Mejora de la eficiencia energética en sistemas de computación de altas prestaciones*

Belén Casanova[1], Javier Balladini[1], Armando De Giusti[2 4], Remo Suppi[3], Dolores Rexachs[3], Emilio Luque[3]

[1]Departamento de Ingeniería de Computadoras, Universidad Nacional del Comahue
Buenos Aires 1400, 8300 Neuquén, Argentina
mb.casanova.retamal@gmail.com, javier.balladini@fai.uncoma.edu.ar
[2]III LIDI, Facultad de Informática, Universidad Nacional de La Plata
Calle 50 y 120, 1900 La Plata (Buenos Aires), Argentina
degiusti@lidi.info.unlp.edu.ar
[3]Departamento de Arquitectura de Computadores y Sistemas Operativos, Universitat Autònoma de Barcelona
Campus UAB, Edifici Q, 08193 Bellaterra (Barcelona), Spain
{remo.suppi, dolores.rexachs, emilio.luque}@uab.es
[4] CONICET

Resumen En la actualidad, los sistemas de cómputo paralelo de altas prestaciones ofrecen un gran rendimiento, impensado hasta hace pocos años, pero que consumen enormes cantidades de energía eléctrica. La tecnología de escalado dinámico de tensión (DVS, *Dynamic Voltage Scaling*) permite reducir el consumo energético mediante cambios dinámicos de la frecuencia de reloj de las CPUs. Las frecuencias más bajas requieren menos potencia, que lleva a una reducción del calor generado, e indirectamente a un aumento del tiempo medio entre fallos de los componentes del sistema de cómputo, menos energía necesaria para la refrigeración, y la posibilidad de aumentar la densidad de componentes. Normalmente, los algoritmos DVS para sistemas de altas prestaciones buscan reducir el consumo energético prácticamente sin reducir el rendimiento. Sin embargo, muchas veces es deseable llevar el consumo energético a niveles inferiores aunque se pierda rendimiento.
En este artículo, se exponen las ideas para desarrollar un nuevo algoritmo DVS, capaz de mejorar la eficiencia energética y el rendimiento (velocidad) resultante de una ejecución a la menor frecuencia disponible de las CPUs. Los resultados experimentales muestran que, para cierto tipo de aplicaciones, es posible reducir el consumo energético como así también aumentar el rendimiento en comparación a una ejecución a la menor frecuencia de las CPUs.

1. Introducción

En la actualidad, los sistemas de cómputo paralelo de altas prestaciones ofrecen un gran rendimiento, impensado hasta hace pocos años, pero consumen

* Esta investigación es soportada por el MINECO (MICINN) España bajo el contrato TIN2011-24384.

enormes cantidades de energía eléctrica. Por ejemplo, las dos primeras supercomputadoras de la lista del Top500 [12], de junio de 2012, consumen 7,9 y 12,6 megawatts (MW) de potencia pico respectivamente. Se estima que en cuatro años el costo relacionado al gasto energético es equivalente al costo de adquisición del sistema.

La tecnología de escalado dinámico de tensión (*Dynamic Voltage Scaling*, DVS) intenta resolver parte del problema mediante cambios dinámicos de la frecuencia de reloj de las CPUs. Las frecuencias más bajas requieren menos potencia, que lleva a una reducción del calor generado, e indirectamente a un aumento del tiempo medio entre fallos de los componentes del sistema de cómputo, menos energía necesaria para la refrigeración, y la posibilidad de aumentar la densidad de componentes.

Normalmente, los algoritmos DVS para sistemas de Computación de Altas Prestaciones (*High Performance Computing*, HPC) buscan reducir el consumo energético prácticamente sin reducir el rendimiento (generalmente, no más de un 5 %). Sin embargo, muchas veces es deseable llevar el consumo energético a niveles inferiores aunque se pierda rendimiento. Es posible citar las siguientes razones, entre otras. Un déficit energético puede llevar a interrupciones del servicio, por lo tanto debe mantenerse el consumo de energía por debajo de la energía disponible. Además, con el fin de mejorar el factor de carga del sistema, los suministradores de energía a menudo proporcionan electricidad en periodos de baja carga a un costo relativamente bajo. Ellos también pueden ofrecer incentivos, a través de programas de conservación y administración de carga, que favorezca la eliminación o el desplazamiento de los picos de potencia [2].

Los sistemas de cómputo normalmente logran menores potencias y mayor eficiencia energética al ejecutar las aplicaciones a la menor frecuencia de reloj de las CPUs. Nuestro objetivo es desarrollar un sistema DVS capaz de mejorar la eficiencia energética y el rendimiento obtenido (tiempo de ejecución) resultante de una ejecución a la menor frecuencia disponible de las CPUs. En este artículo, exponemos las ideas de este nuevo sistema (aún en fase de desarrollo) y resultados experimentales que muestran que, para cierto tipo de aplicaciones, es posible reducir el consumo energético como así también aumentar el rendimiento en comparación a una ejecución a la menor frecuencia de las CPUs.

Nuestra propuesta se sostiene en los siguientes principios. Muchas aplicaciones paralelas tienen fases que se ejecutan en serie o con un bajo grado de paralelismo, ya sea por una deficiente programación o porque las características intrínsecas de los cálculos a realizar lo impiden. Estos períodos de bajo grado de paralelismo significa que se hace un bajo uso de los recursos de cómputo: las CPUs (o cores). Como se explica a continuación, esta ineficiencia en el uso de los recursos lleva a una baja eficiencia energética (que puede ser mejorada).

Hay dos formas de consumo de energía: dinámica y estática. El consumo dinámico surge de la actividad de los circuitos, mientras que el consumo estático se produce durante el estado de inactividad de los circuitos. A modo de ejemplo, una plataforma Intel Server System SC5650BCDP, con dos procesadores dual core Intel Xeon E5502 y 16GB de memoria principal (8GB por socket) consume

aproximadamente una media de 90W en estado ocioso y puede llegar a 210W en estado activo (sin considerar fases de uso del sistema de almacenamiento secundario). En momentos de baja utilización del sistema, el consumo dinámico disminuye pero el estático se mantiene, es decir, la proporción del consumo estático respecto al dinámico aumenta considerablemente. Debido a que el número de operaciones realizadas por unidad de energía es menor, la eficiencia energética disminuye. Por lo tanto, los períodos de uso ineficiente de recursos deberían ser minimizados para evitar el alto costo de mantener en funcionamiento (consumo estático) un sistema completo del cual se utiliza solo una pequeña porción. Para disminuir la duración de los períodos de baja eficiencia de recursos, un algoritmo DVS podría aumentar la frecuencia de las CPUs ocupadas para intentar una aceleración del cómputo de estas fases.

El presente trabajo se organiza como sigue. En la sección 2 se describen los trabajos relacionados, en la sección 3 se presenta el esquema DVS propuesto, que es validado mediante los experimentos detallados en la sección 4. Finalmente, en la sección 5 se exponen las conclusiones y trabajos futuros.

2. Trabajos relacionados

Un DVS manual normalmente implica obtener un perfil del comportamiento energético de la ejecución de una aplicación en todas las frecuencias de reloj de CPU disponibles. Luego, utilizando este perfil, se selecciona la frecuencia que satisfaga la restricción de rendimiento y energía deseada para ejecutar la aplicación. Algunos ejemplos de este método fueron mostrados en [3,5]. Sin embargo, seleccionar un única frecuencia para todo el programa es un esquema muy limitado que podría no lograr la solución esperada.

Un esquema más sofisticado al DVS manual es analizar el código para observar la estructura de la aplicación, y obtener un perfil de cada parte o fase significativa (en tiempo de ejecución) de la aplicación, tal es el caso de [4]. Sin embargo, puede ser una tarea muy tediosa cuando existe un número elevado de frecuencias disponibles y la estructura del programa es muy compleja.

En [9] se propone un esquema basado en el mismo principio pero implementado dentro de un compilador. Generalmente se requiere el código fuente para ser modificado, y se basa en información obtenida por medio de *profilers*. El problema es que ciertas aplicaciones muestran perfiles diferentes según los datos de entrada, por lo tanto la elección de frecuencias podría no ser la adecuada para este tipo de aplicaciones.

Es normal encontrar sistemas de DVS de tiempo de ejecución en dispositivos móviles como las notebooks. Estos sistemas implementan diferentes políticas seleccionables por el usuario. Por ejemplo, la política "On-demand" ahorra energía disminuyendo la frecuencia cuando el uso de la CPU está por debajo de cierto límite; cuando el uso de la CPU supera ese límite, se aumenta la frecuencia para mejorar el rendimiento. Estos esquemas son efectivos para el uso interactivo que se lleva a cabo en dispositivos móviles, pero fallan al trabajar sobre aplicaciones científicas [7].

Específicamente para HPC, se han propuesto una variedad de trabajos como [8,6,10]. Sin embargo, ellos priorizan el rendimiento de las aplicaciones a la energía. En cambio, nuestro esquema plantea una nueva alternativa cuando la necesidad primordial es disminuir el consumo de energía, mientras el rendimiento pasa a ocupar un segundo lugar.

3. Esquema propuesto de escalado dinámico de frecuencia

Es normal encontrar que las aplicaciones paralelas, al ser ejecutadas a la menor frecuencia de reloj de CPU disponible, produzcan un menor consumo energético del sistema de cómputo [1]. No obstante, esto se produce principalmente en momentos de uso eficiente de los recursos. El uso ineficiente de los recursos es uno de los problemas que afectan notablemente el rendimiento de las aplicaciones de HPC. Pero no solo afectan el rendimiento, también producen un innecesario gasto energético.

Nuestra propuesta esencialmente pretende mejorar la eficiencia energética acelerando los períodos de baja utilización de los cores del sistema y desacelerando los períodos de alta utilización de los mismos. Esto se logra ejecutando las partes ineficientes de una aplicación a la mayor frecuencia disponible, y la parte eficiente a la menor frecuencia disponible. Ejecutar los momentos de bajo uso de recursos a la mayor frecuencia producirá un aumento en la potencia media, sin embargo, al reducir el tiempo de ejecución de estas zonas (en caso de ser posible), se espera que la energía utilizada disminuya.

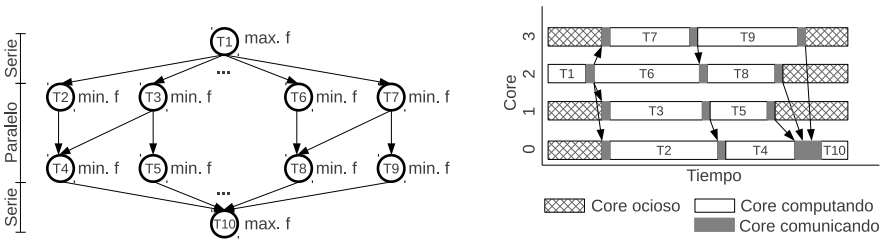

Figura 1. Tareas, asignación a cores, y frecuencias de CPU

A modo de ejemplo, en la figura 1 se muestra una aplicación dividida en tareas, y cómo éstas son asignadas a distintos cores que las ejecutan a distintas frecuencias en diferentes momentos. Para simplificar, la asignación de frecuencias se hizo por tarea, sin embargo, podría realizarse una asignación más fina observando el nivel de actividad de los cores (en el gráfico de la derecha).

A continuación se detallan alternativas de implementación y consideraciones.

3.1. Alternativas y consideraciones de implementación

Para poder implementar nuestro esquema DVS, en primera instancia, se requiere una metodología para distinguir momentos de alto y bajo uso de recursos,

tema en el que principalmente se centra esta sección. Por otra parte, también es necesario definir qué se considera "alto" y "bajo" uso de recursos, es decir, es importante precisar, para cada sistema en particular, la cantidad de cores que deben estar inactivos para establecer si en un determinado momento el sistema está haciendo un alto uso de los recursos (cores) o no. Esto depende tanto de los niveles de consumo energético como del número de cores del sistema. Además, se requiere definir un límite de tiempo inferior a partir del cual sea recomendable realizar el cambio de frecuencia, ya que esta modificación no se produce de forma instantánea sino que añade un retardo en el orden de los microsegundos (dependiente de la arquitectura) [11], como así también incurre en un consumo de energía extra. En síntesis, la variación de frecuencia sólo debe ser realizada para cargas de trabajo tales que la mejora de la eficiencia energética supere los costos de conmutación.

La distinción de momentos de alto y bajo uso de recursos, podría realizarse por medio de dos alternativas: estática (fuera de línea) y dinámica (en línea o tiempo de ejecución). La primera consiste en modificar el código fuente, instrumentándolo en sitios apropiados para que realice los cambios de frecuencia previstos. Esta técnica se caracteriza por una inspección intensiva del código, cuya aplicación requiere conocimientos previos del lenguaje de programación utilizado para paralelizarlo y dificulta, además, la tarea de definir si el cambio de frecuencia resulta o no conveniente. Además, los momentos en que podría cambiarse la frecuencia se verían limitados por la estructura del programa que posiblemente no refleja exactamente la ejecución (por ejemplo, *threads* lanzados por OpenMP para resolver un bucle podrían terminar en diferentes momentos).

La segunda alternativa implica trabajar con recursos brindados por la arquitectura base; en particular, se hace uso de los tipos de estado-C, estados de baja potencia (o estados ociosos), definidos por ACPI (*Advanced Configuration and Power Interface*). La idea básica de los estados-C es que una CPU, cuando está ociosa, puede ahorrar energía de diferentes formas a través de, por ejemplo, la detención de los relojes internos, la reducción de tensión y el apagado de unidades ociosas. Cada estado presenta un equilibrio diferente en términos de ahorro de energía y rendimiento (a causa de la latencia requerida para activar nuevamente al procesador). Esta segunda técnica, haciendo uso de los estados-C, puede, a su vez, plantearse en dos opciones diferentes: realizar una implementación a nivel del núcleo (*kernel*) del sistema operativo, en la que se tomen acciones en el momento exacto en que se realice un cambio de estado-C; o en espacio de usuario, usando también los contadores de rendimiento provistos por hardware (PMC, *Performance Monitoring Counters)*, guardando los datos a modo de información histórica, para procesarlos junto a datos de los estados-C de los demás cores.

En cuanto a la alternativa que utiliza los estados-C, la ACPI establece diferentes estados, el primero *(C-0)* representa el estado activo y los restantes definen diferentes estados de ociosidad (a mayor número, el procesador estará más "profundamente dormido"). Una implementación que trabaje con ésta información, podría aumentar la frecuencia de ciertos cores activos al detectar que una determinada cantidad de ellos están inactivos, y viceversa. Se debe considerar que

cada arquitectura aplica los estados bajo diferentes condiciones, por tanto es necesario definir qué estados-C sirven a los propósitos de nuestro esquema.

Por último, el uso de los PMC nos permite un análisis más minucioso ajustando el intervalo de muestreo. De forma específica, Intel ofrece una herramienta denominada *Performance Counters Monitor* (PCM) basada sobre los PMC, que nos permite consultar contadores como EXEC e IPC entre otros, durante la ejecución de la aplicación a evaluar. El valor de EXEC indica la cantidad instrucciones por ciclo de CPU nominal $\left(\frac{INST\_RETIRED\_ANY}{InvariantTSC}\right)$ e IPC muestra las instrucciones por ciclo de CPU $\left(\frac{INST\_RETIRED\_ANY}{CPU\_CLK\_UNHALTED}\right)$, éste último refleja la cantidad de ciclos activos. Al dividir EXEC por IPC obtenemos un porcentaje de uso del core $\left(\frac{ciclos\_activos}{ciclos\_totales}\right)$ y podemos así determinar, de forma empírica, un umbral sobre el cual un core puede considerarse ocioso.

4. Validación experimental

4.1. Plataforma de evaluación

Evaluamos la plataforma Intel Server System SC5650BCDP, con dos procesadores dual core Intel Xeon E5502 y 16GB de memoria principal (8GB por socket). Es un sistema NUMA (*Non-Uniform Memory Access*), cada procesador tiene un controlador de memoria integrado, y el sistema de interconexión entre los procesadores es el *Intel QuickPath Interconnect* (QPI), que provee enlaces punto a punto de alta velocidad. Las frecuencias de reloj de CPU disponibles son: 1,6, 1,73 y 1,86 GHz. El procesador soporta estados-C por core. La tecnología *Intel Turbo Boost* permite a un core ejecutar por sobre su frecuencia de operación máxima base (si está operando por debajo de los límites de potencia, corriente, y temperatura especificados); sin embargo, los procesadores E5502 no soportan esta tecnología.

4.2. Metodología de medición de potencia

En esta sección explicamos algunas definiciones sobre potencia y energía, y la metodología e instrumentos utilizados para medir la potencia eléctrica de un sistema completo.

Potencia es la tasa a la cual el sistema consume energía eléctrica. El watt (W) es la unidad de potencia real, equivalente a 1 joule por segundo (1 J/s), y es el producto de la corriente por la tensión. Energía es la cantidad total de energía eléctrica que el sistema consume a lo largo del tiempo, y es medida en joules o watt-hora (Wh).

Una vez obtenida la potencia, la energía puede ser calculada integrando la potencia sobre el tiempo. Nosotros medimos el consumo energético del sistema completo. Para esto, utilizamos el osciloscopio PicoScope 2203, la sonda diferencial activa TA041, y la pinza de corriente PP264 60 A AC/DC, todos productos de Pico Technology. Las señales eléctricas capturadas por el osciloscopio de dos

canales son transmitidas en tiempo real a una notebook vía USB. La tensión se mide utilizando la sonda TA041 que se conecta a un canal de entrada del osciloscopio. La corriente del conductor fase se mide usando la pinza de corriente PP264 que se conecta al otro canal de entrada del osciloscopio. Entonces, la potencia se calcula como el producto de las mediciones de tensión y corriente. La tasa de muestreo para los experimentos fue de 1000 Hz.

4.3. Problemas del uso ineficiente de los recursos

Realizamos un experimento para mostrar el alto consumo energético que produce una aplicación que utiliza ineficientemente los recursos. En la figura 2 se muestra el resultado de un experimento con dos aplicaciones que realizan una misma tarea, una aplicación utilizando los 4 cores mientras que la otra utilizando un único core. En la figura se observa que la potencia estática es muy significativa, hasta el punto de ser superior a la dinámica utilizando 1 core. Su efecto se ve en la energía total requerida para ambas versiones, que es mucho mayor (171,5 % más) en la de 1 core. También puede observarse la cantidad de energía estática total utilizada, que en ambos casos es significativa, pero en 1 core es excesiva. Claramente, el bajo uso de los recursos debería tratar de evitarse, sin embargo, muchas veces no es posible.

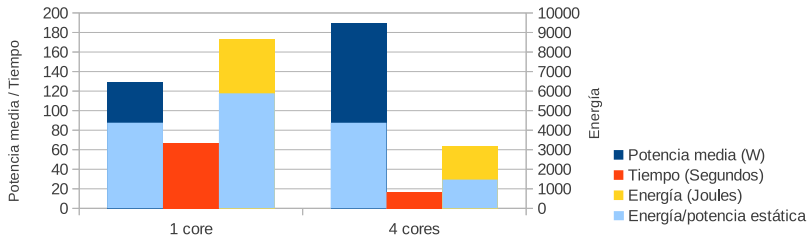

Figura 2. Impacto del uso ineficiente de los recursos en la energía

4.4. La propuesta

Como primer trabajo experimental, tomamos la opción estática de implementación: instrumentación en código fuente. Utilizamos el conjunto de benchmarks ofrecidos por el NAS (*NASA Advanced Supercomputing*), los NPB (*NAS Parallel Benchmarks*); específicamente, utilizamos *MG (Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive)* compilado como clase C, cuyo tamaño de problema puede resolverse con la memoria principal disponible en nuestro sistema. Con el fin de hacer el benchmark útil a nuestros propósitos, fue modificada la subrutina *psinv;* específicamente se eliminó la directiva de paralelismo en OpenMP, de modo que ésta se ejecute en serie. A este benchmark lo denominamos *mgClp (mg class C, Low Parallelism version)*. Introduciendo instrucciones que efectúen los cambios de frecuencias que establece el esquema propuesto, obtuvimos la versión *mgClpDVS*.

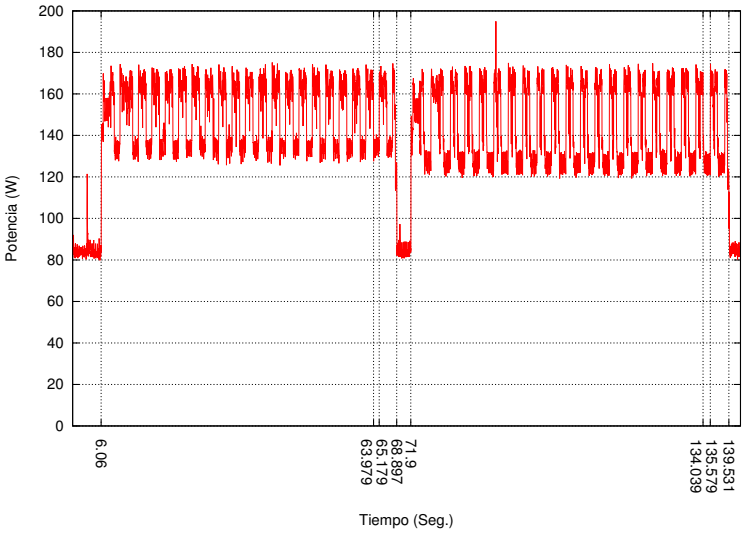

Figura 3. Potencia del benchmark con DVS vs. ejecución a la menor frecuencia

En la figura 3 se observan los valores de potencia medidos a lo largo de la ejecución del benchmark *mgClpDVS*, desde el segundo 6,06 al 68,897, seguido del *mgClp*, del segundo 71,9 al 139,531. Cada valor de potencia corresponde al promedio de la potencia de un ciclo completo de la señal de corriente alterna (en nuestro caso, 20ms por ser una señal de 50Hz). Se puede observar que las partes bajas de la curva son diferentes para ambos benchmarks, y corresponden justamente a la ejecución de la subrutina *psinv*. En especial, se ha marcado una sección dentro del tiempo de duración de cada benchmarks que corresponde a una misma instancia de ejecución de la subrutina *psinv* que analizaremos de manera diferenciada. Para el benchmark *mgClpDVS*, la potencia relacionada a la instancia de *psinv* es claramente superior a la de la instancia de *psinv* del benchmark *mgClp*, efecto que se produce por el aumento de la frecuencia de reloj del core activo (que ejecuta *psinv*).

En el cuadro 1 se detallan los resultados energéticos y tiempos de ejecución correspondientes a los benchmarks *mgClpDVS* y *mgClp*, y las instancias de ejecución de la subrutina *psinv*. Para las ejecuciones completas de los benchmarks se muestra el rendimiento en MFLOPS (*Mega Floating Point Operations Per Second*) que representa el número de operaciones de coma flotante por segundo, y la eficiencia energética en $MFLOPS/WATT$ (rendimiento por watt).

En la figura 4 se muestran los porcentajes de mejora obtenidos utilizando el esquema DVS, tanto para el benchmark completo como para la instancia de ejecución de la subrutina *psinv*. La mejora en los momentos de bajo grado de paralelismo, en que se ejecuta la instancia de la subrutina *psinv*, es notable tanto en eficiencia energética (17,57 %) como en rendimiento (22,08 %), y el impacto

del esquema DVS en el benchmark completo produjo una mejora del 2,63 % en la eficiencia energética y del 7,09 % en el rendimiento.

Benchmark	Potencia media (W)	Energía (Joules)	Tiempo de ejecución (Seg.)	MFLOPS	MFLOPS/W
mgClpDVS	151,03	9.490,64	62,84	2.703,62	17,90
mgClp	144,12	9.747,00	67,63	2.507,04	17,40
mgClpDVS_psinv	134,37	161,37	1,20	-	-
mgClp_psinv	127,04	195,76	1,54	-	-

Cuadro 1. Resultados de la ejecución de los benchmarks completos y de una instancia de ejecución de la subrutina *psinv*

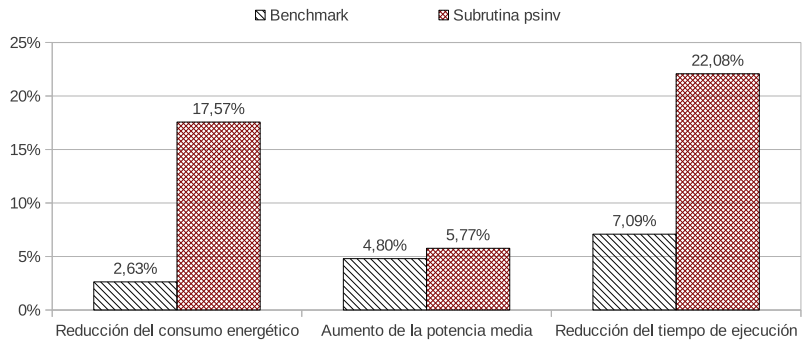

Figura 4. Grado de mejora utilizando el DVS propuesto

5. Conclusiones y trabajos futuros

En los últimos años la computación ecológica se ha vuelto uno de los principales desafíos para la computación de altas prestaciones. Los trabajos actuales sobre escalados dinámico de frecuencia y tensión disminuyen el consumo energético pero siguen priorizando el rendimiento. Sin embargo, en ciertos casos es necesario priorizar el consumo energético al rendimiento. Ante esta situación, normalmente la mejor eficiencia energética se logra ejecutando las aplicaciones paralelas a la menor frecuencia disponible del reloj de las CPUs. En este trabajo se han presentado las ideas para el desarrollo de un nuevo esquema de escalado dinámico de frecuencia y tensión que prioriza la energía y deja en segundo lugar el rendimiento. Además, se han presentado las alternativas y consideraciones para su implementación. Utilizando un benchmark, hemos mostrado que, en aplicaciones con periodos de bajo grado de paralelismo, es posible lograr una mejora de la eficiencia energética y el rendimiento en relación a las misma aplicación ejecutada a la menor frecuencia disponible del reloj de las CPUs.

Como trabajo futuro, nos resta profundizar en las técnicas de implementación de las diferentes alternativas. Especialmente, hemos decidido enfocarnos por una

solución que se ejecute de forma transparente al usuario y que sea independiente de la arquitectura utilizada.

Referencias

1. Balladini, J., Suppi, R., Rexachs, D., Luque, E.: Impact of parallel programming models and cpus clock frequency on energy consumption of hpc systems. Computer Systems and Applications, ACS/IEEE International Conference on 0, 16–21 (2011)
2. Capehart, B.L. (ed.): Encyclopedia of Energy Engineering and Technology. CRC Press (2007)
3. Feng, X., Ge, R., Cameron, K.W.: Power and energy profiling of scientific applications on distributed systems. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01. pp. 34–. IPDPS '05, IEEE Computer Society, Washington, DC, USA (2005), http://dx.doi.org/10.1109/IPDPS.2005.346
4. Freeh, V.W., Lowenthal, D.K.: Using multiple energy gears in mpi programs on a power-scalable cluster. In: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming. pp. 164–173. PPoPP '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1065944.1065967
5. Freeh, V.W., Pan, F., Kappiah, N., Lowenthal, D.K., Springer, R.: Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01. pp. 4.1–. IPDPS '05, IEEE Computer Society, Washington, DC, USA (2005), http://dx.doi.org/10.1109/IPDPS.2005.214
6. Ge, R., Feng, X., Feng, W.c., Cameron, K.W.: Cpu miser: A performance-directed, run-time system for power-aware clusters. In: Proceedings of the 2007 International Conference on Parallel Processing. pp. 18–. ICPP '07, IEEE Computer Society, Washington, DC, USA (2007), http://dx.doi.org/10.1109/ICPP.2007.29
7. Hsu, C.H., Feng, W.C.: Effective dynamic voltage scaling through cpu-boundedness detection. In: Proceedings of the 4th international conference on Power-Aware Computer Systems. pp. 135–149. PACS'04, Springer-Verlag, Berlin, Heidelberg (2005), http://dx.doi.org/10.1007/11574859_10
8. 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. pp. 1–. SC '05, IEEE Computer Society, Washington, DC, USA (2005), http://dx.doi.org/10.1109/SC.2005.3
9. Hsu, C.H., Kremer, U.: The design, implementation, and evaluation of a compiler algorithm for cpu energy reduction. SIGPLAN Not. 38(5), 38–48 (May 2003), http://doi.acm.org/10.1145/780822.781137
10. 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. pp. 460–469. ICS '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1542275.1542340
11. Senger, R., Marsman, E., Carichner, G., Kubba, S., McCorquodale, M., Brown, R.: Low-latency, hdl-synthesizable dynamic clock frequency controller with self-referenced hybrid clocking. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '06) (2006)
12. The TOP500 website (Accedido en 2012), http://www.top500.org/