Incidencia de los modelos de programación paralela y escalado de frecuencia de CPUs en el consumo energético de los sistemas de HPC

Javier Balladini[1], Eduardo Grosclaude[1], Mauricio Hanzich[2], Remo Suppi[3], Dolores Rexachs[3], and Emilio Luque[3]

[1] Departamento de Ingeniería de Computadoras, Universidad Nacional del Comahue, Argentina, {jballadi,oso}@uncoma.edu.ar
[2] Barcelona Supercomputing Center, Barcelona, España, mauricio.hanzich@bsc.es
[3] Departamento de Arquitectura de Computadores y Sistemas Operativos, Universidad Autónoma de Barcelona, España, {remo.suppi, dolores.rexachs@uab.es, emilio.luque}@uab.es

Resumen El consumo energético se ha vuelto uno de los mayores desafíos en el campo de la computación de altas prestaciones (HPC). El costo energético producido por las supercomputadoras, durante el tiempo de vida de la instalación, es similar al de adquisición. Así, además de su incidencia en el medio ambiente, la energía es un factor limitante para el HPC.

Nuestra línea de investigación se orienta a intentar reducir el consumo energético de los sistemas de cómputo paralelo, a través de modificaciones en los algoritmos de las aplicaciones. En este artículo, analizamos el consumo energético de las aplicaciones paralelas, buscando la posible influencia (en el consumo) de los paradigmas de programación paralela de memoria compartida (OpenMP) y paso de mensajes (MPI), y sus variaciones a diferentes niveles de escalado de frecuencia de las CPUs.

Los resultados muestran que el modelo de programación tiene una incidencia importante en el consumo energético de los sistemas de cómputo, y que, reducir la frecuencia de las CPUs no siempre lleva a una reducción en el consumo y hasta puede aumentarlo. Creemos que este estudio puede ser un punto de partida importante para futuros trabajos en el área.

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. Es decir, se han considerado las "prestaciones" como sinónimo de "velocidad". Las supercomputadoras eran diseñadas exclusivamente con la intención de aumentar la cantidad de operaciones de coma flotante por segundo (FLOPS, *floating-point operations per second*). Esto se ve reflejado en la lista del TOP500 [1], que utiliza la métrica FLOPS para determinar el orden de clasificación de las supercomputadoras. Sólo importaban las "prestaciones" y, principalmente para el dueño de la supercomputadora, la relación precio/prestaciones. Así, se ha propiciado la aparición de supercomputadoras cada vez mas grandes, que consumen enormes cantidades de energía eléctrica.

No solo se ha doblado el número de transistores cada 18–24 meses para incrementar las prestaciones de un nodo (ley de Moore), sino que también se ha doblado el consumo energético [2]. Consecuentemente, para garantizar el correcto funcionamiento de las supercomputadoras, se requieren sistemas de refrigeración de dimensiones extravagantes. Según datos del Lawrence Livermore National Laboratory (LLNL), por cada watt (W)

de energía consumido, se gastan 0,7 W de refrigeración para disipar la energía. Dado el tamaño actual de las supercomputadoras, el consumo energético de las mismas es tan elevado que producen un tremendo impacto económico y ecológico. En 2005, el gasto anual de energía eléctrica del LLNL ya era de 14,6 millones de dolares [2]. El Dr. Eric Schmidt, CEO de Google, dijo "lo que más importa a los diseñadores de computadoras de Google no es la velocidad sino el consumo energético, porque los centros de datos pueden consumir tanta electricidad como una ciudad" [3].

Actualmente, los fabricantes no solo piensan en la clasificación del TOP500, sino que también se disputan la posición en el Green500 [4], una lista de las supercomputadoras de mayor eficiencia energética del mundo. Los inicios del Green500 datan del 2005, con una conferencia del Dr. Wu-chun Feng en el "*IEEE IPDPS Workshop on High-Performance, Power-Aware Computing*". En 2006 dos artículos presentados en conferencias, titulados "*Making a Case for a Green500 List*" y "*Global Climate Warming? Yes ... In The Machine Room*" terminaron de impulsar la creación del Green500. Un año más tarde, en 2007, se publicó la primera lista del Green500, dando inicio a la nueva era de la supercomputación verde o ecológica.

El Green500 clasifica a las supercomputadoras según el *rendimiento por watt*, es decir, la tasa de cómputo, en millones de FLOPS (MFLOPS), que el sistema puede proporcionar por cada W consumido [5]. El rendimiento (MFLOPS) considerado es el máximo alcanzado al ejecutar el benchmark Linpack, y la potencia (W) considerada es la media consumida durante la ejecución completa del benchmark.

Dada la enorme relevancia del problema del consumo energético en HPC, nuestro grupo de investigación ha decidido comenzar a dar sus primeros pasos en este tema. Nuestra línea de investigación se orienta a intentar reducir el consumo energético de los sistemas de cómputo paralelo, a través de modificaciones en los algoritmos de las aplicaciones. De esta manera, esperamos contribuir tanto con el factor económico como el ecológico. En este artículo, nos proponemos analizar el consumo energético de las aplicaciones paralelas, buscando la posible influencia (en el consumo) de los paradigmas de programación paralela de memoria compartida y paso de mensajes, y sus variaciones a diferentes niveles de escalado de frecuencia de las CPUs. Específicamente, utilizamos implementaciones en OpenMP [6] y MPI [7] de los benchmark del NAS, ejecutando en dos tipos de plataformas, una notebook con un procesador dual core y un servidor dual socket con procesadores dual core. Los resultados muestran que el modelo de programación cumple un rol importante en el consumo energético de las aplicaciones paralelas, al igual que el escalado de frecuencia de CPUs.

El resto de este artículo se organiza como sigue: en la sección 2 se discute brevemente el estado del arte respecto al consumo energético de los sistemas de HPC. En la sección 3 se analiza el estado actual de los modelos de programación paralela, cuya influencia en el consumo energético de los sistemas computacionales se estudia en la sección 4. Finalmente, en la sección 5, se exponen las conclusiones y trabajos futuros.

2. HPC ecológico

En esta sección se describen las métricas relacionadas a la energía, luego se explican las técnicas de ahorro energético en plataformas de HPC, y finalmente se detallan las características o eficiencia energética de los microprocesadores actuales.

2.1. Consideraciones de métricas de energía

El watt (W) es la unidad de potencia, equivalente a 1 joule sobre segundo (1 J/s). Potencia y energía se confunden fácilmente. Se puede decir que la *potencia* es el ritmo

al que se consume (o genera) la *energía*. Por ejemplo, si una computadora tiene una potencia de consumo de 30W, y la mantenemos encendida durante 15 segundos, se habrá consumido una energía de $15 \times 30/3600 = 0{,}125 Watt - hora(Wh)$.

La métrica más común para medir eficiencia energética es la que utiliza el Green500: rendimiento por watt. Sun Microsystems ha definido otra métrica de eficiencia energética para los centros de datos, en la cual incorpora la medida de espacio. La métrica se llama SWaP (*Space, Wattage and Performance*) y se calcula según la ecuación: $rendimiento/(espacio \times potencia)$. En esta ecuación, *rendimiento* es el obtenido para el benchmark deseado, *espacio* es la altura del servidor expresada en unidades de Rack, y *potencia* es la consumida al ejecutar el benchmark.

En HPC, no solo se requiere un bajo consumo energético sino también disminuir el pico máximo de consumo. Este último requerimiento, por ejemplo, puede determinar la capacidad de la infraestructura eléctrica del sistema de computación y la máxima capacidad del equipo de refrigeración [8].

2.2. Técnicas de ahorro energético en HPC

Existen varias técnicas para ahorrar energía en sistemas de HPC, de las cuales tal vez las más importantes son:

Reducción del consumo de los transistores y cables. Esta técnica se basa en reducir el tamaño de los transistores y afinar los cables para consumir menos energía. Sin embargo, las compañías que desarrollan procesadores han tenido grandes dificultades para obtener beneficios directos por la influencia de otras características físicas de los dispositivos.

Reducción del voltaje. La reducción del voltaje permite reducir la energía consumida. El escalado dinámico de voltaje (DVS, *Dynamic Voltage Scaling*) es una técnica estándar para gestionar el consumo energético de un sistema[9]. Algunos ejemplos son la tecnología SpeedStep de Intel, y PowerNow! y Cool'n'Quiet de AMD. El principio básico que se utiliza para ahorrar energía proviene de la ecuación de potencia (P) [10,11], que es proporcional al cuadrado del voltaje y la frecuencia del reloj: $P \propto fV^2$. Asumiendo que la cantidad de ciclos de reloj para un cierto cálculo es independiente de la frecuencia, el tiempo de ejecución es inversamente proporcional a la frecuencia. Así, la energía total E para realizar el cálculo es proporcional al cuadrado del voltaje: $E \propto V^2$. En este modelo, E no depende de la frecuencia, pero si se quiere reducir el voltaje, también se deberá reducir la frecuencia del reloj. La idea es disminuir el voltaje y la frecuencia del reloj para ahorrar energía, basándose en la utilización de la CPU. Cuando el uso de la CPU baja de cierto límite, el voltaje y la frecuencia de la CPU se disminuyen para ahorrar energía; cuando el uso de la CPU supera cierto límite, se aumenta el voltaje y la frecuencia de la CPU para mejorar el rendimiento. Esto es simple para un uso de aplicaciones interactivas. Sin embargo, para aplicaciones científicas de HPC, resulta muy complejo determinar cuándo realizar una transición entre frecuencias y voltajes de la CPU. Esto se debe a que la transición entre frecuencias y voltajes tarda un tiempo relativamente elevado en comparación a los tiempos ociosos de las CPUs [2].

Corte de energía. Otra metodología es cortar el suministro de energía a los dispositivos cuando ellos no están siendo utilizados. Esta técnica es complementaria al DVS para ahorrar energía durante los momentos ociosos de los dispositivos. Sin embargo, dado que en HPC es normal intentar mantener ciertos niveles de eficiencia computacional eliminando los tiempo ociosos de las unidades de cómputo, la presente técnica resulta difícil de aplicar.

Eliminación del *overhead* de la arquitectura. Con el afán de aumentar el poder de cómputo, muchos procesadores incluyen hardware adicional con su correspondiente

costo energético, aún obteniendo pequeños aumentos en el rendimiento. La eliminación del *overhead* de la arquitectura suele realizarse directamente en hardware, tal es el caso de los procesadores embebidos. Sin embargo, podría realizarse por software si el procesador ofrece mecanismos para desactivar los componentes que no deseamos.

Explotación del paralelismo. Un tiempo atrás, los nuevos procesadores aumentaban su frecuencia para ejecutar las aplicaciones más rápidamente. Este aumento de frecuencia era posible aumentando el voltaje, y por ende el consumo energético. Actualmente, los procesadores en vez de tener un único core rápido, tienen varios cores lentos. Siempre y cuando sea posible obtener un buen *speedup*[4], el intercambio de velocidad de reloj por paralelismo permite ahorrar energía [8].

Cambios en los algoritmos. Es posible ahorrar energía modificando los algoritmos. Por ejemplo, disminuyendo la cantidad de cómputo (lo que también disminuiría el tiempo de ejecución), y reduciendo el uso de los componentes de la arquitectura que consumen mucha energía por otros más eficientes. Esto último está relacionado a la eliminación del *overhead* de la arquitectura expuesto anteriormente.

Es importante mencionar que la reducción del consumo energético no debe ser considerada como un problema que atañe únicamente a los nodos computacionales, sino también a la red y al equipo de refrigeración.

2.3. Eficiencia energética de los microprocesadores actuales

Según datos de la última lista publicada por el Green500, en junio de 2010, las primeras ocho supercomputadoras de mayor eficiencia energética están basadas en aceleradoras. Las aceleradoras son microprocesadores que realizan cálculos a una mayor velocidad que los procesadores tradicionales (CPUs).

Las supercomputadoras basadas en aceleradoras tienen en promedio (según datos publicados por el Green500) una eficiencia de 554 MFLOPS/W, mientras que el resto de las supercomputadoras tienen una media de 181 MFLOPS/W. Estos datos indican que las supercomputadoras basadas en aceleradoras son tres veces más eficientes energéticamente que las que no tienen aceleradoras. A pesar del bajo consumo energético, ellas mantienen un gran poder de cómputo. Por ejemplo, la 4^{ta} de la lista del Green500, la Dawning Nebulae (TC3600 blade CB60-G2 cluster, Intel Xeon 5650 / nVidia C2050, Infiniband) del National Supercomputing Centre in Shenzhen (NSCS), se encuentra 2^{da} en el listado del TOP500.

Hay dos tipos de aceleradoras que utilizan estas supercomputadoras: las basadas en procesadores Cell (PowerXCell 8i) de IBM, y las basadas en unidades de procesamiento gráfico (GPU, *Graphics Processing Unit*), de NVIDIA o AMD. Por un lado, los procesadores Cell son híbridos en el sentido de que tienen un procesador de propósito general y varias pequeñas unidades de procesamiento de código vectorizado en coma flotante. Por otro lado, las GPUs, en vez de utilizarse para el procesamiento de gráficos, son utilizadas por los supercomputadores para realizar cómputo de propósito general.

De las diez supercomputadoras de mayor eficiencia energética, siete están basadas en el PowerXCell 8i y ocupan los tres primeros lugares, y del quinto al séptimo; dos están basadas en GPUs Tesla de NVidia, ocupando el cuarto y octavo lugar; y las dos restantes son CPUs ubicadas en el noveno y décimo lugar.

En [12], un estudio del que participó uno de los autores del presente artículo, se propone una nueva implementación del núcleo de una técnica de imagen acústica sísmica (*Reverse-Time Migration*) utilizada para la prospección petrolera y gasífera. La implementación en un blade QS21 basado en Cell/B.E. mejoró a la del blade JS21 basado en

[4] esta métrica representa el incremento de la velocidad de ejecución de un algoritmo paralelo respecto a su correspondiente algoritmo secuencial.

el procesador multi-core convencional PowerPC 970MP en términos de rendimiento, aumentando el *speedup* en 15×, y una eficiencia energética de MFLOPS/W incrementada en 10×. Además, se ha demostrado que, utilizando códigos aritméticamente intensivos, el Cell/B.E. presenta una eficiencia energética mejorada en un factor de 3 o 4 respecto a otras plataformas de HPC [13].

3. Estado actual de los modelos de programación paralela

Un modelo o paradigma de programación paralela provee, al programador, un marco abstracto que permite expresar un algoritmo paralelo, y efectuar una correspondencia entre componentes lógicos de la aplicación con los componentes físicos de la máquina paralela subyacente. Normalmente, un determinado paradigma de programación resulta en mayores prestaciones que otros para ciertas arquitecturas de sistemas paralelos. Además, estos paradigmas generalmente están orientados a la explotación de un tipo particular de paralelismo. Así, las prestaciones obtenidas al utilizar cierto paradigma de programación, están estrechamente relacionadas al tipo de paralelismo que puede explotarse de la aplicación en cuestión, y a la arquitectura del sistema paralelo donde la misma se ejecute.

Los modelos de programación difieren en la forma de tratar la comunicación y sincronización de procesos, ya sea como memoria compartida o memoria distribuida. En el caso de memoria compartida, los procesos se comunican a través de un espacio común de direcciones de memoria. El estándar más utilizado del *modelo de memoria compartida* es OpenMP. En cambio, en memoria distribuida, se adopta un *modelo de paso de mensajes* entre procesos, donde los datos se mueven desde un espacio de direcciones a otro. Dentro de este último modelo, la implementación más utilizada es MPI.

Se han presentado implementaciones de modelos de programación de memoria compartida que funcionan sobre máquinas de memoria distribuida [14]. Sin embargo, esta "virtualización" tiene un alto costo reflejado en el menor rendimiento de las aplicaciones, siendo el problema de coherencia de cache un factor preponderante de la pérdida de prestaciones. Algo similar ocurre cuando se utiliza MPI en memoria compartida, resultando generalmente en una pérdida de rendimiento. Así, surge el *modelo de programación paralela híbrido*, que es una mezcla de un modelo de paso de mensajes y un modelo de memoria compartida.

Tanto MPI como OpenMP no son adecuados para utilizarse en aceleradoras. El paralelismo interno de la aceleradora suele ser MIMD (*Multiple Instruction Multiple Data*) y SIMD (*Single Instruction Multiple Data*), donde este último no es tratado por MPI ni OpenMP. Además, las GPUs no están diseñadas para ejecutar un sistema operativo, necesitando para ello una CPU. Así, las GPUs agregan un nivel más de paralelismo determinado por la concurrencia entre la CPU y la GPU, tampoco considerado nativamente por MPI y OpenMP. Se han propuestos extensiones a los lenguajes para trabajar con aceleradoras, tal es el caso de la extensión para OpenMP propuesta por [15]. Sin embargo, aún no se encuentra disponible un lenguaje de alto nivel estandarizado para la programación de tales microprocesadores.

Creemos importante estudiar el comportamiento energético de los estándares de los modelos de paso de mensajes y memoria compartida (MPI y OpenMP), en pos de establecer una metodología de uso de los mismos que permita desarrollar programas con mayor eficiencia energética.

4. Experimentación

En esta sección se describen y analizan los experimentos llevados a cabo para verificar la influencia de los modelos de programación paralela en el consumo energético de los sistemas de computación. Para estos experimentos utilizamos los benchmark del NAS, implementados tanto en MPI como OpenMP. Evaluamos dos tipos de sistemas: una *notebook* (Samsung NP-Q310-FA01ES) con un procesador dual-core (Intel Core 2 Duo T5750) y 2GB de memoria principal; y un *servidor* (Intel Server System SC5650BCDP) dual socket con procesadores dual core (Intel Xeon E5502) y 16GB de memoria principal (8GB por socket). Ambos sistemas ejecutando el sistema operativo Linux, OpenMPI 1.4.1 (estándar MPI-2), y la librería de soporte GOMP (GCC OpenMP) 4.4.3 (estándar OpenMP v3.0).

A continuación se describe la metodología utilizada para medir el consumo energético y escalar la frecuencia de las CPUs. Luego analizamos los rangos de consumo energético de las plataformas, y finalmente describimos y comparamos el comportamiento energético y de rendimiento de los benchmarks del NAS en OpenMP y OpenMPI a diferentes frecuencias de CPUs.

4.1. Metodología

Aquí se describe la metodología utilizada para medir el consumo de energía en los dos tipos de sistemas utilizados, y se describe la forma de escalar la frecuencia de las CPUs.

Medición del consumo energético. En una notebook, es muy fácil medir la potencia consumida accediendo los datos energéticos de la batería. El acceso a estos datos se realiza a través de la interfaz avanzada de configuración y energía (ACPI, *Advanced Configuration and Power Interface*). El archivo `/proc/acpi/battery/BAT1/state` ofrece la tasa actual de intensidad (I) en miliamperes (mA) y la tensión (T) en milivoltios (mV). Previa conversión a las unidades de medida adecuadas, y aplicando la siguiente ecuación de potencia:

$$P = I \times T \tag{1}$$

donde I está expresada en amperes (A) y T en voltios, se obtiene la tasa de consumo energético o potencia P, expresada en W.

En un servidor, la medición es más compleja. Esto se debe a que típicamente ellos no disponen de un hardware integrado (como la batería), del cual se puedan obtener datos de consumos energéticos. Así, decidimos utilizar un amperímetro para medir la intensidad de la energía que fluye por el sistema. El amperímetro es conectado en serie, interrumpiendo una de las líneas del circuito de alimentación. Conociendo la intensidad, la tensión (220V), y aplicando la ecuación 1, se obtiene la potencia.

Escalado de frecuencia. Los procesadores modernos permiten escalar la frecuencia de cada core individualmente. El acceso se hace a través de la ACPI. Es posible conocer, para un cierto core, las frecuencias disponibles y la actualmente en uso, respectivamente, a través de los siguientes dos archivos:

`/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies`
`/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq`

Para modificar la frecuencia se puede utilizar el comando `cpufreq-selector` (es necesario tener privilegios de *root*). La ejecución de "`cpufreq-selector -c 0 -f 1000000`" pondría al core 0 en una frecuencia de 1GHz.

4.2. Rangos de consumo energético de las plataformas

Realizamos experimentos para cuantificar la influencia de los algoritmos en el consumo energético de los sistemas de cómputo. Medimos el consumo energético de ambos sistemas computacionales en estado ocioso, es decir, ejecutando únicamente los procesos del sistema operativo. También medimos los picos máximos de potencia de cada plataforma para distintas combinaciones de benchmarks del NAS y frecuencias de CPUs.

El consumo de la notebook fue de 15W en estado ocioso y tuvo un pico de potencia máximo de 42,45W. El servidor consumió 97W en estado ocioso y tuvo un pico máximo de potencia de 185W. Es decir, la notebook, dependiendo del programa que ejecute, puede llegar casi a triplicar el consumo energético del estado ocioso. Mientras, el servidor casi duplica el consumo energético respecto al del estado ocioso. Estos resultados demuestran la alta capacidad del software para alterar el consumo energético de las plataformas, justificando la importancia de su estudio.

4.3. Comparación de OpenMP y MPI a diferentes frecuencias de CPUs

Hemos seleccionado los benchmarks CG, IS y EP, los cuales son limitados en cómputo (*computation bound*), y el benchmark MG que es limitado en comunicación (*communication bound*) [16]. Determinamos cuidadosamente el tamaño del problema de los benchmarks de tal manera que sus requerimientos de memoria principal durante la ejecución sean satisfechos y que no ocurra intercambio (*swapping*) de memoria. Así, optamos por un tamaño de datos clase C, a excepción del benchmark MG clase B utilizado en la notebook por el alto requerimiento de memoria principal del benchmark MG clase C.

Definimos 4 experimentos por benchmark, determinados por la combinación de los dos modelos de programación (OpenMP y MPI), y la máxima y mínima frecuencia de CPUs soportada por cada plataforma. Los resultados de ejecutar cada benchmark en la notebook y el servidor son mostrados en los gráficos de la figuras 1 y 2, respectivamente.

Definimos la eficiencia energética *Ee,* para un experimento i de un cierto benchmark, mediante la fórmula: $Ee_i = e_{max}/e_i$. En esta ecuación, e_i es el consumo energético del experimento i, y e_{max} es el mayor consumo energético que produjo cualquiera de los cuatro experimentos realizados para el benchmark. Similarmente, se define el rendimiento R de un experimento i mediante la fórmula: $R_i = t_{max}/t_i$; donde t_i es el tiempo de ejecución de i y t_{max} es el mayor tiempo de ejecución de los cuatro experimentos realizados para el benchmark en cuestión.

De los resultados experimentales se desprenden las siguientes particularidades para cada arquitectura evaluada:

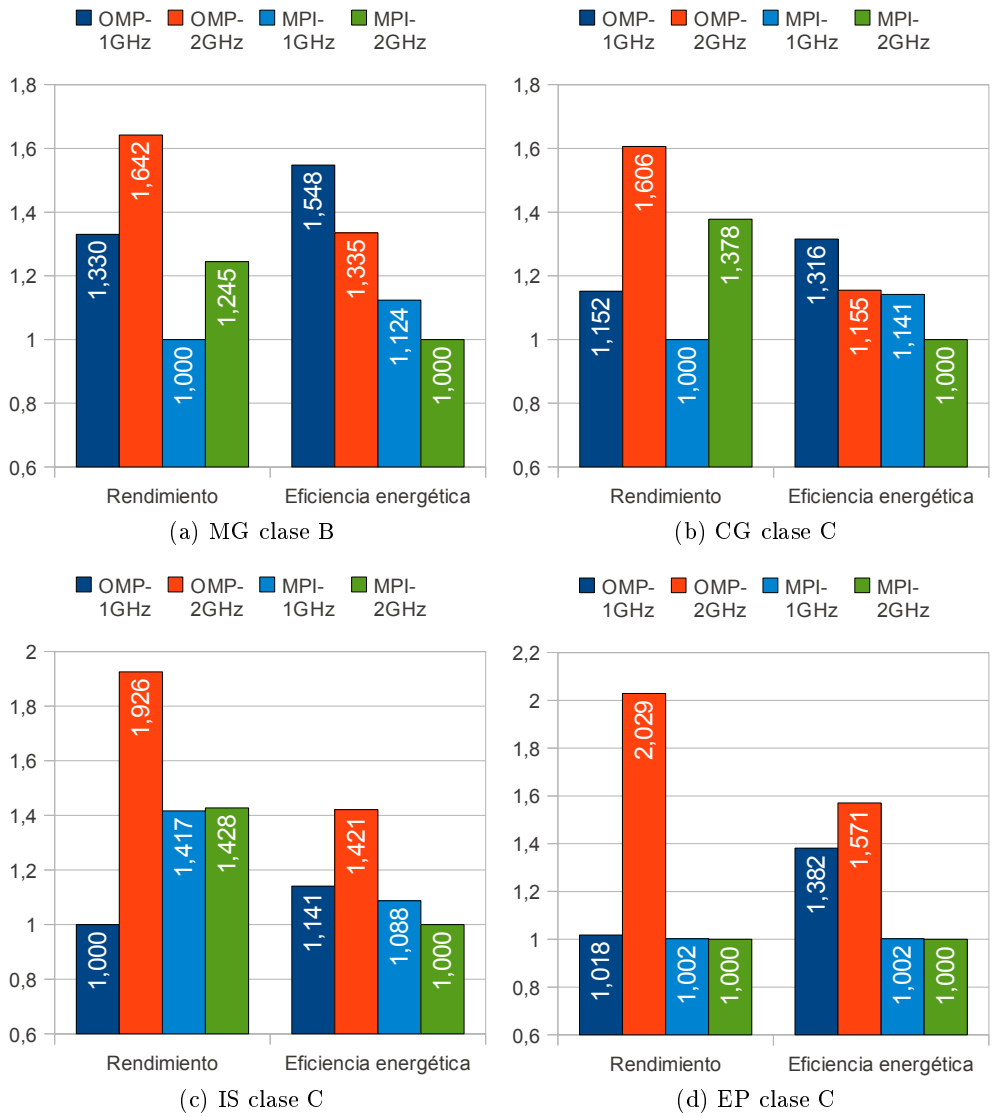
Notebook OpenMP supera en rendimiento y eficiencia energética a OpenMPI. Los benchmarks EP e IS implementados en OpenMP presentan una mayor eficiencia energética para la frecuencia de CPU más alta; es decir, se bajó la frecuencia de CPU pero el consumo energético aumentó.

Servidor OpenMP supera en rendimiento y eficiencia energética a OpenMPI, excepto para el benchmark MG, donde OpenMPI con las CPUs a 1,6GHz resultó ser levemente superior en eficiencia energética. Para cada implementación de los benchmarks (es decir, OpenMP y OpenMPI), bajar la frecuencia de las CPUs resultó en una mayor eficiencia energética. Sin embargo, a veces la ganancia es mínima y la pérdida de rendimiento es muy alta, tal como ocurre con el benchmark EP.

A partir de los resultados expuestos, podemos decir que el modelo de programación tiene una alta influencia en el consumo energético producido por la máquina que ejecuta

la aplicación paralela. El comportamiento de los benchmarks utilizados también fue diferente en cada arquitectura. La reducción de la frecuencia de las CPUs no siempre resultó en una mejora en la eficiencia energética de las aplicaciones y, por el contrario, algunas veces la empeoró.

Es importante mencionar que, idealmente, debería considerarse el consumo energético del sistema de refrigeración en el cálculo de eficiencia energética. Asimismo, las características del equipo de refrigeración podrían influir en la elección (basada en el rendimiento y la eficiencia energética) del algoritmo/aplicación a ejecutar.



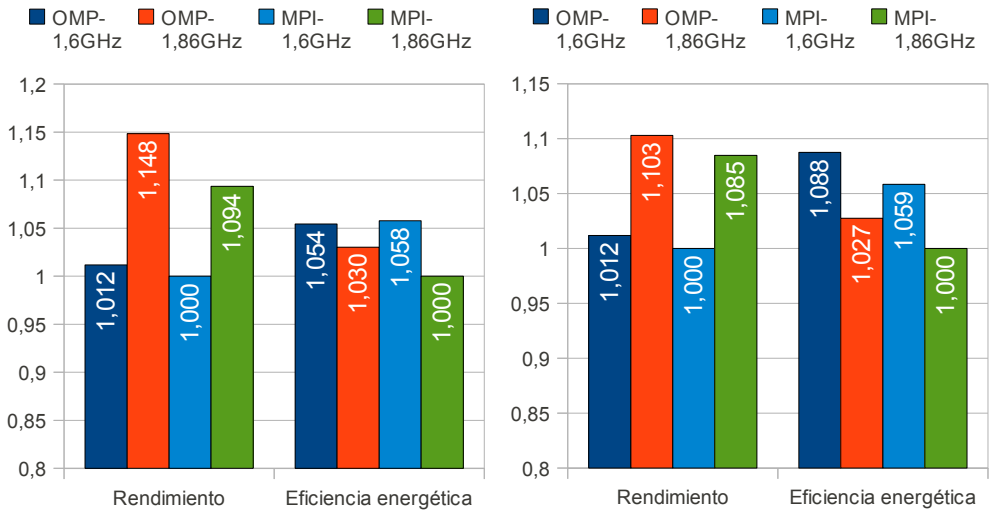
(a) MG clase B

(b) CG clase C

(c) IS clase C

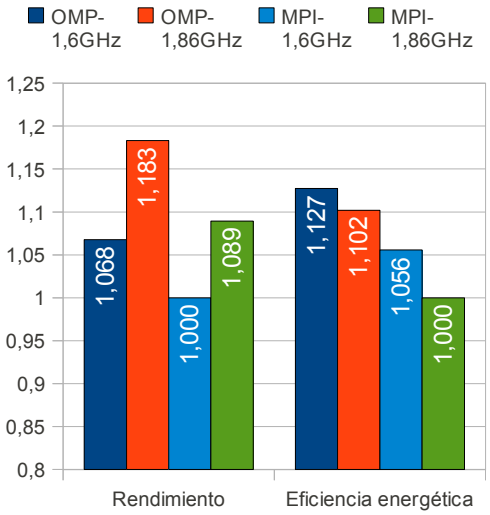
(d) EP clase C

Figura 1: Resultados de la ejecución de los benchmarks del NAS en la notebook

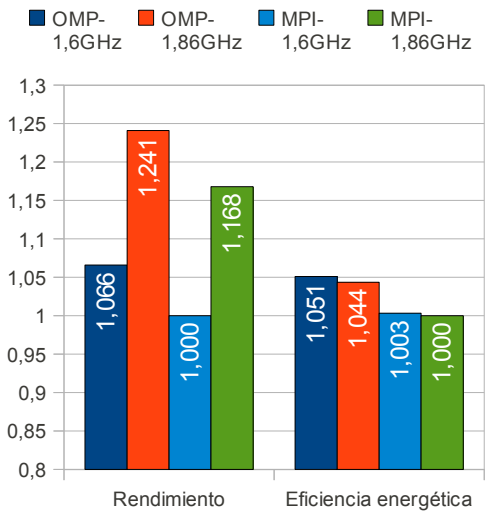


(a) MG clase B

(b) CG clase C



(c) IS clase C



(d) EP clase C

Figura 2: Resultados de la ejecución de los benchmarks del NAS en el servidor

5. Conclusiones y trabajos futuros

Inicialmente mostramos la alta capacidad del software para alterar el consumo energético de los sistemas de computación. Luego analizamos el consumo energético de dos sistemas de cómputo al ejecutar aplicaciones científicas paralelas de HPC. Las aplicaciones seleccionadas fueron benchmarks del NAS. Buscamos evidencia de la influencia, en el consumo energético, de la elección del modelo de programación paralela utilizado para tales aplicaciones. Se evaluó la implementación OpenMP del modelo de memoria compartida, y MPI del modelo de paso de mensajes. También se estudió la incidencia en el consumo energético que se produce al utilizar distintas frecuencias de CPUs.

Los resultados demuestran que el modelo de programación tiene una incidencia importante en el consumo energético de las plataformas de cómputo. La reducción de la frecuencia de las CPUs no siempre lleva a una reducción en el consumo de energía y hasta puede aumentarlo. Este estudio es un punto de partida para futuros trabajos de nuestro grupo en el área del HPC ecológico. Como paso siguiente, tenemos pensado estudiar y relacionar el consumo energético con eventos de las aplicaciones, permitiéndonos avanzar hacia una metodología que nos oriente a construir programas paralelos más "verdes".

Referencias

1. : Sitio web del top500: http://www.top500.org/ (accedido en julio de 2010)
2. Feng, W.C.: The importance of being low power in high-performance computing. Cyberinfrastructure Technology Watch Quarterly (CTWatch Quarterly) **1**(3) (August 2005)
3. Markoff, J., Lohr, S.: Intel's huge bet turns iffy. New York Times (September 29, 2002)
4. : Sitio web de la lista del green500: http://www.green500.org/ (accedido en julio de 2010)
5. : Power measurement tutorial for the green500 list (2007): http://www.green500.org/docs/tutorials/tutorial.pdf (accedido en julio de 2010)
6. : Openmp, http://openmp.org (accedido en julio de 2010)
7. : Mpi forum, http://www.mpi-forum.org (accedido en diciembre de 2009)
8. Feng, W.C.: Low power computing for fleas, mice, and mammoth - do they speak the same language? Cyberinfrastructure Technology Watch Quarterly (CTWatch Quarterly) **1**(3) (August 2005)
9. Weiser, M., Welch, B., Demers, A., Shenker, S.: Scheduling for reduced cpu energy. In: OSDI '94: Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation, Berkeley, CA, USA, USENIX Association (1994) 2
10. Rabaey, J.M., Chandrakasan, A., Nikolic, B.: Digital integrated circuits- A design perspective. 2ed edn. Prentice Hall (2004)
11. Snowdon, D., Ruocco, S., Heiser, G.: Power management and dynamic voltage scaling: Myths and facts. In: Proceedings of the 2005 Workshop on Power Aware Real-time Computing, New Jersey, USA (2005)
12. 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
13. Williams, S., Shalf, J., Oliker, L., Kamil, S., Husbands, P., Yelick, K.: Scientific computing kernels on the cell processor. Int. J. Parallel Program. **35**(3) (2007) 263–298
14. Hoeflinger, J.P.: Extending openmp to clusters. White Paper, Intel Corporation (2006)
15. Ayguade, E., Badia, R.M., Cabrera, D., Duran, A., Gonzalez, M., Igual, F., Jimenez, D., Labarta, J., Martorell, X., Mayo, R., Perez, J.M., Quintana-Ortí, E.S.: A proposal to extend the openmp tasking model for heterogeneous architectures. In: IWOMP '09: Proceedings of the 5th International Workshop on OpenMP, Berlin, Heidelberg, Springer-Verlag (2009) 154–167
16. Jin, H., Hood, R., Chang, J., Djomehri, J., Jespersen, D., Taylor, K.: Characterizing application performance sensitivity to resource contention in multicore architectures. Technical report, NASA Advanced Supercomputing (NAS) Division (2009)