Memory Optimizations For Fast Power-Aware Sparse Computations

Konrad Malkowski, Padma Raghavan and Mary Jane Irwin
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA, 16802. Tel: 814-865-9505
E-mail:{malkowsk, raghavan, mji}@cse.psu.edu

***Abstract*— We consider memory subsystem optimizations for improving the performance of sparse scientific computation while reducing the power consumed by the CPU and memory. We first consider a sparse matrix vector multiplication kernel that is at the core of most sparse scientific codes, to evaluate the impact of prefetchers and power-saving modes of the CPU and caches. We show that performance can be improved at significantly lower power levels, leading to over a factor of five improvement in the operations/Joule metric of energy efficiency. We then indicate that these results extend to more complex codes such as a multigrid solver. We also determine a functional representation of the impacts of such optimizations and we indicate how it can be used toward further tuning. Our results thus indicate the potential for cross-layer tuning for multiobjective optimizations by considering both features of the application and the architecture.**

I. Introduction

Sparse scientific algorithms and codes enable the linear scaling of the computational costs of modeling and simulation applications when the problem size is increased through refinements required to capture phenomena of interest [7], [12]. However, the performance of such codes depends to a large extent on the memory subsystem design of the computer. Unlike dense codes [13], which inherently have a large number of floating point operations per data access, sparse codes are typically dominated by data access operations [8].

In this paper, we consider in detail the interactions between sparse code features, as represented by the sparse-matrix vector multiplication kernel (SMV), and memory optimizations. We discuss how memory optimizations that we have developed earlier [11], [15], [16] can affect the performance of tuned and un-tuned versions of sparse matrix vector multiplication. We consider the use of such optimizations with power-saving modes of the hardware such as Dynamic Voltage and Frequency Scaling (DVFS) [5] to improve performance at significantly lower power levels. We next develop a functional representation of metrics, such as performance and power, for parameters of the application and the hardware. We then indicate how this functional form could be used to select optimal feature sets for multiple objectives. This is particularly important because the impacts of multiple optimizations on multiple metrics are not independent of each other. Such analysis captures interactions between all parameters, including those representing code features and hardware optimizations, to enable the determination of minimal feature sets to maximize impact.

Section II discusses our methodology, Section III contains our main results and we end with brief concluding remarks in Section IV.

The work was supported in part by the National Science Foundation through the grant CCF-0444345.

II. Methodology

We use instruction-level simulation with SimpleScalar [3] and Wattch [2] to model our memory subsystem optimizations.

We use a standard sparse matrix vector kernel (SMV-U) and its tuned form (SMV-O) from Sparsity [8], and the multigrid code MG from the NAS benchmark [1]. Both SMV kernels compute $y \leftarrow A \times x$ which requires one floating-point multiplication and addition per nonzero element in A; x, y are $N-$vectors and A is a sparse $N \times N$ matrix. In both cases, re-use of elements of x can be enhanced by reordering A, as shown in Figure 1. We use such reordered forms for our test matrices with both SMV-U and SMV-O; SMV-O includes optimizations that increase floating-point operations while decreasing loads from memory.

The sparse kernels are emulated by SimpleScalar3.0 [3] and Wattch1.02d [2] with extensions to model memory subsystem enhancements. We use SimpleScalar configured to accept PISA compiled programs to model a single-core processor (such as the one in BlueGene [18]), starting from a PowerPC440 embedded core. We use Wattch [2] to calculate the power consumption with extrapolations for .13 um technology [11], [15], [16]. We also developed a DDR2 type memory performance and power simulator for use with our modified

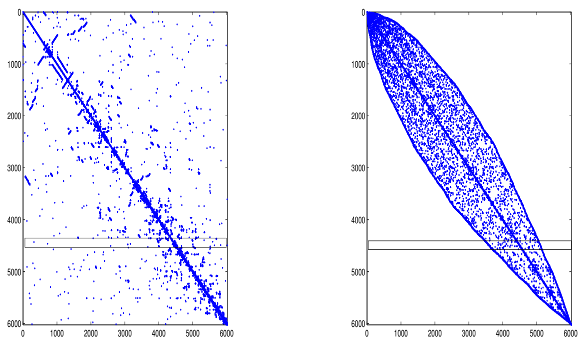

Fig. 1. The location of nonzeroes in a sparse matrix in the original order (left) and in the reordered matrix (right). Nonzeroes in a horizontal box indicate the access pattern in the source vector for processing a row of the matrix. Observe the nearly random access pattern in the original form (left) and the more localized pattern in the permuted form (right).

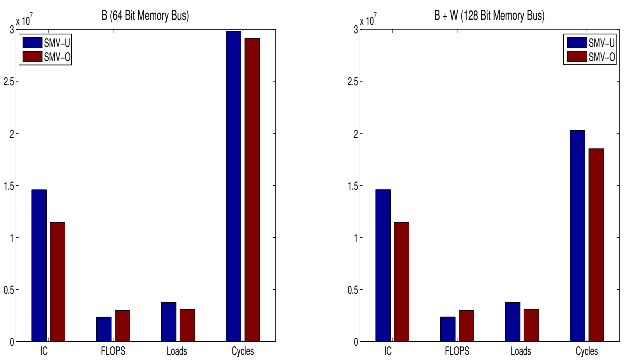

Fig. 2. Impact of code tuning of sparse matrix vector multiplication for the base B at 1 GHz with a 4MB L3-cache, with a 64-bit memory bus (left) and the wider bus (W) (right). Observe that SMV-O trades-off fewer loads, and total number of instructions (IC) for an increase in floating point operations and a 10% decrease in total cycles over SMV-U.

versions of SimpleScalar and Wattch.

Our base architecture has two floating-point units (FPUs) and two integer arithmetic-logic units (IALUs). Each FPU has a multiplication/division module and other arithmetic-logic modules. Thus, our base system can issue four floating-point instructions at each cycle. The data paths between memory and L3 cache are 64 bit wide with cache lines of 64 bytes, i.e., 8 double precision operands or 16 integer operands. We model a cache hierarchy with three levels on chip, including a 32KB data/32KB instruction level 1 cache (L1), a 2KB level 2 cache (L2), and a 4MB unified level 3 cache (L3). Wattch is configured to model only two levels of cache, but we added new functions to model our hierarchy. More details of our system can be found in [11], [15].

Starting with the base architecture (B) we consider the effects of (i) doubling the width of the data paths (W), (ii) Memory page policy: open (MO) or closed (default), (ii) memory prefetching at the memory controller (MP), and (iii) L2-cache prefetching (LP). Many of these optimizations have been considered in other contexts [9], [10], [14], [17], [19]–[21]. All prefetchers are stride-1 and we simulate utilizing power control modes of caches by simply varying cache sizes.

III. Empirical Results and Analysis

We now evaluate the impact on performance (time), power, and energy, of memory subsystem optimizations (W,MO,MP,LP) for SMV-U, SMV-O and MG as discussed in Section II. We first indicate how we can significantly improve the energy efficiency in terms of the number of floating-point-operations/Joule by improving performance at reduced power levels. Next, we consider how we can use our observed data to derive a functional representations which can be used for constrained multi-objective optimizations.

Figure 2 indicates the benefits of code tuning when combined with a wider bus (B) for increased memory bandwidth. Observe that SMV-O with a 2 by 1 blocking increases the floating point operations (useful work) by operating on known zeroes inserted into the matrix in order to reduce the number of loads. However, both SMV-U and SMV-O benefit from the increased memory bandwidth.

Figure 3 indicates that L3 cache miss rates remain nearly unchanged as L3 size is decreased from 4MB to 256KB for a given set of optimizations at two different CPU frequencies (1GHz and 600MHz). Among the optimizations, the wider bus (W) and the L2-cache prefetcher (LP) result in the most dramatic decreases in L3-miss rates.

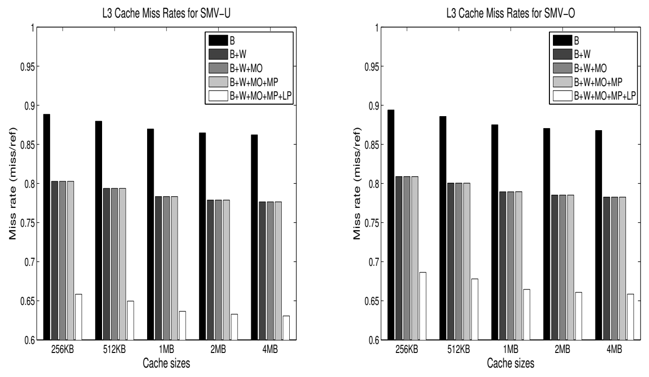

Fig. 3. SMV-U and SMV-O: Impact of optimizations on L3 cache miss rates for 4MB, 2MB, 1MB, 512KB and 256KB cache sizes.

In Figure 4, we illustrate the impact on average load store queue latencies, i.e., memory clock cycles per instruction (memory CPI), as the optimizations are added in the sequence W, MO, MP and LP to the base with a 256KB L3 at either

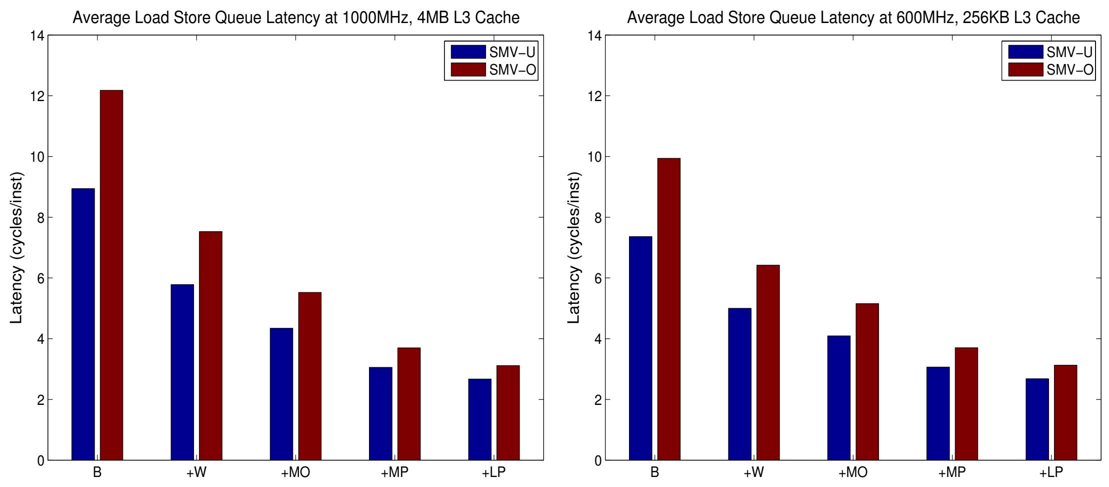

Fig. 4. SMV-U and SMV-O: Impact of optimizations on average load-store queue latency (effective memory CPI) at 1GHz 4MB L3 cache (left) and at 600MHz 256KB L3 cache (right) configurations, as optimizations are added in the order (W, MO, MP, LP). Thus, all optimizations are included in the configuration labeled +LP.

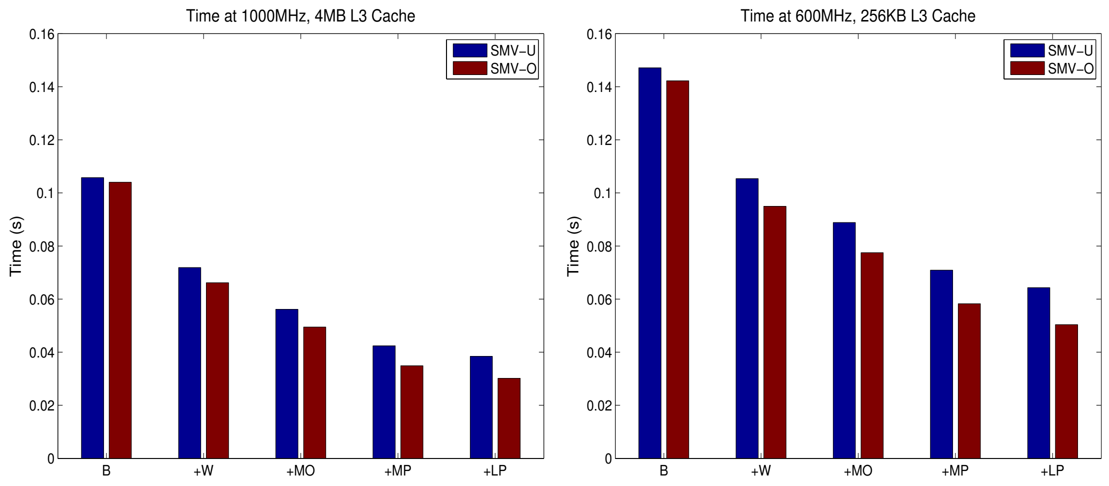

Fig. 5. SMV-U and SMV-O: Impact of optimizations on time (seconds) at 1GHz 4MB L3 cache (left) and at 600MHz 256KB L3 cache (right) configurations, as optimizations are added in the order shown with all included at label +LP.

1GHz or 600MHz. We thus show the combined impacts of memory subsystem optimizations with power saving modes of the caches and DVFS. Observe that the memory CPI is lower for the base B at 600MHz indicating a better balance between CPU and memory service times. Memory CPIs decrease dramatically when the optimizations are added, with greater benefits for the faster CPU at 1GHz. As indicated in Figure 5, these reductions in memory CPI translate to faster execution (time, in seconds). Furthermore, with even just a few of the optimizations, execution is faster at 600MHz compared to the base at 1GHz.

We indicate the impact of optimizations (at 1GHz with 4MB L3, and at 600MHz with a 256KB cache) on: power in Figure 6, energy in Figure 7, and energy efficiency, i.e., floating point operations/J, in Figure 8. Observe that most optimizations result only in small increases in power with MO reducing power (as expected, when the data layout is selected to reduce bank conflicts). The power reductions from DVFS and a smaller cache are particulary impressive, translating to reduced energy levels and over factors of 5 improvements in energy efficiency. For SMV-U, the energy efficiency scales from $.75 \times 10^7$ at B, 1GHz, 4MG L3 to 4.2×10^7 with all optimizations at 600MHz, 256KB L3. Likewise, the energy efficiency of SMV-O scales from $.9 \times 10^7$ at B, 1GHz, 4MG L3 to 6.7×10^7 with all optimizations at 600MHz, 256KB L3.

We consider in summary (see Figure 9), the impact on performance and energy delay product (EDP, energy $\times$ time) for SMV-U, SMV-O (L3 cache size 256KB), and MG (with L3 cache size of 512KB, the smallest size without performance degradations) across the frequency range from 300 MHz to 1GHz. The values (shown relative to the B at 1GHz, 4MB L3 at 1) indicate faster execution than at the base starting at

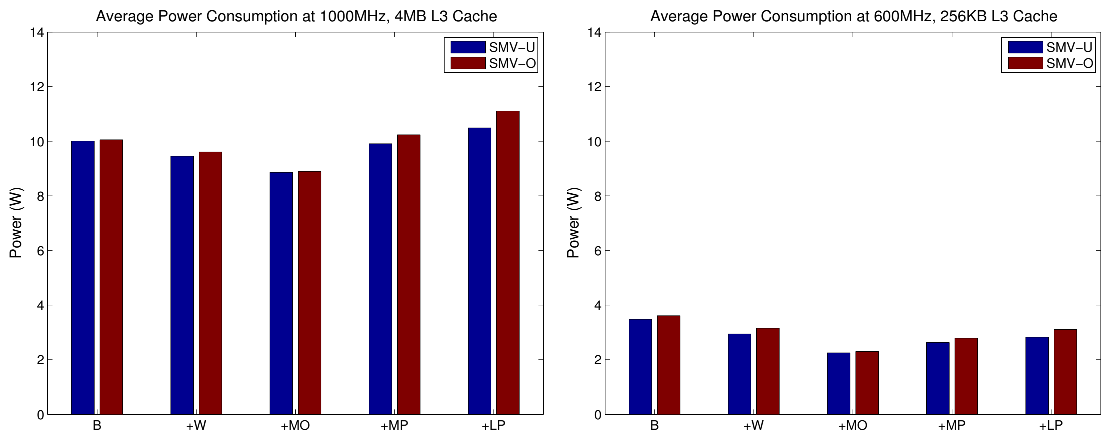

Fig. 6. SMV-U and SMV-O: Impact on power (in Watts) at 1GHz 4MB L3 cache (left) and at 600MHz 256KB L3 cache (right) configurations, as optimizations are added in the order shown with all included at label +LP.

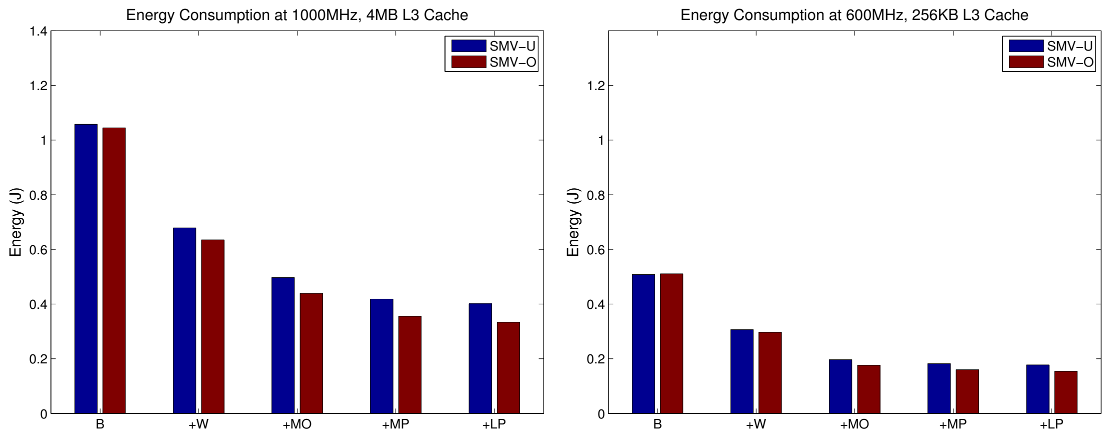

Fig. 7. SMV-U and SMV-O: Impact of optimizations on energy (in Joule), at 1GHz 4MB L3 cache (left) and at 600MHz 256KB L3 cache (right) configurations.

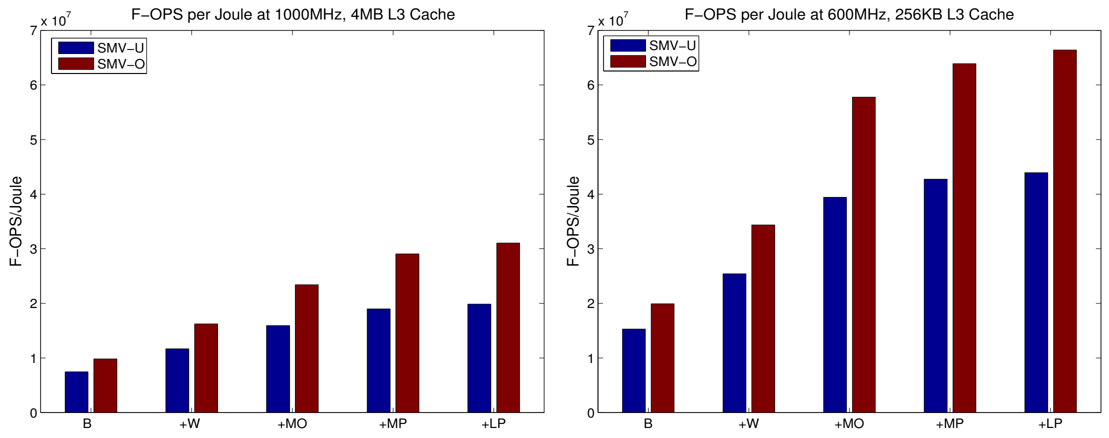

Fig. 8. SMV-U and SMV-O: Impact of optimizations on energy efficiency, i.e., floating point operations/Joule, at 1GHz 4MB L3 cache (left) and at 600MHz 256KB L3 cache (right) configurations.

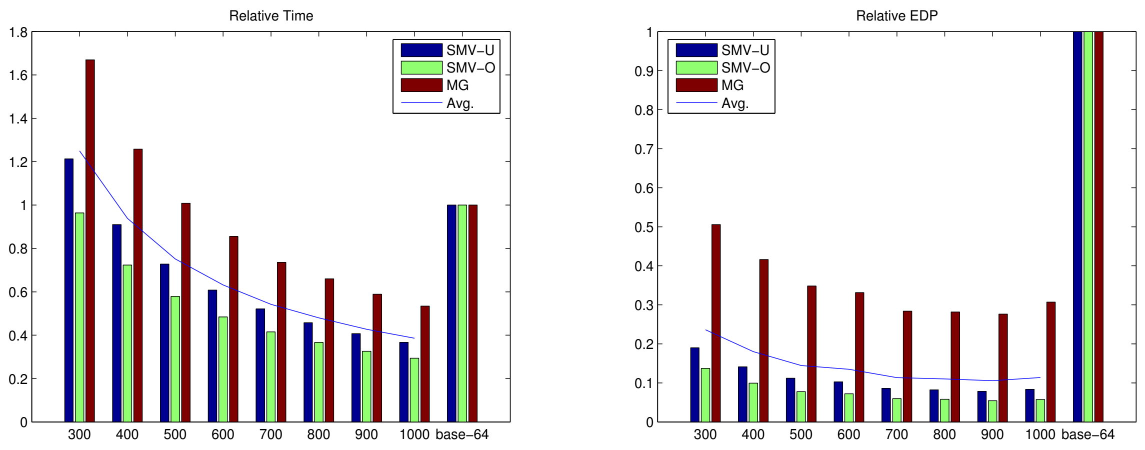

Fig. 9. SMV-U, SMV-O and MG: Time and EDP (energy x time) at each CPU frequencies from 300MHz to 1GHz, with all optimizations (W, MO, MP, LP) and a 256KB L3 cache (512KB L3 cache for MG). Values are shown relative to 1 at B.

600MHz, with the minimal EDP observed at 700MHz.

Functional Representations and Optimization. The data represented in the earlier figures, provides a sampling of the parameter space of codes and architecture and their relationship to metrics such as time, power and energy. We can use data-fitting techniques [6] to derive a functional representation to model the metrics in terms of their parameters. We used a least-squares scheme to derive (as simple approximations), functions of the form $P = c_0 {V_{dd}}^2 f \times \sum_{i=i}^{k} c_i \pi_i$ and $T = b_0 f \times \sum_{i=i}^{q} b_i \pi_i$, where f is the CPU frequency and V_{dd} its supply voltage, π_i represent parameters, and c_i and b_i represent constants. Such data-fitting can help determine the relative impact of each optimization and their combinations, independent of the order in which they are considered.

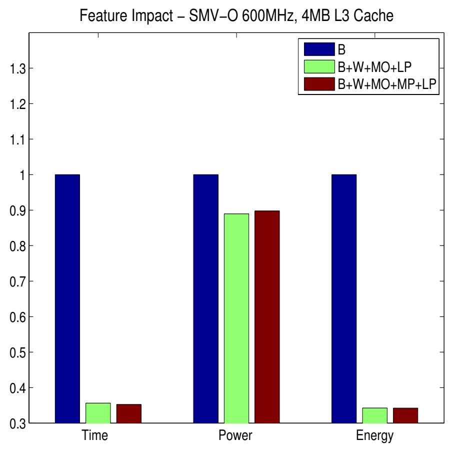

Fig. 10. SMV-O: Relative time, power and energy at 600MHz, 4MB L3 for (i) B (at 1), at (ii) optimal energy 3 feature set B+W+MO+MP, and (iii) with all optimizations (B+W+MO+MP+LP).

The functional representations can be used with a mixed

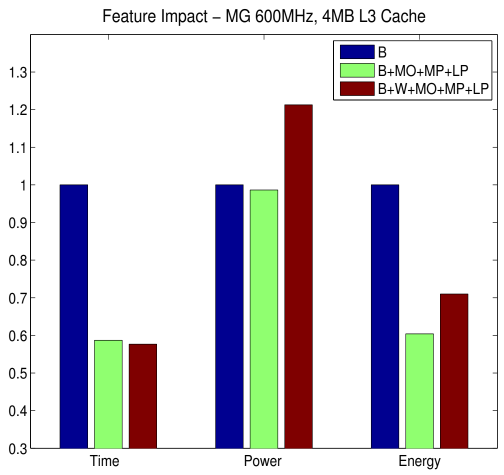

Fig. 11. MG: Relative time, power and energy at 600MHz, 4MB L3 for (i) B (at 1), at (ii) optimal energy 3 feature set B+MO+MP+LP, and (iii) with all optimizations (B+W+MO+MP+LP).

integer program (for optimization), to select, for example, a set of exactly 3 optimizations that minimize energy at execution times no slower than at the base B (at 600 MHz, 4MB L3). Our analysis indicated that such an optimal configuration is given by B+W+MO+MP for SMV-O and B+MO+MP+LP for MG. Figures 10 and 11 show relative time, power and energy for these configurations, the base B, and the configuration with all optimizations. Observe that these optimal configurations perform just as well as the configuration with all features, at equal or lower power levels. Such analysis indicates the potential of numeric techniques for multiobjective optimizations.

IV. Conclusions

The results in this paper indicate the significance of memory subsystem optimizations for power-aware high performance of sparse codes. We conjecture that such codes will be impose even greater demands on the memory subsystem of emerging chip multiprocessor (CMP) architectures, especially as they scale to larger numbers of CPUs. We plan to extend our work to evaluate performance and power trade-offs of sparse computations on such CMPs, with particular attention on developing accurate functional representations for efficient exploration of the high dimensional space of multiobjective optimizations. Such functional representations will necessarily be more complex than the ones indicated here. Additionally, they need to be incorporated into a numerical optimization framework to model the effect of uncertainties in the parameters and observed metrics [4].

REFERENCES

[1] D. H. Bailey, L. Dagum, E. Barszcz, and H. D. Simon. NAS parallel benchmark results. In *Supercomputing '92: Proceedings of the 1992 ACM/IEEE conference on Supercomputing*, pages 386–393, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[2] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *ISCA '00: Proceedings of the 27th annual international symposium on Computer architecture*, pages 83–94. ACM Press, 2000.

[3] Doug Burger and Todd M. Austin. The SimpleScalar tool set, version 2.0. *SIGARCH Comput. Archit. News*, 25(3):13–25, 1997.

[4] D. G. Cacuci. *Sensitivity and Uncertainty Analysis*, volume 1. Chapman and Hall/CRC, 2003.

[5] Kihwan Choi, Ramakrishna Soma, and Massoud Pedram. Fine-grained dynamic voltage and frequency scaling for precise energy and performance trade-off based on the ratio of off-chip access to on-chip computation times. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 10004. IEEE Computer Society, 2004.

[6] M. T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill, 2nd edition, 2002.

[7] M. Heroux, P. Raghavan, and H. D. Simon, editors. *Parallel Processing for Scientific Computing*. SIAM Publications, 2006. Book website, `http://www.ec-securehost.com/SIAM/SE20.html`.

[8] E.-J. Im and K. A. Yelick. SPARSITY. `http://www.cs.berkely/yelick/sparsity`.

[9] Teresa L. Johnson and Wen mei W. Hwu. Run-time adaptive cache hierarchy management via reference analysis. In *ISCA '97: Proceedings of the 24th annual international symposium on Computer architecture*, pages 315–326. ACM Press, 1997.

[10] Norman P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *ISCA '90: Proceedings of the 17th annual international symposium on Computer Architecture*, pages 364–373. ACM Press, 1990.

[11] K. Malkowski and I. Lee and P. Raghavan and M.J. Irwin. On improving performance and energy profiles of sparse scientific applications. In *Proceedings of the 20th IEEE International Parallel and Distributed Symposium, IPDPS'06, Next Generation Software Workshop, accepted, to appear*, April 2006.

[12] D. Keyes. Terascale Optimal PDE Simulations (TOPS) Center. `http://tops-scidac.org/`, 2004.

[13] C. Lazou. LINPACK results refuel IBM/INTEL chip debate. *HPCWire*, 12(29), July 2003.

[14] Wei-Fen Lin, Steven K. Reinhardt, and Doug Burger. Designing a modern memory hierarchy with hardware prefetching. *IEEE Trans. Comput.*, 50(11):1202–1218, 2001.

[15] K. Malkowski, I. Lee, P. Raghavan, and M.J. Irwin. Conjugate gradient sparse solvers: Performance-power characteristics. In *Proceedings of the 20th IEEE International Parallel and Distributed Symposium, IPDPS'06, Second High-Performance, Power-Aware Computing Workshop (to appear)*, April 2006.

[16] K. Malkowski, G. Link, P. Raghavan, and M.J. Irwin. 'load miss prediction - exploiting power performance trade-offs. In *Proceedings of the 21st IEEE International Parallel and Distributed Symposium, IPDPS'07, Second High-Performance, Power-Aware Computing Workshop (to appear)*, 2007.

[17] S. A. McKee, R. H. Klenke, K. L. Wright, W. A. Wulf, M. H. Salinas, J. H. Aylor, and A. P. Batson. Smarter memory: Improving bandwith for streamed references. *IEEE Computer*, pages 54–63, February 1998.

[18] The Bluegene Team. An overview of the BlueGene/l Supercomputer. In *SC '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–22. IEEE Computer Society Press, 2002.

[19] G. Tyson, M. Farrens, J. Matthews, and A. R. Pleszkun. A modified approach to data cache management. In *MICRO 28: Proceedings of the 28th annual international symposium on Microarchitecture*, pages 93–103. IEEE Computer Society Press, 1995.

[20] Y. Wu, R. Rakvic, Li-Ling Chen, Chyi-Chang Miao, G. Chrysos, and J. Fang. Compiler managed micro-cache bypassing for high performance EPIC processors. In *MICRO 35: Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, pages 134–145. IEEE Computer Society Press, 2002.

[21] Chengqiang Zhang and Sally A. McKee. Hardware-only stream prefetching and dynamic access ordering. In *ICS '00: Proceedings of the 14th international conference on Supercomputing*, pages 167–175. ACM Press, 2000.