Modelado de potencia en placas SBC: integración de diferentes generaciones Raspberry Pi

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

[1] Instituto de Investigación en Informática LIDI (CEA-CIC), Universidad Nacional de La Plata, 1900 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. Monitorear la potencia de los procesadores es una tarea importante para definir estrategias que permitan disminuir los gastos de energía en los sistemas informáticos. Hoy en día, los procesadores disponen de un elevado número de contadores que permiten monitorear eventos del sistema tales como uso de CPU, memoria, cache, entre otros. Anteriormente se ha demostrado que es posible predecir el consumo de aplicaciones paralelas a través de estos eventos, pero únicamente para una determinada arquitectura de placas SBC. El presente trabajo analiza la portabilidad de un modelo estadístico de predicción de potencia sobre una nueva generación de placas Raspberry. La experimentación destaca las optimizaciones llevadas a cabo con el objetivo de reducir sistemáticamente el error final de estimación en las arquitecturas analizadas. El modelo final arroja un error promedio de 4.76% sobre ambas placas.

Palabras Claves: Potencia, Raspberry Pi, Contadores de Hardware, Modelado, Regresión estadística.

1 Introducción

Uno de los desafíos principales en el diseño de sistemas es la necesidad de una predicción rápida y precisa del consumo energético. En los últimos años, diversas innovaciones ayudaron a cumplir con este requisito.
La idea subyacente es que el consumo de energía no solo depende de las características del hardware, sino también del uso del software y las características internas del mismo. Por ejemplo, un software más complejo requerirá más ciclos de CPU, o una sola operación enorme de escritura en el disco puede consumir menos energía en lugar de varias operaciones de escritura pequeñas. Por lo general, si una persona es consciente de cuánto está consumiendo, puede encontrar su propia solución adecuada para ahorrar energía en el uso de su dispositivo [1][2].
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 se encuentra disponible frecuentemente.

No obstante, obteniendo información de la ejecución de la aplicación, se puede estimar la energía consumida. Estos datos se recopilan a través de los 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. Mientras que algunos eventos representan actividades con poco impacto en la energía, otros exhiben una alta correlación. Aunque el número de contadores suele ser limitado, los mismos permiten contabilizar una amplia diversidad de eventos.

Una forma de estimar el consumo de energía es generar un modelo estadístico basado en estos contadores. Con el objetivo de obtener una predicción en tiempo real se debe seleccionar un conjunto restringido de eventos que describa la mayor parte de la variación en potencia.

En el trabajo [13] se desarrolló un modelo de estimación de potencia utilizando contadores de hardware, el cual no contemplaba la variación en el número de hilos de la aplicación paralela. Asimismo, se utilizó el modelo estadístico de regresión lineal, el cual es intrínsecamente simple de aplicar, pero puede generar un mayor error en las estimaciones. A su vez, este modelo fue implementado utilizando como base la placa de desarrollo Raspberry Pi 3 modelo B (RPI3B).

En el presente trabajo se analiza el error obtenido usando el modelo desarrollado anteriormente, sobre la placa sucesora Raspberry Pi 3 modelo B+ (RPI3B+). Luego, con el objetivo de mejorar las predicciones, se optimiza el modelo de potencia desarrollado para aplicaciones multihilo con compatibilidad en ambas placas.

El 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 en diferentes arquitecturas CPU y GPU. En la Sección 3 se presenta la obtención de un modelo estadístico que sea compatible con las placas RPI3B y RPI3B+. En la Sección 4 se muestra la validación del modelo utilizando diferentes métodos estadísticos presentes en la herramienta de desarrollo. Finalmente, en la Sección 5 se comentan las conclusiones y trabajos futuros.

2 Trabajos Relacionados

En esta sección se presentan algunas de las investigaciones previas relacionadas con este trabajo. Lee et al. [3] propuso el modelado de regresión como un enfoque eficiente para predecir con precisión el rendimiento y la potencia para diversas aplicaciones que presentan cualquier configuración de microprocesador considerando varios diseños de microarquitectura, abordando como desafío fundamental el costo de simulación para obtener valores correctos en la predicción. Con la aparición de los contadores de hardware, Weaver et al [4] analizó los valores obtenidos de los mismos para comprobar si presentan una alta relación con lo que sucedía dentro de la arquitectura del procesador. A partir de sus resultados pudo observar que es razonable esperar que los contadores reflejen el comportamiento del procesador. Esto permitió que muchos investigadores utilicen herramientas para extraer los valores de rendimiento.

Singh et al [5] desarrolló un modelo de medición en tiempo real del consumo de energía del procesador recopilando la información provista por los contadores.

Por otra parte, Bircher et al. [6][7][8] 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. [9] 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 otra basada en bajo consumo.
Por otro lado, Lively et al. [10] 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.
Las arquitecturas asimétricas de núcleos han surgido recientemente como una alternativa prometedora en un entorno con limitaciones de potencia y térmicas. Por lo general, integran núcleos con diferentes características de potencia y rendimiento, lo que hace que la asignación de cargas de trabajo a los núcleos apropiados sea una tarea desafiante. Pricopi et al [11] presentó un modelo para multi-cores asimétricos en el cual se puede obtener el rendimiento y consumo de energía de las cargas de trabajo asignadas a cada núcleo utilizando los contadores de hardware.
En este último tiempo, con la utilización de las FPGA, O'Neal et al [12] desarrolló modelos predictivos de rendimiento y consumo energético para CPU, GPU y FPGA permitiendo ahorrar costos de simulación.
Con la aparición de las placas de bajo consumo Single Board Computers (SBCs), en [13] se diseñó un modelo estadístico para predecir el consumo energético de aplicaciones ejecutadas sobre la placa RPI3B. La limitación que presenta este modelo es que solo permite predecir el consumo para la ejecución secuencial y con cuatro núcleos, sumado a la desventaja de que se utilizan coeficientes de predicción distintos de acuerdo al número de hilos. El actual artículo es una evolución de este trabajo previo, y se centra en desarrollar un modelo que permita evaluar las aplicaciones paralelas teniendo en cuenta la variación en cuanto al número de núcleos utilizados. Asimismo, el trabajo resalta la necesidad de construir un modelo multi-arquitectura que considere los cambios tecnológicos de nuevas generaciones de placas SBC.

3 Modelado

Debido a que el trabajo se basa en modelos estadísticos, se necesita extraer información de aplicaciones con diferentes comportamientos computacionales, con la finalidad de obtener datos relacionados con el rendimiento y consumo energético para las arquitecturas analizadas.
En particular, se utiliza la metodología empleada en el trabajo previo [13]: utilización de los benchmarks NAS [22] y RODINIA [23], los cuales presentan diferentes comportamientos computacionales; instrumentación y compilación del código fuente de las aplicaciones paralelas; recolección de contadores de performance y muestreo de potencia instantánea; correlación contadores-potencia y correlación mutua entre contadores; entrenamiento del modelo de regresión lineal y validación del mismo a través de la técnica de dejar uno afuera.

Teniendo en cuenta el modelo obtenido en [13], se aplican las modificaciones necesarias de acuerdo a la inclusión de la nueva generación Raspberry, generando un nuevo modelo estadístico que permita disminuir el error final de estimación.
Debido a que RPI3B y RPI3B+ son placas SBC compatibles, se reutiliza el código fuente instrumentado para las distintas aplicaciones.
El modelo se encuentra basado en regresión lineal, motor estadístico que presenta una variable dependiente, una constante de offset y variables predictoras o independientes. De acuerdo a la arquitectura utilizada, las variables independientes utilizadas son cinco, las cuales corresponden a los contadores de rendimiento L2_DCM, L2_DCA, SR_INS, BR_INS y TOT_INS. Además, se incluye el contador TOT_CYC con el objetivo de normalizar los eventos anteriores, obteniendo cinco ratios de performance.
En primer lugar, se estudian las predicciones obtenidas utilizando el modelo anterior sobre la nueva generación de placas Raspberry. Luego se realizan las optimizaciones necesarias para reducir el error y obtener de esta manera una estimación precisa.

3.1 Predicción con el modelo anterior

El modelo generado presenta una constante que se suma a los valores normalizados de los cinco contadores, los cuales tienen un peso asociado. En la Ecuación 1 puede observarse cómo obtener la potencia estimada de las aplicaciones paralelas.

$$Y_i = 1.595 + 14.696\ L2\_DCM + 2.308\ L2\_DCA + \ 0.108\ SR\_INS + 0.093\ BR\_INS + 0.058\ TOT\_INS \quad (1)$$

Para realizar la estimación de potencia sobre la placa RPI3B+, en primer lugar, se efectúa la compilación de todas las aplicaciones instrumentadas. Luego se procede a recolectar la potencia consumida y los contadores de rendimiento en cada aplicación ejecutando con 1, 2 y 4 hilos.
Una vez generada la información, se utilizan los valores obtenidos en cada aplicación paralela de los de los seis contadores usados en el modelo para estimar la potencia en la nueva placa.
La predicción arroja un error porcentual medio de 30% en la RPI3B+, a diferencia del 6.8% obtenido en la RPI3B [13]. Este incremento del error en las nuevas placas se explica por la diferencia arquitectónica entre generaciones, ya que la nueva versión presenta una frecuencia de reloj más elevada.

3.2 Generación del nuevo modelo

En la Figura 1 se puede observar la potencia real consumida por las diferentes aplicaciones paralelas en cada placa. El aumento de potencia a medida que se utilizan más hilos de ejecución es similar en ambas arquitecturas. Las 23 aplicaciones ejecutadas secuencialmente (1 hilo), ubicadas en el primer tercio de la figura, representan el menor consumo de potencia en ambas placas. Por otra parte, la ejecución paralela utilizando 2 y 4 hilos (segundo y tercer tercio de la figura respectivamente) genera un mayor aprovechamiento de los dispositivos y, por lo tanto, un mayor consumo energético.

Para la placa RPI3B, el rango de consumo se ubica entre 2 y 3 watts, mientras que para el modelo RPI3B+, el intervalo comprende 3 y 4 watts aproximadamente. El primer paso para encontrar un modelo de predicción de consumo unificado para ambas placas es estudiar la causa de esta diferencia.

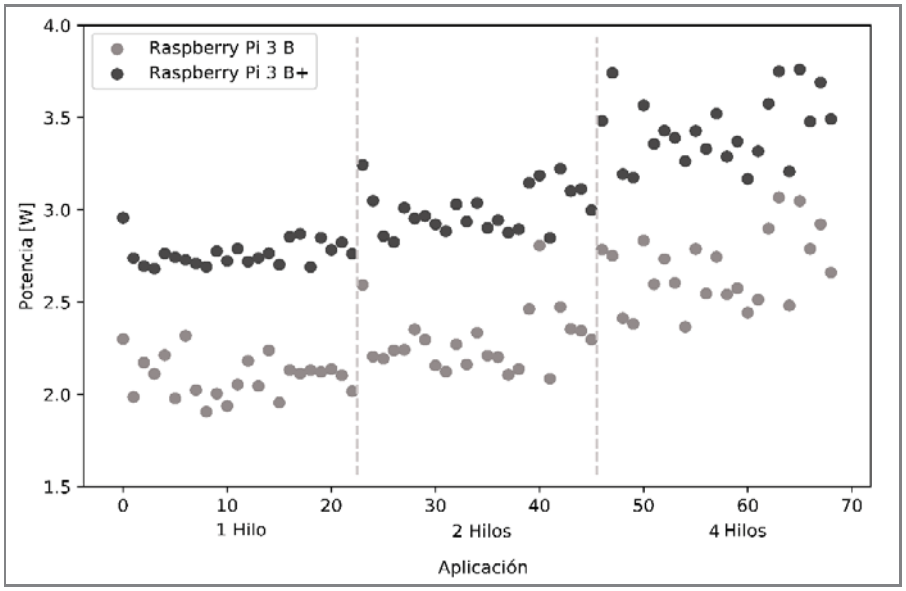

Fig. 1. Potencia real consumida de cada aplicación en cada placa considerando el número de hilos (1 - primer tercio, 2 – segundo tercio, 4 – tercer tercio)

En primer lugar, se analizan los valores que presentan los contadores de rendimiento que alimentan el modelo para cada aplicación. En este caso se elige como ejemplo la aplicación CFD, no obstante, los demás algoritmos presentan el mismo comportamiento en sus contadores. En la Figura 2 se puede observar que todos los eventos muestran la misma tendencia en ambas placas de desarrollo. Al ser placas que presentan varios aspectos arquitectónicos similares (niveles y tamaños de caché, memoria volátil, pipeline de ejecución, entre otras), la ejecución de las mismas aplicaciones arroja resultados semejantes respecto a su comportamiento. Por lo tanto, al no producirse variación en los valores de los contadores de rendimiento entre las placas, el modelo no permite distinguir la ejecución de la aplicación entre ellas.
Posteriormente, se procede a analizar los cambios generacionales entre ambas placas con el objetivo de evaluar su influencia en la predicción. La modificación significativa que se presenta es el aumento en la frecuencia del reloj de 1.2 GHz a 1.4 GHz. Esto genera que la ejecución de las aplicaciones requiera de un mayor consumo energético, lo que puede generar el aumento analizado en la Figura 1. Para comprobar esto se añade al modelo estadístico el predictor correspondiente al valor de la frecuencia máxima al momento de la ejecución. Este cambio disminuyó el error en la estimación, alcanzando el 14 % en promedio.

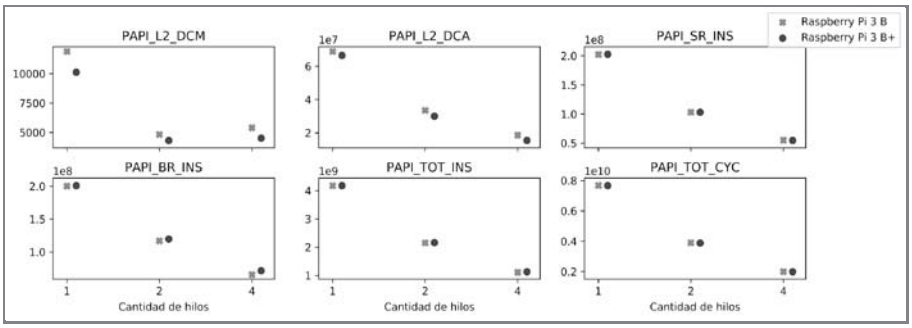

Fig. 2. Valores de contadores de rendimiento en cada placa de desarrollo considerando el número de hilos de ejecución en la aplicación CFD.

En el entrenamiento del modelo original solo se tuvo en cuenta la ejecución de las aplicaciones con 1 y 4 hilos, debido a que en [13] se buscaba evaluar la potencia para ejecuciones secuenciales y paralelas con la máxima cantidad de núcleos disponibles en el procesador. Por lo tanto, para considerar el impacto del nivel de paralelización utilizado en la aplicación, se incorpora el predictor que contempla el número de hilos (núcleos) necesarios para la ejecución ya que las aplicaciones están desarrolladas con la interfaz de programación multihilo de memoria compartida OpenMP [14].
Estas optimizaciones permiten estimar la potencia en ambas versiones de la placa de desarrollo. Por lo tanto, el modelo estadístico final basado en regresión lineal es descrito en la Ecuación 2.

$$Y_i = \beta_0 + \beta_1\, L2\_DCM + \beta_2\, L2\_DCA + \beta_3\, SR\_INS + \beta_4\, BR\_INS + \beta_5\, TOT\_INS + \beta_6\, THREADS + \beta_7\, MAX\_FREQUENCY \quad (2)$$

Para la construcción del modelo estadístico, se sustituyó la herramienta de desarrollo RapidMiner por Python. Esto permite optimizar el tiempo de limpieza de datos y creación del modelo de predicción de acuerdo a la inclusión de librerías embebidas dentro de la herramienta. Asimismo, proporciona diferentes modelos estadísticos con un entrenamiento eficiente. Además, permite personalizar de manera sencilla los parámetros para el entrenamiento de cada modelo. La Tabla 1 muestra los valores de los pesos $\beta_i$una vez entrenado el modelo.

Tabla 1. Pesos obtenidos para cada predictor.

Predictor	Contador	Coeficiente	Valor
-	INTERCEPT	β_0	-2.656
X_1	L2_DCM	β_1	18.437
X_2	L2_DCA	β_2	4.381
X_3	SR_INS	β_3	0.037
X_4	BR_INS	β_4	-0.032
X_5	TOT_INS	β_5	0.169
X_6	THREADS	β_6	0.221
X_7	MAX_FREQUENCY	β_7	3.546

4 Validación del modelo

4.1 Modelo de Regresión Lineal

Luego del entrenamiento del modelo, lo cual resulta en un conjunto de pesos que se aplican sobre los predictores mencionados, se evalúa el resultado del mismo ante los valores reales de potencia observados. De esta manera, en la Figura 3 puede apreciarse las predicciones que se obtuvieron para ambas generaciones Raspberry, junto a la potencia real consumida. Se observan dos tipos de muestras, las representadas en forma circular son las aplicaciones ejecutadas en la placa RPI3B mientras que las triangulares en la RPI3B+.

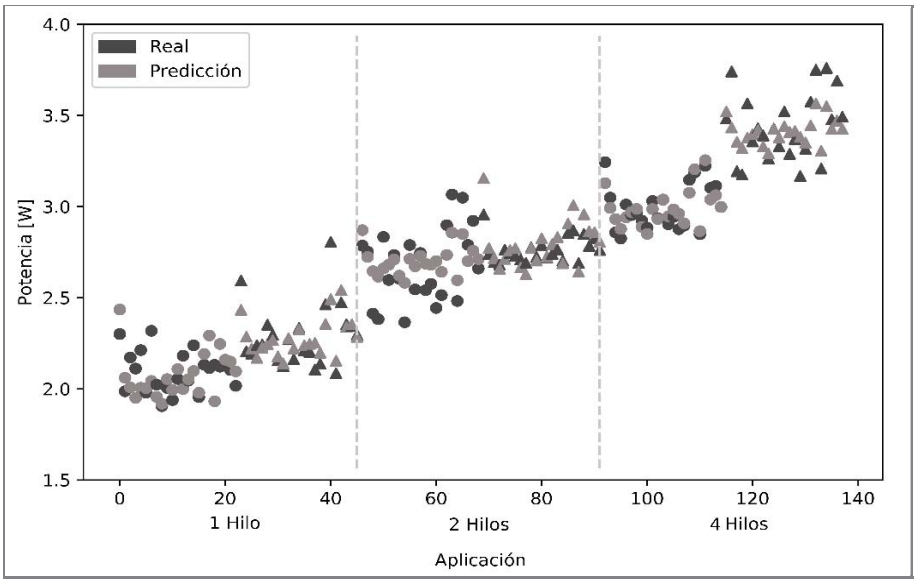

Fig. 3. Predicción y potencia consumida real de cada aplicación en cada placa (círculos - RPI3B, triángulos – RPI3B+) considerando el número de hilos (1 - primer tercio, 2 – segundo tercio, 4 – tercer tercio)

Para estimar la precisión del modelo se implementó una 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. La misma es conocida por el nombre de leave-one-out cross-validation o LOOCV.

En la Figura 4 puede observarse el error para cada iteración de la técnica de validación utilizada. El máximo error de predicción es de 18.46%. La dispersión del error o desviación estándar del conjunto de muestras es de 4.14%. El error promedio de las iteraciones es de 4.76%, el cual se considera aceptable y no genera grandes diferencias entre la potencia real y estimada de la ejecución de una aplicación paralela.

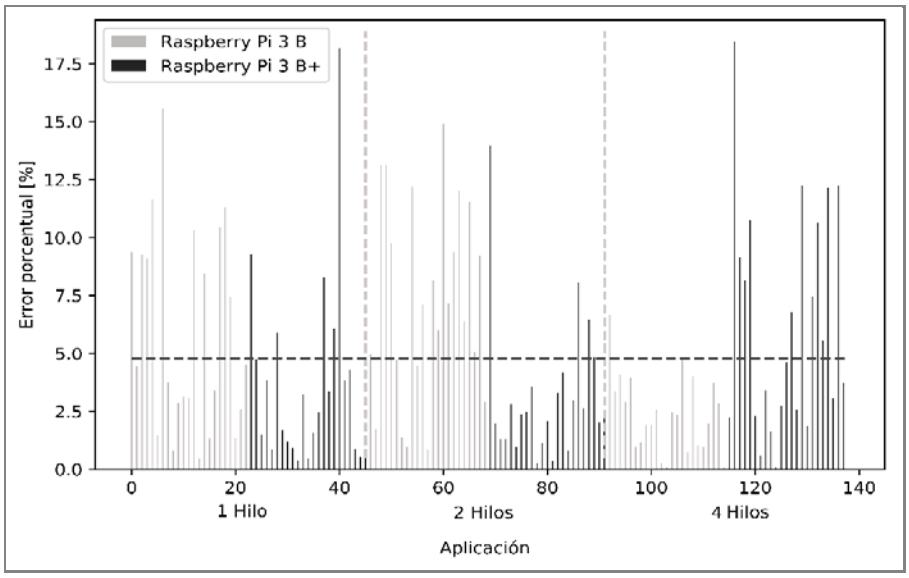

Fig. 4. Porcentaje de error de predicción en las ejecuciones de cada aplicación en cada placa considerando el número de hilos (1 - primer tercio, 2 – segundo tercio, 4 – tercer tercio)

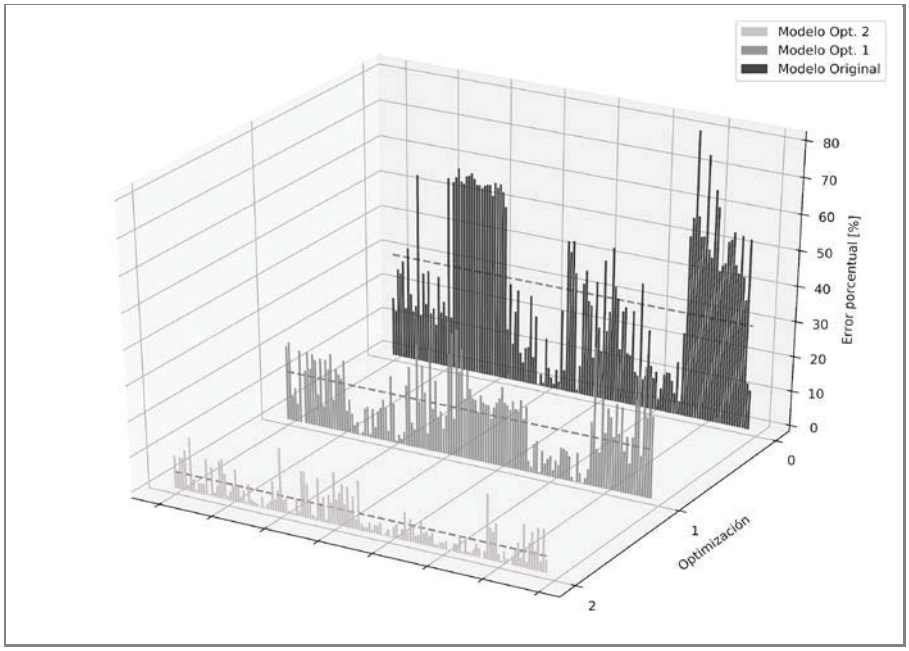

Fig. 5. Comparación de error de predicción de las aplicaciones entre las mejoras propuestas para el modelo estadístico.

Por último, puede observarse en la Figura 5 los errores de predicción para cada modelo evaluado. El eje X enumera cada aplicación analizada, el eje Y representa el modelo utilizado para la ejecución de las aplicaciones y el eje Z corresponde al porcentaje de error en la predicción. El modelo 0 corresponde al desarrollado en [13]. En el modelo 1 se agrega como predictor la frecuencia de reloj. Como última mejora, el modelo 2 adiciona la variable independiente que se corresponde con el número de hilos que utiliza la aplicación para su ejecución paralela.

4.2 Análisis de otros modelos estadísticos

Para minimizar el error de estimación producido por el método estadístico de regresión lineal se estudiaron diferentes modelos presentes en la librería Sklearn [15] de Python. Entre las técnicas analizadas se encuentran SVR (Support Vector Regression) [16], Regresión Gaussiana [17], Regresión Bayesiana [18], Regresión Kernel Ridge [19], Gradient Boosting [20] y Ridge CV [21]. En la Tabla 2 se detallan los errores de estimación obtenidos para cada modelo, en la cual se puede observar que no se lograron mejoras sustanciales en comparación con el modelo de regresión lineal.

Tabla 2. Error de predicción promedio para los modelos complementarios.

Técnica	Error Promedio
SVR	4.14
Regresión Gaussiana	4.38
Regresión Bayesiana	4.83
Regresión Kernel Ridge	4.41
Gradient Boosting	4.51
Ridge CV	4.86

5 Conclusiones y Trabajos Futuros

El presente artículo se basó en la predicción de la potencia consumida por la ejecución de diversas aplicaciones paralelas, utilizando como información de análisis los valores de los contadores de rendimiento presentes en las placas de desarrollo RPI3B y RPI3B+.

Se estudió la posibilidad de añadir a un modelo de potencia previamente construido (compatible únicamente con la placa RPI3B), nuevos predictores que permitan integrar las nuevas generaciones de la placa, así como también contemplar el nivel de paralelismo aplicado a cada algoritmo.

Se diseñó un modelo estadístico de potencia para aplicaciones multihilo basado en regresión lineal y validado a través de muestras obtenidas de las placas analizadas. Se optimizó el modelo previamente creado añadiendo el número de hilos utilizados por la aplicación y la frecuencia máxima de las placas como predictores.

La validación del modelo resultó en un error promedio en la predicción de 4.76%, permitiendo estimar el consumo de potencia de ambas placas con un alto grado de precisión.

Se analizaron diferentes técnicas estadísticas para buscar reducir el error de estimación: SVR, Kernel Ridge, Gaussian y Gradient Boosting. Las mismas disminuyen entre un 0.2% y 0.6% el error obtenido en la estimación, los cuales no producen mejoras significativas en la predicción.
Como línea de trabajo futuro se propone aplicar la metodología desarrollada sobre la nueva placa de desarrollo Raspberry Pi 4. Del mismo modo, se planea implementar una herramienta que recolecte en tiempo real la información necesaria, aplique el modelo estadístico propuesto e informe el consumo de potencia de una determinada aplicación paralela.

Referencias

1. Bekaroo, G., & Santokhee, A. Power consumption of the Raspberry Pi: A comparative analysis. In 2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech) (pp. 361-366). IEEE. (2016)
2. Procaccianti, G., Ardito, L., Vetro, A., Morisio, M., & Eissa, M. Energy efficiency in the ict-profiling power consumption in desktop computer systems. Energy Efficiency-The Innovative Ways for Smart Energy, the Future Towards Modern Utilities/InTech. Helwan: Prof. Moustafa Eissa, 353-372. (2012)
3. Lee, B. C., & Brooks, D. M. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *ACM SIGOPS Operating Systems Review* (Vol. 40, No. 5, pp. 185-194). ACM. (2006)
4. Weaver, V. M., & McKee, S. A. Can hardware performance counters be trusted?. In *2008 IEEE International Symposium on Workload Characterization* (pp. 141-150). IEEE. (2008)
5. Singh, K., Bhadauria, M., & McKee, S. A. Real time power estimation and thread scheduling via performance counters. *ACM SIGARCH Computer Architecture News*, *37*(2), 46-55. (2009)
6. Bircher, W. L., & John, L. K. Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, *61*(4), 563-577. (2011)
7. Bircher, W. L., & John, L. K. Complete system power estimation: A trickle-down approach based on performance events. In *2007 IEEE International Symposium on Performance Analysis of Systems & Software* (pp. 158-168). IEEE. (2007)
8. Bircher, W., Law, J., Valluri, M., & John, L. K. Effective use of performance monitoring counters for run-time prediction of power. *University of Texas at Austin Technical Report TR-041104*, *1*. (2004)
9. Rodrigues, R., Annamalai, A., Koren, I., & Kundu, S. A study on the use of performance counters to estimate power in microprocessors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *60*(12), 882-886. (2013)
10. Lively, C., Wu, X., Taylor, V., Moore, S., Chang, H. C., Su, C. Y., & Cameron, K. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. *Computer Science-Research and Development*, *27*(4), 245-253. (2012)
11. Pricopi, M., Muthukaruppan, T. S., Venkataramani, V., Mitra, T., & Vishin, S. . Power-performance modeling on asymmetric multi-cores. In *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems* (p. 15). IEEE Press. (2013)
12. O'Neal, K., & Brisk, P. Predictive Modeling for CPU, GPU, and FPGA Performance and Power Consumption: A Survey. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (pp. 763-768). IEEE. (2018)

13. Paniego, J. M., Libutti, L., Pi Puig, M., Chichizola, F., De Giusti, L. C., Naiouf, M., & De Giusti, A. E. (2018). Modelado estadístico de potencia usando contadores de rendimiento sobre Raspberry Pi. In XXIV Congreso Argentino de Ciencias de la Computación. (2018)
14. OpenMP, https://www.openmp.org/
15. Scikit-learn, https://scikit-learn.org/stable/
16. Regresion Gaussiana, https://scikit-learn.org/stable/modules/gaussian_process.html
17. Epsilon-Support Vector Regression, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
18. Ridge CV, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html
19. Naive Bayes Classification with Sklearn, https://blog.sicara.com/naive-bayes-classifier-sklearn-python-example-tips-42d100429e44
20. Kernel Ridge Regression, https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html
21. Gradient Boosting Regressor, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html
22. 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).
23. 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).