Selección de nodos de cómputo multicore, para una aplicación paralela de memoria compartida *

John Corredor, Juan Carlos Moure, Dolores Rexachs, Dani Franco, and Emilio Luque

Departamento de Arquitectura de Computadores y Sistemas Operativos,
Universidad Autónoma de Barcelona, España
john.corredor@caos.uab.es
{JuanCarlos.Moure,Dolores.Rexachs,Dani.Franco,Emilio.Luque}@uab.es
http://www.caos.uab.es

Abstract. Con la llegada de una amplia variedad de arquitecturas multicore (NUMA, UMA), seleccionar la mejor configuración del nodo de cómputo para una cierta aplicación paralela de memoria compartida, se convierte en la actualidad en un gran reto. Nuestro trabajo hace frente a este tema caracterizando los nodos de cómputo y las aplicaciones. Los nodos se caracterizan ejecutando pequeños programas (o microbenchmarks, μB), que contienen núcleos de estructuras representativas del comportamiento de programas paralelos de memoria compartida. Los $\mu B's$ ejecutados en cada uno de los nodos nos proporcionan perfiles de rendimiento, o datos medidos del comportamiento, que se almacena en una base de datos y se utiliza con para estimar el comportamiento de nuevas aplicaciones La aplicación es ejecutada sobre un nodo base para identificar sus fases representativas. Para cada fase se extrae información de rendimiento comparable con la de los $\mu B's$, con el fin de caracterizar dicha fase. En la base de datos de los perfiles de rendimiento se localizan $\mu B's$ con características similares en comportamiento para cada fase de la aplicación sobre el nodo base. Finalmente, los perfiles seleccionados, pero ejecutados sobre los otros nodos candidatos, se usan para comparar el rendimiento de los nodos de cómputo y seleccionar el nodo de cómputo apropiado para la aplicación

Key words: Evaluación de prestaciones, modelos de rendimiento, sistemas de cómputo multicore

1 Introducción

En la actualidad, para configurar o seleccionar los recursos de cómputo a utilizar en un computador paralelo, nos encontramos con que existen una gran variedad de componentes y tecnologías a escoger. Con el actual desarrollo de procesadores multicore, es aún mayor la variedad de configuraciones de los nodos, y por ende, de la influencia en las prestaciones que pueden tener estas posibles configuraciones para una aplicación. Por tanto, es importante caracterizar los sistemas.

* Este trabajo ha sido subvencionado por el MEC (España), proyecto TIN 2007-64974

Sin embargo, la mejor configuración del sistema es dependiente de la aplicación, y los benchmarks usados para la caracterización de recursos deben ser cuidadosamente seleccionados.

A su vez debemos tener en cuenta que a lo largo de una aplicación va cambiando el comportamiento y necesitaremos la caracterización del comportamiento de la CPU en diferentes puntos, lo que aconseja la utilización de microbenchmarks ($\mu B's$).

En las aplicaciones que requieren gran capacidad de cómputo, la influencia sobre el rendimiento en las diferentes configuraciones de nodos de cómputo, es muy importante tanto para los administradores como para los usuarios finales [1]. Es deseable tener un índice para comparar el rendimiento de la aplicación en diferentes configuraciones de nodos de cómputo y, de esa manera, tener criterios para seleccionar el más adecuado. Esta es una tarea que puede requerir mucho tiempo. Por ello, es importante disponer de una estrategia que nos permita, hacerlo en un tiempo razonable.

La metodología se resume en obtener: los *perfiles de rendimiento de los nodos de cómputo* ejecutando diferentes microbenchmarks ($\mu B's$), la *caracterización de la aplicación* buscando las características de las fases significativas que influyen en el rendimiento y finalmente combinar los resultados para hacer la estimación de rendimiento sobre diferentes nodos de cómputo que permite compararlos, sin que requiera ejecutar la aplicación. Para ello se propone generar un conjunto de $\mu B's$ bien caracterizados, que permitan obtener información acerca de los nodos de cómputo y obtener su Perfil de Rendimiento. La caracterización de la aplicación es un modelo de la aplicación que se extrae analizando la traza de accesos a memoria y otras características cualitativas y cuantitativas de la aplicación, independientes de la arquitectura. Finalmente, se modela el comportamiento con $\mu B's$ que tengan características de comportamiento similares a las de cada fase de la aplicación.

El artículo está organizado como sigue: En la sección 2, se describen otros trabajos relacionados. En la sección 3 presentamos la metodología propuesta para caracterizar el comportamiento. En la sección 4, se muestra la validación experimental. Finalmente en la sección 5, se presentan las conclusiones y trabajos futuros.

2 Trabajos relacionados

La predicción de rendimiento de aplicaciones sobre sistemas de cómputo, es un área de estudio muy consolidada. Las estrategias que se utilizan normalmente: modelos analíticos, la medición mediante la ejecución de benchmarks y la simulación.

Un gran conjunto de benchmarks han sido propuestos para medir las prestaciones generales de aplicaciones. Entre los más conocidos destacamos los NAS Parallel [4] y los SPEC [5]. Los SPEC son empleados para evaluar características de microarquitectura en sistemas paralelos. Gustafson y Todi [6] relacionan las aplicaciones a la curva del benchmark HINT sobre una máquina y así predecir

el rendimiento de la aplicación. El problema de los benchmarks es buscar su relación con la aplicación. McCalpin [7] hace un estudio más completo entre la relación de los benchmarks y el rendimiento de las aplicaciones, pero ellos no extienden sus ideas a aplicaciones paralelas.

En el trabajo de Carrington y otros [2], elabora modelos computacionales para predecir el rendimiento sobre múltiples sistemas de cómputo. De igual forma, Lau y otros [8], estudian la correlación existente entre el código de una aplicación y la predicción de rendimiento.

Sharkawi y otros [1], presentan un método para proyección de rendimiento de aplicaciones HPC sobre el nodo de cómputo, usando datos publicados del SPEC CPF2006, y contadores hardware de la máquina base. Su esquema utiliza el algoritmo genético como herramienta para generar un modelo de rendimiento de la aplicación HPC como una función de benchmarks sustitutos o similares, en este caso los SPEC CPF2006.

Nuestro enfoque, pretende combinar la información de los perfiles de rendimiento obtenidas con un conjunto de microbenchmarks significativos, de múltiples plataformas o nodos de cómputo, para estimar sobre ellas las características de rendimiento de una aplicación, como una función de datos obtenidos con los microbenchmarks previamente seleccionados.

3 Metodología: Comparación de prestaciones

El objetivo de este trabajo es presentar una metodología para estimar las posibles diferencias en el comportamiento, al ejecutar una aplicación en distintos nodos multicore, desde el punto de vista del rendimiento (tiempo de ejecución) de dicha aplicación paralela de memoria compartida, en diferentes configuraciones de nodos de cómputo con sistemas multiprocesador. La figura 1 muestra el esquema general de la metodología.

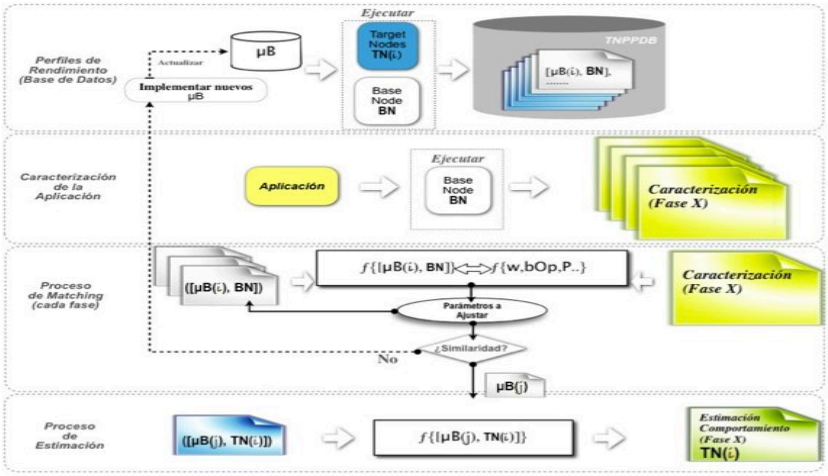

Fig. 1. Esquema de la metodología para estimar el rendimiento

Los perfiles de rendimiento de los nodos de cómputo (Nodo Base y Nodos Target), se obtienen ejecutando un conjunto de microbenchmarks ($\mu B's$), o programas sintéticos de prueba, bien caracterizados, compuestos por estructuras representativas de aplicaciones reales.

Para obtener la caracterización de la aplicación, se ejecuta sobre el Nodo Base (BN) con el fin de analizar y extraer sus fases representativas y obtener información del comportamiento en cada fase.

Basándonos en la información de rendimiento almacenada en la NPPDB y la caracterización de la aplicación, hacemos un proceso de comparación (matching) para identificar aquellos perfiles de rendimiento, obtenidos sobre el BN al ejecutar los microbenchmarks, que tienen un comportamiento similar con cada una de las fases representativas de la aplicación. Cuando se encuentra similaridad, el μB seleccionado será el candidato para representar a la fase en el proceso de estimación.

En el caso de no ser posible encontrar similaridad entre los perfiles de rendimiento obtenidos con los $\mu B's$ y la caracterización de la fase, será necesario implementar nuevos $\mu B's$ y actualizar la base de datos NPPDB.

Por último, tenemos el Proceso de Estimación, que consiste en localizar en la Base de Datos, las métricas de los $\mu B's$ seleccionados en el paso anterior en la NPPDB, y estimar el comportamiento de la aplicación como una función de los perfiles de rendimiento seleccionados en el paso anterior sobre el BN, considerando el peso de cada una de las fases.

Una vez validada la estimación como composición de los perfiles sobre el BN, se han de buscar ahora los perfiles de rendimiento para los $\mu B's$ seleccionados, pero de los nodos candidatos o Nodo Target y reemplazar en el modelo del tiempo de ejecución de la aplicación y así estimar el *perfil de comportamiento* en otros nodos. A continuación describimos el esquema con mayor detalle cada una de las fases.

3.1 Perfiles de rendimiento de los nodos

Los perfiles son obtenidos estresando el nodo de cómputo con programas sintéticos paralelos de memoria compartida implementados en C y OpenMP. El objetivo es caracterizar el nodo de cómputo frente estructuras comunes encontradas en varias aplicaciones. Cada ejecución de un μB nos da una información del perfil de rendimiento del nodo de cómputo.

La Figura 2, muestra una descripción de algunos de los $\mu B's$ que hemos usado, con el fin de visualizar un ejemplo de Perfil de Rendimiento del Nodo.

Los $\mu B's$ son diseñados a partir de estudios previos realizados en un conjunto de aplicaciones paralelas en función de sus comportamientos y sus patrones de acceso a memoria. El diseño de los $\mu B's$ está guiado por características relevantes de la aplicación (por ejemplo, la forma que accede a memoria, el stride, etc).

Un perfil de rendimiento es una lista de valores, que caracterizan el μB junto con valores promedios obtenidos en la ejecución. La idea es obtener información cualitativa y cuantitativa en cada nodo de cómputo.

	… μB2 (k)	μB3 (k)		μB9 (k)…
Patrón de acceso a memoria	V1 - V2: stride-1, secuencial	V1: stride-1; V2: stride-√k, secuencial		V1: stride-1; V2: stride-1, secuencial; P1 stride-1, secuencial
Tipo de acceso	Directo	Directo		V1: indirecto; V2 directo
Tipo de datos	Doble	Doble		Doble
Densidad de los datos	Denso	Denso		Denso
Acceso a memoria	Compartido	Privado		Privado
Operación básica (OpB)	+=V1[k]xV2[k]	+=V1[k]xV2[k]		+=V1[P1[k]]xV2[k]
Complejidad del algoritmo	O(n)	O(n)		O(n)
Nivel de paralelismo (Parámetro a ajustar)	Threads	Threads		Threads
Carga de trabajo (Parámetro a ajustar)	Tamaño (k); nº de iteraciones (it)	Tamaño (k); nº de iteraciones (it)		Tamaño (k); nº de iteraciones (it)
Tiempo de ejecución (Valor medido)	*	*		*
Tiempo por operación	Tiempo / (Nº de OpB x it)	Tiempo / (Nº de OpB x it)		Tiempo / (Nº de OpB x it)

Fig. 2. Ejemplo de descripción del perfil de rendimiento a partir de tres $\mu B's$

- Características cualitativas tales como: patrón de accesos a memoria, tipo de datos, operaciones básicas, complejidad del algoritmo y la densidad de los datos (densos o dispersas).
- Características cuantitativas tales como: tamaño de los datos de entrada, el nivel de paralelismo, tiempos de ejecución, tiempo por operación. Para obtener los perfiles de rendimiento, los $\mu B's$ son ejecutados previamente en todos los nodos de cómputo disponibles (BN y TNi) para diferentes tamaños de datos de entrada.

3.2 Caracterización de la Aplicación

Como hemos visto, en esta fase, se analizan aplicaciones paralelas de memoria compartida, implementada en C y con paradigma de programación paralela OpenMP[9] (Fork Join).

Necesitamos obtener las fases representativas de la aplicación. Para identificar cada fase, buscamos estructuras/primitivas/constructores de paralelismo en el código [10].

Esta caracterización nos permitirá contrastar y comparar con la información de los perfiles de rendimiento disponibles en la NPPDB del Nodo Base en el Proceso de Comparación (Matching).

3.3 Proceso de Comparación.

La idea es encontrar el μB o los $\mu B's$ que puedan representar el comportamiento de la aplicación, se busca el perfil de comportamiento de cada una de las fases representativas de la aplicación. El proceso consiste en identificar para cada una de las fases caracterizadas de la aplicación, que μB tiene del nodo base tienen características similares, para ello se utilizan los perfiles de rendimiento obtenidos de la ejecución de los $\mu B's$ y que están almacenados en la NPPDB.

El proceso comienza por comparar ordenadamente la lista de características mostradas en la tabla 1 para los diferentes $\mu B's$ con las características de cada

una de las fases seleccionadas de la aplicación. El orden dependerá del tipo de fase de la aplicación, distinguimos en este caso en fases limitadas por memoria, limitada por cómputo, limitada por llamadas a funciones.

Se pueden encontrar dos posibilidades frente al Matching:

- En caso de encontrarse similitud entre una de las fases de la aplicación y un μB en el BN, se ajustan los parámetros de entrada (tamaño de los datos de entrada y número de threads) del μB seleccionado. La idea es sintonizar los parámetros de entrada del perfil de rendimiento frente al de la fase.
- En caso de no encontrar similitud entre los perfiles en la NPPDB y la caracterización de la fase de la aplicación, será necesario implementar nuevos μB con el objeto de actualizar la NPPDB, con perfiles correspondientes a estos nuevos comportamientos.

Una vez ajustados los parámetros de entrada al perfil de rendimiento se ha de estimar sobre el mismo BN el tiempo y compararlo con la ejecución real, para validar si este μB seleccionado permite estimar.

3.4 Proceso de Estimación

La estimación del rendimiento corresponde a predecir el comportamiento de una aplicación paralela sobre un nodo de cómputo (nodo target), teniendo en cuenta que la aplicación no se ha ejecutado en este nodo.

Hasta ahora, hemos dividido el programa en fases representativas y cada una de las fases posee un correspondiente peso, sobre el total del tiempo de ejecución de la aplicación. Entonces, debemos seleccionar cada uno de los μB que han sido relacionados por su similaridad a éstas fases pero han sido ejecutados sobre los TN o nodos candidatos y así poder estimar el tiempo de ejecución. Esta información (el tiempo de ejecución) nos permitirá comparar el comportamiento de la aplicación sobre diferentes nodos de cómputo con el fin de seleccionar el nodo más conveniente.

4 Validación experimental

4.1 Entorno Experimental

Experimentalmente validamos la metodología usando las configuraciones de nodos de cómputo que se describen en la Figura 3.

Los sistemas TN poseen dual-socket, y para este estudio hemos utilizado solo 4 cores, tanto para obtener los perfiles de rendimiento, como para la caracterización de las aplicaciones a estudiar en el presente artículo. En el caso en que tengamos mas de 4 cores (por ejemplo: TN2 y TN3), hemos realizado los experimentos enlazando los procesos dentro de un procesador o pastilla de silicio.

Para seleccionar la afinidad del proceso, es decir, enlazar un proceso o un thread específico a un core o procesador específico, usamos la biblioteca sched.h. El TN3, es un sistema con procesador AMD de cache L2 privada para cada core. Posee una L3 compartida para cada 4 cores y un sistema operativo con 32bits (i686).

	BN	TN1	TN2	TN3
Processor	Quad-Core Q9400 Intel	Dual-Core 5160 Intel	Quad-core E5430 Intel (x2)	Quad-core 2352 AMD (x2)
Cores	4	4	8	8
On-chip Private	32KB (L1)	32KB (L1)	32KB (L1)	512KB (L2)
On-chip Shared	3MB (x2)	4MB (x2)	6MB (x2)	2MB (L3)
Memory	4GB	8GB	16GB	4x2 GB NUMA
Ghz	1,99	2,66	2,00	2,11
O.S.	x86_64 GNU/Linux	x86_64 GNU/Linux	x86_64 GNU/Linux	i686 GNU/Linux
Kernel	2.6.26-2	2.6.16-46	2.6.18-8	2.6.24-16
Compiler	gcc-4.3	gcc-4.3	gcc-4.3	gcc-4.3

Fig. 3. Configuración de Nodos de cómputo: Nodo Base BN y Nodos Target TN(i)

4.2 Perfiles de rendimiento

La Figuras 4, presentan los perfiles de rendimiento del $\mu B3$, caracterizados en la Figura 2; ejecutados en los nodos de cómputo descritos en la Figura 3.

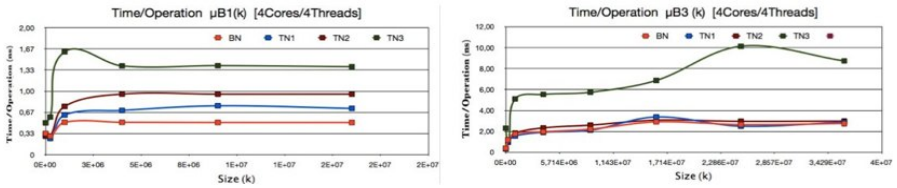

Fig. 4. Perfiles de Rendimiento: $\mu B1$ y $\mu B3$, sobre diferentes configuraciones de nodos de cómputo

Los perfiles se obtienen ejecutando previamente sobre todos los nodos de cómputo. Para obtener validación estadística se hace un numero de repeticiones de cada ejecución superior a 30 veces, y de esa manera extraer el promedio de los valores. Los parámetros a ajustar en los $\mu B's$ son el Nivel de Paralelismo (4threads) y diferentes tamaños de los vectores (k entre 1000 y 49 millones de elementos).

4.3 Caracterización de la Aplicación

Debido a que las operaciones con matrices son fundamentales para la ciencia computacional, hemos considerado como ejemplo para probar nuestra metodología dos variantes del algoritmo paralelo de Multiplicación de Matrices.

A continuación,mostraremos la descripción de un algoritmo MM1. La implementación del algoritmo ha sido hecho por descomposición row-wise (es decir, paralelismo por filas), cada subconjunto de filas es distribuido para cada thread.

$$c_{i,j} = \sum_{l=1}^{N} a_{i,l} \mathrm{x} b_{j,l} \tag{1}$$

Selección de nodos de cómputo multicore,

En (1), se presenta el algoritmo MM1, la matriz B es almacenada por filas, A(filas) por B(filas).

	MM1(N)			
Patrón de acceso a memoria	A: stride-1 (secuencial) B: stride-1 (secuencial)			
Tipo de acceso (Directo, Indirecto)	Directo			
Tipo de Dato (float, double, int, long)	Double			
Densidad de los datos	Denso			
Operación Básica	$C_{i,j} += A_{i,1} \times B_{j,1}$			
Acceso a Memoria (Compartido/Privado)	A:Privado B:Compartido			
Complejidad Algoritmo	$O(n^3)$			
Nivel de Paralelismo	4 (Th)			
Carga de Trabajo (N)	1000	2000	3000	4000
Tiempo de Ejecución (s)	522.157	4.159.967	1,38E+07	3,28E+07
Tiempo/Operación (ns)	0,522	0,520	0,512	0,512

Fig. 5. Caracterización de la Aplicación: Multiplicación de Matrices (algoritmo MM1)

El algoritmo es ejecutado en el BN, con el fin de obtener la caracterización y los valores a medir y calcular como el Tiempo/Operación. Para este ejemplo hemos usado como datos de entrada N=1000, 2000, 3000 y 6000.

4.4 Proceso de Matching

Después de ejecutar el algoritmo sobre el BN, debemos empezar a comparar entre la caracterización de la aplicación (ver Figura 6) y el perfil de rendimiento dado por los $\mu B's$.

1. Empezamos por buscar la similitud entre los Patrones de Acceso a Memoria, entre las Figuras 2 y 5. Los perfiles candidatos vienen dados por $\mu B1$ y $\mu B2$, debido a que tienen "stride-1" secuencial para ambos vectores como el algoritmo MM1.
2. En la Figura 7, se presentan los valores calculados del Tiempo/Operación y la diferencia (%) entre el algoritmo MM1(N) y los dos $\mu B(k)$ seleccionados en el paso anterior. Ajustamos el parámetro según los valores de entrada al algoritmo MM1(N) (N=1000, 2000, 3000 y 6000) en los $\mu B(k)$ seleccionando k=N^2.

Como se puede apreciar en la Figura 7 el comportamiento $\mu B2$ hace un muy buen match con el algoritmo MM1.

4.5 Proceso de Estimación

Hasta ahora hemos obtenido los $\mu B's$ cuyos perfiles de rendimiento hacen match con los algoritmos en el BN (Sección 4.4). Para hacer la estimación sobre los nodos candidatos o TN(i), comenzamos a buscar los valores de los Perfiles de

MM1(N)	μB1(k)	μB2(k)
A: stride-1 (seq.) B: stride-1 (seq.)	V1:stride-1 sequential	V1,V2:stride-1 sequential
Direct	Direct	Direct
Double	Double	Double
A:Private B:Shared	Shared	Shared
$c_{i,j}$ += $a_{i,j}$ x $b_{j,j}$	+=V1[k]	+=V1[k]xV2[k]

	MM1(N)	μb1(k)		μb2(k)	
N	Time/Operation (ns)	Time/Operation (ns)	Diff.	Time/Operation (ns)	Diff.
1E+03	0,525	0,510	2,84%	0,526	0,19%
2E+03	0,528	0,508	3,79%	0,520	1,52%
3E+03	0,513	0,480	6,43%	0,499	2,73%
6E+03	0,519	0,506	2,41%	0,507	2,31%

Match: Application and Profile from μB2

Fig. 6. Tiempo/Operación de MM1 y $\mu B2$ obtenidos en el BN

Rendimiento en la base de datos NPPDB de éstos $\mu B's$ pero que han sido ejecutados sobre los nodos candidatos. De esta forma obtenemos la estimación de comportamiento de los algoritmos sobre los TN(i). En la Figura 7 a) se presentan las predicciones de comportamiento MM1 los nodos candidatos o TN.

Con esta predicción de comportamiento podríamos seleccionar el nodo mas conveniente para esta aplicación. Para validar los resultados, en la Figura 7 b), mostramos el Tiempo/Operación de los algoritmos de Multiplicación de Matrices MM1 sobre los nodos candidatos.

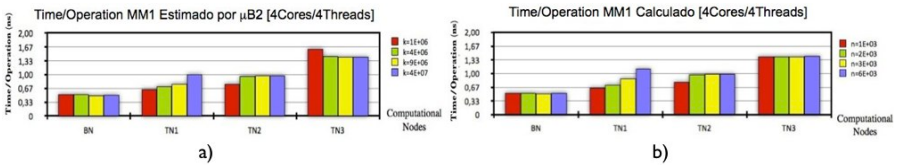

Fig. 7. Comportamientos: a) Estimado y b)Real Calculado para la aplicación MM1 sobre diferentes configuraciones de nodos de cómputo.

4.6 Conclusiones

Hemos presentado una metodología para la Selección del Nodo de cómputo Multicore, para una Aplicación Paralela de Memoria Compartida, mediante los perfiles de rendimiento de variedad de nodos de cómputo con diferentes configuraciones. Los perfiles de rendimiento son extraídos usando microbenchmarks $\mu B's$,basados en la idea de kernels o núcleos los cuales son primitivas en comportamiento de zonas representativas de aplicaciones paralelas. Los tiempos de ejecución de los tres $\mu B's$ son muy cortos, en relación al tiempo que toma la ejecución de la aplicación. El objetivo de los $\mu B's$ es estresar entre el sistema de memoria de los nodos de cómputo como también el cómputo, de manera de

poder identificar diferencias significativas en términos de prestaciones entre los nodos.

Hemos creado un método para caracterizar la aplicación, y hemos mostrado el funcionamiento de la metodología sobre un algoritmo de multiplicación de matrices, para diferentes tamaños de entrada. La caracterización del algoritmos resultan en un buen match o emparejamiento con los $\mu B's$ propuestos, y a su vez permite una selección rápida de una configuración de nodo de cómputo apropiada.

References

1. Sameh Sharkawi, Don Desota, Raj Panda, Rajeev Indukuru, Stephen Stevens, Valerie Taylor and Xingfu Wu: Performance Projection of HPC Applicacions Using SPEC CFP2006 Benchmarks. IPDPS (2009)
2. Snavely, A., Wolter, N., and Carrington, L. 2001. Modeling application performance by convolving machine signatures with application profiles. In Proceedings of the Workload Characterization, 2001. Wwc-4. 2001 IEEE international Workshop 2001. WWC. IEEE Computer Society.
3. Hollingsworth, J. K., Snavely, A., Sbaraglia, S., and Ekanadham, K. 2005. EMPS: An Environment for Memory Performance Studies. In Proceedings of the 19th IEEE international Parallel and Distributed Processing Symposium - Workshop 10 - IPDPS 2005. IEEE Computer Society.
4. Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., Simon, H. D., Venkatakrishnan, V., and Weeratunga, S. K. The NAS parallel benchmarks. In Proceedings of the 1991 ACM/IEEE Conference on Supercomputing 1991. Supercomputing '91.
5. SPEC, http://www.spec.org/
6. Gustafson, J. L. and Todi, R. Conventional Benchmarks as a Sample of the Performance Spectrum. J. Supercomput. 13, 3 (May. 1999).
7. J. McCalpin, Memory bandwidth and machine balance in current high performance computers. IEEE Technical Committee on Computer Architecture Newsletter. 1995
8. J.Lau, J.Sampson, E. Perelman, G. Hamerly, and B.Calder. The strong correlation between code signatures and performance. Performance Analysis of Systems and Software, 2005. ISPASS2005. International Symposiumon Performance Analysis of Systems and Software, 2005.
9. R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald. Parallel Programming in OpenMP. Morgan Kaufmann, 2000.
10. J. Corredor, J. C. Moure, D. Franco, D. Rexachs, E. Luque.: Selecting a suitable multicore system for shared-memory parallel applications. In: Proceedings PDPTA 10 - The 2010 International Conference on Parallel and Distributed Processing Techniques and Applications. Las Vegas, Nevada - USA (July 2010).