

The Impact of Voltage-Frequency Scaling for the Matrix-Vector Product on the IBM POWER8

Sandra Catalán¹(✉), A. Cristiano I. Malossi²,
Costas Bekas², and Enrique S. Quintana-Ortí¹

¹ Dpto. de Ingeniería y Ciencia de Computadores,
Universidad Jaume I, 12071 Castellón, Spain
{catalans,quintana}@uji.es

² IBM Research–Zurich, Foundations of Cognitive Solutions,
8803 Rüschlikon, Switzerland
{acm,bek}@zurich.ibm.com

Abstract. The physical limitations of CMOS miniaturization have promoted understanding the interplay between performance and energy into a primary challenge. In this paper we contribute towards this goal by assessing the effect of voltage and frequency scaling (VFS) on the energy consumption of the dense and sparse matrix-vector products. The optimization of the sparse kernel, from the perspective of both performance and energy efficiency, is especially difficult due to its irregular memory access pattern, but the potential benefits are remarkable because of its varied applications.

Our experiments with a small synthetic training set show that it is possible to build a general classification of sparse matrices that governs the optimal VFS level from the point of view of energy efficiency. More importantly, this characterization can be leveraged to tune VFS for a major portion of the University of Florida Matrix Collection, when executed on the IBM Power8, yielding significant gains with respect to a (power-hungry) configuration that simply favours performance.

Keywords: Energy efficiency · Voltage-frequency scaling · Performance prediction · Performance metrics · Matrix-vector product · IBM POWER8

1 Introduction

The matrix-vector product is an important numerical kernel as well as one of the 7+ dwarfs [3] proposed for the evaluation of parallel programming models and architectures. In particular, the sparse instance of the matrix-vector multiplication (SPMV) underlies the HPCG benchmark [8], and is also a crucial kernel for

IBM POWER8: IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp. in the United States, other countries, or both. Other product and service names might be trademarks of IBM or other companies.

the solution of linear systems and eigenvalue problems arising in many scientific and engineering applications [21]. Furthermore, the connection between sparse linear algebra and graph algorithms has been recently exploited by a new class of algorithms, based among others on SPMV, to tackle the vast volume of information that is common in social networks and other data analytic processes [5, 14].

In this paper, we perform a complete experimental analysis of a generic implementation of the matrix-vector product on a current multi-threaded architecture with support for dynamic voltage-frequency scaling (VFS). Our study focuses on the energy efficiency using the energy-per-flop metric as reference. This analysis is timely because even though the matrix-vector product kernel has been extensively analyzed and optimized from the point of view of performance, the number of studies that investigate its energy consumption is limited. This is especially relevant since power/energy are factors which constrain the performance of current processor designs for the high performance computing systems running the aforementioned numerical and graph-related applications [9, 17].

In particular, this work makes the following specific contributions with respect to the energy optimization of SPMV:

- We analyze several critical parameters with respect to matrix size, sparsity degree, and non-zero clustering of the sparse matrix that drive the energy efficiency of this kernel on a modern multithreaded architecture.
- We derive a simple recipe to optimize VFS for a SPMV operation involving any given sparse matrix that exploits the aforementioned relevant parameters from the graph representing the sparse matrix.
- We demonstrate the robustness of our VFS optimization recipe by applying it to optimize the energy performance of the entire University of Florida Sparse Matrix Collection (UFMC) [1] on the IBM Power8.
- We evaluate the energy gains that an appropriate adaption of VFS can yield with respect to an energy-oblivious approach that considers performance as the only optimization goal.

Our driving motivation is the design of an easy-to-use and widely-applicable strategy to significantly reduce energy consumption of SPMV without the need of a per-case analysis. Indeed, finding the optimal VFS for each individual sparse problem is unrealistic as well as unaffordable in practical production scenarios, where the sparsity pattern may change from one application to the next, and time cannot be spent on preliminary fine tuning energy-benchmarks.

The rest of the paper is structured as follows. In Sect. 2 we briefly review some related works and next, in Sect. 3, we describe the experimental setup. In Sects. 4 and 5 we analyze the energy consumption of the sparse matrix-vector kernel with dense and sparse matrices, respectively. We close the paper with a few concluding remarks in Sect. 6.

2 Related Work

There exists a large volume of work addressing the performance optimization of SPMV; see, e.g., [5, 16, 22, 23] and the references therein. Among others,

pOSKI [6] is a multithreaded library for SPMV that leverages automatic search over multiple implementations and sparse data layouts to optimize performance on multicore processors. Model-driven optimization of SPMV for data-parallel accelerators has been studied in [7, 10].

The number of efforts dedicated to the energy modeling and optimization of SPMV is significantly more reduced. In [20] we introduced a systematic methodology to derive reliable time and power models for algebraic kernels employing a bottom-up approach. However, the recipe resulted a bit cumbersome to leverage, requiring a large number of calibration tests. In [19] we devised a systematic machine learning algorithm to classify and predict the performance and energy costs of the SPMV kernel, but did not consider the effect of VFS. The work in [2] presents an extensive experimental study of the interactions occurring in the triangle performance-power-energy for the execution of a pivotal numerical algorithm, the iterative Conjugate Gradient (CG) method, on an ample collection of parallel multithreaded architectures. However, that work does not produce a recipe to optimize VFS for any given sparse problem. Other work related to modeling sparse linear algebra operations can be found in [11, 13, 18].

3 Experimental Setup

3.1 Hardware

The target platform for our analysis is the IBM Power System S812L. The POWER8 processor in this system, fabricated on 22 nm silicon, features 2 sockets, each with 6 cores, offering hardware support for up to 8-way simultaneous multi-threading (SMT) as well as dynamic VFS. Each core in the IBM POWER8 is furnished with a 512-KB L2 cache. Furthermore, the chip contains a shared L3 cache of 8 MB per core, and 16 MB of L4 cache per buffer, with up to 8 buffers per socket [4]. The server was also equipped with 64 GB of DDR RAM.

Table 1 displays the frequency-voltage configuration pairs and the idle power dissipated by the system, measured during the execution of a sleep test,¹ for two scenarios: socket+DDR (“SD”) only vs the full server (“Node”). Power measures were obtained using AMESTER [15]. This tool runs on a separate server and connects to the service processor of the node in order to obtain voltage/frequency/power-per-core samples from several sensors, while avoiding interference with the workload. Using this information, we calculate the time-per-flop and *net* power-per-core (without the idle power), yielding the *net* energy per-flop-and-core from their product. All energy consumption values reported next refer to the net energy-per-flop and core.

The codes were compiled using IBM’s `mpcc 13.01.0003.0000`, with the flags: `-O3 -qprefetch=dscr=0 -qhot -qstrict -qsmp=noauto:omp -qthreaded -qsimd=auto -qaltivec -q64 -qarch=pwr8`. Each test was repeated for at least 60s,

¹ Although the idle power could be determined with higher accuracy by via an extrapolation to the power usage with 0 cores, we believe that the sleep-based test provides enough precision for our purposes.

Table 1. Voltage-frequency pairs and idle power in the IBM POWER8 processor.

Config.	Frequency F (GHz)	Voltage V (mV)	Idle power, P^{idle} (W)	
			SD	Node
C_1 ●	2.13	875.0	147.9	358.3
C_2 ●	2.53	931.3	151.9	367.7
C_3 ●	2.96	987.5	158.1	380.1
C_4 ●	3.36	1,037.5	165.3	392.6
C_5 ●	3.79	1,093.8	173.2	411.2
C_6 ●	4.22	1,187.5	192.6	451.9

and the results average the values from these runs. The experiments targeted a single “socket” of the IBM POWER8 chip (i.e., 6 cores), with either 1 thread or 8 threads per core (1-SMT or 8-SMT, respectively), and all threads/cores collaborating to compute one instance of SPMV.

3.2 Kernel and Implementation

We analyze an implementation of the SPMV $y := A \cdot x$, with sparse matrix $A \in \mathbb{R}^{n \times n}$ and dense vectors $x, y \in \mathbb{R}^n$, based on the CSR (compressed sparse row) storage format [21]. For sparse matrices, CSR offers a fair balance between compression efficiency (as it is one of the most efficient formats for generic sparse matrices on cache-based microprocessors) and architecture-independent performance (since it does not directly exploit graph characteristics that may emerge from the specific physical problem) [22]. The CSR data layout employs a real array for the values of A (`A_val`), and two auxiliary integer arrays (`col_ind` and `row_ptr`) to maintain (respectively) the column indices of the nonzero entries in A and the initial/final index of each row of A within the other two arrays [21]. All our experiments employ double precision floating-point arithmetic so that the values of A, x, y occupy $s_d = 8$ bytes each. Each component of the indexing integer arrays occupies $s_i = 4$ bytes. Therefore, storing an $n \times n$ matrix with n_z nonzero entries in this format requires $M_S = n_z(s_d + s_i) + (n + 1)s_i$ bytes, and x, y occupy $V_S = ns_d$ bytes each.

The implementation of SPMV in CSR format is illustrated in Fig. 1. The optimization is left to the compiler, except for some minor details omitted for simplicity. The parallelization strategy distributes the computation of the entries of y among the threads/cores (via the OpenMP `#pragma omp` directive before the outer loop).

In the operation $y := A \cdot x$, there is no reuse of the entries of A and the only opportunity to exploit data locality is in the accesses to x, y . In the CSR version of SPMV, the entries of `A_val` and `col_ind` are streamed from the memory layer where they reside into the processor register file with unit stride; each entry of y is loaded into a register once and re-used until it has been completely updated; and the re-use factor of x depends on the sparsity pattern of A .

```

1 void SpMV_CSR( int n, int * row_ptr, int * col_ind,
2               double * A_val, double * x, double * y ) {
3     int i, j;    double tmp;
4
5     #pragma omp parallel for private ( tmp, j ) schedule static
6     for ( i = 0; i < n; i++ ) {
7         tmp = 0.0;
8         for ( j = row_ptr [i]; j < row_ptr [i+1]; j++ )
9             tmp += A_val [j] * x [col_ind[j]];
10        y[i] += tmp;
11    }
12 }

```

Fig. 1. SPMV based on the CSR format.

4 Tuning VFS for the Dense Matrix-Vector Product

We commence our analysis by considering an $n \times n$ dense matrix-vector product kernel, GEMV, computed via the code in Fig. 1. While a dense matrix can be more efficiently stored as a conventional 1-D array, in column- or row-major order, this initial study will offer us some preliminary insights on the energy behavior of this memory-bound operation. For the following experiments, we use two square dense matrices, of dimension $n = 312$ and $30,512$ (with $n_z = n^2$ nonzeros). Taking into account the CSR memory layout, and the fact that all cores collaborate in the execution of the same matrix-vector product, the data for these two problems respectively requires about 1.15 MB and 10.6 GB. Thus, the small case easily fits into the on-chip L3 cache (8 MB/core), while the larger problem can only be stored in the off-chip DDR RAM.

Scaling the voltage and frequency (VFS) can be expected to produce an effect on performance and power dissipation which, in turn, produces an impact on the energy consumption. In principle, one could expect that a change of frequency results in a proportional variation of the GFLOPS (billions of flops per second). The left-hand side plot in Fig. 2 investigates the behaviour as the frequency is increased, and the socket is populated with 1 or 8 threads per core (1-SMT and 8-SMT). To capture the theoretical linear relation between the GFLOPS and the frequency, both metrics are normalized in the figure with respect to those observed for C_1 . On one hand, when operating with the off-chip problem, the GFLOPS rate attained with 1-SMT stagnates for the two higher frequency rates while, for 8-SMT, the performance does not vary with the frequency. These results show scenarios where the DDR bandwidth is saturated for our GEMV code. On the other hand, the GFLOPS rate grows linearly with the frequency for the L3 on-chip case, independently of the number of threads.

The analysis from the point of view of power dissipation is more complex. Concretely, for a given voltage-frequency configuration pair $C_A = (V_A, F_A)$, the power dissipation can be decomposed into its static and dynamic components which depend, respectively, on V_A^2 and $V_A^2 \cdot F_A$ [12]. We can assume that the idle power (see Table 1) is mostly due to leakage (static power), while a substantial fraction of the net power is due to the application's activity (dynamic power). The right-hand side plot in Fig. 2 compares the experimental net power ratio

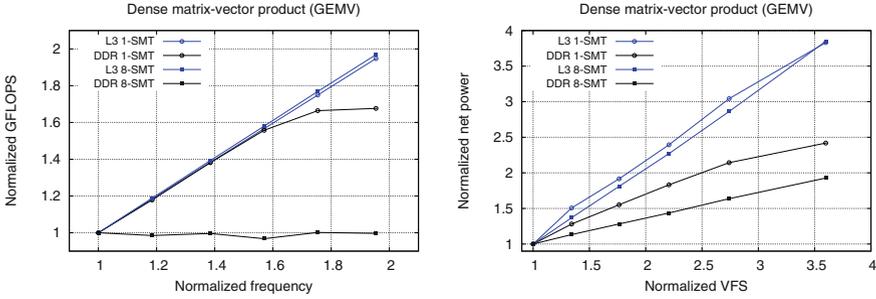


Fig. 2. Performance (left) and SD power consumption (right) for GEMV. (Color figure online)

Table 2. Normalized energy consumption with respect to C_1 for 8-SMT and different VFS levels. Best choices are highlighted in green.

Config.	SD		Node	
	L3	DDR	L3	DDR
C_1 ●	1.00	1.00	1.00	1.00
C_2 ●	0.90	1.06	0.87	1.04
C_3 ●	0.83	1.11	0.79	1.08
C_4 ●	0.79	1.22	0.73	1.16
C_5 ●	0.78	1.26	0.70	1.19
C_6 ●	0.82	1.43	0.71	1.32

P_A^{NET} , normalized with respect to that observed when running in C_1 , against the theoretical VFS ratio $(V_A^2 \cdot F_A)/(V_1^2 \cdot F_1)$ during the execution of GEMV. Considering only the 8-SMT cases, there is a clear difference between the L3 case, which shows a perfect match between the theory and the experimental behaviour, and the DDR problem, for which the net power grows at a much lower pace due to the memory bottleneck.

Table 2 illustrates the combined effect of the variations in performance and power dissipation on the energy consumption, identifying the best VFS configuration depending on the problem dimension. The values there are again normalized with respect to those observed for the configuration C_1 . Hereafter, we only report the results obtained with 8-SMT, as they are consistently superior to those obtained with 1-SMT in both performance and energy efficiency.

The previous experiments reveal that the execution time as well as the power consumption of GEMV can be accurately modeled when the problem data fits on-chip. This leads to a straight-forward derivation of energy-related metrics (such as our target net energy-per-flop) and, as illustrated in Table 2, paves the road to a direct optimization of VFS from the perspective of energy efficiency. The behavior of performance and power consumption is more difficult to predict for the DDR problems; however, if the goal is to optimize energy efficiency, the best strategy for GEMV simply runs this kernel at the lowest voltage-frequency pair. Compared with this energy-aware VFS configuration,

an execution of GEMV that simply aims to enhance performance consumes significantly more energy: $1.43\times$ for SD and $1.32\times$ for the node.

5 Tuning VFS for the Sparse Matrix-Vector Product

Unfortunately, the sparsity exhibited by most real applications generally results in irregular access patterns (in CSR, to the entries of x), which may render an imbalanced workload distribution yielding the guidelines derived to adjust VFS for GEMV in Sect. 4 suboptimal for the sparse case.

In [19], we identified a reduced set of critical structural parameters which impact the performance, power, and energy consumption of the CSR implementation of SPMV. These properties were used to generate a small synthetic sparse benchmark (or training set) which was then employed to build a model that accurately predicts performance and energy consumption of any sparse problem. In the following, we investigate whether the same approach, based on a synthetic sparse training set, can produce a strategy to select a (close-to-)optimal VFS configuration for any sparse problem.

5.1 Training Set

In order to determine an appropriate configuration for SPMV, we leverage the benchmark introduced in [19], which is characterized by five parameters:

- Number of rows/columns n and nonzeros n_z .
- Block size: b_s . Many applications lead to sparse matrices where the non-zeros are clustered into a few compact dense blocks in each row. This parameter specifies the number of entries in these blocks, and determines the number of elements of x accessed with unit stride.
- Block density: $b_d = b_s/n_{zr} = b_s n/n_z \in [0, 1]$ is the inverse of the number of blocks per row. With b_s fixed, b_d defines the re-use factor for y .
- Row density: $r_d = n_{zr}/n \in [0, 1]$ is the number of non-zeros per row relative to the row size. With n_{zr} fixed, r_d is an indicator of the probability of finding an entry of x already fetched into a higher level of the cache hierarchy during the computation of a previous entry of y .

For the analysis, we distribute the matrix instances of the training set evenly in the \log_2 space comprised by $b_s \in \{2^0, 2^2, 2^4, \dots, 2^{14}\}$, $b_d \in \{2^0, 2^{-2}, \dots, 2^{-14}\}$, and $r_d \in \{2^0, 2^{-2}, \dots, 2^{-28}\}$. Thus, for a matrix in the training set, the triplet-coordinates (b_s, b_d, r_d) identify a matrix of dimension $n = n_{zr}/r_d = b_s/(b_d r_d)$ with $n_z = n n_{zr}$ nonzeros. As $b_s \leq n_{zr} \leq n$, this distribution yields a total of 162 samples only, yet offers enough variability to characterize sparse matrices from real applications while avoiding a costly calibration.

Figure 3 illustrates a compact representation of the training set, with each matrix identified by a single point (b_s, b_d, r_d) in the 3-D space. The colors of the points identify the optimal VFS configuration, from the perspective of energy consumption, taking into account two scenarios: SD power only or the full node

consumption. The matrix instances in the training set are divided into three categories, according to the size of the SPMV data ($P_S = M_S + 2V_S$): “L3”, “DDR”, and “transition”. The first category includes those instances where the matrix data occupies less than 18 MB and involve small vectors x, y that easily fit into the L3 cache; the second category contains those instances requiring more than 200 MB, which therefore can only reside in the DDR; all the remaining cases are assigned to the transition category.

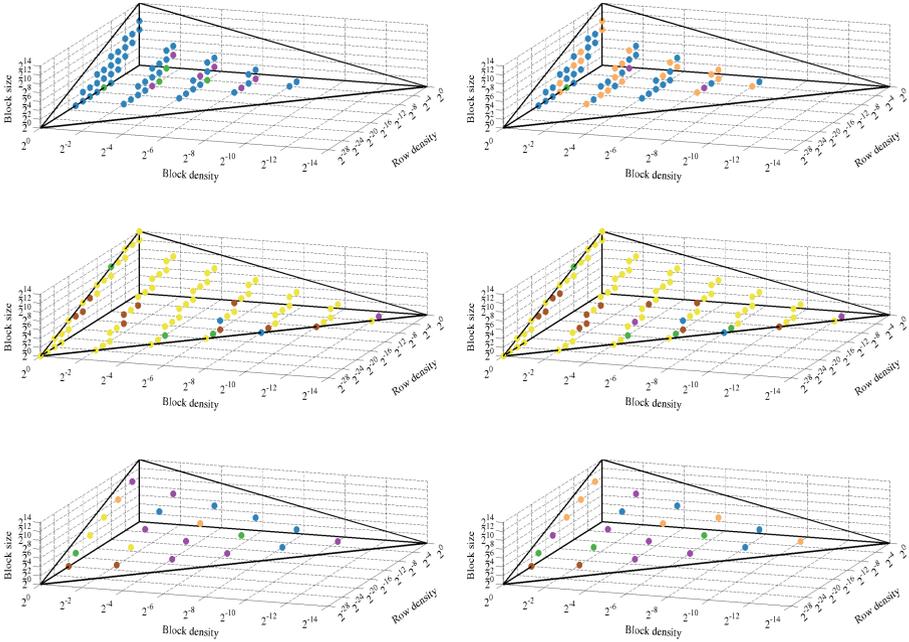


Fig. 3. Optimization of SD (left) and node (right) energy consumption via VFS for the synthetic training set, with matrices in the L3, DDR, and transition categories (top, middle, and bottom, resp.). Color codes: ●: C_1 ; ●: C_2 ; ●: C_3 ; ●: C_4 ; ●: C_5 ; ●: C_6 . (Color figure online)

These results show that, for any scenario/category, there is no single configuration that optimizes VFS for all cases in it. For brevity, let us examine the energy consumed by the SD (scenario) in some detail (left column of plots):

- L3: in most cases, the C_5 configuration (●) is optimal.
- DDR: the C_1 configuration (●) is possibly the best compromise solution. However, we can observe that, as the row density r_c decreases, the C_2 configuration (●) seems to offer a fair alternative.
- Transition: the figures expose no clear winner in this case.

The plots in Fig. 3 offer a quick glance of the impact of VFS depending on the scenario (SD or Node) and category (L3, DDR or transition). We next refine

this information to quantify the overhead incurred by an approach that chooses a single configuration for each scenario/category combination. Consider in particular a category consisting of s problem instances $P = \{p_1, p_2, \dots, p_s\}$, which are executed under a given VFS configuration C_A and scenario S , resulting in a range of values for the energy consumption: $E^S \in \{E^{\text{SD}}, E^{\text{Node}}\}$. Let us denote by $C_{\text{OPT}}(p, S)$ the configuration that minimizes energy consumption for a given problem instance/scenario p/S . With these premises, Table 3 reports the *relative deviation from the energy-optimal VFS configuration* for each problem category and scenario, given (in %) by

$$\frac{100}{s} \cdot \sum_{i=1}^s \left(\frac{E^S(p_i, C_A) - E^S(p_i, C_{\text{OPT}}(p_i, S))}{E^S(p_i, C_{\text{OPT}}(p_i, S))} \right).$$

The ‘‘L3’’, ‘‘DDR’’, and ‘‘Trans(ition)’’ columns in the table thus show the additional energy (overhead in %) that is spent if one chooses a single configuration for all problems in the category, instead of the specific optimal configuration for each problem instance/scenario. Let us analyze one of the scenarios in more detail, namely the SD energy consumption. The results in the table reveal that this excess is acceptable for the instances in the L3 and DDR categories when choosing C_5 (0.9%) and C_1 (2.4%), respectively, but they also expose a significant loss if a single configuration is selected for all cases in the transition category (15.3% in the best case, corresponding to C_2).

Table 3. Deviation (in %) of energy consumption with respect to the optimal configuration for the synthetic training set. Best choices are highlighted in green.

Config.	SD					Node				
	L3	DDR	Trans.	HRD	LRD	L3	DDR	Trans.	HRD	LRD
C_1 	26.7	2.4	18.9	23.6	7.1	39.9	3.2	27.4	34.1	10.3
C_2 	14.7	5.4	15.3	19.8	4.1	24.4	5.0	21.9	28.2	6.1
C_3 	6.9	11.2	18.9	24.6	4.9	12.3	8.8	22.4	29.8	3.6
C_4 	2.2	17.2	17.1	18.8	12.8	5.2	13.6	18.7	21.9	10.4
C_5 	0.9	23.1	17.0	14.9	22.3	0.7	17.1	15.1	14.2	17.1
C_6 	6.2	38.4	21.9	20.0	26.7	2.2	28.7	15.9	15.0	17.8

Motivated by the blurry behaviour of the instances in the transition category, we next analyze these cases in more detail. Table 4 reports the SD energy consumption for the 21 instances in this problem category, normalized with respect to the configuration C_1 . The values in the table show no clear relation between problem size, defined by n and P_S , and the optimal configuration for these instances. On the other hand, among the remaining three problem parameters, only the row density seems to play a role, as high row densities (HRD: $r_d > 2^{-14}$) favour high frequencies, while low row densities (LRD: $r_d \leq 2^{-14}$) benefit more from a lower frequency. This criteria pushes us to split the transition category into two subcategories, LRD and HRD, yielding the deviations in the columns labelled as ‘‘HRD’’ and ‘‘LRD’’ of Table 3, which reveal a significant

Table 4. Normalized energy consumption of SD with respect to C_1 for 8-SMT and different VFS levels. Best choices are highlighted in green.

	Problem parameters					Configuration					
	n	P_S (MB)	b_s	b_d	r_d	C_1	C_2	C_3	C_4	C_5	C_6
HRD	4,096	48.1	1	-10	-2	1	0.98	1.10	0.71	0.78	0.76
	4,096	48.1	4	-8	-2	1	0.79	1.01	1.12	0.70	1.12
	4,096	48.1	16	-6	-2	1	1.01	0.76	0.96	0.75	0.77
	4,096	48.1	64	-4	-2	1	1.06	1.20	1.04	0.74	0.83
	4,096	48.1	256	-2	-2	1	0.91	0.77	0.74	1.08	1.18
	4,096	48.1	1,024	0	-2	1	0.82	1.02	0.79	0.83	0.82
	16,384	48.3	1	-8	-6	1	0.91	0.92	0.87	0.75	0.83
	16,384	48.3	4	-6	-6	1	0.93	0.84	1.05	1.09	1.19
	16,384	48.3	16	-4	-6	1	1.09	1.07	1.07	1.16	0.88
	16,384	48.3	64	-2	-6	1	0.90	1.15	1.12	0.80	1.10
	16,384	48.3	256	0	-6	1	1.04	0.96	1.15	0.94	0.78
	65,536	49.2	1	-6	-10	1	0.97	0.91	0.91	1.06	1.00
	65,536	49.2	4	-4	-10	1	0.99	1.20	0.84	1.14	1.10
	65,536	49.2	16	-2	-10	1	1.11	0.99	0.95	1.16	1.11
65,536	49.2	64	0	-10	1	1.04	1.20	1.03	1.10	1.09	
LRD	262,144	53.0	1	-4	-14	1	0.95	0.94	0.93	1.03	1.23
	262,144	53.0	4	-2	-14	1	1.08	1.00	1.18	1.23	1.25
	262,144	53.0	16	0	-14	1	1.04	1.07	1.03	1.14	1.20
	10,485,766	32.0	1	0	-20	1	0.94	0.99	1.04	1.18	1.32
	10,485,766	68.0	1	-2	-18	1	0.80	0.92	1.05	1.14	0.93
	10,485,766	68.0	4	0	-18	1	1.03	0.94	1.05	1.10	1.17

improvement for the LRD subcategory but no relevant gain for the HRD. To conclude this analysis of the transition cases, we note that LRD consist of just 6 instances which may be too small to derive a strong conclusion.

5.2 Validation with UPMC

The synthetic benchmark is only a small collection of sparse problems, that aims to provide a rough approximation of the sparsity patterns present in real applications, in order to offer some guidance on the energy-optimal VFS configuration. The motivation for this is that choosing the optimal VFS for each individual sparse problem can be unrealistic, as that may require to execute each case at each VFS level to make an appropriate choice. As an alternative, we trade off accuracy (and, therefore, energy efficiency) for flexibility, by using the classification and VFS configuration obtained with synthetic benchmark to dictate the selection for the real applications.

We validate the energy efficiency that can be attained if we base our VFS selection for real sparse matrices on the previous problem classification. For this purpose, we employ 1,202 problem instances from very different real applications in the set UPMC. Among these, 1,044 cases fit into the L3 cache, 75 can be classified as DDR cases, and the rest belong to the transition category, with 69 in HRD and 14 only in LRD.

Table 5 displays the *relative deviation from the energy-optimal VFS configuration* and the performance-optimal configuration for the problem instances in the UPMC. Regarding the energy, when comparing the optimal global VFS

Table 5. Deviation (in %) of energy consumption and performance with respect to the optimal configuration for the UPMC. Best choices are highlighted in green.

Config.	SD energy				Node energy				Performance			
	L3	DDR	HRD	LRD	L3	DDR	HRD	LRD	L3	DDR	HRD	LRD
C_1 	28.2	1.9	17.2	17.2	41.3	3.6	26.0	34.7	-47.91	-11.63	-37.43	-44.96
C_2 	15.8	4.9	10.9	10.9	24.8	4.9	16.6	23.4	-38.62	-10.09	-30.59	-37.44
C_3 	7.1	7.7	7.7	7.4	12.6	5.9	10.8	13.1	-28.39	-6.87	-23.50	-28.17
C_4 	2.4	11.6	6.5	4.3	5.1	7.9	7.0	7.3	-19.21	-4.15	-17.12	-20.50
C_5 	0.5	19.1	5.9	2.8	0.5	13.1	3.8	2.9	-9.95	-3.18	-9.14	-11.89
C_6 	5.1	32.2	11.6	5.2	1.5	22.4	5.8	1.5	-0.01	-0.65	-0.28	0

configuration in the table with those in Table 3, we observe that the training set did actually offer an appropriate guidance to select the energy-optimal VFS for the L3, DDR and HRD cases, for both the SD and node scenarios. On the other hand, the global energy-optimal VFS options for the UPMC problem instances in the LRD category are C_5 (SD) or C_6 (node) instead of those pointed out by the synthetic benchmark. With respect to performance, that when applying the energy-optimal VFS this metric decreases up to 11.63%, except for the LRD cases where it could reach 37.44%. Again, this is due to the fact that the training set prediction does not match the energy-optimal VFS for the UPMC cases that fall into the LRD category. At this point, it is worth reminding that the synthetic collection included only 6 problem instances in this category, with the data for five of them occupying in the range of ($P_S=$)53–68 MB. Compared with this, the UPMC set has 14 instances in the LRD category, but only 4 of this are in the same P_S -dimension range. The fact that the samples in the (synthetic and real) LRD category are few, and that the problem sizes for the synthetic set and the UPMC do not overlap, explains the different energy-optimal VFS configuration determined for each case. However, we emphasize that the LRD category contains only 14 cases out of 1,202 real problems, which is less than 1.2%! For the remaining 98.8% cases, the training set did actually identify a fair classification into categories as well as offer a good VFS selection.

A final question to investigate is the balance between the energy gains vs the performance loss that an energy-aware VFS configuration, based on the categories/VFS levels determined from the previous experimental studies, can yield compared with a conventional performance-oriented VFS selection that simply runs all (real) cases at the highest VFS level. Table 6 shows that, when considering the socket+DDR (SD), an energy-aware VFS configuration can yield savings between 3.0% and 21.0% with respect to the performance-oriented option, and more reduced if we consider the full node consumption. These savings come at a certain cost from the perspective of execution time, reporting a loss of the energy-aware VFS configuration with respect to the performance-oriented case that is between 9.6% and 22.7%.

Table 6. Average energy savings vs performance loss between the energy-optimal and the performance-optimal VFS configurations for the UFMC (denoted as C_E and $C_P = C_6$, respectively). From previous experiments with UFMC, $C_E = C_5$ for the problem instances in the L3 and transition categories, while $C_E = C_1$ for the DDR cases. Performance is measured in time-per-flop.

Metric	SD				Node			
	L3	DDR	HRD	LRD	L3	DDR	HRD	LRD
Energy ratio: C_E/C_P	1.046	1.210	1.054	1.030	1.011	1.097	1.019	0.995
Performance ratio: C_P/C_E	1.104	1.227	1.096	1.125	Same as SD scenario			

6 Concluding Remarks

Voltage-frequency scaling (VFS) is an energy-oriented technology present in current hardware that the operating system/programmer can leverage to adapt the execution pace of an application without modifying the code. Unfortunately, selecting the energy-optimal VFS configuration is both architecture- and application-dependent. For the (sparse) matrix-vector product kernel, our work shows that it is possible to rely on a portable benchmark, consisting of a reduced number of synthetic sparse matrices, to establish a general classification of the problems data, (according to criteria related to problem dimension and sparsity pattern,) and to determine a global energy-optimal VFS configuration for the matrices in each group. Our experiments on a multicore server equipped with an IBM POWER8 show a strong dependence between energy consumption and problem dimension, exposing an interesting trade-off between energy efficiency and performance for this particular kernel.

Our work also analyzed the energy-delay product, with similar conclusions to those presented in the paper for the energy efficiency. As part of future work, we plan to investigate the energy savings that can be attained with a limited loss in performance as well as prediction of the optimal level of concurrency throttling from the point of view of energy efficiency.

Acknowledgements. This work was supported by project Exa2Green (under grant agreement n°318793) 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. The researchers from Universidad Jaume I were supported by project TIN2014-53495-R of the MINECO and FEDER, and the FPU program of MECED.

References

1. The University of Florida Sparse Matrix Collection, January 2016. <http://www.cise.ufl.edu/research/sparse/matrices/>

2. Aliaga, J.I., Anzt, H., Castillo, M., Fernández, J., León, G., Pérez, J., Quintana-Ortí, E.S.: Unveiling the performance-energy trade-off in iterative linear system solvers for multithreaded processors. *Concurr. Comput. Pract. Exper.* **27**(4), 895–904 (2015)
3. Asanovic, K., et al.: The landscape of parallel computing research: a view from berkeley. Technical report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, December 2006
4. Bergner, P., et al.: Performance Optimization and Tuning Techniques for IBM Power Systems Processors Including IBM POWER8. IBM (2015). IBM Red Books
5. Buono, D., et al.: Optimizing sparse linear algebra for large-scale graph analytics. *Computer* **48**(8), 26–34 (2015)
6. Byun, J.-H., Lin, R., Yelick, K.A., Demmel, J.: Autotuning sparse matrix-vector multiplication for multicore. Technical report UCB/EECS-2012-215, EECS Dept., Univ. California, Berkeley (2012)
7. Choi, J.W., Singh, A., Vuduc, R.W.: Model-driven autotuning of sparse matrix-vector multiply on GPUs. In *Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPopp 2010*, pp. 115–126 (2010)
8. Dongarra, J., Heroux, M.A.: Toward a new metric for ranking high performance computing systems. Sandia report SAND2013-4744, Sandia National Laboratories, June 2013
9. Duranton, M., De Bosschere, K., Cohen, A., Maebe, J., Munk, H.: HiPEAC vision 2015 (2015). <https://www.hipeac.org/publications/vision/>
10. Guo, P., Wang, L., Chen, P.: A performance modeling and optimization analysis tool for sparse matrix-vector multiplication on GPUs. *IEEE Trans. Parallel Distrib. Syst.* **25**(5), 1112–1123 (2013)
11. Hager, G., Treibig, J., Habich, J., Wellein, G.: Exploring performance and power properties of modern multi-core chips via simple machine models. *Concurr. Comput. Pract. Exper.* **28**(2), 189–210 (2016)
12. Hennessy, J.L., Patterson, D.A.: *Computer Architecture: A Quantitative Approach*, 5th edn. Morgan Kaufmann, Waltham (2012)
13. Karakasis, V., Goumas, G., Koziris, N.: Exploring the performance-energy tradeoffs in sparse matrix-vector multiplication. In: *Workshop on Emerging Supercomputing Technologies (WEST) - ICS 2011* (2011)
14. Kepner, J., Gilbert, J. (eds.): *Graph Algorithms in the Language of Linear Algebra*. SIAM, Philadelphia (2011)
15. Lefurgy, C., Wang, X., Ware, M.: Server-level power control. In: *Proceedings of the 4th IEEE Conference on Autonomic Computing (ICAC 2007)*, Jacksonville, Florida, USA, 11–15 June, 2007
16. Liu, X., Smelyanskiy, M., Chow, E., Dubey, P.: Efficient sparse matrix-vector multiplication on x86-based many-core processors. In: *Proceedings of the 27th International Conference on Supercomputing*, Eugene, Oregon, USA, pp. 273–282, June 2013
17. Lucas, R.: Top ten Exascale research challenges (2014). <http://science.energy.gov/~media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf>
18. Malkowski, K.: *Co-adapting Scientific Applications and Architectures Toward Energy-efficient High Performance Computing*. Ph.D. thesis, University Park, PA, USA (2008) AI3346339

19. Malossi, A.C.I., Ineichen, Y., Bekas, C., Curioni, A., Quintana-Ortí, E.S.: Performance and energy-aware characterization of the sparse matrix-vector multiplication on multithreaded architectures. In Proceedings of 43rd International Conference on Parallel Processing (ICCP), Minneapolis (MN), USA, pp. 139–148 (2014)
20. Malossi, A.C.I., Ineichen, Y., Bekas, C., Curioni, A., Quintana-Ortí, E.S.: Systematic derivation of time and power models for linear algebra kernels on multicore architectures. *Sustainable Comput. Inf. Syst.* **7**, 24–40 (2016)
21. Saad, Y.: *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia (2003)
22. Vuduc, R.: *Automatic performance tuning of sparse matrix kernels*. Ph.D. dissertation, Univ. California, Berkeley, January 2004
23. Williams, S., et al.: Optimization of sparse matrix-vector multiplication on emerging multicore platforms. *Parallel Comput.* **35**(3), 178–194 (2009)