Modelado estadístico de potencia usando contadores de rendimiento sobre Raspberry Pi

Juan Manuel Paniego[1], Leandro Libutti[1], Martin Pi Puig[1], Franco Chichizola[1], Laura De Giusti[1,2], Marcelo Naiouf[1], Armando De Giusti[1,3]

[1]Instituto de Investigación en Informática III-LIDI, CEA-CIC, Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina.
[2]Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC), La Plata, Argentina.
[3]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), CABA, Argentina.
{jmpaniego, llibutti, mpipuig, francoch, ldgiusti, mnaiouf, degiusti}@lidi.info.unlp.edu.ar

Resumen. Controlar la disipación de potencia y la temperatura es una gran preocupación en todos los sistemas informáticos modernos. No obstante, obtener información sobre el consumo de energía del procesador y del sistema puede no ser una tarea trivial. Afortunadamente, los procesadores de hoy en día cuentan con una gran cantidad de contadores de hardware para monitorear diferentes eventos en CPU y memoria. En este trabajo, se diseña un nuevo modelo estadístico de estimación de potencia destinado a la placa de desarrollo embebido Raspberry Pi 3. El modelo mapea valores de ciertos contadores de rendimiento a consumo de potencia del dispositivo a través de regresión lineal. Se analizan decenas de aplicaciones correspondientes a benchmarks clásicos, obteniendo un error promedio menor al 6.8% tanto para soluciones secuenciales como para algoritmos paralelos utilizando OpenMP.

Palabras claves: Consumo energético, Estimación de potencia, Raspberry Pi, Contadores de rendimiento, Modelo estadístico**.**

1. Introducción

En la actualidad, el consumo energético se ha convertido en una métrica de gran importancia que juega un papel fundamental a la hora de especificar el diseño de un sistema. Si bien hay muchas situaciones en las que es deseable ceder rendimiento en pos de un menor consumo energético, la performance sigue siendo el objetivo principal, por lo que la energía pasa a ser una preocupación de segundo nivel para los usuarios finales. Sin embargo, la constitución de un equilibrio entre el consumo de energía y los requisitos de rendimiento es de suma importancia para aprovechar de manera eficiente los recursos disponibles en muchos entornos, desde pequeños sistemas embebidos hasta grandes centros de cómputo de alto rendimiento (HPC).

Aunque los procesadores multi-core proporcionan una mejor relación rendimiento/consumo respecto a los single-core, la disipación de potencia continúa siendo un limitador de rendimiento clave para ambas arquitecturas. Recientemente, se han propuesto varias técnicas micro arquitectónicas para minimizar los problemas energéticos, como la adaptación dinámica de recursos del procesador [1][2][3] y el escalado dinámico de tensión y frecuencia (DVFS) [4][5].

La monitorización energética del procesador es importante para generar una correcta administración y, por consiguiente, reducir su consumo cumpliendo con los límites de disipación térmica de potencia del dispositivo. Luego, el control de energía requiere una supervisión en línea de la potencia consumida.

Si bien es posible realizar mediciones por hardware al sistema para implementar un control de potencia de grano fino, tal instrumentación es costosa y no comúnmente disponible. Generalmente, se basa en empotrar instrumentos de medición para monitorear cada nodo. Además, la latencia y los períodos de muestreo de las herramientas utilizadas pueden ser grandes en comparación con las escalas de tiempo en las que las cargas de trabajo pueden variar.

No obstante, los procesadores proporcionan un conjunto de contadores de hardware que se pueden utilizar para monitorear una amplia variedad de eventos relacionados con la performance del sistema con una alta precisión. Dado que cada evento está asociado con un determinado gasto de energía, cualquiera de ellos puede ser utilizado como parámetro en un modelo de rendimiento. Aunque el número de contadores suele ser limitado, los mismos permiten contabilizar una amplia diversidad de eventos. Por lo tanto, el problema de crear un modelo útil es encontrar un conjunto restringido de eventos que describa la mayor parte de la variación en potencia. Aunque algunos eventos representan actividades con poco impacto en la energía, otros exhiben una alta correlación.

Dadas las restricciones que presenta la instrumentación por hardware, construir un modelo estadístico es una alternativa viable para realizar la estimación del consumo energético del sistema en tiempo real. En este trabajo se entrena un modelo de potencia basado en regresión lineal y eventos de rendimiento para la placa de desarrollo Raspberry Pi 3. El modelo es validado a través de un conjunto de aplicaciones con diferentes cargas de trabajo. Esta placa presenta un set limitado de contadores, los cuales a su vez soportan una cantidad reducida de eventos. Desafortunadamente, esta placa no presenta contadores para monitorear el consumo energético del sistema.

Este artículo se organiza de la siguiente manera: en la Sección 2 se recopilan los trabajos relacionados en el ámbito de la predicción del consumo energético. Luego, en la Sección 3 se presenta la metodología para el diseño del modelo estadístico. Seguidamente, en la Sección 4 se detallan los procedimientos realizados para la construcción del modelo. Luego, en la Sección 5 se demuestra la validez de la estimación alcanzada. Finalmente, en la Sección 6 se comentan las conclusiones y las líneas de trabajo futuras.

2. Trabajos relacionados

En esta sección se presentan algunas de las investigaciones previas relacionadas con el presente trabajo. Isci et al. [5][6] construyó una metodología de modelado de potencia en tiempo real basada en el uso de contadores de rendimiento para identificar las diferentes fases de potencia dentro de un determinado código científico. Bellosa analizó el consumo energético total del sistema y lo comparó frente a la información obtenida por los contadores de rendimiento en [7]. Por otro lado, Joseph et al. [8] propuso el uso de los contadores de rendimiento como una base para la estimación de potencia del procesador y sus subcomponentes. Luego, Contreras et al. [9] utilizó contadores de rendimiento para estimar en tiempo real el consumo de potencia sobre un determinado procesador y su memoria. Además, propone que este esquema puede servir como base para la estimación en sistemas embebidos.

Por otra parte, Bircher et al. [10][13][19] exploró el uso de los contadores de rendimiento para predecir el consumo energético de diversos subsistemas como la CPU, memoria, chipset, E/S, disco y GPU. El mismo, fue desarrollado y validado sobre dos plataformas distintas. Asimismo, Rodrigues et al. [18] estudió el uso de los contadores de rendimiento aplicados en la estimación del consumo energético en tiempo real sobre dos arquitecturas diferentes: una orientada a alto rendimiento y la restante basada en bajo consumo.

Singh et al. [11] construyó una estimación del consumo de potencia en tiempo real haciendo uso de los contadores de rendimiento. Los mismos son recolectados a través de micro-benchmarks sobre códigos secuenciales y multihilo. Lee et al. [12] sugirió crear modelos de predicción de potencia y rendimiento usando diferentes parámetros de diseño de hardware y contadores de rendimiento. En el mismo, se realiza una combinación de modelos lineales y no lineales. Se utilizó, además, un framework de simulación para evaluar los errores de predicción.

Pusukuri et al. [14] desarrolló un modelo de potencia lineal basado en solo un par de predictores. Funciona para diversas frecuencias de CPU con una gran precisión. El modelo posee una alta portabilidad sobre arquitecturas x86 ya que los predictores utilizados se encuentran presentes en la mayoría de los nuevos procesadores. También, Jiménez et al. [15] caracterizó el comportamiento térmico y energético de un sistema basado en IBM POWER 6. De esta manera, diseñó el primer modelo analítico de consumo de potencia sobre esta arquitectura, el cual está basado en regresión lineal y contadores de rendimiento. En este sentido, Lim et al. [16] presentó y evaluó un enfoque general para construir modelos de estimación de potencia del sistema basados en contadores de rendimiento. A través de esta técnica, construyó un modelo estadístico de regresión lineal para el Intel Core I7 utilizando diferentes frecuencias, cantidad de núcleos y tamaño de problema. Aunque el artículo presenta resultados interesantes, los procesadores Intel ya disponen de un software de estimación de potencia propio conocido como Intel RAPL (Running Average Power Limit), previamente analizado en trabajos como [20][21][22].

Por otro lado, Lively et al. [17] desarrolló un conjunto de modelos híbridos de estimación de performance y consumo centrados en las aplicaciones. El mismo analiza

un conjunto de códigos científicos en su implementación MPI/OpenMP y genera un procedimiento adecuado para llevar a cabo el modelado y la validación.

Si bien los trabajos anteriores analizaron e implementaron diferentes técnicas de estimación energética, la arquitectura objetivo de los mismos son los procesadores CPU tradicionales. No obstante, este artículo se centra en realizar un estudio de similares características sobre procesadores de arquitectura ARM.

3. Metodología

Generalmente, el diseño de estos modelos estadísticos comprende dos aspectos básicos. En primer lugar, analizar una serie de aplicaciones que incorporen diferentes comportamientos computacionales, con el fin de extraer información importante sobre el rendimiento de una determinada arquitectura. Por otra parte, evaluar la performance del sistema (tiempo de ejecución) y/o el consumo de potencia del mismo.

Luego de la recolección de los datos, se selecciona el método estadístico a utilizar para la estimación objetivo. Los modelos existentes abarcan regresiones lineales, gaussianas, máquinas de soporte vectorial, redes neuronales, entre otros. Particularmente, en este trabajo se utiliza el método matemático de regresión lineal, el cual permite aproximar la relación de dependencia entre la potencia (variable dependiente) y la información que provee el hardware (variables independientes). Este modelo puede ser descrito de la siguiente manera:

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

El conjunto de n observaciones se encuentra denotado por $\{Y_i, X_1, X_2, \ldots, X_n\}$ donde Y_i es la variable dependiente y X_i son las variables predictoras. Además, $\{\beta_0, \beta_1, \ldots, \beta_n\}$ son los coeficientes de las variables predictoras. Cabe destacar que cada uno de los predictores X_i se encuentra normalizo de acuerdo con la cantidad de ciclos ocurridos.

Para los datos de entrada del modelo se utilizan los contadores de rendimiento provistos por la arquitectura de hardware utilizada, los cuales son un conjunto de registros de propósito especial cuya función es almacenar en forma acumulativa ciertos eventos del sistema.

La selección de los contadores se realiza mediante un análisis de correlación individual contra la variable dependiente y, además, una correlación mutua entre cada par de contadores de manera tal de detectar información redundante entre ellos.

Generalmente, para implementar el modelo predictivo se emplean diferentes herramientas de desarrollo como Weka, Matlab, RapidMiner, R Studio, entre otras. En este caso, se utiliza la suite RapidMiner Studio 8.2, constituido como el producto mejor valorado frente a sus competidores por su facilidad de uso, calidad de soporte, rápida administración, entre otras, según la lista de los top 16 software para análisis estadístico y predictivo [28].

4. Desarrollo del modelo de estimación

En este trabajo, el modelo predictivo se construye con el objetivo de estimar el consumo de potencia de la placa de desarrollo Raspberry Pi 3. Es una placa embebida que presenta el microprocesador, la memoria, las comunicaciones inalámbricas y los puertos en un mismo circuito. El modelo utilizado dispone de un procesador Broadcom BCM2837B0, Cortex-A53 (ARMv8) con 4 cores de 64 bits a 1.4GHz. Posee una memoria de 1GB SDRAM y una GPU VideoCore IV a 400 MHz. Además, cuenta con conectividad inalámbrica y cableada, así como también, con un vasto número de pines de E/S de propósito general. La placa utiliza el sistema operativo Raspbian 8 Jessie (kernel 4.4.50), una distribución GNU/Linux basada en Debian.
El procesador ARM Cortex-A53 solo dispone de 6 contadores de rendimiento. Con ellos se puede contabilizar los distintos eventos disponibles en el chip, los cuales se encuentran listados en la Tabla I.

Tabla I. Eventos de rendimiento disponibles en Raspberry Pi 3 (ARM Cortex-A53).

Contador	Descripción
TOT_CYC	Totalidad de ciclos del procesador
TOT_INS	Totalidad de instrucciones completadas
LD_INS	Instrucciones de carga completadas
SR_INS	Instrucciones de almacenamiento completadas
BR_INS	Instrucciones de salto completadas
BR_MSP	Fallos en predicción de saltos
L1_ICM	Fallos en caché de instrucciones L1
L1_DCA	Cantidad de accesos a la caché de datos L1
L1_DCM	Fallos en caché de datos L1
L2_DCA	Cantidad de accesos a la caché de datos L2
L2_DCM	Fallos en caché de datos L2
TLB_IM	Fallos en instrucciones en TLB
TLB_DM	Fallos en datos en TLB
HW_INT	Cantidad de interrupciones de hardware

Para llevar a cabo la recolección de datos, se realizó el conexionado de los diferentes dispositivos involucrados, como se muestra en la Figura 1.

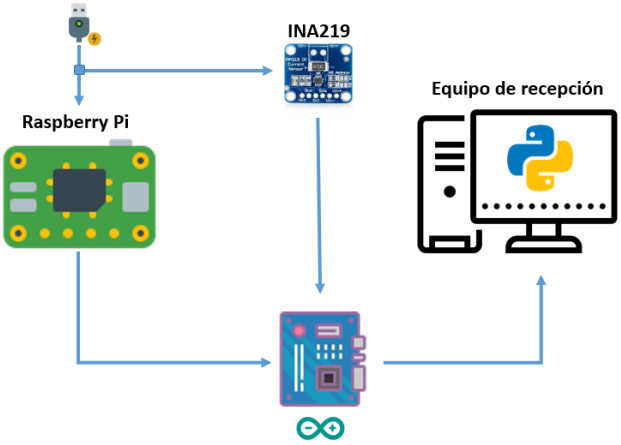

Figura 1. Esquema de conexión.

El sistema de medición está compuesto por una placa Arduino UNO conjunto al sensor INA219. En particular, Arduino UNO es una placa de desarrollo basada en el microcontrolador ATmega328 y muy utilizada para crear dispositivos digitales y objetos interactivos que permiten sensar y manipular diferentes variables del medio en el que se desarrollan. Por otra parte, el dispositivo INA219 permite analizar tensión y corriente continua a través del protocolo I^2C. De esta manera, combinando ambas variables se obtiene la potencia instantánea.

Luego, como se puede apreciar, la Raspberry Pi se alimenta por USB desde la línea eléctrica a través de un transformador convencional. Simultáneamente, el sensor mencionado mide la corriente y la tensión instantánea que consume la placa embebida. En este sentido, al momento de ejecutar cada uno de los benchmarks, la Raspberry Pi genera una señal digital por uno de sus pines de E/S para que la placa Arduino comience a almacenar la información proveniente del sensor INA219. Al mismo tiempo, el equipo de recepción ejecuta un script Python, el cual genera un archivo de muestras con los datos recibidos.

Se analizan las soluciones secuenciales y multihilo (OpenMP) para cada una de las aplicaciones de los benchmarks seleccionados. En primera instancia, se evaluó NAS [23], un conjunto de benchmarks originados por el grupo de investigación NASA Advanced Supercomputing. Además, se analizó RODINIA [24], una suite destinada a computación heterogénea que incluye diferentes aplicaciones según los patrones algorítmicos de Berkeley [25]. Por último, se ejecutaron varias de las aplicaciones que provee la suite LINPACK [26], las cuales permiten medir la potencia de cálculo de punto flotante en el sistema.

En este punto, se lleva a cabo una instrumentación sobre el núcleo de cada uno de los algoritmos, lo que permite supervisar el nivel de rendimiento de una aplicación. Este proceso se realiza a través de PAPI y genera un archivo con la información correspondiente a los contadores de rendimiento evaluados durante la ejecución del algoritmo.

Una vez analizados los diferentes benchmarks, se procede a la carga de los datos de potencia e información de los eventos de rendimiento en la suite RapidMiner. Luego,

se normalizan todos los contadores de acuerdo con el contador de ciclos. Posteriormente, se realiza la correlación de cada contador normalizado con la potencia medida, como se observa en la Figura 2. Además, se analiza la correlación mutua entre contadores. Ambos procesos permiten definir los contadores que dan lugar a la estimación de potencia del sistema. Generalmente, es deseable, que el modelo de estimación involucre el menor número de contadores, de manera de disminuir la cantidad de eventos monitorizados en simultáneo sobre el sistema. Particularmente, TOT_INS_NORM, BR_INS_NORM, SR_INS_NORM, L2_DCA_NORM, L2_DCM_NORM son los contadores seleccionados que, sumados a TOT_CYC (para implementar la normalización), representan las variables de entrada del modelo de regresión lineal.

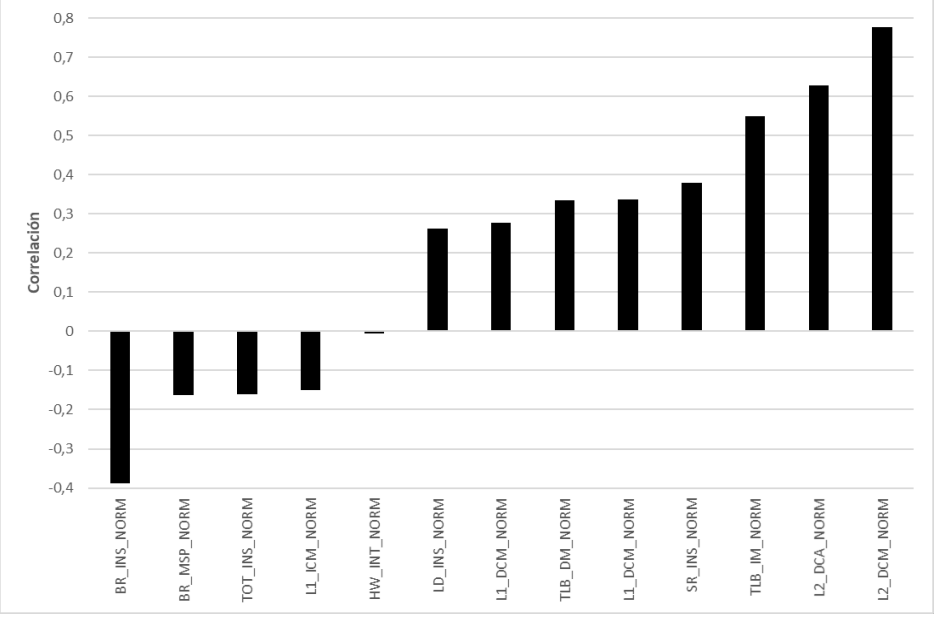

Figura 2. Correlación contador-potencia.

El proceso anterior genera un modelo de regresión lineal que utiliza los contadores seleccionados. Como se pretende modelar tanto soluciones secuenciales como multihilo, se entrenaron dos modelos estadísticos, uno para cada enfoque. Ambos utilizan el mismo set de contadores de rendimiento.

En la Tabla II se muestra el detalle de los pesos asociados a cada contador de rendimiento, según se utilice una solución secuencial u OpenMP.

Tabla II. Pesos obtenidos para cada predictor.

Predictor	Contador	Peso	Algoritmo secuencial	Algoritmo OpenMP
	Intercept	β_0	1.595	1.782
X_1	L2_DCM_NORM	β_1	14.696	79.001
X_2	L2_DCA_NORM	β_2	2.358	8.527
X_3	SR_INS_NORM	β_3	0.108	1.206
X_4	BR_INS_NORM	β_4	0.093	-1.626
X_5	TOT_INS_NORM	β_5	0.058	0.676

De esta manera, ambos modelos de predicción pueden ser descriptos de la siguiente manera:

$$Y_{SEC} = 1.595 + 14.696\,X_1 + 2.358\,X_2 + 0.108\,X_3 + 0.093\,X_4 + 0.058\,X_5$$
$$Y_{OMP} = 1.782 + 79.001\,X_1 + 8.527\,X_2 + 1.206\,X_3 - 1.626\,X_4 + 0.676\,X_5$$

La información reflejada anteriormente puede trasladarse fácilmente a una implementación por software o hardware para permitir que las aplicaciones en ejecución estimen el consumo energético del dispositivo en tiempo real.

5. Validación

Luego de realizado el proceso de entrenamiento para la obtención de los pesos, se contrastaron ambos modelos con los correspondientes valores observados. En este sentido, en la Figura 3 y 4 se detalla la potencia real, obtenida por el proceso de medición, y la potencia estimada, para un conjunto de las aplicaciones analizadas.

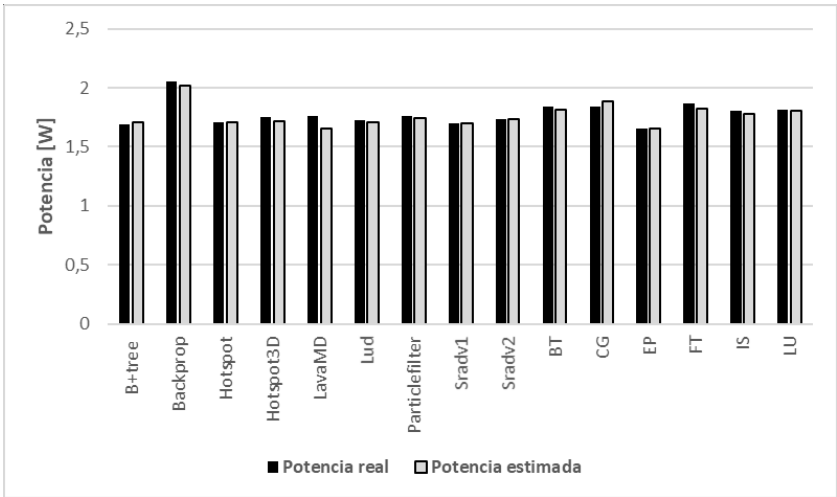

Figura 3. Potencia real vs. estimada para el modelo secuencial.

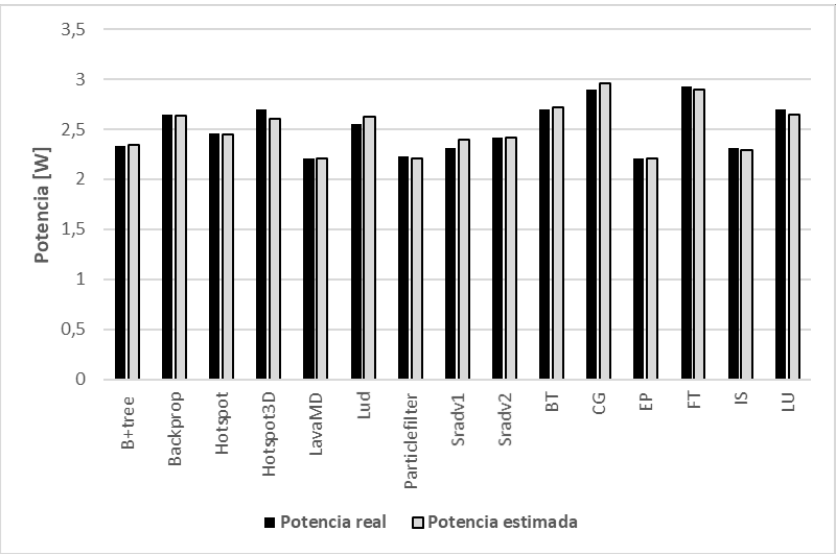

Figura 4. Potencia real vs. estimada para el modelo OpenMP.

Para estimar la precisión del modelo se implementó una validación leave-one-out cross-validation (LOOCV), técnica que permite separar la información de manera tal que, para cada iteración, se destina una única muestra para los datos de prueba mientras que el conjunto restante conforma los datos de entrenamiento del modelo actual.

Como resultado se obtuvo un error promedio de 6,3% para aplicaciones secuenciales y 6,8% para algoritmos OpenMP. Adicionalmente, en pos de clarificar la precisión y robustez de ambos modelos de estimación, se evaluó la raíz del error cuadrático medio (RMSE) [27]. En este caso, el modelo secuencial alcanza un RMSE de 0.033 mientras que la estimación paralela produce un error de 0.042.

Parte del error cometido puede deberse al hecho de que el modelo fue diseñado para funcionar online, razón por la cuál algunos eventos de rendimiento son marginados de la estimación. De esta manera, ciertos comportamientos no habituales de una determinada aplicación generan una desviación mayor entre la predicción y la potencia real.

6. Conclusiones y Líneas de Trabajo Futuro

Este artículo trata la utilización de contadores de rendimiento, específicamente sobre Raspberry Pi 3, con el fin de obtener una estimación confiable del consumo de potencia.

Se construyó un modelo estadístico para aplicaciones secuenciales y otro para aplicaciones paralelas utilizando OpenMP. Ambos basados en regresión lineal y validados frente a diferentes muestras obtenidas por experimentación. En este sentido, los datos generados para los modelos se adquieren a través de la instrumentación de

diferentes benchmarks. Además, la información correspondiente a la potencia real del dispositivo se extrajo a partir de un sistema físico de medición.

Como resultado del proceso de correlación, se seleccionaron los contadores de rendimiento que actúan como predictores en los modelos de regresión. Se restringió la cantidad de predictores seleccionables de acuerdo con la cantidad máxima de contadores de hardware que dispone el dispositivo en estudio. De esta manera, la estimación puede efectuarse en tiempo real.

La validación de ambos modelos estadísticos arrojó un error promedio menor a 6.8% tanto para algoritmos secuenciales como OpenMP.

Finalmente, se plantea como trabajo futuro la creación de un único modelo de predicción que se desligue de la cantidad de hilos empleada por cada aplicación. Además, evaluar la utilización de diferentes esquemas de regresión, los cuales permitirían eventualmente alcanzar errores menores.

Referencias

1. A. Buyuktosunoglu, S. Schuster, D. Brooks, P. Bose, P. Cook, D. Albonesi. An adaptive issue queue for reduced power at high performance. In: Workshop on Power-Aware Computers Systems, (2000).
2. D. Folegnani, A. Gonzalez. Energy-effective issue logic. In: 28th International Symposium on Computer Architecture, (2001).
3. F. D. Rossi, M. Storch, I. de Oliveira, C. A. De Rose. Modeling power consumption for DVFS policies. In: IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1879-1882), (2015).
4. J. S. Lee, K. Skadron, S. W. Chung. Predictive temperature-aware DVFS. IEEE Transactions on Computers (pp. 127-133), (2010).
5. C. Isci, M. Martonosi. Identifying Program Power Phase Behavior Using Power Vectors. In: Proceedings of the 6th Annual IEEE International Workshop on Workload Characterization, (2003).
6. C. Isci, M. Martonosi. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. In: Proceedings of the 36th International Symposium on Microarchitecture, (2003).
7. F. Bellosa. The benefits of event-driven energy accounting in power- sensitive systems. In: Proceedings of 9th ACM SICOPS European Workshop, (2000).
8. R. Joseph, M. Martonosi. Run-time Power Estimation in High Performance Microprocessors. Proceedings of the 2001 international symposium on Low power electronics and Design (ISLPED), (2001).
9. G. Contreras, M. Martonosi, Power prediction for intel XScale processors using performance monitoring unit events. In: Proceeding of ISLPED (pp. 221–226), (2005).
10. W. L. Bircher, L. K. John. Complete system power estimation using processor performance events. IEEE Trans. Comput., vol. 61, no. 4 (pp. 563–577), (2012).
11. K. Singh, M. Bhadauria, S.A. McKee, Real time power estimation and thread scheduling via performance counters. SIGARCH Comput. Archit. News, vol. 37, no. 2 (pp. 46–55), (2009).
12. B. Lee, D. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. SIGOPS Oper. Syst. Rev. (pp. 185–194), (2006).
13. W. Lloyd Bircher, Lizy K. John. Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events. In: Proc. of ISPASS (pp. 158-168), (2007).

14. K. Pusukuri K., D. Vengerov, A. Fedorova. A methodology for developing simple and robust power models using performance monitoring events. In: Proceedings of WIOSCA, (2009).
15. V. Jiménez, F. J. Cazorla, R. Gioiosa, M. Valero, C. Boneti, E. Kursun, C. Cher, C. Isci, A. Buyuktosunoglu, P. Bose. Power and thermal characterization of POWER6 system. In: Proceedings of the 19th international conference on Parallel architectures and compilation techniques (pp. 7-18), (2010).
16. M. Y. Lim, A. Porterfield, R. Fowler. SoftPower: fine-grain power estimations using performance counters. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (pp. 308-311), (2010).
17. C. Lively, X. Wu, V. Taylor, S. Moore, H. C Chang, C. Y. Su, K. Cameron. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. Computer Science-Research and Development (pp. 245-253), (2012).
18. R. Rodrigues, A. Annamalai, I. Koren, S. Kundu. A study on the use of performance counters to estimate power in microprocessors. IEEE Transactions on Circuits and Systems II: Express Briefs (pp. 882-886), (2013).
19. W. Bircher, J. Law, M. Valluri, L. K. John. Effective use of performance monitoring counters for run-time prediction of power. University of Texas at Austin Technical Report TR-041104, (2004).
20. J. M. Paniego, S. Gallo, M. Pi Puig, F. Chichizola, L. De Giusti, J. Balladini. Analysis of RAPL Energy Prediction Accuracy in a Matrix Multiplication Application on Shared Memory. In: Argentine Congress of Computer Science (pp. 37-46). Springer, Cham, (2017).
21. S. Desrochers, C. Paradis, V. M. Weaver. A validation of DRAM RAPL Power Measurements. In: Second International Symposium on Memory Systems (pp. 445-470), (2016).
22. M. Hahnel, B. Dobel, M. Volp, H. Hartigl. Measuring Energy Consumption for Short Code Paths Using RAPL. Technische Universität Dresden. ACM SIGMETRICS Performance Evaluation Review Volume 40. (pp. 13-17), (2012).
23. D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, V. Venkatakrishan, S. K. Weeratunga, H. D. Simon. The NAS parallel benchmarks. The International Journal of Supercomputing Applications, (pp. 63-73), (1991).
24. S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. H. Lee, K. Skadron. Rodinia: A benchmark suite for heterogeneous computing. In: IEEE International Symposium on Workload Characterization (IISWC) (pp. 44-54), (2009).
25. K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, K. A. Yelick. The landscape of parallel computing research: A view from berkeley (Vol. 2). Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, (2006).
26. J. J. Dongarra, P. Luszczek, A. Petitet. The LINPACK benchmark: past, present and future. Concurrency and Computation: practice and experience (pp. 803-820), (2003).
27. T. Chai, R. R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE)?. Geoscientific Model Development Discussions, (pp. 1525-1534), (2014).
28. The Top 16 Predictive Analytics Software, https://www.g2crowd.com/categories/predictive-analytics?utm_content=buffer22c4a&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer#highest_rated.