Survey on Techniques used to Measure Power and Energy Consumption

Manzoor Ahmed[1] Dr. Loganathan R[2]
[1]M.Tech Student [2]Professor & Head of Dept.
[1,2]Department of Computer Science & Engineering
[1,2]HKBK College of Engineering, VTU, Bangalore, India

Abstract— In today's world of big data where we moving up from terascale computing to petascale and approaching exascale computing. Power consumption in such large scale computing is a critical concern. We need to measure power and energy consumption to optimize the power consumed without affecting the performance of the application. Most of the scientific applications use HPC architecture for computation, these applications consume large amount of power. As the power consumed is more we need some sort of tools and techniques to reduce the amount of power consumption. We understand the importance of energy efficiency and power optimization. To optimize the power consumption we first need to measure power and energy usage. The survey mainly focuses on different techniques used in calculating and measuring power and energy consumed by software applications. Applications run on different architectures need to be dynamically monitored. The methodologies can be adopted on different processors to predict power usage. The portability of these techniques is achieved through PMU's (Performance monitoring units). Many of the existing solutions make use of external wattmeter or hardware meters. To optimize power and cost, it is better to use software tools rather than any other hardware meters, as additional cost of hardware can be reduced. This in turn will reduce the operational cost, as the additional meters will use a certain amount of power for its own functionality. The study also highlights the fact that most of the research work is done on Linux platform, as Linux platforms are open source it is easier for researchers to avail all the possible operating system support through packages and other related details, however this is scalable to other platforms.

Key words: Power consumption, Measurement, PMU, RAPL

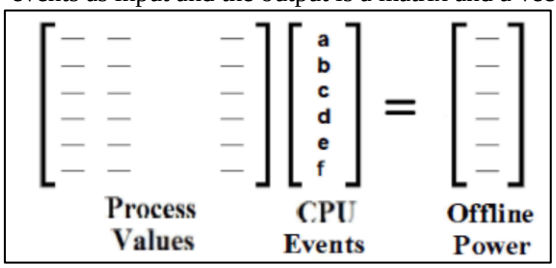
I. INTRODUCTION

There are many software and hardware-based techniques that have been proposed in recent years to lower power consumption and make computing application systems more energy efficient. To achieve power efficiency we first have to measure the power, identify power consumed for an application so that it can be made energy efficient. The various techniques available include calculations using RAPL (Running Average Power Limit), PMU (Performance Monitoring Unit), analyzing power usage from *procfs* file system and Unified Power Profiling API (UPPAPI) using Rodinia and SHOC benchmarks.

IT power consumption is a significant concern has roughly 332 billion kWh [4] as estimated in 2010 [5]. As the demand for power is increasing day by day this field old technology is the hot topic of today. The objective of this survey is to look deeper in the applications and to primarily achieve CPU power optimization. This paper gives an insight on three modern techniques used in measuring power consumption and energy efficiency of software applications.

II. LITERATURE SURVEY

[1] In this paper the author discusses about a model for developing power aware system. It uses an algorithm for power estimation which takes input an application and generate values for multiple runs of a particular process with different environments (eg. No. of processors, size of the input). It proposes a matrix model with process values and CPU events as input and the output is a matrix and a vector.



Where above model is stated as

$$A.X=Y$$

Here, 'A' is counts of variables in 'X', each row represents one run of a process. X is a value of each event. 'Y' is the offline power calculated by Running Average Power Limit (RAPL) for that process. To obtain the matrix A and vector Y we have to estimate the value of each variable in X. Therefore equations designed as,

$$A_{11}X_1+A_{12}X_2+\ldots\ldots+A_{1n}X_n-Y_1 \ldots \text{and so on.}$$

In this way value of vector X can be obtained for simplest CPU operations. In CDAC the Xeon server provides following counters which are of most interest,

1) Instructions: No. of instructions processed by a program. To establish the relation between power and CPU instruction execution is important.
2) Cycles: No. of clock cycles (i.e. Speed) of a process are important in calculating power as different processors have different clock cycles. Heat dissipation has forced the processor manufacturers to avoid further increase of clock cycles.
3) Cache Misses: As the no. of cache misses increase more memory, more time and more power is required.
4) Load-Store Instructions: It is the load and store instructions between register and memory.

Cache misses are considered for the above model. In this paper the author uses a simple and easy implementation method for monitoring application runs for hardware metadata. The matrix vector model is scalable and can be manipulated to for our purpose.

[2] In this paper the author has discuss about the energy consumption of executing software processes. The power consumed by hardware components of a server depends on software running on it. The hardware resources are shared through the operating system and processes under execution.

In a Linux operating system the resources usage of a process including memory, disk, network usage and CPU can availed in detail from the Procfs [18] filesystem. Therefore we have energy consumed by a process p as a function of overall hardware resource usage by that particular process.

The proposed approach has two phases:

1) Calibration Phase: The OS level resource usage that calculates total resource usage by the operating system.
2) Process-Level: The process level resource usage to estimate energy consumption.

In first phase, it explains the usage parameters for disk, memory, CPU and network. It also describes the usage of system-wide OS data. To calculate overall CPU utilization it uses following parameters from procfs filesystem.

- P_{user}: CPU utilization in user space from user programs.
- P_{nice}: CPU utilization in user space from nice priority processes.
- P_{system}: CPU used in execution of instructions in kernel space.
- P_{iowait}: CPU used waiting for I/'O to finish.
- P_{irq}: Time used to service interrupts.
- P_{soft}: Time used to service software interrupts.
- P_{idle}: Time CPU is idle.
- Intr/Sec: No. of interrupts per second

The above information contains the time spent by each CPU in categories like user, irq, soft, idle, nice, system, intr/sec, iowait.

In second phase, it explains the process specific resource usage for the following resources memory, disk, network and CPU. It uses the following parameters.

- utime: Duration of time the process is scheduled in user mode.
- stime: Duration of time the process is scheduled in kernel mode.
- cutime: If the process has child threads this time represents the duration when the child was in user mode.
- cstime: Duration of time process is waiting when the child threads are in kernel mode.

CPU usage is computed for each process by calculating total user time and total system time by reading /proc/<PID>/stat file periodically.

Memory usage is represented by amount memory used by a particular process by parsing the /proc/<PID>/status file in the procfs file system.

Disk I/O usage for a process is read from /proc/<PID>/io file of the Procfs file system. This file provides the amount of data read from and written to the disk at any time.

Network usage can be monitored through data transfers from sockets connected between different processes using the files /proc/net/tcp, /proc/net/tcp6, /proc/net/raw and /proc/net/udp. Each file indicates data transferred and received by an inode.

Based on the above parameters the author summarizes that finding power and energy consumption the main resources is CPU. Along with CPU the memory and disk also plays a vital role.

[3] In this paper the author uses PMU's (performance monitoring unit) for calculating power consumption. This paper indicates in using neural network concept for estimating power.

The author proposes PMU-based neural network model and selection techniques. It uses the following analogous techniques:

1) Linear Regression Model: They have developed this model as a baseline model, for comparison with neural network method.

$$Y_{pwr}=a_1p_1+a_2p_2+\ldots..+a_6p_6$$

Where a_i are parameters to be trained, p_i are selected PMU events, here a_i p_i is added one term at a time for each step.

2) Neural Network Model: It is difficult to train this model but is much more than flexible than linear model. The model develop is a resilient back propagation with weight backtracking and multi-layer neural network with feed-forward structure. This model has one output layer for power value and one input layer for each PMU event, a max of four hidden layers where tested.

   A weighted graph is use to represent mapping between nodes of successive layers. The calculation for values of each node are values from previous layer, weights and squashing function.
3) Generating Training Data: For generating training data it utilize script of workload generation to create parallel workloads in large numbers with variety of characteristics.
4) Feature Extraction: They use target processor to measure all performance events and use them to characterize workload as the initial feature. It uses pruning algorithm in [3]. The features generated are normalized to event counts per million executed instructions. This is required to compare different programs and different runs of the same program.
5) Feature Selection: It uses algorithm 1 to reduce the feature set. Final reprocessing step is called as feature scaling before the learning stage. The experiment was performed to find out whether there is similarity in patterns of power prediction between different architectures. It is best to train the model on one architecture and use it for prediction on other.

The proposed model was able to predict CPU power consumption with little overhead. The PMU based neural network method results showed common patterns in spite of difference in architectures. Portability of the model is analyzed by training one processor and implementing it for prediction on other.

[4] In this the author proposes a technique called Unified Power Profiling API (UPPAPI) using Rodinia and SHOC benchmarks. With such advances in the HPC foundation, logical and business applications are progressively abusing the colossal figure capacities of such parallel processing frameworks. Ideal usage of parallel models result in lessening in application calculation time, while making complete utilize of the equipment close by. This prompts adaptable execution, combined with lower costs. An imperative part of growing exceedingly parallel projects is their energy and vitality utilization. High power utilization prompts colossal measures of warmth dispersal,

which cause expansive expenses and abundant assets for cooling. Clearly, vitality proficient applications help increment the life of the hidden equipment. Given that the co-processors are progressively being utilized as a part of compact devices1, a vitality proficient application on a compact gadget is basic to the life of a battery.

The primary goal of UPPAPI is to provide programmer an easy to use software tool for mapping and analyzing average power, peak power and energy consumption of applications.

To compute the power consumption of the co-processors for individual application the author uses:

1) Power measurement by disabling the coprocessor execution.
2) The program is executed with the co-processor call enabled.

The difference between (1) and (2) gives the power consumed by the co-processors.

The API UPPAPI is tested on ten applications to validate its working, which are Gaussian, SRAD, Hotspot, CFD and LUD from Rodinia Benchmark and FFT, MaxFlops, GEMM, MD and Device Memory from SHOC. An in-depth analysis of SRAD, Gaussian, MaxFlops and FFT applications support the statistical findings.

However, the UPPAPI estimates power and energy reasonably well but it is not very accurate.

Sl #	Measurement of Power		
	Method Used	Input	Output
1	Matrix model	Process values and CPU events	Matrix and a vector.
2	Procfs filesystem	CPU utilization and Time	Time taken per component (Disk, I/O Network & CPU).
3	PMU Method	Different Neural Network Model and Selection Techniques	Different Output for different techniques.
4	UPPAPI	-	Power and Energy measurement.

Table 1: Comparative Study of measuring power consumption

III. CONCLUSION

The survey concludes that by referring the above papers we conclude that for measuring the power consumption of an application it is better to use software tools rather than hardware meters, as the hardware meters consumes a small amount of power for itself. Using hardware meters adds to the additional cost.

Using software tools we can measure power at the process level, CPU level and the RAM, without any external hardware or any additional cost. For compute intensive applications energy consumption is crucial, therefore these applications are run on HPC architecture. To make our world energy efficient and to reduce the operational cost we recommend software tools over hardware meter. However we must ensure that to analyze the power usage of the system as a whole, there is a need to measure power of individual components. To get accurate results we must ensure software approach is as accurate as hardware approach.

From the above study we come to the conclusion that Procfs filesystem methodology calculates power by considering all the aspects of operating systems such as Disk, CPU, Network and I/O. As it calculates power at component level we feel that this approach is more accurate.

ACKNOWLEDGMENT

The survey is carried out to gain knowledge about power consumption techniques and tools for the academic purpose.

REFERENCES

[1] Pooja R. Pawar Prof. U. B. Chavan and Anil Kumar Gupta "A Basic Model for Profiling Power Consumption in HPC Sub-systems" 2016.
[2] Vivek Kumar Singh, Kaushik Dutta and Debra VanderMeer "Estimating the Energy Consumption of Executing Software Processes" 2013 pp. 94-101.
[3] Mario Gutierrez, Saami Rahman, Dan Tamir and Apan Qasem "Neural Network Methods for Fast and Portable Prediction of CPU Power Consumption" 2015
[4] Travis Schluessler, Jacky Romano, Stas Gurtovoy, Guy Zadicario and James Fox "Limiting CPU Power Consumption for Efficient Computation of 3D Workloads" 2012.
[5] Mohak Chadha, Abhishek Srivastava, and Santonu Sarkar "Unified Power and Energy Measurement API for HPC Co-processors" 2016.
[6] Bhatkar, Vijay P., "PARAM parallel supercomputer: architecture, programming environment, and applications," in Parallel Processing Symposium, 1994. Proceedings, Eighth international, vol., no., pp.388-389,26-29 Apr 1994.
[7] Bircher, W.L.; John, L.K., "Complete System Power Estimation Using Processor Perfonnance Events," in Computers, iEEE Transactions on , vo1.61, no.4, pp.563-577, Aplil 2012 doi: 10.1 109/TC.201 1.47.
[8] W. L. Bircher and L. K. John, "Complete System Power Estimation: A Trickle-Down Approach Based on Pelfonnance Events," 2007 iEEE International Symposium on Performance Analysis of Systems & Software, San Jose, CA, 2007, pp. 158- 168.
[9] J. G. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters*, vol. 3, no. 3, p. 034008, 2008.
[10] T. C. I. Agency, "The world factbook: Country comparison:: Electricity - consumption," 2012.
[11] O. C. Project., "Hacking conventional computing infrastructure," 2011.
[12] Kihwan Choi, Ramakrishna Soma and Massoud Pedram, "Dynamic Voltage and Frequency Scaling based on Workload Decomposition", ISLPED '04 Proceedings of the 2004 international symposium on Low power electronics and design.
[13] Hewlett-Packard, Intel Corporation, Microsoft, Phoenix Technologies, Toshiba, "Advanced Configuration and Power Interface Specification, revision 5.0," December 2011.

[14] Sunpyo Hong and Hyesoon Kim, "An integrated GPU power and performance model," Proceedings of the 37th annual international symposium on Computer architecture, p. 280-289, 2010.
[15] R. A. Bridges, N. Imam, and T. M. Mintz, "Understanding gpu power: A survey of profiling, modeling, and simulation methods," ACM Comput. Surv., vol. 49, no. 3, pp. 41:1–41:27, Sep. 2016. [Online]. Available: http://doi.acm.org/10.1145/2962131
[16] M. Burtscher, I. Zecena, and Z. Zong, "Measuring gpu power with the k20 built-in sensor," Proceedings of Workshop on General Purpose Processing Using GPUs - GPGPU-7, 2014.
[17] M. S. Y. S. Gary Lawson, Vaibhav Sundriyal, "Modeling performance and energy for applications offloaded to intel xeon phi," Proceedings of the 2nd International Workshop on Hardware-Software Co-Design for High Performance Computing, 2015.
[18] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi,"McPAT: An Integrated Power, Area, and Timing Modeling Frameworkfor Multicore and Manycore Architectures," in the IEEElACM InternationalSymposium on Microarchitecture, 2009.
[19] J. Treibig, G. Hager, and G. Wellein, "Likwid: A lightweight performance-oriented tool suite for x86 multicore environments," in 39th International Conference on Parallel Processing Workshops, 2010.
[20] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and responsive power models for multicore processors using performance counters," in 24th ACM International Conference on Supercomputing, 2010.

