Power Modeling for Phytium FT-2000+/64 Multi-core Architecture

Zhixin Ou
National University of Defense Technology
Changsha, Hunan, China
ouzhixin16@nudt.edu.cn

Juan Chen*
National University of Defense Technology
Changsha, Hunan, China
juanchen@nudt.edu.cn

Yunfang Zhang
National University of Defense Technology
Changsha, Hunan, China
zhangyunfang1995@163.com

Yong Dong
National University of Defense Technology
Changsha, Hunan, China
yongdong@nudt.edu.cn

Zheng Wang
University of Leeds
United Kingdom
z.wang5@leeds.ac.uk

Yuan Yuan
National University of Defense Technology
Changsha, Hunan, China
yuanyuan@nudt.edu.cn

Abstract

Power and energy consumption is the first-class constraint on high-performance-computing (HPC) systems. Understanding the power consumption on such systems is crucial for software optimization and hardware architecture design. Unfortunately, the subtle interactions between the CPU and the memory subsystem make precise power modeling highly challenging on emerging multi-core architectures. This paper presents the first software-based power model for Phytium FT-2000+/64, an ARM-based HPC multi-core architecture. We show that by carefully choosing and modeling a set of system-wide metrics, one can build an accurate power model for the multi-core CPU and the DRAM memory subsystem on a FT-2000+/64 based computer node, two major energy consumers of an HPC system computing node. We evaluate our approach by applying it to HPCC benchmarks and comparing our results against real power measurement. Experimental results show that our approach is highly accurately in modeling power consumption of FT-2000+/64. The average error rate for CPU power modeling in all the scales (8, 16, 32, 64 processes) is 2%, and the average error rate for memory power modeling is about 7.5%.

CCS Concepts. • **Hardware → Power estimation and optimization**.

Keywords. Power modeling, FT-2000+/64, ARM-based multicore processors

PPoPP '20 Workshop, February 22–26, 2020, San Diego, CA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

ACM Reference Format:
Zhixin Ou, Juan Chen*, Yunfang Zhang, Yong Dong, Zheng Wang, and Yuan Yuan. 2020. Power Modeling for Phytium FT-2000+/64 Multi-core Architecture. In *PPoPP '20 Workshop: Principles and Practice of Parallel Programming 2020, Workshop: Benchmarking in the Datacenter, February 22–26, 2020, San Diego, CA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

In an era where computing hardware hits the power wall, increasing the energy efficiency of our computing systems is of paramount importance. Indeed, power and energy consumption has become the first-class design constraint of computing architectures [20]. Although performance optimization is a familiar topic for developers, few are aware of the application power profiles of the underlying hardware. Therefore, developers require tools that analyze the power and energy consumption.

While there is an extensive body of work on building power models for Intel processors [4] and embedded ARM CPUs [3], there is currently little effort in understanding and modeling the power profiles of the emerging HPC multicores built upon the high-performance ARM architecture. Because ARM-based processors are emerging as an alternative building block for HPC systems [22], it is important to have a software tool to help developers to track and monitor the power consumption of their application, without the need of access to expensive power instrument. Having such knowledge is useful not only for benchmarking and better utilizing the computation resources but also for justifying a further increase in the processor core provision.

This paper presents the first software-based power model for the ARM-based Phytium FT-2000+/64 multi-core architecture [12]. At the core of our approach is a set of analytical methods that can precisely estimate the power consumption of the CPU and the DRAM memory subsystem. Our model is based on a set of widely available operating system metrics and hardware performance counters. To develop an accurate

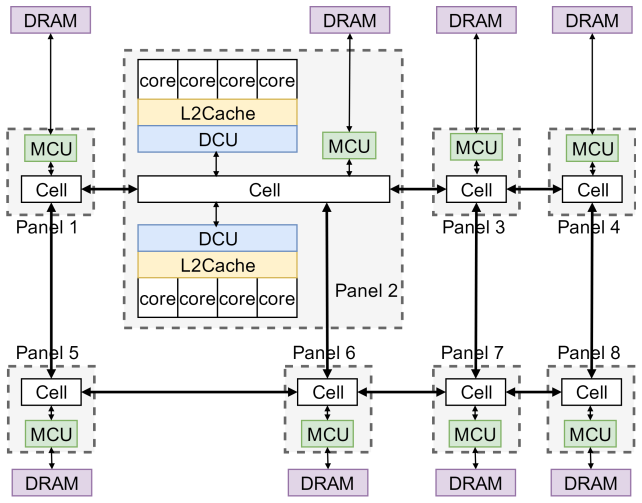

Figure 1. CPU panel (or cluster) and DRAM of the FT-2000+/64 processor.

power model, we apply well-established statistical methods to choose power metrics.

We evaluate our approach by applying it to five representative HPC workloads from the HPCC benchmark suite [9]. We compare the power estimations given by our models against the power reading reported by instrumented hardware power monitors. Experimental results show that our approach can accurately model the power consumption of the CPU and the memory subsystem. The average error rate for CPU power modeling in all the scales (8, 16, 32, 64 processes) is 2%, and the average error rate for memory power modeling is about 7.5%.

The core contribution of this paper is a novel software-based power model for the Phytium FT-2000+/64 multi-core architecture. Our results show that it is possible to build an accurate software-based power model for high-performance ARM-based multi-cores, without instrumenting the hardware or the application code.

2 Experimental Setup

2.1 Evaluation Platform

Our work targets the ARM-based FT-2000+/64 multi-core processor. This processor integrates 64 ARMv8 based cores on a single chip. It offers a peak performance of 588.8 Gflops for double-precision operations, with a typical power consumption of 90 Watts. As illustrated in Figure 1, the CPU has eight panels (or clusters) with eight 2.3 GHz cores per panel. Each core has a private 32KB L1 data cache, and a 2MB L2 cache is shared among four cores (core-group). Each panel has an on-chip network router node (Cell) and a tightly coupled MCU memory access controller[12]. The panels are connected through two directory control units (DCU) .

We run a customized Linux OS with Linux Kernel v4.0.2 on FT-2000+/64. For compilation, we use gcc v6.4.0 with the "-O3" compiler option.

2.2 Workloads

We apply our approach to five representative benchmarks from the HPCC benchmark suite [9]. This benchmark set is designed for measuring the performance attributes of high-performance computer systems. We use the following five MPI parallel benchmarks in our experiments:

- **DGEMM:** This benchmark measures the floating-point arithmetic performance by performing double-precision matrix multiplication [8].
- **STREAM:** This benchmark measures the memory bandwidth of computing nodes.
- **PTRANS:** This benchmark tests the network communication capability of parallel matrix transpose.
- **RandomAccess:** This benchmark is designed to measure the random update frequency of the memory subsystem.
- **FFT:** This measures the floating-point arithmetic capability by performing one-dimensional discrete Fourier Transform using double-precision operations.

2.3 Performance Report

To measure power consumption, we use Baseboard Management Controller (BMC) to take real power readings every second. BMC reads current and voltage of CPU/DRAM power chipset by I^2C bus. Then the product of current and voltage is taken as the real power readings of CPU and DRAM. We compare the power estimation given by our power model against the real power readings. We calculate and report the average error of the estimated power consumption, x_i, and the real power measurement, y_i, over the sampling period, as:

$$error = \frac{1}{n}\sum_{i=1}^{n}\frac{|(x_i - y_i)|}{y_i} \tag{1}$$

3 Power Modeling

3.1 Realtime Power Monitoring

Our power model is based on a set of system-wide performance metrics. We make use of exiting Linux performance tools including *dstat* and *perf* to collect realtime sampling information about the computing systems. Specifically, we use *dstat* to gather information of the system load, such as the CPU utilization in the user, system and idle modes, the memory usage, network communication rates, system interrupts and context switch frequencies. Such information is useful for understanding the load of the CPU and the memory subsystems. We also utilize Linux *perf* to collect hardware performance counter events including the L1 cache miss rate, the number of instructions, and the last-level cache (LLC, which represents L2 cache in FT-2000+/64 because it has no L3 cache) and TLB miss rate etc., as well as operating system kernel level information like the number of system calls. Both tools provide a wide range of metrics and indicators for performance and power modeling. The overhead for extra power increase by using *dstat* and *perf* is very little, which can be ignored. In order to analyze such an overhead, we recorded the CPU and DRAM power consumption without and with *dstat* and *perf* tools and then compared the two values. The difference of these two values is less than 1 Watts.

The key challenge is to choose the right metrics to drive an accurate power model. We achieve this by applying correlation analysis. This is described in the next section.

3.2 Selection of Performance Indicators

One of the key aspects of building a successful power model is developing the right metrics to characterize system behavior. In this work, we considered over 60 candidate metrics or events reported by *dstat* and *perf*.

3.2.1 Correlation coefficient analysis. Our goal is to choose metrics that carry the most useful information for power modeling. By reducing the number of model parameters, we are also improving the generalization ability of our approach, i.e. reducing the likelihood of over-fitting on the training data. Initially, we use correlation-based feature selection. If the pairwise correlation is high for any pair of parameters, we drop one of them and keep the other; retaining most of the information. We performed this by constructing a matrix of correlation coefficients using the Pearson coefficient [25]. This metric quantifies the linear correlation between two variables. The coefficient value falls between -1 and +1. The closer the absolute value is to 1, the stronger the correlation between the two features being tested.

To do so, we profile our benchmarks and record the information given by *dstat* and *perf* during each sampling window. We set the sampling window to one second. We found that this frequency is sufficient for building a good model for FT-2000+/64. We then construct a matrix of correlation coefficients to evaluate the correlation coefficient of each metric to the power reading for each sampling period. We then calculate the average correlation coefficient per metric pair across the sampling periods.

3.2.2 Metrics for CPU power modeling. For choosing metrics for CPU power modeling, we set a correlation coefficient threshold of 0.2 and only keep metric that has a correlation coefficient value higher than that threshold with the power reading. Doing this leaves us with 11 metrics. We then calculate the correlation coefficient values among each metric pair. If any two metrics have a correlation coefficient greater than 0.9, we keep one metric. As a result, we keep four metrics given by *dstat* for CPU power modeling:

- `usr`: CPU utilization in the user mode.
- `used`: The amount of memory used.
- `csw`: The number of context switches.
- `int`: the number of system interrupts.

Besides the above four metrics, we also use the hardware performance counter event L2 data cache writeback rate (l2d_cache_wb) for CPU power modeling. Although l2d_cache_wb strongly correlates with memory power (See Section 3.2.3), it also benefits CPU power modeling from training data.

3.2.3 Metrics for DRAM power modeling. For memory power modeling, we found that the set of *dstat* metrics used by the CPU power model has a correlation coefficient value of less than 0.5 to the memory power measurement. This suggests using this set of metrics only is not sufficient to build a good power model for the DRAM. We then turn to look at the hardware performance counter events given by *perf*. By applying the correlation analysis to the hardware performance counters, we found that the L2 data cache writeback rate (l2d_cache_wb) has the strongest correlation with the memory power reading. Intuitively, this metric record the number of cache stores to the memory, which thus strongly correlates with the power consumption of the memory subsystems. This finding is line with the observation reported in prior work [1] that shows the LLC missing rate is a performance indicator closely related to memory power consumption. Putting together, for memory power estimation, we use the four *dstat* metrics used for CPU power modeling as well as the l2d_cache_wb event given by *perf*.

3.3 Power Prediction

After we have determined the most useful parameters for power modeling, we are able to build analytical models to estimate the power consumption for the CPU and DRAM based on the chosen metrics. In this work, we develop a simple yet effective linear model based on the selected metrics for CPU and DRAM power consumption as:

$$Power_{pred} = w_0 x_{usr} + w_1 x_{used} + w_2 x_{csw} + w_3 x_{int} + w_4 x_{l2d\_cache\_wb} \tag{2}$$

where w_i are the model coefficients and x_i are the chosen metrics for power modeling.

The model coefficients are determined by profiling the benchmarks on training inputs. We store performance metric values that are read in time period $[1, t]$ into matrix X whose size is $5 \times t$. We also store the real-time power readings of during $[1, t]$ as P_C and P_M for the CPU and the memory respectively. We then use the least square method to calculate the coefficients of the linear model:

$$\begin{aligned} W_C &= (X^T X)^{-1} X^T P_C \\ W_M &= (X^T X)^{-1} X^T P_M \end{aligned} \tag{3}$$

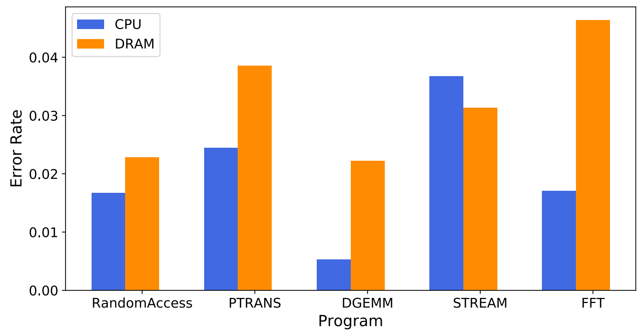

Figure 2. Prediction error rates when running benchmarks using eight processes on an eight-core CPU panel. Our approach delivers an average error rate of 2.0% and 3.2% for CPU and DRAM memory power consumption respectively.

where W_C and W_M are coefficient vectors containing the chosen performance metrics for the CPU and memory respectively. Each element of the vector correspond to a performance metric. We estimat power consumption as:

$$\begin{aligned} P_C^{pre} &= X^T W_C \\ P_M^{pre} &= X^T W_M \end{aligned} \tag{4}$$

To use our models, we profile the target program on training inputs and then apply Formula (3) to fit the training data to determine the model coefficient for the program. Once the model coefficients are instantiated, the linear model can then be used to estimate power consumption for any *new, unseen* inputs for the target program using Formula (4).

4 Experimental Results

This section presents the preliminary results of our approach, showing that it is possible to build an accurate software-based power model for FT-2000+/64.

4.1 Overall Results

In this experiment, we run all benchmarks using eight processes, one process per core on the eight-core CPU panel (Figure 1). We run the programs with an input size of 40,000 matrix elements.

Figure 2 reports the error rate given by our model to the real power measurement. Our approach is highly accurate in estimating the power consumption of the CPU subsystem, with an error rate of as low as 0.5% (up to 3.7%), and an average error of 2.0%. We observe that STREAM has a relatively higher estimation error of around 3.7%. This is largely due to the dynamic behaviour of the program, e.g., changing program phases and memory access patterns. We observe a slight increase in the prediction error when modeling the memory power consumption. The error rate for memory power ranges from 2.1% to 4.6%, and the average error rate is 3.2%. Our future work will address the dynamic program behaviour. Nevertheless, our software-based model is accurate in modeling power consumption of the CPU and the DRAM memory for FT-2000+/64.

Figure 3 shows all the details of real CPU and DRAM power prediction results for these five benchmarks with input size of 40,000 matrix elements. Real-time power prediction values are compared by real power readings. Time interval is one second.

When input size is changed to 80,000, the results of these five benchmarks are shown in Figure 4. Comparing Figure 4 and Figure 3, we can see that our model can get the expected power prediction results with various data sizes.

4.2 Scalability Study

In this experiment, we investigate how the error rate changes as the number of parallel processes increases. We use benchmark ~~DEGMM~~ as a case study by running it with 8, 16, 32 and 64 processes, one process per processor core.

The results are shown in Figure 5. As expected, the error rate increases as the number of parallel processes to use increases, which can reach 10.8% when running the program using all the 64 processor cores. When using more than eight processors, the program will run on more than one CPU panel. Using more than eight processors will involve data communications and synchronization across CPU clusters. In this scenario, the routing and communication overhead between network interfaces will also increase. Such overhead does not provide by the performance metrics used by our model. To improve the performance of power modeling, we will need to find ways to explicitly or implicitly capture the communication and routing overhead. Unfortunately, the current performance counters provided by FT-2000+/64 do not track cross-cluster data communications. We believe the performance of our software-based power model can be further improved if future hardware supports monitoring of cross-cluster communications.

5 Discussion

Our work represents the first software-based power model for FT-2000+/64. Naturally, there is further room for improvement, and we discuss a few points here.

As briefly outlined in Section 4.2, the limits information provided by the performance counters of FT-2000+/64 has a negative impact on the performance of our power model. As depicted in Figure 1, there are a number of energy consumers on the FT-2000+/64 processor, including the processor core, L2 cache, directory and network, memory controller, IO and other peripherals. The current hardware performance counters of FT-2000+/64 only track information for each individual CPU panel (or cluster) but not across CPU clusters. Having such information will be useful for modeling the interactions among parallel processes running on different CPU panels and their power profiles. An alternative approach is to develop a predictive model to capture

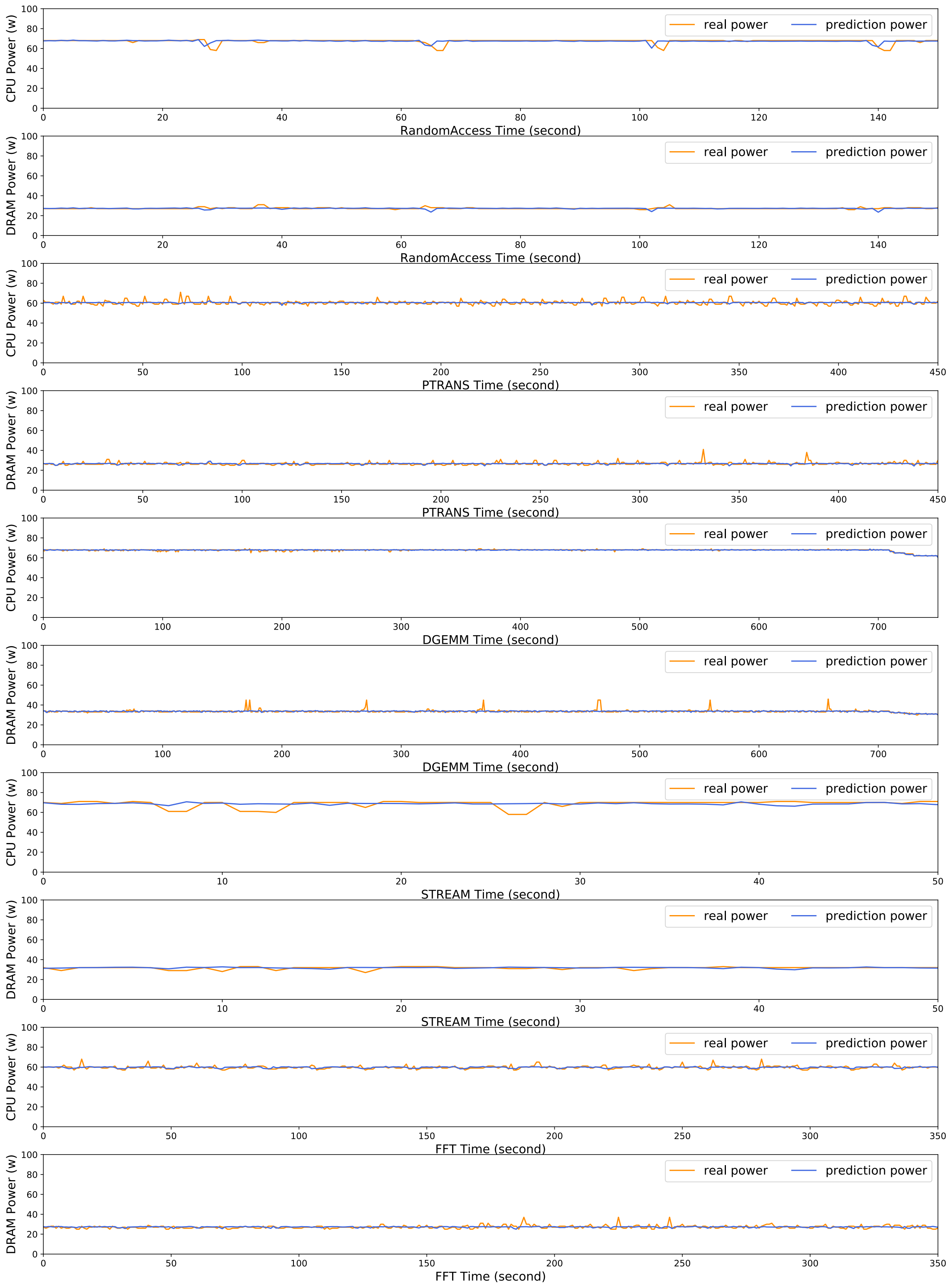

Figure 3. Real power and prediction power of CPU and DRAM for five benchmarks with input size of 40,000 matrix elements, where time interval is one second.

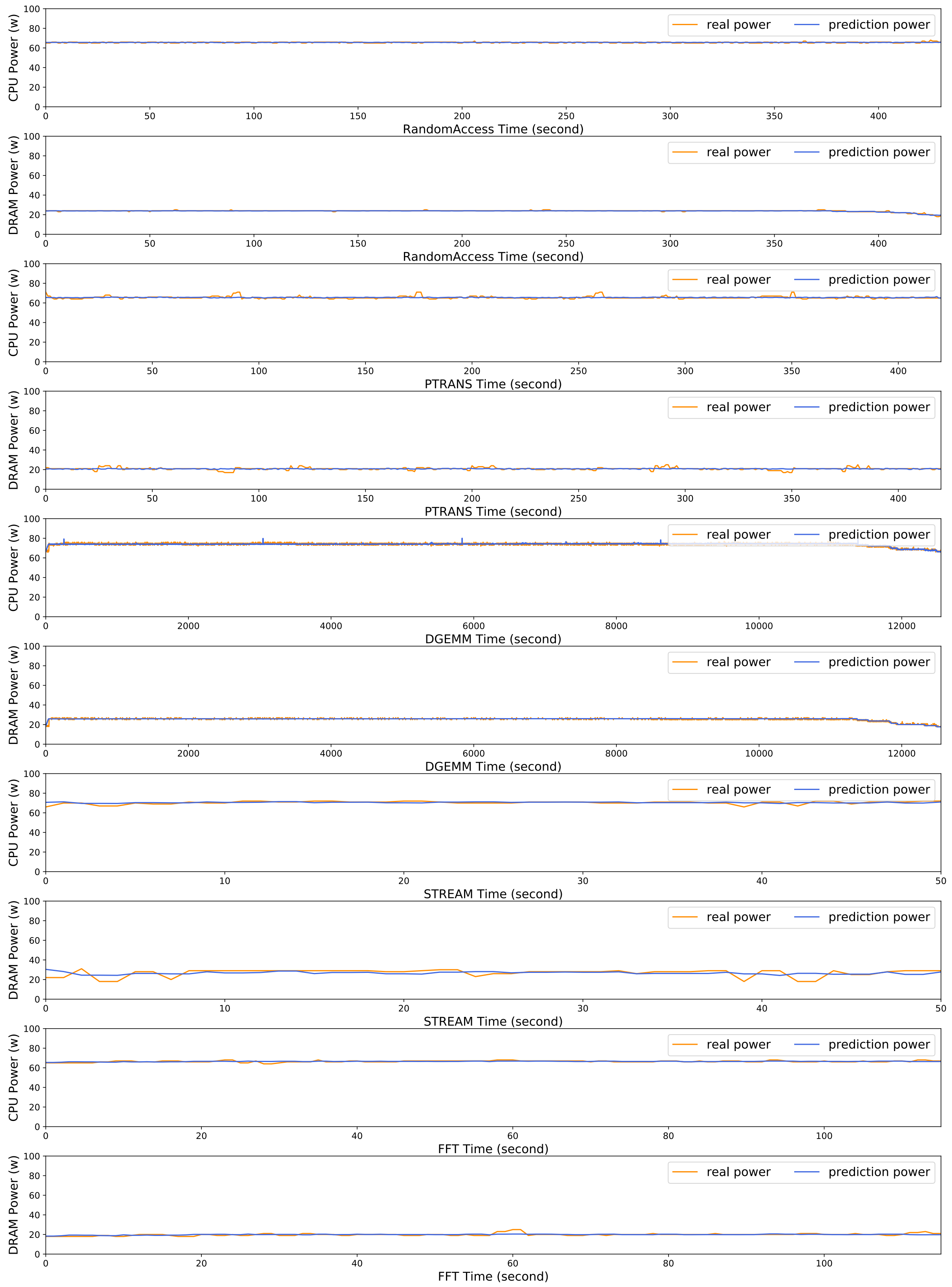

Figure 4. Real power and prediction power of CPU and DRAM for five benchmarks with input size of 80,000 matrix elements, where time interval is one second.

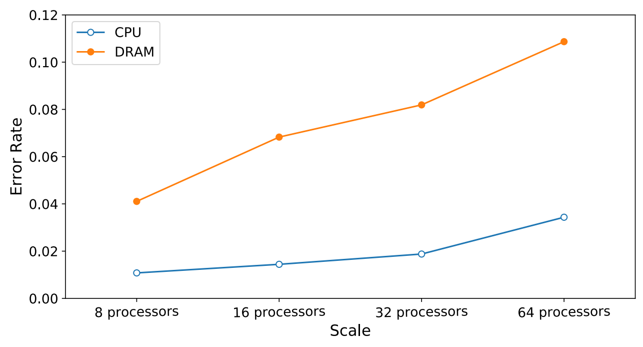

Figure 5. Prediction error rates under different number of parallel processes for five benchmarks.

the interactions of the software and hardware. Prior work in the area of machine-learning-based systems optimisation provides evidence showing that this could be a viable means to overcome the limits in the hardware implementation [5, 6, 15, 16, 21, 23, 24]. Effort in this direction includes performance modeling for the FT-2000+/64 architecture [2]. We leave this as our future work.

6 Related Work

Energy and power modeling is certainly not a new topic. A common approach is to utilize hardware power sensors to report energy consumption [10]. Such an approach has the advantage of being accurate in power reading, but it increases the hardware cost.

Another common approach is to monitor the power consumption of hardware components through software-based modeling [13, 26]. A wide range of methods has been developed to collect samples of hardware event rates from hardware performance counters and system-wide metrics, which are then used to derive an estimation of power consumption [7, 18, 27]. These approaches often employ analytical or predictive models to estimate the power composition of hardware components or program activities. Our approach falls into this category. While there are studies for power modeling and monitoring that target Intel processors [4] and embedded ARM CPUs [3, 19], there is currently little work in power modeling for the high-performance ARM architecture. The Mont-Blanc 2020 project [19] aims at delivering a processor with high energy efficiency for HPC and server workloads, and some energy-efficient computing methods are used in some classic scientific computing application implementations [17]. Given that ARM-based multi-cores are emerging as an alternative architecture for HPC and data centers, it is attractive to have a low-cost technique to characterize the power profiles of such systems. Our work is among the first attempts in this direction, by specifically targeting the ARMv8 based FT-2000+/64 multi-core processor. Our preliminary results show that one can accurately estimate the power consumption of the CPU and the memory sub-system through a set of system-wide performance indicators. We hope our study can encourage further research along this line.

There are also researchers like Donglin Chen et al. [2] presented a quantitative study for characterizing the scalability of sparse matrix-vector multiplications (SpMV) on Phytium FT-2000+, and show that while many computation-intensive SpMV applications contain extensive parallelism, achieving a linear speedup is non-trivial on this platform. They developed a performance analytical model based on a regression tree, and showed that their model is highly effective in characterizing SpMV scalability. Mantovani et al. [14] presented a performance and energy-efficiency study aimed at demonstrating how a single tool can be used to collect most of the relevant metrics, and show how the same analysis techniques can be applicable on different architectures, analyzing the same HPC application on a high-end and a low-power cluster. Thomas Ilsche et al. [11] identified five partly contradictory requirements that characterize different infrastructures, and pushed the boundaries of two measurements in two projects.

7 Conclusion

This paper has presented a novel software-based power model for the ARM-based Phytium FT-2000+/64 multi-core architecture. Our model is based on a set of operating system performance indicator and a hardware performance counter. We evaluate our approach by applying it to five representative HPC workloads and compared the power estimation with real power measurement. Experimental results show that our approach can derive accurate power estimation. The average error rate for CPU power modeling in all the scales (8, 16, 32, 64 processes) is 2%, and the average error rate for memory power modeling is about 7.5%.

Acknowledgments

This work is supported in part by the Advanced Research Project of China under grant number 31511010203 and the Research Program of NUDT grant number ZK18-03-10.

References

[1] William Lloyd Bircher and Lizy K. John. 2007. Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events. In *the Proceedings of 2007 IEEE International Symposium on Performance Analysis of Systems and Software, April 25-27, 2007, San Jose, California, USA.*

[2] Donglin Chen, Jianbin Fang, Chuanfu Xu, Shizhao Chen, and Zheng Wang. 2019. Characterizing Scalability of Sparse Matrix–Vector Multiplications on Phytium FT-2000+. *International Journal of Parallel Programming* (2019), 1–18.

[3] Jee Whan Choi, Marat Dukhan, Xing Liu, and Richard Vuduc. 2014. Algorithmic Time, Energy, and Power on Candidate HPC Compute Building Blocks. In *the Proceedings of 2014 IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS 2014).* 447–457.

[4] Gilberto Contreras and Margaret Martonosi. 2005. Power prediction for intel XScale® processors using performance monitoring unit events. In *the Proceedings of the 2005 International Symposium on Low Power Electronics and Design (ISLPED 2005)*.

[5] Chris Cummins, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. 2017. End-to-end deep learning of optimization heuristics. In *the Proceedings of 26th International Conference on Parallel Architectures and Compilation Techniques (PACT 2017)*. IEEE, 219–232.

[6] Chris Cummins, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. 2017. Synthesizing benchmarks for predictive modeling. In *2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 86–99.

[7] Matthew Curtis-Maury, Ankur Shah, Filip Blagojevic, Dimitrios S Nikolopoulos, Bronis R De Supinski, and Martin Schulz. 2008. Prediction models for multi-dimensional power-performance optimization on many cores. In *the Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*. 250–259.

[8] Paolo D'Alberto and Alex Nicolau. 2005. Adaptive Strassen and ATLAS's DGEMM: A fast square-matrix multiply for modern high-performance systems. In *the Proceedings of the 8th International Conference on High Performance Computing in Asia Pacific Region (HPC Asia)*.

[9] Dongarra and P J., Luszczek. 2005. Introduction to the HPCChallenge Benchmark Suite. In *ICL Technical Report, ICL-UT-05-01, 2005*.

[10] A. Ekin, S. Pankanti, and A. Hampapur. 2004. Initialization-independent spectral clustering with applications to automatic video analysis. In *the Proceedings of 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*.

[11] Thomas Ilsche, Robert Schöne, Joseph Schuchart, Daniel Hackenberg, Marc Simon, Yiannis Georgiou, and Wolfgang E. Nagel. 2019. Power measurement techniques for energy-efficient computing: reconciling scalability, resolution, and accuracy. *SICS Software-Intensive Cyber-Physical Systems* 34 (2019), 45–52.

[12] Feiteng Information Technology Company Limited. [n.d.]. FT-2000 Series of Enterprise Application Processors -> FT-2000+/64. http://www.phytium.com.cn/Product/detail?language=1&product_id=7. Feb. 2020.

[13] Anita Lungu, Pradip Bose, Alper Buyuktosunoglu, and Daniel Sorin. 2009. Dynamic Power Gating with Quality Guarantees. In *the Proceedings of the International Symposium on Low Power Electronics and Design*.

[14] Filippo Mantovani and Enrico Calore. 2018. Performance and Power Analysis of HPC Workloads on Heterogeneous Multi-Node Clusters. *J. Low Power Electron.* 8, 2 (2018).

[15] Vicent Sanz Marco, Ben Taylor, Barry Porter, and Zheng Wang. 2017. Improving spark application throughput via memory aware task co-location: A mixture of experts approach. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*. 95–108.

[16] Lev Mukhanov, Pavlos Petoumenos, Zheng Wang, Nikos Parasyris, Dimitrios S Nikolopoulos, Bronis R De Supinski, and Hugh Leather. 2017. ALEA: A fine-grained energy profiling tool. *ACM Transactions on Architecture and Code Optimization (TACO)*. 14, 1 (2017), 1–25.

[17] G. Oyarzun, R. Borrell, A. Gorobets, F. Mantovani, and A. Oliva. 2018. Efficient CFD code implementation for the ARM-based Mont-Blanc architecture. *Future Generation Computer Systems*. 79 (2018), 786–796. https://doi.org/10.1016/j.future.2017.09.029

[18] Rolf Rabenseifner, Georg Hager, and Gabriele Jost. 2009. Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In *the Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. IEEE, 427–436.

[19] N. Rajovic, A. Rico, F. Mantovani, D. Ruiz, J. O. Vilarrubi, C. Gomez, L. Backes, D. Nieto, H. Servat, X. Martorell, J. Labarta, E. Ayguade, C. Adeniyi-Jones, S. Derradji, H. Gloaguen, P. Lanucara, N. Sanna, J. Mehaut, K. Pouget, B. Videau, E. Boyer, M. Allalen, A. Auweter, D. Brayford, D. Tafani, V. Weinberg, D. Brömmel, R. Halver, J. H. Meinke, R. Beivide, M. Benito, E. Vallejo, M. Valero, and A. Ramirez. 2016. The Mont-Blanc Prototype: An Alternative Approach for HPC Systems. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 444–455. https://doi.org/10.1109/SC.2016.37

[20] Krishna K. Rangan, Gu Yeon Wei, and David Brooks. 2009. Thread motion: fine-grained power management for multi-core systems. In *36th International Symposium on Computer Architecture (ISCA 2009), June 20-24, 2009, Austin, TX, USA*.

[21] Jie Ren, Xiaoming Wang, Jianbin Fang, Yansong Feng, Dongxiao Zhu, Zhunchen Luo, Jie Zheng, and Zheng Wang. 2018. Proteus: network-aware web browsing on heterogeneous mobile systems. In *the Proceedings of the 14th International Conference on Emerging Networking Experiments and Technologies*. 379–392.

[22] Andrew Sloss, Dominic Symes, and Chris Wright. 2004. ARM System Developer's Guide. (2004).

[23] Zheng Wang and Michael O'Boyle. 2018. Machine learning in compiler optimization. 106, 11 (2018), 1879–1901.

[24] Zheng Wang and Michael FP O'Boyle. 2010. Partitioning streaming parallelism for multi-cores: a machine learning based approach. In *the Proceedings of the 19th international conference on Parallel architectures and compilation techniques*. 307–318.

[25] Wikipedia. [n.d.]. Pearson Correlation Coefficient. http://en.wikipedia.org/wiki/Pearson_correlation_coefficient. Feb. 2020.

[26] Feihao Wu, Juan Chen, Yong Dong, Wenxu Zheng, Xiaodong Pan, Yuan Yuan, Zhixin Ou, and Yuyang Sun. 2019. A holistic energy-efficient approach for a processor-memory system. In *Tsinghua Science and Technology*.

[27] Xingfu Wu, Valerie Taylor, Jeanine Cook, and Philip J Mucci. 2016. Using Performance-Power modeling to improve energy efficiency of HPC applications. *Computer* 49, 10 (2016), 20–29.