
**Escola d'Enginyeria**

**Departament d'Arquitectura de Computadors i Sistemes Operatius**

# Predicción de Perfiles de Comportamiento de Aplicaciones Científicas en Nodos Multicore

Memoria presentada por **John Jairo Corredor Franco** para optar al grado de *Doctor por la Universitat Autònoma de Barcelona* dentro del Programa de Doctorado en Computación de Altas Prestaciones.

Bellaterra, Junio 2011

# Predicción de Perfiles de Comportamiento de Aplicaciones Científicas en Nodos Multicore

Tesis Doctoral presentada por ***John Jairo Corredor Franco*** para optar al grado de *Doctor por la Universitat Autònoma de Barcelona*, dentro del Programa de Doctorado en Computación de Altas Prestaciones. Este trabajo ha sido desarrollado en el departamento de Arquitectura de Computadores y Sistemas Operativos de la Escuela de Ingeniería de la Universitat Autònoma de Barcelona, bajo la dirección del Dr. Juan Carlos Moure López y la Dra. Dolores Rexachs del Rosario.

Dirección de la Tesis:

Dr. Juan Carlos Moure López

Dr. Dolores Isabel Rexachs del Rosario

Bellaterra, Junio de 2011

# Resumen

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, $\mu B$), que contienen núcleos de estructuras representativas del comportamiento de programas paralelos de memoria compartida. Los $\mu Bs$ ejecutados en cada uno de los nodos nos proporcionan perfiles de rendimiento, o datos medidos del comportamiento, que se almacenan en una base de datos y se utilizan 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 $\mu Bs$, con el fin de caracterizar dicha fase. En la base de datos de los perfiles de rendimiento se localizan aquellos $\mu Bs$ que tienen características similares en comportamiento a cada fase de la aplicación, ejecutados todos sobre el nodo base. Finalmente, se usan los perfiles seleccionados, pero ejecutados sobre los otros nodos candidatos, para comparar el rendimiento de los nodos de cómputo y seleccionar el nodo de cómputo apropiado para la aplicación.

# Abstract

With the advent of multicore architectures, there arises a need for comparative evaluations of the performance of well-understood parallel programs. It is necessary to gain an insight into the potential advantages of the available computing node configurations in order to select the appropriate computing node for a particular shared-memory parallel application. This paper presents a methodology to resolve this issue, by constructing a database with behavior information for various representative shared-memory programming structures, and then estimating the application behavior as a combination of these data. These structures are represented by small specific chunks of code called microbenchmarks ($\mu B$) based on the study of memory access patterns of shared-memory parallel applications. $\mu Bs$ set is run on each candidate node, and all execution *performance profiles* are stored in a database for future comparisons. Then, applications are executed on a *base node* to identify different execution phases and their weights, and to collect performance and functional data for each phase. Information to compare behavior is always obtained on the same node (*Base Node (BN)*). The best matching *performance profile* (from *performance profile database*) for each phase, is then searched. Finally, the candidates nodes *performance profiles* identify in the *match process* are used to project performance behavior in order to select the appropriate node for running the application. Resource centers own different machine configurations. This methodology helps the users or systems administrator of data centers to schedule the jobs efficiently.

# Agradecimientos

Primero que nada, debo agradecer y dedicar este trabajo a mis padres *Fabiola y Freddy*(q.e.p.d.). A mis hermanos (*Liliana, Claudia, Carolina y Freddy Alonso*). Sin la ayuda de ellos jamás podría haber logrado nada en la vida, gracias por su amor, ayuda y comprensión. A mi abuela *Rosa María* que siempre la llevo en mis recuerdos de infancia, y a toda mi familia.

También agradecer a *Juan Carlos* por su gran colaboración.

A *Lola* con su gran paciencia, amabilidad, motivación y orientación, sin duda un gran apoyo moral.

A *Emilio* por su gran capacidad de orientar en todo momento. A *Daniel Franco* por su objetividad y recomendaciones.

A todos los profesores, compañeros y funcionarios del Departamento CAOS.

No puedo dejar de nombrar a algunos amigos, que compartimos la misma lucha diaria y que de alguna u otra manera han contribuido a la realización de este trabajo: gracias *Dario* por su gran colaboración en todo momento, *Alexander* (*El Negro*), *Alexandre* y *Roberto*.

A mis amigos y amigas que siempre me colaboraron en todo momento *Ricky*, *Nathalie*, *Gemma*, *Javi* y *Carlos*.

*"Cien veces al día me recuerdo a mi mismo, que me vida interior y exterior se alimentan de los esfuerzos de otros hombre vivos y muertos, y que debo tratar de dar en la misma medida en que he recibido".*

*Albert Einstein*

# Índice general

**1. Introducción** **1**
1.1. Contribución de la Tesis . . . . . 3
1.2. Organización . . . . . 4

**2. Marco teórico y trabajos relacionados** **5**
2.1. Métodos de Predicción . . . . . 5
2.1.1. Benchmarking . . . . . 6
2.1.2. Modelos Analíticos . . . . . 10
2.1.3. Simulación . . . . . 11
2.2. *Frameworks* de Predicción de Rendimiento . . . . . 12
2.2.1. Caracterización de las Fases en las Aplicaciones . . . 17

**3. Metodología Propuesta** **19**
3.1. Comportamiento Prestacional . . . . . 19
3.1.1. Perfiles de rendimiento de los nodos de cómputo . . 23
3.1.2. Caracterización de la Aplicación . . . . . 37
3.1.3. Proceso de Comparación. . . . . 43
3.1.4. Proceso de Estimación . . . . . 56

**4. Validación Experimental** **60**
4.1. Configuración de los nodos de cómputo . . . . . 60
4.2. Descripción de las aplicaciones seleccionadas . . . . . 62
4.3. Aplicación de la metodología para predecir el perfil de comportamiento prestacional . . . . . 63

4.3.1. Multiplicación de matrices: análisis de un núcleo con una única fase representativa . . . . . 64
4.3.2. Simulador de Partículas N-Body: análisis de un núcleo con una única fase representativa . . . . . 81
4.3.3. Bloque Tridiagonal Nas Parallel Benchmark: análisis de con varias fases representativas . . . . . 94

**5. Conclusiones** **108**
5.1. Principales Contribuciones . . . . . 109
5.2. Líneas Abiertas . . . . . 110

**Bibliografía** **111**

# Índice de figuras


3.1. Propuesta para la predicción del rendimiento. . . . . 19
3.2. Propuesta para la obtención de la predicción del rendimiento. 20
3.3. Esquema General: *metodología para estimar el rendimiento* 21
3.4. Tiempo de ejecución de 3 $\mu Bs$: operaciones de suma **Add**, operaciones de producto **Mul** y operaciones de acceso a memoria **Load**, en los nodos de cómputo que denominamos BN y TN1. . . . . 25
3.5. Tiempo de ejecución de 3 $\mu Bs$: operaciones de suma **Add**, operaciones de producto **Mul** y operaciones de acceso a memoria **Load**, en los nodos de cómputo que denominamos TN2 y TN3. . . . . 26
3.8. Esquema para la obtención de los *perfiles de rendimiento* . 29
3.9. Descripción de algunos $\mu B$s (serán usados en la Validación Experimental). . . . . 31
3.10. Descripción del $\mu B$: *Mbw1c* para diferentes *stride* y *workloads*. 33
3.11. Perfiles de Rendimiento $\mu B$: *Mbw1c* del Nodo Base (BN) para diferentes *stride* y *workloads*. . . . . 34
3.12. Diferencias de comportamiento del tiempo de ejecución del $\mu B$ *Mbw1c* en 4 nodos de cómputo diferentes, para dos *workloads* de entradas diferentes $w1$ y $w2$. . . . . 35
3.13. Esquema para la *caracterización de la aplicación*. . . . . 38
3.17. Caracterización de la Fase MM1: *workload 4000x4000* . . . 43
3.18. Esquema del *proceso de comparación* . . . . . 45
3.19. Umbrales para el algoritmo de comparación. . . . . 46

3.20. Caracterización de la fase App: *workload wi* con $n$ flujos de acceso a memoria . . . . . 47
3.21. Valores iniciales de los umbrales usados en la parte experimental . . . . . 50
3.22. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$ y $q2$. Todas las pruebas se han realizado para $th = 4$. . . . . 51
3.23. Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 3.22. . . . . 52
3.24. Algoritmo de Comparación: consulta $q2$ frente a información de *perfiles de rendimiento* Tabla 3.22. . . . . 53
3.25. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q3$. Todas las pruebas se han realizado para $th = 4$. . . . . 54
3.26. Algoritmo de Comparación: consulta $q3$ con la información de *perfiles de rendimiento* Tabla 3.25. . . . . 55
3.27. Tabla de *índices de similitud* de los $perfiles$ candidatos. . . 56
3.28. Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación). . . . . 58
3.29. Estimación y comportamiento real de la Fase (aplicación MM $workload$ $4000x4000$) sobre los diferentes nodos de cómputo caracterizados. . . . . 59

4.1. Descripción y diseño de los nodos de cómputo . . . . . 61
4.4. Caracterización de la Fase MM *workload 5000x5000* . . . . 66
4.5. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$ y $q2$. Todas las pruebas se han realizado para $th = 4$. . . . . 67
4.6. Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.5. . . . . 67
4.7. Algoritmo de Comparación: consulta $q2$ frente a información de *perfiles de rendimiento* Tabla 4.5. . . . . 68

4.8. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q3$. Todas las pruebas se han realizado para $th = 4$. . . . . 69
4.9. Algoritmo de Comparación: consulta $q3$ con la información de *perfiles de rendimiento* Tabla 4.8. . . . . 69
4.10. Tabla de *índices de similitud* de los $perfiles$ candidatos. . . 70
4.11. Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación). . . . . 71
4.12. Estimación y comportamiento real de la Fase (aplicación MM workload 5000x5000) sobre los diferentes nodos de cómputo caracterizados. . . . . 72
4.14. Caracterización de la Fase MM *workload 6000x6000* . . . . 75
4.15. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$ y $q2$. Todas las pruebas se han realizado para $th = 4$. . . . . 76
4.16. Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.15. . . . . 76
4.17. Algoritmo de Comparación: consulta $q2$ frente a información de *perfiles de rendimiento* Tabla 4.15 . . . . . 77
4.18. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q3$. Todas las pruebas se han realizado para $th = 4$. . . . . 77
4.19. Algoritmo de Comparación: consulta $q3$ frente a información de *perfiles de rendimiento* Tabla 4.18. . . . . 78
4.20. Tabla de *índices de similitud* de los $perfiles$ candidatos. . . 78
4.21. Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación). . . . . 79
4.22. Estimación y comportamiento real de la Fase (aplicación MM $workload$ $6000x6000$) sobre los diferentes nodos de cómputo caracterizados. . . . . 81

4.24. Caracterización de la Fase NBody *workload 200000p* . . . . 83
4.25. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$. Todas las pruebas se han realizado para $th = 4$. . . . . 84
4.26. Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.25. . . . . 84
4.27. Tabla de *índices de similitud* de los $perfiles$ candidatos. . . 85
4.28. Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación). . . . . 86
4.29. Estimación y comportamiento real de la Fase (aplicación N-Body) sobre los diferentes nodos de cómputo caracterizados. 87
4.31. Caracterización de la Fase NBody *workload 320000p* . . . . 89
4.32. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$. Todas las pruebas se han realizado para $th = 4$. . . . . 90
4.33. Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.32. . . . . 90
4.34. Tabla de *índices de similitud* de los $perfiles$ candidatos. . . 91
4.35. Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación). . . . . 92
4.36. Estimación y comportamiento real de la Fase (aplicación N-Body) sobre los diferentes nodos de cómputo caracterizados. 93
4.38. Caracterización de la Fase BT *workload ClaseA* . . . . . 96
4.39. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$. Todas las pruebas se han realizado para $th = 4$. . . . . 97
4.40. Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla. . . . . 97
4.41. Tabla de *índices de similitud* de los $perfiles$ candidatos. . . 98

4.42. Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación). . . . . 98
4.43. Estimación y comportamiento real de la Fase (aplicación BT NAS Parallel Benchmark), sobre los diferentes nodos de cómputo caracterizados. . . . . 100
4.45. Caracterización de la Fase BT *workload ClaseB* . . . . . 102
4.46. Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$. Todas las pruebas se han realizado para $th = 4$. . . . . 103
4.47. Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.46. . . . . 103
4.48. Tabla de *índices de similitud* de los $perfiles$ candidatos. . . 104
4.49. Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación). . . . . 105
4.50. Estimación y comportamiento real de la Fase (aplicación BT NAS Parallel Benchmark), sobre los diferentes nodos de cómputo caracterizados. . . . . 106

# Capítulo 1

# Introducción

La arquitectura de procesadores multicore o multinúcleo se ha convertido en una tendencia dominante en la industria de fabricación de chips. Se ha generalizado, ya que proporciona una mejor relación entre rendimiento y coste y un uso más eficiente de la energía [21]. Los procesadores multicore y *multithreaded* demandan un mayor ancho de banda de memoria dentro y fuera del chip. A su vez, en promedio sufre más retrasos de acceso a memoria, a pesar del aumento de tamaño de la caché en el chip. Se ejerce una enorme presión sobre el sistema de jerarquía de memoria, al suministrar las instrucciones y los datos necesarios.

Los diferentes diseños de nodos de cómputo producen diferencias significativas en el rendimiento de una aplicación. La tarea de encontrar el mejor diseño de nodo de cómputo para una aplicación es muy compleja, debido a la gran cantidad de alternativas de componentes individuales y al impacto diferente que tienen en las aplicaciones. Aplicaciones diferentes se benefician de diseños diferentes [1]. Con la llegada de una gran variedad de arquitecturas multi-core (NUMA, UMA), aumenta aún más esta variedad, y por ende aumenta la dificultad de esta selección.

Encontrar un diseño de nodo de cómputo o una configuración adecuada es un problema de ingeniería informática que se ha tratado mediante diferentes enfoques de evaluación de rendimiento. Los enfoques basados en modelos analíticos tienen limitaciones, por ejemplo, en términos de su capa-

cidad para tener en cuenta las características de la aplicación o la interacción entre varios procesadores. Los basados en simulación son métodos más comúnmente utilizados para la evaluación del rendimiento, pero requieren mucho tiempo en obtener los resultados de cada simulación [26].

En general, los actuales centros de recursos poseen distintos nodos de cómputo de diferentes diseños. Una metodología que ayude a predecir el comportamiento de las aplicaciones en diferentes sistemas de cómputo, basada en información del sistema y de la aplicación, sin necesidad de ejecutar en todos los sistemas, facilitará a los usuarios o administradores la selección de los recursos para ejecutar los trabajos de manera eficiente. Obtener conocimiento del rendimiento de las aplicaciones de altas prestaciones sobre varias plataformas de cómputo. Es importante para los proveedores, los usuarios de computación de altas prestaciones (*High Performance Computing* (HPC)), así como también para los administradores de centros de recursos. Las proyecciones de rendimiento ayudan a su vez a los proveedores de *hardware* en el diseño de sistemas futuros. Estas proyecciones permiten a su vez comparar el rendimiento de la aplicación frente a diferentes diseños de sistemas de cómputo existentes y futuros [42].

El nodo de cómputo es un componente fundamental en HPC. El diseño de los nodos de cómputo varía respecto a la velocidad y capacidad de la unidad de procesamiento o chip, al número de chips y cores dentro del chip, al tamaño de la memoria y la propia configuración de la jerarquía de memoria, entre otros componentes. El comportamiento del nodo de cómputo dependerá de la aplicación.

Medir el rendimiento de una aplicación paralela en diferentes diseños de nodos de cómputo es complicado y exige tener disponibles los recursos en el momento de realizar cada evaluación y mucho tiempo, incluso si el usuario dispone de todas las configuraciones posibles. Una estrategia que nos permite estimar el rendimiento de una aplicación en un plazo razonable y sin necesidad de tener los recursos disponibles para su evaluación, puede ayudar a seleccionar el mejor nodo respecto al rendimiento.

Se presenta una metodología para predecir el comportamiento del rendimiento de las aplicaciones HPC en el nivel de nodo, para diferentes diseños de nodos. La principal ventaja de esta metodología es que la aplicación no

necesita ser ejecutada en todos los diseños disponibles. Además, nuestro método no implica simulaciones, que a menudo consumen mucho tiempo.

Nuestra investigación tiene como objetivo diseñar una metodología con el fin de obtener una predicción de comportamiento de una aplicación paralela de memoria compartida. La metodología asocia la información del perfil de nodos de cómputo con información de la aplicación. La información del perfil se obtiene ejecutando microbenchmarks ($\mu B$), que son programas sintéticos de prueba, los cuales se ejecutan en todos los nodos de cómputo para caracterizarlos. Por otro lado, la aplicación se ejecuta únicamente en un nodo de cómputo o nodo referencia. El trabajo se ha centrado en el análisis de cómo afectan los diferentes diseños de la jerarquía de memoria en los nodos de cómputo en el comportamiento prestacional de las aplicaciones, cuando dichas aplicaciones acceden con diferentes patrones regulares de acceso a memoria.

El trabajo se ha centrado en este análisis porque existe un amplio conjunto de aplicaciones científicas cuyas prestaciones están limitadas por los accesos a memoria. A medida que el número de cores aumenta, la contención causada al acceder a memoria se incrementa, por lo que el rendimiento de estas aplicaciones estará muy influenciado por el diseño del sistema.

## 1.1. Contribución de la Tesis

La diversidad de diseños de nodos de cómputo ha resaltado la importancia y necesidad de tener criterios para evaluar los sistemas, y para seleccionar el nodo de cómputo donde prevemos que se ejecutará la aplicación con mejores prestaciones. El objetivo de este trabajo es diseñar un método que permita seleccionar el nodo de cómputo apropiado para una aplicación paralela de memoria compartida. Se hacen las siguientes contribuciones:

- Analizar las características de la aplicación en un nodo base. Minimizar el uso de recursos en el momento de la predicción utilizando un método que no requiera ejecutar la aplicación en el momento de tomar la decisión, una vez analizada la aplicación sea rápida la toma de decisiones;

- Capturar y almacenar las características de comportamiento de los nodos candidatos;
- Analizar las características de la aplicación en un nodo base.
- Validar la metodología propuesta por medio de un estudio experimental.

## 1.2. Organización

Este trabajo está organizado de la siguiente manera:

*El capítulo 2* presenta brevemente los principales trabajos relacionados en la evaluación y los métodos de predicción comúnmente empleados. Posteriormente, se analiza su relación con este trabajo.

*El capítulo 3* presenta la metodología propuesta para estimar las posibles diferencias en el comportamiento de aplicaciones paralelas de memoria compartida sobre nodos de cómputo multicore. La idea principal de la metodología para la estimación, es ordenar/comparar el comportamiento de los nodos de cómputo en función del rendimiento de la aplicación, basándose en una caracterización previa de los nodos y la ejecución de la aplicación en un nodo, sin necesidad de ejecutar la aplicación sobre el resto de nodos.

*El capítulo 4* describe la metodología para la validación experimental y se valida experimentalmente la metodología haciendo descripción del entorno empleado como también el seguimiento paso a paso de la metodología, de cara a la predicción del comportamiento de las aplicaciones de ejemplo sobre las configuraciones disponibles.

*El capítulo 5* muestra las conclusiones alcanzadas en el presente trabajo y algunas de las líneas abiertas son también descritas.

# Capítulo 2

# Marco teórico y trabajos relacionados

Una forma de seleccionar el mejor nodo de cómputo para una aplicación es predecir el comportamiento que tendrá dicha aplicación en los diferentes nodos de cómputo. Existen muchos trabajos que proponen métodos para evaluar el rendimiento de una aplicación en un computador. A continuación revisaremos brevemente los principales trabajos relacionados en la evaluación del rendimiento y los métodos de predicción comúnmente empleados. En primer lugar, haremos la descripción de los métodos usados para la evaluación del rendimiento. Algunos han servido de base para el desarrollo de nuestro trabajo de investigación. Posteriormente, se expondrá su relación con este trabajo.

## 2.1. Métodos de Predicción

En la siguiente sección se hará un breve resumen de los métodos más usados en la predicción y evaluación de sistemas de cómputo. El resumen comprende métodos empleados comúnmente tales como *benchmarking*, los *modelos analíticos* y la *simulación*. *Benchmarking* es una técnica comúnmente empleada para medir rendimiento sin ejecutar la aplicación real;

los modelos analíticos o funcionales emplean modelos matemáticos para analizar y predecir el rendimiento; los *modelos de simulación* hacen la descripción funcional de los componentes y consideran sus interacciones. A continuación se describirá brevemente cada uno de estos métodos.

### 2.1.1. Benchmarking

El mejor benchmark para medir el rendimiento de los sistemas de cómputo es la aplicación real[21]. El usuario de un sistema de cómputo que ejecuta los mismos programas diariamente es un buen candidato para diferentes sistemas de cómputo y analizar cual es el sistema que le dará mejores prestaciones. El conjunto de programas que ejecuta es lo que llamamos *carga de trabajo "workloads"*. Para evaluar dos sistemas de cómputo el usuario necesitaría tener disponibles los sistemas, ejecutar y comparar el tiempo de ejecución del *workload* en cada uno de los sistemas de cómputo. Sin embargo, no todos los usuarios disponen o de tiempo o de todos los sistemas de cómputo para poder evaluarlos. En su lugar, se debe contar con otros métodos que midan o predigan el rendimiento del sistema de cómputo candidato, esperando que las predicciones reflejen cómo trabaja el sistema con la carga de trabajo del usuario. Esto se hace habitualmente evaluando el sistema de cómputo; para ello se utiliza un conjunto de programas de prueba (*benchmarks*) para medir el rendimiento. Los programas de prueba forman una carga de trabajo que el usuario espera que prediga el rendimiento de su carga de trabajo real.

La ejecución de simples programas de prueba o *benchmarks* es mucho más simple y directo que ejecutar la aplicación entera sobre todos los sistemas de cómputo a evaluar. Sin embargo, los programas de prueba generan dificultades al relacionar la diferencia de comportamiento frente a la aplicación real. Como ejemplo de los programas de prueba [21]:

- Kernels, son pequeños trozos, piezas claves aplicaciones reales;
- Programas de juguete, programas de al menos 100-lineas de código tales como *quicksort* o algoritmos de búsqueda, y

- Benchmarks sintéticos, falsos programas que intentan ser similares a los perfiles y comportamientos de las aplicaciones reales, tales como Dhrystone.

Los primeros benchmarks sintéticos, tales como Whetston [13] y Dhrystone [55], fueron implementados para consolidar el comportamiento de las aplicaciones en un solo programa. Varios enfoques tales como (Ferrari [18], Curnow y Wichman [13], Sreenivasan y Kleinman [51]) han propuesto construir cargas sintéticas de trabajo que sean representativas de cargas de trabajo reales bajo un sistema de multiprograma. Con estas técnicas, se obtienen a partir de los datos del sistema las características de la carga real de trabajo y se construyen un conjunto de trabajos de carga similar en demanda de rendimiento en los recursos del sistema.

**SPEC CPU2000**

Es un Benchmark producido por la *Standard Performance Evaluation Corporation* (SPEC). Fue creado con el fin de proveer una medida de rendimiento que pueda ser usado para comparar cargas de trabajo intensivas en cómputo en distintos sistemas de computadora. Contiene dos benchmark suites: CINT2000 para medir y comparar el rendimiento de computación intensiva de enteros, y CFP2000 para medir y compara el rendimiento de computación intensiva en flotantes.

La "C" en CINT2000 y CFP2000 denotan que son benchmarks a nivel de componentes, en oposición a benchmarks de todo el sistema. Al ser intensivos en cómputos, miden el rendimiento del procesador de la computadora, la arquitectura de la memoria y el compilador. El CINT2000 y el CFP2000 no fuerzan la entrada/salida (unidades de disco), trabajo en red o gráficos.

Los benchmarks de CPU de SPEC proveen de una medida de rendimiento para comparar sistemas sobre la base de una carga de trabajo intensiva en cómputo bien conocida, con énfasis en la capacidad del procesador del sistema, la jerarquía de memoria y el compilador. SPEC ha mantenido el principio de que los mejores benchmarks están basados en aplicaciones, y estas aplicaciones pueden provenir de cualquier área de trabajo. Por ejemplo, el actual SPEC CPU2000 suite incluye aplicaciones de las siguientes

áreas:

- Teoría de juegos de la IA.
- Compiladores.
- Intérpretes.
- Compresión de datos.
- Bases de datos.
- Predicción del clima.
- Dinámica de fluidos.
- Física.
- Química.
- Procesamiento de imágenes.

**NAS Benchmark**

El programa *Numerical Aerodynamic Simulation* (NAS) de la NASA, propone un conjunto de benchmarks: el NAS *Parallel Benchmarks* (NPB)[3], con el propósito de evaluar prestaciones de computadoras altamente paralelas (*highly parallel supercomputers*). Estos benchmarks son usados en la industria para proyectar el rendimiento, ya que son compuestos por varios núcleos de aplicaciones científicas y por su naturaleza paralela.

Se compone de cinco problemas con núcleos paralelos de mediano tamaño computacional y tres diferentes esquemas de solución implícita. Estos esquemas son representativos de los códigos CFD actualmente en uso en la NASA *Ames Research Center*, debido a que imitan las actividades de cálculo y movimientos de datos típicos de aplicaciones reales de CFD.

Hay seis clases de problemas: S, W, A, B, C y D. Clase S es el problema más fácil y es para propósito de prueba solamente. Clase D es el más difícil y por tanto de mayor requerimiento de cómputo. Las clases C y D, son consideradas para grandes sistemas cluster, donde requieren comunicación inter-nodos o paso de mensajes MPI.

**EP** núcleo *embarrassingly parallel* (embarazosamente paralelo), genera valores pseudo-aleatorios de acuerdo a una curva Gaussian y sistemas uniformes. Este núcleo, a diferencia de otros en la lista, prácticamente no requiere la comunicación entre procesadores.

**MG** usa el método Vcycle MultiGrid para calcular la solución de la ecuación 3D *escalar Poisson*. El algoritmo funciona de forma continua en un conjunto de redes que se hacen entre gruesas y finas. Pone a prueba tanto movimiento de corta y larga distancia de datos.

**CG** emplea el método del Conjugado del Gradiente. Este algoritmo usa el método de las potencias inversas para encontrar una estimación del mayor autovalor o *eigenvalue* [1] de una matriz dispersa definida positiva con una distribución aleatoria de elementos no nulos.

**FT** contiene el núcleo computacional en 3D de la Transformada Rápida de Fourier (FFT) basado en el método espectral. FT realiza tres unidimensionales FFT, una para cada dimensión.

**IS** Este núcleo realiza una operación de ordenación que es importante en códigos que emplean métodos de partículas. Pone a prueba tanto la velocidad de cálculo de enteros como el rendimiento de la comunicación.

**LU** Una solución de un sistema inferior-superior triangular, regular-dispersa de un bloque (5x5). Este problema representa los cálculos asociados a un operador implícito de una nueva clase de algoritmos implícitos CFD.

**SP** Solución de sistemas múltiples, independientes de la no diagonal dominante, ecuaciones escalares, pentadiagonal. SP y el siguiente problema BT son representantes de los cálculos relacionados con los operadores implícitos de CFD. SP y BT son similares en muchos aspectos, pero

[1] conjunto especial de escalares asociadas a un sistema de ecuaciones lineales (es decir, una ecuación de la matriz) que a veces también es conocido como raíces características, valores característicos (Hoffman y Kunze 1971), valores propios, o de las raíces latentes[56].

hay una diferencia fundamental con respecto a la relación entre la comunicación y el cómputo.

**BT** Solución de sistemas múltiples, independientes de la no diagonal dominante, bloque de ecuaciones tridiagonal con un tamaño de bloque (5x5).

### 2.1.2. Modelos Analíticos

Una segunda área de la evaluación de rendimiento es la de modelado funcional y analítico. El rendimiento de una aplicación sobre un sistema candidato o destino puede ser descrito como una compleja ecuación matemática. Cuando las variables de la ecuación son sustituidas por los valores de entrada apropiados del sistema de cómputo destino o candidato, el cálculo de rendimiento es obtenido en un tiempo inmediato para la aplicación sobre ese sistema de cómputo candidato. Varios trabajos han sido utilizados desarrollando modelos matemáticos, brevemente se describirán algunos de estos trabajos.

Saavedra y Smith [40] [41] proponen modelar las aplicaciones como una colección de tareas independientes*Abstract FORTRAN Machine*. Cada tarea es medida en un sistema de cómputo candidato y se usa para predecir el tiempo de ejecución. Para incluir los efectos del sistema de memoria y hacer más real las medidas, los autores tomaron medidas de los fallos y aciertos a la jerarquía de memoria para incluirlos como "*overhead*" o sobrecarga de tiempo. Estos modelos funcionaron bien sobre los modelos simples o no tan complejos de procesadores y sistemas de memoria de mediados de los años 90. Los modelos actuales necesitan mejoras significativas en lo que respecta a sistema de memoria, sistema de interconexión y la aparición del procesamiento paralelo, el procesador multicore.

Para predicciones de sistemas paralelos, Mendes y Reed [34] [33] proponen un trabajo enfocado a múltiples plataformas. Los autores obtenían las trazas de comunicaciones explícitas en los nodos de cómputo, y generaban grafos dirigidos basados en las trazas. A partir de aqui, los autores usaban los *isomorfismos de subgrafos* para estudiar la traza y transformarla según los diferentes especificaciones de los sistemas de cómputo.

Simon y Wierun [46] proponen el uso de *grafos de tareas concurrentes* (TGC) para realizar modelos de las aplicaciones. Un grafo de tareas concurrentes es un grafo acíclico dirigido cuyos bordes representan la relación de dependencia entre los nodos. Con el fin de predecir el tiempo de ejecución, los autores proponen tener diferentes modelos para calcular el *overhead* en la comunicación, con modelos de rendimiento entre los eventos de comunicación.

En el trabajo de Adve y Vernon [2], consideran las técnicas analíticas para predecir en detalle características de rendimiento de un programa paralelo de memoria compartida para unos datos de entrada definidos. Los autores desarrollan análisis de grafos de tareas deterministas los cuales proveen predicciones de rendimiento para programas paralelos de memoria compartida compuestos de grafos de tareas arbitrarias, una amplia variedad de políticas de planificación, entre otras.

En general, los modelos que presentan los autores de los trabajos descritos anteriormente, son significativamente precisos en sus predicciones, sin embargo crear estos modelos requiere de gran experiencia y consumo de tiempo y no es una tarea fácilmente asumible por usuarios no expertos. Otra consideración a tener en cuenta es que se deben crear tantos modelos como posibles cambios en la entrada de los datos [50]. La complejidad de los sistemas actuales, dificulta el desarrollo de estos modelos analíticos.

### 2.1.3. Simulación

Los modelos estructurales han sido utilizados para las evaluaciones de rendimiento, ya que hacen la descripción funcional de cada componente individual del sistema y su interacciones, tales como modelos detallados de simulación. El uso de simuladores detallados o *"cycle-accurate"* ha sido usado en muchos trabajos de investigación como método de evaluación de rendimiento [6]. Los simuladores detallados son construidos por los fabricantes durante la etapa de diseño de las arquitecturas. Dichos sistemas tienen la ventaja de la automatización de la predicción del rendimiento desde el punto de vista del usuario. La desventaja es que estos simuladores son privados del fabricante o diseñadores y, a menudo no están a disposición

de los usuarios de centros HPC. Además, debido a la captura de todos los eventos de los componentes a simular, las simulaciones pueden asumir hasta 1.000.000 veces más, el tiempo real de ejecución de la aplicación [31].

Para evitar estos costos computacionales, los simuladores se utilizan para simular pocos segundos representativos de la aplicación. Simuladores detallados se limitan al modelo de comportamiento del procesador para el que fueron desarrollados, por lo que no son aplicables a otras arquitecturas. Además, la exactitud de la simulación detallada puede ser cuestionable.

La idea de *simulación estadística*, introducida por Oskin y otros [37], Eeckhout y De Bosschere [16], y Nussbaum [35], son los que constituyen el grupo de *workloads* sintéticos. El enfoque usado en simulación estadística es generar una traza sintética dada por perfiles estadísticos. Estos perfiles estadísticos son atributos del *workload*, tales como distribución del tamaño de los bloques básicos, predicción de saltos, fallos de instrucciones/datos, mezcla de instrucciones, etc, y entonces simular las trazas sintéticas usando un simulador estadístico. Eeckhout y otros [15], mejoran la simulación estadística usando *statistical flow graph* (SFG), como también [19] en la mejora de los modelos de flujo de datos en la memoria dado por la simulación estadística.

La limitación de la simulación estadística es que genera trazas sintéticas en lugar de benchmarks sintéticos. Las trazas sintéticas no se puede ejecutar en simuladores o sistemas reales [25], a diferencia de los benchmarks.

## 2.2. *Frameworks* de Predicción de Rendimiento

El trabajo de Todi y Gustafson[20], HINT o Hierarchical INTegration presenta una herramienta de benchmarking para computadores desarrollado en el *Scalable Computing Laboratory* (SCL) de Ames Laboratory. El objetivo es asignar la aplicación a la curva del HINT benchmark, y usar la curva HINT como referencia para dado un nodo de cómputo predecir el rendimiento de la aplicación. Este trabajo dio a conocer que HINT es un gran conjunto referencia para otros benchmarks, tales como el NAS *parallel*

*benchmark*, SPEC, STREAM y otros.

En el trabajo de Krishnaswamy [27] se presenta un *framework* para la evaluación del rendimiento de sistemas de cómputo usando un conjunto de benchmarks. La metodología que proponen se basa en identificar los parámetros claves de rendimiento, para construir lo que los autores llaman "*vectores de rendimiento*". Un vector de rendimiento es definido como un vector de índices que representan el rendimiento que ofrece un sistema para ejecutar operaciones primitivas. Para medir el vector de rendimiento, los autores proponen un modelo geométrico el cual define el comportamiento del sistema usando los conceptos de puntos de soporte, red de contexto y puntos de apoyo. La metodología es validada experimentalmente con cargas de trabajo secuencial, y con arquitectura Sun UltraSparc.

En los trabajos de Snavely y otros autores [47] y [48], proponen un *framework* para modelar el rendimiento de una aplicación paralela, basados en el rendimiento del procesador y el uso de las redes de interconexión. Ellos predicen el rendimiento por *convolución* de los *perfiles de la aplicación* con la *firma del nodo de cómputo*. Los autores proponen caracterizar el procesador o nodo de cómputo, usando el benchmark MAPS para extraer las operaciones fundamentales de la aplicación. A su vez, los autores obtienen el perfil de la aplicación, a través de simuladores MetaSim y DIMEMAS, para determinar el patrón de acceso a memoria y el tráfico de la red de interconexión, respectivamente.

Marin y otros autores [32], presenta un *toolkit* para predecir el rendimiento de aplicaciones científicas usando modelos parametrizados. El *toolkit* que proponen los autores comprende tres componentes: (1) análisis estático del binario de la aplicación, (2) instrumentación de binarios con medidas de tiempo de ejecución y análisis para caracterizar el comportamiento de la aplicación, y (3) modelar con parámetros del comportamiento de la aplicación (basados en una arquitectura neutral); este modelo se hace a partir de análisis exhaustivo del rendimiento con la información estática y las mediciones dinámicas conjuntamente con una descripción de los recursos de un equipo de destino.

El trabajo de Toomula y otros [53] presenta una metodología para estimar la tasa de fallos de cache de una aplicación. Los autores capturan los

patrones de accesos a memoria de la aplicación, para reproducir el comportamiento de memoria de la aplicación en *memory skeleton program* o programas esqueletos de memoria. Ellos proponen con el método, predecir el comportamiento de memoria de una aplicación, a través de múltiples arquitecturas de memoria con los *performance skeletons*. La metodología que proponen no estima el tiempo total de la aplicación, sin embargo ellos pretenden extender su trabajo a la creación de *performance skeletons* que permitan predecir el tiempo de ejecución de la aplicación.

En el trabajo de Yang y otros [57], los autores proponen ejecutar "trozos de prueba" de aplicaciones sobre una gran variedad de plataformas de nodos de cómputo, de forma que se pueda derivar el tiempo de ejecución total de la aplicación a partir del tiempo de ejecución del trozo ejecutado. Los tiempos de ejecución de estos "trozos de prueba", según los autores, podrían ser almacenados en una base de datos para futuras predicciones. En otras palabras, los autores predicen con su método el tiempo total de ejecución de una aplicación de gran escala en un sistema de destino o sistema candidato, mediante la combinación del rendimiento en un sistema referencia y el rendimiento derivado de una muy breve ejecución de la aplicación. A diferencia de lo que proponen los autores, nuestra metodología no ejecuta "trozos de prueba" de las aplicaciones sobre todos los nodos de cómputo disponibles o candidatos a hacer la predicción. Previamente caracteriza los nodos de cómputo ejecutando $\mu Bs$, cuyo tiempo de duración es muy breve y almacenamos la información de todos los perfiles de nodos de cómputos en una base de datos para comparaciones con las características de la aplicación. La aplicación la ejecutamos en un nodo de referencia, en donde extraemos las características de comportamiento de la aplicación que consideramos significativas para ser comparadas y buscar su similitud con los perfiles de rendimiento almacenados en la base de datos.

En el trabajo de Bell y John [5], presenta un "*framework*" para la síntesis automática de benchmarks en miniatura de aplicaciones. La idea principal de este método es capturar la estructura principal de un programa usando la teoría de simulación estadística y generar en código C e instrucciones en lenguaje ensamblador que modele de forma precisa los atributos o características de la carga de trabajo. Sin embargo, el trabajo de los autores

esta enfocado en predecir el consumo de energía en los procesadores, y el modelo es dependiente de la arquitectura.

En el trabajo de Hoste y otros [22], los autores proponen el uso de las métricas de rendimiento de los *benchmarks* SPEC CPU2000 para la proyección del rendimiento de las aplicaciones. Su objetivo era medir un número de características independientes de la microarquitectura dadas por la aplicación de interés, y relacionar éstas características con las de los SPEC 2006. Basados en la similitud de la aplicación de interés con los benchmarks SPEC, los autores hacen la predicción del rendimiento de la aplicación. Los autores proponen y evalúan tres enfoques: Normalización, Análisis de componentes principales y Algoritmo genético, para transformar los datos de las características independientes de la microarquitectura en un espacio de referencia en el que la distancia relativa es una medida de las diferencias de rendimiento relativo. La principal diferencia fundamental entre la propuesta de Hoste y otros, y nuestra metodología es que el rango de medición y los grupos de indicadores no sólo se basa en las propiedades de la aplicación, sino también se basa en las propiedades del sistema al construir los $\mu Bs$. Estas características son las que definen los perfiles de rendimiento de los nodos de cómputo en nuestra metodología, las cuales son similares a las características extraídas de la aplicación.

El trabajo de Tikir y otros, en [52], usan algoritmos genéticos para modelar el rendimiento de aplicaciones limitadas por memoria. Los autores proponen un esquema para predecir el rendimiento de aplicaciones HPC basados en los resultados de *benchmarks MultiMAPS*. *benchmarks MultiMAPS* mide el máximo ancho de banda de operaciones de memoria alcanzable por un nodo de cómputo, como una función de fallos de cache [23]. El algoritmo genético es usado para "aprender" los anchos de banda como una función de fallos de cache por nodo de cómputo, dados por la ejecución del *benchmarks MultiMAPS*. El enfoque de los autores requiere simular diferentes tamaños de la jerarquía de memoria, para entender las características de la aplicación sobre la cache. Nuestra metodología no necesita simulación, utiliza la ejecución de un conjunto de $\mu Bs$ y la recolección de los perfiles de rendimiento de cada uno de los nodos de cómputo para futuras comparaciones. Por otro lado, la propuesta que hacen en [52], esta directamente

relacionado con MultiMaps como único conjunto de benchmarks. Nuestro enfoque permite caracterizar cualquier conjunto de benchmarks

El *framework* que presenta [25], es un trabajo con el objeto de clonar el rendimiento de la aplicación en diminutos benchmarks. La idea de los autores es obtener características de la aplicación, basadas en la captura del control de flujo por Grafos de Flujos Estadísticos (*Statistical Flow Graphs SFG)* descritos en [15]. El benchmark generado como clon sintético posee las mismas características de rendimiento/consumo energético de la aplicación. Los autores modelan las características independientes de la micro-arquitectura, tales como accesos a memoria y predicción de saltos con la finalidad de capturar el comportamiento de la localidad de datos y el control de flujo del programa. La idea es que una vez generados los benchmarks, sean ejecutados en simuladores para poder hacer la predicción del rendimiento. Los tiempos de comunicación son siempre aproximados, no tiene en cuenta los tiempos de solapamiento de las ejecuciones y por ende uno de los factores que sobre todo incide en el tiempo de ejecución.

En [38] presentan un método para modelar el rendimiento, consiste en aproximar el tiempo de ejecución de una aplicación sobre un computador paralelo como una combinación lineal de predictores. Cada predictor es la latencia obtenida de unos microkernels o características del sistema, con todos los microkernels del conjunto HPCC benchmark [24]. El modelo tiene la desventaja de que no tienen en cuenta las diferencias de la jerarquía de memoria entre los diferentes diseños de nodos de cómputo, de manera que la precisión para aplicaciones dominadas por accesos a memoria se ve comprometida.

En el trabajo de Sharkawi y otros autores [42], presentan un esquema para proyectar el rendimiento de aplicaciones HPC usando cargas de trabajo "workloads" sustitutas dadas por los SPEC CFP2006. En su trabajo, al igual que el nuestro ellos emplean un nodo de cómputo de referencia en donde ejecutan la aplicación y los benchmarks SPEC CPF2006. Los autores caracterizan la aplicación y los benchmarks SPEC utilizando la información de los contadores hardware. Una vez caracterizados los benchmarks y la aplicación, usan las métricas recolectadas para encontrar la similitud de comportamiento y así seleccionar el SPEC que mas se le parezca en

comportamiento a la aplicación. En nuestra metodología no empleamos los contadores de rendimiento hardware (ya que son diferentes en diferentes procesadores y dependientes de los fabricantes de procesadores), extraemos información de características de rendimiento, como principales en el modelo del comportamiento de la aplicación. Las mismas características son usadas para desarrollar los $\mu Bs$. Al igual que lo que proponen los autores, nosotros buscamos entonces similitud entre las características del perfil de rendimiento de los nodos de cómputo y características de rendimiento de la aplicación.

### 2.2.1. Caracterización de las Fases en las Aplicaciones

Existe un gran número de trabajos relacionados con la caracterización de las fases en las aplicaciones. En esta sección, se discutirán los más significativos con respecto a nuestro trabajo de investigación.

#### Características dependientes versus independientes de la micro-arquitectura

Algunos trabajos para identificar el comportamiento de las fases de una aplicación se realizan analizando la dependencia del comportamiento con la microarquitectura. Por ejemplo, Balasubramonian y otros [4], adquieren datos de los Ciclos Por Instrucción (CPI) y los fallos de cache para cada una de las fases detectadas. El trabajo de Duesterwald y otros [14], obtienen las métricas de las Instrucciones Por Ciclo (IPC), la tasa de fallos de cache junto con los fallos de los predictores de saltos. Una preocupación en la detección de la fase dependiente de la microarquitectura, es que una vez que el comportamiento de la fase está siendo estudiado, puede afectar a la métrica que se mide (dependiente de la microarquitectura), lo que potencialmente lleva al problema que no está claro si la variación del tiempo de ejecución de la fase (comportamiento) es el resultado del comportamiento natural de los programas o es una consecuencia directa por la recolección de las métricas en cada una de las fases observadas.

Una alternativa al enfoque anterior es obtener métricas que sean independientes de la micro-arquitectura para inferir sobre el comportamiento

de las fases de la aplicación. El trabajo de Sherwood y otros [44], usan los vectores de bloques básicos (*Basic Block Vectors BBVs*) con el fin de estudiar los bloques básicos ejecutados; a su vez Lau y otros [28], muestran que los BBVs tienen una buena correlación con el rendimiento de las aplicaciones. En otros trabajos tambien son usadas métricas como las direcciones de memoria y las distancias en reutilizar los datos [43], estructuras del control de flujo del programa, tales como *loops* y *funciones y procedimientos*, colección de características del programa tales como la mezcla de instrucciones, ILP, y los patrones de accesos a memoria, entre otras [17] y [29].

**Predicción del comportamiento de las fases**

Un aspecto importante a relacionar con nuestro trabajo, es el poder predecir el comportamiento de las fases. Sherwood y otros [45], proponen la última fase utilizada RLE y predictores de fases Markov. En el trabajo de Lau y otros [30], añaden contadores a los predictores de fases, junto con las actualizaciones condicionales dados por el trabajo de Vandeputte y otros [54]. Este último trabajo [54], introduce cuatro características del comportamiento de las fases del programa: la escala del tiempo, el número de fases, la variabilidad de las fases y la predictibilidad secuencial y transitoria de las fases. Los autores hacen uso de superficies como forma de describir la complejidad del comportamiento de las fases, en función de las cuatro características propuestas.

# Capítulo 3

# Metodología Propuesta

## 3.1. Comportamiento Prestacional

El objetivo de este trabajo es presentar una metodología para *estimar* las posibles diferencias en el comportamiento de aplicaciones paralelas de memoria compartida sobre diferentes nodos de cómputo multicore [10].

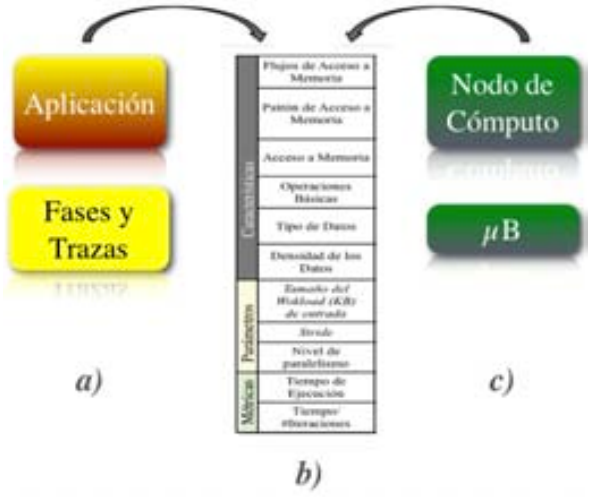



Figura 3.1: Propuesta para la predicción del rendimiento.

La idea principal de la metodología es ordenar/comparar el comportamiento de los nodos de cómputo que pueden ser candidatos a ejecutar la aplicación en función del rendimiento esperado de la aplicación. En la Figura 3.1 se presenta la propuesta para la estimación del rendimiento: ***a)*** a partir de la ejecución de la aplicación y la identificación de las fases representativas junto con las trazas, obtener ***b)*** información que caracterice las fases en términos de rendimiento y ***c)*** obtener información que caracterice los nodos de cómputo a través de $\mu Bs$ y almacenar múltiples comportamientos. Como se observa en la Figura 3.1 la Tabla ***b)*** posee la caracterización tanto de la fase de la aplicación como del sistema, de forma tal que si se tienen un número significativo de comportamientos del sistema, se buscará por aquel comportamiento del $\mu B$ que sea similar en rendimiento a la fase de la aplicación.

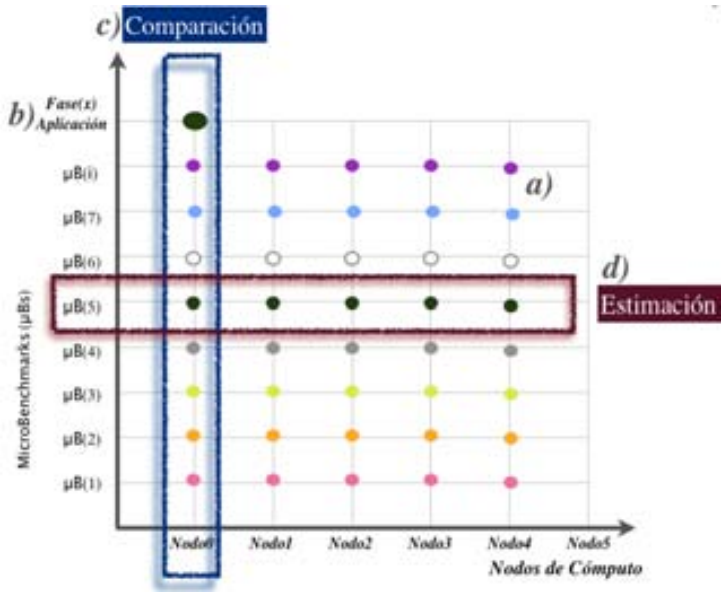



Figura 3.2: Propuesta para la obtención de la predicción del rendimiento.

La Figura 3.2 presenta la propuesta para la obtención de la predicción, el

eje X representa las distintas configuraciones o diseños de nodos de cómputo en donde se desea estimar el rendimiento de cualquier aplicación; el eje Y representa los múltiples $\mu Bs$ con diferentes comportamientos a ser ejecutados en todos los nodos de cómputo; se tiene entonces en la Figura 3.2: ***a)*** la ejecución previa y obtención de todos los comportamientos dados por los $\mu Bs$ sobre diferentes nodos de cómputo; ***b)*** la ejecución de la aplicación sobre un nodo de referencia y la obtención de las fases representativas con su respectiva información de caracterización; ***c)*** La comparación entre las características de los $\mu Bs$ ejecutados sobre el nodo referencia y la fase de aplicación, con el objeto de seleccionar el $\mu B$ que represente la aplicación por su comportamiento similar en términos de rendimiento; y ***d)*** la estimación que corresponde a seleccionar los $\mu Bs$ obtenidos en ***c)*** y estimar el comportamiento sobre los diferentes nodos de cómputo.

La metodología considera entonces cuatro etapas: (I) *obtención de los perfiles de rendimiento de los nodos de cómputo*, (II) *caracterización de las fases de la aplicación*, (III) *proceso de comparación* y (IV) *proceso de estimación*. En la Figura 3.3 se presenta el Esquema General de la Metodología.

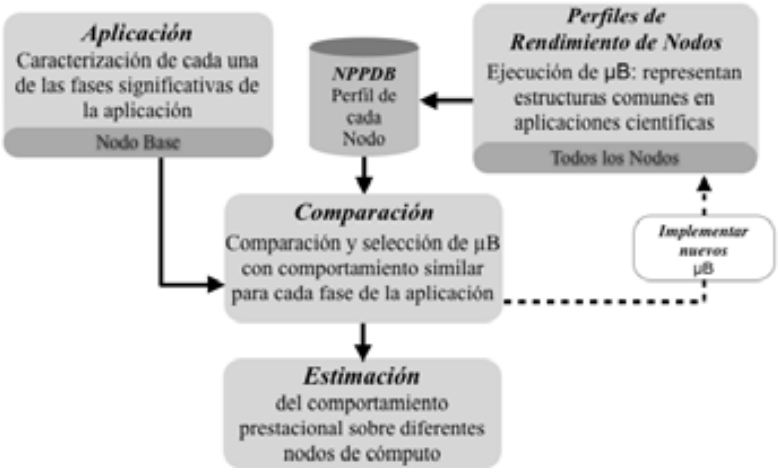



Figura 3.3: Esquema General: *metodología para estimar el rendimiento*

La estimación pretende dar información que permita comparar el comportamiento de una aplicación sobre diferentes nodos de cómputo multicore. Se busca ordenar los recursos o sistemas de cómputo considerando el tiempo de ejecución como la característica de prestaciones relevante. Para ello se consideran características cualitativas y cuantitativas del sistema y se estima su impacto en el tiempo de ejecución [9].

La idea es dar información para la toma de decisiones cuando se quiere comparar y seleccionar un sistema de cómputo, en este caso es importante obtener criterios para la comparación en un tiempo corto (mucho menor al que supone ejecutar toda la aplicación en cada uno de los nodos a comparar), que permitan seleccionar un nodo de cómputo *apropiado* para una aplicación, teniendo información sobre su influencia en el rendimiento.

Consideramos aplicaciones que resuelven problemas de cómputo científico de álgebra matricial. Se analizan las versiones paralelas de las aplicaciones científicas. Se ha seleccionado este campo de aplicación porque son núcleos computacionales comunes en las aplicaciones científicas que utilizan bibliotecas numéricas de altas prestaciones. Se analizan las versiones paralelas implementadas con el modelo de programación paralela OpenMP [36] (estructura de paralelismo tipo Fork-Join), y para unos ciertos valores de entrada o *workload*. El análisis se realiza para diversos nodos de cómputo multicore candidatos para la estimación (Nodos Target), junto con un nodo de cómputo de referencia (Nodo Base), donde ejecutamos la aplicación.

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.

Cada $\mu B$ al ejecutarse, proporciona una información que nos da un perfil de rendimiento particular. Los $\mu B's$ son ejecutados con diferentes workload o parámetros de entrada (tales como tamaños de datos, *stride*). Los *perfiles de rendimiento* obtenidos son almacenados en una base de datos que denominamos Node Performance Profile Data Base (NPPDB).

Para obtener la *caracterización de la aplicación*, ésta 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 (tales como tamaño,

stride, tipo de datos, peso porcentual de la fase). Es importante mencionar que los sistemas en donde serán ejecutadas las aplicaciones, son entornos de cómputo dedicados, y por lo tanto cada *workload* de la aplicación usará el total de los recursos del sistema.

Basándonos en la información de rendimiento almacenada en la NPPDB y en la información obtenida para cada fase en la caracterización de la aplicación, hacemos un proceso de comparación para seleccionar aquellos *perfiles de rendimiento*, obtenidos sobre el nodo base al ejecutar los micro-benchmarks ($\mu B$), que tienen un comportamiento similar con cada una de las fases representativas de la aplicación. En este paso es necesario hacer un ajuste de parámetros con el fin de obtener el punto que permite asociar o correlacionar el *workload* del $\mu B$ y los datos de entrada o *workload* de cada fase de la aplicación.

Cuando se encuentra similitud, el $\mu B$ seleccionado será el candidato para representar a la fase en el *proceso de estimación*.

En el caso de no ser posible encontrar similitud entre los *perfiles de rendimiento* obtenidos con los $\mu B's$ y la información de *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 los *perfiles de rendimiento* identificados en el *proceso de comparación* correspondientes a los nodos candidatos en la NPPDB, y estimar el comportamiento de la aplicación. La estimación para cada nodo se calculará utilizando el tiempo/#Iteraciones de cada $\mu B$ para el *workload* seleccionado en cada una de las fases y el peso de dicha fase. A continuación describimos el esquema de cada una de las etapas de la metodología con mayor detalle.

### 3.1.1. Perfiles de rendimiento de los nodos de cómputo

El objetivo es caracterizar el nodo de cómputo frente a estructuras comunes encontradas en aplicaciones.

*Benchmarking* es el proceso de ejecutar programas específicos sobre un nodo de cómputo concreto o sistemas de cómputo y medir el rendimiento.

Esta técnica claramente provee una evaluación precisa del rendimiento del nodo de cómputo para ese "*workload*"[39]. Los *microbenchmarks*, es decir núcleos de cómputo muy pequeños y de corta duración, son usados para obtener medidas cuantitativas de rendimiento sobre los nodos de cómputo en clusters [49].

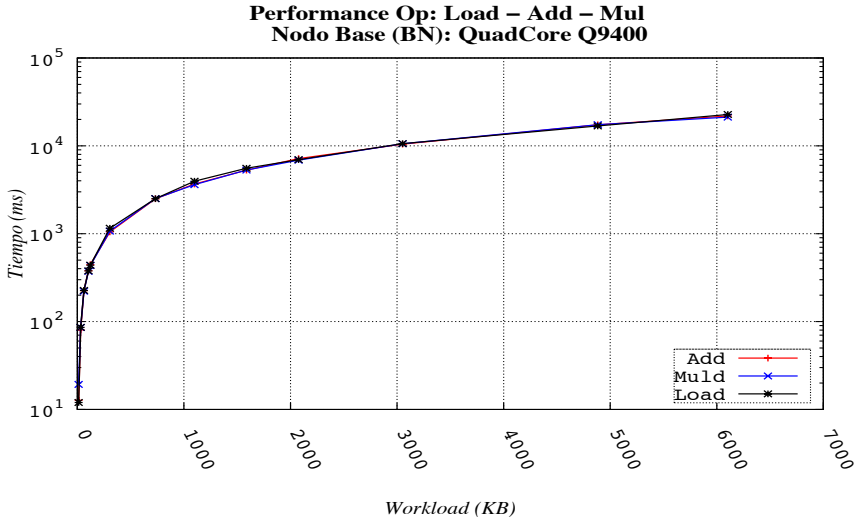
Para obtener el perfil de rendimiento o comportamiento de los nodos frente a determinadas estructuras de cómputo, se ejecutan $\mu Bs$ con diferentes *workloads* que permiten estresar el nodo de cómputo y caracterizar el perfil de comportamiento en un rango de datos. Analizar este perfil nos permite identificar las zonas en las que podemos predecir su comportamiento para un *workload* en concreto, es decir, nos permite seleccionar los rangos de datos de entrada en los que el comportamiento es predecible.

Los $\mu Bs$, al igual que las aplicaciones que queremos predecir (aplicaciones que resuelven problemas de cómputo científico de álgebra matricial), son programas paralelos de memoria compartida implementados en C y modelo de programación paralela OpenMP. Cada ejecución de un $\mu B$ nos da una información del comportamiento o *perfil de rendimiento* del nodo de cómputo.

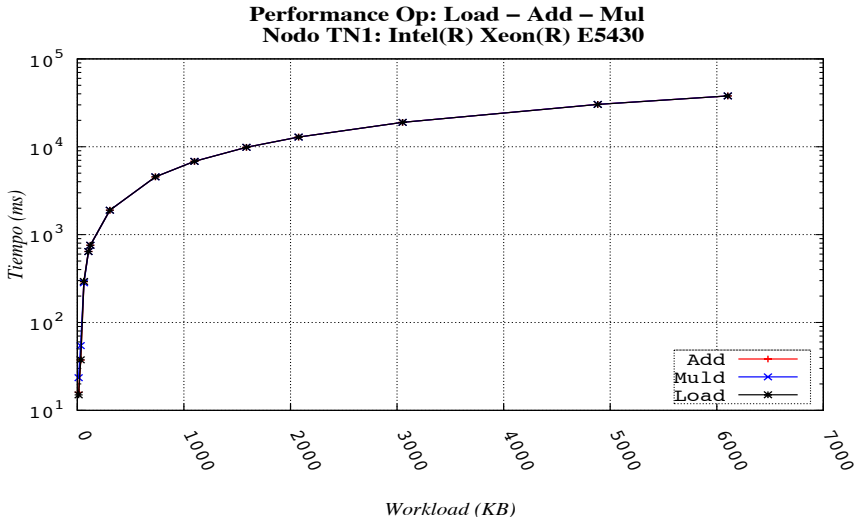
Los $\mu B's$ son clasificados de acuerdo a los factores que limitan el rendimiento en las aplicaciones: Limitadas por Memoria, Limitadas por Cómputo y Limitadas por Entrada/Salida.

La validación experimental de este trabajo se ha enfocado en la predicción de aplicaciones limitadas por los accesos a memoria. Se analizan aplicaciones con patrones regulares de acceso a memoria. Los $\mu Bs$ que se han diseñado tienen características para estresar la jerarquía de memoria. El sistema de jerarquía de memoria en los nodos multicore representa un factor significativo en el comportamiento de las aplicaciones paralelas de memoria compartida [7].

Los accesos a memoria estresan a la jerarquía de memoria representando el principal cuello de botella en el rendimiento de las aplicaciones paralelas de memoria compartida. La metodología presentada en este trabajo está basada en el estudio del patrón de accesos a memoria, en aplicaciones limitadas por memoria. Para este tipo de aplicaciones el patrón de accesos a memoria es información suficiente para obtener el comportamiento.

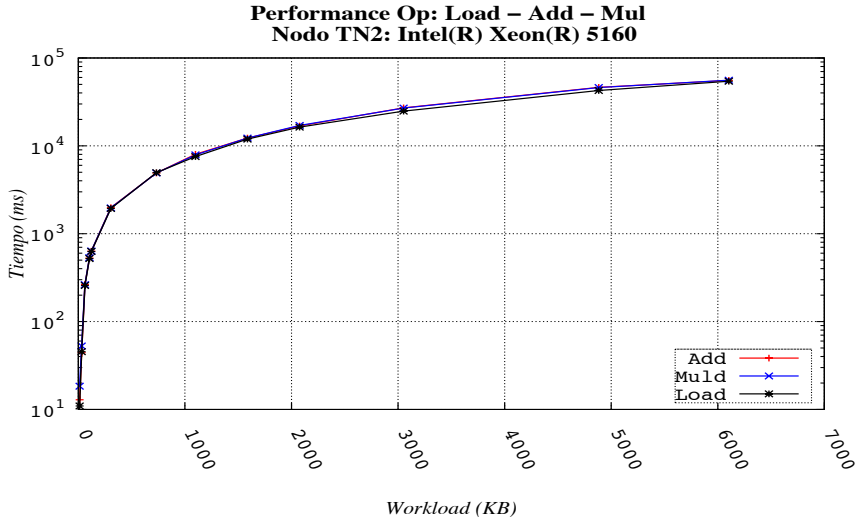



(a) BN: Intel QuadCore Q9400

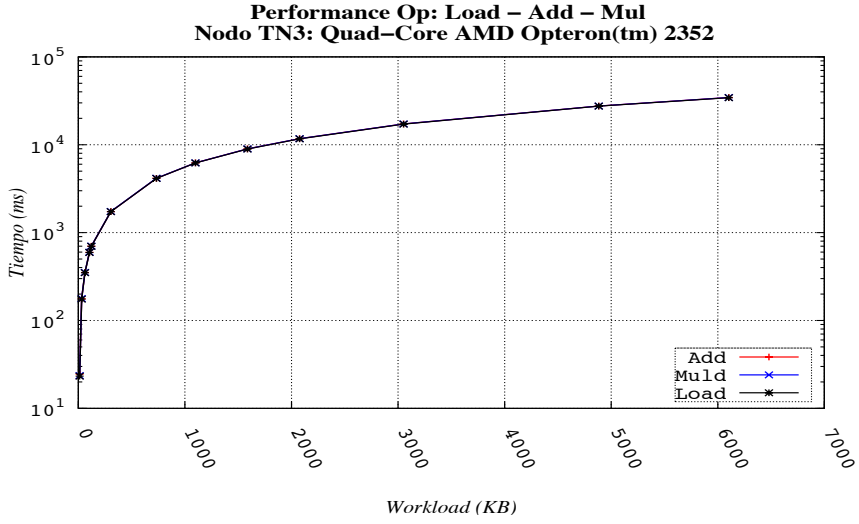



(b) TN1: Intel Xeon 5160

Figura 3.4: Tiempo de ejecución de 3 $\mu Bs$: operaciones de suma **Add**, operaciones de producto **Mul** y operaciones de acceso a memoria **Load**, en los nodos de cómputo que denominamos BN y TN1.




(a) TN2: Intel Xeon E5430




(b) TN3: Intel AMD QuadCore 2352

Figura 3.5: Tiempo de ejecución de 3 $\mu Bs$: operaciones de suma **Add**, operaciones de producto **Mul** y operaciones de acceso a memoria **Load**, en los nodos de cómputo que denominamos TN2 y TN3.

Las Figuras 3.4 y 3.5 presentan la relación entre el impacto que tiene el tiempo de acceso a memoria (*Loads*) y las operaciones Muld/Sumd en cuatro diferentes diseños de nodos de cómputo[1].

El experimento se realizó con un $\mu B$ con las siguientes características: operaciones de suma Sumd en punto flotante de un vector con k elementos del tipo doble. A continuación se muestra la operación básica (3.1) y en la Tabla 3.6 su correspondiente implementación en lenguaje C:

$$v = \sum_{i=0}^{n} a_i \tag{3.1}$$

Se compila el código mostrado en la Tabla 3.6, y se ejecuta 30 veces para posteriormente extraer el promedio de cada ejecución. A su vez, el código es compilado con la variable -S para analizar el código ensamblador. A continuación se muestra el código ensamblador del núcleo.

```
                    ...
for (i=0; i<N; i+=4 {
                v += a[i+0];
                v += a[i+1];
                v += a[i+2];
                v += a[i+3];
                }
                    ...
```

Tabla 3.6: Implementación del $\mu B$, un-rolling de 4 elementos por iteración

En la Tabla 3.7 se presenta el código ensamblador, en la cual se puede observar las operaciones *addsd* (suma en punto flotante) y *movsd* (acceso a memoria). Este experimento es el que refleja las gráficas de las Figuras 3.4 y 3.5 con la etiqueta **Add**. Para los siguientes experimentos hemos repetido el mismo procedimiento, pero hemos sustituido todas las operaciones *addsd* por *muldsd* para las gráficas de operaciones de multiplicación **Mul**. A su

[1]La descripción de cada configuración es presentada en el Capitulo 4 "Validación Experimental"

```
....
.L5:
        movsd   (%rdx,    %r10),    %xmm0
        addq    $1,       %r11
        movsd   (%rax,    %r10),    %xmm1
        addq    %rcx,     %r10
        movapd  %xmm0,    %xmm2
        cmpq    %rsi,     %r11
        addsd   %xmm1,    %xmm2
        addsd   %xmm2,    %xmm0
        addsd   %xmm0,    %xmm1
        addsd   %xmm1,    %xmm3
        jl      .L5
....
```

Tabla 3.7: Código ensamblador para arquitectura x86_64 de la implementación del $\mu B$ de la Tabla 3.6

vez, para las operaciones de acceso a memoria **Load**, se han sustituido *addsd* por *movsd*. Las tres operaciones, para diferentes tamaños de vectores (*worksize*), hemos ejecutado en cuatro configuraciones de nodos multicore diferentes. Los resultados se aprecian en las Figuras 3.4 y 3.5.

Se puede observar en las Figuras 3.4 y 3.5 que para unos valores de entrada superiores a un cierto tamaño de memoria (que coincide con el tamaño de la cache L1 para todas las configuraciones), el comportamiento dado por el $\mu B$ que ejecuta sólo Loads es similar en comportamiento a los $\mu Bs$, que además, también ejecutan las operaciones de Muld/Sumd. Si la implementación de la aplicación de memoria compartida no hace uso de suficiente memoria ("aplicaciones de juguete", que sólo usen cache L1 de la jerarquía de memoria), la metodología no garantiza una buena predicción.

Uno de los objetivos a alcanzar por este estudio es la selección del nodo de cómputo apropiado para una aplicación paralela de memoria compartida. Esta selección viene guiada a partir de los *perfiles de rendimiento* de los nodos de cómputo. Los perfiles son obtenidos usando $\mu B's$, los cuales permiten el análisis de rendimiento para diferentes cargas de trabajo o "*workloads* ". Tal y como hemos mencionado, nuestro enfoque considera

*workloads* que estresan la memoria, con diferentes patrones de acceso a memoria regulares que reflejan el comportamiento de un conjunto significativo de aplicaciones paralelas de memoria compartida.

La Figura 3.8 presenta el esquema para la obtención de los perfiles de rendimiento de los nodos de cómputo.

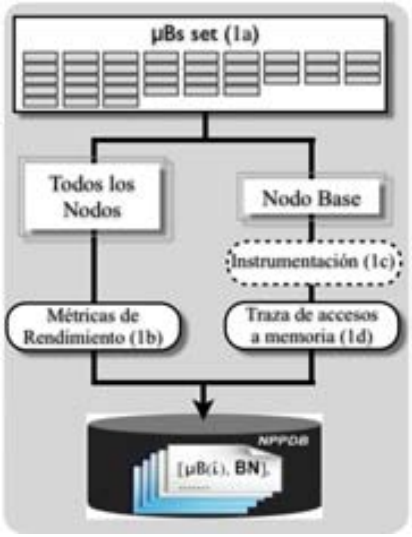



Figura 3.8: Esquema para la obtención de los *perfiles de rendimiento*

### Implementación de Microbenchmarks: $\mu Bs$ *set (1a)*

Se implementan un conjunto de microbenchmarks ($\mu Bs$), que son programas sintéticos de prueba de corta duración, como hemos mencionado, estos programas están desarrollados en C y con modelo de programación OpenMP. Cada $\mu B$ tiene a su vez un conjunto de parámetros con el objeto de caracterizar el rendimiento de los nodos de cómputo. Los primeros $\mu Bs$ se han desarrollado bajo diferentes características de los patrones de acceso a memoria, mostrando el comportamiento del nodo en términos de rendimiento. Los $\mu Bs$ son diseñados de acuerdo a patrones y comportamientos

previamente estudiados en las aplicaciones científicas.

Con los $\mu Bs$ definimos características tales como las que se presentan en la Tabla 3.9. A su vez, la Tabla muestra una descripción detallando las diferencias de algunos $\mu Bs$ que hemos diseñado y las respectivas métricas que utilizamos para describir el perfil de rendimiento de los nodos de cómputo [8]. Las características son las entradas o requisitos que se definen al ser construidos los $\mu Bs$, las métricas son los valores resultado que describen los diferentes comportamientos o perfiles de rendimiento.

Tenemos dos escenarios posibles para la obtención de información de los $\mu Bs$:

- los que nosotros diseñamos e implementamos, los cuales comprenden todas las características que describen el $\mu B$, estos $\mu Bs$ son ejecutados con diferentes parámetros, en todos los nodos, y se obtienen las medidas de rendimiento que se almacenan junto con su descripción en la base de datos, y
- por otro lado, cuando identificamos un núcleo de una aplicación para el que no tenemos un $\mu B$ similar intentamos crear un nuevo $\mu B$, caracterizándolo y ejecutándolo para obtener las medidas de rendimiento y almacenando la información en la base de datos

Un perfil de rendimiento es una lista de valores que caracterizan el $\mu B$ junto con valores promedios de las medidas de rendimiento (tiempo de ejecución y tiempo/#iteraciones), obtenidos en la ejecución para diferentes valores de los parámetros. La idea es utilizar información cualitativa y cuantitativa para analizar el comportamiento del nodo de cómputo.

- Características cualitativas tales como: tener información de cómo afecta los diferentes flujos de acceso a memoria, el patrón de acceso de cada uno de ellos, el tipo de dato, las operaciones básicas, densidad de los datos (densos o dispersos), forma en que acceden a memoria *privada* (P) o *compartida* (C).
- Información cuantitativa, encontramos dos tipos:

**Parámetros de entrada** tamaño del *workload* de entrada, el tamaño del *stride* y el nivel de paralelismo

**Métricas de salida** Tiempos de ejecución y Tiempo/#Iteraciones.

Para obtener los perfiles de rendimiento, los $\mu Bs$ son ejecutados previamente en todos los nodos de cómputo disponibles (BN y TNi) para diferentes tamaños de datos de entrada.

| | µB ID | ... | ***mbw1c***(*k,s,th*) | ***mbw2hc***(*k1,k2,s1,s2,th*) | ***mbw2pc***(*k1,k2,s1,s2,th*) | ***mbw2r***(*k1,k2,s,r,th*) | ... |
|---|---|---|---|---|---|---|---|
| Características | Flujos de Acceso a Memoria | | *stream* ***a*** | *stream* ***a***, *stream* ***b*** | *stream* ***a***, *stream* ***b*** | *stream* ***a***, *stream* ***b*** | |
| | Patrón de Acceso a Memoria | | ***a***:*Stride-s secuencial* | ***a***:*Stride-s1 secuencial* ***b***:*Stride-s2 secuencial* | ***a***:*Stride-s1 secuencial* ***b***:*Stride-s2 secuencial* | ***a***:*Stride-s secuencial* ***b***:*Stride-r aleatorio* | |
| | Acceso a Memoria | | ***a***:*compartido* | ***a***:*compartido* ***b***:*compartido* | ***a***:*compartido* ***b***:*privado* | ***a***:*compartido* ***b***:*privado* | |
| | Operaciones Básicas | | $\sum_{i=0}^{l} \boldsymbol{a}_i$ | $\sum_{i=0}^{l} (\boldsymbol{a}_i \times \boldsymbol{b}_i)$ | $\sum_{i=0}^{l} (\boldsymbol{a}_i + \boldsymbol{b}_i)$ | $\sum_{i=0}^{l} (\boldsymbol{a}_i + \boldsymbol{b}_i)$ | |
| | Tipo de Datos | | ***a***: doble | ***a***: doble ***b***: doble | ***a***: float ***b***: float | ***a***: doble ***b***: doble | |
| | Densidad de los Datos | | ***a***: denso | ***a***: denso ***b***: denso | ***a***: denso ***b***: denso | ***a***: denso ***b***: nodenso | |
| Parámetros | *Tamaño del Wokload (KB) de entrada* | | ***k*** | ***k1,k2*** | ***k1,k2*** | ***k1,k2*** | |
| | *Stride* | | ***s*** | ***s1, s2*** | ***s1, s2*** | ***s,r*** | |
| | Nivel de paralelismo | | ***th*** | ***th*** | ***th*** | ***th*** | |
| Métricas | Tiempo de Ejecución | | | | | | |
| | #Iteraciones | | | | | | |
| | Tiempo/ #Iteraciones | | | | | | |

Tabla 3.9: Descripción de algunos $\mu Bs$ (serán usados en la Validación Experimental).

En la Tabla 3.9, se pueden observar algunas descripciones de los $\mu Bs$ que serán utilizados en la validación experimental:

- $\mu B$: ***mbw1c****(k,s,th)*, tiene 3 parámetros que iremos cambiando para tomar valores en ejecución, el tamaño del *workload* (k), el *stride* (s) y el nivel de paralelismo o número de threads (th). El $\mu B$ hace una operación básica compuesta por un flujo de acceso a memoria (*stream*), que realiza una lectura y una suma. El dato es leído desde el flujo de acceso a memoria *stream* ***a***, con ***k*** elementos de tipo doble (8B), accediendo con paso de salto o *stride* **s** secuencial. El flujo de acceso a memoria se produce en una zona de memoria compartida (C) para todos los *threads*.

- $\mu B$: ***mbw2hc****(k1,k2,s1,s2,th)* tiene 5 parámetros que iremos cambiando para tomar valores en ejecución, para cada flujo de acceso: *workload* (k1 y k2) y *stride* (s1 y s2), a su vez, el parámetro de nivel de paralelismo o numero de threads (th). El $\mu B$ tiene dos operaciones básicas: producto y suma, entre elementos de los flujos de acceso a memoria *stream* ***a*** y *stream* ***b***. El dato es leído desde cada uno de los flujos de acceso a memoria, y cada flujo de acceso a memoria tiene tamaño **k** elementos, cada uno del tipo doble (8B). El flujo de acceso a memoria ***a*** tiene un *stride* variable **s1**, mientras que el flujo de acceso a memoria ***b*** se accede con *stride* variable **s2**. Los flujos de acceso a memoria son de zonas de memoria compartidas para todos los *threads*.

- $\mu B$: ***mbw2pc****(k1,k2,s1,s2,th)* es similar al anterior, su diferencia es que la operación es la suma acumulada de cada flujo de acceso a memoria, y el tipo de datos de cada flujo corresponde al tipo float (4B).

- $\mu B$: ***mbw2r****(k1,k2,s1,r,th)* es similar al anterior, el flujo de acceso a memoria *stream* ***a***, accede con *stride* variable **s1** y el flujo de acceso a memoria *stream* ***b*** accede con *stride* aleatorio ($r$).

Hemos comenzado por analizar un conjunto reducido de aplicaciones paralelas para implementar los $\mu Bs$; pero la idea es, que si se identifican nuevos comportamientos que no son similares a los encontrados en la base de datos NPPDB, analizar estos nuevos patrones, e implementar nuevos y apropiados $\mu Bs$ que contemplen estos nuevos comportamientos. Cada nuevo

$\mu B$ generará nuevos perfiles de rendimiento e incrementará el conocimiento de la NPPDB.

| | µB ID | ***mbw1c***(*k,s,th*) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Características | Flujos de Acceso a Memoria | *stream* ***a*** | | | | | | | |
| | Patrón de Acceso a Memoria | ***a:****Stride-s secuencial* | | | | | | | |
| | Acceso a Memoria | ***a:****compartido* | | | | | | | |
| | Operaciones Básicas | $\sum_{i=0}^{l} \boldsymbol{a_i}$ | | | | | | | |
| | Tipo de Datos | ***a***: doble | | | | | | | |
| | Densidad de los Datos | ***a***: denso | | | | | | | |
| Parámetros | *Tamaño del Wokload (KB) de entrada* | ... | ***k=4000*** | ... | ***k=8000*** | ... | ***k=16000*** | ... | |
| | *Stride* | | ***s=8*** | | ***s=64*** | | ***s=1024*** | | |
| | Nivel de paralelismo | | ***th=4*** | | ***th=4*** | | ***th=4*** | | |
| Métricas | Tiempo de Ejecución (s) | | ***1.6763*** | | ***4.5790*** | | ***11.3945*** | | |
| | #Iteraciones (xE6) | | ***399*** | | ***178*** | | ***323*** | | |
| | Tiempo/#Iteraciones (µs) | | ***0.0042*** | | ***0.0257*** | | ***0.0353*** | | |

Tabla 3.10: Descripción del $\mu B$: *Mbw1c* para diferentes *stride* y *workloads*.

En la Tabla 3.10, se presenta la información de caracterización del $\mu B$: ***mbw1c***, para diferentes valores de entrada (los cuales se mostrarán con mucho más detalle en la siguiente Figura 3.11).

La Figura 3.11 **(a)** presenta los *perfiles de rendimiento* obtenidos por el $\mu B$: *Mbw1c* ejecutados en el Nodo Base (BN). El eje $\boldsymbol{x}$ representa el tamaño de los datos de entrada o *workload* (KB) y el eje $\boldsymbol{y}$ representa el tiempo de ejecución ($\mu$s). Se observa como varían los *perfiles de rendimiento* al variar los valores tanto del *workload* como del *stride*. Este tipo de información es el que se utiliza para comparar con el análisis de la aplicación.

En la parte inferior izquierda de la Figura 3.11 **(b)** se presenta el esquema del nodo de cómputo BN o nodo referencia. El nodo BN posee 4 núcleos y al ejecutar el $\mu B$ se obtienen los valores que caracterizan los *per-*

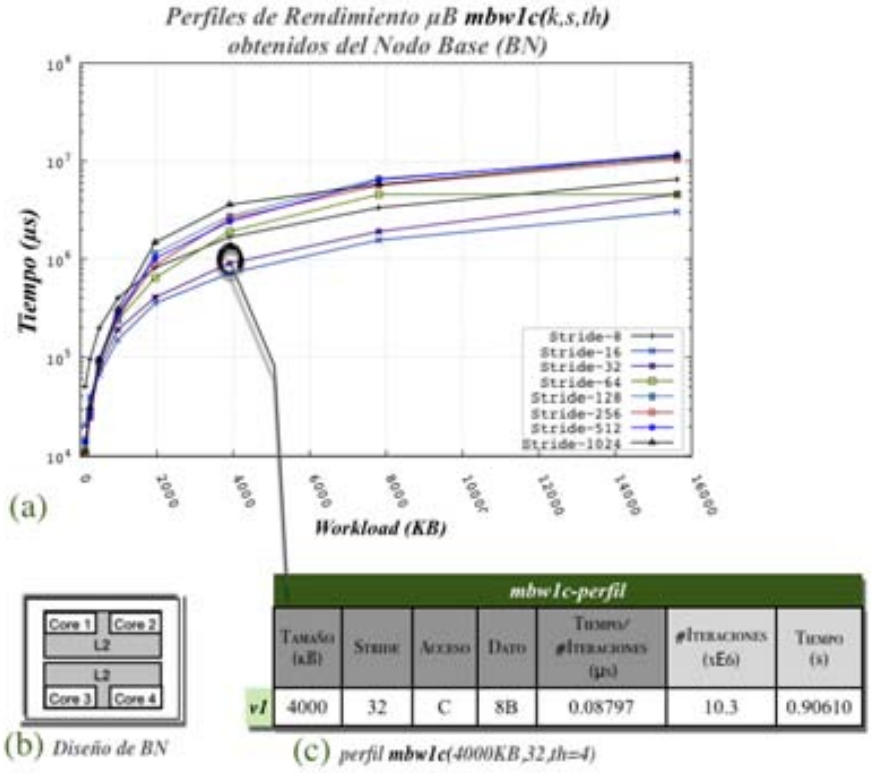



Figura 3.11: Perfiles de Rendimiento $\mu B$: *Mbw1c* del Nodo Base (BN) para diferentes *stride* y *workloads*.

*files de rendimiento* del nodo, ejecutados en 4 *threads* o hilos de ejecución. La descripción en detalle de los diseños de nodos de cómputo se presenta en el capítulo 4.

En la parte inferior derecha de la Figura 3.11 **(c)** se muestra un ejemplo de la información que se almacena en la base de datos como *perfil de rendimiento* para cada uno de los tamaños del *workload* de entrada, para el flujo de acceso a memoria *stream* ***v*** la cual se describirá en detalle en los siguientes apartados.

Para analizar el impacto del *workload*, la Figura 3.12 **(a)** presenta el

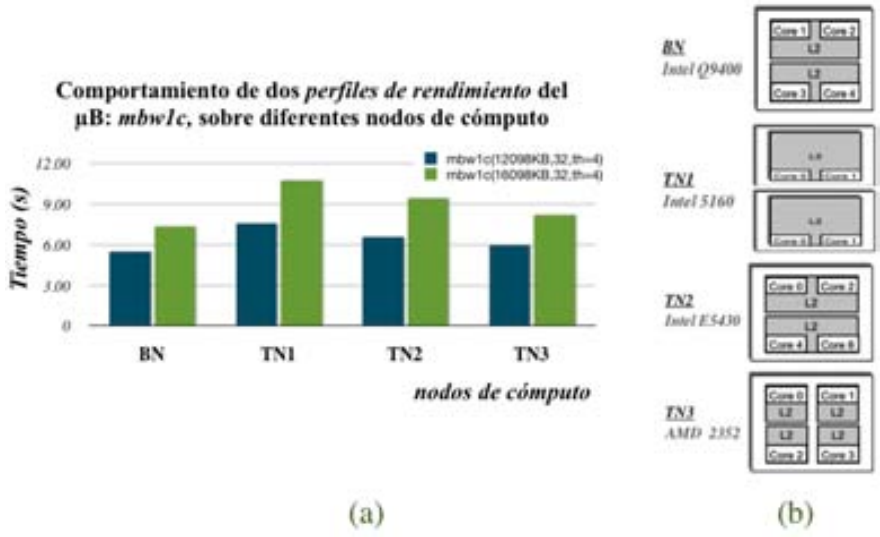



Figura 3.12: Diferencias de comportamiento del tiempo de ejecución del $\mu B$ *Mbw1c* en 4 nodos de cómputo diferentes, para dos *workloads* de entradas diferentes $w1$ y $w2$.

comportamiento dado por 2 *perfiles de rendimiento* obtenidos a partir de la ejecución del $\mu B$: *Mbw1c* sobre cuatro diseños diferentes de nodos de cómputo, mostrados en la parte derecha[2] de la Figura 3.12 **(b)**.

Se puede observar en la Figura 3.12 **(a)** las diferencias de comportamiento al variar los párametros de entrada (tales como el tamaño del vector o el *stride*), tales como:

- la variación del comportamiento sobre el mismo nodo de cómputo al variar los parámetros de entrada del $\mu B$.
- la variación del comportamiento entre diferentes nodos de cómputo para el *perfil de rendimiento* sin cambiar los parámetros de entrada del $\mu B$: $mbw1c$.

El $\mu B$ retrata el perfil de rendimiento del *hardware* para cada *workload*

[2]La descripción en detalle de los diseños de nodos de cómputo es presentada en la sección 4.

específico, el tiempo por iteración nos permitirá comparar con el perfil de las aplicaciones.

#### *Métricas de rendimiento (1b)*

Para obtener el tiempo de ejecución, los $\mu Bs$ son ejecutados en todos los nodos de cómputo (*nodos candidatos* y *nodo base*) para diferentes conjuntos de *workloads*. Con el objeto de tener precisión en las medidas, ejecutamos los $\mu Bs$ al menos 30 veces y se obtiene el promedio de cada medida, ya que los datos presentan poca dispersión.

Para todos los $\mu Bs$ hemos usado el parámetro *th*, *nivel de paralelismo*, con un valor de 4 en todos los casos. A su vez, se ha asignado un *thread* a un único core o núcleo de procesamiento.

#### *Base de Datos de Perfiles de Rendimiento (NPPDB)*

La información que describe el patrón de accesos a memoria y las métricas de rendimiento son almacenadas en la base de datos Node Performance Profile Data Base (NPPDB). El tipo de consultas a la base de datos son hechas a partir de la búsqueda secuencial.

Para normalizar los resultados usamos el *Tiempo/#Iteraciones*, calculado como el tiempo total de ejecución del $\mu B$ (para un determinado *workload*) dividido por el número de iteraciones del bucle.

La información a almacenar por cada flujo de acceso a memoria de los *perfiles de rendimiento* estará conformada por la tupla:

Perfil($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

En el caso de incorporar nuevos $\mu Bs$ extraídos de núcleos de aplicaciones, se han de extraer todas las características mencionadas en la Tabla 3.9. Una vez caracterizados, cuando se tenga acceso, se ejecutan sin instrumentar en todos los nodos de cómputo incluidos el nodo base, para obtener las medidas de rendimiento. Para extraer las características de comportamiento, se han de instrumentar los nuevos $\mu Bs$, y se han de analizar los flujos de acceso a memoria identificados. El $\mu B$ instrumentado se ejecuta sólo en

el nodo base, en esta primera etapa se obtiene una traza de accesos a memoria, se analiza y se extraen las características, tal y como se muestra en la Figura 3.8 (1c) y (1d). Una vez obtenidas todas las características de los nuevos $\mu Bs$ se ejecutan en todos los nodos de cómputo y sus *perfiles de rendimiento* se almacenan en la Base de Datos de los Perfiles de Rendimiento para futuras comparaciones.

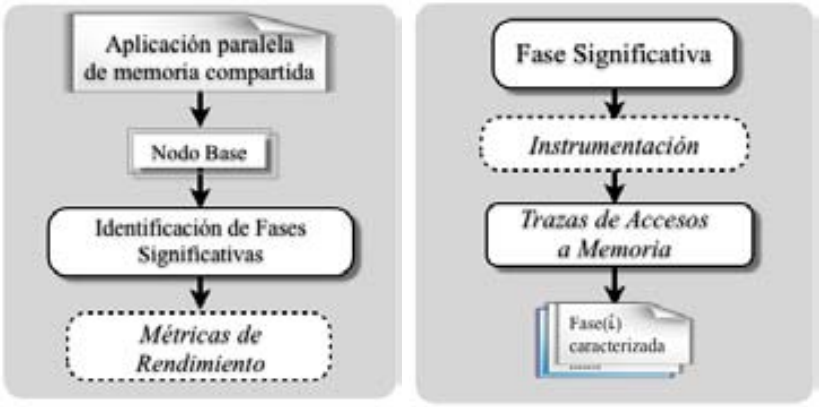
### 3.1.2. Caracterización de la Aplicación

El comportamiento de una aplicación no es aleatorio. En la literatura, encontramos publicaciones donde los investigadores han mostrado que es posible identificar y predecir las fases en un programa [29].

La Figura 3.13 presenta el esquema para la *caracterización de la aplicación*. Se ha dividido en dos etapas, una primera etapa de identificación de las fases significativas de la aplicación (Figura 3.13a), en esta etapa se analiza la aplicación de memoria compartida ejecutándola en el Nodo Base, se identifican las fases significativas (requiere medida del tiempo de ejecución de la fase) y se obtienen las métricas de rendimiento (Tiempo total de ejecución de la aplicación y el cálculo del peso porcentual de cada fase). La Figura 3.13b presenta la segunda etapa para *caracterizar cada fase significativa*: Para extraer las características de las fases significativas se *instrumenta* la aplicación, se *trazan los accesos a memoria* y se organiza toda la información de caracterización de la fase.

La caracterización de la aplicación es una descripción en términos de atributos cualitativos y cuantitativos de cada una de las fases significativas de la aplicación similar a la caracterización realizada para los $\mu Bs$.

Como hemos visto las características cualitativas que nos interesan son características tales como: los flujos de acceso a memoria, patrón de acceso a memoria, tipo de datos, operaciones básicas, densidad de los datos y el efecto *multithreading* si los hilos o procesos de ejecución *comparten* (C) una zona de memoria o acceden a zonas *privadas* (P) de memoria. Características cuantitativas y métricas de rendimiento: tamaño del *workload*, nivel de paralelismo, tiempo de ejecución y el número de iteraciones. A su vez,




(a) Identificación de fases significativas (b) Caracterización de fases significativas

Figura 3.13: Esquema para la *caracterización de la aplicación.*

se obtiene el valor del $Tiempo/\#Iteraciones$. Estas características cuantitativas son obtenidas ejecutando la aplicación en el Nodo Base (BN) o nodo referencia, para un *workload* determinado por el usuario [12].

### Identificación de las *fases representativas*

Cada fase corresponde a un trozo o núcleo de la aplicación, buscamos las fases que son significativas en términos de rendimiento, son fases representativas con respecto al tiempo total de ejecución. Para identificar cada fase, buscamos estructuras/primitivas/constructores de paralelismo en el código [11] y su peso correspondiente respecto al tiempo total de ejecución de la aplicación.

Consideramos que una fase es representativa porque en una región paralela o un $for$ en paralelo, el más lento marcará el tiempo final de la operación. Éste tiempo es el tiempo de la fase, y es el que dividimos por el número de iteraciones completadas por ese *thread* y nos dará el $Tiempo/\#Iteraciones$. Dentro de esa iteración están los accesos y las operaciones del núcleo de la fase. El tiempo de cada fase es el que comparamos.

La relación entre el tiempo de la fase y el tiempo total de ejecución nos da el peso de la fase.

Una vez obtenidas las fases, el análisis es realizado de forma separada para cada fase (Ver Figura 3.13b). En cada fase, se identifican sus características, tales como los flujos de acceso a memoria y por cada flujo se obtiene su traza de accesos a memoria respectiva.

*Ejemplo Ilustrativo Aplicación Multiplicación de Matrices*

A continuación, iremos describiendo la metodología utilizando un ejemplo como guía para la obtención de la *caracterización de la aplicación* con un algoritmo de la Multiplicación de Matrices. La implementación del algoritmo se realiza por descomposición de filas, es decir, un subconjunto de filas es asignado a cada *thread* o hilo de ejecución (modelo de programación paralela).

$$c_{i,j} = \sum_{l=1}^{N} a_{i,l} \mathrm{x} b_{l,j} \tag{3.2}$$

En 3.2, se muestra el algoritmo clásico de la multiplicación de matrices, MM. Producto ordinario de A(filas) por B(columnas). En este caso, esta aplicación sólo tiene una fase significativa con 3 flujos de acceso a memoria o *streams*: *stream* $\boldsymbol{a}$, *stream* $\boldsymbol{b}$ y *stream* $\boldsymbol{c}$.

### *Métricas de rendimiento*

Para obtener el tiempo de ejecución de cada fase y el peso correspondiente, la aplicación es ejecutada en el *nodo base*. El *workload* utilizado como base para predecir o estimar el comportamiento de la aplicación sobre diferentes nodos de cómputo, es la entrada de la aplicación, es un dato de entrada a la fase de caracterización de la aplicación , para el análisis utilizamos el tamaño de cada uno de los *stream* de datos.

El *workload* usado como ejemplo para la caracterización del algoritmo, se pueden observar en la Tabla 3.14, como se muestra para este ejemplo son matrices de 4000x4000.

| Aplicación | *workload* |
|---|---|
| MM | 4000x4000 |

Tabla 3.14: Datos de entrada para la aplicación: MM *workload 4000x4000*.

En este paso se ejecuta la aplicación en el nodo base, y se identifican las fases significativas, para ello, se mide el tiempo de ejecución de la fase (identificada por el $fork\ join$) y el tiempo total de ejecución.

| Aplicación | Tiempo Total (s) | Tiempo Fase ($fork\ join$ principal) (s) |
|---|---|---|
| MM | 216.06140 | 212.02540 |
| Peso Porcentual | 100 % | 98.13 % |

Tabla 3.15: Tiempo de ejecución, tiempo de la fase identificada y peso porcentual de la aplicación MM *workload 4000x4000*.

En la Tabla 3.15 se observan tanto el tiempo de ejecución total de la aplicación MM, como el tiempo de la fase o $fork\ join$ principal, con esta información observamos que el peso de la fase es significativo (98.13 % del tiempo total de ejecución) y pasamos a analizar la fase. Adicionalmente, se normaliza la métrica de comportamiento, se calcula el *Tiempo/#Iteración* como tiempo de ejecución de cada fase dividido por el total de iteraciones.

**Análisis de cada una de las fases significativas**

***Instrumentación*** Se ubican puntos de prueba o sondas programadas por cada *Load/Store* identificados (esto nos permite identificar los *streams* o ***flujos de acceso a memoria***). Estas sondas capturan las direcciones

de memoria de cada flujo de acceso a memoria en cada fase representativa de la aplicación. Estos flujos de accesos a memoria (*"data streams"*) *Load/Store* son caracterizados con respecto al paso de salto local o *local data stride*. Un paso de salto o *stride* local se define en la memoria de datos como la diferencia entre los accesos de direcciones de memoria temporal adyacente que provienen de una sola instrucción [29] —esto se hace mediante el seguimiento de las direcciones de memoria para cada operación de memoria.

***Traza de accesos a memoria*** Para cada *flujo de acceso a memoria* o *stream* se extraen diferentes parámetros: longitud del *stride*, dirección menor y mayor de memoria. Para detectar si los *flujos de accesos a memoria* son compartidos, se comprueba si la región de datos (área de memoria) accedida por un *thread* es la misma para todos los *threads*. De lo contrario los *flujos de accesos a memoria* son privados.

El proceso de capturar las trazas utiliza estructuras de datos para el almacenamiento y la recuperación rápida de los datos.

***Tipo de Dato*** En el lenguaje de programación C, el tipo de dato es un objeto que pertenece a un conjunto específico de valores con los cuales se pueden realizar un conjunto de operaciones determinadas.

Existen cinco tipos básicos que identificaremos: carácter, entero, coma flotante, coma flotante de doble precisión y void. Los demás tipos se basan en algunos de estos tipos básicos. El tamaño y el rango de estos tipos de datos varían con cada tipo de procesador y con la implementación del compilador de C.

En la Tabla 3.16 se presenta la identificación de los tipos de datos con la siguiente especificación, desde el código fuente del programa.

***Fase caracterizada*** Una vez obtenida la información de la traza de la fase, se analiza para obtener toda la información que caracteriza la fase. Es importante obtener la información del patrón de accesos a memoria de cada flujo de acceso a memoria, métricas de rendimiento y tiempo de ejecución

| | |
|---|---|
| char<br>unsigned char, signed char | 1B |
| int<br>unsigned int, signed int<br>short int<br>unsigned short int, signed short int | 2B |
| long int<br>signed long int, unsigned long int<br>float | 4B |
| double:<br>long double: | 8B |

Tabla 3.16: Tipos de Datos

de cada fase, conforman la *fase caracterizada*.

Se tiene entonces la siguiente tupla, como información de caracterización de la fase, por cada flujo de acceso a memoria (*stream* ***v***):

Fase($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

Esta tupla será usada como *entrada* al *proceso de comparación*, en donde se buscará de el $\mu B$ con un *perfil de rendimiento* similar en comportamiento, para ello se analizará la información almacenada en la base de datos.

Para el ejemplo ilustrativo de la caracterización de la aplicación MM, se puede observar en la Tabla 3.17, la información relativa a la caracterización para el *workload w1*, con dos *flujos de acceso a memoria* significativos (*stream* ***v1*** y *stream* ***v2***).

La Tabla 3.17 presenta los valores obtenidos después de ejecutar la fase instrumentada y una vez analizada la información de la traza, tal y como se comentó, se obtienen los datos que caracterizan el patrón de acceso a memoria. El resultado del análisis nos muestra que el número de accesos es significativo y el tamaño de la zona de memoria a la que acceden está dentro del rango que tiene un comportamiento predecible. Se identifican dos *flujos de accesos a memoria* que hemos denominado **v1** y **v2** que acceden a zonas de memoria de diferente tamaño.

| | Fase MM *workload 4000x4000* | | | | | | |
|---|---|---|---|---|---|---|---|
| | Tamaño (kB) | Stride | Tipo de Dato | Acceso | Tiempo/ #Iteraciones (µs) | #Iteraciones (xE6) | Tiempo (s) |
| ***stream v1*** | 15624 | 32000 | 8B | C | 0.01325 | 16000 | 212.02540 |
| ***stream v2*** | 3906 | 8 | 8B | P | | | |

Tabla 3.17: Caracterización de la Fase MM1: *workload 4000x4000*

Éstos flujos de acceso a memoria son representativos ya que el tamaño de la zona de memoria a la que acceden es significativa. Un flujo de acceso a memoria es representativo si supera el tamaño donde el comportamiento es variable debido al impacto de la cache.

Los actuales diseños de nodos de cómputo integran cache con un tamaño de la cache L1 superiores a 1MB.

El flujo de acceso a memoria **v1** tiene un *stride* de 32000, y la zona de memoria a la que accede **v1** es *compartida* (C). El flujo de acceso a memoria **v2**, tiene un *stride* de 8 y la zona de memoria a la que accede es *privada* (P).

### 3.1.3. Proceso de Comparación.

El objetivo es hallar los $\mu Bs$ similares a la fase de la aplicación para la que se va a predecir el comportamiento en otros nodos. Para ello se mira la similitud o distancia considerando cada una de las características seleccionadas para describir el comportamiento computacional.

El proceso consiste en encontrar un *índice de similitud* entre los *perfiles de comportamiento* del nodo base (BN) al ejecutar los $\mu Bs$ y la *caracterizacion de la fase* de la aplicación. El *índice de similitud* $h_i$ corresponderá

a un valor resultado de la evaluación de la similitud entre el perfil y el comportamiento de la fase.

Este índice de similitud es utilizado para cuantificar la distancia entre el comportamiento de la fases y el comportamiento de los diferentes $\mu Bs$ en la base de datos. Con esto se seleccionará el $\mu B$ o $\mu Bs$ que tenga la menor distancia o que la distancia esté por debajo de un umbral.

Los *perfiles de rendimiento* con los que se contrasta la similitud son los ejecutados en el nodo base o BN, el mismo nodo en donde se ha caracterizado cada fase de la aplicación.

Para obtener el índice de similitud no se está utilizando un algoritmo clásico de distancia como la distancia *euclideana* o la distancia del *valor absoluto*, ya que disponemos de variables cuantitativas y cualitativas (donde se utiliza una comparación *booleana*), las diferentes variables no tienen la misma importancia o el mismo peso en el momento de analizar la similitud.

La distancia entre cada uno de los elementos de la tabla debe cumplir una serie de propiedades:

- si es cuantitativa debe ser inferior a un umbral previamente definido, para que se considere una característica significativa
- si es cualitativa se hace una comparación booleana

A estos resultados se les da un peso que marca la importancia de dicha característica en la comparación.

La Figura 3.18 presenta el esquema del *proceso de comparación*. Se utiliza la información de una serie de variables para cada fase y, conforme a estas variables se mide la similitud con los valores obtenidos al ejecutar los $\mu Bs$. Si no existen similares, se puede ajustar los parámetros, y volver a comenzar el proceso o decidir generar nuevos $\mu Bs$.

Si el número de similares es elevado (superior de un umbral, para este trabajo se ha considerado 5), se puede ajustar el margen de similitud de las variables, para que el algoritmo sea más selectivo.

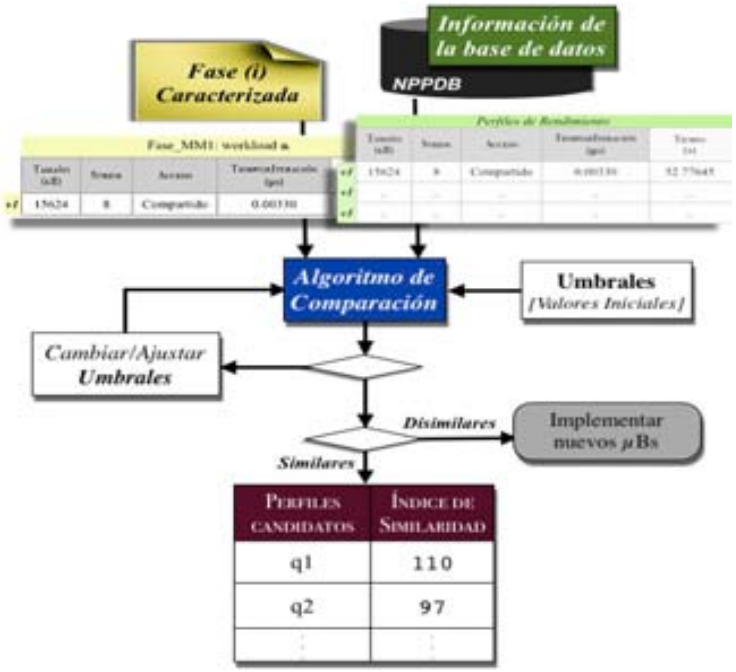



Figura 3.18: Esquema del *proceso de comparación*

Una vez seleccionados los candidatos se analiza la calidad de la predicción de cada uno, para seleccionar qué $\mu B$ se va a utilizar para predecir el comportamiento en los nodos destino.

El esquema de la Figura 3.18 se compone de:

- *Características de la fase (i)*, para este trabajo se han analizando elementos que caracterizan los flujos de acceso a memoria que han sido caracterizados en la sección 3.1.2 de *caracterización de la aplicación.*
- *Información de la base de datos*, la cual contiene todos los *perfiles de rendimiento*, descritos en la sección 3.1.1.
- *Umbrales*. Cada elemento a comparar la distancia posee un umbral con el fin de obtener un rango de precisión en la estimación. Si se supera un cierto valor umbral, se le da un peso que afectará al índice de similitud (Ver Tabla 3.19). El ajuste de parámetros requiere la

intervención del experto, para ajustar los valores por defecto. Para aplicar la metodología hemos puesto unos valores iniciales con el fin de obtener un rango aceptable en la similitud y, por ende, de la estimación. A su vez, y como se puede observar en la Tabla 3.19, hemos propuesto como criterio de restricción, que si los umbrales sobrepasan un valor porcentual de diferencia, se descarta de forma directa el *perfil de rendimiento.*

| *Tamaño* | | *Stride* | | *TipoDato* | | *Acceso* | | *Tiempo/ #Iteraciones* | |
|---|---|---|---|---|---|---|---|---|---|
| Umbral | $\acute{\imath}_T$ | Umbral | $\acute{\imath}_S$ | Umbral | $\acute{\imath}_D$ | Umbral | $\acute{\imath}_A$ | Umbral | $\acute{\imath}_Z$ |
| $\leq\alpha_1$ | ***valor*** | $\leq\beta_1$ | ***valor*** | $\theta$ | ***valor*** | $\Psi$ | ***valor*** | $\leq\delta_1$ | ***valor*** |
| $>\alpha_2$ | ***Descartar*** | $>\beta_2$ | ***Descartar*** | $\neg\theta$ | ***valor*** | $\neg\Psi$ | ***valor*** | $>\delta_2$ | ***Descartar*** |

Tabla 3.19: Umbrales para el algoritmo de comparación.

- *Algoritmo de comparación.* Se compara cada una de las variables para dar un índice de similitud entre los *perfiles de rendimiento* y la *caracterización de las fases*
- *Tabla de índices de similitud.* Tabla resumen de *perfiles de rendimiento* candidatos para la estimación de la *fase caracterizada.*

Las variables a comparar son las características de cada flujo de acceso a memoria ($\boldsymbol{v}$). Para ello se consideran todas las combinaciones sin repetición de los flujos de acceso a memoria de la fase caracterizada y se compara cada combinación con la información de la base de datos de los *perfiles de rendimiento.*

Las combinaciones sin repetición de los $\boldsymbol{v}$ flujos tomados de p en p ($1 \leq p \leq n$) se definen como las distintas agrupaciones formadas con p elementos distintos, eligiéndolos de entre los **n** flujos de accesos a memoria ($\boldsymbol{v}$) que disponemos, considerando una variación distinta a otra sólo si difieren en algún $\boldsymbol{v}$.

$$q=\{C_v^{p=1},C_v^{p=2},..,C_v^{p=n}\}$$

En la Tabla 3.20 se observa un ejemplo de la caracterización de la fase App: *workload* $w$ con $n$ flujos de acceso a memoria ($\boldsymbol{v}$).

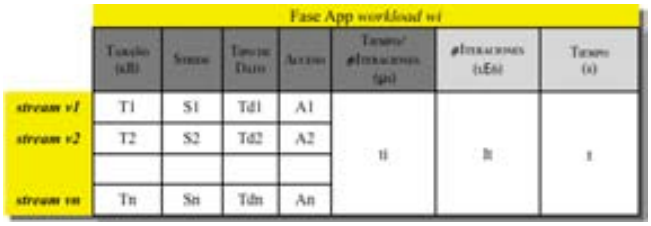



Tabla 3.20: Caracterización de la fase App: *workload wi* con $n$ flujos de acceso a memoria

Las consultas posibles (*queries* o $q$) al *proceso de comparación* de la *fase caracterizada* en la Tabla 3.20 , vendrán dadas por el conjunto de combinatorias de flujos de acceso a memoria de la siguiente manera:

$$q = \{\binom{v}{1}, \binom{v}{2}, \ldots, \binom{v}{n}\}$$

Como ejemplo, supongamos que se identifican 3 flujos de acceso a memoria ($\boldsymbol{v}$) de una *aplicación caracterizada*: **v1**, **v2** y **v3**. El conjunto de consultas ($q$) para cada comparación en la base de datos vendrán dadas por las siguientes combinaciones:

$$q = \{[v_1], [v_2], [v_3], [v_1,v_2], [v_1,v_3], [v_2,v_3], [v_1,v_2,v_3]\}$$

Cada combinación de flujos de acceso a memoria ($\boldsymbol{v}$), junto con la información a comparar de la base de datos de los *perfiles de rendimiento* comprende la tupla o registro conformado por las características de cada flujo de acceso a memoria ($\boldsymbol{v}$) (Véase sección 3.1.2 y 3.1.1).

Fase($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

Perfil($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

Las consultas conformadas por más de un elemento, se harán con su elemento correspondiente. Es decir, para consultas de n elementos *stride*, se consideran n elementos *stride* a comparar dados por la base de datos. El valor de cada comparación correspondiente estará sujeto al algoritmo de comparación (el cual se describirá a continuación) y su valor final será el promedio de entre los valores individuales.

**Algoritmo de Comparación**

Una vez obtenidas las combinaciones, se procede a comparar cada elemento de las tuplas de *Perfil v* con la tupla de la *Fase v* de forma ordenada y con el siguiente criterio:

a) $|Fase\ v_{[\text{Tamaño}]}$ - $Perfil\ v_{[\text{Tamaño}]}| \leq \alpha$

El tamaño de la zona de memoria de cada flujo de acceso a memoria debe tener una diferencia no superior a un umbral $\alpha$ del valor al que accede la zona de memoria de la $Fase\ v[Tamaño]$. El índice resultado $i_T$ del elemento $v[Tamaño]$ será el promedio entre todos los índices dados por el grupo de flujos de acceso a memoria a comparar.

b) $|Fase\ v_{[\text{Stride}]}$ - $Perfil\ v_{[\text{Stride}]}| \leq \beta$

El *Stride* para cada flujo de acceso a memoria debe tener una diferencia no superior a un umbral $\beta$ del *stride* de la fase. El índice resultado $i_S$ del elemento $v[Stride]$ será el promedio entre todos los índices dados por el grupo de flujos de accesos a memoria.

c) $Fase\ v_{[\text{TipoDato}]} \oplus Perfil\ v_{[\text{TipoDato}]} = \theta$

Los flujos de acceso a memoria poseen un índice *booleano* en el elemento TipoDato para la función de similitud.

d) $Fase\ v_{[\text{Acceso}]} \oplus Perfil\ v_{[\text{Acceso}]} = \psi$

Privado (P) o Compartido (C), los flujos de acceso a memoria tendrán un índice *booleano* para la función de similitud.

e) $|Fase\ v_{[\text{Tiempo/\#Iteraciones}]}$ - $Perfil\ v_{[\text{Tiempo/\#Iteraciones}]}| \leq \delta$

El Tiempo/#Iteraciones debe tener una diferencia absoluta no superior a un umbral $\delta$ del valor de la $Fase\ v[Tiempo/\#Iteraciones]$.

**Tabla de índices de similitud**

El ***índice de similitud*** $\dot{i}$ de cada *perfil candidato* vendrá dado por la suma de cada uno de los índices parciales de los elementos obtenidos en la comparación; tal y como se muestra en la siguiente Ecuación 3.3:

$$\dot{i} = iT + iS + iD + iA + iZ \tag{3.3}$$

El candidato con más probabilidad de ser elegido o seleccionado será aquel con mayor ***índice de similitud*** $\dot{i}$, obtenido en 3.3.

A continuación, se ilustra el *proceso de comparación* con el ejemplo del algoritmo de multiplicación de matrices.

**Proceso de comparación: para la fase caracterizada de la aplicación MM** *workload* $4000x4000$

En la sección 3.1.2 se presenta la fase caracterizada para la aplicación MM (Ver Tabla 3.17). El *proceso de comparación* está conformado por las consultas $q$ dadas por la combinación sin repetición de la información de los flujos de accesos a memoria. Como p=2, se buscarán $\mu Bs$ que acceden a 1 ó a 2 stream.

$$q = \{[v1],[v2],[v1v2]\}$$

Se tienen entonces 3 consultas, que se compararán con los $\mu Bs$ con un flujo y un patrón de comportamiento similar a $v1$, o con un flujo y un

patrón de comportamiento similar a $v2$, o con 2 flujos y patrones similares a $v1$ y $v2$. Identificamos cada consulta como $q1$, $q2$ y $q3$, corresponden a la consulta $[v1]$, $[v2]$ y $[v1v2]$ respectivamente.

**Algoritmo de Comparación: Valores Iniciales** Con el objeto de poder tener una mejor precisión en la similitud en el ejemplo ilustrativo, se han dividido los umbrales no *booleanos* en cuatro zonas. Cada zona posee un índice como se muestra en la Tabla 3.21. Adicionalmente la tabla muestra valores a descartar cuando superen el 80 % del umbral considerado. Vemos que con los valores máximos de similitud que se han definido se puede obtener un índice de 125

| *Tamaño* | | *Stride* | | *TipoDato* | | *Acceso* | | *Tiempo/ #Iteraciones* | |
|---|---|---|---|---|---|---|---|---|---|
| Umbral | $\acute{I}_T$ | Umbral | $\acute{I}_S$ | Umbral | $\acute{I}_D$ | Umbral | $\acute{I}_A$ | Umbral | $\acute{I}_Z$ |
| ≤2% | ***25*** | ≤5% | ***25*** | θ | ***25*** | ψ | ***25*** | ≤2% | ***25*** |
| ≤5% | ***15*** | | | | | | | ≤5% | ***15*** |
| ≤10% | ***10*** | ≤10% | ***10*** | ~θ | ***0*** | ~ψ | ***0*** | ≤10% | ***10*** |
| ≤20% | ***5*** | | | | | | | ≤20% | ***5*** |
| >80% | ***Descartar*** | >80% | ***Descartar*** | | | | | >80% | ***Descartar*** |

Tabla 3.21: Valores iniciales de los umbrales usados en la parte experimental

### Consulta *q1*

Los *perfiles de rendimiento* de la Tabla 3.22 son algunos ejemplos de $perfiles$ de la base de datos de *perfiles de rendimiento*, compuestos por un flujo de acceso a memoria *v1*. Estos perfiles sirven para ilustrar el *algoritmo de comparación* para la consulta $q1$.

| | Tamaño | Stride | Tipo Dato | Acceso | Tiempo/ #Iteraciones |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw1hc*** | 4101 | 8 | 8B | C | 0.00337 |
| | 15869 | 8 | 8B | C | 0.00299 |
| | 3901 | 32000 | 8B | C | 0.00180 |
| | 16264 | 32000 | 8B | C | 0.12454 |
| ***mbw1hp*** | 4101 | 8 | 8B | P | 0.00430 |
| | 15869 | 8 | 8B | P | 0.00431 |
| | 3901 | 32000 | 8B | P | 0.00180 |
| | 15864 | 32000 | 8B | P | 0.12455 |
| ***mbw1c*** | 4101 | 8 | 8B | C | 0.00411 |
| | 15869 | 8 | 8B | C | 0.00421 |
| | 3901 | 32000 | 8B | C | 0.00749 |
| | 16864 | 32000 | 8B | C | 0.01006 |
| ***mbw1p*** | 4101 | 8 | 8B | P | 0.00453 |
| | 15869 | 8 | 8B | P | 0.00452 |
| | 3901 | 32000 | 8B | P | 0.00747 |
| | 15864 | 32000 | 8B | P | 0.02998 |
| ***mbw1cX*** | 4101 | 30000 | 8B | C | 0.00925 |
| | 15869 | 30000 | 8B | C | 0.01242 |
| ***mbw1pX*** | 4101 | 30000 | 8B | P | 0.00000 |
| | 15869 | 30000 | 8B | P | 0.00000 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 3.22: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$ y $q2$. Todas las pruebas se han realizado para $th = 4$.

La Tabla 3.22 refleja *perfiles de rendimiento* de un sólo flujo de acceso a memoria candidatos a ser comparados en la consulta. Cada $perfil$ tiene la descripción de comportamiento, para unos valores concretos característicos y $workloads$ específicos. Se pueden observar 6 $perfiles$: **mbw1hc(**$k, s, th$**)**, **mbw1hp(**$k, s, th$**)**, **mbw1c(**$k, s, th$**)**, **mbw1p(**$k, s, th$**)**, **mbw1cX(**$k, s, th$**)** y **mbw1pX(**$k, s, th$**)**, y cada $perfil$ tiene $workloads$ (k) y $strides$ (s) diferentes.

La Tabla 3.23 presenta el resultado después de aplicar el algoritmo de comparación para la primera consulta $q1$ con la información de la Tabla

3.22.

| | | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | *i*T | % | *i*S | *i*D | *i*A | % | *i*Z | *i* |
| | | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *q1* | ∩ | **mbw1hc** | 73.75% | 0 | 99.98% | Desc. | 25 | 25 | 74.57% | 0 | Desc. |
| *q1* | ∩ | **mbw1hc** | 1.57% | 25 | 99.98% | Desc. | 25 | 25 | 77.44% | 0 | Desc. |
| *q1* | ∩ | **mbw1hc** | 75.03% | 0 | 0% | 25 | 25 | 25 | 86.44% | Desc. | Desc. |
| *q1* | ∩ | **mbw1hc** | 4.10% | 15 | 0% | 25 | 25 | 25 | 839.92% | Desc. | Desc. |
| *q1* | ∩ | **mbw1hp** | 73.75% | 0 | 99.98% | Desc. | 25 | 0 | 67.54% | 0 | Desc. |
| *q1* | ∩ | **mbw1hp** | 1.57% | 25 | 99.98% | Desc. | 25 | 0 | 67.46% | 0 | Desc. |
| *q1* | ∩ | **mbw1hp** | 75.03% | 0 | 0% | 25 | 25 | 0 | 86.42% | Desc. | Desc. |
| *q1* | ∩ | **mbw1hp** | 1.54% | 25 | 0% | 25 | 25 | 0 | 840.03% | Desc. | Desc. |
| *q1* | ∩ | **mbw1c** | 73.75% | 0 | 99.98% | Desc. | 25 | 25 | 69.00% | 0 | Desc. |
| *q1* | ∩ | **mbw1c** | 1.57% | 25 | 99.98% | Desc. | 25 | 25 | 68.25% | 0 | Desc. |
| *q1* | ∩ | **mbw1c** | 75.03% | 0 | 0% | 25 | 25 | 25 | 43.44% | 0 | 75 |
| *q1* | ∩ | **mbw1c** | 7.94% | 10 | 0% | 25 | 25 | 25 | 24.04% | 0 | 85 |
| *q1* | ∩ | **mbw1p** | 73.75% | 0 | 99.98% | Desc. | 25 | 0 | 65.83% | 0 | Desc. |
| *q1* | ∩ | **mbw1p** | 1.57% | 25 | 99.98% | Desc. | 25 | 0 | 65.87% | 0 | Desc. |
| *q1* | ∩ | **mbw1p** | 75.03% | 0 | 0% | 25 | 25 | 0 | 43.62% | 0 | 50 |
| *q1* | ∩ | **mbw1p** | 1.54% | 25 | 0% | 25 | 25 | 0 | 126.26% | Desc. | Desc. |
| *q1* | ∩ | **mbw1cx** | 73.75% | 0 | 6.25% | 10 | 25 | 25 | 30.20% | 0 | 60 |
| *q1* | ∩ | **mbw1cx** | 1.57% | 25 | 6.25% | 10 | 25 | 25 | 6.26% | 10 | **95** |
| *q1* | ∩ | **mbw1px** | 73.75% | 0 | 6.25% | 10 | 25 | 0 | 100.00% | Desc. | Desc. |
| *q1* | ∩ | **mbw1px** | 1.57% | 25 | 6.25% | 10 | 25 | 0 | 100.00% | Desc. | Desc. |
| | | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 3.23: Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 3.22.

En la Tabla 3.23 se pueden apreciar las comparaciones disponibles de la información de la base de datos ejemplo (Tabla 3.22) con la consulta $q1$. A su vez, la Tabla muestra las diferencias en porcentaje de cada elemento de la tupla de vectores comparados, los índices por cada elemento y finalmente el *índice de similitud* $i$, siendo el *perfil de rendimiento* "candidato" **mbw1cx**$(k = 15869, s = 30000, th = 4)$ con un *índice de similitud* de 95.

**Consulta *q2***

La Tabla 3.24 presenta el resultado después de aplicar el algoritmo de comparación para la consulta $q2$ con la información de la Tabla 3.22.

| | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|
| | | % | iT | % | iS | iD | iA | % | iZ | i |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| q2 | mbw1hc | 4.99% | 15 | 0% | 25 | 25 | 0 | 74.57% | 0 | 65 |
| q2 | mbw1hc | 306.27% | Desc. | 0% | 25 | 25 | 0 | 77.44% | 0 | Desc. |
| q2 | mbw1hc | 0.13% | 25 | 399900% | Desc. | 25 | 0 | 86.44% | Desc. | Desc. |
| q2 | mbw1hc | 316.39% | 25 | 399900% | Desc. | 25 | 0 | 839.92% | Desc. | Desc. |
| q2 | mbw1hp | 4.99% | 15 | 0% | 25 | 25 | 25 | 67.54% | 0 | 90 |
| q2 | mbw1hp | 306.27% | Desc. | 0% | 25 | 25 | 25 | 67.46% | 0 | Desc. |
| q2 | mbw1hp | 0.13% | 25 | 399900% | Desc. | 25 | 25 | 86.42% | Desc. | Desc. |
| q2 | mbw1hp | 306.14% | Desc. | 399900% | Desc. | 25 | 25 | 840.03% | Desc. | Desc. |
| q2 | mbw1c | 4.99% | 15 | 0% | 25 | 25 | 0 | 69.00% | 0 | 65 |
| q2 | mbw1c | 306.27% | Desc. | 0% | 25 | 25 | 0 | 68.25% | 0 | Desc. |
| q2 | mbw1c | 0.13% | 25 | 399900% | Desc. | 25 | 0 | 43.44% | 0 | Desc. |
| q2 | mbw1c | 331.75% | Desc. | 399900% | Desc. | 25 | 0 | 24.04% | 0 | Desc. |
| q2 | mbw1p | 4.99% | 15 | 0% | 25 | 25 | 25 | 65.83% | 0 | 90 |
| q2 | mbw1p | 306.27% | Desc. | 0% | 25 | 25 | 25 | 65.87% | 0 | Desc. |
| q2 | mbw1p | 0.13% | 25 | 399900% | Desc. | 25 | 25 | 43.62% | 0 | Desc. |
| q2 | mbw1p | 306.14% | Desc. | 399900% | Desc. | 25 | 25 | 126.26% | Desc. | Desc. |
| q2 | mbw1cX | 4.99% | 15 | 374900% | Desc. | 25 | 0 | 30.20% | 0 | Desc. |
| q2 | mbw1cX | 306.27% | Desc. | 374900% | Desc. | 25 | 0 | 6.26% | 10 | Desc. |
| q2 | mbw1pX | 4.99% | 15 | 374900% | Desc. | 25 | 25 | 100.00% | Desc. | Desc. |
| q2 | mbw1pX | 306.27% | Desc. | 374900% | 25 | 25 | 25 | 100.00% | Desc. | Desc. |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 3.24: Algoritmo de Comparación: consulta $q2$ frente a información de *perfiles de rendimiento* Tabla 3.22.

La Tabla 3.24 presenta la segunda comparación, mostrando a los *perfiles de rendimiento* "candidatos" **mbw1hp**$(k = 4101, s = 8, th = 4)$ y **mbw1p**$(k = 4101, s = 8, th = 4)$ ambos con un *índice de similitud* de 90.

**Consulta *q3***

Los *perfiles de rendimiento* de la Tabla 3.25 son algunos ejemplos de $perfiles$ de la base de datos de *perfiles de rendimiento*, compuestos por dos flujo de acceso a memoria. Se usan para ilustrar el funcionamiento del *algoritmo de comparación* para la tercera consulta.

| | *Tamaño* | *Stride* | *Tipo Dato* | *Acceso* | *Tiempo/ #Iteraciones* |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw2c*** | 15622 | 8 | 8B | C | 0.00618 |
| | 4024 | 8 | 8B | C | |
| ***mbw2p*** | 15622 | 8 | 8B | C | 0.00640 |
| | 4024 | 8 | 8B | P | |
| ***mbw2hc*** | 14840 | 8 | 8B | C | 0.07268 |
| | 4170 | 32000 | 8B | C | |
| ***mbw2hp*** | 14840 | 8 | 8B | C | 0.08169 |
| | 4170 | 32000 | 8B | P | |
| ***mbwXA2hc*** | 3906 | 8 | 8B | C | 0.00477 |
| | 3906 | 8 | 8B | C | |
| ***mbwXA2hc*** | 15617 | 32000 | 8B | C | 0.01312 |
| | 3124 | 8 | 8B | C | |
| ***mbwXA2hp*** | 3906 | 8 | 8B | C | 0.00692 |
| | 3906 | 8 | 8B | P | |
| ***mbwXA2hp*** | 15617 | 32000 | 8B | C | 0.01229 |
| | 3124 | 8 | 8B | P | |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 3.25: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q3$. Todas las pruebas se han realizado para $th = 4$.

La Tabla 3.25 refleja *perfiles de rendimiento* de dos flujos de acceso a memoria candidatos que van a ser comparados en la tercera consulta. Cada $perfil$ tiene la descripción de comportamiento, para unos valores concretos característicos y *workloads* específicos. Se pueden observar 8 *perfiles de rendimiento* a comparar: **mbw2c**, **mbw2p**, **mbw2hc**, **mbw2hp**, **mbwXA2hc** y **mbwXA2hp**.

La Tabla 3.26 presenta el resultado después de aplicar el algoritmo de comparación para la consulta $q3$ con la información de la Tabla 3.25.

| | | | Tamaño | | | Stride | | | TipoDato | | Acceso | | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | iT(1,2) | iT | % | iS(1,2) | iS | iD(1,2) | iD | iA(1,2) | iA | % | iZ | i |
| q3 | ∩ | mbw2c | 0.01% | 25 | 20 | 100% | Desc. | Desc. | 25 | 25 | 25 | 12.5 | 53% | 0 | Desc. |
| | | | 3.02% | 15 | | 0% | 25 | | 25 | | 0 | | | | |
| q3 | ∩ | mbw2p | 0.01% | 25 | 20 | 100% | Desc. | Desc. | 25 | 25 | 25 | 25 | 52% | 0 | Desc. |
| | | | 3.02% | 15 | | 0% | 25 | | 25 | | 25 | | | | |
| q3 | ∩ | mbw2hc | 5.02% | 10 | 10 | 100% | Desc. | Desc. | 25 | 25 | 25 | 12.5 | 449% | Desc. | Desc. |
| | | | 6.76% | 10 | | 399900% | Desc. | | 25 | | 0 | | | | |
| q3 | ∩ | mbw2hp | 5.02% | 10 | 10 | 100% | Desc. | Desc. | 25 | 25 | 25 | 25 | 517% | Desc. | Desc. |
| | | | 6.76% | 10 | | 399900% | Desc. | | 25 | | 25 | | | | |
| q3 | ∩ | mbwXA2hc | 75.00% | 0 | 12.5 | 100% | Desc. | Desc. | 25 | 25 | 25 | 12.5 | 64% | 0 | Desc. |
| | | | 0.00% | 25 | | 0% | 25 | | 25 | | 0 | | | | |
| q3 | ∩ | mbwXA2hc | 0.04% | 25 | 12.5 | 0% | 25 | 25 | 25 | 25 | 25 | 12.5 | 1% | 25 | 100.00 |
| | | | 20.02% | 0 | | 0% | 25 | | 25 | | 0 | | | | |
| q3 | ∩ | mbwXA2hp | 75.00% | 0 | 12.5 | 100% | Desc. | Desc. | 25 | 25 | 25 | 25 | 48% | 0 | Desc. |
| | | | 0.00% | 25 | | 0% | 25 | | 25 | | 25 | | | | |
| q3 | ∩ | mbwXA2hp | 0.04% | 25 | 12.5 | 0% | 25 | 25 | 25 | 25 | 25 | 25 | 7% | 10 | 97.5 |
| | | | 20.02% | 0 | | 0% | 25 | | 25 | | 25 | | | | |

Tabla 3.26: Algoritmo de Comparación: consulta $q3$ con la información de *perfiles de rendimiento* Tabla 3.25.

La Tabla 3.26 muestra al *perfil de rendimiento* "candidato" **mbwXA2hc** $(k1 = 15617, s1 = 32000, k2 = 3124, s2 = 8, th = 4)$ con un *índice de similitud* de 100.

### Tabla de *índices de similitud* para la Aplicación MM

*workload* $4000x4000$

La Tabla 3.27 presenta los índices de similitud de los *perfiles candidatos* la cual refleja los máximos de cada una de las comparaciones.

| Perfiles Candidatos<br>Fase MM *workload 4000x4000* | Índice de Similitud |
|---|---|
| *mbw1cX*<br>*(k=15869,s=30000,th=4)* | 95 |
| *mbw1hp*<br>*(k=4101,s=8,th=4)* | 90 |
| *mbw1p*<br>*(k=4101,s=8,th=4)* | 90 |
| *mbwXA2hc*<br>*(k1 =15617, s1 = 32000, k2 = 3124, s2 = 8, th = 4)* | ***100*** |

Tabla 3.27: Tabla de *índices de similitud* de los $perfiles$ candidatos.

La Tabla 3.27 presenta el resumen de los $perfiles$ candidatos después del proceso de comparación. La Tabla nos índica cual de los *perfiles de rendimiento* que han sido comparados posee un mayor *índice de similitud*, y por tanto este $perfil$ con mayor índice es el candidato a representar esta Fase de la Aplicación, para ese $workload$ específico. En caso de que el resultado comprenda índices de similitud máximos pero con valores duplicados, se procede a ajustar/cambiar los valores de los umbrales para poder encontrar un único índice de similitud máximo.

Es de destacar que el *proceso de comparación* se ha realizado sobre ejecuciones en el **Nodo Base (NB)** tanto de la aplicación como de los *perfiles de rendimiento* a comparar.

Una vez obtenido el *perfil de rendimiento* candidato se procede a la siguiente etapa de la metodología, la cual es la *estimación del comportamiento* sobre otros nodos de cómputo.

### 3.1.4. Proceso de Estimación

La estimación del rendimiento corresponde a predecir el comportamiento de una aplicación paralela sobre un nodo de cómputo (nodo candidato), teniendo en cuenta que la aplicación no se ha ejecutado en ese nodo. Hasta ahora hemos dividido el programa en fases representativas y cada una de las fases posee un peso, sobre el tiempo total de ejecución de la aplicación.

El proceso consiste en utilizar la información de los nodos destino donde

se desea hacer la estimación, del *perfil de rendimiento* seleccionado correspondiente al $\mu B$ y el workload en el proceso de comparación, pero han sido obtenidos cuando se ejecutó sobre nodos donde se desea hacer la estimación. Esta información se extrae de la base de datos, y nos permite estimar el comportamiento de la aplicación sobre todos los nodos de cómputo. 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 o "apropiado"para ejecutar la aplicación.

**Estimación MM** *workload* $4000x4000$ **sobre todos los nodos de cómputo**

Siguiendo con el ejemplo la de multiplicación de matrices, seleccionamos el *perfil de rendimiento* **mbwXA2hc(**$k1 = 15617, s1 = 32000, k2 = 3124, s2 = 8, th = 4$**)** de la Base de Datos de Perfiles de Rendimiento que contiene los valores obtenidos previamente para este $\mu B$ o perfil en el resto de los nodos de cómputo, y con esta información, y considerando el peso, se puede hacer la estimación para esta fase de la aplicación MM, sobre todos los nodos. En este caso, esta fase era la única fase significativa identificada, por lo tanto permite hacer la estimación del comportamiento de la aplicación MM.

La Tabla 3.28 presenta el resumen de las medidas obtenidas del perfil de rendimiento sobre los diferentes nodos de cómputo, las medidas de rendimiento real de la caracterización de la fase de la aplicación en el nodo base (BN) y el tiempo estimado del rendimiento de la fase de la aplicación sobre los diferentes nodos de cómputo.

La primera columna (BN) de la Tabla 3.28 refleja las medidas obtenidas sobre el nodo de referencia y los cálculos obtenidos siguiendo la metodología propuesta. Además, las columnas TN1, TN2 y TN3, de la Tabla 3.28 presentan los tiempos de ejecución de la fase de la aplicación sobre los diferentes nodos de cómputo, estas medidas se han realizado sólo con el propósito de validar la estimación del rendimiento.

A su vez, en la Tabla 3.28 se muestran las diferencias en porcentaje entre

| | | BN | TN1 | TN2 | TN3 |
|---|---|---|---|---|---|
| *Perfil de Rendimiento* | µB: mbwXA2hc (s) | 3.96 | 3.96 | 3.46 | 8.22 |
| | #Iteraciones (xE6) | 300 | | | |
| | Tiempo /#Iteraciones (µs) | 0.01316 | 0.01315 | 0.01148 | 0.02730 |
| *Caracterización de la fase* | *Tiempo Real de la Fase (s)* | 212.03 | ( 216.02 )* | ( 193.92 )* | ( 458.92 )* |
| | #Iteraciones (xE6) | 16000 | | | |
| | Tiempo /#Iteraciones (µs) | 0.01325 | | | |
| *Estimación* | *Tiempo Estimado de la Fase (s)* | 210.62 | 210.46 | 183.74 | 436.87 |
| | Diferencia (%) | 0.66% | 2.57% | 5.25% | 4.81% |

Tabla 3.28: Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación).

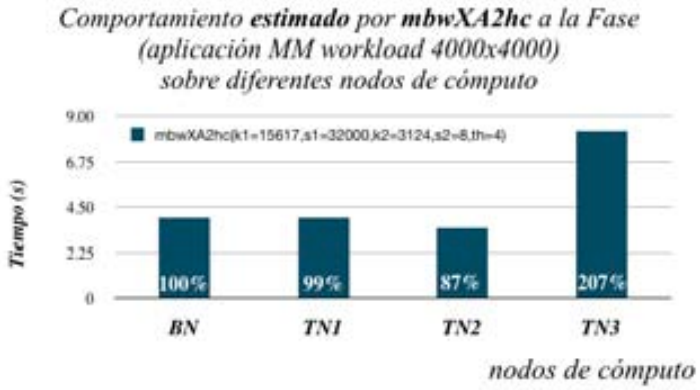
el comportamiento estimado y el real. Se puede apreciar que las diferencias en porcentaje de la estimación no superan el 5.3 %.

En la Figura 3.29 se presentan los comportamientos **estimado** (Figura 3.29a) y **real** (Figura 3.29b) de la aplicación MM. En el eje $x$, los nodos de cómputo BN o sobre el que se ha realizado el análisis, TN1, TN2 y TN3 [3]. En el eje $y$, el tiempo de ejecución en segundos.

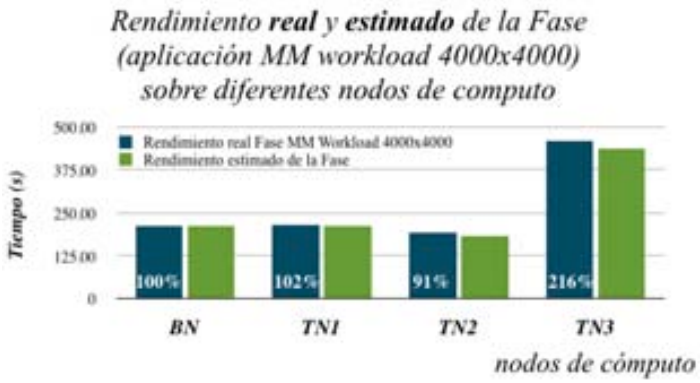
En la Figura 3.29a se muestra la estimación del comportamiento dado por el *perfil de rendimiento* $\mu B$ **mbwXA2hc**$(k1 = 15617, s1 = 32000, k2 = 3124, s2 = 8, th = 4)$ sobre los diferentes nodos de cómputo caracterizados. En cada barra se ha colocado el % en relación al nodo de referencia BN 100 %. La gráfica de estimación muestra la diferencia que existe entre los nodos, el nodo que el método estima que ejecutará la aplicación en menos tiempo es el nodo TN2, que para esta aplicación muestra que es un 30 % más rápido que el nodo BN.

En la Figura 3.29b se presenta el comportamiento real de la aplicación

[3]la descripción de estos nodos de cómputo se hará en la siguiente sección




(a) Comportamiento estimado por **mbwXA2hc**




(b) Comportamiento real de la Fase (aplicación MM *workload* 4000$x$4000)

Figura 3.29: Estimación y comportamiento real de la Fase (aplicación MM *workload* 4000$x$4000) sobre los diferentes nodos de cómputo caracterizados.

MM *workload 4000x4000* y el estimado. Las medidas de rendimiento real de la fase de la aplicación se han realizado sólo con el propósito de validar la metodología propuesta. De igual forma que en la Figura 3.29a, el 100 % representa el comportamiento sobre el nodo referencia BN. Se puede apreciar comportamientos similares entre la estimación, resultado de aplicar la metodología propuesta, y el comportamiento real, donde se observa que el nodo más rápido al ejecutar la aplicación también es el TN2.

# Capítulo 4

# Validación Experimental

Con el propósito de validar experimentalmente la metodología, en este capítulo se presenta la descripción y configuración de los nodos de cómputo utilizados, así como el proceso de caracterización de su *perfil de rendimiento*, utilizando microbenchmarks. A continuación se aplica el proceso detallado de cada una de las etapas de la metodología con diferentes aplicaciones.

Las aplicaciones consideradas para este apartado son el algoritmo de multiplicación de matrices, el simulador de partículas N-Body y el Bloque Tridiagonal del NAS *parallel benchmark*. Al final del capítulo, se presentan los resultados obtenidos de la estimación del comportamiento y el respectivo comportamiento real.

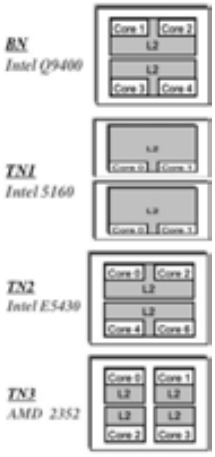
## 4.1. Configuración de los nodos de cómputo

En la Figura 4.1, se presentan las configuraciones o diseños de los nodos de cómputo usados en la parte experimental. El nodo referencia es identificado como BN y los nodos en donde se realiza la estimación son identificados como TN1, TN2 y TN3 respectivamente.

BN, TN1 y TN2 son sistemas que poseen procesadores Intel, con una cache L2 compartida entre 2 cores y con sistema operativo Linux de 64 bits x86_64. Los nodos candidatos incluyen sistemas *"dual-socket"*, es decir, poseen dos procesadores por cada nodo. La configuración TN3, es un

| | BN | TN1 | TN2 | TN3 |
|---|---|---|---|---|
| Procesador | Quad-Core Q9400 Intel | Dual-Core 5160 Intel | Quad-core E5430 Intel (x2) | Quad-core 2352 AMD (x2) |
| Cores | 4 | 4 | 8 | 8 |
| Memoria privada interna | 32KB (L1) | 32KB (L1) | 32KB (L1) | 512KB (L2) |
| Memoria compartida interna | 3MB (x2) | 4MB (x2) | 6MB (x2) | 2MB (L3) |
| Memoria Principal | 4GB | 8GB | 16GB | 4x2 GB NUMA |
| Ghz | 1.99 | 2.66 | 2.00 | 2.11 |
| Kernel | 2.6.26-2 | 2.6.16-46 | 2.6.18-8 | 2.6.24-16 |

(a) Descripción de los nodos de cómputo




(b) Configuraciones

Figura 4.1: Descripción y diseño de los nodos de cómputo

sistema con procesador AMD, que posee una cache L2 privada por cada core y una cache L3 compartida en ambos procesadores. A su vez, el TN3 también tiene un sistema operativo de 64 bits con Linux x86_64.

En los sistemas se han ejecutado los $\mu Bs$ y las aplicaciones usando en todos los casos 4 cores por cada nodo.

Para seleccionar la afinidad de cada proceso, es decir, enlazar un proceso o un *thread* específico a un core o procesador determinado, usamos la biblioteca sched.h. En caso de que el nodo de cómputo posean mas de 4 cores (ejemplo: TN2 y TN3), escogemos los cores dentro del mismo procesador. Esto se hace, debido a que se quiere hacer la estimación al usar 4 procesos o *threads* de las aplicaciones o $\mu Bs$.

Previamente se ejecutan todos los $\mu Bs$ en los sistemas de cómputo antes descrito, esto nos permite obtener los *Perfiles de Rendimiento de los Nodos de cómputo* y almacenados en la base de datos del ***NPPDB***.

Como hemos visto en el Capítulo 3, hemos diseñado y ejecutado un conjunto de $\mu Bs$ para caracterizar el comportamiento de las aplicaciones que están limitadas por sus accesos a memoria.

Centrados en el patrón de acceso a memoria se han diseñado $\mu Bs$ de

carácter específico, cuya finalidad es la de medir las prestaciones con diversos patrones de acceso a memoria, secuencial, con stride, aleatorio; accesos a matrices densas y dispersas, entre otras, las cuales estresan el ancho de banda en las comunicaciones procesador-memoria.

## 4.2. Descripción de las aplicaciones seleccionadas

Hemos seleccionado un conjunto significativo de aplicaciones paralelas de memoria compartida, con el propósito de mostrar la validación experimental.

- La Multiplicación de Matrices aparece en un gran número de algoritmos en el álgebra lineal (sistemas de ecuaciones, cálculo de estructuras, determinantes, etc), como también en núcleos principales de aplicaciones científicas. Los algoritmos de la MM son muy estudiados en la computación de altas prestaciones.

- Simulador de Partículas N-Body es un simulador de partículas n-cuerpos donde el comportamiento del sistema evoluciona en base a su gravedad interna. Se dan las condiciones iniciales e inicia la simulación. Los valores de entrada son el número de partículas, la velocidad de la simulación, la política de colisiones, la fuerza de la gravedad, la masa de las partículas, etc.

- El Bloque Tridiagonal del NAS Parallel Benchmark, es una aplicación empleada en dinámica de fluidos computacional *CFD*, el cual usa un algoritmo implícito para solucionar las ecuaciones de *Navier-Stokes*. Se trata de un conjunto de ecuaciones en derivadas parciales no lineales que describen el movimiento del fluido en 3 dimensiones.

A continuación se presenta la Tabla 4.2, la cual muestra la identificación de cada una de las aplicaciones utilizadas para predecir las diferencias de comportamiento al ejecutarse en diferentes nodos de cómputo.

Para cada aplicación se ha seleccionado dos *workloads* de entrada, que nos permitirán analizar el impacto del *workloads* en el comportamiento.

| Aplicación | Aplicación ID | *workload* |
|---|---|---|
| *Multiplicación de Matrices* ***Algoritmo MM*** | MM | *5000x5000* |
| *Multiplicación de Matrices* ***Algoritmo MM*** | MM | *6000x6000* |
| *Simulador de partículas* ***N-Body*** | N-Body | *200000p* (200.000 partículas) |
| *Simulador de partículas* ***N-Body*** | N-Body | *320000p* (320.000 partículas) |
| *Nas Parallel Benchmark:* ***Block-Tridiagonal*** | BT Clase A | *ClaseA* |
| *Nas Parallel Benchmark:* ***Block-Tridiagonal*** | BT Clase B | *ClaseB* |

Tabla 4.2: Descripción de las aplicaciones

## 4.3. Aplicación de la metodología para predecir el perfil de comportamiento prestacional

En las siguientes secciones se aplicará la metodología a cada una de las aplicaciones mostradas en la Tabla 4.2.

### 4.3.1. Multiplicación de matrices: análisis de un núcleo con una única fase representativa

El siguiente experimento tiene como propósito aplicar la metodología a los casos de multiplicación de matrices (MM). Al igual que en el capítulo anterior, las MM se usan con el propósito de ilustrar la metodología para una aplicación no muy compleja y estudiada ampliamente en la literatura.

#### Caracterización Aplicación: Algoritmo MM *workload 5000x5000*

Caracterizamos la aplicación MM para un tamaño de *workload* 5000x5000. La implementación del algoritmo se realiza por descomposición de filas, es decir, un subconjunto de filas es asignado a cada *thread* o hilo de ejecución (modelo de programación paralela).

Las matrices a multiplicar serán A y B, ambas tratadas como matrices densas (con muy pocos 0's), el resultado lo almacenaremos en la matriz C. A continuación se presenta el algoritmo MM:

$$c_{i,j} = \sum_{l=1}^{N} a_{i,l} \mathrm{x} b_{j,l} \tag{4.1}$$

La matriz B esta almacenada por columnas o matriz con almacenamiento conforme al acceso, por lo tanto, la matriz B se recorre con un *stride* 1. La aplicación sólo tiene una fase significativa con 3 flujos de acceso a memoria o *streams*: *stream* $\boldsymbol{a}$, *stream* $\boldsymbol{b}$ y *stream* $\boldsymbol{c}$.

El objetivo como vimos en el capítulo 3 es obtener la tupla o registro conformado por las características de cada *flujo de acceso a memoria* (v)

Fase($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

#### Métricas de rendimiento

Se ejecuta la MM en el BN, como vimos en el Capítulo 3, el primer paso es la Identificación de fases significativas (Figura 3.9a), para ello ejecutamos la aplicación y obtenemos las fases, el tiempo de ejecución de la fase y el tiempo de ejecución total.

Se analiza y se calcula el peso correspondiente. La Tabla 4.3, presenta en resumen el tiempo de ejecución, los tiempos de las fases identificadas y sus respectivos pesos porcentuales.

| Aplicación | Tiempo Total (s) | Tiempo Fase (*fork join* principal) (s) |
|---|---|---|
| MM | 95.7915 | 93.8905 |
| Peso Porcentual | 100 % | 98.02 % |

Tabla 4.3: Tiempo de ejecución, tiempo de la fase identificada y peso porcentual de la aplicación MM *workload 5000x5000*

En la Tabla 4.3 se observan tanto el tiempo de ejecución total de la aplicación MM, como el tiempo de la fase o *fork join* principal, con esta información observamos que el peso de la fase es significativo (98.02 % del tiempo total de ejecución) y pasamos a analizar la fase. Adicionalmente, se normaliza la métrica de comportamiento, se calcula el *Tiempo/#Iteraciones* como tiempo de ejecución de cada fase dividido por el total de iteraciones.

A la fase identificada, procedemos a instrumentar para obtener los flujos de acceso a memoria, a partir de esa traza de acceso a memoria, se analizará para en cada flujo el patrón de accesos a memoria y su respectivo tipo de dato, con el objeto de tener la información de caracterización de la fase.

***Fase caracterizada***

La Tabla 4.4, presenta la información relativa a la caracterización de la aplicación MM *workload 5000x5000*, con dos *flujos* significativos *de acceso a memoria* (*stream* ***v1*** y *stream* ***v2***) cuyo *tipo de dato* es *double* (8B).

Se identifican dos *flujos de accesos a memoria* que hemos denominado *stream* ***v1*** y *stream* ***v2*** que acceden a zonas de memoria de diferente tamaño. Los dos flujos de acceso a memoria (*stream* ***v1*** *y stream* ***v2***) tienen *stride* de 8B. La zona de memoria a la que accede *stream* **v1** es *compartida* (C). El flujo de acceso a memoria *stream* **v2** accede a una zona de memoria *privada* (P).

| | Fase MM *workload 5000x5000* | | | | | | |
|---|---|---|---|---|---|---|---|
| | Tamaño (kB) | Stride | Tipo de Dato | Acceso | Tiempo/ #Iteraciones (μs) | #Iteraciones (xE6) | Tiempo (s) |
| ***stream v1*** | 24414 | 8 | 8B | C | 0.00300 | 31250 | 93.89051 |
| ***stream v2*** | 6103 | 8 | 8B | P | | | |

Tabla 4.4: Caracterización de la Fase MM *workload 5000x5000*

### Proceso de comparación: Algoritmo MM *workload 5000x5000*

El *proceso de comparación* busca identificar en la base de datos, todos los $\mu Bs$ ejecutados en el nodo en el que se está ejecutando la aplicación y que tienen un patrón de acceso similar a la memoria. El proceso está conformado por las consultas $q$ dadas por la combinatoria sin repetición de la información de los flujos de accesos a memoria, como p=2 se buscarán $\mu Bs$ que acceden a 1 ó a 2 stream.

$$\text{MM } \textit{workload 5000x5000} = \{[v1],[v2],[v1v2]\}$$

Se tiene entonces 3 consultas para realizar el *algoritmo de comparación*, dado por los tres grupos resultado de la combinatoria. Identificamos cada consulta como $q1$, $q2$ y $q3$, corresponden a la consulta $[v1]$, $[v2]$ y $[v1v2]$ respectivamente.

#### Consulta *q1*

Los *perfiles de rendimiento* de la Tabla 4.5 son algunos ejemplos de $perfiles$ de la base de datos de *perfiles de rendimiento*, compuestos por un flujo de acceso a memoria *v1*, para ilustrar el *algoritmo de comparación* para la consulta $q1$.

| | Tamaño | Stride | Tipo Dato | Acceso | Tiempo/ #Iteraciones |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw1hc*** | 5981 | 8 | 8B | C | 0.00293 |
| | 23193 | 8 | 8B | C | 0.00283 |
| ***mbw1c*** | 5981 | 8 | 8B | C | 0.00417 |
| | 23193 | 8 | 8B | C | 0.00414 |
| ***mbw1p*** | 5981 | 8 | 8B | P | 0.00452 |
| | 23193 | 8 | 8B | P | 0.00552 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.5: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$ y $q2$. Todas las pruebas se han realizado para $th = 4$.

La Tabla 4.5 presenta 4 cuatro *perfiles de rendimiento* seleccionados de la base de datos NPPDB para aplicar el *algoritmo de comparación.*

La Tabla 4.6 presenta el algoritmo de comparación para la primera consulta $q1$ con la información de la Tabla 4.5.

| | | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | *i*T | % | *i*S | *i*D | *i*A | % | *i*Z | *i* |
| ⋮ | | | ⋮ | | ⋮ | | ⋮ | ⋮ | ⋮ | | ⋮ |
| ***q1*** | ∩ | ***mbw1hc*** | 75.50% | 0 | 0% | 25 | 25 | 25 | 2.19% | 15 | 90 |
| ***q1*** | ∩ | ***mbw1hc*** | 5.00% | 10 | 0% | 25 | 25 | 25 | 5.55% | 10 | 95 |
| ***q1*** | ∩ | ***mbw1c*** | 75.50% | 0 | 0% | 25 | 25 | 25 | 38.98% | 0 | 75 |
| ***q1*** | ∩ | ***mbw1c*** | 5.00% | 10 | 0% | 25 | 25 | 25 | 37.86% | 0 | 85 |
| ***q1*** | ∩ | ***mbw1p*** | 75.50% | 0 | 0% | 25 | 25 | 0 | 50.72% | 0 | 50 |
| ***q1*** | ∩ | ***mbw1p*** | 5.00% | 10 | 0% | 25 | 25 | 0 | 84.05% | Desc. | Desc. |
| ⋮ | | | ⋮ | | ⋮ | | ⋮ | ⋮ | ⋮ | | ⋮ |

Tabla 4.6: Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.5.

En la Tabla 4.6 se puede apreciar las diferencias en porcentajes de cada elemento de la tupla de vectores comparados, los índices por cada elemento y finalmente el *índice de similitud* $i$, siendo el **mbw1hc(**$k = 23193, s =$

$8, th = 4$) con un *índice de similitud* de 95 el *perfil de rendimiento* "candidato" de la consulta.

### Consulta *q2*

La Tabla 4.7 presenta el algoritmo de comparación para la consulta $q2$ con la información de la Tabla 4.5.

| | | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | iT | % | iS | iD | iA | % | iZ | i |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| q2 | ∩ | mbw1hc | 2.00% | 25 | 0% | 25 | 25 | 0 | 2.19% | 15 | 90 |
| q2 | ∩ | mbw1hc | 280.00% | Desc. | 0% | 25 | 25 | 0 | 5.55% | 10 | Desc. |
| q2 | ∩ | mbw1c | 75.50% | 0 | 0% | 25 | 25 | 0 | 38.98% | 0 | 50 |
| q2 | ∩ | mbw1c | 2.00% | 25 | 0% | 25 | 25 | 0 | 37.86% | 0 | 75 |
| q2 | ∩ | mbw1p | 280.03% | Desc. | 0% | 25 | 25 | 25 | 50.72% | 0 | Desc. |
| q2 | ∩ | mbw1p | 2.00% | 25 | 0% | 25 | 25 | 25 | 84.05% | Desc. | Desc. |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.7: Algoritmo de Comparación: consulta $q2$ frente a información de *perfiles de rendimiento* Tabla 4.5.

La Tabla 4.7 presenta la segunda comparación, mostrando a los *perfiles de rendimiento* "candidatos", seleccionando al $perfil$ **mbw1hc**($k = 5981, s = 8, th = 4$) con un *índice de similitud* de 90 como candidato.

### Consulta *q3*

Los *perfiles de rendimiento* de la Tabla 4.8 son algunos ejemplos de $perfiles$ de la base de datos de *perfiles de rendimiento*, compuestos por dos flujo de acceso a memoria. Se usan para ilustrar el funcionamiento del *algoritmo de comparación* para la tercera consulta.

| | Tamaño | Stride | Tipo Dato | Acceso | Tiempo/ #Iteraciones |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *mbw2c* | 5981 | 8 | 8B | C | 0.00511 |
| | 5981 | 8 | 8B | C | |
| *mbw2p* | 23193 | 8 | 8B | C | 0.00496 |
| | 23193 | 8 | 8B | P | |
| *mbw2hc* | 5981 | 8 | 8B | C | 0.05150 |
| | 5981 | 8 | 8B | C | |
| *mbw2hp* | 23193 | 8 | 8B | C | 0.00494 |
| | 23193 | 8 | 8B | P | |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.8: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q3$. Todas las pruebas se han realizado para $th = 4$.

| | Tamaño | | | Stride | | | TipoDato | | Acceso | | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | iT(1,2) | iT | % | iS(1,2) | iS | iD(1,2) | iD | iA(1,2) | iA | % | iZ | i |
| ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | | ⋮ | | ⋮ | | ⋮ | ⋮ | ⋮ |
| *q3* ∩ *mbw2c* | 76.00% | 0 | 7.5 | 0% | 25 | 25 | 25 | 25 | 25 | 12.5 | 53% | 0 | 70 |
| | 2.00% | 15 | | 0% | 25 | | 25 | | 0 | | | | |
| *q3* ∩ *mbw2p* | 5.00% | 10 | Desc. | 0% | 25 | 25 | 25 | 25 | 25 | 25 | 52% | 0 | Desc. |
| | 280.00% | Desc. | | 0% | 25 | | 25 | | 25 | | | | |
| *q3* ∩ *mbw2hc* | 75.50% | 0 | 7.5 | 0% | 25 | 25 | 25 | 25 | 25 | 12.5 | 449% | Desc. | Desc. |
| | 2.00% | 15 | | 0% | 25 | | 25 | | 0 | | | | |
| *q3* ∩ *mbw2hp* | 5.00% | 10 | Desc. | 0% | 25 | 25 | 25 | 25 | 25 | 25 | 517% | Desc. | Desc. |
| | 280.00% | Desc. | | 0% | 25 | | 25 | | 25 | | | | |
| ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | | ⋮ | | ⋮ | | ⋮ | ⋮ | ⋮ |

Tabla 4.9: Algoritmo de Comparación: consulta $q3$ con la información de *perfiles de rendimiento* Tabla 4.8.

En la Tabla 4.9 muestra al *perfil de rendimiento* "candidato" **mbw2c** $(k1 = 5981, s1 = 8, k2 = 5981, s2 = 8, th = 4)$ con un *índice de similitud* de 70.

### Tabla de *índices de similitud* para la Aplicación MM *workload 5000x5000*

La Tabla 4.10 presenta los índices de similitud de los *perfiles candidatos* como resultado del *algoritmo de comparación*.

| Perfiles Candidatos Fase MM *workload 5000x5000* | Índice de Similitud |
|---|---|
| *mbw1hc (k=23193,s=8,th=4)* | 95 |
| *mbw1hc (k=5981,s=8,th=4)* | 90 |
| *mbw2c (k1=23193, s1,=8,k2=23193,s2=8,th=4)* | 70 |

Tabla 4.10: Tabla de *índices de similitud* de los $perfiles$ candidatos.

### Estimación MM sobre todos los nodos de cómputo

Una vez identificado el *perfil de rendimiento* candidato, seleccionamos el *perfil de rendimiento* **mbw1hc**($k = 23193, s = 8, th = 4$) de la base de datos NPPDB que contiene los valores obtenidos previamente para este $\mu B$ o perfil en el resto de los nodos de cómputo, y con esta información, y considerando el peso, se puede hacer la estimación para esta fase de la aplicación MM, sobre todos los nodos. En este caso, esta fase era la única fase significativa identificada, por lo tanto permite hacer la estimación de la aplicación MM.

La Tabla 4.11 presenta el resumen de las medidas obtenidas del perfil de rendimiento sobre los diferentes nodos de cómputo, las medidas de rendimiento real de la caracterización de la fase de la aplicación en el nodo base (BN) y el tiempo estimado del rendimiento de la fase de la aplicación sobre los diferentes nodos de cómputo.

La primera columna (BN) de la Tabla 4.11 refleja las medidas obtenidas sobre el nodo de referencia y los cálculos obtenidos siguiendo la metodología propuesta. Además, las columnas TN1, TN2 y TN3, de la Tabla 4.11 presentan los tiempos de ejecución de la fase de la aplicación sobre los

| | | BN | TN1 | TN2 | TN3 |
|---|---|---|---|---|---|
| *Perfil de Rendimiento* | µB: mbw1hc (s) | 6.71 | 10.74 | 9.49 | 12.57 |
| | #Iteraciones (xE6) | 2370 | | | |
| | Tiempo /#Iteraciones (µs) | 0.00283 | 0.00453 | 0.00401 | 0.00530 |
| *Caracterización de la fase* | *Tiempo Real de la Fase (s)* | 93.89 | ( 170.37 )* | ( 143.76 )* | ( 165.86 )* |
| | #Iteraciones (xE6) | 31250 | | | |
| | Tiempo /#Iteraciones (µs) | 0.00300 | | | |
| *Estimación* | *Tiempo Estimado de la Fase (s)* | 88.49 | 141.60 | 125.18 | 165.69 |
| | Diferencia (%) | 5.75% | 16.89% | 12.93% | 0.10% |

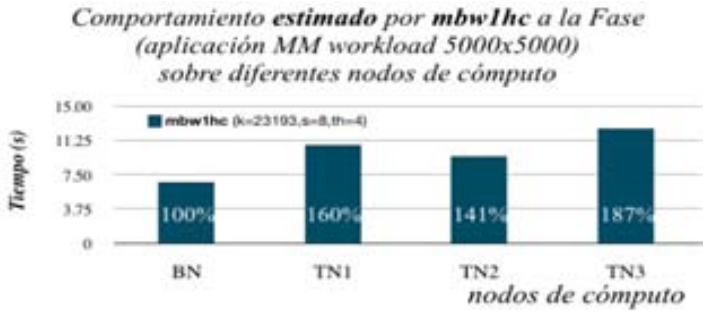
Tabla 4.11: Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación).

diferentes nodos de computo, éstas medidas se han realizado sólo con el propósito de validar la estimación del rendimiento.

A su vez, en la Tabla 4.11 se muestran las diferencias en porcentaje entre el comportamiento estimado y el real. Se puede apreciar que las diferencias en porcentaje de la estimación no superan el 17 %.

En la Figura 4.12 se presentan los comportamientos **estimado** (Figura 4.12a) y **real** (Figura 4.12b) de la aplicación MM. En el eje $x$, los nodos de cómputo BN o sobre el que se ha realizado el análisis, TN1, TN2 y TN3. En el eje $y$, el tiempo de ejecución en segundos.

En la Figura 4.12a se muestra la estimación del comportamiento dado por el *perfil de rendimiento* $\mu B$ **mbw1hc**$(k = 23193, s = 8, th = 4)$ sobre los diferentes nodos de cómputo caracterizados. En cada barra se ha colocado el % en relación al nodo de referencia BN 100 %. La gráfica de estimación muestra la diferencia que existe entre los nodos, el nodo que el método estima que ejecutará la aplicación en menos tiempo es el nodo BN, que para esta aplicación muestra que es un 60 %, 41 % y 87 % más rápido




(a) Comportamiento estimado por **mbw1hc**


(b) Comportamiento real Fase (aplicación MM workload 5000x5000)

Figura 4.12: Estimación y comportamiento real de la Fase (aplicación MM workload 5000x5000) sobre los diferentes nodos de cómputo caracterizados.

que los nodos TN1, TN2 y TN3 respectivamente.

En la Figura 4.12b se presenta el comportamiento real de la aplicación MM *workload 5000x5000*, estas medidas se han realizado con el objetivo de validar la metodología. De igual forma que en la figura anterior el 100 % representa el comportamiento sobre el nodo referencia BN. Se puede apreciar comportamientos similares entre la estimación, resultado de aplicar la metodología propuesta, y el comportamiento real, donde se ve que el nodo más rápido al ejecutar la aplicación también es el BN.

### Caracterización Aplicación: Algoritmo MM *workload 6000x6000*

Caracterizamos la aplicación MM para un tamaño de *workload* 5000x5000. La implementación del algoritmo se realiza por descomposición de filas, es decir, un subconjunto de filas es asignado a cada *thread* o hilo de ejecución (modelo de programación paralela).

Las matrices a multiplicar serán A y B, ambas tratadas como matrices densas (con muy pocos 0's), el resultado lo almacenaremos en la matriz C. A continuación se presenta el algoritmo MM:

$$c_{i,j} = \sum_{l=1}^{N} a_{i,l} \mathrm{x} b_{j,l} \tag{4.2}$$

La matriz B esta almacenada por columnas o matriz con almacenamiento conforme al acceso. La aplicación sólo tiene una fase significativa con 3 flujos de acceso a memoria o *streams*: *stream* ***a***, *stream* ***b*** y *stream* ***c***.

El objetivo como vimos en el capítulo 3 es obtener la tupla o registro conformado por las características de cada *flujo de acceso a memoria* (v)

Fase($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

#### Métricas de rendimiento

Se ejecuta la MM en el BN, como vimos en el Capítulo 3, el primer paso es la Identificación de fases significativas (Figura 3.9a), para ello ejecutamos la aplicación y obtenemos las fases, el tiempo de ejecución de la fase y el tiempo de ejecución total.

Se analiza y se calcula el peso correspondiente. La Tabla 4.13, presenta en resumen el tiempo de ejecución, los tiempos de las fases identificadas y sus respectivos pesos porcentuales.

| Aplicación | Tiempo Total (s) | Tiempo Fase ($fork\ join$ principal) (s) |
|---|---|---|
| MM | 149.1915 | 147.2781 |
| Peso Porcentual | 100 % | 98.71 % |

Tabla 4.13: Tiempo de ejecución, tiempo de la fase identificada y peso porcentual de la aplicación MM *workload 6000x6000*

En la Tabla 4.13 se observan tanto el tiempo de ejecución total de la aplicación MM, como el tiempo de la fase o $fork\ join$ principal, con esta información observamos que el peso de la fase es significativo (98.71 % del tiempo total de ejecución) y pasamos a analizar la fase. Adicionalmente, se normaliza la métrica de comportamiento, se calcula el *Tiempo/#Iteraciones* como tiempo de ejecución de cada fase dividido por el total de iteraciones.

A la fase identificada, procedemos a instrumentar para obtener los flujos de acceso a memoria, a partir de esa traza de acceso a memoria, se analizará para en cada flujo el patrón de accesos a memoria y su respectivo tipo de dato, con el objeto de tener la información de caracterización de la fase.

***Fase caracterizada***

La Tabla 4.14, presenta la información relativa a la caracterización de la aplicación MM *workload 6000x6000*, con dos *flujos* significativos *de acceso a memoria* (*stream **v1*** y *stream **v2***).

Se identifican dos *flujos de accesos a memoria* que hemos denominado *stream **v1*** y *stream **v2*** que acceden a zonas de memoria de diferente tamaño. Los dos flujos de acceso a memoria (*stream **v1** y stream **v2***) tienen *stride* de 8. La zona de memoria a la que accede *stream* **v1** es *compartida* (C). El flujo de acceso a memoria *stream* **v2** accede a una zona de memoria *privada* (P).

| | Fase MM *workload 6000x6000* | | | | | | |
|---|---|---|---|---|---|---|---|
| | Tamaño (xE3) | Stride | Tipo de Dato | Acceso | Tiempo/ #Iteraciones (μs) | #Iteraciones (xE6) | Tiempo (s) |
| ***stream v1*** | 35156 | 8 | 8B | C | 0.00273 | 54000 | 147.27806 |
| ***stream v2*** | 8789 | 8 | 8B | P | | | |

Tabla 4.14: Caracterización de la Fase MM *workload 6000x6000*

### Proceso de comparación: Algoritmo MM *workload 6000x6000*

El *proceso de comparación* busca identificar en la base de datos, todos los $\mu Bs$ ejecutados en el nodo en el que se está ejecutando la aplicación y que tienen un patrón de acceso similar a la memoria. El *proceso de comparación* esta conformado por las consultas $q$ dadas por la combinatoria sin repetición de la información de los flujos de accesos a memoria, como p=2 se buscarán $\mu Bs$ que acceden a 1 ó a 2 stream.

$$q = \{[v1],[v2],[v1v2]\}$$

Se tienen entonces 3 consultas, que se compararán con los $\mu Bs$ con un flujo y un patrón de comportamiento similar a $v1$, o con un flujo y un patrón de comportamiento similar a $v2$, o con 2 flujos y patrones similares a $v1$ y $v2$. Identificamos cada consulta como $q1$, $q2$ y $q3$, corresponden a la consulta $[v1]$, $[v2]$ y $[v1v2]$ respectivamente.

#### Consulta *q1*

Los *perfiles de rendimiento* de la Tabla 4.15 son algunos ejemplos de $perfiles$ de la base de datos de *perfiles de rendimiento*, compuestos por un flujo de acceso a memoria *v1*, para ilustrar el *algoritmo de comparación* para la consulta $q1$ y $q2$.

| | *Tamaño* | *Stride* | *Tipo Dato* | *Acceso* | *Tiempo/ #Iteraciones* |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw1hc*** | 9033 | 8 | 8B | C | 0.00285 |
| | 33691 | 8 | 8B | C | 0.00274 |
| ***mbw1c*** | 9033 | 8 | 8B | C | 0.00408 |
| | 33691 | 8 | 8B | C | 0.00406 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.15: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$ y $q2$. Todas las pruebas se han realizado para $th = 4$.

| | | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | *i*T | % | *i*S | *i*D | *i*A | % | *i*Z | *i* |
| ⋮ | | ⋮ | | ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ |
| ***q1*** | ∩ | ***mbw1hc*** | 74.31% | 0 | 0% | 25 | 25 | 25 | 4.34% | 15 | 90 |
| ***q1*** | ∩ | ***mbw1hc*** | 4.17% | 15 | 0% | 25 | 25 | 25 | 0.55% | 25 | 115 |
| ***q1*** | ∩ | ***mbw1c*** | 74.31% | 0 | 0% | 25 | 25 | 25 | 49.31% | 0 | 75 |
| ***q1*** | ∩ | ***mbw1c*** | 4.17% | 15 | 0% | 25 | 25 | 25 | 48.72% | 0 | 90 |
| | ⋮ | | | ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ |

Tabla 4.16: Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.15.

En la Tabla 4.16 se puede apreciar las diferencias en porcentajes de cada elemento de la tupla de vectores comparados, los índices por cada elemento y finalmente el *índice de similitud* $i$, siendo el **mbw1hc**($k = 33691, s = 8, th = 4$) con un *índice de similitud* de 115 el *perfil de rendimiento* "candidato".

### Consulta *q2*

La Tabla 4.17 presenta el algoritmo de comparación para la consulta $q2$ con la información de la Tabla 4.15.

| | | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | iT | % | iS | iD | iA | % | iZ | i |
| ⋮ | | ⋮ | ⋮ | | ⋮ | | ⋮ | ⋮ | ⋮ | | ⋮ |
| *q2* | ∩ | ***mbw1hc*** | 2.78% | 15 | 0% | 25 | 25 | 0 | 4.34% | 15 | 80 |
| *q2* | ∩ | ***mbw1hc*** | 283.33% | Desc. | 0% | 25 | 25 | 0 | 0.55% | 25 | Desc. |
| *q2* | ∩ | ***mbw1c*** | 2.78% | 15 | 0% | 25 | 25 | 0 | 49.31% | 0 | 65 |
| *q2* | ∩ | ***mbw1c*** | 283.33% | Desc. | 0% | 25 | 25 | 0 | 48.72% | 0 | Desc. |
| ⋮ | | ⋮ | ⋮ | | ⋮ | | ⋮ | ⋮ | ⋮ | | ⋮ |

Tabla 4.17: Algoritmo de Comparación: consulta $q2$ frente a información de *perfiles de rendimiento* Tabla 4.15

La Tabla 4.17 presenta segunda consulta, siendo el **mbw1hc**($k = 9033, s = 8, th = 4$) con un *índice de similitud* de 80 el *perfil de rendimiento* "candidato".

### Consulta *q3*

Los *perfiles de rendimiento* de la Tabla 4.18 son algunos ejemplos de $perfiles$ de la base de datos de *perfiles de rendimiento*, compuestos por dos flujo de acceso a memoria. Se usan para ilustrar el funcionamiento del *algoritmo de comparación* para la tercera consulta.

| | *Tamaño* | *Stride* | *Tipo Dato* | *Acceso* | *Tiempo/ #Iteraciones* |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw2c*** | 9033 | 8 | 8B | C | 0.00520 |
| | 9033 | 8 | 8B | C | |
| ***mbw2c*** | 23193 | 8 | 8B | C | 0.00497 |
| | 23193 | 8 | 8B | P | |
| ***mbw2hc*** | 9033 | 8 | 8B | C | 0.05150 |
| | 9033 | 8 | 8B | C | |
| ***mbw2hc*** | 23193 | 8 | 8B | C | 0.00500 |
| | 23193 | 8 | 8B | P | |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.18: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q3$. Todas las pruebas se han realizado para $th = 4$.

| | | | Tamaño | | | Stride | | | TipoDato | | | Acceso | | | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | iT(1,2) | iT | % | iS(1,2) | iS | iD(1,2) | iD | | iA(1,2) | iA | | % | iZ | i |
| q3 | ∩ | mbw2c | 74.31% | 0 | 7.5 | 0% | 25 | 25 | 25 | 25 | | 25 | 12.5 | | 90% | Desc. | Desc. |
| | | | 2.78% | 15 | | 0% | 25 | | 25 | | | 0 | | | | | |
| q3 | ∩ | mbw2p | 34.03% | 0 | Desc. | 0% | 25 | 25 | 25 | 25 | | 25 | 25 | | 82% | Desc. | Desc. |
| | | | 163.89% | Desc. | | 0% | 25 | | 25 | | | 25 | | | | | |
| q3 | ∩ | mbw2hc | 74.31% | 0 | 7.5 | 0% | 25 | 25 | 25 | 25 | | 25 | 12.5 | | 89% | Desc. | Desc. |
| | | | 2.78% | 15 | | 0% | 25 | | 25 | | | 0 | | | | | |
| q3 | ∩ | mbw2hp | 34.03% | 0 | Desc. | 0% | 25 | 25 | 25 | 25 | | 25 | 25 | | 83% | Desc. | Desc. |
| | | | 163.89% | Desc. | | 0% | 25 | | 25 | | | 25 | | | | | |

Tabla 4.19: Algoritmo de Comparación: consulta $q3$ frente a información de *perfiles de rendimiento* Tabla 4.18.

La Tabla 4.19 presenta que el proceso de comparación para esta consulta no posee ninguna similitud según el algoritmo, por lo tanto no hay *perfil de rendimiento* candidato dado $q3$.

### Tabla de *índices de similitud* para la Aplicación MM *workload 6000x6000*

La Tabla 4.20 presenta los índices de similitud de los *perfiles candidatos* como resultado del *algoritmo de comparación*.

| Perfiles Candidatos Fase MM *workload 5000x5000* | Índice de Similitud |
|---|---|
| *mbw1hc (k=33691,s=8,ph=4)* | 115 |
| *mbw1hc (k=9033,s=8,ph=4)* | 80 |

Tabla 4.20: Tabla de *índices de similitud* de los $perfiles$ candidatos.

**Estimación MM sobre todos los nodos de cómputo**

Una vez identificado el *perfil de rendimiento* candidato, seleccionamos el *perfil de rendimiento* **mbw1hc**($k = 33691, s = 8, th = 4$) de la base de datos NPPDB que contiene los valores obtenidos previamente para este $\mu B$ o perfil en el resto de los nodos de cómputo, y con esta información, y considerando el peso, se puede hacer la estimación para esta fase de la aplicación MM, sobre todos los nodos. En este caso, esta fase era la única fase significativa identificada, por lo tanto permite hacer la estimación de la aplicación MM.

| | | BN | TN1 | TN2 | TN3 |
|---|---|---|---|---|---|
| *Perfil de Rendimiento* | µB: mbw1hc (s) | 9.47 | 14.87 | 14.36 | 16.84 |
| | #Iteraciones (xE6) | 3450 | | | |
| | Tiempo /#Iteraciones (µs) | 0.00274 | 0.00431 | 0.00416 | 0.00488 |
| *Caracterización de la fase* | *Tiempo Real de la Fase (s)* | 147.28 | ( 247.68 )* | ( 231.28 )* | ( 257.20 )* |
| | #Iteraciones (xE6) | 54000 | | | |
| | Tiempo /#Iteraciones (µs) | 0.00273 | | | |
| *Estimación* | *Tiempo Estimado de la Fase (s)* | 148.23 | 232.75 | 224.77 | 263.58 |
| | Diferencia (%) | 0.64% | 6.03% | 2.82% | 2.48% |

Tabla 4.21: Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación).

La Tabla 4.21 presenta el resumen de las medidas obtenidas del perfil de rendimiento sobre los diferentes nodos de cómputo, las medidas de rendimiento real de la caracterización de la fase de la aplicación en el nodo base (BN) y el tiempo estimado del rendimiento de la fase de la aplicación sobre los diferentes nodos de cómputo.

La primera columna (BN) de la Tabla 4.21 refleja las medidas obtenidas

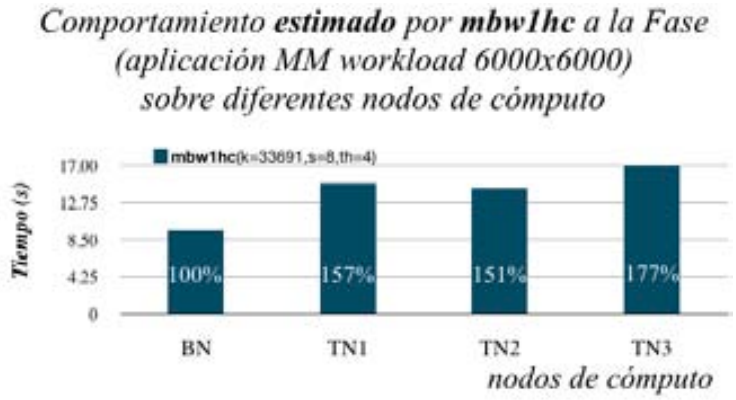
sobre el nodo de referencia y los cálculos obtenidos siguiendo la metodología propuesta. Además, las columnas TN1, TN2 y TN3, de la Tabla 4.21 presentan los tiempos de ejecución de la fase de la aplicación sobre los diferentes nodos de computo, éstas medidas se han realizado sólo con el propósito de validar la estimación del rendimiento.

A su vez, en la Tabla 4.21 se muestran las diferencias en porcentaje entre el comportamiento estimado y el real. Se puede apreciar que las diferencias en porcentaje de la estimación no superan el 7 %.

En la Figura 4.22 se presentan los comportamientos **estimado** (Figura 4.22a) y **real** (Figura 4.22b) de la aplicación MM. En el eje $x$, los nodos de cómputo BN o sobre el que se ha realizado el análisis, TN1, TN2 y TN3. En el eje $y$, el tiempo de ejecución en segundos.

En la Figura 4.22a se muestra la estimación del comportamiento dado por el *perfil de rendimiento* $\mu B$ **mbw1hc**$(k = 33691, s = 8, th = 4)$ sobre los diferentes nodos de cómputo caracterizados. En cada barra se ha colocado el % en relación al nodo de referencia BN 100 %. La gráfica de estimación muestra la diferencia que existe entre los nodos, el nodo que el método estima que ejecutará la aplicación en menos tiempo es el nodo BN, que para esta aplicación muestra que es un 57 %, 51 % y 77 % más rápido que los nodos TN1, TN2 y TN3 respectivamente.

En la Figura 4.22b se presenta el comportamiento real de la aplicación MM *workload 6000x6000*, estas medidas se han realizado con el objetivo de validar la metodología. De igual forma que en la figura anterior el 100 % representa el comportamiento sobre el nodo referencia BN. Se puede apreciar comportamientos similares entre la estimación, resultado de aplicar la metodología propuesta, y el comportamiento real, donde se ve que el nodo más rápido al ejecutar la aplicación también es el BN.




(a) Comportamiento estimado por **mbw1hc**


(b) Comportamiento real y estimado de la Fase (aplicación MM workload 6000x6000)

Figura 4.22: Estimación y comportamiento real de la Fase (aplicación MM *workload* $6000x6000$) sobre los diferentes nodos de cómputo caracterizados.

### 4.3.2. Simulador de Partículas N-Body: análisis de un núcleo con una única fase representativa

#### Caracterización Aplicación: N-Body *workload 200000p*

Se definen como valores de entrada el numero de partículas: 200000. El total de pasos de simulación y el incremento del tiempo son fijados en 10

para todos los casos.

El objetivo es obtener la tupla o registro conformado por las características de cada *flujo de acceso a memoria* (v).

Fase($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

### Métricas de rendimiento

Se ejecuta el simulador N-Body en el BN, para identificar las fases significativas (Figura 3.9a), para ello ejecutamos la aplicación y obtenemos las fases, el tiempo de ejecución de la fase y el tiempo de ejecución total.

Se analiza y se calcula el peso correspondiente. La Tabla 4.23, presenta en resumen el tiempo de ejecución, los tiempos de las fases identificadas y sus respectivos pesos porcentuales.

| Aplicación | Tiempo Total (s) | Tiempo Fase ($fork\ join$ principal) (s) |
|---|---|---|
| N-Body | 1656,78 | 1563,37 |
| Peso Porcentual | 100 % | 94,36 % |

Tabla 4.23: Tiempo de ejecución, tiempo de la fase identificada y peso porcentual de la aplicación N-Body *workload 200000p*

En la Tabla 4.23 se observan tanto el tiempo de ejecución total de la aplicación N-Body, como el tiempo de la fase o $fork\ join$ principal, con esta información observamos que el peso de la fase es significativo (94,36 % del tiempo total de ejecución) y pasamos a analizar la fase. Adicionalmente, se normaliza la métrica de comportamiento, se calcula el *Tiempo/#Iteraciones* como tiempo de ejecución de cada fase dividido por el total de iteraciones.

A la fase identificada, procedemos a instrumentar para obtener los flujos de acceso a memoria, a partir de esa traza de acceso a memoria, se analizará

para en cada flujo el patrón de accesos a memoria y su respectivo tipo de dato, con el objeto de tener la información de caracterización de la fase.

***Fase caracterizada***

La Tabla 4.24, presenta la información relativa a la caracterización de la aplicación N-Body *workload 200000p*, con un sólo *flujo* significativo *de acceso a memoria* (*stream* ***v1***).

| | Fase N-Body *workload 200000p* | | | | | | |
|---|---|---|---|---|---|---|---|
| | Tamaño (kB) | Stride | Tipo de Dato | Acceso | Tiempo/ #Iteraciones (µs) | #Iteraciones (xE6) | Tiempo (s) |
| ***stream v1*** | 781 | 32 | 8B | C | 0.01563 | 99999 | 1563.37 |

Tabla 4.24: Caracterización de la Fase NBody *workload 200000p*

Se identifican un *flujo de acceso a memoria* que hemos denominado *stream* ***v1***, accede a memoria con un *stride* de 32. La zona de memoria a la que accede *stream* ***v1*** es *compartida* (C).

**Proceso de comparación: N-Body *workload 200000p***

El *proceso de comparación* busca identificar en la base de datos, todos los $\mu Bs$ ejecutados en el nodo en el que se está ejecutando la aplicación y que tienen un patrón de acceso similar a la memoria. El *proceso de comparación* esta conformado por la consulta $q$ dadas por la combinatoria sin repetición de la información de los flujos de accesos a memoria, como p=1 se buscarán $\mu Bs$ que acceden a 1 stream.

$$q = \{[v1]\}$$

Se tiene entonces una entrada para realizar el *algoritmo de comparación*, identificamos el grupo a comparar como $q1$.

**Consulta *q1***

Los *perfiles de rendimiento* de la Tabla 4.25 son compuestos por un flujo de acceso a memoria. Hemos seleccionado estos perfiles para ilustrar *algoritmo de comparación* en la consulta $q1$.

| | *Tamaño* | *Stride* | *Tipo Dato* | *Acceso* | *Tiempo/ #Iteraciones* |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw1c*** | 820 | 32 | 8B | C | 0.01500 |
| | 1367 | 32 | 8B | C | 0.01575 |
| ***mbw1p*** | 820 | 32 | 8B | P | 0.02753 |
| | 1367 | 32 | 8B | P | 0.03318 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.25: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$. Todas las pruebas se han realizado para $th = 4$.

La Tabla 4.25 presenta 4 cuatro *perfiles de rendimiento* seleccionados de la base de datos NPPDB para aplicar el *algoritmo de comparación*.

| | | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | /T | % | /S | /D | /A | % | /Z | *i* |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***q1*** | ∩ | ***mbw1c*** | 4.99% | 15 | 0% | 25 | 25 | 25 | 4.03% | 15 | 105 |
| ***q1*** | ∩ | ***mbw1c*** | 75.03% | 0 | 0% | 25 | 25 | 25 | 0.75% | 25 | 100 |
| ***q1*** | ∩ | ***mbw1p*** | 4.99% | 15 | 0% | 25 | 25 | 0 | 76.14% | 0 | 65 |
| ***q1*** | ∩ | ***mbw1p*** | 75.03% | 0 | 0% | 25 | 25 | 0 | 112.26% | Desc. | Desc. |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.26: Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.25.

La Tabla 4.26 presenta la comparación de la información de la Tabla 4.15 para la consulta $q1$. Se pueden apreciar las diferencias en porcentajes de cada elemento de la tupla de vectores comparados, los índices por cada elemento y finalmente el *índice de similitud* $i$, siendo el

**mbw1c**$(k = 820, s = 32, th = 4)$ con un *índice de similitud* de 105 el *perfil de rendimiento* "candidato".

### Tabla de *índices de similitud* para la Aplicación N-Body *workload 200000p*

La Tabla 4.27 presenta los índices de similitud de los *perfiles candidatos* como resultado del *algoritmo de comparación*.

| Perfiles Candidatos Fase N-Body 200000p | Índice de Similitud |
|---|---|
| *mbw1hc (k=820,s=32,th=4)* | 105 |

Tabla 4.27: Tabla de *índices de similitud* de los $perfiles$ candidatos.

### Estimación N-Body sobre todos los nodos de cómputo

Una vez identificado el *perfil de rendimiento* candidato, seleccionamos el *perfil de rendimiento* **mbw1hc**$(k = 820, s = 32, th = 4)$ de la base de datos NPPDB que contiene los valores obtenidos previamente para este $\mu B$ o perfil en el resto de los nodos de cómputo, y con esta información, y considerando el peso, se puede hacer la estimación para esta fase de la aplicación N-Body, sobre todos los nodos. En este caso, esta fase era la única fase significativa identificada, por lo tanto permite hacer la estimación de la aplicación N-Body.

La Tabla 4.28 presenta el resumen de las medidas obtenidas del perfil de rendimiento sobre los diferentes nodos de cómputo, las medidas de rendimiento real de la caracterización de la fase de la aplicación en el nodo base (BN) y el tiempo estimado del rendimiento de la fase de la aplicación sobre los diferentes nodos de cómputo.

La primera columna (BN) de la Tabla 4.28 refleja las medidas obtenidas sobre el nodo de referencia y los cálculos obtenidos siguiendo la metodología propuesta. Además, las columnas TN1, TN2 y TN3, de la Tabla 4.28 presentan los tiempos de ejecución de la fase de la aplicación sobre los

| | | BN | TN1 | TN2 | TN3 |
|---|---|---|---|---|---|
| *Perfil de Rendimiento* | µB: mbw1hc (s) | 2.85 | 2.48 | 3.98 | 3.73 |
| | #Iteraciones (xE6) | 190 | | | |
| | Tiempo /#Iteraciones (µs) | 0.01502 | 0.01305 | 0.02095 | 0.01963 |
| *Caracterización de la fase* | *Tiempo Real de la Fase (s)* | 1563.37 | ( 1342.75 )* | ( 2506.79 )* | ( 2425.63 )* |
| | #Iteraciones (xE6) | 99999 | | | |
| | Tiempo /#Iteraciones (µs) | 0.01563 | | | |
| *Estimación* | *Tiempo Estimado de la Fase (s)* | 1501.56 | 1305.25 | 2094.72 | 1963.14 |
| | Diferencia (%) | 3.95% | 2.79% | 16.41% | 19.05% |

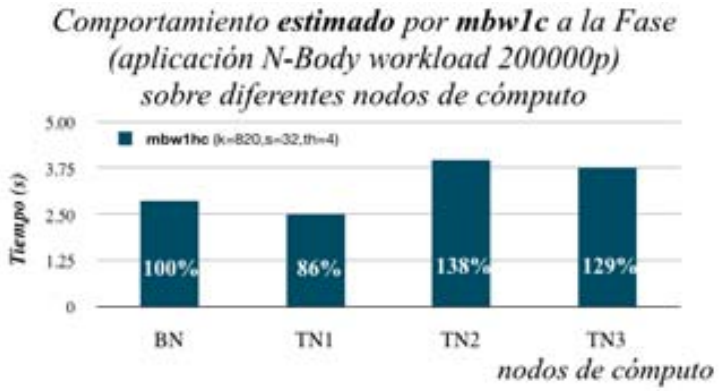
Tabla 4.28: Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación).

diferentes nodos de computo, éstas medidas se han realizado sólo con el propósito de validar la estimación del rendimiento.

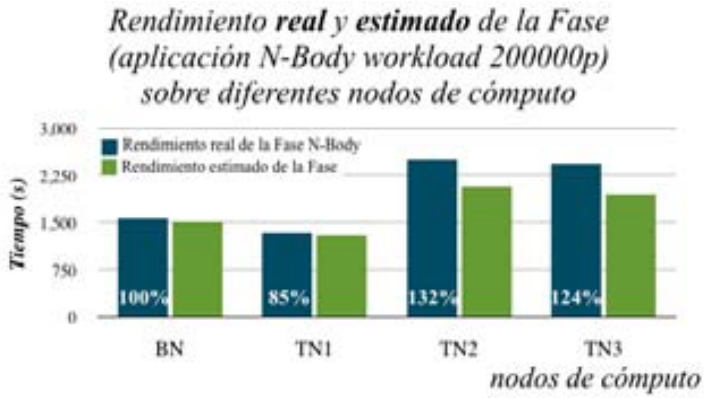
A su vez, en la Tabla 4.28 se muestran las diferencias en porcentaje entre el comportamiento estimado y el real. Se puede apreciar que las diferencias en porcentaje de la estimación no superan el 20 %.

En la Figura 4.29 se presentan los comportamientos **estimado** (Figura 4.29a) y **real** (Figura 4.29b) de la aplicación N-Body. En el eje $x$, los nodos de cómputo BN o sobre el que se ha realizado el análisis, TN1, TN2 y TN3. En el eje $y$, el tiempo de ejecución en segundos.

En la Figura 4.29a se muestra la estimación del comportamiento dado por el *perfil de rendimiento* $\mu B$ **mbw1c(**$k = 820, s = 32, th = 4$**)** sobre los diferentes nodos de cómputo caracterizados. En cada barra se ha colocado el % en relación al nodo de referencia BN 100 %. La gráfica de estimación muestra la diferencia que existe entre los nodos, el nodo que el método estima que ejecutará la aplicación en menos tiempo es el nodo TN1, que




(a) Comportamiento estimado por **mbw1c**




(b) Comportamiento real de la Fase (aplicación N-Body workload 200000p)

Figura 4.29: Estimación y comportamiento real de la Fase (aplicación N-Body) sobre los diferentes nodos de cómputo caracterizados.

para esta aplicación muestra que es un 24 %, 52 % y 43 % más rápido que los nodos BN, TN2 y TN3 respectivamente.

En la Figura 4.29b se presenta el comportamiento real de la aplicación N-Body *workload 200000p*, estas medidas se han realizado con el objetivo de validar la metodología. De igual forma que en la figura anterior el 100 % representa el comportamiento sobre el nodo referencia BN. Se puede apreciar comportamientos similares entre la estimación, resultado de aplicar la

metodología propuesta, y el comportamiento real, donde se ve que el nodo más rápido al ejecutar la aplicación también es el TN1.

### Caracterización Aplicación: N-Body *workload 320000p*

Se definen como valores de entrada el numero de partículas: 320000p. El total de pasos de simulación y el incremento del tiempo son fijados en 10 para todos los casos.

El objetivo es obtener la tupla o registro conformado por las características de cada *flujo de acceso a memoria* (v).

Fase($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

#### Métricas de rendimiento

Se ejecuta el simulador N-Body en el BN, para identificar las fases significativas (Figura 3.9a), para ello ejecutamos la aplicación y obtenemos las fases, el tiempo de ejecución de la fase y el tiempo de ejecución total.

Se analiza y se calcula el peso correspondiente. La Tabla 4.23, presenta en resumen el tiempo de ejecución, los tiempos de las fases identificadas y sus respectivos pesos porcentuales.

En la Tabla 4.30 se observan tanto el tiempo de ejecución total de la aplicación N-Body, como el tiempo de la fase o $fork\ join$ principal, con esta información observamos que el peso de la fase es significativo (97,22 % del tiempo total de ejecución) y pasamos a analizar la fase. Adicionalmente, se normaliza la métrica de comportamiento, se calcula el *Tiempo/#Iteraciones* como tiempo de ejecución de cada fase dividido por el total de iteraciones.

| Aplicación | Tiempo Total (s) | Tiempo Fase (*fork join* principal) (s) |
|---|---|---|
| N-Body | 4101,01 | 3987,99 |
| Peso Porcentual | 100 % | 97,22 % |

Tabla 4.30: Tiempo de ejecución, tiempo de la fase identificada y peso porcentual de la aplicación N-Body *workload 320000p*

A la fase identificada, procedemos a instrumentar para obtener los flujos de acceso a memoria, a partir de esa traza de acceso a memoria, se analizará para en cada flujo el patrón de accesos a memoria y su respectivo tipo de dato, con el objeto de tener la información de caracterización de la fase.

***Fase caracterizada***

La Tabla 4.31, presenta la información relativa a la caracterización de la aplicación N-Body *workload 320000p*, con un sólo *flujo* significativo *de acceso a memoria* (*stream* ***v1***).

| | Fase N-Body *workload 320000p* | | | | | | |
|---|---|---|---|---|---|---|---|
| | Tamaño (kB) | Stride | Tipo de Dato | Acceso | Tiempo/ #Iteraciones (µs) | #Iteraciones (xE6) | Tiempo (s) |
| ***stream v1*** | 1249 | 32 | 8B | C | 0.01558 | 255997 | 3,987.99 |

Tabla 4.31: Caracterización de la Fase NBody *workload 320000p*

Se identifican un *flujo de acceso a memoria* que hemos denominado *stream* ***v1***, accede a memoria con un *stride* de 32. La zona de memoria a la que accede *stream* ***v1*** es *compartida* (C).

**Consulta *q1***

Los *perfiles de rendimiento* de la Tabla 4.32 son compuestos por un flujo de acceso a memoria. Hemos seleccionado estos perfiles para ilustrar *algoritmo de comparación* para la consulta $q1$.

| | *Tamaño* | *Stride* | *Tipo Dato* | *Acceso* | *Tiempo/ #Iteraciones* |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw1c*** | 820 | 32 | 8B | C | 0.01500 |
| | 1367 | 32 | 8B | C | 0.01575 |
| ***mbw1p*** | 820 | 32 | 8B | P | 0.02753 |
| | 1367 | 32 | 8B | P | 0.03318 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.32: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$. Todas las pruebas se han realizado para $th = 4$.

La Tabla 4.32 presenta 4 cuatro *perfiles de rendimiento* seleccionados de la base de datos NPPDB para aplicar el *algoritmo de comparación*.

| | | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | % | *i*T | % | *i*S | *i*D | *i*A | % | *i*Z | *I* |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***q1*** | ∩ | ***mbw1c*** | 34.35% | 0 | 0% | 25 | 25 | 25 | 3.72% | 15 | 90 |
| ***q1*** | ∩ | ***mbw1c*** | 9.45% | 10 | 0% | 25 | 25 | 25 | 1.08% | 25 | 110 |
| ***q1*** | ∩ | ***mbw1p*** | 34.35% | 0 | 0% | 25 | 25 | 0 | 76.14% | 0 | 50 |
| ***q1*** | ∩ | ***mbw1p*** | 9.45% | 10 | 0% | 25 | 25 | 0 | 112.26% | Desc. | Desc. |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.33: Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.32.

En la Tabla 4.33 se pueden apreciar las diferencias en porcentajes de

cada elemento de la tupla de vectores comparados, los índices por cada elemento y finalmente el *índice de similitud* $i$, siendo el **mbw1c**($k = 1367, s = 8, th = 4$) con un *índice de similitud* de 110 el *perfil de rendimiento* "candidato".

### Tabla de *índices de similitud* para la Aplicación N-Body *workload 320000p*

La Tabla 4.34 presenta los índices de similitud de los *perfiles candidatos* como resultado del *algoritmo de comparación*.

| Perfiles Candidatos Fase N-Body 320000p | Índice de Similitud |
|---|---|
| *mbw1hc (k=1367,s=32,th=4)* | 110 |

Tabla 4.34: Tabla de *índices de similitud* de los $perfiles$ candidatos.

### Estimación N-Body sobre todos los nodos de cómputo

Una vez identificado el *perfil de rendimiento* candidato, seleccionamos el *perfil de rendimiento* **mbw1hc**($k = 1367, s = 32, th = 4$) de la base de datos NPPDB que contiene los valores obtenidos previamente para este $\mu B$ o perfil en el resto de los nodos de cómputo, y con esta información, y considerando el peso, se puede hacer la estimación para esta fase de la aplicación N-Body, sobre todos los nodos. En este caso, esta fase era la única fase significativa identificada, por lo tanto permite hacer la estimación de la aplicación N-Body.

La Tabla 4.35 presenta el resumen de las medidas obtenidas del perfil de rendimiento sobre los diferentes nodos de cómputo, las medidas de rendimiento real de la caracterización de la fase de la aplicación en el nodo base (BN) y el tiempo estimado del rendimiento de la fase de la aplicación sobre los diferentes nodos de cómputo.

| | | BN | TN1 | TN2 | TN3 |
|---|---|---|---|---|---|
| *Perfil de Rendimiento* | µB: mbw1hc (s) | 5.49 | 5.59 | 7.95 | 7.88 |
| | #Iteraciones (xE6) | 350 | | | |
| | Tiempo /#Iteraciones (µs) | 0.01568 | 0.01597 | 0.02271 | 0.02251 |
| *Caracterización de la fase* | *Tiempo Real de la Fase (s)* | 3987.99 | ( 4230.39 )* | ( 7333.89 )* | ( 6838.62 )* |
| | #Iteraciones (xE6) | 255997 | | | |
| | Tiempo /#Iteraciones (µs) | 0.01558 | | | |
| *Estimación* | *Tiempo Estimado de la Fase (s)* | 4013.30 | 4088.53 | 5812.81 | 5763.33 |
| | Diferencia (%) | 0.63% | 3.35% | 20.74% | 15.72% |

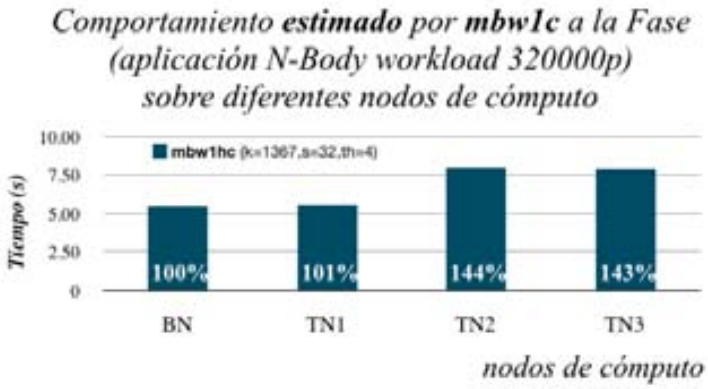
Tabla 4.35: Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación).

La primera columna (BN) de la Tabla 4.35 refleja las medidas obtenidas sobre el nodo de referencia y los cálculos obtenidos siguiendo la metodología propuesta. Además, las columnas TN1, TN2 y TN3, de la Tabla 4.35 presentan los tiempos de ejecución de la fase de la aplicación sobre los diferentes nodos de computo, éstas medidas se han realizado sólo con el propósito de validar la estimación del rendimiento.

A su vez, en la Tabla 4.35 se muestran las diferencias en porcentaje entre el comportamiento estimado y el real. Se puede apreciar que las diferencias en porcentaje de la estimación no superan el 21 %.

En la Figura 4.36 se presentan los comportamientos **estimado** (Figura 4.36a) y **real** (Figura 4.36b) de la aplicación N-Body. En el eje $x$, los nodos de cómputo BN o sobre el que se ha realizado el análisis, TN1, TN2 y TN3. En el eje $y$, el tiempo de ejecución en segundos.

En la Figura 4.36a se muestra la estimación del comportamiento dado por el *perfil de rendimiento* $\mu B$ **mbw1c**$(k = 1367, s = 32, th = 4)$ sobre los diferentes nodos de cómputo caracterizados. En cada barra se ha colocado




(a) Comportamiento estimado por **mbw1c**


(b) Comportamiento real Fase (Aplicación N-Body workload 320000p)

Figura 4.36: Estimación y comportamiento real de la Fase (aplicación N-Body) sobre los diferentes nodos de cómputo caracterizados.

el % en relación al nodo de referencia BN 100 %. La gráfica de estimación muestra la diferencia que existe entre los nodos, el nodo que el método estima que ejecutará la aplicación en menos tiempo es el nodo BN, que para esta aplicación muestra que es un 1 %, 44 % y 43 % más rápido que los nodos TN1, TN2 y TN3 respectivamente.

En la Figura 4.36b se presenta el comportamiento real de la aplicación N-Body *workload 320000p*, estas medidas se han realizado con el objetivo

de validar la metodología. De igual forma que en la figura anterior el 100 % representa el comportamiento sobre el nodo referencia BN. Se puede apreciar comportamientos similares entre la estimación, resultado de aplicar la metodología propuesta, y el comportamiento real, donde se ve que el nodo más rápido al ejecutar la aplicación también es el BN.

### 4.3.3. Bloque Tridiagonal Nas Parallel Benchmark: análisis de con varias fases representativas

#### Caracterización Aplicación: BT Nas Parallel Benchmark *workload ClaseA*

El Bloque Tridiagonal BT del Nas Parallel Benchmark se analiza para la Clase A.

Buscamos la tupla o registro conformado por las características de cada *flujo de acceso a memoria* (v), para la aplicación BT Clase A.

Fase($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

#### Métricas de rendimiento: Clase A

Se ejecuta la aplicación BT en el BN, para identificar las fases significativas (Figura 3.9a), para ello ejecutamos la aplicación y obtenemos las fases, el tiempo de ejecución de la fase y el tiempo de ejecución total.

Se analiza y se calcula el peso correspondiente. La Tabla 4.37, presenta en resumen el tiempo de ejecución, los tiempos de las fases identificadas y sus respectivos pesos porcentuales.

En la Tabla 4.37 se observan tanto el tiempo de ejecución total de la aplicación BT NAS Parallel Benchmark, como el tiempo de la fase o $fork\ join$ principal, con esta información observamos que el peso de la fase es significativo (97,22 % del tiempo total de ejecución) y pasamos a analizar la

| | Workload ClaseA (s) | Peso (%) |
|---|---|---|
| *Tiempo total de ejecución* | 123.68 | |
| *Tiempo fase bt.1* | 17.63 | 14.25 |
| *Tiempo fase bt.2* | 13.01 | 10.52 |
| *Tiempo fase bt.3* | 12.88 | 10.41 |
| *Promedio tiempo de fases* | 14.51 | |
| *Tiempo total de las 3 fases* | 43.52 | 35.19 |

Tabla 4.37: Tiempo de ejecución, tiempo de la fase identificada y peso porcentual de la aplicación BT NAS Parallel Benchmark *workload ClaseA* y *workload ClaseB*

fase. Adicionalmente, se normaliza la métrica de comportamiento, se calcula el *Tiempo/#Iteraciones* como tiempo de ejecución de cada fase dividido por el total de iteraciones.

El promedio del tiempo de las fases es de un 35 % del tiempo total de ejecución de la aplicación. A la fase identificada, procedemos a instrumentar para obtener los flujos de acceso a memoria, a partir de esa traza de acceso a memoria, se analizará para en cada flujo el patrón de accesos a memoria y su respectivo tipo de dato, con el objeto de tener la información de caracterización de la fase para cada *workload*. El tiempo restante de la aplicación o 65 % esta divido en pequeñas o poco significativas fases de ejecución con respecto al tiempo total de ejecución; por lo tanto no son consideradas como fases representativas a caracterizar ya que no influyen significativamente en el rendimiento de la aplicación.

#### ***Fase caracterizada BT* workload ClaseA**

La Tabla 4.38, presenta la información relativa a la caracterización de la aplicación BT *workload ClassA*, con un sólo *flujo* significativo *de acceso a memoria* (*stream* ***v1***).

| | Fase BT *workload ClaseA* | | | | | | |
|---|---|---|---|---|---|---|---|
| | Tamaño (kB) | Stride | Tipo de Dato | Acceso | Tiempo/ #Iteraciones (μs) | #Iteraciones (xE6) | Tiempo (s) |
| ***stream v1*** | 4717 | 600 | 8B | P | 1.21177 | 12 | 14.5412 |

Tabla 4.38: Caracterización de la Fase BT *workload ClaseA*

Se identifican un *flujo de acceso a memoria* que hemos denominado *stream* ***v1***, accede a memoria con un *stride* de 600. La zona de memoria a la que accede *stream* ***v1*** es *compartida* (P).

### Proceso de comparación: BT *ClaseA*

El *proceso de comparación* busca identificar en la base de datos, todos los $\mu Bs$ ejecutados en el nodo en el que se está ejecutando la aplicación y que tienen un patrón de acceso similar a la memoria. La consulta esta compuesta por la combinación sin repetición para un solo *stream*.

$$q = \{[v1]\}$$

Se tiene entonces una consulta a realizar el *algoritmo de comparación*, identificamos el grupo a comparar como $q1$.

#### Consulta *q1*

Los *perfiles de rendimiento* de la Tabla 4.39 son compuestos por un flujo de acceso a memoria *stream v1*. Hemos seleccionado estos perfiles para ilustrar *algoritmo de comparación* para la consulta $q1$.

En la Tabla 4.40 se pueden apreciar las diferencias en porcentajes de cada elemento de la tupla de vectores comparados, los índices por cada elemento y finalmente el *índice de similitud* $i$, siendo el **mbw1Up(**$k = 4925, s = 600, th = 4$**)** con un *índice de similitud* de 100 el *perfil de rendimiento* "candidato".

| | *Tamaño* | *Stride* | *Tipo Dato* | *Acceso* | *Tiempo/ #Iteraciones* |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw1Uc*** | 4925 | 600 | 8B | C | 1.10073 |
| | 18831 | 600 | 8B | C | 1.00718 |
| ***mbw1Up*** | 4925 | 600 | 8B | P | 1.12469 |
| | 18831 | 600 | 8B | P | 1.22920 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.39: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$. Todas las pruebas se han realizado para $th = 4$.

| | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|
| | % | $i$T | % | $i$S | $i$D | $i$A | % | $i$Z | $i$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $q1$ ∩ ***mbw1Uc*** | 4.41% | 15 | 0% | 25 | 25 | 0 | 9.16% | 10 | 75 |
| $q1$ ∩ ***mbw1Uc*** | 299.22% | Desc. | 0% | 25 | 25 | 0 | 16.88% | 0 | Desc. |
| $q1$ ∩ ***mbw1Pc*** | 4.41% | 15 | 0% | 25 | 25 | 25 | 7.19% | 10 | 100 |
| $q1$ ∩ ***mbw1Pc*** | 299.22% | Desc. | 0% | 25 | 25 | 25 | 1.44% | Desc. | Desc. |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.40: Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla.

### Tabla de *índices de similitud* para la Aplicación BT *workload ClaseA*

La Tabla 4.41 presenta los índices de similitud de los *perfiles candidatos* como resultado del *algoritmo de comparación*.

### Estimación BT *workload ClaseA* sobre todos los nodos de cómputo

Una vez identificado el *perfil de rendimiento* candidato, seleccionamos el *perfil de rendimiento* **mbw1Up**$(k = 4925, s = 600, th = 4)$ de la base

| Perfiles Candidatos Fase BT workload ClaseA | Índice de Similitud |
|---|---|
| *mbw1Up* *(k = 4925,x = 600,th = 4)* | 100 |

Tabla 4.41: Tabla de *índices de similitud* de los $perfiles$ candidatos.

de datos NPPDB que contiene los valores obtenidos previamente para este $\mu B$ o perfil en el resto de los nodos de cómputo, y con esta información, y considerando el peso, se puede hacer la estimación para esta fase de la aplicación BT, sobre todos los nodos. Permite hacer la estimación del promedio de tiempo de la ejecución de la aplicación BT *workload ClaseA*.

| | | BN | TN1 | TN2 | TN3 |
|---|---|---|---|---|---|
| *Perfil de Rendimiento* | µB: mbw1Up (s) | 1.43 | 1.76 | 1.27 | 1.17 |
| | #Iteraciones (xE6) | 1.27 | | | |
| | Tiempo /#Iteraciones (µs) | 1.12653 | 1.39214 | 1.00264 | 0.92134 |
| *Caracterización de la fase* | *Tiempo Real de la Fase (s)* | 14.54 | ( 16.39 )* | ( 13.11 )* | ( 8.35 )* |
| | #Iteraciones (xE6) | 12 | | | |
| | Tiempo /#Iteraciones (µs) | 1.21177 | | | |
| *Estimación* | *Tiempo Estimado de la Fase (s)* | 13.52 | 16.71 | 12.03 | 11.06 |
| | Diferencia (%) | 7.03% | 1.92% | 8.23% | 31.93% |

Tabla 4.42: Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación).

La Tabla 4.42 presenta el resumen de las medidas obtenidas del perfil de rendimiento sobre los diferentes nodos de cómputo, las medidas de rendimiento real de la caracterización de la fase de la aplicación en el nodo base (BN) y el tiempo estimado del rendimiento de la fase de la aplicación sobre los diferentes nodos de cómputo.

La primera columna (BN) de la Tabla 4.42 refleja las medidas obtenidas sobre el nodo de referencia y los cálculos obtenidos siguiendo la metodología propuesta. Además, las columnas TN1, TN2 y TN3, de la Tabla 4.42 presentan los tiempos de ejecución de la fase de la aplicación sobre los diferentes nodos de computo, éstas medidas se han realizado sólo con el propósito de validar la estimación del rendimiento.

A su vez, en la Tabla 4.42 se muestran las diferencias en porcentaje entre el comportamiento estimado y el real. Se puede apreciar que las diferencias en porcentaje de la estimación no superan el 32 %.

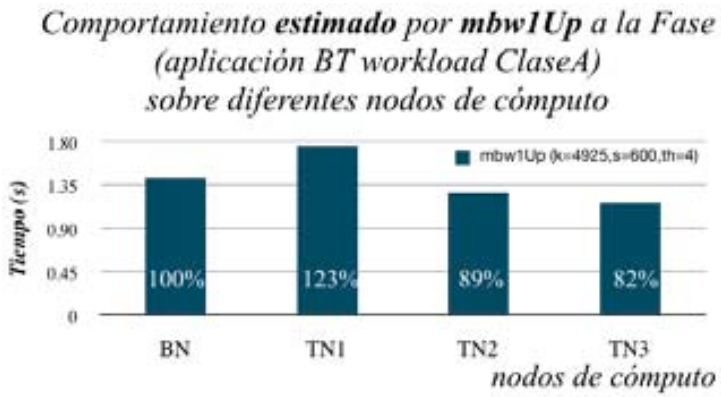
En la Figura 4.43 se presentan los comportamientos **estimado** (Figura 4.43a) y **real** (Figura 4.43b) de la aplicación BT NAS Parallel Benchmark. En el eje $x$, los nodos de cómputo BN o sobre el que se ha realizado el análisis, TN1, TN2 y TN3. En el eje $y$, el tiempo de ejecución en segundos.

En la Figura 4.43a se muestra la estimación del comportamiento dado por el *perfil de rendimiento* $\mu B$ **mbw1c(**$k = 4925, s = 600, th = 4$**)** sobre los diferentes nodos de cómputo caracterizados. En cada barra se ha colocado el % en relación al nodo de referencia BN 100 %. La gráfica de estimación muestra la diferencia que existe entre los nodos, el nodo que el método estima que ejecutará la aplicación en menos tiempo es el nodo TN3, que para esta aplicación muestra que es un 18 %, 41 % y 7 % más rápido que los nodos BN, TN1 y TN2 respectivamente.

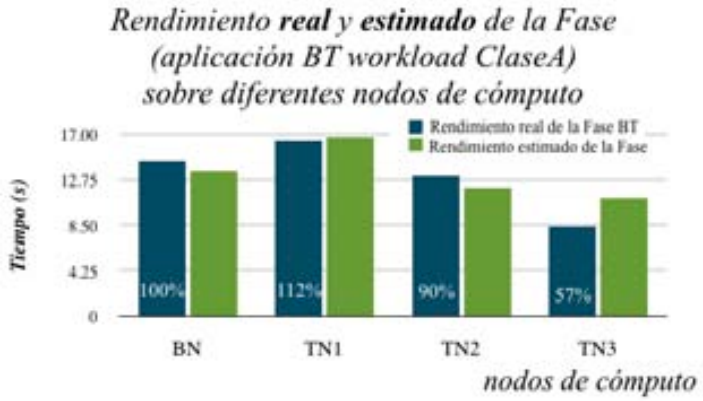
En la Figura 4.12b se presenta el comportamiento real de la aplicación BT *workload ClaseA*, estas medidas se han realizado con el objetivo de validar la metodología. De igual forma que en la figura anterior el 100 % representa el comportamiento sobre el nodo referencia BN. Se puede apreciar comportamientos similares entre la estimación, resultado de aplicar la metodología propuesta, y el comportamiento real, donde se ve que el nodo más rápido al ejecutar la aplicación también es el TN3.

### Caracterización Aplicación: BT Nas Parallel Benchmark *workload ClaseB*

El Bloque Tridiagonal BT del Nas Parallel Benchmark se analiza para la Clase B.




(a) Comportamiento estimado por **mbw1Up**




(b) Comportamiento real Fase (aplicación BT *workload* ClaseA)

Figura 4.43: Estimación y comportamiento real de la Fase (aplicación BT NAS Parallel Benchmark), sobre los diferentes nodos de cómputo caracterizados.

Buscamos la tupla o registro conformado por las características de cada *flujo de acceso a memoria* (v), para la aplicación BT Clase B.

Fase($v$)=("Tamaño","Stride","TipoDato",“Acceso","Tiempo/#Iteraciones")

**Métricas de rendimiento: Clase B**

Se ejecuta la aplicación BT en el BN, para identificar las fases signifi-

cativas (Figura 3.9a), para ello ejecutamos la aplicación y obtenemos las fases, el tiempo de ejecución de la fase y el tiempo de ejecución total.

| | Workload ClaseB (s) | Peso (%) |
|---|---|---|
| *Tiempo total de ejecución* | 550.40 | |
| *Tiempo fase bt.1* | 73.42 | 13.34 |
| *Tiempo fase bt.2* | 53.92 | 9.80 |
| *Tiempo fase bt.3* | 53.47 | 9.71 |
| *Promedio tiempo de fases* | 60.27 | |
| *Tiempo total de las 3 fases* | 180.81 | 32.85 |

Tabla 4.44: Tiempo de ejecución, tiempo de la fase identificada y peso porcentual de la aplicación BT NAS Parallel Benchmark *workload ClaseA* y *workload ClaseB*

Se analiza y se calcula el peso correspondiente. La Tabla 4.44, presenta en resumen el tiempo de ejecución, los tiempos de las fases identificadas y sus respectivos pesos porcentuales.

En la Tabla 4.44 se observan tanto el tiempo de ejecución total de la aplicación BT NAS Parallel Benchmark, como el tiempo de la fase o $fork$ $join$ principal, con esta información observamos que el peso de la fase es significativo (97,22 % del tiempo total de ejecución) y pasamos a analizar la fase. Adicionalmente, se normaliza la métrica de comportamiento, se calcula el *Tiempo/#Iteraciones* como tiempo de ejecución de cada fase dividido por el total de iteraciones.

El promedio del tiempo de las fases es de un 35 % del tiempo total de ejecución de la aplicación. A la fase identificada, procedemos a instrumentar para obtener los flujos de acceso a memoria, a partir de esa traza de acceso a memoria, se analizará para en cada flujo el patrón de accesos a memoria y su respectivo tipo de dato, con el objeto de tener la información de caracterización de la fase para cada *workload*. El tiempo restante de la aplicación o 65 % esta divido en pequeñas o poco significativas fases de ejecución con respecto al tiempo total de ejecución; por lo tanto no son consideradas como fases representativas a caracterizar ya que no influyen

significativamente en el rendimiento de la aplicación.

| Fase BT *workload ClaseB* | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Tamaño (kB) | Stride | Tipo de Dato | Acceso | Tiempo/ #Iteraciones (μs) | #Iteraciones (xE6) | Tiempo (s) |
| ***stream v1*** | 18836 | 600 | 8B | P | 1.20568 | 50 | 60.2838 |

Tabla 4.45: Caracterización de la Fase BT *workload ClaseB*

#### ***Fase caracterizada BT*** **workload ClaseB**

La Tabla 4.45, presenta la información relativa a la caracterización de la aplicación BT *workload ClassB*, con un sólo *flujo* significativo *de acceso a memoria* (*stream* ***v1***).

Se identifican un *flujo de acceso a memoria* que hemos denominado *stream* ***v1***, accede a memoria con un *stride* de 600. La zona de memoria a la que accede *stream* ***v1*** es *compartida* (P).

### Proceso de comparación: BT *ClaseB*

El *proceso de comparación* busca identificar en la base de datos, todos los $\mu Bs$ ejecutados en el nodo en el que se está ejecutando la aplicación y que tienen un patrón de acceso similar a la memoria. La consulta esta compuesta por la combinación sin repetición para un solo *stream*.

$$q = \{[v1]\}$$

Se tiene entonces una consulta a realizar el *algoritmo de comparación*, identificamos el grupo a comparar como $q1$.

**Consulta *q1***

Los *perfiles de rendimiento* de la Tabla 4.46 son compuestos por un flujo de acceso a memoria *stream v1*. Hemos seleccionado estos perfiles para ilustrar *algoritmo de comparación* para la consulta $q1$.

| | *Tamaño* | *Stride* | *Tipo Dato* | *Acceso* | *Tiempo/ #Iteraciones* |
|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***mbw1Uc*** | 4925 | 600 | 8B | C | 1.10073 |
| | 18831 | 600 | 8B | C | 1.00718 |
| ***mbw1Up*** | 4925 | 600 | 8B | P | 1.12469 |
| | 18831 | 600 | 8B | P | 1.22920 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.46: Ejemplo de información de la base de datos: *perfiles de rendimiento* a comparar con $q1$. Todas las pruebas se han realizado para $th = 4$.

| | | Tamaño | | Stride | | TipoDato | Acceso | Tiempo/ #Iteraciones | | Índice de Similitud |
|---|---|---|---|---|---|---|---|---|---|---|
| | | % | *i*T | % | *i*S | *i*D | *i*A | % | *i*Z | *i* |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ***q1*** | ∩ ***mbw1Uc*** | 73.85% | 0 | 0% | 25 | 25 | 0 | 8.70% | 10 | 60 |
| ***q1*** | ∩ ***mbw1Uc*** | 0.03% | 25 | 0% | 25 | 25 | 0 | 16.46% | 0 | 75 |
| ***q1*** | ∩ ***mbw1Pc*** | 73.85% | 0 | 0% | 25 | 25 | 25 | 6.72% | 10 | 85 |
| ***q1*** | ∩ ***mbw1Pc*** | 0.03% | 25 | 0% | 25 | 25 | 25 | 1.95% | 25 | 125 |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Tabla 4.47: Algoritmo de Comparación: consulta $q1$ frente a información de *perfiles de rendimiento* Tabla 4.46.

En la Tabla 4.47 se pueden apreciar las diferencias en porcentajes de cada elemento de la tupla de vectores comparados, los índices por cada elemento y finalmente el *índice de similitud* $i$, siendo el **mbw1Up**

$(k = 18831, s = 600, th = 4)$ con un *índice de similitud* de 125 el *perfil de rendimiento* "candidato".

### Tabla de *índices de similitud* para la Aplicación BT *workload ClaseB*

La Tabla 4.41 presenta los índices de similitud de los *perfiles candidatos* como resultado del *algoritmo de comparación.*

| Perfiles Candidatos Fase BT workload ClaseB | Índice de Similitud |
|---|---|
| *mbw1Up* *(k = 18831,s = 600,th = 4)* | 125 |

Tabla 4.48: Tabla de *índices de similitud* de los $perfiles$ candidatos.

### Estimación BT *workload ClaseB* sobre todos los nodos de cómputo

Una vez identificado el *perfil de rendimiento* candidato, seleccionamos el *perfil de rendimiento* **mbw1Up**$(k = 18831, s = 600, th = 4)$ de la base de datos NPPDB que contiene los valores obtenidos previamente para este $\mu B$ o perfil en el resto de los nodos de cómputo, y con esta información, y considerando el peso, se puede hacer la estimación para esta fase de la aplicación BT, sobre todos los nodos. Permite hacer la estimación del promedio de tiempo de la ejecución de la aplicación BT *workload ClaseB*.

La Tabla 4.49 presenta el resumen de las medidas obtenidas del perfil de rendimiento sobre los diferentes nodos de cómputo, las medidas de rendimiento real de la caracterización de la fase de la aplicación en el nodo base (BN) y el tiempo estimado del rendimiento de la fase de la aplicación sobre los diferentes nodos de cómputo.

La primera columna (BN) de la Tabla 4.49 refleja las medidas obtenidas sobre el nodo de referencia y los cálculos obtenidos siguiendo la metodología propuesta. Además, las columnas TN1, TN2 y TN3, de la Tabla 4.49

| | | BN | TN1 | TN2 | TN3 |
|---|---|---|---|---|---|
| *Perfil de Rendimiento* | µB: mbw1Up (s) | 6.06 | 7.80 | 5.31 | 4.48 |
| | #Iteraciones (xE6) | 4.93 | | | |
| | Tiempo /#Iteraciones (µs) | 1.22920 | 1.58369 | 1.07708 | 0.91009 |
| *Caracterización de la fase* | *Tiempo Real de la Fase (s)* | 60.28 | ( 60.28 )* | ( 68.22 )* | ( 36.99 )* |
| | #Iteraciones (xE6) | 50 | | | |
| | Tiempo /#Iteraciones (µs) | 1.20568 | | | |
| *Estimación* | *Tiempo Estimado de la Fase (s)* | 61.46 | 79.18 | 53.85 | 45.50 |
| | Diferencia (%) | 1.95% | 16.06% | 0.51% | 22.99% |

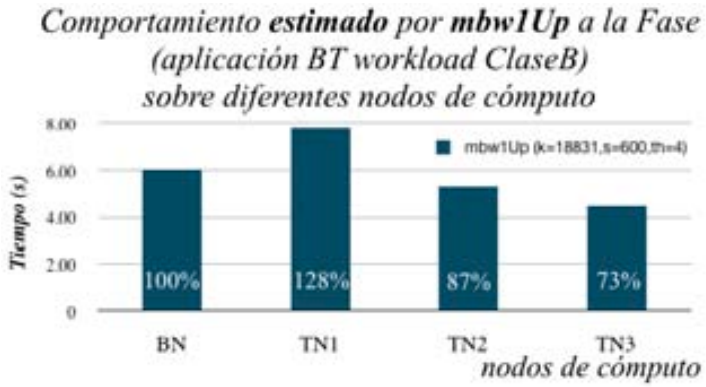
Tabla 4.49: Resumen resultados obtenidos del *perfil de rendimiento* candidato y medidas de rendimiento real de la fase de la aplicación sobre diferentes nodos de cómputo. (*Resultados obtenidos con objeto de validar la estimación).

presentan los tiempos de ejecución de la fase de la aplicación sobre los diferentes nodos de computo, éstas medidas se han realizado sólo con el propósito de validar la estimación del rendimiento.

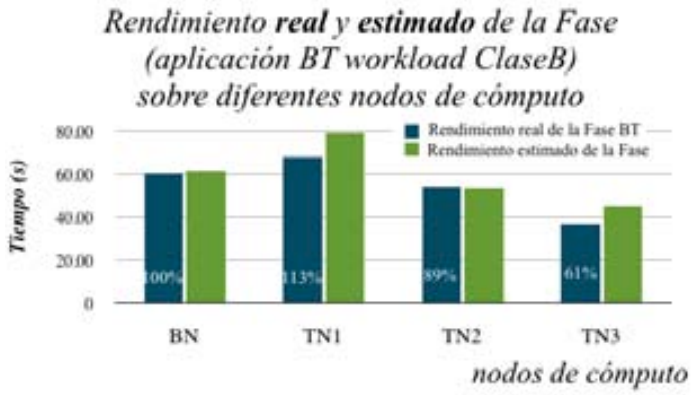
A su vez, en la Tabla 4.49 se muestran las diferencias en porcentaje entre el comportamiento estimado y el real. Se puede apreciar que las diferencias en porcentaje de la estimación no superan el 23 %.

En la Figura 4.50 se presentan los comportamientos **estimado** (Figura 4.50a) y **real** (Figura 4.50b) de la aplicación BT NAS Parallel Benchmark. En el eje $x$, los nodos de cómputo BN o sobre el que se ha realizado el análisis, TN1, TN2 y TN3. En el eje $y$, el tiempo de ejecución en segundos.

En la Figura 4.50a se muestra la estimación del comportamiento dado por el *perfil de rendimiento* $\mu B$ **mbw1c(**$k = 18831, s = 600, th = 4$**)** sobre los diferentes nodos de cómputo caracterizados. En cada barra se ha colocado el % en relación al nodo de referencia BN 100 %. La gráfica de estimación muestra la diferencia que existe entre los nodos, el nodo que el método estima que ejecutará la aplicación en menos tiempo es el nodo TN3,




(a) Comportamiento estimado por **mbw1Up**




(b) Comportamiento real de la Fase (Aplicación BT *workload* ClaseB)

Figura 4.50: Estimación y comportamiento real de la Fase (aplicación BT NAS Parallel Benchmark), sobre los diferentes nodos de cómputo caracterizados.

que para esta aplicación muestra que es un 31 %, 54 % y 17 % más rápido que los nodos BN, TN1 y TN2 respectivamente.

En la Figura 4.12b se presenta el comportamiento real de la aplicación BT *workload ClaseB*, estas medidas se han realizado con el objetivo de validar la metodología. De igual forma que en la figura anterior el 100 % representa el comportamiento sobre el nodo referencia BN. Se puede apre-

ciar comportamientos similares entre la estimación, resultado de aplicar la metodología propuesta, y el comportamiento real, donde se ve que el nodo más rápido al ejecutar la aplicación también es el TN3.

# Capítulo 5

# Conclusiones

Los centros de cómputo en la actualidad disponen de diferentes configuraciones o diseños de nodos de cómputo, recursos disponibles para las aplicaciones científicas paralelas de memoria compartida. La gran variedad de recursos supone la elección de una configuración de nodos de cómputo rápida para ejecutar una aplicación dada. Nuestra metodología tiene como objetivo reflejar el comportamiento de una aplicación y predecirlo sobre diferentes configuraciones de nodos de cómputo previamente caracterizados y así seleccionar la configuración apropiada, en términos de rendimiento.

Una forma de seleccionar el mejor nodo de cómputo para una aplicación es comparar el comportamiento que tendrá dicha aplicación en los diferentes nodos de cómputo que se deseen considerar.

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 una gran variedad de configuraciones de nodos de cómputo.

Los perfiles de rendimiento son extraídos usando microbenchmarks ($\mu Bs$), que son núcleos diseñados para aislar ciertas características que afectan al rendimiento de aplicaciones paralelas. Los tiempos de ejecución de los $\mu B's$ son muy cortos, en relación al tiempo que toma la ejecución de la aplicación. El objetivo de los $\mu B's$ que se han diseñado durante esta fase del trabajo es estresar el sistema de memoria de los nodos de cómputo para

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

Hemos creado un modelo simple para caracterizar la aplicación, hemos diseñado un método para buscar la similitud y hemos mostrado el funcionamiento de la metodología sobre un algoritmo de multiplicación de matrices, para diferentes tamaños de entrada. La caracterización de los algoritmos resulta en un buen emparejamiento con los $\mu B's$ propuestos, y a su vez permite una selección rápida de una configuración de nodo de cómputo apropiada. Para las aplicaciones con las que se ha trabajado la estimación del comportamiento es muy acertada, ya que los *perfiles de rendimiento* seleccionados o candidatos del *proceso de comparación* ordenan en función de prestaciones los diferentes nodos de cómputo, tal y como lo valida la ejecución real.

## 5.1. Principales Contribuciones

Las principales aportaciones de este trabajo están resumidas en las publicaciones realizadas:

- John Corredor, Juan C. Moure, Dolores Rexachs, Daniel Franco, Emilio Luque: Methodology to Predict the Performance Behavior of Shared-memory Parallel Applications on Multicore Systems. PDPTA 2011 [pendiente de publicación]
- John Corredor, Juan C. Moure, Dolores Rexachs, Daniel Franco, Emilio Luque: Selecting a Suitable Multicore System for Shared-memory Parallel Applications. PDPTA 2010: 228-234
- John Corredor, Juan C. Moure, Dolores Rexachs, Daniel Franco, Emilio Luque: Active learning processes to study memory hierarchy on Multicore systems. Procedia CS 1(1): 921-930 (2010)
- John Corredor, Juan C. Moure, Dolores Rexachs, Daniel Franco, Emilio Luque: Predicción del comportamiento de aplicaciones paralelas de memoria compartida sobre nodos multicore. XXI Jornadas de Paralelismo (2010)

- John Corredor, Juan C. Moure, Dolores Rexachs, Daniel Franco, Emilio Luque: Selección de nodos de cómputo multicore para una aplicación paralela de memoria compartida. CACIC-2010: 152-161 (2010)
- John Corredor, Juan C. Moure, Dolores Rexachs, Daniel Franco, Emilio Luque: Behavior prediction of shared-memory parallel applications on multicore sytems. PICCIT-2010 (3rd Palestinian International Conference on Computer and Information Technology) (2010)

## 5.2. Líneas Abiertas

Este trabajo se ha centrado en analizar aplicaciones cuyo rendimiento está limitado por los accesos a memoria, en concreto con matrices densas. En un futuro se quieren analizar otros tipos de aplicaciones como las limitadas por accesos a matrices dispersas, las aplicaciones limitadas por cómputo, o las limitadas por E/S.

Esta tesis presenta una primera versión del algoritmo de similitud. Actualmente se está trabajando para refinar este algoritmo, con la intención de que considere nuevas características que definan aún mejor el comportamiento de las aplicaciones, como también la calidad de la precisión.

Otro aspecto que supone una línea abierta de nuestro trabajo es el diseño de esqueletos que permitan la creación de nuevos $\mu Bs$ a partir del análisis de las aplicaciones. Este proceso es necesario cuando para ciertas aplicaciones no se encuentren *perfiles de rendimiento* similares.

# Bibliografía


[1] "Multi-core, threads & message passing." [Online]. Available: http://www.igvita.com/2010/08/18/multi-core-threads-message-passing

[2] V. S. Adve and M. K. Vernon, "Parallel program performance prediction using deterministic task graph analysis," *ACM Trans. Comput. Syst.*, vol. 22, pp. 94–136, February 2004. [Online]. Available: http://doi.acm.org/10.1145/966785.966788

[3] D. Bailey, E. Barszcz, D. Browning, R. Carter, L. Dagun, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga, "The NAS parallel benchmarks," in *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing 1991*. New York, NY, USA: ACM, 1991.

[4] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas, "Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures," in *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, ser. MICRO 33. New York, NY, USA: ACM, 2000, pp. 245–257. [Online]. Available: http://doi.acm.org/10.1145/360128.360153

[5] R. Bell and L. John, "Efficient power analysis using synthetic testcases," *IEEE Workload Characterization Symposium*, vol. 0, pp. 110–118, 2005.

[6] L. Carrington, A. Snavely, and N. Wolter, "A performance prediction framework for scientific applications," *Future Generation Computer Systems*, vol. 22, no. 3, pp. 336 – 346, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/B6V06-4F53N8K-3/2/9166a099a551a9e86a58105d0a60be8e

[7] J. Corredor, J. C. Moure, D. Rexachs, D. Franco, and E. Luque, "Active learning processes to study memory hierarchy on multicore systems," *Procedia Computer Science*, vol. 1, no. 1, pp. 921 – 930, 2010, iCCS 2010.

[8] ——, "Behavior prediction of shared-memory parallel application on multicore systems," *The 3erd Palestinian International Conference on Computer and Information Technology*, vol. 1, no. 1, pp.–, 2010, pICCIT 2010.

[9] ——, "Prediccion del comportamiento de aplicaciones paralelas de memoria compartida, sobre nodos de computo multicore," *III Congreso Espanol de Informatica. SO2 XXI Jornadas de Paralelismo SARTECO*, 2010, cEDI 2010 - JP2010.

[10] ——, "Seleccion de nodos de computo multicore, para una aplicacion paralela de memoria compartida," *XVI Congreso Argentino de Ciencias de la Computacion*, vol. 1, no. 1, pp. 152 – 161, 2010, cACIC 2010.

[11] ——, "Selecting a suitable multicore system for shared-memory parallel application on multicore systems," in *PDPTA 2010: Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications, CSRA Press*, Las Vegas, Nevada, USA, 2010, pp. 228–234.

[12] ——, "Methodology to predict the performance behavior of shared-memory parallel application on multicore systems," in *PDPTA 2011: Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications, CSRA Press*, Las Ve-

gas, Nevada, USA, 2011, pp. accepted for publication in the proceedings and formal presentation in PDPTA '2011: july 18–21.

[13] H. Curnow and B. Wichman, "A synthetic benchmark," New York, NY, USA, pp. 43–49, 1976.

[14] E. Duesterwald, C. Cascaval, and S. Dwarkadas, "Characterizing and predicting program behavior and its variability," in *PACT '03: Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques*. Washington, DC, USA: IEEE Computer Society, 2003, p. 220.

[15] L. Eeckhout, R. H. Bell Jr., B. Stougie, K. D. Bosschere, and L. K. John, "Control flow modeling in statistical simulation for accurate and efficient processor design studies," in *Proceedings of the 31st annual international symposium on Computer architecture*, ser. ISCA '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 350–. [Online]. Available: http://portal.acm.org/citation.cfm?id=998680.1006730

[16] L. Eeckhout and K. D. Bosschere, "Hybrid analytical-statistical modeling for efficiently exploring architecture and workload design spaces," in *Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 25–. [Online]. Available: http://portal.acm.org/citation.cfm?id=645988.674156

[17] L. Eeckhout and J. Sampson, "Exploiting program microarchitecture independent characteristics and phase behavior for reduced benchmark suite simulation," in *In Proceedings of the 2005 IEEE International Symposium on Workload Characterization (IISWC*, 2005, pp. 2–12.

[18] D. Ferrari, "On the foundations of artificial workload design," Berkeley, CA, USA, Tech. Rep., 1982.

[19] D. Genbrugge and L. Eeckhout, "Memory data flow modeling in statistical simulation for the efficient exploration of microprocessor design spaces," *IEEE Trans. Com-*

*put.*, vol. 57, pp. 41–54, January 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1340077.1340115

[20] J. L. Gustafson and R. Todi, "Conventional benchmarks as a sample of the performance spectrum," *J. Supercomput.*, vol. 13, no. 3, pp. 321–342, 1999.

[21] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, 2007.

[22] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John, and K. De Bosschere, "Performance prediction based on inherent program similarity," in *Proceedings of the 15th international conference on Parallel architectures and compilation techniques*, ser. PACT '06. New York, NY, USA: ACM, 2006, pp. 114–122.

[23] http://www.sdsc.edu/pmac/projects/mmaps.html, "Multimaps."

[24] icl.cs.utk.edu/hpcc, "Hpcc." [Online]. Available: http://www.openmp.org

[25] A. Joshi, L. Eeckhout, R. H. Bell, Jr., and L. K. John, "Distilling the essence of proprietary workloads into miniature benchmarks," vol. 5, no. 2. New York, NY, USA: ACM, 2008, pp. 1–33.

[26] C. P. Joshi, A. Kumar, and M. Balakrishnan, "A new performance evaluation approach for system level design space exploration," in *ISSS '02: Proceedings of the 15th international symposium on System Synthesis.* New York, NY, USA: ACM, 2002, pp. 180–185.

[27] U. Krishnaswamy and I. D. Scherson, "A framework for computer performance evaluation using benchmark sets," *IEEE Trans. Comput.*, vol. 49, pp. 1325–1338, December 2000.

[28] J. Lau, J. Sampson, E. Perelman, G. Hamerly, and B. Calder, "The strong correlation between code signatures and performance," in *ISPASS '05: Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, 2005.* Washington, DC, USA: IEEE Computer Society, 2005, pp. 236–247.

[29] J. Lau, S. Schoemackers, and B. Calder, "Structures for phase classification," in *Performance Analysis of Systems and Software, 2004 IEEE International Symposium on - ISPASS*, 2004, pp. 57 – 67.

[30] J. Lau, S. Schoenmackers, and B. Calder, "Transition phase classification and prediction," in *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 278–289. [Online]. Available: http://portal.acm.org/citation.cfm?id=1042442.1043427

[31] J. L. Lo, J. S. Emer, H. M. Levy, R. L. Stamm, D. M. Tullsen, and S. J. Eggers, "Converting thread-level parallelism to instruction-level parallelism via simultaneous multithreading," *ACM Trans. Comput. Syst.*, vol. 15, pp. 322–354, August 1997. [Online]. Available: http://doi.acm.org/10.1145/263326.263382

[32] G. Marin and J. Mellor-Crummey, "Cross-architecture performance predictions for scientific applications using parameterized models," in *SIGMETRICS '04/Performance '04: Proceedings of the joint international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2004, pp. 2–13.

[33] C. Mendes and D. A. Reed, "Performance stability and prediction," in *In IEEE International Workshop on High Performance Computing (WHPC '94)*, 1994.

[34] C. L. Mendes and D. A. Reed, "Integrated compilation and scalability analysis for parallel systems," in *Proceedings of the 1998 International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 385–. [Online]. Available: http://portal.acm.org/citation.cfm?id=522344.825710

[35] Nussbaum and J. Smith, "Modeling superscalar processors via statistical simulation," in *Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '01.

Washington, DC, USA: IEEE Computer Society, 2001, pp. 15–24. [Online]. Available: http://portal.acm.org/citation.cfm?id=645988.674176

[36] OpenMP, "Openmp application program interface." [Online]. Available: http://www.openmp.org

[37] M. Oskin, F. T. Chong, and M. Farrens, "Hls: Combining statistical and symbolic simulation to guide microprocessor designs," in *In Proceedings of the International Symposium on Computer Architecture*, 2000.

[38] W. Pfeiffer and N. J. Wright, "Modeling and predicting application performance on parallel computers using hpc challenge benchmarks," in *22nd IEEE International Parallel and Distributed Processing Symposium, Hyatt Regency Hotel, Miami, FL, 2008*, 2008.

[39] R. H. Saavedra and A. J. Smith, "Analysis of benchmark characteristics and benchmark performance prediction," *ACM Trans. Comput. Syst.*, vol. 14, pp. 344–384, November 1996.

[40] ——, "Measuring cache and tlb performance and their effect of benchmark run times," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-93-767, Aug 1993. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1993/6283.html

[41] R. Saavedra and A. Smith, "Performance characterization of optimizing compilers," *Software Engineering, IEEE Transactions on*, vol. 21, no. 7, pp. 615 –628, Jul. 1995.

[42] S. Sharkawi, D. DeSota, R. Panda, R. Indukuru, S. Stevens, V. Taylor, and X. Wu, "Performance projection of hpc applications using spec cfp2006 benchmarks," in *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–12.

[43] X. Shen, Y. Zhong, and C. Ding, "Locality phase prediction," in *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS-XI.

New York, NY, USA: ACM, 2004, pp. 165–176. [Online]. Available: http://doi.acm.org/10.1145/1024393.1024414

[44] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," *SIGPLAN Not.*, vol. 37, pp. 45–57, October 2002. [Online]. Available: http://doi.acm.org/10.1145/605432.605403

[45] T. Sherwood, S. Sair, and B. Calder, "Phase tracking and prediction," in *Proceedings of the 30th annual international symposium on Computer architecture*, ser. ISCA '03. New York, NY, USA: ACM, 2003, pp. 336–349. [Online]. Available: http://doi.acm.org/10.1145/859618.859657

[46] J. Simon and J. michael Wierum, "Accurate performance prediction for massively parallel systems and its applications," in *Proceedings of European Conference on Parallel Processing EURO-PAR 96*. Springer-Verlag, 1996, pp. 675–688.

[47] A. Snavely, N. Wolter, and L. Carrington, "Modeling application performance by convolving machine signatures with application profiles," 2001.

[48] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. Badia, and A. Purkayastha, "A framework for performance modeling and prediction," in *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, ser. Supercomputing '02. Los Alamitos, CA, USA: IEEE Computer Society Press, 2002, pp. 1–17. [Online]. Available: http://portal.acm.org/citation.cfm?id=762761.762785

[49] M. Sottile and R. Minnich, "Analysis of microbenchmarks for performance tuning of clusters," in *Proceedings of the 2004 IEEE International Conference on Cluster Computing*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 371–377.

[50] D. Spooner and D. Kerbyson, "Identification of performance characteristics from multi-view trace analysis," in *Proc. Of Int. Conf. On Computational Science (ICCS) part*, 2003, pp. 936–945.

[51] K. Sreenivasan and A. J. Kleinman, "On the construction of a representative synthetic workload," *Commun. ACM*, vol. 17, pp. 127–133, March 1974. [Online]. Available: http://doi.acm.org/10.1145/360860.360863

[52] M. Tikir, L. Carrington, E. Strohmaier, and A. Snavely, "A genetic algorithms approach to modeling the performance of memory-bound computations," in *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM, 2007, pp. 1–12.

[53] A. Toomula and J. Subhlok, "Replicating memory behavior for performance prediction," in *Proceedings of the 7th workshop on Workshop on languages, compilers, and run-time support for scalable systems*, ser. LCR '04. New York, NY, USA: ACM, 2004, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1066650.1066666

[54] F. Vandeputte and L. Eeckhout, "Phase complexity surfaces: characterizing time-varying program behavior," in *Proceedings of the 3rd international conference on High performance embedded architectures and compilers*, ser. HiPEAC'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 320–334.

[55] R. P. Weicker, "Dhrystone: a synthetic systems programming benchmark," *Commun. ACM*, vol. 27, pp. 1013–1030, October 1984. [Online]. Available: http://doi.acm.org/10.1145/358274.358283

[56] E. W. Weisstein, "Eigenvalue." [Online]. Available: http://mathworld.wolfram.com/Eigenvalue.html

[57] L. Yang, X. Ma, and F. Mueller, "Cross-platform performance prediction of parallel applications using partial execution," in *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. Washington, DC, USA: IEEE Computer Society, 2005, p. 40.