0^{T_{W\,(n)}} P_{node\,(n,t)} \cdot dt \right). \tag{10}$$

4.3.5 Peak Power Deviation from the Average (MD)

The peak power deviation from the average is obtained as reported in (11):

$$MD = \frac{P_{peak} - P_{avg}}{P_{avg}} \cdot 100, \tag{11}$$

where the average power is computed following (12):

$$P_{avg} = \frac{E}{\sum_{n=0}^{N-1} T_{W\,(n)}}. \tag{12}$$

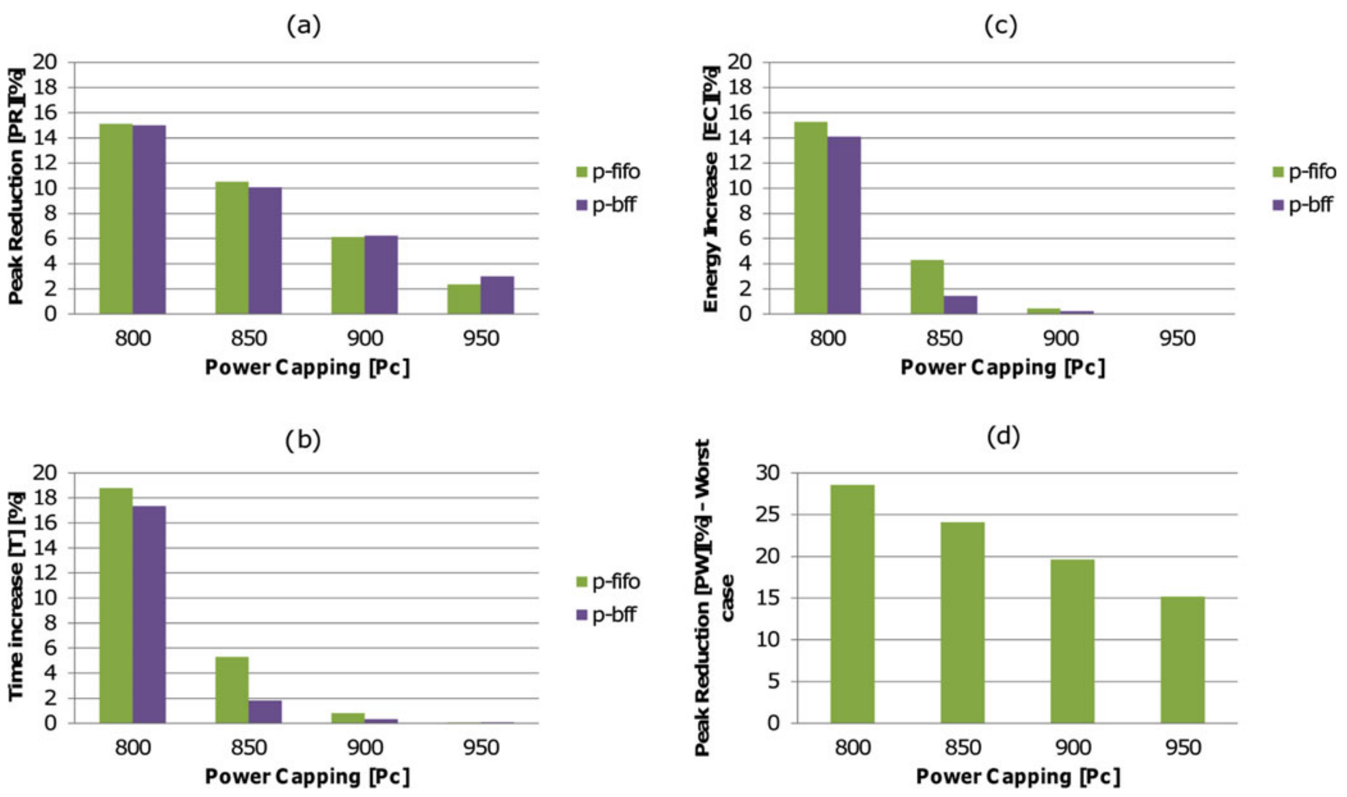


Fig. 6. Average performance obtained using the power-aware scheduling algorithm. In (a) one sees the average peak reduction obtained by the algorithm, while (b) shows the increase in time. Fig. (c) shows the increase in energy consumption and Fig. (d) the power reduction with respect to the worst case scenario.

Fig. 6 shows the experimental results measured by executing the workloads. As shown in Fig. 6a, by setting the power capping value at 800 W a peak power reduction of around 15 percent is measured. However this reduction is paid for by a time increase between 17 and 19 percent (Fig. 6b) and an increase in energy consumption between 14 and 15 percent (Fig. 6c), depending on the scheduling policy. This means that the threshold chosen (P_C) limits full exploitation of the computational parallelism available. By using a higher power budget (850 W), better results can be obtained. In this case the increase in time taken to compute the entire workload is less than 2 percent with the power-aware version of the BFF algorithm, with a measured peak power reduction of up to 10 percent. The increase in energy is negligible since it is less than 2 percent. Using the power-aware FIFO approach, results are slightly worse because each time the power constraint is not met all jobs are delayed. On further increasing the P_C threshold, a peak power reduction between 6 and 7 percent is recorded without any impact on system performance. This means that the algorithm removes the sporadic peaks that take place during workload execution, thus avoiding power capacity overload. Another advantage introduced by the algorithm is that it mitigates the worst case power scenario. As shown in Fig. 6d, by using the power-aware approach (with $P_C = 850$ W) the worst case scenario is reduced by up to 24 percent with a negligible impact on performance.

Fig. 7 explains the benefit of the algorithm from another point of view. The plot shows the peak power deviation from the average (MD) as a function of the increase in computational time (T). When no capping is forced a peak power 60 percent higher than the average value is measured. The first part of the curve (i.e., when power capping is set between 950 and 900 W) shows how the peak deviation from the mean can be reduced by 10 percent without any significant increase in the execution time of the workload, by scheduling jobs taking their power consumption into account. In the last part of the curve the value set in the algorithm is closest to the average power value of the workload, so that peak reduction is obtained at the cost of a significant time increase. As expected, the BFF scheduling policy allows one to achieve better results than the FIFO policy because it introduces fewer constraints on queue management. A summary of the measurements recorded is shown in Table 3.

5 Discussion

Although thorough modeling of GPU kernel power dissipation does not lie within the scope of this work, several

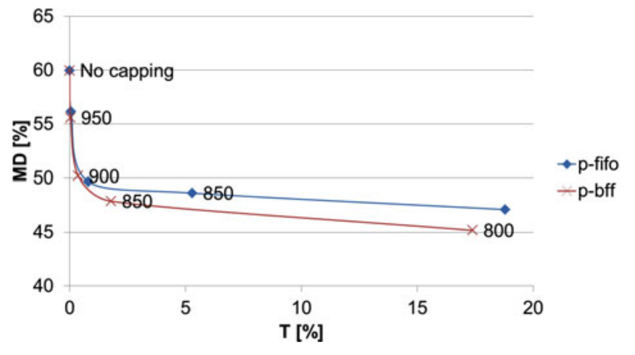


Fig. 7. Power-performance comparison.

TABLE 3
Power-Performance Comparison

	p-FIFO				p-BFF			
P_C	P_R [%]	T [%]	E [%]	MD [%]	P_R [%]	T [%]	E [%]	MD [%]
800	12.0-18.5	11.7-22.6	10.5-17.8	45.5-52.5	11.2-18.7	11.3-21.8	9.5-16.9	45.0-54.6
850	6.9-14.6	2.8-9.7	2.3-7.9	43.4-52.9	6.4-13.2	0-4.4	0-3.6	42.2-49.5
900	2.0-9.1	0-2.6	0-2.0	45.4-56.6	3.5-9.0	0-2.9	0-2.4	44.9-56.0
950	0-6.2	0-2.8	0-2.3	49.9-67.5	0-8.9	0-2.5	0-2.3	49.9-67.5

important considerations can be drawn from the measurement of power profiles taken by the measuring system:

- Execution of a GPU job (for the architecture studied) is much more peak power-consuming than execution of the same job on a CPU (although the execution time is significantly reduced);
- during execution of a kernel the power profile can be assumed constant;
- power consumption of jobs is not sensitive to changes in the input data set values;
- different input data sizes for the same kernel lead to the same trend with different durations. The runtime depends on the computational complexity of the algorithm used in the kernel.

Fig. 8 shows what has been discussed above. The first plot (Fig. 8a) was obtained when computing the same matrix multiplication three times, using three different algorithms. The first two are mapped on GPU (*matrixMul cublas* and *matrixMul kernel*), while the third is obtained by computing the same operation on CPU (*host execution*). Since the computation on GPU is much more power consuming than on CPU, in order to reduce peak power one should focus on concurrent executions of GPU kernels rather than on CPU tasks. Since the difference between the idle power state and the maximum power state of a CPU is small (compared to that of a GPU), the CPU power consumption can be assumed to be equal to its maximum power consumption each time a new job is activated on CPU. Fig. 8b shows the execution of different jobs performed on GPU. A GPU job can be composed of a single kernel or multiple call to the same kernel performed with different sets of input data to overcome the limited space of the GPU memory. Fig. 8b makes it clear that the power consumption of a GPU job can be considered as a constant contribution (dashed line) that has to be added to the total power consumption. Figs. 8c and 8d provide some additional considerations as to the values and sizes of input data. Fig. 8c shows four matrix multiplications performed with different input values. The graph shows that the power profile of a job depends only on the kernel computed and is independent of the input values. The last plot (Fig. 8d) shows that the same kernel, performed with different input data sizes, leads to different durations of the kernel, but with comparable peak power values. These considerations helped to streamline development of the algorithm (which is in fact based on these hypotheses) so as to make it general.

5.1 Application Case

As previously pointed out, the purpose of the algorithm is not to save energy, which increases, albeit slightly. The

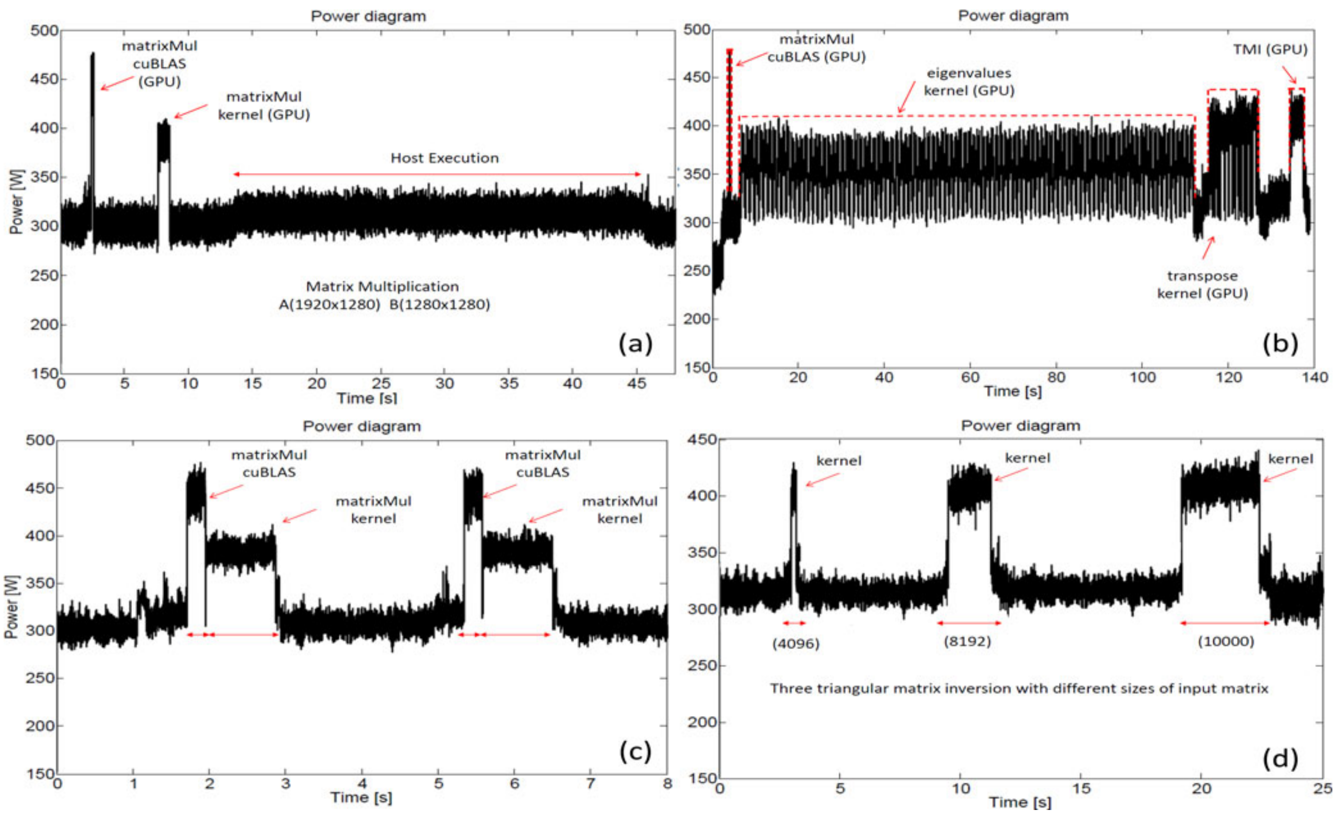


Fig. 8. Analysis of power requirements of different jobs; (a) Comparison between GPU and CPU execution of matrix multiplication; (b) Four different GPU kernels; (c) Triangular matrix inversion with different sizes of matrix; (d) Analysis of power requirements of different jobs.

approach aims to reduce the supply cost due to high peak power whilst having negligible impact on the parallelism of computational nodes.

From another point of view the developed model allows designers to increase the number of cores without increasing the capacity of the power supply unit. For example, each node used in this work is equipped with a power supply unit of 1,400 W. As shown in Section 4 the worst case power scenario is around 1,120 W. In this scenario, since each GPU GTX 590 used in this work consumes up to 365 W, to equip the system with another GPU could lead to system failure caused by power capacity overload. Using the proposed approach, it would be possible to add one GPU GTX 590 to the system, without overloading the power capacity, thereby reducing supply costs, cooling systems and power distribution units. The algorithm can be used as a level of adjustable power state software services [6], [30] in order to provide an efficient solution during high-demand periods.

5.2 Limitations of the Approach

Experimental results shown in this paper depend on the target architecture utilized in this work. The execution of these jobs on a different architecture could lead to a different peak power values. This is because the technique is based on an a priori characterization that is architecture dependent. If the target system changes, the characterization has to be repeated on the new target system. In addition a new characterization is needed each time a new kernel is added to the library. These considerations highlight how the power-aware scheduling algorithm is related to the low-cost monitoring system proposed.

6 Conclusion

This work presents a new algorithm for parallel scheduling, executed on GPU cluster nodes. The idea proposed is to manage both power consumption and GPUs as finite resources. Since the power configuration may vary widely, there is the likelihood that job overlapping will result in power spikes high enough to exceed the specifications of the nodes, causing catastrophic failures in systems designed to a better-than-worst-case policy. In addition, peaks synchronized across several nodes could cause localized power outage. Compared to a system without any power-aware policy, the model allows one to obtain a peak power reduction of as much as 10 percent. Executing workloads that usually involve high power peaks can be avoided at the cost of a very slight time increase, making it possible to reduce the power supply cost.

References

[1] M. Showerman, J. Enos, A. Pant, V. Kindratenko, C. Steffen, R. Pennington, and W.-m. Hwu, "QP: A heterogeneous multi-accelerator cluster," in *Proc. 10th LCI Int. Conf. High-Perform. Clustered Comput.*, 2009, pp. 1–8.

[2] V. V. Kindratenko, J. J. Enos, G. Shi, M. T. Showerman, G. W. Arnold, J. E. Stone, J. C. Phillips, and W. -M. Hwu, "Gpu clusters for high-performance computing," in *Proc. IEEE Int. Conf. Cluster Comput. Workshops*, 2009, pp. 1–8.

[3] ORNL. (2012, Dec.). Titan project timeline. [Online]. Available: http://www.olcf.ornl.gov/titan/

[4] Federal Energy Management Program, "Quick start guide to increase data center energy efficiency," U.S. Department of Energy, Tech. Rep., 2012. [Online]. Available: http://hightech.lbl.gov/documents/data_centers/Quick-Start-Guide.pdf

[5] C. Lefurgy, X. Wang, and M. Ware, "Power capping: A prelude to power shifting," *Cluster Comput.*, vol. 11, no. 2, pp. 183–195, 2008.

[6] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller, "SHIP: A scalable hierarchical power control architecture for large-scale data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 168–176, Jan. 2012.

[7] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "PowerPack: Energy profiling and analysis of high-performance systems and applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 658–671, May 2010.

[8] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal, "Adaptive, transparent frequency and voltage scaling of communication phases in MPI programs," in *Proc. ACM/IEEE Conf. Supercomput.*, 2006, p. 14.

[9] N. Kappiah, V. W. Freeh, and D. K. Lowenthal, "Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs," in *Proc. ACM/IEEE Conf. Supercomput.*, 2005, p. 33.

[10] C.-H. Hsu and W.-C. Feng, "A power-aware run-time system for high-performance computing," in *Proc. ACM/IEEE Conf. Supercomput.*, 2005, p. 1.

[11] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters," in *Proc. 7th IEEE Int. Symp. Cluster Comput. Grid*, 2007, pp. 541–548.

[12] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *Proc. IEEE 14th Int. Symp. High Perform. Comput. Archit.*, 2008, pp. 101–110.

[13] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, "Parallel job scheduling for power constrained HPC systems," *Parallel Comput.*, vol. 38, pp. 615–630, 2012.

[14] B. Lin, A. Mallik, P. Dinda, G. Memik, and R. Dick, "User-and process-driven dynamic voltage and frequency scaling," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2009, pp. 11–22.

[15] W. L. Bircher and L. K. John, "Core-level activity prediction for multicore power management," *IEEE J. Emerging Select. Topics Circuits Syst.*, vol. 1, no. 3, pp. 218–227, Sep. 2011.

[16] H. Ltaief, P. Luszczek, and J. Dongarra, "Profiling high performance dense linear algebra algorithms on multicore architectures for power and energy efficiency," *Comput. Sci.-Res.Develop.*, vol. 27, no. 4, pp. 277–287, 2012.

[17] E. Anderson, *LAPACK Users' Guide*. SIAM, Philadelphia, PA, USA, vol. 9, 1999.

[18] PLASMA—Parallel linear algebra software for multicore architectures, Version 2.4.5, 2011.

[19] C. Lively, X. Wu, V. Taylor, S. Moore, H.-C. Chang, C.-Y. Su, and K. Cameron, "Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems," *Comput. Sci.-Res. Develop.*, vol. 27, no. 4, pp. 245–253, 2012.

[20] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi, "Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster," in *Proc. 20th Int. Parallel Distrib. Process. Symp.*, 2006, pp. 298–298.

[21] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and use of simplepower: A cycle-accurate energy estimation tool," in *Proc. 37th ACM Annu. Desi. Autom. Conf.*, 2000, pp. 340–345.

[22] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," *ACM SIGARCH Comput. Archit. News*, vol. 28, no. 2, pp. 83–94, 2000.

[23] R. Suda and D. Q. Ren, "Accurate measurements and precise modeling of power dissipation of CUDA kernels toward power optimized high performance CPU-GPU computing," in *Proc. Int. Conf. Parallel Distrib. Comput., Appl. Technol.*, 2009, pp. 432–438.

[24] X. Ma, M. Dong, L. Zhong, and Z. Deng, "Statistical power consumption analysis and modeling for GPU-based computing," in *Proc. ACM SOSP Workshop Power Aware Comput. Syst.*, 2009, pp. 1–6.

[25] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, "The case for power management in web servers," in *Power Aware Computing*, New York, NY, USA: Springer-Verlag, 2002, pp. 261–289.

[26] *'Nvidia CUDA Programming Guide*, Nvidia, Santa Clara, CA, USA, 2011.

[27] F. Ries, T. De Marco, and R. Guerrieri, "Triangular matrix inversion on heterogeneous multicore systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 177–184, Jan. 2012.

[28] A. Krampe, J. Lepping, and W. Sieben, "A hybrid Markov chain model for workload on parallel computers," in *Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput.*, 2010, pp. 589–596.

[29] N. Sharifimehr and S. Sadaoul, "Markovian workload modeling for enterprise application servers," in *Proc. 2nd Canadian Conf. Comput. Sci. Softw. Eng.*, 2009, pp. 161–168.

[30] J. Heo, P. Jayachandran, I. Shin, D. Wang, T. Abdelzaher, and X. Liu, "OptiTuner: On performance composition and server farm energy minimization application," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 11, pp. 1871–1878, Nov. 2011.

Matteo Chiesi received the MS degree in electrical engineering from the University of Modena and Reggio Emilia, Modena, Italy, in 2009. He has worked for STMicroelectronics and is currently working toward the PhD degree at the Advanced Research Center on Electronic Systems (ARCES), part of the University of Bologna, Italy. In 2013, he was a visiting student at the Department of Computing, Imperial College London. His research interests include heterogeneous multi- core systems for high performance computing.

Luca Vanzolini received the informatics and communication technologies engineering degree from the University of Bologna, Bologna, Italy in 2006. Since 2006 he has been a consultant for STMicroelectronics, in the field of reconfigurable architectures first, and then in the field of energy harvesting and low-power systems. His main research interests include software and tools development for reconfigurable architectures.

Claudio Mucci received the electronics engineering degree from the University of Bologna, Italy, in February 2003 and the PhD degree at the same university in 2007. Since 2003, he has been with the Advanced Research Center on Electronic Systems E. De Castro (ARCES), Bologna, and during that period he has been a STMicroelectronics consultant in the field of reconfigurable computing. Since the 2009, he joined STMicroelectronics Technology R&D, Agrate Brianza, Italy. His main research interests include configurable platforms based on run-time and metal-programmable technologies, digital signal processing, application development, and related methodologies applied to embedded programmable system. He is a co-author of about 30 publications on international conferences and journals in the same field.

Eleonora Franchi Scarselli (M'98) received the MS degree in electrical engineering and the PhD degree in "electrical engineering and computer science" from the University of Bologna, Bologna, Italy, in 1992. From 1994 to 2004, she was a research assistant with the Faculty of Engineering, University of Bologna, where she has been an associate professor since 2005 and she currently teaches design of digital integrated circuits. She is with the Advanced Research Center on Electronic Systems (ARCES) at the same University where she has been and is presently involved in research activities concerning design of circuits and system for RF and sensor applications, and for enabling wireless 3-D communication. She is a member of the IEEE.

Roberto Guerrieri received the DrEng and PhD degrees from the University of Bologna, Italy, in 1980 and 1986, respectively. After working at the Department of Electrical Engineering and Computer Sciences of the University of California at Berkeley as a visiting researcher and at the MIT as a visiting scientist, he joined the University of Bologna where he is a full professor and teaches design of integrated systems. His research interests include various aspects of integrated circuit modeling and design, including digital systems and biometric sensors, and applications of microelectronics to biotechnology. His work on VLSI design has been cited by widely read magazines, such as *the Nikkei and Electronic Design,* and documented in more than 90 scientific papers. He is a director of the joint laboratory between STMicroelectronics and the University of Bologna. In this role, he is involved in the development of new microprocessors, computational systems, and packaging technologies for advanced silicon circuits. He has founded two start-up companies in the field of biomedical devices for patient monitoring in cancer therapy. In 1992, he received the best paper award of the *IEEE Transactions on Semiconductor Manufacturing* for his research carried out on issues related to the modeling of various IC manufacturing steps. In 2004, he received an ISSCC best paper award for his work in the area of silicon-based lab-on-a-chip.