# Portable Power/Performance Benchmarking and Analysis with WattProf

Amir Farzad
University of Oregon
farzad@cs.uoregon.edu

Boyana Norris
University of Oregon
norris@cs.uoregon.edu

Mohammad Rashti
RNET Technologies, Inc.
mrashti@rnet-tech.com

## ABSTRACT

Energy efficiency is becoming increasingly important in high-performance computing. Hardware and software tools that enable fine-grained measurement of power for real applications can help understand the power attributes of application components. This paper introduces new software infrastructure based on the WattProf measurement hardware that enables high-resolution, per-component power measurements. We demonstrate this new infrastructure by analyzing the power, and energy efficiency of the the MiniFE proxy application built with different optimization levels. We create models that can be used to guide power-aware optimizations.

## Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

## General Terms

Tools, Performance, Power

## Keywords

power measurement, analysis, performance

## 1. INTRODUCTION

Energy efficiency is becoming increasingly important in high performance computing. In order to optimize the behavior of computations with respect to power and energy, we must be able to perform fine-grained measurements of real applications, which would enable accurate analysis and modeling of their power and energy requirements. The majority of power measurement tools are platform-specific and support varying types and granularities of measurement, which can make analysis difficult. Compared to performance measurement and analysis, power monitoring and subsequent analysis is still a relatively new and rapidly changing area.



We are developing a general-purpose fine-grained power measurement system and associated software infrastructure for the analysis and modeling of power and energy of real HPC applications. The ultimate goal is to provide a complete system that can be used to generate scalable power or energy models based on small-scale accurate power measurements and other more easily measured parameters (e.g., hardware performance counters). Instead of instrumenting the hardware of very large-scale parallel systems, our approach relies on adding sensors to just a few representative nodes and then creating empirical models of per-component (e.g., CPU, memory, network interface) power or energy based on smaller-scale runs on the instrumented nodes, using performance counters or problem parameters as the independent variables. Models can be generated for the entire application or for individual functions. The models can be used for selecting algorithms, for source code optimization (e.g., through manual or automated tuning), or for runtime parameter tuning.

The specific contributions described in this paper include the following.

- Use of the new WattProf board [9] to collect fine-grained power and energy measurements.
- Automated source code instrumentation of C and Fortran codes for collecting function-level power and energy measurements;
- Power and energy analysis and modeling use cases based on this infrastructure.

## 2. RELATED WORK

We briefly overview some existing hardware and software tools that can be used to measure, analyze and model power and energy.

*Power measurement hardware.* The WattProf card used in our system is described in more detail in [9], where it is also compared with other power measurement devices, such as PowerMon2 [1] and PowerInsight [7].

*Measurement software.* Many vendors provide utilities for measuring power and power-related system characteristics. These are typically specific to each platform and do not share a common interface. The Performance API (PAPI) analysis library supports measurement of power and energy values through the PAPI-C interface [12], e.g., through integration with Intel's RAPL. The Energy Measurement Library [3] is a

recent effort aimed at providing a portable, uniform interface to a diverse and comprehensive power and energy measurement tools. Integrating WattProf measurement with PAPI and/or EML is a future goal for our work.

*Analysis and Modeling.* Several research efforts have targeted estimating power or energy based on direct measurement of performance counters. Lim et al. [8] use a small set of counters to correlate well, and independently, with fine-grain measurements that distinguish among power consumption by system components such as CPU, memory, disk, and I/O devices. Their goal is system power estimation, while we are targeting more fine-grain (per function) models that would be useful for tuning applications. In previous work, we [5] and others [2] have also used counter-based power estimation – at that time, the inability to measure power at a fine enough granularity presented obstacles to both building and validating models. Tiwari et. al [10] use a neural network approach to model power and energy of HPC kernels such as matrix multiplication and LU factorization.

# 3. POWER MEASUREMENT

In this section we briefly overview the hardware used for measurement and our instrumentation-based approach.

## 3.1 WattProf Board

The WattProf system consists of a programmable monitoring board (a PCIe expansion card), voltage/current sensor boards for individual system components, host driver, and user-level API [9]. The monitoring board collects power and energy samples from system components (e.g., CPU, memory, compute accelerators, network interface cards, disks, and fans) by using measurement sensors attached to power lines (rails). The board can collect data from up to 128 sensors at up to 12KHz. Raw or processed power and energy data and statistics can be collected using multiple mechanisms, including transfer to the host (via PCIe), transfer to a remote agent via on-board Ethernet, or buffered on the board's local storage for future retrieval.

## 3.2 Source Code Instrumentation Toolkit

The WattProf software API enables us to measure the power and energy consumption of a piece of code through starting and stopping a measurement window by calling the corresponding API functions. Considering the accuracy and the sampling frequency of this card, we are able to measure power and energy of individual function calls of HPC kernels. The granularity of the information that WattProf can gather is similar to performance tools such as PAPI, TAU, and HPCToolkit. Because the WattProf API consists of basic functions that just start or stop the measurement window, we also developed a tool that instruments the source code for power and energy measurement.

This power instrumentation is applied to source code during compilation by using the `-finstrument-functions` flag provided by GNU and Intel compilers, which works with C, C++ and Fortran. This option automatically calls the specified routines at the beginning and end of each function in the source code. Note that this option does not require any manual changes in the target source code. We exploited this functionality for WattProf power monitoring API. It starts a measurement window at the beginning of each function and closes the window at the end of the function.

Our power instrumentation software includes a function responsible for initializing the WattProf card and measurement data structures, and another function responsible for gathering the measurements and generating the corresponding log files at the end of the program. The power instrumentation also keeps track of the measurement window in the context of call stack. Hence, the output log files also report the call path information along the power and energy information.

One of the important issues in instrumentation and profiling source codes is the overhead of this extra measurement. In particular in our power and energy measurement, the instrumentation is not aware of context switching and there are no registers to switch on/off the measurement. Hence we were careful to minimize this overhead. Almost all the extra operations are done either before starting a measurement window or after closing the window.

We used this power instrumentation toolkit to measure, analyze and model the power/energy of HPC benchmarks and proxy applications. Generally this toolkit can be used in automated power and energy analysis and modeling. The data collected with WattProf can be used to help identify power-inefficient code regions and guide power and energy optimization.

# 4. ANALYSIS

Using the *WattProf* platform we have the option to measure the power and energy of different hardware components within a fine-grained window. This level of measurement accuracy helps us gather data about power and energy consumption of entire long-running HPC applications or for short-running fragments, e.g., individual functions. Moreover, we can acquire valuable information which might not be revealed by coarser-grained power and energy measurement. For example we found evidence that reductions on execution time through different optimizations are not linearly related to power or energy for individual functions. This result challenges the commonly accepted result of "the instruction count was directly proportional to energy consumption" which is commonly used in prior work (for example [11]).

In this paper we evaluate our analysis and modeling approach on the miniFE proxy application from the Mantevo benchmark suite [4]. We used the WattProf card on a machine with two Intel Xeon CPUs E5620 with 24GB memory and running Ubuntu 14.04.2 with Linux kernel 3.13. We considered problem sizes ranging from 30x30x30 to 150x150x150 and different numbers of MPI processes ranging from 1 to 8. We compiled the MPI-based miniFE with GCC 4.8.2 with optimization levels `-O0, -O1, -O2` and `-O3` in order to study the effects of optimization on power and energy consumption. For each experiment we ran the corresponding instrumented binary file three times and computed the average value. We collected per-function measurement profiles for the CPU, motherboard, memory and hard disk. While WattProf also supports GPU and network card measurement, because miniFE does not use the GPU and we ran on a single node only, we omitted the graphics and network card measures.

## 4.1 Power

The power efficiency of the CPU and the maximum power usage are two important measures that are used on profiling

and optimization of an HPC application. In some previous studies it is mentioned that the more aggressive optimization levels (`-O3`) may increase the power dissipation while they decrease the energy consumption due to shorter runtimes [6]. Our experiments show that this conclusion for power might not be valid for computations similar to miniFE. The power level depends on many factors including the size of the problem relative to cache size and also on number of MPI processes. Moreover, the *Intel turbo boost* technology introduces complex dynamic behavior that depends on the type of the workload, number of active cores, processor temperature and estimated power and energy consumption. For brevity we show the results for one experiment on miniFE, which are representative of the results we obtained for most problem sizes that do not fit in cache. Figure 1 compares the power consumption for miniFE with size 150x150x150 and different compiler optimizations.

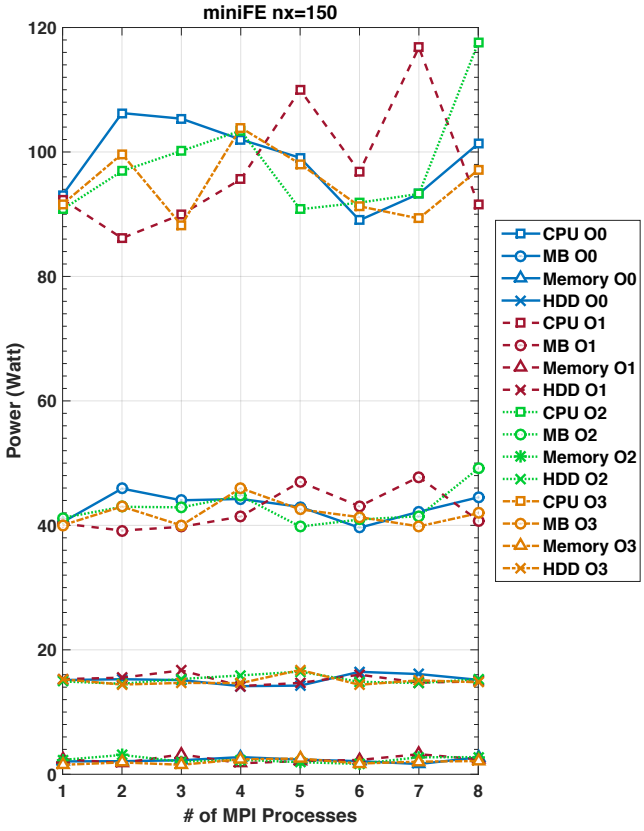



**Figure 1: Power consumption of miniFE bechmmark (problem size 150x150x150) with varying numbers of MPI processes and different levels of compiler optimization.**

These results show that the power does not necessarily increase with more aggressive optimization levels nor with number of processes. For example, CPU power for `-O1` exhibits some unexpected peaks at 5 and 7 processes, which warrants more detailed investigation into the effects of certain optimizations on power consumption.

## 4.2 Energy Measurement

We measured the energy consumption in our experiments for different hardware components. Figure 2 depicts the total energy consumption of miniFE (problem size 150x150x150) and MPI processes $p = 2, 4, 6, 8$. Experiments for other problem sizes showed similar trends.

In Figure 2 we can see that the `-O0` version consumes significantly more energy than other optimization levels. Unsurprisingly, the majority of the energy for miniFE is consumed by the CPU for miniFE. Also for each MPI task count, we can see that `-O3` consistently consumes less energy than the `-O2` optimization level, but `-O1` is sometimes better and sometimes worse than either of the other two optimized versions.

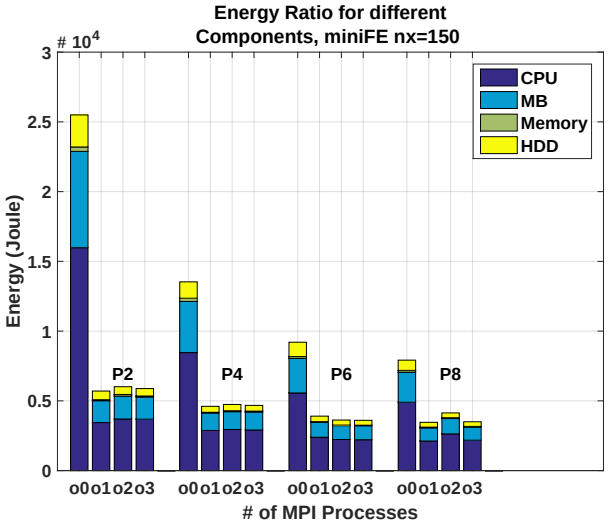


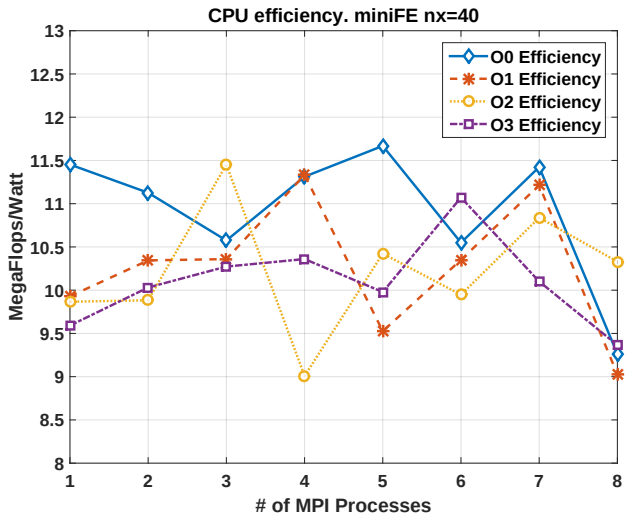

**Figure 2: Energy consumption of different hardware components for miniFE problem size 150x150x150 and MPI processes p=2,4,6,8.**

## 4.3 CPU Efficiency

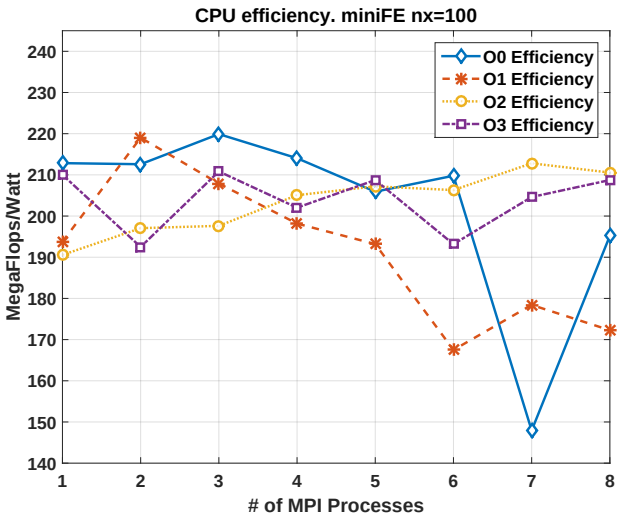
CPU efficiency is typically defined as amount of useful work per Watt. For HPC applications, a common efficiency metric is floating-point operations per Watt. Generally it is desirable to maximize the CPU efficiency.

We computed the *MFLOPs/Watt* measure for miniFE experiments with different optimization levels and varying numbers of MPI processes. We obtained the FLOPS values by using TAU and PAPI to collect performance hardware counter measurements. Figure 3 depicts two problem sizes; 40x40x40 and 100x100x100. The interesting result is that the CPU efficiency has a complex relationship to the optimization level and number of processes. In general, there does not seem to be an easily predictable trend. Although more aggressive optimization levels may produce more FLOP/s they also reduce the total number of operations to solve the same problem size. On the other hand, the power level of the CPU depends on some other factors including the number of active cores and the CPU load and varies dynamically. We suspect more independent variables and parameters should be considered to effectively analyze and model CPU efficiency.

While here we focus on the CPU energy efficiency, we can apply a similar approach to other system components. In particular beyond a single node, models for the network interface power and energy use can help analyze and potentially optimize communications. Moreover, our infrastruc-




(a) CPU efficiency for problem size 40x40x40.




(b) CPU efficiency for problem size 100x100x100.

**Figure 3: CPU efficiency (MFLOPs/Watt) of different optimization levels for two problem sizes.**

ture enables easy definitions of other metrics of efficiency.

## 4.4 Profiling and Optimization

Next we demonstrate the ability of WattProf to profile the power of individual functions. The resolution of this data is comparable to performance metrics. For example the information that we can collect with hardware performance counters for the same functions can be analyzed along with the power and energy measurements. These results can be used to help identify power-inefficient code regions and guide power and energy optimizations.

Figure 4 shows the CPU power usage for top three time-consuming functions in miniFE for different optimization levels ranging from `O1` to `O3`. An interesting observation in this experiment is that for the `miniFE::mytimer()` power decreases with increasing the optimization level ($O1 > O2 > O3$), while for the `miniFE::driver()`, power increases with the optimization level ($O1 < O2 < O3$). Note that this reflects inclusive measurements (i.e., the `mytimer` function is somewhat anomalous in that it wraps other functions for profiling purposes). In future work we will enable exclusive measurement, which would accurately reflect the power and energy of just the instructions local to each function (and not including the code in other functions called from the current function).

This example shows the potential benefit of fine-grained power and energy measurement with WattProf, which can be used to guide more localized optimizations of HPC applications with respect to performance, power and energy.

# 5. MODELING CPU ENERGY

A mathematical model for energy/power consumption is of interest to build predictive models. The fine-grain results that we gathered from WattProf can be used for building accurate models. For example, Figure 5 depicts the energy consumption for miniFE compiled with `-O3` flag. It shows the energy along the number of MPI processes ($p = 1, 2, ..., 8$) and the problem size $nx = 30, 40, ..., 150$. The mesh surface in Figure 5 shows the fitted model and the

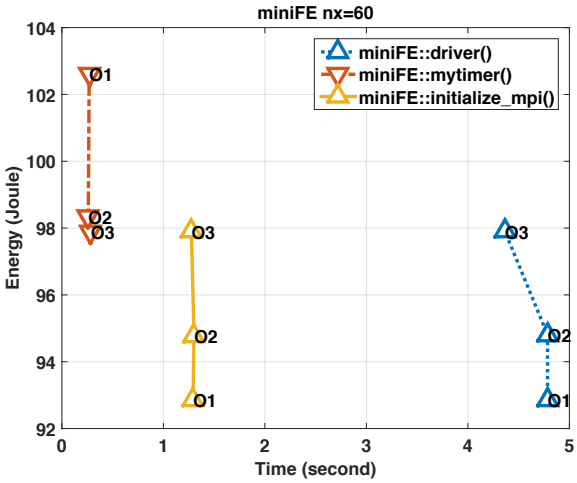



**Figure 4: Energy consumption of the top three miniFE functions vs. execution time.**

points show actual measurements.

We fitted a bivariate cubic polynomial model. Equation 1 shows the fitted cubic polynomial for estimating CPU energy for varying numbers of processes and problem dimensions. For this level of optimization, energy is highly correlated with time ($r = 97.41\%$); the model for time is similar and hence not shown here.

$$\begin{aligned} f(x,y) = &-141.2 + 68.68x + 6.387y + 31.31x^2 \\ &- 5.443xy + 0.07479y^2 - 5.877x^3 \\ &+ 0.9003x^2y - 0.03153xy^2 + 0.001114y^3, \end{aligned} \quad (1)$$

where $x$ is the the number of processes and $y$ is the problem size. The goodness-of-fit statistics for this fitting model shows this bivariate cubic polynomial is acceptable: the $R^2$ value is 97.85% and the $MSE$ is 105.1477 (80% training, 20% testing data split from a total of 104 data points). Figure 6

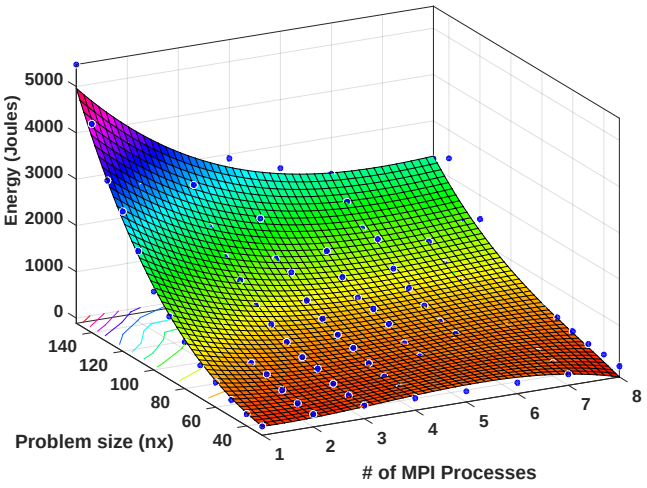



**Figure 5: Bivariate cubic polynomial model on CPU energy consumption for miniFE compiled with O3.**

depicts the absolute error between the experimental energy data and the model.

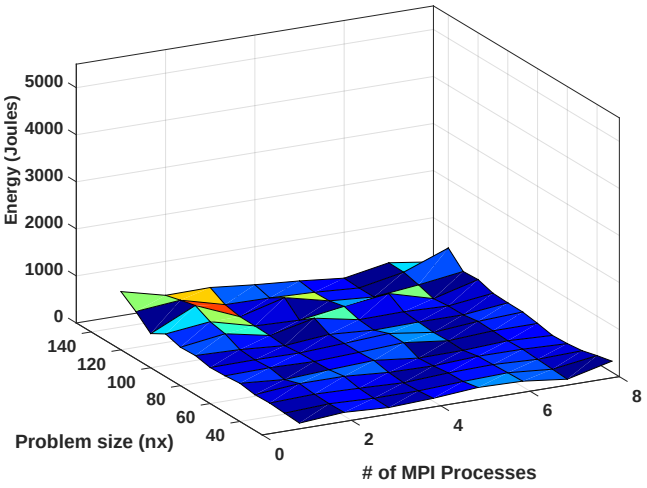



**Figure 6: Absolute error value between the model and experimental energy data ($R^2 = 97.85\%$).**

## 6. CONCLUSION AND FUTURE WORK

We showed that a fine-grained portable performance measurement infrastructure based on the new WattProf card can be used successfully for accurate measurement and analysis of realistic applications. We also constructed a model for the energy consumption of individual system components. This new infrastructure aims to automate the data gathering, analysis and model-generation process for power and energy. We are also working on integrating power measurement and modeling in the Orio autotuning framework.

## Acknowledgments

This work was supported in part by DOE Grant DE-SC0004510.

## 7. REFERENCES

[1] D. Bedard, M. Y. Lim, R. Fowler, and A. Porterfield. Powermon: Fine-grained and integrated power monitoring for commodity computer systems. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pages 479–484, March 2010.

[2] W. L. Bircher and L. K. John. Complete system power estimation using processor performance events. *IEEE Trans. Comput.*, 61(4):563–577, Apr. 2012.

[3] A. Cabrera, F. Almeida, J. Arteaga, and V. Blanco. Measuring energy consumption using EML (Energy Measurement Library). *Computer Science - Research and Development*, 30(2):135–143, 2015.

[4] M. A. Heroux, D. W. Doerer, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, and R. W. Numrich. Improving performance via mini-applications. Technical Report SAND2009-5574, Sandia National Laboratories, Sept. 2009.

[5] K. A. Huck, O. Hernandez, V. Bui, S. Chandrasekaran, B. Chapman, A. D. Malony, L. C. McInnes, and B. Norris. Capturing performance knowledge for automated analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'08)*, 2008.

[6] M. E. Ibrahim, M. Rupp, and S.-D. Habib. Compiler-based optimizations impact on embedded software power consumption. In *Circuits and Systems and TAISA Conference, 2009. NEWCAS-TAISA'09. Joint IEEE North-East Workshop on*, pages 1–4. IEEE, 2009.

[7] J. H. Laros, P. Pokorny, and D. DeBonis. PowerInsight–a commodity power measurement capability. In *Green Computing Conference (IGCC), 2013 International*, pages 1–6, 2013.

[8] M. Y. Lim, A. Porterfield, and R. Fowler. Softpower: Fine-grain power estimations using performance counters. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 308–311. ACM, 2010.

[9] M. Rashti, G. Sabin, and B. Norris. Power and energy analysis and modeling of high performance computing systems using WattProf. In *Proceedings of the 2015 IEEE National Aerospace and Electronics Conference (NAECON)*, July 2015.

[10] A. Tiwari, M. Laurenzano, L. Carrington, and A. Snavely. Modeling power and energy usage of hpc kernels. 2012.

[11] M. Valluri and L. K. John. Is compiling for performance compiling for power? In *Interaction between Compilers and Computer Architectures*, pages 101–115. Springer, 2001.

[12] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore. Measuring energy and power with PAPI. In *Proceedings of the 2012 41st International Conference on Parallel Processing Workshops*, ICPPW '12, pages 262–268. IEEE Computer Society, 2012.