Energy Footprint of Advanced Dense Numerical Linear Algebra using Tile Algorithms on Multicore Architecture

Jack Dongarra[*][†][‡][¶], Hatem Ltaief[§], Piotr Luszczek[*], and Vincent M. Weaver[*]

[*]*Innovative Computing Laboratory, University of Tennessee, Knoxville, TN 37996, USA*
[†]*Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee*
[‡]*School of Mathematics & School of Computer Science, University of Manchester*
[¶]*Research reported here was partially supported by the National Science Foundation and Microsoft Research.*
Email: dongarra,luszczek,vweaver1@eecs.utk.edu

[§]*KAUST Supercomputing Laboratory, Thuwal, Saudi Arabia*
Email: Hatem.Ltaief@kaust.edu.sa

***Abstract*—We propose to study the impact on the energy footprint of two advanced algorithmic strategies in the context of high performance dense linear algebra libraries: (1) mixed precision algorithms with iterative refinement allow to run at the peak performance of single precision floating-point arithmetic while achieving double precision accuracy and (2) tree reduction technique exposes more parallelism when factorizing tall and skinny matrices for solving overdetermined systems of linear equations or calculating the singular value decomposition. Integrated within the PLASMA library using tile algorithms, which will eventually supersede the block algorithms from LAPACK, both strategies further excel in performance in the presence of a dynamic task scheduler while targeting multicore architecture. Energy consumption measurements are reported along with parallel performance numbers on a dual-socket quad-core Intel Xeon as well as a quad-socket quad-core Intel Sandy Bridge chip, both providing component-based energy monitoring at all levels of the system, through the PowerPack framework and the Running Average Power Limit model, respectively.**

***Keywords*-Power Consumption, Dense Linear Algebra, Mixed Precision Algorithms, Tree Reduction, Tile Algorithms, Dynamic Scheduling, PowerPack, RAPL**

INTRODUCTION

The K computer achieves more than 10 Pflop/s with over half a million of cores and most importantly, consumes over 12 MW. The roadmap to exascale [1] lays out the future exascale system by the end of this decade, achieving 10^{18} floating point operations per second with a 20 MegaWatt power consumption. This is a formidable challenge: gaining three order of magnitude in performance compared to today's systems while doubling the power consumption by a factor of two. This challenge arises from the fact that incremental evolution in today's computer systems will not allow to reach exascale systems within a reasonable power budget envelope. Even with technology scaling and the emergence of next generation low frequency cores (GPUs, ARM, BlueGene/Q, etc.), further shrinking the observed power gap appears to be a major obstacle to achieve an exascale system at reasonable power requirements. While hardware architecture components are attaining the limit of the physics, the burden has turned back to the software community to help in designing new power-aware high performance numerical methods. These new algorithms need essentially to shorten the time-to-solution metric of an application as much as possible.

This is in this context that the dense linear algebra community started couple years ago to revisit the standard block algorithms as implemented in LAPACK [2], a widely-accepted numerical library which provides solvers for systems of linear equations, eigenvalue problems and singular value decompositions. Two main numerical libraries were created PLASMA [3] and FLAME [4] to undergo this major redesign. The core ideas are threefold: increasing fine-grained parallelism using tile algorithms, reducing data motion and driving the execution using a dynamic runtime system. The ultimate goal is to eliminate the LAPACK overheads engendered by the artifactual synchronization points, located between the different computational stages.In a previous work from some of the authors [5], the power efficiency study of the standard factorizations and reductions has been performed and has permitted to highlight the impact on energy consumption of the various algorithmic stages in the corresponding functions from LAPACK as well as PLASMA. In this paper, we propose to extend the energy footprint of this class of algorithms to a more advanced category based on tree reduction techniques first introduced in [6] and mixed precision solvers with iterative refinement mechanism, described first in [7]. Both techniques have been integrated in the PLASMA library using tile algorithms. While the tree reduction strategy increases the degree of parallelism, exposing more fine-grained tasks to the dynamic scheduler, the mixed precision solvers with iterative refinement run at the rate of single precision arithmetic while getting the solution accuracy at double precision arithmetic. To monitor and collect the power data, we used PowerPack [8], a home-brewed framework from Virginia Tech on a dual-socket quad-core Intel Xeon, and the RAPL (Running Average Power Limit) model [9] associated with PAPI [10], [11] on a dual-socket eight-core Intel Sandy Bridge chip.

The remainder of this paper is organized as follows. Section I gives a detailed overview of previous research works in this area. Section II describes the fundamental principles of mixed precision solvers with iterative refinement and tree reduction techniques. Section III recalls the concepts of block and tile algorithms within LAPACK and PLASMA, respectively. Section IV highlights the experimental environment in terms of hardware and software settings and performance numbers are reported in Section V. Section VI demonstrates the impact of both numerical techniques on power consumption. Section VII outlines the result interpretations and correlates parallel performance with energy efficiency. Finally, Section VIII summarizes the results of this paper and presents some future work.

I. Related Work

As mentioned in the introduction, our previous work provided a similar analysis of energy efficiency of various implementations of commonly used linear algebra solvers [5]. Here, we extend this study to include less commonly used numerical algorithms and also add power profiles based on Intel's RAPL technology that allows to measure energy seamlessly by using hardware counter technology available on Intel Sand Bridge line of multicore processors.

Dense linear algebra solvers were first to benefit from the mixed-precision iterative refinement algorithm [12]. Mixed precisions techniques are also used for further approximating contributions from farther particles in the context of fast multipole methods as well as LQCD computation [13], resulting in speeding up the computation while reducing memory traffic. Moreover, tree reduction techniques were naturally identified as a way to increase concurrency while reducing data motion on multicore architecture [14]–[16]. In this work, we present detailed power analysis of these mixed precision and tree reduction codes.

Furthermore, the use of mixed precision iterative refinement for sparse matrix problems [17] has lead to interesting conclusions in terms of energy efficiency and power consumption [18], [19]. We dive into these issues in this paper by studying in detail dense linear algebra solvers that feature mixed precision iterative refinement technique [12], [20].

A number of energy metrics have been proposed [21] and we believe that this study will be helpful in comparing these metrics in an accurate way based on the optimized numerical techniques aforementioned.

Last but not least, a comparative study of energy consumption by commodity hardware benchmarks and consumer games [9] complements our study presented here. We refer the reader to the vendor analysis [9] for a more detailed description of Running Average Power Level (RAPL) from Intel and, instead, we mainly turn our focus on various dense linear algebra solvers.

II. Background

This Section recalls the fundamental principles of mixed precision solvers with iterative refinement and tree reduction based algorithms. The goal of both algorithmic optimizations is to eventually speed up the numerical solver.

Mixed Precision Solvers with Iterative Refinement. Mixed precision solvers are ubiquitous numerical algorithms for solving systems of linear equations. Let us say we want to solve the system $Ax = b$, where A is the matrix, b the right hand side and x the corresponding vector solution. There are two steps to calculate the final solution. The matrix A is first reduced using the LU factorization ($A = LU$), respectively (L and U are triangular matrices). This step is performed in single precision arithmetic. Once factorized, two triangular solves are applied (i.e., forward and backward substitution) again in single precision arithmetic. Secondly, the corresponding residual is computed in double precision arithmetic and the iterative refinement procedure can initiate until the appropriate accuracy has been reached. While the first step is compute intensive (O(n^3)) and represents the bulk of the computation, the second step is memory bound (depending on the size of the right side b) and counts for (O(n^2)). Depending on the conditioning of the original matrix, the iterative refinement may not converge to the desired solution accuracy. Further details on the error analysis and the error bounds are available in [7], [22].

Tree Reduction Algorithms. Tree reduction is a numerical mechanism used to break a sequence of successive tasks in order to expose more parallelism. This may require the implementation of new kernels. The list of successive tasks can now be expressed through a bottom-up tree reduction, where at each level of the tree, the contributions from each leaf are merged until the final result is obtained at the top of the tree. This turns out to be critical when solving overdetermined systems of equations using the QR factorization, in which the matrix describing the numerical problem is tall-and-skinny [6], [14]–[16]. The matrix is split into sub-blocks, in which a local QR factorization is performed. Then, the tree reduction algorithm computes the final factors after successive merging steps. Therefore, due to the matrix shape, introducing a tree reduction strategy allows to substantially increase the level of concurrency.

III. From Block to Tile Algorithms

This section recalls the essence of block and tile algorithms and describes how the block algorithmic versions of mixed precision solvers and tree reduction based factorizations (as implemented in LAPACK) had to be reformulated into tile algorithms (as implemented in PLASMA) to effectively exploit parallelism from multicore architectures.

Fundamental Principles of Block Algorithms. The block algorithms implemented in LAPACK leverage the idea of blocking to limit the amount of bus traffic in favor of a high reuse of the data that is present in the higher level memories which are also the fastest ones. Block algorithms are characterized by two successive phases: panel factorization and update of the trailing submatrix. During the panel factorization, the transformations are only applied within the panel. The panel factorization is very rich in Level 2 BLAS operations because the transformations are singly applied. Once accumulated within the panel, those transformations are applied to the rest of the matrix (the trailing submatrix) in a blocking manner leading to Level 3 BLAS operations. The idea of blocking revolves around an important property of Level-3 BLAS operations, the so called surface-to-volume property, that states that $O(n^3)$ floating point operations are performed on $O(n^2)$ data. Because of this property, Level-3 BLAS operations can be implemented in such a way that data movement is limited and reuse of data in the cache is maximized. Blocking algorithms consists in recasting linear algebra algorithms in a way that only a negligible part of computations is done in Level-2 BLAS operations (where no data reuse possible) while the most part is done in Level-3 BLAS. Last but not least, the parallelism within LAPACK occurs only at the level of the BLAS routines, which follows the expensive fork-join model. Basically, all processing units need to synchronize before and after each call to BLAS kernels.

Fundamental Principles of Tile Algorithms. The core idea of tile algorithms is to transform the original matrix stored in column-major data layout to tile data layout (TDL) where each data tile is contiguous in memory.This may demand a complete reshaping of the standard numerical algorithm. The panel factorization as well as the update of the trailing submatrix are then decomposed into several fine-grained tasks, which better fit the memory of the small core caches. The parallelism is then no longer hidden inside the BLAS routines but rather is brought to the fore. The whole computation can then be represented with a directed acyclic graph (DAG), where nodes are computational tasks and edges represent the data dependencies among them. Next, it becomes critical to efficiently schedule the sequential fine-grained tasks across the processing units. The dynamic runtime environment system QUARK [23] is used to distribute the tasks as soon as the data dependencies are satisfied.

Mixed Precision Solvers and Tree Reduction Techniques using Tile Algorithms. Once the necessary kernels developed, expressing both numerical techniques using tile algorithms was a trivial task, thanks to the high productivity provided by the dynamic runtime system [15], [16], [24]. Moreover, QUARK was able to exploit the lookahead opportunities popping up during the single precision LU factorization of the mixed precision solvers. It also initiates the computation of the next tree level while the previous one is still under processing. The original synchronization points seen in the LAPACK library between the panel and update phases of the factorization become now obsolete and are safely removed, leaving the execution dataflow proceeds smoothly.

The next Section describes the experimental environment used to perform the parallel performance as well as the power profiling of mixed precision solvers with iterative refinement and tree reduction algorithms using block and tile algorithms.

IV. Experimental Setting

This Section describes the experimental environment composed of computer systems, software libraries and tools for power data collection.

Machine Descriptions. We have conducted our performance and power measurements on two shared-memory platforms. The first machine is a dual-socket quad-core Intel Xeon system from Virginia Tech, clocked at 2.8GHz with 8GB of memory. The second system is an Intel Sandy Bridge processor with 8 cores, model Xeon® E5-2690 with nominal frequency 2.90 GHz and Turbo Boost 2.0 technology. The motherboard featured two sockets for a total of 16 cores.

Software Environment. On the tested systems, we used Intel compilers and Intel MKL (Math Kernel Library) BLAS version dated 2011.3.174. The LAPACK V3.2 and PLASMA V2.4 libraries have been tuned and compiled against MKL BLAS. The testing matrices are all randomly generated. In the dense mixed precision solver tests, the system matrix size is of 20000 and only a single right-hand side is used for all experiments. The code performs two iterations in the iterative refinement stage of the algorithm. In the tree reduction experimentations, the tall-and-skinny matrices are of size 1152000 by 288 in order to provide significant insight.

Power Collection Infrastructures. We measured the power consumption of our numerical algorithms using two frameworks: PowerPack and RAPL associated with PAPI.

PowerPack. PowerPack [8] provides power profiling information for advanced execution systems. The PowerPack frameworkis a collection of software components, including libraries and APIs. Together, they enable system and component-level power profiling correlated to application functions. The correlation of various measurements that happen at different sampling rates (average of 100 ms) are performed out-of-band on a dedicated computer. PowerPack obtains measurements from power meters attached to the hardware of the system by intercepting all available connections from the power supply to the system units on the motherboard. As multicore systems evolve, the framework can be used to indicate the application parameters and the system components that affect the power consumption on the multicore unit. PowerPack allows the user to obtain direct measurements of the major system components' power consumption, including the CPU, memory, hard disk, and motherboard. This fine-grain measurement allows power consumption to be measured on a per-component basis.

PAPI RAPL. On the other hand, the PAPI RAPL counter monitor [25] should be considered as an in-band measurement tool with an adjustable sampling rate that we set to 100 ms to match PowerPack as closely as possible. Because PowerPack requires physical modification of the monitored hardware,it was too complicated for us to install it inside a single system that features RAPL. Instead, we rely on the comparison performed by Intel that testify to the close match between physical power measurements and RAPL on a wide range of benchmarks [9]. We add to this set of tested codes our own tuned implementations of dense linear algebra solvers.

The next Section highlights the power profiling of block and tile algorithms.

V. Performance Results

We refrain from publishing a comprehensive set of performance charts for the LU factorizations as we have done elsewhere [26]. Figure 1 shows the achieved performance

# Cores	Library	Performance
8	LAPACK	**SP** 110.8 **DP** 89.4 **MIX** 92.9
	PLASMA	**SP** 143.2 **DP** 73.61 **MIX** 126.4
16	LAPACK	**SP** 393.4 **DP** 214.9 **MIX** 305.9
	PLASMA	**SP** 613.5 **DP** 302.8 **MIX** 537.4

Figure 1. Asymptotic performance in Gflop/s of mixed precision solvers.

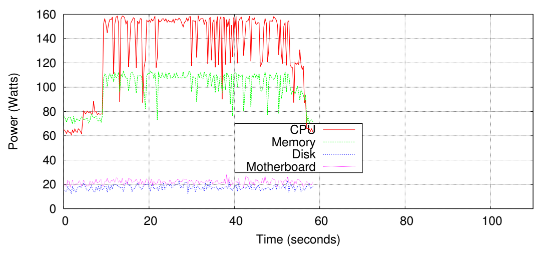
of mixed precision solvers with iterative refinement on both systems for non-symmetric matrices (LU). The mixed precision solvers from PLASMA is more more than 30% faster for LU-based compared to LAPACK on the dual-socket quad-core system. On the Intel Sandy Bridge machine, the mixed precision solvers from PLASMA is more than 1.75X faster for LU-based compared to LAPACK using the total of 16 cores. The next Section focusses on a detailed study of temporal energy and power characteristics. To the best of our knowledge, it is the first such comprehensive study in the context of these advanced dense linear algebra algorithms reported using PowerPack and RAPL.

VI. Power Profiling and Result Interpretations

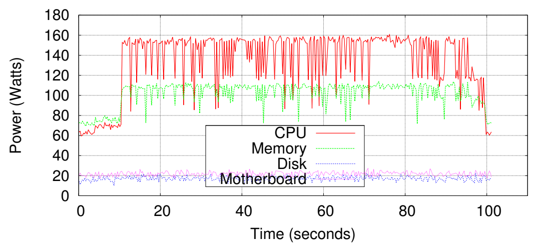
This Section presents the power profiles of tree reduction based QR factorization and LU-based mixed precision solvers with iterative refinement.

Power Profiling of Communication-Reducing Algorithms. Figure 2 shows four graphs drawing the power rate of the standard QR factorization on a tall-and-skinny matrix size of 1152000 by 288 using the dual-socket quad-core Intel Xeon system from Virginia Tech. Figure 2(a) shows the LAPACK version. The pics and dips are representative of the panels and updates computational stages. Lots of pressure is put on the memory bus due to the high number of level 2 BLAS operations happening for such matrix shape. Moreover, the degree of parallelism is hindered by the block algorithms, in which parallelism happens only at the BLAS call level. Figure 2(b) pictures the vendor version of the QR factorization on the same matrix. The power rate curves are smoothed, probably by a better memory usage, which reduces data movement. The elapsed time is reduced roughly by 50% compared to the LAPACK version. Figure 2(c) introduces the PLASMA implementation of the QR factorization using tile algorithms. The tile data layout allows for a better cache usage. The initialization of the matrix is performed in parallel which explains the high power rate in the beginning of the CPU curve. Although the degree of parallelism is still limited by the successive tasks, the elapsed time is yet reduced by more than 50% compared to the vendor version. Figure 2(d) shows the impact on the power rate when the tree reduction technique is integrated. This generates a tremendous amount of parallel tasks, especially during the merging steps of the tree reduction. In fact, the CPU curve reaches the highest power rate when compared with the previous implementations. The elapsed time is again reduced by 50%, compared to the tile QR factorization from PLASMA. Similar power rate curves can be generated using RAPL procedure on the Intel Sandy Bridge machine.

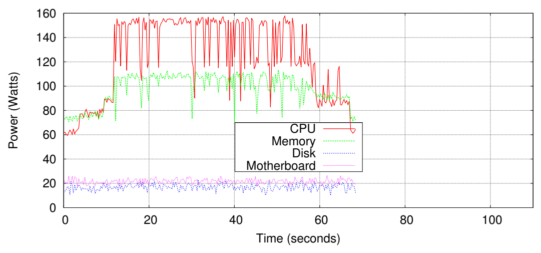
Power Profiling of Mixed Precision Solvers with Iterative Refinement. In this Section, we study the mixed precisions solvers with iterative refinement using LU factorizations (non-symmetric matrix). Figures 3 and 4 highlight the power profile generated through PowerPack using the LAPACK and PLASMA implementations, respectively. The CPU and memory power rate variations are removed in the corresponding PLASMA power profiles due to a better memory reuse. In fact, the PLASMA CPU power curve is higher than the LAPACK one because the dynamic runtime QUARK tries to utilize as much as possible the available cores. The single precision power curves in Figures 3(a) 4(a) are about half those of double precisions in Figures 3(b) 4(b). Similar remarks can be done for the power profiles generated through RAPL on the Intel Sandy Bridge system with regards to single and double precision arithmetics (see Figures 5 and 6). Figures 3(c), 4(c), 5(c), 6(c) show the profiles of the mixed-precision algorithms [12], [17], [20], [27], [28]. Each of them contains the stage of a standard dense factorization stage followed by an iterative refinement stage that brings the desired accuracy to the solution. This is clearly visible as the power consumption pattern at the end of the graphs from these figures. The power draw by the CPU drops appreciatively while the energy consumption related to the main memory increases. Even if we only look at the memory controller as is the limitation for the RAPL measurements.



(a) Single precision solver.

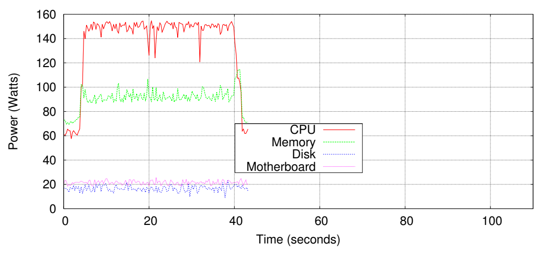


(b) Double precision solver.

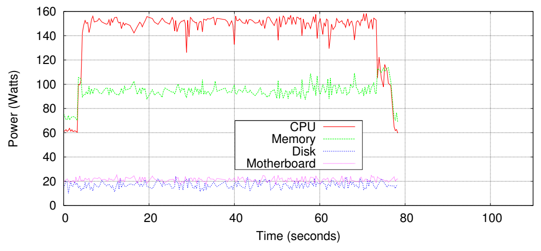


(c) Mixed precision solver.

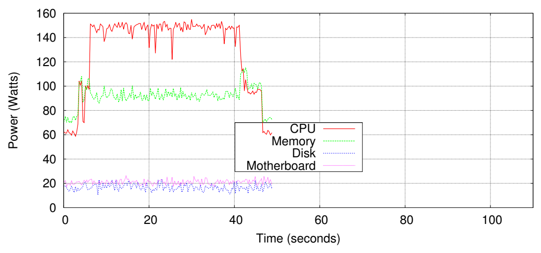
Figure 3. Power Profiling of LAPACK LU Factorization with PowerPack.



(a) Single precision solver.



(b) Double precision solver.

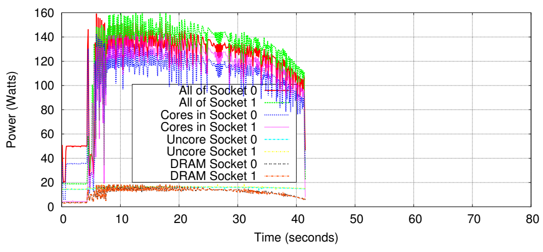


(c) Mixed precision solver.

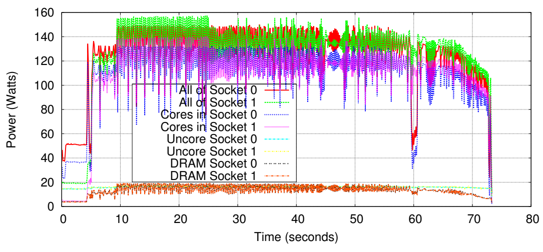
Figure 4. Power Profiling of PLASMA LU Factorization with PowerPack.

VII. PowerPack/RAPL Qualitative Comparisons

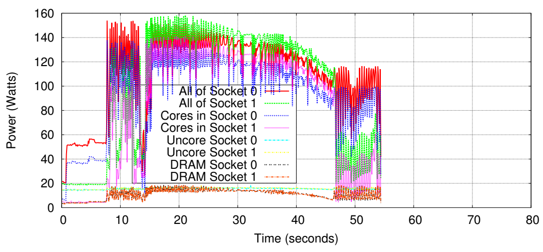
Our intention is to obtain a comparative perspective on the results between the system with the PowerPack installation and an Intel Sandy Bridge system capable of RAPL diagnostics. For our experiments, we have created a software monitoring tool [25] that matches closely the available sampling rate of the standard PowerPack installation that we used for our PowerPack experiments. Internally, the average sampling rate is higher for RAPL [9] but purposefully limit it to match the average rate of PowerPack and achieve as close as possible correspondence between the tested system. The limiting factor for RAPL's sampling rate is delay of transitions between power states, thermal inertia of sensing infrastructure, and the need for synchronization between internal hardware counters that are used for measuring energy consumption of a chip. PowerPack on the other hand, has to correlate in time multiple sources of power readings that have varying sampling rates. In our multiple experiments, the average sampling rate of PowerPack was about 100 ms which was subsequently chosen for our RAPL monitoring tool. As can be seen in the power consumption figures, this sampling rate was sufficient to show important transitions undergone by the dense linear solvers that we have tested. In fact, both measuring techniques offer similar level of detail as proven by the general shape of power curves in the presented Figures. In its core, Intel RAPL is an energy model implemented in hardware with high degree of accuracy [9]. One practical consequence of this is fairly large variability around the power-draw's steady state as well as downward shift of average power reported by the counters. This can be seen as multiple horizontal ledges in Figure 6(b) which are not present in the corresponding measurement performed with PowerPack and shown in Figure 4(b). It may be explained by the way that RAPL the model adapts to the internal measurements with the Exponential Weighted Moving Average algorithm [9], which makes it more sensitive to power fluctuations. One obvious drawback of Intel's RAPL is lack of power measurements for the main memory DIMMs. In Figures 5 and 6, the curves labelled "DRAM" report only the power drawn by the memory controller inside the processor chip – the actual power fed into the main memory DIMMs is not included in the measurement. Needless to say, this is an issue because power consumption for the main memory DIMMs is quickly becoming a substantial portion of the energy budget of a modern and future HPC



(a) Single precision solver.

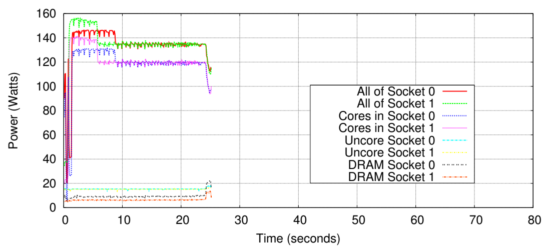


(b) Double precision solver.

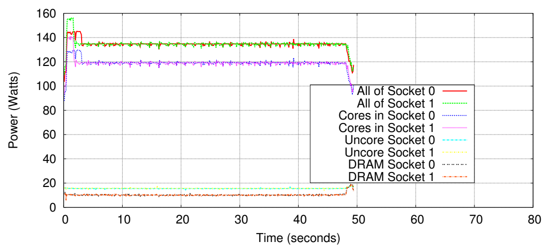


(c) Mixed precision solver.

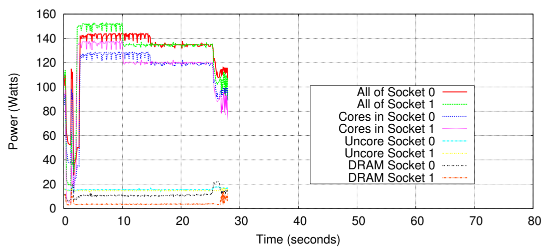
Figure 5. Power Profiling of LAPACK LU Factorization with RAPL.



(a) Single precision solver.



(b) Double precision solver.



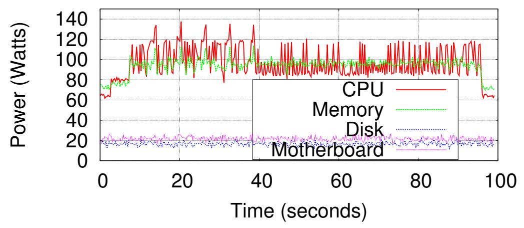
(c) SP-DP.

Figure 6. Power Profiling of PLASMA LU Factorization with RAPL.

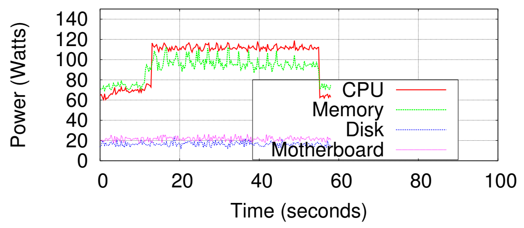
systems [29]. And this large energy consumption may easily be seen by the area below main memory curves from the PowerPack installation as indicated in Figures 3 and 4. Just as PAPI have become an indispensable performance evaluation tool, RAPL could be regarded as yet another aide performance monitoring tools. By showing a compound reading of multiple counters at once, RAPL avoids the limitation of having only few counters available and the associated problems of counter multiplexing. If a given algorithm and its implementation are believed to be optimal, then saving energy consumption requires minimizing the execution time. Under such assumptions and with maximum possible power-draw, RAPL indication of consumed energy may be viewed as a gauge of sufficient performance level. If not enough energy is consumed, it could be easily deduced that the performance level measured, say, in Tflop/s, is not satisfactory. This argument is easily obtained by following the internal implementation of the RAPL algorithm and the energy model it is based on [9]. In years to come, this might not be the case as the users received more comprehensive tools to gauge energy consumption of the chip. In the future, RAPL might offer even finer resolution and the performance monitoring tools will adapt to include first class support for it. This will allow a more energy cognizant application tuning and monitoring. We believe that understanding application energy needs and its overall power profile will enrich the understanding of critical portions of scientific codes and give complementing perspective on performance and time to solution of computationally intensive applications. In the end, we hope it will have offered a helping hand in addressing software challenges at ExaScale [30].

VIII. Summary and Future Directions

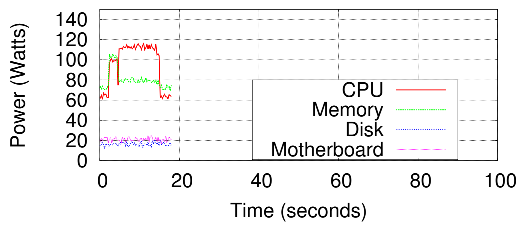
We have presented comparison of measurement techniques for detailed energy consumption and power profiles. We have used some more advanced and less commonly used routines for efficiently solving systems of linear equations. Our finding indicates that RAPL offers a viable alternative to physical power meters for the tested algorithms. The only drawback we observe is lack of interfaces that would allow measuring the energy consumption of the main memory as RAPL only offers counters related to the operation of the main memory controller. We would like also to experiment Dynamic Voltage Frequency Scaling (DVFS) and assess its impact on the overall application performance. A very promising future direction is to use NVIDIA



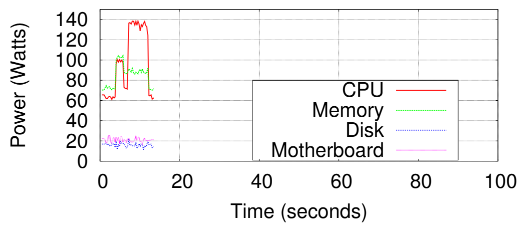
(a) LAPACK with multithreaded BLAS.



(b) MKL with multithreaded BLAS.



(c) PLASMA with sequential BLAS.



(d) PLASMA Tree Reduction with sequential BLAS.

Figure 2. Power Profiling of QR Factorization with PowerPack on a tall-and-skinny matrix using eight cores.

Management Library: Recent NVIDIA GPUs can report power usage via the NVIDIA Management Library (NVML) [31]. The `nvmlDeviceGetPowerUsage()` routine exports the current power; on Fermi C2075 GPUs it has milliwatt resolution within $\pm$5W and is updated at its sampling rate is roughly 60Hz. The power reported is that for the entire board, including the GPU and memory. We would also like to extend our work to distributed memory environment whereby a comprehensive view of energy efficiency may be obtained for large scale clusters. This, combined with energy information from the interconnect, will be the subject of our future investigation of the distributed memory versions of the solvers we tested in this current paper. Finally, RAPL has been designed not just for monitoring instantaneous energy consumption but also for capping the total energy and power at the software level. In this regard it is somewhat similar to Dynamic Frequency and Voltage Scaling (DFVS) – initially only available on mobile versions of processors and now a main stay on the server editions of multicore chips. We would like to look into comparing effects of DVFS with that of RAPL on the platforms that features these technologies side by side.

References

[1] J. Dongarra and P. e. a. Beckman, "The international exascale software project roadmap," University of Tennessee EECS Technical Report, Tech. Rep. UT-CS-10-654, May 30 2010.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1999.

[3] *PLASMA Users' Guide, Parallel Linear Algebra Software for Multicore Architectures, Version 2.4.5, University of Tennessee Knoxville*, University of Tennessee Knoxville, November 2011.

[4] F. G. V. Zee, E. Chan, and R. A. van de Geijn, "libflame," in *Encyclopedia of Parallel Computing*, D. A. Padua, Ed. Springer, 2011, pp. 1010–1014. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-09766-4

[5] H. Ltaief, P. Luszczek, and J. Dongarra, "Profiling High Performance Dense Linear Algebra Algorithms on Multicore Architectures for Power and Energy Efficiency," in *EnA-HPC 2011: Second International Conference on Energy-Aware High Performance Computing*, Hamburg, Germany, Sept 7-9 2011.

[6] A. Pothen and P. Raghavan, "Distributed orthogonal factorization: Givens and Householder algorithms," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, pp. 1113–1134, 1989.

[7] C. B. Moler, "Iterative refinement in floating point," *J. ACM*, vol. 14, no. 2, pp. 316–321, Apr. 1967. [Online]. Available: http://doi.acm.org/10.1145/321386.321394

[8] 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 Transactions on Parallel and Distributed Systems*, vol. PDS-21, no. 5, pp. 658–671, May 2010.

[9] E. Rotem, A. Naveh, D. Rajwan, A. Anathakrishnan, and E. Weissmann, "Power-management architecture of the Intel microarchitecture code-named Sandy Bridge," *IEEE Micro*, vol. 32, no. 2, p. 2027, 2012.

[10] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A portable programming interface for performance evaluation on modern processors," *International Journal of High Performance Computing Applications*, vol. 14, no. 3, p. 189204, 2000.

[11] "Performance Application Programming Interface (PAPI). Innovative Computing Laboratory, University of Tennessee. Available at http://icl.cs.utk.edu/papi/."

[12] A. Buttari, J. Dongarra, J. Langou, J. Langou, P. Luszczek, and J. Kurzak, "Mixed precision iterative refinement techniques for the solution of dense linear systems," *IJHPCA*, vol. 21, no. 4, pp. 457–466, 2007. [Online]. Available: http://dx.doi.org/10.1177/1094342007084026

[13] R. Babich, M. A. Clark, and B. Joó, "Parallelizing the QUDA library for multi-GPU calculations in lattice quantum chromodynamics," in *SC*. IEEE, 2010, pp. 1–11. [Online]. Available: http://dx.doi.org/10.1109/SC.2010.40

[14] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, "Communication-optimal parallel and sequential QR and LU factorizations," *SIAM J. Scientific Computing*, vol. 34, no. 1, 2012. [Online]. Available: http://dx.doi.org/10.1137/080731992

[15] B. Hadri, H. Ltaief, E. Agullo, and J. Dongarra, "Tile QR factorization with parallel panel processing for multicore architectures," in *IPDPS*. IEEE, 2010, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2010.5470443

[16] H. Ltaief, P. Luszczek, A. Haidar, and J. Dongarra, "Enhancing Parallelism of Tile Bidiagonal Transformation on Multicore Architectures using Tree Reduction," in *Parallel Processing and Applied Mathematics*, Torun, Poland, 2011.

[17] A. Buttari, J. Dongarra, J. Kurzak, P. Luszczek, and S. Tomov, "Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy," vol. 34, no. 4, p. 17, Jul. 2008, article 17, 22 pages.

[18] H. Anzt, V. Heuveline, B. Rocker, M. Castillo, J. C. Fernández, R. Mayo, and E. S. Quintana-Orti, "Power consumption of mixed precision in the iterative solution of sparse linear systems," in *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, ser. IPDPSW '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 829–836. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2011.226

[19] H. Anzt, B. Rocker, and V. Heuveline, "Energy efficiency of mixed precision iterative refinement methods using hybrid hardware platforms," *Computer Science - Research and Development*, vol. 25, pp. 141–148, 2010, 10.1007/s00450-010-0124-2. [Online]. Available: http://dx.doi.org/10.1007/s00450-010-0124-2

[20] J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, and J. Dongarra, "Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems)," in *ACM/IEEE SC 2006 Conference (SC'06)*, Nov. 2006, p. 50.

[21] C. Bekas and A. Curioni, "A New Energy Aware Performance Metric," *Computer Science - R&D*, vol. 25, no. 3-4, pp. 187–195, 2010. [Online]. Available: http://dx.doi.org/10.1007/s00450-010-0119-z

[22] N. Higham, *Accuracy and Stability of Numerical Algorithms, Second Edition*. SIAM, 2002.

[23] A. YarKhan, J. Kurzak, and J. Dongarra, "QUARK Users' Guide: QUeueing And Runtime for Kernels," *University of Tennessee Innovative Computing Laboratory Technical Report ICL-UT-11-02*, 2011.

[24] E. Agullo, B. Hadri, H. Ltaief, and J. Dongarrra, "Comparative study of one-sided factorizations with multiple software packages on multi-core hardware," in *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. New York, NY, USA: ACM, 2009, pp. 1–12.

[25] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, "Measuring energy and power with PAPI," in *Power-Aware Systems and Architectures (PASA) PASA2012 in conjunction with the 41st Annual International Conference on Parallel Processing (ICPP 2012)*, Pittsburgh, USA., September 10-13 2012.

[26] J. Dongarra, M. Faverge, H. Ltaief, and P. Luszczek, "Achieving numerical accuracy and high performance using recursive tile lu factorization," *University of Tennessee Computer Science Technical Report UT-CS-11-688 (also LAPACK Working Note 259), Submitted to Journal of Concurrency and Computation: Practice and Experience*.

[27] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, and S. Tomov, "Accelerating scientific computations with mixed precision algorithms," *Computer Physics Communications*, vol. 180, no. 12, pp. 2526 – 2533, 2009, 40 YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures. [Online]. Available: http://www.sciencedirect.com/science/article/B6TJ5-4TX7976-1/2/a85cf5566583a3c10996c05925efa9c9

[28] J. Kurzak and J. J. Dongarra, "Implementation of mixed precision in solving systems of linear equations on the Cell processor," *Concurrency Computat.: Pract. Exper.*, vol. 19, no. 10, pp. 1371–1385, 2007.

[29] P. Kogge, 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, R. S. Williams, and K. Yelick, "Exascale computing study: Technology challenges in achieving exascale systems," Department of Computer Science and Engineering, University of Notre Dame, Tech. Rep. TR-2008-13, September 28 2008.

[30] S. Amarasinghe, D. Campbell, W. Carlson, A. Chien, W. Dally, E. Elnohazy, M. Hall, R. Harrison, W. Harrod, K. Hill, J. Hiller, S. Karp, C. Koelbel, D. Koester, P. Kogge, J. Levesque, D. Reed, V. Sarkar, R. Schreiber, M. Richards, A. Scarpelli, J. Shalf, A. Snavely, and T. Sterling, "Exascale software study: Software challenges in extreme scale systems," Rice University, Tech. Rep., September 14 2009.

[31] *NVML Reference Manual*, 2012, nVIDIA.