Changing computing paradigms towards power efficiency

Pavel Klavík¹, A. Cristiano I. Malossi², Costas Bekas² and Alessandro Curioni²

¹Faculty of Mathematics and Physics, Computer Science Institute, Charles University in Prague, Malostranské nám. 25, 118 00 Prague, Czech Republic
²IBM Research-Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland

Power awareness is fast becoming immensely important in computing, ranging from the traditional high-performance computing applications to the new generation of data centric workloads. In this work, we describe our efforts towards a power-efficient computing paradigm that combines low- and high-precision arithmetic. We showcase our ideas for the widely used kernel of solving systems of linear equations that finds numerous applications in scientific and engineering disciplines as well as in large-scale data analytics, statistics and machine learning. Towards this goal, we developed tools for the seamless power profiling of applications at a fine-grain level. In addition, we verify here previous work on post-FLOPS/W metrics and show that these can shed much more light in the power/energy profile of important applications.

1. Introduction

Since the 1970s, the high-performance computing community, including industry and academia, has tremendously benefited from the existence of a single, simple, and easy to deploy benchmark: the LINPACK benchmark [1]. Its original target was the solution of a dense linear system of 1000 equations in the minimum time, thus generating a ranking of the most powerful supercomputers based on the measured number of floating-point operations per unit of time (FLOPS). As supercomputers became increasingly more potent, the LINPACK benchmark evolved allowing...
the problem size to be scaled, while keeping the same goal. Indeed, while early machines achieved mega-FLOPS, today’s most powerful supercomputers reach peta-FLOPS, which constitutes an astonishing increase of nine orders of magnitude in less than 35 years.

Since 1993, a list of the most powerful supercomputers has been kept and updated biannually in June and November: this is the famous TOP500 list (http://top500.org). It graphically portrays this exponential increase in performance, which is attributed mostly to Moore’s law, which, in turn, has been widely (and vaguely) interpreted as an exponential performance improvement in CMOS technology. However, during the past decade, the core technology has dramatically changed. Voltage and thus frequency scaling has stopped, pointing out the true nature of Moore’s law, which essentially predicted our technological capacity to exponentially increase the number of logical gates on a chip. To continue the exponential overall improvements, technologists have turned to multi-core chips and parallelism at all scales. However, this new trend has led to a series of new issues and questions, which nowadays represent the most critical aspects in the development of future architectures and software. These questions are mainly: (i) how to program such complex machines? and (ii) how to cope with the very fast increase in power requirements? In this work, we concentrate on the latter, although we believe that programmability and energy efficiency will ultimately prove to be closely connected.

Bekas & Curioni [2] have shown that very important power savings can be achieved by a careful combination of different precision levels for floating-point arithmetic. Those observations led to the introduction of new energy-aware performance metrics. This was motivated by the fact that the traditional energy-aware performance metrics (derived from the classic FLOPS performance metric) have serious shortcomings and can potentially draw a picture that is far from reality. For instance, the widespread FLOPS/W metric, which is used to rank supercomputers in the Green500 list (http://green500.org), might still promote power-hungry algorithms instead of other more energy-efficient techniques. By contrast, by using metrics based on both time-to-solution and energy, as proposed in [2], it is possible to get a much more realistic picture, which can drive and help the design of future energy-efficient machines and algorithms. At the same time, we also need to abandon the (convenient and simple) LINPACK benchmark and make use of a relatively small selected set of benchmarks, characterized by widely different features and thus representative of the majority of the current applications [3,4].

The reasoning for the mixed precision computing paradigm and the associated energy-aware performance metrics in [2] is based on a model of power consumption for the various subsystems of a computing platform, such as the floating-point unit, the interconnect and the memory; this model is an extrapolation of future exascale machines [5]. At the same time, it is widely acknowledged that actual online measurements can be complicated and require external circuits that cast noise in the measurements. Recent advances in chip technology, however, have allowed the on-chip measurement of instantaneous power consumption by means of integrated sensors [6–9]. In this work, we advance the initial contribution in [2] and describe a fully automated, online, minimally invasive system for measuring power performance. Our tool follows a similar philosophy to other tools such as the POWERPACK3.0 (http://scape.cs.vt.edu/software/powerpack-3-0/) package [10], however with more emphasis on noise measurement minimization and fine granularity. Our system allows users to easily instrument their code as well as to directly distinguish and separately measure the power performance of various code segments. The tool is completed by a carefully designed graphical interface, which provides real-time, easy-to-read informative plots and diagrams.

The target platform of our analysis is the IBM POWER7 processor that offers a wide variety of on-chip sensors. On this machine, we perform a series of analyses that confirm and extend findings and theories devised in [2] on an IBM Blue Gene/P system. More importantly, we outline a general energy-aware framework for solving linear systems that is based on a careful combination of mixed-precision arithmetics. In addition, we take a further step and show that our framework allows the use of inexact data as well as inexact computations. This is particularly crucial as data movement is expected to completely dominate running time and power consumption in future systems.
In summary, the three main new contributions provided in this work are (i) the development of a new class of iterative methods based on iterative refinement and data approximation, (ii) the extension of the metrics analysis in [2] towards new architectures and algorithms, and (iii) the description of a new minimally invasive power consumption tool which is able to fully exploit the new IBM POWER7 measurement system.

The work is organized as follows. In §2, we describe our framework for online, on-chip power consumption measurements. In §3, we describe the inexact computing/data framework for solving systems of linear equations. Then, in §4, we present our experimental results applied to covariance matrices from applications in data analytics. We conclude with remarks in §5.

2. Online, on-chip power measurement

In the following, we describe the framework that we set up to perform power measurements. Our system consists of an IBM blade server PS702 (http://www-03.ibm.com/systems/bladecenter/hardware/servers/ps70series/specs.html) [11], with two IBM POWER7 chips (octa core, 3.00 GHz CPU and 4 MB L3-cache each) and 64 GB of system memory. Each POWER7 chip has several embedded thermal sensors that measure the temperature of the different components. Data from sensors are transferred to a separate service processor that is embedded in the POWER7 server. The service processor is running a firmware called AME driver, which, in turn, calculates power from the thermal sensors as described in [7–9]. We remark that it is crucial that power readings are calculated on the service processor, so they do not add background noise to the actual user computation; in this regard, the POWER7 server is an ideal platform to perform power measurements. Finally, on a remote machine, we run a software named Automated Measurement of Systems for Energy and Temperature Reporting (AMESTER). This is an experimental application developed by IBM which periodically retrieves power measures from the service processor of the POWER7 server and collects them into a file (for more details, see [8,9]). A graphical description of the power measurement framework is depicted in figure 1.

To ease the instrumentation of the code, we have developed a library that provides a black-box interface between AMESTER and the running application. This layer uses TCP/IP sockets hidden behind a series of simple calls. Issues of TCP/IP latency and resolution of instrumentation are directly resolvable, because the AME driver allows for a store/trace forward mode, where a bunch of power data are measured and collected such that they can be subsequently read remotely. Thus, the instrumentation permits us to set-up and run AMESTER, and then mark power data during computation; using this framework, we can map measured data to different parts of the code and get a clear picture of the specific power consumption for each analysed kernel in our application. The resolution of power is less than 1 W, whereas the sampling frequency is approximately 1 kHz
Figure 2. Power of chips and memories on the IBM blade server PS702 during an idle state.

Table 1. Average power (over 300 s) of chips and memories on the IBM blade server PS702 during an idle state.

<table>
<thead>
<tr>
<th>sensor</th>
<th>average power (W)</th>
<th>standard error (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>142.1</td>
<td>0.6</td>
</tr>
<tr>
<td>processor 1</td>
<td>42.8</td>
<td>0.2</td>
</tr>
<tr>
<td>processor 2</td>
<td>44.2</td>
<td>0.2</td>
</tr>
<tr>
<td>memory 1</td>
<td>12.5</td>
<td>0.4</td>
</tr>
<tr>
<td>memory 2</td>
<td>13.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

(i.e. up to 1000 samples per second). The error of the power measurements is stated to be lower than 2% [7,11,12].

An example of power analysis is illustrated in figure 2, where we plot measurements from five power sensors during an idle state of the machine. From top to bottom, these are: total power of the system, power of each of the two processors and power of each of the corresponding memory banks. Note that the total power of the system includes additional sources of power consumption, such as disks and fans, which in the idle state consume approximately 30 W. These values are averaged in time in table 1, providing us with the lower bound limit of power consumption of the PS702 blade server. In the examples presented in the forthcoming sections, we observe that chip power can reach up to 80 W, whereas memory power can get up to 40 W. Total power is thus always between 140 and 200 W.
3. Inexact computing for the solution of dense linear systems

Let us focus on the solution of a dense system of $n$ linear equations in the form

$$Ax = b,$$ (3.1)

where $A \in \mathbb{R}^{n \times n}$ is a symmetric and positive definite matrix, $x \in \mathbb{R}^n$ is the solution vector, and $b \in \mathbb{R}^n$ is the right-hand side. This problem can be generalized to the case of $m$ right-hand sides $b_1, \ldots, b_m$ such that we can write

$$AX = B,$$ (3.2)

where $B = [b_1, b_2, \ldots, b_m] \in \mathbb{R}^{n \times m}$ and $X = [x_1, x_2, \ldots, x_m] \in \mathbb{R}^{n \times m}$ are two rectangular matrices containing all the right-hand sides and solution vectors, respectively.

The target of our analysis is the solution of large size covariance matrix problems, which are crucial for uncertainty quantification and applications in data analytics (e.g. [13] and references therein). With this aim, in the following, we recall from the literature two iterative refinement procedures derived from the Cholesky and the conjugate gradient (CG) methods for the solution of this class of problems. These schemes represent the starting point for our new developments, which are presented later in §4. More details on iterative refinement algorithms can be found in previous studies [2,14,15] and references therein.

(a) Cholesky with iterative refinement

The Cholesky decomposition $A = R^T R$ [16] offers a cache-friendly and high-performance approach for the solution of problems (3.1) and (3.2). The overall complexity of the decomposition is $\frac{1}{3}n^3 + O(n^2)$ FLOP. Once it is done, the solution to the problem can be computed very fast, even for multiple right-hand sides,

$$X = R^{-1}R^{-T}B,$$ (3.3)

where we remark that $R$ is an upper-triangular matrix; consequently, the cost of (3.3) is $O(n^2)$ FLOP per right-hand side vector, thus, in general, it is negligible because in most of the applications $m \ll n$.

In previous studies [13,17] (see also references therein), it has been shown that, to cope with the formidable cubic FLOP cost of the Cholesky decomposition, mixed precision iterative refinement can be used. For instance, let us consider the procedure described in algorithm 1 for the case of a single right-hand side, where we use the ‘~’ symbol to denote matrices or vectors stored in low precision. On modern multi-core chips with accelerators, the computation of the Cholesky decomposition at the very first step can be sped up by using IEEE single-precision arithmetics. The resulting inexact decomposition is then also used at line 5 of algorithm 1 for each iteration until convergence.

Algorithm 1. Cholesky method with iterative refinement for one right-hand side.

1: Compute Cholesky decomposition $A \approx \tilde{R}^T \tilde{R}$. [low precision]
2: Generate approximate solution $x = \tilde{R}^{-1} \tilde{R}^{-T} b$. [low precision]
3: Compute residual $r = b - Ax$. [high precision]
4: while $\|r\|_2 \geq$ tolerance do
5: Compute error components $z = \tilde{R}^{-1} \tilde{R}^{-T} r$. [low precision]
6: Update solution $x = x + z$. [high precision]
7: Compute residual $r = b - Ax$. [high precision]
8: end while

Observe, however, that the complexity of the overall scheme remains $O(n^3)$ FLOP, owing to the matrix factorization. Therefore, even with significant single-precision acceleration, the time-to-solution increases cubically with the matrix size and dominates the computation in the
case of large problems. A second important remark concerns the very nature of dense matrix factorizations, such as the Cholesky decomposition. These are rich in BLAS-3, cache-friendly linear algebra primitives, a property that has been the focus of attention for optimizing system performance. It reflects that codes such as the Cholesky decomposition tend to make full use of the computational resources of the system, stressing its components to their limits.

As an example, figure 3 shows power measurement for solving a linear system with 32 right-hand sides using the Cholesky decomposition and mixed precision iterative refinement. It is evident that the majority of time and energy is spent for the Cholesky decomposition. In addition, note that power consumption peaks around 200 W, owing to the BLAS-3 primitives that ensure high sustained performance in the arithmetic computations.

The burning question, thus, is whether it is possible to fundamentally reduce the computational cost in terms of FLOP. Iterative refinement theory predicts that the only requirement for the process to converge is that the solver used for the inner step at line 5 of algorithm 1 is not too ill-conditioned [18]. In particular, the following conditions need to be satisfied:

\[(A + \Delta A)z = r \quad \text{and} \quad \|A^{-1}\Delta A\|_\infty < 1,\]  

(3.4)

where \(\Delta A\) is a given perturbation of the matrix coefficients. In the iterative refinement setting, this perturbation corresponds to the error introduced by storing the matrix (or its factorization) in low precision. Bound (3.4) implies that if the problem is not too ill-conditioned, then the iterative scheme would still be able to converge with the desired high precision, even if the accuracy of the internal solver is quite low.

To see this, consider the results of [13,17] depicted in figure 3b: there we plot the number of iterations to solve a dense linear system up to working precision (in our case, IEEE double precision), by using a perturbed Cholesky algorithm. In particular, the factor \(R\) (computed in double precision) is perturbed with a range of upper-triangular matrices \(E\), whose norm is such that

\[
\|R + E\|_2 / \|R\|_2 \leq 10^\gamma,
\]

with \(\gamma \in \{-1, -3.25, -5.5, -7.75, -10\}\). This experiment evidently demonstrates that even in presence of large perturbations, e.g. \(\gamma = -1\), the method is still able to converge. Moreover, even if the number of iterations increases, then each one of them costs \(O(n^2)\) FLOP, which is generally several orders of magnitude less than the Cholesky decomposition.

Figure 3. (a) Power diagram on IBM PS702 blade server during the solution of a dense linear system with 32 right-hand sides, using the Cholesky-based approach with iterative refinement. In the timeline, each computational phase is displayed with a different color. (b) The number of iterations to solution of a perturbed Cholesky algorithm, for a range of matrix sizes and perturbation exponents.
This example shows clearly that the accuracy of the used solver can be lowered without compromising the robustness of the numerical scheme. In fact, the very nature of the solver we choose is quite irrelevant, with the only crucial property being that of bound (3.4).

(b) Conjugate gradient with iterative refinement

In [13] (see also [17]), the CG method with iterative refinement (see algorithm 2) is proposed as a possible energy-efficient replacement of the Cholesky-based approach. Particularly, it is shown that the overall computational cost in terms of FLOP can be reduced down to quadratic levels from the original cubic cost of dense solvers.

**Algorithm 2.** Conjugate gradient method with iterative refinement for one right-hand side.

1: Set initial solution $x = x_0$. 
2: Compute residual $r = b - Ax$. 
3: Set initial direction $p = r$. 
4: while $||r||_2 \geq$ tolerance do 
5: Compute matrix-vector multiplication $z = \tilde{A}p$. 
6: Compute $\rho = r^T r$ and $\alpha = \rho / (p^T z)$. 
7: Update solution $x = x + \alpha p$. 
8: Update residual $r = r - \alpha z$. 
9: Compute $\beta = (r^T z) / \rho$ and update direction $p = r + \beta p$. 
10: end while

Figure 4 illustrates power measurements for this approach, solving the same problem as in the Cholesky-based version, but with one right-hand side. First of all, we observe that time-to-solution is strongly reduced when compared with the Cholesky-based method. Indeed, there is no initial factorization phase any longer that dominates the run time. A second observation regards the peak power consumption which, with respect to the idle state, is about 50% less than the peak power observed for the Cholesky case in figure 3a.

When increasing the number of right-hand sides from 1 to 32, as shown in figure 4b, the major internal kernel of the CG-based algorithm becomes the matrix–matrix multiplication, and thus peak power characteristics resemble those of the Cholesky-based approach. The sudden decreases
in power are due to executing vector operations (BLAS-2), which do not stress the system as much as BLAS-3-based computations. We also note that the first few steps last longer. Indeed, after a few cycles the machine is able to optimize the use of the cache levels to access the vectors in the memory, thus reducing the step time. As a consequence, in the case of the multiple right-hand sides, it is by far the decreased time-to-solution that drives the decrease in total energy of the CG-based method with respect to the Cholesky one.

4. Iterative refinement on approximate covariance matrices

Bound (3.4) offers a second reading: we can cast the error not to the used solver, but to the data. Indeed, this is a statement akin to the classical backward stability. In other words, we can use an exact solver that works on inexact data, as shown in the example of the perturbed Cholesky factor discussed in §3.

A possible strategy to take advantage of this approach is to approximate the system matrix by following the pattern in its structure. For instance, in several applications, matrices exhibit a progressive decay of their elements away from the main diagonal. In those cases, a reduced banded version of the matrix (with just a few diagonals) can be used, in place of the complete one. Observe that the resulting complexity of the inner solver is drastically reduced as well as data movement requirements which are lowered by almost one order of magnitude.

In general, we can opt for sampled versions of the matrices involved for the inner solver. To this end, we can borrow approaches developed in the literature of randomized linear algebra (e.g. [19] and references therein). The main idea is that matrix A is replaced by \( \tilde{A} = PAP^T \), where \( P \) is a suitably chosen fat sketching matrix \( P \in \mathbb{R}^{k \times n} \), where \( k \ll n \) [20].

In the following, we showcase the validity of our arguments with a set of numerical experiments using model covariance matrices with controlled degree of decay \( d \) away from the main diagonal:

\[
A(i,j) = \begin{cases} 
1 & i = j, \\
\frac{1}{|i-j|^d} & i \neq j,
\end{cases}
\]

In all our experiments, the right-hand-side vectors are generated randomly with coefficients in \([0, 1]\), whereas the tolerance for the stopping criteria is set equal to \(10^{-5}\).

Note that matrices with a different degree of decay are equally difficult for the standard Cholesky-based solver. By contrast, for iterative solvers, such as the CG-based method, matrices that exhibit strong decay (i.e. high \( d \)) are easier to solve with respect to those with a small or null decay, because they become strongly diagonally dominant, and thus the scheme converges faster. To ease the comparison of the results for different decay levels, all the banded CG tests have been run with the same bandwidth (33 diagonals, 16 on each side of the main diagonal). In practical applications, the size of the bandwidth should be adapted to the specific problem, either a priori or by using an automatic tuning mechanism.

To optimize performances, we use 16 threads (eight per chip, one per core) for BLAS-3-rich computations (i.e. Cholesky and CG with 32 right-hand sides), and eight threads (four per chip) for the other cases (i.e. CG with one right-hand side). Parallelism is, in general, exploited through the fully optimized IBM ESSL (http://www-03.ibm.com/systems/software/essl/) routines. However, for the banded case a big part of the parallel code is hand-coded; there, parallelism is achieved through simple OpenMP calls without further optimizations. The symmetry of the matrix was not exploited in the CG implementation.

(a) Time and energy analysis

Figure 5 illustrates time-to-solution and energy consumption to solve problem (3.1) or (3.2) with different iterative refinement-based methods and for a range of complete and approximate covariance matrices, with \( d = [1, 2, 4] \). All the quantities are calculated without accounting for
Figure 5. Time and energy comparison between different methods for the solution of power model covariance matrices of different size. (a) $d = 1$. (b) $d = 2$. (c) $d = 4$. (Online version in colour.)

the initialization phase. The Cholesky-based method shows no difference with respect to the degree of decay, because it has a fixed complexity that depends only on the matrix size. A similar result is also obtained for the non-banded CG-based method. However, by using the banded CG version, we observe a sensible progressive reduction in the time-to-solution, and thus in the total consumed energy.

To quantify the benefits of the banded CG-based method with respect to the other approaches, in table 2, we report the values obtained for a covariance matrix with $n = 70\,000$. In most of the cases, even if it is not fully optimized through ESSL routines, the banded CG-based version is able to reduce the time-to-solution by a factor of 10 or more, with respect to the non-banded one. This gain is mainly due to the reduced time cost of each low-precision iteration. This is evident in the case $d = 4$, where the banded CG-based version converges between 15 and 30 times faster than the non-banded version, despite the number of low- and high-precision iterations being exactly the same for both methods. In addition, the peak power consumption does not increase when solving for more than one right-hand side, leading to an additional reduction in the total energy consumed by the problem. For instance, in the case of $d = 4$ and $n = 70\,000$, the energy consumed by the banded CG version with 32 right-hand sides is just 6% of the energy consumed by the non-banded CG version to solve exactly the same problem.
(b) Standard performance metrics analysis

Despite the energy analysis §4d highlighted the strong potential of the inexact iterative refinement strategy, by ranking the methods with standard metrics, i.e. the TOP500 (GFLOPS) and the Green500 (GFLOPS/W), we obtain a surprising result: the banded CG-based method scores quite poorly in both rankings (table 2, rightmost columns).

This behaviour is also confirmed in figure 6, for a large range of matrix sizes. The motivations behind these results are linked to the nature of the methods used. On the one hand, the Cholesky and the non-banded CG (especially with multiple right-hand sides)-based methods make extensive use of computational resources, performing a very large number of operations per second to factorize the matrix and to perform dense matrix–vector multiplications, respectively. On the other hand, the banded CG-based method performs much less multiplications per second, owing to the reduced effective size of the matrix, which is also stored in a different format. Therefore, the number of BLAS-2 and BLAS-3 operations for the banded CG-based method is drastically reduced, leading to a poor score in the TOP500 ranking.

Regarding energy, we observe that the curves in the right-hand side graphs are quite similar to those for the time on the left. The reason is that average power differs for methods by at most 20 W, so that it has a very low impact on the energy trend with respect to time-to-solution, which is dominant. This also reflects in the Green500 ranking, where the result is dominated by the number of GFLOPS, regardless of the effective average power consumption, and thus leading to similar conclusions with respect to the TOP500 ranking. However, those 20 W actually correspond to a significant 33% difference with respect to the net power consumption (from 142.1 up to 200 W).

Table 2. Comparison of different methods to solve a model covariance matrix, with \( n = 70\,000 \).

<table>
<thead>
<tr>
<th>( d ) = 1</th>
<th>( d ) = 2</th>
<th>( d ) = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RHS</strong></td>
<td><strong>iterations</strong></td>
<td><strong>time (s)</strong></td>
</tr>
<tr>
<td>Cholesky</td>
<td>32</td>
<td>1/1</td>
</tr>
<tr>
<td>CG</td>
<td>1</td>
<td>179/2</td>
</tr>
<tr>
<td>CG</td>
<td>32</td>
<td>83/1</td>
</tr>
<tr>
<td>banded CG</td>
<td>1</td>
<td>691/8</td>
</tr>
<tr>
<td>banded CG</td>
<td>32</td>
<td>692/8</td>
</tr>
<tr>
<td><strong>Cholesky</strong></td>
<td>32</td>
<td>1/1</td>
</tr>
<tr>
<td><strong>CG</strong></td>
<td>1</td>
<td>83/1</td>
</tr>
<tr>
<td><strong>CG</strong></td>
<td>32</td>
<td>85/1</td>
</tr>
<tr>
<td><strong>banded CG</strong></td>
<td>1</td>
<td>173/2</td>
</tr>
<tr>
<td><strong>banded CG</strong></td>
<td>32</td>
<td>177/2</td>
</tr>
<tr>
<td><strong>Cholesky</strong></td>
<td>32</td>
<td>1/1</td>
</tr>
<tr>
<td><strong>CG</strong></td>
<td>1</td>
<td>85/1</td>
</tr>
<tr>
<td><strong>CG</strong></td>
<td>32</td>
<td>88/1</td>
</tr>
<tr>
<td><strong>banded CG</strong></td>
<td>1</td>
<td>85/1</td>
</tr>
<tr>
<td><strong>banded CG</strong></td>
<td>32</td>
<td>88/1</td>
</tr>
</tbody>
</table>
In summary, our results show clearly that standard metrics are not able to capture the real behaviour of applications, and this is mainly due to the fact that they rank methods and machines on the basis of GFLOPS and GFLOPS/W rather than real quantities of practical interest, i.e. time-to-solution and energy. In this regard, the new strategies proposed in [2] represent a first step towards the development of concrete energy-aware metrics. However, to drive the development of future hardware in the correct direction, this effort should be combined with the introduction of new benchmarks based on, for example, the CG method [21] or some of the Dwarfs proposed in [3,4], as replacements of the current power-hungry and FLOP-driven LINPACK benchmark.

5. Conclusion

Power awareness and energy consumption is quickly becoming a central issue in HPC systems and research. There is rich recent activity towards solutions that allow us to continue historical exponential improvements in performance while at the same time keeping power requirements at affordable levels. Towards this direction, we described here our recent advancements in the
context of energy-aware performance metrics for HPC systems. More in detail, we developed a framework for accurate, on-chip and online power measurements, with minimal distortion of user runs. Our tool allows the easy instrumentation of user code which subsequently enables its detailed power profiling. We demonstrated the validity and superiority of our proposed performance metrics on actual energy measurements. Finally, we described a framework that combines different precision levels and relaxes data fidelity, and thus has the potential to reduce energy by a few orders of magnitude.

Acknowledgements. The project Exa2Green (under grant agreement no. 318793) acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the ICT theme of the Seventh Framework Programme for Research (FP7/2007-2013) of the European Commission. IBM, ibm.com, ESSL and POWER7 are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at ‘Copyright and trademark information’ at: www.ibm.com/legal/copytrade.shtml.

Funding statement. This work was completed during an internship of the first author at the Computational Sciences Group, IBM Research-Zurich. Partially supported by CE-ITI (P202/12/G061 of GAČR).

References


