Characterizing the Performance and Energy Attributes of Scientific Simulations*

Sayaka Akioka[1], Konrad Malkowski[1], Padma Raghavan[1], Mary Jane Irwin[1], Lois Curfman McInnes[2], and Boyana Norris[2]

[1] The Pennsylvania State University
{tobita, malkowsk, raghavan, mji}@cse.psu.edu
[2] Argonne National Laboratory
{mcinnes, norris}@mcs.anl.gov

Abstract. We characterize the performance and energy attributes of scientific applications based on nonlinear partial differential equations (PDEs). where the dominant cost is that of sparse linear system solution. We obtain performance and energy metrics using cycle-accurate emulations on a processor and memory system derived from the PowerPC RISC architecture with extensions to resemble the processor in the BlueGene/L. These results indicate that low-power modes of CPUs such as Dynamic Voltage Scaling (DVS) can indeed result in energy savings at the expense of performance degradation. We then consider the impact of certain memory subsystem optimizations to demonstrate that these optimizations in conjunction with DVS can provide faster execution time and lower energy consumption. For example, on the optimized architecture, if DVS is used to scale down the processor to 600MHz, execution times are faster by 45% with energy reductions of 75% compared to the original architecture at 1GHz. The insights gained from this study can help scientific applications better utilize the low-power modes of processors as well as guide the selection of hardware optimizations in future power-aware, high-performance computers.

1 Introduction

As microprocessors' clock frequencies have increased in recent years, their corresponding power consumption has also increased dramatically. The peak power dissipation and consequent thermal output often limit processor performance and hinder system reliability. Thus, due to thermal and cost concerns, architectural approaches that reduce overall energy consumption have become an important issue in high-performance parallel computing. The IBM BlueGene/L [17] is one example of power-aware supercomputer architectures, where lower-power, lower-performance processors are integrated into a massively parallel architecture, resulting in performance sufficient to claim the top spot in the Top500 list of supercomputer sites [27]. To fully exploit the performance potential of existing and future low-power architectures in scientific computations, the community must investigate the effects of hardware features on both performance and power.

* This work is supported by the National Science Foundation through grants ACI-0102537, CCF-0444345 and DMR-0205232.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part I, LNCS 3991, pp. 242–249, 2006.

Such work will have a two-fold impact: first, we will be able to target performance optimizations effectively; second, understanding the effects of new hardware features on realistic applications can help in the design of the next generation of power-aware architectures.

In this paper we profile a simulation of driven cavity flow [10] that uses fully implicit Newton-Krylov methods to solve the resulting system of nonlinear PDEs. We have selected this model problem because it has properties that are representative of many large-scale, nonlinear PDE-based applications in domains such as computational aerodynamics [1], astrophysics [16], and fusion [26]. The most time-consuming portion of the simulation is the solution of large, sparse linear systems of equations. Thus, we profile this kernel and interpret the impact of different memory hardware features and algorithm choices on the performance and power characteristics of the model problem.

The remainder of this paper is organized as follows. Section 2 summarizes related power-performance studies. Section 3 provides an overview of the driven cavity application, while Section 4 describes our approach to profiling its performance and power consumption. In Section 5 we present the results of these simulations. We conclude in Section 6 with observations of profiling trends and comments on future work.

2 Related Work

There are a number of studies on power-aware software optimization. The principal approach targets the use of dynamic voltage scaling (DVS) [25] to save central processing unit (CPU) energy. The basic idea of DVS is to reduce runtime energy as much as possible without a significant reduction in application performance; this technique allows the reduction of CPU idle time and consequently enables machines to spend more work time in a lower-power mode. Several studies investigate scheduling for DVS [9, 29, 15, 28, 18] or DVS-aware algorithms [19, 20]. However, we are not aware of any profiling and optimization approach that focuses explicitly on high-performance scientific computing.

There are few existing efforts to build power consumption models of applications [6, 11]. These approaches utilize performance counters on CPUs and actual power samples to build a model, thereby making this approach viable only when such counters and power measurements are available – usually not the case for newly released architectures. Rose et al. [13] are incorporating some temperature-based power estimates in the SvPablo tool, in addition to traditional hardware counter performance metrics. Feng et al. [14] are building a profiling tool that characterizes the power consumption and performance of scientific benchmarks, including SPEC CPU2000 [12] and the NAS parallel benchmarks [2]. Further investigation is needed to characterize and understand the behavior of real scientific applications.

3 A Model PDE-Based Application: Driven Cavity Flow

To explore power issues arising in the solution of large-scale nonlinear PDEs, we focus on the model problem of two-dimensional flow in a driven cavity, which has properties

that are representative of our broader class of target applications. This problem is realistic yet sufficiently simple to provide a good a starting point for understanding the performance and energy attributes of PDE-based simulations.

The model is a combination of lid-driven flow and buoyancy-driven flow in a two-dimensional rectangular cavity. The lid moves with a steady and spatially uniform velocity and sets a principal vortex by viscous forces. The differentially heated lateral walls of the cavity invoke a buoyant vortex flow, opposing the principal lid-driven vortex. The nonlinear system can be expressed in the form $f(u) = 0$, where $f : R^n \rightarrow R^n$. We discretize this system using finite differences with the usual five-point stencil on a uniform Cartesian mesh, resulting in four unknowns per mesh point (two-dimensional velocity, vorticity, and temperature). Further details are discussed in [10]. Experiments presented in Section 5 employ a 64×64 mesh and the following nonlinearity parameters: lid velocity 10, Grashof number 600, and Prandtl number 1.

We solve the nonlinear system using an inexact Newton method (see, e.g., [23]) provided by the PETSc library [3], which (approximately) solves the Newton correction equation

$$f'(u^{k-1})\, \delta u^k = -f(u^{k-1}) \tag{1}$$

and then updates the iterate via $u^k = u^{k-1} + \alpha \cdot \delta u^k$, where α is a scalar determined by a line search technique such that $0 < \alpha \leq 1$. The overall time to solution is typically dominated by the time for repeatedly solving the sparse linearized systems (1) using preconditioned Krylov methods; such sparse solution comprises over 90% of the application execution time [5]. Consequently, the experiments in Section 5 focus on this important computational kernel. The dominance of sparse linear solves in overall time to solution is typical of many applications solving nonlinear PDEs using implicit and semi-implicit schemes.

4 Methodology

Our goal is to characterize the performance and power attributes of nonlinear PDE-based applications when low power modes, such as DVS, are used. An additional goal is to characterize the impact of memory subsystem optimization on performance and power. We use architectural emulation with cycle-accurate simulation of our application code to derive performance and power characteristics. For this purpose, we use SimpleScalar [8] and Wattch [7], which are two well-accepted tools in the computer architecture community. SimpleScalar is a cycle-accurate emulator of C code on a RISC architecture, which can be specified in full detail. Wattch is an extension of SimpleScalar and provides power consumption and energy metrics.

We start with a base architecture, denoted as **B**, to represent a processor and memory subsystem similar to that of the IBM BlueGene/L [17], designed specifically for power-aware scientific computing. **B** has two floating-point units (FPUs) and two integer arithmetic logic units (ALUs). The cache hierarchy consists of 32KB data / 32KB instruction Level 1 cache, 2KB Level 2 cache (L2), and 4MB unified Level 3 cache (L3). The L3 cache uses Enhanced Dynamic Random Access Memory (eDRAM) technology, similar to the BlueGene/L [17]. The main memory has 256MB of Double Data Rate SDRAM 2 (DDR2). The system is configured for a CPU frequency of 1000MHz

with corresponding nominal V_{dd} voltages to model the effect of DVS. The frequency - V_{dd} pairs we use are: 400MHz-0.66V; 600MHz-0.84V; 800MHz-0.93V; 1000MHz-1.20V. We can then simulate the effects of DVS by scaling the frequency to 800MHz, 600MHz, and 400MHz with corresponding scaling of V_{dd}.

We employ a set of memory subsystem optimizations considered first in the evaluation of power and performance trade-offs of simple sparse kernels [21]:

- **W:** A wider memory bus of 128 bytes, as opposed to the 64-byte memory bus in the base architecture **B**.
- **MO:** Open memory page policy. This policy determines whether memory banks stay open or closed after a read/write operation. An open policy is more appropriate when there is some locality of data accesses.
- **LP:** Level 2 cache prefetching. After a read operation, the next line is prefetched. This optimization can improve performance if the data access pattern has locality.
- **MP:** Memory prefetching. This option is similar to **LP** above, but now the prefetching is done by the memory controller to avoid the latency of accessing the memory.
- **LM:** Load miss prediction logic. This feature can avoid L2 and L3 latencies when data is not present in the cache hierarchy.

We use labels of the form **B+W+MO** to specify a configuration that augments the basic architecture (**B**) with the wider data path (**W**) and memory-open (**MO**) features. Such memory subsystem optimizations are feasible and have been studied earlier in the architecture community. We evaluate the impact of these architectural configurations on the performance of sparse solvers in conjunction with DVS. More specifically, we focus on the seven configurations specified by: **B**, **B+W**, **B+W+MO**, **B+W+LP**, **B+W+MP**, **B+W+LM**, and **B+W+MO+LP+MP+LM** (or **All**).

5 Empirical Results

We simulate the driven cavity flow code introduced in Section 3 on SimpleScalar and Wattch to understand the performance and power attributes of its dominant computation, namely solving the sparse linear system given by Equation (1). We focus on observed performance and power metrics during the first iteration of Newton's method for different choices of the linear solver. We consider four popular schemes which are provided within PETSc [3]: the Krylov methods GMRES(30) and BiCG in combination with the incomplete factorization preconditioners ILU(0) and ILU(1) (see, e.g., [4,24]).

We begin by characterizing performance and power for **B**, at 1000MHz and when DVS is used to scale down to 800MHz, 600MHz, and 400MHz. Figure 1 shows the execution time in seconds (left), and the average system power per cycle in Watts (center) for the four sparse linear solution schemes, namely GMRES(30) + ILU(0), GMRES(30) + ILU(1), BiCG + ILU(0), and BiCG + ILU(1). The y axis represents values of the execution time or power at frequencies of 400MHz, 600MHz, 800MHz, 1000MHz for **B**. As expected, with frequency scaling down to lower values, the execution time for all methods increases. However, the relative increase for the same change in the frequency is different for different solvers. For example, BiCG+ILU(0) shows a greater increase in execution time than BiCG+ILU(1) when the frequency is scaled down from 600MHz

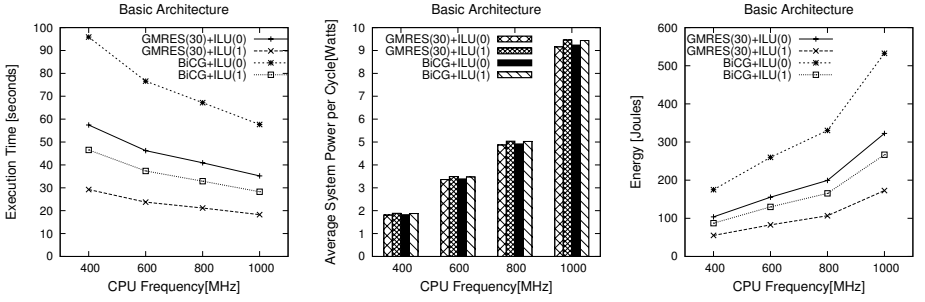

Fig. 1. Execution time in seconds (left), average system power per cycle in Watts (center), and energy in Joules (right) for one linear system solution with the base architecture at frequencies of 400MHz, 600MHz, 800MHz, and 1000MHz

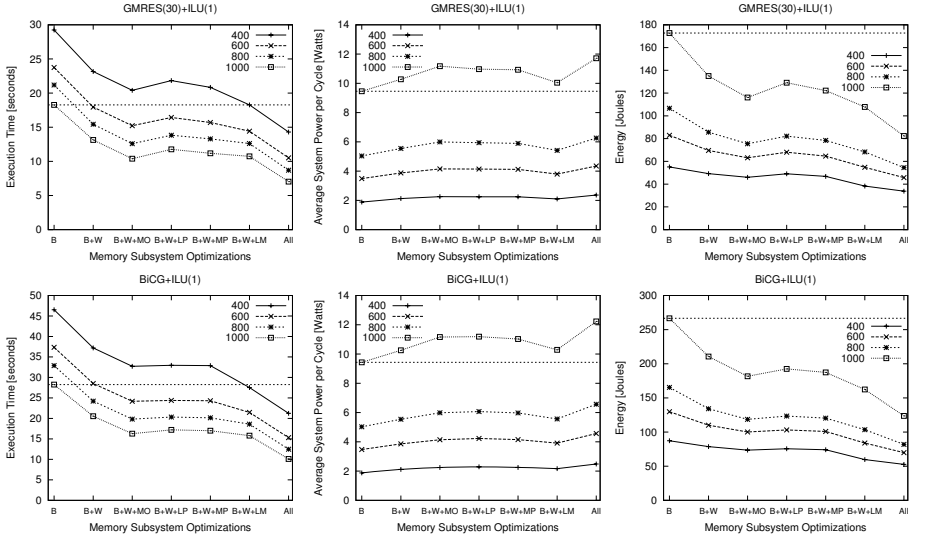

Fig. 2. Execution time in seconds (left), average system power per cycle in Watts (center), and total energy in Joules (right) for one linear system solution using GMRES(30)+ILU(1) (top) and BiCG+ILU(1) (bottom) at frequencies of 400MHz, 600MHz, 800MHz, and 1000MHz when memory subsystem optimizations are added sequentially starting with the base architecture

to 400MHz. The average power values are not substantially different across the solvers. This situation is not surprising as the underlying operations in the solvers are similar and therefore utilize the components of the architecture similarly. The corresponding energy values are shown in Figure 1 (right). As expected, the solvers consume substantially less energy at lower frequencies. However, such energy reductions through DVS come at the expense of degradation in the performance of the solvers, as indicated by substantial increases in execution time at the lowest frequency.

We next explore how the memory subsystem optimizations described in Section 4 help to improve performance. In Figure 2 (left), we focus on GMRES(30) + ILU(1) and BiCG + ILU(1). Each plot shows the execution time of a specific solver as the frequency is kept fixed and the architecture is changed to reflect the seven configurations, **B**, **B+W**, **B+W+MO**, **B+W+LP**, **B+W+MP**, **B+W+LM**, and **All**. These configurations are shown along the x axis, while the y axis values indicate execution time in seconds.

Again, the execution times for all architectural configurations and solvers increase as frequency scales down to lower values. The trends are similar for both solvers. More significantly, the execution times for **B** at 1000MHz are higher than the execution times when even the simplest optimization is added at lower frequency values down to 600MHz. A dotted flat line indicates the time for the solver on **B** at 1000MHz. When all optimizations are included, i.e., the configuration **All**, even at 400MHz the performance is improved over that observed for **B** at 1000MHz.

Figure 2 (center) shows the average system power per cycle in Watts corresponding to the execution times shown earlier in Figure 2 (left). Adding architectural optimizations increases the power incrementally. However, these increases are less significant than the substantial decreases obtained from DVS. Thus, system power values for the base architecture at 1000MHz are higher than at 400MHz, 600MHz and 800MHz consistently.

Figure 2 (right) shows the energy consumed by GMRES(30)+ILU(1) and BiCG+ILU(1). Once again we use a dotted line to denote the value for the base architecture at 1000MHz. Observe that with memory subsystem optimizations, the energy is reduced even without DVS. This effect, which is observed at all frequencies, is primarily due to improvements in execution time. For example, when using all optimizations at 600MHz, substantial reductions in energy are realized for both solvers. From these results, we conclude that the proper selection of memory subsystem optimizations in conjunction with DVS can reduce both execution time and energy consumed.

To quantify the relative improvements in execution time and energy, we define the following relative reduction (RR) metric. Consider a specific solution scheme, i.e., a specific combination of a Krylov method and preconditioner. Let $T_{f,q}$ denote the observed execution time at frequency f MHz for an architectural configuration labeled q. We define RR with respect to the execution time for **B** at 1000MHz, i.e., $T_{1000,B}$ as:

$$RR(T)_{f,q} = (T_{1000,B} - T_{f,q})/T_{1000,B}. \tag{2}$$

Positive RR values indicate reductions in execution time, while negative values indicate increases. We can also define a corresponding RR metric for energy; we denote this for frequency f MHz and architectural configuration q as $RR(E)_{f,q}$.

Figure 3 shows $RR(T)$ and $RR(E)$ for each solver at each frequency for **All** relative to values for **B** at 1000MHz. RR values for the four solvers are indicated by a group of four bars at each frequency; the average value at each frequency is used in the line plot. Observe that on average, execution time is improved by approximately 20% at 400MHz with energy improvements of approximately 80%. Furthermore, if DVS is used to scale down to 600MHz, performance improvements are in excess of 45% on average with energy reductions of approximately 75%.

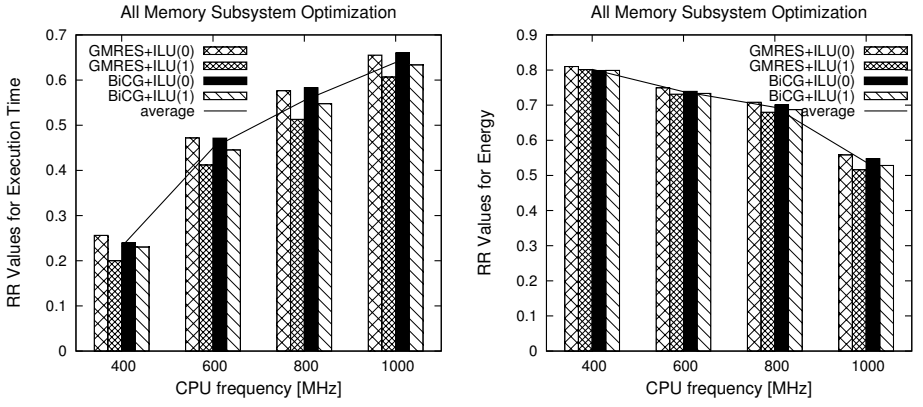

Fig. 3. Relative reductions (RR) in execution time (left) and energy (right) for one linear system solution with the architectural configuration **All** with respect to time/energy values for the basic architecture at 1000MHz. The line indicates the average reductions at each CPU frequency.

6 Conclusions

With the need for low-power, high-performance architectures, the combined performance and power characteristics of scientific applications on such architectures are an important area of study. In this paper, we observe the impact of memory subsystem optimizations and different linear solvers on the performance and power consumption of a nonlinear PDE-based simulation of flow in a driven cavity. Results using the SimpleScalar and Wattch simulators indicate that memory subsystem optimization plays a more important role than CPU frequency in reducing both execution time and power consumption. These results, taken together with our earlier studies [5,22] demonstrating the impact of different solution schemes on quality and performance, indicate that there is significant potential for developing adaptive techniques to co-manage performance, power and quality trade-offs for such scientific applications.

References

1. W. K. Anderson, W. D. Gropp, D. K. Kaushik, and et al. Achieving high sustained performance in an unstructured mesh CFD application. In *SC99*, 1999.
2. D. Bailey, T. Harris, W. Saphir, and et al. The NAS parallel benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center, 1995.
3. S. Balay, K. Buschelman, V. Eijkhout, and et al. PETSc users manual. Technical Report ANL-95/11 - Revision 2.3.0, Argonne National Laboratory, 2005. See `http://www.mcs.anl.gov/petsc`.
4. R. Barrett, M. Berry, T. F. Chan, and et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, Software, Environments, Tools*. SIAM, 1994.
5. S. Bhowmick, L. C. McInnes, B. Norris, and et al. The role of multi-method linear solvers in PDE-based simulations. In V. Kumar, M. L. Gavrilova, C. J. K. Tan, and P. L'Ecuyer, editors, *Lecture Notes in Computer Science, Computational Science and its Applications-ICCSA 2003*, volume 2677. Springer Verlag, 2003.

6. W. L. Bircher, M. Valluri, L. John, and et al. Runtime identification of microprocessor energy saving opportunities. In *ISLPED'05*, 2005.
7. D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *ISCA'00*, 2000.
8. D. C. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. Technical Report 1342, UW Madison Computer Sciences, 1997.
9. J. Casmira and D. Grunwald. Dynamic instruction scheduling slack. In *2000 KoolChips Workshop*, 2000.
10. T. S. Coffey, C. T. Kelley, and D. E. Keyes. Pseudo-transient continuation and differential-algebraic equations. *SIAM J. Sci. Comput.*, 25(2), 2003.
11. G. Contreras and M. Martonosi. Power prediction for Intel XScale processors using performance monitoring unit events. In *ISLPED'05*, 2005.
12. Standard Performance Evaluation Corporation. The SPEC benchmark suite. `http://www.spec.org`.
13. L. A. de Rose and D. A. Reed. SvPablo: A multi-language architecture-independent performance analysis system. In *ICPP'99*, 1999.
14. X. Feng, R. Ge, and K. W. Cameron. Power and energy profiling of scientific applications on distributed systems. In *IPDPS'05*, 2005.
15. B. Fields, R. Bodik, and M. M. Hill. Slack: Maximizing performance under technological constraints. In *ISCA'02*, 2002.
16. B. Fryxell, K. Olson, P. Ricker, and et al. FLASH: An adaptive-mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *Astrophys. J. Suppl.*, 2000.
17. A. Gara, M. A. Blumrich, D. Chen, and et al. Overview of the Blue Gene/L system architecture. *IBM J. Res. & Dev.*, 49(2/3), 2005.
18. R. Ge, X. Feng, and K. W. Cameron. Performance-constrained, distributed DVS scheduling for scientific applications on power-aware clusters. In *SC05*, 2005.
19. C. Hsu and W. Feng. A power-aware run-time system for high-performance computing. In *SC05*, 2005.
20. N. Kappiah, V. W. Freeh, and D. K. Lowenthal. Just-in-time dynamic voltage scaling: Exploiting inter-node slack to save energy in MPI programs. In *SC05*, 2005.
21. K. Malkowski, I. Lee, P. Raghavan, and et al. Memory optimizations for tuned scientific applications: An evaluation of performance-power characteristics. Submitted to ISPASS'06.
22. L. McInnes, B. Norris, S. Bhowmick, and et al. Adaptive sparse linear solvers for implicit CFD using Newton-Krylov algorithms. volume 2, Boston, USA, June 17-20, 2003.
23. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, 1999.
24. Y. Saad. *Iterative Methods for Sparse Liner Systems*. SIAM, second edition, 2003.
25. G. Semeraro, D. H. Albonesi, S. G. Dropsho, and et al. Dynamic frequency and voltage control for a multiple clock domain microarchitecture. In *MICRO 2002*, 2002.
26. X. Z. Tang, G. Y. Fu, S. C. Jardin, and et al. Resistive magnetohydrodynamics simulation of fusion plasmas. Technical Report PPPL-3532, Princeton Plasma Physics Laboratory, 2001.
27. Top500.org. Top 500 supercomputer sites. `http://top500.org`, 2005.
28. W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *SOSP'03*, 2003.
29. D. Zhu, R. Melhem, and B. R. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems. In *RTSS'01*, 2001.