Factores influyentes en el consumo energético de los sistemas de Computación de Altas Prestaciones basados en CPUs y GPUs

Javier Balladini[1], Federico Uribe[1], Remo Suppi[2], Dolores Rexachs[2], and Emilio Luque[2]

[1] Facultad de Informática, Universidad Nacional del Comahue, Argentina
jballadi@uncoma.edu.ar, fede.uribe83@gmail.com
[2] Departamento de Arquitectura de Computadores y Sistemas Operativos, Universidad Autónoma de Barcelona, España
{remo.suppi, dolores.rexachs@uab.es, emilio.luque}@uab.es

Resumen Actualmente, la energía es un factor limitante para la computación de altas prestaciones (HPC). La reducción del consumo energético se ha vuelto uno de los mayores desafíos en este campo. Nuestra línea de investigación se orienta a construir un sistema de gestión de energía que proporcione diferentes alternativas de ejecución, determinadas por valores predichos de: consumo energético, potencia máxima, y rendimiento. Para esto necesitamos predecir energía y rendimiento. Sin embargo, los modelos existentes de predicción energía o no son adecuados para nuestros requerimientos o son dependientes de la plataforma. Antes de diseñar un nuevo modelo energético (para nuestros sistemas), es necesario estudiar los factores influyentes en el consumo energético y el rendimiento. Así, en el presente trabajo se propone una metodología para evaluar plataformas de HPC basadas en CPUs y GPUs. Los resultados de aplicar nuestra metodología podrían utilizarse para establecer recomendaciones que permitan desarrollar aplicaciones energéticamente eficientes en la plataforma destino.

1. Introducción

La computación de altas prestaciones (HPC, *High Performance Computing*) ha tenido, por décadas, el único objetivo de incrementar la velocidad de procesamiento de las aplicaciones científicas. Así, se ha propiciado la aparición de supercomputadoras cada vez mas grandes, que consumen enormes cantidades de energía eléctrica. Además del daño ecológico, el costo económico es altísimo. Actualmente, el gasto energético producido por las supercomputadoras, durante su tiempo de vida, es similar al costo de adquisición de las mismas. De esta manera, el consumo energético se a vuelto un factor limitante para el HPC [1].

En 2007 se publicó la primera lista del Green500 que clasifica a las 500 supercomputadoras de mayor eficiencia energética del mundo, dando inicio a la nueva era de la supercomputación verde o ecológica. En la última lista publicada por este organismo, en Junio de 2011, de las 10 supercomputadoras más

eficientes energéticamente, 7 están basadas en aceleradoras y 3 en procesadores de propósito general (CPU, *Central Processing Unit*). De las aceleradoras, 3 están basadas en procesadores Cell (PowerXCell 8i) de IBM y las restantes 4 se basan en unidades de procesamiento gráfico (GPU, *Graphics Processing Units*) de NVIDIA o AMD. El uso de una GPU para computo científico e ingenieril de propósito general se denomina GPGPU (*General-Purpose Computing on* GPU).

La eficiencia en el consumo energético no involucra solo al hardware, sino también al software. El software requiere ser diseñado y ejecutado pensando en el consumo energético. En este campo se ubica nuestra línea principal de investigación cuyo objetivo es la propuesta de un sistema de gestión de energía para una plataforma de HPC. El escalado dinámico de voltaje (DVFS, *Dynamic Voltage and Frequency Scaling*) es una técnica estándar para gestionar el consumo energético de un sistema [2]. Se basa en la reducción del voltaje para reducir la energía consumida.

Las tecnología SpeedStep de Intel, y PowerNow! y Cool'n'Quiet de AMD son ejemplos de algoritmos DVFS. Sin embargo, ellos están orientados a aplicaciones interactivas. Así, han surgido estudios destinados a aplicaciones de HPC como [3,4,5], los cuales intentan reducir el consumo energético manteniendo el rendimiento. Teniendo en cuenta que el consumo energético es un factor limitante para el HPC, creemos que este esquema resulta muy rígido ya que impide resignar rendimiento para optar por niveles menores de consumo energético. Además, los sistemas existentes se limitan solo a reducir la energía pero no consideran la potencia máxima, un factor importante que determina la capacidad de la infraestructura eléctrica y del equipo de refrigeración. A mayor potencia, mayor generación de calor que implica un mayor consumo energético en refrigeración, cuyo costo comúnmente alcanza hasta el 40 % del costo de operación de la plataforma paralela [6].

El objetivo de nuestra línea de investigación es la propuesta de un nuevo sistema de gestión energético que proporcione, para una aplicación y plataforma de HPC dada, las diferentes alternativas de ejecución determinadas por valores predichos de: consumo energético, potencia máxima, y rendimiento. Así, es necesario disponer de modelos de predicción del consumo energético, la potencia máxima y el rendimiento para la plataforma destino.

Los contadores de rendimiento (PMCs, *Performance Monitoring Counters*) junto con métodos de regresión son normalmente utilizados para modelar y estimar el consumo energético. Sin embargo, relacionar el consumo energético, cargas de trabajo y PMCs no es una tarea trivial y generalmente se logra mediante estudios de la arquitectura, tal son los casos de [7,8,9,10]. Otros, como [11], intentan ser independientes de la plataforma utilizando modelos estocásticos, pero necesitan retroalimentar su modelo con información de consumo energético en línea, capturada por un dispositivo externo. Sin embargo, mantener un dispositivo externo que permanentemente esté midiendo el consumo energético es viable para una pequeña plataforma pero totalmente inviable para una supercomputadora con un elevado número de nodos de cómputo.

Con el fin de obtener información que nos permita construir un modelo ener-

gético y de rendimiento de nuestras plataformas de HPC, nos propusimos estudiar los factores influyentes en el consumo energético y el rendimiento de las mismas. El estudio de dichos factores es el objetivo del presente trabajo, enfocado en dos tipos de plataformas, una basada en CPUs y otra en GPU. Nuestro trabajo se limita al estudio de microprocesadores multicore (CPUs) y many-core (GPUs) y sus interacciones con la jerarquía de memoria, mientras que las interacciones con los dispositivos de entrada y salida (como almacenamiento en disco y dispositivos de red) están fuera del alcance del mismo. Adicionalmente, estudiamos el comportamiento energético y de rendimiento de la plataforma de CPUs con diferentes cargas de trabajo y frecuencias de reloj. Este último estudio solo se limita a la plataforma de CPUs debido a que actualmente no disponemos de una GPU que permita modificar su frecuencia de reloj.

Creemos que la metodología aquí expuesta facilitará el estudio de los factores influyentes en el consumo energético y de rendimiento de otras arquitecturas. El conocer dichos factores y el comportamiento de los microprocesadores ante diversas cargas de trabajo, permitiría construir aplicaciones de bajo consumo energético. Además, permitiría encontrar explicaciones a, por ejemplo, el impacto del modelo de programación paralela en el consumo energético de los sistemas de HPC [12].

2. Trabajos relacionados

Muchos trabajos estudian evaluaciones energéticas de sistemas, ya sea en GPUs [13], CPUs y Cells [14]. No obstante, no buscan explicación al menor o mayor consumo energético de sus experimentos.

En [10] se presenta un modelo energético y de rendimiento de una GPU basado en [7]. En [7] se presenta un modelo de energía y rendimiento para microprocesadores, que asume a los diferentes componentes arquitecturales como determinantes del consumo energético. El modelo depende de parámetros que deben ser determinados empíricamente y para lo cual no se describe ninguna metodología. Nuestro trabajo es complementario al recientemente citado dado que verifica la validez de asumir que los componentes arquitecturales son determinantes en el consumo.

Para lo mejor de nuestro conocimiento, nadie ha presentado una metodología para estudiar los factores influyentes en la energía y el rendimiento, ni estudian el comportamiento del sistema ante los posibles y diversos tipos de carga de trabajo y frecuencias de reloj.

3. Descripción de las plataformas bajo estudio

3.1. Plataforma basada en GPU

El modelo de computación de GPU es utilizar una CPU y GPUs juntas en un modelo de cómputo de coprocesamiento heterogéneo. La parte secuencial de la aplicación se ejecuta en la CPU y la parte de cómputo intensivo se ejecuta en la

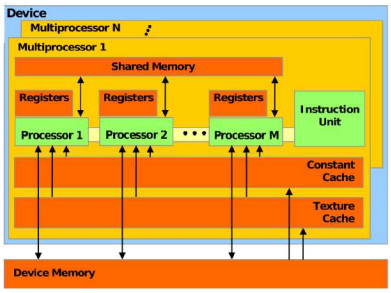

Figura 1. Arquitectura de una GPU

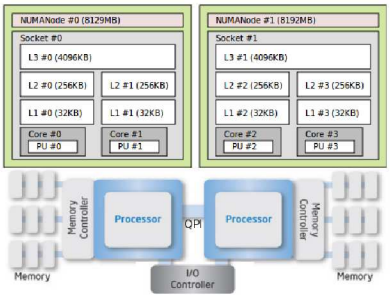

Figura 2. Plataforma de CPUs

GPU. En la figura 1 se muestra la arquitectura de una GPU, la cual tiene N multiprocesadores con M núcleos cada uno. Los núcleos dentro del multiprocesador comparten una única unidad de instrucción. Hay una memoria global (SDRAM) ubicada fuera del chip, dos memorias caché (constant y texture) de solo lectura (conectadas a la memoria global), y cada multiprocesador tiene una memoria compartida (entre sus núcleos). Cada procesador tiene sus propios registros de 32 bits. Las GPU se conectan con el host a través de conexiones PCIExpress.

Nuestra plataforma basada en GPU tiene un procesador dualcore Intel Pentium 4 E7400 con 2GB de memoria principal, y una GPU NVIDIA GeForce 9500GT de 32 núcleos y 1 GB de memoria. Los 32 núcleos están divididos en 4 multiprocesadores.

3.2. Plataforma basada en CPUs

La plataforma paralela basada en CPUs que utilizamos para los experimentos es un servidor Intel de dos sockets modelo SC5650BCDP, con procesadores de dos núcleos Intel Xeon E5502 y 16 GB de memoria principal (8 GB por socket). La figura 2 muestra la arquitectura y jerarquía de memoria de la plataforma. Tiene acceso a memoria no uniforme (NUMA, *Non-Uniform Memory Access*), cada procesador tiene un controlador de memoria integrado, y el sistema de interconexión entre los procesadores es *Intel QuickPath Interconnect* (QPI), una conexión punto a punto de alta velocidad. Las frecuencias de reloj disponibles son: 1,6, 1,73 y 1,86 GHz.

4. Metodología de evaluación de los factores energéticamente influyentes

Para encontrar los factores influyentes en el consumo energético y de rendimiento, es necesario determinar cuáles y qué características tienen los componentes de la arquitectura bajo estudio (en nuestro caso, las expuestas en la

sección 3). Una vez identificados los componentes, se construyen pequeñas aplicaciones sintéticas (*microbenchmarks*) cuyas operaciones evalúen alguna característica especial y puntual de cada componente.

Una vez construidas estas aplicaciones, se ejecutan en la plataforma destino mientras se efectúan las mediciones. Las variables de interés para los experimentos son: el rendimiento, la potencia máxima y la energía. Las mediciones energéticas se realizaron para el nodo completo. Para esto, utilizamos el registrador oscilográfico Yokogawa OR1400 y la sonda de corriente Fluke 80i-110s. El voltaje fue medido directamente por uno de los canales de entrada del registrador, mientras que la corriente del conductor fase fue medida con la pinza Fluke conectada a otro de los canales de entrada del mismo. La tasa de muestreo fue determinada en 10 Hz. Para facilitar las comparaciones entre experimentos, a veces las mediciones se presentan en valores relativos o porcentajes.

A continuación extendemos la metodología expuesta a las dos plataformas evaluadas.

4.1. Metodología de evaluación para GPU

Para construir los diferentes microbenchmarks se utilizaron las APIs (*Application Programming Interface*) de JOCL (Java OpenCL) y JCUDA (Java CUDA) corriendo sobre la máquina virtual Java SE 6. De acuerdo a la arquitectura hardware de la GPU, consideramos desarrollar una serie de microbenchmarks que pueden ser clasificados o agrupados en:

Bus de transferencia Para ejecutar cualquier programa en la GPU se necesita transferir datos de la memoria principal a la memoria de la GPU, y luego a la inversa para retornar los resultados. Así, consideramos importante estudiar el consumo energético de dichas transferencias. Desarrollamos un microbenchmark en JCUDA (por facilidad de implementación) que transfiere un arreglo de 10.000 elementos de 32 bits cada uno (40.000 bytes). El microbenchmark ejecuta 26.000 veces esta transferencia para que la ejecución dure un tiempo considerable que permita capturar el consumo energético.

Operación y tipo de dato Las diferentes operaciones sobre algún tipo de dato tienen distinta complejidad de implementación en hardware. Por lo tanto, se decidió construir microbenchmarks en JOCL para las operaciones de suma, resta, división y multiplicación, para los tipos de datos entero y flotantes, cuyos operandos se obtienen de dos arreglos de 10 elementos y el resultado es almacenado en un tercer arreglo. Las operaciones se aplican 10.000.000 de veces para medir la energía.

Número de núcleos El modelo de ejecución OpenCL de las GPU distribuye un kernel (o programa) a cierto número de núcleos del dispositivo, y cada uno realiza un cálculo individual. Se decidió construir cuatro microbenchmarks, dos de 8 y 32 núcleos limitados en cómputo, y otros dos de 8 y 32 núcleos limitados por memoria.

Dispersión de datos El modelo de memoria de OpenCL define una memoria global que reside en la memoria del dispositivo, la cual es accedida por

transacciones de memoria de 32, 64 o 128 bytes. Es decir, cuando un thread (ejecutando en un núcleo) solicita un dato, la transacción retorna múltiples datos que otros threads podrían necesitar. Este servicio de múltiples peticiones en una sola transacción se llama *coalescing*. Esta propiedad es indicio de que la dispersión de datos es un factor influyente en la energía. Por lo tanto, diseñamos un benchmark con acceso a datos secuencial (acceso *coalescing*) y uno con accesos dispersos o con pérdida de proximidad espacial (acceso no *coalescing*). El benchmark *coalescing* suma dos arreglos de 100 elementos, mientras que el no *coalescing* recibe dos arreglos de 100.000 elementos y suma únicamente los elementos de posiciones múltiplos de 1.000 (ambos suman en total 100 elementos). Los elementos son flotantes y el resultado se retorna en otro arreglo. Las operaciones se aplican 100.000 veces para medir la energía.

4.2. Metodología de evaluación para la plataforma de CPUs

Los microbenchmarks se desarrollaron en lenguaje C. Salvo casos particulares, cada microbenchmark lanza 4 threads, cada uno ejecutando un bucle con 50 Gigaoperaciones. Utilizamos threads con planificación FIFO_SCHED para evitar que los threads sean desalojados de las CPUs y, además, que los datos en memoria sean reemplazados. De acuerdo a las características hardware de la plataforma de CPUs, consideramos desarrollar una serie de microbenchmarks que pueden ser clasificados o agrupados en:

Operación y tipo de dato Las diferentes operaciones sobre algún tipo de dato tienen distinta complejidad de implementación en hardware. Se construyeron microbenchmarks para las operaciones de suma, división y multiplicación, para los tipos de datos entero y flotantes. Los operandos se obtienen de un valor inmediato y un registro.

Jerarquía de memoria El acceso a las distintas memorias (con distinta tecnología y ubicación en la jerarquía) de un microprocesador y el modo de acceso (lectura/escritura) son factores candidatos a influir en el consumo energético. Diseñamos dos tipos de benchmarks. Uno donde los accesos mayoritariamente aciertan en la L1 (mediante accesos secuenciales a un arreglo), y otro cuyos accesos fallan en la L1, L2 y L3, y se terminan resolviendo en la memoria principal. Este último tipo de benchmark se logra mediante accesos a datos espacialmente alejados, a una distancia determinada por el tamaño de la L3. Por cada tipo de benchmark se definieron dos benchmarks, uno con accesos del tipo lectura y el otro de escritura.

Acceso NUMA En esta arquitectura se tiene un único espacio de direcciones, pero los accesos pueden ser locales o foráneos. El acceso foráneo implica utilizar el enlace de interconexión QPI. Por lo tanto, decidimos construir dos benchmarks que escriben datos en memoria, uno accediendo de manera local y el otro foránea.

Eficiencia en el uso de recursos Muchas veces los sistemas no son planificados correctamente y la eficiencia en el uso de recursos disminuye. Así,

decidimos estudiar la influencia en el consumo energético de un benchmark con alta eficiencia y otro con baja eficiencia. El de alta eficiencia ejecuta escrituras en memoria utilizando los 4 núcleos de la plataforma, mientras que el de baja eficiencia lo hace con 1 núcleo.

Frecuencia de reloj La frecuencia de reloj de las CPUs es disminuida cuando se disminuye el voltaje, lo que indudablemente modifica el consumo energético. De esta manera, estudiamos la influencia de la frecuencia de reloj en todos los benchmarks diseñados.

5. Resultados experimentales

5.1. Resultados de evaluación de la plataforma con GPU

La figura 6 muestra la influencia del factor "operación y tipo de dato" en el consumo energético, la potencia máxima y el tiempo de ejecución. Se aprecia que las operaciones sobre enteros consumen menos energía que las operaciones sobre flotantes. El orden de menor a mayor consumo energético es: resta, suma, multiplicación y división. La potencia máxima presenta variaciones inferiores a 4 W.

En la figura 4 se observa la influencia del "número de núcleos" en el consumo energético, la potencia máxima, y el tiempo de ejecución. Para los benchmarks limitados por cómputo, ejecutar en más núcleos consume menos energía, tarda menos tiempo y la potencia máxima es mayor. Para los benchmarks limitados por memoria, el consumo energético, la potencia máxima y el tiempo de ejecución son iguales para el caso de 8 y 32 núcleos. Es decir, si el programa está limitado por memoria, el aumento del número de núcleos no produciría cambios en el comportamiento de los 3 factores estudiados.

La influencia de la "dispersión de datos" se presenta en la figura 5. Se observa que el acceso a elementos alejados presenta un mayor consumo energético y tiempo de ejecución que el acceso a elementos cercanos, mientras que se comporta de forma opuesta en relación a la potencia demandada.

La prueba del "bus de transferencia" tuvo un consumo energético de 0,67 Wh, una potencia máxima de 176 W, y un tiempo de ejecución de 13,94 seg. Además, comprobamos que conviene, en los tres aspectos, realizar pocas transferencias de muchos datos en vez de muchas transferencias de pocos datos.

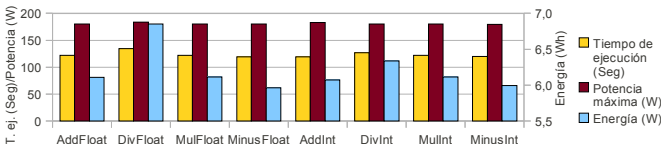

Figura 3. Evaluación de operación y tipo de dato en GPU

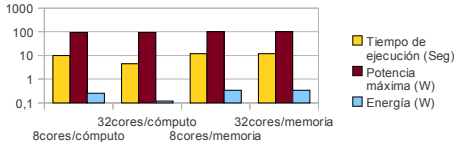

Figura 4. Evaluación del número de núcleos en GPU

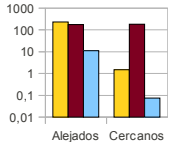

Figura 5. Evaluación de la dispersión de datos en GPU

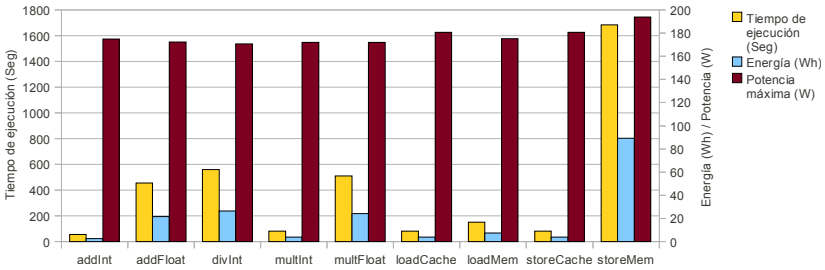

Figura 6. Evaluación de operación y tipo de dato, y jerarquía de memoria en CPUs

5.2. Resultados de evaluación de la plataforma con CPUs

El resultado de evaluar los microbenchmarks de "operación y tipo de dato", y "jerarquía de memoria" a la máxima frecuencia de CPUs son mostrados en la figura 6. Tanto el consumo energético, la potencia máxima y el tiempo de ejecución varían para todos los casos, con lo cual confirmamos la influencia de los factores analizados. En la figura 7 se observa el impacto de ejecutar los microbenchmarks con la menor frecuencia de reloj. Se aprecia una reducción energética clara para la escritura de datos en memoria (storeMem) y en menor medida para la escritura en la caché L1.

En cuanto a la "eficiencia en el uso de recursos" en la figura 8 se observa que el benchmark más eficiente disminuye su consumo energético y el tiempo de ejecución pero aumenta la potencia máxima. Al disminuir la frecuencia del reloj, el benchmark menos eficiente tarda bastante más tiempo en ejecutar, consume más energía y disminuye levemente la potencia máxima. En cambio, el más eficiente tarda solo un poco más, y disminuye drásticamente el consumo de energía y la potencia máxima.

El resultado de la evaluación del factor "acceso NUMA" se muestra en la figura 9, donde se observan las conveniencias en cuanto a consumo energético y de rendimiento de realizar accesos locales. No obstante, hay un pequeño aumento en la potencia máxima. A mayor frecuencia de reloj, mayor es la ventaja en ahorro energético y rendimiento pero mayor es el aumento de la potencia máxima.

En general los aumentos de la potencia se producen cuanto más rápido las operaciones acceden sus operandos. Esto demuestra que el procesador es más

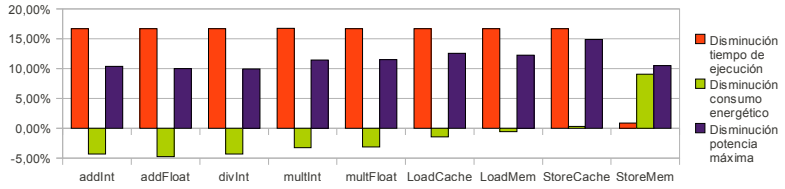

Figura 7. Impacto de la frecuencia de reloj en distintos tipos de operaciones en CPUs

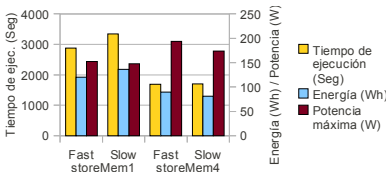

Figura 8. Escritura en memoria con 1 o 4 cores en CPUs

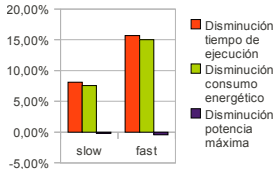

Figura 9. Ventajas por a memoria local en CPUs

demandante de energía que las memorias. Se recomienda evitar accesos no locales (tráfico NUMA) y solo disminuir la frecuencia de reloj en operaciones de escritura (cache o memoria).

6. Conclusiones y trabajo futuro

En este trabajo se presenta una metodología para la evaluación de factores influyentes en el consumo energético de sistemas de computación de altas prestaciones. Para encontrar los factores influyentes en el consumo energético y de rendimiento, la metodología plantea el análisis de las características de los componentes de la arquitectura. Una vez identificados los componentes y sus propiedades, se construyen pequeñas aplicaciones sintéticas (*microbenchmarks*) cuyas operaciones evalúan alguna característica especial y puntual de cada componente. Extendimos la metodología a dos casos prácticos, un sistemas basado en CPUs y otro basado en GPUs. Los resultados aquí presentados serán utilizados como conocimiento para la construcción de un modelo energético y de rendimiento de ambos sistemas de cómputo. Previo a esto, deberemos completar el conjunto de experimentaciones a las aquí presentadas; por ejemplo: estudiar los accesos a L2 y L3 en la plataforma de CPUs, y en GPU, evaluar diferentes agrupaciones de threads y el acceso a memoria compartida y cachés.

Referencias

1. Feng, W.c., Feng, X., Ge, R.: Green supercomputing comes of age. IT Professional **10** (January 2008) 17–23

2. Weiser, M., Welch, B., Demers, A., Shenker, S.: Scheduling for reduced cpu energy. In: Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation. OSDI '94, Berkeley, CA, USA, USENIX Association (1994)
3. Rountree, B., Lownenthal, D.K., de Supinski, B.R., Schulz, M., Freeh, V.W., Bletsch, T.: Adagio: making dvs practical for complex hpc applications. In: Proceedings of the 23rd international conference on Supercomputing. ICS '09, New York, NY, USA, ACM (2009) 460–469
4. Hsu, C.h., Feng, W.c.: A power-aware run-time system for high-performance computing. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing. SC '05, Washington, DC, USA, IEEE Computer Society (2005) 1–9
5. Ge, R., Feng, X., chun Feng, W., Cameron, K.W.: Cpu miser: A performance-directed, run-time system for power-aware clusters. Parallel Processing, International Conference on **0** (2007) 18
6. Cameron, K.W., Ge, R., Feng, X.: High-performance, power-aware distributed computing for scientific applications. Computer **38** (November 2005) 40–47
7. Isci, C., Martonosi, M.: Runtime power monitoring in high-end processors: Methodology and empirical data. In: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture. MICRO 36, Washington, DC, USA, IEEE Computer Society (2003) 93–104
8. Cho, Y., Kim, Y., Park, S., Chang, N.: System-level power estimation using an on-chip bus performance monitoring unit. In: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design. ICCAD '08, Piscataway, NJ, USA, IEEE Press (2008) 149–154
9. Contreras, G.: Power prediction for intel xscale processors using performance monitoring unit events. In: In Proceedings of the International symposium on Low power electronics and design (ISLPED, ACM Press (2005) 221–226
10. Hong, S., Kim, H.: An integrated gpu power and performance model. SIGARCH Comput. Archit. News **38** (June 2010) 280–289
11. Zamani, R., Afsahi, A.: Adaptive estimation and prediction of power and performance inÂ high performance computing. Computer Science - Research and Development **25** (2010) 177–186 10.1007/s00450-010-0125-1.
12. Balladini, J., Grosclaude, E., Hanzich, M., Suppi, R., Rexachs, D., Luque, E.: Incidencia de los modelos de programación paralela y escalado de frecuencia de cpus en el consumo energético de los sistemas de hpc. XVI Congreso Argentino de Ciencias de la Computación (CACIC 2010) (Octubre 2010) 172–181
13. Huang, S., Xiao, S., Feng, W.: On the energy efficiency of graphics processing units for scientific computing. In: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing, Washington, DC, USA, IEEE Computer Society (2009) 1–8
14. Araya-Polo, M., Rubio, F., de la Cruz, R., Hanzich, M., Cela, J.M., Scarpazza, D.P.: 3d seismic imaging through reverse-time migration on homogeneous and heterogeneous multi-core processors. Scientific Programming **17**(1-2) (2009) 185–198