Performance Analisys on MultiCore System Using Papi

Esteban Hernández Barragán[1,2], Josep Jorva Steves[1]

[1]Universidad Oberta de Cataluña, Cataluña, España
[2]Universidad Autónoma de Bucaramanga, Buramanga, Colombia

The rapid growth of multicore systems, and approaches that they have taken, allow complex processes that before were only possible to run on supercomputers, can now be run on low-cost solutions also called "commodity hardware". Such solutions can be implemented using processors consumer market (Intel and AMD). When these solutions scale to scientific computing requirements, is essential to have methods to measure the performance that they offer and how these behave under different workloads. Due to the large number of load types on the market, and even within the scientific computing, it is necessary to introduce "typical workload" that can serve as support in the acquisition and evaluation processes solutions, having a high degree of certainty on this operation. In this research proposes a practical approach to the evaluation and presents the results of tests performed on equipment AMD and Intel multicore architectures

***Index Terms*—Performance, OpenMP, Multicore, SMT, Benchmark,Multicore Programming, PAPI**

I. Introduction

Las arquitecturas multicore han tenido su mayor crecimiento en los últimos 2 años y las tendencias indican que se llegará a algún grado de convergencia entre sistemas Multicore y ManyCore [16]. El número de soluciones que poseen procesadores de tecnología x86 (AMD-x86, Intel-EMT64, IntelCore) en la lista top500.org del mes de Noviembre de 2010 corresponden a cerca del 90% del total de soluciones listadas (Figura No. 6), lo cual indica un cambio de dirección hacia procesadores de consumo masivo con más de 4 cores por sockets (Figura No. 7). Esta tendencia hacia sistemas multicore requiere que dentro del desarrollo de aplicativos se tomen en cuenta nuevos elementos en el momento de medir los niveles de desempeño ofrecidos, especialmente aquellos que afectan de manera directa el trabajo de procesos en paralelo, la distribución de cargas y la sincronización de datos entre los diferentes cores. Estos nuevos desafíos deben ser tenidos en cuenta desde el momento del análisis y diseño, evitando que los cuellos de botella impacten de manera negativa las soluciones productivas. En la evaluación de los sistemas multicore se debe prestar especial atención al tamaño y la jerarquía de los sistemas de caches, a los algoritmos de coherencia entre ellos, la arquitectura de comunicación entre procesadores, memorias y elementos de entrada salida. El presente trabajo propone un modelo de evaluación de los sistemas multicore basados en arquitecturas x86, para aplicaciones que realizan calculo científico intensivo basados en matrices cuadradas o no, mostrando el incremento en desempeño cuando se usa openmp como librería para el desarrollo paralelo y recopilando información mediante el profiling ompp.

II. Trabajos Previos

Se han propuesto metodologías y benchmark en los últimos dos años conducentes a determinar el grado de eficiencia en el uso y la distribución de los caches L1, L2 y L3. Hackenberg y Molka [2] enfocan su análisis en arquitecturas x86-64 evaluando los algoritmos usados en la coherencia de caches y las estrategias de reemplazo de información en los caches de nivel L2 y L3, los resultadosfueron obtenidos aplicando benchmarks específicos de bajo nivel, encargados de mover datos entre la jerarquía de caches y algunos con la memoría principal. Carloza y Pajuelo [1], presentan una nueva metodología llamada FAME para medir el desempeño en procesadores multicore, generalizando cualquier tipo de carga; esta investigación se muestra que la evaluación de sistemas multicore dependiendo los elementos que se midan puede producir conclusiones diversas sobre las mismas arquitecturas, razón por la cual se pueden encontrar resultados donde 2 sistemas comerciales resultan tener el mejor desempeño midiendo el mismo elemento, por esta razón FAME evalúa todas las cargas presentadas (tanto grandes y muy pequeñas) para luego proponer la elaboración de métricas que pueden mostrar el real comportamiento de las cargas de trabajo presentadas.

Hackenberg y Molka [9] también realizan un análisis detallado del comportamiento del nuevo algoritmo de coherencia de caché implementado en la arquitectura Intel Nehalem, tomando como base el modelo de interconexión QPI, el protocolo de coherencia de caché MESIF y el ancho de banda ofrecido. En esta investigación se muestran las ventajas y el incremento en desempeño de esta nueva arquitectura respecto a las anteriores, al mismo tiempo que se realiza una comparación con arquitecturas AMD.

Mallón, Taboada, Tejeiro y otros [3], realizan una comparación del rendimiento de OpenMP, MPI y UPC en sistemas multicore, tomando como base una arquitectura Intel NUMA, conectados a una red de baja latencia Infinibad. En su investigación usan 142 nodos, aplicando benchmark conocidos como NPB Multizone [12], y demuestran que los rendimientos dependen del tipo de operación a realizar; cuando se trata de localidad de datos fuera del mismo nodo, los mejores rendimientos son obtenidos usando MPI, pero cuando se trata

de búsqueda y localización de datos en el mismo nodo, los mejores resultados son obtenidos por OpenMP. Aunque es un trabajo muy interesante, usan equipos que ya no forman parte de la tendencias de los sistemas multicore, además su enfoque de comparación entre acceso a memoria directa local y memoria distribuida resulta más útil para sistemas distribuidos m que para sistemas multicore.

III. Sistemas Evaluados

Para la presente evaluación se eligieron servidores con procesadores Intel Core i7 QuadCore (Nehalem) y procesadores AMD Opteron QuadCore (Shanghai) con las características descritas en el Cuadro No. 1. De especial cuidado es el hecho que los procesadores AMD opteron posean caches de nivel L1 y L2 de mayor tamaño pero de nivel L3 de menor tamaño. Si bien en principio pareciera una desventaja cuando se analiza el esquema de colocación de información (excluyente), se puede observar un mejor aprovechamiento de los espacios de caché evitando duplicidad de los registros, como lo describe ampliamente Conway y Kalyanasundharam[17]

Table 1: Características de las arquitecturas evaluadas

Característica	**Servidor 1**	**Servidor 2**
Procesador	Intel Core i7	AMD Opteron
Micro arquitectura	Nehalem	Barcelona
Frecuencia de Cores	1.6 GHz	1.1 GHZ
Tamaño cache L1	32 KB	64 KB
Tamaño Cache L2	256 KB	512 KB
Tamaño Cache L3	6MB	2MB
Tamaño de Memoria	6 GB	4 GB
Compilador	GCC 4.4.5	GCC 4.4.5
Arquitectura S.O	64 Bits	64 Bits
Threads x Core	2	1
Número de Cores	4	4

IV. Metodologia

En la presente investigación se siguió un enfoque experimental, tomando como base un conjunto de operaciones con matrices cuadradas, haciendo variar su tamaño para que inicialmente fuera igual al tamaño del cache L1 de cada sistema evaluado; el tamaño de la matriz se fue incrementando secuencialmente hasta lograr tamaños que superaban los niveles de cache de segundo nivel (L2) de cada sistema evaluado, observando el grado de éxitos en las búsquedas de acuerdo al tamaño de la matriz. Una vez evaluado este aspecto se realizó modificaciones a el número de threads del procesador, con el fin de medir el impacto que tiene al usar la tecnología SMT (Simultaneus Multithreading) de Intel que hace aparecer cada core físico como 2 "virtuales"; en primer lugar se hizo coincidir el número de threads con los hilos de ejecución nativo de cada procesador (8 para Intel, 4 para AMD)y luego deshabilitando las opciones de SMT en los procesadores Intel. Esn cada escenario se midió el comportamiento de los caches L1 y L2 y se realizó una comparativa de los tiempo de respuesta de las pruebas. Concluida esta etapa se comparó la versión paralela de la prueba con una versión serie de la misma para determinar el grado de aumento en el desempeño que se presentaba usando código paralelo.

Para el trabajo paralelo de la multiplicación de matrices se utilizó las directivas de openmp. Openmp ofrece excepcional rendimiento cuando se trata de acceso a la memoria local como se observa en [10] y [3]. Para la toma de las estadística de los contadores se utilizó la librería PAPI 4.2 [5], que exponen de una manera clara la información sobre los contadores disponibles en Linux kernels 2.6.31 o superiores con soporte a perf_events. Dada la complejidad de tomar datos usando directamente la API y ante la dificil que resulta distinguir los tiempos de ejecución involucrados (tiempo del contador, tiempo de inicio del evento, tiempo de sistema, tiempo de aplicativo), se utilizó el profiler ompp [6]. Este profiler permite revisar el comportamiento de cada hilo en particular, establecer los contadores mediante PAPI y además distinguir los diferentes tiempos que se presentan en la prueba. Para cada ejercicio se realizaron 10 corridas con 8 matrices de diferentes tamaños, 20% de ellas eran de menor o igual tamaño que los caches L1 y el 80% restante de mayor tamaño. Los contadores de hardware que se tuvieron en cuenta corresponde a los presentados en la Cuadro No 2, para los niveles de caché L1 y L2.

Table 2: Contadores de hardware evaluados

Contador PAPI	***Elemento Medido***
PAPI_LX_DCA	Cantidad Total de accesos al caché
PAPI_LX_DCH	Cantidad de aciertos en caché
PAPI_LX_DCM	Cantidad de faltas en el caché

Dentro del proceso de ejecución se realizaron los ajustes de las zonas paralelas utilizadas en openmp evitando que fuera necesario recrear hilos de ejecución, de tal manera que solo se utilizó una región paralela con 4 for internos aplicando agendamiento estático con grupos de 10 hilos. Este tipo de agendamiento fue utilizado debido al comportamiento predecible de las operaciones realizadas (multiplicación de matrices cuadradas), logrando una carga equitativa en las tareas ejecutadas por cada uno de los hilos lanzados. La inicialización de cada matriz se realiza utilizando for paralelos también y en un for paralelo independiente; aunque existen esquemas de agendamiento más dinámicos, estos deben ser usados en entornos donde el comportamiento de la aplicación no es predecible y se hace necesario que los grupos de hilos sean recalculados en tiempo de ejecución, lo que agrega una sobrecarga considerable en los tiempo de ejecución del proceso. Las matrices a,b,c se declaran compartidas para que puedan ser usadas por cada pool de hilos indistintamente y su valor pueda ser compartido entre todos ellos; los índices para

el recorrido de las matrices se declararon privados para asegurarse que cada ciclo de llenado se cumple totalmente.

```
#1Sección de código paralelo
```

```
#pragma omp parallel shared(a,b,c,nthread
s,chunk)
            private(tid,i,j,k)
{
/*** Inicializando matrices ***/
/*** inicializando la primera matriz ***/
#pragma omp for schedule (static, chunk)
  for (i = 0; i < NRA; i++)
      for (j = 0; j < NCA; j++)
             a[i][j] = i + j;
/*** inicializando la segunda matriz ***/
#pragma omp for schedule (static, chunk)
for (i = 0; i < NCA; i++)
     for (j = 0; j < NCB; j++) b[i][j] = i
+ j;
/*** inicializando la matriz resultado **
*/
#pragma omp for schedule (static, chunk)
for (i = 0; i < NRA; i++)
    for (j = 0; j < NCB; j++) c[i][j] = 0;

/***Operando la matriz resultado ***/
#pragma omp for schedule (static, chunk)
for (i = 0; i < NRA; i++) {
     for (j = 0; j < NCB; j++)
          for (k = 0; k < NCA; k++)
               c[i][j] += a[i][k] * b[k]
[j];
   }
} /***Fin de la region paralela ***/
```

A.

V. Resultados

Una vez realizadas las pruebas se puede observar que el tamaño del caché L1 y L2 de las arquitecturas AMD Figuras No. 1 y Figura N3 , influye de una forma determinante en la cantidad de aciertos que se logra obtener al operar matrices que no logran colocarse en su totalidad en el cache L1. La Figura No. 1, muestra que aún con tamaños de matrices que superan el tamaño de caché L2, el porcentaje de acierto sigue siendo superior en las arquitecturas AMD (Figura No. 1). Aunque este comportamiento es el esperado, también se puede observar que el mayor número de thread de proceso de la arquitectura Intel, afectan la efectividad de uso de los caches, de tal manera que al eliminar las características SMT del procesador y dejar un hilo de procesamiento por core en procesadores Intel, el porcentaje de acierto aumenta hasta en un 5% en matrices que no superan en tamaño el nivel de cache L2, como se observa en la Figura No. 2.
La razón para que el nivel de acierto por fuera de los niveles de caché L1 y L2, son explicados a fondo por [2] y especialmente en los métodos de localización de información dentro de los caches (inclusivo y no inclusivo) y los algoritmos de coherencia de los mismos.
Al medir el tiempo de procesamiento y cómo este se relacionaba con la cantidad de threads de proceso, se puede comprobar que al utilizar sistemas SMT(llamados en Intel Hyperthreading o HT), aunque se logra un mayor nivel de ocupación del procesador y un mayor nivel de paralelismo, el tiempo total de proceso aumenta, debido a la sobrecarga de supone para el procesador el manejo de hilos de procesamiento y el agendamiento de los mismos, como se observa en la figura No. 4 y Figura No. 5. El uso de más de un thread por core puede castigar el tiempo de procesamiento hasta en 10% adicional.
Al comparar el incremento de rendimiento de la versión serial del algoritmo contra la versión paralela, se observa que las mejoras en desempeño se notan en su mayoría con matrices de tamaño superior al caché de nivel L2, mientras que en las matrices pequeñas que estén cercanas al tamaño del caché L1, la diferencia en tiempo de procesamiento es mínima (inferior al 1%) lo cual comparado con uso de matrices grandes superiores a 1MB, resulta en una diferencia que puede ser superior al 50%, como se observa en la figura No. 2.

Conclusiones y Trabajos Futuros

Uno de los factores más importantes al construir programas de cálculo científico que serán ejecutados en arquitecturas de procesadores multicore, es la manera como se construyen los tamaños de las matrices. Si estas pueden ser colocada en el caché local, especialmente de nivel L1 y L2, los tiempos de respuesta aumentarán considerablemente debido al uso de localidad privada. En las arquitecturas Intel y AMD los sistemas multicore mantienen esquemas de cache privados para los niveles L1 y L2 y dadas sus prestaciones de velocidad de acceso y cercanía con los registros del procesador pueden hacer la diferencia para lograr un rendimiento superior.

Los programadores y diseñadores de software que tengan en mente operaciones con matrices de gran tamaño, debe diseñar estrategias de subdivisión de las mismas o particionamiento tendientes a lograr que estas al ser operadas puedan entrar dentro de los tamaños de caché específicos, evitando acceder a información que está localizada en un cache compartido (como es el caso del L3) o que se encuentre dentro de un caché de core distinto.

La utilización de esquemas de SMT especialmente el Hyperthreading de Intel afecta de manera negativa el tiempo de ejecución, cuando se operan matrices que superan los tamaños del caché de nivel 2, originando sobrecargas producidas en la coordinación de los hilos de ejecución emulados. Las pruebas realizadas demuestran que al aumentar los thread de ejecución por encima del número de cores el tiempo de respuesta se ve afectado de una manera considerable (hasta en un 7%), lo cual proyectado a soluciones donde se tiene que operar matrices que GigaBytes afectarán de manera fuerte el rendimiento global de la aplicación.
Se deben diseñar benchmark específicos que simulen cargas con matrices tridimensional semejantes las realizadas en NAS

Parallel Benchmark [7], que permitirán evaluar la eficiencia en los algoritmos de coherencia de caches y de colocación de reemplazos en los niveles compartidos. Los profiling existentes deben ser adaptados para que dichas cargas puedan ser analizadas rigurosamente para determinar sus cuellos de botellas y secciones objeto de optimización.

El contadores de hardware presenta un gran reto para el analista de desempeño, sobre todo cuando se trata de arquitecturas de procesadores no X86 comparadas con arquitecturas X86, por lo tanto se requiere un proceso de estandarización de un conjunto de eventos base que deberán ser implementados en todas las arquitecturas de procesadores e implementadas en los sistemas que se ajustan a POSIX. Tales APIS debe tener un muy bajo costo para su invocación. Cada una de las iniciativas existentes como PAPI y perf poseen elementos importantes que deben ser implementados o ajustadas en las nuevas soluciones planteadas [19]

Adicional al presente trabajo se debe adicionar mediciones utilizando otros esquemas de paralelismo como es el caso de Intel TBB [8] y experimentos con las nuevas arquitecturas basadas en GPU como CUDA y OpenCL, para observar el comportamiento de gran cantidad de hilos corriendo fuera de la Procesador Principal y copiando registros desde un procesador tipo GPU.

References

[1] Francisco J. Cazorla, Alex Pajuelo, Oliverio J. Santana, Enrique Fernández, Mateo Valero, On the Problem of Evaluating the Performance of Multiprogrammed Workloads, IEEE Transactions on Computers, vol. 59, no. 12, pp. 1722-1728, Feb. 2010, doi:10.1109/TC.2010.62

[2] Hackenberg Daniel, Molka Daniel, E. Nagel Wolfgang. "Comparing Cache Architectures and Coherency Protocols on x86-64 Multicore SMP Systems",Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium. Jan. 2010.

[3] Mallon, A Damian,Taboada Guillermo, Teijeiro Carlos, Tourino Juan. "Performance Evaluation of MPI, UPC and OpenMP on Multicore Architectures",Lecture Notes in Computer Science. vol 5759, pp. 174-184 2009. .doi: 10.1007/978-3-642-03770-2_24

[4] Dan Terpstra , Heike Jagode , Haihang You , Jack Dongarra.Collecting Performance Data with PAPI-C. Tools for high performance computing, 2010, 157-173, DOI: 10.1007/978-3-642-11261-4_1.

[5] Karl Fuerlinger.The OpenMP Profiler ompP: User Guide and Manual Version 0.7.0, January 2010. University of California at Berkeley

[6] Jin, H.; Van der Wijngaart, R.F.; Performance Characteristics of the Multi-Zone NAS Parallel Benchmarks. NASA Advanced Supercomputing Division, M/S T27A-1, NASA Ames Research Center,

[7] Jaime Randers. Intel Threading Building Blocks. O'Reilly. 2007.

[8] Hackenberg Daniel, Molka Daniel, Robert Schöne y Matthias S. Müller "Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System ", International Conference on Parallel Architectures and Compilation Techniques. Technische Universität Dresden, Alemania.

[9] Bourgeois, Anu,Zheng, S.Marowka, Ami. Performance of OpenMP Benchmarks on Multicore Processors, Algorithms and Architectures for Parallel Processing.Lecture Notes in Computer Science. 2008, Vol 5022. Springer Berlin / Heidelberg.

[10] Jin, H.; Van der Wijngaart, R.F.; , Performance characteristics of the multi-zone NAS parallel benchmarks, Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International , vol., no., pp. 6, 26-30 April 2004 doi: 10.1109/IPDPS.2004.1302906

[11] Terpstra, Dan and Jagode, Heike and You, Haihang and Dongarra, Jack. "Collecting Performance Data with PAPI-C", Tools for High Performance Computing 2009. Springer Berlin Heidelberg.

[12] R Van der Pas, "Getting OpenMP Up To Speed", the 4th International Workshop on OpenMP. Purdue University West Lafayette, IN, USA, May 12-14, 2008.

[13] Ryoo, Shane and Rodrigues, Christopher I. and Baghsorkhi, Sara S. and Stone, Sam S. and Kirk, David B. and Hwu, Wen-mei W.. Optimization Principles and Application Performance Evaluation of a Multithreaded GPU Using CUDA. PPoPP '08 Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming.ACM New York, NY, USA 2008. Pag. 73-82.

[14] Torres Lionel, Benoit Pascal, Sassatelli Giles and Robert Michel. An introduction to MultiCore System On Chip. Trends and Challenges. Multiprocessor System-on-Chip: Hardware Design and Tool Integration. Pag. 1-18, Springer 2010.Berlin Heidelberg.

[15] Conway, P. and Kalyanasundharam, N. and Donley, G. and Lepak, K. and Hughes, B. "Cache hierarchy and memory subsystem of the amd opteron processor".Micro, IEEE , vol.30, no.2, pp.16-29, March-April 2010 doi: 10.1109/MM.2010.31.

[16] Cakarevic, V.; Radojkovic, P.; Verdu, J.; Pajuelo, A, "Characterizing the Resource-Sharing Levels in the UltraSPARC T2 Processor".Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on. Pag 481-492.

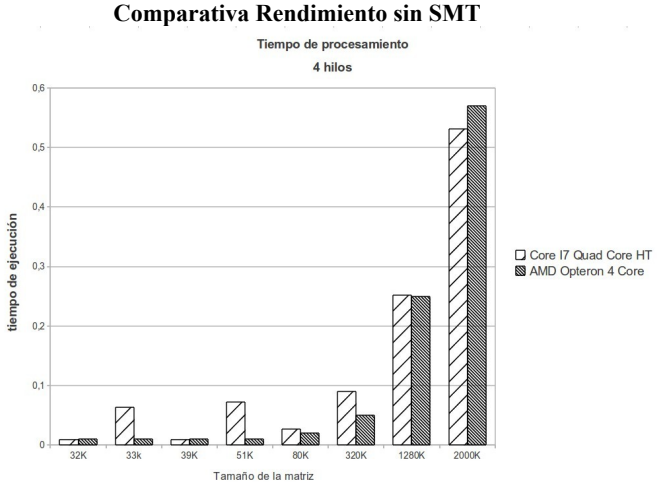
[17] Zaparanuks, D. and Jovic, M. and Hauswirth, M. "Accuracy of performance counter measurements, IEEE International Symposium on Performance Analysis of Systems and Software", 2009. ISPASS 2009.

Esteban Hernández Barragán. Es ingeniero de Redes de la Universidad Distrital Francisco José de Caldas de Bogotá, con estudios de postgrado en matemáticas . Actualmente cursa el Máster en software libre de la Universidad Oberta de Cataluña. Se desempeña como consultor en los temas de performance tunning para servidores de aplicaciones basado en Java. Sus principales áreas de interés radican en modelos y metodologías de tunning en soluciones con alta concurrencia. En la actualidad trabaja temas relacionados con desarrollo en paralelo usando software libre y la utilización de soluciones multicore para modelamiento y cálculo científico .ehernandezba@uoc.edu

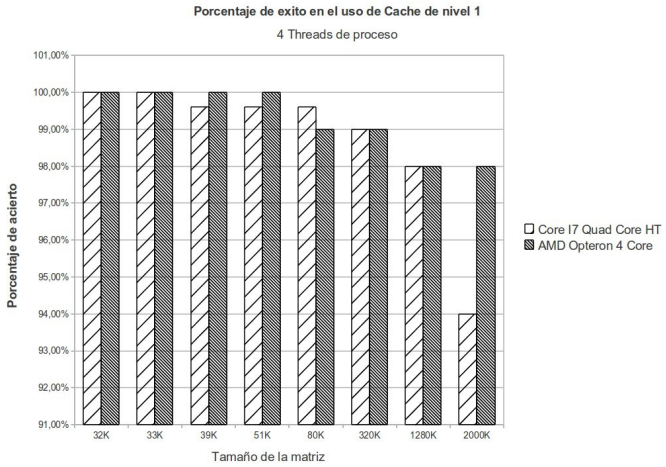
Josep Jorba Esteve Es Ingeniero superior en Informática por la UAB. Magíster en Arquitectura y Procesamiento Paralelo por la UAB y se desempeña como consultor en la Universidad Oberta de Cataluña. Sus principales intereses radican en los temas relacionados con el procesamiento paralelo, la optimización en sistemas operativos libres y la utilización de herramientas para desarrollo en paralelo. jjorbae@uoc.edu**.**

ANEXOS

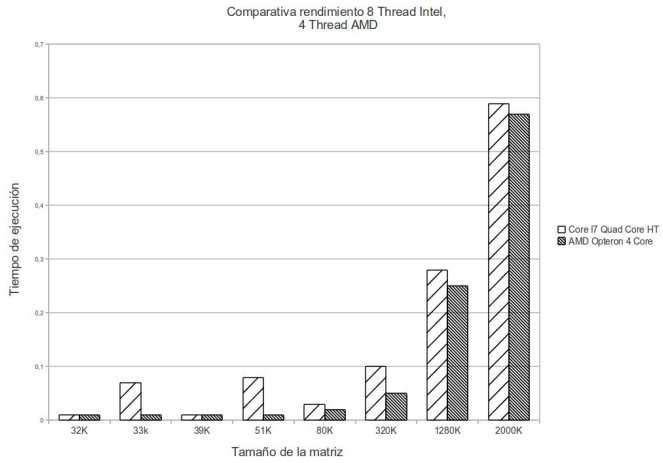
Comparativa Rendimiento sin SMT



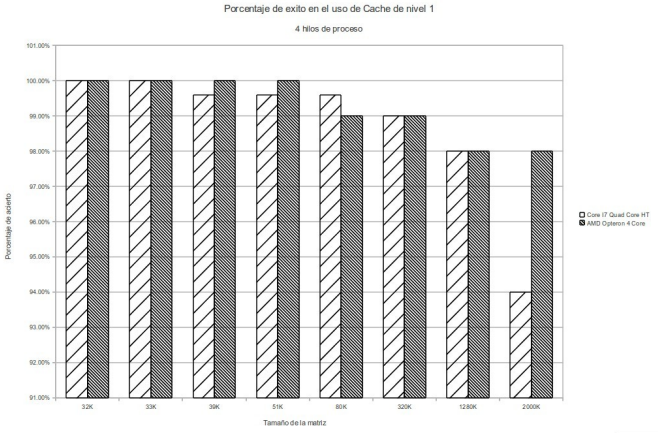
Porcentaje éxito en el uso de cache L1 con SMT activo



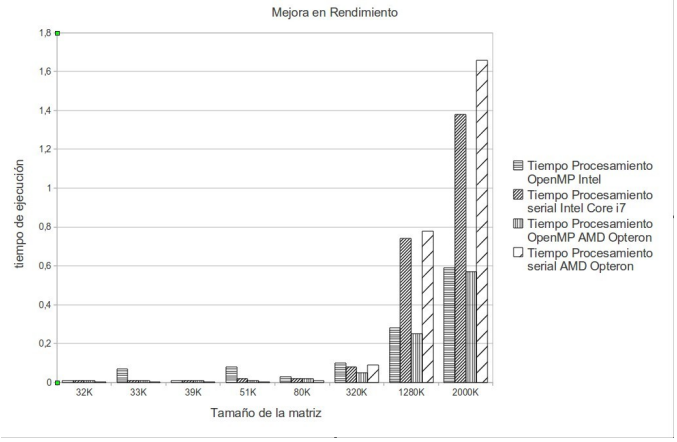
Comparativa rendimiento con SMT



Porcentaje de éxito en uso de caché L1 con SMT inactivo



Tiempos de procesamiento algoritmos serie y paralelo usando openmp



Distribución de arquitectura de procesadores en la lista top500.org